



개발자 가이드

Amazon Simple Storage Service



API 버전 2006-03-01

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Simple Storage Service: 개발자 가이드

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 상표 및 트레이드 드레스는 Amazon 외 제품 또는 서비스와 함께 사용되어서는 안되며, 고객에게 혼동을 일으키거나 Amazon 브랜드 이미지를 떨어뜨리고 폄하하는 방식으로 이용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 해당 상표의 소유자의 자산이며, 해당 상표의 소유자가 Amazon의 계열사이거나 Amazon과 제휴 관계에 있거나 Amazon의 후원을 받는 업체일 수 있습니다.

Table of Contents

Amazon S3란 무엇인가요?	1
API 직접 호출 권장 사항	2
직접 REST API 직접 호출 실행	2
권한	3
Amazon S3를 사용한 개발	4
요청 만들기	4
액세스 키 정보	5
요청 엔드포인트	6
IPv6을 통해 요청	6
AWS SDK를 사용하여 요청	17
REST API를 사용하여 요청	55
AWS CLI 사용	66
AWS CLI에 대해 자세히 알아보기	67
AWS SDK를 사용하여 개발	67
SDK 프로그래밍 인터페이스	69
요청 인증에서 서명 버전 지정	69
지원에 대한 Amazon S3 요청 ID 가져오기	78
AWS CLI을 이용한 요청 ID 가져오기	78
Windows PowerShell을 사용하여 요청 ID 가져오기	79
AWS CloudTrail 데이터 이벤트를 이용한 요청 ID 가져오기	79
S3 서버 액세스 로깅을 이용한 요청 ID 가져오기	79
HTTP를 이용한 요청 ID 가져오기	79
웹 브라우저를 이용한 요청 ID 가져오기	80
AWS SDK를 이용한 요청 ID 가져오기	80
S3 Tables에 지원되는 S3 객체 수준 API 작업	85
코드 예제	90
Amazon S3	96
기본 사항	101
시나리오	787
서버리스 예제	1239
Amazon S3 Control	1251
기본 사항	1252
S3 디렉터리 버킷	1331
기본 사항	1333

시나리오	1458
요청 인증(AWS 서명 버전 4)	1462
인증 방법	1463
서명 요청 소개	1464
Authorization 헤더 사용	1465
개요	1465
서명 계산: 페이로드를 단일 청크로 전송	1469
서명 계산: 페이로드를 여러 청크로 전송	1488
서명 계산: 후행 헤더 포함	1501
쿼리 파라미터 사용	1507
서명 계산	1510
예제	1514
예제 2	1516
예제: 서명 계산	1517
Java를 사용한 서명 계산 예제	1518
C#을 사용한 서명 계산 예제	1519
HTTP POST 요청 인증	1520
서명 계산	1522
Amazon S3 서명 버전 4 인증 관련 정책 키	1524
서명 버전 4 관련 조건 키를 사용하는 버킷 정책 예제	1526
POST를 사용한 브라우저 기반 업로드	1529
POST Object	1530
설명	1530
버전 관리	1531
요청	1531
예	1548
관련 리소스	1549
POST Object restore	1550
설명	1550
Select 요청으로 아카이브 쿼리	1550
아카이브 복원	1552
요청	1553
응답	1568
예제	1569
추가 정보	1572
HTTP POST를 사용한 브라우저 기반 업로드	1572

서명 계산	1574
HTML 양식 생성	1575
HTML 양식 선언	1576
HTML 양식 필드	1576
POST 정책	1582
Expiration	1583
조건 매칭	1583
조건	1585
문자 이스케이프	1589
POST Upload 예제	1591
HTTP POST를 사용하여 Amazon S3에 파일 업로드	1591
AWS Amplify를 사용한 브라우저 기반 업로드	1594
AWS Amplify JavaScript 라이브러리를 사용하여 Amazon S3에 파일 업로드	1594
추가 정보	1595
공통 요청 헤더	1596
공통 응답 헤더	1599
오류 응답	1604
REST 오류 응답	1605
오류 코드 목록	1606
SELECT 객체 콘텐츠 오류 코드 목록	1632
복제 관련 오류 코드 목록	1644
태그 지정 관련 오류 코드 목록	1647
Amazon S3 주식 관련 오류 코드 목록	1648
Amazon S3 on Outposts 오류 코드 목록	1650
Amazon S3 Storage Lens 오류 코드 목록	1652
Amazon S3 Object Lambda 오류 코드 목록	1659
Amazon S3 비동기 오류 코드 목록	1663
Amazon S3 Access Grants 오류 코드 목록	1664
Amazon FSx 관련 오류 코드 목록	1667
Amazon S3 Tables 오류 코드 목록	1668
Amazon S3 오류 모범 사례	1669
InternalErrors 재시도	1669
반복되는 SlowDown 오류에 대한 애플리케이션 튜닝	1669
오류 격리	1670
AWS 용어집	1671
리소스	1672

부록	1674
부록: SelectObjectContent 응답	1675
설명	1675
응답	1675
관련 리소스	1684
부록: OPTIONS 객체	1685
설명	1685
요청	1685
응답	1686
예제	1687
관련 리소스	1688
부록: SOAP API	1688
서비스에 대한 작업(SOAP API)	1689
버킷에 대한 작업(SOAP API)	1691
객체에 대한 작업(SOAP API)	1703
SOAP 요청 인증	1727
SOAP를 사용한 액세스 정책 설정	1728
공통 요소	1730
SOAP 오류 응답	1730
부록: 요청 인증(AWS 서명 버전 2)	1732
REST API를 사용하여 요청 인증(AWS 서명 버전 2)	1734
REST 요청 서명 및 인증(AWS 서명 버전 2)	1737
POST를 사용한 브라우저 기반 업로드(AWS 서명 버전 2)	1750
부록: 수명 주기 구성 API(사용되지 않음)	1774
PUT 버킷 수명 주기(사용되지 않음)	1776
GET 버킷 수명 주기(사용되지 않음)	1792
문서 기록	1803

Amazon S3란 무엇인가요?

Amazon Simple Storage Service(Amazon S3)는 업계 최고의 확장성, 데이터 가용성, 보안 및 성능을 제공하는 객체 스토리지 서비스입니다. 모든 규모와 업종의 고객은 Amazon S3를 사용하여 데이터 레이크, 웹 사이트, 모바일 애플리케이션, 백업 및 복원, 아카이브, 엔터프라이즈 애플리케이션, IoT 디바이스, 빅 데이터 분석 등 다양한 사용 사례에서 원하는 양의 데이터를 저장하고 보호할 수 있습니다. Amazon S3는 특정 비즈니스, 조직 및 규정 준수 요구 사항에 맞게 데이터에 대한 액세스를 최적화, 구조화 및 구성할 수 있는 관리 기능을 제공합니다.

이 안내서에서는 Amazon S3 REST API를 사용하여 클라우드에 데이터를 저장 및 검색하는 애플리케이션을 구축하는 방법에 대한 개발자 중심 정보를 제공합니다.

HTTP를 지원하는 임의의 도구 키트를 사용하여 REST API를 사용할 수 있습니다. 심지어 브라우저를 사용하여 객체를 가져올 수도 있습니다. 단, 객체를 익명으로 읽을 수 있어야 합니다.

REST API는 표준 HTTP 헤더 및 상태 코드를 사용하므로 표준 브라우저 및 도구 키트가 예상대로 작동합니다. Amazon은 일부 영역에서 HTTP에 기능을 추가했습니다. 예를 들어, 액세스 제어를 지원하기 위해 헤더를 추가했습니다. 이 경우 새로운 기능은 최대한 표준 HTTP 사용법과 일치하는 방식으로 추가되었습니다.

Note

HTTP를 통한 SOAP 지원은 중단되었지만 HTTPS를 통해 계속해서 사용할 수 있습니다. 하지만 새로운 Amazon S3 기능은 SOAP에 대해 지원되지 않습니다. 다음 링크에서 REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

<https://aws.amazon.com/developer/tools/>

이 개발자 안내서에는 다음이 포함되어 있습니다.

- [공통 요청 헤더](#) 및 [공통 응답 헤더](#)
- [오류 응답](#)
- [POST를 사용한 브라우저 기반 업로드\(AWS 서명 버전 4\)](#)

Important

특정 API 주제로 이동하기 전에 인증 및 액세스 제어에 관한 다음 내용을 읽어보세요.

Amazon S3에 대한 요청은 인증되거나 익명일 수 있습니다. 인증된 액세스에는 AWS가 요청을 인증하는 데 사용할 수 있는 자격 증명이 필요합니다.

API 직접 호출 권장 사항

코드에서 바로 REST API를 직접적으로 호출하는 것은 번거로울 수 있습니다. 요청을 인증하려면 유효한 서명을 계산하는 데 필요한 코드를 작성해야 합니다. 대신 다음과 같은 대안을 사용하는 것이 좋습니다.

- [AWS SDK](#)를 사용하여 요청을 전송합니다.

또한 [샘플 코드 및 라이브러리](#)를 참조하세요.

SDK를 사용하는 경우 SDK 클라이언트에서 사용자가 제공한 액세스 키를 사용하여 요청을 인증하기 때문에 요청 인증을 위한 서명을 계산하는 코드를 작성할 필요가 없습니다. 특별한 이유가 없다면 항상 AWS SDK를 사용해야 합니다.

- AWS CLI를 사용하여 Amazon S3 API 직접 호출을 실행합니다. AWS CLI 및 예제 Amazon S3 명령을 설정하는 방법에 대한 자세한 내용은 다음 주제를 참조하세요.

Amazon Simple Storage Service 사용 설명서의 [AWS CLI 설정](#).

AWS Command Line Interface 사용 설명서의 [AWS 명령줄 인터페이스에서 Amazon S3 사용](#).

직접 REST API 직접 호출 실행

Note

PUT 요청 헤더는 크기가 8KB 이하여야 합니다. PUT 요청 헤더에 포함되는 시스템 정의 메타데이터의 크기는 2KB 이하여야 합니다. 시스템 정의 메타데이터의 크기는 US-ASCII로 인코딩된 각 키와 값의 바이트 수를 더하여 측정됩니다.

위의 대안 중 하나를 사용하는 대신 직접 REST API 직접 호출을 실행하려고 한다면 몇 가지 유의할 사항이 있습니다.

- 코드에서 직접 REST API 직접 호출을 실행하려면 유효한 자격 증명을 사용하여 서명을 생성하고 요청에 서명을 포함합니다. 다양한 인증 방법 및 서명 계산에 대한 자세한 내용은 [요청 인증\(AWS 서명 버전 4\)](#) 섹션을 참조하세요.

- REST API는 표준 HTTP 헤더 및 상태 코드를 사용하므로 표준 브라우저 및 도구 키트가 예상대로 작동합니다. Amazon은 일부 영역에서 HTTP에 기능을 추가했습니다. 예를 들어, 액세스 제어를 지원하기 위해 헤더를 추가했습니다. 이 경우 새로운 기능은 최대한 표준 HTTP 사용법과 일치하는 방식으로 추가되었습니다. 요청 생성에 대한 자세한 내용은 [요청 만들기](#) 섹션을 참조하세요.

권한

요청을 인증하는 데 유효한 자격 증명이 있더라도 계정 소유자 또는 버킷 소유자로부터 S3 권한을 얻지 못하면 Amazon S3 리소스를 생성하거나 액세스할 수 없습니다. 이러한 권한은 일반적으로 버킷 정책과 같은 AWS Identity and Access Management(IAM) [정책](#)을 통해 부여됩니다. 예를 들어 S3 버킷을 생성하거나 버킷의 객체를 가져올 수 있는 권한이 있어야 합니다. S3 권한의 전체 목록은 [Amazon S3에 사용되는 작업, 리소스 및 조건 키](#)를 참조하세요.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [Amazon S3 API 작업에 필요한 권한](#)을 참조하세요.

AWS 계정의 루트 사용자 자격 증명을 사용하는 경우 모든 권한이 확보됩니다. 그러나 루트 사용자 자격 증명은 사용하지 않는 것이 좋습니다. 대신 계정에 AWS Identity and Access Management(IAM) 역할을 생성하고 사용자 권한을 관리하는 것이 좋습니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [액세스 관리](#)를 참조하세요.

Amazon S3를 사용한 개발

이 섹션에서는 Amazon S3 사용에 대한 개발자 관련 주제를 다룹니다. 자세한 내용은 아래 주제를 검토하세요.

주제

- [요청 만들기](#)
- [AWS CLI를 사용하여 Amazon S3에서 개발](#)
- [AWS SDK를 사용하여 Amazon S3에서 개발](#)
- [AWS Support에 대한 Amazon S3 요청 ID 가져오기](#)
- [S3 Tables에 지원되는 Amazon S3 객체 수준 API 작업](#)

요청 만들기

Amazon S3은 REST 서비스입니다. REST API 또는 프로그래밍 태스크를 간소화하기 위해 기본 Amazon S3 REST API를 래핑하는 AWS SDK 래퍼 라이브러리([샘플 코드 및 라이브러리](#) 참조)를 사용하여 Amazon S3에 요청을 전송할 수 있습니다.

Amazon S3와의 모든 상호 작용은 인증을 거치거나 익명으로 할 수 있습니다. 인증은 Amazon Web Services(AWS) 제품에 액세스하려는 요청자의 ID를 확인하는 프로세스입니다. 인증된 요청에는 요청 발신자를 인증하는 서명 값이 포함되어 있어야 합니다. 서명 값의 일부는 요청자의 AWS 액세스 키(액세스 키 ID 및 비밀 액세스 키)에서 생성됩니다. 액세스 키를 받는 방법에 대한 자세한 내용은 AWS 일반 참조의 [보안 인증 정보를 가져오려면 어떻게 해야 하나요?](#)를 참조하세요.

AWS SDK를 사용할 경우 라이브러리에서 제공된 키를 바탕으로 서명을 컴퓨팅합니다. 그러나 애플리케이션에서 바로 REST API를 호출할 경우 서명을 구하는 코드를 작성하여 요청에 추가해야 합니다.

주제

- [액세스 키 정보](#)
- [요청 엔드포인트](#)
- [IPv6을 통해 Amazon S3에 요청](#)
- [AWS SDK를 사용하여 요청](#)
- [REST API를 사용하여 요청](#)

액세스 키 정보

다음 단원에서는 요청 인증에 사용할 수 있는 액세스 키의 유형에 대해 설명합니다.

AWS 계정 액세스 키

계정 액세스 키는 계정에서 소유한 AWS 리소스에 대한 모든 액세스를 제공합니다. 다음은 액세스 키의 예제입니다.

- 액세스 키 ID(20자, 영숫자 문자열). 예: AKIAIOSFODNN7EXAMPLE
- 보안 액세스 키(40자 문자열). 예: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

액세스 키 ID는 개별 AWS 계정을 식별합니다. 이러한 액세스 키를 사용하여 Amazon S3에 인증된 요청을 전송할 수 있습니다.

IAM 사용자 액세스 키

회사를 대표하는 하나의 AWS 계정을 만들 수 있지만 조직 내 여러 명의 직원이 조직의 AWS 리소스에 액세스해야 할 수 있습니다. 이 경우 AWS 계정 계정 액세스 키를 공유하면 보안에 문제가 될 수 있으며, 각 직원에 대해 별도의 AWS 계정을 만드는 것은 실용적이지 않습니다. 또한 버킷, 객체와 같은 각 계정에서 소유하고 있는 리소스를 쉽게 공유할 수 없으며, 리소스를 공유하기 위해서는 권한을 부여하는 추가 작업이 필요합니다.

이러한 경우 AWS Identity and Access Management(IAM)을 사용하여 AWS 계정 아래에서 개별 액세스 키를 갖는 사용자를 만들고, 이러한 사용자에게 적절한 리소스 액세스 권한을 부여하는 IAM 사용자 정책은 연결할 수 있습니다. 이러한 사용자를 효과적으로 관리하기 위해 IAM은 사용자 그룹을 만들어 이 그룹의 모든 사용자에게 적용되는 그룹 수준 권한을 부여할 수 있습니다.

이러한 사용자를 IAM 사용자라고 하며, 계정에서 만들고 관리합니다. AWS 상위 계정은 사용자의 액세스 권한을 제어합니다. AWS IAM 사용자가 만드는 모든 리소스는 상위 AWS 계정에 의해 제어되고 비용이 청구됩니다. 이러한 IAM 사용자는 개별 보안 자격 증명을 사용하여 Amazon S3에 인증된 요청을 전송할 수 있습니다. AWS 계정 아래 사용자의 생성 및 관리에 대한 자세한 내용은 [AWS Identity and Access Management 제품 세부 정보 페이지](#)를 참조하세요.

임시 보안 자격 증명

개별 액세스 키를 갖는 IAM 사용자를 생성하는 것 외에, IAM을 사용하여 IAM 사용자에게 AWS 서비스 및 리소스에 액세스할 수 있도록 임시 보안 자격 증명(임시 액세스 키 및 보안 토큰)을 부여할 수 있습니다. 외부의 시스템에서 사용자를 관리할 수도 있습니다. AWS 이러한 사용자를 연합된 사용자라고 합니다. AWS 리소스에 액세스하도록 생성한 애플리케이션도 사용자가 될 수 있습니다.

IAM은 임시 보안 자격 증명을 요청하기 위한 AWS Security Token Service API를 제공합니다. AWS STS API 또는 AWS SDK를 사용하여 이러한 자격 증명을 요청할 수 있습니다. API는 임시 보안 자격 증명(액세스 키 ID 및 보안 액세스 키)과 보안 토큰을 반환합니다. 이러한 자격 증명은 요청 시 지정한 기간 동안에만 사용할 수 있습니다. AWS 계정 또는 IAM 사용자 액세스 키를 사용하여 요청을 전송할 때와 동일한 방식으로 액세스 키 ID 및 보안 키를 사용합니다. 또한 Amazon S3에 전송하는 각 요청에 토큰을 포함해야 합니다.

IAM 사용자는 직접 사용하기 위해 또는 연합된 사용자나 애플리케이션에 전달하기 위해 이러한 임시 보안 자격 증명을 요청할 수 있습니다. 연합된 사용자를 위해 임시 보안 자격 증명을 요청할 경우 사용자 이름, 그리고 해당 임시 보안 자격 증명에 연결할 권한을 정의한 IAM 정책을 제공해야 합니다. 연합된 사용자는 임시 자격 증명을 요청한 상위 IAM 사용자보다 더 많은 권한을 가질 수 없습니다.

Amazon S3에 요청할 때 이 임시 보안 자격 증명을 사용할 수 있습니다. API 라이브러리는 요청 인증 시 이 자격 증명을 사용하여 필요한 서명 값을 계산합니다. 만료된 자격 증명으로 요청을 보내면 Amazon S3은 요청을 거부합니다.

REST API 요청에서 임시 보안 자격 증명을 사용한 요청 서명에 대한 자세한 내용은 [REST 요청 서명 및 인증\(AWS 서명 버전 2\)](#) 단원을 참조하십시오. AWS SDK를 사용한 요청 전송에 대한 자세한 내용은 [AWS SDK를 사용하여 요청](#) 섹션을 참조하세요.

임시 보안 자격 증명에 대한 IAM 지원에 관한 자세한 내용은 IAM 사용 설명서의 [임시 보안 자격 증명](#)을 참조하세요.

보안을 강화하기 위해 Amazon S3 리소스에 액세스할 때 Multi-Factor Authentication(MFA)을 요구하도록 버킷 정책을 구성할 수 있습니다. 자세한 내용은 [예제 버킷 정책: MFA 요구](#)를 참조하세요. Amazon S3 리소스에 액세스하려면 MFA를 거치도록 구성할 경우 이러한 리소스에 액세스하려면 항상 MFA 키로 생성된 임시 자격 증명을 제공해야 합니다. 자세한 내용은 [AWS 멀티 팩터 인증](#) 세부 정보 페이지 및 IAM 사용 설명서의 [MFA 보호 API 액세스 구성](#)을 참조하세요.

요청 엔드포인트

사전 정의된 서비스의 엔드포인트로 REST 요청을 전송합니다. AWS 서비스 및 해당 엔드포인트의 전체 목록은 AWS 일반 참조의 [리전 및 엔드포인트](#)를 참조하세요.

IPv6을 통해 Amazon S3에 요청

Amazon Simple Storage Service(Amazon S3)는 IPv4 프로토콜 외에도 IPv6(인터넷 프로토콜 버전 6)을 사용하여 S3 버킷에 액세스할 수 있는 기능을 지원합니다. Amazon S3 듀얼 스택 엔드포인트는

IPv6 및 IPv4를 통한 S3 버킷 요청을 지원합니다. IPv6을 통해 Amazon S3에 액세스하는 데 대한 추가 요금은 없습니다. 요금에 대한 자세한 내용은 [Amazon S3 요금](#)을 참조하세요.

 Note

다음은 퍼블릭 Amazon S3 IPv6 엔드포인트에 대한 정보입니다. AWS Virtual Private Cloud(VPC) 엔드포인트에 대한 자세한 내용은 [AWS PrivateLink for Amazon S3](#)를 참조하세요.

주제

- [IPv6을 통해 요청하기](#)
- [IAM 정책에서 IPv6 주소 사용](#)
- [IP 주소 호환성 테스트](#)
- [Amazon S3 듀얼 스택 엔드포인트 사용](#)

IPv6을 통해 요청하기

IPv6을 통해 S3 버킷에 요청하려면 듀얼 스택 엔드포인트를 사용해야 합니다. 다음 단원에서는 듀얼 스택 엔드포인트를 사용하여 IPv6을 통해 요청하는 방법을 설명합니다.

다음은 IPv6을 통해 버킷에 액세스하기 전에 알아야 할 몇 가지 사항입니다.

- 버킷에 액세스하는 클라이언트와 네트워크가 IPv6을 사용하도록 설정되어 있어야 합니다.
- 가상 호스팅 방식과 경로 방식 요청이 모두 IPv6 액세스를 지원합니다. 자세한 내용은 [Amazon S3 듀얼 스택 엔드포인트](#) 섹션을 참조하세요.
- AWS Identity and Access Management(IAM) 사용자 또는 버킷 정책에서 소스 IP 주소 필터링을 사용하는 경우, IPv6 주소 범위를 포함하도록 정책을 업데이트해야 합니다. 자세한 내용은 [IAM 정책에서 IPv6 주소 사용](#) 섹션을 참조하세요.
- IPv6을 사용하는 경우, 서버 액세스 로그 파일이 IPv6 형식으로 IP 주소를 출력합니다. IPv6 형식의 Remote IP 주소를 구문 분석할 수 있도록 Amazon S3 로그 파일을 구문 분석하는 데 사용하는 기존 도구, 스크립트 및 소프트웨어를 업데이트해야 합니다. 자세한 내용은 [서버 액세스 로깅을 사용하여 요청 로깅](#)을 참조하세요.

Note

로그 파일에 IPv6 주소가 있는 것과 관련하여 문제가 있는 경우, [AWS Support](#)에 문의하세요.

듀얼 스택 엔드포인트를 사용하여 IPv6을 통해 요청

듀얼 스택 엔드포인트를 사용하여 IPv6을 통해 Amazon S3 API 호출로 요청합니다. Amazon S3 API 작업은 IPv6을 통해 Amazon S3에 액세스하든 IPv4를 통해 액세스하든 동일하게 실행됩니다. 성능 또한 동일합니다.

REST API를 사용하는 경우, 듀얼 스택 엔드포인트에 직접 액세스합니다. 자세한 내용은 [듀얼 스택 엔드포인트](#) 섹션을 참조하세요.

AWS Command Line Interface(AWS CLI) 및 AWS SDK를 사용하는 경우, 파라미터 또는 플래그를 사용하여 듀얼 스택 엔드포인트를 변경할 수 있습니다. 구성 파일의 Amazon S3 엔드포인트를 재정의하여 듀얼 스택 엔드포인트를 직접 지정할 수도 있습니다.

듀얼 스택 엔드포인트를 사용하여 다음에서 IPv6을 통해 버킷에 액세스할 수 있습니다.

- AWS CLI에 대한 내용은 [AWS CLI의 듀얼 스택 엔드포인트 사용](#)을 참조하십시오.
- AWS SDK([AWS SDK의 듀얼 스택 엔드포인트 사용](#) 참조).
- REST API, [REST API를 사용하여 듀얼 스택 엔드포인트에 요청](#) 참조.

IPv6을 통해 사용할 수 없는 기능

S3 버킷에서 정적 웹 사이트를 호스팅하는 기능은 현재 IPv6을 통해 S3 버킷에 액세스할 때 지원되지 않습니다.

IAM 정책에서 IPv6 주소 사용

IPv6을 사용하여 버킷에 액세스하기 전에 IP 주소 필터링에 사용되는 IAM 사용자 또는 S3 버킷 정책이 IPv6 주소 범위를 포함하도록 업데이트되었는지 확인해야 합니다. IP 주소 필터링 정책이 IPv6 주소를 처리하도록 업데이트되지 않은 경우, 클라이언트가 IPv6을 사용할 때 버킷에 대한 액세스 권한을 잘못 잃거나 얻을 수 있습니다. IAM를 통한 액세스 권한 관리에 대한 자세한 내용은 [Amazon S3의 ID 및 액세스 관리](#) 섹션을 참조하세요.

IP 주소를 필터링하는 IAM 정책은 [IP 주소 조건 연산자](#)를 사용합니다. 다음 버킷 정책은 IP 주소 조건 연산자를 사용하여 허용되는 IPv4 주소인 54.240.143.* 범위를 식별합니다. 이 범위를 벗어난 모든 IP 주소는 버킷(examplebucket)에 대한 액세스가 거부됩니다. 모든 IPv6 주소가 허용되는 범위 밖에 있으므로 이 정책은 IPv6 주소가 examplebucket에 액세스하지 못하도록 합니다.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::examplebucket/*",
      "Condition": {
        "IpAddress": {"aws:SourceIp": "54.240.143.0/24"}
      }
    }
  ]
}
```

다음 예제와 같이 IPv4(Condition) 및 IPv6(54.240.143.0/24) 주소 범위를 모두 허용하도록 버킷 정책의 2001:DB8:1234:5678::/64 요소를 수정할 수 있습니다. 예제에 나와 있는 동일한 Condition 블록 유형을 사용하여 IAM 사용자와 버킷 정책 모두를 업데이트할 수 있습니다.

```
"Condition": {
  "IpAddress": {
    "aws:SourceIp": [
      "54.240.143.0/24",
      "2001:DB8:1234:5678::/64"
    ]
  }
}
```

IPv6을 사용하기 전에 IP 주소 필터링을 사용하는 관련된 모든 IAM 사용자와 버킷 정책을 업데이트해야 합니다. 버킷 정책에서는 IP 주소 필터링을 사용하지 않는 것이 좋습니다.

<https://console.aws.amazon.com/iam/>의 IAM 콘솔을 사용하여 IAM 사용자 정책을 검토할 수 있습니다. IAM에 대한 자세한 내용은 [IAM 사용 설명서](#)를 참조하세요. S3 버킷 정책에 대한 자세한 내용은 [버킷 정책 추가](#)를 참조하세요.

IP 주소 호환성 테스트

Linux/Unix 또는 Mac OS X를 사용하는 경우, 다음 예제에 나와 있는 `curl` 명령을 사용하여 IPv6을 통해 듀얼 스택 엔드포인트에 액세스할 수 있는지 테스트할 수 있습니다.

Example

```
curl -v http://s3.dualstack.us-west-2.amazonaws.com/
```

다음 예제와 유사한 정보를 가져옵니다. IPv6을 통해 연결된 경우, 연결된 IP 주소가 IPv6 주소가 됩니다.

```
* About to connect() to s3-us-west-2.amazonaws.com port 80 (#0)
* Trying IPv6 address... connected
* Connected to s3.dualstack.us-west-2.amazonaws.com (IPv6 address) port 80 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.18.1 (x86_64-unknown-linux-gnu) libcurl/7.18.1 OpenSSL/1.0.1t
zlib/1.2.3
> Host: s3.dualstack.us-west-2.amazonaws.com
```

Windows 7 또는 10을 사용하는 경우, 다음 예제에 나와 있는 `ping` 명령을 사용하여 IPv6 또는 IPv4를 통해 듀얼 스택 엔드포인트에 액세스할 수 있는지 테스트할 수 있습니다.

```
ping ipv6.s3.dualstack.us-west-2.amazonaws.com
```

Amazon S3 듀얼 스택 엔드포인트 사용

Amazon S3 듀얼 스택 엔드포인트는 IPv6 및 IPv4를 통한 S3 버킷 요청을 지원합니다. 이 단원에서는 듀얼 스택 엔드포인트를 사용하는 방법을 설명합니다.

주제

- [Amazon S3 듀얼 스택 엔드포인트](#)
- [AWS CLI의 듀얼 스택 엔드포인트 사용](#)

- [AWS SDK의 듀얼 스택 엔드포인트 사용](#)
- [REST API의 듀얼 스택 엔드포인트 사용](#)

Amazon S3 듀얼 스택 엔드포인트

듀얼 스택 엔드포인트에 요청하는 경우, 버킷 URL이 IPv6 또는 IPv4 주소로 확인됩니다. IPv6을 통해 버킷에 액세스하는 방법에 대한 자세한 내용은 [IPv6을 통해 Amazon S3에 요청](#) 단원을 참조하세요.

REST API를 사용하는 경우, 엔드포인트 이름(URI)을 사용하여 Amazon S3 엔드포인트에 직접 액세스합니다. 가상 호스팅 방식 또는 경로 방식 엔드포인트 이름을 사용하여 듀얼 스택 엔드포인트를 통해 S3 버킷에 액세스할 수 있습니다. Amazon S3은 리전 듀얼 스택 엔드포인트 이름만 지원합니다. 이는 이름의 일부로 리전을 지정해야 함을 뜻합니다.

듀얼 스택 가상 호스트 방식 및 경로 방식 엔드포인트 이름에는 다음과 같은 명명 규칙을 사용합니다.

- 가상 호스팅 방식 듀얼 스택 엔드포인트:

bucketname.s3.dualstack.*aws-region*.amazonaws.com

- 경로 방식 듀얼 스택 엔드포인트:

s3.dualstack.*aws-region*.amazonaws.com/*bucketname*

엔드포인트 이름 스타일에 대한 자세한 내용은 [Amazon S3 버킷 액세스 및 나열](#)을 참조하세요.

Amazon S3 엔드포인트 목록은 AWS 일반 참조의 [리전 및 엔드포인트](#)를 참조하세요.

Important

듀얼 스택 엔드포인트를 사용할 경우 Transfer Acceleration을 사용할 수 있습니다. 자세한 내용은 [Amazon S3 Transfer Acceleration 시작하기](#)를 참조하세요.

Note

Amazon S3에 액세스하는 두 가지 유형의 Virtual Private Cloud(VPC) 엔드포인트(인터페이스 VPC 엔드포인트 및 게이트웨이 VPC 엔드포인트)는 이제 듀얼 스택이 지원됩니다. Amazon S3용 VPC 엔드포인트에 대한 자세한 내용은 [Amazon S3용 AWS PrivateLink](#)를 참조하세요.

AWS Command Line Interface(AWS CLI) 및 AWS SDK를 사용하는 경우, 파라미터 또는 플래그를 사용하여 듀얼 스택 엔드포인트를 변경할 수 있습니다. 구성 파일의 Amazon S3 엔드포인트를 재정의하여 듀얼 스택 엔드포인트를 직접 지정할 수도 있습니다. 다음 섹션에서는 AWS CLI 및 AWS SDK의 듀얼 스택 엔드포인트를 사용하는 방법을 설명합니다.

AWS CLI의 듀얼 스택 엔드포인트 사용

이 단원에서는 듀얼 스택 엔드포인트에 요청하는 데 사용되는 AWS CLI 명령의 예를 보여 줍니다. AWS CLI를 설치하는 지침은 [AWS CLI를 사용하여 Amazon S3에서 개발](#) 단원을 참조하십시오.

AWS Config 파일의 프로파일에서 구성 값 `use_dualstack_endpoint`를 `true`로 설정하여 `s3` 및 `s3api` AWS CLI 명령의 모든 Amazon S3 요청을 지정된 리전의 듀얼 스택 엔드포인트로 보냅니다. `--region` 옵션을 사용하여 구성 파일 또는 명령에 리전을 지정합니다.

AWS CLI에서 듀얼 스택 엔드포인트를 사용하는 경우, `path` 및 `virtual` 주소 형식이 모두 지원됩니다. 구성 파일에 설정된 주소 스타일은 버킷 이름이 호스트 이름에 있는지 URL의 일부인지 제어합니다. 기본적으로 CLI는 가능한 경우 가상 방식을 사용하나 필요한 경우 경로 방식으로 대체합니다. 자세한 내용은 [AWS CLI Amazon S3 구성](#)을 참조하세요.

다음 예제와 같이 기본 프로파일에 `use_dualstack_endpoint`를 `true`로, `addressing_style`을 `virtual`로 설정하는 명령을 사용하여 구성을 변경할 수도 있습니다.

```
$ aws configure set default.s3.use_dualstack_endpoint true
$ aws configure set default.s3.addressing_style virtual
```

(모든 명령이 아닌) 지정된 AWS CLI 명령에 대해서만 듀얼 스택 엔드포인트를 사용하려는 경우, 다음 방법 중 하나를 사용할 수 있습니다.

- `--endpoint-url` 또는 `https://s3.dualstack.aws-region.amazonaws.com` 명령에 대해 `http://s3.dualstack.aws-region.amazonaws.com` 파라미터를 `s3` 또는 `s3api`으로 설정하여 명령별로 듀얼 스택 엔드포인트를 사용할 수 있습니다.

```
$ aws s3api list-objects --bucket bucketname --endpoint-url https://s3.dualstack.aws-region.amazonaws.com
```

- AWS Config 파일에 별도의 프로파일을 설정할 수 있습니다. 예를 들어, `use_dualstack_endpoint`를 `true`로 설정하는 프로파일을 하나 작성하고, `use_dualstack_endpoint`를 설정하지 않는 프로파일을 하나 작성합니다. 명령을 실행할 때 듀얼 스택 엔드포인트를 사용할지 여부에 따라 사용할 프로파일을 지정합니다.

Note

현재는 AWS CLI에서 듀얼 스택 엔드포인트로 Transfer Acceleration을 사용할 수 없습니다. 그러나 곧 AWS CLI를 지원할 예정입니다. 자세한 내용은 [S3 Transfer Acceleration 활성화 및 사용](#)을 참조하세요.

AWS SDK의 듀얼 스택 엔드포인트 사용

이 섹션에서는 AWS SDK를 사용하여 듀얼 스택 엔드포인트에 액세스하는 방법의 예제를 보여줍니다.

AWS SDK for Java 듀얼 스택 엔드포인트 예제

다음 예제에서는 AWS SDK for Java를 사용하여 Amazon S3 클라이언트 생성 시 듀얼 스택 엔드포인트를 사용하는 방법을 보여 줍니다.

실제 Java 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 AWS SDK for Java 개발자 안내서의 [시작하기](#) 섹션을 참조하세요.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;

public class DualStackEndpoints {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "**** Bucket name ****";

        try {
            // Create an Amazon S3 client with dual-stack endpoints enabled.
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .withDualstackEnabled(true)
                .build();

            s3Client.listObjects(bucketName);
        } catch (AmazonServiceException e) {
```

```

        // The call was transmitted successfully, but Amazon S3 couldn't process
        // it, so it returned an error response.
        e.printStackTrace();
    } catch (SdkClientException e) {
        // Amazon S3 couldn't be contacted for a response, or the client
        // couldn't parse the response from Amazon S3.
        e.printStackTrace();
    }
}
}
}

```

Windows에서 AWS SDK for Java를 사용하는 경우, 다음과 같은 Java 가상 머신(JVM) 속성을 설정해야 할 수 있습니다.

```
java.net.preferIPv6Addresses=true
```

AWS .NET SDK 듀얼 스택 엔드포인트 예제

AWS SDK for .NET을 사용하는 경우, 다음 예제와 같이 AmazonS3Config 클래스를 사용하여 듀얼 스택 엔드포인트를 사용하도록 설정합니다.

```

using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class DualStackEndpointTest
    {
        private const string bucketName = "**** bucket name ****";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USSouth1;
        private static IAmazonS3 client;

        public static void Main()
        {
            var config = new AmazonS3Config
            {
                UseDualstackEndpoint = true,
                RegionEndpoint = bucketRegion
            }
        }
    }
}

```



```

    };
    client = new AmazonS3Client(config);
    Console.WriteLine("Listing objects stored in a bucket");
    ListingObjectsAsync().Wait();
}

private static async Task ListingObjectsAsync()
{
    try
    {
        var request = new ListObjectsV2Request
        {
            BucketName = bucketName,
            MaxKeys = 10
        };
        ListObjectsV2Response response;
        do
        {
            response = await client.ListObjectsV2Async(request);

            // Process the response.
            foreach (S3Object entry in response.S3Objects)
            {
                Console.WriteLine("key = {0} size = {1}",
                    entry.Key, entry.Size);
            }
            Console.WriteLine("Next Continuation Token: {0}",
                response.NextContinuationToken);
            request.ContinuationToken = response.NextContinuationToken;
        } while (response.IsTruncated == true);
    }
    catch (AmazonS3Exception amazonS3Exception)
    {
        Console.WriteLine("An AmazonS3Exception was thrown. Exception: " +
            amazonS3Exception.ToString());
    }
    catch (Exception e)
    {
        Console.WriteLine("Exception: " + e.ToString());
    }
}
}

```

코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 [AWS SDK for .NET 시작하기](#)를 참조하세요.

REST API의 듀얼 스택 엔드포인트 사용

REST API를 사용하여 듀얼 스택 엔드포인트에 요청하는 방법에 대한 자세한 내용은 [REST API를 사용하여 듀얼 스택 엔드포인트에 요청](#) 단원을 참조하십시오.

AWS SDK를 사용하여 요청

주제

- [AWS 계정 또는 IAM 사용자 자격 증명을 사용하여 요청](#)
- [IAM 사용자 임시 자격 증명을 사용하여 요청하기](#)
- [연합된 사용자의 임시 자격 증명을 사용하여 요청하기](#)

AWS SDK를 사용하거나 애플리케이션에서 직접 REST API를 호출하여 Amazon S3로 인증된 요청을 보낼 수 있습니다. AWS SDK API는 사용자가 제공하는 자격 증명을 사용하여 인증을 위한 서명을 계산합니다. 애플리케이션에서 직접 REST API를 사용하는 경우 요청 인증용 서명을 계산하기 위해 필요한 코드를 작성해야 합니다. 사용 가능한 AWS SDK 목록은 [샘플 코드 및 라이브러리](#)를 참조하세요.

AWS 계정 또는 IAM 사용자 자격 증명을 사용하여 요청

AWS 계정 또는 IAM 사용자 보안 자격 증명을 사용하여 인증된 요청을 Amazon S3로 보낼 수 있습니다. 이 단원에서는 AWS SDK for Java, AWS SDK for .NET 및 AWS SDK for PHP를 사용하여 인증된 요청을 보낼 수 있는 방법의 예제를 제공합니다. 사용 가능한 AWS SDK 목록은 [샘플 코드 및 라이브러리](#)를 참조하세요.

이러한 AWS SDK는 각기 SDK별 자격 증명 공급자 체인을 사용하여 자격 증명을 찾아서 사용하고 자격 증명 소유자를 대신하여 작업을 수행합니다. 이러한 모든 자격 증명 공급자 체인은 공통적으로 로컬 AWS 자격 증명 파일을 검색할 수 있는 기능을 가지고 있습니다.

자세한 내용은 아래 주제를 참조하세요.

주제

- [로컬 AWS 자격 증명 파일 생성](#)
- [AWS SDK를 사용하여 인증된 요청 보내기](#)
- [관련 리소스](#)

로컬 AWS 자격 증명 파일 생성

AWS SDK에 대해 자격 증명을 구성하는 가장 간편한 방법은 AWS 자격 증명 파일을 사용하는 것입니다. AWS Command Line Interface(AWS CLI)를 사용하는 경우에는 이미 로컬 AWS 자격 증명 파일이 구성되어 있을 수 있습니다. 그렇지 않으면 다음 절차를 사용하여 자격 증명 파일을 설정합니다.

1. AWS Management Console에 로그인하여 <https://console.aws.amazon.com/iam/>에서 IAM 콘솔을 엽니다.
2. 코드가 액세스 권한을 갖도록 하고 싶은 서비스 및 작업으로 권한을 제한하여 새로운 사용자를 생성합니다. 새로운 사용자 생성에 대한 자세한 내용은 [IAM 사용자 생성\(콘솔\)](#)을 참조하고 8단계의 지침을 따릅니다.
3. [.csv 다운로드(Download .csv)]를 선택하여 AWS 자격 증명의 로컬 복사본을 저장합니다.
4. 컴퓨터에서 홈 디렉터리를 탐색하고 .aws 디렉터리를 생성합니다. Linux 또는 OS X 같은 Unix 기반 시스템의 경우 이 디렉터리의 위치는 다음과 같습니다.

```
~/ .aws
```

Windows에서 이 디렉터리의 위치는 다음과 같습니다.

```
%HOMEPATH%\ .aws
```

5. .aws 디렉터리에서 credentials라는 파일을 새로 생성합니다.
6. IAM 콘솔에서 다운로드한 자격 증명 .csv 파일을 열어서 다음 형식을 사용하는 credentials 파일로 내용을 복사합니다.

```
[default]
aws_access_key_id = your_access_key_id
aws_secret_access_key = your_secret_access_key
```

7. credentials 파일을 저장하고 3단계에서 다운로드한 .csv 파일을 삭제합니다.

이제 공유 자격 증명 파일이 로컬 컴퓨터에 구성이 되고, AWS SDK에서 사용할 준비가 됩니다.

AWS SDK를 사용하여 인증된 요청 보내기

AWS SDK를 사용하여 인증된 요청을 보냅니다. 인증된 요청을 보내는 방법에 대한 자세한 내용은 [AWS 보안 자격 증명](#) 또는 [IAM Identity Center 인증](#)을 참조하세요.

Java

AWS SDK for Java를 사용하는 요청 인증에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 [공유 구성 및 자격 증명 파일을 사용하여 AWS SDK 및 도구를 전역적으로 구성](#) 및 [AWS SDK 및 도구를 사용한 인증 및 액세스](#)를 참조하세요.

.NET

AWS 계정 또는 IAM 사용자 자격 증명을 사용하여 인증된 요청을 보내려면 다음을 수행합니다.

- AmazonS3Client 클래스의 인스턴스를 만듭니다.
- AmazonS3Client 메서드 중 하나를 실행하여 Amazon S3으로 요청을 보냅니다. 클라이언트는 사용자가 제공하는 자격 증명에서 필요한 서명을 생성하여 Amazon S3으로 보내는 요청에 포함 시킵니다.

자세한 내용은 [AWS 계정 또는 IAM 사용자 자격 증명을 사용하여 요청하기](#)를 참조하세요.

Note

- 보안 자격 증명을 제공하지 않고 AmazonS3Client 클라이언트를 만들 수 있습니다. 이 클라이언트를 사용하여 보낸 요청은 서명이 없는 익명 요청입니다. Amazon S3는 공개적으로 사용할 수 없는 리소스에 대해 익명 요청을 보내면 오류를 반환합니다.
- AWS 계정을 생성하고 필요한 사용자를 생성할 수 있습니다. 이러한 사용자의 자격 증명을 관리할 수도 있습니다. 다음 예제의 작업을 수행하려면 이러한 자격 증명이 필요합니다. 자세한 내용은 SDK for .NET 개발자 안내서의 [AWS 자격 증명 구성](#)을 참조하세요.

그런 다음 프로파일과 자격 증명을 미리 검색하고 AWS 서비스 클라이언트를 생성할 때 해당 자격 증명을 명시적으로 사용하도록 애플리케이션을 구성할 수도 있습니다. 자세한 내용은 SDK for .NET 개발자 안내서의 [애플리케이션의 자격 증명 및 프로파일 액세스](#)를 참조하세요.

다음 C# 예제에서는 선행 작업을 수행하는 방법을 보여줍니다. 코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 [AWS SDK for .NET 시작하기](#)를 참조하세요.

Example

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
```

```
class MakeS3RequestTest
{
    private const string bucketName = "**** bucket name ****";
    // Specify your bucket region (an example region is shown).
    private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
    private static IAmazonS3 client;

    public static void Main()
    {
        using (client = new AmazonS3Client(bucketRegion))
        {
            Console.WriteLine("Listing objects stored in a bucket");
            ListingObjectsAsync().Wait();
        }
    }

    static async Task ListingObjectsAsync()
    {
        try
        {
            ListObjectsRequest request = new ListObjectsRequest
            {
                BucketName = bucketName,
                MaxKeys = 2
            };
            do
            {
                ListObjectsResponse response = await
client.ListObjectsAsync(request);
                // Process the response.
                foreach (S3Object entry in response.S3Objects)
                {
                    Console.WriteLine("key = {0} size = {1}",
                        entry.Key, entry.Size);
                }

                // If the response is truncated, set the marker to get the next
                // set of keys.
                if (response.IsTruncated)
                {
                    request.Marker = response.NextMarker;
                }
            }
            else
            {
            }
```

```

        {
            request = null;
        }
    } while (request != null);
}
catch (AmazonS3Exception e)
{
    Console.WriteLine("Error encountered on server. Message:'{0}' when
writing an object", e.Message);
}
catch (Exception e)
{
    Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
}
}
}
}

```

PHP

이 섹션에서는 AWS 계정 또는 IAM 사용자 자격 증명을 사용하여 인증된 요청을 보내기 위해 AWS SDK for PHP 버전 3의 클래스를 사용하는 방법을 설명합니다. AWS SDK for Ruby API에 대한 자세한 내용은 [AWS SDK for Ruby – 버전 2](#)를 참조하세요.

다음 PHP 예제에서는 클라이언트에서 보안 자격 증명을 사용하여 계정에 대한 모든 버킷을 나열하는 요청을 실행하는 방법을 보여줍니다.

Example

```

require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
]);

```

```
// Retrieve the list of buckets.
$result = $s3->listBuckets();

try {
    // Retrieve a paginator for listing objects.
    $objects = $s3->getPaginator('ListObjects', [
        'Bucket' => $bucket
    ]);

    echo "Keys retrieved!" . PHP_EOL;

    // Print the list of objects to the page.
    foreach ($objects as $object) {
        echo $object['Key'] . PHP_EOL;
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Note

보안 자격 증명을 제공하지 않고 S3Client 클라이언트를 만들 수 있습니다. 이 클라이언트를 사용하여 보낸 요청은 서명이 없는 익명 요청입니다. Amazon S3는 공개적으로 사용할 수 없는 리소스에 대해 익명 요청을 보내면 오류를 반환합니다. 자세한 내용은 [AWS SDK for PHP 설명서](#)의 [익명 클라이언트 생성](#)을 참조하세요.

Ruby

AWS SDK for Ruby 버전 3을 사용하여 Amazon S3에 호출할 수 있으려면 먼저 SDK가 버킷 및 객체에 대한 액세스를 확인하기 위해 사용하는 AWS 액세스 자격 증명을 설정해야 합니다. 로컬 시스템에서 AWS 자격 증명 프로파일에 설정된 자격 증명을 공유하는 경우 SDK for Ruby 버전 3이 코드에서 선언하지 않고도 이러한 자격 증명을 사용할 수 있습니다. 공유 자격 증명 설정에 대한 자세한 내용은 [AWS 계정 또는 IAM 사용자 자격 증명을 사용하여 요청하기](#)를 참조하세요.

다음 Ruby 코드 조각은 로컬 컴퓨터에서 공유되는 AWS 자격 증명 파일의 자격 증명을 사용하여 특정 버킷에서 모든 객체 키 이름을 가져오기 위한 요청을 인증합니다. 이 예제에서는 작업을 수행합니다.

1. `Aws::S3::Client` 클래스의 인스턴스를 만듭니다.

2. `list_objects_v2`의 `Aws::S3::Client` 메서드를 사용하여 버킷의 객체를 열거함으로써 Amazon S3에 요청합니다. 클라이언트는 컴퓨터의 AWS 자격 증명 파일에 있는 자격 증명에서 필요한 서명 값을 생성하여 Amazon S3로 전송하는 요청에 포함시킵니다.
3. 객체 키 이름 배열을 터미널에 인쇄합니다.

Example

```
# Prerequisites:
# - An existing Amazon S3 bucket.

require 'aws-sdk-s3'

# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if all operations succeed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_bucket_objects?(s3_client, 'amzn-s3-demo-bucket')
def list_bucket_objects?(s3_client, bucket_name)
  puts "Accessing the bucket named '#{bucket_name}'..."
  objects = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if objects.count.positive?
    puts 'The object keys in this bucket are (first 50 objects):'
    objects.contents.each do |object|
      puts object.key
    end
  else
    puts 'No objects found in this bucket.'
  end

  true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
  false
end

# Example usage:
```

```
def run_me
  region = 'us-west-2'
  bucket_name = 'BUCKET_NAME'
  s3_client = Aws::S3::Client.new(region: region)

  exit 1 unless list_bucket_objects?(s3_client, bucket_name)
end

run_me if $PROGRAM_NAME == __FILE__
```

로컬 AWS 자격 증명 파일을 가지고 있지 않더라도 여전히 `Aws::S3::Client` 리소스를 생성하고 Amazon S3 버킷 및 객체에 대해 코드를 실행할 수 있습니다. Ruby용 SDK 버전 3을 사용하여 보낸 요청은 기본적으로 서명이 없는 익명 요청입니다. Amazon S3은 공개적으로 사용할 수 없는 리소스에 대해 익명 요청을 보내면 오류를 반환합니다.

다음과 같이 Ruby용 SDK 애플리케이션을 위한 이전의 코드 조각을 사용 및 확장하는 등 보다 강력한 용도로 사용할 수 있습니다.

```
# Prerequisites:
# - An existing Amazon S3 bucket.

require 'aws-sdk-s3'

# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if all operations succeed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_bucket_objects?(s3_client, 'amzn-s3-demo-bucket')
def list_bucket_objects?(s3_client, bucket_name)
  puts "Accessing the bucket named '#{bucket_name}'..."
  objects = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if objects.count.positive?
    puts 'The object keys in this bucket are (first 50 objects):'
    objects.contents.each do |object|
      puts object.key
    end
  end
end
```

```

    else
      puts 'No objects found in this bucket.'
    end

    true
  rescue StandardError => e
    puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
    false
  end

  # Example usage:
  def run_me
    region = 'us-west-2'
    bucket_name = 'BUCKET_NAME'
    s3_client = Aws::S3::Client.new(region: region)

    exit 1 unless list_bucket_objects?(s3_client, bucket_name)
  end

  run_me if $PROGRAM_NAME == __FILE__

```

Go

Example

다음 예제에서는 공유 자격 증명 파일에서 Go용 SDK가 자동으로 로드한 AWS 자격 증명을 사용합니다.

```

package main

import (
    "context"
    "errors"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/smithy-go"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.

```

```
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    s3Client := s3.NewFromConfig(sdkConfig)
    count := 10
    fmt.Printf("Let's list up to %v buckets for your account.\n", count)
    result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
    if err != nil {
        var ae smithy.APIError
        if errors.As(err, &ae) && ae.ErrorCode() == "AccessDenied" {
            fmt.Println("You don't have permission to list buckets for this account.")
        } else {
            fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
        }
        return
    }
    if len(result.Buckets) == 0 {
        fmt.Println("You don't have any buckets!")
    } else {
        if count > len(result.Buckets) {
            count = len(result.Buckets)
        }
        for _, bucket := range result.Buckets[:count] {
            fmt.Printf("\t%v\n", *bucket.Name)
        }
    }
}
```

관련 리소스

- [AWS SDK를 사용하여 Amazon S3에서 개발](#)
- [AWS SDK for PHP Amazon S3 Aws\S3\S3Client Class용](#)
- [AWS SDK for PHP 설명서](#)

IAM 사용자 임시 자격 증명을 사용하여 요청하기

AWS 계정 또는 IAM 사용자는 임시 보안 자격 증명을 요청한 후 이 자격 증명을 사용하여 Amazon S3에 인증된 요청을 전송할 수 있습니다. 이 섹션에서는 AWS SDK for Java, .NET 및 PHP를 사용하여 임시 보안 자격 증명을 구하고 이 자격 증명을 사용하여 Amazon S3 요청을 인증하는 방법에 대한 예제를 제공합니다.

Java

IAM 사용자 또는 AWS 계정은 AWS SDK for Java를 사용하여 임시 보안 자격 증명을 요청([요청 만들기](#) 참조)한 후 이 자격 증명을 사용하여 Amazon S3에 액세스할 수 있습니다. 세션의 지정된 유효 기간이 만료된 후 이 자격 증명도 만료됩니다.

세션은 기본적으로 1시간 동안 지속됩니다. IAM 사용자 자격 증명을 사용하는 경우, 임시 보안 자격 증명을 요청할 수 있는 기간을 15분에서 역할의 최대 세션 기간까지 지정할 수 있습니다. 임시 보안 자격 증명에 대한 자세한 내용은 IAM 사용 설명서의 [임시 보안 자격 증명](#)을 참조하세요. 요청 생성에 대한 자세한 내용은 [요청 만들기](#) 섹션을 참조하세요.

임시 보안 자격 증명 가져오기 및 Amazon S3에 액세스

1. AWSSecurityTokenService 클래스의 인스턴스를 만듭니다.
2. Security Token Service(STS) 클라이언트의 `assumeRole()` 메서드를 호출하여 원하는 역할에 대한 임시 보안 자격 증명을 검색합니다.
3. `BasicSessionCredentials` 객체에 임시 보안 자격 증명을 패키지로 포함합니다. 이 객체를 사용하여 Amazon S3 클라이언트에 임시 보안 자격 증명을 제공합니다.
4. 임시 보안 자격 증명을 사용하여 `AmazonS3Client` 클래스의 인스턴스를 만듭니다. 이 클라이언트를 사용하여 Amazon S3에 요청을 보냅니다. 만료된 자격 증명을 사용하여 요청을 보낼 경우 Amazon S3에서 오류를 반환합니다.

Note

AWS 계정 보안 자격 증명을 사용하여 획득한 임시 자격 증명의 유효 기간은 1시간입니다. IAM 사용자 자격 증명을 사용하여 세션을 요청할 경우에만 세션 기간을 지정할 수 있습니다.

다음 예제는 지정된 버킷의 객체 키 세트를 나열합니다. 이 예제에서는 세션에 대한 임시 보안 자격 증명을 구한 다음 이러한 자격 증명을 사용하여 Amazon S3에 인증된 요청을 보냅니다.

IAM 사용자 자격 증명을 사용하여 샘플을 테스트하려면 AWS 계정에서 IAM 사용자를 생성해야 합니다. IAM 사용자를 만드는 방법에 대한 자세한 내용은 IAM 사용 설명서의 [첫 번째 IAM 사용자 및 관리자 그룹 생성](#)을 참조하세요.

실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 AWS SDK for Java 개발자 안내서의 [시작하기](#) 섹션을 참조하세요.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.securitytoken.AWSSecurityTokenService;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.AssumeRoleRequest;
import com.amazonaws.services.securitytoken.model.AssumeRoleResult;
import com.amazonaws.services.securitytoken.model.Credentials;

public class MakingRequestsWithIAMTempCredentials {
    public static void main(String[] args) {
        String clientRegion = "**** Client region ****";
        String roleARN = "**** ARN for role to be assumed ****";
        String roleSessionName = "**** Role session name ****";
        String bucketName = "**** Bucket name ****";

        try {
            // Creating the STS client is part of your trusted code. It has
            // the security credentials you use to obtain temporary security
            credentials.
            AWSSecurityTokenService stsClient =
            AWSSecurityTokenServiceClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .build();

            // Obtain credentials for the IAM role. Note that you cannot assume the
            role of
            // an AWS root account;
```

```

// Amazon S3 will deny access. You must use credentials for an IAM user
or an
// IAM role.
AssumeRoleRequest roleRequest = new AssumeRoleRequest()
    .withRoleArn(roleARN)
    .withRoleSessionName(roleSessionName);
AssumeRoleResult roleResponse = stsClient.assumeRole(roleRequest);
Credentials sessionCredentials = roleResponse.getCredentials();

// Create a BasicSessionCredentials object that contains the credentials
you
// just retrieved.
BasicSessionCredentials awsCredentials = new BasicSessionCredentials(
    sessionCredentials.getAccessKeyId(),
    sessionCredentials.getSecretAccessKey(),
    sessionCredentials.getSessionToken());

// Provide temporary security credentials so that the Amazon S3 client
// can send authenticated requests to Amazon S3. You create the client
// using the sessionCredentials object.
AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
    .withCredentials(new
AWSStaticCredentialsProvider(awsCredentials))
    .withRegion(clientRegion)
    .build();

// Verify that assuming the role worked and the permissions are set
correctly
// by getting a set of object keys from the bucket.
ObjectListing objects = s3Client.listObjects(bucketName);
System.out.println("No. of Objects: " +
objects.getObjectSummaries().size());
    } catch (AmazonServiceException e) {
        // The call was transmitted successfully, but Amazon S3 couldn't process
        // it, so it returned an error response.
        e.printStackTrace();
    } catch (SdkClientException e) {
        // Amazon S3 couldn't be contacted for a response, or the client
        // couldn't parse the response from Amazon S3.
        e.printStackTrace();
    }
}
}
}

```


.NET

IAM 사용자 또는 AWS 계정은 AWS SDK for .NET을 사용하여 임시 보안 자격 증명을 요청한 후 이 자격 증명을 사용하여 Amazon S3에 액세스할 수 있습니다. 세션의 유효 기간이 만료된 후 이 자격 증명도 만료됩니다.

세션은 기본적으로 1시간 동안 지속됩니다. IAM 사용자 자격 증명을 사용하는 경우, 임시 보안 자격 증명을 요청할 수 있는 기간을 15분에서 역할의 최대 세션 기간까지 지정할 수 있습니다. 임시 보안 자격 증명에 대한 자세한 내용은 IAM 사용 설명서의 [임시 보안 자격 증명](#)을 참조하세요. 요청 생성에 대한 자세한 내용은 [요청 만들기](#) 섹션을 참조하세요.

임시 보안 자격 증명 가져오기 및 Amazon S3에 액세스

1. AWS Security Token Service 클라이언트의 인스턴스 `AmazonSecurityTokenServiceClient`를 만듭니다.
2. 이전 단계에서 만든 STS 클라이언트의 `GetSessionToken` 메서드를 호출하여 세션을 시작합니다. `GetSessionTokenRequest` 객체를 사용하여 이 메서드에 세션 정보를 제공합니다.

메서드는 임시 보안 자격 증명을 반환합니다.

3. `SessionAWSCredentials` 객체의 인스턴스에 임시 보안 자격 증명을 패키지로 포함합니다. 이 객체를 사용하여 Amazon S3 클라이언트에 임시 보안 자격 증명을 제공합니다.
4. 임시 보안 자격 증명을 전달하여 `AmazonS3Client` 클래스의 인스턴스를 만듭니다. 이 클라이언트를 사용하여 Amazon S3에 요청을 보냅니다. 만료된 자격 증명을 사용하여 요청을 보낼 경우 Amazon S3에서 오류를 반환합니다.

Note

AWS 계정 보안 자격 증명을 사용하여 획득한 해당 자격 증명의 유효 기간은 1시간입니다. IAM 사용자 자격 증명을 사용하여 세션을 요청할 경우에만 세션 기간을 지정할 수 있습니다.

다음 C# 예제는 지정된 버킷의 객체 키를 나열합니다. 예를 들어, 이 예제에서는 기본 1시간의 세션에 대한 임시 보안 자격 증명을 구한 다음 이 자격 증명을 사용하여 Amazon S3에 인증된 요청을 보냅니다.

IAM 사용자 자격 증명을 사용하여 샘플을 테스트하려면 AWS 계정에서 IAM 사용자를 생성해야 합니다. IAM 사용자를 만드는 방법에 대한 자세한 내용은 IAM 사용 설명서의 [첫 번째 IAM 사용자 및 관리자 그룹 생성](#)을 참조하세요. 요청 생성에 대한 자세한 내용은 [요청 만들기](#) 섹션을 참조하세요.

코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 [AWS SDK for .NET 시작하기](#)를 참조하세요.

```
using Amazon;
using Amazon.Runtime;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TempCredExplicitSessionStartTest
    {
        private const string bucketName = "**** bucket name ****";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
        RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {
            ListObjectsAsync().Wait();
        }

        private static async Task ListObjectsAsync()
        {
            try
            {
                // Credentials use the default AWS SDK for .NET credential search
                chain.
                // On local development machines, this is your default profile.
                Console.WriteLine("Listing objects stored in a bucket");
                SessionAWSCredentials tempCredentials = await
                GetTemporaryCredentialsAsync();

                // Create a client by providing temporary security credentials.
```

```
        using (s3Client = new AmazonS3Client(tempCredentials, bucketRegion))
        {
            var listObjectRequest = new ListObjectsRequest
            {
                BucketName = bucketName
            };
            // Send request to Amazon S3.
            ListObjectsResponse response = await
s3Client.ListObjectsAsync(listObjectRequest);
            List<S3Object> objects = response.S3Objects;
            Console.WriteLine("Object count = {0}", objects.Count);
        }
    }
    catch (AmazonS3Exception s3Exception)
    {
        Console.WriteLine(s3Exception.Message, s3Exception.InnerException);
    }
    catch (AmazonSecurityTokenServiceException stsException)
    {
        Console.WriteLine(stsException.Message,
stsException.InnerException);
    }
}

private static async Task<SessionAWSCredentials>
GetTemporaryCredentialsAsync()
{
    using (var stsClient = new AmazonSecurityTokenServiceClient())
    {
        var getSessionTokenRequest = new GetSessionTokenRequest
        {
            DurationSeconds = 7200 // seconds
        };

        GetSessionTokenResponse sessionTokenResponse =
            await
stsClient.GetSessionTokenAsync(getSessionTokenRequest);

        Credentials credentials = sessionTokenResponse.Credentials;

        var sessionCredentials =
            new SessionAWSCredentials(credentials.AccessKeyId,
                                       credentials.SecretAccessKey,
                                       credentials.SessionToken);
    }
}
```

```

        return sessionCredentials;
    }
}
}
}

```

PHP

AWS SDK for Ruby API에 대한 자세한 내용은 [AWS SDK for Ruby – 버전 2](#)를 참조하세요.

IAM 사용자 또는 AWS 계정은 AWS SDK for PHP 버전 3을 사용하여 임시 보안 자격 증명을 요청할 수 있습니다. 그런 다음 얻은 임시 자격 증명을 사용하여 Amazon S3에 액세스할 수 있습니다. 세션의 유효 기간이 만료되면 이 자격 증명도 만료됩니다.

세션은 기본적으로 1시간 동안 지속됩니다. IAM 사용자 자격 증명을 사용하는 경우, 임시 보안 자격 증명을 요청할 수 있는 기간을 15분에서 역할의 최대 세션 기간까지 지정할 수 있습니다. 임시 보안 자격 증명에 대한 자세한 내용은 IAM 사용 설명서의 [임시 보안 자격 증명](#)을 참조하세요. 요청 생성에 대한 자세한 내용은 [요청 만들기](#) 섹션을 참조하세요.

Note

AWS 계정 보안 자격 증명을 사용하여 획득한 임시 보안 자격 증명의 유효 기간은 1시간입니다. IAM 사용자 자격 증명을 사용하여 세션을 요청할 경우에만 세션 기간을 지정할 수 있습니다.

Example

다음 PHP 예제는 임시 보안 자격 증명을 사용하여 지정된 버킷의 객체 키를 나열합니다. 이 예제에서는 기본 1시간의 세션에 대한 임시 보안 자격 증명을 구한 다음 이 자격 증명을 사용하여 Amazon S3에 인증된 요청을 보냅니다. AWS SDK for Ruby API에 대한 자세한 내용은 [AWS SDK for Ruby – 버전 2](#)를 참조하세요.

IAM 사용자 자격 증명을 사용하여 예제를 테스트하려면 AWS 계정에서 IAM 사용자를 생성해야 합니다. IAM 사용자를 만드는 방법에 대한 자세한 내용은 IAM 사용 설명서의 [첫 번째 IAM 사용자 및 관리자 그룹 생성](#)을 참조하세요. IAM 사용자의 자격 증명을 사용하여 세션을 요청할 경우 세션 기간을 설정하는 방법에 대한 예제는 [IAM 사용자 임시 자격 증명을 사용하여 요청하기](#) 섹션을 참조하세요.

```
require 'vendor/autoload.php';
```

```
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;

$bucket = '*** Your Bucket Name ***';

$sts = new StsClient([
    'version' => 'latest',
    'region' => 'us-east-1'
]);

$sessionToken = $sts->getSessionToken();

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
    'credentials' => [
        'key' => $sessionToken['Credentials']['AccessKeyId'],
        'secret' => $sessionToken['Credentials']['SecretAccessKey'],
        'token' => $sessionToken['Credentials']['SessionToken']
    ]
]);

$result = $s3->listBuckets();

try {
    // Retrieve a paginator for listing objects.
    $objects = $s3->getPaginator('ListObjects', [
        'Bucket' => $bucket
    ]);

    echo "Keys retrieved!" . PHP_EOL;

    // List objects
    foreach ($objects as $object) {
        echo $object['Key'] . PHP_EOL;
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Ruby

IAM 사용자 또는 AWS 계정은 AWS SDK for Ruby를 사용하여 임시 보안 자격 증명을 요청한 후 이 자격 증명을 사용하여 Amazon S3에 액세스할 수 있습니다. 세션의 유효 기간이 만료된 후 이 자격 증명도 만료됩니다.

세션은 기본적으로 1시간 동안 지속됩니다. IAM 사용자 자격 증명을 사용하는 경우, 임시 보안 자격 증명을 요청할 수 있는 기간을 15분에서 역할의 최대 세션 기간까지 지정할 수 있습니다. 임시 보안 자격 증명에 대한 자세한 내용은 IAM 사용 설명서의 [임시 보안 자격 증명](#)을 참조하세요. 요청 생성에 대한 자세한 내용은 [요청 만들기](#) 섹션을 참조하세요.

Note

AWS 계정 보안 자격 증명을 사용하여 획득한 임시 보안 자격 증명의 유효 기간은 1시간입니다. IAM 사용자 자격 증명을 사용하여 세션을 요청할 경우에만 세션 기간을 지정할 수 있습니다.

다음 Ruby 예제는 한 시간 동안 지정된 버킷의 항목을 나열하기 위해 임시 사용자를 생성합니다. 이 예제를 사용하려면 새 AWS Security Token Service(AWS STS) 클라이언트를 생성하고 Amazon S3 버킷을 나열하는데 필요한 권한을 가진 AWS 자격 증명에 있어야 합니다.

```
# Prerequisites:
# - A user in AWS Identity and Access Management (IAM). This user must
#   be able to assume the following IAM role. You must run this code example
#   within the context of this user.
# - An existing role in IAM that allows all of the Amazon S3 actions for all of the
#   resources in this code example. This role must also trust the preceding IAM
#   user.
# - An existing S3 bucket.

require 'aws-sdk-core'
require 'aws-sdk-s3'
require 'aws-sdk-iam'

# Checks whether a user exists in IAM.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [Boolean] true if the user exists; otherwise, false.
```

```

# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   exit 1 unless user_exists?(iam_client, 'my-user')
def user_exists?(iam_client, user_name)
  response = iam_client.get_user(user_name: user_name)
  return true if response.user.user_name
rescue Aws::IAM::Errors::NoSuchEntity
  # User doesn't exist.
rescue StandardError => e
  puts 'Error while determining whether the user ' \
    "'#{user_name}' exists: #{e.message}"
end

# Creates a user in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [AWS::IAM::Types::User] The new user.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   user = create_user(iam_client, 'my-user')
#   exit 1 unless user.user_name
def create_user(iam_client, user_name)
  response = iam_client.create_user(user_name: user_name)
  response.user
rescue StandardError => e
  puts "Error while creating the user '#{user_name}': #{e.message}"
end

# Gets a user in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [AWS::IAM::Types::User] The existing user.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   user = get_user(iam_client, 'my-user')
#   exit 1 unless user.user_name
def get_user(iam_client, user_name)
  response = iam_client.get_user(user_name: user_name)
  response.user
rescue StandardError => e
  puts "Error while getting the user '#{user_name}': #{e.message}"
end

```

```

# Checks whether a role exists in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The role's name.
# @return [Boolean] true if the role exists; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   exit 1 unless role_exists?(iam_client, 'my-role')
def role_exists?(iam_client, role_name)
  response = iam_client.get_role(role_name: role_name)
  return true if response.role.role_name
rescue StandardError => e
  puts 'Error while determining whether the role ' \
    "'#{role_name}' exists: #{e.message}"
end

# Gets credentials for a role in IAM.
#
# @param sts_client [Aws::STS::Client] An initialized AWS STS client.
# @param role_arn [String] The role's Amazon Resource Name (ARN).
# @param role_session_name [String] A name for this role's session.
# @param duration_seconds [Integer] The number of seconds this session is valid.
# @return [AWS::AssumeRoleCredentials] The credentials.
# @example
#   sts_client = Aws::STS::Client.new(region: 'us-west-2')
#   credentials = get_credentials(
#     sts_client,
#     'arn:aws:iam::123456789012:role/AmazonS3ReadOnly',
#     'ReadAmazonS3Bucket',
#     3600
#   )
#   exit 1 if credentials.nil?
def get_credentials(sts_client, role_arn, role_session_name, duration_seconds)
  Aws::AssumeRoleCredentials.new(
    client: sts_client,
    role_arn: role_arn,
    role_session_name: role_session_name,
    duration_seconds: duration_seconds
  )
rescue StandardError => e
  puts "Error while getting credentials: #{e.message}"
end

```



```

# Checks whether a bucket exists in Amazon S3.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The name of the bucket.
# @return [Boolean] true if the bucket exists; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless bucket_exists?(s3_client, 'amzn-s3-demo-bucket')
def bucket_exists?(s3_client, bucket_name)
  response = s3_client.list_buckets
  response.buckets.each do |bucket|
    return true if bucket.name == bucket_name
  end
rescue StandardError => e
  puts "Error while checking whether the bucket '#{bucket_name}' " \
    "exists: #{e.message}"
end

# Lists the keys and ETags for the objects in an Amazon S3 bucket.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if the objects were listed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_objects_in_bucket?(s3_client, 'amzn-s3-demo-bucket')
def list_objects_in_bucket?(s3_client, bucket_name)
  puts "Accessing the contents of the bucket named '#{bucket_name}'..."
  response = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if response.count.positive?
    puts "Contents of the bucket named '#{bucket_name}' (first 50 objects):"
    puts 'Name => ETag'
    response.contents.each do |obj|
      puts "#{obj.key} => #{obj.etag}"
    end
  else
    puts "No objects in the bucket named '#{bucket_name}'."
  end
  true
rescue StandardError => e

```

```
puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"  
end
```

연합된 사용자의 임시 자격 증명을 사용하여 요청하기

임시 보안 자격 증명을 요청하고 이를 AWS 리소스에 액세스해야 하는 페더레이션 사용자 또는 애플리케이션에 제공할 수 있습니다. 이 섹션에서는 AWS SDK를 사용하여 페더레이션 사용자 또는 애플리케이션에 대한 임시 보안 자격 증명을 가져오고 이러한 자격 증명을 사용하여 인증된 요청을 Amazon S3로 보내는 방법에 대한 예제를 제공합니다. 사용 가능한 AWS SDK 목록은 [샘플 코드 및 라이브러리](#)를 참조하세요.

Note

AWS 계정 및 IAM 사용자 모두 페더레이션 사용자에게 대해 임시 보안 자격 증명을 요청할 수 있습니다. 그러나 보안 강화를 위해 필요한 권한이 있는 IAM 사용자만 이러한 임시 자격 증명을 요청하여 연합된 사용자가 요청하는 IAM 사용자의 권한을 최대한 얻을 수 있도록 해야 됩니다. 일부 애플리케이션에서는 연합된 사용자 및 애플리케이션에 임시 보안 자격 증명을 부여하기 위한 목적으로만 특정 권한이 있는 IAM 사용자를 만드는 것이 적합할 수도 있습니다.

Java

AWS 리소스에 액세스하기 위해 인증된 요청을 보낼 수 있도록 페더레이션 사용자 및 애플리케이션에 대해 임시 보안 자격 증명을 제공할 수 있습니다. 이러한 임시 자격 증명을 요청할 때는 사용자 이름과 부여할 리소스 권한을 설명하는 IAM 정책을 제공해야 합니다. 세션은 기본적으로 1시간 동안 지속됩니다. 연합된 사용자 및 애플리케이션에 대한 임시 보안 자격 증명을 요청할 때는 다른 기간 값을 명시적으로 설정할 수 있습니다.

Note

연합된 사용자 및 애플리케이션에 대한 임시 보안 자격 증명을 요청할 때 보안 강화를 위해 필요한 액세스 권한만 가진 전용 IAM 사용자를 사용하는 것이 좋습니다. 생성하는 임시 사용자는 임시 보안 자격 증명을 요청한 IAM 사용자보다 더 많은 권한을 가질 수 없습니다. 자세한 내용은 [AWS Identity and Access Management FAQ](#)를 참조하세요.

리소스에 액세스하기 위해 보안 자격 증명을 제공하고 인증된 요청을 보내려면 다음을 수행합니다.

- `AWSSecurityTokenServiceClient` 클래스의 인스턴스를 만듭니다.
- 보안 토큰 서비스(STS) 클라이언트의 `getFederationToken()` 메서드를 호출하여 세션을 시작합니다. 임시 자격 증명에 연결하려는 IAM 정책 및 사용자 이름을 비롯한 세션 정보를 제공합니다. 선택적 세션 기간을 제공할 수 있습니다. 이 메서드는 임시 보안 자격 증명을 반환합니다.

- `BasicSessionCredentials` 객체의 인스턴스에 임시 보안 자격 증명을 패키지로 포함합니다. 이 객체를 사용하여 Amazon S3 클라이언트에 임시 보안 자격 증명을 제공합니다.
- 임시 보안 자격 증명을 사용하여 `AmazonS3Client` 클래스의 인스턴스를 만듭니다. 이 클라이언트를 사용하여 Amazon S3에 요청을 보냅니다. 만료된 자격 증명을 사용하여 요청을 보낼 경우 Amazon S3에서 오류를 반환합니다.

Example

다음 예제는 지정된 S3 버킷의 키를 나열합니다. 이 예제에서는 연합된 사용자를 위해 2시간의 세션에 대한 임시 보안 자격 증명을 가져오고 이 자격 증명을 사용하여 인증된 요청을 Amazon S3으로 보냅니다. 이 예제를 실행하려면 사용자가 임시 보안 자격 증명을 요청하고 AWS 리소스를 나열하도록 허용하는 연결된 정책을 사용해 IAM 사용자를 생성해야 합니다. 다음 정책을 통해 이러한 작업을 수행할 수 있습니다.

```
{
  "Statement": [{
    "Action": ["s3:ListBucket",
      "sts:GetFederationToken"],
    "Effect": "Allow",
    "Resource": "*"
  }]
}
```

IAM 사용자를 만드는 방법에 대한 자세한 내용은 IAM 사용 설명서의 [첫 번째 IAM 사용자 및 관리자 그룹 생성](#)을 참조하세요.

IAM 사용자를 생성한 후 이전 정책을 연결하면 다음 예제를 실행할 수 있습니다. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 AWS SDK for Java 개발자 안내서의 [시작하기](#) 섹션을 참조하세요.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSSessionCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Resource;
import com.amazonaws.auth.policy.Statement;
```

```
import com.amazonaws.auth.policy.Statement.Effect;
import com.amazonaws.auth.policy.actions.S3Actions;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.securitytoken.AWSSecurityTokenService;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.Credentials;
import com.amazonaws.services.securitytoken.model.GetFederationTokenRequest;
import com.amazonaws.services.securitytoken.model.GetFederationTokenResult;

import java.io.IOException;

public class MakingRequestsWithFederatedTempCredentials {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "**** Specify bucket name ****";
        String federatedUser = "**** Federated user name ****";
        String resourceARN = "arn:aws:s3:::" + bucketName;

        try {
            AWSSecurityTokenService stsClient = AWSSecurityTokenServiceClientBuilder
                .standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .build();

            GetFederationTokenRequest getFederationTokenRequest = new
            GetFederationTokenRequest();
            getFederationTokenRequest.setDurationSeconds(7200);
            getFederationTokenRequest.setName(federatedUser);

            // Define the policy and add it to the request.
            Policy policy = new Policy();
            policy.withStatements(new Statement(Effect.Allow)
                .withActions(S3Actions.ListObjects)
                .withResources(new Resource(resourceARN)));
            getFederationTokenRequest.setPolicy(policy.toJson());

            // Get the temporary security credentials.
```

```

        GetFederationTokenResult federationTokenResult =
stsClient.getFederationToken(getFederationTokenRequest);
        Credentials sessionCredentials = federationTokenResult.getCredentials();

        // Package the session credentials as a BasicSessionCredentials
        // object for an Amazon S3 client object to use.
        BasicSessionCredentials basicSessionCredentials = new
BasicSessionCredentials(
            sessionCredentials.getAccessKeyId(),
            sessionCredentials.getSecretAccessKey(),
            sessionCredentials.getSessionToken());
        AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
            .withCredentials(new
AWSStaticCredentialsProvider(basicSessionCredentials))
            .withRegion(clientRegion)
            .build();

        // To verify that the client works, send a listObjects request using
        // the temporary security credentials.
        ObjectListing objects = s3Client.listObjects(bucketName);
        System.out.println("No. of Objects = " +
objects.getObjectSummaries().size());
    } catch (AmazonServiceException e) {
        // The call was transmitted successfully, but Amazon S3 couldn't process
        // it, so it returned an error response.
        e.printStackTrace();
    } catch (SdkClientException e) {
        // Amazon S3 couldn't be contacted for a response, or the client
        // couldn't parse the response from Amazon S3.
        e.printStackTrace();
    }
}
}

```

.NET

AWS 리소스에 액세스하기 위해 인증된 요청을 보낼 수 있도록 페더레이션 사용자 및 애플리케이션에 대해 임시 보안 자격 증명을 제공할 수 있습니다. 이러한 임시 자격 증명을 요청할 때는 사용자 이름과 부여할 리소스 권한을 설명하는 IAM 정책을 제공해야 합니다. 세션은 기본적으로 1시간 동안 지속됩니다. 연합된 사용자 및 애플리케이션에 대한 임시 보안 자격 증명을 요청할 때는 다른 기간 값을 명시적으로 설정할 수 있습니다. 인증된 요청 전송에 대한 자세한 내용은 [요청 만들기](#) 단원을 참조하십시오.

Note

연합된 사용자 및 애플리케이션에 대한 임시 보안 자격 증명을 요청할 때 보안 강화를 위해 필요한 액세스 권한만 가진 전용 IAM 사용자를 사용하는 것이 좋습니다. 생성하는 임시 사용자는 임시 보안 자격 증명을 요청한 IAM 사용자보다 더 많은 권한을 가질 수 없습니다. 자세한 내용은 [AWS Identity and Access Management FAQ](#)를 참조하세요.

다음을 수행합니다.

- AWS Security Token Service 클라이언트의 인스턴스인 `AmazonSecurityTokenServiceClient` 클래스를 만듭니다.
- STS 클라이언트의 `GetFederationToken` 메서드를 호출하여 세션을 시작합니다. 임시 자격 증명에 연결하려는 IAM 정책 및 사용자 이름을 비롯한 세션 정보를 제공해야 합니다. 경우에 따라 세션 기간을 제공할 수 있습니다. 이 메서드는 임시 보안 자격 증명을 반환합니다.
- `SessionAWSCredentials` 객체의 인스턴스에 임시 보안 자격 증명을 패키지로 포함합니다. 이 객체를 사용하여 Amazon S3 클라이언트에 임시 보안 자격 증명을 제공합니다.
- 임시 보안 자격 증명을 전달하여 `AmazonS3Client` 클래스의 인스턴스를 만듭니다. 이 클라이언트를 사용하여 Amazon S3에 요청을 보냅니다. 만료된 자격 증명을 사용하여 요청을 보낼 경우 Amazon S3에서 오류를 반환합니다.

Example

다음 C# 예제는 지정된 버킷의 키를 나열합니다. 이 예제에서는 먼저 연합된 사용자(User1)를 위해 2시간의 세션에 대한 임시 보안 자격 증명을 가져오고 이 자격 증명을 사용하여 인증된 요청을 Amazon S3으로 보냅니다.

- 이 연습에서는 최소한의 권한을 가진 IAM 사용자를 생성합니다. 이 IAM 사용자의 자격 증명을 사용하여 다른 사용자에 대한 임시 자격 증명을 요청합니다. 이 예제는 특정 버킷의 객체만 나열합니다. 다음과 같은 정책을 연결하여 IAM 사용자를 생성합니다.

```
{
  "Statement": [{
    "Action": ["s3:ListBucket",
      "sts:GetFederationToken*"],
    "Effect": "Allow",
    "Resource": "*"
  }]
```

```

    }
  ]
}
```

이 정책은 IAM 사용자에게 임시 보안 자격 증명을 요청하고, AWS 리소스를 나열할 수 있는 액세스 권한만 허용합니다. IAM 사용자를 만드는 방법에 대한 자세한 내용은 IAM 사용 설명서의 [IAM 사용자 및 관리자 그룹 생성](#)을 참조하세요.

- IAM 사용자 보안 자격 증명을 사용하여 다음 예제를 테스트합니다. 이 예제에서는 임시 보안 자격 증명을 사용하여 인증된 요청을 Amazon S3으로 보냅니다. 이 예제에서는 연합된 사용자(User1)에 대한 임시 보안 자격 증명을 요청할 때 특정 버킷(YourBucketName)의 객체를 나열하는 데 필요한 액세스 권한을 제한하는 다음 정책을 지정합니다. 정책을 업데이트하고 자신의 기존 버킷 이름을 제공해야 합니다.

```

{
  "Statement": [
    {
      "Sid": "1",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::YourBucketName"
    }
  ]
}
```

• Example

다음 샘플을 업데이트하고 이전 연동 사용자 액세스 정책에 지정한 버킷 이름을 제공합니다.

코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 [AWS SDK for .NET 시작하기](#)를 참조하세요.

```

using Amazon;
using Amazon.Runtime;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
```



```
{
    class TempFederatedCredentialsTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            ListObjectsAsync().Wait();
        }

        private static async Task ListObjectsAsync()
        {
            try
            {
                Console.WriteLine("Listing objects stored in a bucket");
                // Credentials use the default AWS SDK for .NET credential search
chain.

                // On local development machines, this is your default profile.
                SessionAWSCredentials tempCredentials =
                    await GetTemporaryFederatedCredentialsAsync();

                // Create a client by providing temporary security credentials.
                using (client = new AmazonS3Client(bucketRegion))
                {
                    ListObjectsRequest listObjectRequest = new
ListObjectsRequest();
                    listObjectRequest.BucketName = bucketName;

                    ListObjectsResponse response = await
client.ListObjectsAsync(listObjectRequest);
                    List<S3Object> objects = response.S3Objects;
                    Console.WriteLine("Object count = {0}", objects.Count);

                    Console.WriteLine("Press any key to continue...");
                    Console.ReadKey();
                }
            }
            catch (AmazonS3Exception e)
            {
            }
        }
    }
}
```

```

        Console.WriteLine("Error encountered ***. Message:'{0}' when
writing an object", e.Message);
    }
    catch (Exception e)
    {
        Console.WriteLine("Unknown encountered on server. Message:'{0}'
when writing an object", e.Message);
    }
}

private static async Task<SessionAWSCredentials>
GetTemporaryFederatedCredentialsAsync()
{
    AmazonSecurityTokenServiceConfig config = new
AmazonSecurityTokenServiceConfig();
    AmazonSecurityTokenServiceClient stsClient =
        new AmazonSecurityTokenServiceClient(
            config);

    GetFederationTokenRequest federationTokenRequest =
        new GetFederationTokenRequest();
    federationTokenRequest.DurationSeconds = 7200;
    federationTokenRequest.Name = "User1";
    federationTokenRequest.Policy = @"{
        ""Statement"":
        [
            {
                ""Sid"":""Stmnt1311212314284"",
                ""Action"":[""s3:ListBucket""],
                ""Effect"":""Allow"",
                ""Resource"":""arn:aws:s3:::" + bucketName + @""
            }
        ]
    }
";

    GetFederationTokenResponse federationTokenResponse =
        await
stsClient.GetFederationTokenAsync(federationTokenRequest);
    Credentials credentials = federationTokenResponse.Credentials;

    SessionAWSCredentials sessionCredentials =
        new SessionAWSCredentials(credentials.AccessKeyId,
            credentials.SecretAccessKey,

```

```

        credentials.SessionToken);

    return sessionCredentials;
}
}
}

```

PHP

이 주제에서는 AWS SDK for PHP 버전 3의 클래스를 사용하여 연합된 사용자 및 애플리케이션에 대한 임시 보안 자격 증명을 요청하고 이를 사용하여 Amazon S3에 저장된 리소스에 액세스하는 방법을 설명합니다. AWS SDK for Ruby API에 대한 자세한 내용은 [AWS SDK for Ruby – 버전 2](#)를 참조하세요.

AWS 리소스에 액세스하기 위해 인증된 요청을 보낼 수 있도록 페더레이션 사용자 및 애플리케이션에 임시 보안 자격 증명을 제공할 수 있습니다. 이러한 임시 자격 증명을 요청할 때는 사용자 이름과 부여할 리소스 권한을 설명하는 IAM 정책을 제공해야 합니다. 세션의 유효 기간이 만료되면 이 자격 증명도 만료됩니다. 세션은 기본적으로 1시간 동안 지속됩니다. 연동 사용자 및 애플리케이션에 대한 임시 보안 자격 증명을 요청할 때는 다른 기간 값을 명시적으로 설정할 수 있습니다. 임시 보안 자격 증명에 대한 자세한 내용은 IAM 사용 설명서의 [임시 보안 자격 증명](#)을 참조하세요. 연동 사용자와 애플리케이션에 임시 보안 자격 증명 제공에 대한 자세한 내용은 [요청 만들기](#) 단원을 참조하십시오.

연합된 사용자 및 애플리케이션에 대한 임시 보안 자격 증명을 요청할 때 보안 강화를 위해 필요한 액세스 권한만 가진 전용 IAM 사용자를 사용하는 것이 좋습니다. 생성하는 임시 사용자는 임시 보안 자격 증명을 요청한 IAM 사용자보다 더 많은 권한을 가질 수 없습니다. 자격 증명 연동에 대한 자세한 내용은 [AWS Identity and Access Management FAQ](#)를 참조하세요.

AWS SDK for Ruby API에 대한 자세한 내용은 [AWS SDK for Ruby – 버전 2](#)를 참조하세요.

Example

다음 PHP 예제는 지정된 버킷의 키를 나열합니다. 이 예제에서는 먼저 연동 사용자(User1)를 위해 1시간짜리 세션에 대한 임시 보안 자격 증명을 가져오고 가져온 임시 보안 자격 증명을 사용하여 인증된 요청을 Amazon S3으로 보냅니다.

다른 사용자에 대한 임시 보안 자격 증명을 요청할 경우, 보안 강화를 위해 임시 자격 증명을 요청할 수 있는 IAM 사용자의 보안 자격 증명을 사용할 수 있습니다. IAM 사용자가 연합된 사용자에게 최소한의 애플리케이션별 권한만 부여하도록 이 IAM 사용자의 액세스 권한을 제한할 수도 있습니다. 이 예제는 특정 버킷의 객체만 나열합니다. 다음과 같은 정책을 연결하여 IAM 사용자를 생성합니다.

```
{
  "Statement": [{
    "Action": ["s3:ListBucket",
      "sts:GetFederationToken*"],
    "Effect": "Allow",
    "Resource": "*"
  }]
}
```

이 정책은 IAM 사용자에게 임시 보안 자격 증명을 요청하고, AWS 리소스를 나열할 수 있는 액세스 권한만 허용합니다. IAM 사용자를 만드는 방법에 대한 자세한 내용은 IAM 사용 설명서의 [첫 번째 IAM 사용자 및 관리자 그룹 생성](#)을 참조하세요.

이제 IAM 사용자 보안 자격 증명을 사용하여 다음 예제를 테스트할 수 있습니다. 이 예제에서는 임시 보안 자격 증명을 사용하여 인증된 요청을 Amazon S3으로 보냅니다. 이 예제에서는 연동 사용자(User1)에 대한 임시 보안 자격 증명을 요청할 때 특정 버킷의 객체를 나열하는 데 필요한 액세스 권한을 제한하는 다음 정책을 지정합니다. 정책을 자신의 버킷 이름으로 업데이트합니다.

```
{
  "Statement": [
    {
      "Sid": "1",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::YourBucketName"
    }
  ]
}
```

다음 예제에서는 정책 리소스를 지정할 때 YourBucketName을 자신의 기존 버킷 이름으로 대체합니다.

```
require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;

$bucket = '*** Your Bucket Name ***';
```

```
// In real applications, the following code is part of your trusted code. It has
// the security credentials that you use to obtain temporary security credentials.
$sts = new StsClient([
    'version' => 'latest',
    'region' => 'us-east-1'
]);

// Fetch the federated credentials.
$sessionToken = $sts->getFederationToken([
    'Name' => 'User1',
    'DurationSeconds' => '3600',
    'Policy' => json_encode([
        'Statement' => [
            'Sid' => 'randomstatementid' . time(),
            'Action' => ['s3:ListBucket'],
            'Effect' => 'Allow',
            'Resource' => 'arn:aws:s3:::' . $bucket
        ]
    ])
]);

// The following will be part of your less trusted code. You provide temporary
// security credentials so the code can send authenticated requests to Amazon S3.

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
    'credentials' => [
        'key' => $sessionToken['Credentials']['AccessKeyId'],
        'secret' => $sessionToken['Credentials']['SecretAccessKey'],
        'token' => $sessionToken['Credentials']['SessionToken']
    ]
]);

try {
    $result = $s3->listObjects([
        'Bucket' => $bucket
    ]);
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Ruby

AWS 리소스에 액세스하기 위해 인증된 요청을 보낼 수 있도록 페더레이션 사용자 및 애플리케이션에 대해 임시 보안 자격 증명을 제공할 수 있습니다. IAM 서비스로부터 이러한 임시 자격 증명을 요청할 경우 사용자 이름과 부여할 리소스 권한을 설명하는 IAM 정책을 제공해야 합니다. 세션은 기본적으로 1시간 동안 지속됩니다. 하지만 IAM 사용자 자격 증명을 사용하여 임시 자격 증명을 요청하면, 연합된 사용자 및 애플리케이션에 대해 임시 보안 자격 증명을 요청할 기간에 대해 다른 값을 명시적으로 설정할 수 있습니다. 연동 사용자와 애플리케이션의 임시 보안 자격 증명에 대한 자세한 내용은 [요청 만들기](#) 단원을 참조하십시오.

Note

연합된 사용자 및 애플리케이션에 대한 임시 보안 자격 증명을 요청할 때 보안 강화를 위해 필요한 액세스 권한만 가진 전용 IAM 사용자를 사용하는 것이 좋습니다. 생성하는 임시 사용자는 임시 보안 자격 증명을 요청한 IAM 사용자보다 더 많은 권한을 가질 수 없습니다. 자세한 내용은 [AWS Identity and Access Management FAQ](#)를 참조하세요.

Example

다음 Ruby 코드 예제에서는 제한된 권한을 가진 연동 사용자가 지정된 버킷의 키를 나열하도록 합니다.

```
# Prerequisites:
# - An existing Amazon S3 bucket.

require 'aws-sdk-s3'
require 'aws-sdk-iam'
require 'json'

# Checks to see whether a user exists in IAM; otherwise,
# creates the user.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [Aws::IAM::Types::User] The existing or new user.
# @example
#   iam = Aws::IAM::Client.new(region: 'us-west-2')
#   user = get_user(iam, 'my-user')
#   exit 1 unless user.user_name
#   puts "User's name: #{user.user_name}"
```

```

def get_user(iam, user_name)
  puts "Checking for a user with the name '#{user_name}'..."
  response = iam.get_user(user_name: user_name)
  puts "A user with the name '#{user_name}' already exists."
  response.user
# If the user doesn't exist, create them.
rescue Aws::IAM::Errors::NoSuchEntity
  puts "A user with the name '#{user_name}' doesn't exist. Creating this user..."
  response = iam.create_user(user_name: user_name)
  iam.wait_until(:user_exists, user_name: user_name)
  puts "Created user with the name '#{user_name}'."
  response.user
rescue StandardError => e
  puts "Error while accessing or creating the user named '#{user_name}':
#{e.message}"
end

# Gets temporary AWS credentials for an IAM user with the specified permissions.
#
# @param sts [Aws::STS::Client] An initialized AWS STS client.
# @param duration_seconds [Integer] The number of seconds for valid credentials.
# @param user_name [String] The user's name.
# @param policy [Hash] The access policy.
# @return [Aws::STS::Types::Credentials] AWS credentials for API authentication.
# @example
#   sts = Aws::STS::Client.new(region: 'us-west-2')
#   credentials = get_temporary_credentials(sts, duration_seconds, user_name,
#     {
#       'Version' => '2012-10-17',
#       'Statement' => [
#         'Sid' => 'Stmt1',
#         'Effect' => 'Allow',
#         'Action' => 's3:ListBucket',
#         'Resource' => 'arn:aws:s3:::amzn-s3-demo-bucket'
#       ]
#     }
#   )
#   exit 1 unless credentials.access_key_id
#   puts "Access key ID: #{credentials.access_key_id}"
def get_temporary_credentials(sts, duration_seconds, user_name, policy)
  response = sts.get_federation_token(
    duration_seconds: duration_seconds,
    name: user_name,
    policy: policy.to_json
  )
end

```

```

    )
    response.credentials
  rescue StandardError => e
    puts "Error while getting federation token: #{e.message}"
  end

# Lists the keys and ETags for the objects in an Amazon S3 bucket.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if the objects were listed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_objects_in_bucket?(s3_client, 'amzn-s3-demo-bucket')
def list_objects_in_bucket?(s3_client, bucket_name)
  puts "Accessing the contents of the bucket named '#{bucket_name}'..."
  response = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if response.count.positive?
    puts "Contents of the bucket named '#{bucket_name}' (first 50 objects):"
    puts 'Name => ETag'
    response.contents.each do |obj|
      puts "#{obj.key} => #{obj.etag}"
    end
  else
    puts "No objects in the bucket named '#{bucket_name}'."
  end
  true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
end

# Example usage:
def run_me
  region = "us-west-2"
  user_name = "my-user"
  bucket_name = "amzn-s3-demo-bucket"

  iam = Aws::IAM::Client.new(region: region)
  user = get_user(iam, user_name)

```



```

exit 1 unless user.user_name

puts "User's name: #{user.user_name}"
sts = Aws::STS::Client.new(region: region)
credentials = get_temporary_credentials(sts, 3600, user_name,
                                         {
                                           'Version' => '2012-10-17',
                                           'Statement' => [
                                             'Sid' => 'Stmt1',
                                             'Effect' => 'Allow',
                                             'Action' => 's3:ListBucket',
                                             'Resource' =>
"arn:aws:s3:::#{bucket_name}"
                                           ]
                                         })

exit 1 unless credentials.access_key_id

puts "Access key ID: #{credentials.access_key_id}"
s3_client = Aws::S3::Client.new(region: region, credentials: credentials)

exit 1 unless list_objects_in_bucket?(s3_client, bucket_name)
end

run_me if $PROGRAM_NAME == __FILE__

```

REST API를 사용하여 요청

이 단원에는 REST API를 사용하여 Amazon S3 엔드포인트에 요청하는 방법에 대한 정보가 포함되어 있습니다. Amazon S3 엔드포인트 목록은 AWS 일반 참조의 [리전 및 엔드포인트](#)를 참조하세요.

REST API 요청에 대한 S3 호스트 이름 구성

Amazon S3 엔드포인트는 아래에 표시된 구조를 따릅니다.

`s3.Region.amazonaws.com`

Amazon S3 액세스 포인트 엔드포인트 및 듀얼 스택 엔드포인트도 다음의 표준 구조를 따릅니다.

- Amazon S3 액세스 포인트 - `s3-accesspoint.Region.amazonaws.com`
- 듀얼 스택 - `s3.dualstack.Region.amazonaws.com`

Amazon S3 리전 및 엔드포인트의 전체 목록은 Amazon Web Services 일반 참조에서 [Amazon S3 엔드포인트 및 할당량](#)을 참조하세요.

가상 호스팅 방식 및 경로 방식 요청

REST API를 사용하여 요청할 때 Amazon S3 엔드포인트에 대해 가상 호스팅 방식 또는 경로 방식 URI를 사용할 수 있습니다. 자세한 내용은 [경로 방식 요청](#)을 참조하세요.

Example가상 호스팅 방식 요청

다음은 미국 서부(오레곤) 리전의 puppy.jpg 버킷에서 examplebucket 파일을 삭제하기 위한 가상 호스팅 방식 요청의 예제입니다. 가상 호스팅 방식 요청에 대한 자세한 내용은 [경로 방식 요청](#)을 참조하세요.

```
DELETE /puppy.jpg HTTP/1.1
Host: examplebucket.s3.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

Example경로 방식 요청

다음은 경로 방식의 메서드로 동일한 요청을 하는 예제입니다.

```
DELETE /examplebucket/puppy.jpg HTTP/1.1
Host: s3.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

현재 Amazon S3는 모든 AWS 리전에서 가상 호스팅 방식 및 경로 방식 URL 액세스를 모두 지원합니다. 그러나 경로 스타일 URL은 향후 중단될 예정입니다. 자세한 내용은 다음 중요 참고 사항을 참조하세요.

경로 방식 요청에 대한 자세한 내용은 [경로 방식 요청](#)을 참조하세요.

⚠ Important

업데이트(2020년 9월 23일) – 고객이 가상 호스팅 스타일 URL로 전환하는 데 필요한 시간을 가질 수 있도록 경로 스타일 URL의 사용 중단을 연기하기로 결정했습니다. 자세한 내용은 AWS 뉴스 블로그에서 [Amazon S3 경로 사용 중지 계획 - 나머지 이야기](#)를 참조하세요.

REST API를 사용하여 듀얼 스택 엔드포인트에 요청

REST API를 사용할 때 가상 호스팅 방식 또는 경로 방식 엔드포인트 이름(URI)을 사용하여 듀얼 스택 엔드포인트에 직접 액세스할 수 있습니다. 모든 Amazon S3 듀얼 스택 엔드포인트 이름에는 리전이 포함되어 있습니다. 표준 IPv4-only 엔드포인트와 달리, 가상 호스팅 방식 및 경로 방식 엔드포인트는 모두 리전별 엔드포인트 이름을 사용합니다.

Example가상 호스팅 방식 듀얼 스택 엔드포인트 요청

미국 서부(오레곤) 리전의 puppy.jpg 버킷에서 examplebucket 객체를 검색하는 다음 예제와 같이 REST 요청에 가상 호스팅 방식 엔드포인트를 사용할 수 있습니다.

```
GET /puppy.jpg HTTP/1.1
Host: examplebucket.s3.dualstack.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

Example경로 방식 듀얼 스택 엔드포인트 요청

또는 다음 예제와 같이 요청에 경로 방식 엔드포인트를 사용할 수 있습니다.

```
GET /examplebucket/puppy.jpg HTTP/1.1
Host: s3.dualstack.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

듀얼 스택 엔드포인트에 대한 자세한 내용은 [Amazon S3 듀얼 스택 엔드포인트 사용](#)을 참조하십시오.

REST API를 사용하여 요청하는 방법에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- [요청 리디렉션 및 REST API](#)

- [라우팅 요청](#)

요청 리디렉션 및 REST API

주제

- [리디렉션 및 HTTP 사용자 에이전트](#)
- [리디렉션 및 100-Continue](#)
- [리디렉션 예제](#)

이 단원에서는 Amazon S3 REST API를 사용하여 HTTP 리디렉션을 처리하는 방법에 대해 설명합니다. Amazon S3 리디렉션에 대한 일반 정보는 Amazon Simple Storage Service API 참조의 [요청 만들기](#) 섹션을 참조하세요.

리디렉션 및 HTTP 사용자 에이전트

Amazon S3 REST API를 사용하는 프로그램은 애플리케이션 계층 또는 HTTP 계층에서 리디렉션을 처리해야 합니다. 리디렉션을 자동으로 올바르게 처리하기 위해 많은 HTTP 클라이언트 라이브러리 및 사용자 에이전트를 구성할 수 있지만 리디렉션 구현이 잘못되거나 불완전한 경우도 많습니다.

라이브러리에 의존하여 리디렉션 요구 사항을 충족시키기 전에 다음 사례를 테스트하십시오.

- 모든 HTTP 요청 헤더가 권한 부여 및 날짜와 같은 HTTP 표준을 포함하며 리디렉션된 요청(리디렉션을 받은 후의 두 번째 요청)에 올바르게 포함되어 있는지 확인합니다.
- PUT 및 DELETE와 같은 비GET 리디렉션이 올바르게 작동하는지 확인합니다.
- 대형 PUT 요청이 올바르게 리디렉션 뒤에 오는지 확인합니다.
- 100-Continue 응답이 도착하는 데 오랜 시간이 걸리는 경우 PUT 요청이 올바르게 리디렉션 뒤에 오는지 확인합니다.

HTTP 요청 메시지가 GET 또는 HEAD가 아니면 RFC 2616을 엄격히 준수하는 HTTP 사용자 에이전트에서 리디렉션을 따르기 전에 명시적 확인을 요구할 수 있습니다. Amazon S3에서 자동으로 생성된 리디렉션을 따르는 것은 일반적으로 안전합니다. 시스템에서 amazonaws.com 도메인 내에 있는 호스트에 대해서만 리디렉션을 발행하며 리디렉션된 요청의 효과가 원래 요청의 효과와 같기 때문입니다.

리디렉션 및 100-Continue

리디렉션 처리를 간소화하고, 효율성을 개선하고, 리디렉션된 요청 본문을 두 번 보내는 작업에 수반되는 비용을 피하려면 PUT 작업에 대해 100-Continue를 사용하도록 애플리케이션을 구성하십시오. 애

폴리케이션에서 100-Continue를 사용하면 승인을 받을 때까지 요청 본문을 보내지 않습니다. 헤더를 기반으로 메시지가 거부될 경우 메시지의 본문이 전송되지 않습니다. 100-Continue에 대한 자세한 내용은 [RFC 2616 Section 8.2.3](#)을 참조하십시오.

Note

RFC 2616에 따르면 알 수 없는 HTTP 서버에 Expect: Continue를 사용할 때 요청 본문을 보내기 전에 무한정으로 기다리지 않아야 합니다. 이는 일부 HTTP 서버에서 100-Continue가 인식되지 않기 때문입니다. 그러나 Amazon S3에서는 요청에 Expect: Continue가 포함되어 있는지를 인식하며 임시 100-Continue 상태 또는 최종 상태 코드로 응답합니다. 또한 임시 100-Continue 진행 신호를 받은 후에는 리디렉션 오류가 발생하지 않습니다. 이는 아직 요청 본문을 작성하고 있는 동안 리디렉션 응답을 받는 것을 방지하는 데 도움이 됩니다.

리디렉션 예제

이 단원에서는 HTTP 리디렉션 및 100-Continue를 사용한 클라이언트-서버 상호 작용의 예제를 제공합니다.

다음은 `quotes.s3.amazonaws.com` 버킷에 대한 샘플 PUT입니다.

```
PUT /nelson.txt HTTP/1.1
Host: quotes.s3.amazonaws.com
Date: Mon, 15 Oct 2007 22:18:46 +0000

Content-Length: 6
Expect: 100-continue
```

Amazon S3에서 다음을 반환합니다.

```
HTTP/1.1 307 Temporary Redirect
Location: http://quotes.s3-4c25d83b.amazonaws.com/nelson.txt?rk=8d47490b
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Mon, 15 Oct 2007 22:18:46 GMT

Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<Error>
```

```
<Code>TemporaryRedirect</Code>
<Message>Please re-send this request to the
specified temporary endpoint. Continue to use the
original request endpoint for future requests.
</Message>
<Endpoint>quotes.s3-4c25d83b.amazonaws.com</Endpoint>
<Bucket>quotes</Bucket>
</Error>
```

클라이언트에서 리디렉션 응답을 따르고 `quotes.s3-4c25d83b.amazonaws.com` 임시 엔드포인트에 대한 새로운 요청을 발행합니다.

```
PUT /nelson.txt?rk=8d47490b HTTP/1.1
Host: quotes.s3-4c25d83b.amazonaws.com
Date: Mon, 15 Oct 2007 22:18:46 +0000

Content-Length: 6
Expect: 100-continue
```

Amazon S3은 클라이언트가 요청 본문 전송을 진행해야 함을 나타내는 100-continue를 반환합니다.

```
HTTP/1.1 100 Continue
```

클라이언트에서 요청 본문을 보냅니다.

```
ha ha\n
```

Amazon S3에서 최종 응답을 반환합니다.

```
HTTP/1.1 200 OK
Date: Mon, 15 Oct 2007 22:18:48 GMT

ETag: "a2c8d6b872054293afd41061e93bc289"
Content-Length: 0
Server: AmazonS3
```

라우팅 요청

[CreateBucketConfiguration](#)을 포함한 [CreateBucket](#) API를 사용하여 만든 버킷에 요청을 하는 프로그램은 리디렉션을 지원해야 합니다. 또한 DNS TTL을 지키지 않는 일부 클라이언트에서 문제가 발생할 수 있습니다.

이 단원에서는 Amazon S3과 함께 사용할 서비스 또는 애플리케이션을 설계할 때 고려할 라우팅 및 DNS 문제에 대해 설명합니다.

요청 리디렉션 및 REST API

Amazon S3에서는 Domain Name System(DNS)을 사용하여 요청을 처리할 수 있는 시설로 요청을 라우팅합니다. 이 시스템은 효율적으로 작동하지만 일시적인 라우팅 오류가 발생할 수 있습니다. 요청이 잘못된 Amazon S3 위치에 도착하는 경우 Amazon S3에서는 요청자에게 새 엔드포인트로 요청을 다시 전송하도록 지시하는 임시 리디렉션으로 응답합니다. 요청이 잘못 구성된 경우 Amazon S3에서는 영구 리디렉션을 사용하여 요청이 올바르게 수행되는 방법에 대한 지침을 제공합니다.

Important

이 기능을 사용하려면 Amazon S3 리디렉션 응답을 처리할 수 있는 애플리케이션이 있어야 합니다. <CreateBucketConfiguration> 없이 생성된 버킷으로만 작동하는 애플리케이션에 대해서는 유일하게 예외가 적용됩니다. 위치 제약 조건에 대한 자세한 내용은 [Amazon S3 버킷 액세스 및 나열](#)을 참조하세요.

2019년 3월 20일 이후에 시작된 모든 리전에 대해 요청이 잘못된 Amazon S3 위치에 도달하면 Amazon S3는 HTTP 400 Bad Request 오류를 반환합니다.

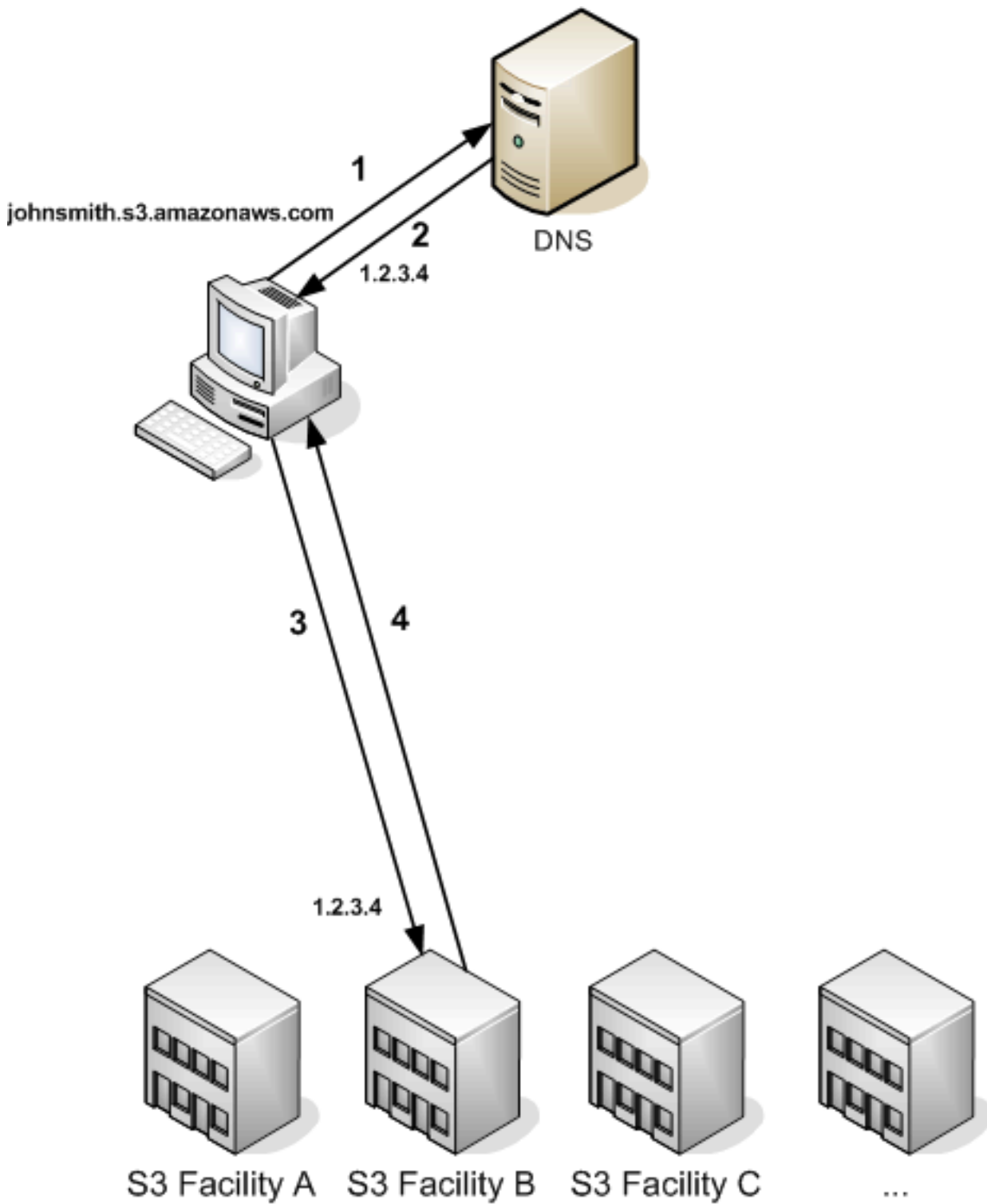
AWS 리전 활성화 및 비활성화에 대한 자세한 내용은 AWS 일반 참조의 [AWS 리전 및 엔드포인트](#)를 참조하세요.

주제

- [DNS 라우팅](#)
- [임시 요청 리디렉션](#)
- [영구 요청 리디렉션](#)
- [요청 리디렉션 예제](#)

DNS 라우팅

DNS 라우팅은 요청을 적절한 Amazon S3 시설로 라우팅합니다. 다음 그림 및 절차에는 DNS 라우팅의 예제가 나와 있습니다.



DNS 라우팅 요청 단계

1. 클라이언트는 Amazon S3에 저장된 객체를 가져오기 위해 DNS 요청을 보냅니다.

2. 클라이언트는 요청을 처리할 수 있는 설비의 IP 주소를 하나 이상 수신합니다. 이 예제에서 IP 주소는 Facility B에 대한 것입니다.
3. 클라이언트는 Amazon S3 Facility B에 요청을 보냅니다.
4. Facility B는 객체의 사본을 클라이언트로 반환합니다.

임시 요청 리디렉션

임시 리디렉션은 요청을 다른 엔드포인트로 다시 전송해야 하는 요청자에게 신호를 보내는 일종의 오류 응답입니다. Amazon S3의 분산 특성 때문에 요청이 일시적으로 잘못된 시설로 라우팅될 수 있습니다. 이러한 경우는 버킷을 만든 직후 또는 삭제한 직후에 발생할 가능성이 가장 높습니다.

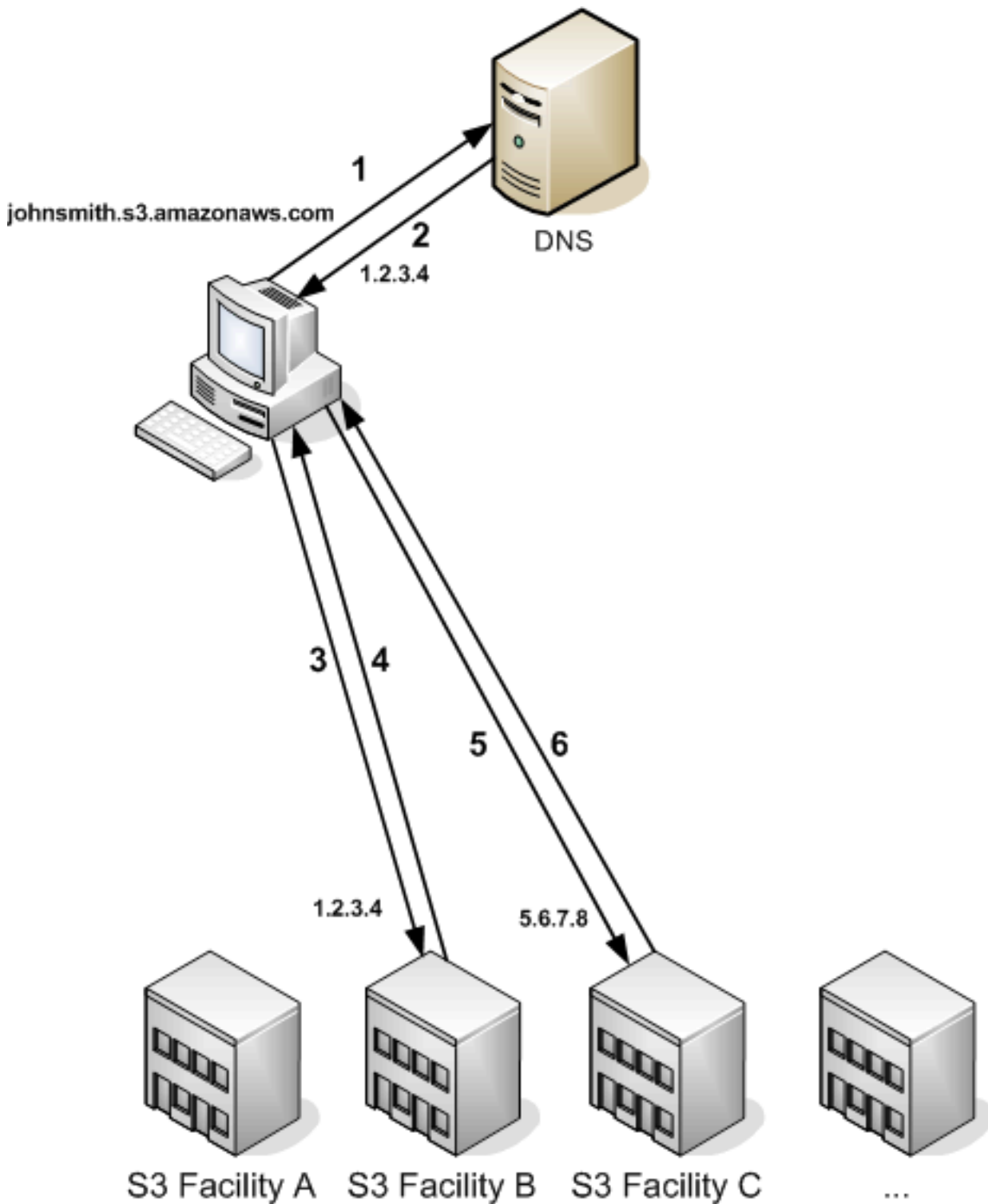
예를 들어, 새 버킷을 만들고 이 버킷에 즉시 요청을 보내는 경우, 버킷의 위치 제한에 따라 임시 리디렉션을 수신할 수 있습니다. 미국 동부(버지니아 북부) AWS 리전에서 버킷을 생성한 경우 이곳 역시 기본 Amazon S3 엔드포인트이므로 리디렉션 현상이 나타나지 않습니다.

그러나 기타 리전에서 버킷을 생성한 경우 버킷의 DNS 항목이 전파되는 동안 이 버킷에 대한 모든 요청은 기본 엔드포인트로 이동합니다. 기본 엔드포인트에서는 HTTP 302 응답과 함께 요청을 정확한 엔드포인트로 리디렉션합니다. 임시 리디렉션에는 정확한 시설에 대한 URI가 포함되어 있어 이를 사용하여 요청을 즉시 재전송할 수 있습니다.

Important

이전 리디렉션 응답에서 제공한 엔드포인트를 재사용하지 마십시오. 긴 시간 동안 작동하는 것처럼 보일 수 있으나 예기치 않은 결과를 제공할 수 있으며 결국 예고 없이 실패하게 됩니다.

다음 그림 및 절차에는 임시 리디렉션의 예제가 나와 있습니다.



임시 요청 리디렉션 단계

1. 클라이언트는 Amazon S3에 저장된 객체를 가져오기 위해 DNS 요청을 보냅니다.
2. 클라이언트는 요청을 처리할 수 있는 설비의 IP 주소를 하나 이상 수신합니다.

3. 클라이언트는 Amazon S3 Facility B에 요청을 보냅니다.
4. Facility B에서는 Location C에서 사용 가능한 객체를 나타내는 리디렉션을 반환합니다.
5. 클라이언트에서 Location C로 요청을 다시 전송합니다.
6. Facility C에서 객체의 사본을 반환합니다.

영구 요청 리디렉션

영구 리디렉션은 요청에 리소스의 주소가 부적절하게 지정되었음을 나타냅니다. 예를 들어, <CreateBucketConfiguration>을 사용하여 생성한 버킷에 경로 스타일 요청을 사용하여 액세스하는 경우 영구 리디렉션이 발생합니다. 자세한 내용은 [Amazon S3 버킷 액세스 및 나열](#)을 참조하세요.

개발 중에 이러한 오류를 쉽게 발견할 수 있도록 요청이 자동으로 올바른 위치를 따라가도록 하는 Location HTTP 헤더를 이 유형의 리디렉션에 포함하지 않습니다. 올바른 Amazon S3 엔드포인트 사용에 대한 도움을 받으려면 결과 XML 오류 문서를 참조하십시오.

요청 리디렉션 예제

다음은 임시 요청 리디렉션 응답의 예제입니다.

REST API 임시 리디렉션 응답

```
HTTP/1.1 307 Temporary Redirect
Location: http://awsexamplebucket1.s3-gztb4pa9sq.amazonaws.com/photos/puppy.jpg?rk=e2c69a31
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Fri, 12 Oct 2007 01:12:56 GMT
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>TemporaryRedirect</Code>
  <Message>Please re-send this request to the specified temporary endpoint. Continue to use the original request endpoint for future requests.</Message>
  <Endpoint>awsexamplebucket1.s3-gztb4pa9sq.amazonaws.com</Endpoint>
</Error>
```

SOAP API 임시 리디렉션 응답

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료 (EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

```
<soapenv:Body>
  <soapenv:Fault>
    <Faultcode>soapenv:Client.TemporaryRedirect</Faultcode>
    <Faultstring>Please re-send this request to the specified temporary endpoint.
    Continue to use the original request endpoint for future requests.</Faultstring>
    <Detail>
      <Bucket>images</Bucket>
      <Endpoint>s3-gztlb4pa9sq.amazonaws.com</Endpoint>
    </Detail>
  </soapenv:Fault>
</soapenv:Body>
```

DNS 고려 사항

Amazon S3의 설계 요구 사항 중 하나는 고가용성입니다. 이 요구 사항을 충족하는 한 가지 방법은 필요에 따라 DNS의 Amazon S3 엔드포인트와 연결된 IP 주소를 업데이트하는 것입니다. 이러한 변경 내용은 수명이 짧은 클라이언트에는 자동으로 반영되지만, 일부 수명이 긴 클라이언트에는 반영되지 않습니다. 수명이 긴 클라이언트는 이러한 변경 사항을 활용하기 위해 주기적으로 Amazon S3 엔드포인트를 재확인하는 특별 작업을 수행해야 합니다. 가상 머신(VM)에 대한 자세한 내용은 다음을 참조하십시오.

- Java의 경우 Sun의 JVM이 기본적으로 DNS 조회를 영구 캐시합니다. 이 동작을 변경하는 방법에 대한 자세한 내용은 [InetAddress 설명서](#)의 "InetAddress Caching" 단원을 참조하십시오.
- PHP의 경우, 가장 인기 있는 배포 구성에서 실행되는 PHP VM이 VM이 재시작될 때까지 DNS 조회를 캐시합니다. [getHostByName PHP 문서](#)를 참조하십시오.

AWS CLI를 사용하여 Amazon S3에서 개발

Amazon S3 AWS CLI 명령은 AWS CLI 명령 참조에서 여러 세트로 구성되어 있으며 각 세트마다 사용할 수 있는 고유한 명령이 있습니다. 한 세트에서 명령을 찾을 수 없으면 다른 세트도 확인해 보세요. 각 세트는 다음과 같습니다.

- `s3` – 객체 및 버킷 작업을 위한 상위 수준 명령을 설명합니다. 여기에는 복사, 나열 및 이동 작업이 포함됩니다. 이 세트의 전체 명령 목록은 AWS CLI 명령 참조의 [s3](#)를 참조하세요.
- `s3api` – 버킷, 객체, 세션, 멀티파트 업로드와 같은 S3 리소스를 관리하는 하위 수준 명령을 설명합니다. 이러한 명령은 [Amazon S3 API](#) 작업에 해당합니다. 이 세트의 전체 CLI 명령 목록은 AWS CLI 명령 참조의 [s3api](#)를 참조하세요.
- `s3control` – Access Grants, Storage Lens 그룹, Amazon S3 on Outposts 버킷과 같은 기타 모든 S3 리소스를 관리하는 하위 수준 명령을 설명합니다. 이러한 명령은 [Amazon S3 Control](#) API 작업에 해당합니다. 이 세트의 전체 CLI 명령 목록은 AWS CLI 명령 참조의 [s3control](#)을 참조하세요.

AWS CLI를 설정하는 데 문제가 있는 경우 AWS re:Post 지식 센터의 [Amazon S3에서 '자격 증명을 찾을 수 없음' 오류 해결 방법](#)을 참조하세요.

Note

Amazon S3와 같은 AWS의 서비스를 사용하려면 액세스할 때 자격 증명을 제공해야 합니다. 이를 통해 서비스는 자신이 소유한 리소스에 액세스할 수 있는 권한이 있는지 확인할 수 있습니다. 콘솔은 암호를 요구합니다. AWS 계정에 대한 액세스 키를 생성하면 AWS CLI 또는 API에 액세스할 수 있습니다. 그러나 AWS 계정용 자격 증명을 사용하여 AWS에 액세스하지 않는 것이 좋습니다. 대신 AWS Identity and Access Management(IAM) 사용을 권장합니다. IAM 사용자를 생성하여 관리자 권한과 함께 IAM 그룹에 추가한 다음, 생성한 IAM 사용자에게 관리자 권한을 부여하십시오. 그러면 해당 IAM 사용자의 특정 URL과 자격 증명을 사용하여 AWS에 액세스할 수 있습니다. 관련 지침은 IAM 사용 설명서의 [첫 번째 IAM 사용자 및 관리자 그룹 생성](#)을 참조하세요.

AWS CLI에 대해 자세히 알아보기

AWS에 대한 자세한 내용은 다음 리소스를 참조하세요.

- [AWS Command Line Interface](#)
- [AWS Command Line Interface 버전 2 사용 설명서](#)

AWS SDK를 사용하여 Amazon S3에서 개발

AWS 다양한 프로그래밍 언어에 대해 소프트웨어 개발 키트(SDK)를 사용할 수 있습니다. 각 SDK는 개발자가 선호하는 언어로 애플리케이션을 쉽게 구축할 수 있도록 하는 API, 코드 예제 및 설명서를 제공합니다.

Note

웹 및 모바일 앱의 엔드-투-엔드 풀스택 개발에 AWS Amplify를 사용할 수 있습니다. Amplify Storage는 파일 스토리지 및 관리 기능을 Amazon S3를 기반으로 구축된 프론트엔드 웹 및 모바일 앱에 원활하게 통합합니다. 자세한 내용은 Amplify 사용 설명서의 [스토리지](#)를 참조하세요.

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	Tools for PowerShell 코드 예시
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

SDK 설명서	코드 예제
AWS SDK for Swift	AWS SDK for Swift 코드 예제

구체적인 예는 [AWS SDK를 사용한 Amazon S3용 코드 예제](#) 섹션을 참조하십시오.

SDK 프로그래밍 인터페이스

각각의 AWS SDK는 Amazon S3 작업을 위한 하나 이상의 프로그래밍 인터페이스를 제공합니다. 각 SDK는 API 작업과 매우 비슷한 메서드를 사용하여 Amazon S3용 하위 수준 인터페이스를 제공합니다. 일부 SDK는 Amazon S3용 상위 수준 인터페이스를 제공합니다. 이러한 인터페이스는 일반 사용 사례를 간소화하기 위해 추상화된 것입니다.

예를 들어 하위 수준 API 작업을 사용하여 멀티파트 업로드를 수행하는 경우, 한 작업을 사용하여 업로드를 시작하고 다른 작업을 사용하여 파트를 업로드하며 최종 작업을 사용하여 업로드를 완료합니다. 상위 수준 멀티파트 업로드 API 작업을 사용하면 업로드에 필요한 모든 작업을 API 직접 호출 한 번으로 수행할 수 있습니다. 예제는 Amazon S3 사용 설명서의 [멀티파트 업로드를 사용하여 객체 업로드](#)를 참조하세요.

하위 수준의 API 작업을 통해 업로드를 보다 효과적으로 제어할 수 있습니다. 업로드를 일시 중지했다가 다시 시작해야 하거나 업로드 중에 파트 크기를 변경해야 하거나 데이터 크기를 사전에 알지 못하고 업로드를 시작하는 경우 하위 수준 API를 사용합니다.

요청 인증에서 서명 버전 지정

대부분의 AWS 리전에서는 Amazon S3가 AWS 서명 버전 4만 지원합니다. 일부 이전 AWS 리전에서는 Amazon S3가 서명 버전 4와 서명 버전 2를 모두 지원합니다. 그러나 서명 버전 2의 사용이 중지되고 있습니다(사용되지 않음). 서명 버전 2 지원 중단에 대한 자세한 내용은 섹션을 참조하세요 [AWS Amazon S3에서 서명 버전 2 사용 중지\(사용되지 않음\)](#)

모든 Amazon S3 리전과 이 리전에서 지원하는 서명 버전의 목록은 AWS 일반 참조의 [리전 및 엔드포인트](#)를 참조하세요.

모든 AWS 리전에서 AWS SDK는 요청 인증에 서명 버전 4를 기본적으로 사용합니다. 2016년 5월 이전에 배포된 AWS SDK를 사용하는 경우에는 다음 표에 있는 서명 버전 4를 요청해야 할 수도 있습니다.

SDK	요청 인증을 위한 서명 버전 4 요청
AWS CLI	기본 프로파일의 경우 다음 명령을 실행합니다: <pre data-bbox="613 344 1380 415">\$ aws configure set default.s3.signature_version s3v4</pre> 사용자 지정 프로파일의 경우 다음 명령을 실행합니다: <pre data-bbox="613 575 1360 646">\$ aws configure set profile.your_profile_name.s3.signature_version s3v4</pre>
Java SDK	코드에 다음을 추가합니다: <pre data-bbox="613 806 1331 877">System.setProperty(SDKGlobalConfiguration.ENABLE_S3_SIGV4_SYSTEM_PROPERTY, "true");</pre> 또는, 명령줄에 다음을 지정합니다: <pre data-bbox="613 1037 1185 1062">-Dcom.amazonaws.services.s3.enableV4</pre>
JavaScript SDK	클라이언트 생성 시에 <code>signatureVersion</code> 파라미터를 v4로 설정합니다: <pre data-bbox="613 1276 1341 1302">var s3 = new AWS.S3({signatureVersion: 'v4'});</pre>
PHP SDK	PHP SDK v2에 대한 Amazon S3 서비스 클라이언트 생성 시 다음과 같이 <code>signature</code> 파라미터를 v4로 설정합니다. <pre data-bbox="613 1514 1088 1743"><?php \$client = S3Client::factory(['region' => 'YOUR-REGION', 'version' => 'latest', 'signature' => 'v4']);</pre>

SDK	요청 인증을 위한 서명 버전 4 요청
	PHP SDK v3를 사용하는 경우 Amazon S3 서비스 클라이언트를 생성하는 동안 <code>signature_version</code> 파라미터를 v4로 설정하십시오. <pre data-bbox="613 380 1507 653"><?php \$s3 = new Aws\S3\S3Client(['version' => '2006-03-01', 'region' => 'YOUR-REGION', 'signature_version' => 'v4']);</pre>
Python-Boto SDK	기본 config 파일인 boto 파일에 다음을 지정합니다: <pre data-bbox="613 768 1507 842">[s3] use-sigv4 = True</pre>
Ruby SDK	Ruby SDK - 버전 1: 클라이언트 생성 시에 <code>:s3_signature_version</code> 파라미터를 v4로 설정합니다: <pre data-bbox="613 1010 1507 1125">s3 = AWS::S3::Client.new(:s3_signature_version => :v4)</pre> Ruby SDK - 버전 3: 클라이언트 생성 시에 <code>signature_version</code> 파라미터를 v4로 설정합니다: <pre data-bbox="613 1283 1507 1356">s3 = Aws::S3::Client.new(signature_version: 'v4')</pre>

SDK	요청 인증을 위한 서명 버전 4 요청
.NET SDK	Amazon S3 클라이언트를 만들기 전에 다음을 코드에 추가합니다. <pre data-bbox="609 367 1263 399">AWSConfigsS3.UseSignatureVersion4 = true;</pre> 또는, config 파일에 다음을 추가합니다: <pre data-bbox="609 556 1425 709"><appSettings> <add key="AWS.S3.UseSignatureVersion4" value="true" /> </appSettings></pre>

AWS Amazon S3에서 서명 버전 2 사용 중지(사용되지 않음)

Amazon S3에서 서명 버전 2의 사용이 중지되고 있습니다(사용되지 않음). 그런 다음 Amazon S3는 서명 버전 4를 사용하여 서명된 API 요청만 수락합니다.

이 단원에서는 서명 버전 2 지원 종료에 대해 자주 묻는 질문에 답변합니다.

서명 버전 2/4란 무엇이며, 요청 서명은 무엇을 의미합니까?

서명 버전 2 또는 서명 버전 4 서명 프로세스는 Amazon S3 API 요청을 인증하는 데 사용됩니다. 요청 서명은 Amazon S3가 요청을 전송하는 사용자를 식별하고 악의적 사용자로부터 요청을 보호할 수 있게 해줍니다.

AWS 요청 서명에 대한 자세한 내용은 AWS 일반 참조의 [AWS API 요청에 서명](#)을 참조하세요.

무엇을 업데이트합니까?

현재 서명 버전 2와 서명 버전 4 프로세스를 사용하여 서명된 Amazon S3 API 요청을 지원합니다. 그 이후로는 Amazon S3가 서명 버전 4를 사용하여 서명된 요청만 수락합니다.

AWS 요청 서명에 대한 자세한 내용은 AWS 일반 참조에서 [서명 버전 4의 변경 사항](#) 섹션을 참조하세요.

업데이트하는 이유는 무엇입니까?

서명 버전 4는 보안 액세스 키 대신 서명 키를 사용하여 개선된 보안을 제공합니다. 서명 버전 4는 현재 모든 AWS 리전에서 지원되는 반면, 서명 버전 2는 2014년 1월 이전에 시작한 리전에서만 지원됩니다. 이번 업데이트를 통해 모든 리전에서 보다 일관된 경험을 제공할 수 있게 되었습니다.

서명 버전 4를 사용 중인지 확인하려면 어떻게 해야 하고 무엇을 업데이트해야 합니까?

요청을 서명하는 데 사용되는 서명 버전은 일반적으로 클라이언트 측에서 도구 또는 SDK에 의해 설정됩니다. 기본적으로 최신 버전의 AWS SDK는 서명 버전 4를 사용합니다. 타사 소프트웨어의 경우, 소프트웨어 담당 지원 팀에 연락하여 필요한 버전을 확인하십시오. Amazon S3로 직접 REST 호출을 전송하는 경우, 서명 버전 4 서명 프로세스를 사용하도록 애플리케이션을 수정해야 합니다.

서명 버전 4로 전환 시 사용할 AWS SDK 버전에 대한 정보는 [서명 버전 2에서 서명 버전 4로 전환](#) 섹션을 참조하세요.

Amazon S3 REST API에서 서명 버전 4를 사용하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 [요청 인증\(AWS 서명 버전 4\)](#) 섹션을 참조하세요.

업데이트하지 않으면 어떻게 됩니까?

그 이후 서명 버전 2로 서명하여 작성된 요청은 Amazon S3에서 인증이 실패합니다. 요청자에게 서명 버전 4를 사용하여 요청에 서명해야 한다는 오류가 표시됩니다.

7일 이상 동안 서명하도록 요구하는 미리 서명된 URL을 사용하는 데도 전환이 필요합니까?

7일 이상 동안 서명하도록 요구하는 미리 서명된 URL을 사용하는 경우 현재 아무 조치도 필요하지 않습니다. 계속해서 AWS 서명 버전 2를 사용하여 미리 서명된 URL을 서명 및 인증할 수 있습니다. 추후 미리 서명된 URL 시나리오에서 서명 버전 4로 전환하는 자세한 방법을 안내하도록 하겠습니다.

추가 정보

- 서명 버전 4 사용에 대한 자세한 내용은 [AWS API 요청에 서명](#) 섹션을 참조하세요.
- 서명 버전 2와 서명 버전 4 간 변경 사항의 목록은 [서명 버전 4의 변경 사항](#)을 참조하십시오.
- AWS 포럼에서 게시물 [Amazon S3 API 요청 서명을 위해 AWS 서명 버전 2를 대체하는 AWS 서명 버전 4](#)를 읽어 보세요.
- 추가 문의 사항은 [지원](#)로 문의하세요.

서명 버전 2에서 서명 버전 4로 전환

현재 Amazon S3 API 요청 인증에 서명 버전 2를 사용 중인 경우 서명 버전 4로 전환해야 합니다. [AWS Amazon S3에서 서명 버전 2 사용 중지\(사용되지 않음\)](#)에서 설명한 대로 서명 버전 2에 대한 지원이 종료될 예정입니다.

Amazon S3 REST API에서 서명 버전 4를 사용하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 [요청 인증\(AWS 서명 버전 4\)](#) 섹션을 참조하세요.

다음 표에는 필수 최소 버전으로 서명 버전 4(SigV4)를 사용하는 SDK가 나열되어 있습니다. AWS Java, JavaScript(Node.js) 또는 Python(Boto/CLI) SDK를 통해 미리 서명된 URL을 사용 중인 경우 올바른 AWS 리전을 설정하고 클라이언트 구성에서 서명 버전 4를 설정해야 합니다. 클라이언트 구성에서 SigV4 설정에 대한 자세한 내용은 [요청 인증에서 서명 버전 지정](#) 단원을 참조하십시오.

현재 사용 중인 SDK/제품	업그레이드할 SDK 버전	Sigv4를 사용하려면 클라이언트에서 코드를 변경해야 하나요?	SDK 설명서 링크
AWS SDK for Java v1	Java 1.11.201+ 또는 v2로 업그레이드합니다.	예	요청 인증에서 서명 버전 지정
AWS SDK for Java v2	SDK 업그레이드가 필요하지 않음.	아니요	AWS SDK for Java
AWS SDK for .NET v1	3.1.10 이상으로 업그레이드.	예	AWS SDK for .NET
AWS SDK for .NET v2	3.1.10 이상으로 업그레이드.	아니요	AWS SDK for .NET v2

현재 사용 중인 SDK/제품	업그레이드할 SDK 버전	Sigv4를 사용하려면 클라이언트에서 코드를 변경해야 합니까?	SDK 설명서 링크
AWS SDK for .NET v3	3.3.0.0 이상으로 업그레이드.	예	AWS SDK for .NET v3
AWS SDK for JavaScript v1	2.68.0 이상으로 업그레이드.	예	AWS SDK for JavaScript
AWS SDK for JavaScript v2	2.68.0 이상으로 업그레이드.	예	AWS SDK for JavaScript
AWS SDK for JavaScript v3	현재 아무 조치도 필요하지 않음. 2019년 Q3에 메이저 버전 V3로 업그레이드.	아니요	AWS SDK for JavaScript
AWS SDK for PHP v1	S3 클라이언트 구성에서 서명 파라미터를 v4로 설정하고 PHP의 최신 버전(v2.7.4 이상)으로 업그레이드할 것을 권장합니다.	예	AWS SDK for PHP

현재 사용 중인 SDK/제품	업그레이드할 SDK 버전	Sigv4를 사용하려면 클라이언트에서 코드를 변경해야 합니까?	SDK 설명서 링크
AWS SDK for PHP v2	S3 클라이언트 구성에서 서명 파라미터를 v4로 설정하고 PHP의 최신 버전(v2.7.4 이상)으로 업그레이드할 것을 권장합니다.	아니요	AWS SDK for PHP
AWS SDK for PHP v3	SDK 업그레이드가 필요하지 않음.	아니요	AWS SDK for PHP
Boto2	Boto2 v2.49.0으로 업그레이드.	예	Boto 2 업그레이드
Boto 3	1.5.71(Botocore), 1.4.6(Boto3)으로 업그레이드.	예	Boto 3 - AWS SDK for Python
AWS CLI	1.11.108로 업그레이드.	예	AWS Command Line Interface
AWS CLI v2(프리뷰)	SDK 업그레이드가 필요하지 않음.	아니요	AWS Command Line Interface 버전 2

현재 사용 중인 SDK/제품	업그레이드할 SDK 버전	Sigv4를 사용하려면 클라이언트에서 코드를 변경해야 합니까?	SDK 설명서 링크
AWS SDK for Ruby v1	Ruby V3로 업그레이드.	예	용 Ruby V3AWS
AWS SDK for Ruby v2	Ruby V3로 업그레이드.	예	용 Ruby V3AWS
AWS SDK for Ruby v3	SDK 업그레이드가 필요하지 않음.	아니요	용 Ruby V3AWS
Go	SDK 업그레이드가 필요하지 않음.	아니요	AWS SDK for Go
C++	SDK 업그레이드가 필요하지 않음.	아니요	AWS SDK for C++

AWS Tools for Windows PowerShell 또는 AWS Tools for PowerShell Core

3.3.0.0 이전의 모듈 버전을 사용하는 경우 3.3.0.0으로 업그레이드해야 합니다.

버전 정보를 확인하려면 Get-Module cmdlet을 사용합니다.

```
Get-Module -Name AWSPowershell
Get-Module -Name AWSPowershell.NetCore
```

3.3.0.0 버전을 업데이트하려면 Update-Module cmdlet을 사용합니다.

```
Update-Module -Name AWSPowershell
```

```
Update-Module -Name AWSPowershell.NetCore
```

서명 버전 2 트래픽을 전송하는 데 7일간 유효한 미리 서명된 URL을 사용할 수 있습니다.

AWS Support에 대한 Amazon S3 요청 ID 가져오기

Amazon S3에서 오류나 예기치 않은 동작이 발생하여 AWS Support에 문의할 때는 실패한 작업과 관련된 요청 ID를 제공해야 합니다. AWS Support이 요청 ID를 사용하여 해당 문제의 해결을 지원합니다.

요청 ID는 쌍으로 되어 있고 Amazon S3에서 처리하는 모든 응답(잘못된 응답이라도)에 반환되며 상세 정보 로그를 통해 액세스할 수 있습니다. 요청 ID를 가져오는 데는 S3 서버 액세스 로그 및 AWS CloudTrail 이벤트 또는 데이터 이벤트 등의 여러 가지 일반적인 방법을 사용할 수 있습니다.

이 로그를 복구한 후에는 에 문의할 때 필요하므로 두 값을 복사하여 보관하세요. AWS Support AWS Support에 문의하는 방법에 대한 자세한 내용은 [AWS 문의](#) 또는 [AWS Support 설명서](#)를 참조하세요.

주제

- [AWS CLI을 이용한 요청 ID 가져오기](#)
- [Windows PowerShell을 사용하여 요청 ID 가져오기](#)
- [AWS CloudTrail 데이터 이벤트를 이용한 요청 ID 가져오기](#)
- [S3 서버 액세스 로깅을 이용한 요청 ID 가져오기](#)
- [HTTP를 이용한 요청 ID 가져오기](#)
- [웹 브라우저를 이용한 요청 ID 가져오기](#)
- [AWS SDK를 이용한 요청 ID 가져오기](#)

AWS CLI을 이용한 요청 ID 가져오기

AWS Command Line Interface(AWS CLI)를 사용할 때 요청 ID를 가져오려면 명령에 `--debug`를 추가하세요. 예를 들어 아래와 같이 디버그 로그에 `x-amz-request-id` 및 `x-amz-id-2`가 표시되어야 합니다.

```
...
2025-04-30 14:35:28,572 - MainThread - botocore.parsers - DEBUG - Response headers:
{'x-amz-id-2': 'a+zm50vDJH3LLmCiXvwEo0u0PtPS/qCJaBvB2ZMH9dzyzTiJhiLZkBFFoRfsPf0KztUKT/garCI=', 'x-amz-request-id': 'N4NMN0MJ4VDFZMX9', 'Date': 'Wed, 30 Apr 2025 21:35:29 GMT', 'Content-Type': 'application/xml', 'Transfer-Encoding': 'chunked', 'Server': 'AmazonS3'}
```


...

Windows PowerShell을 사용하여 요청 ID 가져오기

Windows PowerShell을 사용한 로그 복구에 대한 자세한 내용은 AWS 개발자 도구 블로그에서 [AWS Tools for Windows PowerShell의 응답 로깅](#)을 참조하세요.

AWS CloudTrail 데이터 이벤트를 이용한 요청 ID 가져오기

S3 객체 수준 API 작업을 로깅하도록 CloudTrail 데이터 이벤트를 사용하여 구성된 Amazon S3 버킷은 Amazon S3의 사용자, 역할 또는 AWS 서비스가 수행한 작업의 세부 정보를 제공합니다. Athena를 사용하여 CloudTrail 이벤트를 쿼리함으로써 S3 요청 ID를 식별할 수 있습니다.

자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [CloudTrail을 사용하여 Amazon S3 요청 식별](#)을 참조하세요.

S3 서버 액세스 로깅을 이용한 요청 ID 가져오기

S3 서버 액세스 로깅에 대해 구성된 Amazon S3 버킷은 버킷에 수행된 각 요청에 대한 상세 레코드를 제공합니다. Athena를 사용하여 서버 액세스 로그를 쿼리함으로써 S3 요청 ID를 식별할 수 있습니다.

자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [Amazon Athena를 사용하여 요청에 대한 액세스 로그 쿼리](#)를 참조하세요.

HTTP를 이용한 요청 ID 가져오기

HTTP 요청이 대상 애플리케이션에 도달하기 전에 이 요청을 기록하여 요청 ID x-amz-request-id 및 x-amz-id-2를 가져올 수 있습니다. HTTP 요청의 상세 정보 로그를 복구할 수 있는 다양한 타사 도구가 있습니다. 신뢰할 수 있는 도구를 하나 선택하여 다른 Amazon S3 HTTP 요청을 보낼 때 Amazon S3 트래픽이 통과하는 포트에서 수신 대기할 도구를 실행합니다.

HTTP 요청의 경우 요청 ID 쌍은 다음과 같이 표시됩니다.

```
x-amz-request-id: 79104EXAMPLEB723
x-amz-id-2: IOWQ4fDEXAMPLEQM+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK+Jd1vEXAMPLEa3Km
```

Note

HTTPS 요청은 암호화되어 대부분의 패킷 캡처에서 숨겨집니다.

웹 브라우저를 이용한 요청 ID 가져오기

대부분의 웹 브라우저에는 요청 헤더를 보는 데 사용할 수 있는 개발자 도구가 있습니다.

오류를 반환하는 웹 브라우저 기반 요청의 경우 요청 ID 쌍이 다음 예와 같이 표시됩니다.

```
<Error><Code>AccessDenied</Code><Message>Access Denied</Message>
<RequestId>79104EXAMPLEB723</RequestId><HostId>IOWQ4fDEXAMPLEQM
+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK+Jd1vEXAMPLEa3Km</HostId></Error>
```

성공적인 요청에서 요청 ID 쌍을 가져오려면 브라우저의 개발자 도구를 사용하여 HTTP 응답 헤더를 확인합니다.

AWS SDK를 이용한 요청 ID 가져오기

다음 섹션에는 다양한 AWS SDK를 사용한 로깅 구성에 대한 정보가 있습니다.

Note

모든 요청과 응답에서 상세 정보 로깅을 활성화할 수 있지만 대량 요청 또는 응답은 애플리케이션 속도를 크게 떨어뜨릴 수 있으므로 프로덕션 시스템에서는 로깅을 활성화하지 않는 것이 좋습니다.

AWS SDK 요청의 경우 요청 ID 쌍은 다음과 같이 표시됩니다.

```
Status Code: 403, AWS Service: Amazon S3, AWS Request ID: 79104EXAMPLEB723
AWS Error Code: AccessDenied AWS Error Message: Access Denied
S3 Extended Request ID: IOWQ4fDEXAMPLEQM+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK
+Jd1vEXAMPLEa3Km
```

C++

로거 유형 및 세부 수준은 SDK 초기화 중에 SDKOptions 인수에서 지정됩니다. 다음 예제에서는 세부 수준을 LogLevel::Debug로 지정합니다.

기본 로거는 파일 시스템에 로그를 쓰고 파일은 다음 규칙 aws_sdk_YYYY-MM-DD-HH.log를 사용하여 이름이 지정됩니다. 로거는 매 시간 새 파일을 생성합니다.

```
Aws::SDKOptions options;
options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
```

```
Aws::InitAPI(options);
// ...
Aws::ShutdownAPI(options);
```

자세한 내용은 GitHub의 AWS SDK for C++ wiki에서 [How do I turn on logging?](#)을 참조하세요.

Go

SDK for Go를 사용하여 로깅을 구성할 수 있습니다. 자세한 내용은 AWS SDK for Go v2 개발자 안내서의 [로깅](#)을 참조하세요.

Java

특정 요청 또는 응답에 로깅을 활성화하여 관련 헤더만 포착하고 반환할 수 있습니다. 그러면 `com.amazonaws.services.s3.S3ResponseMetadata` 클래스를 가져오십시오. 그런 다음 실제 요청을 수행하기 전에 변수에 요청을 저장할 수 있습니다. 로깅된 요청이나 응답을 가져오려면 `getCachedResponseMetadata(AmazonWebServiceRequest request).getRequestID()`를 호출합니다.

```
PutObjectRequest req = new PutObjectRequest(bucketName, key, createSampleFile());
s3.putObject(req);
S3ResponseMetadata md = s3.getCachedResponseMetadata(req);
System.out.println("Host ID: " + md.getHostId() + " RequestID: " +
    md.getRequestId());
```

그 대신 모든 Java 요청 및 응답의 상세 정보 로깅을 사용할 수 있습니다. 자세한 내용은 AWS SDK for Java 개발자 안내서의 [상세 유선 로깅](#)을 참조하세요.

JavaScript

AWS SDK for JavaScript에는 기본 제공 로거가 있으므로, 이를 통해 수행하는 API 직접 호출을 로깅할 수 있습니다. 콘솔에서 이 로거를 켜고 로그 항목을 인쇄하려면 코드에 다음 명령문을 추가하세요.

```
AWS.config.logger = console;
```

자세한 내용은 AWS SDK for JavaScript 개발자 안내서의 [AWS SDK for JavaScript 직접 호출 로깅](#)을 참조하세요.

Kotlin

AWS SDK for Kotlin을 사용하면 코드 또는 환경 설정을 사용하여 유선 수준 메시지에 대한 로그 모드를 지정할 수 있습니다. HTTP 요청 및 HTTP 응답에 대한 로그 모드를 설정할 수 있습니다.

추가 로깅을 옵트인하려면 서비스 클라이언트를 구성할 때 `logMode` 속성을 설정합니다.

```
import aws.smithy.kotlin.runtime.client.LogMode

// ...

val client = S3Client {
    // ...
    logMode = LogMode.LogRequestWithBody + LogMode.LogResponse
}
```

또는 환경 변수를 사용하여 로그 모드를 설정할 수 있습니다.

```
export SDK_LOG_MODE=LogRequestWithBody|LogResponse
```

자세한 내용은 AWS SDK for Kotlin 개발자 안내서의 [로깅](#)을 참조하세요.

.NET

기본 제공된 `System.Diagnostics` 로깅 도구를 사용하여 SDK for .NET으로 로깅을 구성할 수 있습니다. 자세한 내용은 [SDK for .NET을 사용하여 로깅](#) AWS 개발자 블로그 게시물을 참조하세요.

Note

반환되는 정보에는 기본적으로 오류 정보만 포함됩니다. 요청 ID를 가져오려면 구성 파일에 `AWSLogMetrics`(및 선택적으로 `AWSResponseLogging`)가 추가되어 있어야 합니다.

PHP

클라이언트 생성자에서 디버그 옵션을 `true`로 설정하여 네트워크를 통해 전송되는 데이터를 비롯한 디버그 정보를 가져올 수 있습니다.

```
$s3Client = new Aws\S3\S3Client([
    'region' => 'us-standard',
    'version' => '2006-03-01',
    'debug'   => true
]);
```

자세한 내용은 AWS SDK for PHP 개발자 안내서의 [네트워크를 통해 전송되는 데이터를 확인하려면 어떻게 해야 하나요?](#)를 참조하세요.

Python (Boto3)

AWS SDK for Python (Boto3)을 사용하여 특정 응답을 로깅할 수 있습니다. 이 기능을 사용하여 관련 헤더만 캡처할 수 있습니다. 다음 코드에서는 응답의 일부를 파일에 로깅하는 방법을 보여줍니다.

```
import logging
import boto3
logging.basicConfig(filename='logfile.txt', level=logging.INFO)
logger = logging.getLogger(__name__)
s3 = boto3.resource('s3')
response = s3.Bucket(bucket_name).Object(object_key).put()
logger.info("HTTPStatusCode: %s", response['ResponseMetadata']['HTTPStatusCode'])
logger.info("RequestId: %s", response['ResponseMetadata']['RequestId'])
logger.info("HostId: %s", response['ResponseMetadata']['HostId'])
logger.info("Date: %s", response['ResponseMetadata']['HTTPHeaders']['date'])
```

예외가 발생할 때 예외를 확인하고 관련 정보를 기록할 수도 있습니다. 자세한 내용은 AWS SDK for Python(Boto) API 참조의 [오류 응답에서 유용한 정보 식별](#)을 참조하세요.

또한 다음 코드를 사용하여 자세한 디버깅 로그를 출력하도록 Boto3를 구성할 수 있습니다.

```
import logging
import boto3
boto3.set_stream_logger('', logging.DEBUG)
```

자세한 내용은 AWS SDK for Python(Boto) API 참조의 [set_stream_logger](#) 섹션을 참조하세요.

Ruby

SDK for Ruby 버전 3을 사용하여 요청 ID를 가져올 수 있습니다. 다음 코드를 사용하여 클라이언트에서 HTTP 유선 로깅을 활성화할 수 있습니다.

```
s3 = Aws::S3::Client.new(http_wire_trace: true)
```

Note

유선 로깅에는 액세스 키 ID와 같은 민감한 정보가 포함될 수 있습니다. 민감한 정보는 AWS Support와 공유하기 전에 제거해야 합니다.

요청 응답 또는 오류에서도 요청 컨텍스트 객체의 요청 ID를 찾을 수 있습니다.

```
# Finding the request ID from an error:
begin
  s3.put_object(bucket: 'bucket', key: 'key', body: 'test')
rescue Aws::S3::Errors::ServiceError => e
  puts e.context[:request_id]
  puts e.context[:s3_host_id]
end

# Finding the request ID from a successful call:
resp = s3.put_object(bucket: 'bucket', key: 'key', body: 'test')
puts resp.context[:request_id]
puts resp.context[:s3_host_id]
```

자세한 내용은 AWS SDK for Ruby 개발자 안내서의 [AWS SDK for Ruby 클라이언트의 유선 추적 정보를 사용하여 디버깅](#)을 참조하세요.

Rust

로깅을 활성화하려면 tracing-subscriber 크레이트를 추가하고 Rust 애플리케이션에서 초기화합니다.

추적 라이브러리를 Cargo.toml 파일에 추가합니다.

```
tracing-subscriber = { version = "0.3", features = ["env-filter"] }
```

그런 다음 SDK 작업을 직접적으로 호출하기 전에 Rust 코드에서 기본 함수의 로거를 초기화합니다.

```
tracing_subscriber::fmt::init();
```

자세한 내용은 AWS SDK for Rust 개발자 안내서의 [AWS SDK for Rust에서 로깅 구성 및 사용](#)을 참조하세요.

S3 Tables에 지원되는 Amazon S3 객체 수준 API 작업

다음 표에는 S3 Tables에 지원되는 S3 객체 수준 API 작업 및 해당 헤더가 포함되어 있습니다. Amazon S3 Tables API의 목록은 [Amazon S3 Tables](#)를 참조하세요. Amazon S3 Tables에 대한 자세한 내용은 Amazon S3 사용 설명서의 [Amazon S3 Tables 및 테이블 버킷 작업](#)을 참조하세요.

객체 수준 API	지원되는 헤더	참고	
AbortMultipartUpload	x-amz-expected-bucket-owner	없음	
CompleteMultipartUpload	x-amz-checksum-crc32	없음	
	x-amz-checksum-crc32c	없음	
	x-amz-checksum-sha1	없음	
	x-amz-checksum-sha256	없음	
	x-amz-expected-bucket-owner	없음	
	If-Match	없음	
	If-None-Match	없음	
CreateMultipartUpload	x-amz-acl: ACL	S3 Tables의 경우 기본값은 bucket-owner-full-control 이며 변경할 수 없습니다.	
	Cache-Control	없음	

객체 수준 API	지원되는 헤더	참고	
	Content-Disposition	없음	
	Content-Encoding	없음	
	Content-Language	없음	
	Content-Type	없음	
	Expires	없음	
	x-amz-checksum-algorithm	없음	
	x-amz-storage-class	S3 Tables의 경우 기본값은 STANDARD이며 변경할 수 없습니다.	
GetObject	x-amz-server-side-encryption	S3 Tables의 경우 기본값은 (SSE-S3) (AES256)이며 변경할 수 없습니다.	
	If-Match	없음	
	If-Modified-Since	없음	
	If-None-Match	없음	
	If-Unmodified-Since	없음	
	Range	없음	
	x-amz-expected-bucket-owner	없음	

객체 수준 API	지원되는 헤더	참고	
	x-amz-checksum-mode	없음	
HeadObject	If-Match	없음	
	If-Modified-Since	없음	
	If-None-Match	없음	
	If-Unmodified-Since	없음	
	Range	없음	
	x-amz-checksum-mode	없음	
	x-amz-expected-bucket-owner	없음	
ListParts	x-amz-expected-bucket-owner	없음	
PutObject	x-amz-acl: ACL	S3 Tables의 경우 기본값은 bucket-owner-full-control 이며 변경할 수 없습니다.	
	Cache-Control	없음	
	Content-Disposition	없음	
	Content-Encoding	없음	
	Content-Language	없음	

객체 수준 API	지원되는 헤더	참고	
	Content-Length	없음	
	Content-MD5	없음	
	Content-Type	없음	
	x-amz-sdk-checksum-algorithm	없음	
	x-amz-checksum-crc32	없음	
	x-amz-checksum-crc32c	없음	
	x-amz-checksum-sha1	없음	
	x-amz-checksum-sha256	없음	
	Expires	없음	
	If-None-Match	없음	
	x-amz-expected-bucket-owner	없음	
	x-amz-storage-class	S3 Tables의 경우 기본값은 STANDARD이며 변경할 수 없습니다.	
	x-amz-server-side-encryption	S3 Tables의 경우 기본값은 (SSE-S3) (AES256)이며 변경할 수 없습니다.	

객체 수준 API	지원되는 헤더	참고	
UploadPart	Content-Length	없음	
	Content-MD5	없음	
	x-amz-checksum-crc32	없음	
	x-amz-checksum-crc32c	없음	
	x-amz-checksum-sha1	없음	
	x-amz-checksum-sha256	없음	
	x-amz-expected-bucket-owner	없음	

AWS SDK를 사용한 Amazon S3용 코드 예제

다음 코드 예제에서는 Amazon S3를 AWS 소프트웨어 개발 키트(SDK)와 함께 사용하는 방법을 보여줍니다.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

코드 예시

- [AWS SDK를 사용한 Amazon S3용 코드 예제](#)
 - [AWS SDK를 사용한 Amazon S3용 기본 예제](#)
 - [Hello Amazon S3](#)
 - [AWS SDK와 함께 사용되는 Amazon S3의 기본 사항 알아보기](#)
 - [AWS SDK를 사용한 Amazon S3에 대한 작업](#)
 - [AWS SDK 또는 CLI와 함께 AbortMultipartUpload 사용](#)
 - [AWS SDK 또는 CLI와 함께 CompleteMultipartUpload 사용](#)
 - [AWS SDK 또는 CLI와 함께 CopyObject 사용](#)
 - [AWS SDK 또는 CLI와 함께 CreateBucket 사용](#)
 - [AWS SDK와 함께 CreateMultiRegionAccessPoint 사용](#)
 - [AWS SDK 또는 CLI와 함께 CreateMultipartUpload 사용](#)
 - [AWS SDK와 함께 CreatePresignedPost 사용](#)
 - [AWS SDK 또는 CLI와 함께 DeleteBucket 사용](#)
 - [CLI로 DeleteBucketAnalyticsConfiguration 사용](#)
 - [AWS SDK 또는 CLI와 함께 DeleteBucketCors 사용](#)
 - [CLI로 DeleteBucketEncryption 사용](#)
 - [CLI로 DeleteBucketInventoryConfiguration 사용](#)
 - [AWS SDK 또는 CLI와 함께 DeleteBucketLifecycle 사용](#)
 - [CLI로 DeleteBucketMetricsConfiguration 사용](#)
 - [AWS SDK 또는 CLI와 함께 DeleteBucketPolicy 사용](#)
 - [CLI로 DeleteBucketReplication 사용](#)
 - [CLI로 DeleteBucketTagging 사용](#)

- [AWS SDK 또는 CLI와 함께 DeleteBucketWebsite 사용](#)
- [AWS SDK 또는 CLI와 함께 DeleteObject 사용](#)
- [CLI로 DeleteObjectTagging 사용](#)
- [AWS SDK 또는 CLI와 함께 DeleteObjects 사용](#)
- [CLI로 DeletePublicAccessBlock 사용](#)
- [CLI로 GetBucketAccelerateConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketAcl 사용](#)
- [CLI로 GetBucketAnalyticsConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketCors 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketEncryption 사용](#)
- [CLI로 GetBucketInventoryConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketLifecycleConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketLocation 사용](#)
- [CLI로 GetBucketLogging 사용](#)
- [CLI로 GetBucketMetricsConfiguration 사용](#)
- [CLI로 GetBucketNotification 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketPolicy 사용](#)
- [CLI로 GetBucketPolicyStatus 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketReplication 사용](#)
- [CLI로 GetBucketRequestPayment 사용](#)
- [CLI로 GetBucketTagging 사용](#)
- [CLI로 GetBucketVersioning 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketWebsite 사용](#)
- [AWS SDK 또는 CLI와 함께 GetObject 사용](#)
- [AWS SDK 또는 CLI와 함께 GetObjectAcl 사용](#)
- [AWS SDK 또는 CLI와 함께 GetObjectAttributes 사용](#)
- [AWS SDK 또는 CLI와 함께 GetObjectLegalHold 사용](#)
- [AWS SDK 또는 CLI와 함께 GetObjectLockConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 GetObjectRetention 사용](#)
- [CLI로 GetObjectTagging 사용](#)

- [CLI로 GetPublicAccessBlock 사용](#)
- [AWS SDK 또는 CLI와 함께 HeadBucket 사용](#)
- [AWS SDK 또는 CLI와 함께 HeadObject 사용](#)
- [CLI로 ListBucketAnalyticsConfigurations 사용](#)
- [CLI로 ListBucketInventoryConfigurations 사용](#)
- [AWS SDK 또는 CLI와 함께 ListBuckets 사용](#)
- [AWS SDK 또는 CLI와 함께 ListMultipartUploads 사용](#)
- [AWS SDK 또는 CLI와 함께 ListObjectVersions 사용](#)
- [CLI로 ListObjects 사용](#)
- [AWS SDK 또는 CLI와 함께 ListObjectsV2 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketAccelerateConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketAcl 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketCors 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketEncryption 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketLifecycleConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketLogging 사용](#)
- [CLI로 PutBucketNotification 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketNotificationConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketPolicy 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketReplication 사용](#)
- [CLI로 PutBucketRequestPayment 사용](#)
- [CLI로 PutBucketTagging 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketVersioning 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketWebsite 사용](#)
- [AWS SDK 또는 CLI와 함께 PutObject 사용](#)
- [AWS SDK 또는 CLI와 함께 PutObjectAcl 사용](#)
- [AWS SDK 또는 CLI와 함께 PutObjectLegalHold 사용](#)
- [AWS SDK 또는 CLI와 함께 PutObjectLockConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 PutObjectRetention 사용](#)
- [AWS SDK 또는 CLI와 함께 RestoreObject 사용](#)

- [AWS SDK 또는 CLI와 함께 SelectObjectContent 사용](#)
- [AWS SDK 또는 CLI와 함께 UploadPart 사용](#)
- [AWS SDK 또는 CLI와 함께 UploadPartCopy 사용](#)
- [AWS SDK를 사용한 Amazon S3에 대한 시나리오](#)
 - [버킷이 존재하는지 확인](#)
 - [객체가 존재하는지 확인](#)
 - [AWS SDK를 사용하여 텍스트를 음성으로, 다시 음성에서 텍스트로 변환](#)
 - [AWS SDK를 사용하여 Amazon S3에 대해 미리 서명된 URL 생성](#)
 - [사용자가 레이블을 사용하여 사진을 관리할 수 있는 사진 자산 관리 애플리케이션 만들기](#)
 - [AWS SDK를 사용하여 Amazon S3 객체를 나열하는 웹 페이지](#)
 - [Amazon Textract 탐색기 애플리케이션 생성](#)
 - [AWS SDK를 사용하여 주어진 Amazon S3 버킷의 모든 객체 삭제](#)
 - [AWS SDK를 사용하여 Amazon S3에 대해 완료되지 않은 멀티파트 업로드 삭제](#)
 - [AWS SDK를 사용하여 Amazon Rekognition을 통해 이미지에서 PPE 감지](#)
 - [AWS SDK를 사용하여 이미지에서 추출한 텍스트의 개체 삭제](#)
 - [AWS SDK를 사용하여 이미지에서 얼굴 감지](#)
 - [AWS SDK를 사용하여 Amazon Rekognition을 통해 이미지에서 객체 감지](#)
 - [Amazon Rekognition을 AWSSDK와 함께 사용하여 동영상의 사람 및 객체 감지](#)
 - [Amazon Simple Storage Service\(Amazon S3\) 버킷에서 S3 '디렉터리' 다운로드](#)
 - [Amazon Simple Storage Service\(S3\) 버킷 내의 모든 객체를 로컬 디렉터리로 다운로드](#)
 - [AWS SDK를 사용하여 Amazon S3 객체에서 알 수 없는 크기의 스트림 다운로드](#)
 - [AWS SDK를 사용하여 다중 리전 액세스 포인트에서 Amazon S3 객체 생성](#)
 - [AWS SDK를 사용하고 If-Modified-Since 헤더를 지정하여 Amazon S3 버킷에서 객체 가져오기](#)
 - [AWS SDK를 사용하여 Amazon S3 객체 암호화 시작하기](#)
 - [AWS SDK를 사용하여 Amazon S3 객체 태깅 시작하기](#)
 - [빅 데이터 처리 클러스터 시작하기](#)
 - [구성 관리 시작하기](#)
 - [문서 텍스트 추출 시작하기](#)
 - [기계 학습 특성 저장소 시작하기](#)

- [객체 스토리지 시작하기](#)
- [쿼리 분석 시작하기](#)
- [AWS SDK를 사용하여 Amazon S3 객체 잠금 기능 작업](#)
- [AWS SDK를 사용하여 Amazon S3 조건부 요청 수행](#)
- [AWS SDK를 사용하여 Amazon S3 버킷의 액세스 제어 목록\(ACL\) 관리](#)
- [AWS SDK를 사용하여 Amazon S3로 대규모 Amazon SQS 메시지 관리](#)
- [AWS SDK를 사용하여 Lambda 함수로 버전이 지정된 Amazon S3 객체를 배치 단위로 관리](#)
- [AWS SDK를 사용하여 Amazon S3 URI 구문 분석](#)
- [AWS SDK를 사용하여 Amazon S3 객체 멀티파트 복사 수행](#)
- [AWS SDK를 사용하여 Amazon S3 이벤트 알림 수신 및 처리](#)
- [AWS SDK를 사용하여 EXIF 및 기타 이미지 정보 저장](#)
- [AWS SDK를 사용하여 Amazon EventBridge에 S3 이벤트 알림 전송](#)
- [AWS SDK를 사용하여 Amazon S3 객체 업로드 또는 다운로드 추적](#)
- [S3 객체 Lambda를 사용하여 애플리케이션의 데이터 변환](#)
- [AWS SDK를 사용한 단위 및 통합 테스트의 예시 접근 방식](#)
- [로컬 디렉터리를 Amazon Simple Storage Service\(S3\) 버킷에 반복 업로드](#)
- [AWS SDK를 사용하여 Amazon S3에 대용량 파일 업로드 또는 Amazon S3에서 대용량 파일 다운로드](#)
- [AWS SDK를 사용하여 Amazon S3 객체에 알 수 없는 크기의 스트림 업로드](#)
- [AWS SDK를 사용하여 Amazon S3 객체 작업 수행](#)
- [AWS SDK를 사용하여 Amazon S3 객체 무결성 기능 작업](#)
- [AWS SDK를 사용하여 버전이 지정된 Amazon S3 객체 작업](#)
- [Amazon S3의 서버리스 예제](#)
 - [Amazon S3 트리거를 사용하여 Lambda 함수 호출](#)
- [AWS SDK를 사용한 Amazon S3 Control 코드 예제](#)
- [AWS SDK를 사용한 Amazon S3 Control 기본 예제](#)
 - [Amazon S3 Control 시작](#)
 - [AWS SDK를 사용하는 Amazon S3 Control의 기본 기능 알아보기](#)
 - [AWS SDK를 사용한 Amazon S3 Control 작업](#)
- [AWS SDK 또는 CLI와 함께 CreateJob 사용](#)

- [AWS SDK와 함께 DeleteJobTagging 사용](#)
- [AWS SDK 또는 CLI와 함께 DescribeJob 사용](#)
- [AWS SDK와 함께 GetJobTagging 사용](#)
- [AWS SDK와 함께 PutJobTagging 사용](#)
- [AWS SDK 또는 CLI와 함께 UpdateJobPriority 사용](#)
- [AWS SDK 또는 CLI와 함께 UpdateJobStatus 사용](#)
- [AWS SDK를 사용한 S3 디렉터리 버킷 코드 예제](#)
- [AWS SDK를 사용한 S3 디렉터리 버킷 기본 예제](#)
 - [Amazon S3 디렉터리 버킷 시작](#)
 - [AWS SDK를 사용하여 S3 디렉터리 버킷의 기본 기능 알아보기](#)
 - [AWS SDK를 사용한 S3 디렉터리 버킷 작업](#)
 - [AWS SDK와 함께 AbortMultipartUpload 사용](#)
 - [AWS SDK와 함께 CompleteMultipartUpload 사용](#)
 - [AWS SDK와 함께 CopyObject 사용](#)
 - [AWS SDK와 함께 CreateBucket 사용](#)
 - [AWS SDK와 함께 CreateMultipartUpload 사용](#)
 - [AWS SDK와 함께 CreateSession 사용](#)
 - [AWS SDK와 함께 DeleteBucket 사용](#)
 - [AWS SDK와 함께 DeleteBucketEncryption 사용](#)
 - [AWS SDK와 함께 DeleteBucketPolicy 사용](#)
 - [AWS SDK와 함께 DeleteObject 사용](#)
 - [AWS SDK와 함께 DeleteObjects 사용](#)
 - [AWS SDK와 함께 GetBucketEncryption 사용](#)
 - [AWS SDK와 함께 GetBucketPolicy 사용](#)
 - [AWS SDK와 함께 GetObject 사용](#)
 - [AWS SDK와 함께 GetObjectAttributes 사용](#)
 - [AWS SDK와 함께 HeadBucket 사용](#)
 - [AWS SDK와 함께 HeadObject 사용](#)
 - [AWS SDK와 함께 ListDirectoryBuckets 사용](#)
 - [AWS SDK와 함께 ListMultipartUploads 사용](#)

- [AWS SDK와 함께 ListObjectsV2 사용](#)
- [AWS SDK와 함께 ListParts 사용](#)
- [AWS SDK와 함께 PutBucketEncryption 사용](#)
- [AWS SDK와 함께 PutBucketPolicy 사용](#)
- [AWS SDK와 함께 PutObject 사용](#)
- [AWS SDK와 함께 UploadPart 사용](#)
- [AWS SDK와 함께 UploadPartCopy 사용](#)
- [AWS SDK를 사용한 S3 디렉터리 버킷 시나리오](#)
 - [AWS SDK를 사용하여 객체를 가져오기 위해 Amazon S3 디렉터리 버킷에 대해 미리 서명된 URL 생성](#)

AWS SDK를 사용한 Amazon S3용 코드 예제

다음 코드 예제에서는 Amazon S3를 AWS 소프트웨어 개발 키트(SDK)와 함께 사용하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접 호출하여 특정 작업을 수행하는 방법을 보여주는 코드 예제입니다.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon S3

- [AWS SDK를 사용한 Amazon S3용 기본 예제](#)
 - [Hello Amazon S3](#)
 - [AWS SDK와 함께 사용되는 Amazon S3의 기본 사항 알아보기](#)
 - [AWS SDK를 사용한 Amazon S3에 대한 작업](#)
 - [AWS SDK 또는 CLI와 함께 AbortMultipartUpload 사용](#)
 - [AWS SDK 또는 CLI와 함께 CompleteMultipartUpload 사용](#)

- [AWS SDK 또는 CLI와 함께 CopyObject 사용](#)
- [AWS SDK 또는 CLI와 함께 CreateBucket 사용](#)
- [AWS SDK와 함께 CreateMultiRegionAccessPoint 사용](#)
- [AWS SDK 또는 CLI와 함께 CreateMultipartUpload 사용](#)
- [AWS SDK와 함께 CreatePresignedPost 사용](#)
- [AWS SDK 또는 CLI와 함께 DeleteBucket 사용](#)
- [CLI로 DeleteBucketAnalyticsConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 DeleteBucketCors 사용](#)
- [CLI로 DeleteBucketEncryption 사용](#)
- [CLI로 DeleteBucketInventoryConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 DeleteBucketLifecycle 사용](#)
- [CLI로 DeleteBucketMetricsConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 DeleteBucketPolicy 사용](#)
- [CLI로 DeleteBucketReplication 사용](#)
- [CLI로 DeleteBucketTagging 사용](#)
- [AWS SDK 또는 CLI와 함께 DeleteBucketWebsite 사용](#)
- [AWS SDK 또는 CLI와 함께 DeleteObject 사용](#)
- [CLI로 DeleteObjectTagging 사용](#)
- [AWS SDK 또는 CLI와 함께 DeleteObjects 사용](#)
- [CLI로 DeletePublicAccessBlock 사용](#)
- [CLI로 GetBucketAccelerateConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketAcl 사용](#)
- [CLI로 GetBucketAnalyticsConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketCors 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketEncryption 사용](#)
- [CLI로 GetBucketInventoryConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketLifecycleConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketLocation 사용](#)
- [CLI로 GetBucketLogging 사용](#)
- [CLI로 GetBucketMetricsConfiguration 사용](#)

- [CLI로 GetBucketNotification 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketPolicy 사용](#)
- [CLI로 GetBucketPolicyStatus 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketReplication 사용](#)
- [CLI로 GetBucketRequestPayment 사용](#)
- [CLI로 GetBucketTagging 사용](#)
- [CLI로 GetBucketVersioning 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketWebsite 사용](#)
- [AWS SDK 또는 CLI와 함께 GetObject 사용](#)
- [AWS SDK 또는 CLI와 함께 GetObjectAcl 사용](#)
- [AWS SDK 또는 CLI와 함께 GetObjectAttributes 사용](#)
- [AWS SDK 또는 CLI와 함께 GetObjectLegalHold 사용](#)
- [AWS SDK 또는 CLI와 함께 GetObjectLockConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 GetObjectRetention 사용](#)
- [CLI로 GetObjectTagging 사용](#)
- [CLI로 GetPublicAccessBlock 사용](#)
- [AWS SDK 또는 CLI와 함께 HeadBucket 사용](#)
- [AWS SDK 또는 CLI와 함께 HeadObject 사용](#)
- [CLI로 ListBucketAnalyticsConfigurations 사용](#)
- [CLI로 ListBucketInventoryConfigurations 사용](#)
- [AWS SDK 또는 CLI와 함께 ListBuckets 사용](#)
- [AWS SDK 또는 CLI와 함께 ListMultipartUploads 사용](#)
- [AWS SDK 또는 CLI와 함께 ListObjectVersions 사용](#)
- [CLI로 ListObjects 사용](#)
- [AWS SDK 또는 CLI와 함께 ListObjectsV2 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketAccelerateConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketAcl 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketCors 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketEncryption 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketLifecycleConfiguration 사용](#)

- [AWS SDK 또는 CLI와 함께 PutBucketLogging 사용](#)
- [CLI로 PutBucketNotification 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketNotificationConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketPolicy 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketReplication 사용](#)
- [CLI로 PutBucketRequestPayment 사용](#)
- [CLI로 PutBucketTagging 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketVersioning 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketWebsite 사용](#)
- [AWS SDK 또는 CLI와 함께 PutObject 사용](#)
- [AWS SDK 또는 CLI와 함께 PutObjectAcl 사용](#)
- [AWS SDK 또는 CLI와 함께 PutObjectLegalHold 사용](#)
- [AWS SDK 또는 CLI와 함께 PutObjectLockConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 PutObjectRetention 사용](#)
- [AWS SDK 또는 CLI와 함께 RestoreObject 사용](#)
- [AWS SDK 또는 CLI와 함께 SelectObjectContent 사용](#)
- [AWS SDK 또는 CLI와 함께 UploadPart 사용](#)
- [AWS SDK 또는 CLI와 함께 UploadPartCopy 사용](#)
- [AWS SDK를 사용한 Amazon S3에 대한 시나리오](#)
 - [버킷이 존재하는지 확인](#)
 - [객체가 존재하는지 확인](#)
 - [AWS SDK를 사용하여 텍스트를 음성으로, 다시 음성에서 텍스트로 변환](#)
 - [AWS SDK를 사용하여 Amazon S3에 대해 미리 서명된 URL 생성](#)
 - [사용자가 레이블을 사용하여 사진을 관리할 수 있는 사진 자산 관리 애플리케이션 만들기](#)
 - [AWS SDK를 사용하여 Amazon S3 객체를 나열하는 웹 페이지](#)
 - [Amazon Textract 탐색기 애플리케이션 생성](#)
 - [AWS SDK를 사용하여 주어진 Amazon S3 버킷의 모든 객체 삭제](#)
 - [AWS SDK를 사용하여 Amazon S3에 대해 완료되지 않은 멀티파트 업로드 삭제](#)
 - [AWS SDK를 사용하여 Amazon Rekognition을 통해 이미지에서 PPE 감지](#)
 - [AWS SDK를 사용하여 이미지에서 추출한 텍스트의 개체 삭제](#)

- [AWS SDK를 사용하여 이미지에서 얼굴 감지](#)
- [AWS SDK를 사용하여 Amazon Rekognition을 통해 이미지에서 객체 감지](#)
- [Amazon Rekognition을 AWSSDK와 함께 사용하여 동영상의 사람 및 객체 감지](#)
- [Amazon Simple Storage Service\(Amazon S3\) 버킷에서 S3 '디렉터리' 다운로드](#)
- [Amazon Simple Storage Service\(S3\) 버킷 내의 모든 객체를 로컬 디렉터리로 다운로드](#)
- [AWS SDK를 사용하여 Amazon S3 객체에서 알 수 없는 크기의 스트림 다운로드](#)
- [AWS SDK를 사용하여 다중 리전 액세스 포인트에서 Amazon S3 객체 생성](#)
- [AWS SDK를 사용하고 If-Modified-Since 헤더를 지정하여 Amazon S3 버킷에서 객체 가져오기](#)
- [AWS SDK를 사용하여 Amazon S3 객체 암호화 시작하기](#)
- [AWS SDK를 사용하여 Amazon S3 객체 태깅 시작하기](#)
- [빅 데이터 처리 클러스터 시작하기](#)
- [구성 관리 시작하기](#)
- [문서 텍스트 추출 시작하기](#)
- [기계 학습 특성 저장소 시작하기](#)
- [객체 스토리지 시작하기](#)
- [쿼리 분석 시작하기](#)
- [AWS SDK를 사용하여 Amazon S3 객체 잠금 기능 작업](#)
- [AWS SDK를 사용하여 Amazon S3 조건부 요청 수행](#)
- [AWS SDK를 사용하여 Amazon S3 버킷의 액세스 제어 목록\(ACL\) 관리](#)
- [AWS SDK를 사용하여 Amazon S3로 대규모 Amazon SQS 메시지 관리](#)
- [AWS SDK를 사용하여 Lambda 함수로 버전이 지정된 Amazon S3 객체를 배치 단위로 관리](#)
- [AWS SDK를 사용하여 Amazon S3 URI 구문 분석](#)
- [AWS SDK를 사용하여 Amazon S3 객체 멀티파트 복사 수행](#)
- [AWS SDK를 사용하여 Amazon S3 이벤트 알림 수신 및 처리](#)
- [AWS SDK를 사용하여 EXIF 및 기타 이미지 정보 저장](#)
- [AWS SDK를 사용하여 Amazon EventBridge에 S3 이벤트 알림 전송](#)
- [AWS SDK를 사용하여 Amazon S3 객체 업로드 또는 다운로드 추적](#)
- [S3 객체 Lambda를 사용하여 애플리케이션의 데이터 변환](#)
- [AWS SDK를 사용한 단위 및 통합 테스트의 예시 접근 방식](#)
- [로컬 디렉터리를 Amazon Simple Storage Service\(S3\) 버킷에 반복 업로드](#)

- [AWS SDK를 사용하여 Amazon S3에 대용량 파일 업로드 또는 Amazon S3에서 대용량 파일 다운로드](#)
- [AWS SDK를 사용하여 Amazon S3 객체에 알 수 없는 크기의 스트림 업로드](#)
- [AWS SDK를 사용하여 Amazon S3 객체 작업 수행](#)
- [AWS SDK를 사용하여 Amazon S3 객체 무결성 기능 작업](#)
- [AWS SDK를 사용하여 버전이 지정된 Amazon S3 객체 작업](#)
- [Amazon S3의 서버리스 예제](#)
 - [Amazon S3 트리거를 사용하여 Lambda 함수 호출](#)

AWS SDK를 사용한 Amazon S3용 기본 예제

다음 코드 예제에서는 AWS SDK와 함께 Amazon Simple Storage Service의 기본 사항을 사용하는 방법을 보여줍니다.

예제

- [Hello Amazon S3](#)
- [AWS SDK와 함께 사용되는 Amazon S3의 기본 사항 알아보기](#)
- [AWS SDK를 사용한 Amazon S3에 대한 작업](#)
 - [AWS SDK 또는 CLI와 함께 AbortMultipartUpload 사용](#)
 - [AWS SDK 또는 CLI와 함께 CompleteMultipartUpload 사용](#)
 - [AWS SDK 또는 CLI와 함께 CopyObject 사용](#)
 - [AWS SDK 또는 CLI와 함께 CreateBucket 사용](#)
 - [AWS SDK와 함께 CreateMultiRegionAccessPoint 사용](#)
 - [AWS SDK 또는 CLI와 함께 CreateMultipartUpload 사용](#)
 - [AWS SDK와 함께 CreatePresignedPost 사용](#)
 - [AWS SDK 또는 CLI와 함께 DeleteBucket 사용](#)
 - [CLI로 DeleteBucketAnalyticsConfiguration 사용](#)
 - [AWS SDK 또는 CLI와 함께 DeleteBucketCors 사용](#)
 - [CLI로 DeleteBucketEncryption 사용](#)
 - [CLI로 DeleteBucketInventoryConfiguration 사용](#)
 - [AWS SDK 또는 CLI와 함께 DeleteBucketLifecycle 사용](#)
 - [CLI로 DeleteBucketMetricsConfiguration 사용](#)

- [AWS SDK 또는 CLI와 함께 DeleteBucketPolicy 사용](#)
- [CLI로 DeleteBucketReplication 사용](#)
- [CLI로 DeleteBucketTagging 사용](#)
- [AWS SDK 또는 CLI와 함께 DeleteBucketWebsite 사용](#)
- [AWS SDK 또는 CLI와 함께 DeleteObject 사용](#)
- [CLI로 DeleteObjectTagging 사용](#)
- [AWS SDK 또는 CLI와 함께 DeleteObjects 사용](#)
- [CLI로 DeletePublicAccessBlock 사용](#)
- [CLI로 GetBucketAccelerateConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketAcl 사용](#)
- [CLI로 GetBucketAnalyticsConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketCors 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketEncryption 사용](#)
- [CLI로 GetBucketInventoryConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketLifecycleConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketLocation 사용](#)
- [CLI로 GetBucketLogging 사용](#)
- [CLI로 GetBucketMetricsConfiguration 사용](#)
- [CLI로 GetBucketNotification 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketPolicy 사용](#)
- [CLI로 GetBucketPolicyStatus 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketReplication 사용](#)
- [CLI로 GetBucketRequestPayment 사용](#)
- [CLI로 GetBucketTagging 사용](#)
- [CLI로 GetBucketVersioning 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketWebsite 사용](#)
- [AWS SDK 또는 CLI와 함께 GetObject 사용](#)
- [AWS SDK 또는 CLI와 함께 GetObjectAcl 사용](#)
- [AWS SDK 또는 CLI와 함께 GetObjectAttributes 사용](#)
- [AWS SDK 또는 CLI와 함께 GetObjectLegalHold 사용](#)

- [AWS SDK 또는 CLI와 함께 GetObjectLockConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 GetObjectRetention 사용](#)
- [CLI로 GetObjectTagging 사용](#)
- [CLI로 GetPublicAccessBlock 사용](#)
- [AWS SDK 또는 CLI와 함께 HeadBucket 사용](#)
- [AWS SDK 또는 CLI와 함께 HeadObject 사용](#)
- [CLI로 ListBucketAnalyticsConfigurations 사용](#)
- [CLI로 ListBucketInventoryConfigurations 사용](#)
- [AWS SDK 또는 CLI와 함께 ListBuckets 사용](#)
- [AWS SDK 또는 CLI와 함께 ListMultipartUploads 사용](#)
- [AWS SDK 또는 CLI와 함께 ListObjectVersions 사용](#)
- [CLI로 ListObjects 사용](#)
- [AWS SDK 또는 CLI와 함께 ListObjectsV2 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketAccelerateConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketAcl 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketCors 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketEncryption 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketLifecycleConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketLogging 사용](#)
- [CLI로 PutBucketNotification 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketNotificationConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketPolicy 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketReplication 사용](#)
- [CLI로 PutBucketRequestPayment 사용](#)
- [CLI로 PutBucketTagging 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketVersioning 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketWebsite 사용](#)
- [AWS SDK 또는 CLI와 함께 PutObject 사용](#)
- [AWS SDK 또는 CLI와 함께 PutObjectAcl 사용](#)
- [AWS SDK 또는 CLI와 함께 PutObjectLegalHold 사용](#)

- [AWS SDK 또는 CLI와 함께 PutObjectLockConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 PutObjectRetention 사용](#)
- [AWS SDK 또는 CLI와 함께 RestoreObject 사용](#)
- [AWS SDK 또는 CLI와 함께 SelectObjectContent 사용](#)
- [AWS SDK 또는 CLI와 함께 UploadPart 사용](#)
- [AWS SDK 또는 CLI와 함께 UploadPartCopy 사용](#)

Hello Amazon S3

다음 코드 예제에서는 Amazon S3를 사용하여 시작하는 방법을 보여줍니다.

.NET

SDK for .NET(v4)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Hello Amazon Simple Storage Service
/// (Amazon S3) example.
/// </summary>
public class HelloS3
{
    /// <summary>
    /// Main method to run the Hello S3 example.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        var s3Client = new AmazonS3Client();

        try
        {
            Console.WriteLine("Hello Amazon S3! Let's list your buckets:");
```

```
        Console.WriteLine(new string('-', 80));

        // Use the built-in paginator to list buckets
        var request = new ListBucketsRequest();
        var paginator = s3Client.Paginators.ListBuckets(request);

        var buckets = new List<S3Bucket>();

        await foreach (var response in paginator.Responses)
        {
            buckets.AddRange(response.Buckets);
        }

        if (buckets.Any())
        {
            Console.WriteLine($"Found {buckets.Count} S3 buckets:");
            Console.WriteLine();

            foreach (var bucket in buckets)
            {
                Console.WriteLine($"- Bucket Name: {bucket.BucketName}");
                Console.WriteLine($"  Creation Date:
{bucket.CreationDate:yyyy-MM-dd HH:mm:ss UTC}");
                Console.WriteLine();
            }
        }
        else
        {
            Console.WriteLine("No S3 buckets found in your account.");
        }

        Console.WriteLine("Hello S3 completed successfully.");
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"S3 service error occurred: {ex.Message}");
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Couldn't list S3 buckets. Here's why:
{ex.Message}");
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 [ListBuckets](#)를 참조하십시오.

C++

SDK for C++

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS s3)

# Set this project's name.
project("hello_s3")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
    libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
        "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})
```

```

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
    for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
    may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
    ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
    hello_s3.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})

```

hello_s3.cpp 소스 파일의 코드입니다.

```

#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <iostream>
#include <aws/core/auth/AWSCredentialsProviderChain.h>
using namespace Aws;
using namespace Aws::Auth;

/*
 * A "Hello S3" starter application which initializes an Amazon Simple Storage
 * Service (Amazon S3) client
 * and lists the Amazon S3 buckets in the selected region.
 *
 * main function
 *
 * Usage: 'hello_s3'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;

```

```

    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        // You don't normally have to test that you are authenticated. But the
        // S3 service permits anonymous requests, thus the s3Client will return "success"
        // and 0 buckets even if you are unauthenticated, which can be confusing to a new
        // user.

        auto provider =
        Aws::MakeShared<DefaultAWSCredentialsProviderChain>("alloc-tag");
        auto creds = provider->GetAWSCredentials();
        if (creds.IsEmpty()) {
            std::cerr << "Failed authentication" << std::endl;
        }

        Aws::S3::S3Client s3Client(clientConfig);
        auto outcome = s3Client.ListBuckets();

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed with error: " << outcome.GetError() <<
std::endl;
            result = 1;
        } else {
            std::cout << "Found " << outcome.GetResult().GetBuckets().size()
                << " buckets\n";
            for (auto &bucket: outcome.GetResult().GetBuckets()) {
                std::cout << bucket.GetName() << std::endl;
            }
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListBuckets](#)를 참조하십시오.

Go

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
package main

import (
    "context"
    "errors"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/smithy-go"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    s3Client := s3.NewFromConfig(sdkConfig)
    count := 10
    fmt.Printf("Let's list up to %v buckets for your account.\n", count)
    result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
    if err != nil {
        var ae smithy.APIError
```

```

if errors.As(err, &ae) && ae.ErrorCode() == "AccessDenied" {
    fmt.Println("You don't have permission to list buckets for this account.")
} else {
    fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
}
return
}
if len(result.Buckets) == 0 {
    fmt.Println("You don't have any buckets!")
} else {
    if count > len(result.Buckets) {
        count = len(result.Buckets)
    }
    for _, bucket := range result.Buckets[:count] {
        fmt.Printf("\t%v\n", *bucket.Name)
    }
}
}
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListBuckets](#)를 참조하세요.

Java

SDK for Java 2.x

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배우보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
```



```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
* <p>
* For more information, see the following documentation topic:
* <p>
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/
public class HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBuckets(s3);
    }

    /**
     * Lists all the S3 buckets associated with the provided AWS S3 client.
     *
     * @param s3 the S3Client instance used to interact with the AWS S3 service
     */
    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
            });

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [ListBuckets](#)를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import {
  paginateListBuckets,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * List the S3 buckets in your configured AWS account.
 */
export const helloS3 = async () => {
  // When no region or credentials are provided, the SDK will use the
  // region and credentials from the local AWS config.
  const client = new S3Client({});

  try {
    /**
     * @type { import("@aws-sdk/client-s3").Bucket[] }
     */
    const buckets = [];

    for await (const page of paginateListBuckets({ client }, {})) {
      buckets.push(...page.Buckets);
    }
    console.log("Buckets: ");
    console.log(buckets.map((bucket) => bucket.Name).join("\n"));
    return buckets;
  } catch (caught) {
    // ListBuckets does not throw any modeled errors. Any error caught
    // here will be something generic like `AccessDenied`.
    if (caught instanceof S3ServiceException) {
      console.error(`${caught.name}: ${caught.message}`);
    } else {
```

```
// Something besides S3 failed.  
throw caught;  
}  
}  
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 [ListBuckets](#)를 참조하세요.

PHP

SDK for PHP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
use Aws\S3\S3Client;  
  
$client = new S3Client(['region' => 'us-west-2']);  
$results = $client->listBuckets();  
var_dump($results);
```

- API 세부 정보는 AWS SDK for PHP API 참조의 [ListBuckets](#)를 참조하십시오.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import boto3
```

```
def hello_s3():
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Simple Storage Service
    (Amazon S3) client and list the buckets in your account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """

    # Create an S3 client.
    s3_client = boto3.client("s3")

    print("Hello, Amazon S3! Let's list your buckets:")

    # Create a paginator for the list_buckets operation.
    paginator = s3_client.get_paginator("list_buckets")

    # Use the paginator to get a list of all buckets.
    response_iterator = paginator.paginate(
        PaginationConfig={
            "PageSize": 50, # Adjust PageSize as needed.
            "StartingToken": None,
        }
    )

    # Iterate through the pages of the response.
    buckets_found = False
    for page in response_iterator:
        if "Buckets" in page and page["Buckets"]:
            buckets_found = True
            for bucket in page["Buckets"]:
                print(f"\t{bucket['Name']}")

    if not buckets_found:
        print("No buckets found!")

if __name__ == "__main__":
    hello_s3()
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [ListBuckets](#)를 참조하십시오.

Ruby

SDK for Ruby

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# frozen_string_literal: true

# S3Manager is a class responsible for managing S3 operations
# such as listing all S3 buckets in the current AWS account.
class S3Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all S3 buckets in the current AWS account.
  def list_buckets
    @logger.info('Here are the buckets in your account:')

    response = @client.list_buckets

    if response.buckets.empty?
      @logger.info("You don't have any S3 buckets yet.")
    else
      response.buckets.each do |bucket|
        @logger.info("- #{bucket.name}")
      end
    end

    rescue Aws::Errors::ServiceError => e
      @logger.error("Encountered an error while listing buckets: #{e.message}")
    end
  end

  if $PROGRAM_NAME == __FILE__
    s3_client = Aws::S3::Client.new
    manager = S3Manager.new(s3_client)
```

```
manager.list_buckets
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 [ListBuckets](#)를 참조하십시오.

Rust

SDK for Rust

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// S3 Hello World Example using the AWS SDK for Rust.
///
/// This example lists the objects in a bucket, uploads an object to that bucket,
/// and then retrieves the object and prints some S3 information about the
/// object.
/// This shows a number of S3 features, including how to use built-in paginators
/// for large data sets.
///
/// # Arguments
///
/// * `client` - an S3 client configured appropriately for the environment.
/// * `bucket` - the bucket name that the object will be uploaded to. Must be
///   present in the region the `client` is configured to use.
/// * `filename` - a reference to a path that will be read and uploaded to S3.
/// * `key` - the string key that the object will be uploaded as inside the
///   bucket.
async fn list_bucket_and_upload_object(
    client: &aws_sdk_s3::Client,
    bucket: &str,
    filepath: &Path,
    key: &str,
) -> Result<(), S3ExampleError> {
    // List the buckets in this account
    let mut objects = client
        .list_objects_v2()
```

```

        .bucket(bucket)
        .into_paginator()
        .send();

println!("key\tetag\tlast_modified\tstorage_class");
while let Some(Ok(object)) = objects.next().await {
    for item in object.contents() {
        println!(
            "{}\t{}\t{}\t{}",
            item.key().unwrap_or_default(),
            item.e_tag().unwrap_or_default(),
            item.last_modified()
                .map(|lm| format!("{lm}"))
                .unwrap_or_default(),
            item.storage_class()
                .map(|sc| format!("{sc}"))
                .unwrap_or_default()
        );
    }
}

// Prepare a ByteStream around the file, and upload the object using that
// ByteStream.
let body = aws_sdk_s3::primitives::ByteStream::from_path(filepath)
    .await
    .map_err(|err| {
        S3ExampleError::new(format!(
            "Failed to create bytestream for {filepath:?} ({err:?})"
        ))
    })?;
let resp = client
    .put_object()
    .bucket(bucket)
    .key(key)
    .body(body)
    .send()
    .await?;

println!(
    "Upload success. Version: {:?}",
    resp.version_id()
        .expect("S3 Object upload missing version ID")
);

```

```
// Retrieve the just-uploaded object.
let resp = client.get_object().bucket(bucket).key(key).send().await?;
println!("etag: {}", resp.e_tag().unwrap_or("(missing)"));
println!("version: {}", resp.version_id().unwrap_or("(missing)"));

Ok(())
}
```

S3ExampleError 유틸리티입니다.

```
/// S3ExampleError provides a From<T: ProvideErrorMetadata> impl to extract
/// client-specific error details. This serves as a consistent backup to handling
/// specific service errors, depending on what is needed by the scenario.
/// It is used throughout the code examples for the AWS SDK for Rust.
#[derive(Debug)]
pub struct S3ExampleError(String);
impl S3ExampleError {
    pub fn new(value: impl Into<String>) -> Self {
        S3ExampleError(value.into())
    }

    pub fn add_message(self, message: impl Into<String>) -> Self {
        S3ExampleError(format!("{}", message.into(), self.0))
    }
}

impl<T: aws_sdk_s3::error::ProvideErrorMetadata> From<T> for S3ExampleError {
    fn from(value: T) -> Self {
        S3ExampleError(format!(
            "{}: {}",
            value
                .code()
                .map(String::from)
                .unwrap_or("unknown code".into()),
            value
                .message()
                .map(String::from)
                .unwrap_or("missing reason".into()),
        ))
    }
}
```



```
impl std::error::Error for S3ExampleError {}

impl std::fmt::Display for S3ExampleError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 [ListBuckets](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 사용되는 Amazon S3의 기본 사항 알아보기

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 버킷을 만들고 버킷에 파일을 업로드합니다.
- 버킷에서 객체를 다운로드합니다.
- 버킷의 하위 폴더에 객체를 복사합니다.
- 버킷의 객체를 나열합니다.
- 버킷 객체와 버킷을 삭제합니다.

.NET

SDK for .NET(v4)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 기능을 보여주는 대화형 시나리오를 실행합니다.

```
public class S3_Basics
{
```

```
public static bool IsInteractive = true;
public static string BucketName = null!;
public static string TempFilePath = null!;
public static S3Wrapper _s3Wrapper = null!;
public static ILogger<S3_Basics> _logger = null!;

public static async Task Main(string[] args)
{
    // Set up dependency injection for the Amazon service.
    using var host = Host.CreateDefaultBuilder(args)
        .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonS3>()
                .AddTransient<S3Wrapper>()
                .AddLogging(builder => builder.AddConsole()))
        .Build();

    _logger = LoggerFactory.Create(builder => builder.AddConsole())
        .CreateLogger<S3_Basics>();

    _s3Wrapper = host.Services.GetRequiredService<S3Wrapper>();

    var sepBar = new string('-', 45);

    Console.WriteLine(sepBar);
    Console.WriteLine("Amazon Simple Storage Service (Amazon S3) basic");
    Console.WriteLine("procedures. This application will:");
    Console.WriteLine("\n\t1. Create a bucket");
    Console.WriteLine("\n\t2. Upload an object to the new bucket");
    Console.WriteLine("\n\t3. Copy the uploaded object to a folder in the
bucket");
    Console.WriteLine("\n\t4. List the items in the new bucket");
    Console.WriteLine("\n\t5. Delete all the items in the bucket");
    Console.WriteLine("\n\t6. Delete the bucket");
    Console.WriteLine(sepBar);

    await RunScenario(_s3Wrapper, _logger);

    Console.WriteLine(sepBar);
    Console.WriteLine("The Amazon S3 scenario has successfully completed.");
    Console.WriteLine(sepBar);
}

/// <summary>
/// Run the S3 Basics scenario with injected dependencies.
```

```

    /// </summary>
    /// <param name="s3Wrapper">The S3 wrapper instance.</param>
    /// <param name="scenarioLogger">The logger instance.</param>
    /// <returns>A Task object.</returns>
    public static async Task RunScenario(S3Wrapper s3Wrapper, ILogger<S3_Basics>
scenarioLogger)
    {
        string bucketName = BucketName;
        string filePath = TempFilePath;
        string keyName = string.Empty;

        var sepBar = new string('-', 45);

        try
        {
            // Create a bucket.
            Console.WriteLine($"\\n{sepBar}");
            Console.WriteLine("\\nCreate a new Amazon S3 bucket.\\n");
            Console.WriteLine(sepBar);

            if (IsInteractive)
            {
                Console.Write("Please enter a name for the new bucket: ");
                bucketName = Console.ReadLine();
            }
            else
            {
                Console.WriteLine($"Using bucket name: {bucketName}");
            }

            var success = await s3Wrapper.CreateBucketAsync(bucketName);
            if (success)
            {
                Console.WriteLine($"Successfully created bucket: {bucketName}.
\\n");
            }
            else
            {
                Console.WriteLine($"Could not create bucket: {bucketName}.\\n");
            }

            Console.WriteLine(sepBar);
            Console.WriteLine("Upload a file to the new bucket.");
            Console.WriteLine(sepBar);
        }
    }
}

```

```
if (IsInteractive)
{
    // Get the local path and filename for the file to upload.
    while (string.IsNullOrEmpty(filePath))
    {
        Console.WriteLine("Please enter the path and filename of the file
to upload: ");
        filePath = Console.ReadLine();

        // Confirm that the file exists on the local computer.
        if (!File.Exists(filePath))
        {
            Console.WriteLine($"Couldn't find {filePath}. Try again.
\n");
            filePath = string.Empty;
        }
    }
}
else
{
    // Use the public variable if set, otherwise create a temp file
    if (!string.IsNullOrEmpty(TempFilePath))
    {
        filePath = TempFilePath;
        Console.WriteLine($"Using provided test file: {filePath}");
    }
    else
    {
        // Create a temporary test file for non-interactive mode
        filePath = Path.GetTempFileName();
        var testContent = "This is a test file for S3 basics
scenario.\nGenerated on: " + DateTime.UtcNow.ToString("yyyy-MM-dd HH:mm:ss
UTC");

        await File.WriteAllTextAsync(filePath, testContent);
        Console.WriteLine($"Created temporary test file:
{filePath}");
    }
}

// Get the file name from the full path.
keyName = Path.GetFileName(filePath);
```

```
        success = await s3Wrapper.UploadFileAsync(bucketName, keyName,
filePath);

        if (success)
        {
            Console.WriteLine($"Successfully uploaded {keyName} from
{filePath} to {bucketName}.\n");
        }
        else
        {
            Console.WriteLine($"Could not upload {keyName}.\n");
        }

        // Set up download path
        string downloadPath = string.Empty;

        if (IsInteractive)
        {
            // Now get a new location where we can save the file.
            while (string.IsNullOrEmpty(downloadPath))
            {
                // First get the path to which the file will be downloaded.
                Console.Write("Please enter the path where the file will be
downloaded: ");
                downloadPath = Console.ReadLine();

                // Confirm that the file doesn't already exist on the local
computer.
                if (File.Exists($"{downloadPath}\\{keyName}"))
                {
                    Console.WriteLine($"Sorry, the file already exists in
that location.\n");
                    downloadPath = string.Empty;
                }
            }
        }
        else
        {
            downloadPath = Path.GetTempPath();
            var downloadFile = Path.Combine(downloadPath, keyName);
            if (File.Exists(downloadFile))
            {
                File.Delete(downloadFile);
            }
        }
    }
}
```

```
        Console.WriteLine($"Using download path: {downloadPath}");
    }

    // Download an object from a bucket.
    success = await s3Wrapper.DownloadObjectFromBucketAsync(bucketName,
keyName, downloadPath);

    if (success)
    {
        Console.WriteLine($"Successfully downloaded {keyName}.\n");
    }
    else
    {
        Console.WriteLine($"Sorry, could not download {keyName}.\n");
    }

    // Copy the object to a different folder in the bucket.
    string folderName = string.Empty;

    if (IsInteractive)
    {
        while (string.IsNullOrEmpty(folderName))
        {
            Console.Write("Please enter the name of the folder to copy
your object to: ");
            folderName = Console.ReadLine();
        }
    }
    else
    {
        folderName = "test-folder";
        Console.WriteLine($"Using folder name: {folderName}");
    }

    await s3Wrapper.CopyObjectInBucketAsync(bucketName, keyName,
folderName);

    // List the objects in the bucket.
    await s3Wrapper.ListBucketContentsAsync(bucketName);

    // Delete the contents of the bucket.
    if (IsInteractive)
    {
```

```
        Console.WriteLine("Press <Enter> when you are ready to delete the
bucket contents.");
        _ = Console.ReadLine();
    }

    var deleteContentsSuccess = await
s3Wrapper.DeleteBucketContentsAsync(bucketName);
    if (deleteContentsSuccess)
    {
        Console.WriteLine($"Successfully deleted contents of
{bucketName}.\n");
    }
    else
    {
        Console.WriteLine($"Sorry, could not delete contents of
{bucketName}.\n");
    }

    if (IsInteractive)
    {
        // Deleting the bucket too quickly after separately deleting its
contents can
        // cause an error that the bucket isn't empty. To delete contents
and bucket in one
        // operation, use AmazonS3Util.DeleteS3BucketWithObjectsAsync
        Console.WriteLine("Press <Enter> when you are ready to delete the
bucket.");
        _ = Console.ReadLine();
    }
    else
    {
        // Add a small delay for non-interactive mode to ensure objects
are fully deleted.
        Console.WriteLine("Waiting a moment for objects to be fully
deleted...");
        await Task.Delay(2000);
    }

    // Delete the bucket.
    var deleteSuccess = await s3Wrapper.DeleteBucketAsync(bucketName);
    if (deleteSuccess)
    {
        Console.WriteLine($"Successfully deleted {bucketName}.\n");
    }
}
```

```
else
{
    Console.WriteLine($"Sorry, could not delete {bucketName}.\n");
}

// Clean up temporary files in non-interactive mode
if (!IsInteractive)
{
    try
    {
        if (File.Exists(filePath))
        {
            File.Delete(filePath);
            Console.WriteLine("Cleaned up temporary test file.");
        }

        var downloadFile = Path.Combine(downloadPath, keyName);
        if (File.Exists(downloadFile))
        {
            File.Delete(downloadFile);
            Console.WriteLine("Cleaned up downloaded test file.");
        }
    }
    catch (Exception ex)
    {
        scenarioLogger.LogWarning(ex, "Failed to clean up temporary
files.");
    }
}
catch (Exception ex)
{
    scenarioLogger.LogError(ex, "An error occurred during the S3 scenario
execution.");

    // Clean up on error - delete bucket if it exists
    try
    {
        if (!string.IsNullOrEmpty(bucketName))
        {
            await s3Wrapper.DeleteBucketContentsAsync(bucketName);
            await s3Wrapper.DeleteBucketAsync(bucketName);
        }
    }
}
```



```

        catch (Exception cleanupEx)
        {
            scenarioLogger.LogError(cleanupEx, "Error during cleanup.");
        }

        // Clean up temporary files in non-interactive mode
        if (!IsInteractive)
        {
            try
            {
                if (!string.IsNullOrEmpty(filePath) && File.Exists(filePath))
                {
                    File.Delete(filePath);
                }
            }
            catch (Exception fileCleanupEx)
            {
                scenarioLogger.LogWarning(fileCleanupEx, "Failed to clean up
temporary files during error handling.");
            }
        }

        throw;
    }
}

```

Amazon S3 SDK 메서드의 래퍼 클래스입니다.

```

using Amazon.S3;
using Amazon.S3.Model;

namespace S3_Actions;

/// <summary>
/// This class contains all of the methods for working with Amazon Simple
/// Storage Service (Amazon S3) buckets.
/// </summary>
public class S3Wrapper
{
    private readonly IAmazonS3 _amazonS3;

```

```
/// <summary>
/// Initializes a new instance of the <see cref="S3Wrapper"/> class.
/// </summary>
/// <param name="amazonS3">An initialized Amazon S3 client object.</param>
public S3Wrapper(IAmazonS3 amazonS3)
{
    _amazonS3 = amazonS3;
}

/// <summary>
/// Shows how to create a new Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <returns>A boolean value representing the success or failure of
/// the bucket creation process.</returns>
public async Task<bool> CreateBucketAsync(string bucketName)
{
    try
    {
        var request = new PutBucketRequest
        {
            BucketName = bucketName,
            UseClientRegion = true,
        };

        var response = await _amazonS3.PutBucketAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error creating bucket: '{ex.Message}'");
        return false;
    }
}

/// <summary>
/// Shows how to upload a file from the local computer to an Amazon S3
/// bucket.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to which the object
```

```

    /// will be uploaded.</param>
    /// <param name="objectName">The object to upload.</param>
    /// <param name="filePath">The path, including file name, of the object
    /// on the local computer to upload.</param>
    /// <returns>A boolean value indicating the success or failure of the
    /// upload procedure.</returns>
    public async Task<bool> UploadFileAsync(
        string bucketName,
        string objectName,
        string filePath)
    {
        try
        {
            var request = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = objectName,
                FilePath = filePath,
            };

            var response = await _amazonS3.PutObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error uploading {objectName}: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Shows how to download an object from an Amazon S3 bucket to the
    /// local computer.
    /// </summary>
    /// <param name="bucketName">The name of the bucket where the object is
    /// currently stored.</param>
    /// <param name="objectName">The name of the object to download.</param>
    /// <param name="filePath">The path, including filename, where the
    /// downloaded object will be stored.</param>
    /// <returns>A boolean value indicating the success or failure of the
    /// download process.</returns>
    public async Task<bool> DownloadObjectFromBucketAsync(

```

```

        string bucketName,
        string objectName,
        string filePath)
    {
        var request = new GetObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
        };

        using GetObjectResponse response = await
            _amazonS3.GetObjectAsync(request);

        try
        {
            // Save object to local file
            await response.WriteResponseStreamToFileAsync($"{filePath}\
\{objectName}", true, CancellationToken.None);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error saving {objectName}: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Copies an object in an Amazon S3 bucket to a folder within the
    /// same bucket.
    /// </summary>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
    /// object to copy is located.</param>
    /// <param name="objectName">The object to be copied.</param>
    /// <param name="folderName">The folder to which the object will
    /// be copied.</param>
    /// <returns>A boolean value that indicates the success or failure of
    /// the copy operation.</returns>
    public async Task<bool> CopyObjectInBucketAsync(
        string bucketName,
        string objectName,
        string folderName)

```

```

    {
        try
        {
            var request = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = objectName,
                DestinationBucket = bucketName,
                DestinationKey = $"{folderName}\\{objectName}",
            };
            var response = await _amazonS3.CopyObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error copying object: '{ex.Message}'");
            return false;
        }
    }
}

/// <summary>
/// Shows how to list the objects in an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket for which to list.
/// <param name="printList">True to print out the list.
/// <returns>The collection of objects.</returns>
public async Task<List<S3Object?>> ListBucketContentsAsync(string bucketName,
bool printList = true)
{
    try
    {
        var request = new ListObjectsV2Request
        {
            BucketName = bucketName,
            MaxKeys = 5,
        };

        if (printList)
        {
            Console.WriteLine("-----");
            Console.WriteLine($"Listing the contents of {bucketName}:");
            Console.WriteLine("-----");
        }
    }
}

```

```

    }

    var listObjectsV2Paginator = _amazonS3.Paginators.ListObjectsV2(new
ListObjectsV2Request
    {
        BucketName = bucketName,
    });
    var s3Objects = new List<S3Object>();
    await foreach (var response in listObjectsV2Paginator.Responses)
    {
        if (response.S3Objects != null)
        {
            s3Objects.AddRange(response.S3Objects);
        }
    }

    if (printList)
    {
        Console.WriteLine($"Number of Objects: {s3Objects.Count}");
        foreach (var entry in s3Objects)
        {
            Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
        }
    }

    return s3Objects;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error encountered on server.
Message: '{ex.Message}' getting list of objects.");
    return null;
}
}

/// <summary>
/// Delete all of the objects stored in an existing Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket from which the
/// contents will be deleted.</param>
/// <returns>A boolean value that represents the success or failure of
/// deleting all of the objects in the bucket.</returns>

```

```

public async Task<bool> DeleteBucketContentsAsync(string bucketName)
{
    // Iterate over the contents of the bucket and delete all objects.
    try
    {
        // Delete all objects in the bucket.
        var deleteList = await ListBucketContentsAsync(bucketName, false);
        if (deleteList != null && deleteList.Any())
        {
            await _amazonS3.DeleteObjectsAsync(new DeleteObjectsRequest()
            {
                BucketName = bucketName,
                Objects = deleteList.Select(o => new KeyVersion { Key =
o.Key }).ToList(),
            });
        }

        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error deleting objects: {ex.Message}");
        return false;
    }
}

/// <summary>
/// Shows how to delete an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the Amazon S3 bucket to delete.</
param>
/// <returns>A boolean value that represents the success or failure of
/// the delete operation.</returns>
public async Task<bool> DeleteBucketAsync(string bucketName)
{
    try
    {
        var request = new DeleteBucketRequest { BucketName = bucketName, };

        await _amazonS3.DeleteBucketAsync(request);
        return true;
    }
}

```

```

        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error deleting bucket: {ex.Message}");
            return false;
        }
    }
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 다음 항목을 참조하세요.

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

Bash

AWS CLIBash 스크립트와 함께 사용

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

#####
# function s3_getting_started
#
# This function creates, copies, and deletes S3 buckets and objects.
#
# Returns:
#     0 - If successful.
#     1 - If an error occurred.

```



```
#####
function s3_getting_started() {
{
    if [ "$BUCKET_OPERATIONS_SOURCED" != "True" ]; then
        cd bucket-lifecycle-operations || exit

        source ./bucket_operations.sh
        cd ..
    fi
}

echo_repeat "*" 88
echo "Welcome to the Amazon S3 getting started demo."
echo_repeat "*" 88
    echo "A unique bucket will be created by appending a Universally Unique
Identifier to a bucket name prefix."
    echo -n "Enter a prefix for the S3 bucket that will be used in this demo: "
    get_input
    bucket_name_prefix=$get_input_result
local bucket_name
bucket_name=$(generate_random_name "$bucket_name_prefix")

local region_code
region_code=$(aws configure get region)

if create_bucket -b "$bucket_name" -r "$region_code"; then
    echo "Created demo bucket named $bucket_name"
else
    errecho "The bucket failed to create. This demo will exit."
    return 1
fi

local file_name
while [ -z "$file_name" ]; do
    echo -n "Enter a file you want to upload to your bucket: "
    get_input
    file_name=$get_input_result

    if [ ! -f "$file_name" ]; then
        echo "Could not find file $file_name. Are you sure it exists?"
        file_name=""
    fi
done

```

```
local key
key="$(basename "$file_name")"

local result=0
if copy_file_to_bucket "$bucket_name" "$file_name" "$key"; then
    echo "Uploaded file $file_name into bucket $bucket_name with key $key."
else
    result=1
fi

local destination_file
destination_file="$file_name.download"
if yes_no_input "Would you like to download $key to the file $destination_file?
(y/n) "; then
    if download_object_from_bucket "$bucket_name" "$destination_file" "$key";
then
        echo "Downloaded $key in the bucket $bucket_name to the file
$destination_file."
    else
        result=1
    fi
fi

if yes_no_input "Would you like to copy $key a new object key in your bucket?
(y/n) "; then
    local to_key
    to_key="demo/$key"
    if copy_item_in_bucket "$bucket_name" "$key" "$to_key"; then
        echo "Copied $key in the bucket $bucket_name to the $to_key."
    else
        result=1
    fi
fi

local bucket_items
bucket_items=$(list_items_in_bucket "$bucket_name")

# shellcheck disable=SC2181
if [[ $? -ne 0 ]]; then
    result=1
fi

echo "Your bucket contains the following items."
echo -e "Name\t\tSize"
```

```

echo "$bucket_items"

if yes_no_input "Delete the bucket, $bucket_name, as well as the objects in it?
(y/n) "; then
    bucket_items=$(echo "$bucket_items" | cut -f 1)

    if delete_items_in_bucket "$bucket_name" "$bucket_items"; then
        echo "The following items were deleted from the bucket $bucket_name"
        echo "$bucket_items"
    else
        result=1
    fi

    if delete_bucket "$bucket_name"; then
        echo "Deleted the bucket $bucket_name"
    else
        result=1
    fi
fi

return $result
}

```

이 시나리오에 사용된 Amazon S3 함수입니다.

```

#####
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#     -b bucket_name  -- The name of the bucket to create.
#     -r region_code  -- The code for an AWS Region in which to
#                        create the bucket.
#
# Returns:
#     The URL of the bucket that was created.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####

```

```

function create_bucket() {
    local bucket_name region_code response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function create_bucket"
        echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
        echo "  -b bucket_name    The name of the bucket. It must be globally
unique."
        echo "  [-r region_code]    The code for an AWS Region in which the bucket is
created."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "b:r:h" option; do
        case "${option}" in
            b) bucket_name="${OPTARG}" ;;
            r) region_code="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done

    if [[ -z "$bucket_name" ]]; then
        errecho "ERROR: You must provide a bucket name with the -b parameter."
        usage
        return 1
    fi

    local bucket_config_arg
    # A location constraint for "us-east-1" returns an error.
    if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
        bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
    fi

```

```

iecho "Parameters:\n"
iecho "    Bucket name:  $bucket_name"
iecho "    Region code:  $region_code"
iecho ""

# If the bucket already exists, we don't want to try to create it.
if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
fi

# shellcheck disable=SC2086
response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

# shellcheck disable=SC2181
if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
fi
}

#####
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket to copy the file to.
#     $2 - The path and file name of the local file to copy to the bucket.
#     $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function copy_file_to_bucket() {
    local response bucket_name source_file destination_file_name
    bucket_name=$1
    source_file=$2
    destination_file_name=$3

```

```

response=$(aws s3api put-object \
  --bucket "$bucket_name" \
  --body "$source_file" \
  --key "$destination_file_name")

# shellcheck disable=SC2181
if [[ ${?} -ne 0 ]]; then
  errecho "ERROR: AWS reports put-object operation failed.\n$response"
  return 1
fi
}

#####
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#   $1 - The name of the bucket to download the object from.
#   $2 - The path and file name to store the downloaded bucket.
#   $3 - The key (name) of the object in the bucket.
#
# Returns:
#   0 - If successful.
#   1 - If it fails.
#####
function download_object_from_bucket() {
  local bucket_name=$1
  local destination_file_name=$2
  local object_name=$3
  local response

  response=$(aws s3api get-object \
    --bucket "$bucket_name" \
    --key "$object_name" \
    "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}

```

```
#####
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#     $1 - The name of the bucket to copy the file from and to.
#     $2 - The key of the source file to copy.
#     $3 - The key of the destination file.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function copy_item_in_bucket() {
    local bucket_name=$1
    local source_key=$2
    local destination_key=$3
    local response

    response=$(aws s3api copy-object \
        --bucket "$bucket_name" \
        --copy-source "$bucket_name/$source_key" \
        --key "$destination_key")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api copy-object operation failed.\n$response"
        return 1
    fi
}

#####
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#     $1 - The name of the bucket.
#
# Returns:
#     The list of files in text format.
```

```

#      And:
#      0 - If successful.
#      1 - If it fails.
#####
function list_items_in_bucket() {
    local bucket_name=$1
    local response

    response=$(aws s3api list-objects \
        --bucket "$bucket_name" \
        --output text \
        --query 'Contents[].{Key: Key, Size: Size}')

    # shellcheck disable=SC2181
    if [[ ${?} -eq 0 ]]; then
        echo "$response"
    else
        errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
        return 1
    fi
}

#####
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.
#     $2 - A list of keys in the bucket to delete.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_items_in_bucket() {
    local bucket_name=$1
    local keys=$2
    local response

    # Create the JSON for the items to delete.
    local delete_items
    delete_items="{\"Objects\":["
    for key in $keys; do

```



```

    delete_items="$delete_items{\"Key\": \"$key\"},"
done
delete_items=${delete_items%?} # Remove the final comma.
delete_items="$delete_items]}"

response=$(aws s3api delete-objects \
    --bucket "$bucket_name" \
    --delete "$delete_items")

# shellcheck disable=SC2181
if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api delete-object operation failed.\n
$response"
    return 1
fi
}

#####
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_bucket() {
    local bucket_name=$1
    local response

    response=$(aws s3api delete-bucket \
        --bucket "$bucket_name")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
        return 1
    fi
}

```

- API 세부 정보는 AWS CLI 명령 참조의 다음 토픽을 참조하세요.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

C++

SDK for C++

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/CopyObjectRequest.h>
#include <aws/s3/model/CreateBucketRequest.h>
#include <aws/s3/model/DeleteBucketRequest.h>
#include <aws/s3/model/DeleteObjectRequest.h>
#include <aws/s3/model/GetObjectRequest.h>
#include <aws/s3/model/ListObjectsV2Request.h>
#include <aws/s3/model/PutObjectRequest.h>
#include <aws/s3/model/BucketLocationConstraint.h>
#include <aws/s3/model/CreateBucketConfiguration.h>
#include <aws/core/utils/UUID.h>
#include <aws/core/utils/StringUtils.h>
#include <aws/core/utils/memory/stl/AWSAllocator.h>
#include <fstream>
#include "s3_examples.h"

namespace AwsDoc {
    namespace S3 {
```

```

    //! Delete an S3 bucket.
    /*!
        \param bucketName: The S3 bucket's name.
        \param client: An S3 client.
        \return bool: Function succeeded.
    */
    static bool
    deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client &client);

    //! Delete an object in an S3 bucket.
    /*!
        \param bucketName: The S3 bucket's name.
        \param key: The key for the object in the S3 bucket.
        \param client: An S3 client.
        \return bool: Function succeeded.
    */
    static bool
    deleteObjectFromBucket(const Aws::String &bucketName, const Aws::String
&key,
                           Aws::S3::S3Client &client);
}

}

//! Scenario to create, copy, and delete S3 buckets and objects.
/*!
    \param bucketNamePrefix: A prefix for a bucket name.
    \param uploadFilePath: Path to file to upload to an Amazon S3 bucket.
    \param saveFilePath: Path for saving a downloaded S3 object.
    \param clientConfig: Aws client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::S3_GettingStartedScenario(const Aws::String &bucketNamePrefix,
      const Aws::String &uploadFilePath,
      const Aws::String &saveFilePath,
      const Aws::Client::ClientConfiguration
&clientConfig) {

    Aws::S3::S3Client client(clientConfig);

    // Create a unique bucket name which is only temporary and will be deleted.
    // Format: <bucketNamePrefix> + "-" + lowercase UUID.
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();

```

```

    Aws::String bucketName = bucketNamePrefix +
                             Aws::Utils::StringUtils::ToLower(uuid.c_str());

    // 1. Create a bucket.
    {
        Aws::S3::Model::CreateBucketRequest request;
        request.SetBucket(bucketName);

        if (clientConfig.region != Aws::Region::US_EAST_1) {
            Aws::S3::Model::CreateBucketConfiguration createBucketConfiguration;
            createBucketConfiguration.WithLocationConstraint(

Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                clientConfig.region));
            request.WithCreateBucketConfiguration(createBucketConfiguration);
        }

        Aws::S3::Model::CreateBucketOutcome outcome =
client.CreateBucket(request);

        if (!outcome.IsSuccess()) {
            const Aws::S3::S3Error &err = outcome.GetError();
            std::cerr << "Error: createBucket: " <<
                err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
            return false;
        } else {
            std::cout << "Created the bucket, '" << bucketName <<
                "'", in the region, '" << clientConfig.region << "'." <<
std::endl;
        }
    }

    // 2. Upload a local file to the bucket.
    Aws::String key = "key-for-test";
    {
        Aws::S3::Model::PutObjectRequest request;
        request.SetBucket(bucketName);
        request.SetKey(key);

        std::shared_ptr<Aws::FStream> input_data =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                uploadFilePath,
                std::ios_base::in |

```

```

        std::ios_base::binary);

    if (!input_data->is_open()) {
        std::cerr << "Error: unable to open file, '" << uploadFilePath <<
        "'."
        << std::endl;
        AwsDoc::S3::deleteBucket(bucketName, client);
        return false;
    }

    request.SetBody(input_data);

    Aws::S3::Model::PutObjectOutcome outcome =
        client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putObject: " <<
            outcome.GetError().GetMessage() << std::endl;
        AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);
        AwsDoc::S3::deleteBucket(bucketName, client);
        return false;
    } else {
        std::cout << "Added the object with the key, '" << key
            << "', to the bucket, '"
            << bucketName << "'." << std::endl;
    }
}

// 3. Download the object to a local file.
{
    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::GetObjectOutcome outcome =
        client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
        std::endl;
    } else {
        std::cout << "Downloaded the object with the key, '" << key

```

```

        << "'", in the bucket, '"
        << bucketName << "'.'" << std::endl;

    Aws::IOStream &iostream = outcome.GetResultWithOwnership().
        GetBody();
    Aws::OStream outStream(saveFilePath,
                           std::ios_base::out | std::ios_base::binary);
    if (!outStream.is_open()) {
        std::cout << "Error: unable to open file, '" << saveFilePath <<
        "'.'"
        << std::endl;
    } else {
        outStream << iostream.rdbuf();
        std::cout << "Wrote the downloaded object to the file '"
        << saveFilePath << "'.'" << std::endl;
    }
}

// 4. Copy the object to a different "folder" in the bucket.
Aws::String copiedToKey = "test-folder/" + key;
{
    Aws::S3::Model::CopyObjectRequest request;
    request.WithBucket(bucketName)
        .WithKey(copiedToKey)
        .WithCopySource(bucketName + "/" + key);

    Aws::S3::Model::CopyObjectOutcome outcome =
        client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error: copyObject: " <<
        outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Copied the object with the key, '" << key
        << "'", to the key, '" << copiedToKey
        << "', in the bucket, '" << bucketName << "'.'" << std::endl;
    }
}

// 5. List objects in the bucket.
{
    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);

```

```

    Aws::String continuationToken;
    Aws::Vector<Aws::S3::Model::Object> allObjects;

    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }
        Aws::S3::Model::ListObjectsV2Outcome outcome = client.ListObjectsV2(
            request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: ListObjects: " <<
                outcome.GetError().GetMessage() << std::endl;
            break;
        } else {
            Aws::Vector<Aws::S3::Model::Object> objects =
                outcome.GetResult().GetContents();
            allObjects.insert(allObjects.end(), objects.begin(),
objects.end());
            continuationToken = outcome.GetResult().GetContinuationToken();
        }
    } while (!continuationToken.empty());

    std::cout << allObjects.size() << " objects in the bucket, " <<
bucketName
        << ":" << std::endl;

    for (Aws::S3::Model::Object &object: allObjects) {
        std::cout << "    " << object.GetKey() << " " << std::endl;
    }
}

// 6. Delete all objects in the bucket.
// All objects in the bucket must be deleted before deleting the bucket.
AwsDoc::S3::deleteObjectFromBucket(bucketName, copiedToKey, client);
AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);

// 7. Delete the bucket.
return AwsDoc::S3::deleteBucket(bucketName, client);
}

bool AwsDoc::S3::deleteObjectFromBucket(const Aws::String &bucketName,
                                        const Aws::String &key,
                                        Aws::S3::S3Client &client) {

```

```

    Aws::S3::Model::DeleteObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::DeleteObjectOutcome outcome =
        client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: deleteObject: " <<
            outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Deleted the object with the key, '" << key
            << "', from the bucket, '"
            << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

bool
AwsDoc::S3::deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client
    &client) {
    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
        client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "Deleted the bucket, '" << bucketName << "'." << std::endl;
    }
    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 다음 항목을 참조하세요.
 - [CopyObject](#)
 - [CreateBucket](#)

- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

Go

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

시나리오가 사용하는 버킷 및 객체 작업을 래핑하는 구조를 정의합니다.

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)  
// actions  
// used in the examples.
```

```
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// ListBuckets lists the buckets in the current account.
func (basics BucketBasics) ListBuckets(ctx context.Context) ([]types.Bucket,
error) {
    var err error
    var output *s3.ListBucketsOutput
    var buckets []types.Bucket
    bucketPaginator := s3.NewListBucketsPaginator(basics.S3Client,
&s3.ListBucketsInput{})
    for bucketPaginator.HasMorePages() {
        output, err = bucketPaginator.NextPage(ctx)
        if err != nil {
            var apiErr smithy.APIError
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "AccessDenied" {
                fmt.Println("You don't have permission to list buckets for this account.")
                err = apiErr
            } else {
                log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
            }
            break
        } else {
            buckets = append(buckets, output.Buckets...)
        }
    }
    return buckets, err
}

// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(ctx context.Context, bucketName string)
(bool, error) {
    _, err := basics.S3Client.HeadBucket(ctx, &s3.HeadBucketInput{
        Bucket: aws.String(bucketName),
    })
    exists := true
}
```

```

if err != nil {
    var apiError smithy.APIError
    if errors.As(err, &apiError) {
        switch apiError.(type) {
            case *types.NotFound:
                log.Printf("Bucket %v is available.\n", bucketName)
                exists = false
                err = nil
            default:
                log.Printf("Either you don't have access to bucket %v or another error
occurred. "+
                    "Here's what happened: %v\n", bucketName, err)
        }
    }
} else {
    log.Printf("Bucket %v exists and you already own it.", bucketName)
}

return exists, err
}

// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(ctx context.Context, name string, region
string) error {
    _, err := basics.S3Client.CreateBucket(ctx, &s3.CreateBucketInput{
        Bucket: aws.String(name),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    })
    if err != nil {
        var owned *types.BucketAlreadyOwnedByYou
        var exists *types.BucketAlreadyExists
        if errors.As(err, &owned) {
            log.Printf("You already own bucket %s.\n", name)
            err = owned
        } else if errors.As(err, &exists) {
            log.Printf("Bucket %s already exists.\n", name)
            err = exists
        }
    } else {
        err = s3.NewBucketExistsWaiter(basics.S3Client).Wait(

```

```

    ctx, &s3.HeadBucketInput{Bucket: aws.String(name)}, time.Minute)
if err != nil {
    log.Printf("Failed attempt to wait for bucket %s to exist.\n", name)
}
}
return err
}

// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    file, err := os.Open(fileName)
    if err != nil {
        log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
    } else {
        defer file.Close()
        _, err = basics.S3Client.PutObject(ctx, &s3.PutObjectInput{
            Bucket: aws.String(bucketName),
            Key:    aws.String(objectKey),
            Body:   file,
        })
        if err != nil {
            var apiErr smithy.APIError
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
                log.Printf("Error while uploading object to %s. The object is too large.\n"+
                    "To upload objects larger than 5GB, use the S3 console (160GB max)\n"+
                    "or the multipart upload API (5TB max).", bucketName)
            } else {
                log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
                    fileName, bucketName, objectKey, err)
            }
        }
    }
    err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
        ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
aws.String(objectKey)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
    }
}
return err
}

```

```
// UploadLargeObject uses an upload manager to upload data to an object in a
// bucket.
// The upload manager breaks large data into parts and uploads the parts
// concurrently.
func (basics BucketBasics) UploadLargeObject(ctx context.Context, bucketName
    string, objectKey string, largeObject []byte) error {
    largeBuffer := bytes.NewReader(largeObject)
    var partMiBs int64 = 10
    uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
        u.PartSize = partMiBs * 1024 * 1024
    })
    _, err := uploader.Upload(ctx, &s3.PutObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
        Body:    largeBuffer,
    })
    if err != nil {
        var apiErr smithy.APIError
        if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
            log.Printf("Error while uploading object to %s. The object is too large.\n"+
                "The maximum size for a multipart upload is 5TB.", bucketName)
        } else {
            log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
                bucketName, objectKey, err)
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
                aws.String(objectKey)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
        }
    }

    return err
}

// DownloadFile gets an object from a bucket and stores it in a local file.
```

```

func (basics BucketBasics) DownloadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    result, err := basics.S3Client.GetObject(ctx, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
    })
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Can't get object %s from bucket %s. No such key exists.\n",
                objectKey, bucketName)
            err = noKey
        } else {
            log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
                objectKey, err)
        }
        return err
    }
    defer result.Body.Close()
    file, err := os.Create(fileName)
    if err != nil {
        log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
        return err
    }
    defer file.Close()
    body, err := io.ReadAll(result.Body)
    if err != nil {
        log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
            err)
    }
    _, err = file.Write(body)
    return err
}

// DownloadLargeObject uses a download manager to download an object from a
// bucket.
// The download manager gets the data in parts and writes them to a buffer until
// all of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
    string, objectKey string) ([]byte, error) {
    var partMiBs int64 = 10

```

```

downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
{
    d.PartSize = partMiBs * 1024 * 1024
})
buffer := manager.NewWriteAtBuffer([]byte{})
_, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
    Bucket: aws.String(bucketName),
    Key:    aws.String(objectKey),
})
if err != nil {
    log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
        bucketName, objectKey, err)
}
return buffer.Bytes(), err
}

// CopyToFolder copies an object in a bucket to a subfolder in the same bucket.
func (basics BucketBasics) CopyToFolder(ctx context.Context, bucketName string,
    objectKey string, folderName string) error {
    objectDest := fmt.Sprintf("%v/%v", folderName, objectKey)
    _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
        Bucket:      aws.String(bucketName),
        CopySource:  aws.String(fmt.Sprintf("%v/%v", bucketName, objectKey)),
        Key:         aws.String(objectDest),
    })
    if err != nil {
        var notActive *types.ObjectNotInActiveTierError
        if errors.As(err, &notActive) {
            log.Printf("Couldn't copy object %s from %s because the object isn't in the
active tier.\n",
                objectKey, bucketName)
            err = notActive
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
aws.String(objectDest)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectDest)
        }
    }
    return err
}

```

```

}

// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(ctx context.Context, sourceBucket string,
    destinationBucket string, objectKey string) error {
    _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
        Bucket:      aws.String(destinationBucket),
        CopySource:  aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
        Key:         aws.String(objectKey),
    })
    if err != nil {
        var notActive *types.ObjectNotInActiveTierError
        if errors.As(err, &notActive) {
            log.Printf("Couldn't copy object %s from %s because the object isn't in the
                active tier.\n",
                objectKey, sourceBucket)
            err = notActive
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(destinationBucket), Key:
                aws.String(objectKey)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
        }
    }
    return err
}

// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(ctx context.Context, bucketName string)
    ([]types.Object, error) {
    var err error
    var output *s3.ListObjectsV2Output
    input := &s3.ListObjectsV2Input{
        Bucket: aws.String(bucketName),
    }
    var objects []types.Object
    objectPaginator := s3.NewListObjectsV2Paginator(basics.S3Client, input)
    for objectPaginator.HasMorePages() {

```



```

output, err = objectPaginator.NextPage(ctx)
if err != nil {
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucketName)
        err = noBucket
    }
    break
} else {
    objects = append(objects, output.Contents...)
}
}
return objects, err
}

// DeleteObjects deletes a list of objects from a bucket.
func (basics BucketBasics) DeleteObjects(ctx context.Context, bucketName string,
    objectKeys []string) error {
    var objectIds []types.ObjectIdentifier
    for _, key := range objectKeys {
        objectIds = append(objectIds, types.ObjectIdentifier{Key: aws.String(key)})
    }
    output, err := basics.S3Client.DeleteObjects(ctx, &s3.DeleteObjectsInput{
        Bucket: aws.String(bucketName),
        Delete: &types.Delete{Objects: objectIds, Quiet: aws.Bool(true)},
    })
    if err != nil || len(output.Errors) > 0 {
        log.Printf("Error deleting objects from bucket %s.\n", bucketName)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucketName)
                err = noBucket
            }
        }
        if len(output.Errors) > 0 {
            for _, outErr := range output.Errors {
                log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
            }
            err = fmt.Errorf("%s", *output.Errors[0].Message)
        }
    } else {
        for _, delObjs := range output.Deleted {

```

```

    err = s3.NewObjectNotExistsWaiter(basics.S3Client).Wait(
        ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key: del0bjs.Key},
        time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s to be deleted.\n",
            *del0bjs.Key)
    } else {
        log.Printf("Deleted %s.\n", *del0bjs.Key)
    }
}
}
return err
}

// DeleteBucket deletes a bucket. The bucket must be empty or an error is
// returned.
func (basics BucketBasics) DeleteBucket(ctx context.Context, bucketName string)
error {
    _, err := basics.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
        Bucket: aws.String(bucketName)})
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucketName)
            err = noBucket
        } else {
            log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
        }
    } else {
        err = s3.NewBucketNotExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(bucketName)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
                bucketName)
        } else {
            log.Printf("Deleted %s.\n", bucketName)
        }
    }
}
return err
}

```

S3 버킷과 객체를 다루는 방법을 보여주는 대화형 시나리오를 실행합니다.

```
import (
    "context"
    "fmt"
    "log"
    "os"
    "strings"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunGetStartedScenario is an interactive example that shows you how to use
// Amazon
// Simple Storage Service (Amazon S3) to create an S3 bucket and use it to store
// objects.
//
// 1. Create a bucket.
// 2. Upload a local file to the bucket.
// 3. Download an object to a local file.
// 4. Copy an object to a different folder in the bucket.
// 5. List objects in the bucket.
// 6. Delete all objects in the bucket.
// 7. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig
// so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ../..demotools folder of this repo.
func RunGetStartedScenario(ctx context.Context, sdkConfig aws.Config, questioner
    demotools.IQuestioner) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
        }
    }()
}
```



```

    log.Println("Bucket created.")
}
}
log.Println(strings.Repeat("-", 88))

fmt.Println("Let's upload a file to your bucket.")
smallFile := questioner.Ask("Enter the path to a file you want to upload:",
    demotools.NotEmpty{})
const smallKey = "doc-example-key"
err = bucketBasics.UploadFile(ctx, bucketName, smallKey, smallFile)
if err != nil {
    panic(err)
}
log.Printf("Uploaded %v as %v.\n", smallFile, smallKey)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download %v to a file.", smallKey)
downloadFileName := questioner.Ask("Enter a name for the downloaded file:",
    demotools.NotEmpty{})
err = bucketBasics.DownloadFile(ctx, bucketName, smallKey, downloadFileName)
if err != nil {
    panic(err)
}
log.Printf("File %v downloaded.", downloadFileName)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's copy %v to a folder in the same bucket.", smallKey)
folderName := questioner.Ask("Enter a folder name: ", demotools.NotEmpty{})
err = bucketBasics.CopyToFolder(ctx, bucketName, smallKey, folderName)
if err != nil {
    panic(err)
}
log.Printf("Copied %v to %v/%v.\n", smallKey, folderName, smallKey)
log.Println(strings.Repeat("-", 88))

log.Println("Let's list the objects in your bucket.")
questioner.Ask("Press Enter when you're ready.")
objects, err := bucketBasics.ListObjects(ctx, bucketName)
if err != nil {
    panic(err)
}
log.Printf("Found %v objects.\n", len(objects))
var objKeys []string
for _, object := range objects {

```


- [PutObject](#)

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

시나리오 예제.

```
import java.io.IOException;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * 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
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an Amazon S3 bucket.
 * 2. Uploads an object to the bucket.
 * 3. Downloads the object to another local file.
 * 4. Uploads an object using multipart upload.
 * 5. List all objects located in the Amazon S3 bucket.
 * 6. Copies the object to another Amazon S3 bucket.
 * 7. Copy the object to another Amazon S3 bucket using multi copy.
```

- * 8. Deletes the object from the Amazon S3 bucket.
- * 9. Deletes the Amazon S3 bucket.
- */

```
public class S3Scenario {

    public static Scanner scanner = new Scanner(System.in);
    static S3Actions s3Actions = new S3Actions();
    public static final String DASHES = new String(new char[80]).replace("\0",
"-");
    private static final Logger logger =
LoggerFactory.getLogger(S3Scenario.class);
    public static void main(String[] args) throws IOException {
        final String usage = ""
            Usage:
                <bucketName> <key> <objectPath> <savePath> <toBucket>

        Where:
            bucketName - The name of the S3 bucket.
            key - The unique identifier for the object stored in the S3
bucket.
            objectPath - The full file path of the object within the S3
bucket (e.g., "documents/reports/annual_report.pdf").
            savePath - The local file path where the object will be
downloaded and saved (e.g., "C:/Users/username/Downloads/annual_report.pdf").
            toBucket - The name of the S3 bucket to which the object will be
copied.

        """;

        if (args.length != 5) {
            logger.info(usage);
            return;
        }

        String bucketName = args[0];
        String key = args[1];
        String objectPath = args[2];
        String savePath = args[3];
        String toBucket = args[4];

        logger.info(DASHES);
        logger.info("Welcome to the Amazon Simple Storage Service (S3) example
scenario.");
        logger.info(""
```


Amazon S3 is a highly scalable and durable object storage service provided by Amazon Web Services (AWS). It is designed to store and retrieve any amount of data, from anywhere on the web, at any time.

The `S3AsyncClient` interface in the AWS SDK for Java 2.x provides a set of methods to programmatically interact with the Amazon S3 (Simple Storage Service) service. This allows developers to automate the management and manipulation of S3 buckets and objects as part of their application deployment pipelines. With S3, teams can focus on building and deploying their applications without having to worry about the underlying storage infrastructure required to host and manage large amounts of data.

This scenario walks you through how to perform key operations for this service.

Let's get started...

```

""");
waitForInputToContinue(scanner);
logger.info(DASHES);

try {
    // Run the methods that belong to this scenario.
    runScenario(bucketName, key, objectPath, savePath, toBucket);

} catch (Throwable rt) {
    Throwable cause = rt.getCause();
    if (cause instanceof S3Exception kmsEx) {
        logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
    } else {
        logger.info("An unexpected error occurred: " + rt.getMessage());
    }
}

private static void runScenario(String bucketName, String key, String
objectPath, String savePath, String toBucket) throws Throwable {
    logger.info(DASHES);
    logger.info("1. Create an Amazon S3 bucket.");
    try {

```

```
        CompletableFuture<Void> future =
s3Actions.createBucketAsync(bucketName);
        future.join();
        waitForInputToContinue(scanner);

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }

    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("2. Upload a local file to the Amazon S3 bucket.");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<PutObjectResponse> future =
s3Actions.uploadLocalFileAsync(bucketName, key, objectPath);
        future.join();
        logger.info("File uploaded successfully to {}/{}", bucketName, key);

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }

    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("3. Download the object to another local file.");
    waitForInputToContinue(scanner);
```

```
        try {
            CompletableFuture<Void> future =
s3Actions.getObjectBytesAsync(bucketName, key, savePath);
            future.join();
            logger.info("Successfully obtained bytes from S3 object and wrote to
file {}", savePath);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Perform a multipart upload.");
        waitForInputToContinue(scanner);
        String multipartKey = "multiPartKey";
        try {
            // Call the multipartUpload method
            CompletableFuture<Void> future =
s3Actions.multipartUpload(bucketName, multipartKey);
            future.join();
            logger.info("Multipart upload completed successfully for bucket '{} '
and key '{}'", bucketName, multipartKey);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
logger.info(DASHES);
logger.info("5. List all objects located in the Amazon S3 bucket.");
waitForInputToContinue(scanner);
try {
    CompletableFuture<Void> future =
s3Actions.listAllObjectsAsync(bucketName);
    future.join();
    logger.info("Object listing completed successfully.");

} catch (RuntimeException rt) {
    Throwable cause = rt.getCause();
    if (cause instanceof S3Exception s3Ex) {
        logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
    } else {
        logger.info("An unexpected error occurred: " + rt.getMessage());
    }
    throw cause;
}
waitForInputToContinue(scanner);
logger.info(DASHES);

logger.info(DASHES);
logger.info("6. Copy the object to another Amazon S3 bucket.");
waitForInputToContinue(scanner);
try {
    CompletableFuture<String> future =
s3Actions.copyBucketObjectAsync(bucketName, key, toBucket);
    String result = future.join();
    logger.info("Copy operation result: {}", result);

} catch (RuntimeException rt) {
    Throwable cause = rt.getCause();
    if (cause instanceof S3Exception s3Ex) {
        logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
    } else {
        logger.info("An unexpected error occurred: " + rt.getMessage());
    }
    throw cause;
}
waitForInputToContinue(scanner);
logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("7. Copy the object to another Amazon S3 bucket using multi
copy.");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<String> future =
s3Actions.performMultiCopy(toBucket, bucketName, key);
            String result = future.join();
            logger.info("Copy operation result: {}", result);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Delete objects from the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, key);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
        throw cause;
    }
    try {
```

```

        CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, "multiPartKey");
        future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("9. Delete the Amazon S3 bucket.");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<Void> future =
s3Actions.deleteBucketAsync(bucketName);
        future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("You successfully completed the Amazon S3 scenario.");
    logger.info(DASHES);
}

private static void waitForInputToContinue(Scanner scanner) {

```

```

        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}

```

작업이 포함된 래퍼 클래스.

```

public class S3Actions {

    private static final Logger logger =
        LoggerFactory.getLogger(S3Actions.class);
    private static S3AsyncClient s3AsyncClient;

    public static S3AsyncClient getAsyncClient() {
        if (s3AsyncClient == null) {
            /*
             * The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
             * version 2,
             * and it is designed to provide a high-performance, asynchronous HTTP
             * client for interacting with AWS services.
             * It uses the Netty framework to handle the underlying network
             * communication and the Java NIO API to
             * provide a non-blocking, event-driven approach to HTTP requests and
             * responses.
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50) // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60)) // Set the connection
            timeout.

```

```

        .readTimeout(Duration.ofSeconds(60)) // Set the read timeout.
        .writeTimeout(Duration.ofSeconds(60)) // Set the write timeout.
        .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ofMinutes(2)) // Set the overall API
call timeout.
        .apiCallAttemptTimeout(Duration.ofSeconds(90)) // Set the
individual call attempt timeout.
        .retryStrategy(RetryMode.STANDARD)
        .build();

        s3AsyncClient = S3AsyncClient.builder()
        .region(Region.US_EAST_1)
        .httpClient(httpClient)
        .overrideConfiguration(overrideConfig)
        .build();
    }
    return s3AsyncClient;
}

/**
 * Creates an S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket to create
 * @return a {@link CompletableFuture} that completes when the bucket is
created and ready
 * @throws RuntimeException if there is a failure while creating the bucket
 */
public CompletableFuture<Void> createBucketAsync(String bucketName) {
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .build();

    CompletableFuture<CreateBucketResponse> response =
getAsyncClient().createBucket(bucketRequest);
    return response.thenCompose(resp -> {
        S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();
    });
}

```



```

        CompletableFuture<WaiterResponse<HeadBucketResponse>>
waiterResponseFuture =
            s3Waiter.waitUntilBucketExists(bucketRequestWait);
        return waiterResponseFuture.thenAccept(waiterResponse -> {
            waiterResponse.matched().response().ifPresent(headBucketResponse
-> {
                logger.info(bucketName + " is ready");
            });
        });
    }).whenComplete((resp, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to create bucket", ex);
        }
    });
}

/**
 * Uploads a local file to an AWS S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket to upload the file to
 * @param key         the key (object name) to use for the uploaded file
 * @param objectPath  the local file path of the file to be uploaded
 * @return a {@link CompletableFuture} that completes with the {@link
PutObjectResponse} when the upload is successful, or throws a {@link
RuntimeException} if the upload fails
 */
public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
bucketName, String key, String objectPath) {
    PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    CompletableFuture<PutObjectResponse> response =
getAsyncClient().putObject(objectRequest,
AsyncRequestBody.fromFile(Paths.get(objectPath)));
    return response.whenComplete((resp, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to upload file", ex);
        }
    });
}
}

```

```
/**
 * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket
and writes them to a local file.
 *
 * @param bucketName the name of the S3 bucket containing the object
 * @param keyName    the key (or name) of the S3 object to retrieve
 * @param path       the local file path where the object's bytes will be
written
 * @return a {@link CompletableFuture} that completes when the object bytes
have been written to the local file
 */
public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
keyName, String path) {
    GetObjectRequest objectRequest = GetObjectRequest.builder()
        .key(keyName)
        .bucket(bucketName)
        .build();

    CompletableFuture<ResponseBytes<GetObjectResponse>> response =
getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
    return response.thenAccept(objectBytes -> {
        try {
            byte[] data = objectBytes.asByteArray();
            Path filePath = Paths.get(path);
            Files.write(filePath, data);
            logger.info("Successfully obtained bytes from an S3 object");
        } catch (IOException ex) {
            throw new RuntimeException("Failed to write data to file", ex);
        }
    }).whenComplete((resp, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to get object bytes from S3",
ex);
        }
    });
}

/**
 * Asynchronously lists all objects in the specified S3 bucket.
 *
 * @param bucketName the name of the S3 bucket to list objects for
```

```

    * @return a {@link CompletableFuture} that completes when all objects have
    been listed
    */
    public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
        ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Publisher paginator =
getAsyncClient().listObjectsV2Paginator(initialRequest);
        return paginator.subscribe(response -> {
            response.contents().forEach(s3Object -> {
                logger.info("Object key: " + s3Object.key());
            });
        }).thenRun(() -> {
            logger.info("Successfully listed all objects in the bucket: " +
bucketName);
        }).exceptionally(ex -> {
            throw new RuntimeException("Failed to list objects", ex);
        });
    }

    /**
    * Asynchronously copies an object from one S3 bucket to another.
    *
    * @param fromBucket the name of the source S3 bucket
    * @param objectKey the key (name) of the object to be copied
    * @param toBucket the name of the destination S3 bucket
    * @return a {@link CompletableFuture} that completes with the copy result as
    a {@link String}
    * @throws RuntimeException if the URL could not be encoded or an S3
    exception occurred during the copy
    */
    public CompletableFuture<String> copyBucketObjectAsync(String fromBucket,
String objectKey, String toBucket) {
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .sourceBucket(fromBucket)
            .sourceKey(objectKey)
            .destinationBucket(toBucket)
            .destinationKey(objectKey)
            .build();
    }

```

```

        CompletableFuture<CopyObjectResponse> response =
getAsyncClient().copyObject(copyReq);
        response.whenComplete((copyRes, ex) -> {
            if (copyRes != null) {
                logger.info("The " + objectKey + " was copied to " + toBucket);
            } else {
                throw new RuntimeException("An S3 exception occurred during
copy", ex);
            }
        });

        return response.thenApply(CopyObjectResponse::copyObjectResult)
            .thenApply(Object::toString);
    }

    /**
     * Performs a multipart upload to an Amazon S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param key         the key (name) of the file to be uploaded
     * @return a {@link CompletableFuture} that completes when the multipart
upload is successful
     */
    public CompletableFuture<Void> multipartUpload(String bucketName, String key)
    {
        int mB = 1024 * 1024;

        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        return
getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
            .thenCompose(createResponse -> {
                String uploadId = createResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);

                // Upload part 1.
                UploadPartRequest uploadPartRequest1 =
UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)

```

```
        .uploadId(uploadId)
        .partNumber(1)
        .contentLength((long) (5 * mB)) // Specify the content length
        .build();

        CompletableFuture<CompletedPart> part1Future =
getAsyncClient().uploadPart(uploadPartRequest1,
        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(5 *
mB)))

        .thenApply(uploadPartResponse -> CompletedPart.builder()
        .partNumber(1)
        .eTag(uploadPartResponse.eTag())
        .build());

        // Upload part 2.
        UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder()
        .bucket(bucketName)
        .key(key)
        .uploadId(uploadId)
        .partNumber(2)
        .contentLength((long) (3 * mB))
        .build();

        CompletableFuture<CompletedPart> part2Future =
getAsyncClient().uploadPart(uploadPartRequest2,
        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(3 *
mB)))

        .thenApply(uploadPartResponse -> CompletedPart.builder()
        .partNumber(2)
        .eTag(uploadPartResponse.eTag())
        .build());

        // Combine the results of both parts.
        return CompletableFuture.allOf(part1Future, part2Future)
        .thenCompose(v -> {
            CompletedPart part1 = part1Future.join();
            CompletedPart part2 = part2Future.join();

            CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                .parts(part1, part2)
                .build();
```

```

        CompleteMultipartUploadRequest
completeMultipartUploadRequest = CompleteMultipartUploadRequest.builder()
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)
    .multipartUpload(completedMultipartUpload)
    .build();

        // Complete the multipart upload
        return
getAsyncClient().completeMultipartUpload(completeMultipartUploadRequest);
    });
    })
    .thenAccept(response -> System.out.println("Multipart upload
completed successfully"))
    .exceptionally(ex -> {
        System.err.println("Failed to complete multipart upload: " +
ex.getMessage());
        throw new RuntimeException(ex);
    });
}

/**
 * Deletes an object from an S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the key (file name) of the object to be deleted
 * @return a {@link CompletableFuture} that completes when the object has
been deleted
 */
public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
String key) {
    DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
    response.whenComplete((deleteRes, ex) -> {
        if (deleteRes != null) {
            logger.info(key + " was deleted");
        } else {

```

```

        throw new RuntimeException("An S3 exception occurred during
delete", ex);
    }
    });

    return response.thenApply(r -> null);
}

/**
 * Deletes an S3 bucket asynchronously.
 *
 * @param bucket the name of the bucket to be deleted
 * @return a {@link CompletableFuture} that completes when the bucket
deletion is successful, or throws a {@link RuntimeException}
 * if an error occurs during the deletion process
 */
public CompletableFuture<Void> deleteBucketAsync(String bucket) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucket)
        .build();

    CompletableFuture<DeleteBucketResponse> response =
getAsyncClient().deleteBucket(deleteBucketRequest);
    response.whenComplete((deleteRes, ex) -> {
        if (deleteRes != null) {
            logger.info(bucket + " was deleted.");
        } else {
            throw new RuntimeException("An S3 exception occurred during
bucket deletion", ex);
        }
    });
    return response.thenApply(r -> null);
}

public CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
    CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
        .bucket(toBucket)
        .key(key)
        .build();

    getAsyncClient().createMultipartUpload(createMultipartUploadRequest)

```

```

        .thenApply(createMultipartUploadResponse -> {
            String uploadId = createMultipartUploadResponse.uploadId();
            System.out.println("Upload ID: " + uploadId);

            UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                .sourceBucket(bucketName)
                .destinationBucket(toBucket)
                .sourceKey(key)
                .destinationKey(key)
                .uploadId(uploadId) // Use the valid uploadId.
                .partNumber(1) // Ensure the part number is correct.
                .copySourceRange("bytes=0-1023") // Adjust range as needed
                .build();

            return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
        })
        .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
        .whenComplete((uploadPartCopyResponse, exception) -> {
            if (exception != null) {
                // Handle any exceptions.
                logger.error("Error during upload part copy: " +
exception.getMessage());
            } else {
                // Successfully completed the upload part copy.
                System.out.println("Upload Part Copy completed successfully.
ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
            }
        });
    return null;
}

private static ByteBuffer getRandomByteBuffer(int size) {
    ByteBuffer buffer = ByteBuffer.allocate(size);
    for (int i = 0; i < size; i++) {
        buffer.put((byte) (Math.random() * 256));
    }
    buffer.flip();
    return buffer;
}
}

```

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

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

먼저 필요한 모듈을 모두 가져옵니다.

```
// Used to check if currently running file is this file.
import { fileURLToPath } from "node:url";
import { readdirSync, readFileSync, writeFileSync } from "node:fs";

// Local helper utils.
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";
import { wrapText } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

import {
  S3Client,
  CreateBucketCommand,
  PutObjectCommand,
  ListObjectsCommand,
  CopyObjectCommand,
  GetObjectCommand,
  DeleteObjectsCommand,
  DeleteBucketCommand,
} from "@aws-sdk/client-s3";
```

이전 가져오기는 일부 도우미 유틸리티를 참조합니다. 이러한 유틸리티는 이 섹션의 시작 부분에 연결된 GitHub 리포지토리에 로컬로 제공됩니다. 참조를 위해 해당 유틸리티의 다음 구현을 참조하세요.

```
export const dirnameFromMetaUrl = (metaUrl) =>
  fileURLToPath(new URL(".", metaUrl));

import { select, input, confirm, checkbox, password } from "@inquirer/prompts";

export class Prompter {
  /**
   * @param {{ message: string, choices: { name: string, value: string }[] }}
   options
   */
  select(options) {
    return select(options);
  }

  /**
   * @param {{ message: string }} options
   */
  input(options) {
    return input(options);
  }

  /**
   * @param {{ message: string }} options
   */
  password(options) {
    return password({ ...options, mask: true });
  }

  /**
   * @param {string} prompt
   */
  checkContinue = async (prompt = "") => {
    const prefix = prompt && `${prompt} `;
    const ok = await this.confirm({
      message: `${prefix}Continue?`,
    });
    if (!ok) throw new Error("Exiting...");
  };
}
```

```

};

/**
 * @param {{ message: string }} options
 */
confirm(options) {
  return confirm(options);
}

/**
 * @param {{ message: string, choices: { name: string, value: string }[] }}
options
 */
checkbox(options) {
  return checkbox(options);
}
}

export const wrapText = (text, char = "=") => {
  const rule = char.repeat(80);
  return `${rule}\n    ${text}\n${rule}\n`;
};

```

객체는 '버킷'에 저장됩니다. 새 버킷을 만들기 위한 함수를 정의해 보겠습니다.

```

export const createBucket = async () => {
  const bucketName = await prompter.input({
    message: "Enter a bucket name. Bucket names must be globally unique:",
  });
  const command = new CreateBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log("Bucket created successfully.\n");
  return bucketName;
};

```

버킷에는 '객체'가 포함됩니다. 이 함수는 디렉터리의 콘텐츠를 버킷에 객체로 업로드합니다.

```

export const uploadFilesToBucket = async ({ bucketName, folderPath }) => {
  console.log(`Uploading files from ${folderPath}\n`);
  const keys = readdirSync(folderPath);
  const files = keys.map((key) => {

```

```

    const filePath = `${folderPath}/${key}`;
    const fileContent = readFileSync(filePath);
    return {
      Key: key,
      Body: fileContent,
    };
  });

  for (const file of files) {
    await s3Client.send(
      new PutObjectCommand({
        Bucket: bucketName,
        Body: file.Body,
        Key: file.Key,
      }),
    );
    console.log(`${file.Key} uploaded successfully.`);
  }
};

```

객체를 업로드한 후 객체가 올바르게 업로드되었는지 확인하세요. 이를 위해 `ListObjects`를 사용할 수 있습니다. 'Key' 속성을 사용하겠지만 응답에는 다른 유용한 속성도 있습니다.

```

export const listFilesInBucket = async ({ bucketName }) => {
  const command = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(command);
  const contentsList = Contents.map((c) => ` • ${c.Key}`).join("\n");
  console.log("\nHere's a list of files in the bucket:");
  console.log(`${contentsList}\n`);
};

```

때로는 한 버킷에서 다른 버킷으로 객체를 복사하고 싶을 수도 있습니다. 이를 위해서는 `CopyObject` 명령을 사용하세요.

```

export const copyFileFromBucket = async ({ destinationBucket }) => {
  const proceed = await prompter.confirm({
    message: "Would you like to copy an object from another bucket?",
  });

  if (!proceed) {

```

```

    return;
  }
  const copy = async () => {
    try {
      const sourceBucket = await prompter.input({
        message: "Enter source bucket name:",
      });
      const sourceKey = await prompter.input({
        message: "Enter source key:",
      });
      const destinationKey = await prompter.input({
        message: "Enter destination key:",
      });

      const command = new CopyObjectCommand({
        Bucket: destinationBucket,
        CopySource: `${sourceBucket}/${sourceKey}`,
        Key: destinationKey,
      });
      await s3Client.send(command);
      await copyFileFromBucket({ destinationBucket });
    } catch (err) {
      console.error("Copy error.");
      console.error(err);
      const retryAnswer = await prompter.confirm({ message: "Try again?" });
      if (retryAnswer) {
        await copy();
      }
    }
  };
  await copy();
};

```

버킷에서 여러 객체를 가져오는 SDK 메서드는 없습니다. 대신, 다운로드 및 반복할 객체 목록을 생성하겠습니다.

```

export const downloadFilesFromBucket = async ({ bucketName }) => {
  const { Contents } = await s3Client.send(
    new ListObjectsCommand({ Bucket: bucketName }),
  );
  const path = await prompter.input({
    message: "Enter destination path for files:",
  });

```

```
});

for (const content of Contents) {
  const obj = await s3Client.send(
    new GetObjectCommand({ Bucket: bucketName, Key: content.Key }),
  );
  writeFileSync(
    `${path}/${content.Key}`,
    await obj.Body.transformToByteArray(),
  );
}
console.log("Files downloaded successfully.\n");
};
```

이제 리소스를 정리할 차례입니다. 삭제하려면 버킷이 비어 있어야 합니다. 이 두 함수는 버킷을 비우고 삭제합니다.

```
export const emptyBucket = async ({ bucketName }) => {
  const listObjectsCommand = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(listObjectsCommand);
  const keys = Contents.map((c) => c.Key);

  const deleteObjectsCommand = new DeleteObjectsCommand({
    Bucket: bucketName,
    Delete: { Objects: keys.map((key) => ({ Key: key })) },
  });
  await s3Client.send(deleteObjectsCommand);
  console.log(`${bucketName} emptied successfully.\n`);
};

export const deleteBucket = async ({ bucketName }) => {
  const command = new DeleteBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log(`${bucketName} deleted successfully.\n`);
};
```

'main' 함수는 모든 것을 한데 가져옵니다. 이 파일을 직접 실행하면 main 함수가 호출됩니다.

```
const main = async () => {
  const OBJECT_DIRECTORY = `${dirnameFromMetaUrl(
    import.meta.url,
```

```
    })../../../../resources/sample_files/.sample_media`;  
  
    try {  
        console.log(wrapText("Welcome to the Amazon S3 getting started example."));  
        console.log("Let's create a bucket.");  
        const bucketName = await createBucket();  
        await prompter.confirm({ message: continueMessage });  
  
        console.log(wrapText("File upload."));  
        console.log(  
            "I have some default files ready to go. You can edit the source code to  
provide your own.",  
        );  
        await uploadFilesToBucket({  
            bucketName,  
            folderPath: OBJECT_DIRECTORY,  
        });  
  
        await listFilesInBucket({ bucketName });  
        await prompter.confirm({ message: continueMessage });  
  
        console.log(wrapText("Copy files."));  
        await copyFileFromBucket({ destinationBucket: bucketName });  
        await listFilesInBucket({ bucketName });  
        await prompter.confirm({ message: continueMessage });  
  
        console.log(wrapText("Download files."));  
        await downloadFilesFromBucket({ bucketName });  
  
        console.log(wrapText("Clean up."));  
        await emptyBucket({ bucketName });  
        await deleteBucket({ bucketName });  
    } catch (err) {  
        console.error(err);  
    }  
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 다음 항목을 참조하세요.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)

- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

Kotlin

SDK for Kotlin

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <bucketName> <key> <objectPath> <savePath> <toBucket>

    Where:
        bucketName - The Amazon S3 bucket to create.
        key - The key to use.
        objectPath - The path where the file is located (for example, C:/AWS/
book2.pdf).
        savePath - The path where the file is saved after it's downloaded (for
example, C:/AWS/book2.pdf).
        toBucket - An Amazon S3 bucket to where an object is copied to (for
example, C:/AWS/book2.pdf).
        """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val bucketName = args[0]
    val key = args[1]
    val objectPath = args[2]
    val savePath = args[3]
```



```
val toBucket = args[4]

// Create an Amazon S3 bucket.
createBucket(bucketName)

// Update a local file to the Amazon S3 bucket.
putObject(bucketName, key, objectPath)

// Download the object to another local file.
getObjectFromMap(bucketName, key, savePath)

// List all objects located in the Amazon S3 bucket.
listBucketObs(bucketName)

// Copy the object to another Amazon S3 bucket
copyBucketOb(bucketName, key, toBucket)

// Delete the object from the Amazon S3 bucket.
deleteBucketObs(bucketName, key)

// Delete the Amazon S3 bucket.
deleteBucket(bucketName)
println("All Amazon S3 operations were successfully performed")
}

suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun putObject(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"
```

```
val request =
    PutObjectRequest {
        bucket = bucketName
        key = objectKey
        metadata = metadataVal
        this.body = Paths.get(objectPath).asByteArray()
    }

S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
    val response = s3.putObject(request)
    println("Tag information is ${response.eTag}")
}

suspend fun getObjectFromMrap(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}

suspend fun listBucketObs(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->

        val response = s3.listObjects(request)
```

```
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The owner is ${myObject.owner}")
        }
    }
}

suspend fun copyBucketOb(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedOperationException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}

suspend fun deleteBucketObs(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }
}
```

```
val request =
    DeleteObjectsRequest {
        bucket = bucketName
        delete = delOb
    }

S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
    s3.deleteObjects(request)
    println("$objectName was deleted from $bucketName")
}
}

suspend fun deleteBucket(bucketName: String?) {
    val request =
        DeleteBucketRequest {
            bucket = bucketName
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하세요.

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

PHP

SDK for PHP

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
echo("\n");
echo("-----\n");
print("Welcome to the Amazon S3 getting started demo using PHP!\n");
echo("-----\n");

$region = 'us-west-2';

$this->s3client = new S3Client([
    'region' => $region,
]);
/* Inline declaration example
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
*/

$this->bucketName = "amzn-s3-demo-bucket-" . uniqid();

try {
    $this->s3client->createBucket([
        'Bucket' => $this->bucketName,
        'CreateBucketConfiguration' => ['LocationConstraint' => $region],
    ]);
    echo "Created bucket named: $this->bucketName \n";
} catch (Exception $exception) {
    echo "Failed to create bucket $this->bucketName with error: " .
    $exception->getMessage();
    exit("Please fix error with bucket creation before continuing.");
}

$fileName = __DIR__ . "/local-file-" . uniqid();
try {
    $this->s3client->putObject([
        'Bucket' => $this->bucketName,
```

```

        'Key' => $fileName,
        'SourceFile' => __DIR__ . '/testfile.txt'
    ]);
    echo "Uploaded $fileName to $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
    exit("Please fix error with file upload before continuing.");
}

try {
    $file = $this->s3client->getObject([
        'Bucket' => $this->bucketName,
        'Key' => $fileName,
    ]);
    $body = $file->get('Body');
    $body->rewind();
    echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
} catch (Exception $exception) {
    echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
    exit("Please fix error with file downloading before continuing.");
}

try {
    $folder = "copied-folder";
    $this->s3client->copyObject([
        'Bucket' => $this->bucketName,
        'CopySource' => "$this->bucketName/$fileName",
        'Key' => "$folder/$fileName-copy",
    ]);
    echo "Copied $fileName to $folder/$fileName-copy.\n";
} catch (Exception $exception) {
    echo "Failed to copy $fileName with error: " . $exception-
>getMessage();
    exit("Please fix error with object copying before continuing.");
}

try {
    $contents = $this->s3client->listObjectsV2([
        'Bucket' => $this->bucketName,
    ]);
    echo "The contents of your bucket are: \n";
    foreach ($contents['Contents'] as $content) {

```

```

        echo $content['Key'] . "\n";
    }
} catch (Exception $exception) {
    echo "Failed to list objects in $this->bucketName with error: " .
    $exception->getMessage();
    exit("Please fix error with listing objects before continuing.");
}

try {
    $objects = [];
    foreach ($contents['Contents'] as $content) {
        $objects[] = [
            'Key' => $content['Key'],
        ];
    }
    $this->s3client->deleteObjects([
        'Bucket' => $this->bucketName,
        'Delete' => [
            'Objects' => $objects,
        ],
    ]);
    $check = $this->s3client->listObjectsV2([
        'Bucket' => $this->bucketName,
    ]);
    if (isset($check['Contents']) && count($check['Contents']) > 0) {
        throw new Exception("Bucket wasn't empty.");
    }
    echo "Deleted all objects and folders from $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to delete $fileName from $this->bucketName with error:
    " . $exception->getMessage();
    exit("Please fix error with object deletion before continuing.");
}

try {
    $this->s3client->deleteBucket([
        'Bucket' => $this->bucketName,
    ]);
    echo "Deleted bucket $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to delete $this->bucketName with error: " . $exception-
    >getMessage();
    exit("Please fix error with bucket deletion before continuing.");
}

```

```
echo "Successfully ran the Amazon S3 with PHP demo.\n";
```

- API 세부 정보는 AWS SDK for PHP API 참조의 다음 항목을 참조하세요.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import io
import os
import uuid

import boto3
from boto3.s3.transfer import S3UploadFailedError
from botocore.exceptions import ClientError

def do_scenario(s3_resource):
    print("-" * 88)
    print("Welcome to the Amazon S3 getting started demo!")
    print("-" * 88)
```



```

bucket_name = f"amzn-s3-demo-bucket-{uuid.uuid4()}"
bucket = s3_resource.Bucket(bucket_name)
try:
    bucket.create(
        CreateBucketConfiguration={
            "LocationConstraint": s3_resource.meta.client.meta.region_name
        }
    )
    print(f"Created demo bucket named {bucket.name}.")
except ClientError as err:
    print(f"Tried and failed to create demo bucket {bucket_name}.")
    print(f"\t{err.response['Error']['Code']}: {err.response['Error']
['Message']}")
    print(f"\nCan't continue the demo without a bucket!")
    return

file_name = None
while file_name is None:
    file_name = input("\nEnter a file you want to upload to your bucket: ")
    if not os.path.exists(file_name):
        print(f"Couldn't find file {file_name}. Are you sure it exists?")
        file_name = None

obj = bucket.Object(os.path.basename(file_name))
try:
    obj.upload_file(file_name)
    print(
        f"Uploaded file {file_name} into bucket {bucket.name} with key
{obj.key}."
    )
except S3UploadFailedError as err:
    print(f"Couldn't upload file {file_name} to {bucket.name}.")
    print(f"\t{err}")

answer = input(f"\nDo you want to download {obj.key} into memory (y/n)? ")
if answer.lower() == "y":
    data = io.BytesIO()
    try:
        obj.download_fileobj(data)
        data.seek(0)
        print(f"Got your object. Here are the first 20 bytes:\n")
        print(f"\t{data.read(20)}")
    except ClientError as err:
        print(f"Couldn't download {obj.key}.")

```

```

        print(
            f"\t{err.response['Error']['Code']}: {err.response['Error']
['Message']}"
        )

    answer = input(
        f"\nDo you want to copy {obj.key} to a subfolder in your bucket (y/n)? "
    )
    if answer.lower() == "y":
        dest_obj = bucket.Object(f"demo-folder/{obj.key}")
        try:
            dest_obj.copy({"Bucket": bucket.name, "Key": obj.key})
            print(f"Copied {obj.key} to {dest_obj.key}.")
        except ClientError as err:
            print(f"Couldn't copy {obj.key} to {dest_obj.key}.")
            print(
                f"\t{err.response['Error']['Code']}: {err.response['Error']
['Message']}"
            )

    print("\nYour bucket contains the following objects:")
    try:
        for o in bucket.objects.all():
            print(f"\t{o.key}")
    except ClientError as err:
        print(f"Couldn't list the objects in bucket {bucket.name}.")
        print(f"\t{err.response['Error']['Code']}: {err.response['Error']
['Message']}")

    answer = input(
        "\nDo you want to delete all of the objects as well as the bucket (y/n)?
"
    )
    if answer.lower() == "y":
        try:
            bucket.objects.delete()
            bucket.delete()
            print(f"Emptied and deleted bucket {bucket.name}.\n")
        except ClientError as err:
            print(f"Couldn't empty and delete bucket {bucket.name}.")
            print(
                f"\t{err.response['Error']['Code']}: {err.response['Error']
['Message']}"
            )

```

```
print("Thanks for watching!")
print("-" * 88)

if __name__ == "__main__":
    do_scenario(boto3.resource("s3"))
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 다음 주제를 참조하세요.

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

Ruby

SDK for Ruby

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-s3'

# Wraps the getting started scenario actions.
class ScenarioGettingStarted
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end
end
```

```
# Creates a bucket with a random name in the currently configured account and
# AWS Region.
#
# @return [Aws::S3::Bucket] The newly created bucket.
def create_bucket
  bucket = @s3_resource.create_bucket(
    bucket: "amzn-s3-demo-bucket-#{Random.uuid}",
    create_bucket_configuration: {
      location_constraint: 'us-east-1' # NOTE: only certain regions permitted
    }
  )
  puts("Created demo bucket named #{bucket.name}.")
rescue Aws::Errors::ServiceError => e
  puts('Tried and failed to create demo bucket.')
  puts("\t#{e.code}: #{e.message}")
  puts("\nCan't continue the demo without a bucket!")
  raise
else
  bucket
end

# Requests a file name from the user.
#
# @return The name of the file.
def create_file
  File.open('demo.txt', w) { |f| f.write('This is a demo file.') }
end

# Uploads a file to an Amazon S3 bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket object representing the upload
destination
# @return [Aws::S3::Object] The Amazon S3 object that contains the uploaded
file.
def upload_file(bucket)
  File.open('demo.txt', 'w+') { |f| f.write('This is a demo file.') }
  s3_object = bucket.object(File.basename('demo.txt'))
  s3_object.upload_file('demo.txt')
  puts("Uploaded file demo.txt into bucket #{bucket.name} with key
#{s3_object.key}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't upload file demo.txt to #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
end
```

```

    raise
  else
    s3_object
  end

  # Downloads an Amazon S3 object to a file.
  #
  # @param s3_object [Aws::S3::Object] The object to download.
  def download_file(s3_object)
    puts("\nDo you want to download #{s3_object.key} to a local file (y/n)? ")
    answer = gets.chomp.downcase
    if answer == 'y'
      puts('Enter a name for the downloaded file: ')
      file_name = gets.chomp
      s3_object.download_file(file_name)
      puts("Object #{s3_object.key} successfully downloaded to #{file_name}.")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't download #{s3_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end

  # Copies an Amazon S3 object to a subfolder within the same bucket.
  #
  # @param source_object [Aws::S3::Object] The source object to copy.
  # @return [Aws::S3::Object, nil] The destination object.
  def copy_object(source_object)
    dest_object = nil
    puts("\nDo you want to copy #{source_object.key} to a subfolder in your
    bucket (y/n)? ")
    answer = gets.chomp.downcase
    if answer == 'y'
      dest_object = source_object.bucket.object("demo-folder/
      #{source_object.key}")
      dest_object.copy_from(source_object)
      puts("Copied #{source_object.key} to #{dest_object.key}.")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't copy #{source_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    dest_object
  end

```

```
end

# Lists the objects in an Amazon S3 bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to query.
def list_objects(bucket)
  puts("\nYour bucket contains the following objects:")
  bucket.objects.each do |obj|
    puts("\t#{obj.key}")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't list the objects in bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end

# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?")
  answer = gets.chomp.downcase
  if answer == 'y'
    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end

end

# Runs the Amazon S3 getting started scenario.
def run_scenario(scenario)
  puts('-' * 88)
  puts('Welcome to the Amazon S3 getting started demo!')
  puts('-' * 88)

  bucket = scenario.create_bucket
  s3_object = scenario.upload_file(bucket)
  scenario.download_file(s3_object)
```

```

scenario.copy_object(s3_object)
scenario.list_objects(bucket)
scenario.delete_bucket(bucket)

puts('Thanks for watching!')
puts('-' * 88)
rescue Aws::Errors::ServiceError
  puts('Something went wrong with the demo!')
end

run_scenario(ScenarioGettingStarted.new(Aws::S3::Resource.new)) if $PROGRAM_NAME
== __FILE__

```

- API 세부 정보는 AWS SDK for Ruby API 참조의 다음 항목을 참조하세요.

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

Rust

SDK for Rust

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

시나리오를 실행하는 바이너리 크레이트(binary crate)용 코드입니다.

```

#![allow(clippy::result_large_err)]

//! Purpose

```

```
#!/ Shows how to use the AWS SDK for Rust to get started using
#!/ Amazon Simple Storage Service (Amazon S3). Create a bucket, move objects
    into and out of it,
#!/ and delete all resources at the end of the demo.
#!/
#!/ This example follows the steps in "Getting started with Amazon S3" in the
    Amazon S3
#!/ user guide.
#!/ - https://docs.aws.amazon.com/AmazonS3/latest/userguide/
    GetStartedWithS3.html

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::{config::Region, Client};
use s3_code_examples::error::S3ExampleError;
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), S3ExampleError> {
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));
    let region = region_provider.region().await.unwrap();
    let shared_config =
aws_config::from_env().region(region_provider).load().await;
    let client = Client::new(&shared_config);
    let bucket_name = format!("amzn-s3-demo-bucket-{}", Uuid::new_v4());
    let file_name = "s3/testfile.txt".to_string();
    let key = "test file key name".to_string();
    let target_key = "target_key".to_string();

    if let Err(e) = run_s3_operations(region, client, bucket_name, file_name,
key, target_key).await
    {
        eprintln!("{:?}", e);
    };

    Ok(())
}

async fn run_s3_operations(
    region: Region,
    client: Client,
    bucket_name: String,
    file_name: String,
    key: String,
```



```

        target_key: String,
    ) -> Result<(), S3ExampleError> {
        s3_code_examples::create_bucket(&client, &bucket_name, &region).await?;
        let run_example: Result<(), S3ExampleError> = (async {
            s3_code_examples::upload_object(&client, &bucket_name, &file_name,
            &key).await?;
            let _object = s3_code_examples::download_object(&client, &bucket_name,
            &key).await?;
            s3_code_examples::copy_object(&client, &bucket_name, &bucket_name, &key,
            &target_key)
                .await?;
            s3_code_examples::list_objects(&client, &bucket_name).await?;
            s3_code_examples::clear_bucket(&client, &bucket_name).await?;
            Ok(())
        })
        .await;
        if let Err(err) = run_example {
            eprintln!("Failed to complete getting-started example: {err:?}");
        }
        s3_code_examples::delete_bucket(&client, &bucket_name).await?;

        Ok(())
    }
}

```

시나리오에서 사용되는 일반적인 작업.

```

pub async fn create_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    region: &aws_config::Region,
) -> Result<Option<aws_sdk_s3::operation::create_bucket::CreateBucketOutput>,
S3ExampleError> {
    let constraint =
aws_sdk_s3::types::BucketLocationConstraint::from(region.to_string().as_str());
    let cfg = aws_sdk_s3::types::CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    let create = client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()

```

```

        .await;

    // BucketAlreadyExists and BucketAlreadyOwnedByYou are not problems for this
    task.
    create.map(Some).or_else(|err| {
        if err
            .as_service_error()
            .map(|se| se.is_bucket_already_exists() ||
se.is_bucket_already_owned_by_you())
            == Some(true)
        {
            Ok(None)
        } else {
            Err(S3ExampleError::from(err))
        }
    })
}

pub async fn upload_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::put_object::PutObjectOutput, S3ExampleError> {
    let body =
aws_sdk_s3::primitives::ByteStream::from_path(std::path::Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await
        .map_err(S3ExampleError::from)
}

pub async fn download_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::get_object::GetObjectOutput, S3ExampleError> {
    client
        .get_object()
        .bucket(bucket_name)

```

```

        .key(key)
        .send()
        .await
        .map_err(S3ExampleError::from)
    }

/// Copy an object from one bucket to another.
pub async fn copy_object(
    client: &aws_sdk_s3::Client,
    source_bucket: &str,
    destination_bucket: &str,
    source_object: &str,
    destination_object: &str,
) -> Result<(), S3ExampleError> {
    let source_key = format!("{source_bucket}/{source_object}");
    let response = client
        .copy_object()
        .copy_source(&source_key)
        .bucket(destination_bucket)
        .key(destination_object)
        .send()
        .await?;

    println!(
        "Copied from {source_key} to {destination_bucket}/{destination_object}
with etag {}",
        response
            .copy_object_result
            .unwrap_or_else(||
aws_sdk_s3::types::CopyObjectResult::builder().build())
            .e_tag()
            .unwrap_or("missing")
    );
    Ok(())
}

pub async fn list_objects(client: &aws_sdk_s3::Client, bucket: &str) ->
Result<(), S3ExampleError> {
    let mut response = client
        .list_objects_v2()
        .bucket(bucket.to_owned())
        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

```

```

while let Some(result) = response.next().await {
    match result {
        Ok(output) => {
            for object in output.contents() {
                println!(" - {}", object.key().unwrap_or("Unknown"));
            }
        }
        Err(err) => {
            eprintln!("{err:?}")
        }
    }
}

Ok(())
}

/// Given a bucket, remove all objects in the bucket, and then ensure no objects
/// remain in the bucket.
pub async fn clear_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<Vec<String>, S3ExampleError> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    // delete_objects no longer needs to be mutable.
    let objects_to_delete: Vec<String> = objects
        .contents()
        .iter()
        .filter_map(|obj| obj.key())
        .map(String::from)
        .collect();

    if objects_to_delete.is_empty() {
        return Ok(vec![]);
    }

    let return_keys = objects_to_delete.clone();

    delete_objects(client, bucket_name, objects_to_delete).await?;

    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    eprintln!("{objects:?}");
}

```

```

    match objects.key_count {
        Some(0) => Ok(return_keys),
        _ => Err(S3ExampleError::new(
            "There were still objects left in the bucket.",
        )),
    }
}

pub async fn delete_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<(), S3ExampleError> {
    let resp = client.delete_bucket().bucket(bucket_name).send().await;
    match resp {
        Ok(_) => Ok(()),
        Err(err) => {
            if err
                .as_service_error()
                .and_then(aws_sdk_s3::error::ProvideErrorMetadata::code)
                == Some("NoSuchBucket")
            {
                Ok(())
            } else {
                Err(S3ExampleError::from(err))
            }
        }
    }
}
}

```

- API 세부 정보는 AWS SDK for Rust API 참조의 다음 주제를 참조하세요.

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

SAP ABAP

SDK for SAP ABAP API

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

" Create an Amazon Simple Storage Service (Amazon S3) bucket. "
TRY.
    " determine our region from our session
    DATA(lv_region) = CONV /aws1/s3_bucketlocationcnstrnt( lo_session-
>get_region( ) ).
    DATA lo_constraint TYPE REF TO /aws1/cl_s3_createbucketconf.
    " When in the us-east-1 region, you must not specify a constraint
    " In all other regions, specify the region as the constraint
    IF lv_region = 'us-east-1'.
        CLEAR lo_constraint.
    ELSE.
        lo_constraint = NEW /aws1/cl_s3_createbucketconf( lv_region ).
    ENDIF.

    lo_s3->createbucket(
        iv_bucket = iv_bucket_name
        io_createbucketconfiguration = lo_constraint ).
    MESSAGE 'S3 bucket created.' TYPE 'I'.
CATCH /aws1/cx_s3_bucketalrdyexists.
    MESSAGE 'Bucket name already exists.' TYPE 'E'.
CATCH /aws1/cx_s3_bktalrdyownedbyyou.
    MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
ENDTRY.

"Upload an object to an S3 bucket."
TRY.
    "Get contents of file from application server."
    DATA lv_file_content TYPE xstring.
```

```

OPEN DATASET iv_key FOR INPUT IN BINARY MODE.
READ DATASET iv_key INTO lv_file_content.
CLOSE DATASET iv_key.

lo_s3->putobject(
    iv_bucket = iv_bucket_name
    iv_key = iv_key
    iv_body = lv_file_content ).
MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" Get an object from a bucket. "
TRY.
    DATA(lo_result) = lo_s3->getobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key ).
    DATA(lv_object_data) = lo_result->get_body( ).
    MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

" Copy an object to a subfolder in a bucket. "
TRY.
    lo_s3->copyobject(
        iv_bucket = iv_bucket_name
        iv_key = |{ iv_copy_to_folder }/{ iv_key }|
        iv_copysource = |{ iv_bucket_name }/{ iv_key }| ).
    MESSAGE 'Object copied to a subfolder.' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

" List objects in the bucket. "
TRY.
    DATA(lo_list) = lo_s3->listobjects(
        iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.

```

```

    CATCH /aws1/cx_s3_nosuchbucket.
      MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
  DATA text TYPE string VALUE 'Object List - '.
  DATA lv_object_key TYPE /aws1/s3_objectkey.
  LOOP AT lo_list->get_contents( ) INTO DATA(lo_object).
    lv_object_key = lo_object->get_key( ).
    CONCATENATE lv_object_key ', ' INTO text.
  ENDLOOP.
  MESSAGE text TYPE 'I'.

" Delete the objects in a bucket. "
TRY.
  lo_s3->deleteobject(
    iv_bucket = iv_bucket_name
    iv_key = iv_key ).
  lo_s3->deleteobject(
    iv_bucket = iv_bucket_name
    iv_key = |{ iv_copy_to_folder }/{ iv_key }| ).
  MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" Delete the bucket. "
TRY.
  lo_s3->deletebucket(
    iv_bucket = iv_bucket_name ).
  MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 다음 주제를 참조하세요.

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)

- [ListObjectsV2](#)
- [PutObject](#)

Swift

SDK for Swift

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSS3

import Foundation
import AWSS3
import Smithy
import ClientRuntime

/// A class containing all the code that interacts with the AWS SDK for Swift.
public class ServiceHandler {
    let configuration: S3Client.S3ClientConfiguration
    let client: S3Client

    enum HandlerError: Error {
        case getObjectBody(String)
        case readGetObjectBody(String)
        case missingContents(String)
    }

    /// Initialize and return a new ``ServiceHandler`` object, which is used to
    drive the AWS calls
    /// used for the example.
    ///
    /// - Returns: A new ``ServiceHandler`` object, ready to be called to
    execute AWS operations.
    public init() async throws {
        do {
            configuration = try await S3Client.S3ClientConfiguration()
```

```

        // configuration.region = "us-east-2" // Uncomment this to set the
region programmatically.
        client = S3Client(config: configuration)
    }
    catch {
        print("ERROR: ", dump(error, name: "Initializing S3 client"))
        throw error
    }
}

/// Create a new user given the specified name.
///
/// - Parameters:
///   - name: Name of the bucket to create.
/// Throws an exception if an error occurs.
public func createBucket(name: String) async throws {
    var input = CreateBucketInput(
        bucket: name
    )

    // For regions other than "us-east-1", you must set the
locationConstraint in the createBucketConfiguration.
    // For more information, see LocationConstraint in the S3 API guide.
    // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_CreateBucket.html#API_CreateBucket_RequestBody
    if let region = configuration.region {
        if region != "us-east-1" {
            input.createBucketConfiguration =
S3ClientTypes.CreateBucketConfiguration(locationConstraint:
S3ClientTypes.BucketLocationConstraint(rawValue: region))
        }
    }

    do {
        _ = try await client.createBucket(input: input)
    }
    catch let error as BucketAlreadyOwnedByYou {
        print("The bucket '\(name)' already exists and is owned by you. You
may wish to ignore this exception.")
        throw error
    }
    catch {
        print("ERROR: ", dump(error, name: "Creating a bucket"))
    }
}

```

```
        throw error
    }
}

/// Delete a bucket.
/// - Parameter name: Name of the bucket to delete.
public func deleteBucket(name: String) async throws {
    let input = DeleteBucketInput(
        bucket: name
    )
    do {
        _ = try await client.deleteBucket(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Deleting a bucket"))
        throw error
    }
}

/// Upload a file from local storage to the bucket.
/// - Parameters:
///   - bucket: Name of the bucket to upload the file to.
///   - key: Name of the file to create.
///   - file: Path name of the file to upload.
public func uploadFile(bucket: String, key: String, file: String) async
throws {
    let fileUrl = URL(fileURLWithPath: file)
    do {
        let fileData = try Data(contentsOf: fileUrl)
        let dataStream = ByteStream.data(fileData)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )

        _ = try await client.putObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Putting an object."))
        throw error
    }
}
```

```
/// Create a file in the specified bucket with the given name. The new
/// file's contents are uploaded from a `Data` object.
///
/// - Parameters:
///   - bucket: Name of the bucket to create a file in.
///   - key: Name of the file to create.
///   - data: A `Data` object to write into the new file.
public func createFile(bucket: String, key: String, withData data: Data)
async throws {
    let dataStream = ByteStream.data(data)

    let input = PutObjectInput(
        body: dataStream,
        bucket: bucket,
        key: key
    )

    do {
        _ = try await client.putObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Putting an object. "))
        throw error
    }
}

/// Download the named file to the given directory on the local device.
///
/// - Parameters:
///   - bucket: Name of the bucket that contains the file to be copied.
///   - key: The name of the file to copy from the bucket.
///   - to: The path of the directory on the local device where you want to
///         download the file.
public func downloadFile(bucket: String, key: String, to: String) async
throws {
    let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)

    let input = GetObjectInput(
        bucket: bucket,
        key: key
    )

    do {
        let output = try await client.getObject(input: input)
```

```

        guard let body = output.body else {
            throw HandlerError.getObjectBody("GetObjectInput missing body.")
        }

        guard let data = try await body.readData() else {
            throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
        }

        try data.write(to: fileUrl)
    }
    catch {
        print("ERROR: ", dump(error, name: "Downloading a file. "))
        throw error
    }
}

/// Read the specified file from the given S3 bucket into a Swift
/// `Data` object.
///
/// - Parameters:
///   - bucket: Name of the bucket containing the file to read.
///   - key: Name of the file within the bucket to read.
///
/// - Returns: A `Data` object containing the complete file data.
public func readFile(bucket: String, key: String) async throws -> Data {
    let input = GetObjectInput(
        bucket: bucket,
        key: key
    )
    do {
        let output = try await client.getObject(input: input)

        guard let body = output.body else {
            throw HandlerError.getObjectBody("GetObjectInput missing body.")
        }

        guard let data = try await body.readData() else {
            throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
        }

        return data
    }
}

```

```

    }
    catch {
        print("ERROR: ", dump(error, name: "Reading a file. "))
        throw error
    }
}

/// Copy a file from one bucket to another.
///
/// - Parameters:
///   - sourceBucket: Name of the bucket containing the source file.
///   - name: Name of the source file.
///   - destBucket: Name of the bucket to copy the file into.
public func copyFile(from sourceBucket: String, name: String, to destBucket:
String) async throws {
    let srcUrl = ("\"(sourceBucket)/
\"(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

    let input = CopyObjectInput(
        bucket: destBucket,
        copySource: srcUrl,
        key: name
    )
    do {
        _ = try await client.copyObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Copying an object. "))
        throw error
    }
}

/// Deletes the specified file from Amazon S3.
///
/// - Parameters:
///   - bucket: Name of the bucket containing the file to delete.
///   - key: Name of the file to delete.
///
public func deleteFile(bucket: String, key: String) async throws {
    let input = DeleteObjectInput(
        bucket: bucket,
        key: key
    )
}

```

```
do {
    _ = try await client.deleteObject(input: input)
}
catch {
    print("ERROR: ", dump(error, name: "Deleting a file. "))
    throw error
}
}

/// Returns an array of strings, each naming one file in the
/// specified bucket.
///
/// - Parameter bucket: Name of the bucket to get a file listing for.
/// - Returns: An array of `String` objects, each giving the name of
///            one file contained in the bucket.
public func listBucketFiles(bucket: String) async throws -> [String] {
    do {
        let input = ListObjectsV2Input(
            bucket: bucket
        )

        // Use "Paginated" to get all the objects.
        // This lets the SDK handle the 'continuationToken' in
        "ListObjectsV2Output".
        let output = client.listObjectsV2Paginated(input: input)
        var names: [String] = []

        for try await page in output {
            guard let objList = page.contents else {
                print("ERROR: listObjectsV2Paginated returned nil contents.")
                continue
            }

            for obj in objList {
                if let objName = obj.key {
                    names.append(objName)
                }
            }
        }

        return names
    }
}
```

```
        catch {
            print("ERROR: ", dump(error, name: "Listing objects. "))
            throw error
        }
    }
}
```

```
import AWSS3

import Foundation
import ServiceHandler
import ArgumentParser

/// The command-line arguments and options available for this
/// example command.
struct ExampleCommand: ParsableCommand {
    @Argument(help: "Name of the S3 bucket to create")
    var bucketName: String

    @Argument(help: "Pathname of the file to upload to the S3 bucket")
    var uploadSource: String

    @Argument(help: "The name (key) to give the file in the S3 bucket")
    var objName: String

    @Argument(help: "S3 bucket to copy the object to")
    var destBucket: String

    @Argument(help: "Directory where you want to download the file from the S3
bucket")
    var downloadDir: String

    static var configuration = CommandConfiguration(
        commandName: "s3-basics",
        abstract: "Demonstrates a series of basic AWS S3 functions.",
        discussion: """
        Performs the following Amazon S3 commands:

        * `CreateBucket`
        * `PutObject`
        * `GetObject`
        * `CopyObject`
    """
    )
}
```



```

        * `ListObjects`
        * `DeleteObjects`
        * `DeleteBucket`
        ""
    )

/// Called by ``main()`` to do the actual running of the AWS
/// example.
func runAsync() async throws {
    let serviceHandler = try await ServiceHandler()

    // 1. Create the bucket.
    print("Creating the bucket \(bucketName)...")
    try await serviceHandler.createBucket(name: bucketName)

    // 2. Upload a file to the bucket.
    print("Uploading the file \(uploadSource)...")
    try await serviceHandler.uploadFile(bucket: bucketName, key: objName,
file: uploadSource)

    // 3. Download the file.
    print("Downloading the file \(objName) to \(downloadDir)...")
    try await serviceHandler.downloadFile(bucket: bucketName, key: objName,
to: downloadDir)

    // 4. Copy the file to another bucket.
    print("Copying the file to the bucket \(destBucket)...")
    try await serviceHandler.copyFile(from: bucketName, name: objName, to:
destBucket)

    // 5. List the contents of the bucket.

    print("Getting a list of the files in the bucket \(bucketName)")
    let fileList = try await serviceHandler.listBucketFiles(bucket:
bucketName)
    let numFiles = fileList.count
    if numFiles != 0 {
        print("\(numFiles) file\(((numFiles > 1) ? "s" : "") in bucket
\(bucketName):")
        for name in fileList {
            print("  \(name)")
        }
    } else {
        print("No files found in bucket \(bucketName)")
    }
}

```

```
    }

    // 6. Delete the objects from the bucket.

    print("Deleting the file \(objName) from the bucket \(bucketName)...")
    try await serviceHandler.deleteFile(bucket: bucketName, key: objName)
    print("Deleting the file \(objName) from the bucket \(destBucket)...")
    try await serviceHandler.deleteFile(bucket: destBucket, key: objName)

    // 7. Delete the bucket.
    print("Deleting the bucket \(bucketName)...")
    try await serviceHandler.deleteBucket(name: bucketName)

    print("Done.")
}
}

//
// Main program entry point.
//
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- API 세부 정보는 AWS SDK for Swift API 참조의 다음 주제를 참조하세요.

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)

- [ListObjectsV2](#)
- [PutObject](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용한 Amazon S3에 대한 작업

다음 코드 예제에서는 AWS SDK를 통해 개별 Amazon S3 작업을 수행하는 방법을 보여줍니다. 각 예제에는 GitHub에 대한 링크가 포함되어 있습니다. 여기에서 코드 설정 및 실행에 대한 지침을 찾을 수 있습니다.

이들 발췌문은 Amazon S3 API를 호출하며, 컨텍스트에서 실행되어야 하는 더 큰 프로그램에서 발췌한 코드입니다. [AWS SDK를 사용한 Amazon S3에 대한 시나리오](#)에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 [Amazon Simple Storage Service API 참조](#)를 참조하세요.

예제

- [AWS SDK 또는 CLI와 함께 AbortMultipartUpload 사용](#)
- [AWS SDK 또는 CLI와 함께 CompleteMultipartUpload 사용](#)
- [AWS SDK 또는 CLI와 함께 CopyObject 사용](#)
- [AWS SDK 또는 CLI와 함께 CreateBucket 사용](#)
- [AWS SDK와 함께 CreateMultiRegionAccessPoint 사용](#)
- [AWS SDK 또는 CLI와 함께 CreateMultipartUpload 사용](#)
- [AWS SDK와 함께 CreatePresignedPost 사용](#)
- [AWS SDK 또는 CLI와 함께 DeleteBucket 사용](#)
- [CLI로 DeleteBucketAnalyticsConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 DeleteBucketCors 사용](#)
- [CLI로 DeleteBucketEncryption 사용](#)
- [CLI로 DeleteBucketInventoryConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 DeleteBucketLifecycle 사용](#)
- [CLI로 DeleteBucketMetricsConfiguration 사용](#)

- [AWS SDK 또는 CLI와 함께 DeleteBucketPolicy 사용](#)
- [CLI로 DeleteBucketReplication 사용](#)
- [CLI로 DeleteBucketTagging 사용](#)
- [AWS SDK 또는 CLI와 함께 DeleteBucketWebsite 사용](#)
- [AWS SDK 또는 CLI와 함께 DeleteObject 사용](#)
- [CLI로 DeleteObjectTagging 사용](#)
- [AWS SDK 또는 CLI와 함께 DeleteObjects 사용](#)
- [CLI로 DeletePublicAccessBlock 사용](#)
- [CLI로 GetBucketAccelerateConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketAcl 사용](#)
- [CLI로 GetBucketAnalyticsConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketCors 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketEncryption 사용](#)
- [CLI로 GetBucketInventoryConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketLifecycleConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketLocation 사용](#)
- [CLI로 GetBucketLogging 사용](#)
- [CLI로 GetBucketMetricsConfiguration 사용](#)
- [CLI로 GetBucketNotification 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketPolicy 사용](#)
- [CLI로 GetBucketPolicyStatus 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketReplication 사용](#)
- [CLI로 GetBucketRequestPayment 사용](#)
- [CLI로 GetBucketTagging 사용](#)
- [CLI로 GetBucketVersioning 사용](#)
- [AWS SDK 또는 CLI와 함께 GetBucketWebsite 사용](#)
- [AWS SDK 또는 CLI와 함께 GetObject 사용](#)
- [AWS SDK 또는 CLI와 함께 GetObjectAcl 사용](#)
- [AWS SDK 또는 CLI와 함께 GetObjectAttributes 사용](#)
- [AWS SDK 또는 CLI와 함께 GetObjectLegalHold 사용](#)

- [AWS SDK 또는 CLI와 함께 GetObjectLockConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 GetObjectRetention 사용](#)
- [CLI로 GetObjectTagging 사용](#)
- [CLI로 GetPublicAccessBlock 사용](#)
- [AWS SDK 또는 CLI와 함께 HeadBucket 사용](#)
- [AWS SDK 또는 CLI와 함께 HeadObject 사용](#)
- [CLI로 ListBucketAnalyticsConfigurations 사용](#)
- [CLI로 ListBucketInventoryConfigurations 사용](#)
- [AWS SDK 또는 CLI와 함께 ListBuckets 사용](#)
- [AWS SDK 또는 CLI와 함께 ListMultipartUploads 사용](#)
- [AWS SDK 또는 CLI와 함께 ListObjectVersions 사용](#)
- [CLI로 ListObjects 사용](#)
- [AWS SDK 또는 CLI와 함께 ListObjectsV2 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketAccelerateConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketAcl 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketCors 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketEncryption 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketLifecycleConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketLogging 사용](#)
- [CLI로 PutBucketNotification 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketNotificationConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketPolicy 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketReplication 사용](#)
- [CLI로 PutBucketRequestPayment 사용](#)
- [CLI로 PutBucketTagging 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketVersioning 사용](#)
- [AWS SDK 또는 CLI와 함께 PutBucketWebsite 사용](#)
- [AWS SDK 또는 CLI와 함께 PutObject 사용](#)
- [AWS SDK 또는 CLI와 함께 PutObjectAcl 사용](#)
- [AWS SDK 또는 CLI와 함께 PutObjectLegalHold 사용](#)

- [AWS SDK 또는 CLI와 함께 PutObjectLockConfiguration 사용](#)
- [AWS SDK 또는 CLI와 함께 PutObjectRetention 사용](#)
- [AWS SDK 또는 CLI와 함께 RestoreObject 사용](#)
- [AWS SDK 또는 CLI와 함께 SelectObjectContent 사용](#)
- [AWS SDK 또는 CLI와 함께 UploadPart 사용](#)
- [AWS SDK 또는 CLI와 함께 UploadPartCopy 사용](#)

AWS SDK 또는 CLI와 함께 **AbortMultipartUpload** 사용

다음 코드 예시는 AbortMultipartUpload의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [미완료 멀티파트 업로드 삭제](#)
- [Amazon S3 객체 무결성 작업](#)

C++

SDK for C++

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Abort a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                       const Aws::String &key,
```

```

        const Aws::String &uploadID,
        const Aws::S3::S3Client &client) {
    Aws::S3::Model::AbortMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);

    Aws::S3::Model::AbortMultipartUploadOutcome outcome =
        client.AbortMultipartUpload(request);

    if (outcome.IsSuccess()) {
        std::cout << "Multipart upload aborted." << std::endl;
    } else {
        std::cerr << "Error aborting multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [AbortMultipartUpload](#)를 참조하세요.

CLI

AWS CLI

지정된 멀티파트 업로드 중단

다음 `abort-multipart-upload` 명령은 `amzn-s3-demo-bucket` 버킷의 `multipart/01` 키에 대한 멀티파트 업로드를 중단합니다.

```

aws s3api abort-multipart-upload \
  --bucket amzn-s3-demo-bucket \
  --key multipart/01 \
  --upload-
id dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZLjF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P

```

이 명령에 필요한 업로드 ID는 `create-multipart-upload`로 출력되며 `list-multipart-uploads`를 사용하여 가져올 수도 있습니다.

- API 세부 정보는 AWS CLI 명령 참조의 [AbortMultipartUpload](#)를 참조하세요.

Java

SDK for Java 2.x

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.utils.builder.SdkBuilder;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.time.Duration;
import java.time.Instant;
import java.util.ArrayList;
```



```
import java.util.Collection;
import java.util.List;
import java.util.Objects;
import java.util.UUID;

import static software.amazon.awssdk.transfer.s3.SizeConstant.KB;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class AbortMultipartUploadExamples {
    static final S3Client s3Client = S3Client.create();
    static final String classPathFilePath = "/multipartUploadFiles/s3-userguide.pdf";
    static final String filePath = getFullFilePath(classPathFilePath);
    private static final Logger logger =
        LoggerFactory.getLogger(AbortMultipartUploadExamples.class);
    private static final String accountId = getAccountId();

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.

        String key = UUID.randomUUID().toString();

        createBucket(bucketName);
        try {
            initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread");
            abortIncompleteMultipartUploadsFromList(bucketName);
            abortMultipartUploadUsingUploadId(bucketName, key);
            abortMultipartUploadsUsingLifecycleConfig(bucketName);
        } catch (S3Exception e) {
            logger.error(e.getMessage());
        } finally {
            deleteResources(bucketName, key);
        }
    }
}
```

```

/**
 * Aborts all incomplete multipart uploads from the specified S3 bucket.
 *
 * @param bucketName the name of the S3 bucket
 */
public static void abortIncompleteMultipartUploadsFromList(String bucketName)
{
    ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
        .bucket(bucketName)
        .build();

    ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
    List<MultipartUpload> uploads = response.uploads();

    AbortMultipartUploadRequest abortMultipartUploadRequest;
    for (MultipartUpload upload : uploads) {
        abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(upload.key())
            .expectedBucketOwner(accountId)
            .uploadId(upload.uploadId())
            .build();

        AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
        if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
        }
    }
}

/**
 * Aborts incomplete multipart uploads older than the specified point in
time.
 *
 * @param bucketName the name of the S3 bucket
 * @param pointInTime the cutoff time; uploads initiated before this are
aborted
 */
static void abortIncompleteMultipartUploadsOlderThan(String bucketName,
Instant pointInTime) {

```

```

        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
    .bucket(bucketName)
    .build();

        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            logger.info("Found multipartUpload with upload ID [{}], initiated
[{}]", upload.uploadId(), upload.initiated());
            if (upload.initiated().isBefore(pointInTime)) {
                abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(upload.key())
                    .expectedBucketOwner(accountId)
                    .uploadId(upload.uploadId())
                    .build();

                AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
                if
(abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                    logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
                }
            }
        }
    }

    /**
     * Aborts a multipart upload using the upload ID.
     *
     * @param bucketName the name of the S3 bucket
     * @param key         the object key
     */
    static void abortMultipartUploadUsingUploadId(String bucketName, String key)
    {
        String uploadId = startUploadReturningUploadId(bucketName, key);
        AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -
> b

```

```

        .uploadId(uploadId)
        .bucket(bucketName)
        .key(key));

    if (response.sdkHttpResponse().isSuccessful()) {
        logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
uploadId, bucketName);
    }
}

/**
 * Configures a lifecycle rule to abort incomplete multipart uploads after 7
days.
 *
 * @param bucketName the name of the S3 bucket
 */
static void abortMultipartUploadsUsingLifecycleConfig(String bucketName) {
    Collection<LifecycleRule> lifeCycleRules =
List.of(LifecycleRule.builder()
        .abortIncompleteMultipartUpload(b -> b.
            daysAfterInitiation(7))
        .status("Enabled")
        .filter(SdkBuilder::build) // Filter element is required.
        .build());

    PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
        .bucket(bucketName)
        .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

    if (response.sdkHttpResponse().isSuccessful()) {
        logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
    } else {
        logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
    }
}

/*****
Multipart upload methods
*****/

```

```

    static void initiateAndInterruptMultiPartUpload(String bucketName, String
key, String threadName) {
        Runnable upload = () -> {
            try {
                doMultipartUpload(bucketName, key);
            } catch (SdkException e) {
                logger.error(e.getMessage());
            }
        };
        Thread uploadThread = new Thread(upload, threadName);
        uploadThread.start();
        try {
            Thread.sleep(Duration.ofSeconds(1).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
        uploadThread.interrupt();
    }

    static Instant initiateAndInterruptTwoUploads(String bucketName, String key)
{
        initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread1");
        try {
            Thread.sleep(Duration.ofSeconds(5).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
        Instant secondUploadInstant = Instant.now();
        initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread2");
        return secondUploadInstant;
    }

    static void doMultipartUpload(String bucketName, String key) {
        String uploadId = step1CreateMultipartUpload(bucketName, key);
        List<CompletedPart> completedParts = step2UploadParts(bucketName, key,
uploadId);
        step3CompleteMultipartUpload(bucketName, key, uploadId, completedParts);
    }

    static String step1CreateMultipartUpload(String bucketName, String key) {
        CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key));
    }

```

```
        return createMultipartUploadResponse.uploadId();
    }

    static List<CompletedPart> step2UploadParts(String bucketName, String key,
String uploadId) {
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(Long.valueOf(1024 * KB).intValue());

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
                long read = file.getChannel().read(bb);

                bb.flip();
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(partNumber)
                    .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                    uploadPartRequest,
                    RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                    .partNumber(partNumber)
                    .eTag(partResponse.eTag())
                    .build();
                completedParts.add(part);
                logger.info("Part {} upload", partNumber);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException | S3Exception e) {
            logger.error(e.getMessage());
            return null;
        }
        return completedParts;
    }
```

```

    }

    static void step3CompleteMultipartUpload(String bucketName, String key,
String uploadId, List<CompletedPart> completedParts) {
        s3Client.completeMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }

    static String startUploadReturningUploadId(String bucketName, String key) {
        String uploadId = step1CreateMultipartUpload(bucketName, key);
        doMultipartUploadWithUploadId(bucketName, key, uploadId);
        return uploadId;
    }

    static void doMultipartUploadWithUploadId(String bucketName, String key,
String uploadId) {
        new Thread(() -> {
            try {
                List<CompletedPart> completedParts = step2UploadParts(bucketName,
key, uploadId);
                step3CompleteMultipartUpload(bucketName, key, uploadId,
completedParts);
            } catch (SdkException e) {
                logger.error(e.getMessage());
            }
        }, "upload thread").start();
        try {
            Thread.sleep(Duration.ofSeconds(2L).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
            System.exit(1);
        }
    }
}

/*****
Resource handling methods
*****/

static void createBucket(String bucketName) {
    logger.info("Creating bucket: [{}]", bucketName);
}

```

```

        s3Client.createBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
            s3Waiter.waitUntilBucketExists(b -> b.bucket(bucketName));
        }
        logger.info("Bucket created.");
    }

    static void deleteResources(String bucketName, String key) {
        logger.info("Deleting resources ...");
        s3Client.deleteObject(b -> b.bucket(bucketName).key(key));
        s3Client.deleteBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
            s3Waiter.waitUntilBucketNotExists(b -> b.bucket(bucketName));
        }
        logger.info("Resources deleted.");
    }

    private static String getAccountId() {
        try (StsClient stsClient = StsClient.create()) {
            return stsClient.getCallerIdentity().account();
        }
    }

    static String getFullFilePath(String filePath) {
        URL uploadDirectoryURL =
        PerformMultiPartUpload.class.getResource(filePath);
        String fullFilePath;
        try {
            fullFilePath =
            Objects.requireNonNull(uploadDirectoryURL).toURI().getPath();
        } catch (URISyntaxException e) {
            throw new RuntimeException(e);
        }
        return fullFilePath;
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [AbortMultipartUpload](#)를 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 5일 이전에 생성된 멀티파트 업로드를 중단합니다.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

예시 2: 이 명령은 2014년 1월 2일 이전에 생성된 멀티파트 업로드를 중단합니다.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"Thursday, January 02, 2014"
```

예시 3: 이 명령은 2014년 1월 2일 10시 45분 37초 이전에 생성된 멀티파트 업로드를 중단합니다.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"2014/01/02 10:45:37"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [AbortMultipartUpload](#)를 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 5일 이전에 생성된 멀티파트 업로드를 중단합니다.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

예시 2: 이 명령은 2014년 1월 2일 이전에 생성된 멀티파트 업로드를 중단합니다.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"Thursday, January 02, 2014"
```

예시 3: 이 명령은 2014년 1월 2일 10시 45분 37초 이전에 생성된 멀티파트 업로드를 중단합니다.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"2014/01/02 10:45:37"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [AbortMultipartUpload](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **CompleteMultipartUpload** 사용

다음 코드 예시는 CompleteMultipartUpload의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [멀티파트 복사 수행](#)
- [체크섬 사용](#)
- [Amazon S3 객체 무결성 작업](#)

C++

SDK for C++

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
#!/ Complete a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param parts: A vector of CompleteParts.
    \param client: The S3 client instance used to perform the upload operation.
    \return CompleteMultipartUploadOutcome: The request outcome.
*/
Aws::S3::Model::CompleteMultipartUploadOutcome
AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,

    const Aws::String &key,

    const Aws::String &uploadID,
```

```

const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,

const Aws::S3::S3Client &client) {
    Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
    completedMultipartUpload.SetParts(parts);

    Aws::S3::Model::CompleteMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetMultipartUpload(completedMultipartUpload);

    Aws::S3::Model::CompleteMultipartUploadOutcome outcome =
        client.CompleteMultipartUpload(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error completing multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }
    return outcome;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CompleteMultipartUpload](#)를 참조하세요.

CLI

AWS CLI

다음 명령은 버킷 `amzn-s3-demo-bucket`의 키 `multipart/01`에 대한 멀티파트 업로드를 완료합니다.

```

aws s3api complete-multipart-upload --multipart-upload file://
mpustruct --bucket amzn-s3-demo-bucket --key 'multipart/01' --upload-
id dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P

```

이 명령에 필요한 업로드 ID는 `create-multipart-upload`로 출력되며 `list-multipart-uploads`를 사용하여 가져올 수도 있습니다.

위 명령의 멀티파트 업로드 옵션은 전체 파일로 리어셈블해야 하는 멀티파트 업로드 부분을 설명하는 JSON 구조를 사용합니다. 이 예시에서는 `file://` 접두사를 사용하여 `mpustruct`라는 로컬 폴더에 있는 파일에서 JSON 구조를 로드합니다.

`mpustruct`:

```
{
  "Parts": [
    {
      "ETag": "e868e0f4719e394144ef36531ee6824c",
      "PartNumber": 1
    },
    {
      "ETag": "6bb2b12753d66fe86da4998aa33fffb0",
      "PartNumber": 2
    },
    {
      "ETag": "d0a0112e841abec9c9ec83406f0159c8",
      "PartNumber": 3
    }
  ]
}
```

업로드되는 각 파트의 ETag 값은 `upload-part` 명령을 사용하여 파트를 업로드할 때마다 출력되며, `list-parts`를 직접 호출하거나 각 파트의 MD5 체크섬으로 계산할 수도 있습니다.

출력:

```
{
  "ETag": "\"3944a9f7a4faab7f78788ff6210f63f0-3\"",
  "Bucket": "amzn-s3-demo-bucket",
  "Location": "https://amzn-s3-demo-bucket.s3.amazonaws.com/multipart%2F01",
  "Key": "multipart/01"
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [CompleteMultipartUpload](#)를 참조하세요.

Rust

SDK for Rust

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
    .set_parts(Some(upload_parts))
    .build();

let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await?;
```

```
// Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
// upload the file.
let multipart_upload_res: CreateMultipartUploadOutput = client
    .create_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .send()
    .await?;

let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
    "Missing upload_id after CreateMultipartUpload",
))?;
```

```
let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();
```

```

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))
        .build()
        .await
        .unwrap();

    // Chunk index needs to start at 0, but part numbers start at 1.
    let part_number = (chunk_index as i32) + 1;
    let upload_part_res = client
        .upload_part()
        .key(&key)
        .bucket(&bucket_name)
        .upload_id(upload_id)
        .body(stream)
        .part_number(part_number)
        .send()
        .await?;

    upload_parts.push(
        CompletedPart::builder()
            .e_tag(upload_part_res.e_tag.unwrap_or_default())
            .part_number(part_number)
            .build(),
    );
}

```

- API 세부 정보는 AWS SDK for Rust API 참조의 [CompleteMultipartUpload](#)를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **CopyObject** 사용

다음 코드 예시는 CopyObject의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [기본 사항 알아보기](#)
- [암호화 시작하기](#)
- [객체 스토리지 시작하기](#)
- [조건부 요청 수행](#)

.NET

SDK for .NET(v4)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Copies an object in an Amazon S3 bucket to a folder within the
/// same bucket.
/// </summary>
/// <param name="bucketName">The name of the Amazon S3 bucket where the
/// object to copy is located.</param>
/// <param name="objectName">The object to be copied.</param>
/// <param name="folderName">The folder to which the object will
/// be copied.</param>
/// <returns>A boolean value that indicates the success or failure of
/// the copy operation.</returns>
public async Task<bool> CopyObjectInBucketAsync(
    string bucketName,
    string objectName,
    string folderName)
{
    try
```

```

    {
        var request = new CopyObjectRequest
        {
            SourceBucket = bucketName,
            SourceKey = objectName,
            DestinationBucket = bucketName,
            DestinationKey = $"{folderName}\\{objectName}",
        };
        var response = await _amazonS3.CopyObjectAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error copying object: '{ex.Message}'");
        return false;
    }
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [CopyObject](#)를 참조하세요.

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

조건부 요청을 사용하여 객체를 복사합니다.

```

/// <summary>
/// Copies an object from one Amazon S3 bucket to another with a conditional
request.
/// </summary>
/// <param name="sourceKey">The key of the source object to copy.</param>
/// <param name="destKey">The key of the destination object.</param>
/// <param name="sourceBucket">The source bucket of the object.</param>
/// <param name="destBucket">The destination bucket of the object.</param>
/// <param name="conditionType">The type of condition to apply, e.g.
'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
'CopySourceIfUnmodifiedSince'.</param>

```



```

    /// <param name="conditionDateValue">The value to use for the condition for
    dates.</param>
    /// <param name="etagConditionalValue">The value to use for the condition for
    etags.</param>
    /// <returns>True if the conditional copy is successful, False otherwise.</
returns>
    public async Task<bool> CopyObjectConditional(string sourceKey, string
destKey, string sourceBucket, string destBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null,
string? etagConditionalValue = null)
    {
        try
        {
            var copyObjectRequest = new CopyObjectRequest
            {
                DestinationBucket = destBucket,
                DestinationKey = destKey,
                SourceBucket = sourceBucket,
                SourceKey = sourceKey
            };

            switch (conditionType)
            {
                case S3ConditionType.IfMatch:
                    copyObjectRequest.ETagToMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfNoneMatch:
                    copyObjectRequest.ETagToNotMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfModifiedSince:
                    copyObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
                case S3ConditionType.IfUnmodifiedSince:
                    copyObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
                default:
                    throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
            }

            await _amazonS3.CopyObjectAsync(copyObjectRequest);

```

```

        _logger.LogInformation($"Conditional copy successful for key
{destKey} in bucket {destBucket}.");
        return true;
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "PreconditionFailed")
        {
            _logger.LogError("Conditional copy failed: Precondition failed");
        }
        else if (e.ErrorCode == "304")
        {
            _logger.LogError("Conditional copy failed: Object not modified");
        }
        else
        {
            _logger.LogError($"Unexpected error: {e.ErrorCode}");
            throw;
        }
        return false;
    }
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [CopyObject](#)를 참조하세요.

Bash

AWS CLIBash 스크립트와 함께 사용

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {

```

```

    printf "%s\n" "$*" 1>&2
}

#####
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#     $1 - The name of the bucket to copy the file from and to.
#     $2 - The key of the source file to copy.
#     $3 - The key of the destination file.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function copy_item_in_bucket() {
    local bucket_name=$1
    local source_key=$2
    local destination_key=$3
    local response

    response=$(aws s3api copy-object \
        --bucket "$bucket_name" \
        --copy-source "$bucket_name/$source_key" \
        --key "$destination_key")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api copy-object operation failed.\n$response"
        return 1
    fi
}

```

- API 세부 정보는 AWS CLI 명령 참조의 [CopyObject](#)를 참조하십시오.

C++

SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::S3::copyObject(const Aws::String &objectKey, const Aws::String
    &fromBucket, const Aws::String &toBucket,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CopyObjectRequest request;

    request.WithCopySource(fromBucket + "/" + objectKey)
        .WithKey(objectKey)
        .WithBucket(toBucket);

    Aws::S3::Model::CopyObjectOutcome outcome = client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: copyObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;

        } else {
            std::cout << "Successfully copied " << objectKey << " from " <<
fromBucket <<
                " to " << toBucket << "." << std::endl;
        }

        return outcome.IsSuccess();
    }
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CopyObject](#)를 참조하세요.

CLI

AWS CLI

다음 명령은 bucket-1에서 bucket-2로 객체를 복사합니다.

```
aws s3api copy-object --copy-source bucket-1/test.txt --key test.txt --  
bucket bucket-2
```

출력:

```
{  
  "CopyObjectResult": {  
    "LastModified": "2015-11-10T01:07:25.000Z",  
    "ETag": "\"589c8b79c230a6ecd5a7e1d040a9a030\""  
  },  
  "VersionId": "YdnYvTCVDqRRFA.NFJjy36p0hxifM1kA"  
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [CopyObject](#)를 참조하세요.

Go

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import (  
  "bytes"  
  "context"  
  "errors"  
  "fmt"  
  "io"  
  "log"  
  "os"  
  "time"
```

```

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(ctx context.Context, sourceBucket string,
    destinationBucket string, objectKey string) error {
    _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
        Bucket:      aws.String(destinationBucket),
        CopySource:  aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
        Key:         aws.String(objectKey),
    })
    if err != nil {
        var notActive *types.ObjectNotInActiveTierError
        if errors.As(err, &notActive) {
            log.Printf("Couldn't copy object %s from %s because the object isn't in the
            active tier.\n",
                objectKey, sourceBucket)
            err = notActive
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(destinationBucket), Key:
            aws.String(objectKey)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
        }
    }
}

```

```
    return err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [CopyObject](#)를 참조하십시오.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

[S3Client](#)를 사용하여 객체를 복사합니다.

```
/**
 * Asynchronously copies an object from one S3 bucket to another.
 *
 * @param fromBucket the name of the source S3 bucket
 * @param objectKey the key (name) of the object to be copied
 * @param toBucket the name of the destination S3 bucket
 * @return a {@link CompletableFuture} that completes with the copy result as
 * a {@link String}
 * @throws RuntimeException if the URL could not be encoded or an S3
 * exception occurred during the copy
 */
public CompletableFuture<String> copyBucketObjectAsync(String fromBucket,
String objectKey, String toBucket) {
    CopyObjectRequest copyReq = CopyObjectRequest.builder()
        .sourceBucket(fromBucket)
        .sourceKey(objectKey)
        .destinationBucket(toBucket)
        .destinationKey(objectKey)
        .build();

    CompletableFuture<CopyObjectResponse> response =
        getAsyncClient().copyObject(copyReq);
}
```

```

        response.whenComplete((copyRes, ex) -> {
            if (copyRes != null) {
                logger.info("The " + objectKey + " was copied to " + toBucket);
            } else {
                throw new RuntimeException("An S3 exception occurred during
copy", ex);
            }
        });

        return response.thenApply(CopyObjectResponse::copyObjectResult)
            .thenApply(Object::toString);
    }

```

[S3TransferManager](#)를 사용하여 한 버킷에서 다른 버킷으로 [객체를 복사](#)합니다. [파일 전체](#)를 보고 [테스트](#)합니다.

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;

public String copyObject(S3TransferManager transferManager, String
bucketName,
    String key, String destinationBucket, String destinationKey) {
    CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
        .sourceBucket(bucketName)
        .sourceKey(key)
        .destinationBucket(destinationBucket)
        .destinationKey(destinationKey)
        .build();

    CopyRequest copyRequest = CopyRequest.builder()
        .copyObjectRequest(copyObjectRequest)
        .build();

    Copy copy = transferManager.copy(copyRequest);

```



```
CompletedCopy completedCopy = copy.completionFuture().join();
return completedCopy.response().copyObjectResult().eTag();
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [CopyObject](#)를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

객체를 복사합니다.

```
import {
  S3Client,
  CopyObjectCommand,
  ObjectNotInActiveTierError,
  waitUntilObjectExists,
} from "@aws-sdk/client-s3";

/**
 * Copy an S3 object from one bucket to another.
 *
 * @param {{
 *   sourceBucket: string,
 *   sourceKey: string,
 *   destinationBucket: string,
 *   destinationKey: string }} config
 */
export const main = async ({
  sourceBucket,
  sourceKey,
  destinationBucket,
  destinationKey,
}) => {
```

```

const client = new S3Client({});

try {
  await client.send(
    new CopyObjectCommand({
      CopySource: `${sourceBucket}/${sourceKey}`,
      Bucket: destinationBucket,
      Key: destinationKey,
    }),
  );
  await waitUntilObjectExists(
    { client },
    { Bucket: destinationBucket, Key: destinationKey },
  );
  console.log(
    `Successfully copied ${sourceBucket}/${sourceKey} to ${destinationBucket}/${destinationKey}`,
  );
} catch (caught) {
  if (caught instanceof ObjectNotInActiveTierError) {
    console.error(
      `Could not copy ${sourceKey} from ${sourceBucket}. Object is not in the active tier.`,
    );
  } else {
    throw caught;
  }
}
};

```

객체의 ETag가 제공된 값과 일치하지 않을 경우에만 조건부로 객체를 복사합니다.

```

import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
'object_name.json'

```

```
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
  destinationBucketName: string, eTag: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
  eTag,
}) => {
  const client = new S3Client({});
  const name = data.name;
  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: `${sourceBucketName}/${sourceKeyName}`,
        Bucket: destinationBucketName,
        Key: `${name}${sourceKeyName}`,
        CopySourceIfMatch: eTag,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
        "${sourceBucketName}". No such key exists.`
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Unable to copy object "${sourceKeyName}" to bucket
        "${sourceBucketName}": ${caught.name}: ${caught.message}`
      );
    } else {
      throw caught;
    }
  }
};
```

```
// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

객체의 ETag가 제공된 값과 일치하지 않을 경우에만 조건부로 객체를 복사합니다.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
  destinationBucketName: string, eTag: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
  eTag,
}) => {
  const client = new S3Client({});
  const name = data.name;

  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: `${sourceBucketName}/${sourceKeyName}`,
        Bucket: destinationBucketName,
        Key: `${name}${sourceKeyName}`,
        CopySourceIfNoneMatch: eTag,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
        "${sourceBucketName}". No such key exists.`
      );
    }
  }
}
```

```
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Unable to copy object "${sourceKeyName}" to bucket
"${sourceBucketName}": ${caught.name}: ${caught.message}`,
    );
  } else {
    throw caught;
  }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
};
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
```

```
const { errors, results } = loadArgs();
if (!errors) {
  main(results.values);
} else {
  console.error(errors.join("\n"));
}
}
```

지정된 기간에 생성되거나 수정된 객체만 조건부로 복사합니다.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
  destinationBucketName: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
}) => {
  const date = new Date();
  date.setDate(date.getDate() - 1);

  const name = data.name;
  const client = new S3Client({});
  const copySource = `${sourceBucketName}/${sourceKeyName}`;
  const copiedKey = name + sourceKeyName;
```

```
try {
  const response = await client.send(
    new CopyObjectCommand({
      CopySource: copySource,
      Bucket: destinationBucketName,
      Key: copiedKey,
      CopySourceIfModifiedSince: date,
    }),
  );
  console.log("Successfully copied object to bucket.");
} catch (caught) {
  if (caught instanceof NoSuchKey) {
    console.error(
      `Error from S3 while copying object "${sourceKeyName}" from
"${sourceBucketName}". No such key exists.`,
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while copying object from ${sourceBucketName}.
${caught.name}: ${caught.message}`,
    );
  } else {
    throw caught;
  }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
  },
```



```

    destinationBucketName: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}

```

지정된 기간에 생성되거나 수정되지 않은 객체만 조건부로 복사합니다.

```

import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
  destinationBucketName: string }}
 */
export const main = async ({
  sourceBucketName,

```

```
    sourceKeyName,
    destinationBucketName,
  }) => {
    const date = new Date();
    date.setDate(date.getDate() - 1);
    const client = new S3Client({});
    const name = data.name;
    const copiedKey = name + sourceKeyName;
    const copySource = `${sourceBucketName}/${sourceKeyName}`;

    try {
      const response = await client.send(
        new CopyObjectCommand({
          CopySource: copySource,
          Bucket: destinationBucketName,
          Key: copiedKey,
          CopySourceIfUnmodifiedSince: date,
        }),
      );
      console.log("Successfully copied object to bucket.");
    } catch (caught) {
      if (caught instanceof NoSuchKey) {
        console.error(
          `Error from S3 while copying object "${sourceKeyName}" from
"${sourceBucketName}". No such key exists.`,
        );
      } else if (caught instanceof S3ServiceException) {
        console.error(
          `Error from S3 while copying object from ${sourceBucketName}.
${caught.name}: ${caught.message}`,
        );
      } else {
        throw caught;
      }
    }
  };

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";
```

```
const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 [CopyObject](#)를 참조하세요.

Kotlin

SDK for Kotlin

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun copyBucketObject(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedOperationException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 [CopyObject](#)를 참조하십시오.

PHP

SDK for PHP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

간단하게 객체를 복사합니다.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
```

```

try {
    $folder = "copied-folder";
    $this->s3client->copyObject([
        'Bucket' => $this->bucketName,
        'CopySource' => "$this->bucketName/$fileName",
        'Key' => "$folder/$fileName-copy",
    ]);
    echo "Copied $fileName to $folder/$fileName-copy.\n";
} catch (Exception $exception) {
    echo "Failed to copy $fileName with error: " . $exception-
>getMessage();
    exit("Please fix error with object copying before continuing.");
}

```

- API 세부 정보는 AWS SDK for PHP API 참조의 [CopyObject](#)를 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 “test-files” 버킷의 “sample.txt” 객체를 동일한 버킷으로 복사하지만 새로운 키인 “sample-copy.txt”를 사용합니다.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey sample-copy.txt
```

예시 2: 이 명령은 “test-files” 버킷의 “sample.txt” 객체를 “backup-files” 버킷으로 복사하며 “sample-copy.txt” 키를 사용합니다.

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -
DestinationKey sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

예시 3: 이 명령은 “test-files” 버킷에서 “local-sample.txt”라는 이름의 로컬 파일로 “sample.txt” 객체를 다운로드합니다.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-sample.txt
```

예시 4: 단일 객체를 지정된 파일로 다운로드합니다. 다운로드한 파일은 c:\downloads\data\archive.zip에서 찾을 수 있습니다

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder c:\downloads
```

예시 5: 지정된 키 접두사와 일치하는 모든 객체를 로컬 폴더로 다운로드합니다. 상대 키 계층 구조는 전체 다운로드 위치에 하위 폴더로 보존됩니다.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:\downloads
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [CopyObject](#)를 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 “test-files” 버킷의 “sample.txt” 객체를 동일한 버킷으로 복사하지만 새로운 키인 “sample-copy.txt”를 사용합니다.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey sample-copy.txt
```

예시 2: 이 명령은 “test-files” 버킷의 “sample.txt” 객체를 “backup-files” 버킷으로 복사하며 “sample-copy.txt” 키를 사용합니다.

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -DestinationKey sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

예시 3: 이 명령은 “test-files” 버킷에서 “local-sample.txt”라는 이름의 로컬 파일로 “sample.txt” 객체를 다운로드합니다.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-sample.txt
```

예시 4: 단일 객체를 지정된 파일로 다운로드합니다. 다운로드한 파일은 c:\downloads\data\archive.zip에서 찾을 수 있습니다

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder c:\downloads
```

예시 5: 지정된 키 접두사와 일치하는 모든 객체를 로컬 폴더로 다운로드합니다. 상대 키 계층 구조는 전체 다운로드 위치에 하위 폴더로 보존됩니다.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:\downloads
```

예시 6: 대용량 파일의 성능을 개선하기 위해 멀티파트 병렬 다운로드를 사용하여 단일 객체를 로컬 파일에 다운로드합니다.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-sample.txt -UseMultipartDownload
```

예시 7: 사용자 지정 64MB 파트 크기로 RANGE 모드를 사용하여 대형 객체를 다운로드합니다. RANGE 모드는 HTTP 바이트 범위 요청을 사용하며 업로드 방식에 관계없이 모든 객체에 대해 작동합니다.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -LocalFile local-large-file.zip -UseMultipartDownload -MultipartDownloadType RANGE -PartSize 64MB
```

예시 8: 동시 파일 다운로드와 함께 멀티파트 다운로드를 사용하여 키 접두사와 일치하는 모든 객체를 로컬 폴더에 다운로드합니다.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -LocalFolder C:\local-data -UseMultipartDownload -DownloadFilesConcurrently
```

예시 9: 고대역폭 네트워크에서 처리량을 극대화하기 위해 20개 동시 연결로 멀티파트 다운로드를 사용하여 단일 객체를 다운로드합니다.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -LocalFile local-large-file.zip -UseMultipartDownload -ConcurrentServiceRequest 20
```

예시 10: 멀티파트 다운로드를 사용하여 키 접두사와 일치하는 모든 객체를 로컬 폴더에 다운로드하며 개별 파일 실패는 중단하는 대신 건너뜁니다.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -LocalFolder C:\local-data -UseMultipartDownload -FailurePolicy ContinueOnFailure
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [CopyObject](#)를 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def copy(self, dest_object):
        """
        Copies the object to another bucket.

        :param dest_object: The destination object initialized with a bucket and
        key.
                           This is a Boto3 Object resource.
        """
        try:
            dest_object.copy_from(
                CopySource={"Bucket": self.object.bucket_name, "Key":
self.object.key}
            )
            dest_object.wait_until_exists()
            logger.info(
                "Copied object from %s:%s to %s:%s.",
                self.object.bucket_name,
```



```

        self.object.key,
        dest_object.bucket_name,
        dest_object.key,
    )
except ClientError:
    logger.exception(
        "Couldn't copy object from %s/%s to %s/%s.",
        self.object.bucket_name,
        self.object.key,
        dest_object.bucket_name,
        dest_object.key,
    )
    raise

```

조건부 요청을 사용하여 객체를 복사합니다.

```

class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def copy_object_conditional(
        self,
        source_key: str,
        dest_key: str,
        source_bucket: str,
        dest_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """

```

Copies an object from one Amazon S3 bucket to another with a conditional request.

```
:param source_key: The key of the source object to copy.
:param dest_key: The key of the destination object.
:param source_bucket: The source bucket of the object.
:param dest_bucket: The destination bucket of the object.
:param condition_type: The type of condition to apply, e.g.
'CopySourceIfMatch', 'CopySourceIfNoneMatch',
'CopySourceIfModifiedSince', 'CopySourceIfUnmodifiedSince'.
:param condition_value: The value to use for the condition.
"""
try:
    self.s3.copy_object(
        Bucket=dest_bucket,
        Key=dest_key,
        CopySource={"Bucket": source_bucket, "Key": source_key},
        **{condition_type: condition_value},
    )
    print(
        f"\tConditional copy successful for key {dest_key} in bucket
{dest_bucket}."
    )
except ClientError as e:
    error_code = e.response["Error"]["Code"]
    if error_code == "PreconditionFailed":
        print("\tConditional copy failed: Precondition failed")
    elif error_code == "304": # Not modified error code.
        print("\tConditional copy failed: Object not modified")
    else:
        logger.error(f"Unexpected error: {error_code}")
        raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [CopyObject](#)를 참조하세요.

Ruby

SDK for Ruby

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

객체를 복사합니다.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
  # used as the source object for
  #                                     copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
  # target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
  # object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
  # nil.
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
    why: #{e.message}"
  end
end

# Example usage:
```

```
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
  #{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

객체를 복사하고 대상 객체에 서버 측 암호화를 추가합니다.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
  # used as the source object for
  #                                     copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the
  # target key, and encrypt it.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
  # object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
  nil.
  def copy_object(target_bucket, target_object_key, encryption)
```

```

    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
server_side_encryption: encryption)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
why: #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key,
target_encryption)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
#{target_object.bucket_name}:#{target_object.key} and "\
    "encrypted the target with #{target_object.server_side_encryption}
encryption."
end

run_demo if $PROGRAM_NAME == __FILE__

```

- API 세부 정보는 AWS SDK for Ruby API 참조의 [CopyObject](#)를 참조하십시오.

Rust

SDK for Rust

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// Copy an object from one bucket to another.
pub async fn copy_object(
    client: &aws_sdk_s3::Client,
    source_bucket: &str,
    destination_bucket: &str,
    source_object: &str,
    destination_object: &str,
) -> Result<(), S3ExampleError> {
    let source_key = format!("{source_bucket}/{source_object}");
    let response = client
        .copy_object()
        .copy_source(&source_key)
        .bucket(destination_bucket)
        .key(destination_object)
        .send()
        .await?;

    println!(
        "Copied from {source_key} to {destination_bucket}/{destination_object}
with etag {}",
        response
            .copy_object_result
            .unwrap_or_else(||
aws_sdk_s3::types::CopyObjectResult::builder().build())
            .e_tag()
            .unwrap_or("missing")
    );
    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 [CopyObject](#)을 참조하십시오.

SAP ABAP

SDK for SAP ABAP API

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.
    lo_s3->copyobject(
        iv_bucket = iv_dest_bucket
        iv_key = iv_dest_object
        iv_copysource = |{ iv_src_bucket }/{ iv_src_object }| ).
    MESSAGE 'Object copied to another bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.
```

- API 세부 정보는 AWSSDK for SAP ABAP API의 [CopyObject](#)을 참조하십시오.

Swift

SDK for Swift

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSS3

public func copyFile(from sourceBucket: String, name: String, to destBucket:
String) async throws {
```

```
let srcUrl = ("\"(sourceBucket)/\n(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)\n\nlet input = CopyObjectInput(\n    bucket: destBucket,\n    copySource: srcUrl,\n    key: name\n)\ndo {\n    _ = try await client.copyObject(input: input)\n}\ncatch {\n    print("ERROR: ", dump(error, name: "Copying an object."))\n    throw error\n}\n}
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 [CopyObject](#)를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **CreateBucket** 사용

다음 코드 예시는 CreateBucket의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [기본 사항 알아보기](#)
- [구성 관리 시작하기](#)
- [기계 학습 특성 저장소 시작하기](#)
- [객체 스토리지 시작하기](#)
- [S3를 사용하여 대규모 메시지 관리](#)
- [버전이 지정된 객체 작업](#)

.NET

SDK for .NET(v4)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Shows how to create a new Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <returns>A boolean value representing the success or failure of
/// the bucket creation process.</returns>
public async Task<bool> CreateBucketAsync(string bucketName)
{
    try
    {
        var request = new PutBucketRequest
        {
            BucketName = bucketName,
            UseClientRegion = true,
        };

        var response = await _amazonS3.PutBucketAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error creating bucket: '{ex.Message}'");
        return false;
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 [CreateBucket](#)을 참조하세요.

SDK for .NET

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

객체 잠금을 활성화한 버킷을 생성합니다.

```
/// <summary>
/// Create a new Amazon S3 bucket with object lock actions.
/// </summary>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <param name="enableObjectLock">True to enable object lock on the
bucket.</param>
/// <returns>True if successful.</returns>
public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
enableObjectLock)
{
    Console.WriteLine($"\\tCreating bucket {bucketName} with object lock
{enableObjectLock}.");
    try
    {
        var request = new PutBucketRequest
        {
            BucketName = bucketName,
            UseClientRegion = true,
            ObjectLockEnabledForBucket = enableObjectLock,
        };

        var response = await _amazonS3.PutBucketAsync(request);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error creating bucket: '{ex.Message}'");
        return false;
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 [CreateBucket](#)을 참조하세요.

Bash

AWS CLIBash 스크립트와 함께 사용

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
#####
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####
function iecho() {
    if [[ $VERBOSE == true ]]; then
        echo "$@"
    fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#     -b bucket_name  -- The name of the bucket to create.
#     -r region_code  -- The code for an AWS Region in which to
```

```

#                               create the bucket.
#
# Returns:
#     The URL of the bucket that was created.
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function create_bucket() {
    local bucket_name region_code response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function create_bucket"
        echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
        echo "  -b bucket_name    The name of the bucket. It must be globally
unique."
        echo "  [-r region_code]    The code for an AWS Region in which the bucket is
created."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "b:r:h" option; do
        case "${option}" in
            b) bucket_name="${OPTARG}" ;;
            r) region_code="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done

    if [[ -z "$bucket_name" ]]; then
        errecho "ERROR: You must provide a bucket name with the -b parameter."
        usage
        return 1
    fi
}

```

```

fi

local bucket_config_arg
# A location constraint for "us-east-1" returns an error.
if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
fi

iecho "Parameters:\n"
iecho "    Bucket name:  $bucket_name"
iecho "    Region code:  $region_code"
iecho ""

# If the bucket already exists, we don't want to try to create it.
if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
fi

# shellcheck disable=SC2086
response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

# shellcheck disable=SC2181
if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
fi
}

```

- API 세부 정보는 AWS CLI 명령 참조의 [CreateBucket](#)을 참조하세요.

C++

SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::S3::createBucket(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CreateBucketRequest request;
    request.SetBucket(bucketName);

    if (clientConfig.region != "us-east-1") {
        Aws::S3::Model::CreateBucketConfiguration createBucketConfig;
        createBucketConfig.SetLocationConstraint(
            Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                clientConfig.region));
        request.SetCreateBucketConfiguration(createBucketConfig);
    }

    Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: createBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
            std::endl;
    } else {
        std::cout << "Created bucket " << bucketName <<
            " in the specified AWS Region." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateBucket](#)을 참조하세요.

CLI

AWS CLI

예 1: 버킷을 생성하는 방법

다음 create-bucket 예시에서는 amzn-s3-demo-bucket이라는 버킷을 생성합니다.

```
aws s3api create-bucket \  
  --bucket amzn-s3-demo-bucket \  
  --region us-east-1
```

출력:

```
{  
  "Location": "/amzn-s3-demo-bucket"  
}
```

자세한 내용은 Amazon S3 사용자 안내서의 [버킷 생성](#)을 참조하세요.

예시 2: 소유자가 적용된 버킷 생성

다음 create-bucket 예시에서는 S3 객체 소유권에 대해 버킷 소유자 적용 설정을 사용하는 amzn-s3-demo-bucket이라는 버킷을 생성합니다.

```
aws s3api create-bucket \  
  --bucket amzn-s3-demo-bucket \  
  --region us-east-1 \  
  --object-ownership BucketOwnerEnforced
```

출력:

```
{  
  "Location": "/amzn-s3-demo-bucket"  
}
```

자세한 내용은 [Amazon S3 사용자 안내서](#)의 객체 소유권 제어 및 ACL 비활성화를 참조하세요.

예시 3: 'us-east-1' 리전 외부에서 버킷 생성

다음 create-bucket 예시에서는 eu-west-1 리전에서 amzn-s3-demo-bucket이라는 버킷을 생성합니다. us-east-1 외부 리전의 경우 원하는 리전에 버킷을 생성하려면 적절한 LocationConstraint를 지정해야 합니다.

```
aws s3api create-bucket \  
  --bucket amzn-s3-demo-bucket \  
  --region eu-west-1 \  
  --create-bucket-configuration LocationConstraint=eu-west-1
```

출력:

```
{  
  "Location": "http://amzn-s3-demo-bucket.s3.amazonaws.com/"  
}
```

자세한 내용은 Amazon S3 사용자 안내서의 [버킷 생성](#)을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [CreateBucket](#)을 참조하세요.

Go

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

기본 구성으로 버킷을 생성합니다.

```
import (  
  "bytes"  
  "context"  
  "errors"  
  "fmt"  
  "io"  
  "log"  
  "os"  
  "time"
```



```
"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(ctx context.Context, name string, region
string) error {
    _, err := basics.S3Client.CreateBucket(ctx, &s3.CreateBucketInput{
        Bucket: aws.String(name),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    })
    if err != nil {
        var owned *types.BucketAlreadyOwnedByYou
        var exists *types.BucketAlreadyExists
        if errors.As(err, &owned) {
            log.Printf("You already own bucket %s.\n", name)
            err = owned
        } else if errors.As(err, &exists) {
            log.Printf("Bucket %s already exists.\n", name)
            err = exists
        }
    } else {
        err = s3.NewBucketExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(name)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to exist.\n", name)
        }
    }
}
```

```
}  
}  
return err  
}
```

객체 잠금 기능이 있는 버킷을 만들고 버킷이 존재할 때까지 기다립니다.

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}  
  
// CreateBucketWithLock creates a new S3 bucket with optional object locking  
// enabled  
// and waits for the bucket to exist before returning.  
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,  
    region string, enableObjectLock bool) (string, error) {  
    input := &s3.CreateBucketInput{  
        Bucket: aws.String(bucket),  
        CreateBucketConfiguration: &types.CreateBucketConfiguration{  
            LocationConstraint: types.BucketLocationConstraint(region),  
        },  
    },
```

```

}

if enableObjectLock {
    input.ObjectLockEnabledForBucket = aws.Bool(true)
}

_, err := actor.S3Client.CreateBucket(ctx, input)
if err != nil {
    var owned *types.BucketAlreadyOwnedByYou
    var exists *types.BucketAlreadyExists
    if errors.As(err, &owned) {
        log.Printf("You already own bucket %s.\n", bucket)
        err = owned
    } else if errors.As(err, &exists) {
        log.Printf("Bucket %s already exists.\n", bucket)
        err = exists
    }
} else {
    err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
        ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
    }
}

return bucket, err
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [CreateBucket](#)을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷을 만듭니다.

```

/**
 * Creates an S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket to create
 * @return a {@link CompletableFuture} that completes when the bucket is
 * created and ready
 * @throws RuntimeException if there is a failure while creating the bucket
 */
public CompletableFuture<Void> createBucketAsync(String bucketName) {
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .build();

    CompletableFuture<CreateBucketResponse> response =
getAsyncClient().createBucket(bucketRequest);
    return response.thenCompose(resp -> {
        S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        CompletableFuture<WaiterResponse<HeadBucketResponse>>
waiterResponseFuture =
            s3Waiter.waitUntilBucketExists(bucketRequestWait);
        return waiterResponseFuture.thenAccept(waiterResponse -> {
            waiterResponse.matched().response().ifPresent(headBucketResponse
-> {
                logger.info(bucketName + " is ready");
            });
        });
    }).whenComplete((resp, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to create bucket", ex);
        }
    });
}

```

객체 잠금을 활성화한 버킷을 생성합니다.

```
// Create a new Amazon S3 bucket with object lock options.
public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
    S3Waiter s3Waiter = getClient().waiter();
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .objectLockEnabledForBucket(enableObjectLock)
        .build();

    getClient().createBucket(bucketRequest);
    HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
        .bucket(bucketName)
        .build();

    // Wait until the bucket is created and print out the response.
    s3Waiter.waitUntilBucketExists(bucketRequestWait);
    System.out.println(bucketName + " is ready");
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [CreateBucket](#)을 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷을 생성합니다.

```
import {
    BucketAlreadyExists,
    BucketAlreadyOwnedByYou,
    CreateBucketCommand,
    S3Client,
    waitUntilBucketExists,
} from "@aws-sdk/client-s3";
```

```
/**
 * Create an Amazon S3 bucket.
 * @param {{ bucketName: string }} config
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { Location } = await client.send(
      new CreateBucketCommand({
        // The name of the bucket. Bucket names are unique and have several other
        constraints.
        // See https://docs.aws.amazon.com/AmazonS3/latest/userguide/
bucketnamingrules.html
        Bucket: bucketName,
      }),
    );
    await waitUntilBucketExists({ client }, { Bucket: bucketName });
    console.log(`Bucket created with location ${Location}`);
  } catch (caught) {
    if (caught instanceof BucketAlreadyExists) {
      console.error(
        `The bucket "${bucketName}" already exists in another AWS account. Bucket
names must be globally unique.`
      );
    }
    // WARNING: If you try to create a bucket in the North Virginia region,
    // and you already own a bucket in that region with the same name, this
    // error will not be thrown. Instead, the call will return successfully
    // and the ACL on that bucket will be reset.
    else if (caught instanceof BucketAlreadyOwnedByYou) {
      console.error(
        `The bucket "${bucketName}" already exists in this AWS account.`
      );
    } else {
      throw caught;
    }
  }
};
```

- 자세한 정보는 [AWS SDK for JavaScript 개발자 안내서](#)를 참조하세요.

- API 세부 정보는 AWS SDK for JavaScript API 참조의 [CreateBucket](#)을 참조하세요.

Kotlin

SDK for Kotlin

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createNewBucket(bucketName: String) {  
    val request =  
        CreateBucketRequest {  
            bucket = bucketName  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.createBucket(request)  
        println("$bucketName is ready")  
    }  
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 [CreateBucket](#)를 참조하세요.

PHP

SDK for PHP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷을 만듭니다.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
```

```

try {
    $this->s3client->createBucket([
        'Bucket' => $this->bucketName,
        'CreateBucketConfiguration' => ['LocationConstraint' => $region],
    ]);
    echo "Created bucket named: $this->bucketName \n";
} catch (Exception $exception) {
    echo "Failed to create bucket $this->bucketName with error: " .
    $exception->getMessage();
    exit("Please fix error with bucket creation before continuing.");
}

```

- API 세부 정보는 AWS SDK for PHP API 참조의 [CreateBucket](#)을 참조하세요.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

기본 설정으로 버킷을 생성합니다.

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def create(self, region_override=None):

```



```

        """
        Create an Amazon S3 bucket in the default Region for the account or in
the
        specified Region.

        :param region_override: The Region in which to create the bucket. If this
is
                                not specified, the Region configured in your
shared
                                credentials is used.
        """
        if region_override is not None:
            region = region_override
        else:
            region = self.bucket.meta.client.meta.region_name
        try:
            self.bucket.create(CreateBucketConfiguration={"LocationConstraint":
region})

            self.bucket.wait_until_exists()
            logger.info("Created bucket '%s' in region=%s", self.bucket.name,
region)
        except ClientError as error:
            logger.exception(
                "Couldn't create bucket named '%s' in region=%s.",
                self.bucket.name,
                region,
            )
            raise error

```

수명 주기 구성으로 버전이 지정된 버킷을 생성합니다.

```

def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
noncurrent versions, which can slow down request performance.

```

Usage is shown in the `usage_demo_single_object` function at the end of this module.

```

:param bucket_name: The name of the bucket to create.
:param prefix: Identifies which objects are automatically expired under the
               configured lifecycle rules.
:return: The newly created bucket.
"""
try:
    bucket = s3.create_bucket(
        Bucket=bucket_name,
        CreateBucketConfiguration={
            "LocationConstraint": s3.meta.client.meta.region_name
        },
    )
    logger.info("Created bucket %s.", bucket.name)
except ClientError as error:
    if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
        logger.warning("Bucket %s already exists! Using it.", bucket_name)
        bucket = s3.Bucket(bucket_name)
    else:
        logger.exception("Couldn't create bucket %s.", bucket_name)
        raise

try:
    bucket.Versioning().enable()
    logger.info("Enabled versioning on bucket %s.", bucket.name)
except ClientError:
    logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
    raise

try:
    expiration = 7
    bucket.LifecycleConfiguration().put(
        LifecycleConfiguration={
            "Rules": [
                {
                    "Status": "Enabled",
                    "Prefix": prefix,
                    "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
                }
            ]
        }
    )

```

```

    }
  )
  logger.info(
    "Configured lifecycle to expire noncurrent versions after %s days "
    "on bucket %s.",
    expiration,
    bucket.name,
  )
except ClientError as error:
  logger.warning(
    "Couldn't configure lifecycle on bucket %s because %s. "
    "Continuing anyway.",
    bucket.name,
    error,
  )

return bucket

```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [CreateBucket](#)를 참조하세요.

Ruby

SDK for Ruby

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
  # This is a client-side object until
  #                               create is called.

```

```

def initialize(bucket)
  @bucket = bucket
end

# Creates an Amazon S3 bucket in the specified AWS Region.
#
# @param region [String] The Region where the bucket is created.
# @return [Boolean] True when the bucket is created; otherwise, false.
def create?(region)
  @bucket.create(create_bucket_configuration: { location_constraint: region })
  true
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create bucket. Here's why: #{e.message}"
  false
end

# Gets the Region where the bucket is located.
#
# @return [String] The location of the bucket.
def location
  if @bucket.nil?
    'None. You must create a bucket before you can get its location!'
  else
    @bucket.client.get_bucket_location(bucket:
@bucket.name).location_constraint
  end
  rescue Aws::Errors::ServiceError => e
    "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("amzn-s3-demo-bucket-
#{Random.uuid}"))
  return unless wrapper.create?(region)

  puts "Created bucket #{wrapper.bucket.name}."
  puts "Your bucket's region is: #{wrapper.location}"
end

run_demo if $PROGRAM_NAME == __FILE__

```

- API 세부 정보는 AWS SDK for Ruby API 참조의 [CreateBucket](#)을 참조하세요.

Rust

SDK for Rust

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn create_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    region: &aws_config::Region,
) -> Result<Option<aws_sdk_s3::operation::create_bucket::CreateBucketOutput>,
S3ExampleError> {
    let constraint =
aws_sdk_s3::types::BucketLocationConstraint::from(region.to_string().as_str());
    let cfg = aws_sdk_s3::types::CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    let create = client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await;

    // BucketAlreadyExists and BucketAlreadyOwnedByYou are not problems for this
    task.
    create.map(Some).or_else(|err| {
        if err
            .as_service_error()
            .map(|se| se.is_bucket_already_exists()) ||
se.is_bucket_already_owned_by_you()
            == Some(true)
        {
            Ok(None)
        }
    })
}
```

```

        } else {
            Err(S3ExampleError::from(err))
        }
    })
}

```

- API 세부 정보는 AWS SDK for Rust API 참조의 [CreateBucket](#)을 참조하세요.

SAP ABAP

SDK for SAP ABAP API

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

TRY.
    " determine our region from our session
    DATA(lv_region) = CONV /aws1/s3_bucketlocationcnstrnt( lo_session-
>get_region( ) ).
    DATA lo_constraint TYPE REF TO /aws1/cl_s3_createbucketconf.
    " When in the us-east-1 region, you must not specify a constraint
    " In all other regions, specify the region as the constraint
    IF lv_region = 'us-east-1'.
        CLEAR lo_constraint.
    ELSE.
        lo_constraint = NEW /aws1/cl_s3_createbucketconf( lv_region ).
    ENDIF.

    lo_s3->createbucket(
        iv_bucket = iv_bucket_name
        io_createbucketconfiguration = lo_constraint ).
    MESSAGE 'S3 bucket created.' TYPE 'I'.
    CATCH /aws1/cx_s3_bucketalrdyexists.
        MESSAGE 'Bucket name already exists.' TYPE 'E'.
    CATCH /aws1/cx_s3_bktalrdyownedbyyou.
        MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
ENDTRY.

```

- API 세부 정보는 AWSSDK for SAP ABAP API의 [CreateBucket](#)을 참조하세요.

Swift

SDK for Swift

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSS3

public func createBucket(name: String) async throws {
    var input = CreateBucketInput(
        bucket: name
    )

    // For regions other than "us-east-1", you must set the
    locationConstraint in the createBucketConfiguration.
    // For more information, see LocationConstraint in the S3 API guide.
    // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_CreateBucket.html#API_CreateBucket_RequestBody
    if let region = configuration.region {
        if region != "us-east-1" {
            input.createBucketConfiguration =
S3ClientTypes.CreateBucketConfiguration(locationConstraint:
S3ClientTypes.BucketLocationConstraint(rawValue: region))
        }
    }

    do {
        _ = try await client.createBucket(input: input)
    }
    catch let error as BucketAlreadyOwnedByYou {
        print("The bucket '\(name)' already exists and is owned by you. You
may wish to ignore this exception.")
        throw error
    }
}
```

```
        catch {
            print("ERROR: ", dump(error, name: "Creating a bucket"))
            throw error
        }
    }
}
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 [CreateBucket](#)을 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **CreateMultiRegionAccessPoint** 사용

다음 코드 예시는 CreateMultiRegionAccessPoint의 사용 방법을 보여 줍니다.

Kotlin

SDK for Kotlin

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

us-west-2 리전으로 요청을 보내도록 S3 제어 클라이언트를 구성합니다.

```
suspend fun createS3ControlClient(): S3ControlClient {
    // Configure your S3ControlClient to send requests to US West
    (Oregon).
    val s3Control = S3ControlClient.fromEnvironment {
        region = "us-west-2"
    }
    return s3Control
}
```

다중 리전 액세스 포인트를 생성합니다.


```

suspend fun createMrap(
    s3Control: S3ControlClient,
    accountIdParam: String,
    bucketName1: String,
    bucketName2: String,
    mrapName: String,
): String {
    println("Creating MRAP ...")
    val createMrapResponse: CreateMultiRegionAccessPointResponse =
        s3Control.createMultiRegionAccessPoint {
            accountId = accountIdParam
            clientToken = UUID.randomUUID().toString()
            details {
                name = mrapName
                regions = listOf(
                    Region {
                        bucket = bucketName1
                    },
                    Region {
                        bucket = bucketName2
                    },
                )
            }
        }
    val requestToken: String? = createMrapResponse.requestTokenArn

    // Use the request token to check for the status of the
    CreateMultiRegionAccessPoint operation.
    if (requestToken != null) {
        waitForSucceededStatus(s3Control, requestToken, accountIdParam)
        println("MRAP created")
    }

    val getMrapResponse =
        s3Control.getMultiRegionAccessPoint(
            input = GetMultiRegionAccessPointRequest {
                accountId = accountIdParam
                name = mrapName
            },
        )
    val mrapAlias = getMrapResponse.accessPoint?.alias
    return "arn:aws:s3:::$accountIdParam:accesspoint/$mrapAlias"
}

```

다중 리전 액세스 포인트가 사용 가능해질 때까지 기다립니다.

```
suspend fun waitForSucceededStatus(
    s3Control: S3ControlClient,
    requestToken: String,
    accountIdParam: String,
    timeBetweenChecks: Duration = 1.minutes,
) {
    var describeResponse: DescribeMultiRegionAccessPointOperationResponse
    describeResponse = s3Control.describeMultiRegionAccessPointOperation(
        input = DescribeMultiRegionAccessPointOperationRequest {
            accountId = accountIdParam
            requestTokenArn = requestToken
        },
    )

    var status: String? = describeResponse.asyncOperation?.requestStatus
    while (status != "SUCCEEDED") {
        delay(timeBetweenChecks)
        describeResponse =
            s3Control.describeMultiRegionAccessPointOperation(
                input = DescribeMultiRegionAccessPointOperationRequest {
                    accountId = accountIdParam
                    requestTokenArn = requestToken
                },
            )
        status = describeResponse.asyncOperation?.requestStatus
        println(status)
    }
}
```

- 자세한 내용은 [AWS SDK for Kotlin 개발자 안내서](#)를 참조하세요.
- API 세부 정보는 Kotlin API 참조용 AWS SDK의 [CreateMultiRegionAccessPoint](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **CreateMultipartUpload** 사용

다음 코드 예시는 CreateMultipartUpload의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [멀티파트 복사 수행](#)
- [체크섬 사용](#)
- [Amazon S3 객체 무결성 작업](#)

C++

SDK for C++

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String
&key,
                                Aws::S3::Model::ChecksumAlgorithm
checksumAlgorithm,
                                const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
}
```

```

    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
        client.CreateMultipartUpload(request);

    Aws::String uploadID;
    if (outcome.IsSuccess()) {
        uploadID = outcome.GetResult().GetUploadId();
    } else {
        std::cerr << "Error creating multipart upload: " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return uploadID;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateMultipartUpload](#)를 참조하세요.

CLI

AWS CLI

다음 명령은 키 multipart/01를 사용하여 버킷 amzn-s3-demo-bucket에 멀티파트 업로드를 생성합니다.

```
aws s3api create-multipart-upload --bucket amzn-s3-demo-bucket --key
'multipart/01'
```

출력:

```

{
  "Bucket": "amzn-s3-demo-bucket",
  "UploadId":
    "dfRtDYU0WwCCcH43C3WfbkR0NycyCpTJJvxu2i5GYkZ1jF.Yxwh6XG7WfS2vC4to6HiV6Yj1x.cph0gtNBtJ8P3
  "Key": "multipart/01"
}

```

완성된 파일은 이름이 01이며 amzn-s3-demo-bucket 버킷의 multipart 폴더에 있습니다. upload-part 명령과 함께 사용할 업로드 ID, 키, 버킷 이름을 저장합니다.

- API 세부 정보는 AWS CLI 명령 참조의 [CreateMultipartUpload](#)를 참조하세요.

Rust

SDK for Rust

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
// upload the file.
let multipart_upload_res: CreateMultipartUploadOutput = client
    .create_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .send()
    .await?;

let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
    "Missing upload_id after CreateMultipartUpload",
))?;
```

```
let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))
        .build()
        .await
        .unwrap();

    // Chunk index needs to start at 0, but part numbers start at 1.
    let part_number = (chunk_index as i32) + 1;
```

```

        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;

        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );
    }

```

```

// upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
let completed_multipart_upload: CompletedMultipartUpload =
    CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();

let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await?;

```

- API 세부 정보는 AWS SDK for Rust API 참조의 [CreateMultipartUpload](#)를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **CreatePresignedPost** 사용

다음 코드 예시는 CreatePresignedPost의 사용 방법을 보여 줍니다.

.NET

SDK for .NET(v4)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

미리 서명된 POST URL을 생성합니다.

```
/// <summary>
/// Create a presigned POST URL with conditions.
/// </summary>
/// <param name="s3Client">The Amazon S3 client.</param>
/// <param name="bucketName">The name of the bucket.</param>
/// <param name="objectKey">The object key (path) where the uploaded file
will be stored.</param>
/// <param name="expires">When the presigned URL expires.</param>
/// <param name="fields">Dictionary of fields to add to the form.</param>
/// <param name="conditions">List of conditions to apply.</param>
/// <returns>A CreatePresignedPostResponse object with URL and form fields.</
returns>
public async Task<CreatePresignedPostResponse> CreatePresignedPostAsync(
    IAmazonS3 s3Client,
    string bucketName,
    string objectKey,
    DateTime expires,
    Dictionary<string, string>? fields = null,
    List<S3PostCondition>? conditions = null)
{
    var request = new CreatePresignedPostRequest
    {
        BucketName = bucketName,
        Key = objectKey,
        Expires = expires
    };
};
```

```
// Add custom fields if provided
if (fields != null)
{
    foreach (var field in fields)
    {
        request.Fields.Add(field.Key, field.Value);
    }
}

// Add conditions if provided
if (conditions != null)
{
    foreach (var condition in conditions)
    {
        request.Conditions.Add(condition);
    }
}

return await s3Client.CreatePresignedPostAsync(request);
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 [CreatePresignedPost](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteBucket** 사용

다음 코드 예시는 DeleteBucket의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [기본 사항 알아보기](#)
- [구성 관리 시작하기](#)
- [기계 학습 특성 저장소 시작하기](#)
- [객체 스토리지 시작하기](#)

.NET

SDK for .NET(v4)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Shows how to delete an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the Amazon S3 bucket to delete.</
param>
/// <returns>A boolean value that represents the success or failure of
/// the delete operation.</returns>
public async Task<bool> DeleteBucketAsync(string bucketName)
{
    try
    {
        var request = new DeleteBucketRequest { BucketName = bucketName, };

        await _amazonS3.DeleteBucketAsync(request);
        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error deleting bucket: {ex.Message}");
        return false;
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 [DeleteBucket](#)을 참조하십시오.

Bash

AWS CLIBash 스크립트와 함께 사용

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.

# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_bucket() {
    local bucket_name=$1
    local response

    response=$(aws s3api delete-bucket \
        --bucket "$bucket_name")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
        return 1
    fi
}
```

```
fi
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [DeleteBucket](#)을 참조하십시오.

C++

SDK for C++

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::S3::deleteBucket(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
        client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
        std::endl;
    } else {
        std::cout << "The bucket was deleted" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteBucket](#)을 참조하세요.

CLI

AWS CLI

다음 명령은 이름이 amzn-s3-demo-bucket인 버킷을 삭제합니다.

```
aws s3api delete-bucket --bucket amzn-s3-demo-bucket --region us-east-1
```

- API 세부 정보는 AWS CLI 명령 참조의 [DeleteBucket](#)을 참조하세요.

Go

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)
```

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// DeleteBucket deletes a bucket. The bucket must be empty or an error is
// returned.
func (basics BucketBasics) DeleteBucket(ctx context.Context, bucketName string)
error {
    _, err := basics.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
        Bucket: aws.String(bucketName)})
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucketName)
            err = noBucket
        } else {
            log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
        }
    } else {
        err = s3.NewBucketNotExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(bucketName)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
                bucketName)
        } else {
            log.Printf("Deleted %s.\n", bucketName)
        }
    }
    return err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteBucket](#)을 참조하십시오.

Java

SDK for Java 2.x

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Deletes an S3 bucket asynchronously.
 *
 * @param bucket the name of the bucket to be deleted
 * @return a {@link CompletableFuture} that completes when the bucket
 * deletion is successful, or throws a {@link RuntimeException}
 * if an error occurs during the deletion process
 */
public CompletableFuture<Void> deleteBucketAsync(String bucket) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucket)
        .build();

    CompletableFuture<DeleteBucketResponse> response =
getAsyncClient().deleteBucket(deleteBucketRequest);
    response.whenComplete((deleteRes, ex) -> {
        if (deleteRes != null) {
            logger.info(bucket + " was deleted.");
        } else {
            throw new RuntimeException("An S3 exception occurred during
bucket deletion", ex);
        }
    });
    return response.thenApply(r -> null);
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [DeleteBucket](#)을 참조하십시오.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷을 삭제합니다.

```
import {
  DeleteBucketCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Delete an Amazon S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new DeleteBucketCommand({
    Bucket: bucketName,
  });

  try {
    await client.send(command);
    console.log("Bucket was deleted.");
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting bucket. The bucket doesn't exist.`
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting the bucket. ${caught.name}:`
        `${caught.message}`
      );
    }
  }
}
```

```
    );  
  } else {  
    throw caught;  
  }  
}  
};
```

- 자세한 정보는 [AWS SDK for JavaScript 개발자 안내서](#)를 참조하세요.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 [DeleteBucket](#)을 참조하십시오.

PHP

SDK for PHP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

빈 버킷을 삭제합니다.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);  
  
try {  
    $this->s3client->deleteBucket([  
        'Bucket' => $this->bucketName,  
    ]);  
    echo "Deleted bucket $this->bucketName.\n";  
} catch (Exception $exception) {  
    echo "Failed to delete $this->bucketName with error: " . $exception->getMessage();  
    exit("Please fix error with bucket deletion before continuing.");  
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 [DeleteBucket](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 'test-files' 버킷에서 모든 객체 및 객체 버전을 제거한 다음 버킷을 삭제합니다. 명령을 실행하면 계속 진행하기 전에 확인하라는 프롬프트가 표시됩니다. 확인 프롬프트를 차단하려면 -Force 스위치를 추가합니다. 비어 있지 않은 버킷은 삭제할 수 없다는 점에 유의하세요.

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [DeleteBucket](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 'test-files' 버킷에서 모든 객체 및 객체 버전을 제거한 다음 버킷을 삭제합니다. 명령을 실행하면 계속 진행하기 전에 확인하라는 프롬프트가 표시됩니다. 확인 프롬프트를 차단하려면 -Force 스위치를 추가합니다. 비어 있지 않은 버킷은 삭제할 수 없다는 점에 유의하세요.

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [DeleteBucket](#)을 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
```

```

        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def delete(self):
        """
        Delete the bucket. The bucket must be empty or an error is raised.
        """
        try:
            self.bucket.delete()
            self.bucket.wait_until_not_exists()
            logger.info("Bucket %s successfully deleted.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete bucket %s.", self.bucket.name)
            raise

```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [DeleteBucket](#)를 참조하십시오.

Ruby

SDK for Ruby

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?")
  answer = gets.chomp.downcase
  if answer == 'y'

```

```

    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end

```

- API 세부 정보는 AWS SDK for Ruby API 참조의 [DeleteBucket](#)을 참조하십시오.

Rust

SDK for Rust

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

pub async fn delete_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<(), S3ExampleError> {
    let resp = client.delete_bucket().bucket(bucket_name).send().await;
    match resp {
        Ok(_) => Ok(()),
        Err(err) => {
            if err
                .as_service_error()
                .and_then(aws_sdk_s3::error::ProvideErrorMetadata::code)
                == Some("NoSuchBucket")
            {
                Ok(())
            } else {
                Err(S3ExampleError::from(err))
            }
        }
    }
}

```

```
}

```

- API 세부 정보는 AWS SDK for Rust API 참조의 [DeleteBucket](#)을 참조하십시오.

SAP ABAP

SDK for SAP ABAP API

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.
```

```
    lo_s3->deletebucket(
        iv_bucket = iv_bucket_name ).
    MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- API 세부 정보는 AWSSDK for SAP ABAP API의 [DeleteBucket](#)을 참조하십시오.

Swift

SDK for Swift

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSS3
```

```
public func deleteBucket(name: String) async throws {
    let input = DeleteBucketInput(
        bucket: name
    )
    do {
        _ = try await client.deleteBucket(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Deleting a bucket"))
        throw error
    }
}
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 [DeleteBucket](#)을 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 DeleteBucketAnalyticsConfiguration 사용

다음 코드 예시는 DeleteBucketAnalyticsConfiguration의 사용 방법을 보여 줍니다.

CLI

AWS CLI

버킷의 분석 구성을 삭제하려면

다음 delete-bucket-analytics-configuration 예시에서는 지정된 버킷 및 ID에 대한 분석 구성을 제거합니다.

```
aws s3api delete-bucket-analytics-configuration \
    --bucket amzn-s3-demo-bucket \
    --id 1
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 참조의 [DeleteBucketAnalyticsConfiguration](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷에서 이름이 'testfilter'인 분석 필터를 제거합니다.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AnalyticsId 'testfilter'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [DeleteBucketAnalyticsConfiguration](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 S3 버킷에서 이름이 'testfilter'인 분석 필터를 제거합니다.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AnalyticsId 'testfilter'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [DeleteBucketAnalyticsConfiguration](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteBucketCors** 사용

다음 코드 예시는 DeleteBucketCors의 사용 방법을 보여 줍니다.

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    /// <summary>
    /// Deletes a CORS configuration from an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used
    /// to delete the CORS configuration from the bucket.</param>
    private static async Task DeleteCORSConfigurationAsync(AmazonS3Client
client)
    {
        DeleteCORSConfigurationRequest request = new
DeleteCORSConfigurationRequest()
        {
            BucketName = BucketName,
        };
        await client.DeleteCORSConfigurationAsync(request);
    }

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [DeleteBucketCors](#)를 참조하세요.

CLI

AWS CLI

다음 명령은 이름이 `amzn-s3-demo-bucket`인 버킷에서 Cross-Origin Resource Sharing 구성을 삭제합니다.

```
aws s3api delete-bucket-cors --bucket amzn-s3-demo-bucket
```

- API 세부 정보는 AWS CLI 명령 참조의 [DeleteBucketCors](#)를 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def delete_cors(self):
        """
        Delete the CORS rules from the bucket.

        :param bucket_name: The name of the bucket to update.
        """
        try:
            self.bucket.Cors().delete()
            logger.info("Deleted CORS from bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete CORS from bucket '%s'.",
self.bucket.name)
            raise

```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [DeleteBucketCors](#)를 참조하십시오.

Ruby

SDK for Ruby

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.


```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
  # an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Deletes the CORS configuration of a bucket.
  #
  # @return [Boolean] True if the CORS rules were deleted; otherwise, false.
  def delete_cors
    @bucket_cors.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete CORS rules for #{@bucket_cors.bucket.name}. Here's why:
    #{e.message}"
    false
  end
end

end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 [DeleteBucketCors](#)를 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

TRY.

```
lo_s3->deletebucketcors(
```

```
        iv_bucket = iv_bucket_name ).
    MESSAGE 'Bucket CORS configuration deleted.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 [DeleteBucketCors](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 DeleteBucketEncryption 사용

다음 코드 예시는 DeleteBucketEncryption의 사용 방법을 보여 줍니다.

CLI

AWS CLI

버킷의 서버 측 암호화 구성을 삭제하려면

다음 delete-bucket-encryption 예시에서는 지정된 버킷의 서버 측 암호화 구성을 삭제합니다.

```
aws s3api delete-bucket-encryption \
    --bucket amzn-s3-demo-bucket
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 참조의 [DeleteBucketEncryption](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이렇게 하면 제공된 S3 버킷에 대해 활성화된 암호화가 비활성화됩니다.

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

출력:

Confirm

Are you sure you want to perform this action?

Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on target "s3casetestbucket".

[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): Y

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [DeleteBucketEncryption](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이렇게 하면 제공된 S3 버킷에 대해 활성화된 암호화가 비활성화됩니다.

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

출력:

Confirm

Are you sure you want to perform this action?

Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on target "s3casetestbucket".

[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): Y

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [DeleteBucketEncryption](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 DeleteBucketInventoryConfiguration 사용

다음 코드 예시는 DeleteBucketInventoryConfiguration의 사용 방법을 보여 줍니다.

CLI

AWS CLI

버킷의 인벤토리 구성을 삭제하려면

다음 delete-bucket-inventory-configuration 예시에서는 지정된 버킷에 대해 ID가 1인 인벤토리 구성을 삭제합니다.

```
aws s3api delete-bucket-inventory-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --id 1
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 참조의 [DeleteBucketInventoryConfiguration](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷에 해당하는 'testInventoryName'이라는 이름의 인벤토리를 제거합니다.

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -  
InventoryId 'testInventoryName'
```

출력:

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3BucketInventoryConfiguration  
(DeleteBucketInventoryConfiguration)" on target "amzn-s3-demo-bucket".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [DeleteBucketInventoryConfiguration](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 S3 버킷에 해당하는 'testInventoryName'이라는 이름의 인벤토리를 제거합니다.

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -  
InventoryId 'testInventoryName'
```

출력:

Confirm

Are you sure you want to perform this action?

Performing the operation "Remove-S3BucketInventoryConfiguration

(DeleteBucketInventoryConfiguration)" on target "amzn-s3-demo-bucket".

[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): Y

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [DeleteBucketInventoryConfiguration](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteBucketLifecycle** 사용

다음 코드 예시는 DeleteBucketLifecycle의 사용 방법을 보여 줍니다.

.NET

SDK for .NET

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    /// <summary>
    /// This method removes the Lifecycle configuration from the named
    /// S3 bucket.
    /// </summary>
    /// <param name="client">The S3 client object used to call
    /// the RemoveLifecycleConfigAsync method.</param>
    /// <param name="bucketName">A string representing the name of the
    /// S3 bucket from which the configuration will be removed.</param>
    public static async Task RemoveLifecycleConfigAsync(IAmazonS3 client,
    string bucketName)
    {
        var request = new DeleteLifecycleConfigurationRequest()

```

```
{
    BucketName = bucketName,
};
await client.DeleteLifecycleConfigurationAsync(request);
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 [DeleteBucketLifecycle](#)을 참조하세요.

CLI

AWS CLI

다음 명령은 `amzn-s3-demo-bucket`이라는 버킷에서 수명 주기 구성을 삭제합니다.

```
aws s3api delete-bucket-lifecycle --bucket amzn-s3-demo-bucket
```

- API 세부 정보는 AWS CLI 명령 참조의 [DeleteBucketLifecycle](#)을 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
```

```

        self.name = bucket.name

    def delete_lifecycle_configuration(self):
        """
        Remove the lifecycle configuration from the specified bucket.
        """
        try:
            self.bucket.LifecycleConfiguration().delete()
            logger.info(
                "Deleted lifecycle configuration for bucket '%s'.",
                self.bucket.name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete lifecycle configuration for bucket '%s'.",
                self.bucket.name,
            )
            raise

```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [DeleteBucketLifecycle](#)를 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

TRY.

```

lo_s3->deletebucketlifecycle(
    iv_bucket = iv_bucket_name ).
MESSAGE 'Bucket lifecycle configuration deleted.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
MESSAGE 'Bucket does not exist.' TYPE 'E'.

```

```
ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 [DeleteBucketLifecycle](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 DeleteBucketMetricsConfiguration 사용

다음 코드 예시는 DeleteBucketMetricsConfiguration의 사용 방법을 보여 줍니다.

CLI

AWS CLI

버킷의 지표 구성을 삭제하려면

다음 delete-bucket-metrics-configuration 예시에서는 지정된 버킷 및 ID에 대한 지표 구성을 제거합니다.

```
aws s3api delete-bucket-metrics-configuration \
  --bucket amzn-s3-demo-bucket \
  --id 123
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 참조의 [DeleteBucketMetricsConfiguration](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷에서 이름이 'testmetrics'인 지표 필터를 제거합니다.

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
'testmetrics'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [DeleteBucketMetricsConfiguration](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 S3 버킷에서 이름이 'testmetrics'인 지표 필터를 제거합니다.

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testmetrics'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [DeleteBucketMetricsConfiguration](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteBucketPolicy** 사용

다음 코드 예시는 DeleteBucketPolicy의 사용 방법을 보여 줍니다.

C++

SDK for C++

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::S3::deleteBucketPolicy(const Aws::String &bucketName,  
                                     const Aws::S3::S3ClientConfiguration  
                                     &clientConfig) {  
    Aws::S3::S3Client client(clientConfig);  
  
    Aws::S3::Model::DeleteBucketPolicyRequest request;  
    request.SetBucket(bucketName);  
  
    Aws::S3::Model::DeleteBucketPolicyOutcome outcome =  
        client.DeleteBucketPolicy(request);  
  
    if (!outcome.IsSuccess()) {  
        const Aws::S3::S3Error &err = outcome.GetError();  
    }  
}
```

```

        std::cerr << "Error: deleteBucketPolicy: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "Policy was deleted from the bucket." << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteBucketPolicy](#)를 참조하세요.

CLI

AWS CLI

다음 명령은 이름이 `amzn-s3-demo-bucket`인 버킷에서 버킷 정책을 삭제합니다.

```
aws s3api delete-bucket-policy --bucket amzn-s3-demo-bucket
```

- API 세부 정보는 AWS CLI 명령 참조의 [DeleteBucketPolicy](#)를 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;

/**
 * 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 DeleteBucketPolicy {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to delete the policy from
(for example, bucket1).""";

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

        String bucketName = args[0];
        System.out.format("Deleting policy from bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteS3BucketPolicy(s3, bucketName);
        s3.close();
    }

    /**
     * Deletes the S3 bucket policy for the specified bucket.
     *
     * @param s3 the {@link S3Client} instance to use for the operation
     * @param bucketName the name of the S3 bucket for which the policy should be
deleted
     *
     * @throws S3Exception if there is an error deleting the bucket policy
     */
}

```

```

public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {
    DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        s3.deleteBucketPolicy(delReq);
        System.out.println("Done!");
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [DeleteBucketPolicy](#)를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷 정책을 삭제합니다.

```

import {
    DeleteBucketPolicyCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Remove the policy from an Amazon S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {

```

```
const client = new S3Client({});

try {
  await client.send(
    new DeleteBucketPolicyCommand({
      Bucket: bucketName,
    }),
  );
  console.log(`Bucket policy deleted from "${bucketName}".`);
} catch (caught) {
  if (
    caught instanceof S3ServiceException &&
    caught.name === "NoSuchBucket"
  ) {
    console.error(
      `Error from S3 while deleting policy from ${bucketName}. The bucket
doesn't exist.`,
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while deleting policy from ${bucketName}. ${caught.name}:
${caught.message}`,
    );
  } else {
    throw caught;
  }
}
};
```

- 자세한 정보는 [AWS SDK for JavaScript 개발자 안내서](#)를 참조하세요.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 [DeleteBucketPolicy](#)를 참조하십시오.

Kotlin

SDK for Kotlin

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {  
    val request =  
        DeleteBucketPolicyRequest {  
            bucket = bucketName  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.deleteBucketPolicy(request)  
        println("Done!")  
    }  
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 [DeleteBucketPolicy](#)를 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷과 연결된 버킷 정책을 제거합니다.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [DeleteBucketPolicy](#)를 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 S3 버킷과 연결된 버킷 정책을 제거합니다.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [DeleteBucketPolicy](#)를 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def delete_policy(self):
        """
        Delete the security policy from the bucket.
        """
        try:
            self.bucket.Policy().delete()
            logger.info("Deleted policy for bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception(
                "Couldn't delete policy for bucket '%s'.", self.bucket.name
            )
            raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [DeleteBucketPolicy](#)를 참조하십시오.

Ruby

SDK for Ruby

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
  configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  def delete_policy
    @bucket_policy.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete the policy from #{@bucket_policy.bucket.name}. Here's
  why: #{e.message}"
    false
  end
end

end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 [DeleteBucketPolicy](#)를 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.  
    lo_s3->deletebucketpolicy(  
        iv_bucket = iv_bucket_name ).  
    MESSAGE 'Bucket policy deleted.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 [DeleteBucketPolicy](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 **DeleteBucketReplication** 사용

다음 코드 예시는 DeleteBucketReplication의 사용 방법을 보여 줍니다.

CLI

AWS CLI

다음 명령은 amzn-s3-demo-bucket이라는 버킷에서 복제 구성을 삭제합니다.

```
aws s3api delete-bucket-replication --bucket amzn-s3-demo-bucket
```

- API 세부 정보는 AWS CLI 명령 참조의 [DeleteBucketReplication](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예제 1: 이름이 'amzn-s3-demo-bucket'인 버킷과 연결된 복제 구성을 삭제합니다. 이 작업에는 s3:DeleteReplicationConfiguration 작업에 대한 권한이 필요합니다. 작업이 진행되기 전에 확인 메시지가 표시됩니다. 확인 메시지를 차단하려면 -Force 스위치를 사용하세요.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [DeleteBucketReplication](#)을 참조하세요.

Tools for PowerShell V5

예제 1: 이름이 'amzn-s3-demo-bucket'인 버킷과 연결된 복제 구성을 삭제합니다. 이 작업에는 s3:DeleteReplicationConfiguration 작업에 대한 권한이 필요합니다. 작업이 진행되기 전에 확인 메시지가 표시됩니다. 확인 메시지를 차단하려면 -Force 스위치를 사용하세요.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [DeleteBucketReplication](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 DeleteBucketTagging 사용

다음 코드 예시는 DeleteBucketTagging의 사용 방법을 보여 줍니다.

CLI

AWS CLI

다음 명령은 amzn-s3-demo-bucket이라는 버킷에서 태그 지정 구성을 삭제합니다.

```
aws s3api delete-bucket-tagging --bucket amzn-s3-demo-bucket
```

- API 세부 정보는 AWS CLI 명령 참조의 [DeleteBucketTagging](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷과 연결된 모든 태그를 제거합니다.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

출력:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
"amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [DeleteBucketTagging](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 S3 버킷과 연결된 모든 태그를 제거합니다.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

출력:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
"amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [DeleteBucketTagging](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteBucketWebsite** 사용

다음 코드 예시는 DeleteBucketWebsite의 사용 방법을 보여 줍니다.

C++

SDK for C++

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::S3::deleteBucketWebsite(const Aws::String &bucketName,
                                       const Aws::S3::S3ClientConfiguration
                                       &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketWebsiteOutcome outcome =
        client.DeleteBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteBucketWebsite: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
        std::endl;
    } else {
        std::cout << "Website configuration was removed." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteBucketWebsite](#)를 참조하세요.

CLI

AWS CLI

다음 명령은 `amzn-s3-demo-bucket`이라는 버킷에서 웹 사이트 구성을 삭제합니다.

```
aws s3api delete-bucket-website --bucket amzn-s3-demo-bucket
```

- API 세부 정보는 AWS CLI 명령 참조의 [DeleteBucketWebsite](#)를 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */

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

        Usage:      <bucketName>
```

```

        Where:
            bucketName - The Amazon S3 bucket to delete the website
configuration from.
        """;

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

        String bucketName = args[0];
        System.out.format("Deleting website configuration for Amazon S3 bucket:
%s\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteBucketWebsiteConfig(s3, bucketName);
        System.out.println("Done!");
        s3.close();
    }

    /**
     * Deletes the website configuration for an Amazon S3 bucket.
     *
     * @param s3 The {@link S3Client} instance used to interact with Amazon S3.
     * @param bucketName The name of the S3 bucket for which the website
configuration should be deleted.
     * @throws S3Exception If an error occurs while deleting the website
configuration.
     */
    public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName)
    {
        DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
            .bucket(bucketName)
            .build();

        try {
            s3.deleteBucketWebsite(delReq);
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

```

```

        System.out.println("Failed to delete website configuration!");
        System.exit(1);
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [DeleteBucketWebsite](#)를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷에서 웹 사이트 구성을 삭제합니다.

```

import {
    DeleteBucketWebsiteCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Remove the website configuration for a bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
    const client = new S3Client({});

    try {
        await client.send(
            new DeleteBucketWebsiteCommand({
                Bucket: bucketName,
            }),
        );
        // The response code will be successful for both removed configurations and
        // configurations that did not exist in the first place.
    }
}

```

```

    console.log(
      `The bucket "${bucketName}" is not longer configured as a website, or it
      never was.`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while removing website configuration from ${bucketName}.
        The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while removing website configuration from ${bucketName}.
        ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

```

- 자세한 정보는 [AWS SDK for JavaScript 개발자 안내서](#)를 참조하세요.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 [DeleteBucketWebsite](#)를 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷의 정적 웹 사이트 호스팅 속성을 비활성화합니다.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

출력:

```

Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
"amzn-s3-demo-bucket".

```



```
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): Y
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [DeleteBucketWebsite](#)를 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 S3 버킷의 정적 웹 사이트 호스팅 속성을 비활성화합니다.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

출력:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
"amzn-s3-demo-bucket".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [DeleteBucketWebsite](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteObject** 사용

다음 코드 예시는 DeleteObject의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [Amazon S3 객체 무결성 작업](#)
- [버전이 지정된 객체 작업](#)

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버저닝되지 않은 S3 버킷에서 객체를 삭제합니다.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to delete an object from a non-versioned Amazon
/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DeleteObject
{
    /// <summary>
    /// The Main method initializes the necessary variables and then calls
    /// the DeleteObjectNonVersionedBucketAsync method to delete the object
    /// named by the keyName parameter.
    /// </summary>
    public static async Task Main()
    {
        const string bucketName = "amzn-s3-demo-bucket";
        const string keyName = "testfile.txt";

        // If the Amazon S3 bucket is located in an AWS Region other than the
        // Region of the default account, define the AWS Region for the
        // Amazon S3 bucket in your call to the AmazonS3Client constructor.
        // For example RegionEndpoint.USWest2.
        IAmazonS3 client = new AmazonS3Client();
        await DeleteObjectNonVersionedBucketAsync(client, bucketName,
keyName);
    }

    /// <summary>
```

```

    /// The DeleteObjectNonVersionedBucketAsync takes care of deleting the
    /// desired object from the named bucket.
    /// </summary>
    /// <param name="client">An initialized Amazon S3 client used to delete
    /// an object from an Amazon S3 bucket.</param>
    /// <param name="bucketName">The name of the bucket from which the
    /// object will be deleted.</param>
    /// <param name="keyName">The name of the object to delete.</param>
    public static async Task DeleteObjectNonVersionedBucketAsync(IAmazonS3
client, string bucketName, string keyName)
    {
        try
        {
            var deleteObjectRequest = new DeleteObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
            };

            Console.WriteLine($"Deleting object: {keyName}");
            await client.DeleteObjectAsync(deleteObjectRequest);
            Console.WriteLine($"Object: {keyName} deleted from
{bucketName}.");
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error encountered on server.
Message: '{ex.Message}' when deleting an object.");
        }
    }
}

```

버저닝된 S3 버킷에서 객체를 삭제합니다.

```

using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example creates an object in an Amazon Simple Storage Service

```

```

/// (Amazon S3) bucket and then deletes the object version that was
/// created.
/// </summary>
public class DeleteObjectVersion
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "verstioned-object.txt";

        // If the AWS Region of the default user is different from the AWS
        // Region of the Amazon S3 bucket, pass the AWS Region of the
        // bucket region to the Amazon S3 client object's constructor.
        // Define it like this:
        //      RegionEndpoint bucketRegion = RegionEndpoint.USSouth1;
        IAmazonS3 client = new AmazonS3Client();

        await CreateAndDeleteObjectVersionAsync(client, bucketName, keyName);
    }

    /// <summary>
    /// This method creates and then deletes a versioned object.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
    /// create and delete the object.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
    /// object will be created and deleted.</param>
    /// <param name="keyName">The key name of the object to create.</param>
    public static async Task CreateAndDeleteObjectVersionAsync(IAmazonS3
client, string bucketName, string keyName)
    {
        try
        {
            // Add a sample object.
            string versionID = await PutAnObject(client, bucketName,
keyName);

            // Delete the object by specifying an object key and a version
ID.

            DeleteObjectRequest request = new DeleteObjectRequest()
            {
                BucketName = bucketName,
                Key = keyName,
                VersionId = versionID,

```

```

        };

        Console.WriteLine("Deleting an object");
        await client.DeleteObjectAsync(request);
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error: {ex.Message}");
    }
}

/// <summary>
/// This method is used to create the temporary Amazon S3 object.
/// </summary>
/// <param name="client">The initialized Amazon S3 object which will be
used
/// to create the temporary Amazon S3 object.</param>
/// <param name="bucketName">The name of the Amazon S3 bucket where the
object
/// will be created.</param>
/// <param name="objectKey">The name of the Amazon S3 object to create.</
param>
/// <returns>The Version ID of the created object.</returns>
public static async Task<string> PutAnObject(IAmazonS3 client, string
bucketName, string objectKey)
{
    PutObjectRequest request = new PutObjectRequest()
    {
        BucketName = bucketName,
        Key = objectKey,
        ContentBody = "This is the content body!",
    };

    PutObjectResponse response = await client.PutObjectAsync(request);
    return response.VersionId;
}
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [DeleteObject](#)를 참조하십시오.

Bash

AWS CLIBash 스크립트와 함께 사용

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function delete_item_in_bucket
#
# This function deletes the specified file from the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.
#     $2 - The key (file name) in the bucket to delete.

# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_item_in_bucket() {
    local bucket_name=$1
    local key=$2
    local response

    response=$(aws s3api delete-object \
        --bucket "$bucket_name" \
        --key "$key")

    # shellcheck disable=SC2181
```

```

if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api delete-object operation failed.\n
$response"
    return 1
fi
}

```

- API 세부 정보는 AWS CLI 명령 참조의 [DeleteBucket](#)을 참조하십시오.

C++

SDK for C++

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::S3::deleteObject(const Aws::String &objectKey,
                               const Aws::String &fromBucket,
                               const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectRequest request;

    request.WithKey(objectKey)
        .WithBucket(fromBucket);

    Aws::S3::Model::DeleteObjectOutcome outcome =
        client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
        std::endl;
    } else {
        std::cout << "Successfully deleted the object." << std::endl;
    }
}

```

```
    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteObject](#)를 참조하세요.

CLI

AWS CLI

다음 명령은 이름이 `amzn-s3-demo-bucket`인 버킷에서 `test.txt`라는 객체를 삭제합니다.

```
aws s3api delete-object --bucket amzn-s3-demo-bucket --key test.txt
```

버킷 버전 관리가 활성화된 경우 출력에는 삭제 마커의 버전 ID가 포함됩니다.

```
{
  "VersionId": "9_gKg5vG56F.TTEUdwkxGpJ3tND1WlGq",
  "DeleteMarker": true
}
```

객체를 삭제하는 방법에 대한 자세한 내용은 Amazon S3 개발자 안내서의 객체 삭제를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [DeleteObject](#)를 참조하세요.

Go

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import (
    "bytes"
    "context"
```



```

"errors"
"fmt"
"log"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
    string, versionId string, bypassGovernance bool) (bool, error) {
    deleted := false
    input := &s3.DeleteObjectInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    _, err := actor.S3Client.DeleteObject(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in %s.\n", key, bucket)
            err = noKey
        } else if errors.As(err, &apiErr) {
            switch apiErr.ErrorCode() {
            case "AccessDenied":

```

```

    log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
    err = nil
case "InvalidArgument":
    if bypassGovernance {
        log.Printf("You cannot specify bypass governance on a bucket without lock
enabled.")
        err = nil
    }
}
} else {
    err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
        ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: aws.String(key)},
        time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s in bucket %s to be deleted.
\n", key, bucket)
    } else {
        deleted = true
    }
}
return deleted, err
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteObject](#)를 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

/**
 * Deletes an object from an S3 bucket asynchronously.
 *

```

```

    * @param bucketName the name of the S3 bucket
    * @param key         the key (file name) of the object to be deleted
    * @return a {@link CompletableFuture} that completes when the object has
    been deleted
    */
    public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
String key) {
        DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
        response.whenComplete((deleteRes, ex) -> {
            if (deleteRes != null) {
                logger.info(key + " was deleted");
            } else {
                throw new RuntimeException("An S3 exception occurred during
delete", ex);
            }
        });

        return response.thenApply(r -> null);
    }

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [DeleteObject](#)를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

객체를 삭제합니다.

```
import {
```

```
DeleteObjectCommand,
S3Client,
S3ServiceException,
waitUntilObjectNotExists,
} from "@aws-sdk/client-s3";

/**
 * Delete one object from an Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new DeleteObjectCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    await waitUntilObjectNotExists(
      { client },
      { Bucket: bucketName, Key: key },
    );
    // A successful delete, or a delete for a non-existent object, both return
    // a 204 response code.
    console.log(
      `The object "${key}" from bucket "${bucketName}" was deleted, or it didn't
exist.`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting object from ${bucketName}. The bucket
doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting object from ${bucketName}. ${caught.name}:
${caught.message}`,
      );
    }
  }
}
```

```

    } else {
        throw caught;
    }
}
};

```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 [DeleteObject](#)를 참조하세요.

PHP

SDK for PHP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

public function deleteObject(string $bucketName, string $fileName, array
$args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
    $args);
    try {
        $this->client->deleteObject($parameters);
        if ($this->verbose) {
            echo "Deleted the object named: $fileName from $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete $fileName from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object deletion before continuing.";
        }
        throw $exception;
    }
}
}

```

- API 세부 정보는 AWS SDK for PHP API 참조의 [DeleteObject](#)를 참조하세요.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

객체를 삭제합니다.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def delete(self):
        """
        Deletes the object.
        """
        try:
            self.object.delete()
            self.object.wait_until_not_exists()
            logger.info(
                "Deleted object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't delete object '%s' from bucket '%s'.",
```

```

        self.object.key,
        self.object.bucket_name,
    )
    raise

```

객체의 최신 버전을 삭제하여 객체를 이전 버전으로 롤백합니다.

```

def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:

```

```

        version.delete()
        print(f"Deleted version {version.version_id}")
    else:
        break

    print(f"Active version is now {bucket.Object(object_key).version_id}")
else:
    raise KeyError(
        f"{version_id} was not found in the list of versions for "
        f"{object_key}."
    )

```

객체의 활성 삭제 마커를 제거하여 삭제된 객체를 다시 활성화합니다.

```

def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
    marker.
    By removing the delete marker, we make the previous version the latest
    version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",

```



```

        object_key,
        latest_version["LastModified"],
    )
    obj = bucket.Object(object_key)
    obj.Version(latest_version["VersionId"]).delete()
    logger.info(
        "Revived %s, active version is now %s with body '%s'",
        object_key,
        obj.version_id,
        obj.get()["Body"].read(),
    )
else:
    logger.warning(
        "Delete marker is not the latest version for %s!", object_key
    )
elif "Versions" in response:
    logger.warning("Got an active version for %s, nothing to do.",
object_key)
else:
    logger.error("Couldn't get any version info for %s.", object_key)

```

S3 객체에서 삭제 마커를 제거하는 Lambda 핸들러를 생성합니다. 이 핸들러를 사용하면 버전이 지정된 버킷에서 불필요한 삭제 마커를 효율적으로 정리할 수 있습니다.

```

import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.client("s3")

def lambda_handler(event, context):
    """
    Removes a delete marker from the specified versioned object.

    :param event: The S3 batch event that contains the ID of the delete marker

```

```

        to remove.
:param context: Context about the event.
:return: A result structure that Amazon S3 uses to interpret the result of
the
        operation. When the result code is TemporaryFailure, S3 retries the
        operation.
"""
# Parse job parameters from Amazon S3 batch operations
invocation_id = event["invocationId"]
invocation_schema_version = event["invocationSchemaVersion"]

results = []
result_code = None
result_string = None

task = event["tasks"][0]
task_id = task["taskId"]

try:
    obj_key = parse.unquote_plus(task["s3Key"], encoding="utf-8")
    obj_version_id = task["s3VersionId"]
    bucket_name = task["s3BucketArn"].split(":")[-1]

    logger.info(
        "Got task: remove delete marker %s from object %s.", obj_version_id,
obj_key
    )

    try:
        # If this call does not raise an error, the object version is not a
delete
        # marker and should not be deleted.
        response = s3.head_object(
            Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
        )
        result_code = "PermanentFailure"
        result_string = (
            f"Object {obj_key}, ID {obj_version_id} is not " f"a delete
marker."
        )

        logger.debug(response)
        logger.warning(result_string)
    except ClientError as error:

```

```

        delete_marker = error.response["ResponseMetadata"]
["HTTPHeaders"].get(
            "x-amz-delete-marker", "false"
        )
        if delete_marker == "true":
            logger.info(
                "Object %s, version %s is a delete marker.", obj_key,
obj_version_id
            )
            try:
                s3.delete_object(
                    Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
                )
                result_code = "Succeeded"
                result_string = (
                    f"Successfully removed delete marker "
                    f"{obj_version_id} from object {obj_key}."
                )
                logger.info(result_string)
            except ClientError as error:
                # Mark request timeout as a temporary failure so it will be
retried.

                if error.response["Error"]["Code"] == "RequestTimeout":
                    result_code = "TemporaryFailure"
                    result_string = (
                        f"Attempt to remove delete marker from "
                        f"object {obj_key} timed out."
                    )
                    logger.info(result_string)
                else:
                    raise
            else:
                raise ValueError(
                    f"The x-amz-delete-marker header is either not "
                    f"present or is not 'true'."
                )
        except Exception as error:
            # Mark all other exceptions as permanent failures.
            result_code = "PermanentFailure"
            result_string = str(error)
            logger.exception(error)
        finally:
            results.append(
                {

```

```

        "taskId": task_id,
        "resultCode": result_code,
        "resultString": result_string,
    }
)
return {
    "invocationSchemaVersion": invocation_schema_version,
    "treatMissingKeysAs": "PermanentFailure",
    "invocationId": invocation_id,
    "results": results,
}

```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [DeleteObject](#)를 참조하십시오.

Rust

SDK for Rust

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

/// Delete an object from a bucket.
pub async fn remove_object(
    client: &aws_sdk_s3::Client,
    bucket: &str,
    key: &str,
) -> Result<(), S3ExampleError> {
    client
        .delete_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await?;

    // There are no modeled errors to handle when deleting an object.

```

```
    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 [DeleteObject](#)을 참조하십시오.

SAP ABAP

SDK for SAP ABAP API

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.
    lo_s3->deleteobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_object_key ).
    MESSAGE 'Object deleted from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- API 세부 정보는 AWSSDK for SAP ABAP API의 [DeleteObject](#)을 참조하십시오.

Swift

SDK for Swift

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSS3
```

```
public func deleteFile(bucket: String, key: String) async throws {
    let input = DeleteObjectInput(
        bucket: bucket,
        key: key
    )

    do {
        _ = try await client.deleteObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Deleting a file. "))
        throw error
    }
}
```

- API 세부 정보는 [Swift용 AWS SDK API 참조](#)의 DeleteObject를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 DeleteObjectTagging 사용

다음 코드 예시는 DeleteObjectTagging의 사용 방법을 보여 줍니다.

CLI

AWS CLI

객체의 태그 세트를 삭제하려면

다음 delete-object-tagging 예시에서는 지정된 키가 있는 태그를 객체 doc1.rtf에서 삭제합니다.

```
aws s3api delete-object-tagging \
    --bucket amzn-s3-demo-bucket \
    --key doc1.rtf
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 참조의 [DeleteObjectTagging](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷에서 'testfile.txt' 키가 있는 객체와 연결된 모든 태그를 제거합니다.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -  
Select '^Key'
```

출력:

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target  
"testfile.txt".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y  
testfile.txt
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [DeleteObjectTagging](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 S3 버킷에서 'testfile.txt' 키가 있는 객체와 연결된 모든 태그를 제거합니다.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -  
Select '^Key'
```

출력:

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target  
"testfile.txt".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y  
testfile.txt
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [DeleteObjectTagging](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteObjects** 사용

다음 코드 예시는 DeleteObjects의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [기본 사항 알아보기](#)
- [버킷의 모든 객체 삭제](#)
- [객체 스토리지 시작하기](#)

.NET

SDK for .NET(v4)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Delete all of the objects stored in an existing Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket from which the
/// contents will be deleted.</param>
/// <returns>A boolean value that represents the success or failure of
/// deleting all of the objects in the bucket.</returns>
public async Task<bool> DeleteBucketContentsAsync(string bucketName)
{
    // Iterate over the contents of the bucket and delete all objects.
    try
```



```

    {
        // Delete all objects in the bucket.
        var deleteList = await ListBucketContentsAsync(bucketName, false);
        if (deleteList != null && deleteList.Any())
        {
            await _amazonS3.DeleteObjectsAsync(new DeleteObjectsRequest()
            {
                BucketName = bucketName,
                Objects = deleteList.Select(o => new KeyVersion { Key =
o.Key }).ToList(),
            });
        }

        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error deleting objects: {ex.Message}");
        return false;
    }
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [DeleteObjects](#)를 참조하세요.

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버저닝되지 않은 S3 버킷에서 여러 개의 객체를 삭제합니다.

```

using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>

```

```

/// This example shows how to delete multiple objects from an Amazon Simple
/// Storage Service (Amazon S3) bucket.
/// </summary>
public class DeleteMultipleObjects
{
    /// <summary>
    /// The Main method initializes the Amazon S3 client and the name of
    /// the bucket and then passes those values to MultiObjectDeleteAsync.
    /// </summary>
    public static async Task Main()
    {
        const string bucketName = "amzn-s3-demo-bucket";

        // If the Amazon S3 bucket from which you wish to delete objects is
        // located in the same AWS Region as the default user, define the
        // AWS Region for the Amazon S3 bucket as a parameter to the client
        // constructor.
        IAmazonS3 s3Client = new AmazonS3Client();

        await MultiObjectDeleteAsync(s3Client, bucketName);
    }

    /// <summary>
    /// This method uses the passed Amazon S3 client to first create and then
    /// delete three files from the named bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
    /// Amazon S3 methods.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket where
    /// objects
    /// will be created and then deleted.</param>
    public static async Task MultiObjectDeleteAsync(IAmazonS3 client, string
    bucketName)
    {
        // Create three sample objects which we will then delete.
        var keysAndVersions = await PutObjectsAsync(client, 3, bucketName);

        // Now perform the multi-object delete, passing the key names and
        // version IDs. Since we are working with a non-versioned bucket,
        // the object keys collection includes null version IDs.
        DeleteObjectsRequest multiObjectDeleteRequest = new
        DeleteObjectsRequest

```

```

        {
            BucketName = bucketName,
            Objects = keysAndVersions,
        };

        // You can add a specific object key to the delete request using the
        // AddKey method of the multiObjectDeleteRequest.
        try
        {
            DeleteObjectsResponse response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
            Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
        }
        catch (DeleteObjectsException e)
        {
            PrintDeletionErrorStatus(e);
        }
    }

    /// <summary>
    /// Prints the list of errors raised by the call to DeleteObjectsAsync.
    /// </summary>
    /// <param name="ex">A collection of exceptions returned by the call to
    /// DeleteObjectsAsync.</param>
    public static void PrintDeletionErrorStatus(DeleteObjectsException ex)
    {
        DeleteObjectsResponse errorResponse = ex.Response;
        Console.WriteLine("x {0}", errorResponse.DeletedObjects.Count);

        Console.WriteLine($"Successfully deleted
{errorResponse.DeletedObjects.Count}.");
        Console.WriteLine($"No. of objects failed to delete =
{errorResponse.DeleteErrors.Count}");

        Console.WriteLine("Printing error data...");
        foreach (DeleteError deleteError in errorResponse.DeleteErrors)
        {
            Console.WriteLine($"Object Key:
{deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
        }
    }

    /// <summary>

```

```

    /// This method creates simple text file objects that can be used in
    /// the delete method.
    /// </summary>
    /// <param name="client">The Amazon S3 client used to call
PutObjectAsync.</param>
    /// <param name="number">The number of objects to create.</param>
    /// <param name="bucketName">The name of the bucket where the objects
    /// will be created.</param>
    /// <returns>A list of keys (object keys) and versions that the calling
    /// method will use to delete the newly created files.</returns>
    public static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
client, int number, string bucketName)
    {
        List<KeyVersion> keys = new List<KeyVersion>();
        for (int i = 0; i < number; i++)
        {
            string key = "ExampleObject-" + new System.Random().Next();
            PutObjectRequest request = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = key,
                ContentBody = "This is the content body!",
            };

            PutObjectResponse response = await
client.PutObjectAsync(request);

            // For non-versioned bucket operations, we only need the
            // object key.
            KeyVersion keyVersion = new KeyVersion
            {
                Key = key,
            };
            keys.Add(keyVersion);
        }

        return keys;
    }
}

```

버저닝된 S3 버킷에서 여러 개의 객체를 삭제합니다.

```

using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to delete objects in a version-enabled Amazon
/// Simple StorageService (Amazon S3) bucket.
/// </summary>
public class DeleteMultipleObjects
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";

        // If the AWS Region for your Amazon S3 bucket is different from
        // the AWS Region of the default user, define the AWS Region for
        // the Amazon S3 bucket and pass it to the client constructor
        // like this:
        // RegionEndpoint bucketRegion = RegionEndpoint.USSouth1;
        IAmazonS3 s3Client;

        s3Client = new AmazonS3Client();
        await DeleteMultipleObjectsFromVersionedBucketAsync(s3Client,
bucketName);
    }

    /// <summary>
    /// This method removes multiple versions and objects from a
    /// version-enabled Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
    /// RemoveDeleteMarkersAsync.</param>
    /// <param name="bucketName">The name of the bucket from which to delete
    /// objects.</param>
    public static async Task
DeleteMultipleObjectsFromVersionedBucketAsync(IAmazonS3 client, string
bucketName)
    {
        // Delete objects (specifying object version in the request).

```

```

        await DeleteObjectVersionsAsync(client, bucketName);

        // Delete objects (without specifying object version in the request).
        var deletedObjects = await DeleteObjectsAsync(client, bucketName);

        // Additional exercise - remove the delete markers Amazon S3 returned
from
        // the preceding response. This results in the objects reappearing
        // in the bucket (you can verify the appearance/disappearance of
        // objects in the console).
        await RemoveDeleteMarkersAsync(client, bucketName, deletedObjects);
    }

    /// <summary>
    /// Creates and then deletes non-versioned Amazon S3 objects and then
deletes
    /// them again. The method returns a list of the Amazon S3 objects
deleted.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// PutObjectsAsync and NonVersionedDeleteAsync.</param>
    /// <param name="bucketName">The name of the bucket where the objects
    /// will be created and then deleted.</param>
    /// <returns>A list of DeletedObjects.</returns>
    public static async Task<List<DeletedObject>>
DeleteObjectsAsync(IAmazonS3 client, string bucketName)
    {
        // Upload the sample objects.
        var keysAndVersions2 = await PutObjectsAsync(client, bucketName, 3);

        // Delete objects using only keys. Amazon S3 creates a delete marker
and
        // returns its version ID in the response.
        List<DeletedObject> deletedObjects = await
NonVersionedDeleteAsync(client, bucketName, keysAndVersions2);
        return deletedObjects;
    }

    /// <summary>
    /// This method creates several temporary objects and then deletes them.
    /// </summary>
    /// <param name="client">The S3 client.</param>
    /// <param name="bucketName">Name of the bucket.</param>

```

```

    /// <returns>Async task.</returns>
    public static async Task DeleteObjectVersionsAsync(IAmazonS3 client,
string bucketName)
    {
        // Upload the sample objects.
        var keysAndVersions1 = await PutObjectsAsync(client, bucketName, 3);

        // Delete the specific object versions.
        await VersionedDeleteAsync(client, bucketName, keysAndVersions1);
    }

    /// <summary>
    /// Displays the list of information about deleted files to the console.
    /// </summary>
    /// <param name="e">Error information from the delete process.</param>
    private static void DisplayDeletionErrors(DeleteObjectsException e)
    {
        var errorResponse = e.Response;
        Console.WriteLine($"No. of objects successfully deleted =
{errorResponse.DeletedObjects.Count}");
        Console.WriteLine($"No. of objects failed to delete =
{errorResponse.DeleteErrors.Count}");
        Console.WriteLine("Printing error data...");
        foreach (var deleteError in errorResponse.DeleteErrors)
        {
            Console.WriteLine($"Object Key:
{deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
        }
    }

    /// <summary>
    /// Delete multiple objects from a version-enabled bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
    /// RemoveDeleteMarkersAsync.</param>
    /// <param name="bucketName">The name of the bucket from which to delete
    /// objects.</param>
    /// <param name="keys">A list of key names for the objects to delete.</
param>
    private static async Task VersionedDeleteAsync(IAmazonS3 client, string
bucketName, List<KeyVersion> keys)
    {

```

```

        var multiObjectDeleteRequest = new DeleteObjectsRequest
        {
            BucketName = bucketName,
            Objects = keys, // This includes the object keys and specific
version IDs.
        };

        try
        {
            Console.WriteLine("Executing VersionedDelete...");
            DeleteObjectsResponse response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
            Console.WriteLine($"Successfully deleted all the
{response.DeletedObjects.Count} items");
        }
        catch (DeleteObjectsException ex)
        {
            DisplayDeletionErrors(ex);
        }
    }

    /// <summary>
    /// Deletes multiple objects from a non-versioned Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
    /// RemoveDeleteMarkersAsync.</param>
    /// <param name="bucketName">The name of the bucket from which to delete
    /// objects.</param>
    /// <param name="keys">A list of key names for the objects to delete.</
param>
    /// <returns>A list of the deleted objects.</returns>
    private static async Task<List<DeletedObject>>
NonVersionedDeleteAsync(IAmazonS3 client, string bucketName, List<KeyVersion>
keys)
    {
        // Create a request that includes only the object key names.
        DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest();
        multiObjectDeleteRequest.BucketName = bucketName;

        foreach (var key in keys)
        {

```



```

        multiObjectDeleteRequest.AddKey(key.Key);
    }

    // Execute DeleteObjectsAsync.
    // The DeleteObjectsAsync method adds a delete marker for each
    // object deleted. You can verify that the objects were removed
    // using the Amazon S3 console.
    DeleteObjectsResponse response;
    try
    {
        Console.WriteLine("Executing NonVersionedDelete...");
        response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
        Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
    }
    catch (DeleteObjectsException ex)
    {
        DisplayDeletionErrors(ex);
        throw; // Some deletions failed. Investigate before continuing.
    }

    // This response contains the DeletedObjects list which we use to
delete the delete markers.
    return response.DeletedObjects;
}

/// <summary>
/// Deletes the markers left after deleting the temporary objects.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used to
call
/// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
/// RemoveDeleteMarkersAsync.</param>
/// <param name="bucketName">The name of the bucket from which to delete
/// objects.</param>
/// <param name="deletedObjects">A list of the objects that were
deleted.</param>
private static async Task RemoveDeleteMarkersAsync(IAmazonS3 client,
string bucketName, List<DeletedObject> deletedObjects)
{
    var keyVersionList = new List<KeyVersion>();

    foreach (var deletedObject in deletedObjects)

```

```

        {
            KeyVersion keyVersion = new KeyVersion
            {
                Key = deletedObject.Key,
                VersionId = deletedObject.DeleteMarkerVersionId,
            };
            keyVersionList.Add(keyVersion);
        }

        // Create another request to delete the delete markers.
        var multiObjectDeleteRequest = new DeleteObjectsRequest
        {
            BucketName = bucketName,
            Objects = keyVersionList,
        };

        // Now, delete the delete marker to bring your objects back to the
        bucket.
        try
        {
            Console.WriteLine("Removing the delete markers .....");
            var deleteObjectResponse = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
            Console.WriteLine($"Successfully deleted the
{deleteObjectResponse.DeletedObjects.Count} delete markers");
        }
        catch (DeleteObjectsException ex)
        {
            DisplayDeletionErrors(ex);
        }
    }

    /// <summary>
    /// Create temporary Amazon S3 objects to show how object deletion works
in an
    /// Amazon S3 bucket with versioning enabled.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// PutObjectAsync to create temporary objects for the example.</param>
    /// <param name="bucketName">A string representing the name of the S3
    /// bucket where we will create the temporary objects.</param>
    /// <param name="number">The number of temporary objects to create.</
param>

```

```

    /// <returns>A list of the KeyVersion objects.</returns>
    private static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
client, string bucketName, int number)
    {
        var keys = new List<KeyVersion>();

        for (var i = 0; i < number; i++)
        {
            string key = "ObjectToDelete-" + new System.Random().Next();
            PutObjectRequest request = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = key,
                ContentBody = "This is the content body!",
            };

            var response = await client.PutObjectAsync(request);
            KeyVersion keyVersion = new KeyVersion
            {
                Key = key,
                VersionId = response.VersionId,
            };

            keys.Add(keyVersion);
        }

        return keys;
    }
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [DeleteObjects](#)를 참조하십시오.

Bash

AWS CLIBash 스크립트와 함께 사용

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.
#     $2 - A list of keys in the bucket to delete.

# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_items_in_bucket() {
    local bucket_name=$1
    local keys=$2
    local response

    # Create the JSON for the items to delete.
    local delete_items
    delete_items="{\"Objects\":["
    for key in $keys; do
        delete_items="$delete_items{\"Key\": \"$key\"},"
    done
    delete_items=${delete_items%?} # Remove the final comma.
    delete_items="$delete_items]}"

    response=$(aws s3api delete-objects \
        --bucket "$bucket_name" \
        --delete "$delete_items")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
```

```

    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n
$response"
    return 1
fi
}

```

- API 세부 정보는 AWS CLI 명령 참조의 [DeleteObjects](#)를 참조하십시오.

C++

SDK for C++

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::S3::deleteObjects(const std::vector<Aws::String> &objectKeys,
                               const Aws::String &fromBucket,
                               const Aws::S3::S3ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectsRequest request;

    Aws::S3::Model::Delete deleteObject;
    for (const Aws::String &objectKey: objectKeys) {
        deleteObject.AddObjects(Aws::S3::Model::ObjectIdentifier().WithKey(objectKey));
    }

    request.SetDelete(deleteObject);
    request.SetBucket(fromBucket);

    Aws::S3::Model::DeleteObjectsOutcome outcome =
        client.DeleteObjects(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error deleting objects. " <<

```

```

        err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "Successfully deleted the objects.";
        for (size_t i = 0; i < objectKeys.size(); ++i) {
            std::cout << objectKeys[i];
            if (i < objectKeys.size() - 1) {
                std::cout << ", ";
            }
        }

        std::cout << " from bucket " << fromBucket << "." << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteObjects](#)를 참조하세요.

CLI

AWS CLI

다음 명령은 이름이 amzn-s3-demo-bucket인 버킷에서 객체를 삭제합니다.

```
aws s3api delete-objects --bucket amzn-s3-demo-bucket --delete file://delete.json
```

delete.json은 삭제할 객체를 지정하는 현재 디렉터리의 JSON 문서입니다.

```

{
  "Objects": [
    {
      "Key": "test1.txt"
    }
  ],
  "Quiet": false
}

```

출력:

```
{
```

```

    "Deleted": [
      {
        "DeleteMarkerVersionId": "mYAT5Mc6F7aeUL8SS7FAAqUP01koHwzU",
        "Key": "test1.txt",
        "DeleteMarker": true
      }
    ]
  }
}

```

- API 세부 정보는 AWS CLI 명령 참조의 [DeleteObjects](#)를 참조하세요.

Go

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

```

```
// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
[]types.ObjectIdentifier, bypassGovernance bool) error {
    if len(objects) == 0 {
        return nil
    }

    input := s3.DeleteObjectsInput{
        Bucket: aws.String(bucket),
        Delete: &types.Delete{
            Objects: objects,
            Quiet:   aws.Bool(true),
        },
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
    if err != nil || len(delOut.Errors) > 0 {
        log.Printf("Error deleting objects from bucket %s.\n", bucket)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucket)
                err = noBucket
            }
        }
        } else if len(delOut.Errors) > 0 {
            for _, outErr := range delOut.Errors {
                log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
            }
            err = fmt.Errorf("%s", *delOut.Errors[0].Message)
        }
    } else {
        for _, delObjs := range delOut.Deleted {
            err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
                ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: delObjs.Key},
                time.Minute)
            if err != nil {
                log.Printf("Failed attempt to wait for object %s to be deleted.\n",
                    *delObjs.Key)
            } else {

```



```

    log.Printf("Deleted %s.\n", *delObjs.Key)
  }
}
return err
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteObjects](#)를 참조하십시오.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

```

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

            Usage:    <bucketName>

            Where:
                bucketName - the Amazon S3 bucket name.
            "";

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

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteBucketObjects(s3, bucketName);
        s3.close();
    }

    /**
     * Deletes multiple objects from an Amazon S3 bucket.
     *
     * @param s3 An Amazon S3 client object.
     * @param bucketName The name of the Amazon S3 bucket to delete objects from.
     */
    public static void deleteBucketObjects(S3Client s3, String bucketName) {
        // Upload three sample objects to the specfied Amazon S3 bucket.
        ArrayList<ObjectIdentifier> keys = new ArrayList<>();
        PutObjectRequest putOb;
        ObjectIdentifier objectId;

        for (int i = 0; i < 3; i++) {
            String keyName = "delete object example " + i;
            objectId = ObjectIdentifier.builder()
                .key(keyName)
                .build();
        }
    }
}
```

```
        putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        s3.putObject(putOb, RequestBody.fromString(keyName));
        keys.add(objectId);
    }

    System.out.println(keys.size() + " objects successfully created.");

    // Delete multiple objects in one request.
    Delete del = Delete.builder()
        .objects(keys)
        .build();

    try {
        DeleteObjectsRequest multiObjectDeleteRequest =
DeleteObjectsRequest.builder()
            .bucket(bucketName)
            .delete(del)
            .build();

        s3.deleteObjects(multiObjectDeleteRequest);
        System.out.println("Multiple objects are deleted!");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [DeleteObjects](#)를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

여러 객체를 삭제합니다.

```
import {
  DeleteObjectsCommand,
  S3Client,
  S3ServiceException,
  waitUntilObjectNotExists,
} from "@aws-sdk/client-s3";

/**
 * Delete multiple objects from an S3 bucket.
 * @param {{ bucketName: string, keys: string[] }}
 */
export const main = async ({ bucketName, keys }) => {
  const client = new S3Client({});

  try {
    const { Deleted } = await client.send(
      new DeleteObjectsCommand({
        Bucket: bucketName,
        Delete: {
          Objects: keys.map((k) => ({ Key: k })),
        },
      }),
    );
    for (const key in keys) {
      await waitUntilObjectNotExists(
        { client },
        { Bucket: bucketName, Key: key },
      );
    }
    console.log(
```

```

        `Successfully deleted ${Deleted.length} objects from S3 bucket. Deleted
objects:`,
    );
    console.log(Deleted.map((d) => ` • ${d.Key}`).join("\n"));
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while deleting objects from ${bucketName}. The bucket
doesn't exist.`,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while deleting objects from ${bucketName}.
${caught.name}: ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};

```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 [DeleteObjects](#)를 참조하십시오.

Kotlin

SDK for Kotlin

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

suspend fun deleteBucketObjects(
    bucketName: String,
    objectName: String,
) {
    val objectId =

```

```

        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}

```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 [DeleteObjects](#)를 참조하십시오.

PHP

SDK for PHP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

키 목록에서 객체 세트를 삭제합니다.

```

$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $objects = [];
    foreach ($contents['Contents'] as $content) {
        $objects[] = [

```

```

        'Key' => $content['Key'],
    ];
}
$this->s3client->deleteObjects([
    'Bucket' => $this->bucketName,
    'Delete' => [
        'Objects' => $objects,
    ],
]);
$check = $this->s3client->listObjectsV2([
    'Bucket' => $this->bucketName,
]);
if (isset($check['Contents']) && count($check['Contents']) > 0) {
    throw new Exception("Bucket wasn't empty.");
}
echo "Deleted all objects and folders from $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to delete $fileName from $this->bucketName with error:
" . $exception->getMessage();
    exit("Please fix error with object deletion before continuing.");
}

```

- API 세부 정보는 AWS SDK for PHP API 참조의 [DeleteObjects](#)를 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 "test-files" 버킷에서 "sample.txt" 객체를 제거합니다. 명령이 실행되기 전에 확인 프롬프트가 표시됩니다. 확인 프롬프트를 차단하려면 -Force 스위치를 사용하세요.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

예시 2: 이 명령은 버킷이 객체 버전을 활성화하도록 구성된 경우 "test-files" 버킷에서 지정된 버전의 "sample.txt" 객체를 제거합니다.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId
HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

예시 3: 이 명령은 단일 배치 작업으로 "test-files" 버킷에서 "sample1.txt", "sample2.txt" 및 "sample3.txt" 객체를 제거합니다. 서비스 응답에는 삭제의 성공 또는 오류 상태에 관계없이 처리된 모든 키가 나열됩니다. 서비스에서 처리하지 못한 키에 대한 오류만 가져오려면 -ReportErrorsOnly 파라미터를 추가합니다. 이 파라미터는 -Quiet라는 별칭으로 지정할 수도 있습니다.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",
"sample2.txt", "sample3.txt" )
```

예시 4: 이 예시는 -keyCollection 파라미터와 함께 인라인 표현식을 사용하여 삭제할 객체의 키를 가져옵니다. Get-S3Object는 Amazon.S3.Model.S3Object 인스턴스의 컬렉션을 반환하며, 각 인스턴스에는 객체를 식별하는 유형 문자열의 키 멤버가 있습니다.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object
"test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

예시 5: 이 예시는 버킷에서 키 접두사 "prefix/subprefix"가 있는 모든 객체를 가져와 삭제합니다. 들어오는 객체는 한 번에 하나씩 처리됩니다. 대규모 컬렉션의 경우, 컬렉션을 cmdlet의 -InputObject(별칭 -S3ObjectCollection) 파라미터에 전달하여 서비스를 한 번 직접적으로 호출함으로써 일괄 삭제가 이루어지도록 하는 것이 좋습니다.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |
Remove-S3Object -Force
```

예시 6: 이 예시는 삭제 마커를 나타내는 Amazon.S3.Model.S3ObjectVersion 인스턴스 컬렉션을 cmdlet으로 파이프하여 삭제합니다. 들어오는 객체는 한 번에 하나씩 처리됩니다. 대규모 컬렉션의 경우, 컬렉션을 cmdlet의 -InputObject(별칭 -S3ObjectCollection) 파라미터에 전달하여 서비스를 한 번 직접적으로 호출함으로써 일괄 삭제가 이루어지도록 하는 것이 좋습니다.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where
{$_ .IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

예시 7: 이 스크립트는 -KeyAndVersionCollection 파라미터와 함께 사용할 객체의 배열을 구성하여 객체 집합(이 경우에는 삭제 마커)을 일괄 삭제하는 방법을 보여 줍니다.

```
$keyVersions = @()
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where
{$_ .IsDeleteMarker -eq "True"}
```



```
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =
    $marker.VersionId } }
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -
Force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [DeleteObjects](#)를 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 "test-files" 버킷에서 "sample.txt" 객체를 제거합니다. 명령이 실행되기 전에 확인 프롬프트가 표시됩니다. 확인 프롬프트를 차단하려면 -Force 스위치를 사용하세요.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

예시 2: 이 명령은 버킷이 객체 버전을 활성화하도록 구성된 경우 "test-files" 버킷에서 지정된 버전의 "sample.txt" 객체를 제거합니다.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId
HLbxxn6V9omT6AQYVpks8mmFKQcejpqt
```

예시 3: 이 명령은 단일 배치 작업으로 "test-files" 버킷에서 "sample1.txt", "sample2.txt" 및 "sample3.txt" 객체를 제거합니다. 서비스 응답에는 삭제의 성공 또는 오류 상태에 관계없이 처리된 모든 키가 나열됩니다. 서비스에서 처리하지 못한 키에 대한 오류만 가져오려면 -ReportErrorsOnly 파라미터를 추가합니다. 이 파라미터는 -Quiet라는 별칭으로 지정할 수도 있습니다.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",
    "sample2.txt", "sample3.txt" )
```

예시 4: 이 예시는 -keyCollection 파라미터와 함께 인라인 표현식을 사용하여 삭제할 객체의 키를 가져옵니다. Get-S3Object는 Amazon.S3.Model.S3Object 인스턴스의 컬렉션을 반환하며, 각 인스턴스에는 객체를 식별하는 유형 문자열의 키 멤버가 있습니다.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object
    "test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

예시 5: 이 예시는 버킷에서 키 접두사 "prefix/subprefix"가 있는 모든 객체를 가져와 삭제합니다. 들어오는 객체는 한 번에 하나씩 처리됩니다. 대규모 컬렉션의 경우, 컬렉션을 cmdlet의 -InputObject(별칭 -S3ObjectCollection) 파라미터에 전달하여 서비스를 한 번 직접적으로 호출함으로써 일괄 삭제가 이루어지도록 하는 것이 좋습니다.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |
Remove-S3Object -Force
```

예시 6: 이 예시는 삭제 마커를 나타내는 Amazon.S3.Model.S3ObjectVersion 인스턴스 컬렉션을 cmdlet으로 파이프하여 삭제합니다. 들어오는 객체는 한 번에 하나씩 처리됩니다. 대규모 컬렉션의 경우, 컬렉션을 cmdlet의 -InputObject(별칭 -S3ObjectCollection) 파라미터에 전달하여 서비스를 한 번 직접적으로 호출함으로써 일괄 삭제가 이루어지도록 하는 것이 좋습니다.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where
{$_ .IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

예시 7: 이 스크립트는 -KeyAndVersionCollection 파라미터와 함께 사용할 객체의 배열을 구성하여 객체 집합(이 경우에는 삭제 마커)을 일괄 삭제하는 방법을 보여 줍니다.

```
$keyVersions = @()
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where
{$_ .IsDeleteMarker -eq "True"}
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =
$marker.VersionId } }
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -
Force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [DeleteObjects](#)를 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

객체 키 목록을 사용하여 객체 세트를 삭제합니다.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
```

```

        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                        that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    @staticmethod
    def delete_objects(bucket, object_keys):
        """
        Removes a list of objects from a bucket.
        This operation is done as a batch in a single request.

        :param bucket: The bucket that contains the objects. This is a Boto3
Bucket
                        resource.
        :param object_keys: The list of keys that identify the objects to remove.
        :return: The response that contains data about which objects were deleted
                and any that could not be deleted.
        """
        try:
            response = bucket.delete_objects(
                Delete={"Objects": [{"Key": key} for key in object_keys]}
            )
            if "Deleted" in response:
                logger.info(
                    "Deleted objects '%s' from bucket '%s'.",
                    [del_obj["Key"] for del_obj in response["Deleted"]],
                    bucket.name,
                )
            if "Errors" in response:
                logger.warning(
                    "Could not delete objects '%s' from bucket '%s'.",
                    [
                        f"{del_obj['Key']}: {del_obj['Code']}"
                        for del_obj in response["Errors"]
                    ],
                    bucket.name,
                )
        except ClientError:
            logger.exception("Couldn't delete any objects from bucket %s.",
bucket.name)

```

```

        raise
    else:
        return response

```

버킷의 모든 객체를 삭제합니다.

```

class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    @staticmethod
    def empty_bucket(bucket):
        """
        Remove all objects from a bucket.

        :param bucket: The bucket to empty. This is a Boto3 Bucket resource.
        """
        try:
            bucket.objects.delete()
            logger.info("Emptied bucket '%s'.", bucket.name)
        except ClientError:
            logger.exception("Couldn't empty bucket '%s'.", bucket.name)
            raise

```

모든 버전을 삭제하여 버전이 지정된 객체를 영구적으로 삭제합니다.

```

def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

```

Usage is shown in the `usage_demo_single_object` function at the end of this module.

```
:param bucket: The bucket that contains the object.
:param object_key: The object to delete.
"""
try:
    bucket.object_versions.filter(Prefix=object_key).delete()
    logger.info("Permanently deleted all versions of object %s.", object_key)
except ClientError:
    logger.exception("Couldn't delete all versions of %s.", object_key)
    raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [DeleteObjects](#)를 참조하십시오.

Ruby

SDK for Ruby

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?")
  answer = gets.chomp.downcase
  if answer == 'y'
    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
```

```
puts("\t#{e.code}: #{e.message}")
raise
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 [DeleteObjects](#)를 참조하십시오.

Rust

SDK for Rust

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// Delete the objects in a bucket.
pub async fn delete_objects(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    objects_to_delete: Vec<String>,
) -> Result<(), S3ExampleError> {
    // Push into a mut vector to use '?' early return errors while building
    // object keys.
    let mut delete_object_ids: Vec<aws_sdk_s3::types::ObjectIdentifier> = vec![];
    for obj in objects_to_delete {
        let obj_id = aws_sdk_s3::types::ObjectIdentifier::builder()
            .key(obj)
            .build()
            .map_err(|err| {
                S3ExampleError::new(format!("Failed to build key for
delete_object: {err:?}"))
            })?;
        delete_object_ids.push(obj_id);
    }

    client
        .delete_objects()
        .bucket(bucket_name)
        .delete(
            aws_sdk_s3::types::Delete::builder()
```

```

        .set_objects(Some(delete_object_ids))
        .build()
        .map_err(|err| {
            S3ExampleError::new(format!("Failed to build delete_object
input {err:?}"))
        })?,
    )
    .send()
    .await?;
    Ok(())
}

```

- API 세부 정보는 AWS SDK for Rust API 참조의 [DeleteObjects](#)를 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

TRY.
    oo_result = lo_s3->deleteobjects(          " oo_result is returned for
testing purposes. "
        iv_bucket = iv_bucket_name
        io_delete = NEW /aws1/cl_s3_delete( it_objects = it_object_keys ) ).
    MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- API 세부 정보는 [AWS SDK for SAP ABAP API 참조](#)의 DeleteObjects를 참조하세요.

Swift

SDK for Swift

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSS3

public func deleteObjects(bucket: String, keys: [String]) async throws {
    let input = DeleteObjectsInput(
        bucket: bucket,
        delete: S3ClientTypes.Delete(
            objects: keys.map { S3ClientTypes.ObjectIdentifier(key: $0) },
            quiet: true
        )
    )

    do {
        _ = try await client.deleteObjects(input: input)
    } catch {
        print("ERROR: deleteObjects:", dump(error))
        throw error
    }
}
```

- API 세부 정보는 [Swift용 AWS SDK API 참조](#)의 DeleteObjects를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 DeletePublicAccessBlock 사용

다음 코드 예시는 DeletePublicAccessBlock의 사용 방법을 보여 줍니다.

CLI

AWS CLI

버킷의 퍼블릭 액세스 차단 구성을 삭제하려면

다음 `delete-public-access-block` 예시에서는 지정된 버킷에서 퍼블릭 액세스 차단 구성을 제거합니다.

```
aws s3api delete-public-access-block \
  --bucket amzn-s3-demo-bucket
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 참조의 [DeletePublicAccessBlock](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 버킷의 퍼블릭 액세스 차단 설정을 끕니다.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select
'^BucketName'
```

출력:

```
amzn-s3-demo-bucket
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [DeletePublicAccessBlock](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 버킷의 퍼블릭 액세스 차단 설정을 끕니다.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select
'^BucketName'
```

출력:

```
amzn-s3-demo-bucket
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [DeletePublicAccessBlock](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 **GetBucketAccelerateConfiguration** 사용

다음 코드 예시는 GetBucketAccelerateConfiguration의 사용 방법을 보여 줍니다.

CLI

AWS CLI

버킷의 가속화 구성을 검색하려면

다음 get-bucket-accelerate-configuration 예시에서는 지정된 버킷에 대한 가속화 구성을 가져옵니다.

```
aws s3api get-bucket-accelerate-configuration \
  --bucket amzn-s3-demo-bucket
```

출력:

```
{
  "Status": "Enabled"
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [GetBucketAccelerateConfiguration](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 버킷에 대해 전송 가속화 설정이 활성화된 경우 Enabled 값을 반환합니다.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

출력:

```
Value
-----
Enabled
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [GetBucketAccelerateConfiguration](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 버킷에 대해 전송 가속화 설정이 활성화된 경우 Enabled 값을 반환합니다.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

출력:

```
Value
-----
Enabled
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [GetBucketAccelerateConfiguration](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetBucketAc1** 사용

다음 코드 예시는 GetBucketAc1의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [버킷이 존재하는지 확인](#)
- [액세스 제어 목록\(ACL\) 관리](#)

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Get the access control list (ACL) for the new bucket.
/// </summary>
/// <param name="client">The initialized client object used to get the
/// access control list (ACL) of the bucket.</param>
/// <param name="newBucketName">The name of the newly created bucket.</
param>
/// <returns>An S3AccessControlList.</returns>
public static async Task<S3AccessControlList>
GetACLForBucketAsync(IAmazonS3 client, string newBucketName)
{
    // Retrieve bucket ACL to show that the ACL was properly applied to
    // the new bucket.
    GetACLResponse getACLResponse = await client.GetACLAsync(new
GetACLRequest
    {
        BucketName = newBucketName,
    });

    return getACLResponse.AccessControlList;
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 [GetBucketAcl](#)을 참조하십시오.

C++

SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::S3::getBucketAcl(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketAclRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketAclOutcome outcome =
        s3Client.GetBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketAcl: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
            outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            std::cout << "For bucket " << bucketName << ": "
                      << std::endl << std::endl;

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:          "
                          << getGranteeTypeString(grantee.GetType()) <<
std::endl;
            }
        }
    }
}
```

```

        if (grantee.DisplayNameHasBeenSet()) {
            std::cout << "Display name: "
                      << grantee.GetDisplayName() << std::endl;
        }

        if (grantee.EmailAddressHasBeenSet()) {
            std::cout << "Email address: "
                      << grantee.GetEmailAddress() << std::endl;
        }

        if (grantee.IDHasBeenSet()) {
            std::cout << "ID: "
                      << grantee.GetID() << std::endl;
        }

        if (grantee.URIHasBeenSet()) {
            std::cout << "URI: "
                      << grantee.GetURI() << std::endl;
        }

        std::cout << "Permission: " <<
                  getPermissionString(grant.GetPermission()) <<
                  std::endl << std::endl;
    }
}

return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable
string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string.
 */

Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:

```

```

        return "Predefined Amazon S3 group";
    case Aws::S3::Model::Type::NOT_SET:
        return "Not set";
    default:
        return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable
    string.
    /*!
        \param permission: Permission enumeration.
        \return String: Human-readable string.
    */

    Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
        switch (permission) {
            case Aws::S3::Model::Permission::FULL_CONTROL:
                return "Can list objects in this bucket, create/overwrite/delete "
                    "objects in this bucket, and read/write this "
                    "bucket's permissions";
            case Aws::S3::Model::Permission::NOT_SET:
                return "Permission not set";
            case Aws::S3::Model::Permission::READ:
                return "Can list objects in this bucket";
            case Aws::S3::Model::Permission::READ_ACP:
                return "Can read this bucket's permissions";
            case Aws::S3::Model::Permission::WRITE:
                return "Can create, overwrite, and delete objects in this bucket";
            case Aws::S3::Model::Permission::WRITE_ACP:
                return "Can write this bucket's permissions";
            default:
                return "Permission unknown";
        }

        return "Permission unknown";
    }
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetBucketAcl](#)을 참조하세요.

CLI

AWS CLI

다음 명령은 이름이 `amzn-s3-demo-bucket`인 버킷의 액세스 제어 목록을 검색합니다.

```
aws s3api get-bucket-acl --bucket amzn-s3-demo-bucket
```

출력:

```
{
  "Owner": {
    "DisplayName": "my-username",
    "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
  },
  "Grants": [
    {
      "Grantee": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
      },
      "Permission": "FULL_CONTROL"
    }
  ]
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [GetBucketAcl](#)을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.services.s3.model.S3Exception;
```



```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAclRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAclResponse;
import software.amazon.awssdk.services.s3.model.Grant;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

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

            Usage:
                <bucketName> <objectKey>

            Where:
                bucketName - The Amazon S3 bucket to get the access control list
(ACL) for.
                objectKey - The object to get the ACL for.\s
            "";

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

        String bucketName = args[0];
        String objectKey = args[1];
        System.out.println("Retrieving ACL for object: " + objectKey);
        System.out.println("in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();
```

```
        getBucketACL(s3, objectKey, bucketName);
        s3.close();
        System.out.println("Done!");
    }

    /**
     * Retrieves the Access Control List (ACL) for an object in an Amazon S3
     * bucket.
     *
     * @param s3 The S3Client object used to interact with the Amazon S3 service.
     * @param objectKey The key of the object for which the ACL is to be
     * retrieved.
     * @param bucketName The name of the bucket containing the object.
     * @return The ID of the grantee who has permission on the object, or an
     * empty string if an error occurs.
     */
    public static String getBucketACL(S3Client s3, String objectKey, String
    bucketName) {
        try {
            GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
            List<Grant> grants = aclRes.grants();
            String grantee = "";
            for (Grant grant : grants) {
                System.out.format("  %s: %s\n", grant.grantee().id(),
                grant.permission());
                grantee = grant.grantee().id();
            }

            return grantee;
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [GetBucketAcl](#)을 참조하십시오.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

ACL 권한을 가져옵니다.

```
import {
  GetBucketAclCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Retrieves the Access Control List (ACL) for an S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetBucketAclCommand({
        Bucket: bucketName,
      }),
    );
    console.log(`ACL for bucket "${bucketName}":`);
    console.log(JSON.stringify(response, null, 2));
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
```

```

        console.error(
            `Error from S3 while getting ACL for ${bucketName}. The bucket doesn't
            exist.` ,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while getting ACL for ${bucketName}. ${caught.name}:
            ${caught.message}` ,
        );
    } else {
        throw caught;
    }
}
};

```

- 자세한 정보는 [AWS SDK for JavaScript 개발자 안내서](#)를 참조하세요.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 [GetBucketAcl](#)을 참조하세요.

PowerShell

Tools for PowerShell V5

예제 1: 이 명령에서는 S3 객체의 객체 소유자 세부 정보를 가져옵니다.

```
(Get-S3BucketACL -BucketName 'amzn-s3-demo-bucket' -Select *).Owner
```

출력:

```

DisplayName Id
----- --
testusername
9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [GetBucketAcl](#)을 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def get_acl(self):
        """
        Get the ACL of the bucket.

        :return: The ACL of the bucket.
        """
        try:
            acl = self.bucket.Acl()
            logger.info(
                "Got ACL for bucket %s. Owner is %s.", self.bucket.name,
                acl.owner
            )
        except ClientError:
            logger.exception("Couldn't get ACL for bucket %s.", self.bucket.name)
            raise
        else:
            return acl
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [GetBucketAcl](#)를 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.  
    oo_result = lo_s3->getbucketacl(          " oo_result is returned for  
testing purposes. "  
    iv_bucket = iv_bucket_name ).  
    MESSAGE 'Retrieved bucket ACL.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- API에 대한 세부 정보는 AWS SDK for SAP ABAP API 참조의 [GetBucketAcl](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 **GetBucketAnalyticsConfiguration** 사용

다음 코드 예시는 GetBucketAnalyticsConfiguration의 사용 방법을 보여 줍니다.

CLI

AWS CLI

특정 ID를 가진 버킷의 분석 구성을 검색하려면

다음 get-bucket-analytics-configuration 예시에서는 지정된 버킷 및 ID에 대한 분석 구성을 표시합니다.

```
aws s3api get-bucket-analytics-configuration \
  --bucket amzn-s3-demo-bucket \
  --id 1
```

출력:

```
{
  "AnalyticsConfiguration": {
    "StorageClassAnalysis": {},
    "Id": "1"
  }
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [GetBucketAnalyticsConfiguration](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷에서 이름이 'testfilter'인 분석 필터의 세부 정보를 반환합니다.

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId
'testfilter'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [GetBucketAnalyticsConfiguration](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 S3 버킷에서 이름이 'testfilter'인 분석 필터의 세부 정보를 반환합니다.

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId
'testfilter'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [GetBucketAnalyticsConfiguration](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetBucketCors** 사용

다음 코드 예시는 GetBucketCors의 사용 방법을 보여 줍니다.

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Retrieve the CORS configuration applied to the Amazon S3 bucket.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used
/// to retrieve the CORS configuration.</param>
/// <returns>The created CORS configuration object.</returns>
private static async Task<CORSConfiguration>
RetrieveCORSConfigurationAsync(AmazonS3Client client)
{
    GetCORSConfigurationRequest request = new
GetCORSConfigurationRequest()
    {
        BucketName = BucketName,
    };
    var response = await client.GetCORSConfigurationAsync(request);
    var configuration = response.Configuration;
    PrintCORSRules(configuration);
    return configuration;
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 [GetBucketCors](#)를 참조하세요.

CLI

AWS CLI

다음 명령은 이름이 `amzn-s3-demo-bucket`인 버킷에 대한 Cross-Origin Resource Sharing 구성을 검색합니다.

```
aws s3api get-bucket-cors --bucket amzn-s3-demo-bucket
```

출력:

```
{
  "CORSRules": [
    {
      "AllowedHeaders": [
        "*"
      ],
      "ExposeHeaders": [
        "x-amz-server-side-encryption"
      ],
      "AllowedMethods": [
        "PUT",
        "POST",
        "DELETE"
      ],
      "MaxAgeSeconds": 3000,
      "AllowedOrigins": [
        "http://www.example.com"
      ]
    },
    {
      "AllowedHeaders": [
        "Authorization"
      ],
      "MaxAgeSeconds": 3000,
      "AllowedMethods": [
        "GET"
      ],
      "AllowedOrigins": [
        "*"
      ]
    }
  ]
}
```

```
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [GetBucketCors](#)를 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷에 대한 CORS 정책을 가져옵니다.

```
import {
  GetBucketCorsCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Log the Cross-Origin Resource Sharing (CORS) configuration information
 * set for the bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new GetBucketCorsCommand({
    Bucket: bucketName,
  });

  try {
    const { CORSRules } = await client.send(command);
    console.log(JSON.stringify(CORSRules));
    CORSRules.forEach((cr, i) => {
      console.log(
        `\nCORSRule ${i + 1}`,
        `\n${"-"}.repeat(10)`,
        `\nAllowedHeaders: ${cr.AllowedHeaders}`,
        `\nAllowedMethods: ${cr.AllowedMethods}`,
      );
    });
  } catch (err) {
    if (err instanceof S3ServiceException) {
      console.log(`S3ServiceException: ${err.message}`);
    } else {
      console.log(`Error: ${err.message}`);
    }
  }
};
```

```

        '\nAllowedOrigins: ${cr.AllowedOrigins}`',
        '\nExposeHeaders: ${cr.ExposeHeaders}`',
        '\nMaxAgeSeconds: ${cr.MaxAgeSeconds}`',
    );
});
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while getting bucket CORS rules for ${bucketName}. The
            bucket doesn't exist.`
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while getting bucket CORS rules for ${bucketName}.
            ${caught.name}: ${caught.message}`
        );
    } else {
        throw caught;
    }
}
};

```

- 자세한 정보는 [AWS SDK for JavaScript 개발자 안내서](#)를 참조하세요.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 [GetBucketCors](#)를 참조하십시오.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""

```

```

def __init__(self, bucket):
    """
    :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
                    that wraps bucket actions in a class-like structure.
    """
    self.bucket = bucket
    self.name = bucket.name

def get_cors(self):
    """
    Get the CORS rules for the bucket.

    :return The CORS rules for the specified bucket.
    """
    try:
        cors = self.bucket.Cors()
        logger.info(
            "Got CORS rules %s for bucket '%s'.", cors.cors_rules,
self.bucket.name
        )
    except ClientError:
        logger.exception(("Couldn't get CORS for bucket %s.",
self.bucket.name))
        raise
    else:
        return cors

```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [GetBucketCors](#)를 참조하십시오.

Ruby

SDK for Ruby

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
  an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Gets the CORS configuration of a bucket.
  #
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS
  configuration for the bucket.
  def cors
    @bucket_cors.data
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
  why: #{e.message}"
    nil
  end
end

end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 [GetBucketCors](#)를 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

TRY.

```

        oo_result = lo_s3->getbucketcors(
            " oo_result is returned for
testing purposes. "
        iv_bucket = iv_bucket_name ).
        MESSAGE 'Retrieved bucket CORS configuration.' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.

```

- API 세부 정보는 [AWS SDK for SAP ABAP API 참조](#)의 GetBucketCors를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetBucketEncryption** 사용

다음 코드 예시는 GetBucketEncryption의 사용 방법을 보여 줍니다.

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

/// <summary>
/// Get and print the encryption settings of a bucket.
/// </summary>
/// <param name="bucketName">Name of the bucket.</param>
/// <returns>Async task.</returns>
public static async Task GetEncryptionSettings(string bucketName)
{
    // Check and print the bucket encryption settings.
    Console.WriteLine($"Getting encryption settings for bucket
{bucketName}.");

    try

```

```

    {
        var settings =
            await _s3Client.GetBucketEncryptionAsync(
                new GetBucketEncryptionRequest() { BucketName =
bucketName });

        foreach (var encryptionSettings in
settings?.ServerSideEncryptionConfiguration?.ServerSideEncryptionRules!)
        {
            Console.WriteLine(
                $"{Algorithm:
{encryptionSettings.ServerSideEncryptionByDefault.ServerSideEncryptionAlgorithm}");
            Console.WriteLine(
                $"{Key:
{encryptionSettings.ServerSideEncryptionByDefault.ServerSideEncryptionKeyManagementService
}
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine(ex.ErrorCode == "InvalidBucketName"
                ? $"Bucket {bucketName} was not found."
                : $"Unable to get bucket encryption for bucket {bucketName},
{ex.Message}");
        }
    }
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [GetBucketEncryption](#)을 참조하세요.

CLI

AWS CLI

버킷의 서버 측 암호화 구성 가져오기

다음 `get-bucket-encryption` 예시에서는 `amzn-s3-demo-bucket` 버킷의 서버 측 암호화 구성을 가져옵니다.

```
aws s3api get-bucket-encryption \
    --bucket amzn-s3-demo-bucket
```

출력:

```
{
  "ServerSideEncryptionConfiguration": {
    "Rules": [
      {
        "ApplyServerSideEncryptionByDefault": {
          "SSEAlgorithm": "AES256"
        }
      }
    ]
  }
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [GetBucketEncryption](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 버킷과 연결된 모든 서버 측 암호화 규칙을 반환합니다.

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [GetBucketEncryption](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 버킷과 연결된 모든 서버 측 암호화 규칙을 반환합니다.

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [GetBucketEncryption](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 **GetBucketInventoryConfiguration** 사용

다음 코드 예시는 GetBucketInventoryConfiguration의 사용 방법을 보여 줍니다.

CLI

AWS CLI

버킷의 인벤토리 구성을 검색하려면

다음 `get-bucket-inventory-configuration` 예시에서는 지정된 버킷에 대해 ID가 1인 인벤토리 구성을 가져옵니다.

```
aws s3api get-bucket-inventory-configuration \
  --bucket amzn-s3-demo-bucket \
  --id 1
```

출력:

```
{
  "InventoryConfiguration": {
    "IsEnabled": true,
    "Destination": {
      "S3BucketDestination": {
        "Format": "ORC",
        "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
        "AccountId": "123456789012"
      }
    },
    "IncludedObjectVersions": "Current",
    "Id": "1",
    "Schedule": {
      "Frequency": "Weekly"
    }
  }
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [GetBucketInventoryConfiguration](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷에 대해 이름이 'testinventory'인 인벤토리의 세부 정보를 반환합니다.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId  
'testinventory'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [GetBucketInventoryConfiguration](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 S3 버킷에 대해 이름이 'testinventory'인 인벤토리의 세부 정보를 반환합니다.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId  
'testinventory'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [GetBucketInventoryConfiguration](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetBucketLifecycleConfiguration** 사용

다음 코드 예시는 GetBucketLifecycleConfiguration의 사용 방법을 보여 줍니다.

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>  
/// Returns a configuration object for the supplied bucket name.  
/// </summary>
```

```

    /// <param name="client">The S3 client object used to call
    /// the GetLifecycleConfigurationAsync method.</param>
    /// <param name="bucketName">The name of the S3 bucket for which a
    /// configuration will be created.</param>
    /// <returns>Returns a new LifecycleConfiguration object.</returns>
    public static async Task<LifecycleConfiguration>
    RetrieveLifecycleConfigAsync(IAmazonS3 client, string bucketName)
    {
        var request = new GetLifecycleConfigurationRequest()
        {
            BucketName = bucketName,
        };
        var response = await client.GetLifecycleConfigurationAsync(request);
        var configuration = response.Configuration;
        return configuration;
    }

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [GetBucketLifecycleConfiguration](#)을 참조하세요.

CLI

AWS CLI

다음 명령은 이름이 amzn-s3-demo-bucket인 버킷의 수명 주기 구성을 검색합니다.

```
aws s3api get-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket
```

출력:

```

{
  "Rules": [
    {
      "ID": "Move rotated logs to Glacier",
      "Prefix": "rotated/",
      "Status": "Enabled",
      "Transitions": [
        {
          "Date": "2015-11-10T00:00:00.000Z",
          "StorageClass": "GLACIER"
        }
      ]
    }
  ]
}

```

```

        }
    ]
},
{
    "Status": "Enabled",
    "Prefix": "",
    "NoncurrentVersionTransitions": [
        {
            "NoncurrentDays": 0,
            "StorageClass": "GLACIER"
        }
    ],
    "ID": "Move old versions to Glacier"
}
]
}

```

- API 세부 정보는 AWS CLI 명령 참조의 [GetBucketLifecycleConfiguration](#)을 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

```

```
def get_lifecycle_configuration(self):
    """
    Get the lifecycle configuration of the bucket.

    :return: The lifecycle rules of the specified bucket.
    """
    try:
        config = self.bucket.LifecycleConfiguration()
        logger.info(
            "Got lifecycle rules %s for bucket '%s'.",
            config.rules,
            self.bucket.name,
        )
    except:
        logger.exception(
            "Couldn't get lifecycle rules for bucket '%s'.", self.bucket.name
        )
        raise
    else:
        return config.rules
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [GetBucketLifecycleConfiguration](#)를 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.
    oo_result = lo_s3->getbucketlifecycleconf(          " oo_result is
returned for testing purposes. "
    iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved bucket lifecycle configuration.' TYPE 'I'.
```

```
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 [GetBucketLifecycleConfiguration](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetBucketLocation** 사용

다음 코드 예시는 GetBucketLocation의 사용 방법을 보여 줍니다.

CLI

AWS CLI

다음 명령은 amzn-s3-demo-bucket이라는 버킷의 위치 제약 조건을 가져옵니다(제약 조건이 있는 경우).

```
aws s3api get-bucket-location --bucket amzn-s3-demo-bucket
```

출력:

```
{  
    "LocationConstraint": "us-west-2"  
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [GetBucketLocation](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예제 1: 이 명령에서는 제약 조건이 있는 경우 'amzn-s3-demo-bucket' 버킷에 대한 위치 제약 조건을 반환합니다.

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

출력:

```
Value
-----
ap-south-1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [GetBucketLocation](#)을 참조하세요.

Tools for PowerShell V5

예제 1: 이 명령에서는 제약 조건이 있는 경우 'amzn-s3-demo-bucket' 버킷에 대한 위치 제약 조건을 반환합니다.

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

출력:

```
Value
-----
ap-south-1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [GetBucketLocation](#)을 참조하세요.

Rust

SDK for Rust

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn show_buckets(
```

```

    strict: bool,
    client: &Client,
    region: BucketLocationConstraint,
) -> Result<(), S3ExampleError> {
    let mut buckets = client.list_buckets().into_paginator().send();

    let mut num_buckets = 0;
    let mut in_region = 0;

    while let Some(Ok(output)) = buckets.next().await {
        for bucket in output.buckets() {
            num_buckets += 1;
            if strict {
                let r = client
                    .get_bucket_location()
                    .bucket(bucket.name().unwrap_or_default())
                    .send()
                    .await?;

                if r.location_constraint() == Some(&region) {
                    println!("{}", bucket.name().unwrap_or_default());
                    in_region += 1;
                }
            } else {
                println!("{}", bucket.name().unwrap_or_default());
            }
        }
    }

    println!();
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            in_region, region, num_buckets
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}

```

- API 세부 정보는 Rust용 AWS SDK API 참조의 [GetBucketLocation](#)을 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 **GetBucketLogging** 사용

다음 코드 예시는 GetBucketLogging의 사용 방법을 보여 줍니다.

CLI

AWS CLI

버킷의 로깅 상태를 검색하려면

다음 get-bucket-logging 예시에서는 지정된 버킷의 로깅 상태를 가져옵니다.

```
aws s3api get-bucket-logging \
  --bucket amzn-s3-demo-bucket
```

출력:

```
{
  "LoggingEnabled": {
    "TargetPrefix": "",
    "TargetBucket": "amzn-s3-demo-bucket-logs"
  }
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [GetBucketLogging](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 버킷의 로깅 상태를 반환합니다.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

출력:

```
TargetBucketName  Grants TargetPrefix
-----
```

```
testbucket1      {}      testprefix
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [GetBucketLogging](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 버킷의 로깅 상태를 반환합니다.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

출력:

```
TargetBucketName  Grants TargetPrefix
-----
testbucket1      {}      testprefix
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [GetBucketLogging](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 **GetBucketMetricsConfiguration** 사용

다음 코드 예시는 GetBucketMetricsConfiguration의 사용 방법을 보여 줍니다.

CLI

AWS CLI

특정 ID를 가진 버킷의 지표 구성을 검색하려면

다음 get-bucket-metrics-configuration 예시에서는 지정된 버킷 및 ID의 지표 구성을 표시합니다.

```
aws s3api get-bucket-metrics-configuration \
  --bucket amzn-s3-demo-bucket \
  --id 123
```

출력:

```
{
  "MetricsConfiguration": {
    "Filter": {
      "Prefix": "logs"
    },
    "Id": "123"
  }
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [GetBucketMetricsConfiguration](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷의 'testfilter'라는 지표 필터에 대한 세부 정보를 반환합니다.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
'testfilter'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [GetBucketMetricsConfiguration](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 S3 버킷의 'testfilter'라는 지표 필터에 대한 세부 정보를 반환합니다.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
'testfilter'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [GetBucketMetricsConfiguration](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 **GetBucketNotification** 사용

다음 코드 예시는 GetBucketNotification의 사용 방법을 보여 줍니다.

CLI

AWS CLI

다음 명령은 이름이 `amzn-s3-demo-bucket`인 버킷의 알림 구성을 검색합니다.

```
aws s3api get-bucket-notification --bucket amzn-s3-demo-bucket
```

출력:

```
{
  "TopicConfiguration": {
    "Topic": "arn:aws:sns:us-west-2:123456789012:my-notification-topic",
    "Id": "YmQzMmEwM2EjZWVlI0NGItNzVtZjI1MC00ZjgyLWZDBiZWNl",
    "Event": "s3:ObjectCreated:*",
    "Events": [
      "s3:ObjectCreated:*"
    ]
  }
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [GetBucketNotification](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 예시는 지정된 버킷의 알림 구성을 검색합니다

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty
TopicConfigurations
```

출력:

```
Id    Topic
--    -
mimo  arn:aws:sns:eu-west-1:123456789012:topic-1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [GetBucketNotification](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 예시는 지정된 버킷의 알림 구성을 검색합니다

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty
TopicConfigurations
```

출력:

```
Id      Topic
--      -
mimo    arn:aws:sns:eu-west-1:123456789012:topic-1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [GetBucketNotification](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetBucketPolicy** 사용

다음 코드 예시는 GetBucketPolicy의 사용 방법을 보여 줍니다.

C++

SDK for C++

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::S3::getBucketPolicy(const Aws::String &bucketName,
                                const Aws::S3::S3ClientConfiguration
                                &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketPolicyRequest request;
```

```

    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketPolicyOutcome outcome =
        s3Client.GetBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketPolicy: "
            << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        Aws::StringStream policy_stream;
        Aws::String line;

        outcome.GetResult().GetPolicy() >> line;
        policy_stream << line;

        std::cout << "Retrieve the policy for bucket '" << bucketName << "':\n\n"
<<
            policy_stream.str() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetBucketPolicy](#)를 참조하세요.

CLI

AWS CLI

다음 명령은 이름이 amzn-s3-demo-bucket인 버킷의 버킷 정책을 검색합니다.

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket
```

출력:

```

{
    "Policy": "{\"Version\":\"2008-10-17\",\"Statement\": [{\"Sid\":\"\", \"Effect\
\": \"Allow\", \"Principal\": \"*\", \"Action\": \"s3:GetObject\", \"Resource\":
\"arn:aws:s3:::amzn-s3-demo-bucket/*\"}, {\"Sid\":\"\", \"Effect\": \"Deny\",

```

```
\\"Principal\\":\\"*\\",\\"Action\\":\\"s3:GetObject\\",\\"Resource\\":\\"arn:aws:s3:::amzn-s3-demo-bucket/secret/*\\"}]}"
}
```

버킷 정책 가져오기 및 넣기 다음 예시에서는 Amazon S3 버킷 정책을 다운로드하고 파일을 수정한 다음 put-bucket-policy를 사용하여 수정된 버킷 정책을 적용하는 방법을 보여줍니다. 버킷 정책을 파일로 다운로드하려면 다음을 실행할 수 있습니다.

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket --query Policy --output text > policy.json
```

그런 다음 필요에 따라 policy.json 파일을 수정할 수 있습니다. 마지막으로 다음을 실행하여 수정된 정책을 S3 버킷에 다시 적용할 수 있습니다.

policy.json 파일을 실행하여 수정된 정책을 S3 버킷에 다시 적용할 수 있습니다. 마지막으로 다음을 실행하여 수정된 정책을 S3 버킷에 다시 적용할 수 있습니다.

파일을 실행하여 수정된 정책을 S3 버킷에 다시 적용할 수 있습니다. 마지막으로 다음을 실행하여 수정된 정책을 S3 버킷에 다시 적용할 수 있습니다.

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://policy.json
```

- API 세부 정보는 AWS CLI 명령 참조의 [GetBucketPolicy](#)를 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
```

```
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

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

            Usage:
                <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to get the policy from.
            "";

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

        String bucketName = args[0];
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        String polText = getPolicy(s3, bucketName);
        System.out.println("Policy Text: " + polText);
        s3.close();
    }

    /**
     * Retrieves the policy for the specified Amazon S3 bucket.
     *
     * @param s3 the {@link S3Client} instance to use for making the request
     */
}
```



```

    * @param bucketName the name of the S3 bucket for which to retrieve the
    policy
    * @return the policy text for the specified bucket, or an empty string if an
    error occurs
    */
    public static String getPolicy(S3Client s3, String bucketName) {
        String policyText;
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
            .bucket(bucketName)
            .build();

        try {
            GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
            policyText = policyRes.policy();
            return policyText;

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [GetBucketPolicy](#)를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷 정책을 가져옵니다.

```
import {
```

```

    GetBucketPolicyCommand,
    S3Client,
    S3ServiceException,
  } from "@aws-sdk/client-s3";

/**
 * Logs the policy for a specified bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { Policy } = await client.send(
      new GetBucketPolicyCommand({
        Bucket: bucketName,
      }),
    );
    console.log(`Policy for "${bucketName}":\n${Policy}`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting policy from ${bucketName}. The bucket
        doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting policy from ${bucketName}.  ${caught.name}:
        ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

```

- 자세한 정보는 [AWS SDK for JavaScript 개발자 안내서](#)를 참조하세요.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 [GetBucketPolicy](#)를 참조하십시오.

Kotlin

SDK for Kotlin

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getPolicy(bucketName: String): String? {
    println("Getting policy for bucket $bucketName")

    val request =
        GetBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val policyRes = s3.getBucketPolicy(request)
        return policyRes.policy
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 [GetBucketPolicy](#)를 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷과 연결된 버킷 정책을 출력합니다.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [GetBucketPolicy](#)를 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 S3 버킷과 연결된 버킷 정책을 출력합니다.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [GetBucketPolicy](#)를 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def get_policy(self):
        """
        Get the security policy of the bucket.

        :return: The security policy of the specified bucket, in JSON format.
        """
        try:
            policy = self.bucket.Policy()
            logger.info(
                "Got policy %s for bucket '%s'.", policy.policy, self.bucket.name
            )
        except ClientError:
```

```

        logger.exception("Couldn't get policy for bucket '%s'.",
self.bucket.name)
        raise
    else:
        return json.loads(policy.policy)

```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [GetBucketPolicy](#)를 참조하십시오.

Ruby

SDK for Ruby

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
  configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Gets the policy of a bucket.
  #
  # @return [Aws::S3::GetBucketPolicyOutput, nil] The current bucket policy.
  def policy
    policy = @bucket_policy.data.policy
    policy.respond_to?(:read) ? policy.read : policy
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get the policy for #{@bucket_policy.bucket.name}. Here's why:
#{e.message}"
    nil
  end
end

```

```
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 [GetBucketPolicy](#)를 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.
    oo_result = lo_s3->getbucketpolicy(          " oo_result is returned for
testing purposes. "
    iv_bucket = iv_bucket_name ).
    DATA(lv_policy) = oo_result->get_policy( ).
    MESSAGE 'Retrieved bucket policy.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- API 세부 정보는 [AWS SDK for SAP ABAP API 참조](#)의 [GetBucketPolicy](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 **GetBucketPolicyStatus** 사용

다음 코드 예시는 GetBucketPolicyStatus의 사용 방법을 보여 줍니다.

CLI

AWS CLI

특정 버킷이 퍼블릭인지 나타내는 버킷 정책 상태를 검색하려면

다음 `get-bucket-policy-status` 예시에서는 버킷 `amzn-s3-demo-bucket`의 정책 상태를 가져옵니다.

```
aws s3api get-bucket-policy-status \
  --bucket amzn-s3-demo-bucket
```

출력:

```
{
  "PolicyStatus": {
    "IsPublic": false
  }
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [GetBucketPolicyStatus](#)를 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 버킷이 퍼블릭 버킷인지 여부를 나타내는 지정된 S3 버킷에 대한 정책 상태를 반환합니다.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [GetBucketPolicyStatus](#)를 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 버킷이 퍼블릭 버킷인지 여부를 나타내는 지정된 S3 버킷에 대한 정책 상태를 반환합니다.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [GetBucketPolicyStatus](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetBucketReplication** 사용

다음 코드 예시는 GetBucketReplication의 사용 방법을 보여 줍니다.

CLI

AWS CLI

다음 명령은 이름이 amzn-s3-demo-bucket인 버킷의 복제 구성을 검색합니다.

```
aws s3api get-bucket-replication --bucket amzn-s3-demo-bucket
```

출력:

```
{
  "ReplicationConfiguration": {
    "Rules": [
      {
        "Status": "Enabled",
        "Prefix": "",
        "Destination": {
          "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket-backup",
          "StorageClass": "STANDARD"
        },
        "ID": "ZmUwNzE4ZmQ4tMjVhOS00MTlkLOGI4NDkzZTIWJjNTUtYTA1"
      }
    ],
    "Role": "arn:aws:iam::123456789012:role/s3-replication-role"
  }
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [GetBucketReplication](#)을 참조하세요.

Java

SDK for Java 2.x

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Retrieves the replication details for the specified S3 bucket.
 *
 * @param s3Client      the S3 client used to interact with the S3
service
 * @param sourceBucketName the name of the S3 bucket to retrieve the
replication details for
 *
 * @throws S3Exception if there is an error retrieving the replication
details
 */
public static void getReplicationDetails(S3Client s3Client, String
sourceBucketName) {
    GetBucketReplicationRequest getRequest =
GetBucketReplicationRequest.builder()
        .bucket(sourceBucketName)
        .build();

    try {
        ReplicationConfiguration replicationConfig =
s3Client.getBucketReplication(getRequest).replicationConfiguration();
        ReplicationRule rule = replicationConfig.rules().get(0);
        System.out.println("Retrieved destination bucket: " +
rule.destination().bucket());
        System.out.println("Retrieved priority: " + rule.priority());
        System.out.println("Retrieved source-bucket replication rule status:
" + rule.status());

    } catch (S3Exception e) {
        System.err.println("Failed to retrieve replication details: " +
e.awsErrorDetails().errorMessage());
    }
}
```

```
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [GetBucketReplication](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예제 1: 이름이 'amzn-s3-demo-bucket'인 버킷에 설정된 복제 구성 정보를 반환합니다.

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [GetBucketReplication](#)을 참조하세요.

Tools for PowerShell V5

예제 1: 이름이 'amzn-s3-demo-bucket'인 버킷에 설정된 복제 구성 정보를 반환합니다.

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [GetBucketReplication](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 **GetBucketRequestPayment** 사용

다음 코드 예시는 GetBucketRequestPayment의 사용 방법을 보여 줍니다.

CLI

AWS CLI

버킷의 지불 요청 구성을 검색하려면

다음 get-bucket-request-payment 예시에서는 지정된 버킷의 requester pays 구성을 가져옵니다.

```
aws s3api get-bucket-request-payment \
  --bucket amzn-s3-demo-bucket
```

출력:

```
{
  "Payer": "BucketOwner"
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [GetBucketRequestPayment](#)를 참조하세요.

PowerShell

Tools for PowerShell V4

예제 1: 이름이 'amzn-s3-demo-bucket'인 버킷에 대한 요청 결제 구성을 반환합니다. 기본적으로 버킷에서 다운로드하는 비용은 버킷 소유자가 지불합니다.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [GetBucketRequestPayment](#)를 참조하세요.

Tools for PowerShell V5

예제 1: 이름이 'amzn-s3-demo-bucket'인 버킷에 대한 요청 결제 구성을 반환합니다. 기본적으로 버킷에서 다운로드하는 비용은 버킷 소유자가 지불합니다.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [GetBucketRequestPayment](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 **GetBucketTagging** 사용

다음 코드 예시는 GetBucketTagging의 사용 방법을 보여 줍니다.

CLI

AWS CLI

다음 명령은 이름이 `amzn-s3-demo-bucket`인 버킷에 대한 태그 지정 구성을 검색합니다.

```
aws s3api get-bucket-tagging --bucket amzn-s3-demo-bucket
```

출력:

```
{
  "TagSet": [
    {
      "Value": "marketing",
      "Key": "organization"
    }
  ]
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [GetBucketTagging](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 버킷과 관련된 모든 태그를 반환합니다.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [GetBucketTagging](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 버킷과 관련된 모든 태그를 반환합니다.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [GetBucketTagging](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 **GetBucketVersioning** 사용

다음 코드 예시는 GetBucketVersioning의 사용 방법을 보여 줍니다.

CLI

AWS CLI

다음 명령은 이름이 amzn-s3-demo-bucket인 버킷에 대한 버전 관리 구성을 검색합니다.

```
aws s3api get-bucket-versioning --bucket amzn-s3-demo-bucket
```

출력:

```
{
  "Status": "Enabled"
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [GetBucketVersioning](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 버킷과 관련된 버전 관리 상태를 반환합니다.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [GetBucketVersioning](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 버킷과 관련된 버전 관리 상태를 반환합니다.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [GetBucketVersioning](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetBucketWebsite** 사용

다음 코드 예시는 GetBucketWebsite의 사용 방법을 보여 줍니다.

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Get the website configuration.
GetBucketWebsiteRequest getRequest = new
GetBucketWebsiteRequest()
{
    BucketName = bucketName,
};
GetBucketWebsiteResponse getResponse = await
client.GetBucketWebsiteAsync(getRequest);
Console.WriteLine($"Index document:
{getResponse.WebsiteConfiguration.IndexDocumentSuffix}");
Console.WriteLine($"Error document:
{getResponse.WebsiteConfiguration.ErrorDocument}");
```

- API 세부 정보는 AWS SDK for .NET API 참조의 [GetBucketWebsite](#)를 참조하십시오.

C++

SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::S3::getWebsiteConfig(const Aws::String &bucketName,
                                   const Aws::S3::S3ClientConfiguration
                                   &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketWebsiteOutcome outcome =
        s3Client.GetBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();

        std::cerr << "Error: GetBucketWebsite: "
                   << err.GetMessage() << std::endl;
    } else {
        Aws::S3::Model::GetBucketWebsiteResult websiteResult =
            outcome.GetResult();

        std::cout << "Success: GetBucketWebsite: "
                  << std::endl << std::endl
                  << "For bucket '" << bucketName << "':"
                  << std::endl
                  << "Index page : "
                  << websiteResult.GetIndexDocument().GetSuffix()
                  << std::endl
                  << "Error page: "
                  << websiteResult.GetErrorDocument().GetKey()
                  << std::endl;
    }
}
```

```
    return outcome.IsSuccess();  
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetBucketWebsite](#)를 참조하세요.

CLI

AWS CLI

다음 명령은 이름이 `amzn-s3-demo-bucket`인 버킷의 정적 웹 사이트 구성을 검색합니다.

```
aws s3api get-bucket-website --bucket amzn-s3-demo-bucket
```

출력:

```
{  
  "IndexDocument": {  
    "Suffix": "index.html"  
  },  
  "ErrorDocument": {  
    "Key": "error.html"  
  }  
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [GetBucketWebsite](#)를 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

웹 사이트 구성을 가져옵니다.

```
import {
```



```

    GetBucketWebsiteCommand,
    S3Client,
    S3ServiceException,
  } from "@aws-sdk/client-s3";

/**
 * Log the website configuration for a bucket.
 * @param {{ bucketName }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetBucketWebsiteCommand({
        Bucket: bucketName,
      }),
    );
    console.log(
      `Your bucket is set up to host a website with the following configuration:
\n${JSON.stringify(response, null, 2)}`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchWebsiteConfiguration"
    ) {
      console.error(
        `Error from S3 while getting website configuration for ${bucketName}. The
bucket isn't configured as a website.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting website configuration for ${bucketName}.
${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 [GetBucketWebsite](#)를 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷의 정적 웹 사이트 구성 세부 정보를 반환합니다.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [GetBucketWebsite](#)를 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 S3 버킷의 정적 웹 사이트 구성 세부 정보를 반환합니다.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [GetBucketWebsite](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetObject** 사용

다음 코드 예시는 GetObject의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [기본 사항 알아보기](#)
- [버킷에서 수정된 객체 가져오기](#)
- [다중 리전 액세스 포인트에서 객체 생성](#)
- [암호화 시작하기](#)
- [객체 스토리지 시작하기](#)
- [조건부 요청 수행](#)
- [업로드 및 다운로드 추적](#)

.NET

SDK for .NET(v4)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Shows how to download an object from an Amazon S3 bucket to the
/// local computer.
/// </summary>
/// <param name="bucketName">The name of the bucket where the object is
/// currently stored.</param>
/// <param name="objectName">The name of the object to download.</param>
/// <param name="filePath">The path, including filename, where the
/// downloaded object will be stored.</param>
/// <returns>A boolean value indicating the success or failure of the
/// download process.</returns>
public async Task<bool> DownloadObjectFromBucketAsync(
    string bucketName,
    string objectName,
    string filePath)
{
    var request = new GetObjectRequest
    {
        BucketName = bucketName,
        Key = objectName,
    };

    using GetObjectResponse response = await
        _amazonS3.GetObjectAsync(request);

    try
    {
        // Save object to local file
        await response.WriteResponseStreamToFileAsync($"{filePath}\
        \{objectName}", true, CancellationToken.None);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
}
```

```

    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error saving {objectName}: {ex.Message}");
        return false;
    }
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [GetObject](#)를 참조하세요.

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

조건부 요청을 사용하여 객체를 가져옵니다.

```

/// <summary>
/// Retrieves an object from Amazon S3 with a conditional request.
/// </summary>
/// <param name="objectKey">The key of the object to retrieve.</param>
/// <param name="sourceBucket">The source bucket of the object.</param>
/// <param name="conditionType">The type of condition: 'IfMatch',
'IfNoneMatch', 'IfModifiedSince', 'IfUnmodifiedSince'.</param>
/// <param name="conditionDateValue">The value to use for the condition for
dates.</param>
/// <param name="etagConditionalValue">The value to use for the condition for
etags.</param>
/// <returns>True if the conditional read is successful, False otherwise.</
returns>
public async Task<bool> GetObjectConditional(string objectKey, string
sourceBucket,
    S3ConditionType conditionType, DateTime? conditionDateValue = null,
string? etagConditionalValue = null)
{
    try
    {
        var getObjectRequest = new GetObjectRequest

```

```
        {
            BucketName = sourceBucket,
            Key = objectKey
        };

        switch (conditionType)
        {
            case S3ConditionType.IfMatch:
                getObjectRequest.EtagToMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfNoneMatch:
                getObjectRequest.EtagToNotMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfModifiedSince:
                getObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                break;
            case S3ConditionType.IfUnmodifiedSince:
                getObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                break;
            default:
                throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
        }

        var response = await _amazonS3.GetObjectAsync(getObjectRequest);
        var sampleBytes = new byte[20];
        await response.ResponseStream.ReadAsync(sampleBytes, 0, 20);
        _logger.LogInformation($"Conditional read
successful. Here are the first 20 bytes of the object:
\\n{System.Text.Encoding.UTF8.GetString(sampleBytes)}");
        return true;
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "PreconditionFailed")
        {
            _logger.LogError("Conditional read failed: Precondition failed");
        }
        else if (e.ErrorCode == "NotModified")
        {
            _logger.LogError("Conditional read failed: Object not modified");
        }
    }
}
```

```

        else
        {
            _logger.LogError($"Unexpected error: {e.ErrorCode}");
            throw;
        }
        return false;
    }
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [GetObject](#)를 참조하세요.

Bash

AWS CLIBash 스크립트와 함께 사용

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#     $1 - The name of the bucket to download the object from.
#     $2 - The path and file name to store the downloaded bucket.
#     $3 - The key (name) of the object in the bucket.
#
# Returns:

```

```
#      0 - If successful.
#      1 - If it fails.
#####
function download_object_from_bucket() {
    local bucket_name=$1
    local destination_file_name=$2
    local object_name=$3
    local response

    response=$(aws s3api get-object \
        --bucket "$bucket_name" \
        --key "$object_name" \
        "$destination_file_name")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
        errecho "ERROR: AWS reports put-object operation failed.\n$response"
        return 1
    fi
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [GetObject](#)를 참조하십시오.

C++

SDK for C++

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::S3::getObject(const Aws::String &objectKey,
                           const Aws::String &fromBucket,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(fromBucket);
    request.SetKey(objectKey);
```

```

    Aws::S3::Model::GetObjectOutcome outcome =
        client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
        std::endl;
    } else {
        std::cout << "Successfully retrieved '" << objectKey << "' from '"
            << fromBucket << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetObject](#)를 참조하세요.

CLI

AWS CLI

다음 예시에서는 get-object 명령을 사용하여 Amazon S3에서 객체를 다운로드합니다.

```
aws s3api get-object --bucket text-content --key dir/
my_images.tar.bz2 my_images.tar.bz2
```

참고로 outfile 파라미터는 “--outfile”과 같은 옵션 이름 없이 지정됩니다. 출력 파일의 이름은 명령의 마지막 파라미터여야 합니다.

아래 예시에서는 --range를 사용하여 객체에서 특정 바이트 범위를 다운로드하는 방법을 보여줍니다. 참고로 바이트 범위에는 “bytes=”라는 접두사가 있어야 합니다.

```
aws s3api get-object --bucket text-content --key dir/my_data --
range bytes=8888-9999 my_data_range
```

객체를 가져오는 방법에 대한 자세한 내용은 Amazon S3 개발자 안내서의 객체 가져오기를 참조하세요.

- API 세부 정보는 AWS CLI명령 참조의 [GetObject](#)를 참조하세요.

Go

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)  
// actions  
// used in the examples.  
// It contains S3Client, an Amazon S3 service client that is used to perform  
// bucket  
// and object actions.  
type BucketBasics struct {  
    S3Client *s3.Client  
}  
  
// DownloadFile gets an object from a bucket and stores it in a local file.
```

```
func (basics BucketBasics) DownloadFile(ctx context.Context, bucketName string,
objectKey string, fileName string) error {
    result, err := basics.S3Client.GetObject(ctx, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:    aws.String(objectKey),
    })
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Can't get object %s from bucket %s. No such key exists.\n",
objectKey, bucketName)
            err = noKey
        } else {
            log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
objectKey, err)
        }
        return err
    }
    defer result.Body.Close()
    file, err := os.Create(fileName)
    if err != nil {
        log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
        return err
    }
    defer file.Close()
    body, err := io.ReadAll(result.Body)
    if err != nil {
        log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
err)
    }
    _, err = file.Write(body)
    return err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [GetObject](#)를 참조하십시오.

Java

SDK for Java 2.x

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

[S3Client](#)를 사용하여 바이트 배열로 데이터를 읽습니다.

```
/**
 * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket
 * and writes them to a local file.
 *
 * @param bucketName the name of the S3 bucket containing the object
 * @param keyName    the key (or name) of the S3 object to retrieve
 * @param path       the local file path where the object's bytes will be
 * written
 * @return a {@link CompletableFuture} that completes when the object bytes
 * have been written to the local file
 */
public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
keyName, String path) {
    GetObjectRequest objectRequest = GetObjectRequest.builder()
        .key(keyName)
        .bucket(bucketName)
        .build();

    CompletableFuture<ResponseBytes<GetObjectResponse>> response =
getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
    return response.thenAccept(objectBytes -> {
        try {
            byte[] data = objectBytes.asByteArray();
            Path filePath = Paths.get(path);
            Files.write(filePath, data);
            logger.info("Successfully obtained bytes from an S3 object");
        } catch (IOException ex) {
            throw new RuntimeException("Failed to write data to file", ex);
        }
    }).whenComplete((resp, ex) -> {
```

```

        if (ex != null) {
            throw new RuntimeException("Failed to get object bytes from S3",
ex);
        }
    });
}

```

[S3TransferManager](#)를 사용하여 S3 버킷 내의 [객체를 로컬 파일로 다운로드](#)합니다. [파일 전체](#)를 보고 [테스트](#)합니다.

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.NoSuchFileException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;

    public Long downloadFile(S3TransferManager transferManager, String
bucketName,

                                String key, String downloadedFilePath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
            .getObjectRequest(b -> b.bucket(bucketName).key(key))
            .destination(Paths.get(downloadedFilePath))
            .build();

        FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

```

```

        CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
        logger.info("Content length [{}]",
downloadResult.response().contentType());
        return downloadResult.response().contentType();
    }

```

[S3Client](#)를 사용하여 객체에 속하는 태그를 읽습니다.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tag;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

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

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {

```

```
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String keyName = args[1];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    listTags(s3, bucketName, keyName);
    s3.close();
}

/**
 * Lists the tags associated with an Amazon S3 object.
 *
 * @param s3 the S3Client object used to interact with the Amazon S3 service
 * @param bucketName the name of the S3 bucket that contains the object
 * @param keyName the key (name) of the S3 object
 */
public static void listTags(S3Client s3, String bucketName, String keyName) {
    try {
        GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        GetObjectTaggingResponse tags =
s3.getObjectTagging(getTaggingRequest);
        List<Tag> tagSet = tags.tagSet();
        for (Tag tag : tagSet) {
            System.out.println(tag.key());
            System.out.println(tag.value());
        }

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

[S3Client](#)를 사용하여 객체의 URL을 가져옵니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetUrlRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.net.URL;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

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

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.
                keyName - A key name that represents the object.\s
            """;

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

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
```

```

        .region(region)
        .build();

        getURL(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Retrieves the URL for a specific object in an Amazon S3 bucket.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket where the object is stored
     * @param keyName the name of the object for which the URL should be
     retrieved
     * @throws S3Exception if there is an error retrieving the URL for the
     specified object
     */
    public static void getURL(S3Client s3, String bucketName, String keyName) {
        try {
            GetUrlRequest request = GetUrlRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            URL url = s3.utilities().getUrl(request);
            System.out.println("The URL for " + keyName + " is " + url);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

[S3Client](#)를 사용한 S3Presigner 클라이언트 객체를 사용하여 객체를 가져옵니다.

```

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;

```



```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import
    software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectPresignedUrl {
    public static void main(String[] args) {
        final String USAGE = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents a text file.\s
            """;

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

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Presigner presigner = S3Presigner.builder()
            .region(region)
            .build();
```

```

        getPresignedUrl(presigner, bucketName, keyName);
        presigner.close();
    }

    /**
     * Generates a pre-signed URL for an Amazon S3 object.
     *
     * @param presigner The {@link S3Presigner} instance to use for generating
     the pre-signed URL.
     * @param bucketName The name of the Amazon S3 bucket where the object is
     stored.
     * @param keyName The key name (file name) of the object in the Amazon S3
     bucket.
     *
     * @throws S3Exception If there is an error interacting with the Amazon S3
     service.
     * @throws IOException If there is an error opening the HTTP connection or
     reading/writing the request/response.
     */
    public static void getPresignedUrl(S3Presigner presigner, String bucketName,
    String keyName) {
        try {
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            GetObjectPresignRequest getObjectPresignRequest =
            GetObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(60))
                .getObjectRequest(getObjectRequest)
                .build();

            PresignedGetObjectRequest presignedGetObjectRequest =
            presigner.presignGetObject(getObjectPresignRequest);
            String theUrl = presignedGetObjectRequest.url().toString();
            System.out.println("Presigned URL: " + theUrl);
            HttpURLConnection connection = (HttpURLConnection)
            presignedGetObjectRequest.url().openConnection();
            presignedGetObjectRequest.httpRequest().headers().forEach((header,
            values) -> {
                values.forEach(value -> {
                    connection.setRequestProperty(header, value);
                });
            });
        }
    }

```

```

    });

    // Send any request payload that the service needs (not needed when
    // isBrowserExecutable is true).
    if (presignedGetObjectRequest.signedPayload().isPresent()) {
        connection.setDoOutput(true);

        try (InputStream signedPayload =
presignedGetObjectRequest.signedPayload().get().asInputStream();
            OutputStream httpOutputStream =
connection.getOutputStream()) {
            IoUtils.copy(signedPayload, httpOutputStream);
        }
    }

    // Download the result of executing the request.
    try (InputStream content = connection.getInputStream()) {
        System.out.println("Service returned response: ");
        IoUtils.copy(content, System.out);
    }

    } catch (S3Exception | IOException e) {
        e.printStackTrace();
    }
}
}
}

```

ResponseTransformer 객체와 [S3Client](#)를 사용하여 객체를 가져옵니다.

```

import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

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

            Usage:
                <bucketName> <keyName> <path>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                path - The path where the file is written to.\s
            "";

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

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getObjectBytes(s3, bucketName, keyName, path);
        s3.close();
    }

    /**
     * Retrieves the bytes of an object stored in an Amazon S3 bucket and saves
     * them to a local file.
     */
}
```

```

    *
    * @param s3 The S3Client instance used to interact with the Amazon S3
    service.
    * @param bucketName The name of the S3 bucket where the object is stored.
    * @param keyName The key (or name) of the S3 object.
    * @param path The local file path where the object's bytes will be saved.
    * @throws IOException If an I/O error occurs while writing the bytes to the
    local file.
    * @throws S3Exception If an error occurs while retrieving the object from
    the S3 bucket.
    */
    public static void getObjectBytes(S3Client s3, String bucketName, String
    keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
            s3.getObject(objectRequest, ResponseTransformer.toBytes());
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [GetObject](#)를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

객체를 다운로드합니다.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    // The Body object also has 'transformToArray' and 'transformToWebStream'
    methods.
    const str = await response.Body.transformToString();
    console.log(str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
        such key exists.`
      );
    }
  }
}
```

```

    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while getting object from ${bucketName}. ${caught.name}:
            ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};

```

객체의 ETag가 제공된 값과 일치하는 경우에만 조건부로 객체를 다운로드합니다.

```

import {
    GetObjectCommand,
    NoSuchKey,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
    const client = new S3Client({});

    try {
        const response = await client.send(
            new GetObjectCommand({
                Bucket: bucketName,
                Key: key,
                IfMatch: eTag,
            }),
        );
        // The Body object also has 'transformToByteArray' and 'transformToWebStream'
        methods.
        const str = await response.Body.transformToString();
        console.log("Success. Here is text of the file:", str);
    } catch (caught) {
        if (caught instanceof NoSuchKey) {

```

```
        console.error(
            `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.`
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`
        );
    } else {
        throw caught;
    }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
    isMain,
    validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
    const options = {
        bucketName: {
            type: "string",
            required: true,
        },
        key: {
            type: "string",
            required: true,
        },
        eTag: {
            type: "string",
            required: true,
        },
    };
};

const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
    const { errors, results } = loadArgs();
```



```

    if (!errors) {
      main(results.values);
    } else {
      console.error(errors.join("\n"));
    }
  }
}

```

객체의 ETag가 제공된 값과 일치하지 않을 경우에만 조건부로 객체를 다운로드합니다.

```

import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfNoneMatch: eTag,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
    methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
        such key exists.`
      );
    }
  }
}

```

```
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}. ${caught.name}:
        ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
};

const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

```
}
}
```

지정된 기간에 생성되거나 수정된 객체만 조건부로 다운로드합니다.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const date = new Date();
  date.setDate(date.getDate() - 1);
  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfModifiedSince: date,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
    methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
        such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
```

```

        `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
        ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
    isMain,
    validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
    const options = {
        bucketName: {
            type: "string",
            required: true,
        },
        key: {
            type: "string",
            required: true,
        },
    };
};

const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
    const { errors, results } = loadArgs();
    if (!errors) {
        main(results.values);
    } else {
        console.error(errors.join("\n"));
    }
}
}

```

지정된 기간에 생성되거나 수정되지 않은 객체만 조건부로 다운로드합니다.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const date = new Date();
  date.setDate(date.getDate() - 1);
  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfUnmodifiedSince: date,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
    methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
}
```

```
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
};

const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- 자세한 정보는 [AWS SDK for JavaScript 개발자 안내서](#)를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 [GetObject](#)를 참조하십시오.

Kotlin

SDK for Kotlin

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getObjectBytes(  
    bucketName: String,  
    keyName: String,  
    path: String,  
) {  
    val request =  
        GetObjectRequest {  
            key = keyName  
            bucket = bucketName  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.getObject(request) { resp ->  
            val myFile = File(path)  
            resp.body?.writeToFile(myFile)  
            println("Successfully read $keyName from $bucketName")  
        }  
    }  
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 [GetObject](#)를 참조하십시오.

PHP

SDK for PHP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

객체를 가져옵니다.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $file = $this->s3client->getObject([
        'Bucket' => $this->bucketName,
        'Key' => $fileName,
    ]);
    $body = $file->get('Body');
    $body->rewind();
    echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
} catch (Exception $exception) {
    echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
    exit("Please fix error with file downloading before continuing.");
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 [GetObject](#)를 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 'amzn-s3-demo-bucket' 버킷에서 'sample.txt' 항목을 검색하여 현재 위치의 'local-sample.txt'라는 파일에 저장합니다. 이 명령을 직접 호출하기 전에 "local-sample.txt" 파일이 없어도 됩니다.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-
sample.txt
```


예시 2: 이 명령은 'amzn-s3-demo-bucket' 버킷에서 가상 디렉터리 'DIR'을 검색하여 현재 위치의 'Local-DIR'이라는 폴더에 저장합니다. 이 명령을 직접 호출하기 전에 "Local-DIR" 폴더가 없어도 됩니다.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

예시 3: 버킷 이름에 'config'가 있는 버킷에서 키가 '.json'으로 끝나는 모든 객체를 지정된 폴더의 파일로 다운로드합니다. 객체 키는 파일 이름을 설정하는 데 사용됩니다.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [GetObject](#)를 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 'amzn-s3-demo-bucket' 버킷에서 'sample.txt' 항목을 검색하여 현재 위치의 'local-sample.txt'라는 파일에 저장합니다. 이 명령을 직접 호출하기 전에 "local-sample.txt" 파일이 없어도 됩니다.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
```

예시 2: 이 명령은 'amzn-s3-demo-bucket' 버킷에서 가상 디렉터리 'DIR'을 검색하여 현재 위치의 'Local-DIR'이라는 폴더에 저장합니다. 이 명령을 직접 호출하기 전에 "Local-DIR" 폴더가 없어도 됩니다.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

예시 3: 버킷 이름에 'config'가 있는 버킷에서 키가 '.json'으로 끝나는 모든 객체를 지정된 폴더의 파일로 다운로드합니다. 객체 키는 파일 이름을 설정하는 데 사용됩니다.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

예시 4: 대용량 파일의 성능을 개선하기 위해 멀티파트 병렬 다운로드를 사용하여 단일 객체를 다운로드합니다.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt -UseMultipartDownload
```

예시 5: 사용자 지정 64MB 파트 크기로 RANGE 모드를 사용하여 대형 객체를 다운로드합니다. RANGE 모드는 HTTP 바이트 범위 요청을 사용하며 업로드 방식에 관계없이 모든 객체에 대해 작동합니다.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -File local-large-file.zip -UseMultipartDownload -MultipartDownloadType RANGE -PartSize 64MB
```

예시 6: 동시 파일 다운로드와 함께 멀티파트 다운로드를 사용하여 키 접두사 'data/'가 있는 모든 객체를 로컬 폴더에 다운로드합니다.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -Folder C:\local-data -UseMultipartDownload -DownloadFilesConcurrently
```

예시 7: 고대역폭 네트워크에서 처리량을 극대화하기 위해 20개 동시 연결로 멀티파트 다운로드를 사용하여 단일 객체를 다운로드합니다.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -File local-large-file.zip -UseMultipartDownload -ConcurrentServiceRequest 20
```

예시 8: 멀티파트 다운로드를 사용하여 키 접두사 'data/'가 있는 모든 객체를 로컬 폴더에 다운로드하며 개별 파일 실패는 중단하는 대신 건너뜁니다.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -Folder C:\local-data -UseMultipartDownload -FailurePolicy ContinueOnFailure
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [GetObject](#)를 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def get(self):
        """
        Gets the object.

        :return: The object data in bytes.
        """
        try:
            body = self.object.get()["Body"].read()
            logger.info(
                "Got object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't get object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
```

```
        raise
    else:
        return body
```

조건부 요청을 사용하여 객체를 가져옵니다.

```
class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def get_object_conditional(
        self,
        object_key: str,
        source_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Retrieves an object from Amazon S3 with a conditional request.

        :param object_key: The key of the object to retrieve.
        :param source_bucket: The source bucket of the object.
        :param condition_type: The type of condition: 'IfMatch', 'IfNoneMatch',
        'IfModifiedSince', 'IfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
            response = self.s3.get_object(
                Bucket=source_bucket,
```

```

        Key=object_key,
        **{condition_type: condition_value},
    )
    sample_bytes = response["Body"].read(20)
    print(
        f"\tConditional read successful. Here are the first 20 bytes of
the object:\n"
    )
    print(f"\t{sample_bytes}")
except ClientError as e:
    error_code = e.response["Error"]["Code"]
    if error_code == "PreconditionFailed":
        print("\tConditional read failed: Precondition failed")
    elif error_code == "304": # Not modified error code.
        print("\tConditional read failed: Object not modified")
    else:
        logger.error(f"Unexpected error: {error_code}")
        raise

```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [GetObject](#)를 참조하세요.

Ruby

SDK for Ruby

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

객체를 가져옵니다.

```

require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.

```

```

def initialize(object)
  @object = object
end

# Gets the object directly to a file.
#
# @param target_path [String] The path to the file where the object is
downloaded.
# @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
successful; otherwise nil.
def get_object(target_path)
  @object.get(response_target: target_path)
rescue Aws::Errors::ServiceError => e
  puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"
  target_path = "my-object-as-file.txt"

  wrapper = ObjectGetWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  obj_data = wrapper.get_object(target_path)
  return unless obj_data

  puts "Object #{object_key} (#{obj_data.content_length} bytes) downloaded to
#{target_path}."
end

run_demo if $PROGRAM_NAME == __FILE__

```

객체를 가져와 서버 측 암호화 상태를 보고합니다.

```

require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.

```

```
def initialize(object)
  @object = object
end

# Gets the object into memory.
#
# @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
successful; otherwise nil.
def object
  @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
    object_key))
  obj_data = wrapper.get_object
  return unless obj_data

  encryption = obj_data.server_side_encryption.nil? ? 'no' :
    obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 [GetObject](#)를 참조하십시오.

Rust

SDK for Rust

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn get_object(client: Client, opt: Opt) -> Result<usize, S3ExampleError> {
    trace!("bucket:      {}", opt.bucket);
    trace!("object:      {}", opt.object);
    trace!("destination: {}", opt.destination.display());

    let mut file = File::create(opt.destination.clone()).map_err(|err| {
        S3ExampleError::new(format!(
            "Failed to initialize file for saving S3 download: {err:?}"
        ))
    })?;

    let mut object = client
        .get_object()
        .bucket(opt.bucket)
        .key(opt.object)
        .send()
        .await?;

    let mut byte_count = 0_usize;
    while let Some(bytes) = object.body.try_next().await.map_err(|err| {
        S3ExampleError::new(format!("Failed to read from S3 download stream:
{err:?}"))
    })? {
        let bytes_len = bytes.len();
        file.write_all(&bytes).map_err(|err| {
            S3ExampleError::new(format!(
                "Failed to write from S3 download stream to local file: {err:?}"
            ))
        })?;
        trace!("Intermediate write of {bytes_len}");
        byte_count += bytes_len;
    }
}
```



```
    Ok(byte_count)
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 [GetObject](#)을 참조하십시오.

SAP ABAP

SDK for SAP ABAP API

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.
    oo_result = lo_s3->getobject(          " oo_result is returned for
testing purposes. "
        iv_bucket = iv_bucket_name
        iv_key = iv_object_key ).
    DATA(lv_object_data) = oo_result->get_body( ).
    MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.
```

- API 세부 정보는 AWSSDK for SAP ABAP API의 [GetObject](#)을 참조하십시오.

Swift

SDK for Swift

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSS3

public func downloadFile(bucket: String, key: String, to: String) async
throws {
    let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)

    let input = GetObjectInput(
        bucket: bucket,
        key: key
    )
    do {
        let output = try await client.getObject(input: input)

        guard let body = output.body else {
            throw HandlerError.getObjectBody("GetObjectInput missing body.")
        }

        guard let data = try await body.readData() else {
            throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
        }

        try data.write(to: fileUrl)
    }
    catch {
        print("ERROR: ", dump(error, name: "Downloading a file. "))
        throw error
    }
}
```

```
import AWSS3
```

```

public func readFile(bucket: String, key: String) async throws -> Data {
    let input = GetObjectInput(
        bucket: bucket,
        key: key
    )
    do {
        let output = try await client.getObject(input: input)

        guard let body = output.body else {
            throw HandlerError.getObjectBody("GetObjectInput missing body.")
        }

        guard let data = try await body.readData() else {
            throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
        }

        return data
    }
    catch {
        print("ERROR: ", dump(error, name: "Reading a file. "))
        throw error
    }
}

```

- API 세부 정보는 AWS SDK for Swift API 참조의 [GetObject](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetObjectAcl** 사용

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

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [액세스 제어 목록\(ACL\) 관리](#)

C++

SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::S3::getObjectAcl(const Aws::String &bucketName,
                              const Aws::String &objectKey,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetObjectAclRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectAclOutcome outcome =
        s3Client.GetObjectAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObjectAcl: "
                    << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
            outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            std::cout << "For object " << objectKey << ": "
                        << std::endl << std::endl;

            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:          "
```

```

        << getGranteeTypeString(grantee.GetType()) <<
std::endl;
    }

    if (grantee.DisplayNameHasBeenSet()) {
        std::cout << "Display name: "
        << grantee.GetDisplayName() << std::endl;
    }

    if (grantee.EmailAddressHasBeenSet()) {
        std::cout << "Email address: "
        << grantee.GetEmailAddress() << std::endl;
    }

    if (grantee.IDHasBeenSet()) {
        std::cout << "ID: "
        << grantee.GetID() << std::endl;
    }

    if (grantee.URIHasBeenSet()) {
        std::cout << "URI: "
        << grantee.GetURI() << std::endl;
    }

    std::cout << "Permission: " <<
        getPermissionString(grant.GetPermission()) <<
        std::endl << std::endl;
    }
}

return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable
string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string
 */
Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:

```

```

        return "Canonical user ID of an AWS account";
    case Aws::S3::Model::Type::Group:
        return "Predefined Amazon S3 group";
    case Aws::S3::Model::Type::NOT_SET:
        return "Not set";
    default:
        return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable
    string.
    /*!
    \param permission: Permission enumeration.
    \return String: Human-readable string
    */
    Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
        switch (permission) {
            case Aws::S3::Model::Permission::FULL_CONTROL:
                return "Can read this object's data and its metadata, "
                    "and read/write this object's permissions";
            case Aws::S3::Model::Permission::NOT_SET:
                return "Permission not set";
            case Aws::S3::Model::Permission::READ:
                return "Can read this object's data and its metadata";
            case Aws::S3::Model::Permission::READ_ACP:
                return "Can read this object's permissions";
            // case Aws::S3::Model::Permission::WRITE // Not applicable.
            case Aws::S3::Model::Permission::WRITE_ACP:
                return "Can write this object's permissions";
            default:
                return "Permission unknown";
        }
    }
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetObjectAcl](#)을 참조하세요.

CLI

AWS CLI

다음 명령은 이름이 `amzn-s3-demo-bucket`인 버킷의 객체에 대한 액세스 제어 목록을 검색합니다.

```
aws s3api get-object-acl --bucket amzn-s3-demo-bucket --key index.html
```

출력:

```
{
  "Owner": {
    "DisplayName": "my-username",
    "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
  },
  "Grants": [
    {
      "Grantee": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
      },
      "Permission": "FULL_CONTROL"
    },
    {
      "Grantee": {
        "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
      },
      "Permission": "READ"
    }
  ]
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [GetObjectAcl](#)을 참조하세요.

Kotlin

SDK for Kotlin

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getBucketACL(
    objectKey: String,
    bucketName: String,
) {
    val request =
        GetObjectAclRequest {
            bucket = bucketName
            key = objectKey
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 [GetObjectAcl](#)를 참조하세요.

PowerShell

Tools for PowerShell V5

예제 1: 이 명령에서는 S3 객체의 객체 소유자 세부 정보를 가져옵니다.

```
(Get-S3ObjectACL -BucketName 'amzn-s3-demo-bucket' -key 'initialize.ps1' -Select
*).Owner
```

출력:


```

DisplayName Id
-----
testusername
9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [GetObjectAcl](#)을 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                                that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def get_acl(self):
        """
        Gets the ACL of the object.

        :return: The ACL of the object.
        """
        try:
            acl = self.object.Acl()
            logger.info(
                "Got ACL for object %s owned by %s.",
                self.object.key,

```

```

        acl.owner["DisplayName"],
    )
except ClientError:
    logger.exception("Couldn't get ACL for object %s.", self.object.key)
    raise
else:
    return acl

```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [GetObjectAcl](#)를 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

TRY.
    oo_result = lo_s3->getobjectacl(           " oo_result is returned for
testing purposes. "
        iv_bucket = iv_bucket_name
        iv_key = iv_object_key ).
    MESSAGE 'Retrieved object ACL.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 [GetObjectAcl](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetObjectAttributes** 사용

다음 코드 예시는 GetObjectAttributes의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [Amazon S3 객체 무결성 작업](#)

C++

SDK for C++

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object is stored.
    \param key: The unique identifier (key) of the object within the S3 bucket.
    \param hashMethod: The hashing algorithm used to calculate the hash value of
the object.
    \param[out] hashData: The retrieved hash.
    \param[out] partHashes: The part hashes if available.
    \param client: The S3 client instance used to retrieve the object.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
&key,
                                     AwsDoc::S3::HASH_METHOD hashMethod,
                                     Aws::String &hashData,
                                     std::vector<Aws::String> *partHashes,
                                     const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
```

```

request.SetBucket(bucket);
request.SetKey(key);

if (hashMethod == MD5) {
    Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
    attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
    request.SetObjectAttributes(attributes);

    Aws::S3::Model::GetObjectAttributesOutcome outcome =
client.GetObjectAttributes(
    request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        hashData = result.GetETag();
    } else {
        std::cerr << "Error retrieving object etag attributes." <<
outcome.GetError().GetMessage() << std::endl;
        return false;
    }
} else { // hashMethod != MD5
    Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
    attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
    request.SetObjectAttributes(attributes);

    Aws::S3::Model::GetObjectAttributesOutcome outcome =
client.GetObjectAttributes(
    request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        switch (hashMethod) {
            case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                break; // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
            case AwsDoc::S3::MD5:
                break; // MD5 is not supported.
#pragma clang diagnostic pop
            case AwsDoc::S3::SHA1:
                hashData = result.GetChecksum().GetChecksumSHA1();
                break;
            case AwsDoc::S3::SHA256:
                hashData = result.GetChecksum().GetChecksumSHA256();

```

```

        break;
    case AwsDoc::S3::CRC32:
        hashData = result.GetChecksum().GetChecksumCRC32();
        break;
    case AwsDoc::S3::CRC32C:
        hashData = result.GetChecksum().GetChecksumCRC32C();
        break;
    default:
        std::cerr << "Unknown hash method." << std::endl;
        return false;
    }
} else {
    std::cerr << "Error retrieving object checksum attributes." <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}

if (nullptr != partHashes) {
    attributes.clear();
    attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
    request.SetObjectAttributes(attributes);
    outcome = client.GetObjectAttributes(request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
result.GetObjectParts().GetParts();
        for (const Aws::S3::Model::ObjectPart &part: parts) {
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // Default is not supported.
NOLINT(*-branch-clone)
                    break;
                case AwsDoc::S3::MD5: // MD5 is not supported.
                    break;
                case AwsDoc::S3::SHA1:
                    partHashes->push_back(part.GetChecksumSHA1());
                    break;
                case AwsDoc::S3::SHA256:
                    partHashes->push_back(part.GetChecksumSHA256());
                    break;
                case AwsDoc::S3::CRC32:
                    partHashes->push_back(part.GetChecksumCRC32());
                    break;
                case AwsDoc::S3::CRC32C:

```

```

        partHashes->push_back(part.GetChecksumCRC32C());
        break;
    default:
        std::cerr << "Unknown hash method." << std::endl;
        return false;
    }
}
} else {
    std::cerr << "Error retrieving object attributes for object
parts." <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}
}
}
return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetObjectAttributes](#)를 참조하세요.

CLI

AWS CLI

객체 자체를 반환하지 않고 객체에서 메타데이터를 검색하려면

다음 `get-object-attributes` 예시에서는 객체 `doc1.rtf`에서 메타데이터를 가져옵니다.

```

aws s3api get-object-attributes \
  --bucket amzn-s3-demo-bucket \
  --key doc1.rtf \
  --object-attributes "StorageClass" "ETag" "ObjectSize"

```

출력:

```

{
  "LastModified": "2022-03-15T19:37:31+00:00",
  "VersionId": "IuCPjXTDzHNfldAuitVBIKJpF2p1fg4P",
  "ETag": "b662d79adeb7c8d787ea7eafb9ef6207",
  "StorageClass": "STANDARD",
  "ObjectSize": 405
}

```

```
}
```

자세한 내용은 Amazon S3 API 참조의 [GetObjectAttributes](#)를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [GetObjectAttributes](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetObjectLegalHold** 사용

다음 코드 예시는 GetObjectLegalHold의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [Amazon S3 객체 잠그기](#)

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Get the legal hold details for an S3 object.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The object key.</param>
/// <returns>The object legal hold details.</returns>
public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
    string objectKey)
{
    try
    {
        var request = new GetObjectLegalHoldRequest()
```

```

        {
            BucketName = bucketName,
            Key = objectKey
        };

        var response = await _amazonS3.GetObjectLegalHoldAsync(request);
        Console.WriteLine($"{objectKey} in {bucketName}: " +
            $"{response.LegalHold.Status}");
        return response.LegalHold;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Unable to fetch legal hold: '{ex.Message}'");
        return new ObjectLockLegalHold();
    }
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [GetObjectLegalHold](#)를 참조하세요.

CLI

AWS CLI

객체의 법적 보류 상태 검색

다음 `get-object-legal-hold` 예시에서는 지정된 객체의 법적 보류 상태를 가져옵니다.

```

aws s3api get-object-legal-hold \
  --bucket amzn-s3-demo-bucket-with-object-lock \
  --key doc1.rtf

```

출력:

```

{
  "LegalHold": {
    "Status": "ON"
  }
}

```

- API 세부 정보는 AWS CLI 명령 참조의 [GetObjectLegalHold](#)를 참조하세요.

Go

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}  
  
// GetObjectLegalHold retrieves the legal hold status for an S3 object.  
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key  
    string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {  
    var status *types.ObjectLockLegalHoldStatus  
    input := &s3.GetObjectLegalHoldInput{  
        Bucket:    aws.String(bucket),  
        Key:       aws.String(key),  
        VersionId: aws.String(versionId),
```

```

}

output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
if err != nil {
    var noSuchKeyErr *types.NoSuchKey
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noSuchKeyErr) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noSuchKeyErr
    } else if errors.As(err, &apiErr) {
        switch apiErr.ErrorCode() {
        case "NoSuchObjectLockConfiguration":
            log.Printf("Object %s does not have an object lock configuration.\n", key)
            err = nil
        case "InvalidRequest":
            log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
            err = nil
        }
    }
} else {
    status = &output.LegalHold.Status
}

return status, err
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [GetObjectLegalHold](#)를 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Get the legal hold details for an S3 object.
```

```

    public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
    objectKey) {
        try {
            GetObjectLegalHoldRequest legalHoldRequest =
            GetObjectLegalHoldRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectLegalHoldResponse response =
            getClient().getObjectLegalHold(legalHoldRequest);
            System.out.println("Object legal hold for " + objectKey + " in " +
            bucketName +
                ":\n\tStatus: " + response.legalHold().status());
            return response.legalHold();

        } catch (S3Exception ex) {
            System.out.println("\tUnable to fetch legal hold: '" +
            ex.getMessage() + "'");
        }

        return null;
    }

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [GetObjectLegalHold](#)를 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

import {
    GetObjectLegalHoldCommand,
    S3Client,
    S3ServiceException,

```

```

} from "@aws-sdk/client-s3";

/**
 * Get an object's current legal hold status.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectLegalHoldCommand({
        Bucket: bucketName,
        Key: key,
        // Optionally, you can provide additional parameters
        // ExpectedBucketOwner: "<account ID that is expected to own the
bucket>",
        // VersionId: "<the specific version id of the object to check>",
      }),
    );
    console.log(`Legal Hold Status: ${response.LegalHold.Status}`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting legal hold status for ${key} in
${bucketName}. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting legal hold status for ${key} in
${bucketName} from ${bucketName}. ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {

```

```

    isMain,
    validateArgs,
  } from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}

```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 [GetObjectLegalHold](#)를 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

객체에 법적 보존을 적용하세요.

```
def get_legal_hold(s3_client, bucket: str, key: str) -> None:
    """
    Get the legal hold status of a specific file in a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket containing the file.
        key: The key of the file to get the legal hold status of.
    """
    print()
    logger.info("Getting legal hold status of file [%s] in bucket [%s]", key,
                bucket)
    try:
        response = s3_client.get_object_legal_hold(Bucket=bucket, Key=key)
        legal_hold_status = response["LegalHold"]["Status"]
        logger.debug(
            "Legal hold status of file [%s] in bucket [%s] is [%s]",
            key,
            bucket,
            legal_hold_status,
        )
    except Exception as e:
        logger.error(
            "Failed to get legal hold status of file [%s] in bucket [%s]: %s",
            key,
            bucket,
            e,
        )
```

- API 세부 정보는 AWS SDK for Python(Boto3) API 참조의 [GetObjectLegalHold](#)를 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.  
    oo_result = lo_s3->getobjectlegalhold(          " oo_result is returned  
for testing purposes. "  
    iv_bucket = iv_bucket_name  
    iv_key = iv_object_key ).  
    MESSAGE 'Retrieved object legal hold status.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
CATCH /aws1/cx_s3_nosuchkey.  
    MESSAGE 'Object key does not exist.' TYPE 'E'.  
ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 [GetObjectLegalHold](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetObjectLockConfiguration** 사용

다음 코드 예시는 GetObjectLockConfiguration의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [Amazon S3 객체 잠그기](#)

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Get the object lock configuration details for an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket to get details.</param>
/// <returns>The bucket's object lock configuration details.</returns>
public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
{
    try
    {
        var request = new GetObjectLockConfigurationRequest()
        {
            BucketName = bucketName
        };

        var response = await
        _amazonS3.GetObjectLockConfigurationAsync(request);
        Console.WriteLine($"\\tBucket object lock config for {bucketName} in
{bucketName}: " +
            $"\\n\\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
            $"\\n\\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

        return response.ObjectLockConfiguration;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to fetch object lock config:
'{ex.Message}'");
        return new ObjectLockConfiguration();
    }
}
```



```
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 [GetObjectLockConfiguration](#)을 참조하세요.

CLI

AWS CLI

버킷의 객체 잠금 구성을 검색하는 방법

다음 get-object-lock-configuration 예시에서는 지정된 버킷의 객체 잠금 구성을 가져옵니다.

```
aws s3api get-object-lock-configuration \
  --bucket amzn-s3-demo-bucket-with-object-lock
```

출력:

```
{
  "ObjectLockConfiguration": {
    "ObjectLockEnabled": "Enabled",
    "Rule": {
      "DefaultRetention": {
        "Mode": "COMPLIANCE",
        "Days": 50
      }
    }
  }
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [GetObjectLockConfiguration](#)을 참조하세요.

Go

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}  
  
// GetObjectLockConfiguration retrieves the object lock configuration for an S3  
// bucket.  
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket  
    string) (*types.ObjectLockConfiguration, error) {  
    var lockConfig *types.ObjectLockConfiguration  
    input := &s3.GetObjectLockConfigurationInput{  
        Bucket: aws.String(bucket),  
    }  
}
```

```

output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
if err != nil {
    var noBucket *types.NoSuchBucket
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
"ObjectLockConfigurationNotFoundError" {
        log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
        err = nil
    }
} else {
    lockConfig = output.ObjectLockConfiguration
}

return lockConfig, err
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [GetObjectLockConfiguration](#)을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

// Get the object lock configuration details for an S3 bucket.
public void getBucketObjectLockConfiguration(String bucketName) {
    GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
        .bucket(bucketName)
        .build();
}

```

```

GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
    System.out.println("Bucket object lock config for "+bucketName+": ");
    System.out.println("\tEnabled:
"+response.getObjectLockConfiguration().objectLockEnabled());
    System.out.println("\tRule: "+
response.getObjectLockConfiguration().rule().defaultRetention());
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [GetObjectLockConfiguration](#)을 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

import {
    GetObjectLockConfigurationCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Gets the Object Lock configuration for a bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
    const client = new S3Client({});

    try {
        const { ObjectLockConfiguration } = await client.send(
            new GetObjectLockConfigurationCommand({
                Bucket: bucketName,
                // Optionally, you can provide additional parameters
            })
        );
    }
}

```

```

        // ExpectedBucketOwner: "<account ID that is expected to own the
        bucket>",
    })),
    );
    console.log(
        `Object Lock Configuration:\n${JSON.stringify(ObjectLockConfiguration)}`,
    );
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while getting object lock configuration for ${bucketName}.
            The bucket doesn't exist.`,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while getting object lock configuration for ${bucketName}.
            ${caught.name}: ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
    isMain,
    validateArgs,
} from "@aws-doc-sdk-examples/lib/utills/util-node.js";

const loadArgs = () => {
    const options = {
        bucketName: {
            type: "string",
            required: true,
        },
    };
};

const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };

```

```
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 [GetObjectLockConfiguration](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷에 대해 객체 잠금 구성이 활성화된 경우 'Enabled' 값을 반환합니다.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select
ObjectLockConfiguration.ObjectLockEnabled
```

출력:

```
Value
-----
Enabled
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [GetObjectLockConfiguration](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 S3 버킷에 대해 객체 잠금 구성이 활성화된 경우 'Enabled' 값을 반환합니다.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select
ObjectLockConfiguration.ObjectLockEnabled
```

출력:

```
Value
-----
Enabled
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [GetObjectLockConfiguration](#)을 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

객체 잠금 구성을 가져오세요.

```
def is_object_lock_enabled(s3_client, bucket: str) -> bool:
    """
    Check if object lock is enabled for a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket to check.

    Returns:
        True if object lock is enabled, False otherwise.
    """
    try:
        response = s3_client.get_object_lock_configuration(Bucket=bucket)
        return (
            "ObjectLockConfiguration" in response
            and response["ObjectLockConfiguration"]["ObjectLockEnabled"] ==
            "Enabled"
        )
    except s3_client.exceptions.ClientError as e:
        if e.response["Error"]["Code"] == "ObjectLockConfigurationNotFoundError":
```

```

        return False
    else:
        raise

```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 [GetObjectLockConfiguration](#)을 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

TRY.
    oo_result = lo_s3->getobjectlockconfiguration(          " oo_result is
returned for testing purposes. "
    iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved object lock configuration.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 [GetObjectLockConfiguration](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetObjectRetention** 사용

다음 코드 예시는 GetObjectRetention의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [Amazon S3 객체 잠그기](#)

.NET

SDK for .NET

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Get the retention period for an S3 object.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The object key.</param>
/// <returns>The object retention details.</returns>
public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
    string objectKey)
{
    try
    {
        var request = new GetObjectRetentionRequest()
        {
            BucketName = bucketName,
            Key = objectKey
        };

        var response = await _amazonS3.GetObjectRetentionAsync(request);
        Console.WriteLine($"{Environment.NewLine}Object retention for {objectKey} in
{bucketName}: " +
                        $"{Environment.NewLine}{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}." );
        return response.Retention;
    }
    catch (AmazonS3Exception ex)
    {
    }
```

```

        Console.WriteLine($"Unable to fetch object lock retention:
'{ex.Message}');
        return new ObjectLockRetention();
    }
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [GetObjectRetention](#)을 참조하세요.

CLI

AWS CLI

객체의 객체 보존 구성을 검색하는 방법

다음 get-object-retention 예시에서는 지정된 객체에 대한 객체 보존 구성을 가져옵니다.

```

aws s3api get-object-retention \
  --bucket amzn-s3-demo-bucket-with-object-lock \
  --key doc1.rtf

```

출력:

```

{
  "Retention": {
    "Mode": "GOVERNANCE",
    "RetainUntilDate": "2025-01-01T00:00:00.000Z"
  }
}

```

- API 세부 정보는 AWS CLI 명령 참조의 [GetObjectRetention](#)을 참조하세요.

Go

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}  
  
// GetObjectRetention retrieves the object retention configuration for an S3  
object.  
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key  
string) (*types.ObjectLockRetention, error) {  
    var retention *types.ObjectLockRetention  
    input := &s3.GetObjectRetentionInput{  
        Bucket: aws.String(bucket),  
        Key:     aws.String(key),  
    }  
  
    output, err := actor.S3Client.GetObjectRetention(ctx, input)  
    if err != nil {  
        var noKey *types.NoSuchKey  
        var apiErr *smithy.GenericAPIError  
        if errors.As(err, &noKey) {  
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)  
            err = noKey  
        } else if errors.As(err, &apiErr) {  
            switch apiErr.ErrorCode() {
```

```

    case "NoSuchObjectLockConfiguration":
        err = nil
    case "InvalidRequest":
        log.Printf("Bucket %s does not have locking enabled.", bucket)
        err = nil
    }
}
} else {
    retention = output.Retention
}

return retention, err
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [GetObjectRetention](#)을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

// Get the retention period for an S3 object.
public ObjectLockRetention getObjectRetention(String bucketName, String key){
    try {
        GetObjectRetentionRequest retentionRequest =
        GetObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        GetObjectRetentionResponse response =
        getClient().getObjectRetention(retentionRequest);
        System.out.println("tObject retention for "+key +"
in "+ bucketName +": " + response.retention().mode() +" until "+
response.retention().retainUntilDate() +".");
    }
}

```

```

        return response.retention();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        return null;
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [GetObjectRetention](#)을 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

import {
    GetObjectRetentionCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Log the "RetainUntilDate" for an object in an S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
    const client = new S3Client({});

    try {
        const { Retention } = await client.send(
            new GetObjectRetentionCommand({
                Bucket: bucketName,
                Key: key,
            }),
        );
    }
};

```

```
    console.log(
      `${key} in ${bucketName} will be retained until
      ${Retention.RetainUntilDate}`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchObjectLockConfiguration"
    ) {
      console.warn(
        `The object "${key}" in the bucket "${bucketName}" does not have an
        ObjectLock configuration.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object retention settings for
        "${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
};
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
```

```

    return { errors, results };
};

if (isMain(import.meta.url)) {
    const { errors, results } = loadArgs();
    if (!errors) {
        main(results.values);
    } else {
        console.error(errors.join("\n"));
    }
}

```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 [GetObjectRetention](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 객체 보존의 모드와 종료 날짜를 반환합니다.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [GetObjectRetention](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 객체 보존의 모드와 종료 날짜를 반환합니다.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [GetObjectRetention](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 **GetObjectTagging** 사용

다음 코드 예시는 GetObjectTagging의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [태깅 시작하기](#)

CLI

AWS CLI

객체에 연결된 태그를 검색하려면

다음 `get-object-tagging` 예시에서는 지정된 객체에서 지정된 키의 값을 가져옵니다.

```
aws s3api get-object-tagging \
  --bucket amzn-s3-demo-bucket \
  --key doc1.rtf
```

출력:

```
{
  "TagSet": [
    {
      "Value": "confidential",
      "Key": "designation"
    }
  ]
}
```

다음 `get-object-tagging` 예시에서는 태그가 없는 객체 `doc2.rtf`의 태그 세트를 가져오려고 시도합니다.

```
aws s3api get-object-tagging \
  --bucket amzn-s3-demo-bucket \
  --key doc2.rtf
```

출력:

```
{
  "TagSet": []
}
```


다음 `get-object-tagging` 예시에서는 태그가 여러 개 있는 객체 `doc3.rtf`의 태그 세트를 가져옵니다.

```
aws s3api get-object-tagging \
  --bucket amzn-s3-demo-bucket \
  --key doc3.rtf
```

출력:

```
{
  "TagSet": [
    {
      "Value": "confidential",
      "Key": "designation"
    },
    {
      "Value": "finance",
      "Key": "department"
    },
    {
      "Value": "payroll",
      "Key": "team"
    }
  ]
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [GetObjectTagging](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 샘플은 지정된 S3 버킷에 있는 객체와 연결된 태그를 반환합니다.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

출력:

```
Key  Value
---  -
```

```
test value
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [GetObjectTagging](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 샘플은 지정된 S3 버킷에 있는 객체와 연결된 태그를 반환합니다.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

출력:

```
Key  Value
---  -
test value
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [GetObjectTagging](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 **GetPublicAccessBlock** 사용

다음 코드 예시는 GetPublicAccessBlock의 사용 방법을 보여 줍니다.

CLI

AWS CLI

버킷의 퍼블릭 액세스 차단 구성을 설정하거나 수정하려면

다음 get-public-access-block 예시에서는 지정된 버킷의 퍼블릭 액세스 차단 구성을 표시합니다.

```
aws s3api get-public-access-block \
  --bucket amzn-s3-demo-bucket
```

출력:

```
{
  "PublicAccessBlockConfiguration": {
    "IgnorePublicAcls": true,
    "BlockPublicPolicy": true,
    "BlockPublicAcls": true,
    "RestrictPublicBuckets": true
  }
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [GetPublicAccessBlock](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷의 퍼블릭 액세스 차단 구성을 반환합니다.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [GetPublicAccessBlock](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 S3 버킷의 퍼블릭 액세스 차단 구성을 반환합니다.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [GetPublicAccessBlock](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **HeadBucket** 사용

다음 코드 예시는 HeadBucket의 사용 방법을 보여 줍니다.

Bash

AWS CLIBash 스크립트와 함께 사용

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
#####
# function bucket_exists
#
# This function checks to see if the specified bucket already exists.
#
# Parameters:
#     $1 - The name of the bucket to check.
#
# Returns:
#     0 - If the bucket already exists.
#     1 - If the bucket doesn't exist.
#####
function bucket_exists() {
    local bucket_name
    bucket_name=$1

    # Check whether the bucket already exists.
    # We suppress all output - we're interested only in the return code.

    if aws s3api head-bucket \
        --bucket "$bucket_name" \
        >/dev/null 2>&1; then
        return 0 # 0 in Bash script means true.
    else
        return 1 # 1 in Bash script means false.
    fi
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [HeadBucket](#)을 참조하세요.

CLI

AWS CLI

다음 명령은 `amzn-s3-demo-bucket`이라는 버킷에 대한 액세스를 확인합니다.

```
aws s3api head-bucket --bucket amzn-s3-demo-bucket
```

버킷이 존재하고 버킷에 대한 액세스 권한이 있는 경우 출력이 반환되지 않습니다. 그렇지 않으면 오류 메시지가 표시됩니다. 예제:

```
A client error (404) occurred when calling the HeadBucket operation: Not Found
```

- API 세부 정보는 AWS CLI 명령 참조의 [HeadBucket](#)을 참조하세요.

Go

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"
```

```
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(ctx context.Context, bucketName string)
    (bool, error) {
    _, err := basics.S3Client.HeadBucket(ctx, &s3.HeadBucketInput{
        Bucket: aws.String(bucketName),
    })
    exists := true
    if err != nil {
        var apiError smithy.APIError
        if errors.As(err, &apiError) {
            switch apiError.(type) {
            case *types.NotFound:
                log.Printf("Bucket %v is available.\n", bucketName)
                exists = false
                err = nil
            default:
                log.Printf("Either you don't have access to bucket %v or another error
occurred. "+
                    "Here's what happened: %v\n", bucketName, err)
            }
        }
    } else {
        log.Printf("Bucket %v exists and you already own it.", bucketName)
    }

    return exists, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [HeadBucket](#)을 참조하세요.

PowerShell

Tools for PowerShell V5

예제 1: 이 명령에서는 사용자에게 액세스 권한이 있을 때 기존 버킷에 대해 HTTP 상태 코드 200 OK로 출력을 반환합니다. BucketArn 파라미터는 S3 디렉터리 버킷에서만 지원됩니다.

```
Get-S3HeadBucket -BucketName amzn-s3-demo-bucket
```

출력:

```
AccessPointAlias    : False
BucketArn           :
BucketLocationName  :
BucketLocationType  :
BucketRegion        : us-east-2
ResponseMetadata    : Amazon.Runtime.ResponseMetadata
ContentLength       : 0
HttpStatusCode       : OK
```

예제 2: 이 명령에서는 존재하지 않는 버킷에 대해 HTTP 상태 코드 NotFound와 함께 오류가 발생합니다.

```
Get-S3HeadBucket -BucketName amzn-s3-non-existing-bucket
```

출력:

```
Get-S3HeadBucket: Error making request with Error Code NotFound and Http Status Code NotFound. No further error information was returned by the service.
```

예제 3: 이 명령에서는 사용자에게 액세스 권한이 없는 기존 버킷에 대해 HTTP 상태 코드 Forbidden 오류가 발생합니다.

```
Get-S3HeadBucket -BucketName amzn-s3-no-access-bucket
```

출력:

Get-S3HeadBucket: Error making request with Error Code Forbidden and Http Status Code Forbidden. No further error information was returned by the service.

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [HeadBucket](#)을 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def exists(self):
        """
        Determine whether the bucket exists and you have access to it.

        :return: True when the bucket exists; otherwise, False.
        """
        try:
            self.bucket.meta.client.head_bucket(Bucket=self.bucket.name)
            logger.info("Bucket %s exists.", self.bucket.name)
            exists = True
        except ClientError:
            logger.warning(
                "Bucket %s doesn't exist or you don't have access to it.",
```



```

        self.bucket.name,
    )
    exists = False
    return exists

```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [HeadBucket](#)를 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

TRY.
    oo_result = lo_s3->headbucket(          " oo_result is returned for
testing purposes. "
    iv_bucket = iv_bucket_name ).
    MESSAGE 'Bucket exists and you have access to it.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- API 세부 정보는 AWSSDK for SAP ABAP API의 [HeadBucket](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **HeadObject** 사용

다음 코드 예시는 HeadObject의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [객체가 존재하는지 확인](#)
- [객체 스토리지 시작하기](#)

CLI

AWS CLI

다음 명령은 `amzn-s3-demo-bucket`이라는 버킷의 객체에 대한 메타데이터를 가져옵니다.

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key index.html
```

출력:

```
{
  "AcceptRanges": "bytes",
  "ContentType": "text/html",
  "LastModified": "Thu, 16 Apr 2015 18:19:14 GMT",
  "ContentLength": 77,
  "VersionId": "null",
  "ETag": "\"30a6ec7e1a9ad79c203d05a589c8b400\"",
  "Metadata": {}
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [HeadObject](#)를 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

객체의 콘텐츠 유형을 결정합니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectContentType {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
            "";

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

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getContentType(s3, bucketName, keyName);
        s3.close();
    }
}
```

```

/**
 * Retrieves the content type of an object stored in an Amazon S3 bucket.
 *
 * @param s3 an instance of the {@link S3Client} class, which is used to
interact with the Amazon S3 service
 * @param bucketName the name of the S3 bucket where the object is stored
 * @param keyName the key (file name) of the object in the S3 bucket
 */
public static void getContentType(S3Client s3, String bucketName, String
keyName) {
    try {
        HeadObjectRequest objectRequest = HeadObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        HeadObjectResponse objectHead = s3.headObject(objectRequest);
        String type = objectHead.contentType();
        System.out.println("The object content type is " + type);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

객체의 복원 상태를 가져옵니다.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

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

        Usage:

```

```

        <bucketName> <keyName>\s

Where:
    bucketName - The Amazon S3 bucket name.\s
    keyName - A key name that represents the object.\s
""";

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

String bucketName = args[0];
String keyName = args[1];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

checkStatus(s3, bucketName, keyName);
s3.close();
}

/**
 * Checks the restoration status of an Amazon S3 object.
 *
 * @param s3          an instance of the {@link S3Client} class used to
interact with the Amazon S3 service
 * @param bucketName the name of the Amazon S3 bucket where the object is
stored
 * @param keyName     the name of the Amazon S3 object to be checked
 * @throws S3Exception if an error occurs while interacting with the Amazon
S3 service
 */
public static void checkStatus(S3Client s3, String bucketName, String
keyName) {
    try {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        HeadObjectResponse response = s3.headObject(headObjectRequest);

```

```

        System.out.println("The Amazon S3 object restoration status is " +
response.restore());

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [HeadObject](#)를 참조하십시오.

Ruby

SDK for Ruby

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectExistsWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Checks whether the object exists.
  #
  # @return [Boolean] True if the object exists; otherwise false.
  def exists?
    @object.exists?
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't check existence of object
#{@object.bucket.name}:#{@object.key}. Here's why: #{e.message}"
  end
end

```

```

    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectExistsWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  exists = wrapper.exists?

  puts "Object #{object_key} #{exists ? 'does' : 'does not'} exist."
end

run_demo if $PROGRAM_NAME == __FILE__

```

- API 세부 정보는 AWS SDK for Ruby API 참조의 [HeadObject](#)를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 **ListBucketAnalyticsConfigurations** 사용

다음 코드 예시는 ListBucketAnalyticsConfigurations의 사용 방법을 보여 줍니다.

CLI

AWS CLI

버킷의 분석 구성 목록을 검색하려면

다음 list-bucket-analytics-configurations는 지정된 버킷의 분석 구성 목록을 가져옵니다.

```
aws s3api list-bucket-analytics-configurations \
  --bucket amzn-s3-demo-bucket
```

출력:

```
{
  "AnalyticsConfigurationList": [
    {
      "StorageClassAnalysis": {},
      "Id": "1"
    }
  ],
  "IsTruncated": false
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [ListBucketAnalyticsConfigurations](#)를 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷의 분석 구성 중 처음 100개를 반환합니다.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [ListBucketAnalyticsConfigurations](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 S3 버킷의 분석 구성 중 처음 100개를 반환합니다.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [ListBucketAnalyticsConfigurations](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 **ListBucketInventoryConfigurations** 사용

다음 코드 예시는 ListBucketInventoryConfigurations의 사용 방법을 보여 줍니다.

CLI

AWS CLI

버킷의 인벤토리 구성을 검색하려면

다음 `list-bucket-inventory-configurations` 예시에서는 지정된 버킷의 인벤토리 구성을 나열합니다.

```
aws s3api list-bucket-inventory-configurations \
  --bucket amzn-s3-demo-bucket
```

출력:

```
{
  "InventoryConfigurationList": [
    {
      "IsEnabled": true,
      "Destination": {
        "S3BucketDestination": {
          "Format": "ORC",
          "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
          "AccountId": "123456789012"
        }
      },
      "IncludedObjectVersions": "Current",
      "Id": "1",
      "Schedule": {
        "Frequency": "Weekly"
      }
    },
    {
      "IsEnabled": true,
      "Destination": {
        "S3BucketDestination": {
          "Format": "CSV",
          "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
          "AccountId": "123456789012"
        }
      },
      "IncludedObjectVersions": "Current",
      "Id": "2",
      "Schedule": {
```

```

        "Frequency": "Daily"
      }
    ],
    "IsTruncated": false
  }

```

- API 세부 정보는 AWS CLI 명령 참조의 [ListBucketInventoryConfigurations](#)를 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷의 인벤토리 구성 중 처음 100개를 반환합니다.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [ListBucketInventoryConfigurations](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 S3 버킷의 인벤토리 구성 중 처음 100개를 반환합니다.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [ListBucketInventoryConfigurations](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListBuckets** 사용

다음 코드 예시는 ListBuckets의 사용 방법을 보여 줍니다.

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
namespace ListBucketsExample
{
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example uses the AWS SDK for .NET to list the Amazon Simple Storage
    /// Service (Amazon S3) buckets belonging to the default account.
    /// </summary>
    public class ListBuckets
    {
        private static IAmazonS3 _s3Client;

        /// <summary>
        /// Get a list of the buckets owned by the default user.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <returns>The response from the ListingBuckets call that contains a
        /// list of the buckets owned by the default user.</returns>
        public static async Task<ListBucketsResponse> GetBuckets(IAmazonS3
client)
        {
            return await client.ListBucketsAsync();
        }

        /// <summary>
        /// This method lists the name and creation date for the buckets in
        /// the passed List of S3 buckets.
        /// </summary>
    }
}
```

```

    /// <param name="bucketList">A List of S3 bucket objects.</param>
    public static void DisplayBucketList(List<S3Bucket> bucketList)
    {
        bucketList
            .ForEach(b => Console.WriteLine($"Bucket name: {b.BucketName},
created on: {b.CreationDate}"));
    }

    public static async Task Main()
    {
        // The client uses the AWS Region of the default user.
        // If the Region where the buckets were created is different,
        // pass the Region to the client constructor. For example:
        // _s3Client = new AmazonS3Client(RegionEndpoint.USEast1);
        _s3Client = new AmazonS3Client();
        var response = await GetBuckets(_s3Client);
        DisplayBucketList(response.Buckets);
    }
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [ListBuckets](#)를 참조하십시오.

C++

SDK for C++

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::S3::listBuckets(const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);

    auto outcome = client.ListBuckets();

    bool result = true;

```

```

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
        result = false;
    } else {
        std::cout << "Found " << outcome.GetResult().GetBuckets().size() << "
buckets\n";
        for (auto &&b: outcome.GetResult().GetBuckets()) {
            std::cout << b.GetName() << std::endl;
        }
    }

    return result;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListBuckets](#)를 참조하세요.

CLI

AWS CLI

다음 명령은 `list-buckets` 명령을 사용하여 모든 Amazon S3 버킷(모든 리전)의 이름을 표시합니다.

```
aws s3api list-buckets --query "Buckets[].Name"
```

쿼리 옵션은 `list-buckets`의 출력을 버킷 이름으로만 필터링합니다.

버킷에 대한 자세한 내용은 Amazon S3 개발자 안내서의 Amazon S3 버킷 작업을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [ListBuckets](#)를 참조하세요.

Go

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)  
// actions  
// used in the examples.  
// It contains S3Client, an Amazon S3 service client that is used to perform  
// bucket  
// and object actions.  
type BucketBasics struct {  
    S3Client *s3.Client  
}  
  
// ListBuckets lists the buckets in the current account.  
func (basics BucketBasics) ListBuckets(ctx context.Context) ([]types.Bucket,  
    error) {  
    var err error  
    var output *s3.ListBucketsOutput  
    var buckets []types.Bucket  
    bucketPaginator := s3.NewListBucketsPaginator(basics.S3Client,  
        &s3.ListBucketsInput{})  
    for bucketPaginator.HasMorePages() {  
        output, err = bucketPaginator.NextPage(ctx)  
        if err != nil {  
            var apiErr smithy.APIError  
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "AccessDenied" {
```

```

    fmt.Println("You don't have permission to list buckets for this account.")
    err = apiErr
} else {
    log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
}
break
} else {
    buckets = append(buckets, output.Buckets...)
}
}
return buckets, err
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListBuckets](#)를 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListBucketsIterable;
/**
 * 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 ListBuckets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
    }
}

```

```

        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listAllBuckets(s3);

    }

    /**
     * Lists all the S3 buckets available in the current AWS account.
     *
     * @param s3 The {@link S3Client} instance to use for interacting with the
     * Amazon S3 service.
     */
    public static void listAllBuckets(S3Client s3) {
        ListBucketsIterable response = s3.listBucketsPaginator();
        response.buckets().forEach(bucket ->
            System.out.println("Bucket Name: " + bucket.name()));
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [ListBuckets](#)를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷을 나열합니다.

```

import {
    paginateListBuckets,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**

```



```

* List the Amazon S3 buckets in your account.
*/
export const main = async () => {
  const client = new S3Client({});
  /** @type {?import('@aws-sdk/client-s3').Owner} */
  let Owner = null;

  /** @type {import('@aws-sdk/client-s3').Bucket[]} */
  const Buckets = [];

  try {
    const paginator = paginateListBuckets({ client }, {});

    for await (const page of paginator) {
      if (!Owner) {
        Owner = page.Owner;
      }

      Buckets.push(...page.Buckets);
    }

    console.log(
      `${Owner.DisplayName} owns ${Buckets.length} bucket${
        Buckets.length === 1 ? "" : "s"
      }`,
    );
    console.log(`${Buckets.map((b) => ` • ${b.Name}`).join("\n")}`);
  } catch (caught) {
    if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while listing buckets.  ${caught.name}:
        ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

```

- 자세한 정보는 [AWS SDK for JavaScript 개발자 안내서](#)를 참조하세요.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 [ListBuckets](#)를 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 모든 S3 버킷을 반환합니다.

```
Get-S3Bucket
```

예시 2: 이 명령은 이름이 “test-files”인 버킷을 반환합니다.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [ListBuckets](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 모든 S3 버킷을 반환합니다.

```
Get-S3Bucket
```

예시 2: 이 명령은 이름이 “test-files”인 버킷을 반환합니다.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [ListBuckets](#)을 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
```

```

        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    @staticmethod
    def list(s3_resource):
        """
        Get the buckets in all Regions for the current account.

        :param s3_resource: A Boto3 S3 resource. This is a high-level resource in
Boto3
                        that contains collections and factory methods to
create
                        other high-level S3 sub-resources.
        :return: The list of buckets.
        """
        try:
            buckets = list(s3_resource.buckets.all())
            logger.info("Got buckets: %s.", buckets)
        except ClientError:
            logger.exception("Couldn't get buckets.")
            raise
        else:
            return buckets

```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [ListBuckets](#)를 참조하십시오.

Ruby

SDK for Ruby

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 resource actions.
class BucketListWrapper
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Lists buckets for the current account.
  #
  # @param count [Integer] The maximum number of buckets to list.
  def list_buckets(count)
    puts 'Found these buckets:'
    @s3_resource.buckets.each do |bucket|
      puts "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list buckets. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  wrapper = BucketListWrapper.new(Aws::S3::Resource.new)
  wrapper.list_buckets(25)
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 [ListBuckets](#)를 참조하십시오.

Rust

SDK for Rust

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn show_buckets(
    strict: bool,
    client: &Client,
    region: BucketLocationConstraint,
) -> Result<(), S3ExampleError> {
    let mut buckets = client.list_buckets().into_paginator().send();

    let mut num_buckets = 0;
    let mut in_region = 0;

    while let Some(Ok(output)) = buckets.next().await {
        for bucket in output.buckets() {
            num_buckets += 1;
            if strict {
                let r = client
                    .get_bucket_location()
                    .bucket(bucket.name().unwrap_or_default())
                    .send()
                    .await?;

                if r.location_constraint() == Some(&region) {
                    println!("{}", bucket.name().unwrap_or_default());
                    in_region += 1;
                }
            } else {
                println!("{}", bucket.name().unwrap_or_default());
            }
        }
    }

    println!();
    if strict {
```

```
println!(
    "Found {} buckets in the {} region out of a total of {} buckets.",
    in_region, region, num_buckets
);
} else {
    println!("Found {} buckets in all regions.", num_buckets);
}

Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 [ListBuckets](#)을 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.
    oo_result = lo_s3->listbuckets(          " oo_result is returned for
testing purposes. "
    ).
    DATA(lv_bucket_count) = lines( oo_result->get_buckets( ) ).
    MESSAGE |Retrieved { lv_bucket_count } buckets in all regions.| TYPE 'I'.
    CATCH /aws1/cx_rt_generic.
    MESSAGE 'Unable to list buckets.' TYPE 'E'.
ENDTRY.
```

- API 세부 정보는 AWSSDK for SAP ABAP API의 [ListBuckets](#)를 참조하세요.

Swift

SDK for Swift

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSS3

/// Return an array containing information about every available bucket.
///
/// - Returns: An array of ``S3ClientTypes.Bucket`` objects describing
/// each bucket.
public func getAllBuckets() async throws -> [S3ClientTypes.Bucket] {
    return try await client.listBuckets(input: ListBucketsInput())
}
```

- API 세부 정보는 AWS SDK for Swift API 참조의 [ListBuckets](#)를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListMultipartUploads** 사용

다음 코드 예시는 ListMultipartUploads의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [미완료 멀티파트 업로드 삭제](#)

CLI

AWS CLI

다음 명령은 amzn-s3-demo-bucket이라는 버킷의 활성 멀티파트 업로드를 모두 나열합니다.

```
aws s3api list-multipart-uploads --bucket amzn-s3-demo-bucket
```

출력:

```
{
  "Uploads": [
    {
      "Initiator": {
        "DisplayName": "username",
        "ID": "arn:aws:iam::0123456789012:user/username"
      },
      "Initiated": "2015-06-02T18:01:30.000Z",
      "UploadId":
        "dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3",
      "StorageClass": "STANDARD",
      "Key": "multipart/01",
      "Owner": {
        "DisplayName": "aws-account-name",
        "ID":
          "100719349fc3b6dcd7c820a124bf7aec408092c3d7b51b38494939801fc248b"
      }
    }
  ],
  "CommonPrefixes": []
}
```

진행 중인 멀티파트 업로드는 Amazon S3에서 스토리지 비용을 발생시킵니다. 활성 멀티파트 업로드를 완료하거나 중단하여 계정에서 해당 파트를 제거하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [ListMultipartUploads](#)를 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.


```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;
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 ListMultipartUploads {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The name of the Amazon S3 bucket where an in-
progress multipart upload is occurring.
            """;

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

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();
        listUploads(s3, bucketName);
        s3.close();
    }
}
```

```

/**
 * Lists the multipart uploads currently in progress in the specified Amazon
 * S3 bucket.
 *
 * @param s3 the S3Client object used to interact with Amazon S3
 * @param bucketName the name of the Amazon S3 bucket to list the multipart
 * uploads for
 */
public static void listUploads(S3Client s3, String bucketName) {
    try {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
            ListMultipartUploadsRequest.builder()
                .bucket(bucketName)
                .build();

        ListMultipartUploadsResponse response =
            s3.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();
        for (MultipartUpload upload : uploads) {
            System.out.println("Upload in progress: Key = \"" + upload.key()
                + "\", id = " + upload.uploadId());
        }

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [ListMultipartUploads](#)를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListObjectVersions** 사용

다음 코드 예시는 ListObjectVersions의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [객체 스토리지 시작하기](#)
- [버전이 지정된 객체 작업](#)

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example lists the versions of the objects in a version enabled
/// Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class ListObjectVersions
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";

        // If the AWS Region where your bucket is defined is different from
        // the AWS Region where the Amazon S3 bucket is defined, pass the
constant
        // for the AWS Region to the client constructor like this:
        //     var client = new AmazonS3Client(RegionEndpoint.USWest2);
        IAmazonS3 client = new AmazonS3Client();
        await GetObjectListWithAllVersionsAsync(client, bucketName);
    }

    /// <summary>
```

```

    /// This method lists all versions of the objects within an Amazon S3
    /// version enabled bucket.
    /// </summary>
    /// <param name="client">The initialized client object used to call
    /// ListVersionsAsync.</param>
    /// <param name="bucketName">The name of the version enabled Amazon S3
bucket
    /// for which you want to list the versions of the contained objects.</
param>
    public static async Task GetObjectListWithAllVersionsAsync(IAmazonS3
client, string bucketName)
    {
        try
        {
            // When you instantiate the ListVersionRequest, you can
            // optionally specify a key name prefix in the request
            // if you want a list of object versions of a specific object.

            // For this example we set a small limit in MaxKeys to return
            // a small list of versions.
            ListVersionsRequest request = new ListVersionsRequest()
            {
                BucketName = bucketName,
                MaxKeys = 2,
            };

            do
            {
                ListVersionsResponse response = await
client.ListVersionsAsync(request);

                // Process response.
                foreach (S3ObjectVersion entry in response.Versions)
                {
                    Console.WriteLine($"key: {entry.Key} size:
{entry.Size}");
                }

                // If response is truncated, set the marker to get the next
                // set of keys.
                if (response.IsTruncated)
                {
                    request.KeyMarker = response.NextKeyMarker;
                    request.VersionIdMarker = response.NextVersionIdMarker;

```

```

        }
        else
        {
            request = null;
        }
    }
    while (request != null);
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error: '{ex.Message}'");
}
}
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [ListObjectVersions](#)를 참조하세요.

CLI

AWS CLI

다음 명령은 이름이 `amzn-s3-demo-bucket`인 버킷의 객체의 버전 정보를 검색합니다.

```
aws s3api list-object-versions --bucket amzn-s3-demo-bucket --prefix index.html
```

출력:

```

{
  "DeleteMarkers": [
    {
      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": true,
      "VersionId": "B2VsEK5saUNNHKc0AJj7hIE86RozToyq",
      "Key": "index.html",
      "LastModified": "2015-11-10T00:57:03.000Z"
    },
  ],
}

```

```

    {
      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": false,
      "VersionId": ".FLQEZscLIcfxSq.jsFJ.szUkmng2Yw6",
      "Key": "index.html",
      "LastModified": "2015-11-09T23:32:20.000Z"
    }
  ],
  "Versions": [
    {
      "LastModified": "2015-11-10T00:20:11.000Z",
      "VersionId": "Rb_l2T8UHDkFEwCgJjhlGPOZC0qJ.vpD",
      "ETag": "\"0622528de826c0df5db1258a23b80be5\"",
      "StorageClass": "STANDARD",
      "Key": "index.html",
      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": false,
      "Size": 38
    },
    {
      "LastModified": "2015-11-09T23:26:41.000Z",
      "VersionId": "rasWWGpgk9E4s0LyTJgusGeRQKLVIAff",
      "ETag": "\"06225825b8028de826c0df5db1a23be5\"",
      "StorageClass": "STANDARD",
      "Key": "index.html",
      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": false,
      "Size": 38
    },
    {
      "LastModified": "2015-11-09T22:50:50.000Z",
      "VersionId": "null",

```

```

        "ETag": "\"d1f45267a863c8392e07d24dd592f1b9\"",
        "StorageClass": "STANDARD",
        "Key": "index.html",
        "Owner": {
            "DisplayName": "my-username",
            "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
        },
        "IsLatest": false,
        "Size": 533823
    }
]
}

```

- API 세부 정보는 AWS CLI 명령 참조의 [ListObjectVersions](#)를 참조하세요.

Go

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

```

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
([]types.ObjectVersion, error) {
    var err error
    var output *s3.ListObjectVersionsOutput
    var versions []types.ObjectVersion
    input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
    versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
    for versionPaginator.HasMorePages() {
        output, err = versionPaginator.NextPage(ctx)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucket)
                err = noBucket
            }
            break
        } else {
            versions = append(versions, output.Versions...)
        }
    }
    return versions, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListObjectVersions](#)를 참조하십시오.

Rust

SDK for Rust

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn show_versions(client: &Client, bucket: &str) -> Result<(), Error> {
    let resp = client.list_object_versions().bucket(bucket).send().await?;

    for version in resp.versions() {
        println!("{}", version.key().unwrap_or_default());
        println!("  version ID: {}", version.version_id().unwrap_or_default());
        println!();
    }

    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 [ListObjectVersions](#)을 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.
    oo_result = lo_s3->listobjectversions(          " oo_result is returned
for testing purposes. "
    iv_bucket = iv_bucket_name
    iv_prefix = iv_prefix ).
```

```
MESSAGE 'Retrieved object versions.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 [ListObjectVersions](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 **ListObjects** 사용

다음 코드 예시는 ListObjects의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [Amazon S3 객체를 나열하는 웹 페이지 생성](#)

CLI

AWS CLI

다음 예시에서는 list-objects 명령을 사용하여 지정된 버킷에 있는 모든 객체의 이름을 표시합니다.

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size: Size}'
```

이 예시에서는 --query 인수를 사용하여 list-objects의 출력을 각 객체의 키 값 및 크기로 필터링합니다.

객체에 대한 자세한 내용은 Amazon S3 개발자 안내서의 Amazon S3 객체 작업을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [ListObjects](#)를 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 “test-files” 버킷에 있는 모든 항목에 대한 정보를 검색합니다.

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

예시 2: 이 명령은 "test-files" 버킷에서 "sample.txt" 항목에 대한 정보를 검색합니다.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

예시 3: 이 명령은 “test-files” 버킷에서 접두사가 “sample”인 모든 항목에 대한 정보를 검색합니다.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [ListObjects](#)를 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 “test-files” 버킷에 있는 모든 항목에 대한 정보를 검색합니다.

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

예시 2: 이 명령은 "test-files" 버킷에서 "sample.txt" 항목에 대한 정보를 검색합니다.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

예시 3: 이 명령은 “test-files” 버킷에서 접두사가 “sample”인 모든 항목에 대한 정보를 검색합니다.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet Reference (V5)의 [ListObjects](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListObjectsV2** 사용

다음 코드 예시는 ListObjectsV2의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [기본 사항 알아보기](#)
- [버킷의 모든 객체 삭제](#)
- [객체 스토리지 시작하기](#)

.NET

SDK for .NET(v4)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Shows how to list the objects in an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket for which to list.
/// <param name="printList">True to print out the list.
/// <returns>The collection of objects.</returns>
public async Task<List<S3Object>?> ListBucketContentsAsync(string bucketName,
bool printList = true)
{
    try
    {
        var request = new ListObjectsV2Request
        {
            BucketName = bucketName,
            MaxKeys = 5,
        };

        if (printList)
```

```

        {
            Console.WriteLine("-----");
            Console.WriteLine($"Listing the contents of {bucketName}:");
            Console.WriteLine("-----");
        }

        var listObjectsV2Paginator = _amazonS3.Paginators.ListObjectsV2(new
ListObjectsV2Request
        {
            BucketName = bucketName,
        });
        var s3Objects = new List<S3Object>();
        await foreach (var response in listObjectsV2Paginator.Responses)
        {
            if (response.S3Objects != null)
            {
                s3Objects.AddRange(response.S3Objects);
            }
        }

        if (printList)
        {
            Console.WriteLine($"Number of Objects: {s3Objects.Count}");
            foreach (var entry in s3Objects)
            {
                Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
            }
        }

        return s3Objects;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error encountered on server.
Message: '{ex.Message}' getting list of objects.");
        return null;
    }
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [ListObjectsV2](#)를 참조하세요.

SDK for .NET

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

페이지네이터를 사용하여 객체를 나열합니다.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// The following example lists objects in an Amazon Simple Storage
/// Service (Amazon S3) bucket.
/// </summary>
public class ListObjectsPaginator
{
    private const string BucketName = "amzn-s3-demo-bucket";

    public static async Task Main()
    {
        IAmazonS3 s3Client = new AmazonS3Client();

        Console.WriteLine($"Listing the objects contained in {BucketName}:
\n");

        await ListingObjectsAsync(s3Client, BucketName);
    }

    /// <summary>
    /// This method uses a paginator to retrieve the list of objects in an
    /// an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">An Amazon S3 client object.</param>
    /// <param name="bucketName">The name of the S3 bucket whose objects
    /// you want to list.</param>
    public static async Task ListingObjectsAsync(IAmazonS3 client, string
bucketName)
    {

```

```

        var listObjectsV2Paginator = client.Paginators.ListObjectsV2(new
        ListObjectsV2Request
        {
            BucketName = bucketName,
        });

        await foreach (var response in listObjectsV2Paginator.Responses)
        {
            Console.WriteLine($"HttpStatusCode: {response.HttpStatusCode}");
            Console.WriteLine($"Number of Keys: {response.KeyCount}");
            foreach (var entry in response.S3Objects)
            {
                Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
            }
        }
    }
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [ListObjectsV2](#)를 참조하십시오.

Bash

AWS CLIBash 스크립트와 함께 사용

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####

```

```

# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#     $1 - The name of the bucket.
#
# Returns:
#     The list of files in text format.
#     And:
#     0 - If successful.
#     1 - If it fails.
#####
function list_items_in_bucket() {
    local bucket_name=$1
    local response

    response=$(aws s3api list-objects \
        --bucket "$bucket_name" \
        --output text \
        --query 'Contents[].{Key: Key, Size: Size}')

    # shellcheck disable=SC2181
    if [[ ${?} -eq 0 ]]; then
        echo "$response"
    else
        errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
        return 1
    fi
}

```

- API 세부 정보는 AWS CLI 명령 참조의 [ListObjectsV2](#)를 참조하십시오.

C++

SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::S3::listObjects(const Aws::String &bucketName,
                             Aws::Vector<Aws::String> &keysResult,
                             const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);

    Aws::String continuationToken; // Used for pagination.
    Aws::Vector<Aws::S3::Model::Object> allObjects;

    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }

        auto outcome = s3Client.ListObjectsV2(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: listObjects: " <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        } else {
            Aws::Vector<Aws::S3::Model::Object> objects =
                outcome.GetResult().GetContents();

            allObjects.insert(allObjects.end(), objects.begin(), objects.end());
            continuationToken = outcome.GetResult().GetNextContinuationToken();
        }
    } while (!continuationToken.empty());
}
```

```
std::cout << allObjects.size() << " object(s) found:" << std::endl;

for (const auto &object: allObjects) {
    std::cout << " " << object.GetKey() << std::endl;
    keysResult.push_back(object.GetKey());
}

return true;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListObjectsV2](#)를 참조하세요.

CLI

AWS CLI

버킷의 객체 목록을 보려면

다음 `list-objects-v2` 예시에서는 지정된 버킷의 객체를 나열합니다.

```
aws s3api list-objects-v2 \
  --bucket amzn-s3-demo-bucket
```

출력:

```
{
  "Contents": [
    {
      "LastModified": "2019-11-05T23:11:50.000Z",
      "ETag": "\"621503c373607d548b37cff8778d992c\"",
      "StorageClass": "STANDARD",
      "Key": "doc1.rtf",
      "Size": 391
    },
    {
      "LastModified": "2019-11-05T23:11:50.000Z",
      "ETag": "\"a2cecc36ab7c7fe3a71a273b9d45b1b5\"",
      "StorageClass": "STANDARD",
      "Key": "doc2.rtf",
      "Size": 373
    },
    {
```

```

        "LastModified": "2019-11-05T23:11:50.000Z",
        "ETag": "\"08210852f65a2e9cb999972539a64d68\"",
        "StorageClass": "STANDARD",
        "Key": "doc3.rtf",
        "Size": 399
    },
    {
        "LastModified": "2019-11-05T23:11:50.000Z",
        "ETag": "\"d1852dd683f404306569471af106988e\"",
        "StorageClass": "STANDARD",
        "Key": "doc4.rtf",
        "Size": 6225
    }
]
}

```

- API 세부 정보는 AWS CLI 명령 참조의 [ListObjectsV2](#)를 참조하세요.

Go

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "io"
    "log"
    "os"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"

```

```
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(ctx context.Context, bucketName string)
([]types.Object, error) {
    var err error
    var output *s3.ListObjectsV2Output
    input := &s3.ListObjectsV2Input{
        Bucket: aws.String(bucketName),
    }
    var objects []types.Object
    objectPaginator := s3.NewListObjectsV2Paginator(basics.S3Client, input)
    for objectPaginator.HasMorePages() {
        output, err = objectPaginator.NextPage(ctx)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucketName)
                err = noBucket
            }
            break
        } else {
            objects = append(objects, output.Contents...)
        }
    }
    return objects, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListObjectsV2](#)를 참조하십시오.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Asynchronously lists all objects in the specified S3 bucket.
 *
 * @param bucketName the name of the S3 bucket to list objects for
 * @return a {@link CompletableFuture} that completes when all objects have
 * been listed
 */
public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
    ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .maxKeys(1)
        .build();

    ListObjectsV2Publisher paginator =
        getAsyncClient().listObjectsV2Paginator(initialRequest);
    return paginator.subscribe(response -> {
        response.contents().forEach(s3Object -> {
            logger.info("Object key: " + s3Object.key());
        });
    }).thenRun(() -> {
        logger.info("Successfully listed all objects in the bucket: " +
            bucketName);
    }).exceptionally(ex -> {
        throw new RuntimeException("Failed to list objects", ex);
    });
}
```

페이지 매김을 사용하여 객체를 나열합니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;

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

            Usage:
                <bucketName>\s

            Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
            """;

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

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    /**
     * Lists the objects in the specified S3 bucket.
     *
     * @param s3 the S3Client instance used to interact with Amazon S3
     * @param bucketName the name of the S3 bucket to list the objects from
     */
    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsV2Request listReq = ListObjectsV2Request.builder()
```

```

        .bucket(bucketName)
        .maxKeys(1)
        .build();

        ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
        listRes.stream()
            .flatMap(r -> r.contents().stream())
            .forEach(content -> System.out.println(" Key: " + content.key() +
" size = " + content.size()));

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [ListObjectsV2](#)를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷의 모든 객체를 나열합니다. 객체가 두 개 이상인 경우, 전체 목록을 반복하는 데 IsTruncated 및 NextContinuationToken이 사용됩니다.

```

import {
    S3Client,
    S3ServiceException,
    // This command supersedes the ListObjectsCommand and is the recommended way to
    list objects.
    paginateListObjectsV2,
} from "@aws-sdk/client-s3";

/**

```

```

* Log all of the object keys in a bucket.
* @param {{ bucketName: string, pageSize: string }}
*/
export const main = async ({ bucketName, pageSize }) => {
  const client = new S3Client({});
  /** @type {string[][]} */
  const objects = [];
  try {
    const paginator = paginateListObjectsV2(
      { client, /* Max items per page */ pageSize: Number.parseInt(pageSize) },
      { Bucket: bucketName },
    );

    for await (const page of paginator) {
      objects.push(page.Contents.map((o) => o.Key));
    }
    objects.forEach((objectList, pageNum) => {
      console.log(
        `Page ${pageNum + 1}\n-----\n${objectList.map((o) => `•
${o}`)}.join("\n")}\n`,
      );
    });
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while listing objects for "${bucketName}". The bucket
doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while listing objects for "${bucketName}".
${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 [ListObjectsV2](#)를 참조하십시오.

Kotlin

SDK for Kotlin

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listBucketObjects(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The object is ${myObject.size?.let { calKb(it) }} KBs")
            println("The owner is ${myObject.owner}")
        }
    }
}

private fun calKb(intValue: Long): Long = intValue / 1024
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 [ListObjectsV2](#)를 참조하십시오.

PHP

SDK for PHP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷의 객체를 나열합니다.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $contents = $this->s3client->listObjectsV2([
        'Bucket' => $this->bucketName,
    ]);
    echo "The contents of your bucket are: \n";
    foreach ($contents['Contents'] as $content) {
        echo $content['Key'] . "\n";
    }
} catch (Exception $exception) {
    echo "Failed to list objects in $this->bucketName with error: " .
    $exception->getMessage();
    exit("Please fix error with listing objects before continuing.");
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 [ListObjectsV2](#)를 참조하십시오.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
```

```

        self.key = self.object.key

    @staticmethod
    def list(bucket, prefix=None):
        """
        Lists the objects in a bucket, optionally filtered by a prefix.

        :param bucket: The bucket to query. This is a Boto3 Bucket resource.
        :param prefix: When specified, only objects that start with this prefix
        are listed.
        :return: The list of objects.
        """
        try:
            if not prefix:
                objects = list(bucket.objects.all())
            else:
                objects = list(bucket.objects.filter(Prefix=prefix))
            logger.info(
                "Got objects %s from bucket '%s'", [o.key for o in objects],
                bucket.name
            )
        except ClientError:
            logger.exception("Couldn't get objects for bucket '%s'.",
                bucket.name)
            raise
        else:
            return objects

```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [ListObjectsV2](#)를 참조하십시오.

Ruby

SDK for Ruby

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketListObjectsWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
  def initialize(bucket)
    @bucket = bucket
  end

  # Lists object in a bucket.
  #
  # @param max_objects [Integer] The maximum number of objects to list.
  # @return [Integer] The number of objects listed.
  def list_objects(max_objects)
    count = 0
    puts "The objects in #{@bucket.name} are:"
    @bucket.objects.each do |obj|
      puts "\t#{obj.key}"
      count += 1
      break if count == max_objects
    end
    count
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list objects in bucket #{bucket.name}. Here's why:
    #{e.message}"
    0
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"

  wrapper = BucketListObjectsWrapper.new(Aws::S3::Bucket.new(bucket_name))
  count = wrapper.list_objects(25)
  puts "Listed #{count} objects."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 [ListObjectsV2](#)를 참조하십시오.

Rust

SDK for Rust

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn list_objects(client: &aws_sdk_s3::Client, bucket: &str) ->
    Result<(), S3ExampleError> {
    let mut response = client
        .list_objects_v2()
        .bucket(bucket.to_owned())
        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

    while let Some(result) = response.next().await {
        match result {
            Ok(output) => {
                for object in output.contents() {
                    println!(" - {}", object.key().unwrap_or("Unknown"));
                }
            }
            Err(err) => {
                eprintln!("{err:?}")
            }
        }
    }

    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 [ListObjectsV2](#)을 참조하십시오.

SAP ABAP

SDK for SAP ABAP API

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.
    oo_result = lo_s3->listobjectsv2(          " oo_result is returned for
testing purposes. "
    iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- API 세부 정보는 AWSSDK for SAP ABAP API의 [ListObjectsV2](#)을 참조하십시오.

Swift

SDK for Swift

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSS3

public func listBucketFiles(bucket: String) async throws -> [String] {
    do {
        let input = ListObjectsV2Input(
            bucket: bucket
        )
```

```

        // Use "Paginated" to get all the objects.
        // This lets the SDK handle the 'continuationToken' in
        "ListObjectsV2Output".
        let output = client.listObjectsV2Paginated(input: input)
        var names: [String] = []

        for try await page in output {
            guard let objList = page.contents else {
                print("ERROR: listObjectsV2Paginated returned nil contents.")
                continue
            }

            for obj in objList {
                if let objName = obj.key {
                    names.append(objName)
                }
            }
        }

        return names
    }
    catch {
        print("ERROR: ", dump(error, name: "Listing objects."))
        throw error
    }
}

```

- API 세부 정보는 Swift용 AWS SDK API 참조의 [ListObjectsV2](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutBucketAccelerateConfiguration** 사용

다음 코드 예시는 PutBucketAccelerateConfiguration의 사용 방법을 보여 줍니다.

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// Amazon Simple Storage Service (Amazon S3) Transfer Acceleration is a
/// bucket-level feature that enables you to perform faster data transfers
/// to Amazon S3. This example shows how to configure Transfer
/// Acceleration.
/// </summary>
public class TransferAcceleration
{
    /// <summary>
    /// The main method initializes the client object and sets the
    /// Amazon Simple Storage Service (Amazon S3) bucket name before
    /// calling EnableAccelerationAsync.
    /// </summary>
    public static async Task Main()
    {
        var s3Client = new AmazonS3Client();
        const string bucketName = "amzn-s3-demo-bucket";

        await EnableAccelerationAsync(s3Client, bucketName);
    }

    /// <summary>
    /// This method sets the configuration to enable transfer acceleration
    /// for the bucket referred to in the bucketName parameter.
    /// </summary>
    /// <param name="client">An Amazon S3 client used to enable the
    /// acceleration on an Amazon S3 bucket.</param>
```



```

    the
    /// <param name="bucketName">The name of the Amazon S3 bucket for which
    /// method will be enabling acceleration.</param>
    private static async Task EnableAccelerationAsync(AmazonS3Client client,
    string bucketName)
    {
        try
        {
            var putRequest = new PutBucketAccelerateConfigurationRequest
            {
                BucketName = bucketName,
                AccelerateConfiguration = new AccelerateConfiguration
                {
                    Status = BucketAccelerateStatus.Enabled,
                },
            };
            await client.PutBucketAccelerateConfigurationAsync(putRequest);

            var getRequest = new GetBucketAccelerateConfigurationRequest
            {
                BucketName = bucketName,
            };
            var response = await
client.GetBucketAccelerateConfigurationAsync(getRequest);

            Console.WriteLine($"Acceleration state = '{response.Status}' ");
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error occurred. Message: '{ex.Message}' when
setting transfer acceleration");
        }
    }
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [PutBucketAccelerateConfiguration](#)을 참조하세요.

CLI

AWS CLI

버킷의 가속화 구성 설정

다음 `put-bucket-accelerate-configuration` 예시에서는 지정된 버킷의 가속화 구성을 활성화합니다.

```
aws s3api put-bucket-accelerate-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --accelerate-configuration Status=Enabled
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 참조의 [PutBucketAccelerateConfiguration](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷의 전송 가속화를 활성화합니다.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')  
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AccelerateConfiguration_Status $statusVal
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [PutBucketAccelerateConfiguration](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 S3 버킷의 전송 가속화를 활성화합니다.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')  
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AccelerateConfiguration_Status $statusVal
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [PutBucketAccelerateConfiguration](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutBucketAcl** 사용

다음 코드 예시는 PutBucketAcl의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [액세스 제어 목록\(ACL\) 관리](#)

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Creates an Amazon S3 bucket with an ACL to control access to the
/// bucket and the objects stored in it.
/// </summary>
/// <param name="client">The initialized client object used to create
/// an Amazon S3 bucket, with an ACL applied to the bucket.
/// </param>
/// <param name="region">The AWS Region where the bucket will be
created.</param>
/// <param name="newBucketName">The name of the bucket to create.</param>
/// <returns>A boolean value indicating success or failure.</returns>
public static async Task<bool> CreateBucketUseCannedACLAsync(IAmazonS3
client, S3Region region, string newBucketName)
{
    try
    {
        // Create a new Amazon S3 bucket with Canned ACL.
```

```

        var putBucketRequest = new PutBucketRequest()
        {
            BucketName = newBucketName,
            BucketRegion = region,
            CannedACL = S3CannedACL.LogDeliveryWrite,
        };

        PutBucketResponse putBucketResponse = await
client.PutBucketAsync(putBucketRequest);

        return putBucketResponse.HttpStatusCode ==
System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Amazon S3 error: {ex.Message}");
    }

    return false;
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [PutBucketAcl](#)을 참조하십시오.

C++

SDK for C++

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::S3::putBucketAcl(const Aws::String &bucketName, const Aws::String
&ownerID,
                                const Aws::String &granteePermission,
                                const Aws::String &granteeType, const Aws::String
&granteeID,
                                const Aws::String &granteeEmailAddress,

```

```
const Aws::String &granteeURI, const
Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutBucketAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);

    Aws::S3::Model::PutBucketAclOutcome outcome =
        s3Client.PutBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &error = outcome.GetError();

        std::cerr << "Error: putBucketAcl: " << error.GetExceptionName()
```

```

        << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the bucket '" << bucketName
        << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type
enumeration.
/*!
 \param access: Human readable string.
 \return Permission: A Permission enum.
 */

Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type
enumeration.
/*!
 \param type: Human readable string.
 \return Type: Type enumeration
 */

Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
}

```

```
    return Aws::S3::Model::Type::NOT_SET;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [PutBucketAcl](#)을 참조하세요.

CLI

AWS CLI

이 예제에서는 두 명의 AWS 사용자(user1@example.com 및 user2@example.com)에게 full control 권한을 부여하고 모든 사용자에게 read 권한을 부여합니다.

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-bucket --grant-full-
control emailaddress=user1@example.com,emailaddress=user2@example.com --grant-
read uri=http://acs.amazonaws.com/groups/global/AllUsers
```

사용자 지정 ACL에 대한 자세한 내용은 <http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html>을 참조하세요(put-bucket-acl과 같은 s3api ACL 명령은 동일한 간편 인수 표기법을 사용함).

- API 세부 정보는 AWS CLI 명령 참조의 [PutBucketAcl](#)을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;
import software.amazon.awssdk.services.s3.model.Grant;
import software.amazon.awssdk.services.s3.model.Permission;
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;
```

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Type;

import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetAcl {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
            <bucketName> <id>\s

            Where:
            bucketName - The Amazon S3 bucket to grant permissions on.\s
            id - The ID of the owner of this bucket (you can get this value
from the AWS Management Console).
            """;

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

        String bucketName = args[0];
        String id = args[1];
        System.out.format("Setting access \n");
        System.out.println(" in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setBucketAcl(s3, bucketName, id);
        System.out.println("Done!");
    }
}
```



```

        s3.close();
    }

    /**
     * Sets the Access Control List (ACL) for an Amazon S3 bucket.
     *
     * @param s3 the S3Client instance to be used for the operation
     * @param bucketName the name of the S3 bucket to set the ACL for
     * @param id the ID of the AWS user or account that will be granted full
control of the bucket
     * @throws S3Exception if an error occurs while setting the bucket ACL
     */
    public static void setBucketAcl(S3Client s3, String bucketName, String id) {
        try {
            Grant ownerGrant = Grant.builder()
                .grantee(builder -> builder.id(id)
                    .type(Type.CANONICAL_USER))
                .permission(Permission.FULL_CONTROL)
                .build();

            List<Grant> grantList2 = new ArrayList<>();
            grantList2.add(ownerGrant);

            AccessControlPolicy acl = AccessControlPolicy.builder()
                .owner(builder -> builder.id(id))
                .grants(grantList2)
                .build();

            PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
                .bucket(bucketName)
                .accessControlPolicy(acl)
                .build();

            s3.putBucketAcl(putAclReq);

        } catch (S3Exception e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [PutBucketAcl](#)을 참조하십시오.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷 ACL을 적용합니다.

```
import {
  PutBucketAclCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Grant read access to a user using their canonical AWS account ID.
 *
 * Most Amazon S3 use cases don't require the use of access control lists (ACLs).
 * We recommend that you disable ACLs, except in unusual circumstances where
 * you need to control access for each object individually. Consider a policy
 * instead.
 * For more information see https://docs.aws.amazon.com/AmazonS3/latest/
 * userguide/bucket-policies.html.
 * @param {{ bucketName: string, granteeCanonicalUserId: string,
 *   ownerCanonicalUserId }}
 */
export const main = async ({
  bucketName,
  granteeCanonicalUserId,
  ownerCanonicalUserId,
}) => {
  const client = new S3Client({});
  const command = new PutBucketAclCommand({
    Bucket: bucketName,
    AccessControlPolicy: {
      Grants: [
        {
          Grantee: {
```

```

        // The canonical ID of the user. This ID is an obfuscated form of
        your AWS account number.
        // It's unique to Amazon S3 and can't be found elsewhere.
        // For more information, see https://docs.aws.amazon.com/AmazonS3/latest/userguide/finding-canonical-user-id.html.
        ID: granteeCanonicalUserId,
        Type: "CanonicalUser",
    },
    // One of FULL_CONTROL | READ | WRITE | READ_ACP | WRITE_ACP
    // https://docs.aws.amazon.com/AmazonS3/latest/API/API\_Grant.html#AmazonS3-Type-Grant-Permission
    Permission: "READ",
},
],
Owner: {
    ID: ownerCanonicalUserId,
},
},
});

try {
    await client.send(command);
    console.log(`Granted READ access to ${bucketName}`);
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while setting ACL for bucket ${bucketName}. The bucket
            doesn't exist.`,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while setting ACL for bucket ${bucketName}.
            ${caught.name}: ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};

```

- 자세한 정보는 [AWS SDK for JavaScript 개발자 안내서](#)를 참조하세요.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 [PutBucketAcl](#)을 참조하십시오.

Kotlin

SDK for Kotlin

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun setBucketAcl(
    bucketName: String,
    idVal: String,
) {
    val myGrant =
        Grantee {
            id = idVal
            type = Type.CanonicalUser
        }

    val ownerGrant =
        Grant {
            grantee = myGrant
            permission = Permission.FullControl
        }

    val grantList = mutableListOf<Grant>()
    grantList.add(ownerGrant)

    val ownerOb =
        Owner {
            id = idVal
        }

    val acl =
        AccessControlPolicy {
            owner = ownerOb
            grants = grantList
        }
}
```

```

    }

    val request =
        PutBucketAclRequest {
            bucket = bucketName
            accessControlPolicy = acl
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.putBucketAcl(request)
        println("An ACL was successfully set on $bucketName")
    }
}

```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 [PutBucketAcl](#)를 참조하십시오.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def grant_log_delivery_access(self):
        """

```

```

Grant the AWS Log Delivery group write access to the bucket so that
Amazon S3 can deliver access logs to the bucket. This is the only
recommended
use of an S3 bucket ACL.
"""
try:
    acl = self.bucket.Acl()
    # Putting an ACL overwrites the existing ACL. If you want to preserve
    # existing grants, append new grants to the list of existing grants.
    grants = acl.grants if acl.grants else []
    grants.append(
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery",
            },
            "Permission": "WRITE",
        }
    )
    acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
    logger.info("Granted log delivery access to bucket '%s'",
self.bucket.name)
except ClientError:
    logger.exception("Couldn't add ACL to bucket '%s'.",
self.bucket.name)
    raise

```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [PutBucketAcl](#)를 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

TRY.

```

" Example: Grant log delivery access to a bucket
" iv_grantwrite = 'uri=http://acs.amazonaws.com/groups/s3/LogDelivery'
lo_s3->putbucketacl(
    iv_bucket = iv_bucket_name
    iv_grantwrite = iv_grantwrite ).
MESSAGE 'Bucket ACL updated.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 [PutBucketAcl](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutBucketCors** 사용

다음 코드 예시는 PutBucketCors의 사용 방법을 보여 줍니다.

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

/// <summary>
/// Add CORS configuration to the Amazon S3 bucket.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used
/// to apply the CORS configuration to an Amazon S3 bucket.</param>
/// <param name="configuration">The CORS configuration to apply.</param>
private static async Task PutCORSConfigurationAsync(AmazonS3Client
client, CORSConfiguration configuration)
{

```

```

        PutCORSConfigurationRequest request = new
PutCORSConfigurationRequest()
    {
        BucketName = BucketName,
        Configuration = configuration,
    };

    _ = await client.PutCORSConfigurationAsync(request);
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [PutBucketCors](#)를 참조하세요.

CLI

AWS CLI

다음 예시에서는 `www.example.com`의 PUT, POST, 및 DELETE 요청을 활성화하고 모든 도메인의 GET 요청을 활성화합니다.

```
aws s3api put-bucket-cors --bucket amzn-s3-demo-bucket --cors-configuration file://cors.json
```

cors.json:

```

{
  "CORSRules": [
    {
      "AllowedOrigins": ["http://www.example.com"],
      "AllowedHeaders": ["*"],
      "AllowedMethods": ["PUT", "POST", "DELETE"],
      "MaxAgeSeconds": 3000,
      "ExposeHeaders": ["x-amz-server-side-encryption"]
    },
    {
      "AllowedOrigins": ["*"],
      "AllowedHeaders": ["Authorization"],
      "AllowedMethods": ["GET"],
      "MaxAgeSeconds": 3000
    }
  ]
}

```


- API 세부 정보는 AWS CLI 명령 참조의 [PutBucketCors](#)를 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;

import java.util.ArrayList;
import java.util.List;

import software.amazon.awssdk.services.s3.model.GetBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.GetBucketCorsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.CORSRule;
import software.amazon.awssdk.services.s3.model.CORSConfiguration;
import software.amazon.awssdk.services.s3.model.PutBucketCorsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class S3Cors {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
        <bucketName> <accountId>\s
```

```

        Where:
            bucketName - The Amazon S3 bucket to upload an object into.
            accountId - The id of the account that owns the Amazon S3 bucket.
        """;

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

    String bucketName = args[0];
    String accountId = args[1];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setCorsInformation(s3, bucketName, accountId);
    getBucketCorsInformation(s3, bucketName, accountId);
    deleteBucketCorsInformation(s3, bucketName, accountId);
    s3.close();
}

/**
 * Deletes the CORS (Cross-Origin Resource Sharing) configuration for an
 * Amazon S3 bucket.
 *
 * @param s3          the {@link S3Client} instance used to interact with
 * the Amazon S3 service
 * @param bucketName  the name of the Amazon S3 bucket for which the CORS
 * configuration should be deleted
 * @param accountId   the expected AWS account ID of the bucket owner
 *
 * @throws S3Exception if an error occurs while deleting the CORS
 * configuration for the bucket
 */
public static void deleteBucketCorsInformation(S3Client s3, String
bucketName, String accountId) {
    try {
        DeleteBucketCorsRequest bucketCorsRequest =
DeleteBucketCorsRequest.builder()
            .bucket(bucketName)
            .expectedBucketOwner(accountId)

```

```

        .build();

        s3.deleteBucketCors(bucketCorsRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

/**
 * Retrieves the CORS (Cross-Origin Resource Sharing) configuration for the
 * specified S3 bucket.
 *
 * @param s3 the S3Client instance to use for the operation
 * @param bucketName the name of the S3 bucket to retrieve the CORS
 * configuration for
 * @param accountId the expected bucket owner's account ID
 *
 * @throws S3Exception if there is an error retrieving the CORS configuration
 */
public static void getBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
    try {
        GetBucketCorsRequest bucketCorsRequest =
GetBucketCorsRequest.builder()
            .bucket(bucketName)
            .expectedBucketOwner(accountId)
            .build();

        GetBucketCorsResponse corsResponse =
s3.getBucketCors(bucketCorsRequest);
        List<CORSRule> corsRules = corsResponse.corsRules();
        for (CORSRule rule : corsRules) {
            System.out.println("allowOrigins: " + rule.allowedOrigins());
            System.out.println("AllowedMethod: " + rule.allowedMethods());
        }

    } catch (S3Exception e) {

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

```

```
/**
 * Sets the Cross-Origin Resource Sharing (CORS) rules for an Amazon S3
bucket.
 *
 * @param s3 The S3Client object used to interact with the Amazon S3 service.
 * @param bucketName The name of the S3 bucket to set the CORS rules for.
 * @param accountId The AWS account ID of the bucket owner.
 */
public static void setCorsInformation(S3Client s3, String bucketName, String
accountId) {
    List<String> allowMethods = new ArrayList<>();
    allowMethods.add("PUT");
    allowMethods.add("POST");
    allowMethods.add("DELETE");

    List<String> allowOrigins = new ArrayList<>();
    allowOrigins.add("http://example.com");
    try {
        // Define CORS rules.
        CORSRule corsRule = CORSRule.builder()
            .allowedMethods(allowMethods)
            .allowedOrigins(allowOrigins)
            .build();

        List<CORSRule> corsRules = new ArrayList<>();
        corsRules.add(corsRule);
        CORSConfiguration configuration = CORSConfiguration.builder()
            .corsRules(corsRules)
            .build();

        PutBucketCorsRequest putBucketCorsRequest =
PutBucketCorsRequest.builder()
            .bucket(bucketName)
            .corsConfiguration(configuration)
            .expectedBucketOwner(accountId)
            .build();

        s3.putBucketCors(putBucketCorsRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [PutBucketCors](#)를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

CORS 규칙을 추가합니다.

```
import {
  PutBucketCorsCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Allows cross-origin requests to an S3 bucket by setting the CORS
 * configuration.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new PutBucketCorsCommand({
        Bucket: bucketName,
        CORSConfiguration: {
          CORSRules: [
            {
              // Allow all headers to be sent to this bucket.
              AllowedHeaders: ["*"],
              // Allow only GET and PUT methods to be sent to this bucket.
            }
          ]
        }
      })
    );
  } catch (err) {
    if (err instanceof S3ServiceException) {
      console.log("S3ServiceException: ", err.message);
    } else {
      console.log("Unknown error: ", err);
    }
  }
}
```

```

        AllowedMethods: ["GET", "PUT"],
        // Allow only requests from the specified origin.
        AllowedOrigins: ["https://www.example.com"],
        // Allow the entity tag (ETag) header to be returned in the
response. The ETag header
        // The entity tag represents a specific version of the object. The
ETag reflects
        // changes only to the contents of an object, not its metadata.
        ExposeHeaders: ["ETag"],
        // How long the requesting browser should cache the preflight
response. After
        // this time, the preflight request will have to be made again.
        MaxAgeSeconds: 3600,
    },
],
},
}),
);
console.log(`Successfully set CORS rules for bucket: ${bucketName}`);
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while setting CORS rules for ${bucketName}. The bucket
doesn't exist.`,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while setting CORS rules for ${bucketName}.
${caught.name}: ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};

```

- 자세한 정보는 [AWS SDK for JavaScript 개발자 안내서](#)를 참조하세요.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 [PutBucketCors](#)를 참조하십시오.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def put_cors(self, cors_rules):
        """
        Apply CORS rules to the bucket. CORS rules specify the HTTP actions that
        are
        allowed from other domains.

        :param cors_rules: The CORS rules to apply.
        """
        try:
            self.bucket.Cors().put(CORSConfiguration={"CORSRules": cors_rules})
            logger.info(
                "Put CORS rules %s for bucket '%s'.", cors_rules,
                self.bucket.name
            )
        except ClientError:
            logger.exception("Couldn't put CORS rules for bucket %s.",
                self.bucket.name)
            raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [PutBucketCors](#)를 참조하십시오.

Ruby

SDK for Ruby

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
  # an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Sets CORS rules on a bucket.
  #
  # @param allowed_methods [Array<String>] The types of HTTP requests to allow.
  # @param allowed_origins [Array<String>] The origins to allow.
  # @returns [Boolean] True if the CORS rules were set; otherwise, false.
  def set_cors(allowed_methods, allowed_origins)
    @bucket_cors.put(
      cors_configuration: {
        cors_rules: [
          {
            allowed_methods: allowed_methods,
            allowed_origins: allowed_origins,
            allowed_headers: %w[*],
            max_age_seconds: 3600
          }
        ]
      }
    )
  end
end
```



```

    }
  )
  true
rescue Aws::Errors::ServiceError => e
  puts "Couldn't set CORS rules for #{@bucket_cors.bucket.name}. Here's why:
#{e.message}"
  false
end
end
end

```

- API 세부 정보는 AWS SDK for Ruby API 참조의 [PutBucketCors](#)를 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

TRY.
  " Example: Allow PUT, POST, DELETE methods from http://www.example.com
  lo_s3->putbucketcors(
    iv_bucket = iv_bucket_name
    io_corsconfiguration = NEW /aws1/cl_s3_corsconfiguration(
      it_corsrules = it_cors_rules ) ).
  MESSAGE 'Bucket CORS configuration set.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 [PutBucketCors](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutBucketEncryption** 사용

다음 코드 예시는 PutBucketEncryption의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [객체 스토리지 시작하기](#)

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Set the bucket server side encryption to use AWSKMS with a customer-
managed key id.
/// </summary>
/// <param name="bucketName">Name of the bucket.</param>
/// <param name="kmsKeyId">The Id of the KMS Key.</param>
/// <returns>True if successful.</returns>
public static async Task<bool> SetBucketServerSideEncryption(string
bucketName, string kmsKeyId)
{
    var serverSideEncryptionByDefault = new ServerSideEncryptionConfiguration
    {
        ServerSideEncryptionRules = new List<ServerSideEncryptionRule>
        {
            new ServerSideEncryptionRule
            {
                ServerSideEncryptionByDefault = new
                ServerSideEncryptionByDefault
```

```

        {
            ServerSideEncryptionAlgorithm =
ServerSideEncryptionMethod.AWSKMS,
            ServerSideEncryptionKeyManagementServiceKeyId = kmsKeyId
        }
    }
};
try
{
    var encryptionResponse = await _s3Client.PutBucketEncryptionAsync(new
PutBucketEncryptionRequest
    {
        BucketName = bucketName,
        ServerSideEncryptionConfiguration =
serverSideEncryptionByDefault,
    });

    return encryptionResponse.HttpStatusCode == HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine(ex.ErrorCode == "AccessDenied"
        ? $"This account does not have permission to set encryption on
{bucketName}, please try again."
        : $"Unable to set bucket encryption for bucket {bucketName},
{ex.Message}");
}
return false;
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [PutBucketEncryption](#)을 참조하세요.

CLI

AWS CLI

버킷의 서버 측 암호화 구성

다음 put-bucket-encryption 예시에서는 AES256 암호화를 지정된 버킷의 기본값으로 설정합니다.

```
aws s3api put-bucket-encryption \  
  --bucket amzn-s3-demo-bucket \  
  --server-side-encryption-configuration '{"Rules":  
  [{"ApplyServerSideEncryptionByDefault": {"SSEAlgorithm": "AES256"}}]}'
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 참조의 [PutBucketEncryption](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 버킷에서 Amazon S3 관리형 키(SSE-S3)를 사용하여 기본 AES256 서버 측 암호화를 활성화합니다.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
  @{ServerSideEncryptionAlgorithm = "AES256"}}  
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -  
  ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [PutBucketEncryption](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 버킷에서 Amazon S3 관리형 키(SSE-S3)를 사용하여 기본 AES256 서버 측 암호화를 활성화합니다.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
  @{ServerSideEncryptionAlgorithm = "AES256"}}  
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -  
  ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [PutBucketEncryption](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutBucketLifecycleConfiguration** 사용

다음 코드 예시는 PutBucketLifecycleConfiguration의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [미완료 멀티파트 업로드 삭제](#)
- [S3를 사용하여 대규모 메시지 관리](#)
- [버전이 지정된 객체 작업](#)

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Adds lifecycle configuration information to the S3 bucket named in
/// the bucketName parameter.
/// </summary>
/// <param name="client">The S3 client used to call the
/// PutLifecycleConfigurationAsync method.</param>
/// <param name="bucketName">A string representing the S3 bucket to
/// which configuration information will be added.</param>
/// <param name="configuration">A LifecycleConfiguration object that
/// will be applied to the S3 bucket.</param>
public static async Task AddExampleLifecycleConfigAsync(IAmazonS3 client,
string bucketName, LifecycleConfiguration configuration)
{
    var request = new PutLifecycleConfigurationRequest()
    {
        BucketName = bucketName,
        Configuration = configuration,
    };
    var response = await client.PutLifecycleConfigurationAsync(request);
}
```

```
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 [PutBucketLifecycleConfiguration](#)을 참조하세요.

CLI

AWS CLI

다음 명령은 amzn-s3-demo-bucket이라는 버킷에 수명 주기 구성을 적용합니다.

```
aws s3api put-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket --
lifecycle-configuration file://lifecycle.json
```

lifecycle.json 파일은 다음 두 규칙을 지정하는 현재 폴더의 JSON 문서입니다.

```
{
  "Rules": [
    {
      "ID": "Move rotated logs to Glacier",
      "Prefix": "rotated/",
      "Status": "Enabled",
      "Transitions": [
        {
          "Date": "2015-11-10T00:00:00.000Z",
          "StorageClass": "GLACIER"
        }
      ]
    },
    {
      "Status": "Enabled",
      "Prefix": "",
      "NoncurrentVersionTransitions": [
        {
          "NoncurrentDays": 2,
          "StorageClass": "GLACIER"
        }
      ],
      "ID": "Move old versions to Glacier"
    }
  ]
}
```

```
]
}
```

첫 번째 규칙은 rotated 접두사가 있는 파일을 지정된 날짜에 Glacier로 옮깁니다. 두 번째 규칙은 이전 객체 버전이 더 이상 최신 버전이 아닌 경우 Glacier로 옮깁니다. 허용되는 타임스탬프 형식에 대한 자세한 내용은 AWS CLI 사용자 안내서의 파라미터 값 지정을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [PutBucketLifecycleConfiguration](#)을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.Transition;
import
    software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;
import
    software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;
import software.amazon.awssdk.services.s3.model.TransitionStorageClass;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```

* <p>
* For more information, see the following documentation topic:
* <p>
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/

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

            Usage:
                <bucketName> <accountId>\s

            Where:
                bucketName - The Amazon Simple Storage Service (Amazon S3) bucket
to upload an object into.
                accountId - The id of the account that owns the Amazon S3 bucket.
            "";

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

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setLifecycleConfig(s3, bucketName, accountId);
        getLifecycleConfig(s3, bucketName, accountId);
        deleteLifecycleConfig(s3, bucketName, accountId);
        System.out.println("You have successfully created, updated, and deleted a
Lifecycle configuration");
        s3.close();
    }

    /**
     * Sets the lifecycle configuration for an Amazon S3 bucket.
     *
     * @param s3          The Amazon S3 client to use for the operation.

```



```
* @param bucketName    The name of the Amazon S3 bucket.
* @param accountId      The expected owner of the Amazon S3 bucket.
*
* @throws S3Exception if there is an error setting the lifecycle
configuration.
*/
public static void setLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
    try {
        // Create a rule to archive objects with the "glacierobjects/" prefix
to the

        // S3 Glacier Flexible Retrieval storage class immediately.
        LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
            .prefix("glacierobjects/")
            .build();

        Transition transition = Transition.builder()
            .storageClass(TransitionStorageClass.GLACIER)
            .days(0)
            .build();

        LifecycleRule rule1 = LifecycleRule.builder()
            .id("Archive immediately rule")
            .filter(ruleFilter)
            .transitions(transition)
            .status(ExpirationStatus.ENABLED)
            .build();

        // Create a second rule.
        Transition transition2 = Transition.builder()
            .storageClass(TransitionStorageClass.GLACIER)
            .days(0)
            .build();

        List<Transition> transitionList = new ArrayList<>();
        transitionList.add(transition2);

        LifecycleRuleFilter ruleFilter2 = LifecycleRuleFilter.builder()
            .prefix("glacierobjects/")
            .build();

        LifecycleRule rule2 = LifecycleRule.builder()
            .id("Archive and then delete rule")
            .filter(ruleFilter2)
```

```

        .transitions(transitionList)
        .status(ExpirationStatus.ENABLED)
        .build();

// Add the LifecycleRule objects to an ArrayList.
ArrayList<LifecycleRule> ruleList = new ArrayList<>();
ruleList.add(rule1);
ruleList.add(rule2);

BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
    .rules(ruleList)
    .build();

PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
    .builder()
    .bucket(bucketName)
    .lifecycleConfiguration(lifecycleConfiguration)
    .expectedBucketOwner(accountId)
    .build();

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

/**
 * Retrieves the lifecycle configuration for an Amazon S3 bucket and adds a
 * new lifecycle rule to it.
 *
 * @param s3 the S3Client instance used to interact with Amazon S3
 * @param bucketName the name of the Amazon S3 bucket
 * @param accountId the expected owner of the Amazon S3 bucket
 */
public static void getLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
    try {
        GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest = GetBucketLifecycleConfigurationRequest

```

```
.builder()
.bucket(bucketName)
.expectedBucketOwner(accountId)
.build();

GetBucketLifecycleConfigurationResponse response = s3

.getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
List<LifecycleRule> newList = new ArrayList<>();
List<LifecycleRule> rules = response.rules();
for (LifecycleRule rule : rules) {
    newList.add(rule);
}

// Add a new rule with both a prefix predicate and a tag predicate.
LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
    .prefix("YearlyDocuments/")
    .build();

Transition transition = Transition.builder()
    .storageClass(TransitionStorageClass.GLACIER)
    .days(3650)
    .build();

LifecycleRule rule1 = LifecycleRule.builder()
    .id("NewRule")
    .filter(ruleFilter)
    .transitions(transition)
    .status(ExpirationStatus.ENABLED)
    .build();

// Add the new rule to the list.
newList.add(rule1);
BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
    .rules(newList)
    .build();

PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
    .builder()
    .bucket(bucketName)
    .lifecycleConfiguration(lifecycleConfiguration)
    .expectedBucketOwner(accountId)
```

```

        .build();

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

/**
 * Deletes the lifecycle configuration for an Amazon S3 bucket.
 *
 * @param s3 the {@link S3Client} to use for the operation
 * @param bucketName the name of the S3 bucket
 * @param accountId the expected account owner of the S3 bucket
 *
 * @throws S3Exception if an error occurs while deleting the lifecycle
configuration
 */
public static void deleteLifecycleConfig(S3Client s3, String bucketName,
String accountId) {
    try {
        DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
DeleteBucketLifecycleRequest
        .builder()
        .bucket(bucketName)
        .expectedBucketOwner(accountId)
        .build();

        s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [PutBucketLifecycleConfiguration](#)을 참조 하십시오.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def put_lifecycle_configuration(self, lifecycle_rules):
        """
        Apply a lifecycle configuration to the bucket. The lifecycle
        configuration can
            be used to archive or delete the objects in the bucket according to
        specified
            parameters, such as a number of days.

        :param lifecycle_rules: The lifecycle rules to apply.
        """
        try:
            self.bucket.LifecycleConfiguration().put(
                LifecycleConfiguration={"Rules": lifecycle_rules}
            )
            logger.info(
                "Put lifecycle rules %s for bucket '%s'.",
                lifecycle_rules,
                self.bucket.name,
            )
```

```
except ClientError:
    logger.exception(
        "Couldn't put lifecycle rules for bucket '%s'.", self.bucket.name
    )
    raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [PutBucketLifecycleConfiguration](#)를 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.
    " Example: Expire objects with prefix 'logs/' after 30 days
    lo_s3->putbucketlifecycleconf(
        iv_bucket = iv_bucket_name
        io_lifecycleconfiguration = NEW /aws1/cl_s3_bucketlcconf(
            it_rules = it_lifecycle_rule ) ).
    MESSAGE 'Bucket lifecycle configuration set.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 [PutBucketLifecycleConfiguration](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutBucketLogging** 사용

다음 코드 예시는 PutBucketLogging의 사용 방법을 보여 줍니다.

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

/// <summary>
/// This example shows how to enable logging on an Amazon Simple Storage
/// Service (Amazon S3) bucket. You need to have two Amazon S3 buckets for
/// this example. The first is the bucket for which you wish to enable
/// logging, and the second is the location where you want to store the
/// logs.
/// </summary>
public class ServerAccessLogging
{
    private static IConfiguration _configuration = null!;

    public static async Task Main()
    {
        LoadConfig();

        string bucketName = _configuration["BucketName"];
        string logBucketName = _configuration["LogBucketName"];
        string logObjectKeyPrefix = _configuration["LogObjectKeyPrefix"];
        string accountId = _configuration["AccountId"];

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
```

```

        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
        IAmazonS3 client = new AmazonS3Client();

        try
        {
            // Update bucket policy for target bucket to allow delivery of
logs to it.
            await SetBucketPolicyToAllowLogDelivery(
                client,
                bucketName,
                logBucketName,
                logObjectKeyPrefix,
                accountId);

            // Enable logging on the source bucket.
            await EnableLoggingAsync(
                client,
                bucketName,
                logBucketName,
                logObjectKeyPrefix);
        }
        catch (AmazonS3Exception e)
        {
            Console.WriteLine($"Error: {e.Message}");
        }
    }

    /// <summary>
    /// This method grants appropriate permissions for logging to the
    /// Amazon S3 bucket where the logs will be stored.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client which will be
used
    /// to apply the bucket policy.</param>
    /// <param name="sourceBucketName">The name of the source bucket.</param>
    /// <param name="logBucketName">The name of the bucket where logging
    /// information will be stored.</param>
    /// <param name="logPrefix">The logging prefix where the logs should be
delivered.</param>
    /// <param name="accountId">The account id of the account where the
source bucket exists.</param>
    /// <returns>Async task.</returns>
    public static async Task SetBucketPolicyToAllowLogDelivery(

```



```

        IAmazonS3 client,
        string sourceBucketName,
        string logBucketName,
        string logPrefix,
        string accountId)
    {
        var resourceArn = @""arn:aws:s3::" + logBucketName + "/" +
logPrefix + @"""";

        var newPolicy = @"{
            ""Statement"": [{
                ""Sid"": ""S3ServerAccessLogsPolicy"",
                ""Effect"": ""Allow"",
                ""Principal"": { ""Service"":
""logging.s3.amazonaws.com"" },
                ""Action"": [""s3:PutObject""],
                ""Resource"": ["" + resourceArn + @""],
                ""Condition"": {
                    ""ArnLike"": { ""aws:SourceArn"":
""arn:aws:s3::" + sourceBucketName + @"" },
                    ""StringEquals"": { ""aws:SourceAccount"": """" +
accountId + @"""" }
                }
            }]
        }";

        Console.WriteLine($"The policy to apply to bucket {logBucketName} to
enable logging:");
        Console.WriteLine(newPolicy);

        PutBucketPolicyRequest putRequest = new PutBucketPolicyRequest
        {
            BucketName = logBucketName,
            Policy = newPolicy,
        };
        await client.PutBucketPolicyAsync(putRequest);
        Console.WriteLine("Policy applied.");
    }

    /// <summary>
    /// This method enables logging for an Amazon S3 bucket. Logs will be
stored
    /// in the bucket you selected for logging. Selected prefix
    /// will be prepended to each log object.
    /// </summary>

```

```

    /// <param name="client">The initialized Amazon S3 client which will be
used
    /// to configure and apply logging to the selected Amazon S3 bucket.</
param>
    /// <param name="bucketName">The name of the Amazon S3 bucket for which
you
    /// wish to enable logging.</param>
    /// <param name="logBucketName">The name of the Amazon S3 bucket where
logging
    /// information will be stored.</param>
    /// <param name="logObjectKeyPrefix">The prefix to prepend to each
    /// object key.</param>
    /// <returns>Async task.</returns>
    public static async Task EnableLoggingAsync(
        IAmazonS3 client,
        string bucketName,
        string logBucketName,
        string logObjectKeyPrefix)
    {
        Console.WriteLine($"Enabling logging for bucket {bucketName}.");
        var loggingConfig = new S3BucketLoggingConfig
        {
            TargetBucketName = logBucketName,
            TargetPrefix = logObjectKeyPrefix,
        };

        var putBucketLoggingRequest = new PutBucketLoggingRequest
        {
            BucketName = bucketName,
            LoggingConfig = loggingConfig,
        };
        await client.PutBucketLoggingAsync(putBucketLoggingRequest);
        Console.WriteLine($"Logging enabled.");
    }

    /// <summary>
    /// Loads configuration from settings files.
    /// </summary>
    public static void LoadConfig()
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
    }

```

```

        .AddJsonFile("settings.local.json", true) // Optionally, load
local settings.
        .Build();
    }
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [PutBucketLogging](#)을 참조하세요.

CLI

AWS CLI

예시 1: 버킷 정책 로깅을 설정하려면

다음 put-bucket-logging 예제에서는 amzn-s3-demo-bucket에 대한 로깅 정책을 설정합니다. 먼저 put-bucket-policy 명령을 사용하여 버킷 정책에서 로깅 서비스 위탁자 권한을 부여합니다.

```

aws s3api put-bucket-policy \
  --bucket amzn-s3-demo-bucket \
  --policy file://policy.json

```

policy.json의 콘텐츠:

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3ServerAccessLogsPolicy",
      "Effect": "Allow",
      "Principal": {"Service": "logging.s3.amazonaws.com"},
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/Logs/*",
      "Condition": {
        "ArnLike": {"aws:SourceARN": "arn:aws:s3:::SOURCE-BUCKET-NAME"},
        "StringEquals": {"aws:SourceAccount": "SOURCE-AWS-ACCOUNT-ID"}
      }
    }
  ]
}

```

```
}
```

로깅 정책을 적용하려면 `put-bucket-logging`을 사용합니다.

```
aws s3api put-bucket-logging \
  --bucket amzn-s3-demo-bucket \
  --bucket-logging-status file://logging.json
```

`logging.json`의 콘텐츠:

```
{
  "LoggingEnabled": {
    "TargetBucket": "amzn-s3-demo-bucket",
    "TargetPrefix": "Logs/"
  }
}
```

로깅 서비스 위탁자에 `s3:PutObject` 권한을 부여하려면 `put-bucket-policy` 명령이 필요합니다.

자세한 내용은 Amazon S3 사용자 안내서의 [Amazon S3 서버 액세스 로깅](#)을 참조하세요.

예시 2: 단일 사용자에게만 액세스 로깅에 대한 버킷 정책 설정

다음 `put-bucket-logging` 예제에서는 `amzn-s3-demo-bucket`에 대한 로깅 정책을 설정합니다. AWS 사용자 `bob@example.com`은 로그 파일을 완전히 제어할 수 있으며 다른 사용자는 액세스 권한을 갖지 않습니다. 먼저 `put-bucket-acl`을 사용하여 S3 권한을 부여합니다.

```
aws s3api put-bucket-acl \
  --bucket amzn-s3-demo-bucket \
  --grant-write URI=http://acs.amazonaws.com/groups/s3/LogDelivery \
  --grant-read-acp URI=http://acs.amazonaws.com/groups/s3/LogDelivery
```

그런 다음 `put-bucket-logging`을 사용하여 로깅 정책을 적용합니다.

```
aws s3api put-bucket-logging \
  --bucket amzn-s3-demo-bucket \
  --bucket-logging-status file://logging.json
```

`logging.json`의 콘텐츠:

```
{
  "LoggingEnabled": {
    "TargetBucket": "amzn-s3-demo-bucket",
    "TargetPrefix": "amzn-s3-demo-bucket-logs/",
    "TargetGrants": [
      {
        "Grantee": {
          "Type": "AmazonCustomerByEmail",
          "EmailAddress": "bob@example.com"
        },
        "Permission": "FULL_CONTROL"
      }
    ]
  }
}
```

S3의 로그 전달 시스템에 필수 권한(write 및 read-acp 권한)을 부여하려면 `put-bucket-acl` 명령이 필요합니다.

자세한 내용은 Amazon S3 개발자 안내서의 [Amazon S3 서버 액세스 로깅](#)을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [PutBucketLogging](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 PutBucketNotification 사용

다음 코드 예시는 PutBucketNotification의 사용 방법을 보여 줍니다.

CLI

AWS CLI

`amzn-s3-demo-bucket`이라는 버킷에 알림 구성을 적용합니다.

```
aws s3api put-bucket-notification --bucket amzn-s3-demo-bucket --notification-configuration file://notification.json
```

`notification.json` 파일은 모니터링할 SNS 주제와 이벤트 유형을 지정하는 현재 폴더의 JSON 문서입니다.

```
{
  "TopicConfiguration": {
    "Event": "s3:ObjectCreated:*",
    "Topic": "arn:aws:sns:us-west-2:123456789012:s3-notification-topic"
  }
}
```

SNS 주제에 IAM 정책이 연결되어 있어야 Amazon S3이 해당 주제에 게시할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Id": "example-ID",
  "Statement": [
    {
      "Sid": "example-statement-ID",
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": [
        "SNS:Publish"
      ],
      "Resource": "arn:aws:sns:us-west-2:123456789012:amzn-s3-demo-bucket",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:s3:*:*:amzn-s3-demo-bucket"
        }
      }
    }
  ]
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [PutBucketNotification](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 예시는 S3 이벤트 ObjectRemovedDelete에 대한 SNS 주제를 구성하고 지정된 S3 버킷에 대한 알림을 활성화합니다

```
$topic = [Amazon.S3.Model.TopicConfiguration] @{
```

```

    Id = "delete-event"
    Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
    Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -TopicConfiguration
$topic

```

예시 2: 이 예시는 지정된 버킷에 대해 ObjectCreatedAll 알림을 활성화하여 Lambda 함수로 전송합니다.

```

$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
    Id = "ObjectCreated-Lambda"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".pem"}
            )
        }
    }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $lambdaConfig

```

예시 3: 이 예시는 서로 다른 키 접미사를 기반으로 2개의 서로 다른 Lambda 구성을 생성하고 단일 명령으로 둘 모두를 구성합니다.

```

#Lambda Config 1

$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
    Id = "ObjectCreated-dada-ps1"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".ps1"}
            )
        }
    }
}

```

```

    )
  }
}
}

#Lambda Config 2

$secondlambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = [Amazon.S3.EventType]::ObjectCreatedAll
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
  Id = "ObjectCreated-dada-json"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".json"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $firstLambdaConfig,$secondlambdaConfig

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [PutBucketNotification](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 예시는 S3 이벤트 ObjectRemovedDelete에 대한 SNS 주제를 구성하고 지정된 S3 버킷에 대한 알림을 활성화합니다

```

$topic = [Amazon.S3.Model.TopicConfiguration] @{
  Id = "delete-event"
  Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
  Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -TopicConfiguration
$topic

```

예시 2: 이 예시는 지정된 버킷에 대해 ObjectCreatedAll 알림을 활성화하여 Lambda 함수로 전송합니다.


```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
    Id = "ObjectCreated-Lambda"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".pem"}
            )
        }
    }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $lambdaConfig
```

예시 3: 이 예시는 서로 다른 키 접미사를 기반으로 2개의 서로 다른 Lambda 구성을 생성하고 단일 명령으로 둘 모두를 구성합니다.

```
#Lambda Config 1

$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
    Id = "ObjectCreated-dada-ps1"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".ps1"}
            )
        }
    }
}

#Lambda Config 2

$secondLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = [Amazon.S3.EventType]::ObjectCreatedAll
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
    Id = "ObjectCreated-dada-json"
    Filter = @{
```

```

    S3KeyFilter = @{
        FilterRules = @(
            @{Name="Prefix";Value="dada"}
            @{Name="Suffix";Value=".json"}
        )
    }
}
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $firstLambdaConfig,$secondLambdaConfig

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [PutBucketNotification](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutBucketNotificationConfiguration** 사용

다음 코드 예시는 PutBucketNotificationConfiguration의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [S3 이벤트 알림 처리](#)
- [EventBridge에 이벤트 알림 전송](#)

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
```

```
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to enable notifications for an Amazon Simple
/// Storage Service (Amazon S3) bucket.
/// </summary>
public class EnableNotifications
{
    public static async Task Main()
    {
        const string bucketName = "amzn-s3-demo-bucket1";
        const string snsTopic = "arn:aws:sns:us-east-2:0123456789ab:bucket-
notify";
        const string sqsQueue = "arn:aws:sqs:us-
east-2:0123456789ab:Example_Queue";

        IAmazonS3 client = new AmazonS3Client(Amazon.RegionEndpoint.USEast2);
        await EnableNotificationAsync(client, bucketName, snsTopic,
sqsQueue);
    }

    /// <summary>
    /// This method makes the call to the PutBucketNotificationAsync method.
    /// </summary>
    /// <param name="client">An initialized Amazon S3 client used to call
    /// the PutBucketNotificationAsync method.</param>
    /// <param name="bucketName">The name of the bucket for which
    /// notifications will be turned on.</param>
    /// <param name="snsTopic">The ARN for the Amazon Simple Notification
    /// Service (Amazon SNS) topic associated with the S3 bucket.</param>
    /// <param name="sqsQueue">The ARN of the Amazon Simple Queue Service
    /// (Amazon SQS) queue to which notifications will be pushed.</param>
    public static async Task EnableNotificationAsync(
        IAmazonS3 client,
        string bucketName,
        string snsTopic,
        string sqsQueue)
    {
        try
        {
            // The bucket for which we are setting up notifications.
```

```
var request = new PutBucketNotificationRequest()
{
    BucketName = bucketName,
};

// Defines the topic to use when sending a notification.
var topicConfig = new TopicConfiguration()
{
    Events = new List<EventType> { EventType.ObjectCreatedCopy },
    Topic = snsTopic,
};
request.TopicConfigurations = new List<TopicConfiguration>
{
    topicConfig,
};
request.QueueConfigurations = new List<QueueConfiguration>
{
    new QueueConfiguration()
    {
        Events = new List<EventType>
{ EventType.ObjectCreatedPut },
        Queue = sqsQueue,
    },
};

// Now apply the notification settings to the bucket.
PutBucketNotificationResponse response = await
client.PutBucketNotificationAsync(request);
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error: {ex.Message}");
}
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 [PutBucketNotificationConfiguration](#)을 참조하세요.

CLI

AWS CLI

버킷에 지정된 알림을 활성화하려면

다음 `put-bucket-notification-configuration` 예시에서는 `amzn-s3-demo-bucket`이라는 버킷에 알림 구성을 적용합니다. `notification.json` 파일은 모니터링할 SNS 주제와 이벤트 유형을 지정하는 현재 폴더의 JSON 문서입니다.

```
aws s3api put-bucket-notification-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --notification-configuration file://notification.json
```

`notification.json`의 콘텐츠:

```
{  
  "TopicConfigurations": [  
    {  
      "TopicArn": "arn:aws:sns:us-west-2:123456789012:s3-notification-  
topic",  
      "Events": [  
        "s3:ObjectCreated:*"  
      ]  
    }  
  ]  
}
```

SNS 주제에 IAM 정책이 연결되어 있어야 Amazon S3이 해당 주제에 게시할 수 있습니다.

```
{  
  "Version": "2012-10-17",  
  "Id": "example-ID",  
  "Statement": [  
    {  
      "Sid": "example-statement-ID",  
      "Effect": "Allow",  
      "Principal": {  
        "Service": "s3.amazonaws.com"  
      },  
      "Action": [  
        "SNS:Publish"  
      ],  
    }  
  ]  
}
```

```

        "Resource": "arn:aws:sns:us-west-2:123456789012::s3-notification-
topic",
        "Condition": {
            "ArnLike": {
                "aws:SourceArn": "arn:aws:s3:*:*:amzn-s3-demo-bucket"
            }
        }
    }
]
}

```

- API 세부 정보는 AWS CLI 명령 참조의 [PutBucketNotificationConfiguration](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutBucketPolicy** 사용

다음 코드 예시는 PutBucketPolicy의 사용 방법을 보여 줍니다.

C++

SDK for C++

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::S3::putBucketPolicy(const Aws::String &bucketName,
                                const Aws::String &policyBody,
                                const Aws::S3::S3ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    std::shared_ptr<Aws::StringStream> request_body =
        Aws::MakeShared<Aws::StringStream>("");
    *request_body << policyBody;
}

```

```

    Aws::S3::Model::PutBucketPolicyRequest request;
    request.SetBucket(bucketName);
    request.SetBody(request_body);

    Aws::S3::Model::PutBucketPolicyOutcome outcome =
        s3Client.PutBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putBucketPolicy: "
            << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Set the following policy body for the bucket '" <<
            bucketName << "':" << std::endl << std::endl;
        std::cout << policyBody << std::endl;
    }

    return outcome.IsSuccess();
}

//! Build a policy JSON string.
/*!
    \param userArn: Aws user Amazon Resource Name (ARN).
        For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/reference\_identifiers.html#identifiers-arns.
    \param bucketName: Name of a bucket.
    \return String: Policy as JSON string.
*/

Aws::String getPolicyString(const Aws::String &userArn,
                           const Aws::String &bucketName) {
    return
        "{\n"
        "  \"Version\": \"2012-10-17\", \n"
        "  \"Statement\": [\n"
        "    {\n"
        "      \"Sid\": \"1\", \n"
        "      \"Effect\": \"Allow\", \n"
        "      \"Principal\": {\n"
        "        \"AWS\": \""
        + userArn +
        "\"\n"
        "      }, \n"
        "      \"Action\": [ \"s3:getObject\" ], \n"
        "      \"Resource\": [ \"arn:aws:s3:::"

```

```

        + bucketName +
        "/*\" ]\n"
        "      }\n"
        "    ]\n"
        "  }";
    }
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [PutBucketPolicy](#)를 참조하세요.

CLI

AWS CLI

이 예제에서는 모든 사용자가 MySecretFolder에 있는 객체를 제외하고 amzn-s3-demo-bucket에 있는 모든 객체를 가져올 수 있습니다. 또한 1234-5678-9012라는 AWS 계정의 루트 사용자에게 put 및 delete 권한을 부여합니다.

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://policy.json
```

policy.json:

```

{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    },
    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/MySecretFolder/*"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
      },
      "Action": [

```



```

        "s3:DeleteObject",
        "s3:PutObject"
    ],
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
}
]
}

```

- API 세부 정보는 AWS CLI 명령 참조의 [PutBucketPolicy](#)를 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;

import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>

```

```

    * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
    */
public class SetBucketPolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <polFile>

            Where:
                bucketName - The Amazon S3 bucket to set the policy on.
                polFile - A JSON file containing the policy (see the Amazon S3
Readme for an example).\s
            """;

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

        String bucketName = args[0];
        String polFile = args[1];
        String policyText = getBucketPolicyFromFile(polFile);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setPolicy(s3, bucketName, policyText);
        s3.close();
    }

    /**
     * Sets the policy for an Amazon S3 bucket.
     *
     * @param s3 the {@link S3Client} object used to interact with the
Amazon S3 service
     * @param bucketName the name of the Amazon S3 bucket
     * @param policyText the text of the policy to be set on the bucket
     * @throws S3Exception if there is an error setting the bucket policy
     */
    public static void setPolicy(S3Client s3, String bucketName, String
policyText) {

```

```

        System.out.println("Setting policy:");
        System.out.println("----");
        System.out.println(policyText);
        System.out.println("----");
        System.out.format("On Amazon S3 bucket: \"%s\"\\n", bucketName);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                .bucket(bucketName)
                .policy(policyText)
                .build();

            s3.putBucketPolicy(policyReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("Done!");
    }

    /**
     * Retrieves the bucket policy from a specified file.
     *
     * @param policyFile the path to the file containing the bucket policy
     * @return the content of the bucket policy file as a string
     */
    public static String getBucketPolicyFromFile(String policyFile) {
        StringBuilder fileText = new StringBuilder();
        try {
            List<String> lines = Files.readAllLines(Paths.get(policyFile),
StandardCharsets.UTF_8);
            for (String line : lines) {
                fileText.append(line);
            }

        } catch (IOException e) {
            System.out.format("Problem reading file: \"%s\"\\n", policyFile);
            System.out.println(e.getMessage());
        }

        try {

```

```

        final JsonParser parser = new
ObjectMapper().getFactory().createParser(fileText.toString());
        while (parser.nextToken() != null) {
            }

        } catch (IOException jpe) {
            jpe.printStackTrace();
        }
        return fileText.toString();
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [PutBucketPolicy](#)를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

정책을 추가합니다.

```

import {
    PutBucketPolicyCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Grant an IAM role GetObject access to all of the objects
 * in the provided bucket.
 * @param {{ bucketName: string, iamRoleArn: string }}
 */
export const main = async ({ bucketName, iamRoleArn }) => {
    const client = new S3Client({});
    const command = new PutBucketPolicyCommand({

```

```
// This is a resource-based policy. For more information on resource-based
policies,
// see https://docs.aws.amazon.com/IAM/latest/UserGuide/
access_policies.html#policies_resource-based.
Policy: JSON.stringify({
  Version: "2012-10-17",
  Statement: [
    {
      Effect: "Allow",
      Principal: {
        AWS: iamRoleArn,
      },
      Action: "s3:GetObject",
      Resource: `arn:aws:s3:::${bucketName}/*`,
    },
  ],
}),
// Apply the preceding policy to this bucket.
Bucket: bucketName,
});

try {
  await client.send(command);
  console.log(
    `GetObject access to the bucket "${bucketName}" was granted to the provided
IAM role.`
  );
} catch (caught) {
  if (
    caught instanceof S3ServiceException &&
    caught.name === "MalformedPolicy"
  ) {
    console.error(
      `Error from S3 while setting the bucket policy for the bucket
"${bucketName}". The policy was malformed.`
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while setting the bucket policy for the bucket
"${bucketName}". ${caught.name}: ${caught.message}`
    );
  } else {
    throw caught;
  }
}
```

```
}
};
```

- 자세한 정보는 [AWS SDK for JavaScript 개발자 안내서](#)를 참조하세요.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 [PutBucketPolicy](#)를 참조하십시오.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def put_policy(self, policy):
        """
        Apply a security policy to the bucket. Policies control users' ability
        to perform specific actions, such as listing the objects in the bucket.

        :param policy: The policy to apply to the bucket.
        """
        try:
            self.bucket.Policy().put(Policy=json.dumps(policy))
            logger.info("Put policy %s for bucket '%s'.", policy,
            self.bucket.name)
```

```

    except ClientError:
        logger.exception("Couldn't apply policy to bucket '%s'.",
            self.bucket.name)
        raise

```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [PutBucketPolicy](#)를 참조하십시오.

Ruby

SDK for Ruby

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
  # configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Sets a policy on a bucket.
  #
  def policy(policy)
    @bucket_policy.put(policy: policy)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set the policy for #{@bucket_policy.bucket.name}. Here's why:
    #{e.message}"
    false
  end
end

```

- API 세부 정보는 AWS SDK for Ruby API 참조의 [PutBucketPolicy](#)를 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.
    " Example policy JSON string
    " iv_policy = '{"Version":"2012-10-17",          "Statement":
[{"Effect":"Allow","Principal":{"AWS":"arn:aws:iam::123456789012:user/
user"},"Action":["s3:GetObject"],"Resource":["arn:aws:s3:::bucketname/*"]}]}'
    lo_s3->putbucketpolicy(
        iv_bucket = iv_bucket_name
        iv_policy = iv_policy ).
    MESSAGE 'Bucket policy set.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 [PutBucketPolicy](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutBucketReplication** 사용

다음 코드 예시는 PutBucketReplication의 사용 방법을 보여 줍니다.

CLI

AWS CLI

S3 버킷의 복제를 구성하려면

다음 `put-bucket-replication` 예시에서는 지정된 S3 버킷에 복제 구성을 적용합니다.

```
aws s3api put-bucket-replication \  
  --bucket amzn-s3-demo-bucket1 \  
  --replication-configuration file://replication.json
```

`replication.json`의 콘텐츠:

```
{  
  "Role": "arn:aws:iam::123456789012:role/s3-replication-role",  
  "Rules": [  
    {  
      "Status": "Enabled",  
      "Priority": 1,  
      "DeleteMarkerReplication": { "Status": "Disabled" },  
      "Filter" : { "Prefix": ""},  
      "Destination": {  
        "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket2"  
      }  
    }  
  ]  
}
```

대상 버킷에 버전 관리가 활성화되어 있어야 합니다. 지정된 역할에는 대상 버킷에 쓰기 위한 권한이 있어야 하며 Amazon S3가 역할을 맡도록 허용하는 신뢰 관계가 있어야 합니다.

예시 역할 권한 정책:

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "s3:GetReplicationConfiguration",  
        "s3:ListBucket"  
      ],  
      "Resource": "*"   
    }  
  ]  
}
```

```

        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket1"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:GetObjectVersion",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"
        ],
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket1/*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ReplicateTags"
        ],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket2/*"
    }
]
}

```

예시 신뢰 관계 정책:

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}

```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Simple Storage Service 콘솔 사용자 안내서의 [주제 제목](#)을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [PutBucketReplication](#)을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Sets the replication configuration for an Amazon S3 bucket.
 *
 * @param s3Client          the S3Client instance to use for the operation
 * @param sourceBucketName  the name of the source bucket
 * @param destBucketName    the name of the destination bucket
 * @param destinationBucketARN the Amazon Resource Name (ARN) of the
destination bucket
 * @param roleARN           the ARN of the IAM role to use for the
replication configuration
 */
public static void setReplication(S3Client s3Client, String sourceBucketName,
String destBucketName, String destinationBucketARN, String roleARN) {
    try {
        Destination destination = Destination.builder()
            .bucket(destinationBucketARN)
            .storageClass(StorageClass.STANDARD)
            .build();

        // Define a prefix filter for replication.
        ReplicationRuleFilter ruleFilter = ReplicationRuleFilter.builder()
            .prefix("documents/")
            .build();

        // Define delete marker replication setting.
        DeleteMarkerReplication deleteMarkerReplication =
DeleteMarkerReplication.builder()
            .status(DeleteMarkerReplicationStatus.DISABLED)
```

```
        .build();

// Create the replication rule.
ReplicationRule replicationRule = ReplicationRule.builder()
    .priority(1)
    .filter(ruleFilter)
    .status(ReplicationRuleStatus.ENABLED)
    .deleteMarkerReplication(deleteMarkerReplication)
    .destination(destination)
    .build();

List<ReplicationRule> replicationRuleList = new ArrayList<>();
replicationRuleList.add(replicationRule);

// Define the replication configuration with IAM role.
ReplicationConfiguration configuration =
ReplicationConfiguration.builder()
    .role(roleARN)
    .rules(replicationRuleList)
    .build();

// Apply the replication configuration to the source bucket.
PutBucketReplicationRequest replicationRequest =
PutBucketReplicationRequest.builder()
    .bucket(sourceBucketName)
    .replicationConfiguration(configuration)
    .build();

s3Client.putBucketReplication(replicationRequest);
System.out.println("Replication configuration set successfully.");

} catch (IllegalArgumentException e) {
    System.err.println("Configuration error: " + e.getMessage());
} catch (S3Exception e) {
    System.err.println("S3 Exception: " +
e.awsErrorDetails().errorMessage());
    System.err.println("Status Code: " + e.statusCode());
    System.err.println("Error Code: " + e.awsErrorDetails().errorCode());

} catch (SdkException e) {
    System.err.println("SDK Exception: " + e.getMessage());
}
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [PutBucketReplication](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예제 1: 이 예제에서는 'amzn-s3-demo-bucket' 버킷에서 키 이름 접두사 'TaxDocs'로 생성된 모든 새 객체를 'amzn-s3-demo-bucket' 버킷에 복제할 수 있도록 하는 단일 규칙을 사용하여 복제 구성을 설정합니다.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

예제 2: 이 예제에서는 키 이름 접두사 'TaxDocs' 또는 'OtherDocs'로 생성된 모든 새 객체를 'amzn-s3-demo-bucket' 버킷에 복제할 수 있도록 하는 여러 규칙을 사용하여 복제 구성을 설정합니다. 키 접두사는 겹치지 않아야 합니다.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
```

```

$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params

```

예제 3: 이 예제에서는 지정된 버킷의 복제 구성을 업데이트하여 키 이름 접두사가 'TaxDocs'인 객체를 'amzn-s3-demo-bucket' 버킷으로 복제하는 작업을 제어하는 규칙을 비활성화합니다.

```

$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [PutBucketReplication](#)을 참조하세요.

Tools for PowerShell V5

예제 1: 이 예제에서는 'amzn-s3-demo-bucket' 버킷에서 키 이름 접두사 'TaxDocs'로 생성된 모든 새 객체를 'amzn-s3-demo-bucket' 버킷에 복제할 수 있도록 하는 단일 규칙을 사용하여 복제 구성을 설정합니다.

```

$rule1 = New-Object Amazon.S3.Model.ReplicationRule

```

```

$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params

```

예제 2: 이 예제에서는 키 이름 접두사 'TaxDocs' 또는 'OtherDocs'로 생성된 모든 새 객체를 'amzn-s3-demo-bucket' 버킷에 복제할 수 있도록 하는 여러 규칙을 사용하여 복제 구성을 설정합니다. 키 접두사는 겹치지 않아야 합니다.

```

$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params

```

예제 3: 이 예제에서는 지정된 버킷의 복제 구성을 업데이트하여 키 이름 접두사가 'TaxDocs'인 객체를 'amzn-s3-demo-bucket' 버킷으로 복제하는 작업을 제어하는 규칙을 비활성화합니다.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [PutBucketReplication](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 PutBucketRequestPayment 사용

다음 코드 예시는 PutBucketRequestPayment의 사용 방법을 보여 줍니다.

CLI

AWS CLI

예시 1: 버킷의 ``요청자 지불`` 구성을 활성화하려면

다음 put-bucket-request-payment 예시에서는 지정된 버킷의 requester pays를 활성화합니다.

```
aws s3api put-bucket-request-payment \
    --bucket amzn-s3-demo-bucket \
```



```
--request-payment-configuration '{"Payer": "Requester"}'
```

이 명령은 출력을 생성하지 않습니다.

예시 2: 버킷의 `requester pays` 구성 비활성화

다음 `put-bucket-request-payment` 예시에서는 지정된 버킷의 `requester pays`를 비활성화합니다.

```
aws s3api put-bucket-request-payment \  
  --bucket amzn-s3-demo-bucket \  
  --request-payment-configuration '{"Payer": "BucketOwner"}'
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 참조의 [PutBucketRequestPayment](#)를 참조하세요.

PowerShell

Tools for PowerShell V4

예제 1: 이름이 'amzn-s3-demo-bucket'인 버킷의 요청 결제 구성을 업데이트하여 버킷에서 다운로드를 요청하는 사람에게 다운로드 요금이 부과되도록 합니다. 기본적으로 다운로드 비용은 버킷 소유자가 지불합니다. 지불 요청을 기본값으로 다시 설정하려면 `RequestPaymentConfiguration_Payer` 파라미터에 'BucketOwner'를 사용하세요.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -  
RequestPaymentConfiguration_Payer Requester
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [PutBucketRequestPayment](#)를 참조하세요.

Tools for PowerShell V5

예제 1: 이름이 'amzn-s3-demo-bucket'인 버킷의 요청 결제 구성을 업데이트하여 버킷에서 다운로드를 요청하는 사람에게 다운로드 요금이 부과되도록 합니다. 기본적으로 다운로드 비용은 버킷 소유자가 지불합니다. 지불 요청을 기본값으로 다시 설정하려면 `RequestPaymentConfiguration_Payer` 파라미터에 'BucketOwner'를 사용하세요.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -  
RequestPaymentConfiguration_Payer Requester
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [PutBucketRequestPayment](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 PutBucketTagging 사용

다음 코드 예시는 PutBucketTagging의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [객체 스토리지 시작하기](#)

CLI

AWS CLI

다음 명령은 amzn-s3-demo-bucket이라는 버킷에 태그 지정 구성을 적용합니다.

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging file://tagging.json
```

tagging.json 파일은 태그를 지정하는 현재 폴더의 JSON 문서입니다.

```
{
  "TagSet": [
    {
      "Key": "organization",
      "Value": "marketing"
    }
  ]
}
```

또는 명령줄에서 태그 지정 구성을 amzn-s3-demo-bucket에 직접 적용할 수도 있습니다.

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging 'TagSet=[{Key=organization, Value=marketing}]'
```

- API 세부 정보는 AWS CLI 명령 참조의 [PutBucketTagging](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 이름이 **cloudtrail-test-2018**인 버킷에 두 개의 태그를 적용합니다. 하나는 키가 Stage이고 값이 Test인 태그이며 다른 하나는 키가 Environment고 값이 Alpha인 태그입니다. 버킷에 태그가 추가되었는지 확인하려면 **Get-S3BucketTagging -BucketName bucket_name**을 실행합니다. 결과에는 첫 번째 명령에서 버킷에 적용한 태그가 표시되어야 합니다. 단, **Write-S3BucketTagging**은 버킷에 설정된 기존 태그 전체를 덮어씁니다. 개별 태그를 추가하거나 삭제하려면 리소스 그룹 및 태그 지정 API cmdlet, **Add-RGTResourceTag** 및 **Remove-RGTResourceTag**를 실행합니다. 또는 AWS Management Console의 태그 편집기를 사용하여 S3 버킷 태그를 관리합니다.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage"; Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

예시 2: 이 명령은 이름이 **cloudtrail-test-2018**인 버킷을 **Write-S3BucketTagging** cmdlet으로 파이프합니다. 이렇게 하면 Stage:Production 및 Department:Finance 태그가 버킷에 적용됩니다. 단, **Write-S3BucketTagging**은 버킷에 설정된 기존 태그 전체를 덮어씁니다.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging -TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department"; Value="Finance" } )
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [PutBucketTagging](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 이름이 **cloudtrail-test-2018**인 버킷에 두 개의 태그를 적용합니다. 하나는 키가 Stage이고 값이 Test인 태그이며 다른 하나는 키가 Environment고 값이 Alpha인 태그입니다. 버킷에 태그가 추가되었는지 확인하려면 **Get-S3BucketTagging -BucketName bucket_name**을 실행합니다. 결과에는 첫 번째 명령에서 버킷에 적용한 태그가 표시되어야 합니다. 단, **Write-S3BucketTagging**은 버킷에 설정된 기존 태그 전체를 덮어씁니다. 개별 태그를 추가하거나 삭제하려면 리소스 그룹 및 태그 지정 API cmdlet, **Add-RGTResourceTag** 및 **Remove-RGTResourceTag**를 실행합니다. 또는 AWS Management Console의 태그 편집기를 사용하여 S3 버킷 태그를 관리합니다.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";
Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

예시 2: 이 명령은 이름이 **cloudtrail-test-2018**인 버킷을 **Write-S3BucketTagging** cmdlet으로 파이프합니다. 이렇게 하면 Stage:Production 및 Department:Finance 태그가 버킷에 적용됩니다. 단, **Write-S3BucketTagging**은 버킷에 설정된 기존 태그 전체를 덮어씁니다.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging
-TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department";
Value="Finance" } )
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [PutBucketTagging](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutBucketVersioning** 사용

다음 코드 예시는 PutBucketVersioning의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [객체 스토리지 시작하기](#)

CLI

AWS CLI

다음 명령은 amzn-s3-demo-bucket이라는 버킷의 버전 관리를 활성화합니다.

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-
configuration Status=Enabled
```

다음 명령은 버전 관리를 활성화하고 mfa 코드를 사용합니다.

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-configuration Status=Enabled --mfa "SERIAL 123456"
```

- API 세부 정보는 AWS CLI 명령 참조의 [PutBucketVersioning](#)을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Enables bucket versioning for the specified S3 bucket.
 *
 * @param s3Client the S3 client to use for the operation
 * @param bucketName the name of the S3 bucket to enable versioning for
 */
public static void enableBucketVersioning(S3Client s3Client, String
bucketName){
    VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
        .status(BucketVersioningStatus.ENABLED)
        .build();

    PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
        .bucket(bucketName)
        .versioningConfiguration(versioningConfiguration)
        .build();

    s3Client.putBucketVersioning(versioningRequest);
    System.out.println("Bucket versioning has been enabled for "+bucketName);
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [PutBucketVersioning](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷의 버전 관리를 활성화합니다.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -  
VersioningConfig_Status Enabled
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [PutBucketVersioning](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 S3 버킷의 버전 관리를 활성화합니다.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -  
VersioningConfig_Status Enabled
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [PutBucketVersioning](#)을 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

TRY.

```
" Example: Enable versioning on a bucket  
" iv_status = 'Enabled'  
lo_s3->putbucketversioning(  
    iv_bucket = iv_bucket_name  
    io_versioningconfiguration = NEW /aws1/cl_s3_versioningconf(  
        iv_status = iv_status ) ).  
MESSAGE 'Bucket versioning enabled.' TYPE 'I'.
```

```
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 [PutBucketVersioning](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutBucketWebsite** 사용

다음 코드 예시는 PutBucketWebsite의 사용 방법을 보여 줍니다.

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Put the website configuration.
PutBucketWebsiteRequest putRequest = new
PutBucketWebsiteRequest()
{
    BucketName = bucketName,
    WebsiteConfiguration = new WebsiteConfiguration()
    {
        IndexDocumentSuffix = indexDocumentSuffix,
        ErrorDocument = errorDocument,
    },
};
PutBucketWebsiteResponse response = await
client.PutBucketWebsiteAsync(putRequest);
```

- API 세부 정보는 AWS SDK for .NET API 참조의 [PutBucketWebsite](#)를 참조하십시오.

C++

SDK for C++

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::S3::putWebsiteConfig(const Aws::String &bucketName,
                                   const Aws::String &indexPage, const Aws::String
                                   &errorPage,
                                   const Aws::S3::S3ClientConfiguration
                                   &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::IndexDocument indexDocument;
    indexDocument.SetSuffix(indexPage);

    Aws::S3::Model::ErrorDocument errorDocument;
    errorDocument.SetKey(errorPage);

    Aws::S3::Model::WebsiteConfiguration websiteConfiguration;
    websiteConfiguration.SetIndexDocument(indexDocument);
    websiteConfiguration.SetErrorDocument(errorDocument);

    Aws::S3::Model::PutBucketWebsiteRequest request;
    request.SetBucket(bucketName);
    request.SetWebsiteConfiguration(websiteConfiguration);

    Aws::S3::Model::PutBucketWebsiteOutcome outcome =
        client.PutBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutBucketWebsite: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Success: Set website configuration for bucket '"
                  << bucketName << "'." << std::endl;
    }
}
```



```

    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [PutBucketWebsite](#)를 참조하세요.

CLI

AWS CLI

amzn-s3-demo-bucket이라는 버킷에 정적 웹 사이트 구성을 적용합니다.

```
aws s3api put-bucket-website --bucket amzn-s3-demo-bucket --website-configuration file://website.json
```

website.json 파일은 웹 사이트의 색인 및 오류 페이지를 지정하는 현재 폴더의 JSON 문서입니다.

```

{
  "IndexDocument": {
    "Suffix": "index.html"
  },
  "ErrorDocument": {
    "Key": "error.html"
  }
}

```

- API 세부 정보는 AWS CLI 명령 참조의 [PutBucketWebsite](#)를 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.IndexDocument;
import software.amazon.awssdk.services.s3.model.PutBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.WebsiteConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

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

            Usage:    <bucketName> [indexdoc]\s

            Where:
                bucketName    - The Amazon S3 bucket to set the website
configuration on.\s
                indexdoc - The index document, ex. 'index.html'
                        If not specified, 'index.html' will be set.

            """;

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

        String bucketName = args[0];
        String indexDoc = "index.html";
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();
```

```
        setWebsiteConfig(s3, bucketName, indexDoc);
        s3.close();
    }

    /**
     * Sets the website configuration for an Amazon S3 bucket.
     *
     * @param s3 The {@link S3Client} instance to use for the AWS SDK operations.
     * @param bucketName The name of the S3 bucket to configure.
     * @param indexDoc The name of the index document to use for the website
     configuration.
     */
    public static void setWebsiteConfig(S3Client s3, String bucketName, String
indexDoc) {
        try {
            WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()
                .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
                .build();

            PutBucketWebsiteRequest pubWebsiteReq =
PutBucketWebsiteRequest.builder()
                .bucket(bucketName)
                .websiteConfiguration(websiteConfig)
                .build();

            s3.putBucketWebsite(pubWebsiteReq);
            System.out.println("The call was successful");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [PutBucketWebsite](#)를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

웹 사이트 구성을 설정합니다.

```
import {
  PutBucketWebsiteCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Configure an Amazon S3 bucket to serve a static website.
 * Website access must also be granted separately. For more information
 * on setting the permissions for website access, see
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/
 * WebsiteAccessPermissionsReqd.html.
 */
@param {{ bucketName: string }}
*/
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new PutBucketWebsiteCommand({
    Bucket: bucketName,
    WebsiteConfiguration: {
      ErrorDocument: {
        // The object key name to use when a 4XX class error occurs.
        Key: "error.html",
      },
      IndexDocument: {
        // A suffix that is appended to a request when the request is
        // for a directory.
        Suffix: "index.html",
      },
    },
  });
};
```

```
try {
  await client.send(command);
  console.log(
    `The bucket "${bucketName}" has been configured as a static website.`
  );
} catch (caught) {
  if (
    caught instanceof S3ServiceException &&
    caught.name === "NoSuchBucket"
  ) {
    console.error(
      `Error from S3 while configuring the bucket "${bucketName}" as a static
website. The bucket doesn't exist.`
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while configuring the bucket "${bucketName}" as a static
website. ${caught.name}: ${caught.message}`
    );
  } else {
    throw caught;
  }
}
};
```

- 자세한 정보는 [AWS SDK for JavaScript 개발자 안내서](#)를 참조하세요.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 [PutBucketWebsite](#)를 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 인덱스 문서가 'index.html', 오류 문서가 'error.html'인 지정된 버킷에 대한 웹 사이트 호스팅을 활성화합니다.

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
-WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [PutBucketWebsite](#)를 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 인덱스 문서가 'index.html', 오류 문서가 'error.html'인 지정된 버킷에 대한 웹 사이트 호스팅을 활성화합니다.

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
  -WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [PutBucketWebsite](#)를 참조하세요.

Ruby

SDK for Ruby

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket website actions.
class BucketWebsiteWrapper
  attr_reader :bucket_website

  # @param bucket_website [Aws::S3::BucketWebsite] A bucket website object
  configured with an existing bucket.
  def initialize(bucket_website)
    @bucket_website = bucket_website
  end

  # Sets a bucket as a static website.
  #
  # @param index_document [String] The name of the index document for the
  website.
```

```

# @param error_document [String] The name of the error document to show for 4XX
errors.
# @return [Boolean] True when the bucket is configured as a website; otherwise,
false.
def set_website(index_document, error_document)
  @bucket_website.put(
    website_configuration: {
      index_document: { suffix: index_document },
      error_document: { key: error_document }
    }
  )
  true
rescue Aws::Errors::ServiceError => e
  puts "Couldn't configure #{@bucket_website.bucket.name} as a website. Here's
why: #{e.message}"
  false
end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  index_document = "index.html"
  error_document = "404.html"

  wrapper = BucketWebsiteWrapper.new(Aws::S3::BucketWebsite.new(bucket_name))
  return unless wrapper.set_website(index_document, error_document)

  puts "Successfully configured bucket #{bucket_name} as a static website."
end

run_demo if $PROGRAM_NAME == __FILE__

```

- API 세부 정보는 AWS SDK for Ruby API 참조의 [PutBucketWebsite](#)를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutObject** 사용

다음 코드 예시는 PutObject의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [기본 사항 알아보기](#)
- [객체 스토리지 시작하기](#)
- [조건부 요청 수행](#)
- [업로드 및 다운로드 추적](#)
- [Amazon S3 객체 무결성 작업](#)

.NET

SDK for .NET(v4)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Shows how to upload a file from the local computer to an Amazon S3
/// bucket.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to which the object
/// will be uploaded.</param>
/// <param name="objectName">The object to upload.</param>
/// <param name="filePath">The path, including file name, of the object
/// on the local computer to upload.</param>
/// <returns>A boolean value indicating the success or failure of the
/// upload procedure.</returns>
public async Task<bool> UploadFileAsync(
    string bucketName,
    string objectName,
    string filePath)
{
    try
    {
        var request = new PutObjectRequest
        {
```



```

        BucketName = bucketName,
        Key = objectName,
        FilePath = filePath,
    };

    var response = await _amazonS3.PutObjectAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error uploading {objectName}: {ex.Message}");
    return false;
}
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [PutObject](#)를 참조하세요.

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

서버 측 암호화를 사용하여 객체를 업로드합니다.

```

using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to upload an object to an Amazon Simple Storage
/// Service (Amazon S3) bucket with server-side encryption enabled.
/// </summary>
public class ServerSideEncryption
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
    }
}

```

```

        string keyName = "samplefile.txt";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2.
        IAmazonS3 client = new AmazonS3Client();

        await WritingAnObjectAsync(client, bucketName, keyName);
    }

    /// <summary>
    /// Upload a sample object include a setting for encryption.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
    /// to upload a file and apply server-side encryption.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
    /// encrypted object will reside.</param>
    /// <param name="keyName">The name for the object that you want to
    /// create in the supplied bucket.</param>
    public static async Task WritingAnObjectAsync(IAmazonS3 client, string
bucketName, string keyName)
    {
        try
        {
            var putRequest = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
                ContentBody = "sample text",
                ServerSideEncryptionMethod =
ServerSideEncryptionMethod.AES256,
            };

            var putResponse = await client.PutObjectAsync(putRequest);

            // Determine the encryption state of an object.
            GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest
            {
                BucketName = bucketName,
                Key = keyName,
            };

```

```

        GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);
        ServerSideEncryptionMethod objectEncryption =
response.ServerSideEncryptionMethod;

        Console.WriteLine($"Encryption method used: {0}",
objectEncryption.ToString());
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error: '{ex.Message}' when writing an
object");
    }
}
}

```

조건부 요청을 사용하여 객체를 저장합니다.

```

/// <summary>
/// Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite using an IfNoneMatch condition for the object key.
/// </summary>
/// <param name="objectKey">The key of the object to upload.</param>
/// <param name="bucket">The source bucket of the object.</param>
/// <param name="content">The content to upload as a string.</param>
/// <returns>The ETag if the conditional write is successful, empty
otherwise.</returns>
public async Task<string> PutObjectConditional(string objectKey, string
bucket, string content)
{
    try
    {
        var putObjectRequest = new PutObjectRequest
        {
            BucketName = bucket,
            Key = objectKey,
            ContentBody = content,
            IfNoneMatch = "*"
        };

        var putResult = await _amazonS3.PutObjectAsync(putObjectRequest);
    }
}

```

```

        _logger.LogInformation($"Conditional write successful for key
{objectKey} in bucket {bucket}.");
        return putResult.ETag;
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "PreconditionFailed")
        {
            _logger.LogError("Conditional write failed: Precondition
failed");
        }
        else
        {
            _logger.LogError($"Unexpected error: {e.ErrorCode}");
            throw;
        }
        return string.Empty;
    }
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [PutObject](#)를 참조하세요.

Bash

AWS CLIBash 스크립트와 함께 사용

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

```

```
#####
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket to copy the file to.
#     $2 - The path and file name of the local file to copy to the bucket.
#     $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function copy_file_to_bucket() {
    local response bucket_name source_file destination_file_name
    bucket_name=$1
    source_file=$2
    destination_file_name=$3

    response=$(aws s3api put-object \
        --bucket "$bucket_name" \
        --body "$source_file" \
        --key "$destination_file_name")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
        errecho "ERROR: AWS reports put-object operation failed.\n$response"
        return 1
    fi
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [PutObject](#)를 참조하십시오.

C++

SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::S3::putObject(const Aws::String &bucketName,
                           const Aws::String &fileName,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    //We are using the name of the file as the key for the object in the bucket.
    //However, this is just a string and can be set according to your retrieval
    needs.
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                       fileName.c_str(),
                                       std::ios_base::in |
std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << fileName << std::endl;
        return false;
    }

    request.SetBody(inputData);

    Aws::S3::Model::PutObjectOutcome outcome =
        s3Client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putObject: " <<
            outcome.GetError().GetMessage() << std::endl;
    } else {
```

```

        std::cout << "Added object '" << fileName << "' to bucket '"
                    << bucketName << "'.";
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [PutObject](#)를 참조하세요.

CLI

AWS CLI

예제 1: Amazon S3에 객체 업로드

다음 put-object 예제에서는 Amazon S3에 객체를 업로드합니다.

```

aws s3api put-object \
    --bucket amzn-s3-demo-bucket \
    --key my-dir/MySampleImage.png \
    --body MySampleImage.png

```

객체 업로드에 대한 자세한 내용은 Amazon S3 개발자 안내서의 객체 업로드 < <http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html>>를 참조하세요.

예제 2: Amazon S3에 비디오 파일 업로드

다음 put-object 예제 명령은 비디오 파일을 업로드합니다.

```

aws s3api put-object \
    --bucket amzn-s3-demo-bucket \
    --key my-dir/big-video-file.mp4 \
    --body /media/videos/f-sharp-3-data-services.mp4

```

객체 업로드에 대한 자세한 내용은 Amazon S3 개발자 안내서의 객체 업로드 < <http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html>>를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [PutObject](#)를 참조하세요.

Go

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

하위 수준 API를 사용하여 버킷에 객체를 추가합니다.

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "io"
    "log"
    "os"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// UploadFile reads from a file and puts the data into an object in a bucket.
```



```

func (basics BucketBasics) UploadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    file, err := os.Open(fileName)
    if err != nil {
        log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
    } else {
        defer file.Close()
        _, err = basics.S3Client.PutObject(ctx, &s3.PutObjectInput{
            Bucket: aws.String(bucketName),
            Key:     aws.String(objectKey),
            Body:    file,
        })
        if err != nil {
            var apiErr smithy.APIError
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
                log.Printf("Error while uploading object to %s. The object is too large.\n"+
                    "To upload objects larger than 5GB, use the S3 console (160GB max)\n"+
                    "or the multipart upload API (5TB max).", bucketName)
            } else {
                log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
                    fileName, bucketName, objectKey, err)
            }
        } else {
            err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
                ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
                    aws.String(objectKey)}, time.Minute)
            if err != nil {
                log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
            }
        }
    }
    return err
}

```

전송 관리자를 사용하여 버킷에 객체를 업로드합니다.

```

import (
    "bytes"
    "context"
    "errors"

```

```
"fmt"
"log"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
    string, contents string) (string, error) {
    var outKey string
    input := &s3.PutObjectInput{
        Bucket:      aws.String(bucket),
        Key:         aws.String(key),
        Body:        bytes.NewReader([]byte(contents)),
        ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
    }
    output, err := actor.S3Manager.Upload(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    } else {
        err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
            Bucket: aws.String(bucket),
            Key:    aws.String(key),
        }, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
                bucket)
        }
    }
}
```

```

    } else {
        outKey = *output.Key
    }
}
return outKey, err
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [PutObject](#)를 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

[S3Client](#)를 사용하여 버킷에 파일을 업로드합니다.

```

/**
 * Uploads a local file to an AWS S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket to upload the file to
 * @param key         the key (object name) to use for the uploaded file
 * @param objectPath  the local file path of the file to be uploaded
 * @return a {@link CompletableFuture} that completes with the {@link
 * PutObjectResponse} when the upload is successful, or throws a {@link
 * RuntimeException} if the upload fails
 */
public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
bucketName, String key, String objectPath) {
    PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

```

```

        CompletableFuture<PutObjectResponse> response =
getAsyncClient().putObject(objectRequest,
AsyncRequestBody.fromFile(Paths.get(objectPath)));
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to upload file", ex);
            }
        });
    }
}

```

[S3TransferManager](#)를 사용하여 버킷에 [파일을 업로드](#)합니다. [파일 전체](#)를 보고 [테스트](#)합니다.

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

    public String uploadFile(S3TransferManager transferManager, String
bucketName,

                            String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }
}

```

[S3Client](#)를 사용하여 버킷에 객체를 업로드하고 태그를 설정합니다.

```
/**
 * Puts tags on an Amazon S3 object.
 *
 * @param s3 An {@link S3Client} object that represents the Amazon S3 client.
 * @param bucketName The name of the Amazon S3 bucket.
 * @param objectKey The key of the Amazon S3 object.
 * @param objectPath The file path of the object to be uploaded.
 */
public static void putS3ObjectTags(S3Client s3, String bucketName, String
objectKey, String objectPath) {
    try {
        Tag tag1 = Tag.builder()
            .key("Tag 1")
            .value("This is tag 1")
            .build();

        Tag tag2 = Tag.builder()
            .key("Tag 2")
            .value("This is tag 2")
            .build();

        List<Tag> tags = new ArrayList<>();
        tags.add(tag1);
        tags.add(tag2);

        Tagging allTags = Tagging.builder()
            .tagSet(tags)
            .build();

        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .tagging(allTags)
            .build();

        s3.putObject(putOb,
RequestBody.fromBytes(getObjectFile(objectPath)));

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
/**
 * Updates the tags associated with an object in an Amazon S3 bucket.
 *
 * @param s3 an instance of the S3Client class, which is used to interact
with the Amazon S3 service
 * @param bucketName the name of the S3 bucket containing the object
 * @param objectKey the key (or name) of the object in the S3 bucket
 * @throws S3Exception if there is an error updating the object's tags
 */
public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {
    try {
        GetObjectTaggingRequest taggingRequest =
GetObjectTaggingRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
        List<Tag> obTags = getTaggingRes.tagSet();
        for (Tag sinTag : obTags) {
            System.out.println("The tag key is: " + sinTag.key());
            System.out.println("The tag value is: " + sinTag.value());
        }

        // Replace the object's tags with two new tags.
        Tag tag3 = Tag.builder()
            .key("Tag 3")
            .value("This is tag 3")
            .build();

        Tag tag4 = Tag.builder()
            .key("Tag 4")
            .value("This is tag 4")
            .build();

        List<Tag> tags = new ArrayList<>();
        tags.add(tag3);
        tags.add(tag4);

        Tagging updatedTags = Tagging.builder()
            .tagSet(tags)
```

```

        .build();

        PutObjectTaggingRequest taggingRequest1 =
PutObjectTaggingRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .tagging(updatedTags)
        .build();

        s3.putObjectTagging(taggingRequest1);
        GetObjectTaggingResponse getTaggingRes2 =
s3.getObjectTagging(taggingRequest);
        List<Tag> modTags = getTaggingRes2.tagSet();
        for (Tag sinTag : modTags) {
            System.out.println("The tag key is: " + sinTag.key());
            System.out.println("The tag value is: " + sinTag.value());
        }

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

/**
 * Retrieves the contents of a file as a byte array.
 *
 * @param filePath the path of the file to be read
 * @return a byte array containing the contents of the file, or null if an
error occurs
 */
private static byte[] getObjectFile(String filePath) {
    FileInputStream fileInputStream = null;
    byte[] byteArray = null;

    try {
        File file = new File(filePath);
        byteArray = new byte[(int) file.length()];
        fileInputStream = new FileInputStream(file);
        fileInputStream.read(byteArray);

    } catch (IOException e) {
        e.printStackTrace();
    } finally {

```

```

        if (fileInputStream != null) {
            try {
                fileInputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        return byteArray;
    }
}

```

[S3Client](#)를 사용하여 버킷에 객체를 업로드하고 메타데이터를 설정합니다.

```

import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutObjectMetadata {
    public static void main(String[] args) {
        final String USAGE = ""

        Usage:
        <bucketName> <objectKey> <objectPath>\s

```



```

        Where:
            bucketName - The Amazon S3 bucket to upload an object into.
            objectKey - The object to upload (for example, book.pdf).
            objectPath - The path where the file is located (for example, C:/
AWS/book2.pdf).\s
        """;

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

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        System.out.println("Putting object " + objectKey + " into bucket " +
bucketName);
        System.out.println("  in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }

    /**
     * Uploads an object to an Amazon S3 bucket with metadata.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket to upload the object to
     * @param objectKey the name of the object to be uploaded
     * @param objectPath the local file path of the object to be uploaded
     */
    public static void putS3Object(S3Client s3, String bucketName, String
objectKey, String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("author", "Mary Doe");
            metadata.put("version", "1.0.0.0");

            PutObjectRequest putOb = PutObjectRequest.builder()
                .bucket(bucketName)

```

```

        .key(objectKey)
        .metadata(metadata)
        .build();

        s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
        System.out.println("Successfully placed " + objectKey + " into bucket
" + bucketName);

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}

```

[S3Client](#)를 사용하여 버킷에 객체를 업로드하고 객체 보존 값을 설정합니다.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */

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

```

```

Usage:
    <key> <bucketName>\s

Where:
    key - The name of the object (for example, book.pdf).\s
    bucketName - The Amazon S3 bucket name that contains the object
(for example, bucket1).\s
""";

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

String key = args[0];
String bucketName = args[1];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

setRentionPeriod(s3, key, bucketName);
s3.close();
}

/**
 * Sets the retention period for an object in an Amazon S3 bucket.
 *
 * @param s3      the S3Client object used to interact with the Amazon S3
service
 * @param key     the key (name) of the object in the S3 bucket
 * @param bucket the name of the S3 bucket where the object is stored
 *
 * @throws S3Exception if an error occurs while setting the object retention
period
 */
public static void setRentionPeriod(S3Client s3, String key, String bucket) {
    try {
        LocalDate localDate = LocalDate.parse("2020-07-17");
        LocalDateTime localDateTime = localDate.atStartOfDay();
        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

        ObjectLockRetention lockRetention = ObjectLockRetention.builder()

```

```

        .mode("COMPLIANCE")
        .retainUntilDate(instant)
        .build();

        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
        .bucket(bucket)
        .key(key)
        .bypassGovernanceRetention(true)
        .retention(lockRetention)
        .build();

        // To set Retention on an object, the Amazon S3 bucket must support
object
        // locking, otherwise an exception is thrown.
        s3.putObjectRetention(retentionRequest);
        System.out.print("An object retention configuration was successfully
placed on the object");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [PutObject](#)를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

객체를 업로드합니다.

```
import { readFile } from "node:fs/promises";
```

```
import {
  PutObjectCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Upload a file to an S3 bucket.
 * @param {{ bucketName: string, key: string, filePath: string }}
 */
export const main = async ({ bucketName, key, filePath }) => {
  const client = new S3Client({});
  const command = new PutObjectCommand({
    Bucket: bucketName,
    Key: key,
    Body: await readFile(filePath),
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "EntityTooLarge"
    ) {
      console.error(
        `Error from S3 while uploading object to ${bucketName}. \
The object was too large. To upload objects larger than 5GB, use the S3 console \
(160GB max) \
or the multipart upload API (5TB max).`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while uploading object to ${bucketName}. ${caught.name}: \
${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

객체의 ETag가 제공된 값과 일치하는 경우에만 조건부로 객체를 업로드합니다.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfMatch: eTag,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
    methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`,
      );
    } else {

```

```
        throw caught;
      }
    }
  };

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
};

const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- 자세한 정보는 [AWS SDK for JavaScript 개발자 안내서](#)를 참조하십시오.

- API 세부 정보는 AWS SDK for JavaScript API 참조의 [PutObject](#)를 참조하십시오.

Kotlin

SDK for Kotlin

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun putS3Object(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            body = File(objectPath).asByteStream()
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 [PutObject](#)를 참조하십시오.

PHP

SDK for PHP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷에 객체를 업로드합니다.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

$fileName = __DIR__ . "/local-file-" . uniqid();
try {
    $this->s3client->putObject([
        'Bucket' => $this->bucketName,
        'Key' => $fileName,
        'SourceFile' => __DIR__ . '/testfile.txt'
    ]);
    echo "Uploaded $fileName to $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to upload $fileName with error: " . $exception-
    >getMessage();
    exit("Please fix error with file upload before continuing.");
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 [PutObject](#)를 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 단일 파일 "local-sample.txt"를 Amazon S3에 업로드하고, "test-files" 버킷에 "sample.txt" 키가 있는 객체를 생성합니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```

예시 2: 이 명령은 단일 파일 "sample.txt"를 Amazon S3에 업로드하고, "test-files" 버킷에 "sample.txt" 키가 있는 객체를 생성합니다. -Key 파라미터가 제공되지 않은 경우 파일 이름이 S3 객체 키로 사용됩니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

예시 3: 이 명령은 단일 파일 "local-sample.txt"를 Amazon S3에 업로드하고, "test-files" 버킷에 "prefix/to/sample.txt" 키가 있는 객체를 생성합니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -  
File .\local-sample.txt
```

예시 4: 이 명령은 하위 디렉터리 "Scripts"의 모든 파일을 "test-files" 버킷에 업로드하고 공통 키 접두사 "SampleScripts"를 각 객체에 적용합니다. 업로드된 각 파일에는 "SampleScripts/filename"이라는 키가 있으며, 여기서 'filename'은 상황에 따라 다릅니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\
```

예시 5: 이 명령은 로컬 디렉터리 "Scripts"의 모든 *.ps1 파일을 "test-files" 버킷에 업로드하고 공통 키 접두사 "SampleScripts"를 각 객체에 적용합니다. 업로드된 각 파일에는 "SampleScripts/filename.ps1"이라는 키가 있으며, 여기서 'filename'은 상황에 따라 다릅니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\ -SearchPattern *.ps1
```

예시 6: 이 명령은 키가 'sample.txt'인 지정된 콘텐츠 문자열을 포함하는 새 S3 객체를 생성합니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object  
contents"
```

예시 7: 이 명령은 지정된 파일(파일 이름이 키로 사용됨)을 업로드하고 지정된 태그를 새 객체에 적용합니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet  
@{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

예시 8: 이 명령은 지정된 폴더를 재귀적으로 업로드하고 지정된 태그를 모든 새 객체에 적용합니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"
-Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [PutObject](#)를 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 단일 파일 "local-sample.txt"를 Amazon S3에 업로드하고, "test-files" 버킷에 "sample.txt" 키가 있는 객체를 생성합니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-sample.txt
```

예시 2: 이 명령은 단일 파일 "sample.txt"를 Amazon S3에 업로드하고, "test-files" 버킷에 "sample.txt" 키가 있는 객체를 생성합니다. -Key 파라미터가 제공되지 않은 경우 파일 이름이 S3 객체 키로 사용됩니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

예시 3: 이 명령은 단일 파일 "local-sample.txt"를 Amazon S3에 업로드하고, "test-files" 버킷에 "prefix/to/sample.txt" 키가 있는 객체를 생성합니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -
File .\local-sample.txt
```

예시 4: 이 명령은 하위 디렉터리 "Scripts"의 모든 파일을 "test-files" 버킷에 업로드하고 공통 키 접두사 "SampleScripts"를 각 객체에 적용합니다. 업로드된 각 파일에는 "SampleScripts/filename"이라는 키가 있으며, 여기서 'filename'은 상황에 따라 다릅니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
SampleScripts\
```

예시 5: 이 명령은 로컬 디렉터리 "Scripts"의 모든 *.ps1 파일을 "test-files" 버킷에 업로드하고 공통 키 접두사 "SampleScripts"를 각 객체에 적용합니다. 업로드된 각 파일에는 "SampleScripts/filename.ps1"이라는 키가 있으며, 여기서 'filename'은 상황에 따라 다릅니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
SampleScripts\ -SearchPattern *.ps1
```

예시 6: 이 명령은 키가 'sample.txt'인 지정된 콘텐츠 문자열을 포함하는 새 S3 객체를 생성합니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object
contents"
```

예시 7: 이 명령은 지정된 파일(파일 이름이 키로 사용됨)을 업로드하고 지정된 태그를 새 객체에 적용합니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet
@{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

예시 8: 이 명령은 지정된 폴더를 재귀적으로 업로드하고 지정된 태그를 모든 새 객체에 적용합니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"
-Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [PutObject](#)를 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
```

```

        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                                that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def put(self, data):
        """
        Upload data to the object.

        :param data: The data to upload. This can either be bytes or a string.
When this
                                argument is a string, it is interpreted as a file name,
which is
                                opened in read bytes mode.
        """
        put_data = data
        if isinstance(data, str):
            try:
                put_data = open(data, "rb")
            except IOError:
                logger.exception("Expected file name or binary data, got '%s'."
data)
                raise

        try:
            self.object.put(Body=put_data)
            self.object.wait_until_exists()
            logger.info(
                "Put object '%s' to bucket '%s'."
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't put object '%s' to bucket '%s'."
                self.object.key,
                self.object.bucket_name,
            )
            raise
        finally:
            if getattr(put_data, "close", None):

```

```
put_data.close()
```

조건부 요청을 사용하여 객체를 업로드합니다.

```
class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def put_object_conditional(self, object_key: str, source_bucket: str, data:
bytes):
        """
        Uploads an object to Amazon S3 with a conditional request. Prevents
        overwrite
        using an IfNoneMatch condition for the object key.

        :param object_key: The key of the object to upload.
        :param source_bucket: The source bucket of the object.
        :param data: The data to upload.
        """
        try:
            self.s3.put_object(
                Bucket=source_bucket, Key=object_key, Body=data, IfNoneMatch="*"
            )
            print(
                f"\tConditional write successful for key {object_key} in bucket
{source_bucket}."
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
```

```

    if error_code == "PreconditionFailed":
        print("\tConditional write failed: Precondition failed")
    else:
        logger.error(f"Unexpected error: {error_code}")
        raise

```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [PutObject](#)를 참조하세요.

Ruby

SDK for Ruby

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

관리형 업로더(Object.upload_file)를 사용하여 파일을 업로드합니다.

```

require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectUploadFileWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e

```

```

    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
    #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

  wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
    object_key))
  return unless wrapper.upload_file(file_path)

  puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__

```

Object.put을 사용하여 파일을 업로드합니다.

```

require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, 'rb') do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{@object.key}. Here's why:
    #{e.message}"
  end
end

```



```

    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)
  return unless success

  puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__

```

Object.put을 사용하여 파일을 업로드하고 서버 측 암호화를 추가합니다.

```

require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:

```

```
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
    object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
    #{encryption}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 [PutObject](#)를 참조하십시오.

Rust

SDK for Rust

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn upload_object(
  client: &aws_sdk_s3::Client,
  bucket_name: &str,
  file_name: &str,
  key: &str,
) -> Result<aws_sdk_s3::operation::put_object::PutObjectOutput, S3ExampleError> {
  let body =
    aws_sdk_s3::primitives::ByteStream::from_path(std::path::Path::new(file_name)).await;
  client
    .put_object()
    .bucket(bucket_name)
    .key(key)
    .body(body.unwrap())
```

```

        .send()
        .await
        .map_err(S3ExampleError::from)
    }

```

- API 세부 정보는 AWS SDK for Rust API 참조의 [PutObject](#)를 참조하십시오.

SAP ABAP

SDK for SAP ABAP API

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

"Get contents of file from application server."
DATA lv_body TYPE xstring.
OPEN DATASET iv_file_name FOR INPUT IN BINARY MODE.
READ DATASET iv_file_name INTO lv_body.
CLOSE DATASET iv_file_name.

"Upload/put an object to an S3 bucket."
TRY.
    lo_s3->putobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_file_name
        iv_body = lv_body ).
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- API 세부 정보는 AWSSDK for SAP ABAP API의 [PutObject](#)를 참조하십시오.

Swift

SDK for Swift

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSS3
import Smithy

    public func uploadFile(bucket: String, key: String, file: String) async
throws {
    let fileUrl = URL(fileURLWithPath: file)
    do {
        let fileData = try Data(contentsOf: fileUrl)
        let dataStream = ByteStream.data(fileData)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )

        _ = try await client.putObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Putting an object. "))
        throw error
    }
}
```

```
import AWSS3
import Smithy

    public func createFile(bucket: String, key: String, withData data: Data)
async throws {
    let dataStream = ByteStream.data(data)
```

```

    let input = PutObjectInput(
        body: dataStream,
        bucket: bucket,
        key: key
    )

    do {
        _ = try await client.putObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Putting an object."))
        throw error
    }
}

```

- API 세부 정보는 AWS SDK for Swift API 참조의 [PutObject](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutObjectAcl** 사용

다음 코드 예시는 PutObjectAcl의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [액세스 제어 목록\(ACL\) 관리](#)

C++

SDK for C++

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::S3::putObjectAcl(const Aws::String &bucketName, const Aws::String
&objectKey, const Aws::String &ownerID,
                                const Aws::String &granteePermission, const
Aws::String &granteeType,
                                const Aws::String &granteeID, const Aws::String
&granteeEmailAddress,
                                const Aws::String &granteeURI, const
Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutObjectAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);
    request.SetKey(objectKey);
```

```

    Aws::S3::Model::PutObjectAclOutcome outcome =
        s3Client.PutObjectAcl(request);

    if (!outcome.IsSuccess()) {
        auto error = outcome.GetError();
        std::cerr << "Error: putObjectAcl: " << error.GetExceptionName()
            << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the object '" << objectKey
            << "' in the bucket '" << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type
    enumeration.
/*!
    \param access: Human readable string.
    \return Permission: Permission enumeration.
*/
Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type
    enumeration.
/*!
    \param type: Human readable string.
    \return Type: Type enumeration.
*/
Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")

```

```

        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [PutObjectAcl](#)을 참조하세요.

CLI

AWS CLI

다음 명령은 두 명의 AWS 사용자(user1@example.com 및 user2@example.com)에게 full control 권한을 부여하고 모든 사용자에게 read 권한을 부여합니다.

```

aws s3api put-object-acl --bucket amzn-s3-demo-bucket --key file.txt --grant-
full-control emailaddress=user1@example.com,emailaddress=user2@example.com --
grant-read uri=http://acs.amazonaws.com/groups/global/AllUsers

```

사용자 지정 ACL에 대한 자세한 내용은 <http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html>을 참조하세요(put-object-acl과 같은 s3api ACL 명령은 동일한 간편 인수 표기법을 사용함).

- API 세부 정보는 AWS CLI 명령 참조의 [PutObjectAcl](#)을 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

class ObjectWrapper:

```



```

"""Encapsulates S3 object actions."""

def __init__(self, s3_object):
    """
    :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                        that wraps object actions in a class-like structure.
    """
    self.object = s3_object
    self.key = self.object.key

def put_acl(self, email):
    """
    Applies an ACL to the object that grants read access to an AWS user
identified
    by email address.

    :param email: The email address of the user to grant access.
    """
    try:
        acl = self.object.Acl()
        # Putting an ACL overwrites the existing ACL, so append new grants
        # if you want to preserve existing grants.
        grants = acl.grants if acl.grants else []
        grants.append(
            {
                "Grantee": {"Type": "AmazonCustomerByEmail", "EmailAddress":
email},
                "Permission": "READ",
            }
        )
        acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
        logger.info("Granted read access to %s.", email)
    except ClientError:
        logger.exception("Couldn't add ACL to object '%s'.", self.object.key)
        raise

```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 [PutObjectAcl](#)를 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.  
  " Example: Grant read access to an AWS user  
  " iv_grantread = 'emailAddress=user@example.com'  
  lo_s3->putobjectacl(  
    iv_bucket = iv_bucket_name  
    iv_key = iv_object_key  
    iv_grantread = iv_grantread ).  
  MESSAGE 'Object ACL updated.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
  MESSAGE 'Bucket does not exist.' TYPE 'E'.  
CATCH /aws1/cx_s3_nosuchkey.  
  MESSAGE 'Object key does not exist.' TYPE 'E'.  
ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 [PutObjectAcl](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutObjectLegalHold** 사용

다음 코드 예시는 PutObjectLegalHold의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [Amazon S3 객체 잠그기](#)

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Set or modify a legal hold on an object in an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The key of the object.</param>
/// <param name="holdStatus">The On or Off status for the legal hold.</param>
/// <returns>True if successful.</returns>
public async Task<bool> ModifyObjectLegalHold(string bucketName,
    string objectKey, ObjectLockLegalHoldStatus holdStatus)
{
    try
    {
        var request = new PutObjectLegalHoldRequest()
        {
            BucketName = bucketName,
            Key = objectKey,
            LegalHold = new ObjectLockLegalHold()
            {
                Status = holdStatus
            }
        };

        var response = await _amazonS3.PutObjectLegalHoldAsync(request);
        Console.WriteLine($"\\tModified legal hold for {objectKey} in
{bucketName}.");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tError modifying legal hold: '{ex.Message}'");
        return false;
    }
}
```

```
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 [PutObjectLegalHold](#)를 참조하세요.

CLI

AWS CLI

객체에 법적 보존을 적용하는 방법

다음 put-object-legal-hold 예시에서는 doc1.rtf 객체에 법적 보존을 설정합니다.

```
aws s3api put-object-legal-hold \
  --bucket amzn-s3-demo-bucket-with-object-lock \
  --key doc1.rtf \
  --legal-hold Status=ON
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 참조의 [PutObjectLegalHold](#)를 참조하세요.

Go

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"
```

```
"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
    string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
{
    input := &s3.PutObjectLegalHoldInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
        LegalHold: &types.ObjectLockLegalHold{
            Status: legalHoldStatus,
        },
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }

    _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        }
    }

    return err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [PutObjectLegalHold](#)를 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Set or modify a legal hold on an object in an S3 bucket.
public void modifyObjectLegalHold(String bucketName, String objectKey,
boolean legalHoldOn) {
    ObjectLockLegalHold legalHold ;
    if (legalHoldOn) {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.ON)
            .build();
    } else {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.OFF)
            .build();
    }

    PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .legalHold(legalHold)
        .build();

    getClient().putObjectLegalHold(legalHoldRequest) ;
    System.out.println("Modified legal hold for " + objectKey + " in
"+bucketName +".");
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [PutObjectLegalHold](#)를 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import {
  PutObjectLegalHoldCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Apply a legal hold configuration to the specified object.
 * @param {{ bucketName: string, objectKey: string, legalHoldStatus: "ON" |
 * "OFF" }}
 */
export const main = async ({ bucketName, objectKey, legalHoldStatus }) => {
  if (!["OFF", "ON"].includes(legalHoldStatus.toUpperCase())) {
    throw new Error(
      "Invalid parameter. legalHoldStatus must be 'ON' or 'OFF'."
    );
  }

  const client = new S3Client({});
  const command = new PutObjectLegalHoldCommand({
    Bucket: bucketName,
    Key: objectKey,
    LegalHold: {
      // Set the status to 'ON' to place a legal hold on the object.
      // Set the status to 'OFF' to remove the legal hold.
      Status: legalHoldStatus,
    },
  });

  try {
    await client.send(command);
    console.log(
```

```

    `Legal hold status set to "${legalHoldStatus}" for "${objectKey}" in
    "${bucketName}"`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying legal hold status for "${objectKey}" in
        "${bucketName}". The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying legal hold status for "${objectKey}" in
        "${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utills/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    objectKey: {
      type: "string",
      required: true,
    },
    legalHoldStatus: {
      type: "string",
      default: "ON",
    },
  },

```



```

};
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
}

```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 [PutObjectLegalHold](#)를 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

객체에 법적 보존을 적용하세요.

```

def set_legal_hold(s3_client, bucket: str, key: str) -> None:
    """
    Set a legal hold on a specific file in a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket containing the file.
        key: The key of the file to set the legal hold on.
    """
    print()
    logger.info("Setting legal hold on file [%s] in bucket [%s]", key, bucket)
    try:

```

```

        before_status = "OFF"
        after_status = "ON"
        s3_client.put_object_legal_hold(
            Bucket=bucket, Key=key, LegalHold={"Status": after_status}
        )
        logger.debug(
            "Legal hold set successfully on file [%s] in bucket [%s]", key,
bucket
        )
        _print_legal_hold_update(bucket, key, before_status, after_status)
    except Exception as e:
        logger.error(
            "Failed to set legal hold on file [%s] in bucket [%s]: %s", key,
bucket, e
        )

```

- API 세부 정보는 AWS SDK for Python(Boto3) API 참조의 [PutObjectLegalHold](#)를 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

TRY.

```

" Example: Set legal hold status to ON
" iv_status = 'ON'
lo_s3->putobjectlegalhold(
    iv_bucket = iv_bucket_name
    iv_key = iv_object_key
    io_legalhold = NEW /aws1/cl_s3_objlocklegalhold(
        iv_status = iv_status ) ).
MESSAGE 'Object legal hold status set.' TYPE 'I'.

```

```
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 [PutObjectLegalHold](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutObjectLockConfiguration** 사용

다음 코드 예시는 PutObjectLockConfiguration의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [Amazon S3 객체 잠그기](#)

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷의 객체 잠금 구성을 설정합니다.

```
/// <summary>
/// Enable object lock on an existing bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket to modify.</param>
/// <returns>True if successful.</returns>
public async Task<bool> EnableObjectLockOnBucket(string bucketName)
```

```

{
    try
    {
        // First, enable Versioning on the bucket.
        await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
        {
            BucketName = bucketName,
            VersioningConfig = new S3BucketVersioningConfig()
            {
                EnableMfaDelete = false,
                Status = VersionStatus.Enabled
            }
        });

        var request = new PutObjectLockConfigurationRequest()
        {
            BucketName = bucketName,
            ObjectLockConfiguration = new ObjectLockConfiguration()
            {
                ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
            },
        };

        var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
        Console.WriteLine($"{bucketName}\tAdded an object lock policy to bucket
{bucketName}.");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
        return false;
    }
}

```

버킷의 기본 보존 기간을 설정합니다.

```

/// <summary>
/// Set or modify a retention period on an S3 bucket.
/// </summary>

```

```

/// <param name="bucketName">The bucket to modify.</param>
/// <param name="retention">The retention mode.</param>
/// <param name="retainUntilDate">The date for retention until.</param>
/// <returns>True if successful.</returns>
public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
{
    var enabledString = enableObjectLock ? "Enabled" : "Disabled";
    var timeDifference = retainUntilDate.Subtract(DateTime.Now);
    try
    {
        // First, enable Versioning on the bucket.
        await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
        {
            BucketName = bucketName,
            VersioningConfig = new S3BucketVersioningConfig()
            {
                EnableMfaDelete = false,
                Status = VersionStatus.Enabled
            }
        });

        var request = new PutObjectLockConfigurationRequest()
        {
            BucketName = bucketName,
            ObjectLockConfiguration = new ObjectLockConfiguration()
            {
                ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                Rule = new ObjectLockRule()
                {
                    DefaultRetention = new DefaultRetention()
                    {
                        Mode = retention,
                        Days = timeDifference.Days // Can be specified in
days or years but not both.
                    }
                }
            }
        };

        var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);

```

```

        Console.WriteLine($"\\tAdded a default retention to bucket
{bucketName}.");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tError modifying object lock: '{ex.Message}'");
        return false;
    }
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [PutObjectLockConfiguration](#)을 참조하세요.

CLI

AWS CLI

버킷에 객체 잠금 구성을 설정하는 방법

다음 put-object-lock-configuration 예시에서는 지정된 버킷에 50일 객체 잠금을 설정합니다.

```

aws s3api put-object-lock-configuration \
  --bucket amzn-s3-demo-bucket-with-object-lock \
  --object-lock-configuration '{ "ObjectLockEnabled": "Enabled", "Rule":
{ "DefaultRetention": { "Mode": "COMPLIANCE", "Days": 50 } } }'

```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 참조의 [PutObjectLockConfiguration](#)을 참조하세요.

Go

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷의 객체 잠금 구성을 설정합니다.

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
    string) error {
    // Versioning must be enabled on the bucket before object locking is enabled.
    verInput := &s3.PutBucketVersioningInput{
        Bucket: aws.String(bucket),
        VersioningConfiguration: &types.VersioningConfiguration{
            MFADelete: types.MFADeleteDisabled,
            Status:    types.BucketVersioningStatusEnabled,
        },
    }
    _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    }
}
```

```

    return err
}

input := &s3.PutObjectLockConfigurationInput{
    Bucket: aws.String(bucket),
    ObjectLockConfiguration: &types.ObjectLockConfiguration{
        ObjectLockEnabled: types.ObjectLockEnabledEnabled,
    },
}
_, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
if err != nil {
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    }
}

return err
}

```

버킷의 기본 보존 기간을 설정합니다.

```

import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {

```



```

S3Client  *s3.Client
S3Manager *manager.Uploader
}

// ModifyDefaultBucketRetention modifies the default retention period of an
// existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
    ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
    retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

    input := &s3.PutObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
        ObjectLockConfiguration: &types.ObjectLockConfiguration{
            ObjectLockEnabled: lockMode,
            Rule: &types.ObjectLockRule{
                DefaultRetention: &types.DefaultRetention{
                    Days: aws.Int32(retentionPeriod),
                    Mode: retentionMode,
                },
            },
        },
    }

    _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    }

    return err
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [PutObjectLockConfiguration](#)을 참조하세요.

Java

SDK for Java 2.x

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷의 객체 잠금 구성을 설정합니다.

```
// Enable object lock on an existing bucket.
public void enableObjectLockOnBucket(String bucketName) {
    try {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        // Enable versioning on the bucket.
        getClient().putBucketVersioning(putBucketVersioningRequest);
        PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(ObjectLockConfiguration.builder()
                .objectLockEnabled(ObjectLockEnabled.ENABLED)
                .build())
            .build();

        getClient().putObjectLockConfiguration(request);
        System.out.println("Successfully enabled object lock on
"+bucketName);

    } catch (S3Exception ex) {
        System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
    }
}
```

```
}  
}
```

버킷의 기본 보존 기간을 설정합니다.

```
// Set or modify a retention period on an S3 bucket.  
public void modifyBucketDefaultRetention(String bucketName) {  
    VersioningConfiguration versioningConfiguration =  
VersioningConfiguration.builder()  
        .mfaDelete(MFADelete.DISABLED)  
        .status(BucketVersioningStatus.ENABLED)  
        .build();  
  
    PutBucketVersioningRequest versioningRequest =  
PutBucketVersioningRequest.builder()  
        .bucket(bucketName)  
        .versioningConfiguration(versioningConfiguration)  
        .build();  
  
    getClient().putBucketVersioning(versioningRequest);  
    DefaultRetention rention = DefaultRetention.builder()  
        .days(1)  
        .mode(ObjectLockRetentionMode.GOVERNANCE)  
        .build();  
  
    ObjectLockRule lockRule = ObjectLockRule.builder()  
        .defaultRetention(rention)  
        .build();  
  
    ObjectLockConfiguration objectLockConfiguration =  
ObjectLockConfiguration.builder()  
        .objectLockEnabled(ObjectLockEnabled.ENABLED)  
        .rule(lockRule)  
        .build();  
  
    PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =  
PutObjectLockConfigurationRequest.builder()  
        .bucket(bucketName)  
        .objectLockConfiguration(objectLockConfiguration)  
        .build();
```

```
getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
    System.out.println("Added a default retention to bucket "+bucketName
+ ".");
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [PutObjectLockConfiguration](#)을 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷의 객체 잠금 구성을 설정합니다.

```
import {
    PutObjectLockConfigurationCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Enable S3 Object Lock for an Amazon S3 bucket.
 * After you enable Object Lock on a bucket, you can't
 * disable Object Lock or suspend versioning for that bucket.
 * @param {{ bucketName: string, enabled: boolean }}
 */
export const main = async ({ bucketName }) => {
    const client = new S3Client({});
    const command = new PutObjectLockConfigurationCommand({
        Bucket: bucketName,
        // The Object Lock configuration that you want to apply to the specified
        bucket.
        ObjectLockConfiguration: {
            ObjectLockEnabled: "Enabled",
```

```
    },
  });

  try {
    await client.send(command);
    console.log(`Object Lock for "${bucketName}" enabled.`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying the object lock configuration for the
        bucket "${bucketName}". The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying the object lock configuration for the
        bucket "${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
  };
};

const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};
```

```

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}

```

버킷의 기본 보존 기간을 설정합니다.

```

import {
  PutObjectLockConfigurationCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Change the default retention settings for an object in an Amazon S3 bucket.
 * @param {{ bucketName: string, retentionDays: string }}
 */
export const main = async ({ bucketName, retentionDays }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: bucketName,
        // The Object Lock configuration that you want to apply to the specified
        bucket.
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
          Rule: {
            // The default Object Lock retention mode and period that you want to
            apply
            // to new objects placed in the specified bucket. Bucket settings
            require
            // both a mode and a period. The period can be either Days or Years
            but
            // you must select one.

```

```

        DefaultRetention: {
            // In governance mode, users can't overwrite or delete an object
version
            // or alter its lock settings unless they have special permissions.
With
            // governance mode, you protect objects against being deleted by
most users,
            // but you can still grant some users permission to alter the
retention settings
            // or delete the objects if necessary.
            Mode: "GOVERNANCE",
            Days: Number.parseInt(retentionDays),
        },
    },
    },
    })),
);
console.log(
    `Set default retention mode to "GOVERNANCE" with a retention period of
${retentionDays} day(s).`,
);
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while setting the default object retention for a bucket.
The bucket doesn't exist.`,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while setting the default object retention for a bucket.
${caught.name}: ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {

```

```

    isMain,
    validateArgs,
  } from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    retentionDays: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}

```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 [PutObjectLockConfiguration](#)을 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

객체 잠금 구성을 적용하세요.

```
s3_client.put_object_lock_configuration(
    Bucket=bucket,
    ObjectLockConfiguration={"ObjectLockEnabled": "Disabled", "Rule":
    {}},
)
```

- API 세부 정보는 AWS SDK for Python(Boto3) API 참조의 [PutObjectLockConfiguration](#)을 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.
    " Example: Enable object lock with default retention
    " iv_enabled = 'Enabled'
    lo_s3->putobjectlockconfiguration(
        iv_bucket = iv_bucket_name
        io_objectlockconfiguration = NEW /aws1/cl_s3_objectlockconf(
            iv_objectlockenabled = iv_enabled ) ).
    MESSAGE 'Object lock configuration set.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 [PutObjectLockConfiguration](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutObjectRetention** 사용

다음 코드 예시는 PutObjectRetention의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [Amazon S3 객체 잠그기](#)

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Set or modify a retention period on an object in an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The key of the object.</param>
/// <param name="retention">The retention mode.</param>
/// <param name="retainUntilDate">The date retention expires.</param>
/// <returns>True if successful.</returns>
public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
    string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
{
    try
    {
        var request = new PutObjectRetentionRequest()
        {
            BucketName = bucketName,
            Key = objectKey,
```

```

        Retention = new ObjectLockRetention()
        {
            Mode = retention,
            RetainUntilDate = retainUntilDate
        }
    };

    var response = await _amazonS3.PutObjectRetentionAsync(request);
    Console.WriteLine($"\\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"\\tError modifying retention period:
'{ex.Message}'");
    return false;
}
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [PutObjectRetention](#)을 참조하세요.

CLI

AWS CLI

객체의 객체 보존 구성을 설정하는 방법

다음 put-object-retention 예시에서는 지정된 객체에 대한 객체 보존 구성을 2025년 1월 1일 전까지로 설정합니다.

```

aws s3api put-object-retention \
  --bucket amzn-s3-demo-bucket-with-object-lock \
  --key doc1.rtf \
  --retention '{ "Mode": "GOVERNANCE", "RetainUntilDate":
"2025-01-01T00:00:00" }'

```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 참조의 [PutObjectRetention](#)을 참조하세요.

Go

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}  
  
// PutObjectRetention sets the object retention configuration for an S3 object.  
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key  
    string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)  
    error {  
    input := &s3.PutObjectRetentionInput{  
        Bucket: aws.String(bucket),  
        Key:     aws.String(key),  
        Retention: &types.ObjectLockRetention{
```

```

    Mode:          retentionMode,
    RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
  },
  BypassGovernanceRetention: aws.Bool(true),
}

_, err := actor.S3Client.PutObjectRetention(ctx, input)
if err != nil {
    var noKey *types.NoSuchKey
    if errors.As(err, &noKey) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noKey
    }
}

return err
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [PutObjectRetention](#)을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

// Set or modify a retention period on an object in an S3 bucket.
public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
{
    // Calculate the instant one day from now.
    Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

    // Convert the Instant to a ZonedDateTime object with a specific time
    zone.
    ZonedDateTime zonedDateTime =
    futureInstant.atZone(ZoneId.systemDefault());
}

```

```

// Define a formatter for human-readable output.
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

// Format the ZonedDateTime object to a human-readable date string.
String humanReadableDate = formatter.format(zonedDateTime);

// Print the formatted date string.
System.out.println("Formatted Date: " + humanReadableDate);
ObjectLockRetention retention = ObjectLockRetention.builder()
    .mode(ObjectLockRetentionMode.GOVERNANCE)
    .retainUntilDate(futureInstant)
    .build();

PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .retention(retention)
    .build();

getClient().putObjectRetention(retentionRequest);
System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [PutObjectRetention](#)을 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

import {
    PutObjectRetentionCommand,

```

```
S3Client,
S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Place a 24-hour retention period on an object in an Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const command = new PutObjectRetentionCommand({
    Bucket: bucketName,
    Key: key,
    BypassGovernanceRetention: false,
    Retention: {
      // In governance mode, users can't overwrite or delete an object version
      // or alter its lock settings unless they have special permissions. With
      // governance mode, you protect objects against being deleted by most
users,
      // but you can still grant some users permission to alter the retention
settings
      // or delete the objects if necessary.
      Mode: "GOVERNANCE",
      RetainUntilDate: new Date(new Date().getTime() + 24 * 60 * 60 * 1000),
    },
  });

  try {
    await client.send(command);
    console.log("Object Retention settings updated.");
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying the governance mode and retention period
on an object. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying the governance mode and retention period
on an object. ${caught.name}: ${caught.message}`,
      );
    }
  }
}
```

```
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
};

const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 [PutObjectRetention](#)을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷의 'testfile.txt' 객체에 대해 '2019년 12월 31일 00:00:00' 날짜까지 거버넌스 보존 모드를 활성화합니다.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 [PutObjectRetention](#)을 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 지정된 S3 버킷의 'testfile.txt' 객체에 대해 '2019년 12월 31일 00:00:00' 날짜까지 거버넌스 보존 모드를 활성화합니다.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 [PutObjectRetention](#)을 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

객체에 보존을 적용하세요.

```
s3_client.put_object_retention(
    Bucket=bucket,
    Key=key,
    VersionId=version_id,
```

```

        Retention={"Mode": "GOVERNANCE", "RetainUntilDate":
far_future_date},
        BypassGovernanceRetention=True,
    )

```

- API 세부 정보는 AWS SDK for Python(Boto3) API 참조의 [PutObjectRetention](#)을 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

TRY.
    " Example: Set retention mode to GOVERNANCE for 30 days
    " iv_mode = 'GOVERNANCE'
    " iv_retain_date should be a timestamp in the future
    lo_s3->putobjectretention(
        iv_bucket = iv_bucket_name
        iv_key = iv_object_key
        io_retention = NEW /aws1/cl_s3_objectlockret(
            iv_mode = iv_mode
            iv_retainuntildate = iv_retain_date )
        iv_bypassgovernanceretention = abap_true ).
    MESSAGE 'Object retention set.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 [PutObjectRetention](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **RestoreObject** 사용

다음 코드 예시는 RestoreObject의 사용 방법을 보여 줍니다.

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to restore an archived object in an Amazon
/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class RestoreArchivedObject
{
    public static void Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string objectKey = "archived-object.txt";

        // Specify your bucket region (an example region is shown).
        RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

        IAmazonS3 client = new AmazonS3Client(bucketRegion);
        RestoreObjectAsync(client, bucketName, objectKey).Wait();
    }
}
```

```

    /// <summary>
    /// This method restores an archived object from an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// RestoreObjectAsync.</param>
    /// <param name="bucketName">A string representing the name of the
    /// bucket where the object was located before it was archived.</param>
    /// <param name="objectKey">A string representing the name of the
    /// archived object to restore.</param>
    public static async Task RestoreObjectAsync(IAmazonS3 client, string
bucketName, string objectKey)
    {
        try
        {
            var restoreRequest = new RestoreObjectRequest
            {
                BucketName = bucketName,
                Key = objectKey,
                Days = 2,
            };
            RestoreObjectResponse response = await
client.RestoreObjectAsync(restoreRequest);

            // Check the status of the restoration.
            await CheckRestorationStatusAsync(client, bucketName, objectKey);
        }
        catch (AmazonS3Exception amazonS3Exception)
        {
            Console.WriteLine($"Error: {amazonS3Exception.Message}");
        }
    }

    /// <summary>
    /// This method retrieves the status of the object's restoration.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// GetObjectMetadataAsync.</param>
    /// <param name="bucketName">A string representing the name of the Amazon
    /// S3 bucket which contains the archived object.</param>
    /// <param name="objectKey">A string representing the name of the
    /// archived object you want to restore.</param>

```

```
public static async Task CheckRestorationStatusAsync(IAmazonS3 client,
string bucketName, string objectKey)
{
    GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest()
    {
        BucketName = bucketName,
        Key = objectKey,
    };

    GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);

    var restStatus = response.RestoreInProgress ? "in-progress" :
"finished or failed";
    Console.WriteLine($"Restoration status: {restStatus}");
}
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 [RestoreObject](#)를 참조하세요.

CLI

AWS CLI

객체에 대한 복원 요청을 생성하는 방법

다음 restore-object 예시에서는 my-glacier-bucket 버킷의 지정된 Amazon S3 Glacier 객체를 10일 동안 복원합니다.

```
aws s3api restore-object \
  --bucket my-glacier-bucket \
  --key doc1.rtf \
  --restore-request Days=10
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 참조의 [RestoreObject](#)를 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.RestoreRequest;
import software.amazon.awssdk.services.s3.model.GlacierJobParameters;
import software.amazon.awssdk.services.s3.model.RestoreObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tier;

/*
 * For more information about restoring an object, see "Restoring an archived
 * object" at
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html
 *
 * 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 RestoreObject {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
            <bucketName> <keyName> <expectedBucketOwner>

        Where:
            bucketName - The Amazon S3 bucket name.\s
```

```

        keyName - The key name of an object with a Storage class value of
        Glacier.\s
        expectedBucketOwner - The account that owns the bucket (you can
        obtain this value from the AWS Management Console).\s
        """;

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

    String bucketName = args[0];
    String keyName = args[1];
    String expectedBucketOwner = args[2];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    restoreS3Object(s3, bucketName, keyName, expectedBucketOwner);
    s3.close();
}

/**
 * Restores an S3 object from the Glacier storage class.
 *
 * @param s3 an instance of the {@link S3Client} to be used
for interacting with Amazon S3
 * @param bucketName the name of the S3 bucket where the object is
stored
 * @param keyName the key (object name) of the S3 object to be
restored
 * @param expectedBucketOwner the AWS account ID of the expected bucket
owner
 */
public static void restoreS3Object(S3Client s3, String bucketName, String
keyName, String expectedBucketOwner) {
    try {
        RestoreRequest restoreRequest = RestoreRequest.builder()
            .days(10)

            .glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
            .build();
    }
}

```

```

        RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
            .expectedBucketOwner(expectedBucketOwner)
            .bucket(bucketName)
            .key(keyName)
            .restoreRequest(restoreRequest)
            .build();

        s3.restoreObject(objectRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [RestoreObject](#)를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **SelectObjectContent** 사용

다음 코드 예시는 SelectObjectContent의 사용 방법을 보여 줍니다.

CLI

AWS CLI

SQL 문을 기반으로 Amazon S3 객체의 콘텐츠를 필터링하는 방법

다음 select-object-content 예시에서는 지정된 SQL 문으로 객체 my-data-file.csv를 필터링하고 출력을 파일로 보냅니다.

```

aws s3api select-object-content \
    --bucket amzn-s3-demo-bucket \
    --key my-data-file.csv \
    --expression "select * from s3object limit 100" \
    --expression-type 'SQL' \
    --input-serialization '{"CSV": {}, "CompressionType": "NONE"}' \

```



```
--output-serialization '{"CSV": {}}' "output.csv"
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 참조의 [SelectObjectContent](#)를 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

다음 예시에서는 JSON 객체를 사용한 쿼리를 보여줍니다. [전체 예시](#)는 CSV 객체의 사용도 보여줍니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.CSVInput;
import software.amazon.awssdk.services.s3.model.CSVOutput;
import software.amazon.awssdk.services.s3.model.CompressionType;
import software.amazon.awssdk.services.s3.model.ExpressionType;
import software.amazon.awssdk.services.s3.model.FileHeaderInfo;
import software.amazon.awssdk.services.s3.model.InputSerialization;
import software.amazon.awssdk.services.s3.model.JSONInput;
import software.amazon.awssdk.services.s3.model.JSONOutput;
import software.amazon.awssdk.services.s3.model.JSONType;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.OutputSerialization;
import software.amazon.awssdk.services.s3.model.Progress;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.SelectObjectContentRequest;
import
    software.amazon.awssdk.services.s3.model.SelectObjectContentResponseHandler;
```

```
import software.amazon.awssdk.services.s3.model.Stats;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

public class SelectObjectContentExample {
    static final Logger logger =
        LoggerFactory.getLogger(SelectObjectContentExample.class);
    static String BUCKET_NAME = "amzn-s3-demo-bucket"; // Replace with your
        bucket name.
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.csv";
    static String URL_JSON = "https://raw.githubusercontent.com/mledoze/
countries/master/dist/countries.json";

    public static void main(String[] args) {
        SelectObjectContentExample selectObjectContentExample = new
        SelectObjectContentExample();
        try {
            SelectObjectContentExample.setUp();
            selectObjectContentExample.runSelectObjectContentMethodForJSON();
            selectObjectContentExample.runSelectObjectContentMethodForCSV();
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
            System.exit(1);
        } finally {
            SelectObjectContentExample.tearDown();
        }
    }

    EventStreamInfo runSelectObjectContentMethodForJSON() {
        // Set up request parameters.
        final String queryExpression = "select * from s3object[*][*] c where
c.area < 350000";
        final String fileType = FILE_JSON;

        InputSerialization inputSerialization = InputSerialization.builder()
```

```

        .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
        .compressionType(CompressionType.NONE)
        .build();

    OutputSerialization outputSerialization = OutputSerialization.builder()
        .json(JSONOutput.builder().recordDelimiter(null).build())
        .build();

    // Build the SelectObjectContentRequest.
    SelectObjectContentRequest select = SelectObjectContentRequest.builder()
        .bucket(BUCKET_NAME)
        .key(FILE_JSON)
        .expression(queryExpression)
        .expressionType(ExpressionType.SQL)
        .inputSerialization(inputSerialization)
        .outputSerialization(outputSerialization)
        .build();

    EventStreamInfo eventStreamInfo = new EventStreamInfo();
    // Call the selectObjectContent method with the request and a response
    handler.
    // Supply an EventStreamInfo object to the response handler to gather
    records and information from the response.
    s3AsyncClient.selectObjectContent(select,
    buildResponseHandler(eventStreamInfo)).join();

    // Log out information gathered while processing the response stream.
    long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record
->
        record.split("\n").length
    ).sum();
    logger.info("Total records {}: {}", fileType, recordCount);
    logger.info("Visitor onRecords for fileType {} called {} times",
fileType, eventStreamInfo.getCountOnRecordsCalled());
    logger.info("Visitor onStats for fileType {}, {}", fileType,
eventStreamInfo.getStats());
    logger.info("Visitor onContinuations for fileType {}, {}", fileType,
eventStreamInfo.getCountContinuationEvents());
    return eventStreamInfo;
}

    static SelectObjectContentResponseHandler
    buildResponseHandler(EventStreamInfo eventStreamInfo) {

```

```

        // Use a Visitor to process the response stream. This visitor logs
        information and gathers details while processing.
        final SelectObjectContentResponseHandler.Visitor visitor =
        SelectObjectContentResponseHandler.Visitor.builder()
            .onRecords(r -> {
                logger.info("Record event received.");
                eventStreamInfo.addRecord(r.payload().asUtf8String());
                eventStreamInfo.incrementOnRecordsCalled();
            })
            .onCont(ce -> {
                logger.info("Continuation event received.");
                eventStreamInfo.incrementContinuationEvents();
            })
            .onProgress(pe -> {
                Progress progress = pe.details();
                logger.info("Progress event received:\n bytesScanned:
{} \n bytesProcessed: {} \n bytesReturned: {}",
                    progress.bytesScanned(),
                    progress.bytesProcessed(),
                    progress.bytesReturned());
            })
            .onEnd(ee -> logger.info("End event received."))
            .onStats(se -> {
                logger.info("Stats event received.");
                eventStreamInfo.addStats(se.details());
            })
            .build();

        // Build the SelectObjectContentResponseHandler with the visitor that
        processes the stream.
        return SelectObjectContentResponseHandler.builder()
            .subscriber(visitor).build();
    }

    // The EventStreamInfo class is used to store information gathered while
    processing the response stream.
    static class EventStreamInfo {
        private final List<String> records = new ArrayList<>();
        private Integer countOnRecordsCalled = 0;
        private Integer countContinuationEvents = 0;
        private Stats stats;

        void incrementOnRecordsCalled() {
            countOnRecordsCalled++;
        }
    }

```

```
    }

    void incrementContinuationEvents() {
        countContinuationEvents++;
    }

    void addRecord(String record) {
        records.add(record);
    }

    void addStats(Stats stats) {
        this.stats = stats;
    }

    public List<String> getRecords() {
        return records;
    }

    public Integer getCountOnRecordsCalled() {
        return countOnRecordsCalled;
    }

    public Integer getCountContinuationEvents() {
        return countContinuationEvents;
    }

    public Stats getStats() {
        return stats;
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [SelectObjectContent](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **UploadPart** 사용

다음 코드 예시는 UploadPart의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [체크섬 사용](#)
- [Amazon S3 객체 무결성 작업](#)

C++

SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Upload a part to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param partNumber:
    \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
    \param calculatedHash: A data integrity hash to set, depending on the
    checksum algorithm,
                           ignored when it is an empty string.
    \param body: An shared_ptr IOSTream of the data to be uploaded.
    \param client: The S3 client instance used to perform the upload operation.
    \return UploadPartOutcome: The outcome.
*/

Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String
    &bucket,
                                                         const Aws::String &key,
                                                         const Aws::String
    &uploadID,
                                                         int partNumber,
                                                         Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
                                                         const Aws::String
    &calculatedHash,
```

```

std::shared_ptr<Aws::IOStream> &body,
const
const Aws::S3::S3Client
&client) {
    Aws::S3::Model::UploadPartRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetPartNumber(partNumber);
    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
    request.SetBody(body);

    if (!calculatedHash.empty()) {
        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
                request.SetContentMD5(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32:
                request.SetChecksumCRC32(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
                request.SetChecksumCRC32C(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA1:
                request.SetChecksumSHA1(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA256:
                request.SetChecksumSHA256(calculatedHash);
                break;
        }
    }

    return client.UploadPart(request);
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [UploadPart](#)를 참조하세요.

CLI

AWS CLI

다음 명령은 create-multipart-upload 명령으로 시작된 멀티파트 업로드의 첫 번째 부분을 업로드합니다.

```
aws s3api upload-part --bucket amzn-s3-demo-bucket --key
'multipart/01' --part-number 1 --body part01 --upload-id
'dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6YjLx.cph0gtNBtJ8P
```

body 옵션은 업로드할 로컬 파일의 이름 또는 경로를 사용합니다. file:// 접두사는 사용하지 마세요. 최소 파트 크기는 5MB입니다. 업로드 ID는 create-multipart-upload에 의해 반환되며 list-multipart-uploads를 사용하여 가져올 수도 있습니다. 멀티파트 업로드를 생성할 때 버킷과 키가 지정됩니다.

출력:

```
{
  "ETag": "\"e868e0f4719e394144ef36531ee6824c\""
}
```

나중을 위해 각 파트의 ETag 값을 저장하세요. 멀티파트 업로드를 완료하는 데 필요합니다.

- API 세부 정보는 AWS CLI 명령 참조의 [UploadPart](#)를 참조하세요.

Rust

SDK for Rust

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
```



```

        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))
        .build()
        .await
        .unwrap();

    // Chunk index needs to start at 0, but part numbers start at 1.
    let part_number = (chunk_index as i32) + 1;
    let upload_part_res = client
        .upload_part()
        .key(&key)
        .bucket(&bucket_name)
        .upload_id(upload_id)
        .body(stream)
        .part_number(part_number)
        .send()
        .await?;

    upload_parts.push(
        CompletedPart::builder()
            .e_tag(upload_part_res.e_tag.unwrap_or_default())
            .part_number(part_number)
            .build(),
    );
}

```

```

// Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
// upload the file.
let multipart_upload_res: CreateMultipartUploadOutput = client
    .create_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .send()
    .await?;

let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(

```

```
"Missing upload_id after CreateMultipartUpload",
))?;
```

```
// upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
    .set_parts(Some(upload_parts))
    .build();

let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await?;
```

- API에 대한 세부 정보는 AWS SDK for Rust API 참조의 [UploadPart](#)을 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **UploadPartCopy** 사용

다음 코드 예시는 UploadPartCopy의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [멀티파트 복사 수행](#)

CLI

AWS CLI

기존 객체의 데이터를 데이터 소스로 복사하여 객체의 일부를 업로드하는 방법

다음 upload-part-copy 예시에서는 기존 객체의 데이터를 데이터 소스로 복사하여 파트를 업로드합니다.

```
aws s3api upload-part-copy \
  --bucket amzn-s3-demo-bucket \
  --key "Map_Data_June.mp4" \
  --copy-source "amzn-s3-demo-bucket/copy_of_Map_Data_June.mp4" \
  --part-number 1 \
  --upload-id "bq0tdE1CDpWQYRPLHuNG50xAT6pA5D.m_RiBy0gg0H6b13pVRY7QjvLlf75iFdJqp_2wztk5hvpUM2SesXgrz"
```

출력:

```
{
  "CopyPartResult": {
    "LastModified": "2019-12-13T23:16:03.000Z",
    "ETag": "\"711470fc377698c393d94aed6305e245\""
  }
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [UploadPartCopy](#) 섹션을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
    CreateMultipartUploadRequest createMultipartUploadRequest =
    CreateMultipartUploadRequest.builder()
        .bucket(toBucket)
        .key(key)
        .build();
```

```

        getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
            .thenApply(createMultipartUploadResponse -> {
                String uploadId = createMultipartUploadResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);

                UploadPartCopyRequest uploadPartCopyRequest =
                    UploadPartCopyRequest.builder()
                        .sourceBucket(bucketName)
                        .destinationBucket(toBucket)
                        .sourceKey(key)
                        .destinationKey(key)
                        .uploadId(uploadId) // Use the valid uploadId.
                        .partNumber(1) // Ensure the part number is correct.
                        .copySourceRange("bytes=0-1023") // Adjust range as needed
                        .build();

                return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
            })
            .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
            .whenComplete((uploadPartCopyResponse, exception) -> {
                if (exception != null) {
                    // Handle any exceptions.
                    logger.error("Error during upload part copy: " +
exception.getMessage());
                } else {
                    // Successfully completed the upload part copy.
                    System.out.println("Upload Part Copy completed successfully.
ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
                }
            });
        return null;
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [UploadPartCopy](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용한 Amazon S3에 대한 시나리오

다음 코드 예제는 AWS SDK로 Amazon S3에서 일반적인 시나리오를 구현하는 방법을 보여줍니다. 이러한 시나리오에서는 Amazon S3 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여줍니다. 각 시나리오에는 전체 소스 코드에 대한 링크가 포함되어 있습니다. 여기에서 코드를 설정 및 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시나리오는 컨텍스트에 맞는 서비스 작업을 이해하는 데 도움이 되도록 중급 수준의 경험을 대상으로 합니다.

예시

- [버킷이 존재하는지 확인](#)
- [객체가 존재하는지 확인](#)
- [AWS SDK를 사용하여 텍스트를 음성으로, 다시 음성에서 텍스트로 변환](#)
- [AWS SDK를 사용하여 Amazon S3에 대해 미리 서명된 URL 생성](#)
- [사용자가 레이블을 사용하여 사진을 관리할 수 있는 사진 자산 관리 애플리케이션 만들기](#)
- [AWS SDK를 사용하여 Amazon S3 객체를 나열하는 웹 페이지](#)
- [Amazon Textract 탐색기 애플리케이션 생성](#)
- [AWS SDK를 사용하여 주어진 Amazon S3 버킷의 모든 객체 삭제](#)
- [AWS SDK를 사용하여 Amazon S3에 대해 완료되지 않은 멀티파트 업로드 삭제](#)
- [AWS SDK를 사용하여 Amazon Rekognition을 통해 이미지에서 PPE 감지](#)
- [AWS SDK를 사용하여 이미지에서 추출한 텍스트의 개체 삭제](#)
- [AWS SDK를 사용하여 이미지에서 얼굴 감지](#)
- [AWS SDK를 사용하여 Amazon Rekognition을 통해 이미지에서 객체 감지](#)
- [Amazon Rekognition을 AWSSDK와 함께 사용하여 동영상의 사람 및 객체 감지](#)
- [Amazon Simple Storage Service\(Amazon S3\) 버킷에서 S3 '디렉터리' 다운로드](#)
- [Amazon Simple Storage Service\(S3\) 버킷 내의 모든 객체를 로컬 디렉터리로 다운로드](#)
- [AWS SDK를 사용하여 Amazon S3 객체에서 알 수 없는 크기의 스트림 다운로드](#)
- [AWS SDK를 사용하여 다중 리전 액세스 포인트에서 Amazon S3 객체 생성](#)
- [AWS SDK를 사용하고 If-Modified-Since 헤더를 지정하여 Amazon S3 버킷에서 객체 가져오기](#)
- [AWS SDK를 사용하여 Amazon S3 객체 암호화 시작하기](#)
- [AWS SDK를 사용하여 Amazon S3 객체 태깅 시작하기](#)

- [빅 데이터 처리 클러스터 시작하기](#)
- [구성 관리 시작하기](#)
- [문서 텍스트 추출 시작하기](#)
- [기계 학습 특성 저장소 시작하기](#)
- [객체 스토리지 시작하기](#)
- [쿼리 분석 시작하기](#)
- [AWS SDK를 사용하여 Amazon S3 객체 잠금 기능 작업](#)
- [AWS SDK를 사용하여 Amazon S3 조건부 요청 수행](#)
- [AWS SDK를 사용하여 Amazon S3 버킷의 액세스 제어 목록\(ACL\) 관리](#)
- [AWS SDK를 사용하여 Amazon S3로 대규모 Amazon SQS 메시지 관리](#)
- [AWS SDK를 사용하여 Lambda 함수로 버전이 지정된 Amazon S3 객체를 배치 단위로 관리](#)
- [AWS SDK를 사용하여 Amazon S3 URI 구문 분석](#)
- [AWS SDK를 사용하여 Amazon S3 객체 멀티파트 복사 수행](#)
- [AWS SDK를 사용하여 Amazon S3 이벤트 알림 수신 및 처리](#)
- [AWS SDK를 사용하여 EXIF 및 기타 이미지 정보 저장](#)
- [AWS SDK를 사용하여 Amazon EventBridge에 S3 이벤트 알림 전송](#)
- [AWS SDK를 사용하여 Amazon S3 객체 업로드 또는 다운로드 추적](#)
- [S3 객체 Lambda를 사용하여 애플리케이션의 데이터 변환](#)
- [AWS SDK를 사용한 단위 및 통합 테스트의 예시 접근 방식](#)
- [로컬 디렉터리를 Amazon Simple Storage Service\(S3\) 버킷에 반복 업로드](#)
- [AWS SDK를 사용하여 Amazon S3에 대용량 파일 업로드 또는 Amazon S3에서 대용량 파일 다운로드](#)
- [AWS SDK를 사용하여 Amazon S3 객체에 알 수 없는 크기의 스트림 업로드](#)
- [AWS SDK를 사용하여 Amazon S3 객체 작업 수행](#)
- [AWS SDK를 사용하여 Amazon S3 객체 무결성 기능 작업](#)
- [AWS SDK를 사용하여 버전이 지정된 Amazon S3 객체 작업](#)

버킷이 존재하는지 확인

다음 코드 예제에서는 버킷이 존재하는지 확인하는 방법을 보여줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

다음 `doesBucketExists` 메서드로 SDK for Java V1

[AmazonS3Client#doesBucketExistV2\(String\)](#) 메서드를 대체할 수 있습니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.http.HttpStatusCode;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.utils.Validate;

public class DoesBucketExist {
    private static final Logger logger =
        LoggerFactory.getLogger(DoesBucketExist.class);

    public static void main(String[] args) {
        DoesBucketExist doesBucketExist = new DoesBucketExist();

        final S3Client s3SyncClient = S3Client.builder().build();
        final String bucketName = "amzn-s3-demo-bucket"; // Replace with the
        bucket name that you want to check.

        boolean exists = doesBucketExist.doesBucketExist(bucketName,
            s3SyncClient);
        logger.info("Bucket exists: {}", exists);
    }

    /**
     * Checks if the specified bucket exists. Amazon S3 buckets are named in a
     * global namespace; use this method to
     * determine if a specified bucket name already exists, and therefore can't
     * be used to create a new bucket.
     */
}
```

```

    * <p>
    * Internally this method uses the <a
    * href="https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/
services/s3/
S3Client.html#getBucketAcl(java.util.function.Consumer)">S3Client.getBucketAcl(String)</
a>
    * operation to determine whether the bucket exists.
    * <p>
    * This method is equivalent to the AWS SDK for Java V1's <a
    * href="https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/
com/amazonaws/services/s3/AmazonS3Client.html#doesBucketExistV2-
java.lang.String-">AmazonS3Client#doesBucketExistV2(String)</a>.
    *
    * @param bucketName The name of the bucket to check.
    * @param s3SyncClient An <code>S3Client</code> instance. The method checks
for the bucket in the AWS Region
    *
    * configured on the instance.
    * @return The value true if the specified bucket exists in Amazon S3; the
value false if there is no bucket in
    *
    * Amazon S3 with that name.
    */
    public boolean doesBucketExist(String bucketName, S3Client s3SyncClient) {
        try {
            Validate.notEmpty(bucketName, "The bucket name must not be null or an
empty string.", "");
            s3SyncClient.getBucketAcl(r -> r.bucket(bucketName));
            return true;
        } catch (AwsServiceException ase) {
            // A redirect error or an AccessDenied exception means the bucket
exists but it's not in this region
            // or we don't have permissions to it.
            if ((ase.statusCode() == HttpStatus.MOVED_PERMANENTLY) ||
"AccessDenied".equals(ase.awsErrorDetails().errorCode())) {
                return true;
            }
            if (ase.statusCode() == HttpStatus.NOT_FOUND) {
                return false;
            }
            throw ase;
        }
    }
}

```


- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [GetBucketAcl](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

객체가 존재하는지 확인

다음 코드 예제에서는 객체가 존재하는지 확인하는 방법을 보여줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

다음 `doesObjectExist` 메서드로 SDK for Java V1 [AmazonS3Client#doesObjectExist\(String, String\)](#) 메서드를 대체할 수 있습니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.NoSuchKeyException;
import software.amazon.awssdk.utils.Validate;

public class DoesObjectExist {
    private static final Logger logger =
        LoggerFactory.getLogger(DoesObjectExist.class);

    public static void main(String[] args) {
        DoesObjectExist doesObjectExist = new DoesObjectExist();

        final S3Client s3SyncClient = S3Client.builder().build();
        final String bucketName = "amzn-s3-demo-bucket"; // Replace with your
        bucket name.
```

```

        final String key = "my-key"; // Replace with your object key.

        boolean exists = doesObjectExist.doesObjectExist(bucketName, key,
s3SyncClient);
        logger.info("Object exists: {}", exists);
    }

    /**
     * Checks if the specified object exists in the specified bucket.
     * <p>
     * Internally this method uses the
     * <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/
services/s3/
S3Client.html#headObject(java.util.function.Consumer)">S3Client.headObject</a>
     * operation to determine whether the object exists.
     * <p>
     * This method is equivalent to the AWS SDK for Java V1's
     * <a href="https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/
amazonaws/services/s3/AmazonS3Client.html#doesObjectExist-java.lang.String-
java.lang.String-">AmazonS3Client#doesObjectExist(String, String)</a>.
     * <p>
     * <b>Note:</b> This method returns {@code false} only when S3 responds with
404 (NoSuchKey). If the caller
     * does not have {@code s3:ListBucket} permission on the bucket, S3 may
return 403 instead of 404 for
     * non-existent objects, which will be thrown as an exception.
     *
     * @param bucketName The name of the bucket containing the object.
     * @param key The key of the object to check.
     * @param s3SyncClient An {@code S3Client} instance.
     * @return {@code true} if the object exists; {@code false} if it does not
exist.
     */
    public boolean doesObjectExist(String bucketName, String key, S3Client
s3SyncClient) {
        try {
            Validate.notEmpty(bucketName, "The bucket name must not be null or an
empty string.", "");
            Validate.notEmpty(key, "The object key must not be null or an empty
string.", "");
            s3SyncClient.headObject(HeadObjectRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build());
        }
    }

```

```
        return true;
    } catch (NoSuchKeyException e) {
        return false;
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [HeadObject](#)를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 텍스트를 음성으로, 다시 음성에서 텍스트로 변환

다음 코드 예시는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- Amazon Polly를 사용하여 일반 텍스트(UTF-8) 입력 파일을 오디오 파일에 합성합니다.
- Amazon S3 버킷에 오디오 파일을 업로드합니다.
- Amazon Transcribe를 사용하여 오디오 파일을 텍스트로 변환합니다.
- 텍스트를 표시합니다.

Rust

SDK for Rust

Amazon Polly를 사용하여 일반 텍스트(UTF-8) 입력 파일을 오디오 파일에 합성하고, 오디오 파일을 Amazon S3 버킷에 업로드하고, Amazon Transcribe를 사용하여 해당 오디오 파일을 텍스트로 변환하고, 텍스트를 표시합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예시를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Polly
- Amazon S3
- Amazon Transcribe

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon S3에 대해 미리 서명된 URL 생성

다음 코드 예제는 Amazon S3에 대해 미리 서명된 URL을 생성하고 객체를 업로드하는 방법을 보여줍니다.

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

제한된 시간 동안 Amazon S3 작업을 수행할 수 있는 미리 서명된 URL을 생성합니다.

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

public class GenPresignedUrl
{
    public static void Main()
    {
        const string bucketName = "amzn-s3-demo-bucket";
        const string objectKey = "sample.txt";

        // Specify how long the presigned URL lasts, in hours
        const double timeoutDuration = 12;

        // Specify the AWS Region of your Amazon S3 bucket. If it is
        // different from the Region defined for the default user,
        // pass the Region to the constructor for the client. For
        // example: new AmazonS3Client(RegionEndpoint.USEast1);
```

```

        // If using the Region us-east-1, and server-side encryption with AWS
        KMS, you must specify Signature Version 4.
        // Region us-east-1 defaults to Signature Version 2 unless explicitly
        set to Version 4 as shown below.
        // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/
        // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/
        TAWSConfigsS3.html
        AWSConfigsS3.UseSignatureVersion4 = true;
        IAmazonS3 s3Client = new AmazonS3Client(RegionEndpoint.USEast1);

        string urlString = GeneratePresignedURL(s3Client, bucketName,
objectKey, timeoutDuration);
        Console.WriteLine($"The generated URL is: {urlString}.");
    }

    /// <summary>
    /// Generate a presigned URL that can be used to access the file named
    /// in the objectKey parameter for the amount of time specified in the
    /// duration parameter.
    /// </summary>
    /// <param name="client">An initialized S3 client object used to call
    /// the GetPresignedUrl method.</param>
    /// <param name="bucketName">The name of the S3 bucket containing the
    /// object for which to create the presigned URL.</param>
    /// <param name="objectKey">The name of the object to access with the
    /// presigned URL.</param>
    /// <param name="duration">The length of time for which the presigned
    /// URL will be valid.</param>
    /// <returns>A string representing the generated presigned URL.</returns>
    public static string GeneratePresignedURL(IAmazonS3 client, string
bucketName, string objectKey, double duration)
    {
        string urlString = string.Empty;
        try
        {
            var request = new GetPreSignedUrlRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Expires = DateTime.UtcNow.AddHours(duration),
            };
            urlString = client.GetPreSignedURL(request);
        }
    }

```

```
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error: '{ex.Message}'");
        }

        return urlString;
    }
}
```

미리 서명된 URL을 생성하고 해당 URL을 사용하여 업로드를 수행합니다.

```
using System;
using System.IO;
using System.Net.Http;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to upload an object to an Amazon Simple Storage
/// Service (Amazon S3) bucket using a presigned URL. The code first
/// creates a presigned URL and then uses it to upload an object to an
/// Amazon S3 bucket using that URL.
/// </summary>
public class UploadUsingPresignedURL
{
    private static HttpClient httpClient = new HttpClient();

    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "samplefile.txt";
        string filePath = $"source\\{keyName}";

        // Specify how long the signed URL will be valid in hours.
        double timeoutDuration = 12;

        // Specify the AWS Region of your Amazon S3 bucket. If it is
        // different from the Region defined for the default user,
        // pass the Region to the constructor for the client. For
```

```

        // example: new AmazonS3Client(RegionEndpoint.USEast1);

        // If using the Region us-east-1, and server-side encryption with AWS
        KMS, you must specify Signature Version 4.
        // Region us-east-1 defaults to Signature Version 2 unless explicitly
        set to Version 4 as shown below.
        // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingAWSSDK.html#specify-signature-version
        // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/TAWSConfigsS3.html
        AWSConfigsS3.UseSignatureVersion4 = true;
        IAmazonS3 client = new AmazonS3Client(RegionEndpoint.USEast1);

        var url = GeneratePreSignedURL(client, bucketName, keyName,
        timeoutDuration);
        var success = await UploadObject(filePath, url);

        if (success)
        {
            Console.WriteLine("Upload succeeded.");
        }
        else
        {
            Console.WriteLine("Upload failed.");
        }
    }

    /// <summary>
    /// Uploads an object to an Amazon S3 bucket using the presigned URL
    passed in
    /// the url parameter.
    /// </summary>
    /// <param name="filePath">The path (including file name) to the local
    /// file you want to upload.</param>
    /// <param name="url">The presigned URL that will be used to upload the
    /// file to the Amazon S3 bucket.</param>
    /// <returns>A Boolean value indicating the success or failure of the
    /// operation, based on the HttpResponseMessage.</returns>
    public static async Task<bool> UploadObject(string filePath, string url)
    {
        using var streamContent = new StreamContent(
            new FileStream(filePath, FileMode.Open, FileAccess.Read));

        var response = await httpClient.PutAsync(url, streamContent);
    }

```

```

        return response.IsSuccessStatusCode;
    }

    /// <summary>
    /// Generates a presigned URL which will be used to upload an object to
    /// an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// GetPreSignedURL.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket to which
the
    /// presigned URL will point.</param>
    /// <param name="objectKey">The name of the file that will be uploaded.</
param>
    /// <param name="duration">How long (in hours) the presigned URL will
    /// be valid.</param>
    /// <returns>The generated URL.</returns>
    public static string GeneratePreSignedURL(
        IAmazonS3 client,
        string bucketName,
        string objectKey,
        double duration)
    {
        var request = new GetPreSignedUrlRequest
        {
            BucketName = bucketName,
            Key = objectKey,
            Verb = HttpVerb.PUT,
            Expires = DateTime.UtcNow.AddHours(duration),
        };

        string url = client.GetPreSignedURL(request);
        return url;
    }
}

```


SDK for .NET(v4)

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

브라우저에서 직접 업로드할 수 있도록 미리 서명된 POST URL을 생성하고 사용합니다.

```
/// <summary>
/// Scenario demonstrating the complete workflow for presigned POST URLs:
/// 1. Create an S3 bucket
/// 2. Create a presigned POST URL
/// 3. Upload a file using the presigned POST URL
/// 4. Clean up resources
/// </summary>
public class CreatePresignedPostBasics
{
    public static ILogger<CreatePresignedPostBasics> _logger = null!;
    public static S3Wrapper _s3Wrapper = null!;
    public static UiMethods _uiMethods = null!;
    public static IHttpClientFactory _httpClientFactory = null!;
    public static bool _isInteractive = true;
    public static string? _bucketName;
    public static string? _objectKey;

    /// <summary>
    /// Set up the services and logging.
    /// </summary>
    /// <param name="host">The IHost instance.</param>
    public static void SetUpServices(IHost host)
    {
        var loggerFactory = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        });
        _logger = new Logger<CreatePresignedPostBasics>(loggerFactory);

        _s3Wrapper = host.Services.GetRequiredService<S3Wrapper>();
        _httpClientFactory =
            host.Services.GetRequiredService<IHttpClientFactory>();
        _uiMethods = new UiMethods();
    }
}
```

```
}

/// <summary>
/// Perform the actions defined for the Amazon S3 Presigned POST scenario.
/// </summary>
/// <param name="args">Command line arguments.</param>
/// <returns>A Task object.</returns>
public static async Task Main(string[] args)
{
    _isInteractive = !args.Contains("--non-interactive");

    // Set up dependency injection for Amazon S3
    using var host =
        Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonS3>()
                    .AddTransient<S3Wrapper>()
                    .AddHttpClient()
            )
            .Build();

    SetupServices(host);

    try
    {
        // Display overview
        _uiMethods.DisplayOverview();
        _uiMethods.PressEnter(_isInteractive);

        // Step 1: Create bucket
        await CreateBucketAsync();
        _uiMethods.PressEnter(_isInteractive);

        // Step 2: Create presigned URL
        _uiMethods.DisplayTitle("Step 2: Create presigned POST URL");
        var response = await CreatePresignedPostAsync();
        _uiMethods.PressEnter(_isInteractive);

        // Step 3: Display URL and fields
        _uiMethods.DisplayTitle("Step 3: Presigned POST URL details");
        DisplayPresignedPostFields(response);
        _uiMethods.PressEnter(_isInteractive);

        // Step 4: Upload file
```

```

        _uiMethods.DisplayTitle("Step 4: Upload test file using presigned
POST URL");
        await UploadFileAsync(response);
        _uiMethods.PressEnter(_isInteractive);

        // Step 5: Verify file exists
        await VerifyFileExistsAsync();
        _uiMethods.PressEnter(_isInteractive);

        // Step 6: Cleanup
        _uiMethods.DisplayTitle("Step 6: Clean up resources");
        await CleanupAsync();

        _uiMethods.DisplayTitle("S3 Presigned POST Scenario completed
successfully!");
        _uiMethods.PressEnter(_isInteractive);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "Error in scenario");
        Console.WriteLine($"Error: {ex.Message}");

        // Attempt cleanup if there was an error
        if (!string.IsNullOrEmpty(_bucketName))
        {
            _uiMethods.DisplayTitle("Cleaning up resources after error");
            await _s3Wrapper.DeleteBucketAsync(_bucketName);
            Console.WriteLine($"Cleaned up bucket: {_bucketName}");
        }
    }
}

/// <summary>
/// Create an S3 bucket for the scenario.
/// </summary>
private static async Task CreateBucketAsync()
{
    _uiMethods.DisplayTitle("Step 1: Create an S3 bucket");

    // Generate a default bucket name for the scenario
    var defaultBucketName = $"presigned-post-demo-
{DateTime.Now:yyyyMMddHHmmss}".ToLower();

    // Prompt user for bucket name or use default in non-interactive mode

```

```
        _bucketName = _uiMethods.GetUserInput(
            $"Enter S3 bucket name (or press Enter for '{defaultBucketName}')": "",
            defaultBucketName,
            _isInteractive);

    // Basic validation to ensure bucket name is not empty
    if (string.IsNullOrEmpty(_bucketName))
    {
        _bucketName = defaultBucketName;
    }

    Console.WriteLine($"Creating bucket: {_bucketName}");

    await _s3Wrapper.CreateBucketAsync(_bucketName);

    Console.WriteLine($"Successfully created bucket: {_bucketName}");
}

/// <summary>
/// Create a presigned POST URL.
/// </summary>
private static async Task<CreatePresignedPostResponse>
CreatePresignedPostAsync()
{
    _objectKey = "example-upload.txt";
    var expiration = DateTime.UtcNow.AddMinutes(10); // Short expiration for
the demo

    Console.WriteLine($"Creating presigned POST URL for {_bucketName}/
{_objectKey}");
    Console.WriteLine($"Expiration: {expiration} UTC");

    var s3Client = _s3Wrapper.GetS3Client();

    var response = await _s3Wrapper.CreatePresignedPostAsync(
        s3Client, _bucketName!, _objectKey, expiration);

    Console.WriteLine("Successfully created presigned POST URL");
    return response;
}

/// <summary>
/// Upload a file using the presigned POST URL.
```

```
/// </summary>
private static async Task UploadFileAsync(CreatePresignedPostResponse
response)
{

    // Create a temporary test file to upload
    string testFilePath = Path.GetTempFileName();
    string testContent = "This is a test file for the S3 presigned POST
scenario.";

    await File.WriteAllTextAsync(testFilePath, testContent);
    Console.WriteLine($"Created test file at: {testFilePath}");

    // Upload the file using the presigned POST URL
    Console.WriteLine("\nUploading file using the presigned POST URL...");
    var uploadResult = await UploadFileWithPresignedPostAsync(response,
testFilePath);

    // Display the upload result
    if (uploadResult.Success)
    {
        Console.WriteLine($"Upload successful! Status code:
{uploadResult.StatusCode}");
    }
    else
    {
        Console.WriteLine($"Upload failed with status code:
{uploadResult.StatusCode}");
        Console.WriteLine($"Error: {uploadResult.Response}");
        throw new Exception("File upload failed");
    }

    // Clean up the temporary file
    File.Delete(testFilePath);
    Console.WriteLine("Temporary file deleted");
}

/// <summary>
/// Helper method to upload a file using a presigned POST URL.
/// </summary>
private static async Task<(bool Success, HttpStatusCode StatusCode, string
Response)> UploadFileWithPresignedPostAsync(
    CreatePresignedPostResponse response,
    string filePath)
```

```
{
    try
    {
        _logger.LogInformation("Uploading file {filePath} using presigned
POST URL", filePath);

        using var httpClient = _httpClientFactory.CreateClient();
        using var formContent = new MultipartFormDataContent();

        // Add all the fields from the presigned POST response
        foreach (var field in response.Fields)
        {
            formContent.Add(new StringContent(field.Value), field.Key);
        }

        // Add the file content
        var fileStream = File.OpenRead(filePath);
        var fileName = Path.GetFileName(filePath);
        var fileContent = new StreamContent(fileStream);
        fileContent.Headers.ContentType = new MediaTypeHeaderValue("text/
plain");
        formContent.Add(fileContent, "file", fileName);

        // Send the POST request
        var httpResponse = await httpClient.PostAsync(response.Url,
formContent);
        var responseContent = await httpResponse.Content.ReadAsStringAsync();

        // Log and return the result
        _logger.LogInformation("Upload completed with status code
{statusCode}", httpResponse.StatusCode);

        return (httpResponse.IsSuccessStatusCode, httpResponse.StatusCode,
responseContent);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "Error uploading file");
        return (false, HttpStatusCode.InternalServerError, ex.Message);
    }
}

/// <summary>
/// Verify that the uploaded file exists in the S3 bucket.
```

```

    /// </summary>
    private static async Task VerifyFileExistsAsync()
    {
        _uiMethods.DisplayTitle("Step 5: Verify uploaded file exists");

        Console.WriteLine($"Checking if file exists at {_bucketName}/
{_objectKey}...");

        try
        {
            var metadata = await _s3Wrapper.GetObjectMetadataAsync(_bucketName!,
            _objectKey!);

            Console.WriteLine($"File verification successful! File exists in the
            bucket.");
            Console.WriteLine($"File size: {metadata.ContentLength} bytes");
            Console.WriteLine($"File type: {metadata.Headers.ContentType}");
            Console.WriteLine($"Last modified: {metadata.LastModified}");
        }
        catch (AmazonS3Exception ex) when (ex.StatusCode ==
        System.Net.HttpStatusCode.NotFound)
        {
            Console.WriteLine($"Error: File was not found in the bucket.");
            throw;
        }
    }

    private static void DisplayPresignedPostFields(CreatePresignedPostResponse
    response)
    {
        Console.WriteLine($"Presigned POST URL: {response.Url}");
        Console.WriteLine("Form fields to include:");

        foreach (var field in response.Fields)
        {
            Console.WriteLine($"  {field.Key}: {field.Value}");
        }
    }

    /// <summary>
    /// Clean up resources created by the scenario.
    /// </summary>
    private static async Task CleanupAsync()
    {

```

```

        if (!string.IsNullOrEmpty(_bucketName))
        {
            Console.WriteLine($"Deleting bucket {_bucketName} and its
contents...");
            bool result = await _s3Wrapper.DeleteBucketAsync(_bucketName);

            if (result)
            {
                Console.WriteLine("Bucket deleted successfully");
            }
            else
            {
                Console.WriteLine("Failed to delete bucket - it may have been
already deleted");
            }
        }
    }
}

```

C++

SDK for C++

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

미리 서명된 URL을 생성하여 객체를 다운로드합니다.

```

//! Routine which demonstrates creating a pre-signed URL to download an object
from an
//! Amazon Simple Storage Service (Amazon S3) bucket.
//!
\param bucketName: Name of the bucket.
\param key: Name of an object key.
\param expirationSeconds: Expiration in seconds for pre-signed URL.
\param clientConfig: Aws client configuration.
\return Aws::String: A pre-signed URL.
*/

```



```

Aws::String AwsDoc::S3::generatePreSignedGetObjectUrl(const Aws::String
    &bucketName,
                                                    const Aws::String &key,
                                                    uint64_t expirationSeconds,
                                                    const
    Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
    Aws::Http::HttpMethod::HTTP_GET,
                                                    expirationSeconds);
}

```

libcurl을 사용하여 다운로드합니다.

```

static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
    *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test getObject with a pre-signed URL.
/*!
    \param presignedURL: A pre-signed URL to get an object from a bucket.
    \param resultString: A string to hold the result.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::getObjectWithPresignedObjectUrl(const Aws::String &presignedURL,
                                                    Aws::String &resultString) {

    CURL *curl = curl_easy_init();
    CURLcode result;

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }
}

```

```

    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }

    resultString = outWriteString.str();

    if (resultString.find("<?xml") == 0) {
        std::cerr << "Failed to get object, response:\n" << resultString <<
std::endl;
        return false;
    }

    return true;
}

```

미리 서명된 URL을 생성하여 객체를 업로드합니다.

```

//! Routine which demonstrates creating a pre-signed URL to upload an object to
an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
    \param bucketName: Name of the bucket.
    \param key: Name of an object key.

```

```

    \param clientConfig: Aws client configuration.
    \return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::generatePreSignedPutObjectUrl(const Aws::String
    &bucketName,
                                                    const Aws::String &key,
                                                    uint64_t expirationSeconds,
                                                    const
    Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
    Aws::Http::HttpMethod::HTTP_PUT,
                                                    expirationSeconds);
}

```

libcurl을 사용하여 업로드합니다.

```

static size_t myCurlReadBack(char *buffer, size_t size, size_t nitems, void
    *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    str->read(buffer, size * nitems);

    return str->gcount();
}

static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
    *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test putObject with a pre-signed URL.
/*!
    \param presignedURL: A pre-signed URL to put an object in a bucket.
    \param data: Body of the putObject request.
    \return bool: Function succeeded.
*/

```

```
bool AwsDoc::S3::PutStringWithPresignedObjectURL(const Aws::String &presignedURL,
                                                  const Aws::String &data) {

    CURL *curl = curl_easy_init();
    CURLcode result;

    Aws::StringStream readStringStream;
    readStringStream << data;
    result = curl_easy_setopt(curl, CURLOPT_READFUNCTION, myCurlReadBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_READDATA, &readStringStream);
    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READDATA" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_INFILESIZE_LARGE,
                              (curl_off_t) data.size());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_INFILESIZE_LARGE" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }
}
```

```

    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_UPLOAD, 1L);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_PUT" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }

    std::string outString = outWriteString.str();
    if (outString.empty()) {
        std::cout << "Successfully put object." << std::endl;
        return true;
    } else {
        std::cout << "A server error was encountered, output:\n" << outString
                    << std::endl;
        return false;
    }
}

```

Go

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

S3 미리 서명 작업을 래핑하는 함수를 만듭니다.

```
import (
    "context"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    v4 "github.com/aws/aws-sdk-go-v2/aws/signer/v4"
    "github.com/aws/aws-sdk-go-v2/service/s3"
)

// Presigner encapsulates the Amazon Simple Storage Service (Amazon S3) presign
// actions
// used in the examples.
// It contains PresignClient, a client that is used to presign requests to Amazon
// S3.
// Presigned requests contain temporary credentials and can be made from any HTTP
// client.
type Presigner struct {
    PresignClient *s3.PresignClient
}

// GetObject makes a presigned request that can be used to get an object from a
// bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) GetObject(
    ctx context.Context, bucketName string, objectKey string, lifetimeSecs int64)
(*v4.PresignedHTTPRequest, error) {
    request, err := presigner.PresignClient.PresignGetObject(ctx,
        &s3.GetObjectInput{
            Bucket: aws.String(bucketName),
            Key:   aws.String(objectKey),
        }, func(opts *s3.PresignOptions) {
            opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
        })
    if err != nil {
        log.Printf("Couldn't get a presigned request to get %v:%v. Here's why: %v\n",
            bucketName, objectKey, err)
    }
    return request, err
}
```

```
}

// PutObject makes a presigned request that can be used to put an object in a
// bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) PutObject(
    ctx context.Context, bucketName string, objectKey string, lifetimeSecs int64)
(*v4.PresignedHTTPRequest, error) {
    request, err := presigner.PresignClient.PresignPutObject(ctx,
        &s3.PutObjectInput{
            Bucket: aws.String(bucketName),
            Key:    aws.String(objectKey),
        }, func(opts *s3.PresignOptions) {
            opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
        })
    if err != nil {
        log.Printf("Couldn't get a presigned request to put %v:%v. Here's why: %v\n",
            bucketName, objectKey, err)
    }
    return request, err
}

// DeleteObject makes a presigned request that can be used to delete an object
// from a bucket.
func (presigner Presigner) DeleteObject(ctx context.Context, bucketName string,
    objectKey string) (*v4.PresignedHTTPRequest, error) {
    request, err := presigner.PresignClient.PresignDeleteObject(ctx,
        &s3.DeleteObjectInput{
            Bucket: aws.String(bucketName),
            Key:    aws.String(objectKey),
        })
    if err != nil {
        log.Printf("Couldn't get a presigned request to delete object %v. Here's why:
        %v\n", objectKey, err)
    }
    return request, err
}
```

```
func (presigner Presigner) PresignPostObject(ctx context.Context, bucketName
    string, objectKey string, lifetimeSecs int64) (*s3.PresignedPostRequest, error)
{
    request, err := presigner.PresignClient.PresignPostObject(ctx,
    &s3.PutObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
    }, func(options *s3.PresignPostOptions) {
        options.Expires = time.Duration(lifetimeSecs) * time.Second
    })
    if err != nil {
        log.Printf("Couldn't get a presigned post request to put %v:%v. Here's why: %v
    \n", bucketName, objectKey, err)
    }
    return request, nil
}
```

미리 서명된 URL을 생성하고 사용하여 S3 객체를 업로드, 다운로드, 삭제하는 대화형 예제를 실행합니다.

```
import (
    "bytes"
    "context"
    "io"
    "log"
    "mime/multipart"
    "net/http"
    "os"
    "strings"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)
```



```
// RunPresigningScenario is an interactive example that shows you how to get
// presigned
// HTTP requests that you can use to move data into and out of Amazon Simple
// Storage
// Service (Amazon S3). The presigned requests contain temporary credentials and
// can
// be used by an HTTP client.
//
// 1. Get a presigned request to put an object in a bucket.
// 2. Use the net/http package to use the presigned request to upload a local
// file to the bucket.
// 3. Get a presigned request to get an object from a bucket.
// 4. Use the net/http package to use the presigned request to download the
// object to a local file.
// 5. Get a presigned request to delete an object from a bucket.
// 6. Use the net/http package to use the presigned request to delete the object.
//
// This example creates an Amazon S3 presign client from the specified sdkConfig
// so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ..\..\demotools folder of this repo.
//
// It uses an IHttpRequester interface to abstract HTTP requests so they can be
// mocked
// during testing.
func RunPresigningScenario(ctx context.Context, sdkConfig aws.Config, questioner
demotools.IQuestioner, httpRequester IHttpRequester) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
            _, isMock := questioner.(*demotools.MockQuestioner)
            if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
                log.Println(r)
            }
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Println("Welcome to the Amazon S3 presigning demo.")
    log.Println(strings.Repeat("-", 88))
}
```

```

s3Client := s3.NewFromConfig(sdkConfig)
bucketBasics := actions.BucketBasics{S3Client: s3Client}
presignClient := s3.NewPresignClient(s3Client)
presigner := actions.Presigner{PresignClient: presignClient}

bucketName := questioner.Ask("We'll need a bucket. Enter a name for a bucket "+
    "you own or one you want to create:", demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
if err != nil {
    panic(err)
}
if !bucketExists {
    err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
    if err != nil {
        panic(err)
    } else {
        log.Println("Bucket created.")
    }
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to upload a file to your bucket.")
uploadFilename := questioner.Ask("Enter the path to a file you want to upload:",
    demotools.NotEmpty{})
uploadKey := questioner.Ask("What would you like to name the uploaded object?",
    demotools.NotEmpty{})
uploadFile, err := os.Open(uploadFilename)
if err != nil {
    panic(err)
}
defer uploadFile.Close()
presignedPutRequest, err := presigner.PutObject(ctx, bucketName, uploadKey, 60)
if err != nil {
    panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedPutRequest.Method,
presignedPutRequest.URL)
log.Println("Using net/http to send the request...")
info, err := uploadFile.Stat()
if err != nil {
    panic(err)
}

```

```

putResponse, err := httpRequester.Put(presignedPutRequest.URL, info.Size(),
uploadFile)
if err != nil {
    panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedPutRequest.Method,
uploadKey, putResponse.StatusCode)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to download the object.")
questioner.Ask("Press Enter when you're ready.")
presignedGetRequest, err := presigner.GetObject(ctx, bucketName, uploadKey, 60)
if err != nil {
    panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedGetRequest.Method,
presignedGetRequest.URL)
log.Println("Using net/http to send the request...")
getResponse, err := httpRequester.Get(presignedGetRequest.URL)
if err != nil {
    panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedGetRequest.Method,
uploadKey, getResponse.StatusCode)
defer getResponse.Body.Close()
downloadBody, err := io.ReadAll(getResponse.Body)
if err != nil {
    panic(err)
}
log.Printf("Downloaded %v bytes. Here are the first 100 of them:\n",
len(downloadBody))
log.Println(strings.Repeat("-", 88))
log.Println(string(downloadBody[:100]))
log.Println(strings.Repeat("-", 88))

log.Println("Now we'll create a new request to put the same object using a
presigned post request")
questioner.Ask("Press Enter when you're ready.")
presignPostRequest, err := presigner.PresignPostObject(ctx, bucketName,
uploadKey, 60)
if err != nil {

```

```
    panic(err)
}
log.Printf("Got a presigned post request to url %v with values %v\n",
presignPostRequest.URL, presignPostRequest.Values)
log.Println("Using net/http multipart to send the request...")
uploadFile, err = os.Open(uploadFilename)
if err != nil {
    panic(err)
}
defer uploadFile.Close()
multiPartResponse, err := sendMultipartRequest(presignPostRequest.URL,
presignPostRequest.Values, uploadFile, uploadKey, httpRequester)
if err != nil {
    panic(err)
}
log.Printf("Presign post object %v with presigned URL returned %v.", uploadKey,
multiPartResponse.StatusCode)

log.Println("Let's presign a request to delete the object.")
questioner.Ask("Press Enter when you're ready.")
presignedDelRequest, err := presigner.DeleteObject(ctx, bucketName, uploadKey)
if err != nil {
    panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedDelRequest.Method,
presignedDelRequest.URL)
log.Println("Using net/http to send the request...")
delResponse, err := httpRequester.Delete(presignedDelRequest.URL)
if err != nil {
    panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.\n",
presignedDelRequest.Method,
uploadKey, delResponse.StatusCode)
log.Println(strings.Repeat("-", 88))

log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

예제가 HTTP 요청을 만드는 데 사용되는 HTTP 요청 래퍼를 정의합니다.

```
// IHttpRequester abstracts HTTP requests into an interface so it can be mocked
// during
// unit testing.
type IHttpRequester interface {
    Get(url string) (resp *http.Response, err error)
    Post(url, contentType string, body io.Reader) (resp *http.Response, err error)
    Put(url string, contentLength int64, body io.Reader) (resp *http.Response, err
    error)
    Delete(url string) (resp *http.Response, err error)
}

// HttpRequester uses the net/http package to make HTTP requests during the
// scenario.
type HttpRequester struct{}

func (httpReq HttpRequester) Get(url string) (resp *http.Response, err error) {
    return http.Get(url)
}

func (httpReq HttpRequester) Post(url, contentType string, body io.Reader) (resp
    *http.Response, err error) {
    postRequest, err := http.NewRequest("POST", url, body)
    if err != nil {
        return nil, err
    }
    postRequest.Header.Set("Content-Type", contentType)
    return http.DefaultClient.Do(postRequest)
}

func (httpReq HttpRequester) Put(url string, contentLength int64, body io.Reader)
    (resp *http.Response, err error) {
    putRequest, err := http.NewRequest("PUT", url, body)
    if err != nil {
        return nil, err
    }
    putRequest.ContentLength = contentLength
    return http.DefaultClient.Do(putRequest)
}

func (httpReq HttpRequester) Delete(url string) (resp *http.Response, err error)
    {
    delRequest, err := http.NewRequest("DELETE", url, nil)
    if err != nil {
```

```
    return nil, err
}
return http.DefaultClient.Do(delRequest)
}
```

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

다음은 미리 서명된 URL을 만들고 HTTP 클라이언트 라이브러리를 이용해 해당 URL을 사용하는 방법의 세 가지 예제를 보여줍니다.

- 3개의 HTTP 클라이언트 라이브러리를 이용해 URL을 사용하는 HTTP GET 요청
- 헤더에 메타데이터가 포함되어 있고 3개의 HTTP 클라이언트 라이브러리를 이용해 URL을 사용하는 HTTP PUT 요청
- 쿼리 파라미터가 있고 하나의 HTTP 클라이언트 라이브러리를 이용해 URL을 사용하는 HTTP PUT 요청

객체에 대해 미리 서명된 URL을 생성한 다음 다운로드(GET 요청)합니다.

가져옵니다.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
    software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.UUID;
```

URL을 생성합니다.

```
/* Create a pre-signed URL to download an object in a subsequent GET request.
*/
public String createPresignedGetUrl(String bucketName, String keyName) {
    try (S3Presigner presigner = S3Presigner.create()) {

        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        GetObjectPresignRequest presignRequest =
            GetObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(10)) // The URL will
                expire in 10 minutes.
                .getObjectRequest(objectRequest)
                .build();

        PresignedGetObjectRequest presignedRequest =
            presigner.presignGetObject(presignRequest);
```

```

        logger.info("Presigned URL: [{}]",
presignedRequest.url().toString());
        logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}

```

다음 세 가지 방법 중 하나를 사용하여 객체를 다운로드합니다.

JDK `URLConnection`(v1.1 이후) 클래스를 사용하여 다운로드합니다.

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
public byte[] useHttpURLConnectionToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setRequestMethod("GET");
        // Download the result of executing the request.
        try (InputStream content = connection.getInputStream()) {
            IoUtils.copy(content, byteArrayOutputStream);
        }
        logger.info("HTTP response code is " + connection.getResponseCode());
    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}

```

JDK `HttpClient`(v1.1 이후) 클래스를 사용하여 다운로드합니다.

```

/* Use the JDK HttpClient (since v1.1) class to do the download. */
public byte[] useHttpClientToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

```



```

HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
HttpClient httpClient = HttpClient.newHttpClient();
try {
    URL presignedUrl = new URL(presignedUrlString);
    HttpResponse<InputStream> response = httpClient.send(requestBuilder
        .uri(presignedUrl.toURI())
        .GET()
        .build(),
        HttpResponse.BodyHandlers.ofInputStream());

    IoUtils.copy(response.body(), byteArrayOutputStream);

    logger.info("HTTP response code is " + response.statusCode());
} catch (URISyntaxException | InterruptedException | IOException e) {
    logger.error(e.getMessage(), e);
}
return byteArrayOutputStream.toByteArray();
}

```

AWS SDK for Java SdkHttpClient 클래스를 사용하여 다운로드합니다.

```

/* Use the AWS SDK for Java SdkHttpClient class to do the download. */
public byte[] useSdkHttpClientToGet(String presignedUrlString) {

    ByteArrayOutputStream byteArrayOutputStream = new
    ByteArrayOutputStream(); // Capture the response body to a byte array.
    try {
        URL presignedUrl = new URL(presignedUrlString);
        SdkHttpRequest request = SdkHttpRequest.builder()
            .method(SdkHttpMethod.GET)
            .uri(presignedUrl.toURI())
            .build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
            sdkHttpClient.prepareRequest(executeRequest).call();

```

```

        response.responseBody().ifPresentOrElse(
            abortableInputStream -> {
                try {
                    IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            },
            () -> logger.error("No response body."));

        logger.info("HTTP Response code is {}",
response.httpResponse().statusCode());
    }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}

```

업로드를 위해 헤더에 메타데이터가 있는 미리 서명된 URL을 생성한 다음 파일을 업로드(PUT 요청)합니다.

가져옵니다.

```

import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

```

```
import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

URL을 생성합니다.

```
/* Create a presigned URL to use in a subsequent PUT request */
public String createPresignedUrl(String bucketName, String keyName,
    Map<String, String> metadata) {
    try (S3Presigner presigner = S3Presigner.create()) {

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .metadata(metadata)
            .build();

        PutObjectPresignRequest presignRequest =
            PutObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(10)) // The URL
                expires in 10 minutes.
                .putObjectRequest(objectRequest)
                .build();

        PresignedPutObjectRequest presignedRequest =
            presigner.presignPutObject(presignRequest);
```

```

        String myURL = presignedRequest.url().toString();
        logger.info("Presigned URL to upload a file to: [{}]", myURL);
        logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}

```

다음 세 가지 방법 중 하나를 사용하여 파일 객체를 업로드합니다.

JDK `URLConnection`(v1.1 이후) 클래스를 사용하여 업로드합니다.

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
public void useHttpURLConnectionToPut(String presignedUrlString, File
fileToPut, Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setDoOutput(true);
        metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-
meta-" + k, v));
        connection.setRequestMethod("PUT");
        OutputStream out = connection.getOutputStream();

        try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
            FileChannel inChannel = file.getChannel()) {
            ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is
8k

            while (inChannel.read(buffer) > 0) {
                buffer.flip();
                for (int i = 0; i < buffer.limit(); i++) {
                    out.write(buffer.get());
                }
                buffer.clear();
            }
        } catch (IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
}

```

```

        out.close();
        connection.getResponseCode();
        logger.info("HTTP response code is " + connection.getResponseCode());

    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
}

```

JDK HttpClient(v11 이후) 클래스를 사용하여 업로드합니다.

```

/* Use the JDK HttpClient (since v11) class to do the upload. */
public void useHttpClientToPut(String presignedUrlString, File fileToPut,
    Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        final HttpResponse<Void> response = httpClient.send(requestBuilder
            .uri(new URL(presignedUrlString).toURI())

        .PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI()))
            .build(),
            HttpResponse.BodyHandlers.discarding());

        logger.info("HTTP response code is " + response.statusCode());

    } catch (URISyntaxException | InterruptedException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}

```

AWS for Java V2 SdkHttpClient 클래스를 사용하여 업로드합니다.

```

/* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
    Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

```

```

try {
    URL presignedUrl = new URL(presignedUrlString);

    SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
        .method(SdkHttpMethod.PUT)
        .uri(presignedUrl.toURI());
    // Add headers
    metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" +
k, v));
    // Finish building the request.
    SdkHttpRequest request = requestBuilder.build();

    HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
        .request(request)
        .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
        .build();

    try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
        HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
        logger.info("Response code: {}",
response.httpResponse().statusCode());
    }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}

```

업로드를 위해 쿼리 파라미터가 있는 미리 서명된 URL을 생성한 다음 파일을 업로드(PUT 요청)합니다.

가져옵니다.

```

import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.awscore.AWSRequestOverrideConfiguration;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;

```

```
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

URL을 생성합니다.

```
/**
 * Creates a presigned URL to use in a subsequent HTTP PUT request. The code
 * adds query parameters
 * to the request instead of using headers. By using query parameters, you
 * do not need to add the
 * the parameters as headers when the PUT request is eventually sent.
 *
 * @param bucketName Bucket name where the object will be uploaded.
 * @param keyName Key name of the object that will be uploaded.
 * @param queryParams Query string parameters to be added to the presigned
 * URL.
 * @return
 */
public String createPresignedUrl(String bucketName, String keyName,
    Map<String, String> queryParams) {
    try (S3Presigner presigner = S3Presigner.create()) {
        // Create an override configuration to store the query parameters.
        AwsRequestOverrideConfiguration.Builder overrideConfigurationBuilder
        = AwsRequestOverrideConfiguration.builder();
```

```

queryParams.forEach(overrideConfigurationBuilder::putRawQueryParameter);

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)

.overrideConfiguration(overrideConfigurationBuilder.build()) // Add the override
configuration.

            .build();

        PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10)) // The URL
expires in 10 minutes.
            .putObjectRequest(objectRequest)
            .build();

        PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
        String myURL = presignedRequest.url().toString();
        logger.info("Presigned URL to upload a file to: [{}]", myURL);
        logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}

```

AWS for Java V2 SdkHttpClient 클래스를 사용하여 업로드합니다.

```

/**
 * Use the AWS SDK for Java V2 SdkHttpClient class to execute the PUT
request. Since the
 * URL contains the query parameters, no headers are needed for metadata, SSE
settings, or ACL settings.
 *
 * @param presignedUrlString The URL for the PUT request.
 * @param fileToPut File to upload
 */

```



```

public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut)
{
    logger.info("Begin [{}] upload", fileToPut.toString());

    try {
        URL presignedUrl = new URL(presignedUrlString);

        SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
            .method(SdkHttpMethod.PUT)
            .uri(presignedUrl.toURI());

        SdkHttpRequest request = requestBuilder.build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
            logger.info("Response code: {}",
response.httpResponse().statusCode());
        }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}

```

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

미리 서명된 URL을 생성하여 버킷에 객체를 업로드합니다.

```

import https from "node:https";

import { XMLParser } from "fast-xml-parser";
import { PutObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
  getSignedUrl,
  S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(
    new HttpRequest({ ...url, method: "PUT" })
  );
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
  const command = new PutObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};

/**
 * Make a PUT request to the provided URL.
 *
 * @param {string} url
 * @param {string} data
 */
const put = (url, data) => {
  return new Promise((resolve, reject) => {
    const req = https.request(

```

```

    url,
    { method: "PUT", headers: { "Content-Length": new Blob([data]).size } },
    (res) => {
      let responseBody = "";
      res.on("data", (chunk) => {
        responseBody += chunk;
      });
      res.on("end", () => {
        const parser = new XMLParser();
        if (res.statusCode >= 200 && res.statusCode <= 299) {
          resolve(parser.parse(responseBody, true));
        } else {
          reject(parser.parse(responseBody, true));
        }
      });
    },
  );
  req.on("error", (err) => {
    reject(err);
  });
  req.write(data);
  req.end();
});
};

/**
 * Create two presigned urls for uploading an object to an S3 bucket.
 * The first presigned URL is created with credentials from the shared INI file
 * in the current environment. The second presigned URL is created using an
 * existing S3Client instance that has already been provided with credentials.
 * @param {{ bucketName: string, key: string, region: string }}
 */
export const main = async ({ bucketName, key, region }) => {
  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      bucket: bucketName,
      key,
      region,
    });

    const clientUrl = await createPresignedUrlWithClient({
      bucket: bucketName,
      region,
      key,

```

```

    });

    // After you get the presigned URL, you can provide your own file
    // data. Refer to put() above.
    console.log("Calling PUT using presigned URL without client");
    await put(noClientUrl, "Hello World");

    console.log("Calling PUT using presigned URL with client");
    await put(clientUrl, "Hello World");

    console.log("\nDone. Check your S3 console.");
  } catch (caught) {
    if (caught instanceof Error && caught.name === "CredentialsProviderError") {
      console.error(
        `There was an error getting your credentials. Are your local credentials
        configured?\n${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

```

미리 서명된 URL을 생성하여 버킷에서 객체를 다운로드합니다.

```

import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
  getSignedUrl,
  S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
  });

```

```
});

const signedUrlObject = await presigner.presign(new HttpRequest(url));
return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
  const command = new GetObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};

/**
 * Create two presigned urls for downloading an object from an S3 bucket.
 * The first presigned URL is created with credentials from the shared INI file
 * in the current environment. The second presigned URL is created using an
 * existing S3Client instance that has already been provided with credentials.
 * @param {{ bucketName: string, key: string, region: string }}
 */
export const main = async ({ bucketName, key, region }) => {
  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      bucket: bucketName,
      region,
      key,
    });

    const clientUrl = await createPresignedUrlWithClient({
      bucket: bucketName,
      region,
      key,
    });

    console.log("Presigned URL without client");
    console.log(noClientUrl);
    console.log("\n");

    console.log("Presigned URL with client");
    console.log(clientUrl);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "CredentialsProviderError") {
      console.error(
        `There was an error getting your credentials. Are your local credentials configured?\n${caught.name}: ${caught.message}`,
      );
    }
  }
}
```

```
    );  
  } else {  
    throw caught;  
  }  
}  
};
```

- 자세한 정보는 [AWS SDK for JavaScript 개발자 안내서](#)를 참조하십시오.

Kotlin

SDK for Kotlin

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

미리 서명된 `GetObject` 요청을 만들고 URL을 사용하여 객체를 다운로드합니다.

```
suspend fun getObjectPresigned(  
    s3: S3Client,  
    bucketName: String,  
    keyName: String,  
): String {  
    // Create a GetObjectRequest.  
    val unsignedRequest =  
        GetObjectRequest {  
            bucket = bucketName  
            key = keyName  
        }  
  
    // Presign the GetObject request.  
    val presignedRequest = s3.presignGetObject(unsignedRequest, 24.hours)  
  
    // Use the URL from the presigned HttpRequest in a subsequent HTTP GET  
    request to retrieve the object.  
    val objectContents = URL(presignedRequest.url.toString()).readText()  
  
    return objectContents  
}
```

```
}
```

고급 옵션을 사용하여 GetObject 미리 서명된 요청을 생성합니다.

```
suspend fun getObjectPresignedMoreOptions(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): HttpRequest {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest =
        s3.presignGetObject(unsignedRequest, signer = CrtAwsSigner) {
            signingDate = Instant.now() + 12.hours // Presigned request can be
            used 12 hours from now.
            algorithm = AwsSigningAlgorithm.SIGV4_ASYMMETRIC
            signatureType = AwsSignatureType.HTTP_REQUEST_VIA_QUERY_PARAMS
            expiresAfter = 8.hours // Presigned request expires 8 hours later.
        }
    return presignedRequest
}
```

미리 서명된 PutObject 요청을 만들고 이를 사용하여 객체를 업로드합니다.

```
suspend fun putObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
    content: String,
) {
    // Create a PutObjectRequest.
    val unsignedRequest =
        PutObjectRequest {
            bucket = bucketName
            key = keyName
        }
```

```

    }

    // Presign the request.
    val presignedRequest = s3.presignPutObject(unsignedRequest, 24.hours)

    // Use the URL and any headers from the presigned HttpRequest in a subsequent
    HTTP PUT request to retrieve the object.
    // Create a PUT request using the OkHttpClient API.
    val putRequest =
        Request
            .Builder()
            .url(presignedRequest.url.toString())
            .apply {
                presignedRequest.headers.forEach { key, values ->
                    header(key, values.joinToString(", "))
                }
            }
            .put(content.toRequestBody())
            .build()

    val response = OkHttpClient().newCall(putRequest).execute()
    assert(response.isSuccessful)
}

```

- 자세한 내용은 [AWS SDK for Kotlin 개발자 안내서](#)를 참조하세요.

PHP

SDK for PHP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

namespace S3;
use Aws\Exception\AwsException;
use AwsUtilities\PrintableLineBreak;
use AwsUtilities\TestableReadline;
use DateTime;

```



```
require 'vendor/autoload.php';

class PresignedURL
{
    use PrintableLineBreak;
    use TestableReadline;

    public function run()
    {
        $s3Service = new S3Service();

        $expiration = new DateTime("+20 minutes");
        $linebreak = $this->getLineBreak();

        echo $linebreak;
        echo ("Welcome to the Amazon S3 presigned URL demo.\n");
        echo $linebreak;

        $bucket = $this->testable_readline("First, please enter the name of the
S3 bucket to use: ");
        $key = $this->testable_readline("Next, provide the key of an object in
the given bucket: ");
        echo $linebreak;
        $command = $s3Service->getClient()->getCommand('GetObject', [
            'Bucket' => $bucket,
            'Key' => $key,
        ]);
        try {
            $preSignedUrl = $s3Service->preSignedUrl($command, $expiration);
            echo "Your preSignedUrl is \n$preSignedUrl\nand will be good for the
next 20 minutes.\n";
            echo $linebreak;
            echo "Thanks for trying the Amazon S3 presigned URL demo.\n";
        } catch (AwsException $exception) {
            echo $linebreak;
            echo "Something went wrong: $exception";
            die();
        }
    }
}

$runner = new PresignedURL();
$runner->run();
```

```
namespace S3;

use Aws\CommandInterface;
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;

    public function __construct(S3Client $client = null, $verbose = false)
    {
        if ($client) {
            $this->client = $client;
        } else {
            $this->client = new S3Client([
                'version' => 'latest',
                'region' => 'us-west-2',
            ]);
        }
        $this->verbose = $verbose;
    }

    public function setVerbose($verbose)
    {
        $this->verbose = $verbose;
    }

    public function isVerbose(): bool
    {
        return $this->verbose;
    }

    public function getClient(): S3Client
    {
        return $this->client;
    }
}
```

```
public function setClient(S3Client $client)
{
    $this->client = $client;
}

public function emptyAndDeleteBucket($bucketName, array $args = [])
{
    try {
        $objects = $this->listAllObjects($bucketName, $args);
        $this->deleteObjects($bucketName, $objects, $args);
        if ($this->verbose) {
            echo "Deleted all objects and folders from $bucketName.\n";
        }
        $this->deleteBucket($bucketName, $args);
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
            echo "\nPlease fix error with bucket deletion before continuing.
\n";
        }
        throw $exception;
    }
}

public function createBucket(string $bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);
    try {
        $this->client->createBucket($parameters);
        if ($this->verbose) {
            echo "Created the bucket named: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
            echo "Please fix error with bucket creation before continuing.";
        }
        throw $exception;
    }
}
```

```

    }
}

public function putObject(string $bucketName, string $key, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
$args);
    try {
        $this->client->putObject($parameters);
        if ($this->verbose) {
            echo "Uploaded the object named: $key to the bucket named:
$bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to create $key in $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object uploading before continuing.";
        }
        throw $exception;
    }
}

public function getObject(string $bucketName, string $key, array $args = []):
Result
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
$args);
    try {
        $object = $this->client->getObject($parameters);
        if ($this->verbose) {
            echo "Downloaded the object named: $key to the bucket named:
$bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to download $key from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object downloading before
continuing.";
        }
    }
}

```

```

        }
        throw $exception;
    }
    return $object;
}

public function copyObject($bucketName, $key, $copySource, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
"CopySource" => $copySource], $args);
    try {
        $this->client->copyObject($parameters);
        if ($this->verbose) {
            echo "Copied the object from: $copySource in $bucketName to:
$key.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to copy $copySource in $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object copying before continuing.";
        }
        throw $exception;
    }
}

public function listObjects(string $bucketName, $start = 0, $max = 1000,
array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
"MaxKeys" => $max], $args);
    try {
        $objects = $this->client->listObjectsV2($parameters);
        if ($this->verbose) {
            echo "Retrieved the list of objects from: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to retrieve the objects from $bucketName with error:
{$exception->getMessage()}\n";

```

```

        echo "Please fix error with list objects before continuing.";
    }
    throw $exception;
}
return $objects;
}

public function listAllObjects($bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);

    $contents = [];
    $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

    foreach ($paginator as $result) {
        if($result['KeyCount'] == 0){
            break;
        }
        foreach ($result['Contents'] as $object) {
            $contents[] = $object;
        }
    }
    return $contents;
}

public function deleteObjects(string $bucketName, array $objects, array $args
= [])
{
    $listOfObjects = array_map(
        function ($object) {
            return ['Key' => $object];
        },
        array_column($objects, 'Key')
    );
    if(!$listOfObjects){
        return;
    }

    $parameters = array_merge(['Bucket' => $bucketName, 'Delete' =>
['Objects' => $listOfObjects]], $args);

```

```

        try {
            $this->client->deleteObjects($parameters);
            if ($this->verbose) {
                echo "Deleted the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete the list of objects from $bucketName with
error: {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }

    public function deleteBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->deleteBucket($parameters);
            if ($this->verbose) {
                echo "Deleted the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket deletion before continuing.";
            }
            throw $exception;
        }
    }

    public function deleteObject(string $bucketName, string $fileName, array
$args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
$args);
        try {
            $this->client->deleteObject($parameters);

```

```

        if ($this->verbose) {
            echo "Deleted the object named: $fileName from $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete $fileName from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object deletion before continuing.";
        }
        throw $exception;
    }
}

public function listBuckets(array $args = [])
{
    try {
        $buckets = $this->client->listBuckets($args);
        if ($this->verbose) {
            echo "Retrieved all " . count($buckets) . "\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
            echo "Please fix error with bucket lists before continuing.";
        }
        throw $exception;
    }
    return $buckets;
}

public function preSignedUrl(CommandInterface $command, DateTimeInterface|
int|string $expires, array $options = [])
{
    $request = $this->client->createPresignedRequest($command, $expires,
$options);
    try {
        $presignedUrl = (string)$request->getUri();
    } catch (AwsException $exception) {
        if ($this->verbose) {

```



```
        echo "Failed to create a presigned url: {$exception-  
>getMessage()}\n";  
        echo "Please fix error with presigned urls before continuing.";  
    }  
    throw $exception;  
}  
return $presignedUrl;  
}  
  
public function createSession(string $bucketName)  
{  
    try{  
        $result = $this->client->createSession([  
            'Bucket' => $bucketName,  
        ]);  
        return $result;  
    }catch(S3Exception $caught){  
        if($caught->getAwsErrorType() == "NoSuchBucket"){  
            echo "The specified bucket does not exist.";  
        }  
        throw $caught;  
    }  
}  
}
```

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

제한된 시간 동안 S3 작업을 수행할 수 있는 미리 서명된 URL을 생성합니다. 요청 패키지를 사용하여 URL로 요청을 수행합니다.

```
import argparse
import logging
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)

def generate_presigned_url(s3_client, client_method, method_parameters,
                           expires_in):
    """
    Generate a presigned Amazon S3 URL that can be used to perform an action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds the presigned URL is valid for.
    :return: The presigned URL.
    """
    try:
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method, Params=method_parameters,
            ExpiresIn=expires_in
        )
        logger.info("Got presigned URL: %s", url)
    except ClientError:
        logger.exception(
            "Couldn't get a presigned URL for client method '%s'.", client_method
        )
        raise
    return url

def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon S3 presigned URL demo.")
    print("-" * 88)
```

```

parser = argparse.ArgumentParser()
parser.add_argument("bucket", help="The name of the bucket.")
parser.add_argument(
    "key",
    help="For a GET operation, the key of the object in Amazon S3. For a "
    "PUT operation, the name of a file to upload.",
)
parser.add_argument("action", choices=("get", "put"), help="The action to
perform.")
args = parser.parse_args()

s3_client = boto3.client("s3")
client_action = "get_object" if args.action == "get" else "put_object"
url = generate_presigned_url(
    s3_client, client_action, {"Bucket": args.bucket, "Key": args.key}, 1000
)

print("Using the Requests package to send a request to the URL.")
response = None
if args.action == "get":
    response = requests.get(url)
    if response.status_code == 200:
        with open(args.key.split("/")[-1], 'wb') as object_file:
            object_file.write(response.content)
elif args.action == "put":
    print("Putting data to the URL.")
    try:
        with open(args.key, "rb") as object_file:
            object_text = object_file.read()
            response = requests.put(url, data=object_text)
    except FileNotFoundError:
        print(
            f"Couldn't find {args.key}. For a PUT operation, the key must be
the "
            f"name of a file that exists on your computer."
        )

if response is not None:
    print(f"Status: {response.status_code}\nReason: {response.reason}")

print("-" * 88)

```

```
if __name__ == "__main__":
    usage_demo()
```

미리 서명된 POST 요청을 생성하여 파일을 업로드합니다.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def generate_presigned_post(self, object_key, expires_in):
        """
        Generate a presigned Amazon S3 POST request to upload a file.
        A presigned POST can be used for a limited time to let someone without an
        AWS
        account upload a file to a bucket.

        :param object_key: The object key to identify the uploaded object.
        :param expires_in: The number of seconds the presigned POST is valid.
        :return: A dictionary that contains the URL and form fields that contain
                required access data.
        """
        try:
            response = self.bucket.meta.client.generate_presigned_post(
                Bucket=self.bucket.name, Key=object_key, ExpiresIn=expires_in
            )
            logger.info("Got presigned POST URL: %s", response["url"])
        except ClientError:
            logger.exception(
                "Couldn't get a presigned POST URL for bucket '%s' and object
                '%s'",
                self.bucket.name,
                object_key,
            )
```

```
        raise
    return response
```

Ruby

SDK for Ruby

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-s3'
require 'net/http'

# Creates a presigned URL that can be used to upload content to an object.
#
# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
# @param object_key [String] The key to give the uploaded object.
# @return [URI, nil] The parsed URI if successful; otherwise nil.
def get_presigned_url(bucket, object_key)
  url = bucket.object(object_key).presigned_url(:put)
  puts "Created presigned URL: #{url}"
  URI(url)
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create presigned URL for #{bucket.name}:#{object_key}. Here's
  why: #{e.message}"
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-file.txt"
  object_content = "This is the content of my-file.txt."

  bucket = Aws::S3::Bucket.new(bucket_name)
  presigned_url = get_presigned_url(bucket, object_key)
```

```

return unless presigned_url

response = Net::HTTP.start(presigned_url.host) do |http|
  http.send_request('PUT', presigned_url.request_uri, object_content,
'content_type' => '')
end

case response
when Net::HTTPSuccess
  puts 'Content uploaded!'
else
  puts response.value
end
end

run_demo if $PROGRAM_NAME == __FILE__

```

Rust

SDK for Rust

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

GET S3 객체에 대한 사전 지정 요청을 만듭니다.

```

/// Generate a URL for a presigned GET request.
async fn get_object(
  client: &Client,
  bucket: &str,
  object: &str,
  expires_in: u64,
) -> Result<(), Box<dyn Error>> {
  let expires_in = Duration::from_secs(expires_in);
  let presigned_request = client
    .get_object()
    .bucket(bucket)
    .key(object)

```

```

        .presigned(PresigningConfig::expires_in(expires_in?))
        .await?;

println!("Object URI: {}", presigned_request.uri());
let valid_until = chrono::offset::Local::now() + expires_in;
println!("Valid until: {valid_until}");

Ok(())
}

```

PUT S3 객체에 대한 사전 지정 요청을 만듭니다.

```

async fn put_object(
    client: &Client,
    bucket: &str,
    object: &str,
    expires_in: u64,
) -> Result<String, S3ExampleError> {
    let expires_in: std::time::Duration =
        std::time::Duration::from_secs(expires_in);
    let expires_in: aws_sdk_s3::presigning::PresigningConfig =
        PresigningConfig::expires_in(expires_in).map_err(|err| {
            S3ExampleError::new(format!(
                "Failed to convert expiration to PresigningConfig: {err:?}"
            ))
        })?;
    let presigned_request = client
        .put_object()
        .bucket(bucket)
        .key(object)
        .presigned(expires_in)
        .await?;

    Ok(presigned_request.uri().into())
}

```

SAP ABAP

SDK for SAP ABAP

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

GET S3 객체에 대한 사전 서명된 요청을 만듭니다.

```
" iv_bucket_name is the bucket name
" iv_key is the object name like "myfile.txt"

DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

"Upload a nice Hello World file to an S3 bucket."
TRY.
    DATA(lv_contents) = cl_abap_codepage=>convert_to( 'Hello, World' ).
    lo_s3->putobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key
        iv_body = lv_contents
        iv_contenttype = 'text/plain' ).
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" now generate a presigned URL with a 600-second expiration
DATA(lo_presigner) = lo_s3->get_presigner( iv_expires_sec = 600 ).
" the presigner getobject() method has the same signature as
" lo_s3->getobject(), but it doesn't actually make the call
" to the service. It just prepares a presigned URL for a future call
DATA(lo_presigned_req) = lo_presigner->getobject(
    iv_bucket = iv_bucket_name
    iv_key = iv_key ).

" You can provide this URL to a web page, user, email etc so they
" can retrieve the file. The URL will expire in 10 minutes.
ov_url = lo_presigned_req->get_url( ).
```


AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

사용자가 레이블을 사용하여 사진을 관리할 수 있는 사진 자산 관리 애플리케이션 만들기

다음 코드 예제는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여 줍니다.

.NET

SDK for .NET

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 [AWS 커뮤니티](#)의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK for C++

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 [AWS 커뮤니티](#)의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK for Java 2.x

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 [AWS 커뮤니티](#)의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK for JavaScript (v3)

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 [AWS 커뮤니티](#)의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK for Kotlin

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 [AWS 커뮤니티](#)의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK for PHP

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 [AWS 커뮤니티](#)의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK for Rust

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 [AWS 커뮤니티](#)의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon S3 객체를 나열하는 웹 페이지

다음 코드 예시는 웹 페이지에 Amazon S3 객체를 나열하는 방법을 보여줍니다.

JavaScript

JavaScript용 SDK(v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

다음 코드는 AWS SDK에 호출을 수행하는 관련 React 구성 요소입니다. 이 구성 요소가 포함된 실행 가능한 애플리케이션 버전은 이전 GitHub 링크에서 찾을 수 있습니다.

```
import { useEffect, useState } from "react";
import {
  ListObjectsCommand,
  type ListObjectsCommandOutput,
  S3Client,
} from "@aws-sdk/client-s3";
import { fromCognitoIdentityPool } from "@aws-sdk/credential-providers";
import "./App.css";

function App() {
  const [objects, setObjects] = useState<
    Required<ListObjectsCommandOutput>["Contents"]
  >([]);

  useEffect(() => {
    const client = new S3Client({
      region: "us-east-1",
      // Unless you have a public bucket, you'll need access to a private bucket.
      // One way to do this is to create an Amazon Cognito identity pool, attach
      // a role to the pool,
      // and grant the role access to the 's3:GetObject' action.
      //
      // You'll also need to configure the CORS settings on the bucket to allow
      // traffic from
      // this example site. Here's an example configuration that allows all
      // origins. Don't
      // do this in production.
      // [
      // {
      //   origin: '*',
      //   methods: 'GET',
      //   allowedHeaders: '*',
      // },
      // ]
    });
```

```

    //    "AllowedHeaders": ["*"],
    //    "AllowedMethods": ["GET"],
    //    "AllowedOrigins": ["*"],
    //    "ExposeHeaders": [],
    //  },
  //]
  //
  credentials: fromCognitoIdentityPool({
    clientConfig: { region: "us-east-1" },
    identityPoolId: "<YOUR_IDENTITY_POOL_ID>",
  }),
});
const command = new ListObjectsCommand({ Bucket: "bucket-name" });
client.send(command).then(({ Contents }) => setObjects(Contents || []));
}, []);

return (
  <div className="App">
    {objects.map((o) => (
      <div key={o.ETag}>{o.Key}</div>
    ))}
  </div>
);
}

export default App;

```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 [ListObjects](#)를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon Textract 탐색기 애플리케이션 생성

다음 코드 예제에서는 대화형 애플리케이션을 통해 Amazon Textract 출력을 탐색하는 방법을 보여줍니다.

JavaScript

JavaScript용 SDK(v3)

Amazon Textract를 사용하여 문서 이미지에서 데이터를 추출하고 대화형 웹 페이지에 표시하는 React 애플리케이션을 AWS SDK for JavaScript로 구축하는 방법을 보여줍니다. 이 예제는 웹 브라우저에서 실행되며 자격 증명을 위해 인증된 Amazon Cognito 자격 증명에 필요합니다. 이 애플리케이션은 스토리지로 Amazon Simple Storage Service(Amazon S3)를 사용하고 알림을 위해 Amazon Simple Notification Service(Amazon SNS) 주제를 구독하는 Amazon Simple Queue Service(Amazon SQS) 대기열을 폴링합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Cognito 자격 증명
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK for Python (Boto3)

AWS SDK for Python (Boto3)을 Amazon Textract와 함께 사용하여 문서 이미지의 텍스트, 양식 및 테이블 요소를 감지하는 방법을 보여줍니다. 입력 이미지와 Amazon Textract 출력은 탐지된 요소를 탐색할 수 있는 Tkinter 애플리케이션에 표시됩니다.

- 문서 이미지를 Amazon Textract에 제출하고 감지된 요소의 출력을 탐색합니다.
- Amazon Textract로 직접, 또는 Amazon Simple Storage Service(Amazon S3) 버킷을 통해 이미지를 제출합니다.
- 비동기식 API를 사용하여 작업이 완료되면 Amazon Simple Notification Service(Amazon SNS) 주제에 알림을 게시하는 작업을 시작합니다.
- Amazon Simple Queue Service(Amazon SQS) 대기열에서 작업 완료 메시지를 폴링하고 결과를 표시합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Cognito 자격 증명
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 주어진 Amazon S3 버킷의 모든 객체 삭제

다음 코드 예제는 Amazon S3 버킷의 모든 객체를 삭제하는 방법을 보여줍니다.

JavaScript

JavaScript용 SDK(v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

주어진 Amazon S3 버킷의 모든 객체를 삭제합니다.

```
import {
  DeleteObjectsCommand,
  paginateListObjectsV2,
  S3Client,
} from "@aws-sdk/client-s3";

/**
 *
 * @param {{ bucketName: string }} config
 */
export const main = async ({ bucketName }) => {
```



```
const client = new S3Client({});
try {
  console.log(`Deleting all objects in bucket: ${bucketName}`);

  const paginator = paginateListObjectsV2(
    { client },
    {
      Bucket: bucketName,
    },
  );

  const objectKeys = [];
  for await (const { Contents } of paginator) {
    objectKeys.push(...Contents.map((obj) => ({ Key: obj.Key })));
  }

  const deleteCommand = new DeleteObjectsCommand({
    Bucket: bucketName,
    Delete: { Objects: objectKeys },
  });

  await client.send(deleteCommand);

  console.log(`All objects deleted from bucket: ${bucketName}`);
} catch (caught) {
  if (caught instanceof Error) {
    console.error(
      `Failed to empty ${bucketName}. ${caught.name}: ${caught.message}`,
    );
  }
}
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const options = {
    bucketName: {
      type: "string",
    },
  };
};

const { values } = parseArgs({ options });
```

```
main(values);  
}
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 다음 주제를 참조하세요.
 - [DeleteObjects](#)
 - [ListObjectsV2](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon S3에 대해 완료되지 않은 멀티파트 업로드 삭제

다음 코드 예시는 미완료 Amazon S3 멀티파트 업로드를 삭제 또는 중지하는 방법을 보여줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

어떤 이유로든 진행 중이거나 완료되지 않은 멀티파트 업로드를 중지하려면 다음 예와 같이 업로드 목록을 가져온 다음 삭제하면 됩니다.

```
/**  
 * Aborts all incomplete multipart uploads from the specified S3 bucket.  
 *  
 * @param bucketName the name of the S3 bucket  
 */  
public static void abortIncompleteMultipartUploadsFromList(String bucketName)  
{  
    ListMultipartUploadsRequest listMultipartUploadsRequest =  
ListMultipartUploadsRequest.builder()  
        .bucket(bucketName)  
        .build();
```

```

        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(upload.key())
                .expectedBucketOwner(accountId)
                .uploadId(upload.uploadId())
                .build();

            AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
            if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
            }
        }
    }
}

```

특정 날짜 전 또는 후에 시작된 미완료 멀티파트 업로드를 삭제하려면 다음 예와 같이 특정 시점을 기준으로 멀티파트 업로드를 선택적으로 삭제할 수 있습니다.

```

/**
 * Aborts incomplete multipart uploads older than the specified point in
time.
 *
 * @param bucketName the name of the S3 bucket
 * @param pointInTime the cutoff time; uploads initiated before this are
aborted
 */
static void abortIncompleteMultipartUploadsOlderThan(String bucketName,
Instant pointInTime) {
    ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
        .bucket(bucketName)
        .build();
}

```

```

        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            logger.info("Found multipartUpload with upload ID [{}], initiated
[{}]", upload.uploadId(), upload.initiated());
            if (upload.initiated().isBefore(pointInTime)) {
                abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(upload.key())
                    .expectedBucketOwner(accountId)
                    .uploadId(upload.uploadId())
                    .build();

                AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
                if
(abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                    logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
                }
            }
        }
    }
}

```

멀티파트 업로드를 시작한 후 업로드 ID에 액세스할 수 있는 경우 ID를 사용하여 진행 중인 업로드를 삭제할 수 있습니다.

```

/**
 * Aborts a multipart upload using the upload ID.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the object key
 */
static void abortMultipartUploadUsingUploadId(String bucketName, String key)
{
    String uploadId = startUploadReturningUploadId(bucketName, key);
    AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -
> b

```

```

        .uploadId(uploadId)
        .bucket(bucketName)
        .key(key));

    if (response.sdkHttpResponse().isSuccessful()) {
        logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
uploadId, bucketName);
    }
}

```

특정 일수보다 오래된 미완료 멀티파트 업로드를 지속적으로 삭제하려면 버킷의 버킷 수명 주기 구성을 설정하세요. 다음 예시에서는 7일보다 오래된 미완료 업로드를 삭제하는 규칙을 생성하는 방법을 보여줍니다.

```

/**
 * Configures a lifecycle rule to abort incomplete multipart uploads after 7
days.
 *
 * @param bucketName the name of the S3 bucket
 */
static void abortMultipartUploadsUsingLifecycleConfig(String bucketName) {
    Collection<LifecycleRule> lifeCycleRules =
List.of(LifecycleRule.builder()
        .abortIncompleteMultipartUpload(b -> b.
            daysAfterInitiation(7))
        .status("Enabled")
        .filter(SdkBuilder::build) // Filter element is required.
        .build());

    PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
        .bucket(bucketName)
        .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

    if (response.sdkHttpResponse().isSuccessful()) {
        logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
    } else {
        logger.error("Unsuccessfully applied rule. HTTP status code is [{})",
response.sdkHttpResponse().statusCode());
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
 - [AbortMultipartUpload](#)
 - [ListMultipartUploads](#)
 - [PutBucketLifecycleConfiguration](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon Rekognition을 통해 이미지에서 PPE 감지

다음 코드 예제에서는 Amazon Rekognition을 사용하여 이미지에서 개인 보호 장비(PPE)를 감지하는 앱을 구축하는 방법을 보여줍니다.

Java

SDK for Java 2.x

개인 보호 장비로 이미지를 감지하는 AWS Lambda 함수를 만드는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 이미지에서 추출한 텍스트의 개체 삭제

다음 코드 예제는 Amazon Comprehend를 사용하여 Amazon S3에 저장된 이미지에서 Amazon Textract를 통해 추출한 텍스트의 개체를 감지하는 방법을 보여줍니다.

Python

SDK for Python(Boto3)

Jupyter Notebook에서 AWS SDK for Python (Boto3)를 사용하여 이미지에서 추출한 텍스트의 개체를 감지하는 방법을 보여줍니다. 이 예제에서는 Amazon Textract를 통해 Amazon Simple Storage Service(Amazon S3) 및 Amazon Comprehend에 저장된 이미지에서 텍스트를 추출하여 추출된 텍스트의 엔터티를 감지합니다.

이 예제는 Jupyter Notebook에 관한 것이며, 노트북을 호스팅할 수 있는 환경에서 실행되어야 합니다. Amazon SageMaker AI를 사용하여 예제를 실행하는 방법에 대한 안내는 [TextractAndComprehendNotebook.ipynb](#)의 지침을 참조하세요.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Comprehend
- Amazon S3
- Amazon Textract

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 이미지에서 얼굴 감지

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- Amazon S3 버킷에 이미지를 저장합니다.
- Amazon Rekognition을 사용하여 연령대, 성별, 감정(예제: 웃음) 등의 얼굴 세부 정보를 감지합니다.
- 이러한 세부 정보를 표시합니다.

Rust

SDK for Rust

uploads 접두사를 사용하여 Amazon S3 버킷에 이미지를 저장하고, Amazon Rekognition을 사용하여 연령대, 성별, 감정(예제: 웃음) 등의 얼굴 세부 정보를 감지한 후 이러한 세부 정보를 표시합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Rekognition
- Amazon S3

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon Rekognition을 통해 이미지에서 객체 감지

다음 코드 예제는 Amazon Rekognition을 사용하여 이미지에서 범주별 객체를 감지하는 앱을 구축하는 방법을 보여줍니다.

.NET

SDK for .NET

Amazon Rekognition .NET을 사용하여 Amazon Simple Storage Service(Amazon S3) 버킷에 있는 이미지에서 범주별로 객체를 식별하기 위해 Amazon Rekognition을 사용하여 앱을 만드는 방법을 보여줍니다. 이 앱은 Amazon Simple Email Service(Amazon SES)를 사용하여 결과와 함께 이메일 알림을 관리자에게 보냅니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

SDK for Java 2.x

Amazon Rekognition을 사용하여 Amazon Simple Storage Service(Amazon S3) 버킷에 있는 이미지에서 범주별로 객체를 식별하기 위해 Amazon Rekognition을 사용하여 앱을 만드는 방법을 보여줍니다. 이 앱은 Amazon Simple Email Service(Amazon SES)를 사용하여 결과와 함께 이메일 알림을 관리자에게 보냅니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

SDK for JavaScript (v3)

AWS SDK for JavaScript로 Amazon Rekognition을 사용하여 Amazon Simple Storage Service(Amazon S3) 버킷에 있는 이미지에서 범주별로 객체를 식별하기 위해 Amazon Rekognition을 사용하여 앱을 만드는 방법을 보여줍니다. 이 앱은 Amazon Simple Email Service(Amazon SES)를 사용하여 결과와 함께 이메일 알림을 관리자에게 보냅니다.

다음 작업을 수행하는 방법에 대해 알아보세요.

- Amazon Cognito를 사용하여 인증되지 않은 사용자를 만듭니다.
- Amazon Rekognition을 사용하여 객체용 이미지를 분석합니다.
- Amazon SES 이메일 주소를 확인합니다.
- Amazon SES를 사용하여 이메일 알림을 전송합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

SDK for Kotlin

Amazon Rekognition Kotlin API를 사용하여 Amazon Simple Storage Service(Amazon S3) 버킷에 있는 이미지에서 범주별로 객체를 식별하기 위해 Amazon Rekognition을 사용하여 앱을 만드

는 방법을 보여줍니다. 이 앱은 Amazon Simple Email Service(Amazon SES)를 사용하여 결과와 함께 이메일 알림을 관리자에게 보냅니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

SDK for Python(Boto3)

다음은 수행할 수 있도록 해주는 웹 애플리케이션을 만들기 위해 AWS SDK for Python (Boto3)을 사용하는 방법을 보여줍니다.

- 사진을 Amazon Simple Storage Service(Amazon S3) 버킷에 업로드합니다.
- Amazon Rekognition을 사용하여 사진을 분석하고 레이블을 지정합니다.
- Amazon Simple Email Service(Amazon SES)를 사용하여 이미지 분석에 대한 이메일 보고서를 보냅니다.

이 예제에는 두 가지 주요 구성 요소가 포함되어 있습니다. 바로 JavaScript로 작성되고 React로 빌드된 웹 페이지와 Python으로 작성되고 Flask-RESTful로 빌드된 REST 서비스입니다.

React 웹 페이지를 사용하여 다음을 수행할 수 있습니다.

- S3 버킷에 저장된 이미지 목록을 표시합니다.
- 컴퓨터에서 S3 버킷에 이미지를 업로드합니다.
- 이미지에서 감지된 항목을 식별하는 이미지와 레이블을 표시합니다.
- S3 버킷의 모든 이미지에 대한 보고서를 받고 보고서의 이메일을 보냅니다.

웹 페이지가 REST 서비스를 호출합니다. 서비스가 다음 작업을 수행하기 위해 AWS에 요청을 전송합니다.

- S3 버킷의 이미지 목록을 가져오고 필터링합니다.
- S3 버킷에 사진을 업로드합니다.

- Amazon Rekognition을 사용하여 개별 사진을 분석하고 사진에서 감지된 항목을 식별하는 레이블 목록을 가져옵니다.
- S3 버킷의 모든 사진을 분석하고 Amazon SES를 사용하여 보고서를 이메일로 보냅니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Rekognition
- Amazon S3
- Amazon SES

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon Rekognition을 AWSSDK와 함께 사용하여 동영상의 사람 및 객체 감지

다음 코드 예제에서는 Amazon Rekognition을 사용하여 동영상에서 사람과 객체를 감지하는 방법을 보여줍니다.

Java

SDK for Java 2.x

Amazon Rekognition Java API를 사용하여 Amazon Simple Storage Service(Amazon S3) 버킷에 있는 동영상에서 얼굴과 객체를 감지하기 위한 앱을 만드는 방법을 보여줍니다. 이 앱은 Amazon Simple Email Service(Amazon SES)를 사용하여 결과와 함께 이메일 알림을 관리자에게 보냅니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK for Python(Boto3)

Amazon Rekognition을 사용하여 비동기식 감지 작업을 시작해 동영상의 얼굴, 객체 및 사람을 감지할 수 있습니다. 또한 이 예제에서는 작업이 완료되고 주제에 대한 Amazon Simple Queue Service(Amazon SQS) 대기열을 구독할 때 Amazon Simple Notification Service(Amazon SNS) 주제를 알리도록 Amazon Rekognition을 구성합니다. 대기열이 작업에 대한 메시지를 받으면 작업이 검색되고 결과가 출력됩니다.

이 예제는 GitHub에서 가장 잘 볼 수 있습니다. 전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon Simple Storage Service(Amazon S3) 버킷에서 S3 '디렉터리' 다운로드

다음 코드 예제에서는 Amazon S3 버킷 '디렉터리'의 콘텐츠를 다운로드하고 필터링하는 방법을 보여줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예제에서는 AWS SDK for Java 2.x에서 [S3TransferManager](#)를 사용하여 Amazon S3 버킷에서 '디렉터리'를 다운로드하는 방법을 보여줍니다. 또한 요청에서 [DownloadFilters](#)를 사용하는 방법을 보여줍니다.

```
/**
 * For standard buckets, S3 provides the illusion of a directory structure
 * through the use of keys. When you upload
 *   * an object to an S3 bucket, you specify a key, which is essentially the
 * "path" to the object. The key can contain
 *   * forward slashes ("/") to make it appear as if the object is stored in a
 * directory structure, but this is just a
 *   * logical representation, not an actual directory.
 * <p><pre>
 * In this example, our S3 bucket contains the following objects:
 *
 * folder1/file1.txt
 * folder1/file2.txt
 * folder1/file3.txt
 * folder2/file1.txt
 * folder2/file2.txt
 * folder2/file3.txt
 * folder3/file1.txt
 * folder3/file2.txt
 * folder3/file3.txt
 *
 * When method `downloadS3Directories` is invoked with
 * `destinationPathURI` set to `/test`, the downloaded
 * directory looks like:
 *
 * |- test
 *   |- folder1
 *     |- file1.txt
 *     |- file2.txt
 *     |- file3.txt
 *   |- folder3
 *     |- file1.txt
 *     |- file2.txt
 *     |- file3.txt
 * </pre>
 *
 * @param transferManager An S3TransferManager instance.
 * @param destinationPathURI local directory to hold the downloaded S3
 * 'directories' and files.
```

```

    * @param bucketName      The S3 bucket that contains the 'directories' to
download.
    * @return The number of objects (files, in this case) that were downloaded.
    */
    public Integer downloadS3Directories(S3TransferManager transferManager,
                                         URI destinationPathURI, String
bucketName) {

        // Define the filters for which 'directories' we want to download.
        DownloadFilter folder1Filter = (S3Object s3Object) ->
s3Object.key().startsWith("folder1/");
        DownloadFilter folder3Filter = (S3Object s3Object) ->
s3Object.key().startsWith("folder3/");
        DownloadFilter folderFilter = s3Object ->
folder1Filter.or(folder3Filter).test(s3Object);

        DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                                .destination(Paths.get(destinationPathURI))
                                .bucket(bucketName)
                                .filter(folderFilter)
                                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

        Integer numFilesInFolder1 =
Paths.get(destinationPathURI).resolve("folder1").toFile().list().length;
        Integer numFilesInFolder3 =
Paths.get(destinationPathURI).resolve("folder3").toFile().list().length;

        try {
            assert numFilesInFolder1 == 3;
            assert numFilesInFolder3 == 3;
            assert !
Paths.get(destinationPathURI).resolve("folder2").toFile().exists(); // `folder2`
was not downloaded.
        } catch (AssertionError e) {
            logger.error("An assertion failed.");
        }

        completedDirectoryDownload.failedTransfers()
            .forEach(fail -> logger.warn("Object failed to transfer  [{ }]",
fail.exception().getMessage()));
        return numFilesInFolder1 + numFilesInFolder3;
    }

```

```
}
```

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon Simple Storage Service(S3) 버킷 내의 모든 객체를 로컬 디렉터리로 다운로드

다음 코드 예제는 Amazon Simple Storage Service(S3) 버킷 내의 모든 객체를 로컬 디렉터리로 다운로드하는 방법을 보여줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

[S3TransferManager](#)를 사용하여 동일한 S3 버킷 내에 있는 [모든 S3 객체를 다운로드](#)합니다. [파일 전체](#)를 보고 [테스트](#)합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
```

```
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;

public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
    URI destinationPathURI, String bucketName) {
    DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
    .destination(Paths.get(destinationPathURI))
    .bucket(bucketName)
    .build());
    CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

    completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryDownload.failedTransfers().size();
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [DownloadDirectory](#)를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon S3 객체에서 알 수 없는 크기의 스트림 다운로드

다음 코드 예제에서는 Amazon S3 객체에서 알 수 없는 크기의 스트림을 다운로드하는 방법을 보여줍니다.

Swift

SDK for Swift

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.


```
import ArgumentParser
import AWSClientRuntime
import AWSS3
import Foundation
import Smithy
import SmithyHTTPAPI
import SmithyStreams

/// Download a file from the specified bucket.
///
/// - Parameters:
///   - bucket: The Amazon S3 bucket name to get the file from.
///   - key: The name (or path) of the file to download from the bucket.
///   - destPath: The pathname on the local filesystem at which to store
///     the downloaded file.
func downloadFile(bucket: String, key: String, destPath: String?) async
throws {
    let fileURL: URL

    // If no destination path was provided, use the key as the name to use
    // for the file in the downloads folder.

    if destPath == nil {
        do {
            try fileURL = FileManager.default.url(
                for: .downloadsDirectory,
                in: .userDomainMask,
                appropriateFor: URL(string: key),
                create: true
            ).appendingPathComponent(key)
        } catch {
            throw TransferError.directoryError
        }
    } else {
        fileURL = URL(fileURLWithPath: destPath!)
    }

    let config = try await S3Client.S3ClientConfiguration(region: region)
    let s3Client = S3Client(config: config)

    // Create a `FileHandle` referencing the local destination. Then
    // create a `ByteStream` from that.
```

```

    FileManager.default.createFile(atPath: fileURL.path, contents: nil,
attributes: nil)
    let fileHandle = try FileHandle(forWritingTo: fileURL)

    // Download the file using `GetObject`.

    let getInput = GetObjectInput(
        bucket: bucket,
        key: key
    )

    do {
        let getOutput = try await s3Client.getObject(input: getInput)

        guard let body = getOutput.body else {
            throw TransferError.downloadError("Error: No data returned for
download")
        }

        // If the body is returned as a `Data` object, write that to the
        // file. If it's a stream, read the stream chunk by chunk,
        // appending each chunk to the destination file.

        switch body {
        case .data:
            guard let data = try await body.readData() else {
                throw TransferError.downloadError("Download error")
            }

            // Write the `Data` to the file.

            do {
                try data.write(to: fileURL)
            } catch {
                throw TransferError.writeError
            }
            break

        case .stream(let stream as ReadableStream):
            while (true) {
                let chunk = try await stream.readAsync(upToCount: 5 * 1024 *
1024)

                guard let chunk = chunk else {

```

```
        break
    }

    // Write the chunk to the destination file.

    do {
        try fileHandle.write(contentsOf: chunk)
    } catch {
        throw TransferError.writeError
    }
}

break
default:
    throw TransferError.downloadError("Received data is unknown
object type")
}
} catch {
    throw TransferError.downloadError("Error downloading the file:
\\(error)")
}

print("File downloaded to \\(fileURL.path).")
}
```

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#) 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 다중 리전 액세스 포인트에서 Amazon S3 객체 생성

다음 코드 예제에서는 다중 리전 액세스 포인트에서 객체를 가져오는 방법을 보여줍니다.

Kotlin

SDK for Kotlin

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

비대칭 Sigv4(Sigv4a) 서명 알고리즘을 사용하도록 S3 클라이언트를 구성합니다.

```
suspend fun createS3Client(): S3Client {  
    // Configure your S3Client to use the Asymmetric SigV4 (SigV4a)  
    signing algorithm.  
    val sigV4aScheme = SigV4AsymmetricAuthScheme(DefaultAwsSigner)  
    val s3 = S3Client.fromEnvironment {  
        authSchemes = listOf(sigV4aScheme)  
    }  
    return s3  
}
```

버킷 이름 대신 다중 리전 액세스 포인트 ARN을 사용하여 객체를 가져옵니다.

```
suspend fun getObjectFromMrap(  
    s3: S3Client,  
    mrapArn: String,  
    keyName: String,  
): String? {  
    val request = GetObjectRequest {  
        bucket = mrapArn // Use the ARN instead of the bucket name for object  
        operations.  
        key = keyName  
    }  
  
    var stringObj: String? = null  
    s3.getObject(request) { resp ->  
        stringObj = resp.body?.decodeToString()  
        if (stringObj != null) {  
            println("Successfully read $keyName from $mrapArn")  
        }  
    }  
}
```

```
    }
    return jsonObj
}
```

- 자세한 내용은 [AWS SDK for Kotlin 개발자 안내서](#)를 참조하세요.
- API 세부 정보는 AWS SDK for Kotlin API 참조의 [GetObject](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하고 If-Modified-Since 헤더를 지정하여 Amazon S3 버킷에서 객체 가져오기

다음 코드 예제는 S3 버킷이 마지막 검색 시간 이후 수정되지 않았을 때에 한하여 해당 버킷 내의 객체에서 데이터를 읽는 방법을 보여줍니다.

Rust

SDK for Rust

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
use aws_sdk_s3::{
    error::SdkError,
    primitives::{ByteStream, DateTime, DateTimeFormat},
    Client,
};

use s3_code_examples::error::S3ExampleError;
use tracing::{error, warn};

const KEY: &str = "key";
const BODY: &str = "Hello, world!";

/// Demonstrate how `if-modified-since` reports that matching objects haven't
```

```
/// changed.
///
/// # Steps
/// - Create a bucket.
/// - Put an object in the bucket.
/// - Get the bucket headers.
/// - Get the bucket headers again but only if modified.
/// - Delete the bucket.
#[tokio::main]
async fn main() -> Result<(), S3ExampleError> {
    tracing_subscriber::fmt::init();

    // Get a new UUID to use when creating a unique bucket name.
    let uuid = uuid::Uuid::new_v4();

    // Load the AWS configuration from the environment.
    let client = Client::new(&aws_config::load_from_env().await);

    // Generate a unique bucket name using the previously generated UUID.
    // Then create a new bucket with that name.
    let bucket_name = format!("if-modified-since-{{uuid}}");
    client
        .create_bucket()
        .bucket(bucket_name.clone())
        .send()
        .await?;

    // Create a new object in the bucket whose name is `KEY` and whose
    // contents are `BODY`.
    let put_object_output = client
        .put_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .body(ByteStream::from_static(BODY.as_bytes()))
        .send()
        .await;

    // If the `PutObject` succeeded, get the eTag string from it. Otherwise,
    // report an error and return an empty string.
    let e_tag_1 = match put_object_output {
        Ok(put_object) => put_object.e_tag.unwrap(),
        Err(err) => {
            error!("{err:?}");
            String::new()
        }
    };
}
```

```

    }
};

// Request the object's headers.
let head_object_output = client
    .head_object()
    .bucket(bucket_name.as_str())
    .key(KEY)
    .send()
    .await;

// If the `HeadObject` request succeeded, create a tuple containing the
// values of the headers `last-modified` and `etag`. If the request
// failed, return the error in a tuple instead.
let (last_modified, e_tag_2) = match head_object_output {
    Ok(head_object) => (
        Ok(head_object.last_modified().cloned().unwrap()),
        head_object.e_tag.unwrap(),
    ),
    Err(err) => (Err(err), String::new()),
};

warn!("last modified: {last_modified:?}");
assert_eq!(
    e_tag_1, e_tag_2,
    "PutObject and first GetObject had differing eTags"
);

println!("First value of last_modified: {last_modified:?}");
println!("First tag: {}\n", e_tag_1);

// Send a second `HeadObject` request. This time, the `if_modified_since`
// option is specified, giving the `last_modified` value returned by the
// first call to `HeadObject`.
//
// Since the object hasn't been changed, and there are no other objects in
// the bucket, there should be no matching objects.

let head_object_output = client
    .head_object()
    .bucket(bucket_name.as_str())
    .key(KEY)
    .if_modified_since(last_modified.unwrap())
    .send()

```

```

        .await;

// If the `HeadObject` request succeeded, the result is a tuple containing
// the `last_modified` and `e_tag_1` properties. This is _not_ the expected
// result.
//
// The _expected_ result of the second call to `HeadObject` is an
// `SdkError::ServiceError` containing the HTTP error response. If that's
// the case and the HTTP status is 304 (not modified), the output is a
// tuple containing the values of the HTTP `last-modified` and `etag`
// headers.
//
// If any other HTTP error occurred, the error is returned as an
// `SdkError::ServiceError`.

let (last_modified, e_tag_2) = match head_object_output {
    Ok(head_object) => (
        Ok(head_object.last_modified().cloned().unwrap()),
        head_object.e_tag.unwrap(),
    ),
    Err(err) => match err {
        SdkError::ServiceError(err) => {
            // Get the raw HTTP response. If its status is 304, the
            // object has not changed. This is the expected code path.
            let http = err.raw();
            match http.status().as_u16() {
                // If the HTTP status is 304: Not Modified, return a
                // tuple containing the values of the HTTP
                // `last-modified` and `etag` headers.
                304 => (
                    Ok(DateTime::from_str(
                        http.headers().get("last-modified").unwrap(),
                        DateTimeFormat::HttpDate,
                    )
                    .unwrap()),
                    http.headers().get("etag").map(|t| t.into()).unwrap(),
                ),
                // Any other HTTP status code is returned as an
                // `SdkError::ServiceError`.
                _ => (Err(SdkError::ServiceError(err)), String::new()),
            }
        }
    }
    // Any other kind of error is returned in a tuple containing the
    // error and an empty string.

```



```

        _ => (Err(err), String::new()),
    },
};

warn!("last modified: {last_modified:?}");
assert_eq!(
    e_tag_1, e_tag_2,
    "PutObject and second HeadObject had different eTags"
);

println!("Second value of last modified: {last_modified:?}");
println!("Second tag: {}", e_tag_2);

// Clean up by deleting the object and the bucket.
client
    .delete_object()
    .bucket(bucket_name.as_str())
    .key(KEY)
    .send()
    .await?;

client
    .delete_bucket()
    .bucket(bucket_name.as_str())
    .send()
    .await?;

Ok(())
}

```

- API 세부 정보는 Rust용 AWS SDK API 참조의 [GetObject](#)를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon S3 객체 암호화 시작하기

다음 코드 예제는 Amazon S3 객체 암호화를 시작하는 방법을 보여줍니다.

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to apply client encryption to an object in an
/// Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class SSEClientEncryption
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "exampleobject.txt";
        string copyTargetKeyName = "examplecopy.txt";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USSouth1.
        IAmazonS3 client = new AmazonS3Client();

        try
        {
            // Create an encryption key.
            Aes aesEncryption = Aes.Create();
            aesEncryption.KeySize = 256;
            aesEncryption.GenerateKey();
            string base64Key = Convert.ToBase64String(aesEncryption.Key);
```

```

        // Upload the object.
        PutObjectRequest putObjectRequest = await
UploadObjectAsync(client, bucketName, keyName, base64Key);

        // Download the object and verify that its contents match what
you uploaded.
        await DownloadObjectAsync(client, bucketName, keyName, base64Key,
putObjectRequest);

        // Get object metadata and verify that the object uses AES-256
encryption.
        await GetObjectMetadataAsync(client, bucketName, keyName,
base64Key);

        // Copy both the source and target objects using server-side
encryption with
        // an encryption key.
        await CopyObjectAsync(client, bucketName, keyName,
copyTargetKeyName, aesEncryption, base64Key);
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error: {ex.Message}");
    }
}

/// <summary>
/// Uploads an object to an Amazon S3 bucket.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used to
call
/// PutObjectAsync.</param>
/// <param name="bucketName">The name of the Amazon S3 bucket to which
the
/// object will be uploaded.</param>
/// <param name="keyName">The name of the object to upload to the Amazon
S3
/// bucket.</param>
/// <param name="base64Key">The encryption key.</param>
/// <returns>The PutObjectRequest object for use by
DownloadObjectAsync.</returns>
public static async Task<PutObjectRequest> UploadObjectAsync(
    IAmazonS3 client,

```

```

        string bucketName,
        string keyName,
        string base64Key)
    {
        PutObjectRequest putObjectRequest = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = keyName,
            ContentBody = "sample text",
            ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
            ServerSideEncryptionCustomerProvidedKey = base64Key,
        };
        PutObjectResponse putObjectResponse = await
client.PutObjectAsync(putObjectRequest);
        return putObjectRequest;
    }

    /// <summary>
    /// Downloads an encrypted object from an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// GetObjectAsync.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
object
    /// is located.</param>
    /// <param name="keyName">The name of the Amazon S3 object to download.</
param>
    /// <param name="base64Key">The encryption key used to encrypt the
    /// object.</param>
    /// <param name="putObjectRequest">The PutObjectRequest used to upload
    /// the object.</param>
    public static async Task DownloadObjectAsync(
        IAmazonS3 client,
        string bucketName,
        string keyName,
        string base64Key,
        PutObjectRequest putObjectRequest)
    {
        GetObjectRequest getObjectRequest = new GetObjectRequest
        {
            BucketName = bucketName,
            Key = keyName,

```

```

        // Provide encryption information for the object stored in Amazon
S3.
        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
        ServerSideEncryptionCustomerProvidedKey = base64Key,
    };

    using (GetObjectResponse getResponse = await
client.GetObjectAsync(getObjectRequest))
        using (StreamReader reader = new
StreamReader(getResponse.ResponseStream))
        {
            string content = reader.ReadToEnd();
            if (string.Compare(putObjectRequest.ContentBody, content) == 0)
            {
                Console.WriteLine("Object content is same as we uploaded");
            }
            else
            {
                Console.WriteLine("Error...Object content is not same.");
            }

            if (getResponse.ServerSideEncryptionCustomerMethod ==
ServerSideEncryptionCustomerMethod.AES256)
            {
                Console.WriteLine("Object encryption method is AES256, same
as we set");
            }
            else
            {
                Console.WriteLine("Error...Object encryption method is not
the same as AES256 we set");
            }
        }
    }

    /// <summary>
    /// Retrieves the metadata associated with an Amazon S3 object.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used
    /// to call GetObjectMetadataAsync.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket containing
the

```

```

    /// object for which we want to retrieve metadata.</param>
    /// <param name="keyName">The name of the object for which we wish to
    /// retrieve the metadata.</param>
    /// <param name="base64Key">The encryption key associated with the
    /// object.</param>
    public static async Task GetObjectMetadataAsync(
        IAmazonS3 client,
        string bucketName,
        string keyName,
        string base64Key)
    {
        GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest
        {
            BucketName = bucketName,
            Key = keyName,

            // The object stored in Amazon S3 is encrypted, so provide the
            necessary encryption information.
            ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
            ServerSideEncryptionCustomerProvidedKey = base64Key,
        };

        GetObjectMetadataResponse getObjectMetadataResponse = await
client.GetObjectMetadataAsync(getObjectMetadataRequest);
        Console.WriteLine("The object metadata show encryption method used
is: {0}", getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
    }

    /// <summary>
    /// Copies an encrypted object from one Amazon S3 bucket to another.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// CopyObjectAsync.</param>
    /// <param name="bucketName">The Amazon S3 bucket containing the object
    /// to copy.</param>
    /// <param name="keyName">The name of the object to copy.</param>
    /// <param name="copyTargetKeyName">The Amazon S3 bucket to which the
object
    /// will be copied.</param>
    /// <param name="aesEncryption">The encryption type to use.</param>
    /// <param name="base64Key">The encryption key to use.</param>

```

```
public static async Task CopyObjectAsync(
    IAmazonS3 client,
    string bucketName,
    string keyName,
    string copyTargetKeyName,
    Aes aesEncryption,
    string base64Key)
{
    aesEncryption.GenerateKey();
    string copyBase64Key = Convert.ToBase64String(aesEncryption.Key);

    CopyObjectRequest copyRequest = new CopyObjectRequest
    {
        SourceBucket = bucketName,
        SourceKey = keyName,
        DestinationBucket = bucketName,
        DestinationKey = copyTargetKeyName,

        // Information about the source object's encryption.
        CopySourceServerSideEncryptionCustomerMethod =
        ServerSideEncryptionCustomerMethod.AES256,
        CopySourceServerSideEncryptionCustomerProvidedKey = base64Key,

        // Information about the target object's encryption.
        ServerSideEncryptionCustomerMethod =
        ServerSideEncryptionCustomerMethod.AES256,
        ServerSideEncryptionCustomerProvidedKey = copyBase64Key,
    };
    await client.CopyObjectAsync(copyRequest);
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 다음 항목을 참조하세요.
 - [CopyObject](#)
 - [GetObject](#)
 - [GetObjectMetadata](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon S3 객체 태깅 시작하기

다음 코드 예제는 Amazon S3 객체 태깅을 시작하는 방법을 보여줍니다.

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to work with tags in Amazon Simple Storage
/// Service (Amazon S3) objects.
/// </summary>
public class ObjectTag
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "newobject.txt";
        string filePath = @"*** file path ***";

        // Specify your bucket region (an example region is shown).
        RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

        var client = new AmazonS3Client(bucketRegion);
        await PutObjectsWithTagsAsync(client, bucketName, keyName, filePath);
    }
}
```



```
}

/// <summary>
/// This method uploads an object with tags. It then shows the tag
/// values, changes the tags, and shows the new tags.
/// </summary>
/// <param name="client">The Initialized Amazon S3 client object used
/// to call the methods to create and change an objects tags.</param>
/// <param name="bucketName">A string representing the name of the
/// bucket where the object will be stored.</param>
/// <param name="keyName">A string representing the key name of the
/// object to be tagged.</param>
/// <param name="filePath">The directory location and file name of the
/// object to be uploaded to the Amazon S3 bucket.</param>
public static async Task PutObjectsWithTagsAsync(IAmazonS3 client, string
bucketName, string keyName, string filePath)
{
    try
    {
        // Create an object with tags.
        var putRequest = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = keyName,
            FilePath = filePath,
            TagSet = new List<Tag>
            {
                new Tag { Key = "Keyx1", Value = "Value1" },
                new Tag { Key = "Keyx2", Value = "Value2" },
            },
        };

        PutObjectResponse response = await
client.PutObjectAsync(putRequest);

        // Now retrieve the new object's tags.
        GetObjectTaggingRequest getTagsRequest = new
GetObjectTaggingRequest()
        {
            BucketName = bucketName,
            Key = keyName,
        };
    }
}
```

```
GetObjectTaggingResponse objectTags = await
client.GetObjectTaggingAsync(getTagsRequest);

// Display the tag values.
objectTags.Tagging
    .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));

Tagging newTagSet = new Tagging()
{
    TagSet = new List<Tag>
    {
        new Tag { Key = "Key3", Value = "Value3" },
        new Tag { Key = "Key4", Value = "Value4" },
    },
};

PutObjectTaggingRequest putObjTagsRequest = new
PutObjectTaggingRequest()
{
    BucketName = bucketName,
    Key = keyName,
    Tagging = newTagSet,
};

PutObjectTaggingResponse response2 = await
client.PutObjectTaggingAsync(putObjTagsRequest);

// Retrieve the tags again and show the values.
GetObjectTaggingRequest getTagsRequest2 = new
GetObjectTaggingRequest()
{
    BucketName = bucketName,
    Key = keyName,
};

GetObjectTaggingResponse objectTags2 = await
client.GetObjectTaggingAsync(getTagsRequest2);

objectTags2.Tagging
    .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));
}
catch (AmazonS3Exception ex)
{
```

```

        Console.WriteLine(
            $"Error: '{ex.Message}'");
    }
}
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 [GetObjectTagging](#)을 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

빅 데이터 처리 클러스터 시작하기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- EC2 키 페어 생성
- 스토리지 설정 및 애플리케이션 준비
- 리소스 정리

Bash

AWS CLIBash 스크립트와 함께 사용

Note

GitHub에 더 많은 내용이 있습니다. [샘플 개발자 튜토리얼](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 배워보세요.

```

#!/bin/bash

# EMR Getting Started Tutorial Script
# This script automates the steps in the Amazon EMR Getting Started tutorial

set -euo pipefail

# Security: Set strict mode and trap errors

```

```
trap 'handle_error "Script interrupted or command failed"' ERR

# Set up logging with secure permissions
LOG_FILE="emr-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EMR Getting Started Tutorial Script"
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "${BUCKET_NAME:-}" ]; then echo "- S3 Bucket: $BUCKET_NAME"; fi
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- EMR Cluster: $CLUSTER_ID"; fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo ""
    echo "=====
    echo "CLEANUP IN PROGRESS"
    echo "=====
    echo "Starting cleanup process..."

    # Terminate EMR cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Terminating EMR cluster: $CLUSTER_ID"
        aws emr terminate-clusters --cluster-ids "$CLUSTER_ID" 2>/dev/null ||
true

        echo "Waiting for cluster to terminate..."
        aws emr wait cluster-terminated --cluster-id "$CLUSTER_ID" 2>/dev/null ||
true
        echo "Cluster terminated successfully."
    fi

    # Delete S3 bucket and contents if it exists and is not shared
```

```

    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" != "true" ];
then
    echo "Deleting S3 bucket contents: $BUCKET_NAME"
    aws s3 rm "s3://$BUCKET_NAME" --recursive 2>/dev/null || true

    echo "Deleting S3 bucket: $BUCKET_NAME"
    aws s3 rb "s3://$BUCKET_NAME" 2>/dev/null || true
fi

# Remove temporary key pair file if created by this script
if [ -f "${KEY_NAME_FILE:-}" ]; then
    rm -f "$KEY_NAME_FILE"
    echo "Removed temporary key pair file."
fi

echo "Cleanup completed."
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed"
fi

# Test AWS credentials
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)

# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null || true)

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="emr-${RANDOM_ID}"

```

```
fi
echo "Using bucket name: $BUCKET_NAME"

# Create S3 bucket with security best practices
echo "Creating S3 bucket: $BUCKET_NAME"
aws s3 mb "s3://$BUCKET_NAME" --region "${AWS_REGION:-us-east-1}" || handle_error
    "Failed to create S3 bucket"

# Tag the bucket
aws s3api put-bucket-tagging --bucket "$BUCKET_NAME" \
    --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=emr-
gs}]'

# Enable bucket versioning for safety
aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled || true

# Block public access to bucket
aws s3api put-public-access-block --bucket "$BUCKET_NAME" \
    --public-access-block-configuration \

    "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
    || true

# Enable encryption on bucket
aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }]
    }' || true

echo "S3 bucket created successfully with security best practices."

# Create PySpark script
echo "Creating PySpark script: health_violations.py"
cat > health_violations.py << 'EOL'
import argparse

from pyspark.sql import SparkSession

def calculate_red_violations(data_source, output_uri):
```

```

"""
Processes sample food establishment inspection data and queries the data to
find the top 10 establishments
with the most Red violations from 2006 to 2020.

:param data_source: The URI of your food establishment data CSV, such as
's3://emr-tutorial-bucket/food-establishment-data.csv'.
:param output_uri: The URI where output is written, such as 's3://emr-
tutorial-bucket/restaurant_violation_results'.
"""

with SparkSession.builder.appName("Calculate Red Health
Violations").getOrCreate() as spark:
    # Load the restaurant violation CSV data
    if data_source is not None:
        restaurants_df = spark.read.option("header", "true").csv(data_source)

    # Create an in-memory DataFrame to query
    restaurants_df.createOrReplaceTempView("restaurant_violations")

    # Create a DataFrame of the top 10 restaurants with the most Red
violations
    top_red_violation_restaurants = spark.sql("""SELECT name, count(*) AS
total_red_violations
FROM restaurant_violations
WHERE violation_type = 'RED'
GROUP BY name
ORDER BY total_red_violations DESC LIMIT 10""")

    # Write the results to the specified output URI
    top_red_violation_restaurants.write.option("header",
"true").mode("overwrite").csv(output_uri)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--data_source', help="The URI for you CSV restaurant data, like an S3
bucket location.")
    parser.add_argument(
        '--output_uri', help="The URI where output is saved, like an S3 bucket
location.")
    args = parser.parse_args()

    calculate_red_violations(args.data_source, args.output_uri)
EOL

```

```
# Secure the script file
chmod 600 health_violations.py

# Upload PySpark script to S3
echo "Uploading PySpark script to S3"
aws s3 cp health_violations.py "s3://$BUCKET_NAME/" --sse AES256 || handle_error
"Failed to upload PySpark script"
echo "PySpark script uploaded successfully."

# Download and prepare sample data
echo "Downloading sample data"
curl -sS -o food_establishment_data.zip "https://docs.aws.amazon.com/emr/latest/
ManagementGuide/samples/food_establishment_data.zip" || handle_error "Failed to
download sample data"

# Verify downloaded file
if [ ! -f food_establishment_data.zip ] || [ ! -s food_establishment_data.zip ];
then
    handle_error "Downloaded file is empty or missing"
fi

unzip -o food_establishment_data.zip || handle_error "Failed to unzip sample
data"
echo "Sample data downloaded and extracted successfully."

# Secure the sample data file
chmod 600 food_establishment_data.csv

# Upload sample data to S3
echo "Uploading sample data to S3"
aws s3 cp food_establishment_data.csv "s3://$BUCKET_NAME/" --sse AES256 ||
handle_error "Failed to upload sample data"
echo "Sample data uploaded successfully."

# Clean up sensitive local files
rm -f food_establishment_data.zip health_violations.py

# Create IAM default roles for EMR
echo "Creating IAM default roles for EMR"
aws emr create-default-roles 2>/dev/null || true
echo "IAM default roles created successfully."

# Check if EC2 key pair exists
```



```

echo "Checking for EC2 key pair"
KEY_PAIRS=$(aws ec2 describe-key-pairs --query "KeyPairs[*].KeyName" --output
text 2>/dev/null || true)

if [ -z "$KEY_PAIRS" ]; then
    echo "No EC2 key pairs found. Creating a new key pair..."
    KEY_NAME="emr-tutorial-key-${RANDOM_ID}"
    KEY_NAME_FILE="${KEY_NAME}.pem"
    aws ec2 create-key-pair --key-name "$KEY_NAME" \
        --tag-specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=emr-gs}]' \
        --query "KeyMaterial" --output text > "$KEY_NAME_FILE"
    chmod 400 "$KEY_NAME_FILE"
    echo "Created new key pair: $KEY_NAME"
else
    # Use the first available key pair
    KEY_NAME=$(echo "$KEY_PAIRS" | awk '{print $1}')
    echo "Using existing key pair: $KEY_NAME"
fi

# Launch EMR cluster with security best practices
echo "Launching EMR cluster with Spark"
CLUSTER_RESPONSE=$(aws emr create-cluster \
    --name "EMR Tutorial Cluster" \
    --release-label emr-6.10.0 \
    --applications Name=Spark \
    --ec2-attributes KeyName="$KEY_NAME" \
    --instance-type m5.xlarge \
    --instance-count 3 \
    --use-default-roles \
    --log-uri "s3://$BUCKET_NAME/logs/" \
    --ebs-root-volume-size 100 \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=emr-gs \
    --security-configuration "EMR-Tutorial-SecurityConfig" 2>/dev/null || true)

# Check for errors in the response
if echo "$CLUSTER_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to create EMR cluster: $CLUSTER_RESPONSE"
fi

# Extract cluster ID using jq if available, otherwise use alternative parsing
if command -v jq &> /dev/null; then
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | jq -r '.ClusterId // empty')
else

```

```
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterId"[[:space:]]*:'  
[[:space:]]*"^[^"]*" | grep -o 'j-[A-Z0-9]*' || true)  
fi  
  
if [ -z "$CLUSTER_ID" ] || [ "$CLUSTER_ID" == "null" ]; then  
    handle_error "Failed to extract cluster ID from response: $CLUSTER_RESPONSE"  
fi  
  
echo "EMR cluster created with ID: $CLUSTER_ID"  
  
# Wait for cluster to be ready  
echo "Waiting for cluster to be ready (this may take several minutes)..."  
aws emr wait cluster-running --cluster-id "$CLUSTER_ID" || handle_error "Cluster  
failed to reach running state"  
  
# Check if cluster is in WAITING state  
CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query  
"Cluster.Status.State" --output text)  
if [ "$CLUSTER_STATE" != "WAITING" ]; then  
    echo "Waiting for cluster to reach WAITING state..."  
    WAIT_COUNT=0  
    MAX_WAIT=120  
    while [ "$CLUSTER_STATE" != "WAITING" ]; do  
        if [ $WAIT_COUNT -ge $MAX_WAIT ]; then  
            handle_error "Cluster did not reach WAITING state within timeout  
period"  
        fi  
        sleep 30  
        CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --  
query "Cluster.Status.State" --output text)  
        echo "Current cluster state: $CLUSTER_STATE"  
  
        # Check for error states  
        if [[ "$CLUSTER_STATE" == "TERMINATED_WITH_ERRORS" || "$CLUSTER_STATE" ==  
"TERMINATED" ]]; then  
            handle_error "Cluster entered error state: $CLUSTER_STATE"  
        fi  
        WAIT_COUNT=$((WAIT_COUNT + 1))  
    done  
fi  
  
echo "Cluster is now in WAITING state and ready to accept work."  
  
# Submit Spark application as a step
```

```
echo "Submitting Spark application as a step"
STEP_RESPONSE=$(aws emr add-steps \
  --cluster-id "$CLUSTER_ID" \
  --steps Type=Spark,Name="Health Violations
  Analysis",ActionOnFailure=CONTINUE,Args=["s3://$BUCKET_NAME/
  health_violations.py","--data_source","s3://$BUCKET_NAME/
  food_establishment_data.csv","--output_uri","s3://$BUCKET_NAME/results/"])

# Check for errors in the response
if echo "$STEP_RESPONSE" | grep -i "error" > /dev/null; then
  handle_error "Failed to submit step: $STEP_RESPONSE"
fi

# Extract step ID using appropriate method
if command -v jq &> /dev/null; then
  STEP_ID=$(echo "$STEP_RESPONSE" | jq -r '.StepIds[0] // empty')
else
  STEP_ID=$(echo "$STEP_RESPONSE" | grep -o 's-[A-Z0-9]*' | head -1 || true)
fi

if [ -z "$STEP_ID" ] || [ "$STEP_ID" == "null" ]; then
  echo "Full step response: $STEP_RESPONSE"
  handle_error "Failed to extract valid step ID from response"
fi

echo "Step submitted with ID: $STEP_ID"

# Wait for step to complete with timeout
echo "Waiting for step to complete (this may take several minutes)..."
aws emr wait step-complete --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID" ||
  handle_error "Step failed to complete"

# Check step status
STEP_STATE=$(aws emr describe-step --cluster-id "$CLUSTER_ID" --step-id
"$STEP_ID" --query "Step.Status.State" --output text)
if [ "$STEP_STATE" != "COMPLETED" ]; then
  handle_error "Step did not complete successfully. Final state: $STEP_STATE"
fi

echo "Step completed successfully."

# View results
echo "Listing output files in S3"
```

```

aws s3 ls "s3://$BUCKET_NAME/results/" || handle_error "Failed to list output
files"

# Download results
echo "Downloading results file"
RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-[0-9]*\.csv"
| head -1 || true)
if [ -z "$RESULT_FILE" ]; then
    echo "No result file found with pattern 'part-[0-9]*.csv'. Trying to find any
    CSV file..."
    RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-.*\.csv"
| head -1 || true)
    if [ -z "$RESULT_FILE" ]; then
        echo "Listing all files in results directory:"
        aws s3 ls "s3://$BUCKET_NAME/results/"
        handle_error "No result file found in the output directory"
    fi
fi

aws s3 cp "s3://$BUCKET_NAME/results/$RESULT_FILE" ./results.csv --sse AES256 ||
handle_error "Failed to download results file"
chmod 600 ./results.csv

echo "Results downloaded to results.csv"
echo "Top 10 establishments with the most red violations:"
cat results.csv

# Display SSH connection information
echo ""
echo "To connect to the cluster via SSH, use the following command:"
echo "aws emr ssh --cluster-id $CLUSTER_ID --key-pair-file ${KEY_NAME_FILE:-./
${KEY_NAME}.pem}"

# Display summary of created resources
echo ""
echo "====="
echo "RESOURCES CREATED"
echo "====="
echo "- S3 Bucket: $BUCKET_NAME"
echo "- EMR Cluster: $CLUSTER_ID"
echo "- Results file: results.csv"
if [ -f "${KEY_NAME_FILE:-}" ]; then
    echo "- EC2 Key Pair: $KEY_NAME (saved to ${KEY_NAME_FILE})"
fi

```

```
# Perform cleanup
cleanup

echo "Script completed successfully."
```

- API 세부 정보는 AWS CLI 명령 참조의 다음 토픽을 참조하세요.
 - [AddSteps](#)
 - [Cp](#)
 - [CreateCluster](#) :
 - [CreateDefaultRoles](#)
 - [:CreateKeyPair](#)
 - [DescribeCluster](#)
 - [DescribeKeyPairs](#)
 - [DescribeStep](#)
 - [Ls](#)
 - [Mb](#)
 - [Rb](#)
 - [Rm](#)
 - [Ssh](#)
 - [TerminateClusters](#)
 - [대기](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

구성 관리 시작하기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- Amazon S3 버킷 생성
- Amazon SNS 주제 생성
- Config용 IAM 역할 생성
- Config 구성 레코더 설정

- Config 전송 채널 설정
- 구성 레코더 시작
- Config 설정 확인

Bash

AWS CLIBash 스크립트와 함께 사용

Note

GitHub에 더 많은 내용이 있습니다. [샘플 개발자 튜토리얼](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 배워보세요.

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}
```

```

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
"true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
"$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
"true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
"$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
"$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi
    fi
}

```

```

        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null ||
true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "=====
echo "CREATED RESOURCES"
    echo "=====
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "=====
}

```



```
# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
    text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1" # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --
create-bucket-configuration LocationConstraint="$AWS_REGION")
    fi
fi
```

```

    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"

    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
    'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
    "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=true"
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-`${RANDOM_ID}`"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags
    Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-`${RANDOM_ID}`"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "config.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}

```

```
]
}
EOF

echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
"$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    }
  ]
}
```

```

    },
    {
        "Effect": "Allow",
        "Action": [
            "sns:Publish"
        ],
        "Resource": "${SNS_TOPIC_ARN}"
    }
]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
"$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-records 2>/dev/
null || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"'
| head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
    "name": "${CONFIG_RECORDER_NAME}",

```

```

    "roleARN": "${ROLE_ARN}",
    "recordingMode": {
        "recordingFrequency": "CONTINUOUS"
    }
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
    "allSupported": true,
    "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags
    Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null ||
echo "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"'
| head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel

```

```

    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

```

```

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/
dev/null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --
configuration-recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "=====
echo "CLEANUP CONFIRMATION"
echo "=====
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."

```

```
cleanup_resources
echo "Cleanup completed."
else
echo "Resources will not be cleaned up. You can manually clean them up
later."
fi

echo "Script completed successfully!"
```

- API 세부 정보는 AWS CLI 명령 참조의 다음 토픽을 참조하세요.

- [AttachRolePolicy](#)
- [:CreateBucket](#)
- [CreateRole](#)
- [CreateTopic](#)
- [:DeleteBucket](#)
- [DeleteConfigurationRecorder](#)
- [DeleteDeliveryChannel](#)
- [DeleteRole](#)
- [DeleteRolePolicy](#)
- [DeleteTopic](#)
- [DescribeConfigurationRecorderStatus](#)
- [DescribeConfigurationRecorders](#)
- [DescribeDeliveryChannels](#)
- [DetachRolePolicy](#)
- [GetCallerIdentity](#)
- [PutConfigurationRecorder](#)
- [PutDeliveryChannel](#)
- [PutPublicAccessBlock](#)
- [PutRolePolicy](#)
- [Rm](#)
- [StartConfigurationRecorder](#)
- [StopConfigurationRecorder](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

문서 텍스트 추출 시작하기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- S3 버킷 생성
- S3에 문서 업로드
- 리소스 정리

Bash

AWS CLIBash 스크립트와 함께 사용

Note

GitHub에 더 많은 내용이 있습니다. [샘플 개발자 튜토리얼](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 배워보세요.

```
#!/bin/bash

# Amazon Textract Getting Started Tutorial Script
# This script demonstrates how to use Amazon Textract to analyze document text

set -euo pipefail

# Set up logging with restricted permissions
LOG_FILE="textract-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====="
echo "Amazon Textract Getting Started Tutorial"
echo "====="
echo "This script will guide you through using Amazon Textract to analyze
document text."
```

```

echo ""

# Function to check for errors in command output and exit code
check_error() {
    local exit_code=$1
    local output=$2
    local cmd=$3

    if [ $exit_code -ne 0 ] || echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output" | sed 's/\(aws_secret_access_key\|Authorization\|X-Amz-Security-Token\).*/\1=***REDACTED***/'
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    # Clean up temporary JSON files
    if [ -f "document.json" ]; then
        rm -f document.json
    fi

    if [ -f "features.json" ]; then
        rm -f features.json
    fi

    if [ -n "${DOCUMENT_NAME:-}" ] && [ -n "${BUCKET_NAME:-}" ]; then
        echo "Deleting document from S3..."
        aws s3 rm "s3://${BUCKET_NAME}/${DOCUMENT_NAME}" || echo "Failed to delete document"
    fi

    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" = "false" ]; then
        echo "Deleting S3 bucket..."
        aws s3 rb "s3://${BUCKET_NAME}" --force || echo "Failed to delete bucket"
    fi
}

# Set up trap for cleanup on exit

```

```

trap cleanup_on_error EXIT

# Verify AWS CLI is installed and configured
echo "Verifying AWS CLI configuration..."
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed."
    exit 1
fi

AWS_CONFIG_OUTPUT=$(aws configure list 2>&1)
AWS_CONFIG_STATUS=$?
if [ $AWS_CONFIG_STATUS -ne 0 ]; then
    echo "ERROR: AWS CLI is not properly configured."
    echo "$AWS_CONFIG_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization
\).*\/1=***REDACTED***/'
    exit 1
fi

# Verify AWS region is configured and supports Textract
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured. Please run 'aws configure' to set a
    default region."
    exit 1
fi

# Check if Textract is available in the configured region
echo "Checking if Amazon Textract is available in region $AWS_REGION..."
TEXTRACT_CHECK=$(aws textract help 2>&1)
TEXTRACT_CHECK_STATUS=$?
if [ $TEXTRACT_CHECK_STATUS -ne 0 ]; then
    echo "ERROR: Amazon Textract may not be available in region $AWS_REGION."
    exit 1
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null || echo "")
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"

```

```

    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="textract-`${RANDOM_ID}`"
fi
DOCUMENT_NAME="document.png"
RESOURCES_CREATED=()

# Step 1: Create S3 bucket
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Creating S3 bucket: $BUCKET_NAME"
    CREATE_BUCKET_OUTPUT=$(aws s3 mb "s3://$BUCKET_NAME" --region "$AWS_REGION"
2>&1)
    CREATE_BUCKET_STATUS=$?
    echo "$CREATE_BUCKET_OUTPUT"
    check_error $CREATE_BUCKET_STATUS "$CREATE_BUCKET_OUTPUT" "aws s3 mb s3://
$BUCKET_NAME"

    aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-textract-gs}]'

    # Apply security settings to bucket
    aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled 2>&1 || true
    aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" --server-side-
encryption-configuration '{"Rules": [{"ApplyServerSideEncryptionByDefault":
{"SSEAlgorithm": "AES256"}]}]' 2>&1 || true
    aws s3api put-bucket-acl --bucket "$BUCKET_NAME" --acl private 2>&1 || true

    RESOURCES_CREATED+=("S3 Bucket: $BUCKET_NAME")
fi

# Step 2: Check if sample document exists, if not create a simple one
if [ ! -f "$DOCUMENT_NAME" ]; then
    echo "Sample document not found. Generating a sample document..."

    # Create a simple PNG document using ImageMagick or convert
    if command -v convert &> /dev/null; then
        convert -size 400x300 xc:white -pointsize 20 -fill black -draw "text
50,50 'Sample Document'" "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
    fi
fi

```

```

        echo "Generated sample document: $DOCUMENT_NAME"
    else
        # Fallback: create a minimal valid PNG using base64
        echo "iVBORw0KGgoAAAANSUgAAAAEAAAABCAyAAAAFfcSJAAAADU1EQVR42mNk
+M9QDwAdhgGAWjR9awAAAABJRu5ErkJggg==" | base64 -d > "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Created minimal sample document: $DOCUMENT_NAME"
    fi
fi

# Step 3: Upload document to S3
echo "Uploading document to S3..."
UPLOAD_OUTPUT=$(aws s3 cp "$DOCUMENT_NAME" "s3://$BUCKET_NAME/" --sse AES256
2>&1)
UPLOAD_STATUS=$?
echo "$UPLOAD_OUTPUT"
check_error $UPLOAD_STATUS "$UPLOAD_OUTPUT" "aws s3 cp $DOCUMENT_NAME s3://
$BUCKET_NAME/"
RESOURCES_CREATED+=("S3 Object: s3://$BUCKET_NAME/$DOCUMENT_NAME")

# Step 4: Analyze document with Amazon Textract
echo "Analyzing document with Amazon Textract..."
echo "This may take a few seconds..."

# Create a JSON file for the document parameter to avoid shell escaping issues
cat > document.json << 'EOF'
{
    "S3Object": {
        "Bucket": "BUCKET_PLACEHOLDER",
        "Name": "DOCUMENT_PLACEHOLDER"
    }
}
EOF

sed -i.bak "s|BUCKET_PLACEHOLDER|$BUCKET_NAME|g; s|DOCUMENT_PLACEHOLDER|
$DOCUMENT_NAME|g" document.json
rm -f document.json.bak
chmod 600 document.json

# Create a JSON file for the feature types parameter
cat > features.json << 'EOF'
["TABLES", "FORMS", "SIGNATURES"]
EOF
chmod 600 features.json

```

```

ANALYZE_OUTPUT=$(aws textract analyze-document --document file://document.json --
feature-types file://features.json 2>&1)
ANALYZE_STATUS=$?

echo "Analysis complete."
if [ $ANALYZE_STATUS -ne 0 ]; then
    echo "ERROR: Document analysis failed"
    echo "$ANALYZE_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization\|Token
\).*\/1=***REDACTED***\/g'
    exit 1
fi

# Save the analysis results to a file with restricted permissions
echo "$ANALYZE_OUTPUT" > textract-analysis-results.json
chmod 600 textract-analysis-results.json
echo "Analysis results saved to textract-analysis-results.json"
RESOURCES_CREATED+=("Local file: textract-analysis-results.json")

# Display a summary of the analysis
echo ""
echo "======"
echo "Analysis Summary"
echo "======"
PAGES=$(echo "$ANALYZE_OUTPUT" | grep -o '"Pages": [0-9]*' | head -1 | awk
'{print $2}')
echo "Document pages: $PAGES"

BLOCKS_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType":' | wc -l)
echo "Total blocks detected: $BLOCKS_COUNT"

# Count different block types using jq if available, fallback to grep
PAGE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "PAGE"' | wc -l ||
echo 0)
LINE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "LINE"' | wc -l ||
echo 0)
WORD_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "WORD"' | wc -l ||
echo 0)
TABLE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "TABLE"' | wc -l ||
echo 0)
CELL_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "CELL"' | wc -l ||
echo 0)
KEY_VALUE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "KEY_VALUE_SET"'
| wc -l || echo 0)

```

```

SIGNATURE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "SIGNATURE"' |
wc -l || echo 0)

echo "Pages: $PAGE_COUNT"
echo "Lines of text: $LINE_COUNT"
echo "Words: $WORD_COUNT"
echo "Tables: $TABLE_COUNT"
echo "Table cells: $CELL_COUNT"
echo "Key-value pairs: $KEY_VALUE_COUNT"
echo "Signatures: $SIGNATURE_COUNT"
echo ""

# Cleanup confirmation
echo ""
echo "====="
echo "RESOURCES CREATED"
echo "====="
for resource in "${RESOURCES_CREATED[@]}"; do
    echo "- $resource"
done
echo ""
echo "====="
echo "CLEANUP CONFIRMATION"
echo "====="
echo "Cleaning up resources..."

# Delete document from S3
echo "Deleting document from S3..."
DELETE_DOC_OUTPUT=$(aws s3 rm "s3://$BUCKET_NAME/$DOCUMENT_NAME" 2>&1)
DELETE_DOC_STATUS=$?
echo "$DELETE_DOC_OUTPUT"
check_error $DELETE_DOC_STATUS "$DELETE_DOC_OUTPUT" "aws s3 rm s3://$BUCKET_NAME/
$DOCUMENT_NAME"

# Delete S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Deleting S3 bucket..."
    DELETE_BUCKET_OUTPUT=$(aws s3 rb "s3://$BUCKET_NAME" --force 2>&1)
    DELETE_BUCKET_STATUS=$?
    echo "$DELETE_BUCKET_OUTPUT"
    check_error $DELETE_BUCKET_STATUS "$DELETE_BUCKET_OUTPUT" "aws s3 rb s3://
$BUCKET_NAME --force"
fi

```

```
# Delete local JSON files
rm -f document.json features.json

echo "Cleanup complete. The analysis results file (textract-analysis-
results.json) has been kept."

echo ""
echo "=====
echo "Tutorial complete!"
echo "=====
echo "You have successfully analyzed a document using Amazon Textract."
echo "The analysis results are available in textract-analysis-results.json"
echo ""

trap - EXIT
```

- API 세부 정보는 AWS CLI 명령 참조의 다음 토픽을 참조하세요.
 - [AnalyzeDocument](#)
 - [Cp](#)
 - [도움말!](#)
 - [Mb](#)
 - [Rb](#)
 - [Rm](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

기계 학습 특성 저장소 시작하기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- IAM 권한 설정
- SageMaker 실행 역할 생성
- 특성 그룹 생성
- 리소스 정리

Bash

AWS CLIBash 스크립트와 함께 사용

Note

GitHub에 더 많은 내용이 있습니다. [샘플 개발자 튜토리얼](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 배워보세요.

```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""

# Track created resources for cleanup
CREATED_RESOURCES=()

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
```

```

wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
            --output text)
        echo "Current status: ${status}"

        if [ "$status" = "Failed" ]; then
            handle_error "Feature group ${feature_group_name} creation failed"
        fi
    done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"

        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
"$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
            ;;
        esac
    done
}

```

```

        ;;
        "IAMRole")
            echo "Detaching policies from role: $resource_name"
            aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
            aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
            echo "Deleting IAM role: $resource_name"
            aws iam delete-role --role-name "$resource_name" 2>/dev/null
            ;;
        *)
            echo "Unknown resource type: $resource_type"
            ;;
    esac
done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
    local trust_policy='{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then

```

```
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
    Key=tutorial,Value=sagemaker-featurestore 2>&1

    # Attach necessary policies
    echo "Attaching policies to role..." >&2

    # SageMaker execution policy
    local policy1_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

    if echo "$policy1_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach SageMaker policy: $policy1_result"
    fi

    # S3 access policy
    local policy2_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)

    if echo "$policy2_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach S3 policy: $policy2_result"
    fi

    # Get account ID for role ARN
    local account_id=$(aws sts get-caller-identity --query Account --output text)
    local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

    echo "Role ARN: $role_arn" >&2
    echo "Waiting 10 seconds for role to propagate..." >&2
    sleep 10

    # Return only the role ARN to stdout
    echo "$role_arn"
}
```

```
# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"

    # Validate the role ARN
    ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\\///')
    ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
    if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
        echo "Creating a new role automatically..."
        ROLE_ARN=$(create_sagemaker_role)
        if [ -z "$ROLE_ARN" ]; then
            handle_error "Failed to create SageMaker execution role"
        fi
    fi
fi

# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
```

```

else
    echo "Using region: $REGION"
fi
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
if [ "$REGION" = "us-east-1" ]; then
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" 2>&1)
else
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" \
        --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

```

```

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
    "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
    exit 1
fi
else
    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-${RANDOM_ID}"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"

CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'$S3_BUCKET_NAME'/'$PREFIX'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \

```

```

    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-${RANDOM_ID}"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "order_id", "FeatureType": "String"},
        {"FeatureName": "order_date", "FeatureType": "String"},
        {"FeatureName": "product", "FeatureType": "String"},
        {"FeatureName": "quantity", "FeatureType": "Integral"},
        {"FeatureName": "amount", "FeatureType": "Fractional"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create orders feature group: $ORDERS_RESPONSE"
    cleanup_resources
    exit 1

```



```
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
  --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
  --record '[
    {"FeatureName": "customer_id", "ValueAsString": "573291"},
    {"FeatureName": "name", "ValueAsString": "John Doe"},
    {"FeatureName": "age", "ValueAsString": "35"},
    {"FeatureName": "address", "ValueAsString": "123 Main St"},
    {"FeatureName": "membership_type", "ValueAsString": "premium"},
    {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
  ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
  echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
  cleanup_resources
  exit 1
fi

echo "$CUSTOMER1_RESPONSE"

CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
  --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
  --record '[
    {"FeatureName": "customer_id", "ValueAsString": "109382"},
    {"FeatureName": "name", "ValueAsString": "Jane Smith"},
    {"FeatureName": "age", "ValueAsString": "28"},
    {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
    {"FeatureName": "membership_type", "ValueAsString": "standard"},
    {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
  ]' 2>&1)
```

```
if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
        {"FeatureName": "product", "ValueAsString": "Laptop"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "1299.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
```

```

        cleanup_resources
        exit 1
    fi

    echo "$ORDER2_RESPONSE"

    # Retrieve records from feature groups
    echo "Retrieving records from feature groups..."

    # Get a single customer record
    echo "Getting customer record with ID 573291:"
    CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
        --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
        --record-identifier-value-as-string "573291" 2>&1)

    if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
        echo "Failed to get customer record: $CUSTOMER_RECORD"
        cleanup_resources
        exit 1
    fi

    echo "$CUSTOMER_RECORD"

    # Get multiple records using batch-get-record
    echo "Getting multiple records using batch-get-record:"
    BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
        --identifiers '[
            {
                "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
                "RecordIdentifiersValueAsString": ["573291", "109382"]
            },
            {
                "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
                "RecordIdentifiersValueAsString": ["573291", "109382"]
            }
        ]' 2>&1)

    if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS"
    | grep -i "Records" > /dev/null; then
        echo "Failed to get batch records: $BATCH_RECORDS"
        cleanup_resources
        exit 1
    fi

```

```

echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi

echo "$FEATURE_GROUPS"

# Display summary of created resources
echo ""
echo "=====
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "=====
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
fi
echo ""
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "=====
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else

```

```

echo "=====
echo "CLEANUP CONFIRMATION"
echo "=====
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Skipping cleanup. Resources will remain in your account."
    echo "To clean up later, delete the following resources:"
    echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
$ORDERS_FEATURE_GROUP_NAME"
    echo "- S3 Bucket: $S3_BUCKET_NAME"
    if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
        echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o
'IAMRole:[^[:space:]]*' | cut -d: -f2)"
    fi
    echo ""
    echo "Estimated ongoing cost: ~$0.01 per month for online store"
fi
fi

echo "Script completed at $(date)"

```

- API 세부 정보는 AWS CLI 명령 참조의 다음 토픽을 참조하세요.

- [AttachRolePolicy](#)
- [:CreateBucket](#)
- [CreateFeatureGroup](#)
- [CreateRole](#)
- [:DeleteBucket](#)
- [DeleteFeatureGroup](#)
- [DeleteRole](#)
- [DescribeFeatureGroup](#)
- [DetachRolePolicy](#)
- [GetCallerIdentity](#)
- [GetRole](#)

- [ListFeatureGroups](#)
- [PutPublicAccessBlock](#)
- [Rm](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

객체 스토리지 시작하기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 첫 번째 S3 버킷 생성
- 객체 업로드
- 버전 관리 사용
- 기본 암호화 구성
- 버킷에 태그 추가
- 객체 및 버전 나열
- 리소스 정리

Bash

AWS CLIBash 스크립트와 함께 사용

Note

GitHub에 더 많은 내용이 있습니다. [샘플 개발자 튜토리얼](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 배워보세요.

```
#!/bin/bash
# S3 Getting Started - Create a bucket, upload and download objects, copy to a
# folder prefix, enable versioning, configure encryption and public access
# blocking, tag the bucket, list objects and versions, and clean up.

set -eE
set -o pipefail
```

```

# =====
# Prerequisites check
# =====

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
"$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured. Run 'aws configure' or set
AWS_DEFAULT_REGION."
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &>/dev/null; then
    echo "ERROR: AWS credentials not configured or invalid. Run 'aws configure'."
    exit 1
fi

# =====
# Setup: logging, temp directory, resource tracking
# =====

UNIQUE_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null || true)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="s3api-${UNIQUE_ID}"
fi

TEMP_DIR=$(mktemp -d)
trap 'rm -rf "$TEMP_DIR"' EXIT
LOG_FILE="$TEMP_DIR/s3-gettingstarted.log"
CREATED_RESOURCES=()

exec > >(tee -a "$LOG_FILE") 2>&1

```

```

echo "=====
echo "S3 Getting Started"
echo "=====
echo "Bucket name: ${BUCKET_NAME}"
echo "Temp directory: ${TEMP_DIR}"
echo "Log file: ${LOG_FILE}"
echo ""

# =====
# Helper functions
# =====

get_region() {
    echo "${AWS_REGION:-${AWS_DEFAULT_REGION:-${CONFIGURED_REGION}}}"
}

delete_object_versions() {
    local bucket=$1
    local query=$2

    local versions
    versions=$(aws s3api list-object-versions \
        --bucket "$bucket" \
        --query "$query" \
        --output json 2>&1) || return 0

    if [ -z "$versions" ] || [ "$versions" = "null" ] || [ "$versions" = "[]" ];
    then
        return 0
    fi

    echo "$versions" | jq -r '.[ ] | "\(.Key)\t\(.VersionId)"' 2>/dev/null | while
IFS=$'\t' read -r key version_id; do
        if [ -n "$key" ] && [ "$key" != "null" ]; then
            aws s3api delete-object --bucket "$bucket" --key "$key" --version-id
"$version_id" >/dev/null 2>&1 || true
        fi
    done

    return 0
}

# =====

```



```
# Error handling and cleanup functions
# =====

cleanup() {
    echo ""
    echo "=====
    echo "CLEANUP"
    echo "=====

    if [ "$BUCKET_IS_SHARED" = "false" ]; then
        echo "Deleting all object versions in bucket..."

        delete_object_versions "$BUCKET_NAME" "Versions[.
{Key:Key,VersionId:VersionId}" || true

        delete_object_versions "$BUCKET_NAME" "DeleteMarkers[.
{Key:Key,VersionId:VersionId}" || true

        echo "Deleting bucket: ${BUCKET_NAME}"
        if ! aws s3api delete-bucket --bucket "$BUCKET_NAME" 2>/dev/null; then
            echo "WARNING: Failed to delete bucket ${BUCKET_NAME}"
        fi

        # Clean up logs bucket
        LOG_TARGET_BUCKET="${BUCKET_NAME}-logs"
        if aws s3api head-bucket --bucket "$LOG_TARGET_BUCKET" 2>/dev/null; then
            echo "Deleting log bucket: ${LOG_TARGET_BUCKET}"
            if ! aws s3api delete-bucket --bucket "$LOG_TARGET_BUCKET" 2>/dev/
null; then
                echo "WARNING: Failed to delete bucket ${LOG_TARGET_BUCKET}"
            fi
        fi
    else
        echo "Keeping shared bucket: ${BUCKET_NAME}"
    fi

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    local line_number=$1
    echo ""
    echo "=====

```

```

    echo "ERROR on line ${line_number}"
    echo "======"
    echo ""
    echo "Resources created before error:"
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        for RESOURCE in "${CREATED_RESOURCES[@}"; do
            echo "  - ${RESOURCE}"
        done
    else
        echo "  (none)"
    fi
    echo ""
    echo "Attempting cleanup..."
    cleanup
    exit 1
}

trap 'handle_error "$LINENO"' ERR

# =====
# Step 1: Create a bucket
# =====

echo "Step 1: Creating bucket ${BUCKET_NAME}..."
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    REGION=$(get_region)
    if [ "$REGION" = "us-east-1" ]; then
        if ! aws s3api create-bucket --bucket "$BUCKET_NAME" >/dev/null 2>&1;
        then
            echo "ERROR: Failed to create bucket $BUCKET_NAME"
            exit 1
        fi
    else
        if ! aws s3api create-bucket \
            --bucket "$BUCKET_NAME" \
            --region "$REGION" \
            --create-bucket-configuration LocationConstraint="$REGION" >/dev/null
        2>&1; then
            echo "ERROR: Failed to create bucket $BUCKET_NAME in region $REGION"
            exit 1
        fi
    fi
    CREATED_RESOURCES+=("s3:bucket:${BUCKET_NAME}")
    echo "Bucket created."

```

```

    if ! aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging '{
            "TagSet": [
                {
                    "Key": "project",
                    "Value": "doc-smith"
                },
                {
                    "Key": "tutorial",
                    "Value": "s3-gettingstarted"
                }
            ]
        }' >/dev/null 2>&1; then
        echo "WARNING: Failed to tag bucket"
    fi
fi
echo ""

# =====
# Step 2: Upload a sample text file
# =====

echo "Step 2: Uploading a sample text file..."

SAMPLE_FILE="${TEMP_DIR}/sample.txt"
cat > "$SAMPLE_FILE" << 'EOF'
Hello, Amazon S3! This is a sample file for the getting started tutorial.
EOF

if ! aws s3api put-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    --body "$SAMPLE_FILE" \
    --server-side-encryption AES256 \
    --metadata "tutorial=s3-gettingstarted" >/dev/null 2>&1; then
    echo "ERROR: Failed to upload sample.txt"
    exit 1
fi
echo "File uploaded."
echo ""

# =====

```

```

# Step 3: Download the object
# =====

echo "Step 3: Downloading the object..."

DOWNLOAD_FILE="${TEMP_DIR}/downloaded-sample.txt"
if ! aws s3api get-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    "$DOWNLOAD_FILE" >/dev/null 2>&1; then
    echo "ERROR: Failed to download sample.txt"
    exit 1
fi
echo "Downloaded to: ${DOWNLOAD_FILE}"
echo "Contents:"
cat "$DOWNLOAD_FILE"
echo ""

# =====
# Step 4: Copy the object to a folder prefix
# =====

echo "Step 4: Copying object to a folder prefix..."

if ! aws s3api copy-object \
    --bucket "$BUCKET_NAME" \
    --copy-source "${BUCKET_NAME}/sample.txt" \
    --key "backup/sample.txt" \
    --server-side-encryption AES256 \
    --metadata-directive COPY >/dev/null 2>&1; then
    echo "ERROR: Failed to copy object to backup/sample.txt"
    exit 1
fi
echo "Object copied to backup/sample.txt."
echo ""

# =====
# Step 5: Enable versioning and upload a second version
# =====

echo "Step 5: Enabling versioning..."

if ! aws s3api put-bucket-versioning \
    --bucket "$BUCKET_NAME" \

```

```

    --versioning-configuration Status=Enabled >/dev/null 2>&1; then
    echo "ERROR: Failed to enable versioning"
    exit 1
fi
echo "Versioning enabled."

echo "Uploading a second version of sample.txt..."
cat > "$SAMPLE_FILE" << 'EOF'
Hello, Amazon S3! This is version 2 of the sample file.
EOF

if ! aws s3api put-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    --body "$SAMPLE_FILE" \
    --server-side-encryption AES256 \
    --metadata "tutorial=s3-gettingstarted,version=2" >/dev/null 2>&1; then
    echo "ERROR: Failed to upload second version of sample.txt"
    exit 1
fi
echo "Second version uploaded."
echo ""

# =====
# Step 6: Configure SSE-S3 encryption
# =====

echo "Step 6: Configuring SSE-S3 default encryption..."

if ! aws s3api put-bucket-encryption \
    --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                },
                "BucketKeyEnabled": true
            }
        ]
    }' >/dev/null 2>&1; then
    echo "ERROR: Failed to configure SSE-S3 encryption"
    exit 1
fi

```

```

echo "SSE-S3 encryption configured."
echo ""

# =====
# Step 7: Block all public access
# =====

echo "Step 7: Blocking all public access..."

if ! aws s3api put-public-access-block \
    --bucket "$BUCKET_NAME" \
    --public-access-block-configuration '{
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    }' >/dev/null 2>&1; then
    echo "ERROR: Failed to block public access"
    exit 1
fi
echo "Public access blocked."
echo ""

# =====
# Step 8: Configure bucket logging
# =====

echo "Step 8: Configuring bucket logging..."

LOG_TARGET_BUCKET="${BUCKET_NAME}-logs"
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    REGION=$(get_region)
    if [ "$REGION" = "us-east-1" ]; then
        aws s3api create-bucket --bucket "$LOG_TARGET_BUCKET" >/dev/null 2>&1 ||
        true
    else
        aws s3api create-bucket \
            --bucket "$LOG_TARGET_BUCKET" \
            --region "$REGION" \
            --create-bucket-configuration LocationConstraint="$REGION" >/dev/null
        2>&1 || true
    fi

    if ! aws s3api put-bucket-tagging \

```

```

        --bucket "$LOG_TARGET_BUCKET" \
        --tagging '{
            "TagSet": [
                {
                    "Key": "project",
                    "Value": "doc-smith"
                },
                {
                    "Key": "tutorial",
                    "Value": "s3-gettingstarted"
                }
            ]
        }' >/dev/null 2>&1; then
        echo "WARNING: Failed to tag log bucket"
    fi

    aws s3api put-bucket-acl --bucket "$LOG_TARGET_BUCKET" --acl log-delivery-
write 2>/dev/null || true

    if ! aws s3api put-bucket-logging \
        --bucket "$BUCKET_NAME" \
        --bucket-logging-status '{
            "LoggingEnabled": {
                "TargetBucket": "'$LOG_TARGET_BUCKET'",
                "TargetPrefix": "logs/"
            }
        }' >/dev/null 2>&1; then
        echo "WARNING: Failed to configure bucket logging"
    else
        echo "Bucket logging configured."
    fi
else
    echo "Skipping logging configuration for shared bucket."
fi
echo ""

# =====
# Step 9: Tag the bucket
# =====

echo "Step 9: Tagging the bucket..."

if ! aws s3api put-bucket-tagging \
    --bucket "$BUCKET_NAME" \

```

```

--tagging '{
    "TagSet": [
        {
            "Key": "project",
            "Value": "doc-smith"
        },
        {
            "Key": "tutorial",
            "Value": "s3-gettingstarted"
        },
        {
            "Key": "Environment",
            "Value": "Tutorial"
        },
        {
            "Key": "Project",
            "Value": "S3-GettingStarted"
        },
        {
            "Key": "ManagedBy",
            "Value": "Bash-Tutorial"
        }
    ]
}' >/dev/null 2>&1; then
echo "ERROR: Failed to tag bucket"
exit 1
fi
echo "Bucket tagged."

echo "Verifying tags..."
if ! aws s3api get-bucket-tagging --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to retrieve bucket tags"
fi
echo ""

# =====
# Step 10: List objects and versions
# =====

echo "Step 10: Listing objects..."

if ! aws s3api list-objects-v2 --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to list objects"
fi

```



```

echo ""

echo "Listing object versions..."

if ! aws s3api list-object-versions --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to list object versions"
fi
echo ""

# =====
# Step 11: Cleanup
# =====

echo ""
echo "=====
echo "TUTORIAL COMPLETE"
echo "=====
echo ""
echo "Resources created:"
if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
    for RESOURCE in "${CREATED_RESOURCES[@}"; do
        echo "  - ${RESOURCE}"
    done
else
    echo "  (none)"
fi
echo ""
echo "=====
echo "CLEANUP"
echo "=====
echo "Cleaning up all created resources..."
cleanup

echo ""
echo "Done."

```

- API 세부 정보는 AWS CLI 명령 참조의 다음 토픽을 참조하세요.

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)

- [GetObject](#)
- [HeadObject](#)
- [ListObjectVersions](#)
- [ListObjectsV2](#)
- [PutBucketEncryption](#)
- [PutBucketTagging](#)
- [PutBucketVersioning](#)
- [PutObject](#)
- [PutPublicAccessBlock](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

쿼리 분석 시작하기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 쿼리 결과용 S3 버킷 생성
- 데이터베이스 생성
- 테이블 생성
- 쿼리 실행
- 명명된 쿼리 생성 및 사용
- 리소스 정리

Bash

AWS CLIBash 스크립트와 함께 사용

Note

GitHub에 더 많은 내용이 있습니다. [샘플 개발자 튜토리얼](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 배워보세요.

```
#!/bin/bash

# Amazon Athena Getting Started Script
# This script demonstrates how to use Amazon Athena with AWS CLI
# It creates a database, table, runs queries, and manages named queries

set -euo pipefail

# Security: Validate AWS credentials are configured
if ! aws sts get-caller-identity &>/dev/null; then
    echo "ERROR: AWS credentials not configured or invalid"
    exit 1
fi

# Security: Restrict umask to prevent world-readable files
umask 0077

# Set up logging with restricted permissions
LOG_FILE="athena-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Athena Getting Started Tutorial..."
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${NAMED_QUERY_ID:-}" ]; then
        echo "- Named Query: $NAMED_QUERY_ID"
    fi
    if [ -n "${DATABASE_NAME:-}" ]; then
        echo "- Database: $DATABASE_NAME"
        if [ -n "${TABLE_NAME:-}" ]; then
            echo "- Table: $TABLE_NAME in $DATABASE_NAME"
        fi
    fi
    if [ -n "${S3_BUCKET:-}" ]; then
        echo "- S3 Bucket: $S3_BUCKET"
    fi
}
```

```

    echo "Exiting..."
    exit 1
}

# Security: Validate bucket name format
validate_bucket_name() {
    local bucket_name="$1"
    if [[ ! "$bucket_name" =~ ^[a-z0-9][a-z0-9.-]*[a-z0-9]$ ]] ||
    [ ${#bucket_name} -lt 3 ] || [ ${#bucket_name} -gt 63 ]; then
        return 1
    fi
    return 0
}

# Security: Validate database and table names
validate_identifier() {
    local identifier="$1"
    if [[ ! "$identifier" =~ ^[a-zA-Z][a-zA-Z0-9_]*$ ]]; then
        return 1
    fi
    return 0
}

# Security: Safely generate random identifier
if ! command -v openssl &>/dev/null; then
    RANDOM_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
else
    RANDOM_ID=$(openssl rand -hex 6)
fi

# Security: Validate random ID format
if [[ ! "$RANDOM_ID" =~ ^[a-f0-9]{12}$ ]]; then
    handle_error "Failed to generate valid random ID"
fi

# Check for shared prereq bucket with proper error handling
PREREQ_BUCKET=""
if aws cloudformation describe-stacks --stack-name tutorial-prereqs-bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null | grep -qv "^$"; then
    PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-
prereqs-bucket \
        --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --
output text 2>/dev/null)

```

```
fi

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET="athena-${RANDOM_ID}"
fi

if ! validate_bucket_name "$S3_BUCKET"; then
    handle_error "Invalid S3 bucket name: $S3_BUCKET"
fi

DATABASE_NAME="mydatabase"
TABLE_NAME="cloudfront_logs"

if ! validate_identifier "$DATABASE_NAME"; then
    handle_error "Invalid database name: $DATABASE_NAME"
fi

if ! validate_identifier "$TABLE_NAME"; then
    handle_error "Invalid table name: $TABLE_NAME"
fi

# Get the current AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No AWS region found in configuration, defaulting to $AWS_REGION"
fi

# Security: Validate region format - expanded regex for newer regions
if [[ ! "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]] && [[ ! "$AWS_REGION" =~ ^[a-z]+-[a-z]+-[0-9]{1}$ ]]; then
    echo "WARNING: Region format may be invalid: $AWS_REGION"
fi

echo "Using AWS Region: $AWS_REGION"

# Create S3 bucket for Athena query results
echo "Creating S3 bucket for Athena query results: $S3_BUCKET"
if [ "$BUCKET_IS_SHARED" = false ]; then
```

```

CREATE_BUCKET_RESULT=$(aws s3 mb "s3://$S3_BUCKET" --region "$AWS_REGION"
2>&1)
if echo "$CREATE_BUCKET_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create S3 bucket: $CREATE_BUCKET_RESULT"
fi

aws s3api put-bucket-tagging \
    --bucket "$S3_BUCKET" \
    --tagging 'TagSet=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-athena-gs}]'

# Security: Enable S3 bucket encryption with KMS validation
echo "Enabling default encryption on S3 bucket..."
if ! aws s3api put-bucket-encryption \
    --bucket "$S3_BUCKET" \
    --server-side-encryption-configuration '{
        "Rules": [{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }]
    }' 2>&1; then
    echo "Warning: Could not enable encryption on bucket"
fi

# Security: Block public access
echo "Blocking public access to S3 bucket..."
if ! aws s3api put-public-access-block \
    --bucket "$S3_BUCKET" \
    --public-access-block-configuration \

"BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
2>&1; then
    echo "Warning: Could not block public access on bucket"
fi

# Security: Enable versioning for data protection
echo "Enabling versioning on S3 bucket..."
if ! aws s3api put-bucket-versioning \
    --bucket "$S3_BUCKET" \
    --versioning-configuration Status=Enabled 2>&1; then
    echo "Warning: Could not enable versioning on bucket"
fi

```

```

    echo "S3 bucket created successfully: $S3_BUCKET"
fi

# Step 1: Create a database
echo "Step 1: Creating Athena database: $DATABASE_NAME"
CREATE_DB_RESULT=$(aws athena start-query-execution \
    --query-string "CREATE DATABASE IF NOT EXISTS $DATABASE_NAME" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$CREATE_DB_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create database: $CREATE_DB_RESULT"
fi

QUERY_ID=$(echo "$CREATE_DB_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/
null || echo "$CREATE_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from database creation response"
fi
echo "Database creation query ID: $QUERY_ID"

# Wait for database creation to complete
echo "Waiting for database creation to complete..."
WAIT_TIMEOUT=60
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Database creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
then
        handle_error "Database creation failed with status: $QUERY_STATUS"
    fi
    echo "Database creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then

```

[illegible]


```

    handle_error "Failed to create table: $CREATE_TABLE_RESULT"
fi

QUERY_ID=$(echo "$CREATE_TABLE_RESULT" | jq -r '.QueryExecutionId // empty' 2>/
dev/null || echo "$CREATE_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*' |
    cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from table creation response"
fi
echo "Table creation query ID: $QUERY_ID"

# Wait for table creation to complete
echo "Waiting for table creation to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Table creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
then
        handle_error "Table creation failed with status: $QUERY_STATUS"
    fi
    echo "Table creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Table creation timed out"
fi

# Verify the table was created
echo "Verifying table creation..."
LIST_TABLE_RESULT=$(aws athena list-table-metadata \
    --catalog-name AwsDataCatalog \
    --database-name "$DATABASE_NAME" \
    --region "$AWS_REGION" 2>&1)
if echo "$LIST_TABLE_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list tables: $LIST_TABLE_RESULT"
fi

```

```

echo "$LIST_TABLE_RESULT"

# Step 3: Query data
echo "Step 3: Running a query on the table..."
QUERY="SELECT os, COUNT(*) count
FROM $DATABASE_NAME.$TABLE_NAME
WHERE date BETWEEN date '2014-07-05' AND date '2014-08-05'
GROUP BY os"

QUERY_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to run query: $QUERY_RESULT"
fi

QUERY_ID=$(echo "$QUERY_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
|| echo "$QUERY_RESULT" | grep -o '"QueryExecutionId": "[^"]*"' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from query execution response"
fi
echo "Query execution ID: $QUERY_ID"

# Wait for query to complete
echo "Waiting for query to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Query completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
then
        handle_error "Query failed with status: $QUERY_STATUS"
    fi
    echo "Query in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

```

```
if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Query execution timed out"
fi

# Get query results
echo "Getting query results..."
RESULTS=$(aws athena get-query-results --query-execution-id "$QUERY_ID" --region
"$AWS_REGION" 2>&1)
if echo "$RESULTS" | grep -qi "error\|failed"; then
    handle_error "Failed to get query results: $RESULTS"
fi
echo "$RESULTS"

# Download results from S3
echo "Downloading query results from S3..."
S3_PATH=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
--query "QueryExecution.ResultConfiguration.OutputLocation" --output text \
--region "$AWS_REGION" 2>&1)
if echo "$S3_PATH" | grep -qi "error\|failed"; then
    handle_error "Failed to get S3 path for results: $S3_PATH"
fi

if [ -z "$S3_PATH" ] || [ "$S3_PATH" = "None" ]; then
    handle_error "S3 path for query results is empty"
fi

DOWNLOAD_RESULT=$(aws s3 cp "$S3_PATH" "./query-results.csv" 2>&1)
if echo "$DOWNLOAD_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to download query results: $DOWNLOAD_RESULT"
fi

# Security: Secure the downloaded file
chmod 600 "./query-results.csv"
echo "Query results downloaded to query-results.csv (permissions: 600)"

# Step 4: Create a named query
echo "Step 4: Creating a named query..."
NAMED_QUERY_RESULT=$(aws athena create-named-query \
--name "OS Count Query" \
--description "Count of operating systems in CloudFront logs" \
--database "$DATABASE_NAME" \
--query-string "$QUERY" \
--region "$AWS_REGION" 2>&1)
```

```

if echo "$NAMED_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create named query: $NAMED_QUERY_RESULT"
fi

NAMED_QUERY_ID=$(echo "$NAMED_QUERY_RESULT" | jq -r '.NamedQueryId // empty' 2>/dev/null || echo "$NAMED_QUERY_RESULT" | grep -o '"NamedQueryId": "[^"]*"' | cut -d'"' -f4)
if [ -z "$NAMED_QUERY_ID" ]; then
    handle_error "Failed to extract Named Query ID from response"
fi
echo "Named query created with ID: $NAMED_QUERY_ID"

# List named queries
echo "Listing named queries..."
LIST_QUERIES_RESULT=$(aws athena list-named-queries --region "$AWS_REGION" 2>&1)
if echo "$LIST_QUERIES_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list named queries: $LIST_QUERIES_RESULT"
fi
echo "$LIST_QUERIES_RESULT"

# Get the named query details
echo "Getting named query details..."
GET_QUERY_RESULT=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    --region "$AWS_REGION" 2>&1)
if echo "$GET_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to get named query: $GET_QUERY_RESULT"
fi
echo "$GET_QUERY_RESULT"

# Execute the named query
echo "Executing the named query..."
QUERY_STRING=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    --query "NamedQuery.QueryString" --output text --region "$AWS_REGION" 2>&1)
if echo "$QUERY_STRING" | grep -qi "error\|failed"; then
    handle_error "Failed to get query string: $QUERY_STRING"
fi

if [ -z "$QUERY_STRING" ] || [ "$QUERY_STRING" = "None" ]; then
    handle_error "Query string is empty"
fi

EXEC_RESULT=$(aws athena start-query-execution \

```

```

--query-string "$QUERY_STRING" \
--result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
--region "$AWS_REGION" 2>&1)

if echo "$EXEC_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to execute named query: $EXEC_RESULT"
fi

QUERY_ID=$(echo "$EXEC_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
|| echo "$EXEC_RESULT" | grep -o '"QueryExecutionId": "[^"]*"' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from named query execution response"
fi
echo "Named query execution ID: $QUERY_ID"

# Wait for named query to complete
echo "Waiting for named query execution to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Named query execution completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
then
        handle_error "Named query execution failed with status: $QUERY_STATUS"
    fi
    echo "Named query execution in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Named query execution timed out"
fi

# Summary of resources created
echo ""
echo "======"
echo "RESOURCES CREATED"
echo "======"

```

```

echo "- S3 Bucket: $S3_BUCKET"
echo "- Database: $DATABASE_NAME"
echo "- Table: $TABLE_NAME"
echo "- Named Query: $NAMED_QUERY_ID"
echo "- Query results saved to: query-results.csv"
echo "======"

# Auto-confirm cleanup
echo ""
echo "======"
echo "CLEANUP CONFIRMATION"
echo "======"
echo "Starting cleanup..."
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup..."

    # Delete named query
    echo "Deleting named query: $NAMED_QUERY_ID"
    DELETE_QUERY_RESULT=$(aws athena delete-named-query --named-query-id
"$NAMED_QUERY_ID" \
    --region "$AWS_REGION" 2>&1)
    if echo "$DELETE_QUERY_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to delete named query: $DELETE_QUERY_RESULT"
    else
        echo "Named query deleted successfully."
    fi

    # Drop table
    echo "Dropping table: $TABLE_NAME"
    DROP_TABLE_RESULT=$(aws athena start-query-execution \
    --query-string "DROP TABLE IF EXISTS $DATABASE_NAME.$TABLE_NAME" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

    if echo "$DROP_TABLE_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to drop table: $DROP_TABLE_RESULT"
    else
        QUERY_ID=$(echo "$DROP_TABLE_RESULT" | jq -r '.QueryExecutionId // empty'
2>/dev/null || echo "$DROP_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*'
| cut -d'"' -f4)
        if [ -n "$QUERY_ID" ]; then
            echo "Waiting for table deletion to complete..."

```

```

        ELAPSED=0
        while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
            QUERY_STATUS=$(aws athena get-query-execution --query-execution-
id "$QUERY_ID" \
                --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
            if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                echo "Table dropped successfully."
                break
            elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
"CANCELLED" ]; then
                echo "Warning: Table deletion failed with status:
$QUERY_STATUS"
                break
            fi
            echo "Table deletion in progress, status: $QUERY_STATUS"
            sleep 2
            ((ELAPSED+=2))
        done
    fi
fi

# Drop database
echo "Dropping database: $DATABASE_NAME"
DROP_DB_RESULT=$(aws athena start-query-execution \
    --query-string "DROP DATABASE IF EXISTS $DATABASE_NAME" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$DROP_DB_RESULT" | grep -qi "error\|failed"; then
    echo "Warning: Failed to drop database: $DROP_DB_RESULT"
else
    QUERY_ID=$(echo "$DROP_DB_RESULT" | jq -r '.QueryExecutionId // empty'
2>/dev/null || echo "$DROP_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*"' |
cut -d'"' -f4)
    if [ -n "$QUERY_ID" ]; then
        echo "Waiting for database deletion to complete..."

        ELAPSED=0
        while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
            QUERY_STATUS=$(aws athena get-query-execution --query-execution-
id "$QUERY_ID" \

```

```

        --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
        if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
            echo "Database dropped successfully."
            break
        elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
"CANCELLED" ]; then
            echo "Warning: Database deletion failed with status:
$QUERY_STATUS"
            break
        fi
        echo "Database deletion in progress, status: $QUERY_STATUS"
        sleep 2
        ((ELAPSED+=2))
    done
fi
fi

# Empty and delete S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Emptying S3 bucket: $S3_BUCKET"
    EMPTY_BUCKET_RESULT=$(aws s3 rm "s3://$S3_BUCKET" --recursive 2>&1)
    if echo "$EMPTY_BUCKET_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to empty S3 bucket: $EMPTY_BUCKET_RESULT"
    else
        echo "S3 bucket emptied successfully."
    fi

    echo "Deleting S3 bucket: $S3_BUCKET"
    DELETE_BUCKET_RESULT=$(aws s3 rb "s3://$S3_BUCKET" 2>&1)
    if echo "$DELETE_BUCKET_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to delete S3 bucket: $DELETE_BUCKET_RESULT"
    else
        echo "S3 bucket deleted successfully."
    fi
else
    echo "Skipping S3 bucket deletion (shared resource)"
fi

# Security: Remove downloaded query results
if [ -f "./query-results.csv" ]; then
    if command -v shred &>/dev/null; then
        shred -vfz -n 3 "./query-results.csv" 2>/dev/null || rm -f "./query-
results.csv"
    fi
fi

```



```
        else
            rm -f "./query-results.csv"
        fi
        echo "Query results file securely removed."
    fi

    echo "Cleanup completed."
fi

echo "Tutorial completed successfully!"
```

- API 세부 정보는 AWS CLI 명령 참조의 다음 토픽을 참조하세요.

- [Cp](#)
- [CreateNamedQuery](#)
- [DeleteNamedQuery](#)
- [GetNamedQuery](#)
- [GetQueryExecution](#)
- [GetQueryResults](#)
- [ListDatabases](#)
- [ListNamedQueries](#)
- [ListTableMetadata](#)
- [Mb](#)
- [Rb](#)
- [Rm](#)
- [StartQueryExecution](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon S3 객체 잠금 기능 작업

다음 코드 예시에는 S3 객체 잠금 기능을 사용하는 방법을 보여 줍니다.

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 객체 잠금 기능을 시연하는 대화형 시나리오를 실행합니다.

```
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace S3ObjectLockScenario;

public static class S3ObjectLockWorkflow
{
    /*
        Before running this .NET code example, set up your development environment,
        including your credentials.

        This .NET example performs the following tasks:
        1. Create test Amazon Simple Storage Service (S3) buckets with different
        lock policies.
        2. Upload sample objects to each bucket.
        3. Set some Legal Hold and Retention Periods on objects and buckets.
        4. Investigate lock policies by viewing settings or attempting to delete
        or overwrite objects.
        5. Clean up objects and buckets.
    */

    public static S3ActionsWrapper _s3ActionsWrapper = null!;
    public static IConfiguration _configuration = null!;
    private static string _resourcePrefix = null!;
```

```

private static string noLockBucketName = null!;
private static string lockEnabledBucketName = null!;
private static string retentionAfterCreationBucketName = null!;
private static List<string> bucketNames = new List<string>();
private static List<string> fileNames = new List<string>();

public static async Task Main(string[] args)
{
    // Set up dependency injection for the Amazon service.
    using var host = Host.CreateDefaultBuilder(args)
        .ConfigureLogging(logging =>
            logging.AddFilter("System", LogLevel.Debug)
                .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
        .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonS3>()
                .AddTransient<S3ActionsWrapper>()
        )
        .Build();

    _configuration = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("settings.json") // Load settings from .json file.
        .AddJsonFile("settings.local.json",
            true) // Optionally, load local settings.
        .Build();

    ConfigurationSetup();

    ServicesSetup(host);

    try
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon Simple Storage Service (S3)
Object Locking Feature Scenario.");
        Console.WriteLine(new string('-', 80));
        await Setup(true);

        await DemoActionChoices();

        Console.WriteLine(new string('-', 80));
    }
}

```

```

        Console.WriteLine("Cleaning up resources.");
        Console.WriteLine(new string('-', 80));
        await Cleanup(true);

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Amazon S3 Object Locking Scenario is complete.");
        Console.WriteLine(new string('-', 80));
    }
    catch (Exception ex)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"There was a problem: {ex.Message}");
        await Cleanup(true);
        Console.WriteLine(new string('-', 80));
    }
}

/// <summary>
/// Populate the services for use within the console application.
/// </summary>
/// <param name="host">The services host.</param>
private static void ServicesSetup(IHost host)
{
    _s3ActionsWrapper = host.Services.GetRequiredService<S3ActionsWrapper>();
}

/// <summary>
/// Any setup operations needed.
/// </summary>
public static void ConfigurationSetup()
{
    _resourcePrefix = _configuration["resourcePrefix"] ?? "dotnet-example";

    noLockBucketName = _resourcePrefix + "-no-lock";
    lockEnabledBucketName = _resourcePrefix + "-lock-enabled";
    retentionAfterCreationBucketName = _resourcePrefix + "-retention-after-creation";

    bucketNames.Add(noLockBucketName);
    bucketNames.Add(lockEnabledBucketName);
    bucketNames.Add(retentionAfterCreationBucketName);
}

// <summary>

```

```
/// Deploy necessary resources for the scenario.
/// </summary>
/// <param name="interactive">True to run as interactive.</param>
/// <returns>True if successful.</returns>
public static async Task<bool> Setup(bool interactive)
{
    Console.WriteLine(
        "\nFor this scenario, we will use the AWS SDK for .NET to create
several S3\n" +
        "buckets and files to demonstrate working with S3 locking features.
\n");

    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Press Enter when you are ready to start.");
    if (interactive)
        Console.ReadLine();

    Console.WriteLine("\nS3 buckets can be created either with or without
object lock enabled.");
    await _s3ActionsWrapper.CreateBucketWithObjectLock(noLockBucketName,
false);
    await _s3ActionsWrapper.CreateBucketWithObjectLock(lockEnabledBucketName,
true);
    await
_s3ActionsWrapper.CreateBucketWithObjectLock(retentionAfterCreationBucketName,
false);

    Console.WriteLine("Press Enter to continue.");
    if (interactive)
        Console.ReadLine();

    Console.WriteLine("\nA bucket can be configured to use object locking
with a default retention period.");
    await
_s3ActionsWrapper.ModifyBucketDefaultRetention(retentionAfterCreationBucketName,
true,
        ObjectLockRetentionMode.Governance, DateTime.UtcNow.AddDays(1));

    Console.WriteLine("Press Enter to continue.");
    if (interactive)
        Console.ReadLine();

    Console.WriteLine("\nObject lock policies can also be added to existing
buckets.");
```

```
await _s3ActionsWrapper.EnableObjectLockOnBucket(lockEnabledBucketName);

Console.WriteLine("Press Enter to continue.");
if (interactive)
    Console.ReadLine();

// Upload some files to the buckets.
Console.WriteLine("\nNow let's add some test files:");
var fileName = _configuration["exampleFileName"] ?? "exampleFile.txt";
int fileCount = 2;
// Create the file if it does not already exist.
if (!File.Exists(fileName))
{
    await using StreamWriter sw = File.CreateText(fileName);
    await sw.WriteLineAsync(
        "This is a sample file for uploading to a bucket.");
}

foreach (var bucketName in bucketNames)
{
    for (int i = 0; i < fileCount; i++)
    {
        var numberedFileName = Path.GetFileNameWithoutExtension(fileName)
+ i + Path.GetExtension(fileName);
        fileNames.Add(numberedFileName);
        await _s3ActionsWrapper.UploadFileAsync(bucketName,
numberedFileName, fileName);
    }
}
Console.WriteLine("Press Enter to continue.");
if (interactive)
    Console.ReadLine();

if (!interactive)
    return true;
Console.WriteLine("\nNow we can set some object lock policies on
individual files:");
foreach (var bucketName in bucketNames)
{
    for (int i = 0; i < fileNames.Count; i++)
    {
        // No modifications to the objects in the first bucket.
        if (bucketName != bucketNames[0])
        {
```

```

        var exampleFileName = fileNames[i];
        switch (i)
        {
            case 0:
            {
                var question =
                    $"\\nWould you like to add a legal hold to
{exampleFileName} in {bucketName}? (y/n)";
                if (GetYesNoResponse(question))
                {
                    // Set a legal hold.
                    await
_s3ActionsWrapper.ModifyObjectLegalHold(bucketName, exampleFileName,
ObjectLockLegalHoldStatus.On);

                }
                break;
            }
            case 1:
            {
                var question =
                    $"\\nWould you like to add a 1 day Governance
retention period to {exampleFileName} in {bucketName}? (y/n)" +
                    "\\nReminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.";
                if (GetYesNoResponse(question))
                {
                    // Set a Governance mode retention period for
1 day.
                    await
_s3ActionsWrapper.ModifyObjectRetentionPeriod(
                        bucketName, exampleFileName,
                        ObjectLockRetentionMode.Governance,
                        DateTime.UtcNow.AddDays(1));
                }
                break;
            }
        }
    }
}
}
}
Console.WriteLine(new string('-', 80));
return true;

```

```

    }

    /// <summary>
    /// List all of the current buckets and objects.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>The list of buckets and objects.</returns>
    public static async Task<List<S3ObjectVersion>> ListBucketsAndObjects(bool
interactive)
    {
        var allObjects = new List<S3ObjectVersion>();
        foreach (var bucketName in bucketNames)
        {
            var objectsInBucket = await
_s3ActionsWrapper.ListBucketObjectsAndVersions(bucketName);
            foreach (var objectKey in objectsInBucket.Versions)
            {
                allObjects.Add(objectKey);
            }
        }

        if (interactive)
        {
            Console.WriteLine("\nCurrent buckets and objects:\n");
            int i = 0;
            foreach (var bucketObject in allObjects)
            {
                i++;
                Console.WriteLine(
                    $"{i}: {bucketObject.Key} \n\tBucket:
{bucketObject.BucketName}\n\tVersion: {bucketObject.VersionId}");
            }
        }

        return allObjects;
    }

    /// <summary>
    /// Present the user with the demo action choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<bool> DemoActionChoices()
    {
        var choices = new string[]{

```



```

        "List all files in buckets.",
        "Attempt to delete a file.",
        "Attempt to delete a file with retention period bypass.",
        "Attempt to overwrite a file.",
        "View the object and bucket retention settings for a file.",
        "View the legal hold settings for a file.",
        "Finish the scenario."};

var choice = 0;
// Keep asking the user until they choose to move on.
while (choice != 6)
{
    Console.WriteLine(new string('-', 80));
    choice = GetChoiceResponse(
        "\nExplore the S3 locking features by selecting one of the
following choices:"
        , choices);
    Console.WriteLine(new string('-', 80));
    switch (choice)
    {
        case 0:
        {
            await ListBucketsAndObjects(true);
            break;
        }
        case 1:
        {
            Console.WriteLine("\nEnter the number of the object to
delete:");

            var allFiles = await ListBucketsAndObjects(true);
            var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
            await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, false, allFiles[fileChoice].VersionId);
            break;
        }
        case 2:
        {
            Console.WriteLine("\nEnter the number of the object to
delete:");

            var allFiles = await ListBucketsAndObjects(true);
            var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());

```

```
        await
        _s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, true, allFiles[fileChoice].VersionId);
        break;
    }
    case 3:
    {
        var allFiles = await ListBucketsAndObjects(true);
        Console.WriteLine("\nEnter the number of the object to
overwrite:");
        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
        // Create the file if it does not already exist.
        if (!File.Exists(allFiles[fileChoice].Key))
        {
            await using StreamWriter sw =
File.CreateText(allFiles[fileChoice].Key);
            await sw.WriteLineAsync(
                "This is a sample file for uploading to a
bucket.");
        }
        await
        _s3ActionsWrapper.UploadFileAsync(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, allFiles[fileChoice].Key);
        break;
    }
    case 4:
    {
        var allFiles = await ListBucketsAndObjects(true);
        Console.WriteLine("\nEnter the number of the object and
bucket to view:");
        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
        await
        _s3ActionsWrapper.GetObjectRetention(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
        await
        _s3ActionsWrapper.GetBucketObjectLockConfiguration(allFiles[fileChoice].BucketName);
        break;
    }
    case 5:
    {
        var allFiles = await ListBucketsAndObjects(true);
```

```

        Console.WriteLine("\nEnter the number of the object to
view:");

        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
        await
_s3ActionsWrapper.GetObjectLegalHold(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
        break;
    }

    }
}
return true;
}

// <summary>
/// Clean up the resources from the scenario.
/// </summary>
/// <param name="interactive">True to run as interactive.</param>
/// <returns>True if successful.</returns>
public static async Task<bool> Cleanup(bool interactive)
{
    Console.WriteLine(new string('-', 80));

    if (!interactive || GetYesNoResponse("Do you want to clean up all files
and buckets? (y/n) "))
    {
        // Remove all locks and delete all buckets and objects.
        var allFiles = await ListBucketsAndObjects(false);
        foreach (var fileInfo in allFiles)
        {
            // Check for a legal hold.
            var legalHold = await
_s3ActionsWrapper.GetObjectLegalHold(fileInfo.BucketName, fileInfo.Key);
            if (legalHold?.Status?.Value == ObjectLockLegalHoldStatus.On)
            {
                await
_s3ActionsWrapper.ModifyObjectLegalHold(fileInfo.BucketName, fileInfo.Key,
ObjectLockLegalHoldStatus.Off);
            }

            // Check for a retention period.
            var retention = await
_s3ActionsWrapper.GetObjectRetention(fileInfo.BucketName, fileInfo.Key);

```

```

        var hasRetentionPeriod = retention?.Mode ==
ObjectLockRetentionMode.Governance && retention.RetainUntilDate >
DateTime.UtcNow.Date;
        await
_s3ActionsWrapper.DeleteObjectFromBucket(fileInfo.BucketName, fileInfo.Key,
hasRetentionPeriod, fileInfo.VersionId);
    }

    foreach (var bucketName in bucketNames)
    {
        await _s3ActionsWrapper.DeleteBucketByName(bucketName);
    }

}
else
{
    Console.WriteLine(
        "Ok, we'll leave the resources intact.\n" +
        "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
    );
}

Console.WriteLine(new string('-', 80));
return true;
}

/// <summary>
/// Helper method to get a yes or no response from the user.
/// </summary>
/// <param name="question">The question string to print on the console.</
param>
/// <returns>True if the user responds with a yes.</returns>
private static bool GetYesNoResponse(string question)
{
    Console.WriteLine(question);
    var ynResponse = Console.ReadLine();
    var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
    return response;
}

/// <summary>
/// Helper method to get a choice response from the user.

```

```

    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <param name="choices">The choices to print on the console.</param>
    /// <returns>The index of the selected choice</returns>
    private static int GetChoiceResponse(string? question, string[] choices)
    {
        if (question != null)
        {
            Console.WriteLine(question);

            for (int i = 0; i < choices.Length; i++)
            {
                Console.WriteLine($"{i + 1}. {choices[i]}");
            }

            var choiceNumber = 0;
            while (choiceNumber < 1 || choiceNumber > choices.Length)
            {
                var choice = Console.ReadLine();
                Int32.TryParse(choice, out choiceNumber);
            }

            return choiceNumber - 1;
        }
    }
}

```

S3 함수의 래퍼 클래스입니다.

```

using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

namespace S3ObjectLockScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper

```

```
{
    private readonly IAmazonS3 _amazonS3;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, IConfiguration configuration)
    {
        _amazonS3 = amazonS3;
    }

    /// <summary>
    /// Create a new Amazon S3 bucket with object lock actions.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <param name="enableObjectLock">True to enable object lock on the
    bucket.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
    enableObjectLock)
    {
        Console.WriteLine($"\\tCreating bucket {bucketName} with object lock
    {enableObjectLock}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
                ObjectLockEnabledForBucket = enableObjectLock,
            };

            var response = await _amazonS3.PutBucketAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }
}
```

```
/// <summary>
/// Enable object lock on an existing bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket to modify.</param>
/// <returns>True if successful.</returns>
public async Task<bool> EnableObjectLockOnBucket(string bucketName)
{
    try
    {
        // First, enable Versioning on the bucket.
        await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
        {
            BucketName = bucketName,
            VersioningConfig = new S3BucketVersioningConfig()
            {
                EnableMfaDelete = false,
                Status = VersionStatus.Enabled
            }
        });

        var request = new PutObjectLockConfigurationRequest()
        {
            BucketName = bucketName,
            ObjectLockConfiguration = new ObjectLockConfiguration()
            {
                ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
            },
        };

        var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
        Console.WriteLine($"{bucketName}\tAdded an object lock policy to bucket
{bucketName}.");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
        return false;
    }
}

/// <summary>
```

```

    /// Set or modify a retention period on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date retention expires.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
        string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
    {
        try
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\\tError modifying retention period:
'{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Set or modify a retention period on an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to modify.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date for retention until.</param>
    /// <returns>True if successful.</returns>

```



```

public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
{
    var enabledString = enableObjectLock ? "Enabled" : "Disabled";
    var timeDifference = retainUntilDate.Subtract(DateTime.Now);
    try
    {
        // First, enable Versioning on the bucket.
        await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
        {
            BucketName = bucketName,
            VersioningConfig = new S3BucketVersioningConfig()
            {
                EnableMfaDelete = false,
                Status = VersionStatus.Enabled
            }
        });

        var request = new PutObjectLockConfigurationRequest()
        {
            BucketName = bucketName,
            ObjectLockConfiguration = new ObjectLockConfiguration()
            {
                ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                Rule = new ObjectLockRule()
                {
                    DefaultRetention = new DefaultRetention()
                    {
                        Mode = retention,
                        Days = timeDifference.Days // Can be specified in
days or years but not both.
                    }
                }
            }
        };

        var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
        Console.WriteLine($"{bucketName}\tAdded a default retention to bucket
{bucketName}.");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)

```

```

        {
            Console.WriteLine($"\\tError modifying object lock: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Get the retention period for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object retention details.</returns>
    public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectRetentionAsync(request);
            Console.WriteLine($"\\tObject retention for {objectKey} in
{bucketName}: " +
                $"\\n\\t{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}.");
            return response.Retention;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\\tUnable to fetch object lock retention:
'{ex.Message}'");
            return new ObjectLockRetention();
        }
    }

    /// <summary>
    /// Set or modify a legal hold on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="holdStatus">The On or Off status for the legal hold.</param>

```

```
/// <returns>True if successful.</returns>
public async Task<bool> ModifyObjectLegalHold(string bucketName,
    string objectKey, ObjectLockLegalHoldStatus holdStatus)
{
    try
    {
        var request = new PutObjectLegalHoldRequest()
        {
            BucketName = bucketName,
            Key = objectKey,
            LegalHold = new ObjectLockLegalHold()
            {
                Status = holdStatus
            }
        };

        var response = await _amazonS3.PutObjectLegalHoldAsync(request);
        Console.WriteLine($"\\tModified legal hold for {objectKey} in
{bucketName}.");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tError modifying legal hold: '{ex.Message}'");
        return false;
    }
}

/// <summary>
/// Get the legal hold details for an S3 object.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The object key.</param>
/// <returns>The object legal hold details.</returns>
public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
    string objectKey)
{
    try
    {
        var request = new GetObjectLegalHoldRequest()
        {
            BucketName = bucketName,
            Key = objectKey
        };
    }
}
```

```

        var response = await _amazonS3.GetObjectLegalHoldAsync(request);
        Console.WriteLine($"\\tObject legal hold for {objectKey} in
{bucketName}: " +
                        $"\\n\\tStatus: {response.LegalHold.Status}");
        return response.LegalHold;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to fetch legal hold: '{ex.Message}'");
        return new ObjectLockLegalHold();
    }
}

/// <summary>
/// Get the object lock configuration details for an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket to get details.</param>
/// <returns>The bucket's object lock configuration details.</returns>
public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
{
    try
    {
        var request = new GetObjectLockConfigurationRequest()
        {
            BucketName = bucketName
        };

        var response = await
        _amazonS3.GetObjectLockConfigurationAsync(request);
        Console.WriteLine($"\\tBucket object lock config for {bucketName} in
{bucketName}: " +
                        $"\\n\\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                        $"\\n\\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

        return response.ObjectLockConfiguration;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to fetch object lock config:
'{ex.Message}'");
    }
}

```

```

        return new ObjectLockConfiguration();
    }
}

/// <summary>
/// Upload a file from the local computer to an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to use.</param>
/// <param name="objectName">The object to upload.</param>
/// <param name="filePath">The path, including file name, of the object to
upload.</param>
/// <returns>True if success.</returns>
public async Task<bool> UploadFileAsync(string bucketName, string objectName,
string filePath)
{
    var request = new PutObjectRequest
    {
        BucketName = bucketName,
        Key = objectName,
        FilePath = filePath,
        ChecksumAlgorithm = ChecksumAlgorithm.SHA256
    };

    var response = await _amazonS3.PutObjectAsync(request);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"\\tSuccessfully uploaded {objectName} to
{bucketName}.");
        return true;
    }
    else
    {
        Console.WriteLine($"\\tCould not upload {objectName} to
{bucketName}.");
        return false;
    }
}

/// <summary>
/// List bucket objects and versions.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to use.</param>
/// <returns>The list of objects and versions.</returns>

```

```

    public async Task<ListVersionsResponse> ListBucketObjectsAndVersions(string
bucketName)
    {
        var request = new ListVersionsRequest()
        {
            BucketName = bucketName
        };

        var response = await _amazonS3.ListVersionsAsync(request);
        return response;
    }

    /// <summary>
    /// Delete an object from a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="hasRetention">True if the object has retention settings.</
param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteObjectFromBucket(string bucketName, string
objectKey, bool hasRetention, string? versionId = null)
    {
        try
        {
            var request = new DeleteObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                VersionId = versionId,
            };
            if (hasRetention)
            {
                // Set the BypassGovernanceRetention header
                // if the file has retention settings.
                request.BypassGovernanceRetention = true;
            }
            await _amazonS3.DeleteObjectAsync(request);
            Console.WriteLine(
                $"Deleted {objectKey} in {bucketName}.");
            return true;
        }
        catch (AmazonS3Exception ex)

```

```

        {
            Console.WriteLine($"\\tUnable to delete object {objectKey} in bucket
{bucketName}: " + ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Delete a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteBucketByName(string bucketName)
    {
        try
        {
            var request = new DeleteBucketRequest() { BucketName = bucketName, };
            var response = await _amazonS3.DeleteBucketAsync(request);
            Console.WriteLine($"\\tDelete for {bucketName} complete.");
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\\tUnable to delete bucket {bucketName}: " +
ex.Message);
            return false;
        }
    }
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 다음 주제를 참조하세요.

- [GetObjectLegalHold](#)
- [GetObjectLockConfiguration](#)
- [GetObjectRetention](#)
- [PutObjectLegalHold](#)
- [PutObjectLockConfiguration](#)

- [PutObjectRetention](#)

Go

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 객체 잠금 기능을 시연하는 대화형 시나리오를 실행합니다.

```
import (
    "context"
    "fmt"
    "log"
    "strings"

    "s3_object_lock/actions"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// ObjectLockScenario contains the steps to run the S3 Object Lock workflow.
type ObjectLockScenario struct {
    questioner demotools.IQuestioner
    resources  Resources
    s3Actions  *actions.S3Actions
    sdkConfig  aws.Config
}

// NewObjectLockScenario constructs a new ObjectLockScenario instance.
func NewObjectLockScenario(sdkConfig aws.Config, questioner
    demotools.IQuestioner) ObjectLockScenario {
    scenario := ObjectLockScenario{
```



```

    questioner: questioner,
    resources: Resources{},
    s3Actions: &actions.S3Actions{S3Client: s3.NewFromConfig(sdkConfig)},
    sdkConfig: sdkConfig,
}
scenario.s3Actions.S3Manager = manager.NewUploader(scenario.s3Actions.S3Client)
scenario.resources.init(scenario.s3Actions, questioner)
return scenario
}

type nameLocked struct {
    name    string
    locked  bool
}

var createInfo = []nameLocked{
    {"standard-bucket", false},
    {"lock-bucket", true},
    {"retention-bucket", false},
}

// CreateBuckets creates the S3 buckets required for the workflow.
func (scenario *ObjectLockScenario) CreateBuckets(ctx context.Context) {
    log.Println("Let's create some S3 buckets to use for this workflow.")
    success := false
    for !success {
        prefix := scenario.questioner.Ask(
            "This example creates three buckets. Enter a prefix to name your buckets\n(remember bucket names must be globally unique):")

        for _, info := range createInfo {
            log.Println(fmt.Sprintf("%s.%s", prefix, info.name))
            bucketName, err := scenario.s3Actions.CreateBucketWithLock(ctx,
                fmt.Sprintf("%s.%s", prefix, info.name), scenario.sdkConfig.Region, info.locked)
            if err != nil {
                switch err.(type) {
                case *types.BucketAlreadyExists, *types.BucketAlreadyOwnedByYou:
                    log.Printf("Couldn't create bucket %s.\n", bucketName)
                default:
                    panic(err)
                }
            }
            break
        }
    }
    scenario.resources.demoBuckets[info.name] = &DemoBucket{

```

```

        name:      bucketName,
        objectKeys: []string{},
    }
    log.Printf("Created bucket %s.\n", bucketName)
}

if len(scenario.resources.demoBuckets) < len(createInfo) {
    scenario.resources.deleteBuckets(ctx)
} else {
    success = true
}
}

log.Println("S3 buckets created.")
log.Println(strings.Repeat("-", 88))
}

// EnableLockOnBucket enables object locking on an existing bucket.
func (scenario *ObjectLockScenario) EnableLockOnBucket(ctx context.Context) {
    log.Println("\nA bucket can be configured to use object locking.")
    scenario.questioner.Ask("Press Enter to continue.")

    var err error
    bucket := scenario.resources.demoBuckets["retention-bucket"]
    err = scenario.s3Actions.EnableObjectLockOnBucket(ctx, bucket.name)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchBucket:
            log.Printf("Couldn't enable object locking on bucket %s.\n", bucket.name)
        default:
            panic(err)
        }
    } else {
        log.Printf("Object locking enabled on bucket %s.", bucket.name)
    }

    log.Println(strings.Repeat("-", 88))
}

// SetDefaultRetentionPolicy sets a default retention governance policy on a
// bucket.
func (scenario *ObjectLockScenario) SetDefaultRetentionPolicy(ctx
context.Context) {

```

```

log.Println("\nA bucket can be configured to use object locking with a default
retention period.")

bucket := scenario.resources.demoBuckets["retention-bucket"]
retentionPeriod := scenario.questioner.AskInt("Enter the default retention
period in days: ")
err := scenario.s3Actions.ModifyDefaultBucketRetention(ctx,
bucket.name, types.ObjectLockEnabledEnabled, int32(retentionPeriod),
types.ObjectLockRetentionModeGovernance)
if err != nil {
    switch err.(type) {
    case *types.NoSuchBucket:
        log.Printf("Couldn't configure a default retention period on bucket %s.\n",
bucket.name)
    default:
        panic(err)
    }
} else {
    log.Printf("Default retention policy set on bucket %s with %d day retention
period.", bucket.name, retentionPeriod)
    bucket.retentionEnabled = true
}

log.Println(strings.Repeat("-", 88))
}

// UploadTestObjects uploads test objects to the S3 buckets.
func (scenario *ObjectLockScenario) UploadTestObjects(ctx context.Context) {
    log.Println("Uploading test objects to S3 buckets.")

    for _, info := range createInfo {
        bucket := scenario.resources.demoBuckets[info.name]
        for i := 0; i < 2; i++ {
            key, err := scenario.s3Actions.UploadObject(ctx, bucket.name,
fmt.Sprintf("example-%d", i),
fmt.Sprintf("Example object content #%d in bucket %s.", i, bucket.name))
            if err != nil {
                switch err.(type) {
                case *types.NoSuchBucket:
                    log.Printf("Couldn't upload %s to bucket %s.\n", key, bucket.name)
                default:
                    panic(err)
                }
            } else {

```

```

    log.Printf("Uploaded %s to bucket %s.\n", key, bucket.name)
    bucket.objectKeys = append(bucket.objectKeys, key)
}
}
}

scenario.questioner.Ask("Test objects uploaded. Press Enter to continue.")
log.Println(strings.Repeat("-", 88))
}

// SetObjectLockConfigurations sets object lock configurations on the test
objects.
func (scenario *ObjectLockScenario) SetObjectLockConfigurations(ctx
context.Context) {
log.Println("Now let's set object lock configurations on individual objects.")

buckets := []*DemoBucket{scenario.resources.demoBuckets["lock-bucket"],
scenario.resources.demoBuckets["retention-bucket"]}
for _, bucket := range buckets {
for index, objKey := range bucket.objectKeys {
switch index {
case 0:
if scenario.questioner.AskBool(fmt.Sprintf("\nDo you want to add a legal hold
to %s in %s (y/n)? ", objKey, bucket.name), "y") {
err := scenario.s3Actions.PutObjectLegalHold(ctx, bucket.name, objKey, "",
types.ObjectLockLegalHoldStatusOn)
if err != nil {
switch err.(type) {
case *types.NoSuchKey:
log.Printf("Couldn't set legal hold on %s.\n", objKey)
default:
panic(err)
}
} else {
log.Printf("Legal hold set on %s.\n", objKey)
}
}
case 1:
q := fmt.Sprintf("\nDo you want to add a 1 day Governance retention period to
%s in %s?\n"+
"Reminder: Only a user with the s3:BypassGovernanceRetention permission is
able to delete this object\n"+
"or its bucket until the retention period has expired. (y/n) ", objKey,
bucket.name)

```

```

    if scenario.questioner.AskBool(q, "y") {
        err := scenario.s3Actions.PutObjectRetention(ctx, bucket.name, objKey,
types.ObjectLockRetentionModeGovernance, 1)
        if err != nil {
            switch err.(type) {
            case *types.NoSuchKey:
                log.Printf("Couldn't set retention period on %s in %s.\n", objKey,
bucket.name)
            default:
                panic(err)
            }
        } else {
            log.Printf("Retention period set to 1 for %s.", objKey)
            bucket.retentionEnabled = true
        }
    }
}
}
log.Println(strings.Repeat("-", 88))
}

const (
    ListAll = iota
    DeleteObject
    DeleteRetentionObject
    OverwriteObject
    ViewRetention
    ViewLegalHold
    Finish
)

// InteractWithObjects allows the user to interact with the objects and test the
// object lock configurations.
func (scenario *ObjectLockScenario) InteractWithObjects(ctx context.Context) {
    log.Println("Now you can interact with the objects to explore the object lock
configurations.")
    interactiveChoices := []string{
        "List all objects and buckets.",
        "Attempt to delete an object.",
        "Attempt to delete an object with retention period bypass.",
        "Attempt to overwrite a file.",
        "View the retention settings for an object.",
        "View the legal hold settings for an object.",
    }

```

```

"Finish the workflow."}

choice := ListAll
for choice != Finish {
    objList := scenario.GetAllObjects(ctx)
    objChoices := scenario.makeObjectChoiceList(objList)
    choice = scenario.questioner.AskChoice("Choose an action from the menu:\n",
interactiveChoices)
    switch choice {
    case ListAll:
        log.Println("The current objects in the example buckets are:")
        for _, objChoice := range objChoices {
            log.Println("\t", objChoice)
        }
    case DeleteObject, DeleteRetentionObject:
        objChoice := scenario.questioner.AskChoice("Enter the number of the object to
delete:\n", objChoices)
        obj := objList[objChoice]
        deleted, err := scenario.s3Actions.DeleteObject(ctx, obj.bucket, obj.key,
obj.versionId, choice == DeleteRetentionObject)
        if err != nil {
            switch err.(type) {
            case *types.NoSuchKey:
                log.Println("Nothing to delete.")
            default:
                panic(err)
            }
        } else if deleted {
            log.Printf("Object %s deleted.\n", obj.key)
        }
    case OverwriteObject:
        objChoice := scenario.questioner.AskChoice("Enter the number of the object to
overwrite:\n", objChoices)
        obj := objList[objChoice]
        _, err := scenario.s3Actions.UploadObject(ctx, obj.bucket, obj.key,
fmt.Sprintf("New content in object %s.", obj.key))
        if err != nil {
            switch err.(type) {
            case *types.NoSuchBucket:
                log.Println("Couldn't upload to nonexistent bucket.")
            default:
                panic(err)
            }
        } else {

```

```

    log.Printf("Uploaded new content to object %s.\n", obj.key)
}
case ViewRetention:
    objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
    obj := objList[objChoice]
    retention, err := scenario.s3Actions.GetObjectRetention(ctx, obj.bucket,
obj.key)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchKey:
            log.Printf("Can't get retention configuration for %s.\n", obj.key)
        default:
            panic(err)
        }
    } else if retention != nil {
        log.Printf("Object %s has retention mode %s until %v.\n", obj.key,
retention.Mode, retention.RetainUntilDate)
    } else {
        log.Printf("Object %s does not have object retention configured.\n", obj.key)
    }
case ViewLegalHold:
    objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
    obj := objList[objChoice]
    legalHold, err := scenario.s3Actions.GetObjectLegalHold(ctx, obj.bucket,
obj.key, obj.versionId)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchKey:
            log.Printf("Can't get legal hold configuration for %s.\n", obj.key)
        default:
            panic(err)
        }
    } else if legalHold != nil {
        log.Printf("Object %s has legal hold %v.", obj.key, *legalHold)
    } else {
        log.Printf("Object %s does not have legal hold configured.", obj.key)
    }
case Finish:
    log.Println("Let's clean up.")
}
log.Println(strings.Repeat("-", 88))
}

```

```

}

type BucketKeyVersionId struct {
    bucket    string
    key       string
    versionId string
}

// GetAllObjects gets the object versions in the example S3 buckets and returns
// them in a flattened list.
func (scenario *ObjectLockScenario) GetAllObjects(ctx context.Context)
[]BucketKeyVersionId {
    var objectList []BucketKeyVersionId
    for _, info := range createInfo {
        bucket := scenario.resources.demoBuckets[info.name]
        versions, err := scenario.s3Actions.ListObjectVersions(ctx, bucket.name)
        if err != nil {
            switch err.(type) {
            case *types.NoSuchBucket:
                log.Printf("Couldn't get object versions for %s.\n", bucket.name)
            default:
                panic(err)
            }
        } else {
            for _, version := range versions {
                objectList = append(objectList,
                    BucketKeyVersionId{bucket: bucket.name, key: *version.Key, versionId:
*version.VersionId})
            }
        }
    }
    return objectList
}

// makeObjectChoiceList makes the object version list into a list of strings that
// are displayed
// as choices.
func (scenario *ObjectLockScenario) makeObjectChoiceList(bucketObjects
[]BucketKeyVersionId) []string {
    choices := make([]string, len(bucketObjects))
    for i := 0; i < len(bucketObjects); i++ {
        choices[i] = fmt.Sprintf("%s in %s with VersionId %s.",
            bucketObjects[i].key, bucketObjects[i].bucket, bucketObjects[i].versionId)
    }
}

```



```

    return choices
}

// Run runs the S3 Object Lock scenario.
func (scenario *ObjectLockScenario) Run(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
            _, isMock := scenario.questioner.(*demotools.MockQuestioner)
            if isMock || scenario.questioner.AskBool("Do you want to see the full error
message (y/n)?", "y") {
                log.Println(r)
            }
            scenario.resources.Cleanup(ctx)
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Println("Welcome to the Amazon S3 Object Lock Feature Scenario.")
    log.Println(strings.Repeat("-", 88))

    scenario.CreateBuckets(ctx)
    scenario.EnableLockOnBucket(ctx)
    scenario.SetDefaultRetentionPolicy(ctx)
    scenario.UploadTestObjects(ctx)
    scenario.SetObjectLockConfigurations(ctx)
    scenario.InteractWithObjects(ctx)

    scenario.resources.Cleanup(ctx)

    log.Println(strings.Repeat("-", 88))
    log.Println("Thanks for watching!")
    log.Println(strings.Repeat("-", 88))
}

```

이 예시에서 사용되는 S3 작업을 래핑하는 구조를 정의합니다.

```

import (
    "bytes"
    "context"

```

```
"errors"
"fmt"
"log"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// CreateBucketWithLock creates a new S3 bucket with optional object locking
// enabled
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
    region string, enableObjectLock bool) (string, error) {
    input := &s3.CreateBucketInput{
        Bucket: aws.String(bucket),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    }

    if enableObjectLock {
        input.ObjectLockEnabledForBucket = aws.Bool(true)
    }

    _, err := actor.S3Client.CreateBucket(ctx, input)
    if err != nil {
        var owned *types.BucketAlreadyOwnedByYou
        var exists *types.BucketAlreadyExists
        if errors.As(err, &owned) {
            log.Printf("You already own bucket %s.\n", bucket)
            err = owned
        } else if errors.As(err, &exists) {
```

```

    log.Printf("Bucket %s already exists.\n", bucket)
    err = exists
}
} else {
    err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
        ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
    }
}

return bucket, err
}

// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
    string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
    var status *types.ObjectLockLegalHoldStatus
    input := &s3.GetObjectLegalHoldInput{
        Bucket:    aws.String(bucket),
        Key:       aws.String(key),
        VersionId: aws.String(versionId),
    }

    output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
    if err != nil {
        var noSuchKeyErr *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noSuchKeyErr) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noSuchKeyErr
        } else if errors.As(err, &apiErr) {
            switch apiErr.ErrorCode() {
            case "NoSuchObjectLockConfiguration":
                log.Printf("Object %s does not have an object lock configuration.\n", key)
                err = nil
            case "InvalidRequest":
                log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
                err = nil
            }
        }
    } else {

```

```
    status = &output.LegalHold.Status
}

return status, err
}

// GetObjectLockConfiguration retrieves the object lock configuration for an S3
// bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
    string) (*types.ObjectLockConfiguration, error) {
    var lockConfig *types.ObjectLockConfiguration
    input := &s3.GetObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
    }

    output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
            "ObjectLockConfigurationNotFoundError" {
            log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
            err = nil
        }
    } else {
        lockConfig = output.ObjectLockConfiguration
    }

    return lockConfig, err
}

// GetObjectRetention retrieves the object retention configuration for an S3
// object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
    string) (*types.ObjectLockRetention, error) {
    var retention *types.ObjectLockRetention
    input := &s3.GetObjectRetentionInput{
```

```

    Bucket: aws.String(bucket),
    Key:     aws.String(key),
}

output, err := actor.S3Client.GetObjectRetention(ctx, input)
if err != nil {
    var noKey *types.NoSuchKey
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noKey) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noKey
    } else if errors.As(err, &apiErr) {
        switch apiErr.ErrorCode() {
        case "NoSuchObjectLockConfiguration":
            err = nil
        case "InvalidRequest":
            log.Printf("Bucket %s does not have locking enabled.", bucket)
            err = nil
        }
    }
} else {
    retention = output.Retention
}

return retention, err
}

// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
{
    input := &s3.PutObjectLegalHoldInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
        LegalHold: &types.ObjectLockLegalHold{
            Status: legalHoldStatus,
        },
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }
}

```

```

_, err := actor.S3Client.PutObjectLegalHold(ctx, input)
if err != nil {
    var noKey *types.NoSuchKey
    if errors.As(err, &noKey) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noKey
    }
}

return err
}

// ModifyDefaultBucketRetention modifies the default retention period of an
// existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
    ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
    retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

    input := &s3.PutObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
        ObjectLockConfiguration: &types.ObjectLockConfiguration{
            ObjectLockEnabled: lockMode,
            Rule: &types.ObjectLockRule{
                DefaultRetention: &types.DefaultRetention{
                    Days: aws.Int32(retentionPeriod),
                    Mode: retentionMode,
                },
            },
        },
    }

    _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    }

    return err
}

```

```
// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
string) error {
    // Versioning must be enabled on the bucket before object locking is enabled.
    verInput := &s3.PutBucketVersioningInput{
        Bucket: aws.String(bucket),
        VersioningConfiguration: &types.VersioningConfiguration{
            MFADelete: types.MFADeleteDisabled,
            Status:    types.BucketVersioningStatusEnabled,
        },
    }
    _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
        return err
    }

    input := &s3.PutObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
        ObjectLockConfiguration: &types.ObjectLockConfiguration{
            ObjectLockEnabled: types.ObjectLockEnabledEnabled,
        },
    }
    _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    }

    return err
}

// PutObjectRetention sets the object retention configuration for an S3 object.
```

```

func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
error {
    input := &s3.PutObjectRetentionInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
        Retention: &types.ObjectLockRetention{
            Mode:          retentionMode,
            RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
        },
        BypassGovernanceRetention: aws.Bool(true),
    }

    _, err := actor.S3Client.PutObjectRetention(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        }
    }

    return err
}

// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
string, contents string) (string, error) {
    var outKey string
    input := &s3.PutObjectInput{
        Bucket:          aws.String(bucket),
        Key:             aws.String(key),
        Body:            bytes.NewReader([]byte(contents)),
        ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
    }

    output, err := actor.S3Manager.Upload(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    }
}

```



```

    } else {
        err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
            Bucket: aws.String(bucket),
            Key:     aws.String(key),
        }, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
                bucket)
        } else {
            outKey = *output.Key
        }
    }
    return outKey, err
}

// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
    ([]types.ObjectVersion, error) {
    var err error
    var output *s3.ListObjectVersionsOutput
    var versions []types.ObjectVersion
    input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
    versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
    for versionPaginator.HasMorePages() {
        output, err = versionPaginator.NextPage(ctx)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucket)
                err = noBucket
            }
            break
        } else {
            versions = append(versions, output.Versions...)
        }
    }
    return versions, err
}

// DeleteObject deletes an object from a bucket.

```

```

func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
string, versionId string, bypassGovernance bool) (bool, error) {
    deleted := false
    input := &s3.DeleteObjectInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    _, err := actor.S3Client.DeleteObject(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in %s.\n", key, bucket)
            err = noKey
        } else if errors.As(err, &apiErr) {
            switch apiErr.ErrorCode() {
            case "AccessDenied":
                log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
                err = nil
            case "InvalidArgument":
                if bypassGovernance {
                    log.Printf("You cannot specify bypass governance on a bucket without lock
enabled.")
                    err = nil
                }
            }
        }
    } else {
        err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: aws.String(key)},
            time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s in bucket %s to be deleted.
\n", key, bucket)
        } else {
            deleted = true
        }
    }
}

```

```

    return deleted, err
}

// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
[]types.ObjectIdentifier, bypassGovernance bool) error {
    if len(objects) == 0 {
        return nil
    }

    input := s3.DeleteObjectsInput{
        Bucket: aws.String(bucket),
        Delete: &types.Delete{
            Objects: objects,
            Quiet:   aws.Bool(true),
        },
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
    if err != nil || len(delOut.Errors) > 0 {
        log.Printf("Error deleting objects from bucket %s.\n", bucket)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucket)
                err = noBucket
            }
        }
        } else if len(delOut.Errors) > 0 {
            for _, outErr := range delOut.Errors {
                log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
            }
            err = fmt.Errorf("%s", *delOut.Errors[0].Message)
        }
    } else {
        for _, delObjs := range delOut.Deleted {
            err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
                ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: delObjs.Key},
                time.Minute)
            if err != nil {

```

```
    log.Printf("Failed attempt to wait for object %s to be deleted.\n",
*del0bjs.Key)
    } else {
        log.Printf("Deleted %s.\n", *del0bjs.Key)
    }
}
}
return err
}
```

리소스를 정리합니다.

```
import (
    "context"
    "log"
    "s3_object_lock/actions"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// DemoBucket contains metadata for buckets used in this example.
type DemoBucket struct {
    name            string
    retentionEnabled bool
    objectKeys      []string
}

// Resources keeps track of AWS resources created during the ObjectLockScenario
// and handles
// cleanup when the scenario finishes.
type Resources struct {
    demoBuckets map[string]*DemoBucket

    s3Actions    *actions.S3Actions
    questioner    demotools.IQuestioner
}
```

```

}

// init initializes objects in the Resources struct.
func (resources *Resources) init(s3Actions *actions.S3Actions, questioner
    demotools.IQuestioner) {
    resources.s3Actions = s3Actions
    resources.questioner = questioner
    resources.demoBuckets = map[string]*DemoBucket{}
}

// Cleanup deletes all AWS resources created during the ObjectLockScenario.
func (resources *Resources) Cleanup(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            log.Printf("Something went wrong during cleanup.\n%v\n", r)
            log.Println("Use the AWS Management Console to remove any remaining resources"
                " +
                "that were created for this scenario.")
        }
    }()

    wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
        resources that were created "+
        "during this demo (y/n)?", "y")
    if !wantDelete {
        log.Println("Be sure to remove resources when you're done with them to avoid
            unexpected charges!")
        return
    }

    log.Println("Removing objects from S3 buckets and deleting buckets...")
    resources.deleteBuckets(ctx)
    //resources.deleteRetentionObjects(resources.retentionBucket,
    resources.retentionObjects)

    log.Println("Cleanup complete.")
}

// deleteBuckets empties and then deletes all buckets created during the
// ObjectLockScenario.
func (resources *Resources) deleteBuckets(ctx context.Context) {
    for _, info := range createInfo {
        bucket := resources.demoBuckets[info.name]
        resources.deleteObjects(ctx, bucket)
    }
}

```

```

_, err := resources.s3Actions.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
    Bucket: aws.String(bucket.name),
})
if err != nil {
    panic(err)
}
}
for _, info := range createInfo {
    bucket := resources.demoBuckets[info.name]
    err := s3.NewBucketNotExistsWaiter(resources.s3Actions.S3Client).Wait(
        ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket.name)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
bucket.name)
    } else {
        log.Printf("Deleted %s.\n", bucket.name)
    }
}
resources.demoBuckets = map[string]*DemoBucket{}
}

// deleteObjects deletes all objects in the specified bucket.
func (resources *Resources) deleteObjects(ctx context.Context, bucket
*DemoBucket) {
    lockConfig, err := resources.s3Actions.GetObjectLockConfiguration(ctx,
bucket.name)
    if err != nil {
        panic(err)
    }
    versions, err := resources.s3Actions.ListObjectVersions(ctx, bucket.name)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchBucket:
            log.Printf("No objects to get from %s.\n", bucket.name)
        default:
            panic(err)
        }
    }
    delObjects := make([]types.ObjectIdentifier, len(versions))
    for i, version := range versions {
        if lockConfig != nil && lockConfig.ObjectLockEnabled ==
types.ObjectLockEnabledEnabled {
            status, err := resources.s3Actions.GetObjectLegalHold(ctx, bucket.name,
*version.Key, *version.VersionId)

```

```
if err != nil {
    switch err.(type) {
    case *types.NoSuchKey:
        log.Printf("Couldn't determine legal hold status for %s in %s.\n",
*version.Key, bucket.name)
    default:
        panic(err)
    }
} else if status != nil && *status == types.ObjectLockLegalHoldStatusOn {
    err = resources.s3Actions.PutObjectLegalHold(ctx, bucket.name, *version.Key,
*version.VersionId, types.ObjectLockLegalHoldStatusOff)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchKey:
            log.Printf("Couldn't turn off legal hold for %s in %s.\n", *version.Key,
bucket.name)
        default:
            panic(err)
        }
    }
}
}
del0bjects[i] = types.ObjectIdentifier{Key: version.Key, VersionId:
version.VersionId}
}
err = resources.s3Actions.DeleteObjects(ctx, bucket.name, del0bjects,
bucket.retentionEnabled)
if err != nil {
    switch err.(type) {
    case *types.NoSuchBucket:
        log.Println("Nothing to delete.")
    default:
        panic(err)
    }
}
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하세요.
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)

- [GetObjectRetention](#)
- [PutObjectLegalHold](#)
- [PutObjectLockConfiguration](#)
- [PutObjectRetention](#)

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 객체 잠금 기능을 시연하는 대화형 시나리오를 실행합니다.

```
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import java.io.BufferedWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;

/*
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/setup.html

This Java example performs the following tasks:
1. Create test Amazon Simple Storage Service (S3) buckets with different lock
policies.
2. Upload sample objects to each bucket.
3. Set some Legal Hold and Retention Periods on objects and buckets.
4. Investigate lock policies by viewing settings or attempting to delete or
overwrite objects.
```


5. Clean up objects and buckets.

```

*/
public class S3ObjectLockWorkflow {

    public static final String DASHES = new String(new char[80]).replace("\0",
"-");
    static String bucketName;
    static S3LockActions s3LockActions;
    private static final List<String> bucketNames = new ArrayList<>();
    private static final List<String> fileNames = new ArrayList<>();

    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <bucketName> \s

            Where:
                bucketName - The Amazon S3 bucket name.
            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        s3LockActions = new S3LockActions();
        bucketName = args[0];
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Simple Storage Service (S3)
Object Locking Feature Scenario.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        configurationSetup();
        System.out.println(DASHES);

        System.out.println(DASHES);
        setup();
        System.out.println("Setup is complete. Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Lets present the user with choices.");
    }
}

```

```
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        demoActionChoices() ;
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Would you like to clean up the resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanup();
            System.out.println("Clean up is complete.");
        }

        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Amazon S3 Object Locking Workflow is complete.");
        System.out.println(DASHES);
    }

    // Present the user with the demo action choices.
    public static void demoActionChoices() {
        String[] choices = {
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the workflow."
        };

        int choice = 0;
        while (true) {
            System.out.println(DASHES);
            choice = getChoiceResponse("Explore the S3 locking features by
selecting one of the following choices:", choices);
            System.out.println(DASHES);
            System.out.println("You selected "+choices[choice]);
            switch (choice) {
                case 0 -> {
                    s3LockActions.listBucketsAndObjects(bucketNames, true);
```

```
        }

        case 1 -> {
            System.out.println("Enter the number of the object to
delete:");

            List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
            List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
            String[] fileKeysArray = fileKeys.toArray(new String[0]);
            int fileChoice = getChoiceResponse(null, fileKeysArray);
            String objectKey = fileKeys.get(fileChoice);
            String bucketName = allFiles.get(fileChoice).getBucketName();
            String version = allFiles.get(fileChoice).getVersion();
            s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
false, version);
        }

        case 2 -> {
            System.out.println("Enter the number of the object to
delete:");

            List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
            List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
            String[] fileKeysArray = fileKeys.toArray(new String[0]);
            int fileChoice = getChoiceResponse(null, fileKeysArray);
            String objectKey = fileKeys.get(fileChoice);
            String bucketName = allFiles.get(fileChoice).getBucketName();
            String version = allFiles.get(fileChoice).getVersion();
            s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
true, version);
        }

        case 3 -> {
            System.out.println("Enter the number of the object to
overwrite:");

            List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
            List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
            String[] fileKeysArray = fileKeys.toArray(new String[0]);
            int fileChoice = getChoiceResponse(null, fileKeysArray);
            String objectKey = fileKeys.get(fileChoice);
```

```
String bucketName = allFiles.get(fileChoice).getBucketName();

// Attempt to overwrite the file.
try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(objectKey))) {
    writer.write("This is a modified text.");

} catch (IOException e) {
    e.printStackTrace();
}
s3LockActions.uploadFile(bucketName, objectKey, objectKey);
}

case 4 -> {
    System.out.println("Enter the number of the object to
overwrite:");

    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
    String[] fileKeysArray = fileKeys.toArray(new String[0]);
    int fileChoice = getChoiceResponse(null, fileKeysArray);
    String objectKey = fileKeys.get(fileChoice);
    String bucketName = allFiles.get(fileChoice).getBucketName();
    s3LockActions.getObjectRetention(bucketName, objectKey);
}

case 5 -> {
    System.out.println("Enter the number of the object to
view:");

    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
    String[] fileKeysArray = fileKeys.toArray(new String[0]);
    int fileChoice = getChoiceResponse(null, fileKeysArray);
    String objectKey = fileKeys.get(fileChoice);
    String bucketName = allFiles.get(fileChoice).getBucketName();
    s3LockActions.getObjectLegalHold(bucketName, objectKey);
    s3LockActions.getBucketObjectLockConfiguration(bucketName);
}

case 6 -> {
    System.out.println("Exiting the workflow...");
```

```

        return;
    }

    default -> {
        System.out.println("Invalid choice. Please select again.");
    }
}

}

}

// Clean up the resources from the scenario.
private static void cleanup() {
    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, false);
    for (S3InfoObject fileInfo : allFiles) {
        String bucketName = fileInfo.getBucketName();
        String key = fileInfo.getKeyName();
        String version = fileInfo.getVersion();
        if (bucketName.contains("lock-enabled") ||
(bucketName.contains("retention-after-creation"))) {
            ObjectLockLegalHold legalHold =
s3LockActions.getObjectLegalHold(bucketName, key);
            if (legalHold != null) {
                String holdStatus = legalHold.status().name();
                System.out.println(holdStatus);
                if (holdStatus.compareTo("ON") == 0) {
                    s3LockActions.modifyObjectLegalHold(bucketName, key,
false);
                }
            }
            // Check for a retention period.
            ObjectLockRetention retention =
s3LockActions.getObjectRetention(bucketName, key);
            boolean hasRetentionPeriod ;
            hasRetentionPeriod = retention != null;
            s3LockActions.deleteObjectFromBucket(bucketName,
key,hasRetentionPeriod, version);
        } else {
            System.out.println(bucketName +" objects do not have a legal
lock");
            s3LockActions.deleteObjectFromBucket(bucketName, key,false,
version);
        }
    }
}

```

```
    }

    // Delete the buckets.
    System.out.println("Delete "+bucketName);
    for (String bucket : bucketNames){
        s3LockActions.deleteBucketByName(bucket);
    }
}

private static void setup() {
    Scanner scanner = new Scanner(System.in);
    System.out.println("""
        For this workflow, we will use the AWS SDK for Java to create
several S3
        buckets and files to demonstrate working with S3 locking
features.
        """);

    System.out.println("S3 buckets can be created either with or without
object lock enabled.");
    System.out.println("Press Enter to continue...");
    scanner.nextLine();

    // Create three S3 buckets.
    s3LockActions.createBucketWithLockOptions(false, bucketNames.get(0));
    s3LockActions.createBucketWithLockOptions(true, bucketNames.get(1));
    s3LockActions.createBucketWithLockOptions(false, bucketNames.get(2));
    System.out.println("Press Enter to continue.");
    scanner.nextLine();

    System.out.println("Bucket "+bucketNames.get(2) +" will be configured to
use object locking with a default retention period.");
    s3LockActions.modifyBucketDefaultRetention(bucketNames.get(2));
    System.out.println("Press Enter to continue.");
    scanner.nextLine();

    System.out.println("Object lock policies can also be added to existing
buckets. For this example, we will use "+bucketNames.get(1));
    s3LockActions.enableObjectLockOnBucket(bucketNames.get(1));
    System.out.println("Press Enter to continue.");
    scanner.nextLine();

    // Upload some files to the buckets.
    System.out.println("Now let's add some test files:");
```

```

        String fileName = "exampleFile.txt";
        int fileCount = 2;
        try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(fileName))) {
            writer.write("This is a sample file for uploading to a bucket.");

        } catch (IOException e) {
            e.printStackTrace();
        }

        for (String bucketName : bucketNames){
            for (int i = 0; i < fileCount; i++) {
                // Get the file name without extension.
                String fileNameWithoutExtension =
java.nio.file.Paths.get(fileName).getFileName().toString();
                int extensionIndex = fileNameWithoutExtension.lastIndexOf('.');
                if (extensionIndex > 0) {
                    fileNameWithoutExtension =
fileNameWithoutExtension.substring(0, extensionIndex);
                }

                // Create the numbered file names.
                String numberedFileName = fileNameWithoutExtension + i +
getFileExtension(fileName);
                fileNames.add(numberedFileName);
                s3LockActions.uploadFile(bucketName, numberedFileName, fileName);
            }
        }

        String question = null;
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        System.out.println("Now we can set some object lock policies on
individual files:");
        for (String bucketName : bucketNames) {
            for (int i = 0; i < fileNames.size(); i++){

                // No modifications to the objects in the first bucket.
                if (!bucketName.equals(bucketNames.get(0))) {
                    String exampleFileName = fileNames.get(i);
                    switch (i) {
                        case 0 -> {
                            question = "Would you like to add a legal hold to " +
exampleFileName + " in " + bucketName + " (y/n)?";

```

```

        System.out.println(question);
        String ans = scanner.nextLine().trim();
        if (ans.equalsIgnoreCase("y")) {
            System.out.println("**** You have selected to put
a legal hold " + exampleFileName);

            // Set a legal hold.
            s3LockActions.modifyObjectLegalHold(bucketName,
exampleFileName, true);
        }
    }
    case 1 -> {
        ""
        Would you like to add a 1 day Governance
retention period to %s in %s (y/n)?
        Reminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.
        "".formatted(exampleFileName, bucketName);
        System.out.println(question);
        String ans2 = scanner.nextLine().trim();
        if (ans2.equalsIgnoreCase("y")) {
            s3LockActions.modifyObjectRetentionPeriod(bucketName, exampleFileName);
        }
    }
}
}
}
}
}
}

// Get file extension.
private static String getFileExtension(String fileName) {
    int dotIndex = fileName.lastIndexOf('.');
    if (dotIndex > 0) {
        return fileName.substring(dotIndex);
    }
    return "";
}

public static void configurationSetup() {
    String noLockBucketName = bucketName + "-no-lock";
    String lockEnabledBucketName = bucketName + "-lock-enabled";

```



```

        String retentionAfterCreationBucketName = bucketName + "-retention-after-creation";
        bucketNames.add(noLockBucketName);
        bucketNames.add(lockEnabledBucketName);
        bucketNames.add(retentionAfterCreationBucketName);
    }

    public static int getChoiceResponse(String question, String[] choices) {
        Scanner scanner = new Scanner(System.in);
        if (question != null) {
            System.out.println(question);
            for (int i = 0; i < choices.length; i++) {
                System.out.println("\t" + (i + 1) + ". " + choices[i]);
            }
        }

        int choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.length) {
            String choice = scanner.nextLine();
            try {
                choiceNumber = Integer.parseInt(choice);
            } catch (NumberFormatException e) {
                System.out.println("Invalid choice. Please enter a valid number.");
            }
        }

        return choiceNumber - 1;
    }
}

```

S3 함수의 래퍼 클래스입니다.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketVersioningStatus;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DefaultRetention;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldRequest;

```

```
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldResponse;
import
    software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationRequest;
import
    software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationResponse;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionResponse;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.MFADelete;
import software.amazon.awssdk.services.s3.model.ObjectLockConfiguration;
import software.amazon.awssdk.services.s3.model.ObjectLockEnabled;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHoldStatus;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.ObjectLockRetentionMode;
import software.amazon.awssdk.services.s3.model.ObjectLockRule;
import software.amazon.awssdk.services.s3.model.PutBucketVersioningRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLegalHoldRequest;
import
    software.amazon.awssdk.services.s3.model.PutObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.VersioningConfiguration;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.stream.Collectors;

// Contains application logic for the Amazon S3 operations used in this workflow.
public class S3LockActions {

    private static S3Client getClient() {
        return S3Client.builder()
```

```
        .region(Region.US_EAST_1)
        .build();
    }

    // Set or modify a retention period on an object in an S3 bucket.
    public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
    {
        // Calculate the instant one day from now.
        Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

        // Convert the Instant to a ZonedDateTime object with a specific time
        zone.
        ZonedDateTime zonedDateTime =
        futureInstant.atZone(ZoneId.systemDefault());

        // Define a formatter for human-readable output.
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
        HH:mm:ss");

        // Format the ZonedDateTime object to a human-readable date string.
        String humanReadableDate = formatter.format(zonedDateTime);

        // Print the formatted date string.
        System.out.println("Formatted Date: " + humanReadableDate);
        ObjectLockRetention retention = ObjectLockRetention.builder()
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .retainUntilDate(futureInstant)
            .build();

        PutObjectRetentionRequest retentionRequest =
        PutObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .retention(retention)
            .build();

        getClient().putObjectRetention(retentionRequest);
        System.out.println("Set retention for "+objectKey+" in "+bucketName+"
        until "+ humanReadableDate +".");
    }

    // Get the legal hold details for an S3 object.
    public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
    objectKey) {
```

```

        try {
            GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
            System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
                ":\n\tStatus: " + response.legalHold().status());
            return response.legalHold();

        } catch (S3Exception ex) {
            System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
        }

        return null;
    }

    // Create a new Amazon S3 bucket with object lock options.
    public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
        S3Waiter s3Waiter = getClient().waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectLockEnabledForBucket(enableObjectLock)
            .build();

        getClient().createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        s3Waiter.waitUntilBucketExists(bucketRequestWait);
        System.out.println(bucketName + " is ready");
    }

    public List<S3InfoObject> listBucketsAndObjects(List<String> bucketNames,
Boolean interactive) {
        AtomicInteger counter = new AtomicInteger(0); // Initialize counter.

```

```

        return bucketNames.stream()
            .flatMap(bucketName ->
listBucketObjectsAndVersions(bucketName).versions().stream()
                .map(version -> {
                    S3InfoObject s3InfoObject = new S3InfoObject();
                    s3InfoObject.setBucketName(bucketName);
                    s3InfoObject.setVersion(version.versionId());
                    s3InfoObject.setKeyName(version.key());
                    return s3InfoObject;
                })
            .peek(s3InfoObject -> {
                int i = counter.incrementAndGet(); // Increment and get the
updated value.
                if (interactive) {
                    System.out.println(i + ": " + s3InfoObject.getKeyName());
                    System.out.printf("%5s Bucket name: %s\n", "",
s3InfoObject.getBucketName());
                    System.out.printf("%5s Version: %s\n", "",
s3InfoObject.getVersion());
                }
            })
            .collect(Collectors.toList());
    }

    public ListObjectVersionsResponse listBucketObjectsAndVersions(String
bucketName) {
        ListObjectVersionsRequest versionsRequest =
ListObjectVersionsRequest.builder()
            .bucket(bucketName)
            .build();

        return getClient().listObjectVersions(versionsRequest);
    }

    // Set or modify a retention period on an S3 bucket.
    public void modifyBucketDefaultRetention(String bucketName) {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .mfaDelete(MFADelete.DISABLED)
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()

```

```
        .bucket(bucketName)
        .versioningConfiguration(versioningConfiguration)
        .build();

getClient().putBucketVersioning(versioningRequest);
DefaultRetention retention = DefaultRetention.builder()
    .days(1)
    .mode(ObjectLockRetentionMode.GOVERNANCE)
    .build();

ObjectLockRule lockRule = ObjectLockRule.builder()
    .defaultRetention(retention)
    .build();

ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
    .objectLockEnabled(ObjectLockEnabled.ENABLED)
    .rule(lockRule)
    .build();

PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
    .bucket(bucketName)
    .objectLockConfiguration(objectLockConfiguration)
    .build();

getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
    System.out.println("Added a default retention to bucket "+bucketName
+ ".");
}

// Enable object lock on an existing bucket.
public void enableObjectLockOnBucket(String bucketName) {
    try {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
```

```

        .build();

        // Enable versioning on the bucket.
        getClient().putBucketVersioning(putBucketVersioningRequest);
        PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(ObjectLockConfiguration.builder()
                .objectLockEnabled(ObjectLockEnabled.ENABLED)
                .build())
            .build();

        getClient().putObjectLockConfiguration(request);
        System.out.println("Successfully enabled object lock on
"+bucketName);

    } catch (S3Exception ex) {
        System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
    }
}

public void uploadFile(String bucketName, String objectName, String filePath)
{
    Path file = Paths.get(filePath);
    PutObjectRequest request = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(objectName)
        .checksumAlgorithm(ChecksumAlgorithm.SHA256)
        .build();

    PutObjectResponse response = getClient().putObject(request, file);
    if (response != null) {
        System.out.println("\tSuccessfully uploaded " + objectName + " to " +
bucketName + ".");
    } else {
        System.out.println("\tCould not upload " + objectName + " to " +
bucketName + ".");
    }
}

// Set or modify a legal hold on an object in an S3 bucket.
public void modifyObjectLegalHold(String bucketName, String objectKey,
boolean legalHoldOn) {

```

```
        ObjectLockLegalHold legalHold ;
        if (legalHoldOn) {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.ON)
                .build();
        } else {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.OFF)
                .build();
        }

        PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .legalHold(legalHold)
            .build();

        getClient().putObjectLegalHold(legalHoldRequest) ;
        System.out.println("Modified legal hold for "+ objectKey +" in
"+bucketName +".");
    }

    // Delete an object from a specific bucket.
    public void deleteObjectFromBucket(String bucketName, String objectKey,
boolean hasRetention, String versionId) {
        try {
            DeleteObjectRequest objectRequest;
            if (hasRetention) {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .bypassGovernanceRetention(true)
                    .build();
            } else {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .build();
            }

            getClient().deleteObject(objectRequest) ;
```



```
        System.out.println("The object was successfully deleted");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

// Get the retention period for an S3 object.
public ObjectLockRetention getObjectRetention(String bucketName, String key){
    try {
        GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
        System.out.println("tObject retention for "+key +"
in "+ bucketName +": " + response.retention().mode() +" until "+
response.retention().retainUntilDate() +".");
        return response.retention();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        return null;
    }
}

public void deleteBucketByName(String bucketName) {
    try {
        DeleteBucketRequest request = DeleteBucketRequest.builder()
            .bucket(bucketName)
            .build();

        getClient().deleteBucket(request);
        System.out.println(bucketName +" was deleted.");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

// Get the object lock configuration details for an S3 bucket.
```

```

public void getBucketObjectLockConfiguration(String bucketName) {
    GetObjectLockConfigurationRequest objectLockConfigurationRequest =
    GetObjectLockConfigurationRequest.builder()
        .bucket(bucketName)
        .build();

    GetObjectLockConfigurationResponse response =
    getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
    System.out.println("Bucket object lock config for "+bucketName+": ");
    System.out.println("\tEnabled:
    "+response.getObjectLockConfiguration().objectLockEnabled());
    System.out.println("\tRule: "+
    response.getObjectLockConfiguration().rule().defaultRetention());
}
}

```

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

- [GetObjectLegalHold](#)
- [GetObjectLockConfiguration](#)
- [GetObjectRetention](#)
- [PutObjectLegalHold](#)
- [PutObjectLockConfiguration](#)
- [PutObjectRetention](#)

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

시나리오의 진입점입니다(index.js). 모든 단계를 오케스트레이션합니다. GitHub를 방문하여 Scenario, ScenarioInput, ScenarioOutput, ScenarioAction에 대한 구현 세부 정보를 확인하세요.

```
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
  loadState,
  saveState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "../welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  confirmSetLegalHoldFileEnabled,
  confirmSetLegalHoldFileRetention,
  confirmSetRetentionPeriodFileEnabled,
  confirmSetRetentionPeriodFileRetention,
  confirmUpdateLockPolicy,
  confirmUpdateRetention,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
  setLegalHoldFileEnabledAction,
  setLegalHoldFileRetentionAction,
  setRetentionPeriodFileEnabledAction,
  setRetentionPeriodFileRetentionAction,
  updateLockPolicy,
  updateLockPolicyAction,
  updateRetention,
  updateRetentionAction,
} from "../setup.steps.js";

/**
 * @param {Scenarios} scenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});

  return {
    deploy: new scenarios.Scenario(
      "S3 Object Locking - Deploy",
      [
```

```

        welcome(scenarios),
        welcomeContinue(scenarios),
        exitOnFalse(scenarios, "welcomeContinue"),
        getBucketPrefix(scenarios),
        createBuckets(scenarios),
        confirmCreateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmCreateBuckets"),
        createBucketsAction(scenarios, client),
        updateRetention(scenarios),
        confirmUpdateRetention(scenarios),
        exitOnFalse(scenarios, "confirmUpdateRetention"),
        updateRetentionAction(scenarios, client),
        populateBuckets(scenarios),
        confirmPopulateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmPopulateBuckets"),
        populateBucketsAction(scenarios, client),
        updateLockPolicy(scenarios),
        confirmUpdateLockPolicy(scenarios),
        exitOnFalse(scenarios, "confirmUpdateLockPolicy"),
        updateLockPolicyAction(scenarios, client),
        confirmSetLegalHoldFileEnabled(scenarios),
        setLegalHoldFileEnabledAction(scenarios, client),
        confirmSetRetentionPeriodFileEnabled(scenarios),
        setRetentionPeriodFileEnabledAction(scenarios, client),
        confirmSetLegalHoldFileRetention(scenarios),
        setLegalHoldFileRetentionAction(scenarios, client),
        confirmSetRetentionPeriodFileRetention(scenarios),
        setRetentionPeriodFileRetentionAction(scenarios, client),
        saveState,
    ],
    initialState,
),
demo: new scenarios.Scenario(
    "S3 Object Locking - Demo",
    [loadState, replAction(scenarios, client)],
    initialState,
),
clean: new scenarios.Scenario(
    "S3 Object Locking - Destroy",
    [
        loadState,
        confirmCleanup(scenarios),
        exitOnFalse(scenarios, "confirmCleanup"),
        cleanupAction(scenarios, client),
    ]

```

```

    ],
    initialState,
  ),
};
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const objectLockingScenarios = getWorkflowStages(Scenarios);
  Scenarios.parseScenarioArgs(objectLockingScenarios, {
    name: "Amazon S3 object locking workflow",
    description:
      "Work with Amazon Simple Storage Service (Amazon S3) object locking features.",
    synopsis:
      "node index.js --scenario <deploy | demo | clean> [-h|--help] [-y|--yes] [-v|--verbose]",
  });
}

```

환영 메시지를 콘솔에 출력합니다(welcome.steps.js).

```

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @param {Scenarios} scenarios
 */
const welcome = (scenarios) =>
  new scenarios.ScenarioOutput(
    "welcome",
    "Welcome to the Amazon Simple Storage Service (S3) Object Locking Feature Scenario. For this workflow, we will use the AWS SDK for JavaScript to create several S3 buckets and files to demonstrate working with S3 locking features.",
    { header: true },
  );

```

```

/**
 * @param {Scenarios} scenarios
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };

```

버킷, 객체, 파일 설정을 배포합니다(setup.steps.js).

```

import {
  BucketVersioningStatus,
  ChecksumAlgorithm,
  CreateBucketCommand,
  MFADeleteStatus,
  PutBucketVersioningCommand,
  PutObjectCommand,
  PutObjectLockConfigurationCommand,
  PutObjectLegalHoldCommand,
  PutObjectRetentionCommand,
  ObjectLockLegalHoldStatus,
  ObjectLockRetentionMode,
  GetBucketVersioningCommand,
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
  S3ServiceException,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client

```

```

*/

/**
 * @param {Scenarios} scenarios
 */
const getBucketPrefix = (scenarios) =>
  new scenarios.ScenarioInput(
    "bucketPrefix",
    "Provide a prefix that will be used for bucket creation.",
    { type: "input", default: "amzn-s3-demo-bucket" },
  );

/**
 * @param {Scenarios} scenarios
 */
const createBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "createBuckets",
    (state) => `The following buckets will be created:
      ${state.bucketPrefix}-no-lock with object lock False.
      ${state.bucketPrefix}-lock-enabled with object lock True.
      ${state.bucketPrefix}-retention-after-creation with object lock False.` ,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmCreateBuckets = (scenarios) =>
  new scenarios.ScenarioInput("confirmCreateBuckets", "Create the buckets?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("createBucketsAction", async (state) => {
    const noLockBucketName = `${state.bucketPrefix}-no-lock`;
    const lockEnabledBucketName = `${state.bucketPrefix}-lock-enabled`;
    const retentionBucketName = `${state.bucketPrefix}-retention-after-creation`;

    try {

```

```

    await client.send(new CreateBucketCommand({ Bucket: noLockBucketName }));
    await waitUntilBucketExists({ client }, { Bucket: noLockBucketName });
    await client.send(
      new CreateBucketCommand({
        Bucket: lockEnabledBucketName,
        ObjectLockEnabledForBucket: true,
      }),
    );
    await waitUntilBucketExists(
      { client },
      { Bucket: lockEnabledBucketName },
    );
    await client.send(
      new CreateBucketCommand({ Bucket: retentionBucketName }),
    );
    await waitUntilBucketExists({ client }, { Bucket: retentionBucketName });

    state.noLockBucketName = noLockBucketName;
    state.lockEnabledBucketName = lockEnabledBucketName;
    state.retentionBucketName = retentionBucketName;
  } catch (caught) {
    if (
      caught instanceof BucketAlreadyExists ||
      caught instanceof BucketAlreadyOwnedByYou
    ) {
      console.error(`${caught.name}: ${caught.message}`);
      state.earlyExit = true;
    } else {
      throw caught;
    }
  }
});

/**
 * @param {Scenarios} scenarios
 */
const populateBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "populateBuckets",
    (state) => `The following test files will be created:
      file0.txt in ${state.bucketPrefix}-no-lock.
      file1.txt in ${state.bucketPrefix}-no-lock.
      file0.txt in ${state.bucketPrefix}-lock-enabled.
      file1.txt in ${state.bucketPrefix}-lock-enabled.`
  );

```



```
        file0.txt in ${state.bucketPrefix}-retention-after-creation.
        file1.txt in ${state.bucketPrefix}-retention-after-creation.` ,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmPopulateBuckets = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmPopulateBuckets",
    "Populate the buckets?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const populateBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("populateBucketsAction", async (state) => {
    try {
      await client.send(
        new PutObjectCommand({
          Bucket: state.noLockBucketName,
          Key: "file0.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.noLockBucketName,
          Key: "file1.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file0.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
```

```

    })),
  );
  await client.send(
    new PutObjectCommand({
      Bucket: state.lockEnabledBucketName,
      Key: "file1.txt",
      Body: "Content",
      ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
    })),
  );
  await client.send(
    new PutObjectCommand({
      Bucket: state.retentionBucketName,
      Key: "file0.txt",
      Body: "Content",
      ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
    })),
  );
  await client.send(
    new PutObjectCommand({
      Bucket: state.retentionBucketName,
      Key: "file1.txt",
      Body: "Content",
      ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
    })),
  );
} catch (caught) {
  if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while uploading object.  ${caught.name}:
${caught.message}`,
    );
  } else {
    throw caught;
  }
}
});

/**
 * @param {Scenarios} scenarios
 */
const updateRetention = (scenarios) =>
  new scenarios.ScenarioOutput(
    "updateRetention",

```

```

    (state) => `A bucket can be configured to use object locking with a default
    retention period.
A default retention period will be configured for ${state.bucketPrefix}-
retention-after-creation.` ,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmUpdateRetention",
    "Configure default retention period?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const updateRetentionAction = (scenarios, client) => {
  new scenarios.ScenarioAction("updateRetentionAction", async (state) => {
    await client.send(
      new PutBucketVersioningCommand({
        Bucket: state.retentionBucketName,
        VersioningConfiguration: {
          MFADelete: MFADeleteStatus.Disabled,
          Status: BucketVersioningStatus.Enabled,
        },
      }),
    );

    const getBucketVersioning = new GetBucketVersioningCommand({
      Bucket: state.retentionBucketName,
    });

    await retry({ intervalInMs: 500, maxRetries: 10 }, async () => {
      const { Status } = await client.send(getBucketVersioning);
      if (Status !== "Enabled") {
        throw new Error("Bucket versioning is not enabled.");
      }
    });
  });
};

```

```

    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: state.retentionBucketName,
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
          Rule: {
            DefaultRetention: {
              Mode: "GOVERNANCE",
              Years: 1,
            },
          },
        },
      }),
    );
  });

/**
 * @param {Scenarios} scenarios
 */
const updateLockPolicy = (scenarios) =>
  new scenarios.ScenarioOutput(
    "updateLockPolicy",
    (state) => `Object lock policies can also be added to existing buckets.
An object lock policy will be added to ${state.bucketPrefix}-lock-enabled.` ,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateLockPolicy = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmUpdateLockPolicy",
    "Add object lock policy?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const updateLockPolicyAction = (scenarios, client) =>
  new scenarios.ScenarioAction("updateLockPolicyAction", async (state) => {
    await client.send(

```

```

        new PutObjectLockConfigurationCommand({
            Bucket: state.lockEnabledBucketName,
            ObjectLockConfiguration: {
                ObjectLockEnabled: "Enabled",
            },
        }),
    );
});

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileEnabled = (scenarios) =>
    new scenarios.ScenarioInput(
        "confirmSetLegalHoldFileEnabled",
        (state) =>
            `Would you like to add a legal hold to file0.txt in
            ${state.lockEnabledBucketName}?`,
        {
            type: "confirm",
        },
    );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileEnabledAction = (scenarios, client) =>
    new scenarios.ScenarioAction(
        "setLegalHoldFileEnabledAction",
        async (state) => {
            await client.send(
                new PutObjectLegalHoldCommand({
                    Bucket: state.lockEnabledBucketName,
                    Key: "file0.txt",
                    LegalHold: {
                        Status: ObjectLockLegalHoldStatus.ON,
                    },
                }),
            );
            console.log(
                `Modified legal hold for file0.txt in ${state.lockEnabledBucketName}.`,
            );
        },
    );

```

```

    },
    { skipWhen: (state) => !state.confirmSetLegalHoldFileEnabled },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetRetentionPeriodFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileEnabled",
    (state) =>
      `Would you like to add a 1 day Governance retention period to file1.txt in
      ${state.lockEnabledBucketName}?
      Reminder: Only a user with the s3:BypassGovernanceRetention permission will be
      able to delete this file or its bucket until the retention period has expired.`
    ,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileEnabledAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileEnabledAction",
    async (state) => {
      const retentionDate = new Date();
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
        new PutObjectRetentionCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
            RetainUntilDate: retentionDate,
          },
        })
      );
      console.log(
        `Set retention for file1.txt in ${state.lockEnabledBucketName} until
        ${retentionDate.toISOString().split("T")[0]}.`,
      );
    },
  );

```

```

    );
  },
  { skipWhen: (state) => !state.confirmSetRetentionPeriodFileEnabled },
);

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetLegalHoldFileRetention",
    (state) =>
      `Would you like to add a legal hold to file0.txt in
      ${state.retentionBucketName}?`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setLegalHoldFileRetentionAction",
    async (state) => {
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: state.retentionBucketName,
          Key: "file0.txt",
          LegalHold: {
            Status: ObjectLockLegalHoldStatus.ON,
          },
        }),
      );
      console.log(
        `Modified legal hold for file0.txt in ${state.retentionBucketName}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetLegalHoldFileRetention },
  );

```

```

/**
 * @param {Scenarios} scenarios
 */
const confirmSetRetentionPeriodFileRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileRetention",
    (state) =>
      `Would you like to add a 1 day Governance retention period to file1.txt in
      ${state.retentionBucketName}?
      Reminder: Only a user with the s3:BypassGovernanceRetention permission will be
      able to delete this file or its bucket until the retention period has expired.`
    ,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileRetentionAction",
    async (state) => {
      const retentionDate = new Date();
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
        new PutObjectRetentionCommand({
          Bucket: state.retentionBucketName,
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
            RetainUntilDate: retentionDate,
          },
          BypassGovernanceRetention: true,
        })
      );
      console.log(
        `Set retention for file1.txt in ${state.retentionBucketName} until
        ${retentionDate.toISOString().split("T")[0]}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetRetentionPeriodFileRetention },
  );

```



```
export {
  getBucketPrefix,
  createBuckets,
  confirmCreateBuckets,
  createBucketsAction,
  populateBuckets,
  confirmPopulateBuckets,
  populateBucketsAction,
  updateRetention,
  confirmUpdateRetention,
  updateRetentionAction,
  updateLockPolicy,
  confirmUpdateLockPolicy,
  updateLockPolicyAction,
  confirmSetLegalHoldFileEnabled,
  setLegalHoldFileEnabledAction,
  confirmSetRetentionPeriodFileEnabled,
  setRetentionPeriodFileEnabledAction,
  confirmSetLegalHoldFileRetention,
  setLegalHoldFileRetentionAction,
  confirmSetRetentionPeriodFileRetention,
  setRetentionPeriodFileRetentionAction,
};
```

버킷의 파일을 조회하고 삭제합니다(repl.steps.js).

```
import {
  ChecksumAlgorithm,
  DeleteObjectCommand,
  GetObjectLegalHoldCommand,
  GetObjectLockConfigurationCommand,
  GetObjectRetentionCommand,
  ListObjectVersionsCommand,
  PutObjectCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
```

```
* @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
*/

const choices = {
  EXIT: 0,
  LIST_ALL_FILES: 1,
  DELETE_FILE: 2,
  DELETE_FILE_WITH_RETENTION: 3,
  OVERWRITE_FILE: 4,
  VIEW_RETENTION_SETTINGS: 5,
  VIEW_LEGAL_HOLD_SETTINGS: 6,
};

/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new scenarios.ScenarioInput(
    "replChoice",
    "Explore the S3 locking features by selecting one of the following choices",
    {
      type: "select",
      choices: [
        { name: "List all files in buckets", value: choices.LIST_ALL_FILES },
        { name: "Attempt to delete a file.", value: choices.DELETE_FILE },
        {
          name: "Attempt to delete a file with retention period bypass.",
          value: choices.DELETE_FILE_WITH_RETENTION,
        },
        { name: "Attempt to overwrite a file.", value: choices.OVERWRITE_FILE },
        {
          name: "View the object and bucket retention settings for a file.",
          value: choices.VIEW_RETENTION_SETTINGS,
        },
        {
          name: "View the legal hold settings for a file.",
          value: choices.VIEW_LEGAL_HOLD_SETTINGS,
        },
        { name: "Finish the workflow.", value: choices.EXIT },
      ],
    },
  );

/**
```

```

    * @param {S3Client} client
    * @param {string[]} buckets
    */
const getAllFiles = async (client, buckets) => {
    /** @type {{bucket: string, key: string, version: string}[]} */
    const files = [];
    for (const bucket of buckets) {
        const objectsResponse = await client.send(
            new ListObjectVersionsCommand({ Bucket: bucket }),
        );
        for (const version of objectsResponse.Versions || []) {
            const { Key, VersionId } = version;
            files.push({ bucket, key: Key, version: VersionId });
        }
    }

    return files;
};

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const replAction = (scenarios, client) =>
    new scenarios.ScenarioAction(
        "replAction",
        async (state) => {
            const files = await getAllFiles(client, [
                state.noLockBucketName,
                state.lockEnabledBucketName,
                state.retentionBucketName,
            ]);

            const fileInput = new scenarios.ScenarioInput(
                "selectedFile",
                "Select a file:",
                {
                    type: "select",
                    choices: files.map((file, index) => ({
                        name: `${index + 1}: ${file.bucket}: ${file.key} (version: ${
                            file.version
                        })`,
                        value: index,
                    })),
                },
            );

```

```

    },
  );

const { replChoice } = state;

switch (replChoice) {
  case choices.LIST_ALL_FILES: {
    const files = await getAllFiles(client, [
      state.noLockBucketName,
      state.lockEnabledBucketName,
      state.retentionBucketName,
    ]);
    state.replOutput = files
      .map(
        (file) =>
          `${file.bucket}: ${file.key} (version: ${file.version})`,
      )
      .join("\n");
    break;
  }
  case choices.DELETE_FILE: {
    /** @type {number} */
    const fileToDelete = await fileInput.handle(state);
    const selectedFile = files[fileToDelete];
    try {
      await client.send(
        new DeleteObjectCommand({
          Bucket: selectedFile.bucket,
          Key: selectedFile.key,
          VersionId: selectedFile.version,
        }),
      );
      state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
    } catch (err) {
      state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
    }
    break;
  }
  case choices.DELETE_FILE_WITH_RETENTION: {
    /** @type {number} */
    const fileToDelete = await fileInput.handle(state);
    const selectedFile = files[fileToDelete];

```

```

    try {
      await client.send(
        new DeleteObjectCommand({
          Bucket: selectedFile.bucket,
          Key: selectedFile.key,
          VersionId: selectedFile.version,
          BypassGovernanceRetention: true,
        })),
    );
    state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
  } catch (err) {
    state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
  }
  break;
}

case choices.OVERWRITE_FILE: {
  /** @type {number} */
  const fileToOverwrite = await fileInput.handle(state);
  const selectedFile = files[fileToOverwrite];
  try {
    await client.send(
      new PutObjectCommand({
        Bucket: selectedFile.bucket,
        Key: selectedFile.key,
        Body: "New content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      })),
    );
    state.replOutput = `Overwrote ${selectedFile.key} in
${selectedFile.bucket}.`;
  } catch (err) {
    state.replOutput = `Unable to overwrite object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
  }
  break;
}

case choices.VIEW_RETENTION_SETTINGS: {
  /** @type {number} */
  const fileToView = await fileInput.handle(state);
  const selectedFile = files[fileToView];
  try {
    const retention = await client.send(

```

```

        new GetObjectRetentionCommand({
            Bucket: selectedFile.bucket,
            Key: selectedFile.key,
            VersionId: selectedFile.version,
        })),
    );
    const bucketConfig = await client.send(
        new GetObjectLockConfigurationCommand({
            Bucket: selectedFile.bucket,
        })),
    );
    state.replOutput = `Object retention for ${selectedFile.key}
in ${selectedFile.bucket}: ${retention.Retention?.Mode} until
${retention.Retention?.RetainUntilDate?.toISOString()}.
Bucket object lock config for ${selectedFile.bucket} in ${selectedFile.bucket}:
Enabled: ${bucketConfig.ObjectLockConfiguration?.ObjectLockEnabled}
Rule:
${JSON.stringify(bucketConfig.ObjectLockConfiguration?.Rule?.DefaultRetention)}`;
    } catch (err) {
        state.replOutput = `Unable to fetch object lock retention:
'${err.message}'`;
    }
    break;
}
case choices.VIEW_LEGAL_HOLD_SETTINGS: {
    /** @type {number} */
    const fileToView = await fileInput.handle(state);
    const selectedFile = files[fileToView];
    try {
        const legalHold = await client.send(
            new GetObjectLegalHoldCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
            })),
        );
        state.replOutput = `Object legal hold for ${selectedFile.key} in
${selectedFile.bucket}: Status: ${legalHold.LegalHold?.Status}`;
    } catch (err) {
        state.replOutput = `Unable to fetch legal hold: '${err.message}'`;
    }
    break;
}
default:

```

```

        throw new Error(`Invalid replChoice: ${replChoice}`);
    }
},
{
    whileConfig: {
        whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
        input: replInput(scenarios),
        output: new scenarios.ScenarioOutput(
            "REPL output",
            (state) => state.replOutput,
            { preformatted: true },
        ),
    },
},
);

export { replInput, replAction, choices };

```

생성된 모든 리소스를 삭제합니다(clean.steps.js).

```

import {
    DeleteObjectCommand,
    DeleteBucketCommand,
    ListObjectVersionsCommand,
    GetObjectLegalHoldCommand,
    GetObjectRetentionCommand,
    PutObjectLegalHoldCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) => {
    new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {

```

```

        type: "confirm",
    });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const cleanupAction = (scenarios, client) =>
    new scenarios.ScenarioAction("cleanupAction", async (state) => {
        const { noLockBucketName, lockEnabledBucketName, retentionBucketName } =
            state;

        const buckets = [
            noLockBucketName,
            lockEnabledBucketName,
            retentionBucketName,
        ];

        for (const bucket of buckets) {
            /** @type {import("@aws-sdk/client-s3").ListObjectVersionsCommandOutput} */
            let objectsResponse;

            try {
                objectsResponse = await client.send(
                    new ListObjectVersionsCommand({
                        Bucket: bucket,
                    }),
                );
            } catch (e) {
                if (e instanceof Error && e.name === "NoSuchBucket") {
                    console.log("Object's bucket has already been deleted.");
                    continue;
                }
                throw e;
            }

            for (const version of objectsResponse.Versions || []) {
                const { Key, VersionId } = version;

                try {
                    const legalHold = await client.send(
                        new GetObjectLegalHoldCommand({
                            Bucket: bucket,
                            Key,

```



```
        VersionId,
      )),
    );

    if (legalHold.LegalHold?.Status === "ON") {
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: bucket,
          Key,
          VersionId,
          LegalHold: {
            Status: "OFF",
          },
        )),
      );
    }
  } catch (err) {
    console.log(
      `Unable to fetch legal hold for ${Key} in ${bucket}:`
    );
  }

  try {
    const retention = await client.send(
      new GetObjectRetentionCommand({
        Bucket: bucket,
        Key,
        VersionId,
      )),
    );

    if (retention.Retention?.Mode === "GOVERNANCE") {
      await client.send(
        new DeleteObjectCommand({
          Bucket: bucket,
          Key,
          VersionId,
          BypassGovernanceRetention: true,
        )),
      );
    }
  } catch (err) {
    console.log(
```

```

        `Unable to fetch object lock retention for ${Key} in ${bucket}:
        '${err.message}'`,
        );
    }

    await client.send(
        new DeleteObjectCommand({
            Bucket: bucket,
            Key,
            VersionId,
        }),
    );
}

await client.send(new DeleteBucketCommand({ Bucket: bucket }));
console.log(`Delete for ${bucket} complete.`);
}
});

export { confirmCleanup, cleanupAction };

```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 다음 주제를 참조하세요.
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon S3 조건부 요청 수행

다음 코드 예제에서는 Amazon S3 요청에 사전 조건을 추가하는 방법을 보여줍니다.

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 조건부 요청 기능을 시연하는 대화형 시나리오를 실행합니다.

```
using Amazon.S3;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace S3ConditionalRequestsScenario;

public static class S3ConditionalRequestsScenario
{
    /*
        Before running this .NET code example, set up your development environment,
        including your credentials.

        This example demonstrates the use of conditional requests for S3 operations.
        You can use conditional requests to add preconditions to S3 read requests to
        return or copy
        an object based on its Entity tag (ETag), or last modified date.
        You can use a conditional write requests to prevent overwrites by ensuring
        there is no existing object with the same key.
    */

    public static S3ActionsWrapper _s3ActionsWrapper = null!;
    public static IConfiguration _configuration = null!;
    public static string _resourcePrefix = null!;
    public static string _sourceBucketName = null!;
    public static string _destinationBucketName = null!;
    public static string _sampleObjectKey = null!;
```

```

public static string _sampleObjectEtag = null!;
public static bool _interactive = true;

public static async Task Main(string[] args)
{
    // Set up dependency injection for the Amazon service.
    using var host = Host.CreateDefaultBuilder(args)
        .ConfigureLogging(logging =>
            logging.AddFilter("System", LogLevel.Debug)
                .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
        .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonS3>()
                .AddTransient<S3ActionsWrapper>()
        )
        .Build();

    _configuration = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("settings.json") // Load settings from .json file.
        .AddJsonFile("settings.local.json",
            true) // Optionally, load local settings.
        .Build();

    ServicesSetup(host);

    try
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon Simple Storage Service (S3)
Conditional Requests Feature Scenario.");
        Console.WriteLine(new string('-', 80));
        ConfigurationSetup();
        _sampleObjectEtag = await Setup(_sourceBucketName,
            _destinationBucketName, _sampleObjectKey);

        await DisplayDemoChoices(_sourceBucketName, _destinationBucketName,
            _sampleObjectKey, _sampleObjectEtag, 0);

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Cleaning up resources.");
    }
}

```

```

        Console.WriteLine(new string('-', 80));
        await Cleanup(true);

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Amazon S3 Conditional Requests Feature Scenario is
complete.");
        Console.WriteLine(new string('-', 80));
    }
    catch (Exception ex)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"There was a problem: {ex.Message}");
        await CleanupScenario(_sourceBucketName, _destinationBucketName);
        Console.WriteLine(new string('-', 80));
    }
}

/// <summary>
/// Populate the services for use within the console application.
/// </summary>
/// <param name="host">The services host.</param>
private static void ServicesSetup(IHost host)
{
    _s3ActionsWrapper = host.Services.GetRequiredService<S3ActionsWrapper>();
}

/// <summary>
/// Any setup operations needed.
/// </summary>
public static void ConfigurationSetup()
{
    _resourcePrefix = _configuration["resourcePrefix"] ?? "dotnet-example";

    _sourceBucketName = _resourcePrefix + "-source";
    _destinationBucketName = _resourcePrefix + "-dest";
    _sampleObjectKey = _resourcePrefix + "-sample-object.txt";
}

/// <summary>
/// Sets up the scenario by creating a source and destination bucket, and
uploading a test file to the source bucket.
/// </summary>
/// <param name="sourceBucket">The name of the source bucket.</param>
/// <param name="destBucket">The name of the destination bucket.</param>

```

```

    /// <param name="objectKey">The name of the test file to add to the source
    bucket.</param>
    /// <returns>The ETag of the uploaded test file.</returns>
    public static async Task<string> Setup(string sourceBucket, string
    destBucket, string objectKey)
    {
        Console.WriteLine(
            "\nFor this scenario, we will use the AWS SDK for .NET to create
    several S3\n" +
            "buckets and files to demonstrate working with S3 conditional
    requests.\n" +
            "This example demonstrates the use of conditional requests for S3
    operations.\r\n" +
            "You can use conditional requests to add preconditions to S3 read
    requests to return or copy\r\n" +
            "an object based on its Entity tag (ETag), or last modified date. \r
    \n" +
            "You can use a conditional write requests to prevent overwrites by
    ensuring \r\n" +
            "there is no existing object with the same key. \r\n\r\n" +
            "This example will allow you to perform conditional reads\r\n" +
            "and writes that will succeed or fail based on your selected options.
    \r\n\r\n" +
            "Sample buckets and a sample object will be created as part of the
    example.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you are ready to start.");
        if (_interactive)
            Console.ReadLine();

        await _s3ActionsWrapper.CreateBucketWithName(sourceBucket);
        await _s3ActionsWrapper.CreateBucketWithName(destBucket);

        var eTag = await _s3ActionsWrapper.PutObjectConditional(objectKey,
        sourceBucket,
            "Test file content.");

        return eTag;
    }

    /// <summary>
    /// Cleans up the scenario by deleting the source and destination buckets.
    /// </summary>

```

```
/// <param name="sourceBucket">The name of the source bucket.</param>
/// <param name="destBucket">The name of the destination bucket.</param>
public static async Task CleanupScenario(string sourceBucket, string
destBucket)
{
    await _s3ActionsWrapper.CleanupBucketByName(sourceBucket);
    await _s3ActionsWrapper.CleanupBucketByName(destBucket);
}

/// <summary>
/// Displays a list of the objects in the test buckets.
/// </summary>
/// <param name="sourceBucket">The name of the source bucket.</param>
/// <param name="destBucket">The name of the destination bucket.</param>
public static async Task DisplayBuckets(string sourceBucket, string
destBucket)
{
    await _s3ActionsWrapper.ListBucketContentsByName(sourceBucket);
    await _s3ActionsWrapper.ListBucketContentsByName(destBucket);
}

/// <summary>
/// Displays the menu of conditional request options for the user.
/// </summary>
/// <param name="sourceBucket">The name of the source bucket.</param>
/// <param name="destBucket">The name of the destination bucket.</param>
/// <param name="objectKey">The key of the test object in the source
bucket.</param>
/// <param name="etag">The ETag of the test object in the source bucket.</
param>
public static async Task DisplayDemoChoices(string sourceBucket, string
destBucket, string objectKey, string etag, int defaultChoice)
{
    var actions = new[]
    {
        "Print a list of bucket items.",
        "Perform a conditional read.",
        "Perform a conditional copy.",
        "Perform a conditional write.",
        "Clean up and exit."
    };

    var conditions = new[]
    {
```

```

        "If-Match: using the object's ETag. This condition should succeed.",
        "If-None-Match: using the object's ETag. This condition should
fail.",
        "If-Modified-Since: using yesterday's date. This condition should
succeed.",
        "If-Unmodified-Since: using yesterday's date. This condition should
fail."
    };

    var conditionTypes = new[]
    {
        S3ConditionType.IfMatch,
        S3ConditionType.IfNoneMatch,
        S3ConditionType.IfModifiedSince,
        S3ConditionType.IfUnmodifiedSince,
    };

    var yesterdayDate = DateTime.UtcNow.AddDays(-1);

    int choice;
    while ((choice = GetChoiceResponse("\nExplore the S3 conditional request
features by selecting one of the following choices:", actions, defaultChoice)) != 4)
    {
        switch (choice)
        {
            case 0:
                Console.WriteLine("Listing the objects and buckets.");
                await DisplayBuckets(sourceBucket, destBucket);
                break;
            case 1:
                int conditionTypeIndex = GetChoiceResponse("Perform a
conditional read:", conditions, 1);
                if (conditionTypeIndex == 0 || conditionTypeIndex == 1)
                {
                    await _s3ActionsWrapper.GetObjectConditional(objectKey,
sourceBucket, conditionTypes[conditionTypeIndex], null, _sampleObjectEtag);
                }
                else if (conditionTypeIndex == 2 || conditionTypeIndex == 3)
                {
                    await _s3ActionsWrapper.GetObjectConditional(objectKey,
sourceBucket, conditionTypes[conditionTypeIndex], yesterdayDate);
                }
                break;
        }
    }

```



```

        case 2:
            int copyConditionTypeIndex = GetChoiceResponse("Perform a
conditional copy:", conditions, 1);
            string destKey = GetStringResponse("Enter an object key:",
"sampleObjectKey");
            if (copyConditionTypeIndex == 0 || copyConditionTypeIndex ==
1)
            {
                await _s3ActionsWrapper.CopyObjectConditional(objectKey,
destKey, sourceBucket, destBucket, conditionTypes[copyConditionTypeIndex], null,
etag);
            }
            else if (copyConditionTypeIndex == 2 ||
copyConditionTypeIndex == 3)
            {
                await _s3ActionsWrapper.CopyObjectConditional(objectKey,
destKey, sourceBucket, destBucket, conditionTypes[copyConditionTypeIndex],
yesterdayDate);
            }
            break;
        case 3:
            Console.WriteLine("Perform a conditional write using
IfNoneMatch condition on the object key.");
            Console.WriteLine("If the key is a duplicate, the write will
fail.");
            string newObjectKey = GetStringResponse("Enter an object
key:", "newObjectKey");
            await _s3ActionsWrapper.PutObjectConditional(newObjectKey,
sourceBucket, "Conditional write example data.");
            break;
    }

    if (!_interactive)
    {
        break;
    }

    Console.WriteLine("Proceeding to cleanup.");
}

// <summary>
/// Clean up the resources from the scenario.
/// </summary>

```

```

    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Cleanup(bool interactive)
    {
        Console.WriteLine(new string('-', 80));

        if (!interactive || GetYesNoResponse("Do you want to clean up all files
and buckets? (y/n) "))
        {
            await _s3ActionsWrapper.CleanUpBucketByName(_sourceBucketName);
            await _s3ActionsWrapper.CleanUpBucketByName(_destinationBucketName);

        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Helper method to get a choice response from the user.
    /// </summary>

```

```

    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="choices">The choices to print on the console.</param>
    /// <returns>The index of the selected choice</returns>
    private static int GetChoiceResponse(string? question, string[] choices, int
defaultChoice)
    {
        if (question != null)
        {
            Console.WriteLine(question);

            for (int i = 0; i < choices.Length; i++)
            {
                Console.WriteLine($"{i + 1}. {choices[i]}");
            }
        }

        if (!_interactive)
            return defaultChoice;

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.Length)
        {
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        return choiceNumber - 1;
    }

    /// <summary>
    /// Get a string response from the user.
    /// </summary>
    /// <param name="question">The question to print.</param>
    /// <param name="defaultAnswer">A default answer to use when not
interactive.</param>
    /// <returns>The string response.</returns>
    public static string GetStringResponse(string? question, string
defaultAnswer)
    {
        string? answer = "";
        if (!_interactive)
        {
            do

```

```

        {
            Console.WriteLine(question);
            answer = Console.ReadLine();
        } while (string.IsNullOrEmpty(answer));
    }
    else
    {
        answer = defaultAnswer;
    }

    return answer;
}
}

```

S3 함수의 래퍼 클래스입니다.

```

using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Logging;

namespace S3ConditionalRequestsScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper
{
    private readonly IAmazonS3 _amazonS3;
    private readonly ILogger<S3ActionsWrapper> _logger;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
    /// <param name="logger">The class logger.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, ILogger<S3ActionsWrapper> logger)
    {
        _amazonS3 = amazonS3;
        _logger = logger;
    }
}

```

```

    /// <summary>
    /// Retrieves an object from Amazon S3 with a conditional request.
    /// </summary>
    /// <param name="objectKey">The key of the object to retrieve.</param>
    /// <param name="sourceBucket">The source bucket of the object.</param>
    /// <param name="conditionType">The type of condition: 'IfMatch',
    'IfNoneMatch', 'IfModifiedSince', 'IfUnmodifiedSince'.</param>
    /// <param name="conditionDateValue">The value to use for the condition for
    dates.</param>
    /// <param name="etagConditionalValue">The value to use for the condition for
    etags.</param>
    /// <returns>True if the conditional read is successful, False otherwise.</
returns>
    public async Task<bool> GetObjectConditional(string objectKey, string
sourceBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null,
string? etagConditionalValue = null)
    {
        try
        {
            var getObjectRequest = new GetObjectRequest
            {
                BucketName = sourceBucket,
                Key = objectKey
            };

            switch (conditionType)
            {
                case S3ConditionType.IfMatch:
                    getObjectRequest.EtagToMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfNoneMatch:
                    getObjectRequest.EtagToNotMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfModifiedSince:
                    getObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
                case S3ConditionType.IfUnmodifiedSince:
                    getObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
                default:

```

```

        throw new ArgumentOutOfRangeException(nameof(conditionType),
            conditionType, null);
    }

    var response = await _amazonS3.GetObjectAsync(getObjectRequest);
    var sampleBytes = new byte[20];
    await response.ResponseStream.ReadAsync(sampleBytes, 0, 20);
    _logger.LogInformation($"Conditional read
successful. Here are the first 20 bytes of the object:
\n{System.Text.Encoding.UTF8.GetString(sampleBytes)}");
    return true;
}
catch (AmazonS3Exception e)
{
    if (e.ErrorCode == "PreconditionFailed")
    {
        _logger.LogError("Conditional read failed: Precondition failed");
    }
    else if (e.ErrorCode == "NotModified")
    {
        _logger.LogError("Conditional read failed: Object not modified");
    }
    else
    {
        _logger.LogError($"Unexpected error: {e.ErrorCode}");
        throw;
    }
    return false;
}
}

/// <summary>
/// Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite using an IfNoneMatch condition for the object key.
/// </summary>
/// <param name="objectKey">The key of the object to upload.</param>
/// <param name="bucket">The source bucket of the object.</param>
/// <param name="content">The content to upload as a string.</param>
/// <returns>The ETag if the conditional write is successful, empty
otherwise.</returns>
public async Task<string> PutObjectConditional(string objectKey, string
bucket, string content)
{
    try

```

```

    {
        var putObjectRequest = new PutObjectRequest
        {
            BucketName = bucket,
            Key = objectKey,
            ContentBody = content,
            IfNoneMatch = "*"
        };

        var putResult = await _amazonS3.PutObjectAsync(putObjectRequest);
        _logger.LogInformation($"Conditional write successful for key
{objectKey} in bucket {bucket}.");
        return putResult.ETag;
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "PreconditionFailed")
        {
            _logger.LogError("Conditional write failed: Precondition
failed");
        }
        else
        {
            _logger.LogError($"Unexpected error: {e.ErrorCode}");
            throw;
        }
        return string.Empty;
    }
}

/// <summary>
/// Copies an object from one Amazon S3 bucket to another with a conditional
request.
/// </summary>
/// <param name="sourceKey">The key of the source object to copy.</param>
/// <param name="destKey">The key of the destination object.</param>
/// <param name="sourceBucket">The source bucket of the object.</param>
/// <param name="destBucket">The destination bucket of the object.</param>
/// <param name="conditionType">The type of condition to apply, e.g.
'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
'CopySourceIfUnmodifiedSince'.</param>
/// <param name="conditionDateValue">The value to use for the condition for
dates.</param>

```

```
/// <param name="etagConditionalValue">The value to use for the condition for
etags.</param>
/// <returns>True if the conditional copy is successful, False otherwise.</
returns>
public async Task<bool> CopyObjectConditional(string sourceKey, string
destKey, string sourceBucket, string destBucket,
    S3ConditionType conditionType, DateTime? conditionDateValue = null,
string? etagConditionalValue = null)
{
    try
    {
        var copyObjectRequest = new CopyObjectRequest
        {
            DestinationBucket = destBucket,
            DestinationKey = destKey,
            SourceBucket = sourceBucket,
            SourceKey = sourceKey
        };

        switch (conditionType)
        {
            case S3ConditionType.IfMatch:
                copyObjectRequest.ETagToMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfNoneMatch:
                copyObjectRequest.ETagToNotMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfModifiedSince:
                copyObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                break;
            case S3ConditionType.IfUnmodifiedSince:
                copyObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                break;
            default:
                throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
        }

        await _amazonS3.CopyObjectAsync(copyObjectRequest);
        _logger.LogInformation($"Conditional copy successful for key
{destKey} in bucket {destBucket}.");
        return true;
    }
}
```



```
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "PreconditionFailed")
        {
            _logger.LogError("Conditional copy failed: Precondition failed");
        }
        else if (e.ErrorCode == "304")
        {
            _logger.LogError("Conditional copy failed: Object not modified");
        }
        else
        {
            _logger.LogError($"Unexpected error: {e.ErrorCode}");
            throw;
        }
        return false;
    }
}

/// <summary>
/// Create a new Amazon S3 bucket with a specified name and check that the
bucket is ready.
/// </summary>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <returns>True if successful.</returns>
public async Task<bool> CreateBucketWithName(string bucketName)
{
    Console.WriteLine($"\\tCreating bucket {bucketName}.");
    try
    {
        var request = new PutBucketRequest
        {
            BucketName = bucketName,
            UseClientRegion = true
        };

        await _amazonS3.PutBucketAsync(request);
        var bucketReady = false;
        var retries = 5;
        while (!bucketReady && retries > 0)
        {
            Thread.Sleep(5000);
        }
    }
}
```

```

        bucketReady = await
Amazon.S3.Util.AmazonS3Util.DoesS3BucketExistV2Async(_amazonS3, bucketName);
        retries--;
    }

    return bucketReady;
}
catch (BucketAlreadyExistsException ex)
{
    Console.WriteLine($"Bucket already exists: '{ex.Message}'");
    return true;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error creating bucket: '{ex.Message}'");
    return false;
}
}

/// <summary>
/// Cleans up objects and deletes the bucket by name.
/// </summary>
/// <param name="bucketName">The name of the bucket.</param>
/// <returns>Async task.</returns>
public async Task CleanupBucketByName(string bucketName)
{
    try
    {
        var listObjectsResponse = await _amazonS3.ListObjectsV2Async(new
ListObjectsV2Request { BucketName = bucketName });
        foreach (var obj in listObjectsResponse.S3Objects)
        {
            await _amazonS3.DeleteObjectAsync(new DeleteObjectRequest
{ BucketName = bucketName, Key = obj.Key });
        }
        await _amazonS3.DeleteBucketAsync(new DeleteBucketRequest
{ BucketName = bucketName });
        Console.WriteLine($"Cleaned up bucket: {bucketName}.");
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "NoSuchBucket")
        {

```

```
        Console.WriteLine($"Bucket {bucketName} does not exist, skipping
cleanup.");
    }
    else
    {
        Console.WriteLine($"Error deleting bucket: {e.ErrorCode}");
        throw;
    }
}

}

/// <summary>
/// List the contents of the bucket with their ETag.
/// </summary>
/// <param name="bucketName">The name of the bucket.</param>
/// <returns>Async task.</returns>
public async Task<List<S3Object>> ListBucketContentsByName(string bucketName)
{
    var results = new List<S3Object>();
    try
    {
        Console.WriteLine($"\\t Items in bucket {bucketName}");
        var listObjectsResponse = await _amazonS3.ListObjectsV2Async(new
ListObjectsV2Request { BucketName = bucketName });
        if (listObjectsResponse.S3Objects.Count == 0)
        {
            Console.WriteLine("\\t\\tNo objects found.");
        }
        else
        {
            foreach (var obj in listObjectsResponse.S3Objects)
            {
                Console.WriteLine($"\\t\\t object: {obj.Key} ETag {obj.ETag}");
            }
        }
        results = listObjectsResponse.S3Objects;
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "NoSuchBucket")
        {
            _logger.LogError($"Bucket {bucketName} does not exist.");
        }
    }
}
```

```

        else
        {
            _logger.LogError($"Error listing bucket and objects:
{e.ErrorCode}");
            throw;
        }
    }

    return results;
}

/// <summary>
/// Delete an object from a specific bucket.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to use.</param>
/// <param name="objectKey">The key of the object to delete.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteObjectFromBucket(string bucketName, string
objectKey)
{
    try
    {
        var request = new DeleteObjectRequest()
        {
            BucketName = bucketName,
            Key = objectKey
        };
        await _amazonS3.DeleteObjectAsync(request);
        Console.WriteLine($"Deleted {objectKey} in {bucketName}.");
        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Unable to delete object {objectKey} in bucket
{bucketName}: " + ex.Message);
        return false;
    }
}

/// <summary>
/// Delete a specific bucket by deleting the objects and then the bucket
itself.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to use.</param>

```

```
/// <param name="objectKey">The key of the object to delete.</param>
/// <param name="versionId">Optional versionId.</param>
/// <returns>True if successful.</returns>
public async Task<bool> CleanUpBucketByName(string bucketName)
{
    try
    {
        var allFiles = await ListBucketContentsByName(bucketName);

        foreach (var fileInfo in allFiles)
        {
            await DeleteObjectFromBucket(fileInfo.BucketName, fileInfo.Key);
        }

        var request = new DeleteBucketRequest() { BucketName = bucketName, };
        var response = await _amazonS3.DeleteBucketAsync(request);
        Console.WriteLine($"\\tDelete for {bucketName} complete.");
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to delete bucket {bucketName}: " +
            ex.Message);
        return false;
    }
}
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 다음 항목을 참조하세요.
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

워크플로의 진입점입니다(index.js). 모든 단계를 오케스트레이션합니다. GitHub를 방문하여 Scenario, ScenarioInput, ScenarioOutput, ScenarioAction에 대한 구현 세부 정보를 확인하세요.

```
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
  loadState,
  saveState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "../welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
} from "../setup.steps.js";

/**
 * @param {Scenarios} scenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});

  return {
    deploy: new scenarios.Scenario(
      "S3 Conditional Requests - Deploy",
```

```

    [
      welcome(scenarios),
      welcomeContinue(scenarios),
      exitOnFalse(scenarios, "welcomeContinue"),
      getBucketPrefix(scenarios),
      createBuckets(scenarios),
      confirmCreateBuckets(scenarios),
      exitOnFalse(scenarios, "confirmCreateBuckets"),
      createBucketsAction(scenarios, client),
      populateBuckets(scenarios),
      confirmPopulateBuckets(scenarios),
      exitOnFalse(scenarios, "confirmPopulateBuckets"),
      populateBucketsAction(scenarios, client),
      saveState,
    ],
    initialState,
  ),
  demo: new scenarios.Scenario(
    "S3 Conditional Requests - Demo",
    [loadState, welcome(scenarios), replAction(scenarios, client)],
    initialState,
  ),
  clean: new scenarios.Scenario(
    "S3 Conditional Requests - Destroy",
    [
      loadState,
      confirmCleanup(scenarios),
      exitOnFalse(scenarios, "confirmCleanup"),
      cleanupAction(scenarios, client),
    ],
    initialState,
  ),
};
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const objectLockingScenarios = getWorkflowStages(Scenarios);
  Scenarios.parseScenarioArgs(objectLockingScenarios, {

```

```

    name: "Amazon S3 object locking workflow",
    description:
      "Work with Amazon Simple Storage Service (Amazon S3) object locking
      features.",
    synopsis:
      "node index.js --scenario <deploy | demo | clean> [-h|--help] [-y|--yes] [-v|--verbose]",
  });
}

```

환영 메시지를 콘솔에 출력합니다(welcome.steps.js).

```

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @param {Scenarios} scenarios
 */
const welcome = (scenarios) =>
  new scenarios.ScenarioOutput(
    "welcome",
    "This example demonstrates the use of conditional requests for S3
    operations." +
    " You can use conditional requests to add preconditions to S3 read requests
    to return " +
    "or copy an object based on its Entity tag (ETag), or last modified
    date.You can use " +
    "a conditional write requests to prevent overwrites by ensuring there is no
    existing " +
    "object with the same key.\n" +
    "This example will enable you to perform conditional reads and writes that
    will succeed " +
    "or fail based on your selected options.\n" +
    "Sample buckets and a sample object will be created as part of the example.
    \n" +
    "Some steps require a key name prefix to be defined by the user. Before you
    begin, you can " +
    "optionally edit this prefix in ./object_name.json. If you do so, please
    reload the scenario before you begin.",
    { header: true },
  );

```



```

/**
 * @param {Scenarios} scenarios
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };

```

버킷 및 객체를 배포합니다(setup.steps.js).

```

import {
  ChecksumAlgorithm,
  CreateBucketCommand,
  PutObjectCommand,
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
  S3ServiceException,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const getBucketPrefix = (scenarios) =>
  new scenarios.ScenarioInput(
    "bucketPrefix",
    "Provide a prefix that will be used for bucket creation.",
    { type: "input", default: "amzn-s3-demo-bucket" },
  );

```

```
/**
 * @param {Scenarios} scenarios
 */
const createBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "createBuckets",
    (state) => `The following buckets will be created:
      ${state.bucketPrefix}-source-bucket.
      ${state.bucketPrefix}-destination-bucket.`
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmCreateBuckets = (scenarios) =>
  new scenarios.ScenarioInput("confirmCreateBuckets", "Create the buckets?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("createBucketsAction", async (state) => {
    const sourceBucketName = `${state.bucketPrefix}-source-bucket`;
    const destinationBucketName = `${state.bucketPrefix}-destination-bucket`;

    try {
      await client.send(
        new CreateBucketCommand({
          Bucket: sourceBucketName,
        }),
      );
      await waitUntilBucketExists({ client }, { Bucket: sourceBucketName });
      await client.send(
        new CreateBucketCommand({
          Bucket: destinationBucketName,
        }),
      );
      await waitUntilBucketExists(
        { client },
        { Bucket: destinationBucketName },
      );
    }
  });
```

```

    );

    state.sourceBucketName = sourceBucketName;
    state.destinationBucketName = destinationBucketName;
  } catch (caught) {
    if (
      caught instanceof BucketAlreadyExists ||
      caught instanceof BucketAlreadyOwnedByYou
    ) {
      console.error(`${caught.name}: ${caught.message}`);
      state.earlyExit = true;
    } else {
      throw caught;
    }
  }
});

/**
 * @param {Scenarios} scenarios
 */
const populateBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "populateBuckets",
    (state) => `The following test files will be created:
      file01.txt in ${state.bucketPrefix}-source-bucket.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmPopulateBuckets = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmPopulateBuckets",
    "Populate the buckets?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const populateBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("populateBucketsAction", async (state) => {

```

```

    try {
      await client.send(
        new PutObjectCommand({
          Bucket: state.sourceBucketName,
          Key: "file01.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
    } catch (caught) {
      if (caught instanceof S3ServiceException) {
        console.error(
          `Error from S3 while uploading object.  ${caught.name}:
${caught.message}`,
        );
      } else {
        throw caught;
      }
    }
  });
});

export {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
};

```

S3 조건부 요청을 사용하여 객체를 가져오고, 복사하고, 업로드합니다(repl.steps.js).

```

import path from "node:path";
import { fileURLToPath } from "node:url";
import { dirname } from "node:path";

import {
  ListObjectVersionsCommand,
  GetObjectCommand,
  CopyObjectCommand,

```

```

    PutObjectCommand,
  } from "@aws-sdk/client-s3";
import data from "./object_name.json" assert { type: "json" };
import { readFile } from "node:fs/promises";
import {
  ScenarioInput,
  Scenario,
  ScenarioAction,
  ScenarioOutput,
} from "../../libs/scenario/index.js";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const choices = {
  EXIT: 0,
  LIST_ALL_FILES: 1,
  CONDITIONAL_READ: 2,
  CONDITIONAL_COPY: 3,
  CONDITIONAL_WRITE: 4,
};

/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new ScenarioInput(
    "replChoice",
    "Explore the S3 conditional request features by selecting one of the following choices",
    {
      type: "select",
      choices: [
        { name: "Print list of bucket items.", value: choices.LIST_ALL_FILES },
        {
          name: "Perform a conditional read.",
          value: choices.CONDITIONAL_READ,
        },
      ],
    }
  )

```

```

        name: "Perform a conditional copy. These examples use the key name
        prefix defined in ./object_name.json.",
        value: choices.CONDITIONAL_COPY,
    },
    {
        name: "Perform a conditional write. This example use the sample file ./
        text02.txt.",
        value: choices.CONDITIONAL_WRITE,
    },
    { name: "Finish the workflow.", value: choices.EXIT },
],
},
);

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */
const getAllFiles = async (client, buckets) => {
    /** @type {{bucket: string, key: string, version: string}[]} */
    const files = [];
    for (const bucket of buckets) {
        const objectsResponse = await client.send(
            new ListObjectVersionsCommand({ Bucket: bucket })),
        );
        for (const version of objectsResponse.Versions || []) {
            const { Key } = version;
            files.push({ bucket, key: Key });
        }
    }
    return files;
};

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 * @param {string} key
 */
const getEtag = async (client, bucket, key) => {
    const objectsResponse = await client.send(
        new GetObjectCommand({
            Bucket: bucket,
            Key: key,
        })),
    );

```

```

    );
    return objectsResponse.ETag;
};

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
export const replAction = (scenarios, client) =>
  new ScenarioAction(
    "replAction",
    async (state) => {
      const files = await getAllFiles(client, [
        state.sourceBucketName,
        state.destinationBucketName,
      ]);

      const fileInput = new scenarios.ScenarioInput(
        "selectedFile",
        "Select a file to use:",
        {
          type: "select",
          choices: files.map((file, index) => ({
            name: `${index + 1}: ${file.bucket}: ${file.key} (Etag: ${
              file.version
            })`,
            value: index,
          })),
        },
      );

      const condReadOptions = new scenarios.ScenarioInput(
        "selectOption",
        "Which conditional read action would you like to take?",
        {
          type: "select",
          choices: [
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should
fail.",

```

```

        "If-Modified-Since: using yesterday's date. This condition should
        succeed.",
        "If-Unmodified-Since: using yesterday's date. This condition should
        fail.",
    ],
    },
);
const condCopyOptions = new scenarios.ScenarioInput(
    "selectOption",
    "Which conditional copy action would you like to take?",
    {
        type: "select",
        choices: [
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should
            fail.",
            "If-Modified-Since: using yesterday's date. This condition should
            succeed.",
            "If-Unmodified-Since: using yesterday's date. This condition should
            fail.",
        ],
    },
);
const condWriteOptions = new scenarios.ScenarioInput(
    "selectOption",
    "Which conditional write action would you like to take?",
    {
        type: "select",
        choices: [
            "IfNoneMatch condition on the object key: If the key is a duplicate,
            the write will fail.",
        ],
    },
);

const { replChoice } = state;

switch (replChoice) {
    case choices.LIST_ALL_FILES: {
        const files = await getAllFiles(client, [
            state.sourceBucketName,
            state.destinationBucketName,
        ]);
        state.replOutput = files

```



```

        .map(
            (file) => `Items in bucket ${file.bucket}: object: ${file.key} `,
        )
        .join("\n");
    break;
}
case choices.CONDITIONAL_READ:
{
    const selectedCondRead = await condReadOptions.handle(state);
    if (
        selectedCondRead ===
        "If-Match: using the object's ETag. This condition should succeed."
    ) {
        const bucket = state.sourceBucketName;
        const key = "file01.txt";
        const ETag = await getEtag(client, bucket, key);

        try {
            await client.send(
                new GetObjectCommand({
                    Bucket: bucket,
                    Key: key,
                    IfMatch: ETag,
                }),
            );
            state.replOutput = `${key} in bucket ${state.sourceBucketName}
read because ETag provided matches the object's ETag.`;
        } catch (err) {
            state.replOutput = `Unable to read object ${key} in bucket
${state.sourceBucketName}: ${err.message}`;
        }
        break;
    }
    if (
        selectedCondRead ===
        "If-None-Match: using the object's ETag. This condition should
fail."
    ) {
        const bucket = state.sourceBucketName;
        const key = "file01.txt";
        const ETag = await getEtag(client, bucket, key);

        try {
            await client.send(

```

```

        new GetObjectCommand({
            Bucket: bucket,
            Key: key,
            IfNoneMatch: ETag,
        })),
    );
    state.replOutput = `${key} in ${state.sourceBucketName} was
returned.`;
    } catch (err) {
        state.replOutput = `${key} in ${state.sourceBucketName} was not
read: ${err.message}`;
    }
    break;
}
if (
    selectedCondRead ===
    "If-Modified-Since: using yesterday's date. This condition should
succeed."
) {
    const date = new Date();
    date.setDate(date.getDate() - 1);

    const bucket = state.sourceBucketName;
    const key = "file01.txt";
    try {
        await client.send(
            new GetObjectCommand({
                Bucket: bucket,
                Key: key,
                IfModifiedSince: date,
            })),
        );
        state.replOutput = `${key} in bucket ${state.sourceBucketName}
read because it has been created or modified in the last 24 hours.`;
    } catch (err) {
        state.replOutput = `Unable to read object ${key} in bucket
${state.sourceBucketName}: ${err.message}`;
    }
    break;
}
if (
    selectedCondRead ===
    "If-Unmodified-Since: using yesterday's date. This condition should
fail."

```

```

    ) {
      const bucket = state.sourceBucketName;
      const key = "file01.txt";

      const date = new Date();
      date.setDate(date.getDate() - 1);
      try {
        await client.send(
          new GetObjectCommand({
            Bucket: bucket,
            Key: key,
            IfUnmodifiedSince: date,
          }),
        );
        state.replOutput = `${key} in ${state.sourceBucketName} was
read.`;
      } catch (err) {
        state.replOutput = `${key} in ${state.sourceBucketName} was not
read: ${err.message}`;
      }
      break;
    }
  }
  break;
case choices.CONDITIONAL_COPY: {
  const selectedCondCopy = await condCopyOptions.handle(state);
  if (
    selectedCondCopy ===
    "If-Match: using the object's ETag. This condition should succeed."
  ) {
    const bucket = state.sourceBucketName;
    const key = "file01.txt";
    const ETag = await getEtag(client, bucket, key);

    const copySource = `${bucket}/${key}`;
    // Optionally edit the default key name prefix of the copied object
    in ./object_name.json.
    const name = data.name;
    const copiedKey = `${name}${key}`;
    try {
      await client.send(
        new CopyObjectCommand({
          CopySource: copySource,
          Bucket: state.destinationBucketName,

```

```

        Key: copiedKey,
        CopySourceIfMatch: ETag,
    )),
    );
    state.replOutput = `${key} copied as ${copiedKey} to bucket
${state.destinationBucketName} because ETag provided matches the object's
ETag.`;
    } catch (err) {
        state.replOutput = `Unable to copy object ${key} as ${copiedKey} to
bucket ${state.destinationBucketName}: ${err.message}`;
    }
    break;
}
if (
    selectedCondCopy ===
    "If-None-Match: using the object's ETag. This condition should fail."
) {
    const bucket = state.sourceBucketName;
    const key = "file01.txt";
    const ETag = await getEtag(client, bucket, key);
    const copySource = `${bucket}/${key}`;
    // Optionally edit the default key name prefix of the copied object
    in ./object_name.json.
    const name = data.name;
    const copiedKey = `${name}${key}`;

    try {
        await client.send(
            new CopyObjectCommand({
                CopySource: copySource,
                Bucket: state.destinationBucketName,
                Key: copiedKey,
                CopySourceIfNoneMatch: ETag,
            )),
        );
        state.replOutput = `${copiedKey} copied to bucket
${state.destinationBucketName}`;
    } catch (err) {
        state.replOutput = `Unable to copy object as ${key} as as
${copiedKey} to bucket ${state.destinationBucketName}: ${err.message}`;
    }
    break;
}
if (

```

```

        selectedCondCopy ===
        "If-Modified-Since: using yesterday's date. This condition should
succeed."
    ) {
        const bucket = state.sourceBucketName;
        const key = "file01.txt";
        const copySource = `${bucket}/${key}`;
        // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
        const name = data.name;
        const copiedKey = `${name}${key}`;

        const date = new Date();
        date.setDate(date.getDate() - 1);

        try {
            await client.send(
                new CopyObjectCommand({
                    CopySource: copySource,
                    Bucket: state.destinationBucketName,
                    Key: copiedKey,
                    CopySourceIfModifiedSince: date,
                }),
            );
            state.replOutput = `${key} copied as ${copiedKey} to bucket
${state.destinationBucketName} because it has been created or modified in the
last 24 hours.`;
        } catch (err) {
            state.replOutput = `Unable to copy object ${key} as ${copiedKey} to
bucket ${state.destinationBucketName} : ${err.message}`;
        }
        break;
    }
    if (
        selectedCondCopy ===
        "If-Unmodified-Since: using yesterday's date. This condition should
fail."
    ) {
        const bucket = state.sourceBucketName;
        const key = "file01.txt";
        const copySource = `${bucket}/${key}`;
        // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
        const name = data.name;

```

```

const copiedKey = `${name}${key}`;

const date = new Date();
date.setDate(date.getDate() - 1);

try {
  await client.send(
    new CopyObjectCommand({
      CopySource: copySource,
      Bucket: state.destinationBucketName,
      Key: copiedKey,
      CopySourceIfUnmodifiedSince: date,
    }),
  );
  state.replOutput = `${copiedKey} copied to bucket
${state.destinationBucketName} because it has not been created or modified in
the last 24 hours.`;
} catch (err) {
  state.replOutput = `Unable to copy object ${key} to bucket
${state.destinationBucketName}: ${err.message}`;
}
break;
}
case choices.CONDITIONAL_WRITE:
{
  const selectedCondWrite = await condWriteOptions.handle(state);
  if (
    selectedCondWrite ===
    "IfNoneMatch condition on the object key: If the key is a
duplicate, the write will fail."
  ) {
    // Optionally edit the default key name prefix of the copied object
    in ./object_name.json.
    const key = "text02.txt";
    const __filename = fileURLToPath(import.meta.url);
    const __dirname = dirname(__filename);
    const filePath = path.join(__dirname, "text02.txt");
    try {
      await client.send(
        new PutObjectCommand({
          Bucket: `${state.destinationBucketName}`,
          Key: `${key}`,
          Body: await readFile(filePath),

```

```

                IfNoneMatch: "*",
            )),
        );
        state.replOutput = `${key} uploaded to bucket
${state.destinationBucketName} because the key is not a duplicate.`;
        } catch (err) {
            state.replOutput = `Unable to upload object to bucket
${state.destinationBucketName}:${err.message}`;
        }
        break;
    }
}
break;

default:
    throw new Error(`Invalid replChoice: ${replChoice}`);
}
},
{
    whileConfig: {
        whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
        input: replInput(scenarios),
        output: new ScenarioOutput("REPL output", (state) => state.replOutput, {
            preformatted: true,
        }),
    },
},
);

export { replInput, choices };

```

생성된 모든 리소스를 삭제합니다(clean.steps.js).

```

import {
    DeleteObjectCommand,
    DeleteBucketCommand,
    ListObjectVersionsCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

```

```
/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) =>
  new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const cleanupAction = (scenarios, client) =>
  new scenarios.ScenarioAction("cleanupAction", async (state) => {
    const { sourceBucketName, destinationBucketName } = state;
    const buckets = [sourceBucketName, destinationBucketName].filter((b) => b);

    for (const bucket of buckets) {
      try {
        let objectsResponse;
        objectsResponse = await client.send(
          new ListObjectVersionsCommand({
            Bucket: bucket,
          }),
        );
        for (const version of objectsResponse.Versions || []) {
          const { Key, VersionId } = version;
          try {
            await client.send(
              new DeleteObjectCommand({
                Bucket: bucket,
                Key,
                VersionId,
              }),
            );
          } catch (err) {
            console.log(`An error occurred: ${err.message} `);
          }
        }
      }
    }
  })
```



```

    } catch (e) {
      if (e instanceof Error && e.name === "NoSuchBucket") {
        console.log("Objects and buckets have already been deleted.");
        continue;
      }
      throw e;
    }

    await client.send(new DeleteBucketCommand({ Bucket: bucket }));
    console.log(`Delete for ${bucket} complete.`);
  }
});

export { confirmCleanup, cleanupAction };

```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 다음 항목을 참조하세요.
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 조건부 요청을 시연하는 대화형 시나리오를 실행합니다.

```

"""
Purpose

Shows how to use AWS SDK for Python (Boto3) to get started using conditional
requests for
Amazon Simple Storage Service (Amazon S3).

```

```

"""

import logging
import random
import sys
import datetime

import boto3
from botocore.exceptions import ClientError

from s3_conditional_requests import S3ConditionalRequests

# Add relative path to include demo_tools in this code example without need for
# setup.
sys.path.append("../..")
import demo_tools.question as q # noqa

# Constants
FILE_CONTENT = "This is a test file for S3 conditional requests."
RANDOM_SUFFIX = str(random.randint(100, 999))

logger = logging.getLogger(__name__)

class ConditionalRequestsScenario:
    """Runs a scenario that shows how to use S3 Conditional Requests."""

    def __init__(self, conditional_requests, s3_client):
        """
        :param conditional_requests: An object that wraps S3 conditional request
        actions.
        :param s3_client: A Boto3 S3 client for setup and cleanup operations.
        """
        self.conditional_requests = conditional_requests
        self.s3_client = s3_client

    def setup_scenario(self, source_bucket: str, dest_bucket: str, object_key:
str):
        """
        Sets up the scenario by creating a source and destination bucket.
        Prompts the user to provide a bucket name prefix.

        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.

```

```

        :param object_key: The name of a test file to add to the source bucket.
        """

    # Create the buckets.
    try:
        self.s3_client.create_bucket(Bucket=source_bucket)
        self.s3_client.create_bucket(Bucket=dest_bucket)
        print(
            f"Created source bucket: {source_bucket} and destination bucket:
{dest_bucket}"
        )
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        logger.error(f"Error creating buckets: {error_code}")
        raise

    # Upload test file into the source bucket.
    try:
        print(f"Uploading file {object_key} to bucket {source_bucket}")
        response = self.s3_client.put_object(
            Bucket=source_bucket, Key=object_key, Body=FILE_CONTENT
        )
        object_etag = response["ETag"]
        return object_etag

    except Exception as e:
        logger.error(
            f"Failed to upload file {object_key} to bucket {source_bucket}:
{e}"
        )

    def cleanup_scenario(self, source_bucket: str, dest_bucket: str):
        """
        Cleans up the scenario by deleting the source and destination buckets.

        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.
        """
        self.cleanup_bucket(source_bucket)
        self.cleanup_bucket(dest_bucket)

    def cleanup_bucket(self, bucket_name: str):
        """

```

```

Cleans up the bucket by deleting all objects and then the bucket itself.

:param bucket_name: The name of the bucket.
"""
try:
    # Get list of all objects in the bucket.
    list_response = self.s3_client.list_objects_v2(Bucket=bucket_name)
    objs = list_response.get("Contents", [])
    for obj in objs:
        key = obj["Key"]
        self.s3_client.delete_object(Bucket=bucket_name, Key=key)
    self.s3_client.delete_bucket(Bucket=bucket_name)
    print(f"Cleaned up bucket: {bucket_name}.")
except ClientError as e:
    error_code = e.response["Error"]["Code"]
    if error_code == "NoSuchBucket":
        logger.info(f"Bucket {bucket_name} does not exist, skipping
cleanup.")
    else:
        logger.error(f"Error deleting bucket: {error_code}")
        raise

def display_buckets(self, source_bucket: str, dest_bucket: str):
    """
    Display a list of the objects in the test buckets.

    :param source_bucket: The name of the source bucket.
    :param dest_bucket: The name of the destination bucket.
    """
    self.list_bucket_contents(source_bucket)
    self.list_bucket_contents(dest_bucket)

def list_bucket_contents(self, bucket_name):
    """
    Display a list of the objects in the bucket.

    :param bucket_name: The name of the bucket.
    """
    try:
        # Get list of all objects in the bucket.
        print(f"\t Items in bucket {bucket_name}")
        list_response = self.s3_client.list_objects_v2(Bucket=bucket_name)
        objs = list_response.get("Contents", [])

```

```

        if not objs:
            print("\t\tNo objects found.")
        for obj in objs:
            key = obj["Key"]
            print(f"\t\t object: {key} ETag {obj['ETag']}")
        return objs
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "NoSuchBucket":
            logger.info(f"Bucket {bucket_name} does not exist.")
        else:
            logger.error(f"Error listing bucket and objects: {error_code}")
            raise

def display_menu(
    self, source_bucket: str, dest_bucket: str, object_key: str, etag: str
):
    """
    Displays the menu of conditional request options for the user.

    :param source_bucket: The name of the source bucket.
    :param dest_bucket: The name of the destination bucket.
    :param object_key: The key of the test object in the source bucket.
    :param etag: The etag of the test object in the source bucket.
    """

    actions = [
        "Print list of bucket items.",
        "Perform a conditional read.",
        "Perform a conditional copy.",
        "Perform a conditional write.",
        "Clean up and exit.",
    ]

    conditions = [
        "If-Match: using the object's ETag. This condition should succeed.",
        "If-None-Match: using the object's ETag. This condition should",
        fail.",
        "If-Modified-Since: using yesterday's date. This condition should",
        succeed.",
        "If-Unmodified-Since: using yesterday's date. This condition should",
        fail.",
    ]

```

```

condition_types = [
    "IfMatch",
    "IfNoneMatch",
    "IfModifiedSince",
    "IfUnmodifiedSince",
]
copy_condition_types = [
    "CopySourceIfMatch",
    "CopySourceIfNoneMatch",
    "CopySourceIfModifiedSince",
    "CopySourceIfUnmodifiedSince",
]

yesterday_date = datetime.datetime.utcnow() - datetime.timedelta(days=1)

choice = 0
while choice != 4:
    print("-" * 88)
    print("Choose an action to explore some example conditional
requests.")
    choice = q.choose("Which action would you like to take? ", actions)
    if choice == 0:
        print("Listing the objects and buckets.")
        self.display_buckets(source_bucket, dest_bucket)
    elif choice == 1:
        print("Perform a conditional read.")
        condition_type = q.choose("Enter the condition type : ",
conditions)
        if condition_type == 0 or condition_type == 1:
            self.conditional_requests.get_object_conditional(
                object_key, source_bucket,
condition_types[condition_type], etag
            )
        elif condition_type == 2 or condition_type == 3:
            self.conditional_requests.get_object_conditional(
                object_key,
                source_bucket,
                condition_types[condition_type],
                yesterday_date,
            )
    elif choice == 2:
        print("Perform a conditional copy.")

```

```

        condition_type = q.choose("Enter the condition type : ",
conditions)
        dest_key = q.ask("Enter an object key: ", q.non_empty)
        if condition_type == 0 or condition_type == 1:
            self.conditional_requests.copy_object_conditional(
                object_key,
                dest_key,
                source_bucket,
                dest_bucket,
                copy_condition_types[condition_type],
                etag,
            )
        elif condition_type == 2 or condition_type == 3:
            self.conditional_requests.copy_object_conditional(
                object_key,
                dest_key,
                copy_condition_types[condition_type],
                yesterday_date,
            )
        elif choice == 3:
            print(
                "Perform a conditional write using IfNoneMatch condition on
the object key."
            )
            print("If the key is a duplicate, the write will fail.")
            object_key = q.ask("Enter an object key: ", q.non_empty)
            self.conditional_requests.put_object_conditional(
                object_key, source_bucket, b"Conditional write example data."
            )
        elif choice == 4:
            print("Proceeding to cleanup.")

def run_scenario(self):
    """
    Runs the interactive scenario.
    """
    print("-" * 88)
    print("Welcome to the Amazon S3 conditional requests example.")
    print("-" * 88)

    print(
        f"""\

```

This example demonstrates the use of conditional requests for S3 operations.

You can use conditional requests to add preconditions to S3 read requests to return or copy

an object based on its Entity tag (ETag), or last modified date.

You can use a conditional write requests to prevent overwrites by ensuring

there is no existing object with the same key.

This example will allow you to perform conditional reads and writes that will succeed or fail based on your selected options.

Sample buckets and a sample object will be created as part of the example.

```

"""
)

bucket_prefix = q.ask("Enter a bucket name prefix: ", q.non_empty)
source_bucket_name = f"{bucket_prefix}-source-{RANDOM_SUFFIX}"
dest_bucket_name = f"{bucket_prefix}-dest-{RANDOM_SUFFIX}"
object_key = "test-upload-file.txt"

try:
    etag = self.setup_scenario(source_bucket_name, dest_bucket_name,
object_key)
    self.display_menu(source_bucket_name, dest_bucket_name, object_key,
etag)
finally:
    self.cleanup_scenario(source_bucket_name, dest_bucket_name)

print("-" * 88)
print("Thanks for watching.")
print("-" * 88)

if __name__ == "__main__":
    scenario = ConditionalRequestsScenario(
        S3ConditionalRequests.from_client(), boto3.client("s3")
    )
    scenario.run_scenario()

```

조건부 요청 작업을 정의하는 래퍼 클래스입니다.


```
import boto3
import logging

from botocore.exceptions import ClientError

# Configure logging
logger = logging.getLogger(__name__)

class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def get_object_conditional(
        self,
        object_key: str,
        source_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Retrieves an object from Amazon S3 with a conditional request.

        :param object_key: The key of the object to retrieve.
        :param source_bucket: The source bucket of the object.
        :param condition_type: The type of condition: 'IfMatch', 'IfNoneMatch',
        'IfModifiedSince', 'IfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
```

```

        response = self.s3.get_object(
            Bucket=source_bucket,
            Key=object_key,
            **{condition_type: condition_value},
        )
        sample_bytes = response["Body"].read(20)
        print(
            f"\tConditional read successful. Here are the first 20 bytes of
the object:\n"
        )
        print(f"\t{sample_bytes}")
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "PreconditionFailed":
            print("\tConditional read failed: Precondition failed")
        elif error_code == "304": # Not modified error code.
            print("\tConditional read failed: Object not modified")
        else:
            logger.error(f"Unexpected error: {error_code}")
            raise

    def put_object_conditional(self, object_key: str, source_bucket: str, data:
bytes):
        """
        Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite
        using an IfNoneMatch condition for the object key.

        :param object_key: The key of the object to upload.
        :param source_bucket: The source bucket of the object.
        :param data: The data to upload.
        """
        try:
            self.s3.put_object(
                Bucket=source_bucket, Key=object_key, Body=data, IfNoneMatch=""
            )
            print(
                f"\tConditional write successful for key {object_key} in bucket
{source_bucket}."
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]

```

```

        if error_code == "PreconditionFailed":
            print("\tConditional write failed: Precondition failed")
        else:
            logger.error(f"Unexpected error: {error_code}")
            raise

def copy_object_conditional(
    self,
    source_key: str,
    dest_key: str,
    source_bucket: str,
    dest_bucket: str,
    condition_type: str,
    condition_value: str,
):
    """
    Copies an object from one Amazon S3 bucket to another with a conditional
    request.

    :param source_key: The key of the source object to copy.
    :param dest_key: The key of the destination object.
    :param source_bucket: The source bucket of the object.
    :param dest_bucket: The destination bucket of the object.
    :param condition_type: The type of condition to apply, e.g.
        'CopySourceIfMatch', 'CopySourceIfNoneMatch',
        'CopySourceIfModifiedSince', 'CopySourceIfUnmodifiedSince'.
    :param condition_value: The value to use for the condition.
    """
    try:
        self.s3.copy_object(
            Bucket=dest_bucket,
            Key=dest_key,
            CopySource={"Bucket": source_bucket, "Key": source_key},
            **{condition_type: condition_value},
        )
        print(
            f"\tConditional copy successful for key {dest_key} in bucket {dest_bucket}."
        )
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "PreconditionFailed":
            print("\tConditional copy failed: Precondition failed")

```

```
elif error_code == "304": # Not modified error code.
    print("\tConditional copy failed: Object not modified")
else:
    logger.error(f"Unexpected error: {error_code}")
    raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 다음 주제를 참조하세요.
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon S3 버킷의 액세스 제어 목록(ACL) 관리

다음 코드 예제는 Amazon S3 버킷의 액세스 제어 목록(ACL)을 관리하는 방법을 보여줍니다.

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to manage Amazon Simple Storage Service
/// (Amazon S3) access control lists (ACLs) to control Amazon S3 bucket
```

```

    /// access.
    /// </summary>
    public class ManageACLs
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket1";
            string newBucketName = "amzn-s3-demo-bucket2";
            string keyName = "sample-object.txt";
            string emailAddress = "someone@example.com";

            // If the AWS Region where your bucket is located is different from
            // the Region defined for the default user, pass the Amazon S3
bucket's
            // name to the client constructor. It should look like this:
            // RegionEndpoint bucketRegion = RegionEndpoint.USEast1;
            IAmazonS3 client = new AmazonS3Client();

            await TestBucketObjectACLsAsync(client, bucketName, newBucketName,
keyName, emailAddress);
        }

        /// <summary>
        /// Creates a new Amazon S3 bucket with a canned ACL, then retrieves the
ACL
        /// information and then adds a new ACL to one of the objects in the
        /// Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// methods to create a bucket, get an ACL, and add a different ACL to
        /// one of the objects.</param>
        /// <param name="bucketName">A string representing the original Amazon S3
        /// bucket name.</param>
        /// <param name="newBucketName">A string representing the name of the
        /// new bucket that will be created.</param>
        /// <param name="keyName">A string representing the key name of an Amazon
S3
        /// object for which we will change the ACL.</param>
        /// <param name="emailAddress">A string representing the email address
        /// belonging to the person to whom access to the Amazon S3 bucket will
be
        /// granted.</param>
        public static async Task TestBucketObjectACLsAsync(

```

```

        IAmazonS3 client,
        string bucketName,
        string newBucketName,
        string keyName,
        string emailAddress)
    {
        try
        {
            // Create a new Amazon S3 bucket and specify canned ACL.
            var success = await CreateBucketWithCannedACLAsync(client,
newBucketName);

            // Get the ACL on a bucket.
            await GetBucketACLAsync(client, bucketName);

            // Add (replace) the ACL on an object in a bucket.
            await AddACLToExistingObjectAsync(client, bucketName, keyName,
emailAddress);
        }
        catch (AmazonS3Exception amazonS3Exception)
        {
            Console.WriteLine($"Exception: {amazonS3Exception.Message}");
        }
    }

    /// <summary>
    /// Creates a new Amazon S3 bucket with a canned ACL attached.
    /// </summary>
    /// <param name="client">The initialized client object used to call
    /// PutBucketAsync.</param>
    /// <param name="newBucketName">A string representing the name of the
    /// new Amazon S3 bucket.</param>
    /// <returns>Returns a boolean value indicating success or failure.</
returns>
    public static async Task<bool> CreateBucketWithCannedACLAsync(IAmazonS3
client, string newBucketName)
    {
        var request = new PutBucketRequest()
        {
            BucketName = newBucketName,
            BucketRegion = S3Region.EUWest1,

            // Add a canned ACL.
            CannedACL = S3CannedACL.LogDeliveryWrite,

```

```

        };

        var response = await client.PutBucketAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Retrieves the ACL associated with the Amazon S3 bucket name in the
    /// bucketName parameter.
    /// </summary>
    /// <param name="client">The initialized client object used to call
    /// PutBucketAsync.</param>
    /// <param name="bucketName">The Amazon S3 bucket for which we want to
get the
    /// ACL list.</param>
    /// <returns>Returns an S3AccessControlList returned from the call to
    /// GetACLAsync.</returns>
    public static async Task<S3AccessControlList> GetBucketACLAsync(IAmazonS3
client, string bucketName)
    {
        GetACLResponse response = await client.GetACLAsync(new GetACLRequest
        {
            BucketName = bucketName,
        });

        return response.AccessControlList;
    }

    /// <summary>
    /// Adds a new ACL to an existing object in the Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized client object used to call
    /// PutBucketAsync.</param>
    /// <param name="bucketName">A string representing the name of the Amazon
S3
    /// bucket containing the object to which we want to apply a new ACL.</
param>
    /// <param name="keyName">A string representing the name of the object
    /// to which we want to apply the new ACL.</param>
    /// <param name="emailAddress">The email address of the person to whom
    /// we will be applying to whom access will be granted.</param>

```

```
public static async Task AddACLToExistingObjectAsync(IAmazonS3 client,
string bucketName, string keyName, string emailAddress)
{
    // Retrieve the ACL for an object.
    GetACLResponse aclResponse = await client.GetACLAsync(new
GetACLRequest
    {
        BucketName = bucketName,
        Key = keyName,
    });

    S3AccessControlList acl = aclResponse.AccessControlList;

    // Retrieve the owner.
    Owner owner = acl.Owner;

    // Clear existing grants.
    acl.Grants.Clear();

    // Add a grant to reset the owner's full permission
    // (the previous clear statement removed all permissions).
    var fullControlGrant = new S3Grant
    {
        Grantee = new S3Grantee { CanonicalUser = acl.Owner.Id },
    };
    acl.AddGrant(fullControlGrant.Grantee, S3Permission.FULL_CONTROL);

    // Specify email to identify grantee for granting permissions.
    var grantUsingEmail = new S3Grant
    {
        Grantee = new S3Grantee { EmailAddress = emailAddress },
        Permission = S3Permission.WRITE_ACP,
    };

    // Specify log delivery group as grantee.
    var grantLogDeliveryGroup = new S3Grant
    {
        Grantee = new S3Grantee { URI = "http://acs.amazonaws.com/groups/
s3/LogDelivery" },
        Permission = S3Permission.WRITE,
    };

    // Create a new ACL.
    var newAcl = new S3AccessControlList
```



```
        {
            Grants = new List<S3Grant> { grantUsingEmail,
grantLogDeliveryGroup },
            Owner = owner,
        };

        // Set the new ACL. We're throwing away the response here.
        _ = await client.PutACLAsync(new PutACLRequest
        {
            BucketName = bucketName,
            Key = keyName,
            AccessControlList = newAcl,
        });
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 다음 주제를 참조하세요.

- [GetBucketAcl](#)
- [GetObjectAcl](#)
- [PutBucketAcl](#)
- [PutObjectAcl](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon S3로 대규모 Amazon SQS 메시지 관리

다음 코드 예시에서는 Amazon SQS 확장 클라이언트 라이브러리를 사용하여 대규모 Amazon SQS 메시지 작업을 수행하는 방법을 보여줍니다.

Java

SDK for Java 2.x

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import com.amazonaws.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazonaws.sqs.javamessaging.ExtendedClientConfiguration;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

import java.util.Arrays;
```

```
import java.util.List;
import java.util.UUID;

/**
 * Example of using Amazon SQS Extended Client Library for Java 2.x.
 */
public class SqsExtendedClientExample {
    private static final Logger logger =
        LoggerFactory.getLogger(SqsExtendedClientExample.class);

    private String s3BucketName;
    private String queueUrl;
    private final String queueName;
    private final S3Client s3Client;
    private final SqsClient sqsExtendedClient;
    private final int messageSize;

    /**
     * Constructor with default clients and message size.
     */
    public SqsExtendedClientExample() {
        this(S3Client.create(), 300000);
    }

    /**
     * Constructor with custom S3 client and message size.
     *
     * @param s3Client The S3 client to use
     * @param messageSize The size of the test message to create
     */
    public SqsExtendedClientExample(S3Client s3Client, int messageSize) {
        this.s3Client = s3Client;
        this.messageSize = messageSize;

        // Generate a unique bucket name.
        this.s3BucketName = UUID.randomUUID() + "-" +
            DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());

        // Generate a unique queue name.
        this.queueName = "MyQueue-" + UUID.randomUUID();

        // Configure the SQS extended client.
        final ExtendedClientConfiguration extendedClientConfig = new
            ExtendedClientConfiguration()
```

```
        .withPayloadSupportEnabled(s3Client, s3BucketName);

        this.sqsExtendedClient = new
AmazonSQSExtendedClient(SqsClient.builder().build(), extendedClientConfig);
    }

    public static void main(String[] args) {
        SqsExtendedClientExample example = new SqsExtendedClientExample();
        try {
            example.setup();
            example.sendAndReceiveMessage();
        } finally {
            example.cleanup();
        }
    }

    /**
     * Send a large message and receive it back.
     *
     * @return The received message
     */
    public Message sendAndReceiveMessage() {
        try {
            // Create a large message.
            char[] chars = new char[messageSize];
            Arrays.fill(chars, 'x');
            String largeMessage = new String(chars);

            // Send the message.
            final SendMessageRequest sendMessageRequest =
SendMessageRequest.builder()
                .queueUrl(queueUrl)
                .messageBody(largeMessage)
                .build();

            sqsExtendedClient.sendMessage(sendMessageRequest);
            logger.info("Sent message of size: {}", largeMessage.length());

            // Receive and return the message.
            final ReceiveMessageResponse receiveMessageResponse =
sqsExtendedClient.receiveMessage(
                ReceiveMessageRequest.builder().queueUrl(queueUrl).build());

            List<Message> messages = receiveMessageResponse.messages();
```

```
        if (messages.isEmpty()) {
            throw new RuntimeException("No messages received");
        }

        Message message = messages.getFirst();
        logger.info("\nMessage received.");
        logger.info("  ID: {}", message.messageId());
        logger.info("  Receipt handle: {}", message.receiptHandle());
        logger.info("  Message body size: {}", message.body().length());
        logger.info("  Message body (first 5 characters): {}",
            message.body().substring(0, 5));

        return message;
    } catch (RuntimeException e) {
        logger.error("Error during message processing: {}", e.getMessage(),
            e);
        throw e;
    }
}
```

- 자세한 정보는 [AWS SDK for Java 2.x 개발자 안내서](#)를 참조하세요.
- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
 - [CreateBucket](#)
 - [PutBucketLifecycleConfiguration](#)
 - [ReceiveMessage](#)
 - [SendMessage](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Lambda 함수로 버전이 지정된 Amazon S3 객체를 배치 단위로 관리

다음 코드 예제는 Lambda 함수를 사용하여 버전이 지정된 S3 객체를 배치 단위로 관리하는 방법을 보여줍니다.

Python

SDK for Python(Boto3)

처리를 수행할 AWS Lambda 함수를 호출하는 작업을 생성하여 버전이 지정된 Amazon Simple Storage Service(Amazon S3) 객체를 배치 단위로 조작하는 방법을 보여줍니다. 이 예제에서는 버전 관리를 사용한 버킷을 생성하고, Lewis Carroll의 You Are Old, Father William이라는 시의 시구를 업로드하며, Amazon S3 배치 작업을 사용하여 다양한 방법으로 시를 번형합니다.

다음 작업을 수행하는 방법에 대해 알아보세요.

- 버전이 지정된 객체에서 작동하는 Lambda 함수를 생성합니다.
- 업데이트할 객체의 매니페스트를 만듭니다.
- 객체를 업데이트하기 위해 Lambda 함수를 호출하는 배치 작업을 생성합니다.
- Lambda 함수를 삭제합니다.
- 버전이 지정된 버킷을 비운 다음 삭제합니다.

이 예제는 GitHub에서 가장 잘 볼 수 있습니다. 전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon S3

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon S3 URI 구문 분석

다음 코드 예제는 버킷 이름 및 객체 키와 같은 중요한 구성 요소를 추출하기 위해 Amazon S3 URI를 구문 분석하는 방법을 보여 줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

[S3Uri](#) 클래스를 사용하여 Amazon S3 URI를 구문 분석합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.S3Uri;
import software.amazon.awssdk.services.s3.S3Utilities;

import java.net.URI;
import java.util.List;
import java.util.Map;

/**
 *
 * @param s3Client      - An S3Client through which you acquire an S3Uri
instance.
 * @param s3objectUrl - A complex URL (String) that is used to demonstrate
S3Uri
 *
capabilities.
 */
public static void parseS3UriExample(S3Client s3Client, String s3objectUrl) {
    logger.info(s3objectUrl);
    // Console output:
    // 'https://s3.us-west-1.amazonaws.com/myBucket/resources/doc.txt?
versionId=abc123&partNumber=77&partNumber=88'.

    // Create an S3Utilities object using the configuration of the s3Client.
    S3Utilities s3Utilities = s3Client.utilities();

    // From a String URL create a URI object to pass to the parseUri()
method.
    URI uri = URI.create(s3objectUrl);
```

```
S3Uri s3Uri = s3Utilities.parseUri(uri);

// If the URI contains no value for the Region, bucket or key, the SDK
returns
// an empty Optional.
// The SDK returns decoded URI values.

Region region = s3Uri.region().orElse(null);
log("region", region);
// Console output: 'region: us-west-1'.

String bucket = s3Uri.bucket().orElse(null);
log("bucket", bucket);
// Console output: 'bucket: myBucket'.

String key = s3Uri.key().orElse(null);
log("key", key);
// Console output: 'key: resources/doc.txt'.

Boolean isPathStyle = s3Uri.isPathStyle();
log("isPathStyle", isPathStyle);
// Console output: 'isPathStyle: true'.

// If the URI contains no query parameters, the SDK returns an empty map.
Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();
log("rawQueryParameters", queryParams);
// Console output: 'rawQueryParameters: {versionId=[abc123],
partNumber=[77,
// 88]]'.

// Retrieve the first or all values for a query parameter as shown in the
// following code.
String versionId =
s3Uri.firstMatchingRawQueryParameter("versionId").orElse(null);
log("firstMatchingRawQueryParameter-versionId", versionId);
// Console output: 'firstMatchingRawQueryParameter-versionId: abc123'.

String partNumber =
s3Uri.firstMatchingRawQueryParameter("partNumber").orElse(null);
log("firstMatchingRawQueryParameter-partNumber", partNumber);
// Console output: 'firstMatchingRawQueryParameter-partNumber: 77'.

List<String> partNumbers =
s3Uri.firstMatchingRawQueryParameters("partNumber");
```



```

        log("firstMatchingRawQueryParameter", partNumbers);
        // Console output: 'firstMatchingRawQueryParameter: [77, 88]'.

        /*
         * Object keys and query parameters with reserved or unsafe characters,
        must be
         * URL-encoded.
         * For example replace whitespace " " with "%20".
         * Valid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object%20key?query=
        %5Bbrackets%5D"
         * Invalid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object key?
        query=[brackets]"
         *
         * Virtual-hosted-style URIs with bucket names that contain a dot, ".",
        the dot
         * must not be URL-encoded.
         * Valid: "https://my.Bucket.s3.us-west-1.amazonaws.com/key"
         * Invalid: "https://my%2EBucket.s3.us-west-1.amazonaws.com/key"
         */
    }

    private static void log(String s3UriElement, Object element) {
        if (element == null) {
            logger.info("{}: {}", s3UriElement, "null");
        } else {
            logger.info("{}: {}", s3UriElement, element);
        }
    }
}

```

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon S3 객체 멀티파트 복사 수행

다음 코드 예제는 Amazon S3 객체의 멀티파트 복사를 수행하는 방법을 보여줍니다.

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to perform a multi-part copy from one Amazon
/// Simple Storage Service (Amazon S3) bucket to another.
/// </summary>
public class MPUApiCopyObj
{
    private const string SourceBucket = "amzn-s3-demo-bucket1";
    private const string TargetBucket = "amzn-s3-demo-bucket2";
    private const string SourceObjectKey = "example.mov";
    private const string TargetObjectKey = "copied_video_file.mov";

    /// <summary>
    /// This method starts the multi-part upload.
    /// </summary>
    public static async Task Main()
    {
        var s3Client = new AmazonS3Client();
        Console.WriteLine("Copying object...");
        await MPUCopyObjectAsync(s3Client);
    }

    /// <summary>
    /// This method uses the passed client object to perform a multipart
    /// copy operation.
    /// </summary>
    /// <param name="client">An Amazon S3 client object that will be used
```

```
/// to perform the copy.</param>
public static async Task MPUCopyObjectAsync(AmazonS3Client client)
{
    // Create a list to store the copy part responses.
    var copyResponses = new List<CopyPartResponse>();

    // Setup information required to initiate the multipart upload.
    var initiateRequest = new InitiateMultipartUploadRequest
    {
        BucketName = TargetBucket,
        Key = TargetObjectKey,
    };

    // Initiate the upload.
    InitiateMultipartUploadResponse initResponse =
        await client.InitiateMultipartUploadAsync(initiateRequest);

    // Save the upload ID.
    string uploadId = initResponse.UploadId;

    try
    {
        // Get the size of the object.
        var metadataRequest = new GetObjectMetadataRequest
        {
            BucketName = SourceBucket,
            Key = SourceObjectKey,
        };

        GetObjectMetadataResponse metadataResponse =
            await client.GetObjectMetadataAsync(metadataRequest);
        var objectSize = metadataResponse.ContentLength; // Length in
bytes.

        // Copy the parts.
        var partSize = 5 * (long)Math.Pow(2, 20); // Part size is 5 MB.

        long bytePosition = 0;
        for (int i = 1; bytePosition < objectSize; i++)
        {
            var copyRequest = new CopyPartRequest
            {
                DestinationBucket = TargetBucket,
                DestinationKey = TargetObjectKey,
```

```

        SourceBucket = SourceBucket,
        SourceKey = SourceObjectKey,
        UploadId = uploadId,
        FirstByte = bytePosition,
        LastByte = bytePosition + partSize - 1 >= objectSize ?
objectSize - 1 : bytePosition + partSize - 1,
        PartNumber = i,
    };

    copyResponses.Add(await client.CopyPartAsync(copyRequest));

    bytePosition += partSize;
}

// Set up to complete the copy.
var completeRequest = new CompleteMultipartUploadRequest
{
    BucketName = TargetBucket,
    Key = TargetObjectKey,
    UploadId = initResponse.UploadId,
};
completeRequest.AddPartETags(copyResponses);

// Complete the copy.
CompleteMultipartUploadResponse completeUploadResponse =
    await client.CompleteMultipartUploadAsync(completeRequest);
}
catch (AmazonS3Exception e)
{
    Console.WriteLine($"Error encountered on server.
Message: '{e.Message}' when writing an object");
}
catch (Exception e)
{
    Console.WriteLine($"Unknown encountered on server.
Message: '{e.Message}' when writing an object");
}
}
}

```

- API 세부 정보는 AWS SDK for .NET API 참조의 다음 주제를 참조하세요.

- [CompleteMultipartUpload](#)
- [CreateMultipartUpload](#)
- [GetObjectMetadata](#)
- [UploadPartCopy](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon S3 이벤트 알림 수신 및 처리

다음 코드 예시에서는 객체 지향적 방식으로 S3 이벤트 알림을 사용하는 방법을 보여줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예시에서는 Amazon SQS를 사용하여 S3 알림 이벤트를 처리하는 방법을 보여줍니다.

```
/**
 * This method receives S3 event notifications by using an SqsAsyncClient.
 * After the client receives the messages it deserializes the JSON payload
 and logs them. It uses
 * the S3EventNotification class (part of the S3 event notification API for
 Java) to deserialize
 * the JSON payload and access the messages in an object-oriented way.
 *
 * @param queueUrl The URL of the AWS SQS queue that receives the S3 event
 notifications.
 * @see <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/
 awssdk/eventnotifications/s3/model/package-summary.html">S3EventNotification
 API</a>.
 * <p>
```

```

    * To use S3 event notification serialization/deserialization to objects, add
    the following
    * dependency to your Maven pom.xml file.
    * <dependency>
    * <groupId>software.amazon.awssdk</groupId>
    * <artifactId>s3-event-notifications</artifactId>
    * <version><LATEST></version>
    * </dependency>
    * <p>
    * The S3 event notification API became available with version 2.25.11 of the
    Java SDK.
    * <p>
    * This example shows the use of the API with AWS SQS, but it can be used to
    process S3 event notifications
    * in AWS SNS or AWS Lambda as well.
    * <p>
    * Note: The S3EventNotification class does not work with messages routed
    through AWS EventBridge.
    */
    static void processS3Events(String bucketName, String queueUrl, String
    queueArn) {
        try {
            // Configure the bucket to send Object Created and Object Tagging
            notifications to an existing SQS queue.
            s3Client.putBucketNotificationConfiguration(b -> b
                .notificationConfiguration(ncb -> ncb
                    .queueConfigurations(qcb -> qcb
                        .events(Event.S3_OBJECT_CREATED,
Event.S3_OBJECT_TAGGING)
                            .queueArn(queueArn)))
                    .bucket(bucketName)
                ).join();

            triggerS3EventNotifications(bucketName);
            // Wait for event notifications to propagate.
            Thread.sleep(Duration.ofSeconds(5).toMillis());

            boolean didReceiveMessages = true;
            while (didReceiveMessages) {
                // Display the number of messages that are available in the
                queue.

                sqsClient.getQueueAttributes(b -> b
                    .queueUrl(queueUrl)

```

```

.attributeNames(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)
                ).thenAccept(attributeResponse ->
                    logger.info("Approximate number of messages in
the queue: {}"),
attributeResponse.attributes().get(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)))
                .join();

// Receive the messages.
ReceiveMessageResponse response = sqsClient.receiveMessage(b -> b
    .queueUrl(queueUrl)
    ).get();
logger.info("Count of received messages: {}",
response.messages().size());
didReceiveMessages = !response.messages().isEmpty();

// Create a collection to hold the received message for deletion
// after we log the messages.
HashSet<DeleteMessageBatchRequestEntry> messagesToDelete = new
HashSet<>();

// Process each message.
response.messages().forEach(message -> {
    logger.info("Message id: {}", message.messageId());
    // Deserialize JSON message body to a S3EventNotification
object
    // to access messages in an object-oriented way.
    S3EventNotification event =
S3EventNotification.fromJson(message.body());

    // Log the S3 event notification record details.
    if (event.getRecords() != null) {
        event.getRecords().forEach(record -> {
            String eventName = record.getEventName();
            String key = record.getS3().getObject().getKey();
            logger.info(record.toString());
            logger.info("Event name is {} and key is {}",
eventName, key);
        });
    }
    // Add logged messages to collection for batch deletion.
    messagesToDelete.add(DeleteMessageBatchRequestEntry.builder()
        .id(message.messageId())
        .receiptHandle(message.receiptHandle())

```

```

        .build());
    });
    // Delete messages.
    if (!messagesToDelete.isEmpty()) {
        sqsClient.deleteMessageBatch(DeleteMessageBatchRequest.builder()
            .queueUrl(queueUrl)
            .entries(messagesToDelete)
            .build()
        ).join();
    }
    } // End of while block.
} catch (InterruptedException | ExecutionException e) {
    throw new RuntimeException(e);
}
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
 - [DeleteMessageBatch](#)
 - [GetQueueAttributes](#)
 - [PutBucketNotificationConfiguration](#)
 - [ReceiveMessage](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 EXIF 및 기타 이미지 정보 저장

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- JPG, JPEG 또는 PNG 파일에서 EXIF 정보를 가져옵니다.
- Amazon S3 버킷에 이미지 파일을 업로드합니다.
- Amazon Rekognition을 사용하여 파일에서 3가지 주요 속성(레이블)을 파악합니다.
- EXIF 및 레이블 정보를 리전의 Amazon DynamoDB 테이블에 추가합니다.

Rust

SDK for Rust

JPG, JPEG 또는 PNG 파일에서 EXIF 정보를 가져오고, 이미지 파일을 Amazon S3 버킷에 업로드하며, Amazon Rekognition을 사용하여 파일에서 3가지 주요 속성(Amazon Rekognition의 레이블)을 파악한 후 EXIF 및 레이블 정보를 리전의 Amazon DynamoDB 테이블에 추가합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- DynamoDB
- Amazon Rekognition
- Amazon S3

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon EventBridge에 S3 이벤트 알림 전송

다음 코드 예시에서는 S3 이벤트 알림을 EventBridge로 보내고 알림을 Amazon SNS 주제 및 Amazon SQS 대기열로 라우팅하도록 버킷을 설정하는 방법을 보여줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/** This method configures a bucket to send events to AWS EventBridge and
    creates a rule
    * to route the S3 object created events to a topic and a queue.
    *
    * @param bucketName Name of existing bucket
    * @param topicArn ARN of existing topic to receive S3 event notifications
```

```

    * @param queueArn ARN of existing queue to receive S3 event notifications
    *
    * An AWS CloudFormation stack sets up the bucket, queue, topic before the
    method runs.
    */
    public static String setBucketNotificationToEventBridge(String bucketName,
String topicArn, String queueArn) {
        try {
            // Enable bucket to emit S3 Event notifications to EventBridge.
            s3Client.putBucketNotificationConfiguration(b -> b
                .bucket(bucketName)
                .notificationConfiguration(b1 -> b1
                    .eventBridgeConfiguration(
                        SdkBuilder::build)
                ).build()).join();

            // Create an EventBridge rule to route Object Created notifications.
            PutRuleRequest putRuleRequest = PutRuleRequest.builder()
                .name(RULE_NAME)
                .eventPattern("""
                    {
                        "source": ["aws.s3"],
                        "detail-type": ["Object Created"],
                        "detail": {
                            "bucket": {
                                "name": ["%s"]
                            }
                        }
                    }
                """).formatted(bucketName))
                .build();

            // Add the rule to the default event bus.
            PutRuleResponse putRuleResponse =
            eventBridgeClient.putRule(putRuleRequest)
                .whenComplete((r, t) -> {
                    if (t != null) {
                        logger.error("Error creating event bus rule: " +
t.getMessage(), t);
                        throw new RuntimeException(t.getCause().getMessage(),
t);
                    }
                    logger.info("Event bus rule creation request sent
successfully. ARN is: {}", r.ruleArn());
                });
        }
    }

```

```

        }).join();

        // Add the existing SNS topic and SQS queue as targets to the rule.
        eventBridgeClient.putTargets(b -> b
            .eventBusName("default")
            .rule(RULE_NAME)
            .targets(List.of (
                Target.builder()
                    .arn(queueArn)
                    .id("Queue")
                    .build(),
                Target.builder()
                    .arn(topicArn)
                    .id("Topic")
                    .build()
            )
        ).join();
        return putRuleResponse.ruleArn();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
- [PutBucketNotificationConfiguration](#)
- [PutRule](#)
- [PutTargets](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon S3 객체 업로드 또는 다운로드 추적

다음 코드 예시는 Amazon S3 객체 업로드 또는 다운로드를 추적하는 방법을 보여 줍니다.

Java

SDK for Java 2.x

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

파일 업로드 진행 상황을 추적합니다.

```
public void trackUploadFile(S3TransferManager transferManager, String
bucketName,
                        String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create()) // Add
listener.
        .source(Paths.get(filePathURI))
        .build();

    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

    fileUpload.completionFuture().join();
    /*
        The SDK provides a LoggingTransferListener implementation of the
        TransferListener interface.
        You can also implement the interface to provide your own logic.

        Configure log4j2 with settings such as the following.
        <Configuration status="WARN">
            <Appenders>
                <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
                    <PatternLayout pattern="%m%n"/>
                </Console>
            </Appenders>

            <Loggers>
                <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
```

```

        <AppenderRef ref="AlignedConsoleAppender"/>
    </logger>
</Loggers>
</Configuration>

```

Log4J2 logs the progress. The following is example output for a 21.3 MB file upload.

```

    Transfer initiated...
    |                      | 0.0%
    |====                 | 21.1%
    |=====              | 60.5%
    |=====              | 100.0%
    Transfer complete!

    */
}

```

파일 다운로드 진행 상황을 추적합니다.

```

public void trackDownloadFile(S3TransferManager transferManager, String
bucketName,
                                String key, String downloadedFilePath) {
    DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
        .getObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create()) // Add
listener.
        .destination(Paths.get(downloadedFilePath))
        .build();

    FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

    CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
    /*
        The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
        You can also implement the interface to provide your own logic.

        Configure log4J2 with settings such as the following.
        <Configuration status="WARN">
            <Appenders>

```

```

        <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
            <PatternLayout pattern="%m%n"/>
        </Console>
    </Appenders>

    <Loggers>
        <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
            <AppenderRef ref="AlignedConsoleAppender"/>
        </logger>
    </Loggers>
</Configuration>

```

Log4J2 logs the progress. The following is example output for a 21.3 MB file download.

```

Transfer initiated...
|=====| 39.4%
|=====| 78.8%
|=====| 100.0%
Transfer complete!

```

```

    */
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
- [GetObject](#)
- [PutObject](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

S3 객체 Lambda를 사용하여 애플리케이션의 데이터 변환

다음 코드 예시는 S3 객체 Lambda를 사용하여 애플리케이션의 데이터를 변환하는 방법을 보여 줍니다.

.NET

SDK for .NET

표준 S3 GET 요청에 사용자 지정 코드를 추가하여, 요청하는 클라이언트 또는 애플리케이션의 요구 사항에 맞게 S3에서 검색된 요청된 객체를 수정하는 방법을 보여 줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Lambda
- Amazon S3

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용한 단위 및 통합 테스트의 예시 접근 방식

다음 코드 예제는 AWS SDK를 사용하여 단위 및 통합 테스트를 작성할 때의 모범 사례 기법에 관한 예시 방법을 보여줍니다.

Rust

SDK for Rust

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

테스트 예제용 Cargo.toml

```
[package]
name = "testing-examples"
version = "0.1.0"
authors = [
  "John Disanti <jdisanti@amazon.com>",
  "Doug Schwartz <dougsch@amazon.com>",
```

```

]
edition = "2021"

[dependencies]
async-trait = "0.1.51"
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-credential-types = { version = "1.0.1", features = [ "hardcoded-credentials", ] }
aws-sdk-s3 = { version = "1.4.0" }
aws-smithy-types = { version = "1.0.1" }
aws-smithy-runtime = { version = "1.0.1", features = ["test-util"] }
aws-smithy-runtime-api = { version = "1.0.1", features = ["test-util"] }
aws-types = { version = "1.0.1" }
clap = { version = "4.4", features = ["derive"] }
http = "0.2.9"
mockall = "0.11.4"
serde_json = "1"
tokio = { version = "1.20.1", features = ["full"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }

[[bin]]
name = "main"
path = "src/main.rs"

```

automock 및 서비스 래퍼를 사용한 유닛 테스트 예제

```

use aws_sdk_s3 as s3;
#[allow(unused_imports)]
use mockall::automock;

use s3::operation::list_objects_v2::{ListObjectsV2Error, ListObjectsV2Output};

#[cfg(test)]
pub use MockS3Impl as S3;
#[cfg(not(test))]
pub use S3Impl as S3;

#[allow(dead_code)]
pub struct S3Impl {
    inner: s3::Client,
}

```



```
#[cfg_attr(test, automock)]
impl S3Impl {
    #[allow(dead_code)]
    pub fn new(inner: s3::Client) -> Self {
        Self { inner }
    }

    #[allow(dead_code)]
    pub async fn list_objects(
        &self,
        bucket: &str,
        prefix: &str,
        continuation_token: Option<String>,
    ) -> Result<ListObjectsV2Output, s3::error::SdkError<ListObjectsV2Error>> {
        self.inner
            .list_objects_v2()
            .bucket(bucket)
            .prefix(prefix)
            .set_continuation_token(continuation_token)
            .send()
            .await
    }
}

#[allow(dead_code)]
pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3_list: S3,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3_list
            .list_objects(bucket, prefix, next_token.take())
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }
    }
}
```

```

        // Handle pagination, and break the loop if there are no more pages
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {
            break;
        }
    }
    Ok(total_size_bytes)
}

#[cfg(test)]
mod test {
    use super::*;
    use mockall::predicate::eq;

    #[tokio::test]
    async fn test_single_page() {
        let mut mock = MockS3Impl::default();
        mock.expect_list_objects()
            .with(eq("test-bucket"), eq("test-prefix"), eq(None))
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(5).build(),
                        s3::types::Object::builder().size(2).build(),
                    ]))
                    .build())
            });

        // Run the code we want to test with it
        let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
            .await
            .unwrap();

        // Verify we got the correct total size back
        assert_eq!(7, size);
    }

    #[tokio::test]
    async fn test_multiple_pages() {
        // Create the Mock instance with two pages of objects now
        let mut mock = MockS3Impl::default();
        mock.expect_list_objects()

```

```

        .with(eq("test-bucket"), eq("test-prefix"), eq(None))
        .return_once(|_, _, _| {
            Ok(ListObjectsV2Output::builder()
                .set_contents(Some(vec![
                    // Mock content for ListObjectsV2 response
                    s3::types::Object::builder().size(5).build(),
                    s3::types::Object::builder().size(2).build(),
                ]))
                .set_next_continuation_token(Some("next".to_string()))
                .build())
        });

mock.expect_list_objects()
    .with(
        eq("test-bucket"),
        eq("test-prefix"),
        eq(Some("next".to_string()))
    )
    .return_once(|_, _, _| {
        Ok(ListObjectsV2Output::builder()
            .set_contents(Some(vec![
                // Mock content for ListObjectsV2 response
                s3::types::Object::builder().size(3).build(),
                s3::types::Object::builder().size(9).build(),
            ]))
            .build())
    });

// Run the code we want to test with it
let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
    .await
    .unwrap();

assert_eq!(19, size);
}
}

```

StatiReplayClient를 사용한 통합 테스트 예제

```

use aws_sdk_s3 as s3;

#[allow(dead_code)]

```

```

pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3: s3::Client,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3
            .list_objects_v2()
            .prefix(prefix)
            .bucket(bucket)
            .set_continuation_token(next_token.take())
            .send()
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }

        // Handle pagination, and break the loop if there are no more pages
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {
            break;
        }
    }
    Ok(total_size_bytes)
}

#[allow(dead_code)]
fn make_s3_test_credentials() -> s3::config::Credentials {
    s3::config::Credentials::new(
        "ATESTCLIENT",
        "atestsecretkey",
        Some("atestsessiontoken".to_string()),
        None,
        "",
    )
}

#[cfg(test)]

```

```

mod test {
  use super::*;
  use aws_config::BehaviorVersion;
  use aws_sdk_s3 as s3;
  use aws_smithy_runtime::client::http::test_util::{ReplayEvent,
StaticReplayClient};
  use aws_smithy_types::body::SdkBody;

  #[tokio::test]
  async fn test_single_page() {
    let page_1 = ReplayEvent::new(
      http::Request::builder()
        .method("GET")
        .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
        .body(SdkBody::empty())
        .unwrap(),
      http::Response::builder()
        .status(200)
        .body(SdkBody::from(include_str!("./testing/
response_1.xml")))
        .unwrap(),
    );

    let replay_client = StaticReplayClient::new(vec![page_1]);
    let client: s3::Client = s3::Client::from_conf(
      s3::Config::builder()
        .behavior_version(BehaviorVersion::latest())
        .credentials_provider(make_s3_test_credentials())
        .region(s3::config::Region::new("us-east-1"))
        .http_client(replay_client.clone())
        .build(),
    );

    // Run the code we want to test with it
    let size = determine_prefix_file_size(client, "test-bucket", "test-
prefix")
      .await
      .unwrap();

    // Verify we got the correct total size back
    assert_eq!(7, size);
    replay_client.assert_requests_match(&[]);
  }
}

```

```

#[tokio::test]
async fn test_multiple_pages() {
    let page_1 = ReplayEvent::new(
        http::Request::builder()
            .method("GET")
            .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
            .body(SdkBody::empty())
            .unwrap(),
        http::Response::builder()
            .status(200)
            .body(SdkBody::from(include_str!("./testing/
response_multi_1.xml")))
            .unwrap(),
    );
    let page_2 = ReplayEvent::new(
        http::Request::builder()
            .method("GET")
            .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix&continuation-token=next")
            .body(SdkBody::empty())
            .unwrap(),
        http::Response::builder()
            .status(200)
            .body(SdkBody::from(include_str!("./testing/
response_multi_2.xml")))
            .unwrap(),
    );
    let replay_client = StaticReplayClient::new(vec![page_1, page_2]);
    let client: s3::Client = s3::Client::from_conf(
        s3::Config::builder()
            .behavior_version(BehaviorVersion::latest())
            .credentials_provider(make_s3_test_credentials())
            .region(s3::config::Region::new("us-east-1"))
            .http_client(replay_client.clone())
            .build(),
    );

    // Run the code we want to test with it
    let size = determine_prefix_file_size(client, "test-bucket", "test-
prefix")
        .await
        .unwrap();

```

```

        assert_eq!(19, size);

        replay_client.assert_requests_match(&[]);
    }
}

```

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

로컬 디렉터리를 Amazon Simple Storage Service(S3) 버킷에 반복 업로드

다음 코드 예제는 Amazon Simple Storage Service(S3) 버킷에 로컬 디렉터리를 반복적으로 업로드하는 방법을 보여줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

[S3TransferManager](#)를 사용하여 [로컬 디렉터리를 업로드](#)합니다. [파일 전체](#)를 보고 [테스트](#)합니다.

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;

```

```

public Integer uploadDirectory(S3TransferManager transferManager,
    URI sourceDirectory, String bucketName) {
    DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
    .source(Paths.get(sourceDirectory))
    .bucket(bucketName)
    .build());

    CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
    completedDirectoryUpload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryUpload.failedTransfers().size();
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [UploadDirectory](#)를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon S3에 대용량 파일 업로드 또는 Amazon S3에서 대용량 파일 다운로드

다음 코드 예제는 Amazon S3에 대용량 파일을 업로드하고 Amazon S3에서 대용량 파일을 다운로드 하는 방법을 보여줍니다.

자세한 내용은 [멀티파트 업로드를 사용하여 객체 업로드](#)를 참조하십시오.

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 TransferUtility를 사용하여 S3 버킷과 파일을 주고받는 함수를 호출합니다.

```
global using System.Text;
global using Amazon.S3;
global using Amazon.S3.Model;
global using Amazon.S3.Transfer;
global using TransferUtilityBasics;

// This Amazon S3 client uses the default user credentials
// defined for this computer.
using Microsoft.Extensions.Configuration;

IAmazonS3 client = new AmazonS3Client();
var transferUtil = new TransferUtility(client);
IConfiguration _configuration;

_configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("settings.json") // Load test settings from JSON file.
    .AddJsonFile("settings.local.json",
        true) // Optionally load local settings.
    .Build();

// Edit the values in settings.json to use an S3 bucket and files that
// exist on your AWS account and on the local computer where you
// run this scenario.
var bucketName = _configuration["BucketName"];
var localPath =
    $"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\
\TransferFolder";

DisplayInstructions();

PressEnter();

Console.WriteLine();

// Upload a single file to an S3 bucket.
DisplayTitle("Upload a single file");

var fileToUpload = _configuration["FileToUpload"];
Console.WriteLine($"Uploading {fileToUpload} to the S3 bucket, {bucketName}.");
```

```
var success = await TransferMethods.UploadSingleFileAsync(transferUtil,
    bucketName, fileToUpload, localPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the file, {fileToUpload} to
    {bucketName}.");
}

PressEnter();

// Upload a local directory to an S3 bucket.
DisplayTitle("Upload all files from a local directory");
Console.WriteLine("Upload all the files in a local folder to an S3 bucket.");
const string keyPrefix = "UploadFolder";
var uploadPath = $"{localPath}\\UploadFolder";

Console.WriteLine($"Uploading the files in {uploadPath} to {bucketName}");
DisplayTitle($"{uploadPath} files");
DisplayLocalFiles(uploadPath);
Console.WriteLine();

PressEnter();

success = await TransferMethods.UploadFullDirectoryAsync(transferUtil,
    bucketName, keyPrefix, uploadPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the files in {uploadPath} to
    {bucketName}.");
    Console.WriteLine($"{bucketName} currently contains the following files:");
    await DisplayBucketFiles(client, bucketName, keyPrefix);
    Console.WriteLine();
}

PressEnter();

// Download a single file from an S3 bucket.
DisplayTitle("Download a single file");
Console.WriteLine("Now we will download a single file from an S3 bucket.");

var keyName = _configuration["FileToDownload"];

Console.WriteLine($"Downloading {keyName} from {bucketName}.");
```

```
success = await TransferMethods.DownloadSingleFileAsync(transferUtil, bucketName,
    keyName, localPath);
if (success)
{
    Console.WriteLine($"Successfully downloaded the file, {keyName} from
    {bucketName}.");
}

PressEnter();

// Download the contents of a directory from an S3 bucket.
DisplayTitle("Download the contents of an S3 bucket");
var s3Path = _configuration["S3Path"];
var downloadPath = $"{localPath}\\{s3Path}";

Console.WriteLine($"Downloading the contents of {bucketName}\\{s3Path}");
Console.WriteLine($"{bucketName}\\{s3Path} contains the following files:");
await DisplayBucketFiles(client, bucketName, s3Path);
Console.WriteLine();

success = await TransferMethods.DownloadS3DirectoryAsync(transferUtil,
    bucketName, s3Path, downloadPath);
if (success)
{
    Console.WriteLine($"Downloaded the files in {bucketName} to
    {downloadPath}.");
    Console.WriteLine($"{downloadPath} now contains the following files:");
    DisplayLocalFiles(downloadPath);
}

Console.WriteLine("\nThe TransferUtility Basics application has completed.");
PressEnter();

// Displays the title for a section of the scenario.
static void DisplayTitle(string titleText)
{
    var sepBar = new string('-', Console.WindowWidth);

    Console.WriteLine(sepBar);
    Console.WriteLine(CenterText(titleText));
    Console.WriteLine(sepBar);
}
```

```
// Displays a description of the actions to be performed by the scenario.
static void DisplayInstructions()
{
    var sepBar = new string('-', Console.WindowWidth);

    DisplayTitle("Amazon S3 Transfer Utility Basics");
    Console.WriteLine("This program shows how to use the Amazon S3 Transfer
Utility.");
    Console.WriteLine("It performs the following actions:");
    Console.WriteLine("\t1. Upload a single object to an S3 bucket.");
    Console.WriteLine("\t2. Upload an entire directory from the local computer to
an\n\t S3 bucket.");
    Console.WriteLine("\t3. Download a single object from an S3 bucket.");
    Console.WriteLine("\t4. Download the objects in an S3 bucket to a local
directory.");
    Console.WriteLine($"\\n{sepBar}");
}

// Pauses the scenario.
static void PressEnter()
{
    Console.WriteLine("Press <Enter> to continue.");
    _ = Console.ReadLine();
    Console.WriteLine("\\n");
}

// Returns the string textToCenter, padded on the left with spaces
// that center the text on the console display.
static string CenterText(string textToCenter)
{
    var centeredText = new StringBuilder();
    var screenWidth = Console.WindowWidth;
    centeredText.Append(new string(' ', (int)(screenWidth -
textToCenter.Length) / 2));
    centeredText.Append(textToCenter);
    return centeredText.ToString();
}

// Displays a list of file names included in the specified path.
static void DisplayLocalFiles(string localPath)
{
    var fileList = Directory.GetFiles(localPath);
    if (fileList.Length > 0)
    {
```

```

        foreach (var fileName in fileList)
        {
            Console.WriteLine(fileName);
        }
    }
}

// Displays a list of the files in the specified S3 bucket and prefix.
static async Task DisplayBucketFiles(IAmazonS3 client, string bucketName, string
    s3Path)
{
    ListObjectsV2Request request = new()
    {
        BucketName = bucketName,
        Prefix = s3Path,
        MaxKeys = 5,
    };

    var response = new ListObjectsV2Response();

    do
    {
        response = await client.ListObjectsV2Async(request);

        response.S3Objects
            .ForEach(obj => Console.WriteLine($"{obj.Key}"));

        // If the response is truncated, set the request ContinuationToken
        // from the NextContinuationToken property of the response.
        request.ContinuationToken = response.NextContinuationToken;
    } while (response.IsTruncated);
}

```

단일 파일을 업로드합니다.

```

/// <summary>
/// Uploads a single file from the local computer to an S3 bucket.
/// </summary>
/// <param name="transferUtil">The transfer initialized TransferUtility
/// object.</param>

```

```
    /// <param name="bucketName">The name of the S3 bucket where the file
    /// will be stored.</param>
    /// <param name="fileName">The name of the file to upload.</param>
    /// <param name="localPath">The local path where the file is stored.</
param>
    /// <returns>A boolean value indicating the success of the action.</
returns>
    public static async Task<bool> UploadSingleFileAsync(
        TransferUtility transferUtil,
        string bucketName,
        string fileName,
        string localPath)
    {
        if (File.Exists($"{localPath}\\{fileName}"))
        {
            try
            {
                await transferUtil.UploadAsync(new
TransferUtilityUploadRequest
                {
                    BucketName = bucketName,
                    Key = fileName,
                    FilePath = $"{localPath}\\{fileName}",
                });

                return true;
            }
            catch (AmazonS3Exception s3Ex)
            {
                Console.WriteLine($"Could not upload {fileName} from
{localPath} because:");
                Console.WriteLine(s3Ex.Message);
                return false;
            }
        }
        else
        {
            Console.WriteLine($"{fileName} does not exist in {localPath}");
            return false;
        }
    }
}
```

전체 로컬 디렉토리를 업로드합니다.

```

    /// <summary>
    /// Uploads all the files in a local directory to a directory in an S3
    /// bucket.
    /// </summary>
    /// <param name="transferUtil">The transfer initialized TransferUtility
    /// object.</param>
    /// <param name="bucketName">The name of the S3 bucket where the files
    /// will be stored.</param>
    /// <param name="keyPrefix">The key prefix is the S3 directory where
    /// the files will be stored.</param>
    /// <param name="localPath">The local directory that contains the files
    /// to be uploaded.</param>
    /// <returns>A Boolean value representing the success of the action.</
returns>
    public static async Task<bool> UploadFullDirectoryAsync(
        TransferUtility transferUtil,
        string bucketName,
        string keyPrefix,
        string localPath)
    {
        if (Directory.Exists(localPath))
        {
            try
            {
                await transferUtil.UploadDirectoryAsync(new
TransferUtilityUploadDirectoryRequest
                {
                    BucketName = bucketName,
                    KeyPrefix = keyPrefix,
                    Directory = localPath,
                });

                return true;
            }
            catch (AmazonS3Exception s3Ex)
            {
                Console.WriteLine($"Can't upload the contents of {localPath}
because:");

                Console.WriteLine(s3Ex?.Message);
                return false;
            }
        }
    }

```

```
    }
    else
    {
        Console.WriteLine($"The directory {localPath} does not exist.");
        return false;
    }
}
```

단일 파일을 다운로드합니다.

```
/// <summary>
/// Download a single file from an S3 bucket to the local computer.
/// </summary>
/// <param name="transferUtil">The transfer initialized TransferUtility
/// object.</param>
/// <param name="bucketName">The name of the S3 bucket containing the
/// file to download.</param>
/// <param name="keyName">The name of the file to download.</param>
/// <param name="localPath">The path on the local computer where the
/// downloaded file will be saved.</param>
/// <returns>A Boolean value indicating the results of the action.</
returns>
public static async Task<bool> DownloadSingleFileAsync(
    TransferUtility transferUtil,
    string bucketName,
    string keyName,
    string localPath)
{
    await transferUtil.DownloadAsync(new TransferUtilityDownloadRequest
    {
        BucketName = bucketName,
        Key = keyName,
        FilePath = $"{localPath}\\{keyName}",
    });

    return (File.Exists($"{localPath}\\{keyName}"));
}
```


S3 버킷의 콘텐츠를 다운로드합니다.

```

    /// <summary>
    /// Downloads the contents of a directory in an S3 bucket to a
    /// directory on the local computer.
    /// </summary>
    /// <param name="transferUtil">The transfer initialized TransferUtility
    /// object.</param>
    /// <param name="bucketName">The bucket containing the files to
download.</param>
    /// <param name="s3Path">The S3 directory where the files are located.</
param>
    /// <param name="localPath">The local path to which the files will be
    /// saved.</param>
    /// <returns>A Boolean value representing the success of the action.</
returns>
    public static async Task<bool> DownloadS3DirectoryAsync(
        TransferUtility transferUtil,
        string bucketName,
        string s3Path,
        string localPath)
    {
        int fileCount = 0;

        // If the directory doesn't exist, it will be created.
        if (Directory.Exists(s3Path))
        {
            var files = Directory.GetFiles(localPath);
            fileCount = files.Length;
        }

        await transferUtil.DownloadDirectoryAsync(new
TransferUtilityDownloadDirectoryRequest
        {
            BucketName = bucketName,
            LocalDirectory = localPath,
            S3Directory = s3Path,
        });

        if (Directory.Exists(localPath))
        {
            var files = Directory.GetFiles(localPath);
            if (files.Length > fileCount)

```

```

        {
            return true;
        }

        // No change in the number of files. Assume
        // the download failed.
        return false;
    }

    // The local directory doesn't exist. No files
    // were downloaded.
    return false;
}

```

TransferUtility를 사용하여 업로드 진행 상황을 추적합니다.

```

using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Transfer;

/// <summary>
/// This example shows how to track the progress of a multipart upload
/// using the Amazon Simple Storage Service (Amazon S3) TransferUtility to
/// upload to an Amazon S3 bucket.
/// </summary>
public class TrackMPUUsingHighLevelAPI
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "sample_pic.png";
        string path = "filepath/directory/";
        string filePath = $"{path}{keyName}";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
        IAmazonS3 client = new AmazonS3Client();
    }
}

```

```
        await TrackMPUAsync(client, bucketName, filePath, keyName);
    }

    /// <summary>
    /// Starts an Amazon S3 multipart upload and assigns an event handler to
    /// track the progress of the upload.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
    /// perform the multipart upload.</param>
    /// <param name="bucketName">The name of the bucket to which to upload
    /// the file.</param>
    /// <param name="filePath">The path, including the file name of the
    /// file to be uploaded to the Amazon S3 bucket.</param>
    /// <param name="keyName">The file name to be used in the
    /// destination Amazon S3 bucket.</param>
    public static async Task TrackMPUAsync(
        IAmazonS3 client,
        string bucketName,
        string filePath,
        string keyName)
    {
        try
        {
            var fileTransferUtility = new TransferUtility(client);

            // Use TransferUtilityUploadRequest to configure options.
            // In this example we subscribe to an event.
            var uploadRequest =
                new TransferUtilityUploadRequest
                {
                    BucketName = bucketName,
                    FilePath = filePath,
                    Key = keyName,
                };

            uploadRequest.UploadProgressEvent +=
                new EventHandler<UploadProgressArgs>(
                    UploadRequest_UploadPartProgressEvent);

            await fileTransferUtility.UploadAsync(uploadRequest);
            Console.WriteLine("Upload completed");
        }
        catch (AmazonS3Exception ex)
        {

```

```

        Console.WriteLine($"Error:: {ex.Message}");
    }
}

/// <summary>
/// Event handler to check the progress of the multipart upload.
/// </summary>
/// <param name="sender">The object that raised the event.</param>
/// <param name="e">The object that contains multipart upload
/// information.</param>
public static void UploadRequest_UploadPartProgressEvent(object sender,
UploadProgressArgs e)
{
    // Process event.
    Console.WriteLine($"{e.TransferredBytes}/{e.TotalBytes}");
}
}

```

암호화를 사용하여 객체를 업로드합니다.

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// Uses the Amazon Simple Storage Service (Amazon S3) low level API to
/// perform a multipart upload to an Amazon S3 bucket.
/// </summary>
public class SSECLowLevelMPUCopyObject
{
    public static async Task Main()
    {
        string existingBucketName = "amzn-s3-demo-bucket";
        string sourceKeyName = "sample_file.txt";
        string targetKeyName = "sample_file_copy.txt";
        string filePath = $"sample\\{targetKeyName}";
    }
}

```

```

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USEast1.
        IAmazonS3 client = new AmazonS3Client();

        // Create the encryption key.
        var base64Key = CreateEncryptionKey();

        await CreateSampleObjUsingClientEncryptionKeyAsync(
            client,
            existingBucketName,
            sourceKeyName,
            filePath,
            base64Key);
    }

    /// <summary>
    /// Creates the encryption key to use with the multipart upload.
    /// </summary>
    /// <returns>A string containing the base64-encoded key for encrypting
    /// the multipart upload.</returns>
    public static string CreateEncryptionKey()
    {
        Aes aesEncryption = Aes.Create();
        aesEncryption.KeySize = 256;
        aesEncryption.GenerateKey();
        string base64Key = Convert.ToBase64String(aesEncryption.Key);
        return base64Key;
    }

    /// <summary>
    /// Creates and uploads an object using a multipart upload.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 object used to
    /// initialize and perform the multipart upload.</param>
    /// <param name="existingBucketName">The name of the bucket to which
    /// the object will be uploaded.</param>
    /// <param name="sourceKeyName">The source object name.</param>
    /// <param name="filePath">The location of the source object.</param>
    /// <param name="base64Key">The encryption key to use with the upload.</
param>
    public static async Task CreateSampleObjUsingClientEncryptionKeyAsync(
        IAmazonS3 client,

```

```
        string existingBucketName,
        string sourceKeyName,
        string filePath,
        string base64Key)
    {
        List<UploadPartResponse> uploadResponses = new
List<UploadPartResponse>();

        InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
        {
            BucketName = existingBucketName,
            Key = sourceKeyName,
            ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
            ServerSideEncryptionCustomerProvidedKey = base64Key,
        };

        InitiateMultipartUploadResponse initResponse =
            await client.InitiateMultipartUploadAsync(initiateRequest);

        long contentLength = new FileInfo(filePath).Length;
        long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

        try
        {
            long filePosition = 0;
            for (int i = 1; filePosition < contentLength; i++)
            {
                UploadPartRequest uploadRequest = new UploadPartRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId,
                    PartNumber = i,
                    PartSize = partSize,
                    FilePosition = filePosition,
                    FilePath = filePath,
                    ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                    ServerSideEncryptionCustomerProvidedKey = base64Key,
                };

                // Upload part and add response to our list.
```

```
        uploadResponses.Add(await
client.UploadPartAsync(uploadRequest));

        filePosition += partSize;
    }

    CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
    {
        BucketName = existingBucketName,
        Key = sourceKeyName,
        UploadId = initResponse.UploadId,
    };
    completeRequest.AddPartETags(uploadResponses);

    CompleteMultipartUploadResponse completeUploadResponse =
        await client.CompleteMultipartUploadAsync(completeRequest);
    }
    catch (Exception exception)
    {
        Console.WriteLine($"Exception occurred: {exception.Message}");

        // If there was an error, abort the multipart upload.
        AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
        {
            BucketName = existingBucketName,
            Key = sourceKeyName,
            UploadId = initResponse.UploadId,
        };

        await client.AbortMultipartUploadAsync(abortMPURequest);
    }
}
}
```

Go

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

업로드 및 다운로드 관리자를 사용하여 데이터를 여러 부분으로 나누고 동시에 전송하는 함수를 생성합니다.

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "io"
    "log"
    "os"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}
```



```
// UploadLargeObject uses an upload manager to upload data to an object in a
// bucket.
// The upload manager breaks large data into parts and uploads the parts
// concurrently.
func (basics BucketBasics) UploadLargeObject(ctx context.Context, bucketName
string, objectKey string, largeObject []byte) error {
    largeBuffer := bytes.NewReader(largeObject)
    var partMiBs int64 = 10
    uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
        u.PartSize = partMiBs * 1024 * 1024
    })
    _, err := uploader.Upload(ctx, &s3.PutObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
        Body:    largeBuffer,
    })
    if err != nil {
        var apiErr smithy.APIError
        if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
            log.Printf("Error while uploading object to %s. The object is too large.\n"+
                "The maximum size for a multipart upload is 5TB.", bucketName)
        } else {
            log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
                bucketName, objectKey, err)
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
aws.String(objectKey)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
        }
    }

    return err
}

// DownloadLargeObject uses a download manager to download an object from a
// bucket.
// The download manager gets the data in parts and writes them to a buffer until
// all of
```

```
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
string, objectKey string) ([]byte, error) {
    var partMiBs int64 = 10
    downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
    {
        d.PartSize = partMiBs * 1024 * 1024
    })
    buffer := manager.NewWriteAtBuffer([]byte{})
    _, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
    })
    if err != nil {
        log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
            bucketName, objectKey, err)
    }
    return buffer.Bytes(), err
}
```

컨텍스트에서 업로드 및 다운로드 관리자를 사용하는 방법을 보여주는 대화형 시나리오를 실행합니다.

```
import (
    "context"
    "crypto/rand"
    "log"
    "strings"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunLargeObjectScenario is an interactive example that shows you how to use
// Amazon
// Simple Storage Service (Amazon S3) to upload and download large objects.
//
// 1. Create a bucket.
```

```
// 3. Upload a large object to the bucket by using an upload manager.
// 5. Download a large object by using a download manager.
// 8. Delete all objects in the bucket.
// 9. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig
// so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ../../demotools folder of this repo.
func RunLargeObjectScenario(ctx context.Context, sdkConfig aws.Config, questioner
demotools.IQuestioner) {
defer func() {
if r := recover(); r != nil {
log.Println("Something went wrong with the demo.")
_, isMock := questioner.(*demotools.MockQuestioner)
if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
log.Println(r)
}
}
}()

log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon S3 large object demo.")
log.Println(strings.Repeat("-", 88))

s3Client := s3.NewFromConfig(sdkConfig)
bucketBasics := actions.BucketBasics{S3Client: s3Client}

bucketName := questioner.Ask("Let's create a bucket. Enter a name for your
bucket:",
demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
if err != nil {
panic(err)
}
if !bucketExists {
err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
if err != nil {
panic(err)
} else {
```

```
    log.Println("Bucket created.")
}
}
log.Println(strings.Repeat("-", 88))

mibs := 30
log.Printf("Let's create a slice of %v MiB of random bytes and upload it to your
bucket. ", mibs)
questioner.Ask("Press Enter when you're ready.")
largeBytes := make([]byte, 1024*1024*mibs)
_, _ = rand.Read(largeBytes)
largeKey := "doc-example-large"
log.Println("Uploading...")
err = bucketBasics.UploadLargeObject(ctx, bucketName, largeKey, largeBytes)
if err != nil {
    panic(err)
}
log.Printf("Uploaded %v MiB object as %v", mibs, largeKey)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download the %v MiB object.", mibs)
questioner.Ask("Press Enter when you're ready.")
log.Println("Downloading...")
largeDownload, err := bucketBasics.DownloadLargeObject(ctx, bucketName,
largeKey)
if err != nil {
    panic(err)
}
log.Printf("Downloaded %v bytes.", len(largeDownload))
log.Println(strings.Repeat("-", 88))

if questioner.AskBool("Do you want to delete your bucket and all of its "+
"contents? (y/n)", "y") {
    log.Println("Deleting object.")
    err = bucketBasics.DeleteObjects(ctx, bucketName, []string{largeKey})
    if err != nil {
        panic(err)
    }
    log.Println("Deleting bucket.")
    err = bucketBasics.DeleteBucket(ctx, bucketName)
    if err != nil {
        panic(err)
    }
} else {
```

```

    log.Println("Okay. Don't forget to delete objects from your bucket to avoid
charges.")
}
log.Println(strings.Repeat("-", 88))

log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}

```

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

S3TransferManager를 사용하여 S3 버킷과 파일을 주고받는 함수를 호출합니다.

```

public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
    URI destinationPathURI, String bucketName) {
    DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
        .destination(Paths.get(destinationPathURI))
        .bucket(bucketName)
        .build());
    CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

    completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryDownload.failedTransfers().size();
}

```

전체 로컬 디렉토리를 업로드합니다.

```

public Integer uploadDirectory(S3TransferManager transferManager,
    URI sourceDirectory, String bucketName) {
    DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
        .source(Paths.get(sourceDirectory))
        .bucket(bucketName)
        .build());

    CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
    completedDirectoryUpload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryUpload.failedTransfers().size();
}

```

단일 파일을 업로드합니다.

```

public String uploadFile(S3TransferManager transferManager, String
bucketName,

    String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .source(Paths.get(filePathURI))
        .build();

    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

    CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
    return uploadResult.response().eTag();
}

```

코드 예제에서는 다음 가져오기를 사용합니다.

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3AsyncClient;

```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
```

콘텐츠 크기가 임계값을 초과할 때 [AWS CRT 기반 S3 클라이언트](#) 위에 있는 [S3 Transfer Manager](#)를 사용하여 멀티파트 업로드를 투명하게 수행할 수 있습니다. 기본 임계값 크기는 8MB입니다.

```
/**
 * Uploads a file to an Amazon S3 bucket using the S3TransferManager.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the object key
 * @param filePath    the file path of the file to be uploaded
 */
public void multipartUploadWithTransferManager(String bucketName, String key,
String filePath) {
    S3TransferManager transferManager = S3TransferManager.create();
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b
            .bucket(bucketName)
            .key(key))
```

```

        .source(Paths.get(filePath))
        .build();
    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
    fileUpload.completionFuture().join();
    transferManager.close();
}

```

멀티파트 업로드를 수행하려면 [S3Client API](#)를 사용합니다.

```

/**
 * Performs a multipart upload to Amazon S3 using the provided S3 client.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the object key
 * @param filePath    the path to the file to be uploaded
 */
public void multipartUploadWithS3Client(String bucketName, String key, String
filePath) {

    // Initiate the multipart upload.
    CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key));
    String uploadId = createMultipartUploadResponse.uploadId();

    // Upload the parts of the file.
    int partNumber = 1;
    List<CompletedPart> completedParts = new ArrayList<>();
    ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

    try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
        long fileSize = file.length();
        long position = 0;
        while (position < fileSize) {
            file.seek(position);
            long read = file.getChannel().read(bb);

            bb.flip(); // Swap position and limit before reading from the
buffer.

            UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                .bucket(bucketName)

```



```

        .key(key)
        .uploadId(uploadId)
        .partNumber(partNumber)
        .build();

        UploadPartResponse partResponse = s3Client.uploadPart(
            uploadPartRequest,
            RequestBody.fromByteBuffer(bb));

        CompletedPart part = CompletedPart.builder()
            .partNumber(partNumber)
            .eTag(partResponse.eTag())
            .build();
        completedParts.add(part);

        bb.clear();
        position += read;
        partNumber++;
    }
} catch (IOException e) {
    logger.error(e.getMessage());
}

// Complete the multipart upload.
s3Client.completeMultipartUpload(b -> b
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)

    .multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
}

```

멀티파트 지원이 활성화된 [S3AsyncClient API](#)를 사용하여 멀티파트 업로드를 수행합니다.

```

/**
 * Uploads a file to an S3 bucket using the S3AsyncClient and enabling
 * multipart support.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the object key
 * @param filePath    the local file path of the file to be uploaded

```

```

    */
    public void multipartUploadWithS3AsyncClient(String bucketName, String key,
String filePath) {
        // Enable multipart support.
        S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
            .multipartEnabled(true)
            .build();

        CompletableFuture<PutObjectResponse> response = s3AsyncClient.putObject(b
-> b
            .bucket(bucketName)
            .key(key),
            Paths.get(filePath));

        response.join();
        logger.info("File uploaded in multiple 8 MiB parts using
S3AsyncClient.");
    }

```

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

대용량 파일을 업로드합니다.

```

import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

import {
    ProgressBar,
    logger,
} from "@aws-doc-sdk-examples/lib/utils/util-log.js";

const twentyFiveMB = 25 * 1024 * 1024;

```

```
export const createString = (size = twentyFiveMB) => {
  return "x".repeat(size);
};

/**
 * Create a 25MB file and upload it in parts to the specified
 * Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const str = createString();
  const buffer = Buffer.from(str, "utf8");
  const progressBar = new ProgressBar({
    description: `Uploading "${key}" to "${bucketName}"`,
    barLength: 30,
  });

  try {
    const upload = new Upload({
      client: new S3Client({}),
      params: {
        Bucket: bucketName,
        Key: key,
        Body: buffer,
      },
    });

    upload.on("httpUploadProgress", ({ loaded, total }) => {
      progressBar.update({ current: loaded, total });
    });

    await upload.done();
  } catch (caught) {
    if (caught instanceof Error && caught.name === "AbortError") {
      logger.error(`Multipart upload was aborted. ${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

대용량 파일을 다운로드합니다.

```
import { fileURLToPath } from "node:url";
import { GetObjectCommand, NoSuchKey, S3Client } from "@aws-sdk/client-s3";
import { createWriteStream, rmSync } from "node:fs";

const s3Client = new S3Client({});
const oneMB = 1024 * 1024;

export const getObjectRange = ({ bucket, key, start, end }) => {
  const command = new GetObjectCommand({
    Bucket: bucket,
    Key: key,
    Range: `bytes=${start}-${end}`,
  });

  return s3Client.send(command);
};

/**
 * @param {string | undefined} contentRange
 */
export const getRangeAndLength = (contentRange) => {
  const [range, length] = contentRange.split("/");
  const [start, end] = range.split("-");
  return {
    start: Number.parseInt(start),
    end: Number.parseInt(end),
    length: Number.parseInt(length),
  };
};

export const isComplete = ({ end, length }) => end === length - 1;

const downloadInChunks = async ({ bucket, key }) => {
  const writeStream = createWriteStream(
    fileURLToPath(new URL(`./${key}`, import.meta.url)),
  ).on("error", (err) => console.error(err));

  let rangeAndLength = { start: -1, end: -1, length: -1 };

  while (!isComplete(rangeAndLength)) {
    const { end } = rangeAndLength;
    const nextRange = { start: end + 1, end: end + oneMB };
  }
}
```

```
const { ContentRange, Body } = await getObjectRange({
  bucket,
  key,
  ...nextRange,
});
console.log(`Downloaded bytes ${nextRange.start} to ${nextRange.end}`);

writeStream.write(await Body.transformToByteArray());
rangeAndLength = getRangeAndLength(ContentRange);
}
};

/**
 * Download a large object from and Amazon S3 bucket.
 *
 * When downloading a large file, you might want to break it down into
 * smaller pieces. Amazon S3 accepts a Range header to specify the start
 * and end of the byte range to be downloaded.
 *
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  try {
    await downloadInChunks({
      bucket: bucketName,
      key: key,
    });
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(`Failed to download object. No such key "${key}".`);
      rmSync(key);
    }
  }
};
```

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

사용 가능한 여러 전송 관리자 설정을 사용하여 파일을 전송하는 함수를 생성합니다. 콜백 클래스를 사용하여 파일 전송 중에 콜백 진행률을 작성합니다.

```
import sys
import threading

import boto3
from boto3.s3.transfer import TransferConfig

MB = 1024 * 1024
s3 = boto3.resource("s3")

class TransferCallback:
    """
    Handle callbacks from the transfer manager.

    The transfer manager periodically calls the __call__ method throughout
    the upload and download process so that it can take action, such as
    displaying progress to the user and collecting data about the transfer.
    """

    def __init__(self, target_size):
        self._target_size = target_size
        self._total_transferred = 0
        self._lock = threading.Lock()
        self.thread_info = {}

    def __call__(self, bytes_transferred):
        """
        The callback method that is called by the transfer manager.
```

Display progress during file transfer and collect per-thread transfer data. This method can be called by multiple threads, so shared instance data is protected by a thread lock.

```

"""
thread = threading.current_thread()
with self._lock:
    self._total_transferred += bytes_transferred
    if thread.ident not in self.thread_info.keys():
        self.thread_info[thread.ident] = bytes_transferred
    else:
        self.thread_info[thread.ident] += bytes_transferred

    target = self._target_size * MB
    sys.stdout.write(
        f"\r{self._total_transferred} of {target} transferred "
        f"({(self._total_transferred / target) * 100:.2f}%)."
    )
    sys.stdout.flush()

```

```

def upload_with_default_configuration(
    local_file_path, bucket_name, object_key, file_size_mb
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, using the default
    configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Callback=transfer_callback
    )
    return transfer_callback.thread_info

```

```

def upload_with_chunksize_and_meta(
    local_file_path, bucket_name, object_key, file_size_mb, metadata=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart chunk size and adding metadata to the Amazon S3 object.

```

The multipart chunk size controls the size of the chunks of data that are sent in the request. A smaller chunk size typically results in the transfer

manager using more threads for the upload.

The metadata is a set of key-value pairs that are stored with the object in Amazon S3.

```
"""
```

```
transfer_callback = TransferCallback(file_size_mb)
```

```
config = TransferConfig(multipart_chunksize=1 * MB)
```

```
extra_args = {"Metadata": metadata} if metadata else None
```

```
s3.Bucket(bucket_name).upload_file(
```

```
    local_file_path,
```

```
    object_key,
```

```
    Config=config,
```

```
    ExtraArgs=extra_args,
```

```
    Callback=transfer_callback,
```

```
)
```

```
return transfer_callback.thread_info
```

```
def upload_with_high_threshold(local_file_path, bucket_name, object_key,
    file_size_mb):
```

```
    """
```

Upload a file from a local folder to an Amazon S3 bucket, setting a multipart threshold larger than the size of the file.

Setting a multipart threshold larger than the size of the file results in the transfer manager sending the file as a standard upload instead of a multipart upload.

```
    """
```

```
transfer_callback = TransferCallback(file_size_mb)
```

```
config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
```

```
s3.Bucket(bucket_name).upload_file(
```

```
    local_file_path, object_key, Config=config, Callback=transfer_callback
```

```
)
```

```
return transfer_callback.thread_info
```

```
def upload_with_sse(
```

```
    local_file_path, bucket_name, object_key, file_size_mb, sse_key=None
```

```
):
```

```
    """
```

Upload a file from a local folder to an Amazon S3 bucket, adding server-side encryption with customer-provided encryption keys to the object.


```
When this kind of encryption is specified, Amazon S3 encrypts the object
at rest and allows downloads only when the expected encryption key is
provided in the download request.
"""

transfer_callback = TransferCallback(file_size_mb)
if sse_key:
    extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey":
sse_key}
else:
    extra_args = None
s3.Bucket(bucket_name).upload_file(
    local_file_path, object_key, ExtraArgs=extra_args,
    Callback=transfer_callback
)
return transfer_callback.thread_info

def download_with_default_configuration(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using the
    default configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Callback=transfer_callback
    )
    return transfer_callback.thread_info

def download_with_single_thread(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using a
    single thread.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(use_threads=False)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

```
def download_with_high_threshold(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard download instead
    of a multipart download.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_sse(
    bucket_name, object_key, download_file_path, file_size_mb, sse_key
):
    """
    Download a file from an Amazon S3 bucket to a local folder, adding a
    customer-provided encryption key to the request.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)

    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey":
sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, ExtraArgs=extra_args, Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

전송 관리자 기능을 시연하고 결과를 보고합니다.

```
import hashlib
import os
import platform
import shutil
import time

import boto3
from boto3.s3.transfer import TransferConfig
from botocore.exceptions import ClientError
from botocore.exceptions import ParamValidationError
from botocore.exceptions import NoCredentialsError

import file_transfer

MB = 1024 * 1024
# These configuration attributes affect both uploads and downloads.
CONFIG_ATTRS = (
    "multipart_threshold",
    "multipart_chunksize",
    "max_concurrency",
    "use_threads",
)
# These configuration attributes affect only downloads.
DOWNLOAD_CONFIG_ATTRS = ("max_io_queue", "io_chunksize", "num_download_attempts")

class TransferDemoManager:
    """
    Manages the demonstration. Collects user input from a command line, reports
    transfer results, maintains a list of artifacts created during the
    demonstration, and cleans them up after the demonstration is completed.
    """

    def __init__(self):
        self._s3 = boto3.resource("s3")
        self._chore_list = []
        self._create_file_cmd = None
        self._size_multiplier = 0
```

```

        self.file_size_mb = 30
        self.demo_folder = None
        self.demo_bucket = None
        self._setup_platform_specific()
        self._terminal_width = shutil.get_terminal_size(fallback=(80, 80))[0]

    def collect_user_info(self):
        """
        Collect local folder and Amazon S3 bucket name from the user. These
        locations are used to store files during the demonstration.
        """
        while not self.demo_folder:
            self.demo_folder = input(
                "Which file folder do you want to use to store " "demonstration
files? "
            )
            if not os.path.isdir(self.demo_folder):
                print(f"{self.demo_folder} isn't a folder!")
                self.demo_folder = None

        while not self.demo_bucket:
            self.demo_bucket = input(
                "Which Amazon S3 bucket do you want to use to store "
"demonstration files? "
            )
            try:
                self._s3.meta.client.head_bucket(Bucket=self.demo_bucket)
            except ParamValidationError as err:
                print(err)
                self.demo_bucket = None
            except ClientError as err:
                print(err)
                print(
                    f"Either {self.demo_bucket} doesn't exist or you don't "
                    f"have access to it."
                )
                self.demo_bucket = None

    def demo(
        self, question, upload_func, download_func, upload_args=None,
download_args=None
    ):
        """Run a demonstration.

```

```

Ask the user if they want to run this specific demonstration.
If they say yes, create a file on the local path, upload it
using the specified upload function, then download it using the
specified download function.
"""
if download_args is None:
    download_args = {}
if upload_args is None:
    upload_args = {}
question = question.format(self.file_size_mb)
answer = input(f"{question} (y/n)")
if answer.lower() == "y":
    local_file_path, object_key, download_file_path =
self._create_demo_file()

    file_transfer.TransferConfig = self._config_wrapper(
        TransferConfig, CONFIG_ATTRS
    )
    self._report_transfer_params(
        "Uploading", local_file_path, object_key, **upload_args
    )
    start_time = time.perf_counter()
    thread_info = upload_func(
        local_file_path,
        self.demo_bucket,
        object_key,
        self.file_size_mb,
        **upload_args,
    )
    end_time = time.perf_counter()
    self._report_transfer_result(thread_info, end_time - start_time)

    file_transfer.TransferConfig = self._config_wrapper(
        TransferConfig, CONFIG_ATTRS + DOWNLOAD_CONFIG_ATTRS
    )
    self._report_transfer_params(
        "Downloading", object_key, download_file_path, **download_args
    )
    start_time = time.perf_counter()
    thread_info = download_func(
        self.demo_bucket,
        object_key,
        download_file_path,
        self.file_size_mb,

```

```

        **download_args,
    )
    end_time = time.perf_counter()
    self._report_transfer_result(thread_info, end_time - start_time)

def last_name_set(self):
    """Get the name set used for the last demo."""
    return self._chore_list[-1]

def cleanup(self):
    """
    Remove files from the demo folder, and uploaded objects from the
    Amazon S3 bucket.
    """
    print("-" * self._terminal_width)
    for local_file_path, s3_object_key, downloaded_file_path in
self._chore_list:
        print(f"Removing {local_file_path}")
        try:
            os.remove(local_file_path)
        except FileNotFoundError as err:
            print(err)

        print(f"Removing {downloaded_file_path}")
        try:
            os.remove(downloaded_file_path)
        except FileNotFoundError as err:
            print(err)

        if self.demo_bucket:
            print(f"Removing {self.demo_bucket}:{s3_object_key}")
            try:
                self._s3.Bucket(self.demo_bucket).Object(s3_object_key).delete()
            except ClientError as err:
                print(err)

def _setup_platform_specific(self):
    """Set up platform-specific command used to create a large file."""
    if platform.system() == "Windows":
        self._create_file_cmd = "fsutil file createnew {} {}"
        self._size_multiplier = MB
    elif platform.system() == "Linux" or platform.system() == "Darwin":

```

```

        self._create_file_cmd = f"dd if=/dev/urandom of={{}} " f"bs={MB}
count={{}}"
        self._size_multiplier = 1
    else:
        raise EnvironmentError(
            f"Demo of platform {platform.system()} isn't supported."
        )

def _create_demo_file(self):
    """
    Create a file in the demo folder specified by the user. Store the local
    path, object name, and download path for later cleanup.

    Only the local file is created by this method. The Amazon S3 object and
    download file are created later during the demonstration.

    Returns:
    A tuple that contains the local file path, object name, and download
    file path.
    """
    file_name_template = "TestFile{}-{}.demo"
    local_suffix = "local"
    object_suffix = "s3object"
    download_suffix = "downloaded"
    file_tag = len(self._chore_list) + 1

    local_file_path = os.path.join(
        self.demo_folder, file_name_template.format(file_tag, local_suffix)
    )

    s3_object_key = file_name_template.format(file_tag, object_suffix)

    downloaded_file_path = os.path.join(
        self.demo_folder, file_name_template.format(file_tag,
download_suffix)
    )

    filled_cmd = self._create_file_cmd.format(
        local_file_path, self.file_size_mb * self._size_multiplier
    )

    print(
        f"Creating file of size {self.file_size_mb} MB "
        f"in {self.demo_folder} by running:"
    )

```

```

    )
    print(f"{'':4}{filled_cmd}")
    os.system(filled_cmd)

    chore = (local_file_path, s3_object_key, downloaded_file_path)
    self._chore_list.append(chore)
    return chore

def _report_transfer_params(self, verb, source_name, dest_name, **kwargs):
    """Report configuration and extra arguments used for a file transfer."""
    print("-" * self._terminal_width)
    print(f"{verb} {source_name} ({self.file_size_mb} MB) to {dest_name}")
    if kwargs:
        print("With extra args:")
        for arg, value in kwargs.items():
            print(f"{'':4}{arg:<20}: {value}")

@staticmethod
def ask_user(question):
    """
    Ask the user a yes or no question.

    Returns:
    True when the user answers 'y' or 'Y'; otherwise, False.
    """
    answer = input(f"{question} (y/n) ")
    return answer.lower() == "y"

@staticmethod
def _config_wrapper(func, config_attrs):
    def wrapper(*args, **kwargs):
        config = func(*args, **kwargs)
        print("With configuration:")
        for attr in config_attrs:
            print(f"{'':4}{attr:<20}: {getattr(config, attr)}")
        return config

    return wrapper

@staticmethod
def _report_transfer_result(thread_info, elapsed):
    """Report the result of a transfer, including per-thread data."""
    print(f"\nUsed {len(thread_info)} threads.")
    for ident, byte_count in thread_info.items():

```



```
        print(f"{'':4}Thread {ident} copied {byte_count} bytes.")
    print(f"Your transfer took {elapsed:.2f} seconds.")

def main():
    """
    Run the demonstration script for s3_file_transfer.
    """
    demo_manager = TransferDemoManager()
    demo_manager.collect_user_info()

    # Upload and download with default configuration. Because the file is 30 MB
    # and the default multipart_threshold is 8 MB, both upload and download are
    # multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "using the default configuration?",
        file_transfer.upload_with_default_configuration,
        file_transfer.download_with_default_configuration,
    )

    # Upload and download with multipart_threshold set higher than the size of
    # the file. This causes the transfer manager to use standard transfers
    # instead of multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "as a standard (not multipart) transfer?",
        file_transfer.upload_with_high_threshold,
        file_transfer.download_with_high_threshold,
    )

    # Upload with specific chunk size and additional metadata.
    # Download with a single thread.
    demo_manager.demo(
        "Do you want to upload a {} MB file with a smaller chunk size and "
        "then download the same file using a single thread?",
        file_transfer.upload_with_chunksize_and_meta,
        file_transfer.download_with_single_thread,
        upload_args={
            "metadata": {
                "upload_type": "chunky",
                "favorite_color": "aqua",
                "size": "medium",
            }
        }
    )
```

```

    },
)

# Upload using server-side encryption with customer-provided
# encryption keys.
# Generate a 256-bit key from a passphrase.
sse_key = hashlib.sha256("demo_passphrase".encode("utf-8")).digest()
demo_manager.demo(
    "Do you want to upload and download a {} MB file using "
    "server-side encryption?",
    file_transfer.upload_with_sse,
    file_transfer.download_with_sse,
    upload_args={"sse_key": sse_key},
    download_args={"sse_key": sse_key},
)

# Download without specifying an encryption key to show that the
# encryption key must be included to download an encrypted object.
if demo_manager.ask_user(
    "Do you want to try to download the encrypted "
    "object without sending the required key?"
):
    try:
        _, object_key, download_file_path = demo_manager.last_name_set()
        file_transfer.download_with_default_configuration(
            demo_manager.demo_bucket,
            object_key,
            download_file_path,
            demo_manager.file_size_mb,
        )
    except ClientError as err:
        print(
            "Got expected error when trying to download an encrypted "
            "object without specifying encryption info:"
        )
        print(f"{'':4}{err}")

# Remove all created and downloaded files, remove all objects from
# S3 storage.
if demo_manager.ask_user(
    "Demonstration complete. Do you want to remove local files " "and S3
objects?"
):
    demo_manager.cleanup()

```

```

if __name__ == "__main__":
    try:
        main()
    except NoCredentialsError as error:
        print(error)
        print(
            "To run this example, you must have valid credentials in "
            "a shared credential file or set in environment variables."
        )

```

Rust

SDK for Rust

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

use std::fs::File;
use std::io::prelude::*;
use std::path::Path;

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::error::DisplayErrorContext;
use aws_sdk_s3::operation::{
    create_multipart_upload::CreateMultipartUploadOutput,
    get_object::GetObjectOutput,
};
use aws_sdk_s3::types::{CompletedMultipartUpload, CompletedPart};
use aws_sdk_s3::{config::Region, Client as S3Client};
use aws_smithy_types::byte_stream::{ByteStream, Length};
use rand::distributions::Alphanumeric;
use rand::{thread_rng, Rng};
use s3_code_examples::error::S3ExampleError;
use std::process;

```

```

use uuid::Uuid;

//In bytes, minimum chunk size of 5MB. Increase CHUNK_SIZE to send larger chunks.
const CHUNK_SIZE: u64 = 1024 * 1024 * 5;
const MAX_CHUNKS: u64 = 10000;

#[tokio::main]
pub async fn main() {
    if let Err(err) = run_example().await {
        eprintln!("Error: {}", DisplayErrorContext(err));
        process::exit(1);
    }
}

async fn run_example() -> Result<(), S3ExampleError> {
    let shared_config = aws_config::load_from_env().await;
    let client = S3Client::new(&shared_config);

    let bucket_name = format!("amzn-s3-demo-bucket-{}", Uuid::new_v4());
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));
    let region = region_provider.region().await.unwrap();
    s3_code_examples::create_bucket(&client, &bucket_name, &region).await?;

    let key = "sample.txt".to_string();
    // Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
    // upload the file.
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await?;

    let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
        "Missing upload_id after CreateMultipartUpload",
    ));

    //Create a file of random characters for the upload.
    let mut file = File::create(&key).expect("Could not create sample file.");
    // Loop until the file is 5 chunks.
    while file.metadata().unwrap().len() <= CHUNK_SIZE * 4 {
        let rand_string: String = thread_rng()
            .sample_iter(&Alphanumeric)

```

```

        .take(256)
        .map(char::from)
        .collect();
    let return_string: String = "\n".to_string();
    file.write_all(rand_string.as_ref())
        .expect("Error writing to file.");
    file.write_all(return_string.as_ref())
        .expect("Error writing to file.");
}

let path = Path::new(&key);
let file_size = tokio::fs::metadata(path)
    .await
    .expect("it exists I swear")
    .len();

let mut chunk_count = (file_size / CHUNK_SIZE) + 1;
let mut size_of_last_chunk = file_size % CHUNK_SIZE;
if size_of_last_chunk == 0 {
    size_of_last_chunk = CHUNK_SIZE;
    chunk_count -= 1;
}

if file_size == 0 {
    return Err(S3ExampleError::new("Bad file size."));
}
if chunk_count > MAX_CHUNKS {
    return Err(S3ExampleError::new(
        "Too many chunks! Try increasing your chunk size.",
    ));
}

let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))

```

```

        .build()
        .await
        .unwrap();

// Chunk index needs to start at 0, but part numbers start at 1.
let part_number = (chunk_index as i32) + 1;
let upload_part_res = client
    .upload_part()
    .key(&key)
    .bucket(&bucket_name)
    .upload_id(upload_id)
    .body(stream)
    .part_number(part_number)
    .send()
    .await?;

upload_parts.push(
    CompletedPart::builder()
        .e_tag(upload_part_res.e_tag.unwrap_or_default())
        .part_number(part_number)
        .build(),
);
}

// upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
    .set_parts(Some(upload_parts))
    .build();

let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await?;

let data: GetObjectOutput =
    s3_code_examples::download_object(&client, &bucket_name, &key).await?;
let data_length: u64 = data
    .content_length()
    .unwrap_or_default()

```

```

        .try_into()
        .unwrap();
    if file.metadata().unwrap().len() == data_length {
        println!("Data lengths match.");
    } else {
        println!("The data was not the same size!");
    }

    s3_code_examples::clear_bucket(&client, &bucket_name)
        .await
        .expect("Error emptying bucket.");
    s3_code_examples::delete_bucket(&client, &bucket_name)
        .await
        .expect("Error deleting bucket.");

    Ok(())
}

```

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#) 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon S3 객체에 알 수 없는 크기의 스트림 업로드

다음 코드 예제에서는 Amazon S3 객체에 알 수 없는 크기의 스트림을 업로드하는 방법을 보여줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

[AWS CRT 기반 S3 클라이언트](#)를 사용합니다.

```
import com.example.s3.util.AsyncExampleUtils;
```

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class PutObjectFromStreamAsync {
    private static final Logger logger =
        LoggerFactory.getLogger(PutObjectFromStreamAsync.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.

        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            PutObjectFromStreamAsync example = new PutObjectFromStreamAsync();
            S3AsyncClient s3AsyncClientCrt = S3AsyncClient.crtCreate();
            PutObjectResponse putObjectResponse =
                example.putObjectFromStreamCrt(s3AsyncClientCrt, bucketName, key);
            logger.info("Object {} etag: {}", key, putObjectResponse.eTag());
            logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
    }

    /**
     * @param s3CrtAsyncClient - To upload content from a stream of unknown
     * size, use can the AWS CRT-based S3 client.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     */
}
```



```

    * @return software.amazon.awssdk.services.s3.model.PutObjectResponse -
    Returns metadata pertaining to the put object operation.
    */
    public PutObjectResponse putObjectFromStreamCrt(S3AsyncClient
s3CrtAsyncClient, String bucketName, String key) {

        // AsyncExampleUtils.randomString() returns a random string up to 100
        characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
randomString.length());
        InputStream inputStream = new
ByteArrayInputStream(randomString.getBytes());

        // Executor required to handle reading from the InputStream on a separate
        thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
        length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream,
null, executor);

        CompletableFuture<PutObjectResponse> responseFuture =
            s3CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
body);

        PutObjectResponse response = responseFuture.join(); // Wait for the
        response.
        logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
        executor.shutdown();
        return response;
    }
}

```

[멀티파트 업로드가 활성화된 표준 비동기 S3 클라이언트](#)를 사용합니다.

```

import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;

```

```
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class PutObjectFromStreamAsyncMp {
    private static final Logger logger =
        LoggerFactory.getLogger(PutObjectFromStreamAsyncMp.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.

        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            PutObjectFromStreamAsyncMp example = new
            PutObjectFromStreamAsyncMp();
            S3AsyncClient s3AsyncClientMp =
            S3AsyncClient.builder().multipartEnabled(true).build();
            PutObjectResponse putObjectResponse =
            example.putObjectFromStreamMp(s3AsyncClientMp, bucketName, key);
            logger.info("Object {} etag: {}", key, putObjectResponse.eTag());
            logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
    }

    /**
     * @param s3AsyncClientMp - To upload content from a stream of unknown size,
     use can the S3 asynchronous client with multipart enabled.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return software.amazon.awssdk.services.s3.model.PutObjectResponse -
     Returns metadata pertaining to the put object operation.
    */
}
```

```

    */
    public PutObjectResponse putObjectFromStreamMp(S3AsyncClient s3AsyncClientMp,
        String bucketName, String key) {

        // AsyncExampleUtils.randomString() returns a random string up to 100
        characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
            randomString.length());
        InputStream inputStream = new
        ByteArrayInputStream(randomString.getBytes());

        // Executor required to handle reading from the InputStream on a separate
        thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
        length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream,
            null, executor);

        CompletableFuture<PutObjectResponse> responseFuture =
            s3AsyncClientMp.putObject(r -> r.bucket(bucketName).key(key),
            body);

        PutObjectResponse response = responseFuture.join(); // Wait for the
        response.
        logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
        executor.shutdown();
        return response;
    }
}

```

[Amazon S3 Transfer Manager](#)를 사용합니다.

```

import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedUpload;

```

```
import software.amazon.awssdk.transfer.s3.model.Upload;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class UploadStream {
    private static final Logger logger =
        LoggerFactory.getLogger(UploadStream.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.

        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            UploadStream example = new UploadStream();
            CompletedUpload completedUpload =
                example.uploadStream(S3TransferManager.create(), bucketName, key);
            logger.info("Object {} etag: {}", key,
                completedUpload.response().eTag());
            logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
    }

    /**
     * @param transferManager - To upload content from a stream of unknown size,
     * you can use the S3TransferManager based on the AWS CRT-based S3 client.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return - software.amazon.awssdk.transfer.s3.model.CompletedUpload - The
     * result of the completed upload.
     */
    public CompletedUpload uploadStream(S3TransferManager transferManager, String
        bucketName, String key) {
```

```

        // AsyncExampleUtils.randomString() returns a random string up to 100
        characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
randomString.length());
        InputStream inputStream = new
        ByteArrayInputStream(randomString.getBytes());

        // Executor required to handle reading from the InputStream on a separate
        thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
        length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream,
        null, executor);

        Upload upload = transferManager.upload(builder -> builder
            .requestBody(body)
            .putObjectRequest(req -> req.bucket(bucketName).key(key))
            .build());

        CompletedUpload completedUpload = upload.completionFuture().join();
        executor.shutdown();
        return completedUpload;
    }
}

```

Swift

SDK for Swift

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

import ArgumentParser
import AWSClientRuntime
import AWSS3

```

```
import Foundation
import Smithy
import SmithyHTTPAPI
import SmithyStreams

/// Upload a file to the specified bucket.
///
/// - Parameters:
///   - bucket: The Amazon S3 bucket name to store the file into.
///   - key: The name (or path) of the file to upload to in the `bucket`.
///   - sourcePath: The pathname on the local filesystem of the file to
///     upload.
func uploadFile(sourcePath: String, bucket: String, key: String?) async
throws {
    let fileURL: URL = URL(fileURLWithPath: sourcePath)
    let fileName: String

    // If no key was provided, use the last component of the filename.

    if key == nil {
        fileName = fileURL.lastPathComponent
    } else {
        fileName = key!
    }

    let s3Client = try await S3Client()

    // Create a FileHandle for the source file.

    let fileHandle = FileHandle(forReadingAtPath: sourcePath)
    guard let fileHandle = fileHandle else {
        throw TransferError.readError
    }

    // Create a byte stream to retrieve the file's contents. This uses the
    // Smithy FileStream and ByteStream types.

    let stream = FileStream(fileHandle: fileHandle)
    let body = ByteStream.stream(stream)

    // Create a `PutObjectInput` with the ByteStream as the body of the
    // request's data. The AWS SDK for Swift will handle sending the
    // entire file in chunks, regardless of its size.
```

```
let putInput = PutObjectInput(  
    body: body,  
    bucket: bucket,  
    key: fileName  
)  
  
do {  
    _ = try await s3Client.putObject(input: putInput)  
} catch {  
    throw TransferError.uploadError("Error uploading the file: \(error)")  
}  
  
print("File uploaded to \(fileURL.path).")  
}
```

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#) 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon S3 객체 작업 수행

다음 코드 예제는 체크섬을 사용하여 Amazon S3 객체로 작업하는 방법을 보여줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

코드 예제에서는 다음 가져오기 하위 집합을 사용합니다.

```
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import software.amazon.awssdk.core.exception.SdkException;
```

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.ChecksumMode;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.security.DigestInputStream;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Base64;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
```

[PutObjectRequest](#)를 빌드할 때 putObject 메서드에 대한 체크섬 알고리즘을 지정합니다.

```
public void putObjectWithChecksum(String bucketName, String key) {
    s3Client.putObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(ChecksumAlgorithm.CRC32),
        RequestBody.fromString("This is a test"));
}
```


[GetObjectRequest](#)를 구축할 때 getObject 메서드의 체크섬을 확인합니다.

```
public GetObjectResponse getObjectWithChecksum(String bucketName, String key)
{
    return s3Client.getObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumMode(ChecksumMode.ENABLED))
        .response();
}
```

[PutObjectRequest](#)를 빌드할 때 putObject 메서드에 대한 체크섬을 미리 계산합니다.

```
public void putObjectWithPrecalculatedChecksum(String bucketName, String key,
String filePath) {
    String checksum = calculateChecksum(filePath, "SHA-256");

    s3Client.putObject((b -> b
        .bucket(bucketName)
        .key(key)
        .checksumSHA256(checksum)),
        RequestBody.fromFile(Paths.get(filePath)));
}
```

콘텐츠 크기가 임계값을 초과할 때 [AWS CRT 기반 S3 클라이언트](#) 위에 있는 [S3 Transfer Manager](#)를 사용하여 멀티파트 업로드를 투명하게 수행할 수 있습니다. 기본 임계값 크기는 8MB입니다.

SDK에서 사용할 체크섬 알고리즘을 지정할 수 있습니다. 기본적으로 SDK는 CRC32 알고리즘을 사용합니다.

```
public void multipartUploadWithChecksumTm(String bucketName, String key,
String filePath) {
    S3TransferManager transferManager = S3TransferManager.create();
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b
            .bucket(bucketName)
            .key(key)
            .checksumAlgorithm(ChecksumAlgorithm.SHA1))
        .source(Paths.get(filePath))
```

```

        .build();
        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
        fileUpload.completionFuture().join();
        transferManager.close();
    }

```

멀티파트 업로드를 수행하려면 [S3Client API](#) 또는 (S3AsyncClient API)를 사용합니다. 추가 체크섬을 지정하는 경우 업로드를 시작할 때 사용할 알고리즘을 지정해야 합니다. 또한 각 파트 요청에 대한 알고리즘을 지정하고 업로드 후 각 파트에 대해 계산된 체크섬을 제공해야 합니다.

```

public void multipartUploadWithChecksumS3Client(String bucketName, String
key, String filePath) {
    ChecksumAlgorithm algorithm = ChecksumAlgorithm.CRC32;

    // Initiate the multipart upload.
    CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(algorithm)); // Checksum specified on initiation.
    String uploadId = createMultipartUploadResponse.uploadId();

    // Upload the parts of the file.
    int partNumber = 1;
    List<CompletedPart> completedParts = new ArrayList<>();
    ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

    try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
        long fileSize = file.length();
        long position = 0;
        while (position < fileSize) {
            file.seek(position);
            long read = file.getChannel().read(bb);

            bb.flip(); // Swap position and limit before reading from the
buffer.

            UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .checksumAlgorithm(algorithm) // Checksum specified on each
part.

```

```

        .partNumber(partNumber)
        .build();

UploadPartResponse partResponse = s3Client.uploadPart(
    uploadPartRequest,
    RequestBody.fromByteBuffer(bb));

CompletedPart part = CompletedPart.builder()
    .partNumber(partNumber)
    .checksumCRC32(partResponse.checksumCRC32()) // Provide the
calculated checksum.
    .eTag(partResponse.eTag())
    .build();
completedParts.add(part);

bb.clear();
position += read;
partNumber++;
    }
} catch (IOException e) {
    System.err.println(e.getMessage());
}

// Complete the multipart upload.
s3Client.completeMultipartUpload(b -> b
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 항목을 참조하세요.
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [UploadPart](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon S3 객체 무결성 기능 작업

다음 코드 예시에서는 S3 객체 무결성 기능을 사용하는 방법을 보여줍니다.

C++

SDK for C++

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 객체 무결성 기능을 시연하는 대화형 시나리오를 실행합니다.

```
#!/ Routine which runs the S3 object integrity workflow.
/*!
    \param clientConfig: Aws client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::s3ObjectIntegrityWorkflow(
    const Aws::S3::S3ClientConfiguration &clientConfiguration) {

    /*
     * Create a large file to be used for multipart uploads.
     */
    if (!createLargeFileIfNotExists()) {
        std::cerr << "Workflow exiting because large file creation failed." <<
std::endl;
        return false;
    }

    Aws::String bucketName = TEST_BUCKET_PREFIX;
    bucketName += Aws::Utils::UUID::RandomUUID();
    bucketName = Aws::Utils::StringUtils::ToLower(bucketName.c_str());

    bucketName.resize(std::min(bucketName.size(), MAX_BUCKET_NAME_LENGTH));

    introductoryExplanations(bucketName);

    if (!AwsDoc::S3::createBucket(bucketName, clientConfiguration)) {
```

```

        std::cerr << "Workflow exiting because bucket creation failed." <<
std::endl;
        return false;
    }

    Aws::S3::S3ClientConfiguration s3ClientConfiguration(clientConfiguration);
    std::shared_ptr<Aws::S3::S3Client> client =
    Aws::MakeShared<Aws::S3::S3Client>("S3Client", s3ClientConfiguration);

    printAsterisksLine();
    std::cout << "Choose from one of the following checksum algorithms."
        << std::endl;

    for (HASH_METHOD hashMethod = DEFAULT; hashMethod <= SHA256; ++hashMethod) {
        std::cout << "  " << hashMethod << " - " <<
stringForHashMethod(hashMethod)
        << std::endl;
    }

    HASH_METHOD chosenHashMethod = askQuestionForIntRange("Enter an index: ",
    DEFAULT,
        SHA256);

    gUseCalculatedChecksum = !askYesNoQuestion(
        "Let the SDK calculate the checksum for you? (y/n) ");

    printAsterisksLine();

    std::cout << "The workflow will now upload a file using PutObject."
        << std::endl;
    std::cout << "Object integrity will be verified using the "
        << stringForHashMethod(chosenHashMethod) << " algorithm."
        << std::endl;
    if (gUseCalculatedChecksum) {
        std::cout
            << "A checksum computed by this workflow will be used for object
integrity verification,"
            << std::endl;
        std::cout << "except for the TransferManager upload." << std::endl;
    } else {
        std::cout
            << "A checksum computed by the SDK will be used for object
integrity verification."

```

```

        << std::endl;
    }

    pressEnterToContinue();
    printAsterisksLine();

    std::shared_ptr<Aws::IOStream> inputData =
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                       TEST_FILE,
                                       std::ios_base::in |
                                       std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    Hasher hasher;
    HASH_METHOD putObjectHashMethod = chosenHashMethod;
    if (putObjectHashMethod == DEFAULT) {
        putObjectHashMethod = MD5; // MD5 is the default hash method for
        PutObject.

        std::cout << "The default checksum algorithm for PutObject is "
                   << stringForHashMethod(putObjectHashMethod)
                   << std::endl;
    }

    // Demonstrate in code how the hash is computed.
    if (!hasher.calculateObjectHash(*inputData, putObjectHashMethod)) {
        std::cerr << "Error calculating hash for file " << TEST_FILE <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }
    Aws::String key = stringForHashMethod(putObjectHashMethod);
    key += "_";
    key += TEST_FILE_KEY;
    Aws::String localHash = hasher.getBase64HashString();

    // Upload the object with PutObject
    if (!putObjectWithHash(bucketName, key, localHash, putObjectHashMethod,
                           inputData, chosenHashMethod == DEFAULT,

```

```

        *client)) {
    std::cerr << "Error putting file " << TEST_FILE << " to bucket "
        << bucketName << " with key " << key << std::endl;
    cleanUp(bucketName, clientConfiguration);
    return false;
}

Aws::String retrievedHash;
if (!retrieveObjectHash(bucketName, key,
    putObjectHashMethod, retrievedHash,
    nullptr, *client)) {
    std::cerr << "Error getting file " << TEST_FILE << " from bucket "
        << bucketName << " with key " << key << std::endl;
    cleanUp(bucketName, clientConfiguration);
    return false;
}

explainPutObjectResults();
verifyHashingResults(retrievedHash, hasher,
    "PutObject upload", putObjectHashMethod);

printAsterisksLine();
pressEnterToContinue();

key = "tr_";
key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;

introductoryTransferManagerUploadExplanations(key);

HASH_METHOD transferManagerHashMethod = chosenHashMethod;
if (transferManagerHashMethod == DEFAULT) {
    transferManagerHashMethod = CRC32; // The default hash method for the
TransferManager is CRC32.

    std::cout << "The default checksum algorithm for TransferManager is "
        << stringForHashMethod(transferManagerHashMethod)
        << std::endl;
}

// Upload the large file using the transfer manager.
if (!doTransferManagerUpload(bucketName, key, transferManagerHashMethod,
    chosenHashMethod == DEFAULT,
        client)) {

```

```

        std::cerr << "Exiting because of an error in doTransferManagerUpload." <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    std::vector<Aws::String> retrievedTransferManagerPartHashes;
    Aws::String retrievedTransferManagerFinalHash;

    // Retrieve all the hashes for the TransferManager upload.
    if (!retrieveObjectHash(bucketName, key,
                           transferManagerHashMethod,
                           retrievedTransferManagerFinalHash,
                           &retrievedTransferManagerPartHashes, *client)) {
        std::cerr << "Exiting because of an error in retrieveObjectHash for
TransferManager." << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    AwsDoc::S3::Hasher locallyCalculatedFinalHash;
    std::vector<Aws::String> locallyCalculatedPartHashes;

    // Calculate the hashes locally to demonstrate how TransferManager hashes are
    computed.
    if (!calculatePartHashesForFile(transferManagerHashMethod,
MULTI_PART_TEST_FILE,
                                   UPLOAD_BUFFER_SIZE,
                                   locallyCalculatedFinalHash,
                                   locallyCalculatedPartHashes)) {
        std::cerr << "Exiting because of an error in calculatePartHashesForFile."
<< std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    verifyHashingResults(retrievedTransferManagerFinalHash,
                        locallyCalculatedFinalHash, "TransferManager upload",
                        transferManagerHashMethod,
                        retrievedTransferManagerPartHashes,
                        locallyCalculatedPartHashes);

    printAsterisksLine();

```



```

    key = "mp_";
    key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;

    multiPartUploadExplanations(key, chosenHashMethod);

    pressEnterToContinue();

    std::shared_ptr<Aws::IOStream> largeFileInputData =
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                      MULTI_PART_TEST_FILE,
                                      std::ios_base::in |
                                      std::ios_base::binary);

    if (!largeFileInputData->good()) {
        std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    HASH_METHOD multipartUploadHashMethod = chosenHashMethod;
    if (multipartUploadHashMethod == DEFAULT) {
        multipartUploadHashMethod = MD5; // The default hash method for
multipart uploads is MD5.

        std::cout << "The default checksum algorithm for multipart upload is "
                    << stringForHashMethod(putObjectHashMethod)
                    << std::endl;
    }

    AwsDoc::S3::Hasher hashData;
    std::vector<Aws::String> partHashes;

    if (!doMultipartUpload(bucketName, key,
                           multipartUploadHashMethod,
                           largeFileInputData, chosenHashMethod == DEFAULT,
                           hashData,
                           partHashes,
                           *client)) {
        std::cerr << "Exiting because of an error in doMultipartUpload." <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

```

```

std::cout << "Finished multipart upload of with hash method " <<
    stringForHashMethod(multipartUploadHashMethod) << std::endl;

std::cout << "Now we will retrieve the checksums from the server." <<
std::endl;

retrievedHash.clear();
std::vector<Aws::String> retrievedPartHashes;
if (!retrieveObjectHash(bucketName, key,
                        multipartUploadHashMethod,
                        retrievedHash, &retrievedPartHashes, *client)) {
    std::cerr << "Exiting because of an error in retrieveObjectHash for
multipart." << std::endl;
    cleanUp(bucketName, clientConfiguration);
    return false;
}

verifyHashingResults(retrievedHash, hashData, "MultiPart upload",
                    multipartUploadHashMethod,
                    retrievedPartHashes, partHashes);

printAsterisksLine();

if (askYesNoQuestion("Would you like to delete the resources created in this
workflow? (y/n)")) {
    return cleanUp(bucketName, clientConfiguration);
} else {
    std::cout << "The bucket " << bucketName << " was not deleted." <<
std::endl;
    return true;
}
}

//! Routine which uploads an object to an S3 bucket with different object
integrity hashing methods.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param hashData: The hash value that will be associated with the uploaded
object.
    \param hashMethod: The hashing algorithm to use when calculating the hash
value.
    \param body: The data content of the object being uploaded.

```

```

\param useDefaultHashMethod: A flag indicating whether to use the default hash
method or the one specified in the hashMethod parameter.
\param client: The S3 client instance used to perform the upload operation.
\return bool: Function succeeded.
*/
bool AwsDoc::S3::putObjectWithHash(const Aws::String &bucket, const Aws::String
&key,
                                const Aws::String &hashData,
                                AwsDoc::S3::HASH_METHOD hashMethod,
                                const std::shared_ptr<Aws::IOStream> &body,
                                bool useDefaultHashMethod,
                                const Aws::S3::S3Client &client) {
    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    if (!useDefaultHashMethod) {
        if (hashMethod != MD5) {
            request.SetChecksumAlgorithm(getChecksumAlgorithmForHashMethod(hashMethod));
        }
    }

    if (gUseCalculatedChecksum) {
        switch (hashMethod) {
            case AwsDoc::S3::MD5:
                request.SetContentMD5(hashData);
                break;
            case AwsDoc::S3::SHA1:
                request.SetChecksumSHA1(hashData);
                break;
            case AwsDoc::S3::SHA256:
                request.SetChecksumSHA256(hashData);
                break;
            case AwsDoc::S3::CRC32:
                request.SetChecksumCRC32(hashData);
                break;
            case AwsDoc::S3::CRC32C:
                request.SetChecksumCRC32C(hashData);
                break;
            default:
                std::cerr << "Unknown hash method." << std::endl;
                return false;
        }
    }
}

```

```

    request.SetBody(body);
    Aws::S3::Model::PutObjectOutcome outcome = client.PutObject(request);
    body->seekg(0, body->beg);
    if (outcome.IsSuccess()) {
        std::cout << "Object successfully uploaded." << std::endl;
    } else {
        std::cerr << "Error uploading object." <<
            outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}

// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object is stored.
    \param key: The unique identifier (key) of the object within the S3 bucket.
    \param hashMethod: The hashing algorithm used to calculate the hash value of
the object.
    \param[out] hashData: The retrieved hash.
    \param[out] partHashes: The part hashes if available.
    \param client: The S3 client instance used to retrieve the object.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
&key,

                                AwsDoc::S3::HASH_METHOD hashMethod,
                                Aws::String &hashData,
                                std::vector<Aws::String> *partHashes,
                                const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (hashMethod == MD5) {
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
client.GetObjectAttributes(
    request);
        if (outcome.IsSuccess()) {

```

```

        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        hashData = result.GetETag();
    } else {
        std::cerr << "Error retrieving object etag attributes." <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }
} else { // hashMethod != MD5
    Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
    attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
    request.SetObjectAttributes(attributes);

    Aws::S3::Model::GetObjectAttributesOutcome outcome =
client.GetObjectAttributes(
    request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        switch (hashMethod) {
            case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                break; // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
            case AwsDoc::S3::MD5:
                break; // MD5 is not supported.
#pragma clang diagnostic pop
            case AwsDoc::S3::SHA1:
                hashData = result.GetChecksum().GetChecksumSHA1();
                break;
            case AwsDoc::S3::SHA256:
                hashData = result.GetChecksum().GetChecksumSHA256();
                break;
            case AwsDoc::S3::CRC32:
                hashData = result.GetChecksum().GetChecksumCRC32();
                break;
            case AwsDoc::S3::CRC32C:
                hashData = result.GetChecksum().GetChecksumCRC32C();
                break;
            default:
                std::cerr << "Unknown hash method." << std::endl;
                return false;
        }
    } else {

```

```

        std::cerr << "Error retrieving object checksum attributes." <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    if (nullptr != partHashes) {
        attributes.clear();
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
        request.SetObjectAttributes(attributes);
        outcome = client.GetObjectAttributes(request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
            const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
result.GetObjectParts().GetParts();
            for (const Aws::S3::Model::ObjectPart &part: parts) {
                switch (hashMethod) {
                    case AwsDoc::S3::DEFAULT: // Default is not supported.
NOLINT(*-branch-clone)
                        break;
                    case AwsDoc::S3::MD5: // MD5 is not supported.
                        break;
                    case AwsDoc::S3::SHA1:
                        partHashes->push_back(part.GetChecksumSHA1());
                        break;
                    case AwsDoc::S3::SHA256:
                        partHashes->push_back(part.GetChecksumSHA256());
                        break;
                    case AwsDoc::S3::CRC32:
                        partHashes->push_back(part.GetChecksumCRC32());
                        break;
                    case AwsDoc::S3::CRC32C:
                        partHashes->push_back(part.GetChecksumCRC32C());
                        break;
                    default:
                        std::cerr << "Unknown hash method." << std::endl;
                        return false;
                }
            }
        } else {
            std::cerr << "Error retrieving object attributes for object
parts." <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }
}

```

```

        }
    }
}

return true;
}

//! Verifies the hashing results between the retrieved and local hashes.
/*!
\param retrievedHash The hash value retrieved from the remote source.
\param localHash The hash value calculated locally.
\param uploadtype The type of upload (e.g., "multipart", "single-part").
\param hashMethod The hashing method used (e.g., MD5, SHA-256).
\param retrievedPartHashes (Optional) The list of hashes for the individual
parts retrieved from the remote source.
\param localPartHashes (Optional) The list of hashes for the individual parts
calculated locally.
*/
void AwsDoc::S3::verifyHashingResults(const Aws::String &retrievedHash,
                                     const Hasher &localHash,
                                     const Aws::String &uploadtype,
                                     HASH_METHOD hashMethod,
                                     const std::vector<Aws::String>
&retrievedPartHashes,
                                     const std::vector<Aws::String>
&localPartHashes) {
    std::cout << "For " << uploadtype << " retrieved hash is " << retrievedHash
<< std::endl;
    if (!retrievedPartHashes.empty()) {
        std::cout << retrievedPartHashes.size() << " part hash(es) were also
retrieved."
<< std::endl;
        for (auto &retrievedPartHash: retrievedPartHashes) {
            std::cout << " Part hash " << retrievedPartHash << std::endl;
        }
    }
    Aws::String hashString;
    if (hashMethod == MD5) {
        hashString = localHash.getHexHashString();
        if (!localPartHashes.empty()) {
            hashString += "-" + std::to_string(localPartHashes.size());
        }
    } else {
        hashString = localHash.getBase64HashString();
    }
}

```

```

    }

    bool allMatch = true;
    if (hashString != retrievedHash) {
        std::cerr << "For " << uploadtype << ", the main hashes do not match" <<
std::endl;
        std::cerr << "Local hash- '" << hashString << "'" << std::endl;
        std::cerr << "Remote hash - '" << retrievedHash << "'" << std::endl;
        allMatch = false;
    }

    if (hashMethod != MD5) {
        if (localPartHashes.size() != retrievedPartHashes.size()) {
            std::cerr << "For " << uploadtype << ", the number of part hashes do
not match" << std::endl;
            std::cerr << "Local number of hashes- '" << localPartHashes.size() <<
""
                        << std::endl;
            std::cerr << "Remote number of hashes - '"
                        << retrievedPartHashes.size()
                        << "'" << std::endl;
        }

        for (int i = 0; i < localPartHashes.size(); ++i) {
            if (localPartHashes[i] != retrievedPartHashes[i]) {
                std::cerr << "For " << uploadtype << ", the part hashes do not
match for part " << i + 1
                        << "." << std::endl;
                std::cerr << "Local hash- '" << localPartHashes[i] << "'"
                        << std::endl;
                std::cerr << "Remote hash - '" << retrievedPartHashes[i] << "'"
                        << std::endl;
                allMatch = false;
            }
        }
    }

    if (allMatch) {
        std::cout << "For " << uploadtype << ", locally and remotely calculated
hashes all match!" << std::endl;
    }
}

```



```

static void transferManagerErrorCallback(const Aws::Transfer::TransferManager *,
                                         const std::shared_ptr<const
                                         Aws::Transfer::TransferHandle> &,
                                         const
                                         Aws::Client::AWSError<Aws::S3::S3Errors> &err) {
    std::cerr << "Error during transfer: '" << err.GetMessage() << "'" <<
    std::endl;
}

static void transferManagerStatusCallback(const Aws::Transfer::TransferManager *,
                                         const std::shared_ptr<const
                                         Aws::Transfer::TransferHandle> &handle) {
    if (handle->GetStatus() == Aws::Transfer::TransferStatus::IN_PROGRESS) {
        std::cout << "Bytes transferred: " << handle->GetBytesTransferred() <<
        std::endl;
    }
}

//! Routine which uploads an object to an S3 bucket using the AWS C++ SDK's
    Transfer Manager.
    /*!
        \param bucket: The name of the S3 bucket where the object will be uploaded.
        \param key: The unique identifier (key) for the object within the S3 bucket.
        \param hashMethod: The hashing algorithm to use when calculating the hash
        value.
        \param useDefaultHashMethod: A flag indicating whether to use the default hash
        method or the one specified in the hashMethod parameter.
        \param client: The S3 client instance used to perform the upload operation.
        \return bool: Function succeeded.
    */
    bool
    AwsDoc::S3::doTransferManagerUpload(const Aws::String &bucket, const Aws::String
    &key,
                                         AwsDoc::S3::HASH_METHOD hashMethod,
                                         bool useDefaultHashMethod,
                                         const std::shared_ptr<Aws::S3::S3Client>
    &client) {
        std::shared_ptr<Aws::Utils::Threading::PooledThreadExecutor> executor =
        Aws::MakeShared<Aws::Utils::Threading::PooledThreadExecutor>(
            "executor", 25);
        Aws::Transfer::TransferManagerConfiguration transfer_config(executor.get());
        transfer_config.s3Client = client;
        transfer_config.bufferSize = UPLOAD_BUFFER_SIZE;
        if (!useDefaultHashMethod) {

```

```

        if (hashMethod == MD5) {
            transfer_config.computeContentMD5 = true;
        } else {
            transfer_config.checksumAlgorithm =
getChecksumAlgorithmForHashMethod(
                hashMethod);
        }
    }
    transfer_config.errorCallback = transferManagerErrorCallback;
    transfer_config.transferStatusUpdatedCallback =
transferManagerStatusCallback;

    std::shared_ptr<Aws::Transfer::TransferManager> transfer_manager =
Aws::Transfer::TransferManager::Create(
        transfer_config);

    std::cout << "Uploading the file..." << std::endl;
    std::shared_ptr<Aws::Transfer::TransferHandle> uploadHandle =
transfer_manager->UploadFile(MULTI_PART_TEST_FILE,

        bucket, key,

        "text/plain",

        Aws::Map<Aws::String, Aws::String>());
    uploadHandle->WaitUntilFinished();
    bool success =
        uploadHandle->GetStatus() ==
Aws::Transfer::TransferStatus::COMPLETED;
    if (!success) {
        Aws::Client::AWSError<Aws::S3::S3Errors> err = uploadHandle-
>GetLastError();
        std::cerr << "File upload failed:  " << err.GetMessage() << std::endl;
    }

    return success;
}

//! Routine which calculates the hash values for each part of a file being
uploaded to an S3 bucket.
/*!
    \param hashMethod: The hashing algorithm to use when calculating the hash
values.

```

```

    \param fileName: The path to the file for which the part hashes will be
    calculated.
    \param bufferSize: The size of the buffer to use when reading the file.
    \param[out] hashDataResult: The Hasher object that will store the concatenated
    hash value.
    \param[out] partHashes: The vector that will store the calculated hash values
    for each part of the file.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::calculatePartHashesForFile(AwsDoc::S3::HASH_METHOD hashMethod,
                                             const Aws::String &fileName,
                                             size_t bufferSize,
                                             AwsDoc::S3::Hasher &hashDataResult,
                                             std::vector<Aws::String> &partHashes)
{
    std::ifstream fileStream(fileName.c_str(), std::ifstream::binary);
    fileStream.seekg(0, std::ifstream::end);
    size_t objectSize = fileStream.tellg();
    fileStream.seekg(0, std::ifstream::beg);
    std::vector<unsigned char> totalHashBuffer;
    size_t uploadedBytes = 0;

    while (uploadedBytes < objectSize) {
        std::vector<unsigned char> buffer(bufferSize);
        std::streamsize bytesToRead =
static_cast<std::streamsize>(std::min(buffer.size(), objectSize -
uploadedBytes));
        fileStream.read((char *) buffer.data(), bytesToRead);
        Aws::Utils::Stream::PreallocatedStreamBuf
preallocatedStreamBuf(buffer.data(),

bytesToRead);
        std::shared_ptr<Aws::IOStream> body =
            Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
                                           &preallocatedStreamBuf);

        Hasher hasher;
        if (!hasher.calculateObjectHash(*body, hashMethod)) {
            std::cerr << "Error calculating hash." << std::endl;
            return false;
        }
        Aws::String base64HashString = hasher.getBase64HashString();
        partHashes.push_back(base64HashString);
    }
}

```

```

        Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();

        totalHashBuffer.insert(totalHashBuffer.end(),
            hashBuffer.GetUnderlyingData(),
            hashBuffer.GetUnderlyingData() +
            hashBuffer.GetLength());

        uploadedBytes += bytesToRead;
    }

    return hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod);
}

//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String
    &key,

                                Aws::S3::Model::ChecksumAlgorithm
checksumAlgorithm,

                                const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }

    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
        client.CreateMultipartUpload(request);

    Aws::String uploadID;
    if (outcome.IsSuccess()) {
        uploadID = outcome.GetResult().GetUploadId();
    } else {
        std::cerr << "Error creating multipart upload: " <<
        outcome.GetError().GetMessage() << std::endl;
    }
}

```

```

        return uploadID;
    }

    //! Upload a part to an S3 bucket.
    /*!
        \param bucket: The name of the S3 bucket where the object will be uploaded.
        \param key: The unique identifier (key) for the object within the S3 bucket.
        \param uploadID: An upload ID string.
        \param partNumber:
        \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
        \param calculatedHash: A data integrity hash to set, depending on the
checksum algorithm,
                                ignored when it is an empty string.
        \param body: An shared_ptr IOStream of the data to be uploaded.
        \param client: The S3 client instance used to perform the upload operation.
        \return UploadPartOutcome: The outcome.
    */

    Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String
        &bucket,
                                                                const Aws::String &key,
                                                                const Aws::String
        &uploadID,
                                                                int partNumber,

        Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
                                                                const Aws::String
        &calculatedHash,
                                                                const
        std::shared_ptr<Aws::IOStream> &body,
                                                                const Aws::S3::S3Client
        &client) {
        Aws::S3::Model::UploadPartRequest request;
        request.SetBucket(bucket);
        request.SetKey(key);
        request.SetUploadId(uploadID);
        request.SetPartNumber(partNumber);
        if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
            request.SetChecksumAlgorithm(checksumAlgorithm);
        }
        request.SetBody(body);

        if (!calculatedHash.empty()) {

```

```

        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
                request.SetContentMD5(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32:
                request.SetChecksumCRC32(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
                request.SetChecksumCRC32C(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA1:
                request.SetChecksumSHA1(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA256:
                request.SetChecksumSHA256(calculatedHash);
                break;
        }
    }

    return client.UploadPart(request);
}

//! Abort a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                       const Aws::String &key,
                                       const Aws::String &uploadID,
                                       const Aws::S3::S3Client &client) {
    Aws::S3::Model::AbortMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);

    Aws::S3::Model::AbortMultipartUploadOutcome outcome =
        client.AbortMultipartUpload(request);

    if (outcome.IsSuccess()) {

```

```

        std::cout << "Multipart upload aborted." << std::endl;
    } else {
        std::cerr << "Error aborting multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Complete a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param parts: A vector of CompleteParts.
    \param client: The S3 client instance used to perform the upload operation.
    \return CompleteMultipartUploadOutcome: The request outcome.
*/
Aws::S3::Model::CompleteMultipartUploadOutcome
AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,

    const Aws::String &key,

    const Aws::String &uploadID,

    const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,

    const Aws::S3::S3Client &client) {
    Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
    completedMultipartUpload.SetParts(parts);

    Aws::S3::Model::CompleteMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetMultipartUpload(completedMultipartUpload);

    Aws::S3::Model::CompleteMultipartUploadOutcome outcome =
        client.CompleteMultipartUpload(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error completing multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }
}

```

```

        return outcome;
    }

    //! Routine which performs a multi-part upload.
    /*!
        \param bucket: The name of the S3 bucket where the object will be uploaded.
        \param key: The unique identifier (key) for the object within the S3 bucket.
        \param hashMethod: The hashing algorithm to use when calculating the hash
        value.
        \param ioStream: An IOStream for the data to be uploaded.
        \param useDefaultHashMethod: A flag indicating whether to use the default
        hash method or the one specified in the hashMethod parameter.
        \param[out] hashDataResult: The Hasher object that will store the
        concatenated hash value.
        \param[out] partHashes: The vector that will store the calculated hash values
        for each part of the file.
        \param client: The S3 client instance used to perform the upload operation.
        \return bool: Function succeeded.
    */
    bool AwsDoc::S3::doMultipartUpload(const Aws::String &bucket,
                                       const Aws::String &key,
                                       AwsDoc::S3::HASH_METHOD hashMethod,
                                       const std::shared_ptr<Aws::IOStream>
                                       &ioStream,
                                       bool useDefaultHashMethod,
                                       AwsDoc::S3::Hasher &hashDataResult,
                                       std::vector<Aws::String> &partHashes,
                                       const Aws::S3::S3Client &client) {

        // Get object size.
        ioStream->seekg(0, ioStream->end);
        size_t objectSize = ioStream->tellg();
        ioStream->seekg(0, ioStream->beg);

        Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm =
        Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
        if (!useDefaultHashMethod) {
            if (hashMethod != MD5) {
                checksumAlgorithm = getChecksumAlgorithmForHashMethod(hashMethod);
            }
        }
        Aws::String uploadID = createMultipartUpload(bucket, key, checksumAlgorithm,
        client);
        if (uploadID.empty()) {
            return false;
        }
    }

```



```

    }

    std::vector<unsigned char> totalHashBuffer;
    bool uploadSucceeded = true;
    std::streamsize uploadedBytes = 0;
    int partNumber = 1;
    Aws::Vector<Aws::S3::Model::CompletedPart> parts;
    while (uploadedBytes < objectSize) {
        std::cout << "Uploading part " << partNumber << "." << std::endl;

        std::vector<unsigned char> buffer(UPLOAD_BUFFER_SIZE);
        std::streamsize bytesToRead =
static_cast<std::streamsize>(std::min(buffer.size(),
objectSize - uploadedBytes));
        ioStream->read((char *) buffer.data(), bytesToRead);
        Aws::Utils::Stream::PreallocatedStreamBuf
preallocatedStreamBuf(buffer.data(),
bytesToRead);
        std::shared_ptr<Aws::IOStream> body =
            Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
                &preallocatedStreamBuf);

        Hasher hasher;
        if (!hasher.calculateObjectHash(*body, hashMethod)) {
            std::cerr << "Error calculating hash." << std::endl;
            uploadSucceeded = false;
            break;
        }

        Aws::String base64HashString = hasher.getBase64HashString();
        partHashes.push_back(base64HashString);

        Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();

        totalHashBuffer.insert(totalHashBuffer.end(),
hashBuffer.GetUnderlyingData(),
                                hashBuffer.GetUnderlyingData() +
hashBuffer.GetLength());

        Aws::String calculatedHash;
        if (gUseCalculatedChecksum) {
            calculatedHash = base64HashString;

```

```

    }
    Aws::S3::Model::UploadPartOutcome uploadPartOutcome = uploadPart(bucket,
key, uploadID, partNumber,

checksumAlgorithm, base64HashString, body,

                                                                    client);

    if (uploadPartOutcome.IsSuccess()) {
        const Aws::S3::Model::UploadPartResult &uploadPartResult =
uploadPartOutcome.GetResult();
        Aws::S3::Model::CompletedPart completedPart;
        completedPart.SetETag(uploadPartResult.GetETag());
        completedPart.SetPartNumber(partNumber);
        switch (hashMethod) {
            case AwsDoc::S3::MD5:
                break; // Do nothing.
            case AwsDoc::S3::SHA1:

completedPart.SetChecksumSHA1(uploadPartResult.GetChecksumSHA1());
                break;
            case AwsDoc::S3::SHA256:

completedPart.SetChecksumSHA256(uploadPartResult.GetChecksumSHA256());
                break;
            case AwsDoc::S3::CRC32:

completedPart.SetChecksumCRC32(uploadPartResult.GetChecksumCRC32());
                break;
            case AwsDoc::S3::CRC32C:

completedPart.SetChecksumCRC32C(uploadPartResult.GetChecksumCRC32C());
                break;
            default:
                std::cerr << "Unhandled hash method for completedPart." <<
std::endl;
                break;
        }

        parts.push_back(completedPart);
    } else {
        std::cerr << "Error uploading part. " <<
            uploadPartOutcome.GetError().GetMessage() << std::endl;
        uploadSucceeded = false;
        break;
    }
}

```

```

        uploadedBytes += bytesToRead;
        partNumber++;
    }

    if (!uploadSucceeded) {
        abortMultipartUpload(bucket, key, uploadID, client);
        return false;
    } else {

        Aws::S3::Model::CompleteMultipartUploadOutcome
completeMultipartUploadOutcome = completeMultipartUpload(bucket,

                                key,

                                uploadID,

                                parts,

                                client);

        if (completeMultipartUploadOutcome.IsSuccess()) {
            std::cout << "Multipart upload completed." << std::endl;
            if (!hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod))
        {
                std::cerr << "Error calculating hash." << std::endl;
                return false;
            }
        } else {
            std::cerr << "Error completing multipart upload." <<
                completeMultipartUploadOutcome.GetError().GetMessage()
                << std::endl;
        }

        return completeMultipartUploadOutcome.IsSuccess();
    }
}

//! Routine which retrieves the string for a HASH_METHOD constant.
/*!
    \param: hashMethod: A HASH_METHOD constant.
    \return: String: A string description of the hash method.
*/
Aws::String AwsDoc::S3::stringForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod) {

```

```

        switch (hashMethod) {
            case AwsDoc::S3::DEFAULT:
                return "Default";
            case AwsDoc::S3::MD5:
                return "MD5";
            case AwsDoc::S3::SHA1:
                return "SHA1";
            case AwsDoc::S3::SHA256:
                return "SHA256";
            case AwsDoc::S3::CRC32:
                return "CRC32";
            case AwsDoc::S3::CRC32C:
                return "CRC32C";
            default:
                return "Unknown";
        }
    }

    /** Routine that returns the ChecksumAlgorithm for a HASH_METHOD constant.
    */
    \param: hashMethod: A HASH_METHOD constant.
    \return: ChecksumAlgorithm: The ChecksumAlgorithm enum.
    */
    Aws::S3::Model::ChecksumAlgorithm
    AwsDoc::S3::getChecksumAlgorithmForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod)
    {
        Aws::S3::Model::ChecksumAlgorithm result =
        Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
        switch (hashMethod) {
            case AwsDoc::S3::DEFAULT:
                std::cerr << "getChecksumAlgorithmForHashMethod- DEFAULT is not
valid." << std::endl;
                break; // Default is not supported.
            case AwsDoc::S3::MD5:
                break; // Ignore MD5.
            case AwsDoc::S3::SHA1:
                result = Aws::S3::Model::ChecksumAlgorithm::SHA1;
                break;
            case AwsDoc::S3::SHA256:
                result = Aws::S3::Model::ChecksumAlgorithm::SHA256;
                break;
            case AwsDoc::S3::CRC32:
                result = Aws::S3::Model::ChecksumAlgorithm::CRC32;
                break;

```

```

        case AwsDoc::S3::CRC32C:
            result = Aws::S3::Model::ChecksumAlgorithm::CRC32C;
            break;
        default:
            std::cerr << "Unknown hash method." << std::endl;
            break;
    }

    return result;
}

//! Routine which cleans up after the example is complete.
/*!
    \param bucket: The name of the S3 bucket where the object was uploaded.
    \param clientConfiguration: The client configuration for the S3 client.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::cleanUp(const Aws::String &bucketName,
                        const Aws::S3::S3ClientConfiguration
                        &clientConfiguration) {

    Aws::Vector<Aws::String> keysResult;
    bool result = true;
    if (AwsDoc::S3::listObjects(bucketName, keysResult, clientConfiguration)) {
        if (!keysResult.empty()) {
            result = AwsDoc::S3::deleteObjects(keysResult, bucketName,
                                                clientConfiguration);
        }
    } else {
        result = false;
    }

    return result && AwsDoc::S3::deleteBucket(bucketName, clientConfiguration);
}

//! Console interaction introducing the workflow.
/*!
    \param bucketName: The name of the S3 bucket to use.
*/
void AwsDoc::S3::introductoryExplanations(const Aws::String &bucketName) {

    std::cout

```

```
<< "Welcome to the Amazon Simple Storage Service (Amazon S3) object
integrity workflow."
    << std::endl;
    printAsterisksLine();
    std::cout
        << "This workflow demonstrates how Amazon S3 uses checksum values to
verify the integrity of data\n";
    std::cout << "uploaded to Amazon S3 buckets" << std::endl;
    std::cout
        << "The AWS SDK for C++ automatically handles checksums.\n";
    std::cout
        << "By default it calculates a checksum that is uploaded with an
object.\n"
        << "The default checksum algorithm for PutObject and MultiPart upload
is an MD5 hash.\n"
        << "The default checksum algorithm for TransferManager uploads is a
CRC32 checksum."
        << std::endl;
    std::cout
        << "You can override the default behavior, requiring one of the
following checksums,\n";
    std::cout << "MD5, CRC32, CRC32C, SHA-1 or SHA-256." << std::endl;
    std::cout << "You can also set the checksum hash value, instead of letting
the SDK calculate the value."
        << std::endl;
    std::cout
        << "For more information, see https://docs.aws.amazon.com/AmazonS3/
latest/userguide/checking-object-integrity.html."
        << std::endl;

    std::cout
        << "This workflow will locally compute checksums for files uploaded
to an Amazon S3 bucket,\n";
    std::cout << "even when the SDK also computes the checksum." << std::endl;
    std::cout
        << "This is done to provide demonstration code for how the checksums
are calculated."
        << std::endl;
    std::cout << "A bucket named '" << bucketName << "' will be created for the
object uploads."
        << std::endl;
}

//! Console interaction which explains the PutObject results.
```

```

    /*!
    */
    void AwsDoc::S3::explainPutObjectResults() {

        std::cout << "The upload was successful.\n";
        std::cout << "If the checksums had not matched, the upload would have
        failed."
                << std::endl;
        std::cout
                << "The checksums calculated by the server have been retrieved using
        the GetObjectAttributes."
                << std::endl;
        std::cout
                << "The locally calculated checksums have been verified against the
        retrieved checksums."
                << std::endl;
    }

    /*! Console interaction explaining transfer manager uploads.
    */
    \param objectKey: The key for the object being uploaded.
    */
    void AwsDoc::S3::introductoryTransferManagerUploadExplanations(
        const Aws::String &objectKey) {
        std::cout
            << "Now the workflow will demonstrate object integrity for
        TransferManager multi-part uploads."
            << std::endl;
        std::cout
            << "The AWS C++ SDK has a TransferManager class which simplifies
        multipart uploads."
            << std::endl;
        std::cout
            << "The following code lets the TransferManager handle much of the
        checksum configuration."
            << std::endl;

        std::cout << "An object with the key '" << objectKey
            << " will be uploaded by the TransferManager using a "
            << BUFFER_SIZE_IN_MEGABYTES << " MB buffer." << std::endl;
        if (gUseCalculatedChecksum) {
            std::cout << "For TransferManager uploads, this demo always lets the SDK
        calculate the hash value."
                << std::endl;
        }
    }

```

```

    }

    pressEnterToContinue();
    printAsterisksLine();
}

//! Console interaction explaining multi-part uploads.
/*!
    \param objectKey: The key for the object being uploaded.
    \param chosenHashMethod: The hash method selected by the user.
*/
void AwsDoc::S3::multiPartUploadExplanations(const Aws::String &objectKey,
                                              HASH_METHOD chosenHashMethod) {

    std::cout
        << "Now we will provide an in-depth demonstration of multi-part
uploading by calling the multi-part upload APIs directly."
        << std::endl;
    std::cout << "These are the same APIs used by the TransferManager when
uploading large files."
        << std::endl;
    std::cout
        << "In the following code, the checksums are also calculated locally
and then compared."
        << std::endl;
    std::cout
        << "For multi-part uploads, a checksum is uploaded with each part.
The final checksum is a concatenation of"
        << std::endl;
    std::cout << "the checksums for each part." << std::endl;
    std::cout
        << "This is explained in the user guide, https://docs.aws.amazon.com/AmazonS3/latest/userguide/checking-object-integrity.html,"
        << " in the section \"Using part-level checksums for multipart
uploads\"." << std::endl;

    std::cout << "Starting multipart upload of with hash method " <<
        stringForHashMethod(chosenHashMethod) << " uploading to with object
key\n"
        << "'" << objectKey << "'," << std::endl;
}

//! Create a large file for doing multi-part uploads.
/*!

```



```

*/
bool AwsDoc::S3::createLargeFileIfNotExists() {
    // Generate a large file by writing this source file multiple times to a new
    file.
    if (std::filesystem::exists(MULTI_PART_TEST_FILE)) {
        return true;
    }

    std::ofstream newFile(MULTI_PART_TEST_FILE, std::ios::out
                           | std::ios::binary);

    if (!newFile) {
        std::cerr << "createLargeFileIfNotExists- Error creating file " <<
MULTI_PART_TEST_FILE <<
        std::endl;
        return false;
    }

    std::ifstream input(TEST_FILE, std::ios::in
                        | std::ios::binary);

    if (!input) {
        std::cerr << "Error opening file " << TEST_FILE <<
        std::endl;
        return false;
    }
    std::stringstream buffer;
    buffer << input.rdbuf();

    input.close();

    while (newFile.tellp() < LARGE_FILE_SIZE && !newFile.bad()) {
        buffer.seekg(std::stringstream::beg);
        newFile << buffer.rdbuf();
    }

    newFile.close();

    return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 다음 주제를 참조하세요.

- [AbortMultipartUpload](#)
- [CompleteMultipartUpload](#)
- [CreateMultipartUpload](#)
- [DeleteObject](#)
- [GetObjectAttributes](#)
- [PutObject](#)
- [UploadPart](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 버전이 지정된 Amazon S3 객체 작업

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 버전이 지정된 S3 버킷을 생성합니다.
- 객체의 모든 버전을 가져옵니다.
- 객체를 이전 버전으로 롤백합니다.
- 버전이 지정된 객체를 삭제하고 복원합니다.
- 객체의 모든 버전을 영구 삭제합니다.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

S3 작업을 래핑하는 함수를 만듭니다.

```
def create_versioned_bucket(bucket_name, prefix):  
    """
```

Creates an Amazon S3 bucket, enables it for versioning, and configures a lifecycle

that expires noncurrent object versions after 7 days.

Adding a lifecycle configuration to a versioned bucket is a best practice. It helps prevent objects in the bucket from accumulating a large number of noncurrent versions, which can slow down request performance.

Usage is shown in the `usage_demo_single_object` function at the end of this module.

```
:param bucket_name: The name of the bucket to create.
:param prefix: Identifies which objects are automatically expired under the
               configured lifecycle rules.
:return: The newly created bucket.
"""
try:
    bucket = s3.create_bucket(
        Bucket=bucket_name,
        CreateBucketConfiguration={
            "LocationConstraint": s3.meta.client.meta.region_name
        },
    )
    logger.info("Created bucket %s.", bucket.name)
except ClientError as error:
    if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
        logger.warning("Bucket %s already exists! Using it.", bucket_name)
        bucket = s3.Bucket(bucket_name)
    else:
        logger.exception("Couldn't create bucket %s.", bucket_name)
        raise

try:
    bucket.Versioning().enable()
    logger.info("Enabled versioning on bucket %s.", bucket.name)
except ClientError:
    logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
    raise

try:
    expiration = 7
    bucket.LifecycleConfiguration().put(
        LifecycleConfiguration={
            "Rules": [
```

```

        {
            "Status": "Enabled",
            "Prefix": prefix,
            "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
        }
    ]
}
)
logger.info(
    "Configured lifecycle to expire noncurrent versions after %s days "
    "on bucket %s.",
    expiration,
    bucket.name,
)
except ClientError as error:
    logger.warning(
        "Couldn't configure lifecycle on bucket %s because %s. "
        "Continuing anyway.",
        bucket.name,
        error,
    )

return bucket

def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
    )

```

```

        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
            f"{object_key}."
        )

def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
    marker.
    By removing the delete marker, we make the previous version the latest
    version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

```

```

:param bucket: The bucket that contains the object.
:param object_key: The object to revive.
"""

# Get the latest version for the object.
response = s3.meta.client.list_object_versions(
    Bucket=bucket.name, Prefix=object_key, MaxKeys=1
)

if "DeleteMarkers" in response:
    latest_version = response["DeleteMarkers"][0]
    if latest_version["IsLatest"]:
        logger.info(
            "Object %s was indeed deleted on %s. Let's revive it.",
            object_key,
            latest_version["LastModified"],
        )
        obj = bucket.Object(object_key)
        obj.Version(latest_version["VersionId"]).delete()
        logger.info(
            "Revived %s, active version is now %s with body '%s'",
            object_key,
            obj.version_id,
            obj.get()["Body"].read(),
        )
    else:
        logger.warning(
            "Delete marker is not the latest version for %s!", object_key
        )
elif "Versions" in response:
    logger.warning("Got an active version for %s, nothing to do.",
object_key)
else:
    logger.error("Couldn't get any version info for %s.", object_key)

def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that contains the object.

```

```

:param object_key: The object to delete.
"""
try:
    bucket.object_versions.filter(Prefix=object_key).delete()
    logger.info("Permanently deleted all versions of object %s.", object_key)
except ClientError:
    logger.exception("Couldn't delete all versions of %s.", object_key)
    raise

```

버전이 지정된 개체에 시의 스타anzas를 업로드하고 해당 개체에 대해 일련의 작업을 수행합니다.

```

def usage_demo_single_object(obj_prefix="demo-versioning/"):
    """
    Demonstrates usage of versioned object functions. This demo uploads a stanza
    of a poem and performs a series of revisions, deletions, and revivals on it.

    :param obj_prefix: The prefix to assign to objects created by this demo.
    """
    with open("father_william.txt") as file:
        stanzas = file.read().split("\n\n")

    width = get_terminal_size((80, 20))[0]
    print("-" * width)
    print("Welcome to the usage demonstration of Amazon S3 versioning.")
    print(
        "This demonstration uploads a single stanza of a poem to an Amazon "
        "S3 bucket and then applies various revisions to it."
    )
    print("-" * width)
    print("Creating a version-enabled bucket for the demo...")
    bucket = create_versioned_bucket("bucket-" + str(uuid.uuid1()), obj_prefix)

    print("\nThe initial version of our stanza:")
    print(stanzas[0])

    # Add the first stanza and revise it a few times.
    print("\nApplying some revisions to the stanza...")
    obj_stanza_1 = bucket.Object(f"{obj_prefix}stanza-1")
    obj_stanza_1.put(Body=bytes(stanzas[0], "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0].upper(), "utf-8"))

```

```

obj_stanza_1.put(Body=bytes(stanzas[0].lower(), "utf-8"))
obj_stanza_1.put(Body=bytes(stanzas[0][::-1], "utf-8"))
print(
    "The latest version of the stanza is now:",
    obj_stanza_1.get()["Body"].read().decode("utf-8"),
    sep="\n",
)

# Versions are returned in order, most recent first.
obj_stanza_1_versions =
bucket.object_versions.filter(Prefix=obj_stanza_1.key)
print(
    "The version data of the stanza revisions:",
    *[
        f"    {version.version_id}, last modified {version.last_modified}"
        for version in obj_stanza_1_versions
    ],
    sep="\n",
)

# Rollback two versions.
print("\nRolling back two versions...")
rollback_object(bucket, obj_stanza_1.key, list(obj_stanza_1_versions)
[2].version_id)
print(
    "The latest version of the stanza:",
    obj_stanza_1.get()["Body"].read().decode("utf-8"),
    sep="\n",
)

# Delete the stanza
print("\nDeleting the stanza...")
obj_stanza_1.delete()
try:
    obj_stanza_1.get()
except ClientError as error:
    if error.response["Error"]["Code"] == "NoSuchKey":
        print("The stanza is now deleted (as expected).")
    else:
        raise

# Revive the stanza
print("\nRestoring the stanza...")
revive_object(bucket, obj_stanza_1.key)

```



```
print(
    "The stanza is restored! The latest version is again:",
    obj_stanza_1.get()["Body"].read().decode("utf-8"),
    sep="\n",
)

# Permanently delete all versions of the object. This cannot be undone!
print("\nPermanently deleting all versions of the stanza...")
permanently_delete_object(bucket, obj_stanza_1.key)
obj_stanza_1_versions =
bucket.object_versions.filter(Prefix=obj_stanza_1.key)
if len(list(obj_stanza_1_versions)) == 0:
    print("The stanza has been permanently deleted and now has no versions.")
else:
    print("Something went wrong. The stanza still exists!")

print(f"\nRemoving {bucket.name}...")
bucket.delete()
print(f"{bucket.name} deleted.")
print("Demo done!")
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 다음 주제를 참조하세요.
 - [CreateBucket](#)
 - [DeleteObject](#)
 - [ListObjectVersions](#)
 - [PutBucketLifecycleConfiguration](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon S3의 서버리스 예제

다음 코드 예제에서는 Amazon S3를 AWS SDK와 함께 사용하는 방법을 보여줍니다.

예제

- [Amazon S3 트리거를 사용하여 Lambda 함수 호출](#)

Amazon S3 트리거를 사용하여 Lambda 함수 호출

다음 코드 예제에서는 S3 버킷에 객체를 업로드하여 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 해당 함수는 이벤트 파라미터에서 S3 버킷 이름과 객체 키를 검색하고 Amazon S3 API를 호출하여 객체의 콘텐츠 유형을 검색하고 로깅합니다.

.NET

SDK for .NET

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

.NET을 사용하여 Lambda로 S3 이벤트를 사용합니다.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Threading.Tasks;
using Amazon.Lambda.Core;
using Amazon.S3;
using System;
using Amazon.Lambda.S3Events;
using System.Web;

// Assembly attribute to enable the Lambda function's JSON input to be converted
// into a .NET class.
[assembly:
    LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSerializer))]

namespace S3Integration
{
    public class Function
    {
        private static AmazonS3Client _s3Client;
        public Function() : this(null)
        {
        }

        internal Function(AmazonS3Client s3Client)
```

```
{
    _s3Client = s3Client ?? new AmazonS3Client();
}

public async Task<string> Handler(S3Event evt, ILambdaContext context)
{
    try
    {
        if (evt.Records.Count <= 0)
        {
            context.Logger.LogLine("Empty S3 Event received");
            return string.Empty;
        }

        var bucket = evt.Records[0].S3.Bucket.Name;
        var key = HttpUtility.UrlDecode(evt.Records[0].S3.Object.Key);

        context.Logger.LogLine($"Request is for {bucket} and {key}");

        var objectResult = await _s3Client.GetObjectAsync(bucket, key);

        context.Logger.LogLine($"Returning {objectResult.Key}");

        return objectResult.Key;
    }
    catch (Exception e)
    {
        context.Logger.LogLine($"Error processing request - {e.Message}");

        return string.Empty;
    }
}
}
```

Go

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda로 S3 이벤트를 사용합니다.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
    "context"
    "log"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
)

func handler(ctx context.Context, s3Event events.S3Event) error {
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        log.Printf("failed to load default config: %s", err)
        return err
    }
    s3Client := s3.NewFromConfig(sdkConfig)

    for _, record := range s3Event.Records {
        bucket := record.S3.Bucket.Name
        key := record.S3.Object.URLDecodedKey
        headOutput, err := s3Client.HeadObject(ctx, &s3.HeadObjectInput{
            Bucket: &bucket,
            Key:    &key,
        })
        if err != nil {
            log.Printf("error getting head of object %s/%s: %s", bucket, key, err)
        }
    }
}
```

```

    return err
}
log.Printf("successfully retrieved %s/%s of type %s", bucket, key,
*headOutput.ContentType)
}

return nil
}

func main() {
    lambda.Start(handler)
}

```

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda로 S3 이벤트를 사용합니다.

```

// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
    com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotifi

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

```

```

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
            S3EventNotificationRecord record = s3event.getRecords().get(0);
            String srcBucket = record.getS3().getBucket().getName();
            String srcKey = record.getS3().getObject().getUrlDecodedKey();

            S3Client s3Client = S3Client.builder().build();
            HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
srcKey);

            logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
type " + headObject.contentType());

            return "Ok";
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucket)
            .key(key)
            .build();
        return s3Client.headObject(headObjectRequest);
    }
}

```

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

JavaScript를 사용하여 Lambda로 S3 이벤트를 사용합니다.

```
import { S3Client, HeadObjectCommand } from "@aws-sdk/client-s3";

const client = new S3Client();

export const handler = async (event, context) => {

    // Get the object from the event and show its content type
    const bucket = event.Records[0].s3.bucket.name;
    const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, ' '));

    try {
        const { ContentType } = await client.send(new HeadObjectCommand({
            Bucket: bucket,
            Key: key,
        }));

        console.log('CONTENT TYPE:', ContentType);
        return ContentType;

    } catch (err) {
        console.log(err);
        const message = `Error getting object ${key} from bucket ${bucket}. Make sure they exist and your bucket is in the same region as this function.`;
        console.log(message);
        throw new Error(message);
    }
};
```

TypeScript를 사용하여 Lambda로 S3 이벤트를 사용합니다.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { S3Event } from 'aws-lambda';
import { S3Client, HeadObjectCommand } from '@aws-sdk/client-s3';

const s3 = new S3Client({ region: process.env.AWS_REGION });

export const handler = async (event: S3Event): Promise<string | undefined> => {
    // Get the object from the event and show its content type
```

```

const bucket = event.Records[0].s3.bucket.name;
const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, '
'));
const params = {
  Bucket: bucket,
  Key: key,
};
try {
  const { ContentType } = await s3.send(new HeadObjectCommand(params));
  console.log('CONTENT TYPE:', ContentType);
  return ContentType;
} catch (err) {
  console.log(err);
  const message = `Error getting object ${key} from bucket ${bucket}. Make sure
they exist and your bucket is in the same region as this function.`;
  console.log(message);
  throw new Error(message);
}
};

```

PHP

SDK for PHP

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda로 S3 이벤트 사용.

```

<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

```



```
class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' .
                $bucket . '. Make sure they exist and your bucket is in the same region as this
                function.' . "\n";
                throw $e;
            }
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Python을 사용하여 Lambda로 S3 이벤트를 사용합니다.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import json
import urllib.parse
import boto3

print('Loading function')

s3 = boto3.client('s3')

def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
    encoding='utf-8')
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        print("CONTENT TYPE: " + response['ContentType'])
        return response['ContentType']
    except Exception as e:
        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and
        your bucket is in the same region as this function.'.format(key, bucket))
        raise e
```

Ruby

SDK for Ruby

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda로 S3 이벤트 사용.

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
  bucket = event['Records'][0]['s3']['bucket']['name']
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
    Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
    and your bucket is in the same region as this function."
    raise e
  end
end
```

Rust

SDK for Rust

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Rust를 사용하여 Lambda로 S3 이벤트를 사용합니다.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::s3::S3Event;
use aws_sdk_s3::{Client};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

/// Main function
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    // Initialize the AWS SDK for Rust
    let config = aws_config::load_from_env().await;
    let s3_client = Client::new(&config);

    let res = run(service_fn(|request: LambdaEvent<S3Event>| {
        function_handler(&s3_client, request)
    })).await;

    res
}

async fn function_handler(
    s3_client: &Client,
    evt: LambdaEvent<S3Event>
) -> Result<(), Error> {
```

```

    tracing::info!(records = ?evt.payload.records.len(), "Received request from
SQS");

    if evt.payload.records.len() == 0 {
        tracing::info!("Empty S3 event received");
    }

    let bucket = evt.payload.records[0].s3.bucket.name.as_ref().expect("Bucket
name to exist");
    let key = evt.payload.records[0].s3.object.key.as_ref().expect("Object key to
exist");

    tracing::info!("Request is for {} and object {}", bucket, key);

    let s3_get_object_result = s3_client
        .get_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await;

    match s3_get_object_result {
        Ok(_) => tracing::info!("S3 Get Object success, the s3GetObjectResult
contains a 'body' property of type ByteStream"),
        Err(_) => tracing::info!("Failure with S3 Get Object request")
    }

    Ok(())
}

```

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용한 Amazon S3 Control 코드 예제

다음 코드 예제에서는 Amazon S3 Control을 AWS 소프트웨어 개발 키트(SDK)와 함께 사용하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon S3 Control

- [AWS SDK를 사용한 Amazon S3 Control 기본 예제](#)
 - [Amazon S3 Control 시작](#)
 - [AWS SDK를 사용하는 Amazon S3 Control의 기본 기능 알아보기](#)
 - [AWS SDK를 사용한 Amazon S3 Control 작업](#)
 - [AWS SDK 또는 CLI와 함께 CreateJob 사용](#)
 - [AWS SDK와 함께 DeleteJobTagging 사용](#)
 - [AWS SDK 또는 CLI와 함께 DescribeJob 사용](#)
 - [AWS SDK와 함께 GetJobTagging 사용](#)
 - [AWS SDK와 함께 PutJobTagging 사용](#)
 - [AWS SDK 또는 CLI와 함께 UpdateJobPriority 사용](#)
 - [AWS SDK 또는 CLI와 함께 UpdateJobStatus 사용](#)

AWS SDK를 사용한 Amazon S3 Control 기본 예제

다음 코드 예제에서는 AWS SDK를 통해 Amazon S3 Control의 기본 기능을 사용하는 방법을 보여줍니다.

예제

- [Amazon S3 Control 시작](#)
- [AWS SDK를 사용하는 Amazon S3 Control의 기본 기능 알아보기](#)
- [AWS SDK를 사용한 Amazon S3 Control 작업](#)
 - [AWS SDK 또는 CLI와 함께 CreateJob 사용](#)
 - [AWS SDK와 함께 DeleteJobTagging 사용](#)
 - [AWS SDK 또는 CLI와 함께 DescribeJob 사용](#)
 - [AWS SDK와 함께 GetJobTagging 사용](#)
 - [AWS SDK와 함께 PutJobTagging 사용](#)

- [AWS SDK 또는 CLI와 함께 UpdateJobPriority 사용](#)
- [AWS SDK 또는 CLI와 함께 UpdateJobStatus 사용](#)

Amazon S3 Control 시작

다음 코드 예제에서는 Amazon S3 Control 사용을 시작하는 방법을 보여줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryMode;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlAsyncClient;
import software.amazon.awssdk.services.s3control.model.JobListDescriptor;
import software.amazon.awssdk.services.s3control.model.JobStatus;
import software.amazon.awssdk.services.s3control.model.ListJobsRequest;
import software.amazon.awssdk.services.s3control.paginators.ListJobsPublisher;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this example:
 * <p/>
 * The SDK must be able to authenticate AWS requests on your behalf. If you have
 * not configured
```

```

    * authentication for SDKs and tools, see https://docs.aws.amazon.com/sdkref/latest/guide/access.html in the AWS SDKs and Tools Reference Guide.
    * <p/>
    * You must have a runtime environment configured with the Java SDK.
    * See https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html in the Developer Guide if this is not set up.
    */
public class HelloS3Batch {
    private static S3ControlAsyncClient asyncClient;

    public static void main(String[] args) {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        try {
            listBatchJobsAsync(accountId)
                .exceptionally(ex -> {
                    System.err.println("List batch jobs failed: " +
ex.getMessage());
                    return null;
                })
                .join();

        } catch (CompletionException ex) {
            System.err.println("Failed to list batch jobs: " + ex.getMessage());
        }
    }

    /**
     * Retrieves the asynchronous S3 Control client instance.
     * <p>
     * This method creates and returns a singleton instance of the {@link
S3ControlAsyncClient}. If the instance
     * has not been created yet, it will be initialized with the following
configuration:
     * <ul>
     * <li>Maximum concurrency: 100</li>
     * <li>Connection timeout: 60 seconds</li>
     * <li>Read timeout: 60 seconds</li>
     * <li>Write timeout: 60 seconds</li>
     * <li>API call timeout: 2 minutes</li>
     * <li>API call attempt timeout: 90 seconds</li>
     * <li>Retry policy: 3 retries</li>
     * <li>Region: US_EAST_1</li>
     * </ul>

```



```

    * <li>Credentials provider: {@link
EnvironmentVariableCredentialsProvider}</li>
    * </ul>
    *
    * @return the asynchronous S3 Control client instance
    */
    private static S3ControlAsyncClient getAsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            asyncClient = S3ControlAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return asyncClient;
    }

    /**
     * Asynchronously lists batch jobs that have completed for the specified
     * account.
     *
     * @param accountId the ID of the account to list jobs for
     * @return a CompletableFuture that completes when the job listing operation
     * is finished
     */
    public static CompletableFuture<Void> listBatchJobsAsync(String accountId) {
        ListJobsRequest jobsRequest = ListJobsRequest.builder()
            .jobStatuses(JobStatus.COMPLETE)
            .accountId(accountId)
            .maxResults(10)

```

```

        .build();

        ListJobsPublisher publisher =
getAsyncClient().listJobsPaginator(jobsRequest);
        return publisher.subscribe(response -> {
            List<JobListDescriptor> jobs = response.jobs();
            for (JobListDescriptor job : jobs) {
                System.out.println("The job id is " + job.jobId());
                System.out.println("The job priority is " + job.priority());
            }
        }).thenAccept(response -> {
            System.out.println("Listing batch jobs completed");
        }).exceptionally(ex -> {
            System.err.println("Failed to list batch jobs: " + ex.getMessage());
            throw new RuntimeException(ex);
        });
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [ListJobs](#)를 참조하십시오.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

def list_jobs(self, account_id: str) -> None:
    """
    List all batch jobs for the account.

    Args:
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.list_jobs(
            AccountId=account_id,

```

```

        JobStatuses=['Active', 'Complete', 'Cancelled', 'Failed', 'New',
'Paused', 'Pausing', 'Preparing', 'Ready', 'Suspended']
    )
    jobs = response.get('Jobs', [])
    for job in jobs:
        print(f"The job id is {job['JobId']}")
        print(f"The job priority is {job['Priority']}")
except ClientError as e:
    print(f"Error listing jobs: {e}")
    raise

```

- API 세부 정보는 AWS SDK for Python(Boto3) API 참조의 [ListJobs](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하는 Amazon S3 Control의 기본 기능 알아보기

다음 코드 예제에서는 Amazon S3 Control의 핵심 작업을 학습하는 방법을 보여줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

핵심 작업에 대해 알아봅니다.

```

package com.example.s3.batch;

import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.IOException;
import java.util.Map;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletionException;

```

```
public class S3BatchScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
    "-");
    private static final String STACK_NAME = "MyS3Stack";
    public static void main(String[] args) throws IOException {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        String uuid = java.util.UUID.randomUUID().toString();
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon S3 Batch basics scenario.");
        System.out.println("""
            S3 Batch operations enables efficient and cost-effective processing
of large-scale
            data stored in Amazon S3. It automatically scales resources to handle
varying workloads
            without the need for manual intervention.

            One of the key features of S3 Batch is its ability to perform tagging
operations on objects stored in
            S3 buckets. Users can leverage S3 Batch to apply, update, or remove
tags on thousands or millions of
            objects in a single operation, streamlining the management and
organization of their data.

            This can be particularly useful for tasks such as cost allocation,
lifecycle management, or
            metadata-driven workflows, where consistent and accurate tagging is
essential.

            S3 Batch's scalability and serverless nature make it an ideal
solution for organizations with
            growing data volumes and complex data management requirements.

            This Java program walks you through Amazon S3 Batch operations.

            Let's get started...

            """);
        waitForInputToContinue(scanner);
        // Use CloudFormation to stand up the resource required for this
scenario.
```

```
        System.out.println("Use CloudFormation to stand up the resource required
for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME);

        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputs(STACK_NAME);
        String iamRoleArn = stackOutputs.get("S3BatchRoleArn");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Setup the required bucket for this scenario.");
        waitForInputToContinue(scanner);
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
name.
        actions.createBucket(bucketName);
        String reportBucketName = "arn:aws:s3:::"+bucketName;
        String manifestLocation = "arn:aws:s3:::"+bucketName+"/job-manifest.csv";
        System.out.println("Populate the bucket with the required files.");
        String[] fileNames = {"job-manifest.csv", "object-key-1.txt", "object-
key-2.txt", "object-key-3.txt", "object-key-4.txt"};
        actions.uploadFilesToBucket(bucketName, fileNames, actions);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a S3 Batch Job");
        System.out.println("This job tags all objects listed in the manifest file
with tags");
        waitForInputToContinue(scanner);
        String jobId ;
        try {
            jobId = actions.createS3JobAsync(accountId, iamRoleArn,
manifestLocation, reportBucketName, uuid).join();
            System.out.println("The Job id is " + jobId);

        } catch (S3Exception e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
```

```

        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Update an existing S3 Batch Operations job's
priority");
        System.out.println("""
            In this step, we modify the job priority value. The higher the
number, the higher the priority.
            So, a job with a priority of `30` would have a higher priority than
a job with
            a priority of `20`. This is a common way to represent the priority
of a task
            or job, with higher numbers indicating a higher priority.

            Ensure that the job status allows for priority updates. Jobs in
certain
            states (e.g., Cancelled, Failed, or Completed) cannot have their
priorities
            updated. Only jobs in the Active or Suspended state typically allow
priority
            updates.
            """);

        try {
            actions.updateJobPriorityAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Update job priority failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to update job priority: " +
ex.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Cancel the S3 Batch job");
        System.out.print("Do you want to cancel the Batch job? (y/n): ");
        String cancelAns = scanner.nextLine();
        if (cancelAns != null && cancelAns.trim().equalsIgnoreCase("y")) {
            try {

```

```
        actions.cancelJobAsync(jobId, accountId)
            .exceptionally(ex -> {
                System.err.println("Cancel job failed: " +
ex.getMessage());
                return null;
            })
            .join();
    } catch (CompletionException ex) {
        System.err.println("Failed to cancel job: " + ex.getMessage());
    }
} else {
    System.out.println("Job " + jobId + " was not canceled.");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Describe the job that was just created");
waitForInputToContinue(scanner);
try {
    actions.describeJobAsync(jobId, accountId)
        .exceptionally(ex -> {
            System.err.println("Describe job failed: " +
ex.getMessage());
            return null;
        })
        .join();
} catch (CompletionException ex) {
    System.err.println("Failed to describe job: " + ex.getMessage());
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Describe the tags associated with the job");
waitForInputToContinue(scanner);
try {
    actions.getJobTagsAsync(jobId, accountId)
        .exceptionally(ex -> {
            System.err.println("Get job tags failed: " +
ex.getMessage());
            return null;
        })
        .join();
} catch (CompletionException ex) {
    System.err.println("Failed to get job tags: " + ex.getMessage());
}
```

```
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("6. Update Batch Job Tags");
    waitForInputToContinue(scanner);
    try {
        actions.putJobTaggingAsync(jobId, accountId)
            .exceptionally(ex -> {
                System.err.println("Put job tagging failed: " +
ex.getMessage());
                return null;
            })
            .join();
    } catch (CompletionException ex) {
        System.err.println("Failed to put job tagging: " + ex.getMessage());
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("7. Delete the Amazon S3 Batch job tagging.");
    System.out.print("Do you want to delete Batch job tagging? (y/n)");
    String delAns = scanner.nextLine();
    if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
        try {
            actions.deleteBatchJobTagsAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Delete batch job tags failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to delete batch job tags: " +
ex.getMessage());
        }
    } else {
        System.out.println("Tagging was not deleted.");
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.print("Do you want to delete the AWS resources used in this
scenario? (y/n)");
```



```
String delResAns = scanner.nextLine();
if (delResAns != null && delResAns.trim().equalsIgnoreCase("y")) {
    actions.deleteFilesFromBucket(bucketName, fileNames, actions);
    actions.deleteBucketFolderAsync(bucketName);
    actions.deleteBucket(bucketName)
        .thenRun(() -> System.out.println("Bucket deletion completed"))
        .exceptionally(ex -> {
            System.err.println("Error occurred: " + ex.getMessage());
            return null;
        });
    CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
} else {
    System.out.println("The AWS resources were not deleted.");
}
System.out.println("The Amazon S3 Batch scenario has successfully
completed.");
System.out.println(DASHES);
}

private static void waitForInputToContinue(Scanner scanner) {
    while (true) {
        System.out.println();
        System.out.println("Enter 'c' followed by <ENTER> to continue:");
        String input = scanner.nextLine();

        if (input.trim().equalsIgnoreCase("c")) {
            System.out.println("Continuing with the program...");
            System.out.println();
            break;
        } else {
            // Handle invalid input.
            System.out.println("Invalid input. Please try again.");
        }
    }
}
}
```

작업을 래핑하는 작업 클래스를 만듭니다.

```
public class S3BatchActions {
```

```

private static S3ControlAsyncClient asyncClient;

private static S3AsyncClient s3AsyncClient ;
/**
 * Retrieves the asynchronous S3 Control client instance.
 * <p>
 * This method creates and returns a singleton instance of the {@link
S3ControlAsyncClient}. If the instance
 * has not been created yet, it will be initialized with the following
configuration:
 * <ul>
 * <li>Maximum concurrency: 100</li>
 * <li>Connection timeout: 60 seconds</li>
 * <li>Read timeout: 60 seconds</li>
 * <li>Write timeout: 60 seconds</li>
 * <li>API call timeout: 2 minutes</li>
 * <li>API call attempt timeout: 90 seconds</li>
 * <li>Retry policy: 3 retries</li>
 * <li>Region: US_EAST_1</li>
 * <li>Credentials provider: {@link
EnvironmentVariableCredentialsProvider}</li>
 * </ul>
 *
 * @return the asynchronous S3 Control client instance
 */
private static S3ControlAsyncClient getAsyncClient() {
    if (asyncClient == null) {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();
    }
}

```

```

        asyncClient = S3ControlAsyncClient.builder()
            .region(Region.US_EAST_1)
            .httpClient(httpClient)
            .overrideConfiguration(overrideConfig)
            .build();
    }
    return asyncClient;
}

private static S3AsyncClient getS3AsyncClient() {
    if (asyncClient == null) {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
            ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

        s3AsyncClient = S3AsyncClient.builder()
            .region(Region.US_EAST_1)
            .httpClient(httpClient)
            .overrideConfiguration(overrideConfig)
            .build();
    }
    return s3AsyncClient;
}

/**
 * Cancels a job asynchronously.
 *
 * @param jobId The ID of the job to be canceled.
 * @param accountId The ID of the account associated with the job.
 * @return A {@link CompletableFuture} that completes when the job status has
 *         been updated to "CANCELLED".
 *         If an error occurs during the update, the returned future will
 *         complete exceptionally.

```

```

    */
    public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId)
    {
        UpdateJobStatusRequest updateJobStatusRequest =
UpdateJobStatusRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))
            .build();

        return asyncClient.updateJobStatus(updateJobStatusRequest)
            .thenAccept(updateJobStatusResponse -> {
                System.out.println("Job status updated to: " +
updateJobStatusResponse.status());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to cancel job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    /**
     * Updates the priority of a job asynchronously.
     *
     * @param jobId      the ID of the job to update
     * @param accountId the ID of the account associated with the job
     * @return a {@link CompletableFuture} that represents the asynchronous
     operation, which completes when the job priority has been updated or an error
     has occurred
     */
    public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
        UpdateJobPriorityRequest priorityRequest =
UpdateJobPriorityRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .priority(60)
            .build();

        CompletableFuture<Void> future = new CompletableFuture<>();
        getAsyncClient().updateJobPriority(priorityRequest)
            .thenAccept(response -> {
                System.out.println("The job priority was updated");
            });
    }

```

```

        future.complete(null); // Complete the CompletableFuture on
successful execution
    })
    .exceptionally(ex -> {
        System.err.println("Failed to update job priority: " +
ex.getMessage());
        future.completeExceptionally(ex); // Complete the
CompletableFuture exceptionally on error
        return null; // Return null to handle the exception
    });

    return future;
}

/**
 * Asynchronously retrieves the tags associated with a specific job in an AWS
account.
 *
 * @param jobId      the ID of the job for which to retrieve the tags
 * @param accountId the ID of the AWS account associated with the job
 * @return a {@link CompletableFuture} that completes when the job tags have
been retrieved, or with an exception if the operation fails
 * @throws RuntimeException if an error occurs while retrieving the job tags
 */
public CompletableFuture<Void> getJobTagsAsync(String jobId, String
accountId) {
    GetJobTaggingRequest request = GetJobTaggingRequest.builder()
        .jobId(jobId)
        .accountId(accountId)
        .build();

    return asyncClient.getJobTagging(request)
        .thenAccept(response -> {
            List<S3Tag> tags = response.tags();
            if (tags.isEmpty()) {
                System.out.println("No tags found for job ID: " + jobId);
            } else {
                for (S3Tag tag : tags) {
                    System.out.println("Tag key is: " + tag.key());
                    System.out.println("Tag value is: " + tag.value());
                }
            }
        })
        .exceptionally(ex -> {

```

```

        System.err.println("Failed to get job tags: " + ex.getMessage());
        throw new RuntimeException(ex); // Propagate the exception
    });
}

/**
 * Asynchronously deletes the tags associated with a specific batch job.
 *
 * @param jobId      The ID of the batch job whose tags should be deleted.
 * @param accountId  The ID of the account associated with the batch job.
 * @return A CompletableFuture that completes when the job tags have been
 * successfully deleted, or an exception is thrown if the deletion fails.
 */
public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {
    DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
        .accountId(accountId)
        .jobId(jobId)
        .build();

    return asyncClient.deleteJobTagging(jobTaggingRequest)
        .thenAccept(response -> {
            System.out.println("You have successfully deleted " + jobId + "
tagging.");
        })
        .exceptionally(ex -> {
            System.err.println("Failed to delete job tags: " +
ex.getMessage());
            throw new RuntimeException(ex);
        });
}

/**
 * Asynchronously describes the specified job.
 *
 * @param jobId      the ID of the job to describe
 * @param accountId  the ID of the AWS account associated with the job
 * @return a {@link CompletableFuture} that completes when the job
 * description is available
 * @throws RuntimeException if an error occurs while describing the job
 */
public CompletableFuture<Void> describeJobAsync(String jobId, String
accountId) {

```

```
DescribeJobRequest jobRequest = DescribeJobRequest.builder()
    .jobId(jobId)
    .accountId(accountId)
    .build();

return getAsyncClient().describeJob(jobRequest)
    .thenAccept(response -> {
        System.out.println("Job ID: " + response.job().jobId());
        System.out.println("Description: " +
response.job().description());
        System.out.println("Status: " + response.job().statusAsString());
        System.out.println("Role ARN: " + response.job().roleArn());
        System.out.println("Priority: " + response.job().priority());
        System.out.println("Progress Summary: " +
response.job().progressSummary());

        // Print out details about the job manifest.
        JobManifest manifest = response.job().manifest();
        System.out.println("Manifest Location: " +
manifest.location().objectArn());
        System.out.println("Manifest ETag: " +
manifest.location().eTag());

        // Print out details about the job operation.
        JobOperation operation = response.job().operation();
        if (operation.s3PutObjectTagging() != null) {
            System.out.println("Operation: S3 Put Object Tagging");
            System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
        }

        // Print out details about the job report.
        JobReport report = response.job().report();
        System.out.println("Report Bucket: " + report.bucket());
        System.out.println("Report Prefix: " + report.prefix());
        System.out.println("Report Format: " + report.format());
        System.out.println("Report Enabled: " + report.enabled());
        System.out.println("Report Scope: " +
report.reportScopeAsString());
    })
    .exceptionally(ex -> {
        System.err.println("Failed to describe job: " + ex.getMessage());
        throw new RuntimeException(ex);
    });
```

```

    }

    /**
     * Creates an asynchronous S3 job using the AWS Java SDK.
     *
     * @param accountId      the AWS account ID associated with the job
     * @param iamRoleArn     the ARN of the IAM role to be used for the job
     * @param manifestLocation the location of the job manifest file in S3
     * @param reportBucketName the name of the S3 bucket to store the job report
     * @param uuid           a unique identifier for the job
     * @return a CompletableFuture that represents the asynchronous creation of
     the S3 job.
     *
     * The CompletableFuture will return the job ID if the job is created
     successfully,
     *
     * or throw an exception if there is an error.
     */
    public CompletableFuture<String> createS3JobAsync(String accountId, String
                                                    iamRoleArn,
                                                    String manifestLocation,
                                                    String reportBucketName, String uuid) {

        String[] bucketName = new String[]{" "};
        String[] parts = reportBucketName.split("::");
        if (parts.length > 1) {
            bucketName[0] = parts[1];
        } else {
            System.out.println("The input string does not contain the expected
format.");
        }

        return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
            .thenCompose(eTag -> {
                ArrayList<S3Tag> tagSet = new ArrayList<>();
                S3Tag s3Tag = S3Tag.builder()
                    .key("keyOne")
                    .value("ValueOne")
                    .build();
                S3Tag s3Tag2 = S3Tag.builder()
                    .key("keyTwo")
                    .value("ValueTwo")
                    .build();
                tagSet.add(s3Tag);
                tagSet.add(s3Tag2);
            });
    }

```



```
S3SetObjectTaggingOperation objectTaggingOperation =
S3SetObjectTaggingOperation.builder()
    .tagSet(tagSet)
    .build();

JobOperation jobOperation = JobOperation.builder()
    .s3PutObjectTagging(objectTaggingOperation)
    .build();

JobManifestLocation jobManifestLocation =
JobManifestLocation.builder()
    .objectArn(manifestLocation)
    .eTag(eTag)
    .build();

JobManifestSpec manifestSpec = JobManifestSpec.builder()
    .fieldsWithStrings("Bucket", "Key")
    .format("S3BatchOperations_CSV_20180820")
    .build();

JobManifest jobManifest = JobManifest.builder()
    .spec(manifestSpec)
    .location(jobManifestLocation)
    .build();

JobReport jobReport = JobReport.builder()
    .bucket(reportBucketName)
    .prefix("reports")
    .format("Report_CSV_20180820")
    .enabled(true)
    .reportScope("AllTasks")
    .build();

CreateJobRequest jobRequest = CreateJobRequest.builder()
    .accountId(accountId)
    .description("Job created using the AWS Java SDK")
    .manifest(jobManifest)
    .operation(jobOperation)
    .report(jobReport)
    .priority(42)
    .roleArn(iamRoleArn)
    .clientRequestToken(uuid)
    .confirmationRequired(false)
```

```

        .build();

        // Create the job asynchronously.
        return getAsyncClient().createJob(jobRequest)
            .thenApply(CreateJobResponse::jobId);
    })
    .handle((jobId, ex) -> {
        if (ex != null) {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof S3ControlException) {
                throw new CompletionException(cause);
            } else {
                throw new RuntimeException(cause);
            }
        }
        return jobId;
    });
}

/**
 * Retrieves the ETag (Entity Tag) for an object stored in an Amazon S3
bucket.
 *
 * @param bucketName the name of the Amazon S3 bucket where the object is
stored
 * @param key the key (file name) of the object in the Amazon S3 bucket
 * @return the ETag of the object
 */
public String getETag(String bucketName, String key) {
    S3Client s3Client = S3Client.builder()
        .region(Region.US_EAST_1)
        .build();

    HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    HeadObjectResponse headObjectResponse =
s3Client.headObject(headObjectRequest);
    return headObjectResponse.eTag();
}

```

```
/**
 * Asynchronously adds tags to a job in the system.
 *
 * @param jobId      the ID of the job to add tags to
 * @param accountId the account ID associated with the job
 * @return a CompletableFuture that completes when the tagging operation is
finished
 */
public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
    S3Tag departmentTag = S3Tag.builder()
        .key("department")
        .value("Marketing")
        .build();

    S3Tag fiscalYearTag = S3Tag.builder()
        .key("FiscalYear")
        .value("2020")
        .build();

    PutJobTaggingRequest putJobTaggingRequest =
PutJobTaggingRequest.builder()
        .jobId(jobId)
        .accountId(accountId)
        .tags(departmentTag, fiscalYearTag)
        .build();

    return asyncClient.putJobTagging(putJobTaggingRequest)
        .thenRun(() -> {
            System.out.println("Additional Tags were added to job " + jobId);
        })
        .exceptionally(ex -> {
            System.err.println("Failed to add tags to job: " +
ex.getMessage());
            throw new RuntimeException(ex); // Propagate the exception
        });
}

// Setup the S3 bucket required for this scenario.
/**
 * Creates an Amazon S3 bucket with the specified name.
 *
 * @param bucketName the name of the S3 bucket to create
 * @throws S3Exception if there is an error creating the bucket
 */
```

```

    */
    public void createBucket(String bucketName) {
        try {
            S3Client s3Client = S3Client.builder()
                .region(Region.US_EAST_1)
                .build();

            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                .bucket(bucketName)
                .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Uploads a file to an Amazon S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param fileName the name of the file to be uploaded
     * @throws RuntimeException if an error occurs during the file upload
     */
    public void populateBucket(String bucketName, String fileName) {
        // Define the path to the directory.
        Path filePath = Paths.get("src/main/resources/batch/",
fileName).toAbsolutePath();
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(fileName)

```

```

        .build();

        CompletableFuture<PutObjectResponse> future =
getS3AsyncClient().putObject(putOb, AsyncRequestBody.fromFile(filePath));
        future.whenComplete((result, ex) -> {
            if (ex != null) {
                System.err.println("Error uploading file: " + ex.getMessage());
            } else {
                System.out.println("Successfully placed " + fileName + " into
bucket " + bucketName);
            }
        }).join();
    }

    // Update the bucketName in CSV.
    public void updateCSV(String newValue) {
        Path csvFilePath = Paths.get("src/main/resources/batch/job-
manifest.csv").toAbsolutePath();
        try {
            // Read all lines from the CSV file.
            List<String> lines = Files.readAllLines(csvFilePath);

            // Update the first value in each line.
            List<String> updatedLines = lines.stream()
                .map(line -> {
                    String[] parts = line.split(",");
                    parts[0] = newValue;
                    return String.join(",", parts);
                })
                .collect(Collectors.toList());

            // Write the updated lines back to the CSV file
            Files.write(csvFilePath, updatedLines);
            System.out.println("CSV file updated successfully.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /**
     * Deletes an object from an Amazon S3 bucket asynchronously.
     *
     * @param bucketName The name of the S3 bucket where the object is stored.

```

```

    * @param objectName The name of the object to be deleted.
    * @return A {@link CompletableFuture} that completes when the object has
    been deleted,
    *         or throws a {@link RuntimeException} if an error occurs during the
    deletion.
    */
    public CompletableFuture<Void> deleteBucketObjects(String bucketName, String
objectName) {
        ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
        toDelete.add(ObjectIdentifier.builder()
            .key(objectName)
            .build());

        DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
            .bucket(bucketName)
            .delete(Delete.builder()
                .objects(toDelete).build())
            .build();

        return getS3AsyncClient().deleteObjects(dor)
            .thenAccept(result -> {
                System.out.println("The object was deleted!");
            })
            .exceptionally(ex -> {
                throw new RuntimeException("Error deleting object: " +
ex.getMessage(), ex);
            });
    }

    /**
     * Deletes a folder and all its contents asynchronously from an Amazon S3
    bucket.
     *
     * @param bucketName the name of the S3 bucket containing the folder to be
    deleted
     * @return a {@link CompletableFuture} that completes when the folder and its
    contents have been deleted
     * @throws RuntimeException if any error occurs during the deletion process
     */
    public void deleteBucketFolderAsync(String bucketName) {
        String folderName = "reports/";
        ListObjectsV2Request request = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .prefix(folderName)

```

```

        .build();

        CompletableFuture<ListObjectsV2Response> listObjectsFuture =
getS3AsyncClient().listObjectsV2(request);
        listObjectsFuture.thenCompose(response -> {
            List<CompletableFuture<DeleteObjectResponse>> deleteFutures =
response.contents().stream()
                .map(obj -> {
                    DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
                        .bucket(bucketName)
                        .key(obj.key())
                        .build();
                    return getS3AsyncClient().deleteObject(deleteRequest)
                        .thenApply(deleteResponse -> {
                            System.out.println("Deleted object: " + obj.key());
                            return deleteResponse;
                        });
                })
                .collect(Collectors.toList());

            return CompletableFuture.allOf(deleteFutures.toArray(new
CompletableFuture[0]))
                .thenCompose(v -> {
                    // Delete the folder.
                    DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
                        .bucket(bucketName)
                        .key(folderName)
                        .build();
                    return getS3AsyncClient().deleteObject(deleteRequest)
                        .thenApply(deleteResponse -> {
                            System.out.println("Deleted folder: " + folderName);
                            return deleteResponse;
                        });
                });
        }).join();
    }

/**
 * Deletes an Amazon S3 bucket.
 *
 * @param bucketName the name of the bucket to delete

```

```

    * @return a {@link CompletableFuture} that completes when the bucket has
    been deleted, or exceptionally if there is an error
    * @throws RuntimeException if there is an error deleting the bucket
    */
    public CompletableFuture<Void> deleteBucket(String bucketName) {
        S3AsyncClient s3Client = getS3AsyncClient();
        return s3Client.deleteBucket(DeleteBucketRequest.builder()
            .bucket(bucketName)
            .build())
            .thenAccept(deleteBucketResponse -> {
                System.out.println(bucketName + " was deleted");
            })
            .exceptionally(ex -> {
                // Handle the exception or rethrow it.
                throw new RuntimeException("Failed to delete bucket: " +
bucketName, ex);
            });
    }

    /**
     * Uploads a set of files to an Amazon S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to upload the files to
     * @param fileNames an array of file names to be uploaded
     * @param actions an instance of {@link S3BatchActions} that provides the
    implementation for the necessary S3 operations
     * @throws IOException if there's an error creating the text files or
    uploading the files to the S3 bucket
     */
    public static void uploadFilesToBucket(String bucketName, String[] fileNames,
S3BatchActions actions) throws IOException {
        actions.updateCSV(bucketName);
        createTextFiles(fileNames);
        for (String fileName : fileNames) {
            actions.populateBucket(bucketName, fileName);
        }
        System.out.println("All files are placed in the S3 bucket " +
bucketName);
    }

    /**
     * Deletes the specified files from the given S3 bucket.
     *
     * @param bucketName the name of the S3 bucket

```



```

    * @param fileNames an array of file names to be deleted from the bucket
    * @param actions the S3BatchActions instance to be used for the file
    deletion
    * @throws IOException if an I/O error occurs during the file deletion
    */
    public void deleteFilesFromBucket(String bucketName, String[] fileNames,
    S3BatchActions actions) throws IOException {
        for (String fileName : fileNames) {
            actions.deleteBucketObjects(bucketName, fileName)
                .thenRun(() -> System.out.println("Object deletion completed"))
                .exceptionally(ex -> {
                    System.err.println("Error occurred: " + ex.getMessage());
                    return null;
                });
        }
        System.out.println("All files have been deleted from the bucket " +
        bucketName);
    }

    public static void createTextFiles(String[] fileNames) {
        String currentDirectory = System.getProperty("user.dir");
        String directoryPath = currentDirectory + "\\src\\main\\resources\\
        \\batch";
        Path path = Paths.get(directoryPath);

        try {
            // Create the directory if it doesn't exist.
            if (Files.notExists(path)) {
                Files.createDirectories(path);
                System.out.println("Created directory: " + path.toString());
            } else {
                System.out.println("Directory already exists: " +
        path.toString());
            }

            for (String fileName : fileNames) {
                // Check if the file is a .txt file.
                if (fileName.endsWith(".txt")) {
                    // Define the path for the new file.
                    Path filePath = path.resolve(fileName);
                    System.out.println("Attempting to create file: " +
        filePath.toString());

                    // Create and write content to the new file.

```

```
        Files.write(filePath, "This is a test".getBytes());

        // Verify the file was created.
        if (Files.exists(filePath)) {
            System.out.println("Successfully created file: " +
filePath.toString());
        } else {
            System.out.println("Failed to create file: " +
filePath.toString());
        }
    }

    } catch (IOException e) {
        System.err.println("An error occurred: " + e.getMessage());
        e.printStackTrace();
    }
}

public String getAccountId() {
    StsClient stsClient = StsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    GetCallerIdentityResponse callerIdentityResponse =
stsClient.getCallerIdentity();
    return callerIdentityResponse.account();
}
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
 - [CreateJob](#).
 - [DeleteJobTagging](#)
 - [DescribeJob](#)
 - [GetJobTagging](#)
 - [ListJobs](#)
 - [PutJobTagging](#)
 - [UpdateJobPriority](#)
 - [UpdateJobStatus](#)

Python

SDK for Python(Boto3)

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

S3 Batch 기본 사항 시나리오에 대해 알아봅니다.

```
class S3BatchWrapper:
    """Wrapper class for managing S3 Batch Operations."""

    def __init__(self, s3_client: Any, s3control_client: Any, sts_client: Any) ->
    None:
        """
        Initializes the S3BatchWrapper with AWS service clients.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-
        level
                           access to AWS S3 services.
        :param s3control_client: A Boto3 Amazon S3 Control client. This client
        provides
                           low-level access to AWS S3 Control services.
        :param sts_client: A Boto3 AWS STS client. This client provides low-level
                           access to AWS STS services.
        """
        self.s3_client = s3_client
        self.s3control_client = s3control_client
        self.sts_client = sts_client
        # Get region from the client for bucket creation logic
        self.region_name = self.s3_client.meta.region_name

    def get_account_id(self) -> str:
        """
        Get AWS account ID.

        Returns:
            str: AWS account ID
        """
        return self.sts_client.get_caller_identity()["Account"]
```

```
def create_bucket(self, bucket_name: str) -> None:
    """
    Create an S3 bucket.

    Args:
        bucket_name (str): Name of the bucket to create

    Raises:
        ClientError: If bucket creation fails
    """
    try:
        if self.region_name and self.region_name != 'us-east-1':
            self.s3_client.create_bucket(
                Bucket=bucket_name,
                CreateBucketConfiguration={
                    'LocationConstraint': self.region_name
                }
            )
        else:
            self.s3_client.create_bucket(Bucket=bucket_name)
        print(f"Created bucket: {bucket_name}")
    except ClientError as e:
        print(f"Error creating bucket: {e}")
        raise

def upload_files_to_bucket(self, bucket_name: str, file_names: List[str]) -> str:
    """
    Upload files to S3 bucket including manifest file.

    Args:
        bucket_name (str): Target bucket name
        file_names (list): List of file names to upload

    Returns:
        str: ETag of the manifest file

    Raises:
        ClientError: If file upload fails
    """
    try:
        for file_name in file_names:
            if file_name != "job-manifest.csv":
```

```

        content = f"Content for {file_name}"
        self.s3_client.put_object(
            Bucket=bucket_name,
            Key=file_name,
            Body=content.encode('utf-8')
        )
        print(f"Uploaded {file_name} to {bucket_name}")

    manifest_content = ""
    for file_name in file_names:
        if file_name != "job-manifest.csv":
            manifest_content += f"{bucket_name},{file_name}\n"

    manifest_response = self.s3_client.put_object(
        Bucket=bucket_name,
        Key="job-manifest.csv",
        Body=manifest_content.encode('utf-8')
    )
    print(f"Uploaded manifest file to {bucket_name}")
    print(f"Manifest content:\n{manifest_content}")
    return manifest_response['ETag'].strip('')

except ClientError as e:
    print(f"Error uploading files: {e}")
    raise

def create_s3_batch_job(self, account_id: str, role_arn: str,
manifest_location: str,
                        report_bucket_name: str) -> str:
    """
    Create an S3 batch operation job.

    Args:
        account_id (str): AWS account ID
        role_arn (str): IAM role ARN for batch operations
        manifest_location (str): Location of the manifest file
        report_bucket_name (str): Bucket for job reports

    Returns:
        str: Job ID

    Raises:
        ClientError: If job creation fails
    """

```

```
try:
    bucket_name = manifest_location.split(':::')[1].split('/')[0]
    manifest_key = 'job-manifest.csv'
    manifest_obj = self.s3_client.head_object(
        Bucket=bucket_name,
        Key=manifest_key
    )
    etag = manifest_obj['ETag'].strip('"')

    response = self.s3control_client.create_job(
        AccountId=account_id,
        Operation={
            'S3PutObjectTagging': {
                'TagSet': [
                    {
                        'Key': 'BatchTag',
                        'Value': 'BatchValue'
                    },
                ],
            }
        },
        Report={
            'Bucket': report_bucket_name,
            'Format': 'Report_CSV_20180820',
            'Enabled': True,
            'Prefix': 'batch-op-reports',
            'ReportScope': 'AllTasks'
        },
        Manifest={
            'Spec': {
                'Format': 'S3BatchOperations_CSV_20180820',
                'Fields': ['Bucket', 'Key']
            },
            'Location': {
                'ObjectArn': manifest_location,
                'ETag': etag
            }
        },
        Priority=10,
        RoleArn=role_arn,
        Description='Batch job for tagging objects',
        ConfirmationRequired=True
    )
    job_id = response['JobId']
```

```

        print(f"The Job id is {job_id}")
        return job_id
    except ClientError as e:
        print(f"Error creating batch job: {e}")
        if 'Message' in str(e):
            print(f"Detailed error message: {e.response['Message']}")
        raise

    def check_job_failure_reasons(self, job_id: str, account_id: str) ->
    List[Dict[str, Any]]:
        """
        Check for any failure reasons of a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID

        Returns:
            list: List of failure reasons

        Raises:
            ClientError: If checking job failure reasons fails
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            if 'FailureReasons' in response['Job']:
                for reason in response['Job']['FailureReasons']:
                    print(f"- {reason}")
            return response['Job'].get('FailureReasons', [])
        except ClientError as e:
            print(f"Error checking job failure reasons: {e}")
            raise

    def wait_for_job_ready(self, job_id: str, account_id: str, desired_status:
    str = 'Ready') -> bool:
        """
        Wait for a job to reach the desired status.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID

```

```

        desired_status (str): Target status to wait for

Returns:
    bool: True if desired status is reached, False otherwise

Raises:
    ClientError: If checking job status fails
    """
    print(f"Waiting for job to become {desired_status}...")
    max_attempts = 60
    attempt = 0
    while attempt < max_attempts:
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            current_status = response['Job']['Status']
            print(f"Current job status: {current_status}")
            if current_status == desired_status:
                return True
            if current_status == 'Suspended':
                print("Job is in Suspended state, can proceed with
activation")
                return True
            if current_status in ['Active', 'Failed', 'Cancelled',
'Complete']:
                print(f"Job is in {current_status} state, cannot reach
{desired_status} status")
                if 'FailureReasons' in response['Job']:
                    print("Failure reasons:")
                    for reason in response['Job']['FailureReasons']:
                        print(f"- {reason}")
                    return False

            time.sleep(20)
            attempt += 1
        except ClientError as e:
            print(f"Error checking job status: {e}")
            raise
    print(f"Timeout waiting for job to become {desired_status}")
    return False

def update_job_priority(self, job_id: str, account_id: str) -> None:

```



```
"""
Update the priority of a batch job and start it.

Args:
    job_id (str): ID of the batch job
    account_id (str): AWS account ID
"""
try:
    response = self.s3control_client.describe_job(
        AccountId=account_id,
        JobId=job_id
    )
    current_status = response['Job']['Status']
    print(f"Current job status: {current_status}")

    if current_status in ['Ready', 'Suspended']:
        self.s3control_client.update_job_priority(
            AccountId=account_id,
            JobId=job_id,
            Priority=60
        )
        print("The job priority was updated")

    try:
        self.s3control_client.update_job_status(
            AccountId=account_id,
            JobId=job_id,
            RequestedJobStatus='Ready'
        )
        print("Job activated successfully")
    except ClientError as activation_error:
        print(f"Note: Could not activate job automatically:
{activation_error}")
        print("Job priority was updated successfully. Job may need
manual activation in the console.")
    elif current_status in ['Active', 'Completing', 'Complete']:
        print(f"Job is in '{current_status}' state - priority cannot be
updated")

    if current_status == 'Completing':
        print("Job is finishing up and will complete soon.")
    elif current_status == 'Complete':
        print("Job has already completed successfully.")
    else:
        print("Job is currently running.")
```

```

        else:
            print(f"Job is in '{current_status}' state - priority update not
allowed")

    except ClientError as e:
        print(f"Error updating job priority: {e}")
        print("Continuing with the scenario...")
        return

def cancel_job(self, job_id: str, account_id: str) -> None:
    """
    Cancel an S3 batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        current_status = response['Job']['Status']
        print(f"Current job status: {current_status}")

        if current_status in ['Ready', 'Suspended', 'Active']:
            self.s3control_client.update_job_status(
                AccountId=account_id,
                JobId=job_id,
                RequestedJobStatus='Cancelled'
            )
            print(f"Job {job_id} was successfully canceled.")
        elif current_status in ['Completing', 'Complete']:
            print(f"Job is in '{current_status}' state - cannot be
cancelled")

            if current_status == 'Completing':
                print("Job is finishing up and will complete soon.")
            elif current_status == 'Complete':
                print("Job has already completed successfully.")
        else:
            print(f"Job is in '{current_status}' state - cancel not allowed")
    except ClientError as e:
        print(f"Error canceling job: {e}")
        raise

```

```

def describe_job_details(self, job_id: str, account_id: str) -> None:
    """
    Describe detailed information about a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        job = response['Job']
        print(f"Job ID: {job['JobId']}")
        print(f"Description: {job.get('Description', 'N/A')}")
        print(f"Status: {job['Status']}")
        print(f"Role ARN: {job['RoleArn']}")
        print(f"Priority: {job['Priority']}")
        if 'ProgressSummary' in job:
            progress = job['ProgressSummary']
            print(f"Progress Summary:
Total={progress.get('TotalNumberOfTasks', 0)}, "
                  f"Succeeded={progress.get('NumberOfTasksSucceeded', 0)}, "
                  f"Failed={progress.get('NumberOfTasksFailed', 0)}")
    except ClientError as e:
        print(f"Error describing job: {e}")
        raise

def get_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Get tags associated with a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.get_job_tagging(
            AccountId=account_id,
            JobId=job_id
        )
        tags = response.get('Tags', [])
    
```

```
        if tags:
            print(f"Tags for job {job_id}:")
            for tag in tags:
                print(f"  {tag['Key']}: {tag['Value']}")
        else:
            print(f"No tags found for job ID: {job_id}")
    except ClientError as e:
        print(f"Error getting job tags: {e}")
        raise

def put_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Add tags to a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        self.s3control_client.put_job_tagging(
            AccountId=account_id,
            JobId=job_id,
            Tags=[
                {'Key': 'Environment', 'Value': 'Development'},
                {'Key': 'Team', 'Value': 'DataProcessing'}
            ]
        )
        print(f"Additional tags were added to job {job_id}")
    except ClientError as e:
        print(f"Error adding job tags: {e}")
        raise

def list_jobs(self, account_id: str) -> None:
    """
    List all batch jobs for the account.

    Args:
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.list_jobs(
            AccountId=account_id,
            JobStatuses=['Active', 'Complete', 'Cancelled', 'Failed', 'New',
                        'Paused', 'Pausing', 'Preparing', 'Ready', 'Suspended']
        )
```

```

    )
    jobs = response.get('Jobs', [])
    for job in jobs:
        print(f"The job id is {job['JobId']}")
        print(f"The job priority is {job['Priority']}")
except ClientError as e:
    print(f"Error listing jobs: {e}")
    raise

def delete_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Delete all tags from a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        self.s3control_client.delete_job_tagging(
            AccountId=account_id,
            JobId=job_id
        )
        print(f"You have successfully deleted {job_id} tagging.")
    except ClientError as e:
        print(f"Error deleting job tags: {e}")
        raise

def cleanup_resources(self, bucket_name: str, file_names: List[str]) -> None:
    """
    Clean up all resources created during the scenario.

    Args:
        bucket_name (str): Name of the bucket to clean up
        file_names (list): List of files to delete

    Raises:
        ClientError: If cleanup fails
    """
    try:
        for file_name in file_names:
            self.s3_client.delete_object(Bucket=bucket_name, Key=file_name)
            print(f"Deleted {file_name}")

        response = self.s3_client.list_objects_v2(

```

```

        Bucket=bucket_name,
        Prefix='batch-op-reports/'
    )
    if 'Contents' in response:
        for obj in response['Contents']:
            self.s3_client.delete_object(
                Bucket=bucket_name,
                Key=obj['Key']
            )
            print(f"Deleted {obj['Key']}")

    self.s3_client.delete_bucket(Bucket=bucket_name)
    print(f"Deleted bucket {bucket_name}")
except ClientError as e:
    print(f"Error in cleanup: {e}")
    raise

```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 다음 주제를 참조하세요.
 - [CreateJob](#).
 - [DeleteJobTagging](#)
 - [DescribeJob](#)
 - [GetJobTagging](#)
 - [ListJobs](#)
 - [PutJobTagging](#)
 - [UpdateJobPriority](#)
 - [UpdateJobStatus](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용한 Amazon S3 Control 작업

다음 코드 예제에서는 AWS SDK를 통해 개별 Amazon S3 Control 작업을 수행하는 방법을 보여줍니다. 각 예시에는 GitHub에 대한 링크가 포함되어 있습니다. 여기에서 코드 설정 및 실행에 대한 지침을 찾을 수 있습니다.

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 [Amazon S3 Control API 참조](#)에서 확인하세요.

예제

- [AWS SDK 또는 CLI와 함께 CreateJob 사용](#)
- [AWS SDK와 함께 DeleteJobTagging 사용](#)
- [AWS SDK 또는 CLI와 함께 DescribeJob 사용](#)
- [AWS SDK와 함께 GetJobTagging 사용](#)
- [AWS SDK와 함께 PutJobTagging 사용](#)
- [AWS SDK 또는 CLI와 함께 UpdateJobPriority 사용](#)
- [AWS SDK 또는 CLI와 함께 UpdateJobStatus 사용](#)

AWS SDK 또는 CLI와 함께 **CreateJob** 사용

다음 코드 예시는 CreateJob의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [기본 사항 알아보기](#)

CLI

AWS CLI

Amazon S3 Batch Operations 작업 생성

다음 create-job 예시에서는 Amazon S3 Batch Operations 작업을 생성하고 객체를 `confidential` in the bucket ``employee-records`로 태그 지정합니다.

```
aws s3control create-job \
  --account-id 123456789012 \
  --operation '{"S3PutObjectTagging": { "TagSet": [{"Key":"confidential",
    "Value":"true"}] }}' \
  --report '{"Bucket":"arn:aws:s3:::employee-records-logs","Prefix":"batch-op-
    create-job",
    "Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}' \
  --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
    ["Bucket","Key"]},"Location":{"ObjectArn":"arn:aws:s3:::employee-records-logs/
```

```
inv-report/7a6a9be4-072c-407e-85a2-
ec3e982f773e.csv", "ETag": "69f52a4e9f797e987155d9c8f5880897"}}' \
--priority 42 \
--role-arn arn:aws:iam::123456789012:role/S3BatchJobRole
```

출력:

```
{
  "JobId": "93735294-df46-44d5-8638-6356f335324e"
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [CreateJob](#)을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

비동기 S3 작업을 만듭니다.

```
/**
 * Creates an asynchronous S3 job using the AWS Java SDK.
 *
 * @param accountId      the AWS account ID associated with the job
 * @param iamRoleArn     the ARN of the IAM role to be used for the job
 * @param manifestLocation the location of the job manifest file in S3
 * @param reportBucketName the name of the S3 bucket to store the job report
 * @param uuid           a unique identifier for the job
 * @return a CompletableFuture that represents the asynchronous creation of
 * the S3 job.
 *
 * The CompletableFuture will return the job ID if the job is created
 * successfully,
 *
 * or throw an exception if there is an error.
 */
public CompletableFuture<String> createS3JobAsync(String accountId, String
iamRoleArn,
```



```

String manifestLocation,
String reportBucketName, String uuid) {

    String[] bucketName = new String[]{" "};
    String[] parts = reportBucketName.split("::");
    if (parts.length > 1) {
        bucketName[0] = parts[1];
    } else {
        System.out.println("The input string does not contain the expected
format.");
    }

    return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
        .thenCompose(eTag -> {
            ArrayList<S3Tag> tagSet = new ArrayList<>();
            S3Tag s3Tag = S3Tag.builder()
                .key("keyOne")
                .value("ValueOne")
                .build();
            S3Tag s3Tag2 = S3Tag.builder()
                .key("keyTwo")
                .value("ValueTwo")
                .build();
            tagSet.add(s3Tag);
            tagSet.add(s3Tag2);

            S3SetObjectTaggingOperation objectTaggingOperation =
S3SetObjectTaggingOperation.builder()
                .tagSet(tagSet)
                .build();

            JobOperation jobOperation = JobOperation.builder()
                .s3PutObjectTagging(objectTaggingOperation)
                .build();

            JobManifestLocation jobManifestLocation =
JobManifestLocation.builder()
                .objectArn(manifestLocation)
                .eTag(eTag)
                .build();

            JobManifestSpec manifestSpec = JobManifestSpec.builder()
                .fieldsWithStrings("Bucket", "Key")

```

```
        .format("S3BatchOperations_CSV_20180820")
        .build();

    JobManifest jobManifest = JobManifest.builder()
        .spec(manifestSpec)
        .location(jobManifestLocation)
        .build();

    JobReport jobReport = JobReport.builder()
        .bucket(reportBucketName)
        .prefix("reports")
        .format("Report_CSV_20180820")
        .enabled(true)
        .reportScope("AllTasks")
        .build();

    CreateJobRequest jobRequest = CreateJobRequest.builder()
        .accountId(accountId)
        .description("Job created using the AWS Java SDK")
        .manifest(jobManifest)
        .operation(jobOperation)
        .report(jobReport)
        .priority(42)
        .roleArn(iamRoleArn)
        .clientRequestToken(uuid)
        .confirmationRequired(false)
        .build();

    // Create the job asynchronously.
    return getAsyncClient().createJob(jobRequest)
        .thenApply(CreateJobResponse::jobId)
        })
        .handle((jobId, ex) -> {
            if (ex != null) {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof S3ControlException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        })
        .return jobId;
    });
```

```
}
```

규정 준수 보존 작업을 만듭니다.

```
/**
 * Creates a compliance retention job in Amazon S3 Control.
 * <p>
 * A compliance retention job in Amazon S3 Control is a feature that allows
 you to
 * set a retention period for objects stored in an S3 bucket.
 * This feature is particularly useful for organizations that need to comply
 with
 * regulatory requirements or internal policies that mandate the retention of
 data for
 * a specific duration.
 *
 * @param s3ControlClient The S3ControlClient instance to use for the API
 call.
 * @return The job ID of the created compliance retention job.
 */
public static String createComplianceRetentionJob(final S3ControlClient
s3ControlClient, String roleArn, String bucketName, String accountId) {
    final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-
bucket/compliance-objects-manifest.csv";
    final String manifestObjectVersionId = "your-object-version-Id";

    Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
    JobOperation jobOperation = JobOperation.builder()
        .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
            .retention(S3Retention.builder()
                .mode(S3ObjectLockRetentionMode.COMPLIANCE)
                .retainUntilDate(jan2025)
                .build())
            .build())
        .build();

    JobManifestLocation manifestLocation = JobManifestLocation.builder()
        .objectArn(manifestObjectArn)
        .eTag(manifestObjectVersionId)
        .build();
```

```
JobManifestSpec manifestSpec = JobManifestSpec.builder()
    .fieldsWithStrings("Bucket", "Key")
    .format("S3BatchOperations_CSV_20180820")
    .build();

JobManifest manifestToPublicApi = JobManifest.builder()
    .location(manifestLocation)
    .spec(manifestSpec)
    .build();

// Report details.
final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
final String jobReportPrefix = "reports/compliance-objects-bops";

JobReport jobReport = JobReport.builder()
    .enabled(true)
    .reportScope(JobReportScope.ALL_TASKS)
    .bucket(jobReportBucketArn)
    .prefix(jobReportPrefix)
    .format(JobReportFormat.REPORT_CSV_20180820)
    .build();

final Boolean requiresConfirmation = true;
final int priority = 10;
CreateJobRequest request = CreateJobRequest.builder()
    .accountId(accountId)
    .description("Set compliance retain-until to 1 Jan 2025")
    .manifest(manifestToPublicApi)
    .operation(jobOperation)
    .priority(priority)
    .roleArn(roleArn)
    .report(jobReport)
    .confirmationRequired(requiresConfirmation)
    .build();

// Create the job and get the result.
CreateJobResponse result = s3ControlClient.createJob(request);
return result.jobId();
}
```

법적 보존 작업을 만듭니다.

```

/**
 * Creates a compliance retention job in Amazon S3 Control.
 * <p>
 * A compliance retention job in Amazon S3 Control is a feature that allows
you to
 * set a retention period for objects stored in an S3 bucket.
 * This feature is particularly useful for organizations that need to comply
with
 * regulatory requirements or internal policies that mandate the retention of
data for
 * a specific duration.
 *
 * @param s3ControlClient The S3ControlClient instance to use for the API
call.
 * @return The job ID of the created compliance retention job.
 */
public static String createComplianceRetentionJob(final S3ControlClient
s3ControlClient, String roleArn, String bucketName, String accountId) {
    final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-
bucket/compliance-objects-manifest.csv";
    final String manifestObjectVersionId = "your-object-version-Id";

    Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
    JobOperation jobOperation = JobOperation.builder()
        .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
            .retention(S3Retention.builder()
                .mode(S3ObjectLockRetentionMode.COMPLIANCE)
                .retainUntilDate(jan2025)
                .build())
            .build())
        .build();

    JobManifestLocation manifestLocation = JobManifestLocation.builder()
        .objectArn(manifestObjectArn)
        .eTag(manifestObjectVersionId)
        .build();

    JobManifestSpec manifestSpec = JobManifestSpec.builder()
        .fieldsWithStrings("Bucket", "Key")
        .format("S3BatchOperations_CSV_20180820")
        .build();

```

```

        JobManifest manifestToPublicApi = JobManifest.builder()
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();

// Report details.
final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
final String jobReportPrefix = "reports/compliance-objects-bops";

JobReport jobReport = JobReport.builder()
    .enabled(true)
    .reportScope(JobReportScope.ALL_TASKS)
    .bucket(jobReportBucketArn)
    .prefix(jobReportPrefix)
    .format(JobReportFormat.REPORT_CSV_20180820)
    .build();

final Boolean requiresConfirmation = true;
final int priority = 10;
CreateJobRequest request = CreateJobRequest.builder()
    .accountId(accountId)
    .description("Set compliance retain-until to 1 Jan 2025")
    .manifest(manifestToPublicApi)
    .operation(jobOperation)
    .priority(priority)
    .roleArn(roleArn)
    .report(jobReport)
    .confirmationRequired(requiresConfirmation)
    .build();

// Create the job and get the result.
CreateJobResponse result = s3ControlClient.createJob(request);
return result.jobId();
}

```

새 거버넌스 보존 작업을 만듭니다.

```

/**
 * 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 CreateGovernanceRetentionJob {

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

        Usage:
            <manifestObjectArn> <jobReportBucketArn> <roleArn> <accountId>
<manifestObjectVersionId>

        Where:
            manifestObjectArn - The Amazon Resource Name (ARN) of the S3
object that contains the manifest file for the governance objects.\s
            bucketName - The ARN of the S3 bucket where the job report will
be stored.
            roleArn - The ARN of the IAM role that will be used to perform
the governance retention operation.
            accountId - Your AWS account Id.
            manifestObjectVersionId = A unique value that is used as the
`eTag` property of the `JobManifestLocation` object.
        """;

        if (args.length != 4) {
            System.out.println(usage);
            return;
        }

        String manifestObjectArn = args[0];
        String jobReportBucketArn = args[1];
        String roleArn = args[2];
        String accountId = args[3];
        String manifestObjectVersionId = args[4];

        S3ControlClient s3ControlClient = S3ControlClient.create();
        createGovernanceRetentionJob(s3ControlClient, manifestObjectArn,
jobReportBucketArn, roleArn, accountId, manifestObjectVersionId);
    }

    public static String createGovernanceRetentionJob(final S3ControlClient
s3ControlClient, String manifestObjectArn, String jobReportBucketArn, String
```

```
roleArn, String accountId, String manifestObjectVersionId) throws ParseException
{
    final JobManifestLocation manifestLocation =
JobManifestLocation.builder()
    .objectArn(manifestObjectArn)
    .eTag(manifestObjectVersionId)
    .build();

    final JobManifestSpec manifestSpec = JobManifestSpec.builder()
    .format(JobManifestFormat.S3_BATCH_OPERATIONS_CSV_20180820)
    .fields(Arrays.asList(JobManifestFieldName.BUCKET,
JobManifestFieldName.KEY))
    .build();

    final JobManifest manifestToPublicApi = JobManifest.builder()
    .location(manifestLocation)
    .spec(manifestSpec)
    .build();

    final String jobReportPrefix = "reports/governance-objects";
    final JobReport jobReport = JobReport.builder()
    .enabled(true)
    .reportScope(JobReportScope.ALL_TASKS)
    .bucket(jobReportBucketArn)
    .prefix(jobReportPrefix)
    .format(JobReportFormat.REPORT_CSV_20180820)
    .build();

    final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
    final Date jan30th = format.parse("30/01/2025");

    final S3SetObjectRetentionOperation s3SetObjectRetentionOperation =
S3SetObjectRetentionOperation.builder()
    .retention(S3Retention.builder()
    .mode(S3ObjectLockRetentionMode.GOVERNANCE)
    .retainUntilDate(jan30th.toInstant())
    .build())
    .build();

    final JobOperation jobOperation = JobOperation.builder()
    .s3PutObjectRetention(s3SetObjectRetentionOperation)
    .build();

    final Boolean requiresConfirmation = true;
```



```

        final int priority = 10;

        final CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Put governance retention")
            .manifest(manifestToPublicApi)
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();

        final CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [CreateJob](#)을 참조하세요.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

def create_s3_batch_job(self, account_id: str, role_arn: str,
manifest_location: str,
                        report_bucket_name: str) -> str:
    """
    Create an S3 batch operation job.

    Args:
        account_id (str): AWS account ID
        role_arn (str): IAM role ARN for batch operations
        manifest_location (str): Location of the manifest file
        report_bucket_name (str): Bucket for job reports
    """

```

Returns:

str: Job ID

Raises:

ClientError: If job creation fails

"""

try:

bucket_name = manifest_location.split(':::')[1].split('/')[0]

manifest_key = 'job-manifest.csv'

manifest_obj = self.s3_client.head_object(

 Bucket=bucket_name,

 Key=manifest_key

)

etag = manifest_obj['ETag'].strip('\"')'

response = self.s3control_client.create_job(

 AccountId=account_id,

 Operation={

 'S3PutObjectTagging': {

 'TagSet': [

 {

 'Key': 'BatchTag',

 'Value': 'BatchValue'

 },

],

 }

 },

 Report={

 'Bucket': report_bucket_name,

 'Format': 'Report_CSV_20180820',

 'Enabled': True,

 'Prefix': 'batch-op-reports',

 'ReportScope': 'AllTasks'

 },

 Manifest={

 'Spec': {

 'Format': 'S3BatchOperations_CSV_20180820',

 'Fields': ['Bucket', 'Key']

 },

 'Location': {

 'ObjectArn': manifest_location,

 'ETag': etag

 }

```

        },
        Priority=10,
        RoleArn=role_arn,
        Description='Batch job for tagging objects',
        ConfirmationRequired=True
    )
    job_id = response['JobId']
    print(f"The Job id is {job_id}")
    return job_id
except ClientError as e:
    print(f"Error creating batch job: {e}")
    if 'Message' in str(e):
        print(f"Detailed error message: {e.response['Message']}")
    raise

```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 [CreateJob](#)을 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

TRY.

```

" iv_manifest_arn  = 'arn:aws:s3::my-bucket/job-manifest.csv'
" iv_manifest_etag = 'abc123def456'
" iv_report_bucket = 'arn:aws:s3::my-report-bucket'
DATA(lo_result) = lo_s3c->createjob(
    iv_accountid      = iv_account_id
    iv_rolearn        = iv_role_arn
    iv_confirmationrequired = abap_true
    iv_priority        = 10
    iv_description     = 'Batch job for tagging objects'
    io_operation       = NEW /aws1/cl_s3cjoboperation(
        io_s3putobjecttagging = NEW /aws1/cl_s3cs3setobjecttagop(
            it_tagset = VALUE /aws1/cl_s3cs3tag=>tt_s3tagset(
                ( NEW /aws1/cl_s3cs3tag(

```

```

        iv_key    = 'BatchTag'
        iv_value  = 'BatchValue' ) )
    )
)
)
io_manifest      = NEW /aws1/cl_s3cjobmanifest(
    io_spec       = NEW /aws1/cl_s3cjobmanifestspec(
        iv_format = 'S3BatchOperations_CSV_20180820'
        it_fields = VALUE /aws1/
cl_s3cjobmanifestfield00=>tt_jobmanifestfieldlist(
    ( NEW /aws1/cl_s3cjobmanifestfield00( 'Bucket' ) )
    ( NEW /aws1/cl_s3cjobmanifestfield00( 'Key' ) )
    )
)
io_location = NEW /aws1/cl_s3cjobmanifestloc(
    iv_objectarn = iv_manifest_arn
    iv_etag      = iv_manifest_etag
)
)
io_report      = NEW /aws1/cl_s3cjobreport(
    iv_bucket    = iv_report_bucket
    iv_format     = 'Report_CSV_20180820'
    iv_enabled    = abap_true
    iv_prefix     = 'batch-op-reports'
    iv_reportscope = 'AllTasks'
)
).
ov_job_id = lo_result->get_jobid( ).
MESSAGE |S3 Batch Operations job created: { ov_job_id }| TYPE 'I'.
CATCH /aws1/cx_s3cbadrequestex INTO DATA(lo_ex_bad).
MESSAGE lo_ex_bad->get_text( ) TYPE 'I'.
RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_bad.
CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_cli.
CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_srv.
ENDTRY.

```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 [CreateJob](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **DeleteJobTagging** 사용

다음 코드 예시는 DeleteJobTagging의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서 컨텍스트 내 이 작업을 확인할 수 있습니다.

- [기본 사항 알아보기](#)

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Asynchronously deletes the tags associated with a specific batch job.
 *
 * @param jobId      The ID of the batch job whose tags should be deleted.
 * @param accountId The ID of the account associated with the batch job.
 * @return A CompletableFuture that completes when the job tags have been
 * successfully deleted, or an exception is thrown if the deletion fails.
 */
public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {
    DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
        .accountId(accountId)
        .jobId(jobId)
        .build();
```

```

        return asyncClient.deleteJobTagging(jobTaggingRequest)
            .thenAccept(response -> {
                System.out.println("You have successfully deleted " + jobId + "
tagging.");
            })
            .exceptionally(ex -> {
                System.err.println("Failed to delete job tags: " +
ex.getMessage());
                throw new RuntimeException(ex);
            });
    }

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [DeleteJobTagging](#)을 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

def delete_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Delete all tags from a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        self.s3control_client.delete_job_tagging(
            AccountId=account_id,
            JobId=job_id
        )
        print(f"You have successfully deleted {job_id} tagging.")
    except ClientError as e:
        print(f"Error deleting job tags: {e}")
        raise

```

- API 세부 정보는 AWS SDK for Python(Boto3) API 참조의 [DeleteJobTagging](#)을 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.
    lo_s3c->deletejobtagging(
        iv_accountid = iv_account_id
        iv_jobid      = iv_job_id
    ).
    MESSAGE |Tags deleted from job { iv_job_id }| TYPE 'I'.
CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
    MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_nf.
CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
    MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_cli.
CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
    MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_srv.
ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 [DeleteJobTagging](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DescribeJob** 사용

다음 코드 예시는 DescribeJob의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [기본 사항 알아보기](#)

CLI

AWS CLI

Amazon S3 Batch Operations 작업 설명

다음 describe-job 예시에서는 지정된 Batch Operations 작업의 구성 파라미터와 상태를 제공합니다.

```
aws s3control describe-job \
  --account-id 123456789012 \
  --job-id 93735294-df46-44d5-8638-6356f335324e
```

출력:

```
{
  "Job": {
    "TerminationDate": "2019-10-03T21:49:53.944Z",
    "JobId": "93735294-df46-44d5-8638-6356f335324e",
    "FailureReasons": [],
    "Manifest": {
      "Spec": {
        "Fields": [
          "Bucket",
          "Key"
        ],
        "Format": "S3BatchOperations_CSV_20180820"
      },
      "Location": {
        "ETag": "69f52a4e9f797e987155d9c8f5880897",
        "ObjectArn": "arn:aws:s3:::employee-records-logs/inv-report/7a6a9be4-072c-407e-85a2-ec3e982f773e.csv"
      }
    }
  },
}
```



```

    "Operation": {
      "S3PutObjectTagging": {
        "TagSet": [
          {
            "Value": "true",
            "Key": "confidential"
          }
        ]
      }
    },
    "RoleArn": "arn:aws:iam::123456789012:role/S3BatchJobRole",
    "ProgressSummary": {
      "TotalNumberOfTasks": 8,
      "NumberOfTasksFailed": 0,
      "NumberOfTasksSucceeded": 8
    },
    "Priority": 42,
    "Report": {
      "ReportScope": "AllTasks",
      "Format": "Report_CSV_20180820",
      "Enabled": true,
      "Prefix": "batch-op-create-job",
      "Bucket": "arn:aws:s3:::employee-records-logs"
    },
    "JobArn": "arn:aws:s3:us-west-2:123456789012:job/93735294-
df46-44d5-8638-6356f335324e",
    "CreationTime": "2019-10-03T21:48:48.048Z",
    "Status": "Complete"
  }
}

```

- API 세부 정보는 AWS CLI 명령 참조의 [DescribeJob](#)을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Asynchronously describes the specified job.
 *
 * @param jobId      the ID of the job to describe
 * @param accountId the ID of the AWS account associated with the job
 * @return a {@link CompletableFuture} that completes when the job
description is available
 * @throws RuntimeException if an error occurs while describing the job
 */
public CompletableFuture<Void> describeJobAsync(String jobId, String
accountId) {
    DescribeJobRequest jobRequest = DescribeJobRequest.builder()
        .jobId(jobId)
        .accountId(accountId)
        .build();

    return getAsyncClient().describeJob(jobRequest)
        .thenAccept(response -> {
            System.out.println("Job ID: " + response.job().jobId());
            System.out.println("Description: " +
response.job().description());
            System.out.println("Status: " + response.job().statusAsString());
            System.out.println("Role ARN: " + response.job().roleArn());
            System.out.println("Priority: " + response.job().priority());
            System.out.println("Progress Summary: " +
response.job().progressSummary());

            // Print out details about the job manifest.
            JobManifest manifest = response.job().manifest();
            System.out.println("Manifest Location: " +
manifest.location().objectArn());
            System.out.println("Manifest ETag: " +
manifest.location().eTag());

            // Print out details about the job operation.
            JobOperation operation = response.job().operation();
            if (operation.s3PutObjectTagging() != null) {
                System.out.println("Operation: S3 Put Object Tagging");
                System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
            }

            // Print out details about the job report.
```

```

        JobReport report = response.job().report();
        System.out.println("Report Bucket: " + report.bucket());
        System.out.println("Report Prefix: " + report.prefix());
        System.out.println("Report Format: " + report.format());
        System.out.println("Report Enabled: " + report.enabled());
        System.out.println("Report Scope: " +
report.reportScopeAsString());
    })
    .exceptionally(ex -> {
        System.err.println("Failed to describe job: " + ex.getMessage());
        throw new RuntimeException(ex);
    });
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [DescribeJob](#)을 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

def describe_job_details(self, job_id: str, account_id: str) -> None:
    """
    Describe detailed information about a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        job = response['Job']
        print(f"Job ID: {job['JobId']}")
    
```

```

        print(f"Description: {job.get('Description', 'N/A')}")
        print(f"Status: {job['Status']}")
        print(f"Role ARN: {job['RoleArn']}")
        print(f"Priority: {job['Priority']}")
        if 'ProgressSummary' in job:
            progress = job['ProgressSummary']
            print(f"Progress Summary:
Total={progress.get('TotalNumberOfTasks', 0)}, "
                f"Succeeded={progress.get('NumberOfTasksSucceeded', 0)}, "
                f"Failed={progress.get('NumberOfTasksFailed', 0)}")
        except ClientError as e:
            print(f"Error describing job: {e}")
            raise

```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 [DescribeJob](#)을 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

TRY.
    oo_result = lo_s3c->describejob(          " oo_result is returned for
testing purposes.
    iv_accountid = iv_account_id
    iv_jobid      = iv_job_id
    ).
    DATA(lo_job) = oo_result->get_job( ).
    DATA(lv_status) = lo_job->get_status( ).
    DATA(lv_priority) = lo_job->get_priority( ).
    DATA(lo_progress) = lo_job->get_progresssummary( ).
    IF lo_progress IS NOT INITIAL.
        MESSAGE |Job { iv_job_id }: status={ lv_status },
priority={ lv_priority }, | &&
                |total={ lo_progress->get_totalnumberoftasks( ) }, | &&
                |succeeded={ lo_progress->get_numberoftaskssucceeded( ) }, | &&

```

```

|failed={ lo_progress->get_numberoftasksfailed( ) }| TYPE 'I'.
ELSE.
  MESSAGE |Job { iv_job_id }: status={ lv_status },
priority={ lv_priority }| TYPE 'I'.
ENDIF.
CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
  MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
  RAISE EXCEPTION TYPE /aws1/cx_rt_generic
  EXPORTING previous = lo_ex_nf.
CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
  MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
  RAISE EXCEPTION TYPE /aws1/cx_rt_generic
  EXPORTING previous = lo_ex_cli.
CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
  MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
  RAISE EXCEPTION TYPE /aws1/cx_rt_generic
  EXPORTING previous = lo_ex_srv.
ENDTRY.

```

- API 세부 정보는 AWS SDK for SAP ABAP API의 [DescribeJob](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **GetJobTagging** 사용

다음 코드 예시는 GetJobTagging의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서 컨텍스트 내 이 작업을 확인할 수 있습니다.

- [기본 사항 알아보기](#)

Java

SDK for Java 2.x

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Asynchronously retrieves the tags associated with a specific job in an AWS
 * account.
 *
 * @param jobId      the ID of the job for which to retrieve the tags
 * @param accountId the ID of the AWS account associated with the job
 * @return a {@link CompletableFuture} that completes when the job tags have
 *         been retrieved, or with an exception if the operation fails
 * @throws RuntimeException if an error occurs while retrieving the job tags
 */
public CompletableFuture<Void> getJobTagsAsync(String jobId, String
accountId) {
    GetJobTaggingRequest request = GetJobTaggingRequest.builder()
        .jobId(jobId)
        .accountId(accountId)
        .build();

    return asyncClient.getJobTagging(request)
        .thenAccept(response -> {
            List<S3Tag> tags = response.tags();
            if (tags.isEmpty()) {
                System.out.println("No tags found for job ID: " + jobId);
            } else {
                for (S3Tag tag : tags) {
                    System.out.println("Tag key is: " + tag.key());
                    System.out.println("Tag value is: " + tag.value());
                }
            }
        })
        .exceptionally(ex -> {
            System.err.println("Failed to get job tags: " + ex.getMessage());
            throw new RuntimeException(ex); // Propagate the exception
        });
}
```

```
    });  
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [GetJobTagging](#)을 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def get_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Get tags associated with a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.get_job_tagging(
            AccountId=account_id,
            JobId=job_id
        )
        tags = response.get('Tags', [])
        if tags:
            print(f"Tags for job {job_id}:")
            for tag in tags:
                print(f"  {tag['Key']}: {tag['Value']}")
        else:
            print(f"No tags found for job ID: {job_id}")
    except ClientError as e:
        print(f"Error getting job tags: {e}")
        raise
```

- API 세부 정보는 AWS SDK for Python(Boto3) API 참조의 [GetJobTagging](#)을 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.
    oo_result = lo_s3c->getjobtagging(          " oo_result is returned for
testing purposes.
    iv_accountid = iv_account_id
    iv_jobid      = iv_job_id
    ).
    DATA(lt_tags) = oo_result->get_tags( ).
    MESSAGE |Retrieved { lines( lt_tags ) } tag(s) for job { iv_job_id }|
TYPE 'I'.
CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
    MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_nf.
CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
    MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_cli.
CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
    MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_srv.
ENDTRY.
```

- API 세부 정보는 [AWS SDK for SAP ABAP API 참조](#)의 GetJobTagging을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **PutJobTagging** 사용

다음 코드 예시는 PutJobTagging의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서 컨텍스트 내 이 작업을 확인할 수 있습니다.

- [기본 사항 알아보기](#)

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Asynchronously adds tags to a job in the system.
 *
 * @param jobId      the ID of the job to add tags to
 * @param accountId the account ID associated with the job
 * @return a CompletableFuture that completes when the tagging operation is
 * finished
 */
public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
    S3Tag departmentTag = S3Tag.builder()
        .key("department")
        .value("Marketing")
        .build();

    S3Tag fiscalYearTag = S3Tag.builder()
        .key("FiscalYear")
        .value("2020")
```

```

        .build();

        PutJobTaggingRequest putJobTaggingRequest =
        PutJobTaggingRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .tags(departmentTag, fiscalYearTag)
            .build();

        return asyncClient.putJobTagging(putJobTaggingRequest)
            .thenRun(() -> {
                System.out.println("Additional Tags were added to job " + jobId);
            })
            .exceptionally(ex -> {
                System.err.println("Failed to add tags to job: " +
                ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [PutJobTagging](#)을 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

def put_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Add tags to a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:

```

```

        self.s3control_client.put_job_tagging(
            AccountId=account_id,
            JobId=job_id,
            Tags=[
                {'Key': 'Environment', 'Value': 'Development'},
                {'Key': 'Team', 'Value': 'DataProcessing'}
            ]
        )
        print(f"Additional tags were added to job {job_id}")
    except ClientError as e:
        print(f"Error adding job tags: {e}")
        raise

```

- API 세부 정보는 AWS SDK for Python(Boto3) API 참조의 [PutJobTagging](#)을 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

TRY.

```

lo_s3c->putjobtagging(
    iv_accountid = iv_account_id
    iv_jobid      = iv_job_id
    it_tags       = VALUE /aws1/cl_s3cs3tag=>tt_s3tagset(
        ( NEW /aws1/cl_s3cs3tag(
            iv_key   = 'Environment'
            iv_value = 'Development' ) )
        ( NEW /aws1/cl_s3cs3tag(
            iv_key   = 'Team'
            iv_value = 'DataProcessing' ) )
    )
).
MESSAGE |Tags added to job { iv_job_id }| TYPE 'I'.
CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.

```

```

        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_nf.
    CATCH /aws1/cx_s3ctoomanytagsex INTO DATA(lo_ex_tags).
    MESSAGE lo_ex_tags->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_tags.
    CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
    MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_cli.
    CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
    MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_srv.
ENDTRY.

```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 [PutJobTagging](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **UpdateJobPriority** 사용

다음 코드 예시는 UpdateJobPriority의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [기본 사항 알아보기](#)

CLI

AWS CLI

Amazon S3 Batch Operations의 작업 우선 순위 업데이트

다음 update-job-priority 예시에서는 지정된 작업을 새 우선 순위로 업데이트합니다.

```

aws s3control update-job-priority \
  --account-id 123456789012 \

```

```
--job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \
--priority 52
```

출력:

```
{
  "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386",
  "Priority": 52
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [UpdateJobPriority](#)를 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Updates the priority of a job asynchronously.
 *
 * @param jobId      the ID of the job to update
 * @param accountId the ID of the account associated with the job
 * @return a {@link CompletableFuture} that represents the asynchronous
 * operation, which completes when the job priority has been updated or an error
 * has occurred
 */
public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
    UpdateJobPriorityRequest priorityRequest =
UpdateJobPriorityRequest.builder()
        .accountId(accountId)
        .jobId(jobId)
        .priority(60)
        .build();

    CompletableFuture<Void> future = new CompletableFuture<>();
```

```

        getAsyncClient().updateJobPriority(priorityRequest)
            .thenAccept(response -> {
                System.out.println("The job priority was updated");
                future.complete(null); // Complete the CompletableFuture on
successful execution
            })
            .exceptionally(ex -> {
                System.err.println("Failed to update job priority: " +
ex.getMessage());
                future.completeExceptionally(ex); // Complete the
CompletableFuture exceptionally on error
                return null; // Return null to handle the exception
            });

        return future;
    }

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [UpdateJobPriority](#)를 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

def update_job_priority(self, job_id: str, account_id: str) -> None:
    """
    Update the priority of a batch job and start it.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id

```

```

    )
    current_status = response['Job']['Status']
    print(f"Current job status: {current_status}")

    if current_status in ['Ready', 'Suspended']:
        self.s3control_client.update_job_priority(
            AccountId=account_id,
            JobId=job_id,
            Priority=60
        )
        print("The job priority was updated")

    try:
        self.s3control_client.update_job_status(
            AccountId=account_id,
            JobId=job_id,
            RequestedJobStatus='Ready'
        )
        print("Job activated successfully")
    except ClientError as activation_error:
        print(f"Note: Could not activate job automatically: {activation_error}")
        print("Job priority was updated successfully. Job may need manual activation in the console.")
        elif current_status in ['Active', 'Completing', 'Complete']:
            print(f"Job is in '{current_status}' state - priority cannot be updated")

            if current_status == 'Completing':
                print("Job is finishing up and will complete soon.")
            elif current_status == 'Complete':
                print("Job has already completed successfully.")
            else:
                print("Job is currently running.")
        else:
            print(f"Job is in '{current_status}' state - priority update not allowed")

    except ClientError as e:
        print(f"Error updating job priority: {e}")
        print("Continuing with the scenario...")
    return

```

- API 세부 정보는 AWS SDK for Python(Boto3) API 참조의 [UpdateJobPriority](#)를 참조하세요.

SAP ABAP

SDK for SAP ABAP

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.
    oo_result = lo_s3c->updatejobpriority(    " oo_result is returned for
testing purposes.
    iv_accountid = iv_account_id
    iv_jobid      = iv_job_id
    iv_priority   = 60
    ).
    MESSAGE |Job { oo_result->get_jobid( ) } priority updated to { oo_result-
>get_priority( ) }| TYPE 'I'.
    CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
    MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_nf.
    CATCH /aws1/cx_s3cbadrequestex INTO DATA(lo_ex_bad).
    MESSAGE lo_ex_bad->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_bad.
    CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
    MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_cli.
    CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
    MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_srv.
ENDTRY.
```

- API에 대한 세부 정보는 AWS SDK for SAP ABAP API 참조의 [UpdateJobPriority](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **UpdateJobStatus** 사용

다음 코드 예시는 UpdateJobStatus의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [기본 사항 알아보기](#)

CLI

AWS CLI

Amazon S3 Batch Operations의 작업 상태 업데이트

다음 update-job-status 예시에서는 승인을 기다리는 지정된 작업을 취소합니다.

```
aws s3control update-job-status \
  --account-id 123456789012 \
  --job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \
  --requested-job-status Cancelled
```

출력:

```
{
  "Status": "Cancelled",
  "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386"
}
```

다음 update-job-status 예시에서는 승인을 기다리는 지정된 작업을 확인하고 실행합니다.

```
aws s3control update-job-status \
  --account-id 123456789012 \
  --job-id 5782949f-3301-4fb3-be34-8d5bab54dbca \
  --requested-job-status Ready
```

Output::

```
{
  "Status": "Ready",
  "JobId": "5782949f-3301-4fb3-be34-8d5bab54dbca"
}
```

다음 update-job-status 예시에서는 실행 중인 지정된 작업을 취소합니다.

```
aws s3control update-job-status \
  --account-id 123456789012 \
  --job-id 5782949f-3301-4fb3-be34-8d5bab54dbca \
  --requested-job-status Cancelled

Output:
{
  "Status": "Cancelling",
  "JobId": "5782949f-3301-4fb3-be34-8d5bab54dbca"
}
```

- API 세부 정보는 AWS CLI 명령 참조의 [UpdateJobStatus](#)를 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Cancels a job asynchronously.
 *
 * @param jobId The ID of the job to be canceled.
 * @param accountId The ID of the account associated with the job.
 * @return A {@link CompletableFuture} that completes when the job status has
 *         been updated to "CANCELLED".
 *         If an error occurs during the update, the returned future will
 *         complete exceptionally.
 */
```

```

    public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId)
    {
        UpdateJobStatusRequest updateJobStatusRequest =
        UpdateJobStatusRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))
            .build();

        return asyncClient.updateJobStatus(updateJobStatusRequest)
            .thenAccept(updateJobStatusResponse -> {
                System.out.println("Job status updated to: " +
                updateJobStatusResponse.status());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to cancel job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [UpdateJobStatus](#)를 참조하세요.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

def cancel_job(self, job_id: str, account_id: str) -> None:
    """
    Cancel an S3 batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:

```

```

        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        current_status = response['Job']['Status']
        print(f"Current job status: {current_status}")

        if current_status in ['Ready', 'Suspended', 'Active']:
            self.s3control_client.update_job_status(
                AccountId=account_id,
                JobId=job_id,
                RequestedJobStatus='Cancelled'
            )
            print(f"Job {job_id} was successfully canceled.")
        elif current_status in ['Completing', 'Complete']:
            print(f"Job is in '{current_status}' state - cannot be
cancelled")

            if current_status == 'Completing':
                print("Job is finishing up and will complete soon.")
            elif current_status == 'Complete':
                print("Job has already completed successfully.")
        else:
            print(f"Job is in '{current_status}' state - cancel not allowed")
    except ClientError as e:
        print(f"Error canceling job: {e}")
        raise

```

- API 세부 정보는 AWS SDK for Python(Boto3) API 참조의 [UpdateJobStatus](#)를 참조하세요.

SAP ABAP

SDK for SAP ABAP

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

TRY.

```
" iv_requested_status = 'Cancelled'
```

```

        oo_result = lo_s3c->updatejobstatus(      " oo_result is returned for
testing purposes.
        iv_accountid          = iv_account_id
        iv_jobid              = iv_job_id
        iv_requestedjobstatus = iv_requested_status
    ).
    MESSAGE |Job { oo_result->get_jobid( ) } status updated to { oo_result-
>get_status( ) }| TYPE 'I'.
    CATCH /aws1/cx_s3cjobstatusexception INTO DATA(lo_ex_js).
    MESSAGE lo_ex_js->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_js.
    CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
    MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_nf.
    CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
    MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_cli.
    CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
    MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_srv.
ENDTRY.

```

- API 세부 정보는 [AWS SDK for SAP ABAP API 참조](#)의 UpdateJobStatus를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용한 S3 디렉터리 버킷 코드 예제

다음 코드 예제에서는 S3 디렉터리 버킷을 AWS 소프트웨어 개발 키트(SDK)와 함께 사용하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적 호출하여 특정 작업을 수행하는 방법을 보여주는 코드 예제입니다.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

S3 디렉터리 버킷

- [AWS SDK를 사용한 S3 디렉터리 버킷 기본 예제](#)
 - [Amazon S3 디렉터리 버킷 시작](#)
 - [AWS SDK를 사용하여 S3 디렉터리 버킷의 기본 기능 알아보기](#)
 - [AWS SDK를 사용한 S3 디렉터리 버킷 작업](#)
 - [AWS SDK와 함께 AbortMultipartUpload 사용](#)
 - [AWS SDK와 함께 CompleteMultipartUpload 사용](#)
 - [AWS SDK와 함께 CopyObject 사용](#)
 - [AWS SDK와 함께 CreateBucket 사용](#)
 - [AWS SDK와 함께 CreateMultipartUpload 사용](#)
 - [AWS SDK와 함께 CreateSession 사용](#)
 - [AWS SDK와 함께 DeleteBucket 사용](#)
 - [AWS SDK와 함께 DeleteBucketEncryption 사용](#)
 - [AWS SDK와 함께 DeleteBucketPolicy 사용](#)
 - [AWS SDK와 함께 DeleteObject 사용](#)
 - [AWS SDK와 함께 DeleteObjects 사용](#)
 - [AWS SDK와 함께 GetBucketEncryption 사용](#)
 - [AWS SDK와 함께 GetBucketPolicy 사용](#)
 - [AWS SDK와 함께 GetObject 사용](#)
 - [AWS SDK와 함께 GetObjectAttributes 사용](#)
 - [AWS SDK와 함께 HeadBucket 사용](#)
 - [AWS SDK와 함께 HeadObject 사용](#)
 - [AWS SDK와 함께 ListDirectoryBuckets 사용](#)
 - [AWS SDK와 함께 ListMultipartUploads 사용](#)
 - [AWS SDK와 함께 ListObjectsV2 사용](#)

- [AWS SDK와 함께 ListParts 사용](#)
- [AWS SDK와 함께 PutBucketEncryption 사용](#)
- [AWS SDK와 함께 PutBucketPolicy 사용](#)
- [AWS SDK와 함께 PutObject 사용](#)
- [AWS SDK와 함께 UploadPart 사용](#)
- [AWS SDK와 함께 UploadPartCopy 사용](#)
- [AWS SDK를 사용한 S3 디렉터리 버킷 시나리오](#)
 - [AWS SDK를 사용하여 객체를 가져오기 위해 Amazon S3 디렉터리 버킷에 대해 미리 서명된 URL 생성](#)

AWS SDK를 사용한 S3 디렉터리 버킷 기본 예제

다음 코드 예제에서는 AWS SDK를 통해 Amazon S3 디렉터리 버킷의 기본 기능을 사용하는 방법을 보여줍니다.

예제

- [Amazon S3 디렉터리 버킷 시작](#)
- [AWS SDK를 사용하여 S3 디렉터리 버킷의 기본 기능 알아보기](#)
- [AWS SDK를 사용한 S3 디렉터리 버킷 작업](#)
 - [AWS SDK와 함께 AbortMultipartUpload 사용](#)
 - [AWS SDK와 함께 CompleteMultipartUpload 사용](#)
 - [AWS SDK와 함께 CopyObject 사용](#)
 - [AWS SDK와 함께 CreateBucket 사용](#)
 - [AWS SDK와 함께 CreateMultipartUpload 사용](#)
 - [AWS SDK와 함께 CreateSession 사용](#)
 - [AWS SDK와 함께 DeleteBucket 사용](#)
 - [AWS SDK와 함께 DeleteBucketEncryption 사용](#)
 - [AWS SDK와 함께 DeleteBucketPolicy 사용](#)
 - [AWS SDK와 함께 DeleteObject 사용](#)
 - [AWS SDK와 함께 DeleteObjects 사용](#)
 - [AWS SDK와 함께 GetBucketEncryption 사용](#)
 - [AWS SDK와 함께 GetBucketPolicy 사용](#)

- [AWS SDK와 함께 GetObject 사용](#)
- [AWS SDK와 함께 GetObjectAttributes 사용](#)
- [AWS SDK와 함께 HeadBucket 사용](#)
- [AWS SDK와 함께 HeadObject 사용](#)
- [AWS SDK와 함께 ListDirectoryBuckets 사용](#)
- [AWS SDK와 함께 ListMultipartUploads 사용](#)
- [AWS SDK와 함께 ListObjectsV2 사용](#)
- [AWS SDK와 함께 ListParts 사용](#)
- [AWS SDK와 함께 PutBucketEncryption 사용](#)
- [AWS SDK와 함께 PutBucketPolicy 사용](#)
- [AWS SDK와 함께 PutObject 사용](#)
- [AWS SDK와 함께 UploadPart 사용](#)
- [AWS SDK와 함께 UploadPartCopy 사용](#)

Amazon S3 디렉터리 버킷 시작

다음 코드 예제에서는 Amazon S3 디렉터리 버킷 사용을 시작하는 방법을 보여줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
package com.example.s3.directorybucket;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
```



```
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.List;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

/**
 * Before running this example:
 * <p>
 * The SDK must be able to authenticate AWS requests on your behalf. If you have
 * not configured
 * authentication for SDKs and tools, see
 * https://docs.aws.amazon.com/sdkref/latest/guide/access.html in the AWS SDKs
 * and Tools Reference Guide.
 * <p>
 * You must have a runtime environment configured with the Java SDK.
 * See
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html in
 * the Developer Guide if this is not set up.
 * <p>
 * To use S3 directory buckets, configure a gateway VPC endpoint. This is the
 * recommended method to enable directory bucket traffic without
 * requiring an internet gateway or NAT device. For more information on
 * configuring VPC gateway endpoints, visit
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-
 \* networking.html#s3-express-networking-vpc-gateway.
 * <p>
 * Directory buckets are available in specific AWS Regions and Zones. For
 * details on Regions and Zones supporting directory buckets, see
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-
 \* networking.html#s3-express-endpoints.
```

```
*/

public class HelloS3DirectoryBuckets {
    private static final Logger logger =
        LoggerFactory.getLogger(HelloS3DirectoryBuckets.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.
        Region region = Region.US_WEST_2;
        String zone = "usw2-az1";
        S3Client s3Client = createS3Client(region);

        try {
            // Create the directory bucket
            createDirectoryBucket(s3Client, bucketName, zone);
            logger.info("Created bucket: {}", bucketName);

            // List all directory buckets
            List<String> bucketNames = listDirectoryBuckets(s3Client);
            bucketNames.forEach(name -> logger.info("Bucket Name: {}", name));
        } catch (S3Exception e) {
            logger.error("An error occurred during S3 operations: {} - Error
            code: {}",
                e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        } finally {
            try {
                // Delete the created bucket
                deleteDirectoryBucket(s3Client, bucketName);
                logger.info("Deleted bucket: {}", bucketName);
            } catch (S3Exception e) {
                logger.error("Failed to delete the bucket due to S3 error: {} -
                Error code: {}",
                    e.awsErrorDetails().errorMessage(),
                e.awsErrorDetails().errorCode(), e);
            } catch (RuntimeException e) {
                logger.error("Failed to delete the bucket due to unexpected
                error: {}", e.getMessage(), e);
            } finally {
                s3Client.close();
            }
        }
    }
}
```

```

/**
 * Creates a new S3 directory bucket in a specified Zone (For example, a
 * specified Availability Zone in this code example).
 *
 * @param s3Client    The S3 client used to create the bucket
 * @param bucketName  The name of the bucket to be created
 * @param zone        The region where the bucket will be created
 * @throws S3Exception if there's an error creating the bucket
 */
public static void createDirectoryBucket(S3Client s3Client, String
bucketName, String zone) throws S3Exception {
    logger.info("Creating bucket: {}", bucketName);

    CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
        .location(LocationInfo.builder()
            .type(LocationType.AVAILABILITY_ZONE)
            .name(zone).build())
        .bucket(BucketInfo.builder()
            .type(BucketType.DIRECTORY)
            .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
            .build())
        .build();

    try {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .createBucketConfiguration(bucketConfiguration).build();
        CreateBucketResponse response = s3Client.createBucket(bucketRequest);
        logger.info("Bucket created successfully with location: {}",
response.location());
    } catch (S3Exception e) {
        logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}

/**
 * Lists all S3 directory buckets.
 *
 * @param s3Client The S3 client used to interact with S3
 * @return A list of bucket names

```

```
    */
    public static List<String> listDirectoryBuckets(S3Client s3Client) {
        logger.info("Listing all directory buckets");

        try {
            // Create a ListBucketsRequest
            ListDirectoryBucketsRequest listBucketsRequest =
ListDirectoryBucketsRequest.builder().build();

            // Retrieve the list of buckets
            ListDirectoryBucketsResponse response =
s3Client.listDirectoryBuckets(listBucketsRequest);

            // Extract bucket names
            List<String> bucketNames = response.buckets().stream()
                .map(Bucket::name)
                .collect(Collectors.toList());

            return bucketNames;
        } catch (S3Exception e) {
            logger.error("Failed to list buckets: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }

    /**
     * Deletes the specified S3 directory bucket.
     *
     * @param s3Client The S3 client used to interact with S3
     * @param bucketName The name of the bucket to delete
     */
    public static void deleteDirectoryBucket(S3Client s3Client, String
bucketName) {
        try {
            DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();
            s3Client.deleteBucket(deleteBucketRequest);
        } catch (S3Exception e) {
            logger.error("Failed to delete bucket: " + bucketName + " - Error
code: " + e.awsErrorDetails().errorCode(),
```

```
        e);  
        throw e;  
    }  
}  
  
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
 - [CreateBucket](#)
 - [ListDirectoryBuckets](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 S3 디렉터리 버킷의 기본 기능 알아보기

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- VPC 및 VPC 엔드포인트를 설정합니다.
- S3 디렉터리 버킷 및 S3 Express One Zone 스토리지 클래스로 작업하도록 정책, 역할 및 사용자를 설정합니다.
- 두 개의 S3 클라이언트를 만듭니다.
- 두 개의 버킷 만들기
- 객체를 만들고 복사합니다.
- 성능 차이를 보여줍니다.
- 버킷을 채워 사전식 순서 차이를 표시합니다.
- 사용자에게 리소스를 정리할지 묻는 프롬프트를 표시합니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 기능을 보여주는 대화형 시나리오를 실행합니다.

```
public class S3DirectoriesScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
    "-");

    private static final Logger logger =
    LoggerFactory.getLogger(S3DirectoriesScenario.class);
    static Scanner scanner = new Scanner(System.in);

    private static S3AsyncClient mS3RegularClient;
    private static S3AsyncClient mS3ExpressClient;

    private static String mdirectoryBucketName;
    private static String mregularBucketName;

    private static String stackName = "cfn-stack-s3-express-basics--" +
    UUID.randomUUID();

    private static String regularUser = "";
    private static String vpcId = "";
    private static String expressUser = "";

    private static String vpcEndpointId = "";

    private static final S3DirectoriesActions s3DirectoriesActions = new
    S3DirectoriesActions();

    public static void main(String[] args) {
        try {
            s3ExpressScenario();
        } catch (RuntimeException e) {
```

```
        logger.info(e.getMessage());
    }
}

// Runs the scenario.
private static void s3ExpressScenario() {
    logger.info(DASHES);
    logger.info("Welcome to the Amazon S3 Express Basics demo using AWS SDK
for Java V2.");
    logger.info("""
        Let's get started! First, please note that S3 Express One Zone works
        best when working within the AWS infrastructure,
        specifically when working in the same Availability Zone (AZ). To see
        the best results in this example and when you implement
        directory buckets into your infrastructure, it is best to put your
        compute resources in the same AZ as your directory
        bucket.
        """);
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    // Create an optional VPC and create 2 IAM users.
    UserNames userNames = createVpcUsers();
    String expressUserName = userNames.getExpressUserName();
    String regularUserName = userNames.getRegularUserName();

    // Set up two S3 clients, one regular and one express,
    // and two buckets, one regular and one directory.
    setupClientsAndBuckets(expressUserName, regularUserName);

    // Create an S3 session for the express S3 client and add objects to the
    buckets.
    logger.info("Now let's add some objects to our buckets and demonstrate
how to work with S3 Sessions.");
    waitForInputToContinue(scanner);
    String bucketObject = createSessionAddObjects();

    // Demonstrate performance differences between regular and directory
    buckets.
    demonstratePerformance(bucketObject);

    // Populate the buckets to show the lexicographical difference between
    // regular and express buckets.
    showLexicographicalDifferences(bucketObject);
}
```

```

        logger.info(DASHES);
        logger.info("That's it for our tour of the basic operations for S3
Express One Zone.");
        logger.info("Would you like to cleanUp the AWS resources? (y/n): ");
        String response = scanner.next().trim().toLowerCase();
        if (response.equals("y")) {
            cleanUp(stackName);
        }
    }

    /**
     Delete resources created by this scenario.
    */
    public static void cleanUp(String stackName) {
        try {
            if (mdirectoryBucketName != null) {

s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
            }
            logger.info("Deleted directory bucket " + mdirectoryBucketName);
            mdirectoryBucketName = null;
            if (mregularBucketName != null) {

s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3RegularClient,
mregularBucketName).join();
            }
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof S3Exception) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
        }

        logger.info("Deleted regular bucket " + mregularBucketName);
        mregularBucketName = null;
        CloudFormationHelper.destroyCloudFormationStack(stackName);
    }

    private static void showLexicographicalDifferences(String bucketObject) {

```



```

        logger.info(DASHES);
        logger.info("""
            7. Populate the buckets to show the lexicographical (alphabetical)
difference
            when object names are listed. Now let's explore how directory buckets
store
            objects in a different manner to regular buckets. The key is in the
name
            "Directory". Where regular buckets store their key/value pairs in a
            flat manner, directory buckets use actual directories/folders.
            This allows for more rapid indexing, traversing, and therefore
            retrieval times!

            The more segmented your bucket is, with lots of
            directories, sub-directories, and objects, the more efficient it
            becomes.

            This structural difference also causes `ListObject` operations to
            behave
            differently, which can cause unexpected results. Let's add a few
            more
            objects in sub-directories to see how the output of
            ListObjects changes.
            """);

        waitForInputToContinue(scanner);

        // Populate a few more files in each bucket so that we can use
        // ListObjects and show the difference.
        String otherObject = "other/" + bucketObject;
        String altObject = "alt/" + bucketObject;
        String otherAltObject = "other/alt/" + bucketObject;

        try {
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, altObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, altObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherAltObject, "").join();

```

```

        s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherAltObject, "").join();

    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof NoSuchBucketException) {
            logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }

    try {
        // List objects in both S3 buckets.
        List<String> dirBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
        List<String> regBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3RegularClient,
mregularBucketName).join();

        logger.info("Directory bucket content");
        for (String obj : dirBucketObjects) {
            logger.info(obj);
        }

        logger.info("Regular bucket content");
        for (String obj : regBucketObjects) {
            logger.info(obj);
        }
    } catch (CompletionException e) {
        logger.error("Async operation failed: {} ",
e.getCause().getMessage());
        return;
    }

    logger.info("""
        Notice how the regular bucket lists objects in lexicographical order,
while the directory bucket does not. This is
        because the regular bucket considers the whole "key" to be the object
identifier, while the directory bucket actually

```

```

        creates directories and uses the object "key" as a path to the
object.
        """);
        waitForInputToContinue(scanner);
    }

    /**
     * Demonstrates the performance difference between downloading an object from
a directory bucket and a regular bucket.
     *
     * <p>This method:
     * <ul>
     *     <li>Prompts the user to choose the number of downloads (default is
1,000).</li>
     *     <li>Downloads the specified object from the directory bucket and
measures the total time.</li>
     *     <li>Downloads the same object from the regular bucket and measures the
total time.</li>
     *     <li>Compares the time differences and prints the results.</li>
     * </ul>
     *
     * <p>Note: The performance difference will be more pronounced if this
example is run on an EC2 instance
     * in the same Availability Zone as the buckets.
     *
     * @param bucketObject the name of the object to download
     */
    private static void demonstratePerformance(String bucketObject) {
        logger.info(DASHES);
        logger.info("6. Demonstrate the performance difference.");
        logger.info("
            Now, let's do a performance test. We'll download the same object from
each
            bucket repeatedly and compare the total time needed.

            Note: the performance difference will be much more pronounced if this
example is run in an EC2 instance in the same Availability Zone as
the bucket.
            """);
        waitForInputToContinue(scanner);

        int downloads = 1000; // Default value.
        logger.info("The default number of downloads of the same object for this
example is set at " + downloads + ".");
    }

```

```
// Ask if the user wants to download a different number.
logger.info("Would you like to download the file a different number of
times? (y/n): ");
String response = scanner.next().trim().toLowerCase();
if (response.equals("y")) {
    int maxDownloads = 1_000_000;

    // Ask for a valid number of downloads.
    while (true) {
        logger.info("Enter a number between 1 and " + maxDownloads + "
for the number of downloads: ");
        if (scanner.hasNextInt()) {
            downloads = scanner.nextInt();
            if (downloads >= 1 && downloads <= maxDownloads) {
                break;
            } else {
                logger.info("Please enter a number between 1 and " +
maxDownloads + ".");
            }
        } else {
            logger.info("Invalid input. Please enter a valid integer.");
            scanner.next();
        }
    }

    logger.info("You have chosen to download {} items.", downloads);
} else {
    logger.info("No changes made. Using default downloads: {}",
downloads);
}
// Simulating the download process for the directory bucket.
logger.info("Downloading from the directory bucket.");
long directoryTimeStart = System.nanoTime();
for (int index = 0; index < downloads; index++) {
    if (index % 50 == 0) {
        logger.info("Download " + index + " of " + downloads);
    }

    try {
        // Get the object from the directory bucket.
        s3DirectoriesActions.getObjectAsync(mS3ExpressClient,
mdirectoryBucketName, bucketObject).join();
    } catch (CompletionException ce) {
```

```

        Throwable cause = ce.getCause();
        if (cause instanceof NoSuchKeyException) {
            logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }
}
long directoryTimeDifference = System.nanoTime() - directoryTimeStart;

// Download from the regular bucket.
logger.info("Downloading from the regular bucket.");
long normalTimeStart = System.nanoTime();
for (int index = 0; index < downloads; index++) {
    if (index % 50 == 0) {
        logger.info("Download " + index + " of " + downloads);
    }

    try {
        s3DirectoriesActions.getObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject).join();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof NoSuchKeyException) {
            logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }
}

long normalTimeDifference = System.nanoTime() - normalTimeStart;
logger.info("The directory bucket took " + directoryTimeDifference
+ " nanoseconds, while the regular bucket took " + normalTimeDifference + "
nanoseconds.");
long difference = normalTimeDifference - directoryTimeDifference;
logger.info("That's a difference of " + difference + " nanoseconds, or");
logger.info(difference / 1_000_000_000.0 + " seconds.");

```

```

        if (difference < 0) {
            logger.info("The directory buckets were slower. This can happen if
you are not running on the cloud within a VPC.");
        }
        waitForInputToContinue(scanner);
    }

    private static String createSessionAdd0Objects() {
        logger.info(DASHES);
        logger.info("""
            5. Create an object and copy it.
            We'll create an object consisting of some text and upload it to the
            regular bucket.
            """);
        waitForInputToContinue(scanner);

        String bucketObject = "basic-text-object.txt";
        try {
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject, "Look Ma, I'm a bucket!").join();
            s3DirectoriesActions.createSessionAsync(mS3ExpressClient,
mdirectoryBucketName).join();

            // Copy the object to the destination S3 bucket.
            s3DirectoriesActions.copyObjectAsync(mS3ExpressClient,
mregularBucketName, bucketObject, mdirectoryBucketName, bucketObject).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof S3Exception) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
        }
        logger.info("""
            It worked! This is because the S3Client that performed the copy
operation
            is the expressClient using the credentials for the user with
permission to
            work with directory buckets.
        """);
    }

```

```

        It's important to remember the user permissions when interacting
with
        directory buckets. Instead of validating permissions on every call
as
        regular buckets do, directory buckets utilize the user credentials
and session
        token to validate. This allows for much faster connection speeds on
every call.
        For single calls, this is low, but for many concurrent calls
        this adds up to a lot of time saved.
        """);
        waitForInputToContinue(scanner);
        return bucketObject;
    }

    /**
     * Creates VPC users for the S3 Express One Zone scenario.
     * <p>
     * This method performs the following steps:
     * <ol>
     *     <li>Optionally creates a new VPC and VPC Endpoint if the application
is running in an EC2 instance in the same Availability Zone as the directory
buckets.</li>
     *     <li>Creates two IAM users: one with S3 Express One Zone permissions
and one without.</li>
     * </ol>
     *
     * @return a {@link UserNames} object containing the names of the created IAM
users
     */
    public static UserNames createVpcUsers() {
        /*
         * Optionally create a VPC.
         * Create two IAM users, one with S3 Express One Zone permissions and one
without.
         */
        logger.info(DASHES);
        logger.info("""
            1. First, we'll set up a new VPC and VPC Endpoint if this program is
running in an EC2 instance in the same AZ as your\s
            directory buckets will be. Are you running this in an EC2 instance
located in the same AZ as your intended directory buckets?
            """);
    }

```

```

        logger.info("Do you want to setup a VPC Endpoint? (y/n)");
        String endpointAns = scanner.nextLine().trim();
        if (endpointAns.equalsIgnoreCase("y")) {
            logger.info("
                Great! Let's set up a VPC, retrieve the Route Table from it, and
create a VPC Endpoint to connect the S3 Client to.
                ");
            try {
                s3DirectoriesActions.setupVPCAsync().join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof Ec2Exception) {
                    logger.error("IamException occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
            }
            waitForInputToContinue(scanner);
        } else {
            logger.info("Skipping the VPC setup. Don't forget to use this in
production!");
        }
        logger.info(DASHES);
        logger.info("
            2. Create a RegularUser and ExpressUser by using the AWS CDK.
            One IAM User, named RegularUser, will have permissions to work only
with regular buckets and one IAM user, named ExpressUser, will have
permissions to work only with directory buckets.
            ");
        waitForInputToContinue(scanner);

        // Create two users required for this scenario.
        Map<String, String> stackOutputs = createUsersUsingCDK(stackName);
        regularUser = stackOutputs.get("RegularUser");
        expressUser = stackOutputs.get("ExpressUser");

        UserNames names = new UserNames();
        names.setRegularUserName(regularUser);
        names.setExpressUserName(expressUser);
        return names;
    }

```



```

/**
 * Creates users using AWS CloudFormation.
 *
 * @return a {@link Map} of String keys and String values representing the
stack outputs,
 * which may include user-related information such as user names and IDs.
 */
public static Map<String, String> createUsersUsingCDK(String stackName) {
    logger.info("We'll use an AWS CloudFormation template to create the IAM
users and policies.");
    CloudFormationHelper.deployCloudFormationStack(stackName);
    return CloudFormationHelper.getStackOutputsAsync(stackName).join();
}

/**
 * Sets up the necessary clients and buckets for the S3 Express service.
 *
 * @param expressUserName the username for the user with S3 Express
permissions
 * @param regularUserName the username for the user with regular S3
permissions
 */
public static void setupClientsAndBuckets(String expressUserName, String
regularUserName) {
    Scanner locscanner = new Scanner(System.in);
    String accessKeyIdforRegUser;
    String secretAccessforRegUser;
    try {
        CreateAccessKeyResponse keyResponse =
s3DirectoriesActions.createAccessKeyAsync(regularUserName).join();
        accessKeyIdforRegUser = keyResponse.accessKey().accessKeyId();
        secretAccessforRegUser = keyResponse.accessKey().secretAccessKey();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof IamException) {
            logger.error("IamException occurred: {}", cause.getMessage(),
ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
    }
    return;
}

```

```

        String accessKeyIdforExpressUser;
        String secretAccessforExpressUser;
        try {
            CreateAccessKeyResponse keyResponseExpress =
s3DirectoriesActions.createAccessKeyAsync(expressUserName).join();
            accessKeyIdforExpressUser =
keyResponseExpress.accessKey().accessKeyId();
            secretAccessforExpressUser =
keyResponseExpress.accessKey().secretAccessKey();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IamException) {
                logger.error("IamException occurred: {}", cause.getMessage(),
ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
            return;
        }

        logger.info(DASHES);
        logger.info("""
            3. Create two S3Clients; one uses the ExpressUser's credentials and
one uses the RegularUser's credentials.
            The 2 S3Clients will use different credentials.
            """);
        waitForInputToContinue(locscanner);
        try {
            mS3RegularClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforRegUser,
secretAccessforRegUser).join();
            mS3ExpressClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforExpressUser,
secretAccessforExpressUser).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IllegalArgumentException) {
                logger.error("An invalid argument exception occurred: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
        }
    }

```

```

        return;
    }

    logger.info("""
        We can now use the ExpressUser client to make calls to S3 Express
        operations.
        """);
    waitForInputToContinue(locscanner);
    logger.info(DASHES);
    logger.info("""
        4. Create two buckets.
        Now we will create a directory bucket which is the linchpin of the S3
        Express One Zone service. Directory buckets
        behave differently from regular S3 buckets which we will explore
        here. We'll also create a regular bucket, put
        an object into the regular bucket, and copy it to the directory
        bucket.
        """);

    logger.info("""
        Now, let's choose an availability zone (AZ) for the directory
        bucket.
        We'll choose one that is supported.
        """);
    String zoneId;
    String regularBucketName;
    try {
        zoneId = s3DirectoriesActions.selectAvailabilityZoneIdAsync().join();
        regularBucketName = "amzn-s3-demo-bucket"; // Replace with your
        bucket name.
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof Ec2Exception) {
            logger.error("EC2Exception occurred: {}", cause.getMessage(),
            ce);
        } else {
            logger.error("An unexpected error occurred: {}",
            cause.getMessage(), ce);
        }
        return;
    }
    logger.info("""
        Now, let's create the actual directory bucket, as well as a regular
        bucket."

```

```

        "");

        String directoryBucketName = "amzn-s3-demo-bucket--" + zoneId + "--x-
s3"; // Replace with your directory bucket name.
        try {
            s3DirectoriesActions.createDirectoryBucketAsync(mS3ExpressClient,
directoryBucketName, zoneId).join();
            logger.info("Created directory bucket {}", directoryBucketName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                return;
            }
        }

        // Assign to the data member.
        mdirectoryBucketName = directoryBucketName;
        try {
            s3DirectoriesActions.createBucketAsync(mS3RegularClient,
regularBucketName).join();
            logger.info("Created regular bucket {} ", regularBucketName);
            mregularBucketName = regularBucketName;
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                return;
            }
        }
        logger.info("Great! Both buckets were created.");
        waitForInputToContinue(locscanner);
    }

    /**

```

```

    * Creates an asynchronous S3 client with the specified access key and secret
    access key.
    *
    * @param accessKeyId      the AWS access key ID
    * @param secretAccessKey the AWS secret access key
    * @return a {@link CompletableFuture} that asynchronously creates the S3
    client
    * @throws IllegalArgumentException if the access key ID or secret access key
    is null
    */
    public static CompletableFuture<S3AsyncClient>
    createS3ClientWithAccessKeyAsync(String accessKeyId, String secretAccessKey) {
        return CompletableFuture.supplyAsync(() -> {
            // Validate input parameters
            if (accessKeyId == null || accessKeyId.isBlank() || secretAccessKey
            == null || secretAccessKey.isBlank()) {
                throw new IllegalArgumentException("Access Key ID and Secret
                Access Key must not be null or empty");
            }

            AwsBasicCredentials awsCredentials =
            AwsBasicCredentials.create(accessKeyId, secretAccessKey);
            return S3AsyncClient.builder()

            .credentialsProvider(StaticCredentialsProvider.create(awsCredentials))
                .region(Region.US_WEST_2)
                .build();
        });
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}

```

```
}  
}
```

Amazon S3 SDK 메서드의 래퍼 클래스입니다.

```
public class S3DirectoriesActions {  
  
    private static IamAsyncClient iamAsyncClient;  
  
    private static Ec2AsyncClient ec2AsyncClient;  
    private static final Logger logger =  
        LoggerFactory.getLogger(S3DirectoriesActions.class);  
  
    private static IamAsyncClient getIAMAsyncClient() {  
        if (iamAsyncClient == null) {  
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()  
                .maxConcurrency(100)  
                .connectionTimeout(Duration.ofSeconds(60))  
                .readTimeout(Duration.ofSeconds(60))  
                .writeTimeout(Duration.ofSeconds(60))  
                .build();  
  
            ClientOverrideConfiguration overrideConfig =  
                ClientOverrideConfiguration.builder()  
                    .apiCallTimeout(Duration.ofMinutes(2))  
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))  
                    .retryStrategy(RetryMode.STANDARD)  
                    .build();  
  
            iamAsyncClient = IamAsyncClient.builder()  
                .httpClient(httpClient)  
                .overrideConfiguration(overrideConfig)  
                .build();  
        }  
        return iamAsyncClient;  
    }  
  
    private static Ec2AsyncClient getEc2AsyncClient() {  
        if (ec2AsyncClient == null) {  
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()  
                .maxConcurrency(100)  
                .connectionTimeout(Duration.ofSeconds(60))
```

```

        .readTimeout(Duration.ofSeconds(60))
        .writeTimeout(Duration.ofSeconds(60))
        .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ofMinutes(2))
        .apiCallAttemptTimeout(Duration.ofSeconds(90))
        .retryStrategy(RetryMode.STANDARD)
        .build();

        ec2AsyncClient = Ec2AsyncClient.builder()
        .httpClient(httpClient)
        .region(Region.US_WEST_2)
        .overrideConfiguration(overrideConfig)
        .build();
    }
    return ec2AsyncClient;
}

/**
 * Deletes the specified S3 bucket and all the objects within it
asynchronously.
 *
 * @param s3AsyncClient the S3 asynchronous client to use for the operations
 * @param bucketName the name of the S3 bucket to be deleted
 * @return a {@link CompletableFuture} that completes with a {@link
WaiterResponse} containing the
 *         {@link HeadBucketResponse} when the bucket has been successfully
deleted
 * @throws CompletionException if there was an error deleting the bucket or
its objects
 */
public CompletableFuture<WaiterResponse<HeadBucketResponse>>
deleteBucketAndObjectsAsync(S3AsyncClient s3AsyncClient, String bucketName) {
    ListObjectsV2Request listRequest = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .build();

    return s3AsyncClient.listObjectsV2(listRequest)
        .thenCompose(listResponse -> {
            if (!listResponse.contents().isEmpty()) {
                List<ObjectIdentifier> objectIdentifiers =
listResponse.contents().stream()

```

```

        .map(s3object ->
ObjectIdentifier.builder().key(s3object.key()).build())
        .collect(Collectors.toList());

        DeleteObjectsRequest deleteRequest =
DeleteObjectsRequest.builder()
        .bucket(bucketName)

.delete(Delete.builder().objects(objectIdentifiers).build())
        .build();

        return s3AsyncClient.deleteObjects(deleteRequest)
            .thenAccept(deleteResponse -> {
                if (!deleteResponse.errors().isEmpty()) {
                    deleteResponse.errors().forEach(error ->
                        logger.error("Couldn't delete object " +
error.key() + ". Reason: " + error.message()));
                }
            });
    }
    return CompletableFuture.completedFuture(null);
})
.thenCompose(ignored -> {
        DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
        .bucket(bucketName)
        .build();
        return s3AsyncClient.deleteBucket(deleteBucketRequest);
    })
    .thenCompose(ignored -> {
        S3AsyncWaiter waiter = s3AsyncClient.waiter();
        HeadBucketRequest headBucketRequest =
HeadBucketRequest.builder().bucket(bucketName).build();
        return waiter.waitUntilBucketNotExists(headBucketRequest);
    })
    .whenComplete((ignored, exception) -> {
        if (exception != null) {
            Throwable cause = exception.getCause();
            if (cause instanceof S3Exception) {
                throw new CompletionException("Error deleting bucket: " +
bucketName, cause);
            }
            throw new CompletionException("Failed to delete bucket and
objects: " + bucketName, exception);
        }
    });
}

```



```

        }
        logger.info("Bucket deleted successfully: " + bucketName);
    });
}

/**
 * Lists the objects in an S3 bucket asynchronously.
 *
 * @param s3Client the S3 async client to use for the operation
 * @param bucketName the name of the S3 bucket containing the objects to list
 * @return a {@link CompletableFuture} that contains the list of object keys
in the specified bucket
 */
public CompletableFuture<List<String>> listObjectsAsync(S3AsyncClient
s3Client, String bucketName) {
    ListObjectsV2Request request = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .build();

    return s3Client.listObjectsV2(request)
        .thenApply(response -> response.contents().stream()
            .map(S3Object::key)
            .toList())
        .whenComplete((result, exception) -> {
            if (exception != null) {
                throw new CompletionException("Couldn't list objects in
bucket: " + bucketName, exception);
            }
        });
}

/**
 * Retrieves an object from an Amazon S3 bucket asynchronously.
 *
 * @param s3Client the S3 async client to use for the operation
 * @param bucketName the name of the S3 bucket containing the object
 * @param keyName the unique identifier (key) of the object to retrieve
 * @return a {@link CompletableFuture} that, when completed, contains the
object's content as a {@link ResponseBytes} of {@link GetObjectResponse}
 */
public CompletableFuture<ResponseBytes<GetObjectResponse>>
getObjectAsync(S3AsyncClient s3Client, String bucketName, String keyName) {
    GetObjectRequest objectRequest = GetObjectRequest.builder()
        .key(keyName)

```

```

        .bucket(bucketName)
        .build();

        // Get the object asynchronously and transform it into a byte array
        return s3Client.getObject(objectRequest,
            AsyncResponseTransformer.toBytes())
            .exceptionally(exception -> {
                Throwable cause = exception.getCause();
                if (cause instanceof NoSuchKeyException) {
                    throw new CompletionException("Failed to get the object.
Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                }
                throw new CompletionException("Failed to get the object",
exception);
            });
    }

    /**
     * Asynchronously copies an object from one S3 bucket to another.
     *
     * @param s3Client          the S3 async client to use for the copy
operation
     * @param sourceBucket      the name of the source bucket
     * @param sourceKey         the key of the object to be copied in the source
bucket
     * @param destinationBucket the name of the destination bucket
     * @param destinationKey    the key of the copied object in the destination
bucket
     * @return a {@link CompletableFuture} that completes when the copy operation
is finished
     */
    public CompletableFuture<Void> copyObjectAsync(S3AsyncClient s3Client, String
sourceBucket, String sourceKey, String destinationBucket, String destinationKey)
    {
        CopyObjectRequest copyRequest = CopyObjectRequest.builder()
            .sourceBucket(sourceBucket)
            .sourceKey(sourceKey)
            .destinationBucket(destinationBucket)
            .destinationKey(destinationKey)
            .build();

        return s3Client.copyObject(copyRequest)
            .thenRun(() -> logger.info("Copied object '" + sourceKey + "' from
bucket '" + sourceBucket + "' to bucket '" + destinationBucket + "'"))
    }

```

```

        .whenComplete((ignored, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof S3Exception) {
                    throw new CompletionException("Couldn't
copy object '" + sourceKey + "' from bucket '" + sourceBucket + "'
to bucket '" + destinationBucket + "'. Reason: " + ((S3Exception)
cause).awsErrorDetails().errorMessage(), cause);
                }
                throw new CompletionException("Failed to copy object",
exception);
            }
        });
    }

    /**
     * Asynchronously creates a session for the specified S3 bucket.
     *
     * @param s3Client the S3 asynchronous client to use for creating the
session
     * @param bucketName the name of the S3 bucket for which to create the
session
     * @return a {@link CompletableFuture} that completes when the session is
created, or throws a {@link CompletionException} if an error occurs
     */
    public CompletableFuture<CreateSessionResponse>
createSessionAsync(S3AsyncClient s3Client, String bucketName) {
        CreateSessionRequest request = CreateSessionRequest.builder()
            .bucket(bucketName)
            .build();

        return s3Client.createSession(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
                        throw new CompletionException("Couldn't create the
session. Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(),
cause);
                    }
                    throw new CompletionException("Unexpected error occurred
while creating session", exception);
                }
                logger.info("Created session for bucket: " + bucketName);
            });
    }

```

```

    });

}

/**
 * Creates a new S3 directory bucket in a specified Zone (For example, a
 * specified Availability Zone in this code example).
 *
 * @param s3Client    The asynchronous S3 client used to create the bucket
 * @param bucketName The name of the bucket to be created
 * @param zone        The Availability Zone where the bucket will be created
 * @throws CompletionException if there's an error creating the bucket
 */
public CompletableFuture<CreateBucketResponse>
createDirectoryBucketAsync(S3AsyncClient s3Client, String bucketName, String
zone) {
    logger.info("Creating bucket: " + bucketName);

    CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
        .location(LocationInfo.builder()
            .type(LocationType.AVAILABILITY_ZONE)
            .name(zone)
            .build())
        .bucket(BucketInfo.builder()
            .type(BucketType.DIRECTORY)
            .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
            .build())
        .build();

    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .createBucketConfiguration(bucketConfiguration)
        .build();

    return s3Client.createBucket(bucketRequest)
        .whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof BucketAlreadyExistsException) {
                    throw new CompletionException("The bucket already exists:
" + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                }
            }
        })
}

```

```

        throw new CompletionException("Unexpected error occurred
while creating bucket", exception);
    }
    logger.info("Bucket created successfully with location: " +
response.location());
    });
}

/**
 * Creates an S3 bucket asynchronously.
 *
 * @param s3Client    the S3 async client to use for the bucket creation
 * @param bucketName  the name of the S3 bucket to create
 * @return a {@link CompletableFuture} that completes with the {@link
WaiterResponse} containing the {@link HeadBucketResponse}
 *         when the bucket is successfully created
 * @throws CompletionException if there's an error creating the bucket
 */
public CompletableFuture<WaiterResponse<HeadBucketResponse>>
createBucketAsync(S3AsyncClient s3Client, String bucketName) {
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .build();

    return s3Client.createBucket(bucketRequest)
        .thenCompose(response -> {
            S3AsyncWaiter s3Waiter = s3Client.waiter();
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();
            return s3Waiter.waitUntilBucketExists(bucketRequestWait);
        })
        .whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof BucketAlreadyExistsException) {
                    throw new CompletionException("The S3 bucket exists: " +
cause.getMessage(), cause);
                } else {
                    throw new CompletionException("Failed to create access
key: " + exception.getMessage(), exception);
                }
            }
            logger.info(bucketName + " is ready");
        });
}

```

```

        });
    }

    /**
     * Uploads an object to an Amazon S3 bucket asynchronously.
     *
     * @param s3Client      the S3 async client to use for the upload
     * @param bucketName    the destination S3 bucket name
     * @param bucketObject  the name of the object to be uploaded
     * @param text          the content to be uploaded as the object
     */
    public CompletableFuture<PutObjectResponse> putObjectAsync(S3AsyncClient
s3Client, String bucketName, String bucketObject, String text) {
        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(bucketObject)
            .build();

        return s3Client.putObject(objectRequest,
AsyncRequestBody.fromString(text))
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof NoSuchBucketException) {
                        throw new CompletionException("The S3 bucket does not
exist: " + cause.getMessage(), cause);
                    } else {
                        throw new CompletionException("Failed to create access
key: " + exception.getMessage(), exception);
                    }
                }
            });
    }

    /**
     * Creates an AWS IAM access key asynchronously for the specified user name.
     *
     * @param userName the name of the IAM user for whom to create the access key
     * @return a {@link CompletableFuture} that completes with the {@link
CreateAccessKeyResponse} containing the created access key
     */
    public CompletableFuture<CreateAccessKeyResponse> createAccessKeyAsync(String
userName) {
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()

```

```

        .userName(userName)
        .build();

    return getIAMAsyncClient().createAccessKey(request)
        .whenComplete((response, exception) -> {
            if (response != null) {
                logger.info("Access Key Created.");
            } else {
                if (exception == null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof IamException) {
                        throw new CompletionException("IAM error while
creating access key: " + cause.getMessage(), cause);
                    } else {
                        throw new CompletionException("Failed to create
access key: " + exception.getMessage(), exception);
                    }
                }
            }
        });
    }

    /**
     * Asynchronously selects an Availability Zone ID from the available EC2
     zones.
     *
     * @return A {@link CompletableFuture} that resolves to the selected
     Availability Zone ID.
     * @throws CompletionException if an error occurs during the request or
     processing.
     */
    public CompletableFuture<String> selectAvailabilityZoneIdAsync() {
        DescribeAvailabilityZonesRequest zonesRequest =
        DescribeAvailabilityZonesRequest.builder()
            .build();

        return getEc2AsyncClient().describeAvailabilityZones(zonesRequest)
            .thenCompose(response -> {
                List<AvailabilityZone> zonesList = response.availabilityZones();
                if (zonesList.isEmpty()) {
                    logger.info("No availability zones found.");
                    return CompletableFuture.completedFuture(null); // Return
null if no zones are found
                }
            })
    }

```

```

        List<String> zoneIds = zonesList.stream()
            .map(AvailabilityZone::zoneId) // Get the zoneId (e.g.,
"usw2-az1")
            .toList();

        return CompletableFuture.supplyAsync(() ->
promptUserForZoneSelection(zonesList, zoneIds))
            .thenApply(selectedZone -> {
                // Return only the selected Zone ID (e.g., "usw2-az1").
                return selectedZone.zoneId();
            });
    })
    .whenComplete((result, exception) -> {
        if (exception == null) {
            if (result != null) {
                logger.info("Selected Availability Zone ID: " + result);
            } else {
                logger.info("No availability zone selected.");
            }
        } else {
            Throwable cause = exception.getCause();
            if (cause instanceof Ec2Exception) {
                throw new CompletionException("EC2 error while selecting
availability zone: " + cause.getMessage(), cause);
            }
            throw new CompletionException("Failed to select availability
zone: " + exception.getMessage(), exception);
        }
    });
}

/**
 * Prompts the user to select an Availability Zone from the given list.
 *
 * @param zonesList the list of Availability Zones
 * @param zoneIds the list of zone IDs
 * @return the selected Availability Zone
 */
private static AvailabilityZone
promptUserForZoneSelection(List<AvailabilityZone> zonesList, List<String>
zoneIds) {
    Scanner scanner = new Scanner(System.in);
    int index = -1;

```



```

while (index < 0 || index >= zoneIds.size()) {
    logger.info("Select an availability zone:");
    IntStream.range(0, zoneIds.size()).forEach(i ->
        logger.info(i + ": " + zoneIds.get(i))
    );

    logger.info("Enter the number corresponding to your choice: ");
    if (scanner.hasNextInt()) {
        index = scanner.nextInt();
    } else {
        scanner.next();
    }
}

AvailabilityZone selectedZone = zonesList.get(index);
logger.info("You selected: " + selectedZone.zoneId());
return selectedZone;
}

/**
 * Asynchronously sets up a new VPC, including creating the VPC, finding the
 * associated route table, and
 * creating a VPC endpoint for the S3 service.
 *
 * @return a {@link CompletableFuture} that, when completed, contains a
 * AbstractMap with the
 *      * VPC ID and VPC endpoint ID.
 */
public CompletableFuture<AbstractMap.SimpleEntry<String, String>>
setupVPCAsync() {
    String cidr = "10.0.0.0/16";
    CreateVpcRequest vpcRequest = CreateVpcRequest.builder()
        .cidrBlock(cidr)
        .build();

    return getEc2AsyncClient().createVpc(vpcRequest)
        .thenCompose(vpcResponse -> {
            String vpcId = vpcResponse.vpc().vpcId();
            logger.info("VPC Created: {}", vpcId);

            Ec2AsyncWaiter waiter = getEc2AsyncClient().waiter();
            DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                .vpcIds(vpcId)

```

```

        .build();

        return waiter.waitForVpcAvailable(request)
            .thenApply(waiterResponse -> vpcId);
    })
    .thenCompose(vpcId -> {
        Filter filter = Filter.builder()
            .name("vpc-id")
            .values(vpcId)
            .build();

        DescribeRouteTablesRequest describeRouteTablesRequest =
DescribeRouteTablesRequest.builder()
            .filters(filter)
            .build();

        return
getEc2AsyncClient().describeRouteTables(describeRouteTablesRequest)
            .thenApply(routeTablesResponse -> {
                if (routeTablesResponse.routeTables().isEmpty()) {
                    throw new CompletionException("No route tables found
for VPC: " + vpcId, null);
                }
                String routeTableId =
routeTablesResponse.routeTables().get(0).routeTableId();
                logger.info("Route table found: {}", routeTableId);
                return new AbstractMap.SimpleEntry<>(vpcId,
routeTableId);
            });
    })
    .thenCompose(vpcAndRouteTable -> {
        String vpcId = vpcAndRouteTable.getKey();
        String routeTableId = vpcAndRouteTable.getValue();
        Region region =
getEc2AsyncClient().serviceClientConfiguration().region();
        String serviceName = String.format("com.amazonaws.%s.s3express",
region.id());

        CreateVpcEndpointRequest endpointRequest =
CreateVpcEndpointRequest.builder()
            .vpcId(vpcId)
            .routeTableIds(routeTableId)
            .serviceName(serviceName)
            .build();
    });

```

```

        return getEc2AsyncClient().createVpcEndpoint(endpointRequest)
            .thenApply(vpcEndpointResponse -> {
                String vpcEndpointId =
vpcEndpointResponse.vpcEndpoint().vpcEndpointId();
                logger.info("VPC Endpoint created: {}", vpcEndpointId);
                return new AbstractMap.SimpleEntry<>(vpcId,
vpcEndpointId);
            });
    })
    .exceptionally(exception -> {
        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
        if (cause instanceof Ec2Exception) {
            logger.error("EC2 error during VPC setup: {}",
cause.getMessage(), cause);
            throw new CompletionException("EC2 error during VPC setup: "
+ cause.getMessage(), cause);
        }

        logger.error("VPC setup failed: {}", cause.getMessage(), cause);
        throw new CompletionException("VPC setup failed: " +
cause.getMessage(), cause);
    });
}
}
}

```

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

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObject](#)
- [GetObject](#)
- [ListObjects](#)
- [PutObject](#)

PHP

SDK for PHP

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 디렉터리 버킷 및 S3 Express One Zone의 기본 사항을 보여주는 시나리오를 실행합니다.

```

echo "\n";
echo "-----\n";
echo "Welcome to the Amazon S3 Express Basics demo using PHP!\n";
echo "-----\n";

// Change these both of these values to use a different region/
availability zone.
$region = "us-west-2";
$az = "usw2-az1";

$this->s3Service = new S3Service(new S3Client(['region' => $region]));
$this->iamService = new IAMService(new IamClient(['region' => $region]));

$uuid = uniqid();

echo <<<INTRO
Let's get started! First, please note that S3 Express One Zone works best when
working within the AWS infrastructure,
specifically when working in the same Availability Zone. To see the best results
in this example, and when you implement
Directory buckets into your infrastructure, it is best to put your Compute
resources in the same AZ as your Directory
bucket.\n
INTRO;
    pressEnter();
    // 1. Configure a gateway VPC endpoint. This is the recommended method to
allow S3 Express One Zone traffic without
    // the need to pass through an internet gateway or NAT device.
echo "\n";

```

```

        echo "1. First, we'll set up a new VPC and VPC Endpoint if this program
        is running in an EC2 instance in the same AZ as your Directory buckets will be.
        \n";

        $ec2Choice = testable_readline("Are you running this in an EC2 instance
        located in the same AZ as your intended Directory buckets? Enter Y/y to setup a
        VPC Endpoint, or N/n/blank to skip this section.");
        if($ec2Choice == "Y" || $ec2Choice == "y") {
            echo "Great! Let's set up a VPC, retrieve the Route Table from it,
            and create a VPC Endpoint to connect the S3 Client to.\n";
            pressEnter();
            $this->ec2Service = new EC2Service(new Ec2Client(['region' =>
            $region]));
            $cidr = "10.0.0.0/16";
            $vpc = $this->ec2Service->createVpc($cidr);
            $this->resources['vpcId'] = $vpc['VpcId'];

            $this->ec2Service->waitForVpcAvailable($vpc['VpcId']);

            $routeTable = $this->ec2Service->describeRouteTables([], [
                [
                    'Name' => "vpc-id",
                    'Values' => [$vpc['VpcId']],
                ],
            ]);

            $serviceName = "com.amazonaws." . $this->ec2Service->getRegion() .
            ".s3express";
            $vpcEndpoint = $this->ec2Service->createVpcEndpoint($serviceName,
            $vpc['VpcId'], [$routeTable[0]]);
            $this->resources['vpcEndpointId'] = $vpcEndpoint['VpcEndpointId'];
        }else{
            echo "Skipping the VPC setup. Don't forget to use this in production!
            \n";
        }

        // 2. Policies, user, and roles with CDK.
        echo "\n";
        echo "2. Policies, users, and roles with CDK.\n";
        echo "Now, we'll set up some policies, roles, and a user. This user will
        only have permissions to do S3 Express One Zone actions.\n";
        pressEnter();

        $this->cloudFormationClient = new CloudFormationClient([]);
        $stackName = "cfn-stack-s3-express-basics-" . uniqid();

```

```

    $file = file_get_contents(__DIR__ . "/../../../../../resources/cfn/
s3_express_basics/s3_express_template.yml");
    $result = $this->cloudFormationClient->createStack([
        'StackName' => $stackName,
        'TemplateBody' => $file,
        'Capabilities' => ['CAPABILITY_IAM'],
    ]);
    $waiter = $this->cloudFormationClient->getWaiter("StackCreateComplete",
['StackName' => $stackName]);
    try {
        $waiter->promise()->wait();
    } catch (CloudFormationException $caught){
        echo "Error waiting for the CloudFormation stack to create: {$caught-
>getAwsErrorMessage()}\n";
        throw $caught;
    }
    $this->resources['stackName'] = $stackName;
    $stackInfo = $this->cloudFormationClient->describeStacks([
        'StackName' => $result['StackId'],
    ]);

    $expressUserName = "";
    $regularUserName = "";
    foreach($stackInfo['Stacks'][0]['Outputs'] as $output) {
        if ($output['OutputKey'] == "RegularUser") {
            $regularUserName = $output['OutputValue'];
        }
        if ($output['OutputKey'] == "ExpressUser") {
            $expressUserName = $output['OutputValue'];
        }
    }
    $regularKey = $this->iamService->createAccessKey($regularUserName);
    $regularCredentials = new Credentials($regularKey['AccessKeyId'],
    $regularKey['SecretAccessKey']);
    $expressKey = $this->iamService->createAccessKey($expressUserName);
    $expressCredentials = new Credentials($expressKey['AccessKeyId'],
    $expressKey['SecretAccessKey']);

    // 3. Create an additional client using the credentials with S3 Express
    permissions.
    echo "\n";
    echo "3. Create an additional client using the credentials with S3
    Express permissions.\n";

```

```

    echo "This client is created with the credentials associated with the
    user account with the S3 Express policy attached, so it can perform S3 Express
    operations.\n";
    pressEnter();
    $s3RegularClient = new S3Client([
        'Region' => $region,
        'Credentials' => $regularCredentials,
    ]);
    $s3RegularService = new S3Service($s3RegularClient);
    $s3ExpressClient = new S3Client([
        'Region' => $region,
        'Credentials' => $expressCredentials,
    ]);
    $s3ExpressService = new S3Service($s3ExpressClient);
    echo "All the roles and policies were created an attached to the user.
    Then, a new S3 Client and Service were created using that user's credentials.
    \n";

    echo "We can now use this client to make calls to S3 Express operations.
    Keeping permissions in mind (and adhering to least-privilege) is crucial to S3
    Express.\n";
    pressEnter();

    // 4. Create two buckets.
    echo "\n";
    echo "3. Create two buckets.\n";
    echo "Now we will create a Directory bucket, which is the linchpin of the
    S3 Express One Zone service.\n";
    echo "Directory buckets behave in different ways from regular S3 buckets,
    which we will explore here.\n";
    echo "We'll also create a normal bucket, put an object into the normal
    bucket, and copy it over to the Directory bucket.\n";
    pressEnter();

    // Create a directory bucket. These are different from normal S3 buckets
    in subtle ways.
    $directoryBucketName = "s3-express-demo-directory-bucket-$uuid--$az--x-
s3";
    echo "Now, let's create the actual Directory bucket, as well as a regular
    bucket.\n";
    pressEnter();
    $s3ExpressService->createBucket($directoryBucketName, [
        'CreateBucketConfiguration' => [
            'Bucket' => [

```

```
'Type' => "Directory", // This is what causes S3 to create a
Directory bucket as opposed to a normal bucket.
    'DataRedundancy' => "SingleAvailabilityZone",
    ],
    'Location' => [
        'Name' => $az,
        'Type' => "AvailabilityZone",
    ],
    ],
]);
$this->resources['directoryBucketName'] = $directoryBucketName;

// Create a normal bucket.
$normalBucketName = "normal-bucket-$uuid";
$s3RegularService->createBucket($normalBucketName);
$this->resources['normalBucketName'] = $normalBucketName;
echo "Great! Both buckets were created.\n";
pressEnter();

// 5. Create an object and copy it over.
echo "\n";
echo "5. Create an object and copy it over.\n";
echo "We'll create a basic object consisting of some text and upload it
to the normal bucket.\n";
echo "Next, we'll copy the object into the Directory bucket using the
regular client.\n";
echo "This works fine, because Copy operations are not restricted for
Directory buckets.\n";
pressEnter();

$objectKey = "basic-text-object";
$s3RegularService->putObject($normalBucketName, $objectKey, $args =
['Body' => "Look Ma, I'm a bucket!"]);
$this->resources['objectKey'] = $objectKey;

// Create a session to access the directory bucket. The SDK Client will
automatically refresh this as needed.
$s3ExpressService->createSession($directoryBucketName);
$s3ExpressService->copyObject($directoryBucketName, $objectKey,
"$normalBucketName/$objectKey");

echo "It worked! It's important to remember the user permissions when
interacting with Directory buckets.\n";
```



```
    echo "Instead of validating permissions on every call as normal buckets
do, Directory buckets utilize the user credentials and session token to
validate.\n";
    echo "This allows for much faster connection speeds on every call. For
single calls, this is low, but for many concurrent calls, this adds up to a lot
of time saved.\n";
    pressEnter();

    // 6. Demonstrate performance difference.
    echo "\n";
    echo "6. Demonstrate performance difference.\n";
    $downloads = 1000;
    echo "Now, let's do a performance test. We'll download the same object
from each bucket $downloads times and compare the total time needed. Note: the
performance difference will be much more pronounced if this example is run in an
EC2 instance in the same AZ as the bucket.\n";
    $downloadChoice = testable_readline("If you would like to download each
object $downloads times, press enter. Otherwise, enter a custom amount and press
enter.");
    if($downloadChoice && is_numeric($downloadChoice) && $downloadChoice <
1000000){ // A million is enough. I promise.
        $downloads = $downloadChoice;
    }

    // Download the object $downloads times from each bucket and time it to
demonstrate the speed difference.
    $directoryStartTime = hrtime(true);
    for($i = 0; $i < $downloads; ++$i){
        $s3ExpressService->getObject($directoryBucketName, $objectKey);
    }
    $directoryEndTime = hrtime(true);
    $directoryTimeDiff = $directoryEndTime - $directoryStartTime;

    $normalStartTime = hrtime(true);
    for($i = 0; $i < $downloads; ++$i){
        $s3RegularService->getObject($normalBucketName, $objectKey);
    }
    $normalEndTime = hrtime(true);
    $normalTimeDiff = $normalEndTime - $normalStartTime;

    echo "The directory bucket took $directoryTimeDiff nanoseconds, while the
normal bucket took $normalTimeDiff.\n";
```

```

    echo "That's a difference of " . ($normalTimeDiff - $directoryTimeDiff) .
    " nanoseconds, or " . (($normalTimeDiff - $directoryTimeDiff)/1000000000) . "
    seconds.\n";
    pressEnter();

    // 7. Populate the buckets to show the lexicographical difference.
    echo "\n";
    echo "7. Populate the buckets to show the lexicographical difference.\n";
    echo "Now let's explore how Directory buckets store objects in a
different manner to regular buckets.\n";
    echo "The key is in the name \"Directory!\".\n";
    echo "Where regular buckets store their key/value pairs in a flat manner,
Directory buckets use actual directories/folders.\n";
    echo "This allows for more rapid indexing, traversing, and therefore
retrieval times!\n";
    echo "The more segmented your bucket is, with lots of directories, sub-
directories, and objects, the more efficient it becomes.\n";
    echo "This structural difference also causes ListObjects to behave
differently, which can cause unexpected results.\n";
    echo "Let's add a few more objects with layered directories as see how
the output of ListObjects changes.\n";
    pressEnter();

    // Populate a few more files in each bucket so that we can use
ListObjects and show the difference.
    $otherObject = "other/$objectKey";
    $altObject = "alt/$objectKey";
    $otherAltObject = "other/alt/$objectKey";
    $s3ExpressService->putObject($directoryBucketName, $otherObject);
    $s3RegularService->putObject($normalBucketName, $otherObject);
    $this->resources['otherObject'] = $otherObject;
    $s3ExpressService->putObject($directoryBucketName, $altObject);
    $s3RegularService->putObject($normalBucketName, $altObject);
    $this->resources['altObject'] = $altObject;
    $s3ExpressService->putObject($directoryBucketName, $otherAltObject);
    $s3RegularService->putObject($normalBucketName, $otherAltObject);
    $this->resources['otherAltObject'] = $otherAltObject;

    $listDirectoryBucket = $s3ExpressService-
>listObjects($directoryBucketName);
    $listNormalBucket = $s3RegularService->listObjects($normalBucketName);

    // Directory bucket content
    echo "Directory bucket content\n";

```

```

        foreach($listDirectoryBucket['Contents'] as $result){
            echo $result['Key'] . "\n";
        }

        // Normal bucket content
        echo "\nNormal bucket content\n";
        foreach($listNormalBucket['Contents'] as $result){
            echo $result['Key'] . "\n";
        }

        echo "Notice how the normal bucket lists objects in lexicographical
order, while the directory bucket does not. This is because the normal bucket
considers the whole \"key\" to be the object identifies, while the directory
bucket actually creates directories and uses the object \"key\" as a path to the
object.\n";
        pressEnter();

        echo "\n";
        echo "That's it for our tour of the basic operations for S3 Express One
Zone.\n";
        $cleanUp = testable_readline("Would you like to delete all the resources
created during this demo? Enter Y/y to delete all the resources.");
        if($cleanUp){
            $this->cleanUp();
        }
    }

namespace S3;

use Aws\CommandInterface;
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;

    public function __construct(S3Client $client = null, $verbose = false)

```

```
{
    if ($client) {
        $this->client = $client;
    } else {
        $this->client = new S3Client([
            'version' => 'latest',
            'region' => 'us-west-2',
        ]);
    }
    $this->verbose = $verbose;
}

public function setVerbose($verbose)
{
    $this->verbose = $verbose;
}

public function isVerbose(): bool
{
    return $this->verbose;
}

public function getClient(): S3Client
{
    return $this->client;
}

public function setClient(S3Client $client)
{
    $this->client = $client;
}

public function emptyAndDeleteBucket($bucketName, array $args = [])
{
    try {
        $objects = $this->listAllObjects($bucketName, $args);
        $this->deleteObjects($bucketName, $objects, $args);
        if ($this->verbose) {
            echo "Deleted all objects and folders from $bucketName.\n";
        }
        $this->deleteBucket($bucketName, $args);
    } catch (AwsException $exception) {
        if ($this->verbose) {
```

```

        echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
        echo "\nPlease fix error with bucket deletion before continuing.
\n";
    }
    throw $exception;
}

}

public function createBucket(string $bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);
    try {
        $this->client->createBucket($parameters);
        if ($this->verbose) {
            echo "Created the bucket named: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
            echo "Please fix error with bucket creation before continuing.";
        }
        throw $exception;
    }
}

public function putObject(string $bucketName, string $key, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
$args);
    try {
        $this->client->putObject($parameters);
        if ($this->verbose) {
            echo "Uploaded the object named: $key to the bucket named:
$bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {

```

```

        echo "Failed to create $key in $bucketName with error:
{$exception->getMessage()}\n";
        echo "Please fix error with object uploading before continuing.";
    }
    throw $exception;
}
}

public function getObject(string $bucketName, string $key, array $args = []):
Result
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
$args);
    try {
        $object = $this->client->getObject($parameters);
        if ($this->verbose) {
            echo "Downloaded the object named: $key to the bucket named:
$bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to download $key from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object downloading before
continuing.";
        }
        throw $exception;
    }
    return $object;
}

public function copyObject($bucketName, $key, $copySource, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
"CopySource" => $copySource], $args);
    try {
        $this->client->copyObject($parameters);
        if ($this->verbose) {
            echo "Copied the object from: $copySource in $bucketName to:
$key.\n";

```

```

    }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to copy $copySource in $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object copying before continuing.";
        }
        throw $exception;
    }
}

public function listObjects(string $bucketName, $start = 0, $max = 1000,
array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
"MaxKeys" => $max], $args);
    try {
        $objects = $this->client->listObjectsV2($parameters);
        if ($this->verbose) {
            echo "Retrieved the list of objects from: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to retrieve the objects from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with list objects before continuing.";
        }
        throw $exception;
    }
    return $objects;
}

public function listAllObjects($bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);

    $contents = [];
    $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

    foreach ($paginator as $result) {

```

```
        if($result['KeyCount'] == 0){
            break;
        }
        foreach ($result['Contents'] as $object) {
            $contents[] = $object;
        }
    }
    return $contents;
}

public function deleteObjects(string $bucketName, array $objects, array $args
= [])
{
    $listOfObjects = array_map(
        function ($object) {
            return ['Key' => $object];
        },
        array_column($objects, 'Key')
    );
    if(!$listOfObjects){
        return;
    }

    $parameters = array_merge(['Bucket' => $bucketName, 'Delete' =>
['Objects' => $listOfObjects]], $args);
    try {
        $this->client->deleteObjects($parameters);
        if ($this->verbose) {
            echo "Deleted the list of objects from: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete the list of objects from $bucketName with
error: {$exception->getMessage()}\n";
            echo "Please fix error with object deletion before continuing.";
        }
        throw $exception;
    }
}
```



```
public function deleteBucket(string $bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);
    try {
        $this->client->deleteBucket($parameters);
        if ($this->verbose) {
            echo "Deleted the bucket named: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
            echo "Please fix error with bucket deletion before continuing.";
        }
        throw $exception;
    }
}

public function deleteObject(string $bucketName, string $fileName, array
$args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
$args);
    try {
        $this->client->deleteObject($parameters);
        if ($this->verbose) {
            echo "Deleted the object named: $fileName from $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete $fileName from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object deletion before continuing.";
        }
        throw $exception;
    }
}

public function listBuckets(array $args = [])
{

```

```

        try {
            $buckets = $this->client->listBuckets($args);
            if ($this->verbose) {
                echo "Retrieved all " . count($buckets) . "\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket lists before continuing.";
            }
            throw $exception;
        }
        return $buckets;
    }

    public function preSignedUrl(CommandInterface $command, DateTimeInterface|
int|string $expires, array $options = [])
    {
        $request = $this->client->createPresignedRequest($command, $expires,
$options);
        try {
            $presignedUrl = (string)$request->getUri();
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
                echo "Please fix error with presigned urls before continuing.";
            }
            throw $exception;
        }
        return $presignedUrl;
    }

    public function createSession(string $bucketName)
    {
        try{
            $result = $this->client->createSession([
                'Bucket' => $bucketName,
            ]);
        }
    }

```

```

        return $result;
    }catch(S3Exception $caught){
        if($caught->getAwsErrorType() == "NoSuchBucket"){
            echo "The specified bucket does not exist.";
        }
        throw $caught;
    }
}
}
}

```

- API 세부 정보는 AWS SDK for PHP API 참조의 다음 항목을 참조하세요.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObject](#)
 - [GetObject](#)
 - [ListObjects](#)
 - [PutObject](#)

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 디렉터리 버킷 및 S3 Express One Zone의 기본 사항을 보여주는 시나리오를 실행합니다.

```

class S3ExpressScenario:
    """Runs an interactive scenario that shows how to get started with S3
    Express."""

```

```

def __init__(
    self,
    cloud_formation_resource: ServiceResource,
    ec2_client: client,
    iam_client: client,
):
    self.cloud_formation_resource = cloud_formation_resource
    self.ec2_client = ec2_client
    self.iam_client = iam_client
    self.region = ec2_client.meta.region_name
    self.stack = None
    self.vpc_id = None
    self.vpc_endpoint_id = None
    self.regular_bucket_name = None
    self.directory_bucket_name = None
    self.s3_express_wrapper = None
    self.s3_regular_wrapper = None

def s3_express_scenario(self):
    """
    Runs the scenario.
    """
    print("")
    print_dashes()
    print("Welcome to the Amazon S3 Express Basics demo using Python (Boto
3)!")
    print_dashes()
    print(
        """
Let's get started! First, please note that S3 Express One Zone works best when
working within the AWS infrastructure,
specifically when working in the same Availability Zone. To see the best results
in this example and when you implement
Directory buckets into your infrastructure, it is best to put your compute
resources in the same AZ as your Directory
bucket.
        """
    )
    press_enter_to_continue()

    # Create an optional VPC and create 2 IAM users.
    express_user_name, regular_user_name = self.create_vpc_and_users()

```

```

        # Set up two S3 clients, one regular and one express, and two buckets,
        one regular and one express.
        self.setup_clients_and_buckets(express_user_name, regular_user_name)

        # Create an S3 session for the express S3 client and add objects to the
        buckets.
        bucket_object = self.create_session_and_add_objects()

        # Demonstrate performance differences between regular and express
        buckets.
        self.demonstrate_performance(bucket_object)

        # Populate the buckets to show the lexicographical difference between
        regular and express buckets.
        self.show_lexicographical_differences(bucket_object)

        print("")
        print("That's it for our tour of the basic operations for S3 Express One
        Zone.")

        if q.ask(
            "Would you like to delete all the resources created during this demo
            (y/n)? ",
            q.is_yesno,
        ):
            self.cleanup()

    def create_vpc_and_users(self) -> None:
        """
        Optionally create a VPC.
        Create two IAM users, one with S3 Express One Zone permissions and one
        without.
        """
        # Configure a gateway VPC endpoint. This is the recommended method to
        allow S3 Express One Zone traffic without
        # the need to pass through an internet gateway or NAT device.
        print(
            """
1. First, we'll set up a new VPC and VPC Endpoint if this program is running in
an EC2 instance in the same AZ as your
Directory buckets will be. Are you running this in an EC2 instance located in the
same AZ as your intended Directory buckets?
"""
        )

```

```

        if q.ask("Do you want to setup a VPC Endpoint? (y/n) ", q.is_yn):
            print(
                "Great! Let's set up a VPC, retrieve the Route Table from it, and
create a VPC Endpoint to connect the S3 Client to."
            )
            self.setup_vpc()
            press_enter_to_continue()
        else:
            print("Skipping the VPC setup. Don't forget to use this in
production!")
            print(
                """

```

2. Policies, users, and roles with CDK.

Now, we'll set up some policies, roles, and a user. This user will only have permissions to do S3 Express One Zone actions.

```

        """
    )
    press_enter_to_continue()
    stack_name = f"cfn-stack-s3-express-basics--{uuid.uuid4()}"
    template_as_string = S3ExpressScenario.get_template_as_string()
    self.stack = self.deploy_cloudformation_stack(stack_name,
template_as_string)
    regular_user_name = None
    express_user_name = None
    outputs = self.stack.outputs
    for output in outputs:
        if output.get("OutputKey") == "RegularUser":
            regular_user_name = output.get("OutputValue")
        elif output.get("OutputKey") == "ExpressUser":
            express_user_name = output.get("OutputValue")
    if not regular_user_name or not express_user_name:
        error_string = f"""
Failed to retrieve required outputs from CloudFormation stack.
'regular_user_name'={regular_user_name},
'express_user_name'={express_user_name}
        """
        logger.error(error_string)
        raise ValueError(error_string)
    return express_user_name, regular_user_name

def setup_clients_and_buckets(
    self, express_user_name: str, regular_user_name: str
) -> None:
    """

```

Set up two S3 clients, one regular and one express, and two buckets, one regular and one express.

```
:param express_user_name: The name of the user with S3 Express
permissions.
```

```
:param regular_user_name: The name of the user with regular S3
permissions.
```

```
"""
```

```
regular_credentials = self.create_access_key(regular_user_name)
```

```
express_credentials = self.create_access_key(express_user_name)
```

```
# 3. Create an additional client using the credentials with S3 Express
permissions.
```

```
print(
```

```
    """
```

3. Create an additional client using the credentials with S3 Express permissions.

This client is created with the credentials associated with the user account with the S3 Express policy attached, so it can perform S3 Express operations.

```
"""
```

```
)
```

```
press_enter_to_continue()
```

```
s3_regular_client = self.create_s3_client_with_access_key_credentials(
    regular_credentials
```

```
)
```

```
self.s3_regular_wrapper = S3ExpressWrapper(s3_regular_client)
```

```
s3_express_client = self.create_s3_client_with_access_key_credentials(
    express_credentials
```

```
)
```

```
self.s3_express_wrapper = S3ExpressWrapper(s3_express_client)
```

```
print(
```

```
    """
```

All the roles and policies were created and attached to the user. Then a new S3 Client were created using

that user's credentials. We can now use this client to make calls to S3 Express operations. Keeping permissions in mind

(and adhering to least-privilege) is crucial to S3 Express.

```
"""
```

```
)
```

```
press_enter_to_continue()
```

```
# 4. Create two buckets.
```

```
print(
```

```
    """
```

3. Create two buckets.

Now we will create a Directory bucket which is the linchpin of the S3 Express One Zone service. Directory buckets

behave in different ways from regular S3 buckets which we will explore here.

We'll also create a normal bucket, put an object into the normal bucket, and copy it over to the Directory bucket.

```
"""
    )

    # Create a directory bucket. These are different from normal S3 buckets
    in subtle ways.
    bucket_prefix = q.ask(
        "Enter a bucket name prefix that will be used for both buckets: ",
        q.re_match(r"[a-z0-9](?:[a-z0-9-\.]*)[a-z0-9]$"),
    )

    # Some availability zones are not supported for Directory buckets. We'll
    choose one that is supported.
    print(
        "Now, let's choose an availability zone for the Directory bucket.
    We'll choose one that is supported."
    )
    while True:
        availability_zone = self.select_availability_zone_id(self.region)
        # Construct the parts of a directory bucket name that is made unique
        with a UUID string.
        directory_bucket_suffix = f"--{availability_zone['ZoneId']}--x-s3"
        max_uuid_length = 63 - len(bucket_prefix) -
        len(directory_bucket_suffix) - 1
        bucket_uuid = str(uuid.uuid4()).replace("-", "").[:max_uuid_length]
        directory_bucket_name = (
            f"{bucket_prefix}-{bucket_uuid}{directory_bucket_suffix}"
        )
        regular_bucket_name = f"{bucket_prefix}-regular-{bucket_uuid}"
        configuration = {
            "Bucket": {
                "Type": "Directory",
                "DataRedundancy": "SingleAvailabilityZone",
            },
            "Location": {
                "Name": availability_zone["ZoneId"],
                "Type": "AvailabilityZone",
            },
        }
        press_enter_to_continue()
        print(
```



```

        "Now, let's create the actual Directory bucket, as well as a
        regular bucket."
    )
    press_enter_to_continue()
    try:
        self.s3_express_wrapper.create_bucket(
            directory_bucket_name, configuration
        )
        break
    except ClientError as client_error:
        if client_error.response["Error"]["Code"] == "InvalidBucketName":
            print(
                f"Bucket '{directory_bucket_name}' is invalid. This may
                be because of selected availability zone."
            )
            if q.ask(
                "Would you like to select a different availability zone?"
            ),
                q.is_yesno,
            ):
                continue
            else:
                raise
        else:
            raise
    print(f"Created directory bucket, '{directory_bucket_name}'")
    self.directory_bucket_name = directory_bucket_name

    self.s3_regular_wrapper.create_bucket(regular_bucket_name)
    print(f"Created regular bucket, '{regular_bucket_name}'")
    self.regular_bucket_name = regular_bucket_name
    print("Great! Both buckets were created.")
    press_enter_to_continue()

def create_session_and_add_objects(self) -> None:
    """
    Create a session for the express S3 client and add objects to the
    buckets.
    """
    print(
        """

```

5. Create an object and copy it over.

We'll create a basic object consisting of some text and upload it to the normal bucket. Next we'll copy the object

into the Directory bucket using the regular client. This works fine because copy operations are not restricted for Directory buckets.

```

    """
    )
    press_enter_to_continue()
    bucket_object = "basic-text-object"
    self.s3_regular_wrapper.put_object(
        self.regular_bucket_name, bucket_object, "Look Ma, I'm a bucket!"
    )
    self.s3_express_wrapper.create_session(self.directory_bucket_name)
    self.s3_express_wrapper.copy_object(
        self.regular_bucket_name,
        bucket_object,
        self.directory_bucket_name,
        bucket_object,
    )
    print(
        """

```

It worked! It's important to remember the user permissions when interacting with Directory buckets. Instead of validating permissions on every call as normal buckets do, Directory buckets utilize the user credentials and session token to validate.

This allows for much faster connection speeds on every call. For single calls, this is low, but for many concurrent calls this adds up to a lot of time saved.

```

    """
    )
    press_enter_to_continue()
    return bucket_object

    def demonstrate_performance(self, bucket_object: str) -> None:
        """
        Demonstrate performance differences between regular and Directory
        buckets.
        :param bucket_object: The name of the object to download from each
        bucket.
        """
        print("")
        print("6. Demonstrate performance difference.")
        print(
            """

```

Now, let's do a performance test. We'll download the same object from each bucket 'downloads' times

and compare the total time needed. Note: the performance difference will be much more pronounced if this example is run in an EC2 instance in the same Availability Zone as the bucket.

```

"""
    )
    downloads = 1000
    print(
        f"The number of downloads of the same object for this example is set
at {downloads}."
    )
    if q.ask("Would you like to download a different number? (y/n) ",
q.is_yesno):
        max_downloads = 1000000
        downloads = q.ask(
            f"Enter a number between 1 and {max_downloads} for the number of
downloads: ",
            q.is_int,
            q.in_range(1, max_downloads),
        )
        # Download the object 'downloads' times from each bucket and time it to
demonstrate the speed difference.
        print("Downloading from the Directory bucket.")
        directory_time_start = time.time_ns()

        for index in range(downloads):
            if index % 10 == 0:
                print(f"Download {index} of {downloads}")

                self.s3_express_wrapper.get_object(
                    self.directory_bucket_name, bucket_object
                )

        directory_time_difference = time.time_ns() - directory_time_start
        print("Downloading from the normal bucket.")
        normal_time_start = time.time_ns()

        for index in range(downloads):
            if index % 10 == 0:
                print(f"Download {index} of {downloads}")
                self.s3_regular_wrapper.get_object(self.regular_bucket_name,
bucket_object)

        normal_time_difference = time.time_ns() - normal_time_start
        print(

```

```

        f"The directory bucket took {directory_time_difference} nanoseconds,
while the normal bucket took {normal_time_difference}."
    )
    difference = normal_time_difference - directory_time_difference
    print(f"That's a difference of {difference} nanoseconds, or")
    print(f"{{(difference) / 1000000000}} seconds.")
    if difference < 0:
        print(
            "The directory buckets were slower. This can happen if you are
not running on the cloud within a vpc."
        )
    press_enter_to_continue()

def show_lexicographical_differences(self, bucket_object: str) -> None:
    """
    Show the lexicographical difference between Directory buckets and regular
    buckets.

    This is done by creating a few objects in each bucket and listing them to
    show the difference.
    :param bucket_object: The object to use for the listing operations.
    """
    print(
        """
7. Populate the buckets to show the lexicographical difference.
Now let's explore how Directory buckets store objects in a different manner to
regular buckets. The key is in the name
"Directory". Where regular buckets store their key/value pairs in a flat manner,
Directory buckets use actual
directories/folders. This allows for more rapid indexing, traversing, and
therefore retrieval times! The more segmented
your bucket is, with lots of directories, sub-directories, and objects, the more
efficient it becomes. This structural
difference also causes ListObjects to behave differently, which can cause
unexpected results. Let's add a few more
objects with layered directories to see how the output of ListObjects changes.
        """
    )
    press_enter_to_continue()
    # Populate a few more files in each bucket so that we can use ListObjects
    and show the difference.
    other_object = f"other/{bucket_object}"
    alt_object = f"alt/{bucket_object}"
    other_alt_object = f"other/alt/{bucket_object}"

```

```

        self.s3_regular_wrapper.put_object(self.regular_bucket_name,
other_object, "")
        self.s3_express_wrapper.put_object(self.directory_bucket_name,
other_object, "")
        self.s3_regular_wrapper.put_object(self.regular_bucket_name, alt_object,
"")
        self.s3_express_wrapper.put_object(self.directory_bucket_name,
alt_object, "")
        self.s3_regular_wrapper.put_object(
            self.regular_bucket_name, other_alt_object, ""
        )
        self.s3_express_wrapper.put_object(
            self.directory_bucket_name, other_alt_object, ""
        )
        directory_bucket_objects = self.s3_express_wrapper.list_objects(
            self.directory_bucket_name
        )

        regular_bucket_objects = self.s3_regular_wrapper.list_objects(
            self.regular_bucket_name
        )

        print("Directory bucket content")
        for bucket_object in directory_bucket_objects:
            print(f"    {bucket_object['Key']}")
        print("Normal bucket content")
        for bucket_object in regular_bucket_objects:
            print(f"    {bucket_object['Key']}")
        print(
            """
Notice how the normal bucket lists objects in lexicographical order, while the
directory bucket does not. This is
because the normal bucket considers the whole "key" to be the object identifier,
while the directory bucket actually
creates directories and uses the object "key" as a path to the object.
            """
        )
        press_enter_to_continue()

    def cleanup(self) -> None:
        """
        Delete resources created by this scenario.
        """
        if self.directory_bucket_name is not None:

```

```

        self.s3_express_wrapper.delete_bucket_and_objects(
            self.directory_bucket_name
        )
        print(f"Deleted directory bucket, '{self.directory_bucket_name}'")
        self.directory_bucket_name = None

    if self.regular_bucket_name is not None:

self.s3_regular_wrapper.delete_bucket_and_objects(self.regular_bucket_name)
        print(f"Deleted regular bucket, '{self.regular_bucket_name}'")
        self.regular_bucket_name = None

    if self.stack is not None:
        self.destroy_cloudformation_stack(self.stack)
        self.stack = None

    self.tear_done_vpc()

def create_access_key(self, user_name: str) -> dict[str, any]:
    """
    Creates an access key for the user.
    :param user_name: The name of the user.
    :return: The access key for the user.
    """
    try:
        access_key = self.iam_client.create_access_key(UserName=user_name)
        return access_key["AccessKey"]
    except ClientError as client_error:
        logging.error(
            "Couldn't create the access key. Here's why: %s",
            client_error.response["Error"]["Message"],
        )
        raise

def create_s3__client_with_access_key_credentials(
    self, access_key: dict[str, any]
) -> client:
    """
    Creates an S3 client with access key credentials.
    :param access_key: The access key for the user.
    :return: The S3 Express One Zone client.
    """
    try:
        s3_express_client = boto3.client(

```

```

        "s3",
        aws_access_key_id=access_key["AccessKeyId"],
        aws_secret_access_key=access_key["SecretAccessKey"],
        region_name=self.region,
    )
    return s3_express_client
except ClientError as client_error:
    logging.error(
        "Couldn't create the S3 Express One Zone client. Here's why: %s",
        client_error.response["Error"]["Message"],
    )
    raise

def select_availability_zone_id(self, region: str) -> dict[str, any]:
    """
    Selects an availability zone.
    :param region: The region to select the availability zone from.
    :return: The availability zone dictionary.
    """
    try:
        response = self.ec2_client.describe_availability_zones(
            Filters=[{"Name": "region-name", "Values": [region]}]
        )
        availability_zones = response["AvailabilityZones"]
        zone_names = [zone["ZoneName"] for zone in availability_zones]
        index = q.choose("Select an availability zone: ", zone_names)
        return availability_zones[index]
    except ClientError as client_error:
        logging.error(
            "Couldn't describe availability zones. Here's why: %s",
            client_error.response["Error"]["Message"],
        )
        raise

def deploy_cloudformation_stack(
    self, stack_name: str, cfn_template: str
) -> ServiceResource:
    """
    Deploys prerequisite resources used by the scenario. The resources are
    defined in the associated `cfn_template.yaml` AWS CloudFormation script
    and are deployed
    as a CloudFormation stack, so they can be easily managed and destroyed.

    :param stack_name: The name of the CloudFormation stack.

```

```

        :param cfn_template: The CloudFormation template as a string.
        :return: The CloudFormation stack resource.
        """
        print(f"Deploying CloudFormation stack: {stack_name}.")
        stack = self.cloud_formation_resource.create_stack(
            StackName=stack_name,
            TemplateBody=cfn_template,
            Capabilities=["CAPABILITY_NAMED_IAM"],
        )
        print(f"CloudFormation stack creation started: {stack_name}")
        print("Waiting for CloudFormation stack creation to complete...")
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_create_complete"
        )
        waiter.wait(StackName=stack.name)
        stack.load()
        print("CloudFormation stack creation complete.")

    return stack

def destroy_cloudformation_stack(self, stack: ServiceResource) -> None:
    """
    Destroys the resources managed by the CloudFormation stack, and the
    CloudFormation
    stack itself.

    :param stack: The CloudFormation stack that manages the example
    resources.
    """
    try:
        print(
            f"CloudFormation stack '{stack.name}' is being deleted. This may
            take a few minutes."
        )
        stack.delete()
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_delete_complete"
        )
        waiter.wait(StackName=stack.name)
        print(f"CloudFormation stack '{stack.name}' has been deleted.")
    except ClientError as client_error:
        logging.error(
            "Couldn't delete the CloudFormation stack. Here's why: %s",
            client_error.response["Error"]["Message"],

```



```

    )

    @staticmethod
    def get_template_as_string() -> str:
        """
        Returns a string containing this scenario's CloudFormation template.
        """
        script_directory = os.path.dirname(os.path.abspath(__file__))
        template_file_path = os.path.join(script_directory,
            "s3_express_template.yaml")
        file = open(template_file_path, "r")
        return file.read()

    def setup_vpc(self):
        cidr = "10.0.0.0/16"
        try:
            response = self.ec2_client.create_vpc(CidrBlock=cidr)
            self.vpc_id = response["Vpc"]["VpcId"]

            waiter = self.ec2_client.get_waiter("vpc_available")
            waiter.wait(VpcIds=[self.vpc_id])
            print(f"Created vpc {self.vpc_id}")

        except ClientError as client_error:
            logging.error(
                "Couldn't create the vpc. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise

        try:
            response = self.ec2_client.describe_route_tables(
                Filters=[{"Name": "vpc-id", "Values": [self.vpc_id]}]
            )
            route_table_id = response["RouteTables"][0]["RouteTableId"]
            service_name = f"com.amazonaws.
{self.ec2_client.meta.region_name}.s3express"

            response = self.ec2_client.create_vpc_endpoint(
                VpcId=self.vpc_id,
                RouteTableIds=[route_table_id],
                ServiceName=service_name,
            )
            self.vpc_endpoint_id = response["VpcEndpoint"]["VpcEndpointId"]
            print(f"Created vpc endpoint {self.vpc_endpoint_id}")

```

```

except ClientError as client_error:
    logging.error(
        "Couldn't create the vpc endpoint. Here's why: %s",
        client_error.response["Error"]["Message"],
    )
    raise

def tear_down_vpc(self) -> None:
    if self.vpc_endpoint_id is not None:
        try:
            self.ec2_client.delete_vpc_endpoints(
                VpcEndpointIds=[self.vpc_endpoint_id]
            )
            print(f"Deleted vpc endpoint {self.vpc_endpoint_id}.")
            self.vpc_endpoint_id = None
        except ClientError as client_error:
            logging.error(
                "Couldn't delete the vpc endpoint %s. Here's why: %s",
                self.vpc_endpoint_id,
                client_error.response["Error"]["Message"],
            )
    if self.vpc_id is not None:
        try:
            self.ec2_client.delete_vpc(VpcId=self.vpc_id)
            print(f"Deleted vpc {self.vpc_id}")
            self.vpc_id = None
        except ClientError as client_error:
            logging.error(
                "Couldn't delete the vpc %s. Here's why: %s",
                self.vpc_id,
                client_error.response["Error"]["Message"],
            )

```

Amazon S3 Express SDK 함수의 래퍼 클래스입니다.

```

class S3ExpressWrapper:
    """Encapsulates Amazon S3 Express One Zone actions using the client
    interface."""

```

```

def __init__(self, s3_client: Any) -> None:
    """
    Initializes the S3ExpressWrapper with an S3 client.

    :param s3_client: A Boto3 Amazon S3 client. This client provides low-
level
                        access to AWS S3 services.
    """
    self.s3_client = s3_client

    @classmethod
    def from_client(cls) -> "S3ExpressWrapper":
        """
        Creates an S3ExpressWrapper instance with a default s3 client.

        :return: An instance of S3ExpressWrapper initialized with the default S3
client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def create_bucket(
        self, bucket_name: str, bucket_configuration: dict[str, any] = None
    ) -> None:
        """
        Creates a bucket.
        :param bucket_name: The name of the bucket.
        :param bucket_configuration: The optional configuration for the bucket.
        """
        try:
            params = {"Bucket": bucket_name}
            if bucket_configuration:
                params["CreateBucketConfiguration"] = bucket_configuration

            self.s3_client.create_bucket(**params)
        except ClientError as client_error:
            # Do not log InvalidBucketName error because it is logged elsewhere.
            if client_error.response["Error"]["Code"] != "InvalidBucketName":
                logging.error(
                    "Couldn't create the bucket %s. Here's why: %s",
                    bucket_name,
                    client_error.response["Error"]["Message"],
                )

```

```

        raise

    def delete_bucket_and_objects(self, bucket_name: str) -> None:
        """
        Deletes a bucket and its objects.
        :param bucket_name: The name of the bucket.
        """
        try:
            # Delete the objects in the bucket first. This is required for a
            bucket to be deleted.
            paginator = self.s3_client.get_paginator("list_objects_v2")
            page_iterator = paginator.paginate(Bucket=bucket_name)
            for page in page_iterator:
                if "Contents" in page:
                    delete_keys = {
                        "Objects": [{"Key": obj["Key"]} for obj in
page["Contents"]]
                    }
                    response = self.s3_client.delete_objects(
                        Bucket=bucket_name, Delete=delete_keys
                    )
                    if "Errors" in response:
                        for error in response["Errors"]:
                            logging.error(
                                "Couldn't delete object %s. Here's why: %s",
                                error["Key"],
                                error["Message"],
                            )

                    self.s3_client.delete_bucket(Bucket=bucket_name)
        except ClientError as client_error:
            logging.error(
                "Couldn't delete the bucket %s. Here's why: %s",
                bucket_name,
                client_error.response["Error"]["Message"],
            )

    def put_object(self, bucket_name: str, object_key: str, content: str) ->
None:
        """
        Puts an object into a bucket.
        :param bucket_name: The name of the bucket.
        :param object_key: The key of the object.
        :param content: The content of the object.

```

```

        """
        try:
            self.s3_client.put_object(Body=content, Bucket=bucket_name,
Key=object_key)
        except ClientError as client_error:
            logging.error(
                "Couldn't put the object %s into bucket %s. Here's why: %s",
                object_key,
                bucket_name,
                client_error.response["Error"]["Message"],
            )
            raise

def list_objects(self, bucket: str) -> list[str]:
    """
    Lists objects in a bucket.
    :param bucket: The name of the bucket.
    :return: The list of objects in the bucket.
    """
    try:
        response = self.s3_client.list_objects_v2(Bucket=bucket)
        return response.get("Contents", [])
    except ClientError as client_error:
        logging.error(
            "Couldn't list objects in bucket %s. Here's why: %s",
            bucket,
            client_error.response["Error"]["Message"],
        )
        raise

def copy_object(
    self,
    source_bucket: str,
    source_key: str,
    destination_bucket: str,
    destination_key: str,
) -> None:
    """
    Copies an object from one bucket to another.
    :param source_bucket: The source bucket.
    :param source_key: The source key.
    :param destination_bucket: The destination bucket.
    :param destination_key: The destination key.
    :return: None
    """

```

```

        """
    try:
        self.s3_client.copy_object(
            CopySource={"Bucket": source_bucket, "Key": source_key},
            Bucket=destination_bucket,
            Key=destination_key,
        )
    except ClientError as client_error:
        logging.error(
            "Couldn't copy object %s from bucket %s to bucket %s. Here's why:
%s",
            source_key,
            source_bucket,
            destination_bucket,
            client_error.response["Error"]["Message"],
        )
        raise

def create_session(self, bucket_name: str) -> None:
    """
    Creates an express session.
    :param bucket_name: The name of the bucket.
    """
    try:
        self.s3_client.create_session(Bucket=bucket_name)
    except ClientError as client_error:
        logging.error(
            "Couldn't create the express session for bucket %s. Here's why:
%s",
            bucket_name,
            client_error.response["Error"]["Message"],
        )
        raise

def get_object(self, bucket_name: str, object_key: str) -> None:
    """
    Gets an object from a bucket.
    :param bucket_name: The name of the bucket.
    :param object_key: The key of the object.
    """
    try:
        self.s3_client.get_object(Bucket=bucket_name, Key=object_key)
    except ClientError as client_error:

```

```
logging.error(
    "Couldn't get the object %s from bucket %s. Here's why: %s",
    object_key,
    bucket_name,
    client_error.response["Error"]["Message"],
)
raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 다음 주제를 참조하세요.

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObject](#)
- [GetObject](#)
- [ListObjects](#)
- [PutObject](#)

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용한 S3 디렉터리 버킷 작업

다음 코드 예제에서는 AWS SDK를 통해 개별 S3 디렉터리 버킷 작업을 수행하는 방법을 보여줍니다. 각 예제에는 GitHub에 대한 링크가 포함되어 있습니다. 여기에서 코드 설정 및 실행에 대한 지침을 찾을 수 있습니다.

S3 디렉터리 버킷 API를 직접적으로 호출하는 이들 발체문은 컨텍스트에서 실행되어야 하는 더 큰 프로그램에서 발체한 코드입니다. [AWS SDK를 사용한 S3 디렉터리 버킷 시나리오](#)에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 [Amazon S3 디렉터리 버킷 API 참조](#)에서 확인하세요.

예제

- [AWS SDK와 함께 AbortMultipartUpload 사용](#)
- [AWS SDK와 함께 CompleteMultipartUpload 사용](#)
- [AWS SDK와 함께 CopyObject 사용](#)
- [AWS SDK와 함께 CreateBucket 사용](#)
- [AWS SDK와 함께 CreateMultipartUpload 사용](#)
- [AWS SDK와 함께 CreateSession 사용](#)
- [AWS SDK와 함께 DeleteBucket 사용](#)
- [AWS SDK와 함께 DeleteBucketEncryption 사용](#)
- [AWS SDK와 함께 DeleteBucketPolicy 사용](#)
- [AWS SDK와 함께 DeleteObject 사용](#)
- [AWS SDK와 함께 DeleteObjects 사용](#)
- [AWS SDK와 함께 GetBucketEncryption 사용](#)
- [AWS SDK와 함께 GetBucketPolicy 사용](#)
- [AWS SDK와 함께 GetObject 사용](#)
- [AWS SDK와 함께 GetObjectAttributes 사용](#)
- [AWS SDK와 함께 HeadBucket 사용](#)
- [AWS SDK와 함께 HeadObject 사용](#)
- [AWS SDK와 함께 ListDirectoryBuckets 사용](#)
- [AWS SDK와 함께 ListMultipartUploads 사용](#)
- [AWS SDK와 함께 ListObjectsV2 사용](#)
- [AWS SDK와 함께 ListParts 사용](#)
- [AWS SDK와 함께 PutBucketEncryption 사용](#)
- [AWS SDK와 함께 PutBucketPolicy 사용](#)
- [AWS SDK와 함께 PutObject 사용](#)
- [AWS SDK와 함께 UploadPart 사용](#)
- [AWS SDK와 함께 UploadPartCopy 사용](#)

AWS SDK와 함께 **AbortMultipartUpload** 사용

다음 코드 예시는 AbortMultipartUpload의 사용 방법을 보여 줍니다.

Java

SDK for Java 2.x

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에서 멀티파트 업로드를 중단합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Aborts a specific multipart upload for the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be uploaded
 * @param uploadId The upload ID of the multipart upload to abort
 * @return True if the multipart upload is successfully aborted, false
 * otherwise
 */
public static boolean abortDirectoryBucketMultipartUpload(S3Client s3Client,
String bucketName,
    String objectKey, String uploadId) {
    logger.info("Aborting multipart upload: {} for bucket: {}", uploadId,
bucketName);
    try {
        // Abort the multipart upload
```

```

        AbortMultipartUploadRequest abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .uploadId(uploadId)
    .build();

    s3Client.abortMultipartUpload(abortMultipartUploadRequest);
    logger.info("Aborted multipart upload: {} for object: {}", uploadId,
objectKey);
    return true;
} catch (S3Exception e) {
    logger.error("Failed to abort multipart upload: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
        e.awsErrorDetails().errorCode(), e);
    return false;
}
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [AbortMultipartUpload](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **CompleteMultipartUpload** 사용

다음 코드 예시는 CompleteMultipartUpload의 사용 방법을 보여 줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에서 멀티파트 업로드를 완료합니다.

```

import com.example.s3.util.S3DirectoryBucketUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

/**
 * This method completes the multipart upload request by collating all the
 * upload parts.
 *
 * @param s3Client    The S3 client used to interact with S3
 * @param bucketName  The name of the directory bucket
 * @param objectKey   The key (name) of the object to be uploaded
 * @param uploadId    The upload ID used to track the multipart upload
 * @param uploadParts The list of completed parts
 * @return True if the multipart upload is successfully completed, false
 *         otherwise
 */
public static boolean completeDirectoryBucketMultipartUpload(S3Client
s3Client, String bucketName, String objectKey,
    String uploadId, List<CompletedPart> uploadParts) {

```

```

        try {
            CompletedMultipartUpload completedMultipartUpload =
            CompletedMultipartUpload.builder()
                .parts(uploadParts)
                .build();

            CompleteMultipartUploadRequest completeMultipartUploadRequest =
            CompleteMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .uploadId(uploadId)
                .multipartUpload(completedMultipartUpload)
                .build();

            CompleteMultipartUploadResponse response =
            s3Client.completeMultipartUpload(completeMultipartUploadRequest);
            logger.info("Multipart upload completed. ETag: {}", response.eTag());
            return true;
        } catch (S3Exception e) {
            logger.error("Failed to complete multipart upload: {} - Error code:
            {}", e.awsErrorDetails().errorMessage(),
                e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [CompleteMultipartUpload](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **CopyObject** 사용

다음 코드 예시는 CopyObject의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서 컨텍스트 내 이 작업을 확인할 수 있습니다.

- [기본 사항 알아보기](#)

Java

SDK for Java 2.x

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷의 객체를 다른 디렉터리 버킷으로 복사합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Copies an object from one S3 general purpose bucket to one S3 directory
 * bucket.
 *
 * @param s3Client      The S3 client used to interact with S3
 * @param sourceBucket  The name of the source bucket
 * @param objectKey     The key (name) of the object to be copied
 * @param targetBucket  The name of the target bucket
 */
public static void copyDirectoryBucketObject(S3Client s3Client, String
sourceBucket, String objectKey,
```

```

        String targetBucket) {
            logger.info("Copying object: {} from bucket: {} to bucket: {}",
                objectKey, sourceBucket, targetBucket);

            try {
                // Create a CopyObjectRequest
                CopyObjectRequest copyReq = CopyObjectRequest.builder()
                    .sourceBucket(sourceBucket)
                    .sourceKey(objectKey)
                    .destinationBucket(targetBucket)
                    .destinationKey(objectKey)
                    .build();

                // Copy the object
                CopyObjectResponse copyRes = s3Client.copyObject(copyReq);
                logger.info("Successfully copied {} from bucket {} into bucket {}.
CopyObjectResponse: {}",
                    objectKey, sourceBucket, targetBucket,
                    copyRes.copyObjectResult().toString());

            } catch (S3Exception e) {
                logger.error("Failed to copy object: {} - Error code: {}",
                    e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
                throw e;
            }
        }
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [CopyObject](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **CreateBucket** 사용

다음 코드 예시는 CreateBucket의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서 컨텍스트 내 이 작업을 확인할 수 있습니다.

- [기본 사항 알아보기](#)

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

S3 디렉터리 버킷을 만듭니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Creates a new S3 directory bucket in a specified Zone (For example, a
 * specified Availability Zone in this code example).
 *
 * @param s3Client The S3 client used to create the bucket
 * @param bucketName The name of the bucket to be created
 * @param zone The region where the bucket will be created
 * @throws S3Exception if there's an error creating the bucket
 */
public static void createDirectoryBucket(S3Client s3Client, String
bucketName, String zone) throws S3Exception {
    logger.info("Creating bucket: {}", bucketName);
```

```

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
    .location(LocationInfo.builder()
        .type(LocationType.AVAILABILITY_ZONE)
        .name(zone).build())
    .bucket(BucketInfo.builder()
        .type(BucketType.DIRECTORY)
        .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
        .build())
    .build();
    try {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .createBucketConfiguration(bucketConfiguration).build();
        CreateBucketResponse response = s3Client.createBucket(bucketRequest);
        logger.info("Bucket created successfully with location: {}",
response.location());
    } catch (S3Exception e) {
        logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [CreateBucket](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **CreateMultipartUpload** 사용

다음 코드 예시는 CreateMultipartUpload의 사용 방법을 보여 줍니다.

Java

SDK for Java 2.x

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에 멀티파트 업로드를 생성합니다.

```
import com.example.s3.util.S3DirectoryBucketUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * This method creates a multipart upload request that generates a unique
 * upload
 * ID used to track
 * all the upload parts.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be uploaded
 * @return The upload ID used to track the multipart upload
 */
public static String createDirectoryBucketMultipartUpload(S3Client s3Client,
String bucketName, String objectKey) {
    logger.info("Creating multipart upload for object: {} in bucket: {}",
objectKey, bucketName);

    try {
        // Create a CreateMultipartUploadRequest
```

```

        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .build();

        // Initiate the multipart upload
        CreateMultipartUploadResponse response =
s3Client.createMultipartUpload(createMultipartUploadRequest);
        String uploadId = response.uploadId();
        logger.info("Multipart upload initiated. Upload ID: {}", uploadId);
        return uploadId;

    } catch (S3Exception e) {
        logger.error("Failed to create multipart upload: {} - Error code:
{}", e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [CreateMultipartUpload](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **CreateSession** 사용

다음 코드 예시는 CreateSession의 사용 방법을 보여 줍니다.

Python

SDK for Python(Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

class S3ExpressWrapper:
    """Encapsulates Amazon S3 Express One Zone actions using the client
    interface."""

    def __init__(self, s3_client: Any) -> None:
        """
        Initializes the S3ExpressWrapper with an S3 client.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-
level
                           access to AWS S3 services.
        """
        self.s3_client = s3_client

    @classmethod
    def from_client(cls) -> "S3ExpressWrapper":
        """
        Creates an S3ExpressWrapper instance with a default s3 client.

        :return: An instance of S3ExpressWrapper initialized with the default S3
client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def create_session(self, bucket_name: str) -> None:
        """
        Creates an express session.
        :param bucket_name: The name of the bucket.
        """
        try:
            self.s3_client.create_session(Bucket=bucket_name)
        except ClientError as client_error:
            logging.error(
                "Couldn't create the express session for bucket %s. Here's why:
%s",
                bucket_name,
                client_error.response["Error"]["Message"],
            )
            raise

```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 [CreateSession](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **DeleteBucket** 사용

다음 코드 예시는 DeleteBucket의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서 컨텍스트 내 이 작업을 확인할 수 있습니다.

- [기본 사항 알아보기](#)

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

S3 디렉터리 버킷을 삭제합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

/**
 * Deletes the specified S3 directory bucket.
```

```
*
* @param s3Client The S3 client used to interact with S3
* @param bucketName The name of the directory bucket to delete
*/
public static void deleteDirectoryBucket(S3Client s3Client, String
bucketName) {
    logger.info("Deleting bucket: {}", bucketName);

    try {
        // Create a DeleteBucketRequest
        DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
                        .bucket(bucketName)
                        .build();

        // Delete the bucket
        s3Client.deleteBucket(deleteBucketRequest);
        logger.info("Successfully deleted bucket: {}", bucketName);

    } catch (S3Exception e) {
        logger.error("Failed to delete bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                        e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [DeleteBucket](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **DeleteBucketEncryption** 사용

다음 코드 예시는 DeleteBucketEncryption의 사용 방법을 보여 줍니다.

Java

SDK for Java 2.x

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷의 암호화 구성을 삭제합니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Deletes the encryption configuration from an S3 bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 */
public static void deleteDirectoryBucketEncryption(S3Client s3Client, String
bucketName) {
    DeleteBucketEncryptionRequest deleteRequest =
DeleteBucketEncryptionRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        s3Client.deleteBucketEncryption(deleteRequest);
        logger.info("Bucket encryption deleted for bucket: {}", bucketName);
    } catch (S3Exception e) {
        logger.error("Failed to delete bucket encryption: {} - Error code:
{}", e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
    }
}
```

```
        throw e;
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [DeleteBucketEncryption](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **DeleteBucketPolicy** 사용

다음 코드 예시는 DeleteBucketPolicy의 사용 방법을 보여 줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에 대한 버킷 정책을 삭제합니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;

/**
 * Deletes the bucket policy for the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 */
public static void deleteDirectoryBucketPolicy(S3Client s3Client, String
bucketName) {
    logger.info("Deleting policy for bucket: {}", bucketName);

    try {
        // Create a DeleteBucketPolicyRequest
        DeleteBucketPolicyRequest deletePolicyReq =
DeleteBucketPolicyRequest.builder()
            .bucket(bucketName)
            .build();

        // Delete the bucket policy
        s3Client.deleteBucketPolicy(deletePolicyReq);
        logger.info("Successfully deleted bucket policy");

    } catch (S3Exception e) {
        logger.error("Failed to delete bucket policy: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [DeleteBucketPolicy](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **DeleteObject** 사용

다음 코드 예시는 DeleteObject의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서 컨텍스트 내 이 작업을 확인할 수 있습니다.

- [기본 사항 알아보기](#)

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷의 객체를 삭제합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Deletes an object from the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
```

```
    * @param objectKey The key (name) of the object to be deleted
    */
    public static void deleteDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey) {
        logger.info("Deleting object: {} from bucket: {}", objectKey,
bucketName);

        try {
            // Create a DeleteObjectRequest
            DeleteObjectRequest deleteObjectRequest =
DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            // Delete the object
            s3Client.deleteObject(deleteObjectRequest);
            logger.info("Object {} has been deleted", objectKey);

        } catch (S3Exception e) {
            logger.error("Failed to delete object: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [DeleteObject](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **DeleteObjects** 사용

다음 코드 예시는 DeleteObjects의 사용 방법을 보여 줍니다.

Java

SDK for Java 2.x

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에서 여러 객체를 삭제합니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectsResponse;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.net.URISyntaxException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Deletes multiple objects from the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKeys The list of keys (names) of the objects to be deleted
```

```
    */
    public static void deleteDirectoryBucketObjects(S3Client s3Client, String
bucketName, List<String> objectKeys) {
        logger.info("Deleting objects from bucket: {}", bucketName);

        try {
            // Create a list of ObjectIdentifier.
            List<ObjectIdentifier> identifiers = objectKeys.stream()
                .map(key -> ObjectIdentifier.builder().key(key).build())
                .toList();

            Delete delete = Delete.builder()
                .objects(identifiers)
                .build();

            DeleteObjectsRequest deleteObjectsRequest =
DeleteObjectsRequest.builder()
                .bucket(bucketName)
                .delete(delete)
                .build();

            DeleteObjectsResponse deleteObjectsResponse =
s3Client.deleteObjects(deleteObjectsRequest);
            deleteObjectsResponse.deleted().forEach(deleted ->
logger.info("Deleted object: {}", deleted.key()));

        } catch (S3Exception e) {
            logger.error("Failed to delete objects: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [DeleteObjects](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **GetBucketEncryption** 사용

다음 코드 예시는 GetBucketEncryption의 사용 방법을 보여 줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷의 암호화 구성을 가져옵니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Retrieves the encryption configuration for an S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @return The type of server-side encryption applied to the bucket (e.g.,
 *         AES256, aws:kms)
 */
public static String getDirectoryBucketEncryption(S3Client s3Client, String
bucketName) {
    try {
        // Create a GetBucketEncryptionRequest
```

```
GetBucketEncryptionRequest getRequest =
GetBucketEncryptionRequest.builder()
    .bucket(bucketName)
    .build();

// Retrieve the bucket encryption configuration
GetBucketEncryptionResponse response =
s3Client.getBucketEncryption(getRequest);
ServerSideEncryptionRule rule =
response.getServerSideEncryptionConfiguration().rules().get(0);

String encryptionType =
rule.applyServerSideEncryptionByDefault().sseAlgorithmAsString();
logger.info("Bucket encryption algorithm: {}", encryptionType);
logger.info("KMS Customer Managed Key ID: {}",
rule.applyServerSideEncryptionByDefault().kmsMasterKeyID());
logger.info("Bucket Key Enabled: {}", rule.bucketKeyEnabled());

return encryptionType;
} catch (S3Exception e) {
    logger.error("Failed to get bucket encryption: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
        e.awsErrorDetails().errorCode(), e);
    throw e;
}
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [GetBucketEncryption](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **GetBucketPolicy** 사용

다음 코드 예시는 GetBucketPolicy의 사용 방법을 보여 줍니다.

Java

SDK for Java 2.x

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷의 정책을 가져옵니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

/**
 * Retrieves the bucket policy for the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @return The bucket policy text
 */
public static String getDirectoryBucketPolicy(S3Client s3Client, String
bucketName) {
    logger.info("Getting policy for bucket: {}", bucketName);

    try {
        // Create a GetBucketPolicyRequest
        GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
            .bucket(bucketName)
```

```
        .build();

        // Retrieve the bucket policy
        GetBucketPolicyResponse response =
s3Client.getBucketPolicy(policyReq);

        // Print and return the policy text
        String policyText = response.policy();
        logger.info("Bucket policy: {}", policyText);
        return policyText;

    } catch (S3Exception e) {
        logger.error("Failed to get bucket policy: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [GetBucketPolicy](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **GetObject** 사용

다음 코드 예시는 GetObject의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- [기본 사항 알아보기](#)
- [미리 서명된 URL을 만들어 객체 가져오기](#)

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에서 객체를 가져옵니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.charset.StandardCharsets;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Retrieves an object from the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be retrieved
 * @return The retrieved object as a ResponseInputStream
 */
```

```
public static boolean getDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey) {
    logger.info("Retrieving object: {} from bucket: {}", objectKey,
bucketName);

    try {
        // Create a GetObjectRequest
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .key(objectKey)
            .bucket(bucketName)
            .build();

        // Retrieve the object as bytes
        ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
        byte[] data = objectBytes.asByteArray();

        // Print object contents to console
        String objectContent = new String(data, StandardCharsets.UTF_8);
        logger.info("Object contents: \n{}", objectContent);

        return true;

    } catch (S3Exception e) {
        logger.error("Failed to retrieve object: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        return false;
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [GetObject](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **GetObjectAttributes** 사용

다음 코드 예시는 GetObjectAttributes의 사용 방법을 보여 줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에서 객체 속성을 가져옵니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesResponse;
import software.amazon.awssdk.services.s3.model.ObjectAttributes;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Retrieves attributes for an object in the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to retrieve attributes for
 * @return True if the object attributes are successfully retrieved, false
 *         otherwise
 */
```

```

    public static boolean getDirectoryBucketObjectAttributes(S3Client s3Client,
String bucketName, String objectKey) {
        logger.info("Retrieving attributes for object: {} from bucket: {}",
objectKey, bucketName);

        try {
            // Create a GetObjectAttributesRequest
            GetObjectAttributesRequest getObjectAttributesRequest =
GetObjectAttributesRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .objectAttributes(ObjectAttributes.E_TAG,
ObjectAttributes.STORAGE_CLASS,
                    ObjectAttributes.OBJECT_SIZE)
                .build();

            // Retrieve the object attributes
            GetObjectAttributesResponse response =
s3Client.getObjectAttributes(getObjectAttributesRequest);
            logger.info("Attributes for object {}:", objectKey);
            logger.info("ETag: {}", response.eTag());
            logger.info("Storage Class: {}", response.storageClass());
            logger.info("Object Size: {}", response.objectSize());
            return true;

        } catch (S3Exception e) {
            logger.error("Failed to retrieve object attributes: {} - Error code:
{}",
                e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [GetObjectAttributes](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 HeadBucket 사용

다음 코드 예시는 HeadBucket의 사용 방법을 보여 줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

지정된 S3 디렉터리 버킷이 존재하고 액세스할 수 있는지 확인합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Checks if the specified S3 directory bucket exists and is accessible.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket to check
 * @return True if the bucket exists and is accessible, false otherwise
 */
public static boolean headDirectoryBucket(S3Client s3Client, String
bucketName) {
    logger.info("Checking if bucket exists: {}", bucketName);

    try {
        // Create a HeadBucketRequest
        HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
            .bucket(bucketName)
```

```

        .build();
        // If the bucket doesn't exist, the following statement throws
        NoSuchBucketException,
        // which is a subclass of S3Exception.
        s3Client.headBucket(headBucketRequest);
        logger.info("Amazon S3 directory bucket: \"{}\" found.", bucketName);
        return true;

    } catch (S3Exception e) {
        logger.error("Failed to access bucket: {} - Error code: {}",
            e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [HeadBucket](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **HeadObject** 사용

다음 코드 예시는 HeadObject의 사용 방법을 보여 줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에 있는 객체의 메타데이터를 가져옵니다.

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Retrieves metadata for an object in the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to retrieve metadata for
 * @return True if the object exists, false otherwise
 */
public static boolean headDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey) {
    logger.info("Retrieving metadata for object: {} from bucket: {}",
objectKey, bucketName);

    try {
        // Create a HeadObjectRequest
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        // Retrieve the object metadata
        HeadObjectResponse response = s3Client.headObject(headObjectRequest);
        logger.info("Amazon S3 object: \"{}\" found in bucket: \"{}\" with
ETag: \"{}\"", objectKey, bucketName,
            response.eTag());
        logger.info("Content-Type: {}", response.contentType());
        logger.info("Content-Length: {}", response.contentLength());
    }
```

```

        logger.info("Last Modified: {}", response.lastModified());
        return true;

    } catch (S3Exception e) {
        logger.error("Failed to retrieve object metadata: {} - Error code:
        {}", e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        return false;
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [HeadObject](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **ListDirectoryBuckets** 사용

다음 코드 예시는 ListDirectoryBuckets의 사용 방법을 보여 줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

모든 디렉터리 버킷을 나열합니다.

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;

```



```

import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.List;
import java.util.UUID;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Lists all S3 directory buckets and no general purpose buckets.
 *
 * @param s3Client The S3 client used to interact with S3
 * @return A list of bucket names
 */
public static List<String> listDirectoryBuckets(S3Client s3Client) {
    logger.info("Listing all directory buckets");

    try {
        // Create a ListBucketsRequest
        ListDirectoryBucketsRequest listDirectoryBucketsRequest =
            ListDirectoryBucketsRequest.builder().build();

        // Retrieve the list of buckets
        ListDirectoryBucketsResponse response =
            s3Client.listDirectoryBuckets(listDirectoryBucketsRequest);

        // Extract bucket names
        List<String> bucketNames = response.buckets().stream()
            .map(Bucket::name)
            .collect(Collectors.toList());

        return bucketNames;
    } catch (S3Exception e) {
        logger.error("Failed to list buckets: {} - Error code: {}",
            e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode());
        throw e;
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [ListDirectoryBuckets](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **ListMultipartUploads** 사용

다음 코드 예시는 ListMultipartUploads의 사용 방법을 보여 줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에 있는 멀티파트 업로드를 나열합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
    com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
```

```
import static
com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

/**
 * Lists multipart uploads for the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @return A list of MultipartUpload objects representing the multipart
uploads
 */
public static List<MultipartUpload>
listDirectoryBucketMultipartUploads(S3Client s3Client, String bucketName) {
    logger.info("Listing in-progress multipart uploads for bucket: {}",
bucketName);

    try {
        // Create a ListMultipartUploadsRequest
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
            .bucket(bucketName)
            .build();

        // List the multipart uploads
        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();
        for (MultipartUpload upload : uploads) {
            logger.info("In-progress multipart upload: Upload ID: {}, Key:
{}, Initiated: {}", upload.uploadId(),
                upload.key(), upload.initiated());
        }
        return uploads;

    } catch (S3Exception e) {
        logger.error("Failed to list multipart uploads: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode());
        return List.of(); // Return an empty list if an exception is thrown
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [ListMultipartUploads](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **ListObjectsV2** 사용

다음 코드 예시는 ListObjectsV2의 사용 방법을 보여 줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에 있는 객체를 나열합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;

import java.nio.file.Path;
import java.util.List;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Lists objects in the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @return A list of object keys in the bucket
 */
public static List<String> listDirectoryBucketObjectsV2(S3Client s3Client,
String bucketName) {
    logger.info("Listing objects in bucket: {}", bucketName);

    try {
        // Create a ListObjectsV2Request
        ListObjectsV2Request listObjectsV2Request =
ListObjectsV2Request.builder()
                        .bucket(bucketName)
                        .build();

        // Retrieve the list of objects
        ListObjectsV2Response response =
s3Client.listObjectsV2(listObjectsV2Request);

        // Extract and return the object keys
        return response.contents().stream()
                        .map(S3Object::key)
                        .collect(Collectors.toList());

    } catch (S3Exception e) {
        logger.error("Failed to list objects: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                        e.awsErrorDetails().errorCode());
        throw e;
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [ListObjectsV2](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **ListParts** 사용

다음 코드 예시는 ListParts의 사용 방법을 보여 줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에 있는 멀티파트 업로드의 일부를 나열합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListPartsRequest;
import software.amazon.awssdk.services.s3.model.ListPartsResponse;
import software.amazon.awssdk.services.s3.model.Part;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
    com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
```

```
import static
com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

/**
 * Lists the parts of a multipart upload for the specified S3 directory
 * bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object being uploaded
 * @param uploadId The upload ID used to track the multipart upload
 * @return A list of Part representing the parts of the multipart upload
 */
public static List<Part> listDirectoryBucketMultipartUploadParts(S3Client
s3Client, String bucketName,
    String objectKey, String uploadId) {
    logger.info("Listing parts for object: {} in bucket: {}", objectKey,
bucketName);

    try {
        // Create a ListPartsRequest
        ListPartsRequest listPartsRequest = ListPartsRequest.builder()
            .bucket(bucketName)
            .uploadId(uploadId)
            .key(objectKey)
            .build();

        // List the parts of the multipart upload
        ListPartsResponse response = s3Client.listParts(listPartsRequest);
        List<Part> parts = response.parts();
        for (Part part : parts) {
            logger.info("Uploaded part: Part number = \"{}\", etag = {}",
part.partNumber(), part.eTag());
        }
        return parts;

    } catch (S3Exception e) {
        logger.error("Failed to list parts: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode());
        return List.of(); // Return an empty list if an exception is thrown
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [ListParts](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **PutBucketEncryption** 사용

다음 코드 예시는 PutBucketEncryption의 사용 방법을 보여 줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에 버킷 암호화를 설정합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryption;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionByDefault;
import
    software.amazon.awssdk.services.s3.model.ServerSideEncryptionConfiguration;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsClient;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsKey;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
```



```
import static com.example.s3.util.S3DirectoryBucketUtils.scheduleKeyDeletion;

/**
 * Sets the default encryption configuration for an S3 bucket as SSE-KMS.
 *
 * @param s3Client    The S3 client used to interact with S3
 * @param bucketName  The name of the directory bucket
 * @param kmsKeyId    The ID of the customer-managed KMS key
 */
public static void putDirectoryBucketEncryption(S3Client s3Client, String
bucketName, String kmsKeyId) {
    // Define the default encryption configuration to use SSE-KMS. For
    directory
    // buckets, AWS managed KMS keys aren't supported. Only customer-managed
    keys
    // are supported.
    ServerSideEncryptionByDefault encryptionByDefault =
    ServerSideEncryptionByDefault.builder()
        .sseAlgorithm(ServerSideEncryption.AWS_KMS)
        .kmsMasterKeyID(kmsKeyId)
        .build();

    // Create a server-side encryption rule to apply the default encryption
    // configuration. For directory buckets, the bucketKeyEnabled field is
    enforced
    // to be true.
    ServerSideEncryptionRule rule = ServerSideEncryptionRule.builder()
        .bucketKeyEnabled(true)
        .applyServerSideEncryptionByDefault(encryptionByDefault)
        .build();

    // Create the server-side encryption configuration for the bucket
    ServerSideEncryptionConfiguration encryptionConfiguration =
    ServerSideEncryptionConfiguration.builder()
        .rules(rule)
        .build();

    // Create the PutBucketEncryption request
    PutBucketEncryptionRequest putRequest =
    PutBucketEncryptionRequest.builder()
        .bucket(bucketName)
        .serverSideEncryptionConfiguration(encryptionConfiguration)
        .build();
```

```
// Set the bucket encryption
try {
    s3Client.putBucketEncryption(putRequest);
    logger.info("SSE-KMS Bucket encryption configuration set for the
directory bucket: {}", bucketName);
} catch (S3Exception e) {
    logger.error("Failed to set bucket encryption: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode());
    throw e;
}
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [PutBucketEncryption](#)을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **PutBucketPolicy** 사용

다음 코드 예시는 PutBucketPolicy의 사용 방법을 보여 줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에 버킷 정책을 적용합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
```

```

import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;

/**
 * Sets the following bucket policy for the specified S3 directory bucket.
 * <pre>
 * {
 *     "Version": "2012-10-17",
 *     "Statement": [
 *         {
 *             "Sid": "AdminPolicy",
 *             "Effect": "Allow",
 *             "Principal": {
 *                 "AWS": "arn:aws:iam::<ACCOUNT_ID>:root"
 *             },
 *             "Action": "s3express:*",
 *             "Resource": "arn:aws:s3express:us-west-2:<ACCOUNT_ID>:bucket/
<DIR_BUCKET_NAME>
 *         }
 *     ]
 * }
 * </pre>
 * This policy grants all S3 directory bucket actions to identities in the
 * same account as the bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param policyText The policy text to be applied
 */
public static void putDirectoryBucketPolicy(S3Client s3Client, String
bucketName, String policyText) {
    logger.info("Setting policy on bucket: {}", bucketName);
    logger.info("Policy: {}", policyText);

    try {
        PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
            .bucket(bucketName)
            .policy(policyText)
            .build();
    }

```

```

        s3Client.putBucketPolicy(policyReq);
        logger.info("Bucket policy set successfully!");

    } catch (S3Exception e) {
        logger.error("Failed to set bucket policy: {} - Error code: {}",
            e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [PutBucketPolicy](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **PutObject** 사용

다음 코드 예시는 PutObject의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서 컨텍스트 내 이 작업을 확인할 수 있습니다.

- [기본 사항 알아보기](#)

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에 객체를 업로드합니다.

```
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsErrorDetails;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.UncheckedIOException;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

/**
 * Puts an object into the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be placed in the bucket
 * @param filePath The path of the file to be uploaded
 */
public static void putDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey, Path filePath) {
    logger.info("Putting object: {} into bucket: {}", objectKey, bucketName);

    try {
        // Create a PutObjectRequest
        PutObjectRequest putObj = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        // Upload the object
        s3Client.putObject(putObj, filePath);
        logger.info("Successfully placed {} into bucket {}", objectKey,
bucketName);

    } catch (UncheckedIOException e) {
        throw S3Exception.builder().message("Failed to read the file: " +
e.getMessage()).cause(e)
    }
}
```

```

        .awsErrorDetails(AwsErrorDetails.builder()
            .errorCode("ClientSideException:FailedToReadFile")
            .errorMessage(e.getMessage())
            .build())
        .build();
    } catch (S3Exception e) {
        logger.error("Failed to put object: {}", e.getMessage(), e);
        throw e;
    }
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [PutObject](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **UploadPart** 사용

다음 코드 예시는 UploadPart의 사용 방법을 보여 줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에 멀티파트 업로드의 일부를 업로드합니다.

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;

```

```

import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.nio.ByteBuffer;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;

import static
    com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

/**
 * This method creates part requests and uploads individual parts to S3.
 * While it uses the UploadPart API to upload a single part, it does so
 * sequentially to handle multiple parts of a file, returning all the
 * completed
 * parts.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be uploaded
 * @param uploadId The upload ID used to track the multipart upload
 * @param filePath The path to the file to be uploaded
 * @return A list of uploaded parts
 * @throws IOException if an I/O error occurs
 */
public static List<CompletedPart> multipartUploadForDirectoryBucket(S3Client
s3Client, String bucketName,
    String objectKey, String uploadId, Path filePath) throws IOException
{
    logger.info("Uploading parts for object: {} in bucket: {}", objectKey,
bucketName);

    int partNumber = 1;

```

```
List<CompletedPart> uploadedParts = new ArrayList<>();
ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

// Read the local file, break down into chunks and process
try (RandomAccessFile file = new RandomAccessFile(filePath.toFile(),
"r")) {
    long fileSize = file.length();
    int position = 0;

    // Sequentially upload parts of the file
    while (position < fileSize) {
        file.seek(position);
        int read = file.getChannel().read(bb);

        bb.flip(); // Swap position and limit before reading from the
buffer

        UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .uploadId(uploadId)
            .partNumber(partNumber)
            .build();

        UploadPartResponse partResponse = s3Client.uploadPart(
            uploadPartRequest,
            RequestBody.fromByteBuffer(bb));

        // Build the uploaded part
        CompletedPart uploadedPart = CompletedPart.builder()
            .partNumber(partNumber)
            .eTag(partResponse.eTag())
            .build();

        // Add the uploaded part to the list
        uploadedParts.add(uploadedPart);

        // Log to indicate the part upload is done
        logger.info("Uploaded part number: {} with ETag: {}", partNumber,
partResponse.eTag());

        bb.clear();
        position += read;
        partNumber++;
    }
}
```



```

    } catch (S3Exception e) {
        logger.error("Failed to list parts: {} - Error code: {}",
            e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode());
        throw e;
    }
    return uploadedParts;
}

```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [UploadPart](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 UploadPartCopy 사용

다음 코드 예시는 UploadPartCopy의 사용 방법을 보여 줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

소스 객체 크기를 기반으로 복사할 부분을 나눈 다음 각 부분을 디렉터리 버킷에 복사합니다.

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartCopyRequest;

```

```

import software.amazon.awssdk.services.s3.model.UploadPartCopyResponse;

import java.io.IOException;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;

import static
    com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static
    com.example.s3.util.S3DirectoryBucketUtils.completeDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

/**
 * Creates copy parts based on source object size and copies over individual
 * parts.
 *
 * @param s3Client      The S3 client used to interact with S3
 * @param sourceBucket  The name of the source bucket
 * @param sourceKey     The key (name) of the source object
 * @param destinationBucket The name of the destination bucket
 * @param destinationKey The key (name) of the destination object
 * @param uploadId      The upload ID used to track the multipart upload
 * @return A list of completed parts
 */
public static List<CompletedPart>
multipartUploadCopyForDirectoryBucket(S3Client s3Client, String sourceBucket,
    String sourceKey, String destinationBucket, String destinationKey,
    String uploadId) {
    // Get the object size to track the end of the copy operation
    HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
        .bucket(sourceBucket)
        .key(sourceKey)
        .build();

```

```

        HeadObjectResponse headObjectResponse =
s3Client.headObject(headObjectRequest);
        long objectSize = headObjectResponse.contentLength();

        logger.info("Source Object size: {}", objectSize);

        // Copy the object using 20 MB parts
        long partSize = 20 * 1024 * 1024; // 20 MB
        long bytePosition = 0;
        int partNum = 1;
        List<CompletedPart> uploadedParts = new ArrayList<>();

        while (bytePosition < objectSize) {
            long lastByte = Math.min(bytePosition + partSize - 1, objectSize -
1);

            logger.info("Part Number: {}, Byte Position: {}, Last Byte: {}",
partNum, bytePosition, lastByte);

            try {
                UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                    .sourceBucket(sourceBucket)
                    .sourceKey(sourceKey)
                    .destinationBucket(destinationBucket)
                    .destinationKey(destinationKey)
                    .uploadId(uploadId)
                    .copySourceRange("bytes=" + bytePosition + "-" +
lastByte)

                    .partNumber(partNum)
                    .build();

                UploadPartCopyResponse uploadPartCopyResponse =
s3Client.uploadPartCopy(uploadPartCopyRequest);

                CompletedPart part = CompletedPart.builder()
                    .partNumber(partNum)
                    .eTag(uploadPartCopyResponse.copyPartResult().eTag())
                    .build();
                uploadedParts.add(part);

                bytePosition += partSize;
                partNum++;
            } catch (S3Exception e) {
                logger.error("Failed to copy part number {}: {} - Error code:
{}", partNum,

```

```
        e.awsErrorDetails().errorMessage(),
        e.awsErrorDetails().errorCode());
        throw e;
    }
}

return uploadedParts;
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [UploadPartCopy](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용한 S3 디렉터리 버킷 시나리오

다음 코드 예제에서는 S3 디렉터리 버킷에서 AWS SDK를 사용하여 일반적인 시나리오를 구현하는 방법을 보여줍니다. 이러한 시나리오에서는 S3 디렉터리 버킷 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 작업을 수행하는 방법을 보여줍니다. 각 시나리오에는 전체 소스 코드에 대한 링크가 포함되어 있습니다. 여기에서 코드를 설정 및 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시나리오는 컨텍스트에 맞는 서비스 작업을 이해하는 데 도움이 되도록 중급 수준의 경험을 대상으로 합니다.

예시

- [AWS SDK를 사용하여 객체를 가져오기 위해 Amazon S3 디렉터리 버킷에 대해 미리 서명된 URL 생성](#)

AWS SDK를 사용하여 객체를 가져오기 위해 Amazon S3 디렉터리 버킷에 대해 미리 서명된 URL 생성

다음 코드 예제에서는 S3 디렉터리 버킷에 대해 미리 서명된 URL을 만들고 객체를 가져오는 방법을 보여줍니다.

Java

SDK for Java 2.x

 Note

GitHub에 더 많은 내용이 있습니다. [AWS코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

S3 디렉터리 버킷의 객체에 액세스하기 위해 미리 서명된 GET URL을 생성합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
    software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;

import java.nio.file.Path;
import java.time.Duration;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Presigner;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Generates a presigned URL for accessing an object in the specified S3
 * directory bucket.
 */
```

```
    * @param s3Presigner The S3 presigner client used to generate the presigned
    URL
    * @param bucketName The name of the directory bucket
    * @param objectKey The key (name) of the object to access
    * @return A presigned URL for accessing the specified object
    */
    public static String generatePresignedGetURLForDirectoryBucket(S3Presigner
s3Presigner, String bucketName,
        String objectKey) {
        logger.info("Generating presigned URL for object: {} in bucket: {}",
objectKey, bucketName);

        try {
            // Create a GetObjectRequest
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            // Create a GetObjectPresignRequest
            GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(10)) // Presigned URL
valid for 10 minutes
                .getObjectRequest(getObjectRequest)
                .build();

            // Generate the presigned URL
            PresignedGetObjectRequest presignedGetObjectRequest =
s3Presigner.presignGetObject(getObjectPresignRequest);

            // Get the presigned URL
            String presignedURL = presignedGetObjectRequest.url().toString();
            logger.info("Presigned URL: {}", presignedURL);
            return presignedURL;

        } catch (S3Exception e) {
            logger.error("Failed to generate presigned URL: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 [GetObject](#)를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 [AWS SDK를 사용하여 Amazon S3에서 개발](#)을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

요청 인증(AWS 서명 버전 4)

주제

- [인증 방법](#)
- [서명 요청 소개](#)
- [요청 인증: Authorization 헤더 사용\(AWS 서명 버전 4\)](#)
- [요청 인증: 쿼리 파라미터 사용\(AWS 서명 버전 4\)](#)
- [예제: AWS 서명 버전 4의 서명 계산](#)
- [요청 인증: POST를 사용한 브라우저 기반 업로드\(AWS 서명 버전 4\)](#)
- [Amazon S3 서명 버전 4 인증 관련 정책 키](#)

Amazon S3와의 모든 상호 작용은 인증을 거치거나 익명으로 할 수 있습니다. 이 섹션에서는 AWS 서명 버전 4 알고리즘을 사용한 요청 인증에 대해 설명합니다.

Note

AWS SDK([샘플 코드 및 라이브러리](#) 참조)를 사용하여 API 요청을 전송하는 경우, SDK 클라이언트는 사용자가 제공한 액세스 키를 사용하여 요청을 인증하므로 이 섹션은 읽을 필요가 없습니다. 특별한 이유가 없다면 항상 AWS SDK를 사용해야 합니다. 두 서명 버전을 모두 지원하는 리전에서는 AWS SDK에 특정 서명 버전을 사용하도록 요청할 수 있습니다. 자세한 내용은 [AWS SDK를 사용하여 인증된 요청 보내기](#) 섹션을 참조하세요. 사용자 지정 클라이언트에서 AWS 서명 버전 4 알고리즘을 구현하는 경우에만 이 섹션을 읽어야 합니다.

AWS 서명 버전 4를 사용한 인증은 요청에 서명하도록 선택한 방법에 따라 다음 중 일부 또는 전부를 제공합니다.

- 요청자 자격 증명 확인 - 인증된 요청에는 액세스 키(액세스 키 ID, 시크릿 액세스 키)를 사용하여 생성한 서명이 필요합니다. 액세스 키를 얻는 방법에 대한 자세한 내용은 AWS 일반 참조의 [보안 자격 증명 이해 및 가져오기](#)를 참조하세요. 임시 보안 인증 정보를 사용하는 경우 서명 계산에 보안 토큰도 필요합니다. 자세한 내용은 IAM 사용 설명서의 [임시 보안 자격 증명 요청](#) 섹션을 참조하세요.
- 전송 중 데이터 보호 - 전송 중 요청의 변조를 방지하기 위해 요청 요소 중 일부를 사용하여 요청 서명을 계산합니다. Amazon S3는 요청을 수신하면 동일한 요청 요소를 사용하여 서명을 계산합니다.

Amazon S3에서 수신한 요청 구성 요소가 서명을 계산하는 데 사용된 구성 요소와 일치하지 않는 경우 Amazon S3는 요청을 거부합니다.

- 요청의 서명된 부분이 재사용되지 않도록 보호 – 요청의 서명된 부분(AWS 서명 사용)은 요청의 타임스탬프로부터 15분까지 유효합니다. 서명된 요청에 액세스할 수 있는 권한 없는 당사자는 이 15분 동안 요청의 유효성에 영향을 주지 않고 요청의 서명되지 않은 부분을 수정할 수 있습니다. 따라서 요청 헤더 및 본문에 서명하고, Amazon S3에 HTTPS 요청을 전송하고, AWS 정책에서 `s3:x-amz-content-sha256` 조건 키([Amazon S3 서명 버전 4 인증 관련 정책 키](#) 참조)를 사용하여 사용자에게 Amazon S3 요청 본문에 서명하도록 요구하여 보호를 극대화하는 것이 좋습니다.

Note

Amazon S3는 모든 AWS 리전에서 AWS 서비스에 대한 인바운드 API 요청을 인증하기 위한 프로토콜인 서명 버전 4를 지원합니다. 현재 2014년 1월 30일 이전에 생성된 AWS 리전이라고 해도 이전 프로토콜인 서명 버전 2에 대한 지원은 계속됩니다. 하지만 2014년 1월 30일 이후에 새롭게 생성된 리전은 모두 서명 버전 4만 지원하며, 이에 따라 새로운 리전에 대한 모든 요청은 서명 버전 4를 이용해야 합니다. AWS 서명 버전 2에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [REST 요청 서명 및 인증](#)을 참조하세요.

인증 방법

다음 방법 중 하나를 사용하여 인증 정보를 표현할 수 있습니다.

- HTTP Authorization 헤더 – HTTP Authorization 헤더를 사용하는 것이 Amazon S3 요청을 인증하는 가장 일반적인 방법입니다. 모든 Amazon S3 REST 작업(POST 요청을 사용한 브라우저 기반 업로드 제외)에는 이 헤더가 필요합니다. Authorization 헤더 값과 서명 계산 방법 및 관련 옵션에 대한 자세한 내용은 [요청 인증: Authorization 헤더 사용\(AWS 서명 버전 4\)](#) 섹션을 참조하세요.
- 쿼리 문자열 파라미터 – 쿼리 문자열을 사용하여 전체 요청을 URL로 표현할 수 있습니다. 이 경우 쿼리 파라미터를 사용하여 인증 정보를 포함한 요청 정보를 제공합니다. 요청 서명이 URL의 일부이므로 이 URL 유형을 미리 서명된 URL이라고 합니다. 미리 서명된 URL을 사용하여 최대 7일 동안 유효할 수 있는 클릭 가능한 링크를 HTML에 포함할 수 있습니다. 자세한 내용은 [요청 인증: 쿼리 파라미터 사용\(AWS 서명 버전 4\)](#) 섹션을 참조하세요.

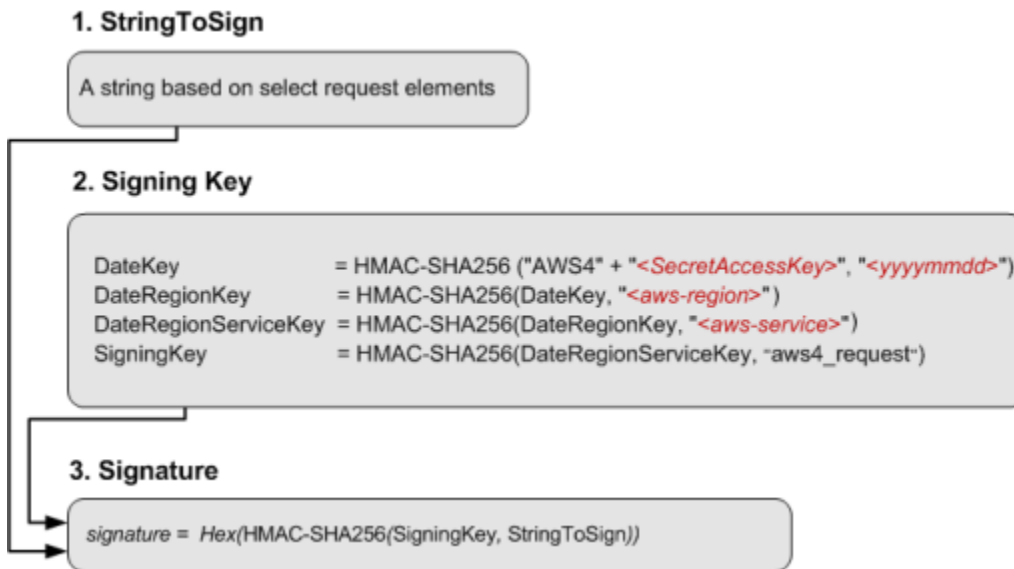
Amazon S3는 HTTP POST 요청을 사용하는 브라우저 기반 업로드도 지원합니다. HTTP POST 요청을 사용하면 콘텐츠를 브라우저에서 직접 Amazon S3에 업로드할 수 있습니다. POST 요청 인증에 대한 자세한 내용은 [POST를 사용한 브라우저 기반 업로드\(AWS 서명 버전 4\)](#) 섹션을 참조하세요.

서명 요청 소개

요청에서 보내는 인증 정보에는 서명이 포함되어야 합니다. 서명을 계산하려면 먼저 select 요청 요소를 연결하여 문자열을 생성하는데, 이를 서명할 문자열이라고 합니다. 이후 서명 키를 사용하여 서명할 문자열의 해시 기반 메시지 인증 코드(HMAC)를 계산합니다.

AWS 서명 버전 4에서는 비밀 액세스 키를 사용하여 요청에 서명하지 않습니다. 대신 비밀 액세스 키를 사용하여 서명 키를 파생합니다. 파생된 서명 키는 날짜, 서비스 및 리전에 따라 다릅니다. 다양한 프로그래밍 언어로 서명 키를 추출하는 방법에 대한 자세한 내용은 [서명 버전 4용 서명 키 추출 방법 예제](#)를 참조하세요.

다음 다이어그램은 서명을 계산하는 일반적인 프로세스를 보여줍니다.



서명할 문자열은 요청 유형에 따라 다릅니다. 예를 들어 HTTP Authorization 헤더 또는 쿼리 파라미터를 인증에 사용하는 경우 다양한 요청 요소 조합을 사용하여 서명할 문자열을 생성합니다. HTTP POST 요청의 경우 요청의 POST 정책은 서명하는 문자열입니다. 서명할 문자열 계산에 대한 자세한 내용은 이 섹션의 마지막에 제공된 링크를 참조하세요.

서명 키의 경우 다이어그램에는 일련의 계산이 표시되며, 각 단계의 결과가 다음 단계로 전달됩니다. 마지막 단계는 서명 키입니다.

Amazon S3 서버는 인증된 요청을 수신하면 요청에 포함된 인증 정보를 사용하여 서명을 다시 생성합니다. 서명이 일치하면 Amazon S3가 요청을 처리하고, 그렇지 않으면 요청이 거부됩니다.

요청 인증에 대한 자세한 내용은 다음 주제를 참조하세요.

- [요청 인증: Authorization 헤더 사용\(AWS 서명 버전 4\)](#)
- [요청 인증: 쿼리 파라미터 사용\(AWS 서명 버전 4\)](#)
- [POST를 사용한 브라우저 기반 업로드\(AWS 서명 버전 4\)](#)

요청 인증: Authorization 헤더 사용(AWS 서명 버전 4)

주제

- [개요](#)
- [Authorization 헤더에 대한 서명 계산: 페이로드를 단일 청크로 전송\(AWS 서명 버전 4\)](#)
- [Authorization 헤더에 대한 서명 계산: 페이로드를 여러 청크로 전송\(청크 업로드\)\(AWS 서명 버전 4\)](#)
- [후행 헤더에 대한 서명 계산\(청크 업로드\)\(AWS 서명 버전 4\)](#)

개요

HTTP Authorization 헤더를 사용하는 것이 인증 정보를 제공하는 가장 일반적인 방법입니다. [POST 요청](#) 및 쿼리 파라미터를 사용하여 서명된 요청을 제외하고 모든 Amazon S3 작업은 Authorization 요청 헤더를 사용하여 인증 정보를 제공합니다.

다음은 Authorization 헤더 값의 예시입니다. 가독성을 위해 이 예시에 줄바꿈이 추가됩니다.

```
Authorization: AWS4-HMAC-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/aws4_request,
SignedHeaders=host;range;x-amz-date,
Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024
```

다음 표에서는 앞선 예제에 있는 Authorization 헤더 값의 다양한 구성 요소를 설명합니다.

구성 요소	설명
AWS4-HMAC-SHA256	

구성 요소	설명
	서명을 계산하는 데 사용된 알고리즘입니다. 인증에 AWS 서명 버전 4를 사용하는 경우 이 값을 제공해야 합니다. 문자열은 AWS 서명 버전 4(AWS4) 및 서명 알고리즘(HMAC-SHA256)을 지정합니다.
Credential	액세스 키 ID 및 범위 정보(날짜, 리전 및 서명 계산에 사용된 서비스 포함). 이 문자열의 형식은 다음과 같습니다. <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> 위치: • <date> 값은 YYYYMMDD 형식을 사용하여 지정됩니다. • <aws-service> 값은 Amazon S3에 요청을 전송하는 경우 s3입니다.
SignedHeaders	Signature 계산에 사용한 세미콜론으로 구분된 요청 헤더의 목록입니다. 목록에는 헤더 이름만 포함되며, 헤더 이름은 소문자여야 합니다. 예제: <pre>host;range;x-amz-date</pre>

구성 요소	설명
Signature	64개의 소문자 16진수 문자로 표현되는 256비트 서명. 예제: <pre>fe5f80f77d5fa3beca038a248ff027d0445342fe2855d dc963176630326f1024</pre> 서명 계산은 페이로드를 전송하기 위해 선택한 옵션에 따라 달라집니다.

서명 계산은 요청 페이로드를 전송하기 위해 선택한 옵션에 따라 달라집니다. S3는 다음 옵션을 지원합니다.

- 페이로드를 단일 청크로 전송 – 이 경우 다음과 같은 서명 계산 옵션이 있습니다.
- 서명된 페이로드 옵션 – 선택적으로 전체 페이로드 체크섬을 계산하여 서명 계산에 포함할 수 있습니다. 이렇게 하면 보안이 강화되지만 페이로드를 두 번 읽거나 메모리에 버퍼링해야 합니다.

예를 들어 파일을 업로드하려면 먼저 파일을 읽고 서명 계산을 위해 페이로드 해시를 계산한 다음 요청을 생성할 때 전송을 위해 다시 읽어야 합니다. 더 작은 페이로드의 경우 이 접근 방식이 유리할 수 있습니다. 그러나 대용량 파일의 경우 파일을 두 번 읽는 것이 비효율적일 수 있으므로 대신 데이터를 청크로 업로드해야 할 수 있습니다.

보안을 강화하기 위해 페이로드 체크섬을 포함할 것을 권장합니다.

- 서명되지 않은 페이로드 옵션 – 서명 계산에 페이로드 체크섬을 포함하지 마세요.

서명을 계산하고 Authorization 헤더 값을 구성하는 단계별 지침은 [Authorization 헤더에 대한 서명 계산: 페이로드를 단일 청크로 전송\(AWS 서명 버전 4\)](#) 섹션을 참조하세요.

- 여러 청크로 페이로드 전송(청크 업로드) – 이 경우 페이로드를 청크로 전송합니다. 페이로드 크기에 관계없이 페이로드를 청크 단위로 전송할 수 있습니다.

페이로드를 청크로 분할할 수 있습니다. 청크는 고정 또는 가변 크기일 수 있습니다. 데이터를 청크로 업로드하면 전체 페이로드를 읽고 서명을 계산하지 않아도 됩니다. 대신 첫 번째 청크에서 요청 헤더만 사용하는 시드 서명을 계산합니다. 두 번째 청크에는 첫 번째 청크에 대한 서명이 포함되고, 이후의 각 청크는 바로 앞선 청크에 대한 서명이 포함됩니다. 업로드가 완료되면 페이로드의 마지막 청크에 대한 서명이 포함된 0바이트 데이터의 최종 청크를 전송합니다. 자세한 내용은 [Authorization](#)

[헤더에 대한 서명 계산: 페이로드를 여러 청크로 전송\(청크 업로드\)\(AWS 서명 버전 4\) 섹션을 참조](#) 하세요.

요청에 서명할 때 AWS 서명 버전 4 또는 AWS 서명 버전 4A를 사용할 수 있습니다. 둘 사이의 주요 차이점은 서명 계산 방법에 따라 결정됩니다. AWS 서명 버전 4A의 경우 서명에 리전별 정보가 포함되지 않고 AWS4-ECDSA-P256-SHA256 알고리즘을 사용하여 계산됩니다.

이러한 옵션 외에도 요청에 트레일러를 포함할 수 있습니다. 요청에 트레일러를 포함하려면 `x-amz-content-sha256`을 적절한 값으로 설정하여 헤더에서 지정해야 합니다. 후행 헤더를 사용하는 경우 헤더에 `x-amz-trailer`를 포함하고 후행 헤더 이름을 쉼표로 구분된 목록의 문자열로 지정해야 합니다. 모든 후행 헤더는 최종 청크 뒤에 기록됩니다. 데이터를 여러 청크로 업로드하는 경우 후행 헤더를 전송하기 전에 0바이트 데이터가 포함된 최종 청크를 전송해야 합니다.

요청을 전송할 때 다음 값 중 하나가 포함된 `x-amz-content-sha256` 헤더를 추가하여 서명 계산에서 위의 옵션 중 무엇을 선택했는지 Amazon S3에 알려야 합니다.

헤더 값	설명
실제 페이로드 체크섬 값	이 값은 객체의 실제 체크섬이며 데이터를 단일 청크로 업로드할 때만 가능합니다.
UNSIGNED-PAYLOAD	객체를 서명되지 않은 단일 청크로 업로드할 때 사용합니다.
STREAMING-UNSIGNED-PAYLOAD-TRAILER	서명되지 않은 페이로드를 여러 청크로 전송할 때 사용합니다. 이 경우 청크가 업로드된 후 후행 헤더도 있습니다.
STREAMING-AWS4-HMAC-SHA256-PAYLOAD	페이로드를 여러 청크로 전송하고 AWS4-HMAC-SHA256 을 사용하여 청크에 서명할 때 사용합니다. 그러면 SigV4 서명이 생성됩니다.
STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER	페이로드를 여러 청크로 전송하고 AWS4-HMAC-SHA256 을 사용하여 청크에 서명할 때 사용합니다. 그러면 SigV4 서명이 생성됩니다. 또한 청크의 다이제스트가 후행 헤더로 포함됩니다.
STREAMING-AWS4-ECDSA-P256-SHA256-PAYLOAD	페이로드를 여러 청크로 전송하고 AWS4-ECDSA-P256-SHA256 을 사용하여 청크에 서명할 때 사용합니다. 그러면 SigV4A 서명이 생성됩니다.

헤더 값	설명
STREAMING-AWS4-ECDSA-P256-SHA256-PAYLOAD-TRAILER	페이로드를 여러 청크로 전송하고 AWS4-ECDSA-P256-SHA256 을 사용하여 청크에 서명할 때 사용합니다. 그러면 SigV4A 서명이 생성됩니다. 또한 청크의 다이제스트가 후행 헤더로 포함됩니다.

Amazon S3는 요청을 수신하면 Authorization 헤더 및 date 헤더의 정보를 사용하여 서명할 문자열을 다시 생성합니다. 그런 다음 인증 서비스를 사용하여 서명이 일치하는지 확인합니다. 요청 날짜는 HTTP Date 또는 x-amz-date 헤더를 사용하여 지정할 수 있습니다. 두 헤더가 모두 있는 경우 x-amz-date가 우선합니다.

서명이 일치하면 Amazon S3가 요청을 처리하고, 그렇지 않으면 요청이 실패합니다.

자세한 내용은 다음 항목을 참조하세요.

[Authorization 헤더에 대한 서명 계산: 페이로드를 단일 청크로 전송\(AWS 서명 버전 4\)](#)

[Authorization 헤더에 대한 서명 계산: 페이로드를 여러 청크로 전송\(청크 업로드\)\(AWS 서명 버전 4\)](#)

[후행 헤더에 대한 서명 계산\(청크 업로드\)\(AWS 서명 버전 4\)](#)

Authorization 헤더에 대한 서명 계산: 페이로드를 단일 청크로 전송(AWS 서명 버전 4)

Authorization 헤더를 사용하여 요청을 인증할 때 헤더 값에는 무엇보다도 서명이 포함됩니다. 서명 계산은 페이로드 전송을 위해 선택한 항목에 따라 달라집니다([개요](#)). 이 섹션에서는 페이로드를 단일 청크로 전송하도록 선택하는 경우의 서명 계산에 대해 설명합니다. 예제 섹션([예제: 서명 계산](#) 참조)에는 코드를 확인하기 위한 테스트 스위트로 사용할 수 있는 서명 계산 및 결과 Authorization 헤더가 나와 있습니다.

Important

페이로드를 단일 청크로 전송할 때 선택적으로 서명 계산에 페이로드 해시를 포함하도록 선택할 수 있으며, 이를 서명된 페이로드라고 합니다(페이로드 해시를 포함하지 않으면 페이로드가 서명되지 않은 것으로 간주됨). 다음 섹션에서 설명하는 서명 절차는 둘 모두에 적용되지만 다음과 같은 차이점이 있습니다.

- 서명된 페이로드 옵션 – 정식 요청을 구성할 때 페이로드 해시를 포함합니다(이 경우 서명 계산 섹션에 설명된 대로 StringToSign의 일부가 됨). 또한 S3에 요청을 전송할 때 x-amz-content-sha256 헤더 값과 동일한 값을 지정합니다.
- 서명되지 않은 페이로드 옵션 - 표준 요청을 구성할 때 리터럴 문자열 UNSIGNED-PAYLOAD를 포함하고, Amazon S3에 요청을 전송할 때 x-amz-content-sha256 헤더 값과 동일한 값을 설정합니다.

Amazon S3에 요청을 전송하면 x-amz-content-sha256 헤더 값은 페이로드가 서명되었는지 여부를 Amazon S3에 알립니다. 그런 다음 Amazon S3는 그에 따라 확인을 위해 서명을 생성할 수 있습니다.

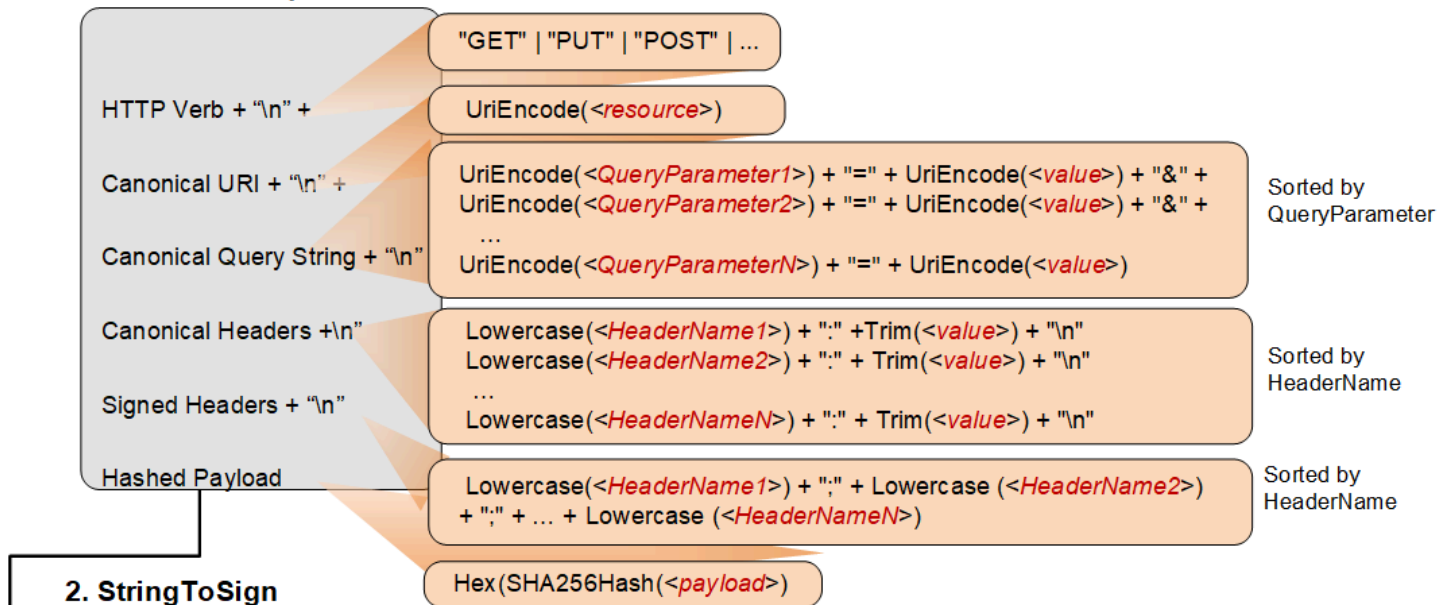
두 경우 모두 x-amz-content-sha256 헤더 값이 이미 HashedPayload의 일부이므로 x-amz-content-sha256 헤더를 정식 헤더로 포함할 필요가 없습니다.

서명 계산

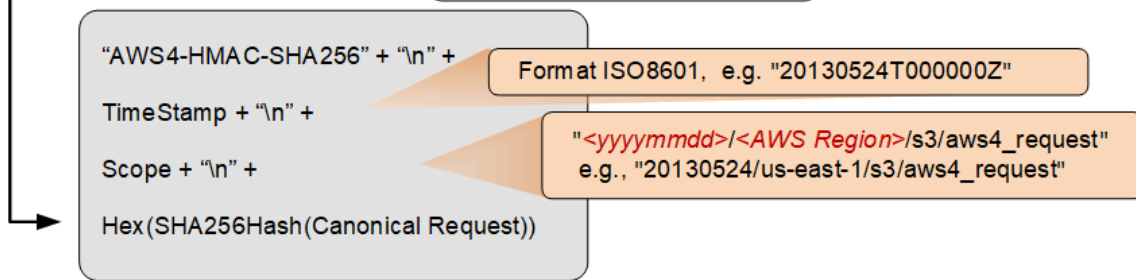
서명을 계산하려면 먼저 서명할 문자열이 필요합니다. 이후 서명 키를 사용하여 서명할 문자열의 HMAC-SHA256 해시를 계산합니다. 다음 다이어그램은 서명을 위해 생성한 문자열의 다양한 구성 요소를 포함한 프로세스를 보여줍니다.

Amazon S3는 인증된 요청을 수신하면 서명을 계산한 다음 요청에 제공된 서명과 비교합니다. 이러한 이유로 인해 Amazon S3가 사용한 방법과 동일한 방법으로 서명을 계산해야 합니다. 요청을 서명을 위해 합의된 형식에 추가하는 과정을 정규화라고 합니다.

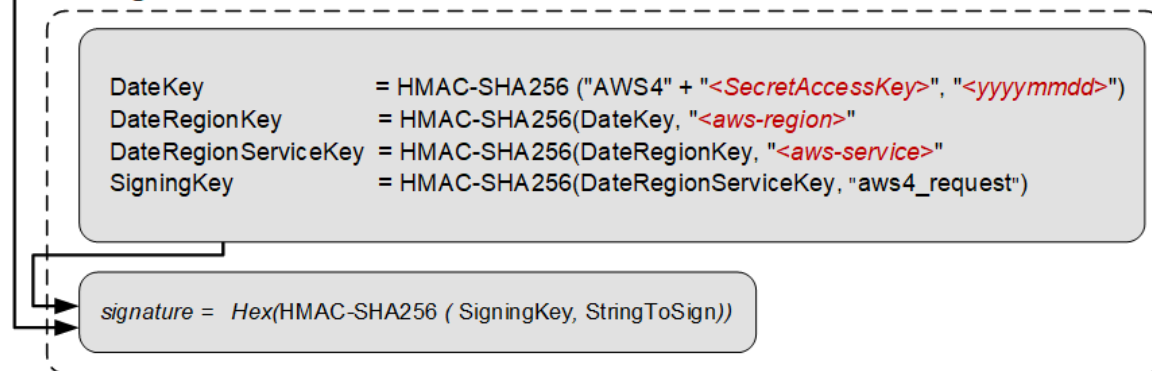
1. Canonical Request



2. StringToSign



3. Signature



다음 표에서는 다이어그램에 표시된 함수를 설명합니다. 이러한 함수의 코드를 구현해야 합니다.

함수	설명
Lowercase()	문자열을 소문자로 변환합니다.
Hex()	소문자 16진법 인코딩.

함수	설명
SHA256Hash()	보안 해시 알고리즘(SHA) 암호화 해시 함수.
HMAC-SHA256()	제공된 서명 키를 포함한 SHA256 알고리즘을 사용하여 HMAC를 계산합니다. 이것이 최종 서명입니다.
Trim()	선행 또는 후행 공백을 모두 제거합니다.

함수	설명
UriEncode()	URI는 모든 바이트를 인코딩합니다. UriEncode()는 다음 규칙을 적용해야 합니다. - URI는 예약되지 않은 문자를 제외한 모든 바이트를 인코딩합니다. 'A'-'Z', 'a'-'z', '0'-'9', '-', '.', '_' 및 '~'. - 공백 문자는 예약된 문자이고 '+'가 아닌 '%20'로 인코딩되어야 합니다. - 각 URI 인코딩 바이트는 '%' 및 바이트의 두 자리 16진수 값으로 구성됩니다. 16진수 값의 문자는 대문자여야 합니다(예: '%1A'). - 슬래시 문자('/')를 객체 키 이름을 제외한 모든 위치에 인코딩합니다. 예를 들어, 객체 키 이름이 photos/Jan/sample.jpg 인 경우 키 이름의 슬래시는 인코딩되지 않습니다.  Important 개발 플랫폼에서 제공하는 표준 UriEncode 함수는 구현상의 차이와 기본 RFC의 관련 모호성으로 인해 효과적이지 않을 수 있습니다. 인코딩이 제대로 작동하도록 사용자 지정 UriEncode 함수를 직접 작성하는 것이 좋습니다. 다음은 Java로 작성된 UriEncode() 함수 예제입니다. <pre> public static String UriEncode(CharSequence input, boolean encodeSlash) { StringBuilder result = new StringBui lder(); for (int i = 0; i < input.length(); i++) { char ch = input.charAt(i); if ((ch >= 'A' && ch <= 'Z') (ch >= 'a' && ch <= 'z') (ch >= '0' && ch <= '9') ch == '_' ch == '-' ch == '~' ch == '.') { result.append(ch); } else if (ch == '/') { </pre>

함수	설명
	<pre> result.append(encodeSlash ? "%2F" : ch); } else { result.append(toHexUTF8(ch)); } } return result.toString(); } </pre>

작업 1: 표준 요청 생성

이 섹션에서는 정식 요청 생성에 대한 개요를 제공합니다.

다음은 Amazon S3가 서명을 계산하는 데 사용하는 정식 요청 형식입니다. 서명을 일치시키려면 이 형식으로 정식 요청을 생성해야 합니다.

```

<HTTPMethod>\n
<CanonicalURI>\n
<CanonicalQueryString>\n
<CanonicalHeaders>\n
<SignedHeaders>\n
<HashedPayload>

```

위치:

- **HTTPMethod**는 GET, PUT, HEAD, DELETE와 같은 HTTP 메서드 중 하나입니다.
- **CanonicalURI**는 도메인 이름 뒤에 오는 '/'로 시작하여 문자열의 끝 또는 쿼리 문자열 파라미터가 있는 경우 물음표 문자('?')까지 URI의 절대 경로 구성 요소를 URI로 인코딩한 버전입니다. 다음 예제에서 URI /examplebucket/myphoto.jpg는 절대 경로이며 절대 경로에는 '/'를 인코딩하지 않습니다.

```
http://s3.amazonaws.com/examplebucket/myphoto.jpg
```

Note

Amazon S3에 대한 요청의 URI 경로를 정규화하지 않습니다. 예를 들어 'my-object//example//photo.user'라는 객체가 있는 버킷이 있습니다. 경로를 정규화하면 요청의 객체 이름이 'my-object/example/photo.user'로 변경됩니다. 이는 해당 객체의 잘못된 경로입니다.

- **CanonicalQueryString**은 URI로 인코딩된 쿼리 문자열 파라미터를 지정합니다. 이름과 값을 개별적으로 URI 인코딩합니다. 또한 표준 쿼리 문자열의 파라미터를 키 이름별로 알파벳순으로 정렬해야 합니다. 정렬은 인코딩 후에 이루어집니다. 다음 URI 예제의 쿼리 문자열은 다음과 같습니다. `prefix=somePrefix&marker=someMarker&max-keys=20`

```
http://s3.amazonaws.com/examplebucket?prefix=somePrefix&marker=someMarker&max-keys=20
```

표준 쿼리 문자열은 다음과 같습니다(가독성을 위해 줄바꿈 추가됨).

```
UriEncode("marker")+"="+UriEncode("someMarker")+"&"+
UriEncode("max-keys")+"="+UriEncode("20") + "&" +
UriEncode("prefix")+"="+UriEncode("somePrefix")
```

요청이 하위 리소스를 대상으로 하는 경우 해당 쿼리 파라미터 값은 빈 문자열("")이 됩니다. 예를 들어, 다음 URI는 examplebucket 버킷에서 ACL 하위 리소스를 식별합니다.

```
http://s3.amazonaws.com/examplebucket?acl
```

이 경우 CanonicalQueryString은 다음과 같습니다.

```
UriEncode("acl") + "=" + ""
```

URI에 '?'가 포함되지 않은 경우 요청에는 쿼리 문자열이 없으며, 표준 쿼리 문자열을 빈 문자열("")로 설정합니다. 여전히 '\n'을 포함해야 합니다.

- **CanonicalHeaders**는 요청 헤더 및 해당 값의 목록입니다. 개별 헤더 이름과 값 페어는 개행 문자 ('\n')로 구분됩니다. 헤더 이름은 소문자여야 합니다. 다음 예제와 같이 헤더 이름을 알파벳순으로 정렬하여 문자열을 구성해야 합니다.

```
Lowercase(<HeaderName1>)+":"+Trim(<value>)+"\n"
Lowercase(<HeaderName2>)+":"+Trim(<value>)+"\n"
...
Lowercase(<HeaderNameN>)+":"+Trim(<value>)+"\n"
```

이 예제에서 사용하는 Lowercase() 및 Trim() 함수는 이전 섹션에서 설명합니다.

CanonicalHeaders 목록에는 다음이 포함되어야 합니다.

- HTTP host 헤더
- Content-MD5 헤더가 요청에 있는 경우 이를 *CanonicalHeaders* 목록에 추가해야 합니다.
- 요청에 포함할 x-amz-* 헤더 또한 추가되어야 합니다. 예를 들어, 임시 보안 인증 정보를 사용하는 경우 요청에 x-amz-security-token을 포함해야 합니다. 이 헤더를 *CanonicalHeaders* 목록에 추가해야 합니다.

Note

- x-amz-content-sha256 헤더는 모든 AWS 서명 버전 4 요청에 필요합니다. 요청 페이로드의 해시를 제공합니다. 그러나 S3는 요청에서 전송된 페이로드 해시를 계산할 때 자동으로 해당 값을 사용하기 때문에 x-amz-content-sha256를 정식 헤더로 포함할 필요는 없습니다.
- 페이로드가 없는 경우 빈 문자열의 해시를 제공해야 합니다.
- S3가 요청의 해시와 대조하지 않도록 하려면 대신 리터럴 문자열 "UNSIGNED-PAYLOAD"를 사용할 수 있습니다.

다음은 CanonicalHeaders 문자열의 예입니다. 헤더 이름은 소문자이고 정렬되어 있습니다.

Example예 1

```
host:s3.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca95991b7852b855
x-amz-date:20130708T220855Z
```

Example예제 2

```
host:s3.amazonaws.com
x-amz-content-sha256:UNSIGNED-PAYLOAD
```

```
x-amz-date:20130708T220855Z
```

Example에 3

```
host:s3.amazonaws.com
x-amz-date:20130708T220855Z
```

Note

권한 부여 서명을 계산하려면 호스트 헤더 그리고 x-amz-content-sha256을 제외한 모든 x-amz-* 헤더가 필요합니다. 그러나 데이터 변조를 방지하려면 서명 계산에 모든 헤더를 포함하는 것을 고려해야 합니다.

수신된 요청 페이로드 해시를 계산할 때 S3가 해당 값을 사용하기 때문에 x-amz-content-sha256 헤더 서명은 선택 사항입니다.

- **SignedHeaders**는 소문자 요청 헤더 이름의 알파벳순으로 정렬되고 세미콜론으로 구분된 목록입니다. 목록의 요청 헤더는 CanonicalHeaders 문자열에 포함한 것과 동일합니다. 이전 예제에서 **SignedHeaders**의 값은 다음과 같습니다.

Example예제 1 및 2

```
host;x-amz-content-sha256;x-amz-date
```

Example예 3

```
host;x-amz-date
```

- **HashedPayload**는 요청 페이로드 SHA256 해시의 16진수 값입니다.

```
Hex(SHA256Hash(<payload>))
```

요청에 페이로드가 없는 경우 다음과 같이 빈 문자열의 해시를 계산합니다.

```
Hex(SHA256Hash(""))
```

해시는 다음 값을 반환합니다.

```
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

예를 들어 PUT 요청을 사용하여 객체를 업로드하는 경우 본문에 객체 데이터를 제공합니다. GET 요청을 사용하여 객체를 검색하는 경우 빈 문자열 해시를 계산합니다.

작업 2: 서명할 문자열 생성

이 섹션에서는 서명할 문자열 생성에 대한 개요를 제공합니다. 단계별 지침은 AWS 일반 참조의 [작업 2: 서명할 문자열 생성](#)을 참조하세요.

서명할 문자열은 다음 문자열을 연결한 것입니다.

```
"AWS4-HMAC-SHA256" + "\n" +  
timeStampISO8601Format + "\n" +  
<Scope> + "\n" +  
Hex(SHA256Hash(<CanonicalRequest>))
```

상수 문자열 AWS4-HMAC-SHA256은 사용 중인 해시 알고리즘인 HMAC-SHA256을 지정합니다. timeStamp는 ISO 8601 형식의 현재 UTC 시간입니다(예: 20130524T000000Z).

Scope는 결과 서명을 특정 날짜, AWS 리전 및 서비스에 바인딩합니다. 따라서 결과 서명은 특정 리전 및 특정 서비스에서만 작동합니다. 서명은 지정된 날짜 이후 7일 동안 유효합니다.

```
date.Format(<YYYYMMDD>) + "/" + <region> + "/" + <service> + "/aws4_request"
```

Amazon S3의 경우 서비스 문자열은 s3입니다. ## 문자열 목록은 AWS 일반 참조의 [리전 및 엔드포인트](#)를 참조하세요. 이 표의 리전 열은 유효한 리전 문자열의 목록입니다.

다음 범위는 결과 서명을 us-east-1 리전 및 Amazon S3로 제한합니다.

```
20130606/us-east-1/s3/aws4_request
```


Note

Scope는 다음 섹션에서 설명한 대로 서명 키를 계산하는 데 사용하는 것과 동일한 날짜를 사용해야 합니다.

작업 3: 서명 계산

AWS 서명 버전 4에서는 AWS 액세스 키를 사용하여 요청에 서명하는 대신, 먼저 특정 리전 및 서비스로 범위가 지정된 서명 키를 생성합니다. 서명 키에 대한 자세한 정보는 [서명 요청 소개](#) 섹션을 참조하세요.

```
DateKey           = HMAC-SHA256("AWS4"+"<SecretAccessKey>", "<YYYYMMDD>")
DateRegionKey     = HMAC-SHA256(<DateKey>, "<aws-region>")
DateRegionServiceKey = HMAC-SHA256(<DateRegionKey>, "<aws-service>")
SigningKey        = HMAC-SHA256(<DateRegionServiceKey>, "aws4_request")
```

Note

일부 사용 사례에서는 서명 키를 최대 7일 동안 처리할 수 있습니다. 자세한 내용은 [다른 사용자 및 객체 공유](#)를 참조하세요.

리전 문자열 목록은 AWS 일반 참조의 [리전 및 엔드포인트](#)를 참조하세요.

서명 키를 사용하면 AWS 자격 증명을 한 곳에 안전하게 보관할 수 있습니다. 예를 들어 Amazon S3와 통신하는 서버가 여러 개 있는 경우 서명 키를 해당 서버와 공유합니다. 그러면 각 서버에 시크릿 액세스 키의 사본을 보관할 필요가 없습니다. 서명 키는 최대 7일 동안 유효합니다. 따라서 서명 키를 계산할 때마다 서명 키를 서버와 공유해야 합니다. 자세한 내용은 [요청 인증\(AWS 서명 버전 4\)](#) 섹션을 참조하세요.

최종 서명은 서명 키를 키로 사용하여 서명할 문자열의 HMAC-SHA256 해시입니다.

```
HMAC-SHA256(SigningKey, StringToSign)
```

서명 생성에 대한 단계별 지침은 AWS 일반 참조의 [작업 3: 서명 생성](#)을 참조하세요.

예제: 서명 계산

이 섹션의 예제는 코드의 서명 계산을 점검하기 위한 참조로 사용할 수 있습니다. 예제에 표시된 계산은 다음 데이터를 사용합니다.

- 예제 액세스 키.

파라미터	값
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

- 요청 타임스탬프 20130524T000000Z(Fri, 24 May 2013 00:00:00 GMT).
- 버킷 이름 examplebucket.
- 버킷은 미국 동부(버지니아 북부) 리전에 있는 것으로 가정합니다. 자격 증명 Scope 및 Signing Key 계산은 us-east-1을 리전 지정자로 사용합니다. 다른 리전에 대한 자세한 내용은 AWS 일반 참조의 [리전 및 엔드포인트](#)를 참조하세요.
- 경로 방식 또는 가상 호스팅 방식 요청을 사용할 수 있습니다. 다음 예제에서는 가상 호스팅 방식 요청에 서명하는 방법을 보여줍니다.

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [버킷의 가상 호스팅](#)을 참조하세요.

예제: GET Object

다음 예제에서는 examplebucket에서 객체(test.txt)의 처음 10바이트를 가져옵니다. 이 API 작업에 대한 자세한 내용은 [GetObject](#)를 참조하세요.

```
GET /test.txt HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Authorization: SignatureToBeCalculated
Range: bytes=0-9
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

```
x-amz-date: 20130524T000000Z
```

이 GET 요청은 본문 콘텐츠를 제공하지 않으므로 x-amz-content-sha256 값은 빈 요청 본문의 해시 또는 리터럴 문자열 "UNSIGNED-PAYLOAD"일 수 있습니다. 다음 단계에서는 빈 문자열의 해시를 사용하여 Authorization 헤더의 서명 계산 및 구성을 보여줍니다.

1. StringToSign

a. CanonicalRequest

```
GET
/test.txt

host:examplebucket.s3.amazonaws.com
range:bytes=0-9
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130524T000000Z

host;range;x-amz-content-sha256;x-amz-date
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

정식 요청 문자열에서 마지막 줄은 빈 요청 본문의 해시입니다. 세 번째 줄은 요청에 쿼리 파라미터가 없기 때문에 비어 있습니다.

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
7344ae5b7ee6c3e7e6b0fe0640412a37625d1fbfff95c48bbb2dc43964946972
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. 서명

```
f0e8bdb87c964420e857bd35b5d6ed310bd44f0170aba48dd91039c6036bdb41
```

4. Authorization 헤더

결과 Authorization 헤더는 다음과 같습니다.

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;range;x-amz-content-sha256;x-amz-
date,Signature=f0e8bdb87c964420e857bd35b5d6ed310bd44f0170aba48dd91039c6036bdb41
```

예제: PUT Object

이 예제 PUT 요청은 examplebucket에서 객체(test\$file.text)를 생성합니다. 이 예시에서는 다음과 같이 가정합니다.

- x-amz-storage-class 요청 헤더를 추가하여 REDUCED_REDUNDANCY를 스토리지 클래스로 요청하고 있습니다. 스토리지 클래스에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [스토리지 클래스](#)를 참조하세요.
- 업로드된 파일의 콘텐츠는 문자열 "Welcome to Amazon S3."입니다. 요청에서 x-amz-content-sha256의 값은 이 문자열을 기반으로 합니다.

이 API 작업에 대한 자세한 내용은 [PutObject](#)를 참조하세요.

```
PUT test$file.text HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Date: Fri, 24 May 2013 00:00:00 GMT
Authorization: SignatureToBeCalculated
x-amz-date: 20130524T000000Z
x-amz-storage-class: REDUCED_REDUNDANCY
x-amz-content-sha256: 44ce7dd67c959e0d3524ffac1771dfbba87d2b6b4b4e99e42034a8b803f8b072
```

<Payload>

다음 단계에서는 서명 계산을 보여줍니다.

1. StringToSign

a. CanonicalRequest

```
PUT
/test%24file.text

date:Fri, 24 May 2013 00:00:00 GMT
host:examplebucket.s3.amazonaws.com
x-amz-content-sha256:44ce7dd67c959e0d3524ffac1771dfbba87d2b6b4b4e99e42034a8b803f8b072
x-amz-date:20130524T000000Z
x-amz-storage-class:REDUCED_REDUNDANCY

date;host;x-amz-content-sha256;x-amz-date;x-amz-storage-class
44ce7dd67c959e0d3524ffac1771dfbba87d2b6b4b4e99e42034a8b803f8b072
```

정식 요청에서 세 번째 줄은 요청에 쿼리 파라미터가 없기 때문에 비어 있습니다. x-amz-content-sha256 정식 헤더는 선택적으로 서명할 수 있습니다. 요청 하단에 페이로드 해시가 이미 제공되기 때문입니다. 마지막 줄은 본문의 해시로, HTTP 요청에서 S3로 전송된 x-amz-content-sha256 header 값과 동일해야 합니다.

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
9e0e90d9c76de8fa5b200d8c849cd5b8dc7a3be3951ddb7f6a76b4158342019d
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. 서명

```
98ad721746da40c64f1a55b78f14c238d841ea1380cd77a1b5971af0ece108bd
```

4. Authorization 헤더

결과 Authorization 헤더는 다음과 같습니다.

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/
aws4_request,SignedHeaders=date;host;x-amz-content-sha256;x-amz-date;x-amz-storage-
class,Signature=98ad721746da40c64f1a55b78f14c238d841ea1380cd77a1b5971af0ece108bd
```

예제: GET Bucket Lifecycle

다음 GET 요청은 examplebucket의 수명 주기 구성을 검색합니다. 이 API 작업에 대한 자세한 내용은 [GetBucketLifecycleConfiguration](#)을 참조하세요.

```
GET ?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Authorization: SignatureToBeCalculated
x-amz-date: 20130524T000000Z
x-amz-content-sha256: e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

요청은 본문 콘텐츠를 제공하지 않으므로 x-amz-content-sha256 헤더 값은 빈 요청 본문의 해시입니다. 다음 단계에서는 서명 계산을 보여줍니다.

1. StringToSign

a. CanonicalRequest

```
GET
/
lifecycle=
host:examplebucket.s3.amazonaws.com
x-amz-content-
sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130524T000000Z

host;x-amz-content-sha256;x-amz-date
```

```
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

정식 요청에서 마지막 줄은 빈 요청 본문의 해시입니다.

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
9766c798316ff2757b517bc739a67f6213b4ab36dd5da2f94eaebf79c77395ca
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. 서명

```
fea454ca298b7da1c68078a5d1bdbfbbe0d65c699e0f91ac7a200a0136783543
```

4. Authorization 헤더

결과 Authorization 헤더는 다음과 같습니다.

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-
date,Signature=fea454ca298b7da1c68078a5d1bdbfbbe0d65c699e0f91ac7a200a0136783543
```

예제: Get Bucket(List Objects)

다음 예제는 examplebucket 버킷에서 객체의 목록을 검색합니다. 이 API 작업에 대한 자세한 내용은 [ListObjects](#)를 참조하세요.

```
GET ?max-keys=2&prefix=J HTTP/1.1
Host: examplebucket.s3.amazonaws.com
```

Authorization: *SignatureToBeCalculated*

x-amz-date: 20130524T000000Z

x-amz-content-sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855

요청은 본문을 제공하지 않으므로 x-amz-content-sha256의 값은 빈 요청 본문의 해시입니다. 다음 단계에서는 서명 계산을 보여줍니다.

1. StringToSign

a. CanonicalRequest

```
GET
/
max-keys=2&prefix=J
host:examplebucket.s3.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130524T000000Z

host;x-amz-content-sha256;x-amz-date
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

정식 문자열에서 마지막 줄은 빈 요청 본문의 해시입니다.

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
df57d21db20da04d7fa30298dd4488ba3a2b47ca3a489c74750e0f1e7df1b9b7
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. 서명


```
34b48302e7b5fa45bde8084f4b7868a86f0a534bc59db6670ed5711ef69dc6f7
```

4. Authorization 헤더

결과 Authorization 헤더는 다음과 같습니다.

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-
date,Signature=34b48302e7b5fa45bde8084f4b7868a86f0a534bc59db6670ed5711ef69dc6f7
```

Authorization 헤더에 대한 서명 계산: 페이로드를 여러 청크로 전송(청크 업로드)(AWS 서명 버전 4)

[개요](#)에 설명된 대로 Authorization 헤더를 사용하여 요청을 인증할 때 페이로드를 청크로 업로드할 수 있습니다. 데이터를 고정 크기 또는 가변 크기 청크로 전송할 수 있습니다. 이 섹션에서는 청크 업로드의 서명 계산 프로세스, 청크 본문을 생성하는 방법, 첫 번째 청크를 업로드하고 후속 청크에서 서명을 전송하는 지연된 서명이 작동하는 방식을 설명합니다. 예제 섹션([예제: PUT Object](#) 참조)에는 코드를 확인하기 위한 테스트 스위트로 사용할 수 있는 서명 계산 및 결과 Authorization 헤더가 나와 있습니다.

Note

데이터를 일련의 청크로 전송할 때는 다음 중 하나를 수행해야 합니다.

- Content-Length HTTP 헤더를 사용하여 총 콘텐츠 길이(바이트 단위의 객체 길이 및 각 청크의 메타데이터)를 명시적으로 지정합니다. 이렇게 하려면 요청을 시작하기 전에 각 청크에서 전송하는 메타데이터를 포함하여 페이로드의 총 길이를 미리 계산해야 합니다.
- Transfer-Encoding HTTP 헤더를 지정합니다. Transfer-Encoding 헤더를 포함하고 identity 이외의 값을 지정하는 경우 Content-Length 헤더를 생략해야 합니다.

모든 요청에 대해 객체의 크기를 바이트 단위로 지정하여 x-amz-decoded-content-length 헤더를 포함해야 합니다.

각 청크 서명 계산에는 이전 청크의 서명이 포함됩니다. 시작하려면 헤더만 사용하여 시드 서명을 생성합니다. 첫 번째 청크의 서명 계산에는 시드 서명을 사용합니다. 이후의 각 청크에서는 이전 청크의 서명이 포함된 청크 서명을 생성합니다. 따라서 청크 서명은 서로 연결(체인화)됩니다. 즉, 청크 n의 서명은 $F(\text{청크 } n, \text{서명}(\text{청크 } n-1))$ 이라는 함수입니다. 체인화를 사용하면 청크를 올바른 순서로 전송할 수 있습니다.

청크 업로드를 수행하려면 다음을 수행합니다.

1. 페이로드 청크 크기를 결정합니다. 이 크기는 코드를 작성할 때 필요합니다.

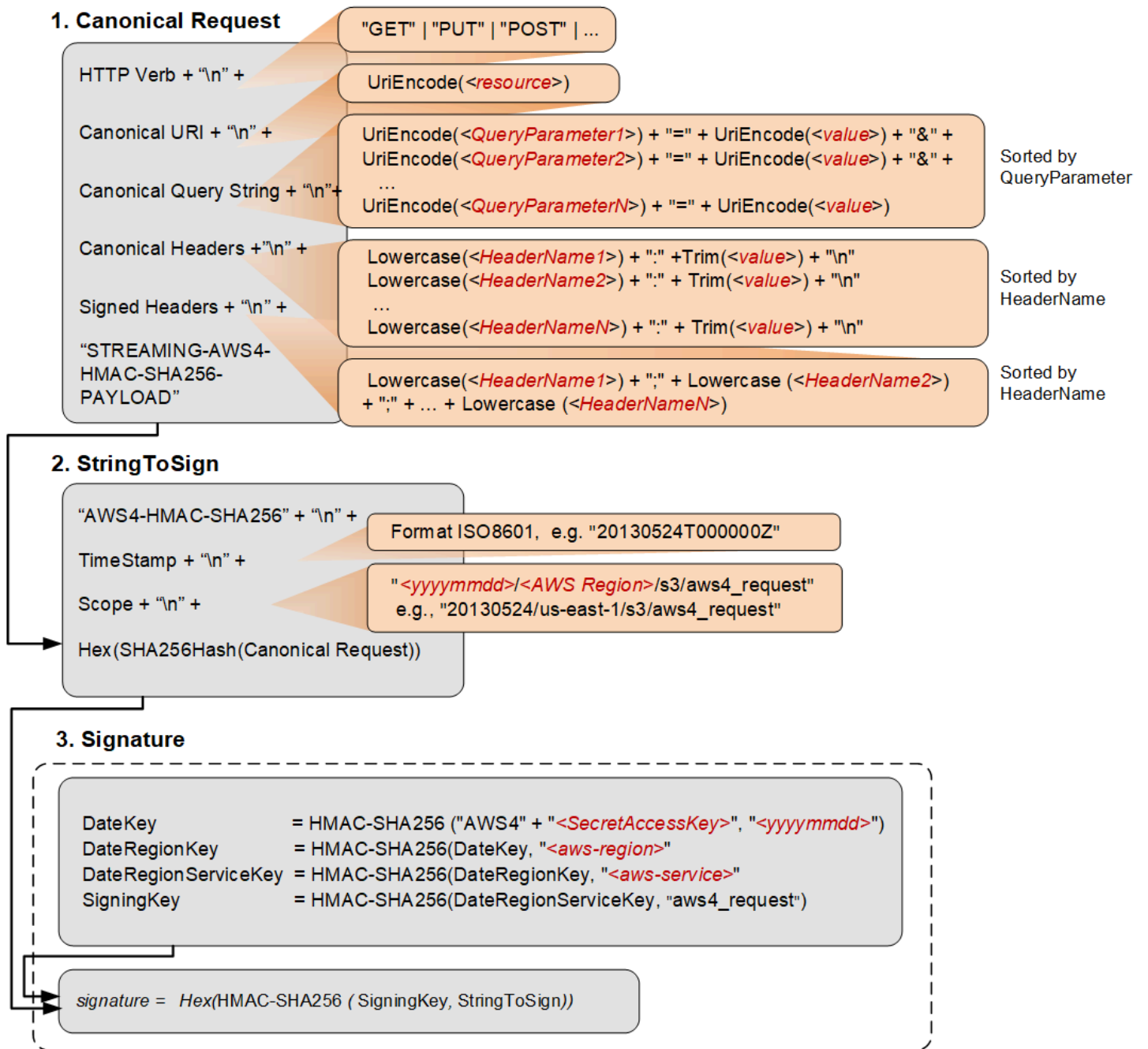
청크 크기는 8KB 이상이어야 합니다. 성능 향상을 위해 청크 크기를 최소 64KB로 설정하는 것이 좋습니다. 이 청크 크기는 마지막 청크를 제외한 모든 청크에 적용됩니다. 마지막으로 전송하는 청크

는 8KB보다 작을 수 있습니다. 페이로드가 작고 하나의 청크에 들어갈 수 있는 경우 8KB보다 작을 수 있습니다.

2. 첫 번째 청크에 포함할 시드 서명을 생성합니다. 자세한 내용은 [시드 서명 계산](#) 섹션을 참조하세요.
3. 첫 번째 청크를 생성하고 스트리밍합니다. 자세한 내용은 [청크 본문 정의](#) 섹션을 참조하세요.
4. 이후의 각 청크에 대해 서명할 문자열에 이전 서명을 포함하는 청크 서명을 계산하고 청크를 구성하여 전송합니다. 자세한 내용은 [청크 본문 정의](#) 섹션을 참조하세요.
5. 구성의 다른 청크와 동일하지만 데이터 바이트가 0인 최종 추가 청크를 전송합니다. 자세한 내용은 [청크 본문 정의](#) 섹션을 참조하세요.

시드 서명 계산

다음 다이어그램은 시드 서명을 계산하는 프로세스를 보여줍니다.



다음 표에서는 다이어그램에 표시된 함수를 설명합니다. 이러한 함수의 코드를 구현해야 합니다.

함수	설명
Lowercase()	문자열을 소문자로 변환합니다.
Hex()	소문자 16진법 인코딩.

함수	설명
SHA256Hash()	보안 해시 알고리즘(SHA) 암호화 해시 함수.
HMAC-SHA256()	제공된 서명 키를 포함한 SHA256 알고리즘을 사용하여 HMAC를 계산합니다. 이것이 최종 서명입니다.
Trim()	선행 또는 후행 공백을 모두 제거합니다.

함수	설명
UriEncode()	URI는 모든 바이트를 인코딩합니다. UriEncode()는 다음 규칙을 적용해야 합니다. • URI는 예약되지 않은 문자를 제외한 모든 바이트를 인코딩합니다. 'A'-'Z', 'a'-'z', '0'-'9', '-', '.', '_' 및 '~'. • 공백 문자는 예약된 문자이고 '+'가 아닌 '%20'로 인코딩되어야 합니다. • 각 URI 인코딩 바이트는 '%' 및 바이트의 두 자리 16진수 값으로 구성됩니다. • 16진수 값의 문자는 대문자여야 합니다(예: '%1A'). • 슬래시 문자('/')를 객체 키 이름을 제외한 모든 위치에 인코딩합니다. 예를 들어, 객체 키 이름이 photos/Jan/sample.jpg 인 경우 키 이름의 슬래시는 인코딩되지 않습니다.  Important 개발 플랫폼에서 제공하는 표준 UriEncode 함수는 구현상의 차이와 기본 RFC의 관련 모호성으로 인해 효과적이지 않을 수 있습니다. 인코딩이 제대로 작동하도록 사용자 지정 UriEncode 함수를 직접 작성하는 것이 좋습니다. 다음은 Java로 작성된 UriEncode() 함수 예제입니다. <pre> public static String UriEncode(CharSequence input, boolean encodeSlash) { StringBuilder result = new StringBui lder(); for (int i = 0; i < input.length(); i++) { char ch = input.charAt(i); if ((ch >= 'A' && ch <= 'Z') (ch >= 'a' && ch <= 'z') (ch >= '0' && ch <= '9') ch == '_' ch == '-' ch == '~' ch == '.') { result.append(ch); } else if (ch == '/') { </pre>

함수	설명
	<pre> result.append(encodeSlash ? "%2F" : ch); } else { result.append(toHexUTF8(ch)); } } return result.toString(); } </pre>

서명 프로세스에 대한 자세한 내용은 [Authorization 헤더에 대한 서명 계산: 페이로드를 단일 청크로 전송\(AWS 서명 버전 4\)](#) 섹션을 참조하세요. CanonicalRequest 생성이 다음과 같이 다르다는 점을 제외하면 프로세스는 동일합니다.

- 추가하려는 요청 헤더 외에도 다음 헤더를 포함해야 합니다.

헤더	설명
x-amz-content-sha256	이 헤더는 모든 AWS 서명 버전 4 요청에서 필수입니다. 서명이 헤더만 포함하고 페이로드가 없음을 나타내려면 값을 STREAMING-AWS4-HMAC-SHA256-PAYLOAD 로 설정합니다.

헤더	설명
Content-Encoding	값을 <code>aws-chunked</code> (으)로 설정합니다. Amazon S3는 여러 콘텐츠 인코딩 값을 지원합니다. 서명 버전 4 스트리밍 API를 사용하는 경우 사용자 지정 콘텐츠 인코딩을 지정할 수 있습니다. 예제: <pre>Content-Encoding : aws-chunked,gzip</pre> Amazon S3는 결과 객체를 <code>content-encoding</code> 헤더에 <code>aws-chunked</code> 값 없이 저장합니다. <code>aws-chunked</code> 가 <code>content-encoding</code> 헤더에 전달하는 유일한 값인 경우 S3는 <code>content-encoding</code> 헤더를 비어 있는 것으로 간주하고 사용자가 객체를 검색할 때 이 헤더를 반환하지 않습니다.
x-amz-decoded-content-length	메타데이터는 포함하지 않고 청크할 데이터의 길이를 바이트 단위로 설정합니다. 예를 들어 4GB 파일을 업로드하는 경우 값을 4294967296으로 설정합니다. 이 값은 업로드할 객체의 원시 크기입니다(Amazon S3에 저장하려는 데이터).
Content-Length	값을 전송된 HTTP 본문의 실제 크기로 설정합니다. 여기에는 데이터 길이(<code>x-amz-decoded-content-length</code>에 설정된 값)와 청크 메타데이터가 포함됩니다. 각 청크에는 이전 청크의 서명과 같은 메타데이터가 있습니다. 청크 계산은 다음 섹션에서 설명합니다. <code>Transfer-Encoding</code> 헤더를 포함하고 <code>identity</code> 이외의 값을 지정하는 경우 <code>Content-Length</code> 헤더를 포함해서는 안 됩니다.

첫 번째 청크를 시드 서명과 함께 전송합니다. 다음 섹션에 설명된 대로 청크를 구성해야 합니다.

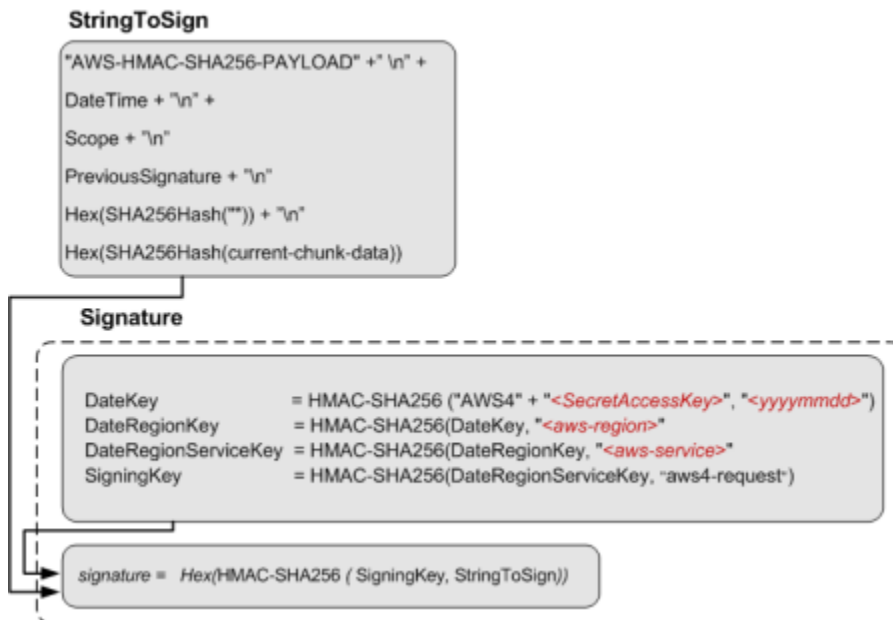
청크 본문 정의

모든 청크에는 일부 메타데이터가 포함됩니다. 각 청크는 다음 구조를 준수해야 합니다.

```
string(IntHexBase(chunk-size)) + ";chunk-signature=" + signature + \r\n + chunk-data + \r\n
```

위치:

- IntHexBase()는 정수 청크 크기를 16진수로 변환하기 위해 작성하는 함수입니다. 예를 들어 청크 크기가 65536인 경우 16진수 문자열은 '10000'입니다.
- *chunk-size*는 메타데이터를 포함하지 않은 청크 데이터의 바이트 단위 크기입니다. 예를 들어 65KB 객체를 업로드하면서 청크 크기를 64KB로 사용하는 경우 데이터를 세 개의 청크로 업로드합니다. 첫 번째 청크는 64KB, 두 번째 청크는 1KB, 마지막 청크는 0바이트입니다.
- *signature* 각 청크에 대해 다음 서명할 문자열을 사용하여 서명을 계산합니다. 첫 번째 청크의 경우 seed-signature를 이전 서명으로 사용합니다.



전송하는 최종 청크 데이터의 크기는 0이지만 청크 본문에는 이전 청크의 서명을 포함한 메타데이터가 여전히 포함되어 있습니다.

예제: PUT Object

이 섹션의 예제는 코드의 서명 계산을 점검하기 위한 참조로 사용할 수 있습니다. 예제를 검토하기 전에 다음 사항에 유의하세요.

- 이러한 예제의 서명 계산은 다음 예제 보안 자격 증명을 사용합니다.

파라미터	값
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

- 모든 예제는 요청 타임스탬프 20130524T000000Z(Fri, 24 May 2013 00:00:00 GMT)를 사용합니다.
- 모든 예제는 examplebucket을 버킷 이름으로 사용합니다.
- 버킷은 미국 동부(버지니아 북부) 리전에 있는 것으로 가정하며, 자격 증명 Scope 및 Signing Key 계산은 us-east-1을 리전 지정자로 사용합니다. 자세한 내용은 Amazon Web Services 일반 참조의 [리전 및 엔드포인트](#)를 참조하세요.
- 경로 방식 또는 가상 호스팅 방식 요청을 사용할 수 있습니다. 다음 예제에서는 가상 호스팅 방식 요청을 사용합니다.

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [버킷의 가상 호스팅](#)을 참조하세요.

다음 예제에서는 객체를 업로드하기 위한 PUT 요청을 전송합니다. 서명 계산은 다음을 가정합니다.

- 65KB 텍스트 파일을 업로드하고 있으며 파일 콘텐츠는 문자 'a'로 구성된 1자 문자열입니다.
- 체크 크기는 64KB입니다. 따라서 페이로드는 세 개의 체크, 즉 64KB 체크, 1KB 체크, 체크 데이터가 0바이트인 최종 체크로 업로드됩니다.
- 결과 객체의 키 이름은 chunk0bject.txt입니다.
- x-amz-storage-class 요청 헤더를 추가하여 REDUCED_REDUNDANCY를 스토리지 클래스로 요청하고 있습니다.

이 API 작업에 대한 자세한 내용은 [PutObject](#)를 참조하세요. 전체 요청 구문은 다음과 같습니다.

```
PUT /examplebucket/chunkObject.txt HTTP/1.1
Host: s3.amazonaws.com
x-amz-date: 20130524T000000Z
x-amz-storage-class: REDUCED_REDUNDANCY
Authorization: SignatureToBeCalculated
x-amz-content-sha256: STREAMING-AWS4-HMAC-SHA256-PAYLOAD
Content-Encoding: aws-chunked
x-amz-decoded-content-length: 66560
Content-Length: 66824
<Payload>
```

다음 단계에서는 서명 계산을 보여줍니다.

1. 시드 서명 - 서명할 문자열 생성

a. CanonicalRequest

```
PUT
/examplebucket/chunkObject.txt

content-encoding:aws-chunked
content-length:66824
host:s3.amazonaws.com
x-amz-content-sha256:STREAMING-AWS4-HMAC-SHA256-PAYLOAD
x-amz-date:20130524T000000Z
x-amz-decoded-content-length:66560
x-amz-storage-class:REDUCED_REDUNDANCY

content-encoding;content-length;host;x-amz-content-sha256;x-amz-date;x-amz-
decoded-content-length;x-amz-storage-class
STREAMING-AWS4-HMAC-SHA256-PAYLOAD
```

정식 요청에서 세 번째 줄은 요청에 쿼리 파라미터가 없기 때문에 비어 있습니다. 마지막 줄은 해시된 페이로드의 값으로 제공된 상수 문자열로, x-amz-content-sha256 header의 값과 동일해야 합니다.

b. StringToSign

```

AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
cee3fed04b70f867d036f722359b0b1f2f0e5dc0efadbc082b76c4c60e316455

```

 Note

서명할 문자열의 각 줄에 대한 자세한 내용은 시드 서명 계산을 설명하는 다이어그램을 참조하세요.

2. SigningKey

```

signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")

```

3. 시드 서명

```

4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9

```

4. Authorization 헤더

결과 Authorization 헤더는 다음과 같습니다.

```

AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/
aws4_request,SignedHeaders=content-encoding;content-length;host;x-amz-
content-sha256;x-amz-date;x-amz-decoded-content-length;x-amz-storage-
class,Signature=4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9

```

5. 청크 1: (65536바이트, 문자 'a'의 경우 값 97)

a. 서명할 청크 문자열:

```

AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z

```

```
20130524/us-east-1/s3/aws4_request
4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
bf718b6f653bebc184e1479f1935b8da974d701b893afcf49e701f3e2f9f9c5a
```

 Note

서명할 문자열의 각 줄에 대한 자세한 내용은 서명할 문자열의 다양한 구성 요소를 보여주는 이전 다이어그램을 참조하세요(예: 마지막 세 줄은 `previous-signature`, `hash("")`, `hash(current-chunk-data)`).

b. 청크 서명:

```
ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
```

c. 전송된 청크 데이터:

```
10000; chunk-
signature=ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
<65536-bytes>
```

6. 청크 2: (1024바이트, 문자 'a'의 경우 값 97)

a. 서명할 청크 문자열:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
2edc986847e209b4016e141a6dc8716d3207350f416969382d431539bf292e4a
```

b. 청크 서명:

```
0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497
```

c. 전송된 청크 데이터:

```
400; chunk-
signature=0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497
```

```
<1024 bytes>
```

7. 청크 3: (0바이트 데이터)

a. 서명할 청크 문자열:

```
AWS4-HMAC-SHA256-PAYLOAD  
20130524T000000Z  
20130524/us-east-1/s3/aws4_request  
0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497  
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855  
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

b. 청크 서명:

```
b6c6ea8a5354eaf15b3cb7646744f4275b71ea724fed81ceb9323e279d449df9
```

c. 전송된 청크 데이터:

```
0; chunk-  
signature=b6c6ea8a5354eaf15b3cb7646744f4275b71ea724fed81ceb9323e279d449df9
```

후행 헤더에 대한 서명 계산(청크 업로드)(AWS 서명 버전 4)

Authorization 헤더를 사용하여 요청을 인증할 때 페이로드를 청크로 업로드할 수도 있습니다. 객체에 대한 데이터를 청크로 전송할 때 후행 헤더를 포함할 수도 있습니다. (자세한 내용은 [Authorization 헤더에 대한 서명 계산: 페이로드를 여러 청크로 전송\(청크 업로드\)\(AWS 서명 버전 4\)](#) 섹션을 참조하세요.) 이 섹션에서는 다중 청크 업로드의 마지막에 후행 헤더를 포함하려는 경우 수행해야 하는 단계를 설명합니다.

Important

후행 헤더를 포함하는 경우 초기 헤더에 다음을 전송해야 합니다.

- x-amz-content-sha256을 후행 헤더가 포함될 것임을 나타내는 적절한 값으로 설정해야 합니다. x-amz-content-sha256에 허용되는 값을 보려면 [요청 인증: Authorization 헤더 사용\(AWS 서명 버전 4\)](#) 섹션을 참조하세요.
- x-amz-trailer를 후행 헤더에 포함하려는 콘텐츠를 나타내도록 설정해야 합니다.

후행 헤더는 청크가 업로드된 후에만 전송됩니다. 이전 청크는 페이로드가 0바이트인 최종 청크 전송을 포함하여 이전 섹션에 설명된 대로 정상적으로 전송되고 서명됩니다. 후행 헤더는 자체 청크로 포함되며 페이로드가 0바이트인 최종 청크 이후에 전송됩니다. 예를 들어 데이터가 100KB 청크로 끝나는 경우 다음을 전송합니다.

- 이전 데이터 청크
- 객체의 100KB 최종 청크
- 객체의 끝을 나타내는 0바이트 청크
- 후행 헤더 청크

예제: 서명 계산 확인

이 섹션의 예제는 코드의 서명 계산을 점검하기 위한 참조로 사용할 수 있습니다. 예제를 검토하기 전에 다음 사항에 유의하세요.

- 이러한 예제의 서명 계산은 다음 예제 보안 자격 증명을 사용합니다.

파라미터	값
AWSSecretAccessKey	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

- 모든 예제는 요청 타임스탬프 20130524T000000Z(Fri, 24 May 2013 00:00:00 GMT)를 사용합니다.
- 모든 예제는 examplebucket을 버킷 이름으로 사용합니다.
- 버킷은 미국 동부(버지니아 북부) 리전에 있는 것으로 가정하며, 자격 증명 Scope 및 Signing Key 계산은 us-east-1을 리전 지정자로 사용합니다. 자세한 내용은 Amazon Web Services 일반 참조의 [리전 및 엔드포인트](#)를 참조하십시오.
- 경로 방식 또는 가상 호스팅 방식 요청을 사용할 수 있습니다. 다음 예제에서는 가상 호스팅 방식 요청을 사용합니다.

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [버킷의 가상 호스팅](#)을 참조하세요.

다음 예제에서는 객체를 업로드하기 위한 PUT 요청을 전송합니다. 서명 계산은 다음을 가정합니다.

- 65KB 텍스트 파일을 업로드하고 있으며 파일 콘텐츠는 문자 'a'로 구성된 1자 문자열입니다.
- 청크 크기는 64KB입니다. 따라서 페이로드는 세 개의 청크, 즉 64KB 청크, 1KB 청크, 청크 데이터가 0바이트인 최종 청크로 업로드됩니다.
- 결과 객체의 키 이름은 chunkObject.txt입니다.
- x-amz-storage-class 요청 헤더를 추가하여 REDUCED_REDUNDANCY를 스토리지 클래스로 요청하고 있습니다.
- 전송에는 CRC32C 체크섬 값이 후행 헤더로 포함됩니다.

이 API 작업에 대한 자세한 내용은 [PutObject](#)를 참조하세요. 전체 요청 구문은 다음과 같습니다.

```
PUT /examplebucket/chunkObject.txt HTTP/1.1
Host: s3.amazonaws.com
```



```
x-amz-date: 20130524T000000Z
x-amz-storage-class: REDUCED_REDUNDANCY
Authorization: SignatureToBeCalculated
x-amz-content-sha256: STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER
Content-Encoding: aws-chunked
x-amz-decoded-content-length: 66560
x-amz-trailer: x-amz-checksum-crc32c
Content-Length: 66946
<Payload>
```

다음 단계에서는 서명 계산을 보여줍니다.

1. 시드 서명 - 서명할 문자열 생성

a. CanonicalRequest

```
PUT
/examplebucket/chunk0bject.txt

content-encoding:aws-chunked
host:s3.amazonaws.com
x-amz-content-sha256:STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER
x-amz-date:20130524T000000Z
x-amz-decoded-content-length:66560
x-amz-storage-class:REDUCED_REDUNDANCY
x-amz-trailer:x-amz-checksum-crc32c

content-encoding;host;x-amz-content-sha256;x-amz-date;x-amz-decoded-content-
length;x-amz-storage-class;x-amz-trailer
STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER
```

정식 요청에서 세 번째 줄은 요청에 쿼리 파라미터가 없기 때문에 비어 있습니다. 마지막 줄은 해시된 페이로드의 값으로 제공된 상수 문자열로, x-amz-content-sha256 header의 값과 동일해야 합니다.

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
```

```
44d48b8c2f70eae815a0198cc73d7a546a73a93359c070abbaa5e6c7de112559
```

Note

서명할 문자열의 각 줄에 대한 자세한 내용은 시드 서명 계산을 설명하는 다이어그램을 참조하세요.

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. 시드 서명

```
106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
```

4. Authorization 헤더

결과 Authorization 헤더는 다음과 같습니다.

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/
aws4_request,SignedHeaders=content-encoding;content-length;host;x-amz-
content-sha256;x-amz-date;x-amz-decoded-content-length;x-amz-storage-
class,Signature=106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
```

5. 청크 1: (65536바이트, 문자 'a'의 경우 값 97)

a. 서명할 청크 문자열:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
bf718b6f653bebc184e1479f1935b8da974d701b893afcf49e701f3e2f9f9c5a
```

Note

서명할 문자열의 각 줄에 대한 자세한 내용은 서명할 문자열의 다양한 구성 요소를 보여주는 이전 주제([시드 서명 계산](#))의 다이어그램을 참조하세요. 예를 들어 마지막 세 줄은 다음으로 구성됩니다.

- `## ##`
- `hash("")`
- `hash(current-chunk-data)`

b. 청크 서명:

```
b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
```

c. 전송된 청크 데이터:

```
10000;chunk-  
signature=b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2  
<65536-bytes>
```

6. 청크 2: (1024바이트, 문자 'a'의 경우 값 97)

a. 서명할 청크 문자열:

```
AWS4-HMAC-SHA256-PAYLOAD  
20130524T000000Z  
20130524/us-east-1/s3/aws4_request  
b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2  
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855  
2edc986847e209b4016e141a6dc8716d3207350f416969382d431539bf292e4a
```

b. 청크 서명:

```
1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
```

c. 전송된 청크 데이터:

```
400;chunk-  
signature=1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
```

```
<1024-bytes>
```

7. 청크 3: (0바이트 데이터)

a. 서명할 청크 문자열:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

b. 청크 서명:

```
2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
```

c. 전송된 청크 데이터:

```
0; chunk-
signature=2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
```

8. 청크 4: 후행 헤더

a. 서명할 트레일러 청크 문자열:

```
AWS4-HMAC-SHA256-TRAILER
20130524T000000Z
20130524/us-east-1/s3/aws4_request
2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
1e376db7e1a34a8ef1c4bcee131a2d60a1cb62503747488624e10995f448d774
```

Note

마지막 두 줄은 `previous-signature`(0바이트 데이터 청크 서명),
`hash(trailing-checksum-header-name:base64-encoded-trailing-checksum-value\n)`입니다.
 해시는 공백 없이 다음과 같이 계산됩니다.

- 후행 체크섬 헤더 이름
- 콜론(:)

- Base64로 인코딩된 후행 체크섬 값
- 줄 바꿈 문자(\n).

이 예제에서는 crc32c 해시 알고리즘, Base64로 인코딩된 체크섬 값 s008/Q==를 사용하여 계산을 다음과 같이 나타낼 수 있습니다. `hash('x-amz-checksum-crc32c:s008/Q==\n')`.

b. 청크 서명:

```
d81f82fc3505edab99d459891051a732e8730629a2e4a59689829ca17fe2e435
```

c. 전송된 청크 데이터:

```
x-amz-checksum-crc32c:s008/Q==
x-amz-trailer-
signature:d81f82fc3505edab99d459891051a732e8730629a2e4a59689829ca17fe2e435
```

요청 인증: 쿼리 파라미터 사용(AWS 서명 버전 4)

인증 개요([인증 방법](#) 참조)에 설명된 대로 쿼리 문자열 파라미터를 사용하여 인증 정보를 제공할 수 있습니다. 쿼리 파라미터를 사용하여 요청을 인증하는 것은 요청을 하나의 URL에서 완전히 표현하려는 경우에 유용합니다. 이 방법을 URL 미리 서명이라고도 합니다.

미리 서명된 URL의 사용 사례 시나리오는 Amazon S3 리소스에 대한 임시 액세스 권한을 부여할 수 있다는 것입니다. 예를 들어 웹 사이트에 미리 서명된 URL을 포함하거나 명령줄 클라이언트(예: Curl)에 이를 사용하여 객체를 다운로드할 수 있습니다.

Note

AWS CLI를 사용하여 미리 서명된 URL을 생성할 수도 있습니다. 자세한 내용은 AWS CLI 명령 참조의 [presign](#) 섹션을 참조하십시오.

다음은 미리 서명된 URL의 예입니다.

```
https://examplebucket.s3.amazonaws.com/test.txt
```

```
?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=<your-access-key-id>/20130721/us-east-1/s3/aws4_request
&X-Amz-Date=20130721T201207Z
&X-Amz-Expires=86400
&X-Amz-SignedHeaders=host
&X-Amz-Signature=<signature-value>
```

예제 URL에서 다음 사항에 유의하세요.

- 가독성을 위해 라인 피드가 추가되어 있습니다.
- URL의 X-Amz-Credential 값은 가독성을 위해 '/' 문자만 표시합니다. 실제로는 %2F로 인코딩되어야 합니다. 예제:

```
&X-Amz-Credential=<your-access-key-id>%2F20130721%2Fus-east-1%2Fs3%2Faws4_request
```

다음 표에서는 인증 정보를 제공하는 URL의 쿼리 파라미터를 설명합니다.

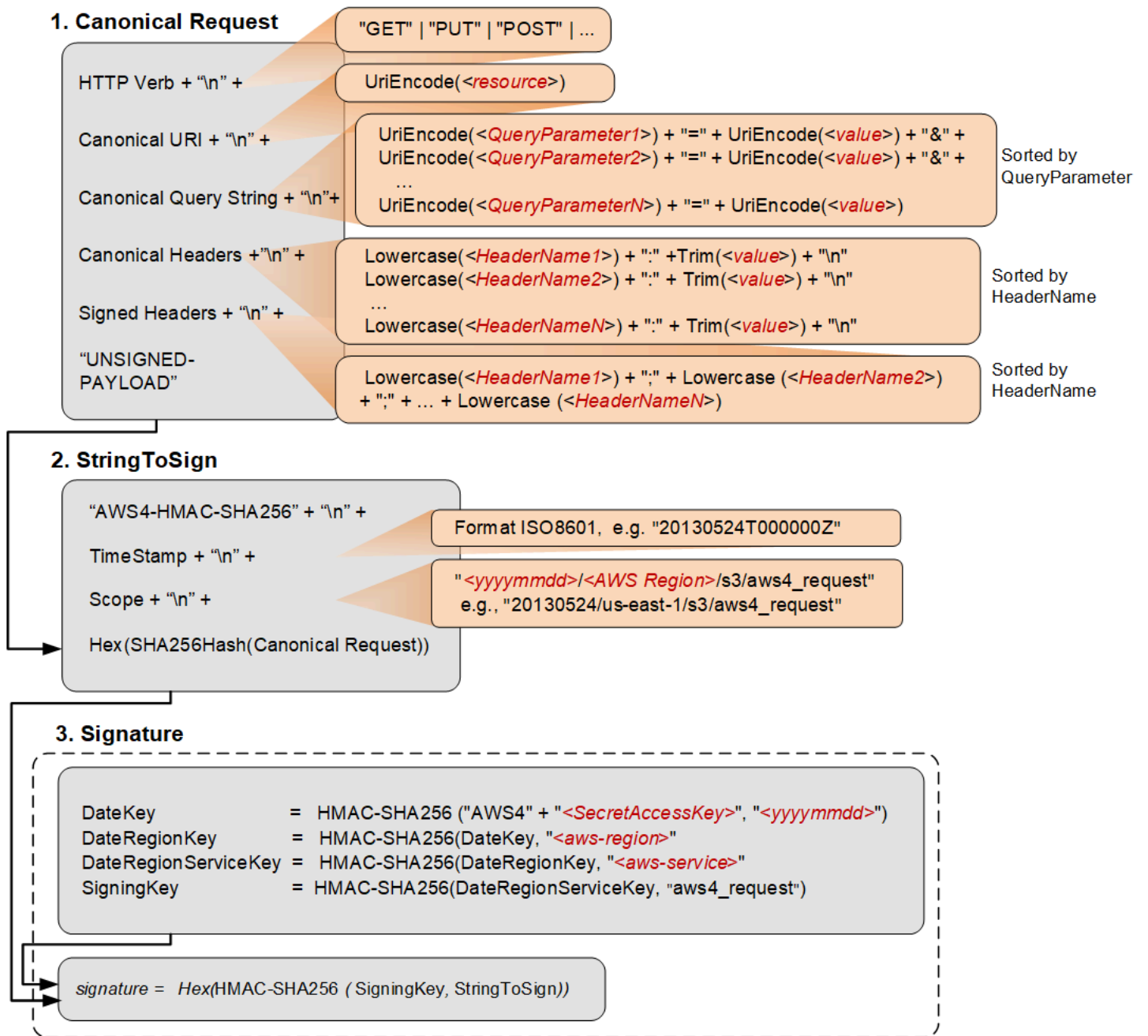
쿼리 문자열 파라미터 이름	예제 값
X-Amz-Algorithm	서명 계산에 사용한 AWS 서명 버전과 알고리즘을 식별합니다. AWS 서명 버전 4의 경우 이 파라미터 값을 AWS4-HMAC-SHA256으로 설정합니다. 이 문자열은 AWS 서명 버전 4(AWS4) 및 HMAC-SHA256 알고리즘(HMAC-SHA256)을 식별합니다.
X-Amz-Credential	이 파라미터는 액세스 키 ID 외에도 서명이 유효한 범위(AWS 리전 및 서비스)를 제공합니다. 이 값은 다음 섹션의 설명과 같이 서명 계산에 사용하는 범위와 일치해야 합니다. 이 파라미터의 값의 일반적인 형식은 다음과 같습니다. <pre><your-access-key-id> /<date>/<AWS ##>/<AWS-service> / aws4_request</pre> 예제:

쿼리 문자열 파라미터 이름	예제 값
	AKIAIOSFODNN7EXAMPLE/20130721/us-east-1/s3/aw s4_request Amazon S3의 경우 AWS-service 문자열은 s3입니다. S3 AWS-region 문자열의 목록은 AWS 일반 참조의 리전 및 엔드 포인트를 참조하세요.
X-Amz-Date	날짜 및 시간 형식은 ISO 8601 표준을 따르고, 'yyyyMMddTHHmmsZ' 형식을 사용해야 합니다. 예를 들어 날짜와 시간이 '08/01/2016 15:32:41.982-700'인 경우 UTC(협정 세계시)로 변환하여 '20160801T223241Z'로 제출해야 합니다.
X-Amz-Expires	생성된 미리 서명된 URL이 유효한 기간(초)을 제공합니다. 예: 86400(24시간). 이 값은 정수입니다. 설정할 수 있는 최소값은 1 이고 최대값은 604800(7일)입니다. 서명 계산에 사용하는 서명 키는 최대 7일 동안 유효하므로 미리 서명된 URL은 최대 7일 동안 유효할 수 있습니다.
X-Amz-SignedHeaders	서명 계산에 사용한 헤더를 나열합니다. 서명 계산에는 다음 헤더가 필요합니다. • HTTP host 헤더. • 요청에 추가하려는 모든 x-amz-* 헤더.  Note 보안을 강화하려면 요청에 포함하려는 모든 요청 헤더에 서명해야 합니다.

쿼리 문자열 파라미터 이름	예제 값
X-Amz-Signature	요청을 인증하기 위한 서명을 제공합니다. 이 서명은 Amazon S3가 계산하는 서명과 일치해야 합니다. 그렇지 않으면 Amazon S3가 요청을 거부합니다. 예: 733255ef022bec3f2a8701cd61d4b371f3f28c9f193a1f02279211d48d5193d7 서명 계산은 다음 섹션에서 설명합니다.
X-Amz-Security-Token	STS 서비스에서 가져온 보안 인증 정보를 사용하는 경우의 선택적 보안 자격 증명 파라미터입니다.

서명 계산

다음 다이어그램은 서명 계산 프로세스를 보여줍니다.



다음 표에서는 다이어그램에 표시된 함수를 설명합니다. 이러한 함수의 코드를 구현해야 합니다.

함수	설명
Lowercase()	문자열을 소문자로 변환합니다.
Hex()	소문자 16진법 인코딩.

함수	설명
SHA256Hash()	보안 해시 알고리즘(SHA) 암호화 해시 함수.
HMAC-SHA256()	제공된 서명 키를 포함한 SHA256 알고리즘을 사용하여 HMAC를 계산합니다. 이것이 최종 서명입니다.
Trim()	선행 또는 후행 공백을 모두 제거합니다.

함수	설명
UriEncode()	URI는 모든 바이트를 인코딩합니다. UriEncode()는 다음 규칙을 적용해야 합니다. - URI는 예약되지 않은 문자를 제외한 모든 바이트를 인코딩합니다. 'A'-'Z', 'a'-'z', '0'-'9', '-', '.', '_' 및 '~'. - 공백 문자는 예약된 문자이고 '+'가 아닌 '%20'로 인코딩되어야 합니다. - 각 URI 인코딩 바이트는 '%' 및 바이트의 두 자리 16진수 값으로 구성됩니다. 16진수 값의 문자는 대문자여야 합니다(예: '%1A'). - 슬래시 문자('/')를 객체 키 이름을 제외한 모든 위치에 인코딩합니다. 예를 들어, 객체 키 이름이 photos/Jan/sample.jpg 인 경우 키 이름의 슬래시는 인코딩되지 않습니다.  Important 개발 플랫폼에서 제공하는 표준 UriEncode 함수는 구현상의 차이와 기본 RFC의 관련 모호성으로 인해 효과적이지 않을 수 있습니다. 인코딩이 제대로 작동하도록 사용자 지정 UriEncode 함수를 직접 작성하는 것이 좋습니다. 다음은 Java로 작성된 UriEncode() 함수 예제입니다. <pre> public static String UriEncode(CharSequence input, boolean encodeSlash) { StringBuilder result = new StringBui lder(); for (int i = 0; i < input.length(); i++) { char ch = input.charAt(i); if ((ch >= 'A' && ch <= 'Z') (ch >= 'a' && ch <= 'z') (ch >= '0' && ch <= '9') ch == '_' ch == '-' ch == '~' ch == '.') { result.append(ch); } else if (ch == '/') { </pre>

함수	설명
	<pre> result.append(encodeSlash ? "%2F" : ch); } else { result.append(toHexUTF8(ch)); } } return result.toString(); } </pre>

서명 프로세스에 대한 자세한 내용(정식 요청 생성, 서명할 문자열 및 서명 계산에 대한 세부 정보)은 [Authorization 헤더에 대한 서명 계산: 페이로드를 단일 청크로 전송\(AWS 서명 버전 4\)](#) 섹션을 참조하세요. 미리 서명된 URL에서 CanonicalRequest를 생성하는 것이 다음과 같이 다르다는 점을 제외하면 프로세스는 전반적으로 동일합니다.

- 미리 서명된 URL을 생성할 때는 임의의 페이로드를 업로드하는 데 해당 URL이 사용되므로 페이로드 콘텐츠를 알 수 없습니다. 따라서 정식 요청에 페이로드 해시를 포함하지 않습니다. 대신 상수 문자열 UNSIGNED-PAYLOAD를 사용합니다.
- 정식 쿼리 문자열에는 X-Amz-Signature를 제외하고 위 표의 모든 쿼리 파라미터가 포함되어야 합니다.
- S3의 경우 STS 서비스에서 가져온 자격 증명을 사용한다면 URL에 X-Amz-Security-Token 쿼리 파라미터를 포함해야 합니다.
- 정식 헤더에는 HTTP host 헤더가 포함되어야 합니다. x-amz-* 헤더를 포함하려는 경우 서명 계산을 위해 이러한 헤더도 추가해야 합니다. 선택적으로 요청에 포함하려는 다른 모든 헤더를 추가할 수 있습니다. 보안을 강화하려면 가능한 한 많은 헤더에 서명해야 합니다. 서명된 쿼리 파라미터이기도 한 서명된 헤더를 추가하는 경우 값이 서로 다르면 입력이 충돌하여 InvalidRequest 오류가 발생합니다.

예제

examplebucket 버킷에 test.txt 객체가 있다고 가정해 보겠습니다. 미리 서명된 URL을 생성하여 24시간(86,400초) 동안 이 객체를 다른 사람과 공유하려고 합니다.

```
https://examplebucket.s3.amazonaws.com/test.txt
```

```
?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request
&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host
&X-Amz-Signature=<signature-value>
```

다음 단계에서는 먼저 서명 계산을 설명한 다음 미리 서명된 URL 구성을 설명합니다. 이 예제에서는 다음을 추가로 가정합니다.

- 요청 타임스탬프는 Fri, 24 May 2013 00:00:00 GMT입니다.
- 버킷은 미국 동부(버지니아 북부) 리전에 있으며, 자격 증명 Scope 및 Signing Key 계산은 us-east-1을 리전 지정자로 사용합니다. 자세한 내용은 AWS 일반 참조의 [리전 및 엔드포인트](#)를 참조하세요.

이 예제를 테스트 사례로 사용하여 코드가 계산하는 서명을 확인할 수 있습니다. 단, 동일한 버킷 이름, 객체 키, 타임스탬프 및 다음 예제 자격 증명을 사용해야 합니다.

파라미터	값
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

1. StringToSign

a. CanonicalRequest

```
GET
/test.txt
X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE
%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host
host:examplebucket.s3.amazonaws.com

host
UNSIGNED-PAYLOAD
```

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
3bfa292879f6447bbcd47001decf97f4a54dc650c8942174ae0a9121cf58ad04
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
  "<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. 서명

```
aeed9bbccd4d02ee5c0109b86d86835f995330da4c265957d157751f604d404
```

이제 미리 서명된 URL을 구성하는 데 필요한 모든 정보를 얻었습니다. 이 예제의 결과 URL은 다음과 같습니다(이 URL을 사용하여 미리 서명된 URL을 비교할 수 있음).

```
https://examplebucket.s3.amazonaws.com/test.txt?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-
Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-
Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Amz-
Signature=aeed9bbccd4d02ee5c0109b86d86835f995330da4c265957d157751f604d404
```

예제 2

다음은 X-Amz-Security-Token 파라미터와 함께 미리 서명된 URL을 보여주는 예제입니다(이전 예제와 관련이 없음).

```
https://examplebucket.s3.us-east-1.amazonaws.com/test.txt  
?X-Amz-Algorithm=AWS4-HMAC-SHA256  
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request  
&X-Amz-Date=20200524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host  
&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEMv%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJGMEQCIBSubvdj9YGs2q0HkHsOHfDkwOozjARSKHL987NhhOC8AiBPepRU1obMvIbGUOT
```

[illegible]

예제: AWS 서명 버전 4의 서명 계산

주제

- Java를 사용한 서명 계산 예제(AWS 서명 버전 4)
- C#을 사용한 서명 계산 예제(AWS 서명 버전 4)

인증된 요청의 경우, AWS SDK를 사용하지 않는 한 요청에 인증 정보를 제공하는 서명을 계산하는 코드를 작성해야 합니다. AWS 서명 버전 4([요청 인증\(AWS 서명 버전 4\)](#) 참조)의 서명 계산은 복잡한 작업일 수 있으므로 가능한 경우 AWS SDK를 사용하는 것이 좋습니다.

이 섹션에서는 Java 및 C#으로 작성된 서명 계산 예제를 제공합니다. 코드 샘플은 다음 요청을 전송하고 HTTP Authorization 헤더를 사용하여 인증 정보를 제공합니다.

- PUT object – 개별 예제가 전체 페이로드를 한 번에 업로드하는 방법과 페이로드를 청크로 업로드하는 방법을 모두 보여줍니다. 인증에 Authorization 헤더를 사용하는 방법에 대한 자세한 내용은 [요청 인증: Authorization 헤더 사용\(AWS 서명 버전 4\)](#) 섹션을 참조하세요.
- GET object – 이 예제에서는 미리 서명된 URL을 생성하여 객체를 가져옵니다. 쿼리 파라미터는 서명 및 기타 인증 정보를 제공합니다. 사용자는 브라우저에 미리 서명된 URL을 붙여 넣어 객체를 검색하거나 URL을 사용하여 클릭 가능한 링크를 생성할 수 있습니다. 인증에 쿼리 파라미터를 사용하는 방법에 대한 자세한 내용은 [요청 인증: 쿼리 파라미터 사용\(AWS 서명 버전 4\)](#) 섹션을 참조하세요.

이 섹션의 나머지 부분에서는 Java 및 C#으로 작성된 예제를 설명합니다. 주제에는 샘플 다운로드 및 실행을 위한 지침이 포함되어 있습니다.

Java를 사용한 서명 계산 예제(AWS 서명 버전 4)

서명 계산을 보여주는 Java 샘플은 <https://docs.aws.amazon.com/AmazonS3/latest/API/samples/AWSS3SigV4JavaSamples.zip>에서 다운로드할 수 있습니다. 이러한 샘플은 AWS SDK for Java v1을 사용하므로 새 애플리케이션에는 [AWS SDK for Java v2](#)를 사용하는 것이 좋습니다. RunAllSamples.java에서 main() 함수는 샘플 요청을 실행하여 객체를 생성하고, 객체를 검색하고, 객체에 대해 미리 서명된 URL을 생성합니다. 샘플은 코드에 제공된 텍스트 문자열에서 객체를 생성합니다.

```
PutS3ObjectSample.putS3Object(bucketName, regionName, awsAccessKey, awsSecretKey);
GetS3ObjectSample.getS3Object(bucketName, regionName, awsAccessKey, awsSecretKey);
PresignedUrlSample.getPresignedUrlToS3Object(bucketName, regionName, awsAccessKey,
awsSecretKey);
PutS3ObjectChunkedSample.putS3ObjectChunked(bucketName, regionName, awsAccessKey,
awsSecretKey);
```

Linux 기반 컴퓨터에서 예제를 테스트하려면

다음은 Linux 운영 체제용 지침입니다.

1. 터미널에서 AWSS3SigV4JavaSamples.zip이 포함된 디렉터리로 이동합니다.
2. .zip 파일의 압축을 풉니다.
3. 텍스트 편집기에서 ./com/amazonaws/services/s3/samples/RunAllSamples.java 파일을 엽니다. 코드를 다음 정보로 업데이트합니다.
 - 새 객체를 생성할 수 있는 버킷의 이름입니다.

Note

이 예제에서는 가상 호스팅 방식 요청을 사용하여 버킷에 액세스합니다. 잠재적 오류를 방지하려면 Amazon Simple Storage Service 사용 설명서의 [버킷 규제 및 제한](#)에 설명된 대로 버킷 이름이 버킷 이름 지정 규칙을 준수하는지 확인하세요.

- AWS 리전 버킷이 상주하는 위치입니다.

버킷이 미국 동부(버지니아 북부) 리전에 있는 경우 us-east-1을 사용하여 리전을 지정합니다. 다른 AWS 리전의 목록은 AWS 일반 참조의 [Amazon Simple Storage Service\(S3\)](#)를 참조하세요.

4. 소스 코드를 컴파일하고 컴파일된 클래스를 bin/ 디렉터리에 저장합니다.


```
javac -d bin -source 6 -verbose com
```

5. 디렉터리를 bin/으로 변경한 다음 RunAllSamples를 실행합니다.

```
java com.amazonaws.services.s3.sample.RunAllSamples
```

코드는 main()의 메서드를 모두 실행합니다. 각 요청에 대해 출력에 정식 요청, 서명할 문자열 및 서명이 표시됩니다.

C#을 사용한 서명 계산 예제(AWS 서명 버전 4)

서명 계산을 보여주는 C# 샘플은 https://docs.aws.amazon.com/AmazonS3/latest/API/samples/AmazonS3SigV4_Samples_CSharp.zip에서 다운로드할 수 있습니다. 이러한 샘플은 AWS SDK for .NET v1을 사용하므로 새 애플리케이션에는 [AWS SDK for .NET v3](#)을 사용하는 것이 좋습니다. Program.cs에서 main() 함수는 샘플 요청을 실행하여 객체를 생성하고, 객체를 검색하고, 객체에 대해 미리 서명된 URL을 생성합니다. 서명 계산을 위한 코드는 \Signers 폴더에 있습니다.

```
PutS3ObjectSample.Run(awsRegion, bucketName, "MySampleFile.txt");

Console.WriteLine("\n\n*****");
PutS3ObjectChunkedSample.Run(awsRegion, bucketName, "MySampleFileChunked.txt");

Console.WriteLine("\n\n*****");
GetS3ObjectSample.Run(awsRegion, bucketName, "MySampleFile.txt");

Console.WriteLine("\n\n*****");
PresignedUrlSample.Run(awsRegion, bucketName, "MySampleFile.txt");
```

Microsoft Visual Studio 2010 이상을 사용하여 예제를 테스트하려면

1. .zip 파일의 압축을 풉니다.
2. Visual Studio를 시작한 다음 .sln 파일을 엽니다.
3. App.config 파일을 유효한 보안 자격 증명으로 업데이트합니다.
4. 다음과 같이 코드를 업데이트합니다.

- Program.cs에서 버킷 이름과 버킷이 상주하는 AWS 리전을 입력합니다. 샘플은 이 버킷에 객체를 생성합니다.
5. 코드를 실행합니다.
 6. 객체가 생성되었는지 확인하려면 프로그램이 생성하는 미리 서명된 URL을 복사하여 브라우저 창에 붙여 넣습니다.

요청 인증: POST를 사용한 브라우저 기반 업로드(AWS 서명 버전 4)

Amazon S3는 사용자가 Amazon S3에 직접 콘텐츠를 업로드할 수 있도록 HTTP POST 요청을 지원합니다. HTTP POST를 사용하여 콘텐츠를 업로드하면 사용자가 Amazon S3에 저장할 데이터를 업로드할 때 업로드 과정이 간소화되고 업로드 지연 시간이 단축됩니다. 이 섹션에서는 HTTP POST 요청을 인증하는 방법을 설명합니다. HTTP POST 요청, 양식 생성 방법, POST 정책 생성 방법 및 예제에 대한 자세한 내용은 [POST를 사용한 브라우저 기반 업로드\(AWS 서명 버전 4\)](#) 섹션을 참조하세요.

HTTP POST 요청을 인증하려면 다음을 수행합니다.

1. 양식에는 Amazon S3가 요청을 수신하면 서명을 다시 계산하는 데 사용할 수 있는 서명 및 관련 정보를 제공하는 다음 필드가 포함되어야 합니다.

요소 이름	설명
policy	요청에서 허용되는 항목을 설명하는 Base64로 인코딩된 보안 정책입니다. 서명 계산의 경우 이 정책은 서명하는 문자열입니다. Amazon S3는 서명을 다시 계산할 수 있도록 이 정책을 가져와야 합니다.
x-amz-algorithm	사용된 서명 알고리즘입니다. AWS 서명 버전 4의 경우 값은 AWS4-HMAC-SHA256 입니다.
x-amz-credential	액세스 키 ID 외에도 서명 계산을 위한 서명 키 계산에 사용한 범위 정보를 제공합니다. 이는 다음 형식의 문자열입니다.

요소 이름	설명
	<code><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</code> 예제: <code>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request . .</code> Amazon S3의 경우 aws-service 문자열은 s3입니다. Amazon S3 aws-region 문자열의 목록은 AWS 일반 참조의 리전 및 엔드포인트를 참조하세요.
x-amz-date	ISO8601 형식의 날짜 값입니다. 예를 들어 20130728T000000Z 입니다. 이 값은 서명 키를 생성할 때 사용한 날짜와 동일합니다. 또한 서명한 정책(x-amz-date)에 제공한 값과도 동일해야 합니다.
x-amz-signature	(AWS 서명 버전 4) 보안 정책의 HMAC-SHA256 해시입니다. 서명 옵션에 대한 자세한 내용은 AWS 일반 참조의 HTTP 요청에 서명 추가를 참조하세요.

2. POST 정책은 다음 요소를 포함해야 합니다.

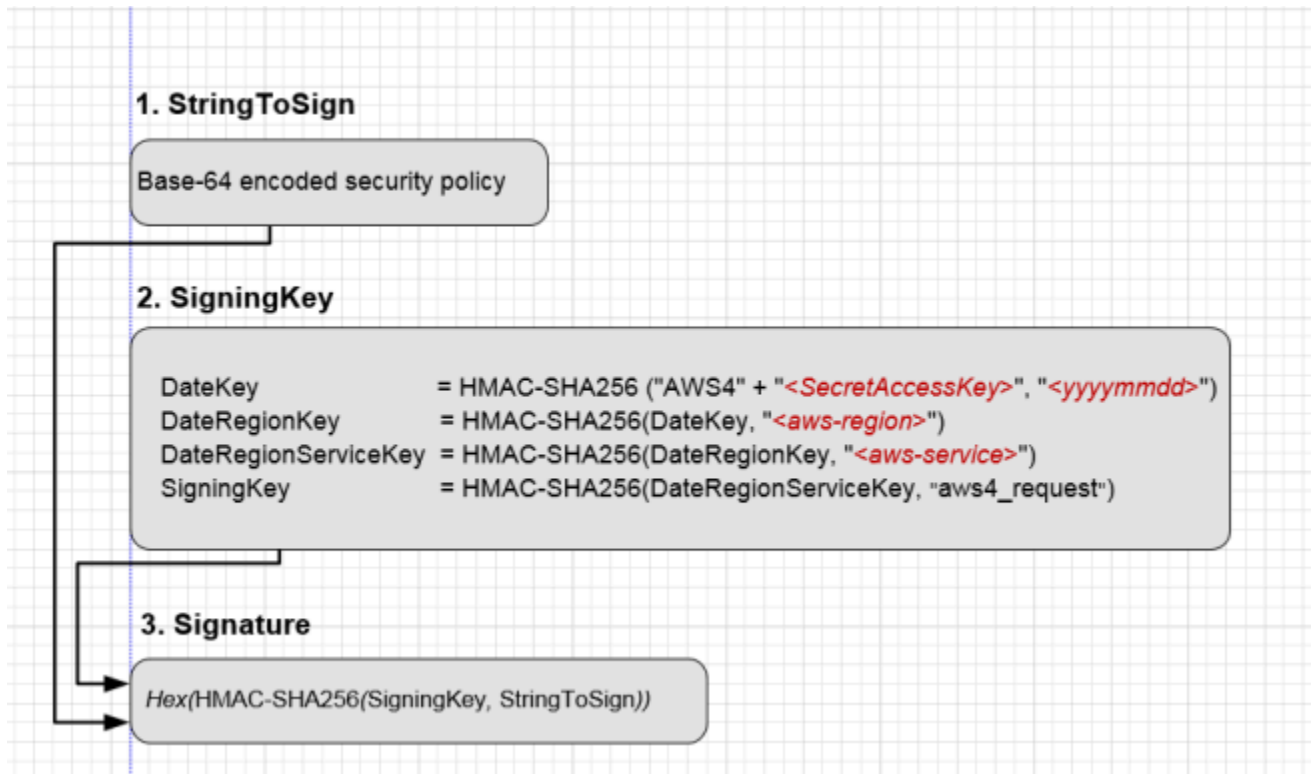
요소 이름	설명
x-amz-algorithm	서명 계산에 사용한 알고리즘입니다. AWS 서명 버전 4의 경우 값은 AWS4-HMAC-SHA256 입니다.
x-amz-credential	액세스 키 ID 외에도 서명 계산을 위한 서명 키 계산에 사용한 범위 정보를 제공합니다.

요소 이름	설명
	이는 다음 형식의 문자열입니다. <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> 예: <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request . .</pre>
x-amz-date	ISO8601 형식 문자열에 지정된 날짜 값입니다. 예를 들어 '20130728T000000Z'입니다. 이 날짜는 서명 계산을 위한 서명 키를 생성할 때 사용한 날짜와 동일해야 합니다.

3. 서명 계산의 경우 POST 정책은 서명할 문자열입니다.

서명 계산

다음 다이어그램은 서명 계산 프로세스를 보여줍니다.



서명을 계산하려면

1. UTF-8 인코딩을 사용하여 정책을 생성합니다.
2. UTF-8로 인코딩된 정책을 Base64로 변환합니다. 결과는 서명할 문자열입니다.
3. 서명할 문자열의 HMAC-SHA256 해시로 서명을 생성합니다. 서명 키를 해시 함수의 키로 제공합니다.
4. 16진수 인코딩을 사용하여 서명을 인코딩합니다.

HTML 양식, 보안 정책 및 예제 생성에 대한 자세한 내용은 다음 하위 주제를 참조하세요.

- [HTML 양식 생성\(AWS 서명 버전 4 사용\)](#)
- [POST 정책](#)
- [예제: HTTP POST를 사용한 브라우저 기반 업로드\(AWS 서명 버전 4 사용\)](#)

Amazon S3 서명 버전 4 인증 관련 정책 키

다음 표에는 Amazon S3 정책에 있을 수 있는 Amazon S3 서명 버전 4 인증과 관련된 정책 키가 나와 있습니다. 버킷 정책에서 이러한 조건을 추가하여 서명 버전 4를 사용하여 요청이 인증될 때 특정 동작을 적용할 수 있습니다. 예시 정책은 [서명 버전 4 관련 조건 키를 사용하는 버킷 정책 예제](#) 섹션을 참조하세요.

적용 가능한 키	설명
s3:signatureversion	인증된 요청에 대해 지원하려는 AWS Signature의 버전을 식별합니다. 인증된 요청의 경우 Amazon S3는 서명 버전 4와 서명 버전 2를 모두 지원합니다. 버킷 정책에 이 조건을 추가하여 특정 서명 버전을 요구할 수 있습니다. 유효한 값: "AWS"는 서명 버전 2를 식별함 "AWS4-HMAC-SHA256" 은 서명 버전 4를 식별함
s3:authType	Amazon S3는 다양한 인증 방법을 지원합니다 (요청 인증(AWS 서명 버전 4) 참조). 특정 인증 방법을 사용하도록 수신 요청을 제한하려면 선택적으로 이 조건 키를 사용할 수 있습니다. 예를 들어 요청 인증에 HTTP Authorization 헤더만 사용하도록 허용할 수 있습니다. 유효한 값: REST-HEADER REST-QUERY-STRING POST
s3:signatureAge	인증된 요청에서 서명이 유효한 시간(밀리초)입니다.

적용 가능한 키	설명
	이 조건은 다음에 적용됩니다. • 미리 서명된 URL - 가장 제한적인 조건이 적용됩니다. 자세한 내용은 미리 서명된 URL로 작업을 참조하세요. • 미리 서명된 POST - 미리 서명된 POST를 사용하여 파일을 S3에 직접 업로드합니다. 자세한 내용은 Amazon S3 POST 정책을 참조하세요. 서명 버전 2에서는 이 값이 항상 0으로 설정됩니다. 서명 버전 4에서는 서명 키가 최대 7일간 유효합니다. 따라서 서명도 최대 7일간 유효합니다. 이 조건을 사용하여 서명 연령을 추가로 제한할 수 있습니다. 자세한 내용은 서명 요청 소개 섹션을 참조하세요. 예시 값: 100

적용 가능한 키	설명
s3:x-amz-content-sha256	이 조건 키를 사용하여 버킷에서 서명되지 않은 콘텐츠를 허용하지 않을 수 있습니다. 서명 버전 4를 사용할 때 Authorization 헤더를 사용하는 요청에 대해 서명 계산에 x-amz-content-sha256 헤더를 추가한 다음 해당 값을 해시 페이로드로 설정합니다. 이 조건 키는 x-amz-content-sha256 헤더를 쿼리 문자열 파라미터로 지원하지 않습니다. 버킷 정책에서 이 조건 키를 사용하여 페이로드에 서명하지 않은 업로드를 거부할 수 있습니다. 예를 들어 Authorization 헤더를 사용하여 요청을 인증하지만 페이로드에 서명하지 않는 업로드를 거부합니다. 자세한 내용은 Authorization 헤더에 대한 서명 계산: 페이로드를 단일 청크로 전송(AWS 서명 버전 4) 섹션을 참조하세요. 유효한 값: UNSIGNED-PAYLOAD

서명 버전 4 관련 조건 키를 사용하는 버킷 정책 예제

다음 버킷 정책은 서명이 10분 이상 지난 경우 examplebucket의 객체에 대한 Amazon S3 미리 서명된 URL 요청을 모두 거부합니다.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Deny a presigned URL request if the signature is more than 10 min old",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*"
```



```

    "Resource": "arn:aws:s3:::examplebucket3/*",
    "Condition": {
      "NumericGreaterThan": {
        "s3:signatureAge": 600000
      }
    }
  }
]
}

```

다음 버킷 정책은 요청 인증에 Authorization 헤더를 사용하는 요청만 허용합니다. 모든 POST 또는 미리 서명된 URL 요청은 거부됩니다.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Allow only requests that use Authorization header for request authentication. Deny POST or presigned URL requests.",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::examplebucket3/*",
      "Condition": {
        "StringNotEquals": {
          "s3:authType": "REST-HEADER"
        }
      }
    }
  ]
}

```

다음 버킷 정책은 요청 인증에 미리 서명된 URL을 사용하는 요청을 거부합니다.

JSON

```

{

```

```

"Version":"2012-10-17",
"Statement":[
  {
    "Sid":"DenyUploadsUsingPresignedURL",
    "Effect":"Deny",
    "Principal":"*",
    "Action":"s3:*",
    "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/*",
    "Condition":{"
      "StringEquals":{"
        "s3:authType":"REST-query-string"
      }
    }
  }
]
}

```

다음 버킷 정책은 서명되지 않은 페이로드가 있는 업로드를 거부합니다.

JSON

```

{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "Deny uploads with unsigned payloads that use the
Authorization header.",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::examplebucket3/*",
      "Condition": {
        "StringEquals": {
          "s3:x-amz-content-sha256": "UNSIGNED-PAYLOAD"
        }
      }
    }
  ]
}

```

POST를 사용한 브라우저 기반 업로드(AWS 서명 버전 4)

이 섹션에서는 HTTP POST 요청을 사용하여 브라우저를 통해 Amazon S3에 직접 파일을 업로드하는 방법을 설명합니다. 또한 Amazon S3에 대한 브라우저 기반 파일 업로드에 AWS Amplify JavaScript 라이브러리를 사용하는 방법에 대한 정보도 포함되어 있습니다.

주제

- [POST Object](#)
- [POST Object restore](#)
- [HTTP POST를 사용한 브라우저 기반 업로드](#)
- [서명 계산](#)
- [HTML 양식 생성\(AWS 서명 버전 4 사용\)](#)
- [POST 정책](#)
- [예제: HTTP POST를 사용한 브라우저 기반 업로드\(AWS 서명 버전 4 사용\)](#)
- [AWS Amplify 라이브러리를 사용하여 Amazon S3에 브라우저 기반 업로드](#)

POST Object

설명

POST 작업은 HTML 양식을 사용하여 지정된 버킷에 객체를 추가합니다. POST는 버킷에 객체를 넣는 방법으로, 브라우저 기반 업로드를 지원하는 PUT의 대체 형식입니다. HTTP 헤더를 통해 PUT에 전달되는 파라미터는 대신 multipart/form-data로 인코딩된 메시지 본문의 POST에 양식 필드로 전달됩니다. 버킷에 객체를 추가하려면 버킷에 대한 WRITE 액세스 권한이 있어야 합니다. Amazon S3는 부분 객체를 저장하지 않습니다. 성공 응답을 받으면 전체 객체가 저장되었다고 확인할 수 있습니다.

Amazon S3는 분산 시스템입니다. 버킷에 대해 버전 관리를 활성화하지 않은 한, Amazon S3가 동일한 객체에 대해 여러 쓰기 요청을 동시에 수신하는 경우 작성된 객체의 마지막 버전만 저장됩니다.

데이터가 네트워크를 통과하는 동안 손상되지 않도록 하려면 Content-MD5 양식 필드를 사용합니다. 이 양식 필드를 사용하면 Amazon S3는 제공된 MD5 값을 기준으로 객체를 확인합니다. 일치하지 않으면 Amazon S3가 오류를 반환합니다. 또한 Amazon S3에 객체를 게시하는 동안 MD5 값을 계산하고 반환된 ETag를 계산된 MD5 값과 비교할 수 있습니다. ETag는 객체의 콘텐츠에 대한 변경 사항만 반영하고 메타데이터에 대한 변경을 반영하지 않습니다.

Note

요청 본문을 전송하기 전에 요청 헤더를 전송하도록 애플리케이션을 구성하려면 HTTP 상태 코드 100(계속)을 사용합니다. POST 작업의 경우 이 상태 코드를 사용하면 헤더를 기반으로 메시지가 거부되는 경우(예: 인증 실패 또는 리디렉션) 메시지 본문이 전송되지 않도록 할 수 있습니다. HTTP 상태 코드 100(계속)에 대한 자세한 내용은 <http://www.ietf.org/rfc/rfc2616.txt>의 섹션 8.2.3을 참조하세요.

Amazon S3는 S3 버킷에 업로드되는 모든 새 객체를 자동으로 암호화합니다. 업로드된 객체의 암호화 설정은 대상 버킷의 기본 암호화 구성에 따라 달라집니다. 기본적으로 모든 버킷은 Amazon S3 관리형 키를 통한 서버 측 암호화(SSE-S3)를 사용하는 기본 암호화 구성을 가지고 있습니다.

대상 버킷에 AWS Key Management Service(AWS KMS) 키를 통한 서버 측 암호화(SSE-KMS), AWS KMS 키를 통한 이중 계층 서버 측 암호화(DSSE-KMS) 또는 고객 제공 암호화 키(SSE-C)를 사용하는 암호화 구성이 있는 경우, Amazon S3는 해당 KMS 키 또는 고객이 제공한 키를 사용하여 업로드된 객체를 암호화합니다. 객체를 업로드할 때 업로드된 객체의 암호화 설정을 변경하려면 서버 측 암호화 유형을 지정할 수 있습니다. SSE-S3, SSE-KMS, DSSE-KMS 또는 SSE-C를 구성할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [서버 측 암호화를 사용하여 데이터 보호](#)를 참조하세요.

⚠ Important

요청을 구성할 때 file 필드가 양식의 마지막 필드여야 합니다.

버전 관리

버킷에 대한 버전 관리를 활성화하면 POST는 추가되는 객체에 대해 고유한 버전 ID를 자동으로 생성합니다. Amazon S3는 응답에서 x-amz-version-id 응답 헤더를 사용하여 이 ID를 반환합니다.

버킷에 대한 버전 관리를 일시 중지하면 Amazon S3는 버킷에 저장된 객체의 버전 ID로 항상 null을 사용합니다.

버킷의 버전 관리 상태 반환에 대한 자세한 내용은 [GetBucketVersioning](#)을 참조하세요.

Amazon S3는 분산 시스템입니다. 버킷에 대해 버전 관리를 활성화하면 Amazon S3가 동일한 객체에 대해 여러 쓰기 요청을 동시에 수신하는 경우 해당 객체의 모든 버전이 저장됩니다.

버전 관리를 사용하는 샘플 요청을 보려면 [샘플 요청](#)을 참조하세요.

요청

구문

```
POST / HTTP/1.1
Host: destinationBucket.s3.amazonaws.com
User-Agent: browser_data
Accept: file_types
Accept-Language: Regions
Accept-Encoding: encoding
Accept-Charset: character_set
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=9431149156168
Content-Length: length

--9431149156168
Content-Disposition: form-data; name="key"

acl
--9431149156168
Content-Disposition: form-data; name="tagging"
```

```
<Tagging><TagSet><Tag><Key>Tag Name</Key><Value>Tag Value</Value></Tag></TagSet></Tagging>
--9431149156168
Content-Disposition: form-data; name="success_action_redirect"

success_redirect
--9431149156168
Content-Disposition: form-data; name="Content-Type"

content_type
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-uuid"

uuid
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-tag"

metadata
--9431149156168
Content-Disposition: form-data; name="AWSAccessKeyId"

access-key-id
--9431149156168
Content-Disposition: form-data; name="Policy"

encoded_policy
--9431149156168
Content-Disposition: form-data; name="Signature"

signature=
--9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename.jpg"
Content-Type: image/jpeg

file_content
--9431149156168
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
--9431149156168--
```

요청 파라미터

이 작업 구현에서는 요청 파라미터를 사용하지 않습니다.

양식 필드

이 작업은 다음 양식 필드를 사용할 수 있습니다.

이름	설명	필수
AWSAccessKeyId	정책에 속하는 일련의 제약 조건을 충족하는 요청에 대해 익명 사용자 액세스를 허용하는 버킷 소유자의 AWS 액세스 키 ID입니다. 유형: 문자열 기본값: None 제약 조건: 요청과 함께 정책 문서가 포함되는 경우 필수입니다.	조건부
acl	지정된 Amazon S3 액세스 제어 목록(ACL)입니다. 지정된 ACL이 유효하지 않으면 오류가 생성됩니다. ACL에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 액세스 제어 목록(ACL) 개요를 참조하세요. 유형: 문자열 기본: 프라이빗 유효한 값: private public-read public-read-write aws-exec-read authenticated-read bucket-owner-read bucket-owner-full-control	아니요
Cache-Control , Content-Type , Content-Disposition , Content-Encoding , Expires	REST 지정 헤더입니다. 자세한 내용은 PutObject를 참조하세요. 유형: 문자열	아니요

이름	설명	필수
	기본값: None	
file	파일 또는 텍스트 콘텐츠입니다. 파일 또는 텍스트 콘텐츠는 양식의 마지막 필드이어야 합니다. 한 번에 파일 하나만 업로드할 수 있습니다. 유형: 파일 또는 텍스트 콘텐츠 기본값: None	예
key	업로드된 키의 이름입니다. 사용자가 제공한 파일 이름을 사용하려면 <code>\${filename}</code> 변수를 사용합니다. 예를 들어, Mary라는 사용자가 example.jpg 파일을 업로드할 때 <code>/user/mary/\${filename}</code> 를 지정하면 키 이름은 <code>/user/mary/example.jpg</code> 가 됩니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 객체 키 및 메타데이터를 참조하십시오. 유형: 문자열 기본값: None	예

이름	설명	필수
policy	요청에서 허용되는 항목을 설명하는 보안 정책입니다. 보안 정책이 없는 요청은 익명으로 간주되며 공개적으로 쓰기 가능한 버킷에서만 작동합니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 HTML 양식 및 업로드 예제를 참조하세요. 유형: 문자열 기본값: None 제약 조건: 버킷을 공개적으로 쓸 수 없는 경우 보안 정책이 필요합니다.	조건부
success_action_redirect , redirect	업로드 성공 시 클라이언트가 리디렉션되는 URL입니다. <code>success_action_redirect</code> 를 지정하지 않을 경우 Amazon S3은 <code>success_action_status</code> 필드에 지정된 빈 문서 유형을 반환합니다. Amazon S3가 URL을 해석할 수 없는 경우 필드가 없는 것처럼 작동합니다. 업로드가 실패하는 경우 Amazon S3이 오류를 표시하고 사용자는 어떤 URL로도 리디렉션되지 않습니다. 유형: 문자열 기본값: None  Note <code>redirect</code> 필드 이름은 더 이상 사용되지 않으며, <code>redirect</code> 필드 이름에 대한 지원이 조만간 중단됩니다.	아니요

이름	설명	필수
success_action_status	success_action_redirect 를 지정하지 않으면 업로드 성공 시 상태 코드가 클라이언트에 반환됩니다. 이 필드는 값 200, 201 또는 204(기본값)를 허용합니다. 200 또는 204로 값을 설정하는 경우 Amazon S3는 상태 코드가 200 또는 204인 빈 문서를 반환합니다. 201로 값을 설정하는 경우 Amazon S3는 상태 코드가 201인 XML 문서를 반환합니다. 값이 설정되지 않거나 유효하지 않은 값으로 설정된 경우 Amazon S3는 상태 코드가 204인 빈 문서를 반환합니다. 유형: 문자열 기본값: None	아니요

이름	설명	필수
tagging	객체에 추가할 지정된 태그 세트입니다. 태그를 추가하려면 다음 인코딩 체계를 사용합니다. <pre><Tagging> <TagSet> <Tag> <Key>TagName</Key> <Value>TagValue</Value> </Tag> ... </TagSet> </Tagging></pre> 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 객체 태그 지정를 참조하세요. 유형: 문자열 기본값: None	아니요
x-amz-storage-class	객체를 저장하는 데 사용할 스토리지 클래스입니다. 클래스를 지정하지 않으면 Amazon S3는 기본 스토리지 클래스 STANDARD를 사용합니다. Amazon S3는 다른 스토리지 클래스를 지원합니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 스토리지 클래스를 참조하세요. 유형: 문자열 기본값: STANDARD 유효한 값: REDUCED_REDUNDANCY EXPRESS_ONEZONE DEEP_ARCHIVE GLACIER GLACIER_IR INTELLIGENT_TIERING ONEZONE_IA STANDARD STANDARD_IA	아니요

이름	설명	필수
x-amz-meta-*	이 접두사로 시작하는 헤더는 사용자 정의 메타데이터입니다. 각 항목은 키-값 페어 세트로 저장되고 반환됩니다. Amazon S3는 사용자 정의 메타데이터를 검증하거나 해석하지 않습니다. 자세한 내용은 PutObject를 참조하세요. 유형: 문자열 기본값: None	아니요
x-amz-security-token	Amazon DevPay 보안 토큰입니다. Amazon DevPay를 사용하는 각 요청에는 제품 토큰과 사용자 토큰에 하나씩 2개의 x-amz-security-token 양식 필드가 필요합니다. 유형: 문자열 기본값: None	아니요
x-amz-signature	(AWS 서명 버전 4) 보안 정책의 HMAC-SHA256 해시입니다. 유형: 문자열 기본값: None	조건부

이름	설명	필수
x-amz-website-redirect-location	버킷이 웹 사이트로 구성된 경우 이 필드는 이 객체에 대한 요청을 동일 버킷의 다른 객체로 혹은 외부 URL로 리디렉션합니다. Amazon S3는 객체 메타데이터에 이 헤더의 값을 저장합니다. 객체 메타데이터에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 객체 키 및 메타데이터를 참조하세요. 다음 예제에서 요청 헤더는 리디렉션을 동일 버킷의 객체(anotherPage.html)로 설정합니다. <pre>x-amz-website-redirect-location: /anotherPage.html</pre> 다음 예제에서 요청 헤더는 객체 리디렉션을 다른 웹 사이트로 설정합니다. <pre>x-amz-website-redirect-location: http://www.example.com/</pre> Amazon S3에서 웹 사이트 호스팅에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 Amazon S3에서 웹 사이트 호스팅 및 웹 사이트 페이지 리디렉션을 구성하는 방법을 참조하세요. 유형: 문자열 기본값: None 제약 조건: 값에는 /, http:// 또는 https:// 접두사가 붙어야 합니다. 값의 길이는 2KB로 제한됩니다.	아니요

추가 체크섬 요청 양식 필드

객체를 업로드할 때 데이터 무결성을 확인하는 데 사용할 다양한 체크섬을 지정할 수 있습니다. Amazon S3가 사용할 체크섬 알고리즘을 하나 더 지정할 수 있습니다. 추가 체크섬 값에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [객체 무결성 확인](#)을 참조하세요.

이름	설명	필수
x-amz-checksum-algorithm	객체의 체크섬을 생성하는 데 사용된 알고리즘을 나타냅니다. 값을 지정한 경우 일치하는 체크섬 헤더를 포함해야 합니다. 그렇지 않으면 요청에 400 오류가 발생합니다. 가능한 값은 CRC32, CRC32C, SHA1, SHA256입니다.	아니요
x-amz-checksum-crc32	객체의 Base64로 인코딩된 32비트 CRC32 체크섬을 지정합니다. 이 파라미터는 x-amz-checksum-algorithm 의 값이 CRC32인 경우 필수입니다.	조건부
x-amz-checksum-crc32c	객체의 Base64로 인코딩된 32비트 CRC32C 체크섬을 지정합니다. 이 파라미터는 x-amz-checksum-algorithm 의 값이 CRC32C인 경우 필수입니다.	조건부
x-amz-checksum-sha1	객체의 Base64로 인코딩된 160비트 SHA-1 다이제스트를 지정합니다. 이 파라미터는 x-amz-checksum-algorithm 의 값이 SHA1인 경우 필수입니다.	조건부
x-amz-checksum-sha256	객체의 Base64로 인코딩된 256비트 SHA-256 다이제스트를 지정합니다. 이 파라미터는 x-amz-checksum-algorithm 의 값이 SHA256인 경우 필수입니다.	조건부

서버 측 암호화 관련 요청 양식 필드

서버 측 암호화는 저장 데이터 암호화입니다. Amazon S3가 AWS 데이터 센터의 디스크에 데이터를 쓸 때 암호화하고 사용자가 해당 데이터에 액세스할 때 복호화합니다. 객체를 업로드할 때 해당 객체를 암호화하는 데 Amazon S3에서 사용할 서버 측 암호화 유형을 지정할 수 있습니다.

서버 측 암호화에는 네 가지 유형이 있습니다.

- Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) - 2022년 5월부터 모든 Amazon S3 버킷에는 기본적으로 암호화가 구성됩니다. 서버 측 암호화의 기본 옵션은 SSE-S3를 사용하는 것입니다. 각 객체가 고유한 키로 암호화됩니다. 또한 추가 보안 조치로 SSE-S3는 주기적으로 교체되는 루트 키를 사용하여 키 자체를 암호화합니다. SSE-S3는 가장 강력한 블록 암호 중 하나인 256비트 Advanced Encryption Standard(AES-256)를 사용하여 데이터를 암호화합니다.
- AWS KMS 키를 사용한 서버 측 암호화(SSE-KMS) - SSE-KMS는 Amazon S3와 AWS KMS 서비스를 통합하여 제공됩니다. AWS KMS를 사용하면 키에 대한 제어 기능이 더 많아집니다. 예를 들어, 개별 키를 보고, 제어 정책을 편집하고, AWS CloudTrail에서 키를 팔로우할 수 있습니다. 또한 고객 관리형 키를 생성하고 관리하거나 사용자, 서비스 및 리전에 고유한 AWS 관리형 키를 사용할 수 있습니다.
- AWS KMS 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS) - AWS KMS keys를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)는 SSE-KMS와 비슷하지만, 한 계층이 아니라 두 계층의 객체 수준 암호화를 적용합니다.
- 고객 제공 키를 사용한 서버 측 암호화(SSE-C) - SSE-C를 사용하면 사용자는 암호화 키를 관리하고 Amazon S3는 암호화(디스크에 쓸 때) 및 복호화(사용자가 객체에 액세스할 때)를 관리합니다.

 Note

범용 버킷에 대해 고객 제공 키를 사용한 서버 측 암호화(SSE-C)를 차단한 경우 버킷에 새 데이터를 쓰는 동안 SSE-C 요청 헤더를 지정하면 HTTP 403 액세스 거부됨 오류가 발생합니다. 자세한 내용은 [범용 버킷에 대한 SSE-C 차단 또는 차단 해제](#)를 참조하세요.

자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [서버 측 암호화를 사용하여 데이터 보호](#)를 참조하세요.

사용할 서버 측 암호화 유형에 따라 다음 양식 필드를 지정합니다.

- SSE-S3, SSE-KMS 또는 DSSE-KMS 사용 - 이러한 유형의 서버 측 암호화를 사용하려면 요청에 다음 양식 필드를 지정합니다.

이름	설명	필수
x-amz-server-side-encryption	Amazon S3가 객체를 생성할 때 사용할 서버 측 암호화 알고리즘을 지정합니다. SSE-S3를 사용하려면 AES256을 지정합니다. SSE-KMS를 사용하려면 aws:kms를 지정합니다. DSSE-KMS를 사용하려면 aws:kms:dsse 를 지정합니다.	예

이름	설명	필수
	유형: 문자열 유효한 값: <code>aws:kms</code>, <code>AES256</code>, <code>aws:kms:dsse</code>	
<code>x-amz-server-side-encryption-aws-kms-key-id</code>	<code>x-amz-server-side-encryption</code> 헤더의 값인 <code>aws:kms</code> 또는 <code>aws:kms:dsse</code> 인 경우 이 헤더는 객체를 암호화하는 데 사용된 AWS KMS key의 ID를 지정합니다. 유형: 문자열	<code>x-amz-server-side-encryption</code> 의 값이 <code>aws:kms</code> 또는 <code>aws:kms:dsse</code> 인 경우, 예
<code>x-amz-server-side-encryption-context</code>	<code>x-amz-server-side-encryption</code>의 값이 <code>aws:kms</code> 또는 <code>aws:kms:dsse</code> 인 경우 이 헤더는 객체에 대한 암호화 컨텍스트를 지정합니다. 이 헤더의 값은 암호화 컨텍스트의 JSON 형식 키-값 페어를 포함하는 Base64로 인코딩된 UTF-8 문자열입니다. 유형: 문자열	아니요
<code>x-amz-server-side-encryption-bucket-key-enabled</code>	<code>x-amz-server-side-encryption</code>의 값이 <code>aws:kms</code> 또는 <code>aws:kms:dsse</code> 인 경우 이 헤더는 Amazon S3가 SSE-KMS 또는 DSSE-KMS와 함께 S3 버킷 키를 사용해야 하는지 여부를 지정합니다. 이 헤더를 <code>true</code>로 설정하면 Amazon S3가 SSE-KMS 또는 DSSE-KMS를 사용한 객체 암호화에 S3 버킷 키를 사용합니다. 유형: Boolean	아니요

Note

`x-amz-server-side-encryption:aws:kms` 또는 `x-amz-server-side-encryption:aws:kms:dsse`를 지정하지만 `x-amz-server-side-encryption-aws-kms-key-id`를 제공하지 않는 경우 Amazon S3는 AWS 관리형 키(`aws/S3`)를 사용하여 데이터를 보호합니다.

- SSE-C 사용 – 자체 암호화 키를 관리하려는 경우 요청에 다음 양식 필드를 모두 제공해야 합니다.

Note

SSE-C를 사용하는 경우 Amazon S3가 응답에서 반환하는 ETag 값은 객체의 MD5가 아닙니다.

이름	설명	필수
<code>x-amz-server-side-encryption-customer-algorithm</code>	객체를 암호화할 때 사용할 알고리즘을 지정합니다. 유형: 문자열 기본값: None 유효한 값: AES256 제약 조건: 유효한 <code>x-amz-server-side-encryption-customer-key</code> 및 <code>x-amz-server-side-encryption-customer-key-MD5</code> 필드를 포함해야 합니다.	예
<code>x-amz-server-side-encryption-customer-key</code>	Amazon S3가 데이터 암호화에 사용할 Base64로 인코딩된 고객 제공 암호화 키를 지정합니다. 이 값은 객체를 저장하는 데 사용된 다음 폐기됩니다. Amazon은 암호화 키를 저장하지 않습니다. 키는 <code>x-amz-server-side-encryption-customer-algorithm</code> 헤더에 지정된 알고리즘과 함께 사용하기에 적합해야 합니다.	예

이름	설명	필수
	유형: 문자열 기본값: None 제약 조건: 유효한 <code>x-amz-server-side-encryption-customer-algorithm</code> 및 <code>x-amz-server-side-encryption-customer-key-MD5</code> 필드를 포함해야 합니다.	
<code>x-amz-server-side-encryption-customer-key-MD5</code>	RFC 1321에 따라 암호화 키의 Base64로 인코딩된 128비트 MD5 다이제스트를 지정합니다. Amazon S3는 메시지 무결성 검사에 이 헤더를 사용하여 암호화 키가 오류 없이 전송되었음을 확인합니다. 유형: 문자열 기본값: None 제약 조건: 유효한 <code>x-amz-server-side-encryption-customer-algorithm</code> 및 <code>x-amz-server-side-encryption-customer-key</code> 필드를 포함해야 합니다.	예

응답

응답 헤더

이 작업의 구현에는 모든 응답에 공통되는 응답 헤더 외에 다음 응답 헤더가 포함될 수 있습니다. 자세한 내용은 [공통 응답 헤더](#)를 참조하세요.

이름	설명
<code>x-amz-checksum-crc32</code>	객체의 Base64로 인코딩된 32비트 CRC32 체크섬입니다. 유형: 문자열

이름	설명
x-amz-checksum-crc32c	객체의 Base64로 인코딩된 32비트 CRC32C 체크섬입니다. 유형: 문자열
x-amz-checksum-sha1	객체의 Base64로 인코딩된 160비트 SHA-1 다이제스트입니다. 유형: 문자열
x-amz-checksum-sha256	객체의 Base64로 인코딩된 256비트 SHA-256 다이제스트입니다. 유형: 문자열
x-amz-expiration	버킷의 수명 주기 구성에서 객체에 대해 Expiration 작업이 구성된 경우 Amazon S3는 이 헤더를 반환합니다. 헤더 값에는 expiry-date 구성 요소 및 URL로 인코딩된 rule-id 구성 요소가 포함됩니다. 버전 관리를 사용하는 버킷의 경우 이 헤더는 현재 버전에만 적용됩니다. Amazon S3는 최신이 아닌 버전을 영구적으로 삭제할 수 있는 시기를 나타내는 헤더를 제공하지 않습니다. 자세한 내용은 PutBucketLifecycle Configuration을 참조하세요. 유형: 문자열
success_action_redirect, redirect	업로드 성공 시 클라이언트가 리디렉션되는 URL입니다. 유형: 문자열 상위 항목: PostResponse

이름	설명
<code>x-amz-server-side-encryption</code>	Amazon S3에 이 객체를 저장할 때 사용된 서버 측 암호화 알고리즘입니다(예: AES256, <code>aws:kms</code>, <code>aws:kms:dsse</code>). 유형: 문자열
<code>x-amz-server-side-encryption-aws-kms-key-id</code>	<code>x-amz-server-side-encryption</code> 헤더의 값이 <code>aws:kms</code>인 경우 이 헤더는 객체를 암호화하는 데 사용된 KMS 키의 ID를 지정합니다. 유형: 문자열
<code>x-amz-server-side-encryption-bucket-key-enabled</code>	<code>x-amz-server-side-encryption</code> 의 값이 <code>aws:kms</code>인 경우 이 헤더는 S3 버킷 키를 사용하여 객체를 SSE-KMS로 암호화할지 여부를 나타냅니다. 이 헤더가 <code>true</code>로 설정된 경우, 객체는 SSE-KMS에 S3 버킷 키를 사용합니다. 유형: Boolean
<code>x-amz-server-side-encryption-customer-algorithm</code>	SSE-C가 요청된 경우 응답에는 사용된 암호화 알고리즘을 확인하는 이 헤더가 포함됩니다. 타입: 문자열 유효 값: AES256
<code>x-amz-server-side-encryption-customer-key-MD5</code>	SSE-C가 요청된 경우 응답에는 고객 제공 암호화 키의 왕복 메시지 무결성을 확인하기 위한 이 헤더가 포함됩니다. 유형: 문자열
<code>x-amz-version-id</code>	객체의 버전입니다. 유형: 문자열

응답 요소

이름	설명
Bucket	객체가 저장되어 있는 버킷의 이름입니다. 유형: 문자열 상위 항목: PostResponse
ETag	엔터티 태그(ETag)는 GET 요청 작업에서 If-Modified 요청 태그를 사용하여 조건부 GET 작업을 수행하는 데 사용할 수 있는, 객체의 MD5 해시입니다. ETag는 객체의 콘텐츠에 대한 변경 사항만 반영하고 메타데이터에 대한 변경을 반영하지 않습니다. 유형: 문자열 상위 항목: PostResponse
Key	객체 키 이름입니다. 유형: 문자열 상위 항목: PostResponse
Location	객체의 URI입니다. 유형: 문자열 상위 항목: PostResponse

특수 오류

이 작업의 구현은 특수 오류를 반환하지 않습니다. 일반적인 Amazon S3 오류에 대한 자세한 내용 및 오류 코드 목록은 [오류 응답](#)을 참조하세요.

예

예 요청

```
POST /Neo HTTP/1.1
Content-Length: 4
Host: quotes.s3.amazonaws.com
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: authorization string
Content-Type: text/plain
Expect: the 100-continue HTTP status code
```

ObjectContent

버전 관리가 일시 중지된 샘플 응답

다음은 버킷 버전 관리가 일시 중지된 경우의 샘플 응답입니다.

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
x-amz-id-2: LriYPLdm0dAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiIOIx1+zbwZ163pt7
x-amz-request-id: 0A49CE4060975EAC
x-amz-version-id: default
Date: Wed, 12 Oct 2009 17:50:00 GMT
ETag: "1b2cf535f27731c974343645a3985328"
Content-Length: 0
Connection: close
Server: AmazonS3
```

이 응답에서 버전 ID는 null입니다.

버전 관리가 활성화된 샘플 응답

다음은 버킷 버전 관리가 활성화된 경우의 샘플 응답입니다.

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
x-amz-id-2: LriYPLdm0dAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiIOIx1+zbwZ163pt7
x-amz-request-id: 0A49CE4060975EAC
x-amz-version-id: 43jfkodU8493jnFJD9fjj3HHNVfdsQUIFDNsidf038jfdsjGFDSIRp
Date: Wed, 01 Mar 2006 12:00:00 GMT
```

```
ETag: "828ef3fdfa96f00ad9f27c383fc9ac7f"  
Content-Length: 0  
Connection: close  
Server: AmazonS3
```

관련 리소스

- [CopyObject](#)
- [POST Object](#)
- [GetObject](#)

POST Object restore

설명

이 작업은 다음과 같은 유형의 요청을 수행합니다.

- `select` - 아카이브 객체에서 Select 쿼리를 수행합니다.
- `restore an archive` - 아카이브된 객체를 복원합니다.

이 작업을 사용하려면 `s3:RestoreObject` 및 `s3:GetObject` 작업을 수행할 수 있는 권한이 있어야 합니다. 버킷 소유자는 기본적으로 이 권한을 보유하며 다른 사용자에게 이 권한을 부여할 수 있습니다. 권한에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [버킷 하위 리소스 작업에 관련된 권한](#) 및 [Amazon S3 리소스에 대한 액세스 권한 관리](#)를 참조하세요.

Select 요청으로 아카이브 쿼리

Select 요청 유형을 사용하여 아카이브된 객체에서 SQL 쿼리를 수행할 수 있습니다. Select 요청으로 쿼리되는 아카이브된 객체는 압축되지 않은, 쉼표로 분리된 값(CSV) 파일 형식이어야 합니다. 데이터를 더 핫한 Amazon S3 계층으로 복원할 필요 없이 아카이브된 데이터에 대해 쿼리 및 사용자 지정 분석을 실행할 수 있습니다. Select 요청에 대한 개요는 Amazon Simple Storage Service 사용 설명서의 [아카이브된 객체 쿼리](#)를 참조하세요.

Select 요청을 실행할 때 다음을 수행합니다.

- Select 쿼리의 출력 위치를 정의합니다. 이 위치는 쿼리하는 아카이브된 객체가 포함된 버킷과 동일한 AWS 리전의 Amazon S3 버킷이어야 합니다. 작업을 시작하는 AWS 계정에는 S3 버킷에 대한 쓰기 권한이 있어야 합니다. 버킷에 저장되는 출력 객체의 스토리지 클래스와 암호화를 지정할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서에서 [아카이브된 객체 쿼리](#)를 참조하세요.

요청 본문의 S3 구조에 대한 자세한 내용은 다음을 참조하세요.

- [PutObject](#)
- Amazon Simple Storage Service 사용 설명서의 [ACL을 사용한 액세스 관리](#)
- Amazon Simple Storage Service 사용 설명서의 [서버 측 암호화를 사용하여 데이터 보호](#)
- 요청 본문의 `SelectParameters` 구조에서 쿼리의 복원 SELECT 유형에 대한 SQL 표현식을 정의합니다. 예를 들어 다음 예제와 같은 표현식을 사용할 수 있습니다.
- 다음 표현식은 지정한 객체에서 모든 레코드를 반환합니다.


```
SELECT * FROM Object
```

- 객체에 저장된 데이터에 헤더를 사용하지 않는다고 가정하면 위치 헤더를 사용하여 열을 지정할 수 있습니다.

```
SELECT s._1, s._2 FROM Object s WHERE s._3 > 100
```

- 헤더가 있고 요청 본문의 CSV에서 fileHeaderInfo를 USE로 설정한 경우, 쿼리에서 헤더를 지정할 수 있습니다. (fileHeaderInfo 필드를 IGNORE로 설정한 경우, 쿼리를 위해 첫 번째 행을 건너뛵니다.) 서수 위치와 헤더 열 이름은 혼용할 수 없습니다.

```
SELECT s.Id, s.FirstName, s.SSN FROM S3Object s
```

Amazon Glacier Select 복원과 함께 SQL을 사용하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [Amazon S3 Select 및 Amazon Glacier Select용 SQL 참조](#)를 참조하세요.

Select 요청을 실행할 때 다음 작업도 수행할 수 있습니다.

- 쿼리를 신속히 처리하기 위해 Expedited 계층을 지정합니다. 계층에 대한 자세한 내용은 이 주제 뒷부분의 '아카이브 복원'을 참조하세요.
- 쿼리되는 입력 객체와 CSV로 인코딩된 쿼리 결과 직렬화의 데이터 직렬화 형식에 대한 세부 정보를 지정합니다.

다음은 Select 기능에 대해 추가로 알아야 할 중요한 사항입니다.

- 출력 결과는 새 Amazon S3 객체입니다. 아카이브 검색과 달리 객체는 수동으로 또는 수명 주기 정책을 통해 명시적으로 삭제될 때까지 저장됩니다.
- 동일한 Amazon S3 객체에 둘 이상의 Select 요청을 실행할 수 있습니다. Amazon S3는 중복 요청을 제거하지 않으므로 중복 요청을 실행하지 마세요.
- Amazon S3는 객체가 이미 복원된 경우에도 Select 요청을 수락합니다. Select 요청은 오류 응답 409를 반환하지 않습니다.

아카이브 복원

GLACIER 및 DEEP_ARCHIVE 스토리지 클래스의 객체는 아카이브됩니다. 아카이브된 객체에 액세스하려면 먼저 복원 요청을 시작해야 합니다. 이 요청은 아카이브된 객체의 임시 복사본을 복원합니다. 복원 요청에서 복원하려는 객체 복사본의 복원 일수를 지정합니다. 지정된 기간이 경과하면 Amazon S3는 임시 복사본을 삭제하지만 객체는 해당 객체가 복원된 GLACIER 또는 DEEP_ARCHIVE 스토리지 클래스에 계속 아카이브됩니다.

특정 객체 버전을 복원하려면 버전 ID를 제공할 수 있습니다. 버전 ID를 제공하지 않으면 Amazon S3가 현재 버전을 복원합니다.

복원 작업을 완료하는 데 걸리는 시간은 객체가 복원되는 스토리지 클래스 및 지정된 데이터 액세스 계층에 따라 달라집니다.

아카이브된 객체를 복원할 때(또는 Select 요청을 사용할 때) 요청 본문의 Tier 요소에 다음 데이터 액세스 계층 옵션 중 하나를 지정할 수 있습니다.

- **Expedited** - 신속 검색을 사용하면 가끔 발생하는 아카이브의 하위 집합에 대한 긴급 요청이 필요할 때 GLACIER 스토리지 클래스에 저장된 데이터에 신속하게 액세스할 수 있습니다. 매우 큰 아카이브된 객체(250MB+)를 제외한 모든 경우, 신속 검색을 사용하여 액세스된 데이터는 일반적으로 1~5분 안에 사용할 수 있게 됩니다. 프로비저닝된 용량을 통해 필요할 때 신속 검색에 대한 검색 용량이 보장됩니다. 신속 검색 및 프로비저닝된 용량은 DEEP_ARCHIVE 스토리지 클래스에 사용할 수 없습니다.
- **Standard** - 표준 검색을 사용하면 몇 시간 내에 아카이브된 모든 객체에 액세스할 수 있습니다. 이는 검색 옵션을 지정하지 않은 GLACIER 및 DEEP_ARCHIVE 검색 요청에 대한 기본 옵션입니다. 표준 검색은 일반적으로 GLACIER 스토리지 클래스에서 3~5시간 이내에 완료되고 DEEP_ARCHIVE 스토리지 클래스에서 12시간 이내에 완료됩니다.
- **Bulk** - 대량 검색은 Amazon Glacier의 가장 저렴한 검색 옵션으로, 페타바이트 단위의 대용량 데이터도 하루 만에 저렴하게 검색할 수 있습니다. 대량 검색은 일반적으로 GLACIER 스토리지 클래스에서 5~12시간 이내에 완료되고 DEEP_ARCHIVE 스토리지 클래스에서 48시간 이내에 완료됩니다.

Expedited 데이터 액세스를 위한 아카이브 검색 옵션 및 프로비저닝된 용량에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [아카이브된 객체 복원](#)을 참조하세요.

Amazon S3 복원 속도 업그레이드를 사용하여 복원이 진행되는 동안 복원 속도를 더 빠른 속도로 변경할 수 있습니다. 동일한 객체에 다른 복원 요청을 실행하면서 새 Tier 요청 요소를 설정하여 진행 중인 복원의 속도를 업그레이드합니다. 복원 티어를 업그레이드하라는 요청을 발행할 때는 진행 중인 복원

이 사용하는 티어보다 빠른 티어를 선택해야 합니다. Days 요청 요소 등 다른 매개 변수는 변경하면 안 됩니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [진행 중인 복원 속도 업그레이드](#)를 참조하세요.

객체 복원의 상태를 가져오려면 HEAD 요청을 전송할 수 있습니다. 작업은 응답에서 복원 상태에 대한 정보를 제공하는 x-amz-restore 헤더를 반환합니다. Amazon S3 이벤트 알림을 사용하여 복원이 시작되거나 완료되면 알림을 받을 수 있습니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [Amazon S3 이벤트 알림 구성](#)을 참조하세요.

아카이브된 객체를 복원한 후 새 기간으로 요청을 다시 실행하여 복원 기간을 업데이트할 수 있습니다. Amazon S3는 현재 시점을 기준으로 복원 기간을 업데이트하고 요청에 대해서만 요금을 부과합니다. 데이터 전송 요금은 없습니다. Amazon S3가 객체에 대한 현재 복원 요청을 적극적으로 처리하는 경우에는 복원 기간을 업데이트할 수 없습니다.

버킷의 수명 주기 구성에 만료 작업이 포함된 규칙이 있는 경우 객체 만료는 복원 요청에서 지정한 수명을 재정의합니다. 예를 들어 10일간 객체 복사본을 복원했지만 객체가 3일 후에 만료되도록 예약된 경우 Amazon S3는 3일 후에 객체를 삭제합니다. 수명 주기 구성에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [PutBucketLifecycleConfiguration](#) 및 [객체 수명 주기 관리](#)를 참조하세요.

요청

구문

```
POST /ObjectName?restore&versionId=VersionID HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see ## ##\(AWS ## ## 4\))
Content-MD5: MD5
```

request body

Note

구문에는 요청 헤더가 일부만 나와 있습니다. 전체 목록은 이 주제 뒷부분의 '요청 헤더'를 참조하세요.

요청 파라미터

이 작업 구현에서는 요청 파라미터를 사용하지 않습니다.

요청 헤더

이름	설명	필수
Content-MD5	데이터의 Base64로 인코딩된 128비트 MD5 다이제스트입니다. 이 헤더를 메시지 무결성 검사로 사용하여 요청 본문이 전송 중에 손상되지 않았는지 확인해야 합니다. 자세한 내용은 RFC 1864를 참조하세요. 유형: 문자열 기본값: None	예

요청 요소

다음은 아카이브 복원을 위한 요청 본문의 XML 예제입니다.

```
<RestoreRequest>
  <Days>2</Days>
  <GlacierJobParameters>
    <Tier>Bulk</Tier>
  </GlacierJobParameters>
</RestoreRequest>
```

다음 표에서는 요청 본문의 아카이브 복원을 위한 XML을 설명합니다.

이름	설명	필수
RestoreRequest	복원 정보를 위한 컨테이너입니다. 유형: 컨테이너	예
Days		아카이브를 복원하는 경우, 예

이름	설명	필수
	복원된(활성) 복사본의 수명입니다. Amazon Glacier에서 객체를 복원할 수 있는 최소 일수는 1입니다. 객체 복사본이 지정된 수명에 도달하면 Amazon S3는 버킷에서 해당 복사본을 제거합니다. 아카이브를 복원하는 경우 이 요소는 필수입니다. 이 요소를 SELECT 요청 유형과 함께 사용하지 마세요. 유형: 양의 정수 상위 항목: RestoreRequest	
GlacierJobParameters	S3 Glacier 스토리지 클래스 작업 파라미터를 위한 컨테이너입니다. 이 요소를 SELECT 요청 유형과 함께 사용하지 마세요. 유형: 컨테이너 상위 항목: RestoreRequest	아니요
Tier	아카이브를 복원할 때 사용할 데이터 액세스 계층입니다. Standard가 기본값입니다. 형식: 열거형 유효한 값: Expedited Standard Bulk 상위 항목: GlacierJobParameters	아니요

다음 XML은 아카이브된 객체에 대한 Select 쿼리의 요청 본문입니다.

```
<RestoreRequest>
  <Type>SELECT</Type>
  <Tier>Expedited</Tier>
  <Description>Job description</Description>
  <SelectParameters>
    <Expression>Select * from Object</Expression>
    <ExpressionType>SQL</ExpressionType>
```

```

    <InputSerialization>
      <CSV>
        <FileHeaderInfo>IGNORE</FileHeaderInfo>
        <RecordDelimiter>\n</RecordDelimiter>
        <FieldDelimiter>,</FieldDelimiter>
        <QuoteCharacter>"</QuoteCharacter>
        <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
        <Comments>#</Comments>
      </CSV>
    </InputSerialization>
  <OutputSerialization>
    <CSV>
      <QuoteFields>ASNEEDED</QuoteFields>
      <RecordDelimiter>\n</RecordDelimiter>
      <FieldDelimiter>,</FieldDelimiter>
      <QuoteCharacter>"</QuoteCharacter>
      <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
    </CSV>
  </OutputSerialization>
</SelectParameters>
<OutputLocation>
  <S3>
    <BucketName>Name of bucket</BucketName>
    <Prefix>Key prefix</Prefix>
    <CannedACL>Canned ACL string</CannedACL>
    <AccessControlList>
      <Grantee>
        <Type>Grantee Type</Type>
      </Grantee>
    </AccessControlList>
    <ID>Grantee identifier</ID>
    <URI>Grantee URI</URI>
    <Permission>Granted permission</Permission>
    <DisplayName>Display Name</DisplayName>
    <EmailAddress>email</EmailAddress>
  </S3>
</OutputLocation>
<Encryption>
  <EncryptionType>Encryption type</EncryptionType>
  <KMSKeyId>KMS Key ID</KMSKeyId>
  <KMSContext>Base64-encoded JSON<KMSContext>
</Encryption>
<UserMetadata>
  <MetadataEntry>
    <Name>Key</Name>
    <Value>Value</Value>
  </MetadataEntry>
</UserMetadata>

```

```
        </MetadataEntry>
      </UserMetadata>
      <Tagging>
        <TagSet>
          <Tag>
            <Key>Tag name</Key>
            <Value>Tag value</Value>
          </Tag>
        </TagSet>
      </Tagging>
      <StorageClass>Storage class</StorageClass>
    </S3>
  </OutputLocation>
</RestoreRequest>
```

다음 표에서는 요청 본문의 SELECT 복원 유형을 위한 XML을 설명합니다.

이름	설명	필수
RestoreRequest	복원 정보를 위한 컨테이너입니다. 유형: 컨테이너	예
Tier	아카이브를 복원할 때 사용할 데이터 액세스 계층입니다. Standard가 기본값입니다. 형식: 열거형 유효한 값: Expedited Standard Bulk 상위 항목: RestoreRequest	아니요
Description	요청에 대한 설명입니다(선택 사항). 유형: 문자열 상위 항목: RestoreRequest	아니요

이름	설명	필수
SelectParameters	Select 작업 요청에 대한 파라미터를 설명합니다. 유형: 컨테이너 상위 항목: RestoreRequest	요청 유형이 SELECT인 경우, 예
OutputLocation	Select 복원 요청의 결과를 수신할 위치를 설명합니다. 유형: Amazon S3용 컨테이너 상위 항목: RestoreRequest	요청 유형이 SELECT인 경우, 예

SelectParameters 컨테이너 요소에는 다음 요소가 포함되어 있습니다.

이름	설명	필수
Expression	SQL 표현식입니다. 예제: - 다음 SQL 표현식은 CSV 형식으로 저장된 객체에서 데이터의 첫 번째 열을 검색합니다. SELECT s._1 FROM Object s - 다음 SQL 표현식은 객체의 모든 것을 반환합니다. SELECT * FROM Object 유형: 문자열 상위 항목: SelectParameters	예
ExpressionType	표현식 유형을 식별합니다. 타입: 문자열	예

이름	설명	필수
InputSerialization	유효한 값: SQL	예
	상위 항목: SelectParameters	
	객체의 직렬화 형식을 설명합니다.	
	유형: CSV용 컨테이너	
OutputSerialization	상위 항목: SelectParameters	예
	선택 작업의 결과가 직렬화되는 방법을 설명합니다.	
	유형: CSV용 컨테이너	
	상위 항목: SelectParameters	

InputSerialization 요소의 CSV 컨테이너 요소에는 다음 요소가 포함되어 있습니다.

이름	설명	필수
RecordDelimiter	입력에서 개별 레코드를 분리하는 데 사용되는 단일 문자입니다. 기본값 대신 임의의 구분 기호를 지정할 수 있습니다. 유형: 문자열 기본값: \n 상위 항목: CSV	아니요
FieldDelimiter	한 레코드 내에서 개별 필드를 분리하는 데 사용되는 단일 문자입니다. 임의의 구분 기호를 지정할 수 있습니다. 유형: 문자열 기본값: ,	아니요

이름	설명	필수
	상위 항목: CSV	
QuoteCharacter	필드 구분 기호가 값의 일부인 경우 이스케이프에 사용되는 단일 문자입니다. CSV 파일에서 이 예제를 생각해 보세요. "a, b" 값을 따옴표로 묶으면 이 값은 단일 필드가 됩니다. 따옴표를 사용하지 않는 경우 쉼표는 필드 구분 기호입니다(이 경우 a 및 b라는 두 개의 개별 필드 값이 됨). 유형: 문자열 기본값: " 상위 항목: CSV	아니요
QuoteEscapeCharacter	이미 이스케이프된 값 내의 따옴표 문자를 이스케이프하는 데 사용되는 단일 문자입니다. 예를 들어 값 "" " a , b "" "는 " a , b "로 구문 분석됩니다. 유형: 문자열 기본값: " 상위 항목: CSV	아니요

이름	설명	필수
FileHeaderInfo	입력 데이터의 첫 번째 줄을 설명합니다. 이는 ENUM 값 중 하나입니다. - NONE: 첫 번째 줄이 헤더가 아닙니다. - IGNORE: 첫 번째 줄이 헤더이지만 헤더 값을 사용하여 표현식의 열을 나타낼 수 없습니다. 열 위치(예: <code>_1</code>, <code>_2</code>, ...)를 사용하여 열(<code>SELECT s._1 FROM OBJECT s</code>)을 나타낼 수 있습니다. - Use: 첫 번째 줄이 헤더이며 헤더 값을 사용하여 표현식(<code>SELECT "name" FROM OBJECT</code>)의 열을 식별할 수 있습니다. 형식: 열거형 유효한 값: NONE USE IGNORE 상위 항목: CSV	아니요
Comments	행의 시작에 존재할 경우, 해당 행을 무시해야 함을 나타내는데 사용되는 단일 문자입니다. 주석 줄을 나타내는 문자를 지정할 수 있습니다. 유형: 문자열 상위 항목: CSV	아니요

CSV 컨테이너 요소(**OutputSerialization** 요소 내)에는 다음 요소가 포함되어 있습니다.

이름	설명	필수
QuoteFields	출력 필드에 따옴표를 사용할지 여부를 나타냅니다.	아니요

이름	설명	필수
	• ALWAYS: 출력 필드에 항상 따옴표를 사용합니다. • ASNEEDED: 필요한 경우 출력 필드에 따옴표를 사용합니다. 형식: 열거형 유효한 값: ALWAYS ASNEEDED 기본값: AsNeeded 상위 항목: CSV	
RecordDelimiter	출력에서 개별 레코드를 분리하는 데 사용되는 단일 문자입니다. 기본값 대신 임의의 구분 기호를 지정할 수 있습니다. 유형: 문자열 기본값: \n 상위 항목: CSV	아니요
FieldDelimiter	한 레코드 내에서 개별 필드를 분리하는 데 사용되는 단일 문자입니다. 임의의 구분 기호를 지정할 수 있습니다. 유형: 문자열 기본값: , 상위 항목: CSV	아니요

이름	설명	필수
QuoteCharacter	필드 구분 기호가 값의 일부인 경우 이스케이프에 사용되는 단일 문자입니다. 예를 들어 값이 a, b인 경우 Amazon S3는 다음과 같이 이 필드 값을 따옴표로 묶습니다. " a , b " . 유형: 문자열 기본값: " 상위 항목: CSV	아니요
QuoteEscapeCharacter	이미 이스케이프된 값 내의 따옴표 문자를 이스케이프하는 데 사용되는 단일 문자입니다. 예를 들어 값이 " a , b " 인 경우 Amazon S3는 다음과 같이 이 값을 따옴표로 묶습니다: "" " a , b "" " . 유형: 문자열 상위 항목: CSV	아니요

S3 컨테이너 요소(**OutputLocation** 요소 내)에는 다음 요소가 포함되어 있습니다.

이름	설명	필수
AccessControlList	스테이징된 결과에 대한 액세스를 제어하는 권한의 목록입니다. 유형: 부여용 컨테이너 상위 항목: S3	아니요
BucketName	Select 복원 결과가 저장되는 S3 버킷의 이름입니다. 버킷은 입력 아카이브 객체를 포함하는 버킷과 동일한 AWS 리전에 있어야 합니다. 유형: 문자열	예

이름	설명	필수
	상위 항목: S3	
CannedACL	Select 복원 결과에 적용되는 미리 준비된 액세스 제어 목록 (ACL)입니다. 타입: 문자열 유효한 값: private public-read public-read-write aws-exec-read authenticated-read bucket-owner-read bucket-owner-full-control 상위 항목: S3	아니요
Encryption	저장된 결과에 대한 암호화 정보를 포함합니다. 유형: 암호화용 컨테이너 상위 항목: S3	아니요
Prefix	이 select 복원 결과 앞에 추가되는 접두사입니다. 접두사의 최대 길이는 512바이트입니다. 유형: 문자열 상위 항목: S3	예

이름	설명	필수
StorageClass	Select 요청 결과를 저장하는 데 사용되는 스토리지 클래스입니다. 유형: 문자열 유효한 값: STANDARD REDUCED_REDUNDANCY STANDARD_IA ONEZONE_IA 상위 항목: S3	아니요
Tagging	태그 정보를 위한 컨테이너입니다. 유형: 태그 구조 상위 항목: S3	아니요
UserMetadata	Select 복원 결과와 함께 저장할 메타데이터 목록을 포함합니다. 유형: MetadataEntry 구조 상위 항목: S3	아니요

피부여자 컨테이너 요소(**AccessControlList** 요소 내)에는 다음 요소가 포함되어 있습니다.

이름	설명	필수
DisplayName	피부여자의 표시 이름입니다. 유형: 문자열 상위 항목: Grantee	아니요
EmailAddress	피부여자의 이메일 주소입니다.	아니요

이름	설명	필수
ID	유형: 문자열	아니요
	상위 항목: Grantee	
	피부여자의 정식 사용자 ID입니다.	
Type	유형: 문자열	아니요
	상위 항목: Grantee	
	피부여자의 유형입니다.	
URI	유형: 문자열	아니요
	상위 항목: Grantee	
	피부여자 그룹의 URI입니다.	
Permission	유형: 문자열	아니요
	상위 항목: Grantee	
	부여된 권한입니다.	

Encryption 컨테이너 요소(S3 내)에는 다음 요소가 포함되어 있습니다.

이름	설명	필수
EncryptionType	작업 결과를 저장할 때 사용되는 서버 측 암호화 알고리즘입니다. 기본값은 암호화 없음입니다. 유형: 문자열	아니요

이름	설명	필수
	유효한 값 <code>aws:kms</code> <code>AES256</code> 상위 항목: Encryption	
KMSContext	선택 사항입니다. 암호화 유형이 <code>aws:kms</code>인 경우 이 값을 사용하여 select 복원 결과의 암호화 컨텍스트를 지정할 수 있습니다. 유형: 문자열 상위 항목: Encryption	아니요
KMSKeyId	개체 암호화에 사용할 AWS Key Management Service(AWS KMS) 키 ID입니다. 유형: 문자열 상위 항목: Encryption	아니요

TagSet 컨테이너 요소(**Tagging** 요소 내)에는 다음 요소가 포함되어 있습니다.

이름	설명	필수
Tag	태그를 포함합니다. 유형: 컨테이너 상위 항목: TagSet	아니요

태그 컨테이너 요소(**TagSet** 요소 내)에는 다음 요소가 포함되어 있습니다.

이름	설명	필수
Key	태그의 이름입니다.	아니요

이름	설명	필수
	유형: 문자열 상위 항목: Tag	
Value	태그의 값입니다. 유형: 문자열 상위 항목: Tag	아니요

MetadataEntry 컨테이너 요소(**UserMetadata** 요소 내)에는 객체와 함께 저장할 다음과 같은 키-값 페어 요소가 포함되어 있습니다.

이름	설명	필수
MetadataKey	메타데이터 키입니다. 유형: 문자열 상위 항목:	아니요
MetadataEntry	메타데이터 값입니다. 유형: 문자열 상위 항목:	아니요

응답

작업이 성공하면 200 OK 또는 202 Accepted 상태 코드가 반환됩니다.

- 객체 복사본이 이전에 복원되지 않은 경우 Amazon S3는 응답으로 202 Accepted를 반환합니다.
- 객체 복사본이 이전에 복원된 경우 Amazon S3는 응답으로 200 OK를 반환합니다.

응답 헤더

이 작업의 구현에서는 대부분의 응답에 공통되는 응답 헤더만 사용합니다. 자세한 내용은 [공통 응답 헤더](#)를 참조하세요.

응답 요소

이 작업은 응답 요소를 반환하지 않습니다.

특수 오류

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
RestoreAlreadyInProgress	객체 복원이 이미 진행 중입니다. (이 오류는 SELECT 유형 요청에는 적용되지 않습니다.)	409 충돌	클라이언트
GlacierExpeditedRetrievalNotAvailable	S3 Glacier 스토리지 클래스 신속 검색을 현재 사용할 수 없습니다. 나중에 다시 시도해 주세요. (Expedited 요청을 처리할 용량이 충분하지 않은 경우 반환됩니다. 이 오류는 Expedited 검색에만 적용되며 Standard 또는 Bulk 검색에는 적용되지 않습니다.)	503	해당 사항 없음

예제

신속 검색 옵션을 사용하여 2일간 객체 복원

다음 복원 요청은 신속 검색 옵션을 사용하여 Amazon Glacier에서 photo1.jpg 객체의 복사본을 2일간 복원합니다.

```
POST /photo1.jpg?restore HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Date: Mon, 22 Oct 2012 01:49:52 GMT
Authorization: authorization string
Content-Length: content length
```

```
<RestoreRequest>
  <Days>2</Days>
  <GlacierJobParameters>
    <Tier>Expedited</Tier>
  </GlacierJobParameters>
</RestoreRequest>
```

examplebucket에 해당 객체의 복원된 복사본이 없는 경우 Amazon S3는 다음 202 Accepted 응답을 반환합니다.

```
HTTP/1.1 202 Accepted
x-amz-id-2: GFihv3y6+kE7KG11GEkQhU7/2/CHR3Yb2fCb2S04nxI423Dqwg2XiQ0B/
UZ1zYQvPiBlZNRcovw=
x-amz-request-id: 9F341CD3C4BA79E0
Date: Sat, 20 Oct 2012 23:54:05 GMT
Content-Length: 0
Server: AmazonS3
```

해당 객체의 복사본이 이미 복원된 경우 Amazon S3는 200 OK 응답을 반환하고 복원된 복사본의 만료 시간만 업데이트합니다.

SELECT 요청을 사용하여 아카이브 쿼리

다음은 예제 Select 복원 요청입니다.

```
POST /object-one.csv?restore HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Date: Date: Sat, 20 Oct 2012 23:54:05 GMT
Authorization: authorization string
Content-Length: content length

<RestoreRequest xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Type>SELECT</Type>
  <Tier>Expedited</Tier>
  <Description>this is a description</Description>
  <SelectParameters>
    <InputSerialization>
      <CSV>
        <FileHeaderInfo>IGNORE</FileHeaderInfo>
        <Comments>#</Comments>
        <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
        <RecordDelimiter>\n</RecordDelimiter>
```

```

        <FieldDelimiter>,</FieldDelimiter>
        <QuoteCharacter>"</QuoteCharacter>
    </CSV>
</InputSerialization>
<ExpressionType>SQL</ExpressionType>
<Expression>select * from object</Expression>
<OutputSerialization>
    <CSV>
        <QuoteFields>ALWAYS</QuoteFields>
        <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
        <RecordDelimiter>\n</RecordDelimiter>
        <FieldDelimiter>\t</FieldDelimiter>
        <QuoteCharacter>\'</QuoteCharacter>
    </CSV>
</OutputSerialization>
</SelectParameters>
<OutputLocation>
    <S3>
        <BucketName>example-output-bucket</BucketName>
        <Prefix>test-s3</Prefix>
        <AccessControlList>
            <Grant>
                <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="AmazonCustomerByEmail">
                    <EmailAddress>jane-doe@example.com</EmailAddress>
                </Grantee>
                <Permission>FULL_CONTROL</Permission>
            </Grant>
        </AccessControlList>
        <UserMetadata>
            <MetadataEntry>
                <Name>test</Name>
                <Value>test-value</Value>
            </MetadataEntry>
            <MetadataEntry>
                <Name>other</Name>
                <Value>something else</Value>
            </MetadataEntry>
        </UserMetadata>
        <StorageClass>STANDARD</StorageClass>
    </S3>
</OutputLocation>
</RestoreRequest>

```

Amazon S3는 다음 202 Accepted 응답을 반환합니다.

```
HTTP/1.1 202 Accepted
x-amz-id-2: GFihv3y6+kE7KG11GEkQhU7/2/CHR3Yb2fCb2S04nxI423Dqwg2XiQ0B/
UZ1zYQvPiB1ZNRcovw=
x-amz-request-id: 9F341CD3C4BA79E0
x-amz-restore-output-path: js-test-s3/qE8nk5M0XIj-LuZE2HXNw6empQm3znLkH1MWInRYPs-
Orl2W0uj6LyYm-neTvm1-btz3wbBxfMhPykd3jk1-lvZE7w42/
Date: Sat, 20 Oct 2012 23:54:05 GMT
Content-Length: 0
Server: AmazonS3
```

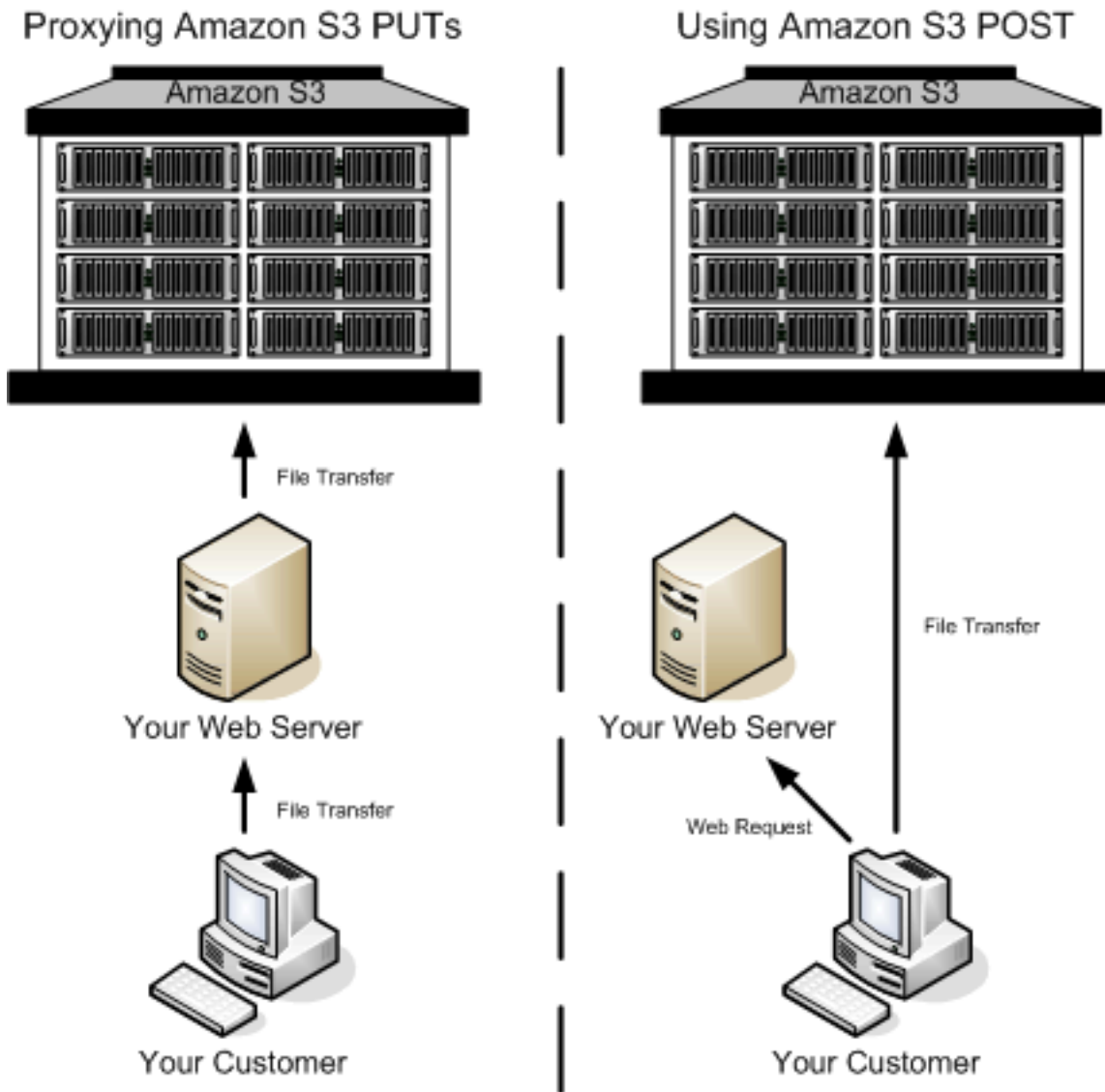
추가 정보

- [GetBucketLifecycleConfiguration](#)
- [PutBucketLifecycleConfiguration](#)
- Amazon Simple Storage Service 사용 설명서의 [Amazon S3 Select](#) 및 [Amazon Glacier Select](#)용 [SQL 참조](#)

HTTP POST를 사용한 브라우저 기반 업로드

Amazon S3는 사용자가 Amazon S3에 직접 콘텐츠를 업로드할 수 있도록 HTTP POST 요청을 지원합니다. POST를 사용하면 최종 사용자가 자격 증명을 보호하는 보안 중개 노드를 통해 데이터를 전달하지 않고도 요청을 인증할 수 있습니다. 따라서 HTTP POST는 지연 시간을 줄일 가능성이 있습니다.

다음 그림은 POST 요청을 사용한 Amazon S3 업로드를 보여 줍니다.



1. 사용자가 웹 브라우저에서 페이지에 액세스합니다.
2. 웹페이지는 사용자가 Amazon S3에 콘텐츠를 업로드하는 데 필요한 모든 정보가 들어 있는 HTML 양식을 포함합니다.
3. 사용자가 웹 브라우저를 통해 Amazon S3에 콘텐츠를 업로드합니다.

브라우저 기반 POST 요청을 전송하는 프로세스는 다음과 같습니다.

1. 객체를 업로드할 수 있는 버킷 이름, 생성 중인 객체에 대해 허용할 키 이름 접두사 등 요청에서 허용할 조건을 지정하는 보안 정책을 생성합니다.
2. 정책을 기반으로 서명을 생성합니다. 인증된 요청의 경우 양식에 유효한 서명과 정책이 포함되어야 합니다.

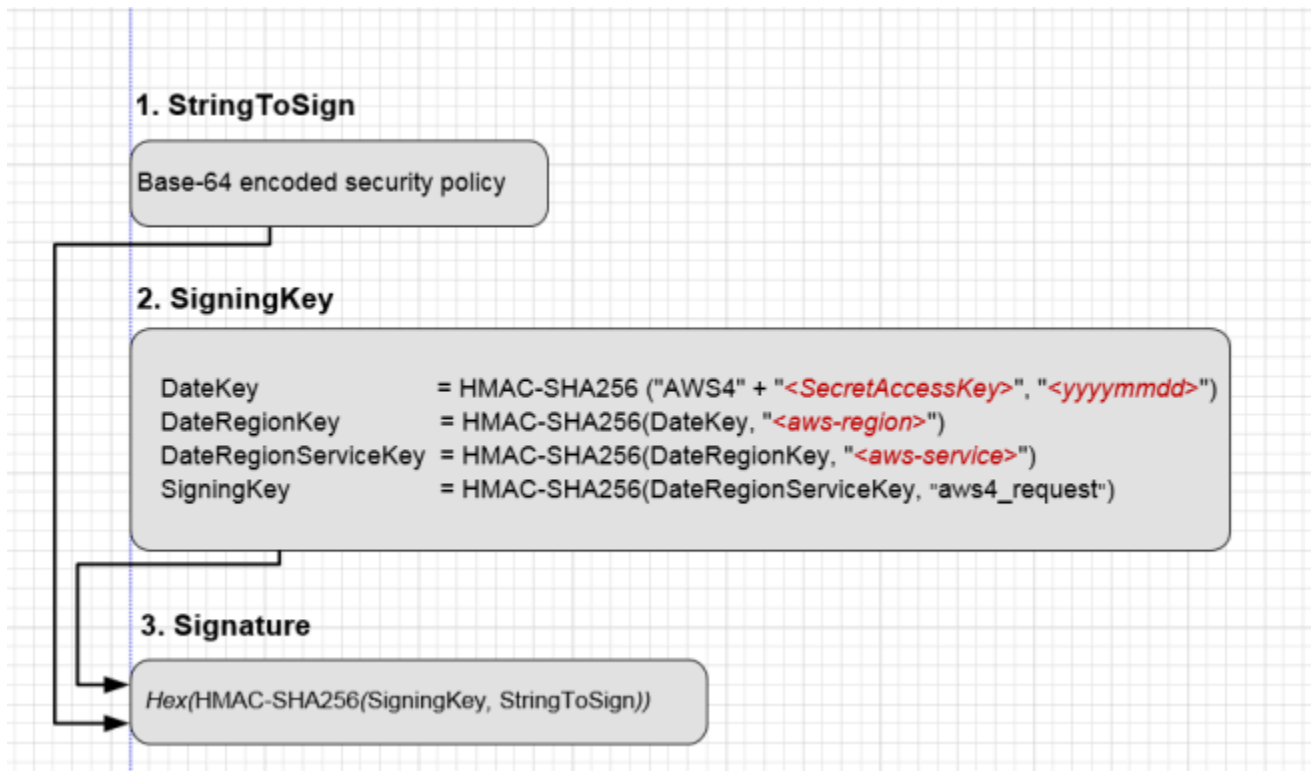
3. 사용자가 Amazon S3 버킷에 객체를 업로드하기 위해 액세스할 수 있는 HTML 양식을 생성합니다.

다음 섹션에서는 요청을 인증하기 위한 서명을 생성하는 방법을 설명합니다. 양식 및 보안 정책 생성에 대한 자세한 내용은 [HTML 양식 생성\(AWS 서명 버전 4 사용\)](#) 섹션을 참조하세요.

서명 계산

인증된 요청의 경우 HTML 양식에 보안 정책 및 서명에 대한 필드가 포함되어야 합니다.

- 보안 정책([POST 정책](#) 참조)은 요청에 허용되는 항목을 제어합니다.
- 보안 정책은 서명 계산의 StringToSign([서명 요청 소개](#) 참조)입니다.



서명을 계산하려면

- UTF-8 인코딩을 사용하여 정책을 생성합니다.
- UTF-8로 인코딩된 정책 바이트를 Base64로 변환합니다. 결과는 StringToSign입니다.
- 서명 키를 생성합니다.

4. 서명 키를 사용하여 HMAC-SHA256 서명 알고리즘으로 StringToSign에 서명합니다.

HTML 양식, 보안 정책 및 예제 생성에 대한 자세한 내용은 다음을 참조하세요.

- [HTML 양식 생성\(AWS 서명 버전 4 사용\)](#)
- [POST 정책](#)
- [예제: HTTP POST를 사용한 브라우저 기반 업로드\(AWS 서명 버전 4 사용\)](#)

HTML 양식 생성(AWS 서명 버전 4 사용)

주제

- [HTML 양식 선언](#)
- [HTML 양식 필드](#)

사용자가 브라우저(HTTP POST 요청)를 사용하여 Amazon S3에 콘텐츠를 업로드할 수 있도록 하려면 HTML 양식을 사용합니다. HTML 양식은 양식 선언과 양식 필드로 구성됩니다. 양식 선언은 요청에 대한 상위 수준 정보를 포함합니다. 양식 필드에는 자세한 요청 정보가 포함됩니다.

이 섹션에서는 HTML 양식을 생성하는 방법을 설명합니다. HTTP POST를 사용한 브라우저 기반 업로드 및 요청 인증을 위한 관련 서명 계산의 작동 예제는 [예제: HTTP POST를 사용한 브라우저 기반 업로드\(AWS 서명 버전 4 사용\)](#) 섹션을 참조하세요.

양식과 정책은 UTF-8로 인코딩되어야 합니다. content 속성에서 charset=UTF-8을 지정하여 양식에 UTF-8 인코딩을 적용할 수 있습니다. 다음은 HTML 헤딩의 UTF-8 인코딩 예제입니다.

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

다음은 요청 헤더의 UTF-8 인코딩 예제입니다.

```
Content-Type: text/html; charset=UTF-8
```

Note

양식 데이터 및 경계(파일의 내용 제외)는 20KB를 초과할 수 없습니다.

HTML 양식 선언

HTML 양식 선언에는 다음과 같은 세 가지 속성이 있습니다.

- **action** - 요청을 처리하는 URL로, 버킷의 URL로 설정되어야 합니다. 예를 들어 버킷의 이름이 `examplebucket`인 경우 URL은 `http://examplebucket.s3.amazonaws.com/`입니다.

Note

키 이름은 양식 필드에 지정됩니다.

- **method** - 메서드는 POST이어야 합니다.
- **enctype** - 엔클로저 유형(enctype)은 파일 업로드와 텍스트 영역 업로드 모두에 대해 `multipart/form-data`로 설정되어야 합니다. enctype에 대한 자세한 내용은 [RFC 1867](#)을 참조하세요.

다음은 버킷 `examplebucket`에 대한 양식 선언입니다.

```
<form action="http://examplebucket.s3.amazonaws.com/" method="post"
enctype="multipart/form-data">
```

HTML 양식 필드

다음 표에서는 양식 내에서 사용할 수 있는 필드의 목록을 설명합니다. 다른 필드 중에는 요청을 인증하는 데 사용할 수 있는 서명 필드가 있습니다. 서명 계산 알고리즘(`x-amz-algorithm`), 서명 키를 생성하는 데 사용한 자격 증명 범위(`x-amz-credential`), 서명을 계산하는 데 사용하는 날짜(`x-amz-date`)를 지정할 수 있는 필드가 있습니다. Amazon S3는 이 정보를 사용하여 서명을 다시 생성합니다. 서명이 일치하면 Amazon S3가 요청을 처리합니다.

Note

변수 `${filename}`은 사용자가 제공한 파일의 이름으로 자동 대체되며 모든 양식 필드가 이 변수를 인식합니다. 브라우저 또는 클라이언트에 파일의 전체 또는 부분 경로가 지정되어 있는 경우 마지막 슬래시(/) 또는 백슬래시(\) 뒤에 오는 텍스트만 사용됩니다(예: `C:\Program Files\directory1\file.txt`가 `file.txt`로 해석됨). 파일이나 파일 이름이 제공되는 경우 변수가 빈 문자열로 대체됩니다.

인증된 요청에 필요한 요소(예: policy 요소)를 제공하지 않으면 요청은 익명으로 간주되고 버킷을 퍼블릭 읽기 및 쓰기로 구성한 경우에만 성공합니다.

요소 이름	설명	필수
acl	Amazon S3 ACL(액세스 제어 목록)입니다. 잘못된 ACL이 지정되면 Amazon S3는 요청을 거부합니다. ACL에 대한 자세한 내용은 Amazon S3 ACL 사용을 참조하세요. 유형: 문자열 기본: 프라이빗 유효한 값: private public-read public-read-write aws-exec-read authenticated-read bucket-owner-read bucket-owner-full-control	아니요
Cache-Control	REST 지정 헤더입니다. 자세한 내용은 PutObject를 참조하세요.	아니요
Content-Type		
Content-Disposition		
Content-Encoding		
Expires		

요소 이름	설명	필수
key	업로드된 객체의 키 이름입니다. 사용자가 제공한 파일 이름을 사용하려면 <code>\${filename}</code> 변수를 사용합니다. 예를 들어 <code>photo1.jpg</code> 파일을 업로드하고 <code>/user/user1/\${filename}</code> 을 키 이름으로 지정하면 파일이 <code>/user/user1/photo1.jpg</code> 로 저장됩니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 객체 키 및 메타데이터를 참조하세요.	예
policy	요청에서 허용되는 항목을 설명하는 Base64로 인코딩된 보안 정책입니다. 인증된 요청의 경우 정책은 필수입니다. 보안 정책이 없는 요청은 익명으로 간주되며 공개적으로 쓰기 가능한 버킷에서만 성공합니다.	인증된 요청의 경우 필수
success_action_redirect	업로드 완료 시 클라이언트가 리디렉션되는 URL입니다. <code>success_action_redirect</code> 를 지정하지 않거나 Amazon S3가 URL을 해석할 수 없는 경우 Amazon S3는 <code>success_action_status</code> 필드에 지정된 빈 문서 유형을 반환합니다. 업로드가 실패하는 경우 Amazon S3는 오류를 반환하고 사용자를 다른 URL로 리디렉션하지 않습니다.	아니요

요소 이름	설명	필수
success_action_status	success_action_redirect 를 지정하지 않을 경우 업로드 완료 시 클라이언트에 반환되는 상태 코드입니다. 유효한 값은 200, 201 또는 204입니다(기본 값). 200 또는 204로 값을 설정하는 경우 Amazon S3는 지정된 상태 코드의 빈 문서를 반환합니다. 201로 값을 설정하는 경우 Amazon S3는 상태 코드가 201인 XML 문서를 반환합니다. XML 문서의 내용에 대한 자세한 내용은 POST Object 섹션을 참조하세요. 값이 설정되지 않거나 잘못된 경우 Amazon S3는 상태 코드가 204인 빈 문서를 반환합니다.  Note 일부 버전의 Adobe Flash Player는 본문이 비어 있는 HTTP 응답을 올바르게 처리하지 못합니다. Adobe Flash를 통한 업로드를 지원하려면 success_action_status 를 201로 설정하는 것이 바람직합니다.	아니요
x-amz-algorithm	요청을 인증하는 데 사용되는 서명 알고리즘입니다. AWS 서명 버전 4의 경우 값은 AWS4-HMAC-SHA256 입니다. 요청과 함께 정책 문서가 포함되는 경우 이 필드는 필수입니다.	인증된 요청의 경우 필수

요소 이름	설명	필수
x-amz-credential	이 필드는 액세스 키 ID 외에도 해당 서명이 유효한 리전 및 서비스를 식별하는 범위 정보를 제공합니다. 이는 서명 계산을 위한 서명 키 계산에 사용한 것과 동일한 범위여야 합니다. 이는 다음 형식의 문자열입니다. <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> 예제: <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request</pre> Amazon S3의 경우 aws-service 문자열은 s3입니다. Amazon S3 aws-region 문자열의 목록은 AWS 일반 참조의 리전 및 엔드포인트를 참조하세요. 요청과 함께 정책 문서가 포함되는 경우 이 필드는 필수입니다.	인증된 요청의 경우 필수
x-amz-date	ISO8601 형식의 날짜 값입니다. 예를 들어 20130728T000000Z 입니다. 이 값은 서명 키를 생성할 때 사용한 날짜와 동일합니다(예: 20130728). 또한 서명한 정책(x-amz-date)에 제공한 값과도 동일해야 합니다. 요청과 함께 정책 문서가 포함되는 경우 이 필드는 필수입니다.	인증된 요청의 경우 필수

요소 이름	설명	필수
x-amz-security-token	Amazon DevPay 및 세션 자격 증명이 사용하는 보안 토큰 요청이 Amazon DevPay를 사용하는 경우 요청에는 제품 토큰과 사용자 토큰에 하나씩 두 개의 x-amz-security-token 양식 필드가 필요합니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 DevPay 사용을 참조하세요. 요청이 세션 자격 증명을 사용 중인 경우 이 필드에는 하나의 x-amz-security-token 양식이 필요합니다. 자세한 내용은 IAM 사용 설명서의 임시 보안 자격 증명 요청 섹션을 참조하세요.	아니요
x-amz-signature	(AWS 서명 버전 4) 보안 정책의 HMAC-SHA256 해시입니다. 요청과 함께 정책 문서가 포함되는 경우 이 필드는 필수입니다.	인증된 요청의 경우 필수
x-amz-meta-*	이 접두사로 시작하는 필드 이름은 사용자 정의 메타데이터입니다. 각 항목은 키-값 페어 세트 로 저장되고 반환됩니다. Amazon S3는 사용자 정의 메타데이터를 검증하거나 해석하지 않습니다. 자세한 내용은 PutObject를 참조하세요.	아니요
x-amz-*	POST 객체를 참조하세요(다른 x-amz-* 헤더는 POST Object 참조).	아니요

요소 이름	설명	필수
file	파일 또는 텍스트 콘텐츠입니다. 파일이나 콘텐츠는 양식의 마지막 필드이어야 합니다. 한 번에 파일 하나만 업로드할 수 있습니다.	예

조건부 항목은 인증된 요청의 경우 필수이고 익명 요청의 경우 선택 사항입니다.

이제 양식을 생성하는 방법을 알았으므로 서명할 수 있는 보안 정책을 생성할 수 있습니다. 자세한 내용은 [POST 정책](#) 섹션을 참조하세요.

POST 정책

주제

- [Expiration](#)
- [조건 매칭](#)
- [조건](#)
- [문자 이스케이프](#)

HTTP POST를 사용하여 인증된 요청을 수행하는 데 필요한 정책은 요청이 충족해야 하는 조건을 지정하는, UTF-8 및 Base64로 인코딩된 JavaScript Object Notation(JSON) 형식 문서입니다. 정책 문서를 디자인하는 방법에 따라 업로드 단위, 사용자 단위, 모든 업로드에 대해 또는 필요에 맞는 다른 디자인에 따라 액세스 세분성을 제어할 수 있습니다.

이 섹션에서는 POST 정책에 대해 설명합니다. POST 정책을 사용한 예제 서명 계산은 [예제: HTTP POST를 사용한 브라우저 기반 업로드\(AWS 서명 버전 4 사용\)](#) 섹션을 참조하세요.

Note

정책 문서는 선택 사항이지만 요청에 허용되는 항목을 제어하려면 정책 문서를 사용하는 것이 좋습니다. 버킷을 공개적으로 쓰기 가능하게 만들면 버킷에 쓸 수 있는 사용자를 전혀 제어할 수 없습니다.

다음은 POST 정책 문서의 예입니다.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"acl": "public-read" },
    {"bucket": "johnsmith" },
    ["starts-with", "$key", "user/eric/"],
  ]
}
```

POST 정책에는 항상 `expiration` 및 `conditions` 요소가 포함됩니다. 이 예제 정책은 두 가지 조건 매칭 유형(정확한 매칭 및 `starts-with` 매칭)을 사용합니다. 다음 섹션에서는 이러한 요소에 대해 설명합니다.

Expiration

`expiration` 요소는 POST 정책의 만료 날짜 및 시간을 ISO8601 GMT 날짜 형식으로 지정합니다. 예를 들어 `2013-08-01T12:00:00.000Z`는 2013년 8월 1일 자정 이후(GMT 기준)에는 POST 정책이 유효하지 않도록 지정합니다.

조건 매칭

다음은 POST 정책 조건(다음 섹션에서 설명)을 지정하는 데 사용할 수 있는 조건 매칭 유형을 설명하는 표입니다. 양식에 지정하는 양식 필드마다 조건을 적어도 하나씩 지정해야 하지만 양식 필드 하나에 여러 조건을 지정하면 좀 더 복잡한 매칭 조건을 만들 수 있습니다.

조건 일치 유형	설명
완전 매치	양식 필드 값이 지정된 값과 일치해야 합니다. 이 예제는 ACL이 <code>public-read</code> 로 설정되어야 한다는 것을 나타냅니다. <pre>{"acl": "public-read" }</pre> 이 예제를 통해 ACL이 <code>public-read</code>로 설정되어야 함을 알 수 있습니다. <pre>["eq", "\$acl", "public-read"]</pre>

조건 일치 유형	설명
Starts With(다음으로 시작)	값이 지정된 값으로 시작해야 합니다. 이 예제는 객체 키가 user/user1로 시작되어야 한다는 것을 나타냅니다. <pre>["starts-with", "\$key", "user/user1/"]</pre>
쉼표로 구분된 목록의 Content-Types 매칭	starts-with 조건에서 쉼표가 포함된 Content-Types 값은 목록으로 해석됩니다. 전체 조건이 통과하려면 목록의 각 값이 조건을 충족해야 합니다. 예를 들어, 조건이 다음과 같은 경우 <pre>["starts-with", "\$Content-Type", "image/"]</pre> 다음 값은 조건을 통과합니다. <pre>"image/jpg,image/png,image/gif"</pre> 다음 값은 조건을 통과하지 못합니다. <pre>["image/jpg,text/plain"]</pre>  Note Content-Type 이외의 데이터 요소는 쉼표의 존재 여부에 관계없이 문자열로 처리됩니다.
내용 무관 매칭	양식 필드 내에 모든 내용을 허용하도록 POST 정책을 구성하려면 빈 값("")이 포함된 starts-with 를 사용합니다. 이 예제에서는 success_action_redirect 에 모든 값을 허용합니다. <pre>["starts-with", "\$success_action_redirect", ""]</pre>

조건 일치 유형	설명
범위 지정	범위를 허용하는 양식 필드의 경우 상한 및 하한을 쉼표로 구분합니다. 이 예제에서는 1~10MiB의 파일 크기가 허용됩니다.
	<code>["content-length-range", 1048576, 10485760]</code>

POST 정책에서 지원되는 특정 조건은 [조건](#) 섹션에 설명되어 있습니다.

조건

POST 정책의 `conditions`는 객체 배열이며, 각 객체는 요청을 검증하는 데 사용됩니다. 이러한 조건을 사용하여 요청에서 허용되는 항목을 제한할 수 있습니다. 예를 들어 위의 정책 조건에는 다음이 필요합니다.

- 요청은 johnsmith 버킷 이름을 지정해야 합니다.
- 객체 키 이름에는 user/eric 접두사가 있어야 합니다.
- 객체 ACL은 public-read로 설정되어야 합니다.

양식에서 지정하는 각 양식 필드(x-amz-signature, file, policy 및 x-ignore- 접두사가 있는 필드 이름 제외)가 조건 목록에 나타나야 합니다.

Note

POST 정책을 검증하기 전에 양식 내의 모든 변수가 확장됩니다. 따라서 모든 조건 매칭은 확장된 양식 필드를 기준으로 해야 합니다. 객체 키 이름을 특정 접두사(user/user1)로 제한한다고 가정해 보겠습니다. 이 경우 키 양식 필드를 user/user1/\${filename}으로 설정합니다. POST 정책은 ["starts-with", "\$key", "user/user1/"]이어야 합니다 (["starts-with", "\$key", "user/user1/\${filename}"]을 입력하지 마세요). 자세한 내용은 [조건 매칭](#) 섹션을 참조하세요.

다음 표에서는 정책 문서 조건을 설명합니다.

요소 이름	설명
acl	양식 제출에 사용해야 하는 ACL 값을 지정합니다. 이 조건은 다음 섹션에서 설명하는 정확한 매칭 및 starts-with 조건 일치 유형을 지원합니다.
bucket	허용되는 버킷 이름을 지정합니다. 이 조건은 정확한 매칭 조건 일치 유형을 지원합니다.
content-length-range	업로드된 콘텐츠의 허용 가능한 최소 및 최대 크기입니다. 이 조건은 content-length-range 조건 일치 유형을 지원합니다.
Cache-Control	REST 지정 헤더입니다. 자세한 내용은 POST Object 섹션을 참조하세요. 이 조건은 정확한 매칭 및 starts-with 조건 일치 유형을 지원합니다.
Content-Type	
Content-Disposition	
Content-Encoding	
Expires	
key	업로드된 객체의 허용 가능한 키 이름 또는 접두사입니다. 이 조건은 정확한 매칭 및 starts-with 조건 일치 유형을 지원합니다.
success_action_redirect	업로드 완료 시 클라이언트가 리디렉션되는 URL입니다. 이 조건은 정확한 매칭 및 starts-with 조건 일치 유형을 지원합니다.
redirect	
success_action_status	

요소 이름	설명
	<code>success_action_redirect</code> 를 지정하지 않을 경우 업로드 완료 시 클라이언트에 반환되는 상태 코드입니다. 이 조건은 정확한 매칭을 지원합니다.
<code>x-amz-algorithm</code>	서명 계산 중에 사용해야 하는 서명 알고리즘입니다. AWS 서명 버전 4의 경우 값은 <code>AWS4-HMAC-SHA256</code> 입니다. 이 조건은 정확한 매칭을 지원합니다.
<code>x-amz-credential</code>	서명 계산에 사용한 자격 증명입니다. 서명이 유효한 리전 및 서비스를 식별하는 액세스 키 ID 및 범위 정보를 제공합니다. 이는 서명 계산을 위한 서명 키 계산에 사용한 것과 동일한 범위여야 합니다. 이는 다음 형식의 문자열입니다. <i><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</i> 예제: <code>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request</code> Amazon S3의 경우 <code>aws-service</code> 문자열은 <code>s3</code>입니다. Amazon S3 <code>aws-region</code> 문자열의 목록은 AWS 일반 참조의 리전 및 엔드포인트를 참조하세요. 요청과 함께 POST 정책 문서가 포함되는 경우 이 필드는 필수입니다. 이 조건은 정확한 매칭을 지원합니다.

요소 이름	설명
x-amz-date	ISO8601 형식 문자열에 지정된 날짜 값입니다. 예를 들어 20130728T000000Z 입니다. 이 날짜는 서명 계산을 위한 서명 키를 생성할 때 사용한 날짜와 동일해야 합니다. 요청과 함께 POST 정책 문서가 포함되는 경우 이 필드는 필수입니다. 이 조건은 정확한 매칭을 지원합니다.
x-amz-security-token	Amazon DevPay 보안 토큰입니다. Amazon DevPay를 사용하는 각 요청에는 제품 토큰과 사용자 토큰에 하나씩 2개의 x-amz-security-token 양식 필드가 필요합니다. 따라서 해당 값을 쉼표로 구분해야 합니다. 예를 들어, 사용자 토큰이 eW91dHViZQ== 이고 제품 토큰이 b0hnNVNKWVJIQTA= 인 경우 POST 정책 항목을 { "x-amz-security-token": "eW91dHViZQ==,b0hnNVNKWVJIQTA=" } 로 설정합니다. Amazon DevPay에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 DevPay 사용을 참조하세요.
x-amz-meta-*	사용자 지정 메타데이터입니다. 이 조건은 정확한 매칭 및 starts-with 조건 일치 유형을 지원합니다.
x-amz-*	POST 객체를 참조하세요(다른 x-amz-* 헤더는 POST Object 참조). 이 조건은 정확한 매칭을 지원합니다.

Note

도구 키트가 양식 필드를 더 추가하는 경우(예: Flash에서 filename 추가) POST 정책 문서에도 추가해야 합니다. 이 기능을 제어할 수 있는 경우 해당 필드에 x-ignore- 접두사를 붙여야 Amazon S3이 해당 기능을 무시하고 나중 버전의 기능에 영향을 주지 않습니다.

문자 이스케이프

다음 표에서는 POST 정책 문서 내에서 이스케이프되어야 하는 문자들을 설명합니다.

이스케이프 시퀀스	설명
\\	백슬래시
\\\$	달러 기호
\\b	백스페이스
\\f	용지 공급
\\n	줄 바꿈
\\r	캐리지 리턴
\\t	가로 탭
\\v	세로 탭
\\uXXXX	모든 유니코드 문자

이제 양식 및 정책을 숙지하고 서명 작동 방식을 이해했으므로 POST 업로드 예제를 시도해 볼 수 있습니다. 서명을 계산하려면 코드를 작성해야 합니다. 이 예제에서는 샘플 양식과 서명 계산을 테스트하는데 사용할 수 있는 POST 정책을 제공합니다. 자세한 내용은 [예제: HTTP POST를 사용한 브라우저 기반 업로드\(AWS 서명 버전 4 사용\)](#) 섹션을 참조하세요.

예제: HTTP POST를 사용한 브라우저 기반 업로드(AWS 서명 버전 4 사용)

이 섹션에서는 HTTP POST 요청을 사용하여 Amazon S3에 콘텐츠를 직접 업로드하는 예제를 보여줍니다.

서명 버전 4에 대한 자세한 내용은 [서명 버전 4 서명 프로세스](#)를 참조하세요.

HTTP POST를 사용하여 Amazon S3에 파일 업로드

이 예제에서는 샘플 POST 정책 그리고 파일을 업로드하는 데 사용할 수 있는 양식을 제공합니다. 이 주제에서는 예제 정책 및 가상 자격 증명을 사용하여 워크플로와 그에 따른 서명 및 정책 해시를 보여줍니다. 이 데이터를 테스트 스위트로 사용하여 서명 계산 코드를 확인할 수 있습니다.

이 예제에서는 다음 예제 자격 증명을 사용하여 서명을 계산합니다. 이러한 자격 증명을 사용하여 서명 계산 코드를 확인할 수 있습니다. 그러나 AWS에 요청을 전송할 때 이러한 자격 증명을 자신의 자격 증명으로 바꾸어야 합니다.

파라미터	값
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

샘플 정책 및 양식

다음 POST 정책은 특정 조건에서 Amazon S3에 대한 업로드를 지원합니다.

```
{ "expiration": "2015-12-30T12:00:00.000Z",
  "conditions": [
    {"bucket": "sigv4examplebucket"},
    ["starts-with", "$key", "user/user1/"],
    {"acl": "public-read"},
    {"success_action_redirect": "http://sigv4examplebucket.s3.amazonaws.com/successful_upload.html"},
    ["starts-with", "$Content-Type", "image/"],
    {"x-amz-meta-uuid": "14365123651274"}]
```

```
{
  "x-amz-server-side-encryption": "AES256"},
  ["starts-with", "$x-amz-meta-tag", ""],

  {"x-amz-credential": "AKIAIOSFODNN7EXAMPLE/20151229/us-east-1/s3/aws4_request"},
  {"x-amz-algorithm": "AWS4-HMAC-SHA256"},
  {"x-amz-date": "20151229T000000Z" }
]
}
```

이 POST 정책은 요청에 대해 다음 조건을 설정합니다.

- 업로드가 2015년 12월 30일 정오 UTC 이전에 발생해야 합니다.
- 콘텐츠는 sigv4examplebucket에만 업로드할 수 있습니다. 제공한 서명은 이 범위 내에서만 유효하므로 버킷은 자격 증명 범위(x-amz-credential 양식 파라미터)에서 지정한 리전에 있어야 합니다.
- user/user1로 시작하는 임의의 키 이름을 제공할 수 있습니다. 예를 들어 user/user1/MyPhoto.jpg입니다.
- ACL은 public-read로 설정해야 합니다.
- 업로드가 성공하면 사용자의 브라우저가 http://sigv4examplebucket.s3.amazonaws.com/successful_upload.html로 리디렉션됩니다.
- 객체가 이미지 파일이어야 합니다.
- x-amz-meta-uuid 태그가 14365123651274로 설정되어야 합니다.
- x-amz-meta-tag에는 모든 값이 포함될 수 있습니다.

다음은 이 POST 정책의 Base64로 인코딩된 버전입니다. 이 값을 서명 계산에서 StringToSign으로 사용합니다.

```
eyJhZiZxhwaXJhdGlvbiI6ICIyMDE1LTExLTUwVDEyOjAwOjAwLjAwMFoiLA0KICAiY29uZGl0aW9ucyI6IFsNCiAgICB7ImJ
```

이전 정책을 복사/붙여 넣기 할 때는 계산된 해시가 이 값과 일치하도록 캐리지 리턴과 새 줄이 있어야 합니다(예: CRLF 행 종결자가 있는 ASCII 텍스트).

예제 자격 증명을 사용하여 서명을 생성하면 서명 값은 다음과 같습니다(서명 계산에서 날짜는 정책의 x-amz-date(20151229)와 동일).

```
8afdbf4008c03f22c2cd3cdb72e4afbb1f6a588f3255ac628749a66d7f09699e
```

다음 예제 양식은 위의 POST 정책을 지정하고 sigv4examplebucket에 대한 POST 요청을 지원합니다. 콘텐츠를 복사하여 텍스트 편집기에 붙여 넣고 exampleform.html로 저장합니다. 그런 다음 exampleform.html을 사용하여 특정 버킷에 이미지 파일을 업로드할 수 있습니다. 제공하는 서명이 Amazon S3에서 계산하는 서명과 일치하면 요청이 성공합니다.

Note

S3에 성공적으로 업로드하려면 버킷 이름, 날짜, 자격 증명, 정책 및 서명을 유효한 값으로 업데이트해야 합니다.

```
<html>
<head>

  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

</head>
<body>

  <form action="http://sigv4examplebucket.s3.amazonaws.com/" method="post"
  enctype="multipart/form-data">
    Key to upload:
    <input type="input" name="key" value="user/user1/${filename}" /><br />
    <input type="hidden" name="acl" value="public-read" />
    <input type="hidden" name="success_action_redirect" value="http://
sigv4examplebucket.s3.amazonaws.com/successful_upload.html" />
    Content-Type:
    <input type="input" name="Content-Type" value="image/jpeg" /><br />
    <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
    <input type="hidden" name="x-amz-server-side-encryption" value="AES256" />
    <input type="text" name="X-Amz-Credential" value="AKIAIOSFODNN7EXAMPLE/20151229/
us-east-1/s3/aws4_request" />
    <input type="text" name="X-Amz-Algorithm" value="AWS4-HMAC-SHA256" />
    <input type="text" name="X-Amz-Date" value="20151229T000000Z" />

    Tags for File:
    <input type="input" name="x-amz-meta-tag" value="" /><br />
    <input type="hidden" name="Policy" value='<Base64-encoded policy string>' />
    <input type="hidden" name="X-Amz-Signature" value="<signature-value>" />
    File:
    <input type="file" name="file" /> <br />
    <!-- The elements after this will be ignored -->
```

```
<input type="submit" name="submit" value="Upload to Amazon S3" />
</form>

</html>
```

POST 파라미터는 대/소문자를 구분하지 않습니다. 예를 들어 `x-amz-signature` 또는 `X-Amz-Signature`를 지정할 수 있습니다.

AWS Amplify 라이브러리를 사용하여 Amazon S3에 브라우저 기반 업로드

이 섹션에서는 AWS Amplify JavaScript 라이브러리를 사용하여 Amazon S3에 파일을 업로드하는 방법을 설명합니다.

AWS Amplify 라이브러리 설정에 대한 자세한 내용은 [AWS Amplify 설치 및 구성](#)을 참조하세요.

AWS Amplify JavaScript 라이브러리를 사용하여 Amazon S3에 파일 업로드

AWS Amplify 라이브러리 Storage 모듈은 퍼블릭 또는 프라이빗 Amazon S3 스토리지에서 사용자 콘텐츠를 관리하기 위한 간단한 브라우저 기반 업로드 메커니즘을 제공합니다.

Example: AWS Amplify 수동 설정

다음 예제에서는 AWS Amplify 스토리지 모듈을 사용하기 위한 수동 설정을 보여줍니다. 스토리지 모듈의 기본 구현에서는 Amazon S3를 사용합니다.

```
import Amplify from 'aws-amplify';
Amplify.configure(
  Auth: {
    identityPoolId: 'XX-XXXX-X:XXXXXXXXX-XXXX-1234-abcd-1234567890ab', //REQUIRED -
    Amazon Cognito Identity Pool ID
    region: 'XX-XXXX-X', // REQUIRED - Amazon Cognito Region
    userPoolId: 'XX-XXXX-X_abcd1234', //OPTIONAL - Amazon Cognito User Pool ID
    userPoolWebClientId: 'XX-XXXX-X_abcd1234', //OPTIONAL - Amazon Cognito Web
    Client ID
  },
  Storage: {
    bucket: '', //REQUIRED - Amazon S3 bucket
    region: 'XX-XXXX-X', //OPTIONAL - Amazon service region
  }
);
```

```
);
```

Example: Amazon S3에 데이터 넣기

다음 예제에서는 Amazon S3에 퍼블릭 데이터를 put하는 방법을 보여 줍니다.

```
Storage.put('test.txt', 'Hello')
  .then (result => console.log(result))
  .catch(err => console.log(err));
```

다음 예제에서는 Amazon S3에 프라이빗 데이터를 put하는 방법을 보여 줍니다.

```
Storage.put('test.txt', 'Private Content', {
  level: 'private',
  contentType: 'text/plain'
})
  .then (result => console.log(result))
  .catch(err => console.log(err));
```

AWS Amplify 스토리지 모듈 사용에 대한 자세한 내용은 [AWS Amplify 스토리지](#)를 참조하세요.

추가 정보

[AWS Amplify 빠른 시작](#)

공통 요청 헤더

다음 표에서는 모든 Amazon S3 REST 요청에서 사용할 수 있는 헤더를 설명합니다.

헤더 이름	설명
Authorization	요청 인증에 필요한 정보. 자세한 내용은 Amazon Simple Storage Service 개발자 안내서의 인증 헤더 를 참조하세요. 익명 요청의 경우 이 헤더는 필요하지 않습니다.
Access-Control-Request-Method	사전 CORS 요청으로 전송되는 HTTP 메서드의 목록입니다. 사전 CORS 평가가 성공하면 지정된 메서드를 다음 CORS 요청에서 사용할 수 있습니다.
Content-Length	RFC 2616에 따른 메시지의 길이(헤더 제외). 이 헤더는 PUT 요청과 XML을 로드하는 작업(예: 로깅 및 ACL)에 필요합니다.
Content-Type	요청의 본문에 콘텐츠가 있는 경우 리소스의 콘텐츠 유형입니다. 예시: text/plain
Content-MD5	RFC 1864에 따른 메시지(헤더 제외)의 Base64로 인코딩된 128 비트 MD5 다이제스트입니다. 이 헤더를 메시지 무결성 검사로 사용하여 데이터가 원래 전송된 데이터와 동일한지 확인할 수 있습니다. 선택 사항이지만 Content-MD5 메커니즘을 엔드 투 엔드 무결성 검사로 사용하는 것이 좋습니다. REST 요청 인증에 대한 자세한 내용은 Amazon Simple Storage Service 개발자 안내서의 REST 인증 을 참조하세요.
Date	Authorization 헤더에 포함되는 서명을 생성할 때 사용할 수 있는 날짜입니다. 서명할 때 Date 헤더를 사용해야 하는 경우에는 ISO 8601 기본 형식에 따라 지정해야 합니다. 이때 x-amz-date 헤더는 필요하지 않습니다. x-amz-date 가 있으면 항상 Date 헤더 값보다 우선하기 때문입니다. 서명할 때 Date 헤더가 사용되지 않는 경우 RFC 2616 섹션 3.3에 지정된 전체 날짜 형식 중 하나일 수 있습니다. 예를 들어 날짜/시간 Wed, 01 Mar 2006 12:00:00 GMT는 Amazon S3에서 사용할 수 있는 유효한 날짜/시간 헤더입니다.

헤더 이름	설명
	서명에 Date 헤더를 사용할 때는 ISO 8601 기본 'YYYYMMDD'T'HHMMSS'Z' 형식을 따라야 합니다. Date를 지정하면서 ISO 8601 기본 형식을 따르지 않는 경우에는 x-amz-date 헤더도 포함해야 합니다. ISO 8601 기본 형식에 따라 Date를 지정하는 경우에는 이 헤더만으로도 요청 서명이 가능하므로 x-amz-date 헤더가 필요 없습니다. 자세한 정보는 Amazon Web Services 용어집의 서명 버전 4의 날짜 처리 섹션을 참조하세요.
Expect	애플리케이션이 100-Continue를 사용하는 경우 승인을 받을 때까지 요청 본문을 전송하지 않습니다. 헤더를 기반으로 메시지가 거부될 경우 메시지의 본문이 전송되지 않습니다. 이 헤더는 본문을 전송하는 경우에만 사용할 수 있습니다. 유효한 값: 100-Continue
Host	경로 방식 요청의 경우 값은 s3.amazonaws.com 입니다. 가상 방식 요청의 경우 값은 BucketName.s3.amazonaws.com 입니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 가상 호스팅을 참조하세요. 이 헤더는 HTTP 1.1의 경우 필수이고(대부분의 도구 키트는 이 헤더를 자동으로 추가함), HTTP/1.0 요청의 경우 선택 사항입니다.
Origin	초기 요청자의 서버 이름을 지정하는 엔드포인트입니다.
x-amz-content-sha256	서명 버전 4를 사용하여 요청을 인증하는 경우 이 헤더는 요청 페이로드의 해시를 제공합니다. 자세한 내용은 Authorization 헤더에 대한 서명 계산: 페이로드를 단일 청크로 전송(AWS 서명 버전 4)을 참조하세요. 객체를 청크로 업로드하는 경우 서명이 헤더만 포함하고 페이로드가 없음을 나타내려면 값을 STREAMING-AWS4-HMAC-SHA256-PAYLOAD 로 설정합니다. 자세한 내용은 Authorization 헤더에 대한 서명 계산: 페이로드를 여러 청크로 전송(청크 업로드)(AWS 서명 버전 4) 섹션을 참조하세요.

헤더 이름	설명
x-amz-date	Authorization 헤더에서 서명을 생성할 때 사용하는 날짜입니다. 형식은 ISO 8601 기본 형식(YYYYMMDD'T'HHMMSS'Z')을 따라야 합니다. 예를 들어 날짜/시간 20170210T120000Z 는 Amazon S3에서 사용할 수 있는 유효한 x-amz-date 입니다. x-amz-date 는 모든 요청에서 선택 사항이지만 서명 요청에 사용되는 날짜보다 우선할 때 사용됩니다. Date 헤더가 ISO 8601 기본 형식으로 지정되면 x-amz-date 는 필요하지 않습니다. x-amz-date 가 있으면 항상 Date 헤더 값보다 우선하기 때문입니다. 자세한 정보는 Amazon Web Services 용어집의 서명 버전 4의 날짜 처리 섹션을 참조하세요.
x-amz-security-token	이 헤더는 다음 시나리오에서 사용할 수 있습니다. - Amazon DevPay 작업에 보안 토큰을 제공 - Amazon DevPay 를 사용하는 각 요청에는 두 개의 x-amz-security-token 헤더가 필요합니다. 하나는 제품 토큰용이고 다른 하나는 사용자 토큰용입니다. Amazon S3는 인증된 요청을 수신하면 계산된 서명을 제공된 서명과 비교합니다. 서명을 계산하는 데 사용되는 잘못된 형식의 다중 값 헤더는 인증 문제를 일으킬 수 있습니다. - 임시 보안 자격 증명을 사용할 때 보안 토큰을 제공 - IAM에서 얻은 임시 보안 자격 증명을 사용하여 요청할 때 이 헤더를 사용하여 보안 토큰을 제공해야 합니다. 임시 보안 자격 증명에 대한 자세한 내용은 요청하기를 참조하세요. 이 헤더는 Amazon DevPay를 사용하는 요청과 임시 보안 자격 증명을 사용하여 서명된 요청에 필요합니다.

공통 응답 헤더

다음 표에서는 대부분의 Amazon S3 응답에 공통적으로 나타나는 응답 헤더를 설명합니다.

이름	설명
Access-Control-Allow-Credentials	서버가 CORS 요청에 자격 증명을 포함하도록 허용하는지 여부를 결정하는 부울입니다. Access-Control-Allow-Origin 요청 헤더가 '*'로 설정된 경우 Access-Control-Allow-Credentials 응답 헤더는 생략되고, 그렇지 않으면 CORS 평가가 성공할 때 true로 설정됩니다. 유형: Boolean 기본값: None
Access-Control-Allow-Headers	CORS 요청에 허용되는 HTTP 헤더의 목록입니다. Access-Control-Allow-Headers 응답 헤더는 성공적인 CORS 평가에 대해 반환되며 허용되는 모든 Access-Control-Request-Headers를 명시적으로 지정합니다. 유형: 문자열 기본값: None
Access-Control-Allow-Methods	허용되는 HTTP 메서드를 지정하는 목록입니다. Amazon S3는 CORS 평가가 성공하는 경우에만 허용된 CORS 메서드의 CORS 요청을 허용합니다. 유형: 문자열 기본값: None
Access-Control-Allow-Origin	허용된 오리진의 위치입니다. Amazon S3는 CORS 평가가 성공한 경우에만 Access-Control-Allow-Origin 응답 헤더를 전송합니다. 요청 오리진이 CORS 구성의 허용된 오리진에 있는 '*'와 일치하는 경우 '*'는 원래 오리진 대신 이 응답 헤더에서 반환됩니다. 유형: 문자열 기본값: None

이름	설명
Access-Control-Expose-Headers	CORS 평가가 성공할 때 서버가 애플리케이션이 액세스할 수 있는 응답 헤더를 식별할 수 있도록 허용하는 목록입니다. 유형: 문자열 기본값: None
Access-Control-Max-Age	브라우저가 리소스, HTTP 메서드, 오리진으로 식별되는 CORS 사전 요청에 대한 응답을 캐시할 수 있는 시간(초)입니다. Access-Control-Max-Age 응답 헤더는 CORS 평가가 성공한 경우에만 반환됩니다. 유형: 정수 기본값: None
Vary	CORS 평가 결과가 어떤 요청 헤더에 따라 달라지는지를 나타내는 목록입니다. Vary 응답 헤더는 CORS 평가가 성공한 경우에만 반환됩니다. 유형: 문자열 기본값: None
Content-Length	응답 본문의 길이(바이트)입니다. 유형: 문자열 기본값: None
Content-Type	콘텐츠의 MIME 유형입니다. 예를 들어 Content-Type: text/html; charset=utf-8 입니다. 유형: 문자열 기본값: None

이름	설명
Connection	서버에 대한 연결이 열려 있는지 또는 닫혀 있는지 지정하는 값입니다. 형식: 열거형 유효한 값: open close 기본값: None
Date	Amazon S3가 응답한 날짜 및 시간(예: 2006년 3월 1일 수요일 12:00:00 GMT)입니다. 유형: 문자열 기본값: None
ETag	엔터티 태그(ETag)는 객체의 특정 버전을 나타냅니다. ETag는 객체의 콘텐츠에 대한 변경 사항만 반영하고 메타데이터에 대한 변경을 반영하지 않습니다. ETag는 객체 데이터의 MD5 다이제스트일 수도 아닐 수도 있습니다. 다이제스트인지 여부는 다음과 같이 객체 생성 방식 및 암호화 방식에 좌우됩니다. • AWS Management Console을 통해 또는 PUT 객체, POST Object 또는 Copy 작업으로 생성된 객체: • Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)로 암호화된 객체 또는 일반 텍스트 객체에는 해당 데이터의 MD5 다이제스트인 ETag가 있습니다. • 고객 제공 키를 사용한 서버 측 암호화(SSE-C) 또는 AWS Key Management Service(AWS KMS) 키를 사용한 서버 측 암호화(SSE-KMS)로 암호화된 객체에는 해당 객체 데이터의 MD5 다이제스트가 아닌 ETag가 있습니다. • 멀티파트 업로드 또는 파트 복사 업로드 작업으로 생성된 객체에는 암호화 방법에 관계없이 MD5 다이제스트가 아닌 ETag가 있습니다. 유형: 문자열

이름	설명
Server	응답을 생성한 서버의 이름입니다. 유형: 문자열 기본값: AmazonS3
x-amz-delete-marker	반환된 객체가 삭제 마커였는지(true) 또는 아니었는지(false) 지정하는 값입니다. 유형: Boolean 유효한 값: true false 기본값: false
x-amz-id-2	AWS가 문제를 해결하는 데 도움이 되도록 x-amz-request-id 헤더와 함께 사용되는 특수 토큰입니다. 이러한 요청 ID를 사용하는 지원에 대한 자세한 내용은 Amazon S3 문제 해결을 참조하세요. 유형: 문자열 기본값: None
x-amz-request-id	요청을 고유하게 식별하기 위해 Amazon S3에서 생성된 값입니다. 이 값은 AWS가 문제를 해결하는 데 도움이 되도록 x-amz-id-2 헤더와 함께 사용됩니다. 이러한 요청 ID를 사용하는 지원에 대한 자세한 내용은 Amazon S3 문제 해결을 참조하세요. 유형: 문자열 기본값: None
x-amz-server-side-encryption	Amazon S3에서 이 객체를 저장할 때 사용되는 서버 측 암호화 알고리즘입니다(예: AES256, aws:kms). 유효한 값: AES256 aws:kms

이름	설명
x-amz-version-id	객체의 버전입니다. 버전 관리를 활성화하면 Amazon S3는 버킷에 추가되는 객체에 대해 난수를 생성합니다. 이 값은 UTF-8로 인코딩되며 URL 지원 형식입니다. 버전 관리가 일시 중지된 버킷에서 객체를 PUT하면 버전 ID는 항상 null입니다. 유형: 문자열 유효한 값: null 모든 URL 지원 UTF-8 인코딩 문자열 기본값: null

오류 응답

이 섹션에서는 Amazon S3 오류에 대한 참조 정보를 제공합니다.

Note

- 일반적으로 S3 버킷 소유자에게는 HTTP 200 OK 성공 응답 및 HTTP 4XX 클라이언트 오류 응답이 반환된 요청에 대해 요금이 청구됩니다. 버킷 소유자에게는 HTTP 503 Slow Down 오류와 같은 HTTP 5XX 서버 오류 응답에 대해서는 요금이 청구되지 않습니다. 요금이 청구되지 않는 HTTP 3XX 및 4XX 상태 코드 아래의 S3 오류 코드에 대한 자세한 내용은 Amazon S3 사용 설명서의 [Amazon S3 오류 응답에 대한 청구](#)를 참조하세요. 버킷이 요청자 지불 버킷으로 구성된 경우 청구 요금에 대한 자세한 내용은 Amazon S3 사용 설명서의 [요청자 지불 요금 작동 방식](#)을 참조하세요.
- HTTP를 통한 SOAP 지원은 중단되었지만 SOAP는 HTTPS를 통해 계속해서 사용할 수 있습니다. Amazon S3의 새로운 기능들은 SOAP에서 지원되지 않습니다. SOAP를 사용하는 대신 REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

주제

- [REST 오류 응답](#)
- [오류 코드 목록](#)
- [SELECT 객체 콘텐츠 오류 코드 목록](#)
- [복제 관련 오류 코드 목록](#)
- [태그 지정 관련 오류 코드 목록](#)
- [Amazon S3 주식 관련 오류 코드 목록](#)
- [Amazon S3 on Outposts 오류 코드 목록](#)
- [Amazon S3 Storage Lens 오류 코드 목록](#)
- [Amazon S3 Object Lambda 오류 코드 목록](#)
- [Amazon S3 비동기 오류 코드 목록](#)
- [Amazon S3 Access Grants 오류 코드 목록](#)
- [Amazon FSx 관련 오류 코드 목록](#)
- [Amazon S3 Tables 오류 코드 목록](#)
- [Amazon S3 오류 모범 사례](#)

REST 오류 응답

오류가 발생하면 헤더 정보에 다음이 포함됩니다.

- 콘텐츠 유형: application/xml
- 적합한 3xx, 4xx 또는 5xx HTTP 상태 코드

응답의 본문에도 오류에 대한 정보가 포함됩니다. 다음 샘플 오류 응답은 모든 REST 오류 응답에 공통된 응답 요소의 구조를 보여줍니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>NoSuchKey</Code>
  <Message>The resource you requested does not exist</Message>
  <Resource>/mybucket/myfoto.jpg</Resource>
  <RequestId>4442587FB7D0A2F9</RequestId>
</Error>
```

다음 표에서는 REST 오류 응답 요소를 설명합니다.

이름	설명
Code	오류 코드는 오류 상태를 식별하는 고유한 문자열입니다. 이 코드는 오류를 감지하여 유형별로 처리하는 프로그램에서 판독하고 이해하기 위한 것입니다. 자세한 내용은 오류 코드 목록을 참조하세요. 유형: 문자열 상위 항목: Error
Error	모든 오류 요소를 위한 컨테이너입니다. 유형: 컨테이너 상위 항목: 없음
Message	오류 메시지는 오류 상태에 관한 일반적인 설명을 담고 있으며 영어로 제공됩니다. 이는 기계가 아닌 사람을 위한 것입니다. 간단한 프로그램은 자체적으로 확인되지 않거나 어떻게 처리해야 할지 모르는 오류 상태가 발생하면 최종 사용자에게

이름	설명
	게 직접 메시지를 표시합니다. 좀 더 포괄적인 오류 처리 기능을 갖고 있고 적절히 국제화된 고급 프로그램은 오류 메시지를 무시할 가능성이 큼니다. 유형: 문자열 상위 항목: Error
RequestId	오류와 관련된 요청의 ID입니다. 유형: 문자열 상위 항목: Error
Resource	오류와 관련된 버킷 또는 객체입니다. 유형: 문자열 상위 항목: Error

많은 오류 응답에는 프로그래밍 오류를 진단하는 개발자가 판독하고 이해하기 위한 체계적 데이터가 추가로 포함되어 있습니다. 예를 들어, REST PUT 요청에 서버에서 계산된 다이제스트와 일치하지 않는 Content-MD5 헤더를 보내면 BadDigest 오류가 반환됩니다. 오류 응답에는 서버가 계산한 다이제스트와 서버가 예상한 다이제스트와 같은 세부 요소도 포함됩니다. 개발 단계 이 정보를 사용하여 오류를 진단할 수 있습니다. 프로덕션 단계에서 모범적인 프로그램은 오류 로그에 이 내용을 포함할 수 있습니다.

일반 응답 요소에 대한 자세한 내용은 [오류 응답](#)을 참조하세요.

오류 코드 목록

다음 표에는 Amazon S3 오류 코드가 나열되어 있습니다.

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
AccessControlListNotSupported	버킷은 ACL을 허용하지 않습니다.	400 Bad Request	클라이언트
AccessDenied	액세스 거부됨	403 Forbidden	클라이언트
AccessPointAlreadyOwnedByYou	계정에 동일한 이름의 액세스 포인트가 이미 있습니다.	409 Conflict	클라이언트
AccountProblem	AWS 계정에 해당 작업을 성공적으로 완료하지 못하도록 방해하는 문제가 있습니다. 추가 지원이 필요한 경우 문의처 를 참조하세요.	403 Forbidden	클라이언트
AllAccessDisabled	이 Amazon S3 리소스에 대한 모든 액세스 권한이 비활성화되었습니다. 추가 지원이 필요한 경우 문의처 를 참조하세요.	403 Forbidden	클라이언트
AmbiguousGrantByEmailAddress	제공한 이메일 주소가 여러 계정과 연결되어 있습니다.	400 Bad Request	클라이언트
AuthorizationHeaderMalformed	제공한 권한 부여 헤더가 유효하지 않습니다.	400 Bad Request	해당 사항 없음

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
AuthorizationQueryParametersError	제공한 권한 부여 쿼리 매개변수가 유효하지 않습니다.	400 Bad Request	해당 사항 없음
BadDigest	지정한 Content-MD5 또는 체크섬 값이 서버가 수신한 값과 일치하지 않습니다.	400 Bad Request	클라이언트
BucketAlreadyExists	요청된 버킷 이름을 사용할 수 없습니다. 버킷 네임스페이스는 시스템의 모든 사용자가 공유합니다. 다른 이름을 지정하고 다시 시도합니다.	409 Conflict	클라이언트
BucketAlreadyOwnedByYou	생성하려는 버킷이 이미 있으며 사용자의 소유입니다. Amazon S3는 미국 동부(버지니아 북부) 리전(us-east-1)을 제외한 모든 AWS 리전에서 이 오류를 반환합니다. 레거시 호환성을 위해 us-east-1에서 이미 소유한 기존 버킷을 다시 생성하는 경우 Amazon S3는 200 OK를 반환하고 버킷 액세스 제어 목록(ACL)을 재설정합니다. Amazon S3 on Outposts의 경우, 생성하려는 버킷이 이미 Outpost에 존재하며 사용자의 소유입니다.	409 Conflict east-1을 제외한 모든 리전)	클라이언트
BucketHasAccessPointsAttached	삭제하려는 버킷에 액세스 포인트가 연결되어 있습니다. 버킷을 삭제하기 전에 액세스 포인트를 삭제하세요.	400 Bad Request	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
BucketNotEmpty	삭제하려는 버킷이 비어 있지 않습니다.	409 Conflict	클라이언트
ClientTokenConflict	다중 리전 액세스 포인트 맥등성 토큰이 이미 다른 요청에 사용되었습니다.	409 Conflict	클라이언트
ConnectionClosedByRequester	WriteGetObjectResponse 본문을 읽는 동안 오류가 발생하여 원래 호출자에게 반환되었습니다.	400 Bad Request	클라이언트
ConditionalRequestConflict	충돌하는 작업이 발생했습니다. PutObject 를 사용하는 경우 요청을 재시도할 수 있습니다. 멀티파트 업로드를 사용하는 경우 다른 CreateMultipartUpload 요청을 시작하고 각 파트를 다시 업로드해야 합니다.	409 Conflict	클라이언트
CredentialsNotSupported	이 요청은 자격 증명을 지원하지 않습니다.	400 Bad Request	클라이언트
CrossLocationLoggingProhibited	교차 리전 로깅은 허용되지 않습니다. 한 AWS 리전의 버킷은 다른 리전의 버킷에 정보를 로깅할 수 없습니다.	403 Forbidden	클라이언트
DeviceNotActiveError	디바이스가 현재 활성화 상태가 아닙니다.	400 Bad Request	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
EndpointNotFound	요청을 올바른 엔드포인트로 전달합니다.	400 Bad Request	클라이언트
EntityTooSmall	제안된 업로드가 허용되는 최소 객체 크기보다 작습니다.	400 Bad Request	클라이언트
EntityTooLarge	제안된 업로드가 허용되는 최대 객체 크기를 초과합니다. 자세한 내용은 AWS 일반 참조의 Amazon Simple Storage Service 엔드포인트 및 할당량 을 참조하세요.	400 Bad Request	클라이언트
	제안된 다운로드가 최대 허용 크기를 초과합니다.	405 Method Not Allowed	클라이언트
ExpiredToken	제공된 토큰이 만료되었습니다.	400 Bad Request	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
<code>IllegalLocationConstraintException</code>	이 오류는 다음과 같은 이유로 발생할 수 있습니다. - 버킷이 있는 리전과 다른 리전의 버킷에 액세스하려고 합니다. - 요청이 전송된 리전 엔드포인트와 다른 리전에 해당하는 위치 제약 조건이 있는 버킷을 생성하려고 시도합니다.	400 Bad Request	클라이언트
<code>IllegalVersioningConfigurationException</code>	요청에 지정된 버전 관리 구성이 유효하지 않습니다.	400 Bad Request	클라이언트
<code>IncompleteBody</code>	Content-Length HTTP 헤더에 지정된 바이트 수를 제공하지 않았습니다.	400 Bad Request	클라이언트
<code>IncorrectEndpoint</code>	지정된 버킷이 다른 리전에 있습니다. 요청을 올바른 엔드포인트로 전달합니다.	400 Bad Request	클라이언트
<code>IncorrectNumberOfFilesInPostRequest</code>	POST는 요청당 정확히 하나의 파일 업로드가 필요합니다.	400 Bad Request	클라이언트
<code>InlineDataTooLarge</code>	인라인 데이터가 최대 허용 크기를 초과했습니다.	400 Bad Request	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
InternalError	내부 오류가 발생했습니다. 다시 시도하세요.	500 Internal Server Error	Server
InvalidAccessKeyId	입력한 AWS 액세스 키 ID가 AWS의 레코드에 존재하지 않습니다.	403 Forbidden	클라이언트
InvalidAccessPoint	지정된 액세스 포인트 이름 또는 계정이 유효하지 않습니다.	400 Bad Request	클라이언트
InvalidAccessPointAliasError	지정된 액세스 포인트 별칭 이름이 유효하지 않습니다.	400 Bad Request	클라이언트
InvalidAddressingHeader	익명 역할을 지정해야 합니다.	해당 사항 없음	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
InvalidArgument	이 오류는 다음과 같은 이유로 발생할 수 있습니다. - ListBuckets 요청이 bucket-region 파라미터에 지정된 리전과 다른 리전 엔드포인트로 전송되었습니다. - 지정된 인수가 유효하지 않습니다. - 요청에 필수 헤더가 누락되었습니다. - 지정된 인수가 불완전하거나 형식이 잘못되었습니다. - 지정된 인수의 길이는 3 이상이어야 합니다.	400 Bad Request	클라이언트
InvalidBucketAclWithObjectOwnership	버킷에서는 ObjectOwnership의 BucketOwnerEnforced 설정으로 ACL을 설정할 수 없습니다.	400 Bad Request	클라이언트
InvalidBucketName	지정된 버킷이 유효하지 않습니다.	400 Bad Request	클라이언트
InvalidBucketOwnerAWSAccountID	예상되는 버킷 소유자 파라미터의 값은 AWS 계정 ID여야 합니다.	400 Bad Request	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
InvalidBucketState	요청이 버킷의 현재 상태에 대해 유효하지 않습니다.	409 Conflict	클라이언트
InvalidDigest	지정한 Content-MD5 또는 체크섬 값이 유효하지 않습니다.	400 Bad Request	클라이언트
InvalidEncryptionAlgorithmError	지정한 암호화 요청이 유효하지 않습니다. 유효한 값은 AES256입니다.	400 Bad Request	클라이언트
InvalidHostHeader	요청에 제공된 호스트 헤더가 잘못된 스타일 주소를 사용했습니다.	400 Bad Request	클라이언트
InvalidHttpMethod	예상치 못한 HTTP 메서드를 사용하여 요청했습니다.	400 Bad Request	클라이언트
InvalidLocationConstraint	지정된 위치(리전) 제약 조건이 유효하지 않습니다. 버킷의 리전 선택에 대한 자세한 내용은 버킷 개요 를 참조하세요.	400 Bad Request	클라이언트
InvalidObjectState	작업이 객체의 현재 상태에 유효하지 않습니다.	403 Forbidden	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
InvalidPart	지정된 파트 중 하나 이상을 찾을 수 없습니다. 파트가 업로드되지 않았거나 지정된 엔터티 태그가 파트의 엔터티 태그와 일치하지 않은 것일 수 있습니다.	400 Bad Request	클라이언트
InvalidPartOrder	파트 목록이 오름차순이 아닙니다. 파트 목록은 파트 번호 순서대로 지정해야 합니다.	400 Bad Request	클라이언트
InvalidPayer	이 개체에 대한 모든 액세스가 비활성화되었습니다. 추가 지원이 필요한 경우 문의처 를 참조하세요.	403 Forbidden	클라이언트
InvalidPolicyDocument	양식의 콘텐츠가 정책 문서에 지정된 조건을 충족하지 않습니다.	400 Bad Request	클라이언트
InvalidRange	요청한 범위를 충족할 수 없습니다.	416 요청 범위가 만족스럽지 않음	클라이언트
InvalidRegion	옵트인하지 않은 리전에서 다중 리전 액세스 포인트를 생성하려고 했습니다.	403 금지됨	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
InvalidRequest	이 오류는 다음과 같은 이유로 발생할 수 있습니다. - 승인된 범용 버킷 할당량이 10,000개를 초과하는 계정에서 페이지가 매겨지지 않은 ListBuckets 요청을 실행합니다. 버킷이 10,000개를 초과하는 계정의 버킷을 나열하려면 페이지가 매겨진 요청을 실행해야 합니다. - 요청에서 잘못된 서명 버전을 사용하고 있습니다. AWS4-HMAC-SHA256(서명 버전 4)을 사용합니다. - 액세스 포인트는 기존 버킷에 대해서만 생성할 수 있습니다. - 액세스 포인트가 삭제할 수 있는 상태가 아닙니다. - 액세스 포인트는 기존 버킷에 대해서만 나열할 수 있습니다. - 다음 토큰이 유효하지 않습니다. - 수명 주기 규칙에 하나 이상의 작업을 지정해야 합니다.	400 Bad Request	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
	하나 이상의 수명 주기 규칙을 지정해야 합니다. - 수명 주기 규칙의 수는 허용된 규칙 수인 1,000개를 초과할 수 없습니다. - MaxResults 파라미터의 범위가 유효하지 않습니다. - SOAP 요청은 HTTPS 연결을 통해 이루어져야 합니다. - Amazon S3 Transfer Acceleration에서는 DNS를 준수하지 않는 이름의 버킷이 지원되지 않습니다. - Amazon S3 Transfer Acceleration에서는 이름에 마침표(.)가 포함된 버킷이 지원되지 않습니다. - Amazon S3 Transfer Acceleration 엔드포인트는 시각적 스타일 요청만 지원합니다. - Amazon S3 Transfer Acceleration이 이 버킷에서 구성되어 있지 않습니다.		

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
	Amazon S3 Transfer Acceleration 이 이 버킷에서 비활성화되어 있습니다. - Amazon S3 Transfer Acceleration 이 이 버킷에서 지원되지 않습니다. 도움이 필요하면 지원로 문의하십시오. - Amazon S3 Transfer Acceleration 을 이 버킷에서 활성화할 수 없습니다. 도움이 필요하면 지원로 문의하십시오. - HTTP 헤더와 쿼리 파라미터에 제공된 값이 서로 충돌합니다. - HTTP 헤더와 POST 양식 필드에 제공된 값이 서로 충돌합니다. - 크기가 5GB보다 큰 객체에 대한 CopyObject 요청이 이루어졌습니다.		
InvalidSessionException	제한 시간이 초과되었거나 만료되어 세션이 더 이상 존재하지 않는 경우 반환됩니다.	400 Bad Request	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
InvalidSignature	서버가 계산한 요청 서명이 사용자가 제공한 서명과 일치하지 않습니다. AWS 비밀 액세스 키와 서명 방법을 확인합니다. 자세한 내용은 REST 요청 서명 및 인증 을 참조하세요.	400 Bad Request	클라이언트
InvalidSecurity	입력한 보안 자격 증명이 유효하지 않습니다.	403 Forbidden	클라이언트
InvalidSOAPRequest	SOAP 요청 본문이 유효하지 않습니다.	400 Bad Request	클라이언트
InvalidStorageClass	지정한 스토리지 클래스가 유효하지 않습니다.	400 Bad Request	클라이언트
InvalidTargetBucketForLogging	로깅을 위한 대상 버킷이 존재하지 않거나, 사용자가 소유하지 않거나, 로그 전송 그룹에 대한 적절한 권한이 없습니다.	400 Bad Request	클라이언트
InvalidToken	입력된 토큰의 형식이 잘못되었거나 다른 이유로 유효하지 않습니다.	400 Bad Request	클라이언트
InvalidURI	지정된 URI를 구문 분석할 수 없습니다.	400 Bad Request	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
KeyTooLongError	키가 너무 길니다.	400 Bad Request	클라이언트
KMS.DisabledException	지정한 KMS 키가 활성화되지 않아서 요청이 거부되었습니다.	400 Bad Request	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
KMS.InvalidKeyUsageException	다음과 같은 이유 중 하나로 요청이 거부되었습니다. - KMS 키의 KeyUsage 값이 API 작업과 호환되지 않습니다. - 작업에 지정된 암호화 알고리즘 또는 서명 알고리즘이 KMS 키 (KeySpec)의 키 구성 요소 유형과 호환되지 않습니다. 암호화, 암호 해독, 재암호화 및 데이터 키 생성을 위해서는 KeyUsage가 ENCRYPT_DECRYPT여야 합니다. 메시지 서명 및 검증을 위해서는 KeyUsage가 SIGN_VERIFY여야 합니다. 메시지 인증 코드(MAC)를 생성하고 확인하려면 KeyUsage가 GENERATE_VERIFY_MAC이어야 합니다. 키 일치 비밀을 도출하려면 KeyUsage가 KEY_AGREEMENT여야 합니다. KMS 키의 KeyUsage를 확인하려면 DescribeKey 작업을 사용합니다. 특정 KMS 키에 지원되는 암호화 또는 서명 알고리즘을 찾으려면 DescribeKey 작업을 사용합니다.	400 Bad Request	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
KMS.KMSInvalidStateException	지정한 리소스의 상태가 이 요청에 유효하지 않아서 요청이 거부되었습니다. 이 예외는 다음 중 하나를 의미합니다. - KMS 키의 키 상태가 작업과 호환되지 않습니다. 키 상태를 확인하려면 DescribeKey 작업을 사용합니다. 각 KMS 작업과 호환되는 키 상태에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 AWS KMS keys 키 상태를 참조하세요. - 사용자 지정 키 저장소의 KMS 키에 대한 암호화 작업 시 이 예외는 여러 가지 원인이 있을 수 있는 일반적인 오류를 나타냅니다. 원인을 식별하려면 예외와 함께 제공되는 오류 메시지를 참조하세요.	400 Bad Request	클라이언트
KMS.NotFoundException	지정한 엔터티 또는 리소스를 찾을 수 없으므로 요청이 거부되었습니다.	400 Bad Request	클라이언트
MalformedACLError	제공한 ACL이 잘못된 형식으로 되어 있고 게시된 스키마를 기준으로 검증되지 않았습니다.	400 Bad Request	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
MalformedPOSTRequest	POST 요청 본문이 올바른 형식의 멀티파트/양식 데이터가 아닙니다.	400 Bad Request	클라이언트
MalformedXML	제공한 XML이 잘못된 형식으로 되어 있고 게시된 스키마를 기준으로 검증되지 않았습니다.	400 Bad Request	클라이언트
MaxMessageLengthExceeded	요청이 너무 큼니다.	400 Bad Request	클라이언트
MaxPostPreDataLengthExceededError	업로드 파일 앞에 있는 POST 요청 필드가 너무 컸습니다.	400 Bad Request	클라이언트
MetadataTooLarge	메타데이터 헤더가 최대 허용 메타데이터 크기를 초과합니다.	400 Bad Request	클라이언트
MethodNotAllowed	지정된 메서드가 이 리소스에 허용되지 않습니다.	405 Method Not Allowed	클라이언트
MissingAttachment	SOAP 첨부 파일이 필요했지만 해당 파일이 없었습니다.	400 Bad Request	클라이언트
MissingAuthenticationToken	요청이 서명되지 않았습니다.	403 금지됨	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
MissingContentLength	콘텐츠 길이 HTTP 헤더를 제공해야 합니다.	411 Length Required	클라이언트
MissingRequestBodyError	빈 XML 문서를 요청으로 보냈습니다.	400 Bad Request	클라이언트
MissingSecurityElement	SOAP 1.1 요청에 보안 요소가 없습니다.	400 Bad Request	클라이언트
MissingSecurityHeader	요청에 필수 헤더가 누락되었습니다.	400 Bad Request	클라이언트
NoLoggingStatusForKey	키에는 로깅 상태 하위 리소스라는 것이 없습니다.	400 Bad Request	클라이언트
NoSuchAsyncRequest	지정된 리소스를 찾을 수 없습니다.	404 Not Found	클라이언트
NoSuchBucket	지정된 버킷이 존재하지 않습니다.	404 Not Found	클라이언트
NoSuchBucketPolicy	지정된 버킷에 버킷 정책이 없습니다.	404 Not Found	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
NoSuchCORSConfiguration	지정된 버킷에 CORS 구성이 없습니다.	404 Not Found	클라이언트
NoSuchKey	지정된 키가 없습니다.	404 Not Found	클라이언트
NoSuchLifecycleConfiguration	지정된 수명 주기 구성이 없습니다.	404 Not Found	클라이언트
NoSuchMultiRegionAccessPoint	지정된 다중 리전 액세스 포인트가 존재하지 않습니다.	404 Not Found	클라이언트
NoSuchObjectLockConfiguration	지정된 객체에 ObjectLock 구성이 없습니다.	404 Not Found	클라이언트
NoSuchWebsiteConfiguration	지정된 버킷에 웹 사이트 구성이 없습니다.	404 Not Found	클라이언트
NoSuchTagSet	지정된 태그가 없습니다.	404 Not Found	클라이언트
NoSuchUpload	지정된 멀티파트 업로드가 존재하지 않습니다. 업로드 ID가 유효하지 않거나 멀티파트 업로드가 중단 또는 완료되었을 수 있습니다.	404 Not Found	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
NoSuchVersion	요청에 지정된 버전 ID가 기존 버전과 일치하지 않습니다.	404 Not Found	클라이언트
NotDeviceOwnerError	토큰을 생성한 디바이스가 인증된 사용자의 소유가 아닙니다.	400 Bad Request	클라이언트
NotImplemented	제공한 헤더가 구현되지 않은 기능을 암시합니다.	501 Not Implemented	Server
NotModified	리소스가 변경되지 않았습니다.	304 Not Modified	Server
NoTransformationDefined	이 객체 Lambda 액세스 포인트에 대한 변환이 없습니다.	404 Not Found	클라이언트
NotSignedUp	계정이 Amazon S3 서비스에 가입되지 않았습니다. Amazon S3를 사용하려면 가입해야 합니다. 다음 URL에서 가입할 수 있습니다. https://aws.amazon.com/s3	403 Forbidden	클라이언트
ObjectLockConfigurationNotFoundError	이 버킷에 대한 객체 잠금 구성이 존재하지 않습니다.	404 Not Found	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
OwnershipControlsNotFound	버킷 소유권 제어를 찾을 수 없습니다.	404 Not Found	클라이언트
OperationAborted	현재 이 리소스에 대해 서로 충돌하는 조건부 작업이 진행 중입니다. 다시 시도하세요.	409 Conflict	클라이언트
PermanentRedirect	액세스하려는 버킷이 지정된 엔드포인트를 사용하여 처리되어야 합니다. 이 엔드포인트로 모든 향후 요청을 보냅니다.	301 Moved Permanently	클라이언트
PermanentRedirectControlError	액세스하려는 API 작업이 지정된 엔드포인트를 사용하여 처리되어야 합니다. 이 엔드포인트로 모든 향후 요청을 보냅니다.	301 Moved Permanently	클라이언트
PreconditionFailed	지정한 사전 조건 중 하나 이상이 충족되지 않았습니다.	412 Precondition Failed	클라이언트
Redirect	임시 리디렉션. 도메인 이름 시스템 (DNS) 서버가 업데이트되는 동안 버킷으로 리디렉션됩니다.	307 Temporary Redirect	클라이언트
RequestHeaderSectionTooLarge	요청을 만드는 데 사용된 요청 헤더 및 쿼리 파라미터가 최대 허용 크기를 초과합니다.	400 Bad Request	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
RequestIsNotMultiPartContent	버킷 POST 요청은 엔클로저 유형의 멀티파트/양식 데이터여야 합니다.	412 Precondition Failed	클라이언트
RequestTimeout	서버에 대한 소켓 연결에서 제한 시간 내에 읽기 또는 쓰기가 수행되지 않았습니다.	400 Bad Request	클라이언트
RequestTimeTooSkewed	요청 시간과 서버 시간 간의 차이가 너무 큼니다.	403 Forbidden	클라이언트
RequestTorrentOfBucketError	버킷의 Torrent 파일 요청은 허용되지 않습니다.	400 Bad Request	클라이언트
ResponseInterrupted	WriteGetObjectResponse 본문을 읽는 동안 오류가 발생하여 원래 호출자에게 반환되었습니다.	400 Bad Request	클라이언트
RestoreAlreadyInProgress	객체 복원이 이미 진행 중입니다.	409 Conflict	클라이언트
ServerSideEncryptionConfigurationNotFound	서버 측 암호화 구성을 찾을 수 없습니다.	400 Bad Request	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
ServiceUnavailable	서비스가 요청을 처리할 수 없습니다.	503 Service Unavailable	Server
SignatureDoesNotMatch	서버가 계산한 요청 서명이 사용자가 제공한 서명과 일치하지 않습니다. AWS 비밀 액세스 키와 서명 방법을 확인합니다. 자세한 내용은 REST 인증 과 SOAP 인증 을 확인하세요.	403 Forbidden	클라이언트
SlowDown	요청 속도를 낮추세요.	503 Slow Down	Server
503 SlowDown	속도 저하	503 Slow Down	Server
TagPolicyException	태그 정책이 다음 태그 키에 지정된 값을 허용하지 않습니다.	400 Bad Request	클라이언트
TemporaryRedirect	도메인 이름 시스템(DNS) 서버가 업데이트되는 동안 버킷으로 리디렉션됩니다.	307 Temporary Redirect	클라이언트
TokenCodeInvalidError	제공한 일련 번호 및/또는 토큰 코드가 유효하지 않습니다.	400 Bad Request	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
TokenRefreshRequired	제공된 토큰을 새로 고쳐야 합니다.	400 Bad Request	클라이언트
TooManyAccessPoints	단일 계정에 허용된 것보다 많은 액세스 포인트를 생성하려고 시도했습니다. 자세한 내용은 AWS 일반 참조의 Amazon Simple Storage Service 엔드포인트 및 할당량 을 참조하세요.	400 Bad Request	클라이언트
TooManyBuckets	단일 계정에 허용된 것보다 많은 버킷을 생성하려고 했습니다. 자세한 내용은 AWS 일반 참조의 Amazon Simple Storage Service 엔드포인트 및 할당량 을 참조하세요.	400 Bad Request	클라이언트
TooManyMultiRegionAccessPointregionsError	단일 계정에 허용된 것보다 많은 리전을 포함하는 다중 리전 액세스 포인트를 생성하려고 시도했습니다. 자세한 내용은 AWS 일반 참조의 Amazon Simple Storage Service 엔드포인트 및 할당량 을 참조하세요.	400 Bad Request	클라이언트
TooManyMultiRegionAccessPoints	단일 계정에 허용된 것보다 많은 다중 리전 액세스 포인트를 생성하려고 시도했습니다. 자세한 내용은 AWS 일반 참조의 Amazon Simple Storage Service 엔드포인트 및 할당량 을 참조하세요.	400 Bad Request	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
UnauthorizedAccessError	중국 지역에만 적용됩니다. ICP 라이선스가 없는 버킷에 요청을 보냈을 때 반환됩니다. 자세한 내용은 ICP 기록 을 참조하세요.	403 Forbidden	클라이언트
UnexpectedContent	이 요청에는 지원되지 않는 콘텐츠가 포함되어 있습니다.	400 Bad Request	클라이언트
UnexpectedIPError	중국 지역에만 적용됩니다. 예기치 않은 IP로 인해 요청이 거부되었습니다.	403 Forbidden	클라이언트
UnsupportedArgument	요청에 지원되지 않는 인수가 포함되어 있습니다.	400 Bad Request	클라이언트
UnsupportedSignature	제공된 요청이 지원되지 않는 STS 토큰 버전으로 서명되었거나 해당 서명 버전이 지원되지 않습니다.	400 Bad Request	클라이언트
UnresolvableGrantByEmailAddress	제공한 이메일 주소가 기록된 계정과 일치하지 않습니다.	400 Bad Request	클라이언트
UserKeyMustBeSpecified	버킷 POST 요청에는 지정된 필드 이름이 포함되어야 합니다. 필드 이름이 지정되었다면 필드의 순서를 확인합니다.	400 Bad Request	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
NoSuchAccessPoint	지정된 액세스 포인트가 존재하지 않습니다.	404 Not Found	클라이언트
InvalidTag	요청에 유효하지 않은 태그 입력이 포함되어 있습니다. 예를 들어 요청에 중복된 키, 너무 긴 키 또는 값, 시스템 태그가 포함되어 있을 수 있습니다.	400 Bad Request	클라이언트
MalformedPolicy	정책에 유효하지 않은 위탁자가 포함되어 있습니다.	400 Bad Request	클라이언트

SELECT 객체 콘텐츠 오류 코드 목록

Important

신규 고객은 더 이상 Amazon S3 Select를 사용할 수 없습니다. Amazon S3 Select 기존 고객은 평소처럼 이 기능을 계속 사용할 수 있습니다. [자세히 알아보기](#)

다음 표에는 SELECT Object Content가 반환할 수 있는 특수 오류가 나와 있습니다. 일반적인 Amazon S3 오류에 대한 자세한 내용 및 오류 코드 목록은 [오류 응답](#) 섹션을 참조하세요.

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
AmbiguousFieldName	필드 이름이 파일의 여러 필드와 일치합니다. SQL 표현식 및 파일을 확인하고 다시 시도하세요.	400	클라이언트
Busy	이 서비스를 사용할 수 없습니다. 나중에 다시 시도해 주세요.	503	클라이언트
CastFailed	CAST를 사용하여 한 데이터 유형에서 다른 데이터 유형으로 변환하려는 시도가 SQL 표현식에서 실패했습니다.	400	클라이언트
ColumnTooLong	결과에 있는 열의 길이가 maxCharsPerColumn 1MB보다 큼니다.	400	클라이언트
CSVEscapingRecordDelimiter	파일에서 따옴표로 묶인 레코드 구분 기호가 발견되었습니다. 따옴표로 묶인 레코드 구분 기호를 허용하려면 AllowQuotedRecordDelimiter를 'TRUE'로 설정하세요.	400	클라이언트
CSVParsingError	CSV 파일을 구문 분석하는 동안 오류가 발생했습니다. 파일을 확인하고 다시 시도하세요.	400	클라이언트
CSVUnescapedQuote	CSV 파일을 구문 분석하는 동안 이스케이프 처리되지 않은 따옴표가 발견되었습니다. 따옴표로 묶인 레코드 구분 기호를 허용하려면 AllowQuotedRecordDelimiter를 'TRUE'로 설정하세요.	400	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
EmptyRequestBody	요청 본문은 비어 있을 수 없습니다.	400	클라이언트
EvaluatorBindingDoesNotExist	제공된 열 이름 또는 경로가 SQL 표현식에 없습니다.	400	클라이언트
EvaluatorInvalidArguments	SQL 표현식의 함수 직접 호출에 잘못된 수의 인수가 있습니다.	400	클라이언트
EvaluatorInvalidTimestampFormatPattern	SQL 표현식의 타임스탬프 형식 문자열이 유효하지 않습니다.	400	클라이언트
EvaluatorInvalidTimestampFormatPatternSymbol	타임스탬프 형식 패턴에 SQL 표현식에는 유효하지 않은 기호가 포함되어 있습니다.	400	클라이언트
EvaluatorInvalidTimestampFormatPatternSymbolForParsing	타임스탬프 형식 패턴에 SQL 표현식의 타임스탬프 구문 분석에 적용할 수 없는 유효한 형식 기호가 포함되어 있습니다.	400	클라이언트
EvaluatorInvalidTimestampFormatPatternToken	타임스탬프 형식 패턴에 SQL 표현식에는 유효하지 않은 토큰이 포함되어 있습니다.	400	클라이언트
EvaluatorLikePatternInvalidEscapeSequence	LIKE 표현식에 지정된 인수가 유효하지 않습니다.	400	클라이언트
EvaluatorNegativeLimit	LIMIT는 음수가 아니어야 합니다.	400	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
EvaluatorTimestampFormatPatternDuplicateFields	타임스탬프 형식 패턴에 SQL 표현식의 타임스탬프 필드를 나타내는 여러 형식 지정자가 포함되어 있습니다.	400	클라이언트
EvaluatorTimestampFormatPatternHourClockAmPmMismatch	타임스탬프 형식 패턴에는 12시간 형식 기호가 포함되어 있지만 AM/PM 필드가 포함되어 있지 않거나, 24시간 형식 지정자가 포함되어 있지만 SQL 표현식에 AM/PM 필드가 포함되어 있습니다.	400	클라이언트
EvaluatorUnterminatedTimestampFormatPatternToken	타임스탬프 형식 패턴이 SQL 표현식에 종료되지 않은 토큰을 포함합니다.	400	클라이언트
ExpressionTooLong	SQL 표현식이 너무 깁니다. SQL 표현식의 최대 길이는 256KB입니다.	400	클라이언트
ExternalEvalException	쿼리를 평가할 수 없습니다. 파일을 확인하고 다시 시도하세요.	400	클라이언트
IllegalSqlFunctionArgument	SQL 함수에 잘못된 인수가 사용되었습니다.	400	클라이언트
IncorrectSqlFunctionArgumentType	SQL 표현식의 함수 직접 호출에 잘못된 인수 유형이 지정되었습니다.	400	클라이언트
IntegerOverflow	SQL 표현식에서 정수 오버플로 또는 언더플로가 발생했습니다.	400	클라이언트
InternalError	내부 오류가 발생했습니다.	500	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
InvalidCast	CAST를 사용하여 한 데이터 유형에서 다른 데이터 유형으로 변환하려는 시도가 SQL 표현식에서 실패했습니다.	400	클라이언트
InvalidColumnIndex	SQL 표현식의 열 인덱스가 유효하지 않습니다.	400	클라이언트
InvalidCompressionFormat	파일이 지원되는 압축 형식이 아닙니다. GZIP 및 BZIP2만 지원됩니다.	400	클라이언트
InvalidDataSource	데이터 소스 유형이 유효하지 않습니다. CSV, JSON 및 Parquet만 지원됩니다.	400	클라이언트
InvalidDataType	SQL 표현식에 유효하지 않은 데이터 형식이 포함되어 있습니다.	400	클라이언트
InvalidExpressionType	ExpressionType 값이 유효하지 않습니다. SQL 표현식만 지원됩니다.	400	클라이언트
InvalidFileHeaderInfo	FileHeaderInfo 값이 유효하지 않습니다. NONE, USE 및 IGNORE만 지원됩니다.	400	클라이언트
InvalidJsonType	JsonType 값이 유효하지 않습니다. DOCUMENT 및 LINES만 지원됩니다.	400	클라이언트
InvalidKeyPath	SQL 표현식의 키 경로가 유효하지 않습니다.	400	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
InvalidQuoteFields	QuoteFields 값이 유효하지 않습니다. ALWAYS 및 ASNEEDED만 지원됩니다.	400	클라이언트
InvalidRequestParameter	SelectRequest 요소의 파라미터 값이 유효하지 않습니다. 서비스 API 설명서를 확인하고 다시 시도하세요.	400	클라이언트
InvalidScanRange	제공된 스캔 범위가 유효하지 않습니다.	400	클라이언트
InvalidTableAlias	SQL 표현식에 유효하지 않은 테이블 별칭이 포함되어 있습니다.	400	클라이언트
InvalidTextEncoding	인코딩 유형이 유효하지 않습니다. UTF-8 인코딩만 지원됩니다.	400	클라이언트
JSONParsingError	JSON 파일을 구문 분석하는 동안 오류가 발생했습니다. 파일을 확인하고 다시 시도하세요.	400	클라이언트
LexerInvalidChar	SQL 표현식에 유효하지 않은 문자가 포함되어 있습니다.	400	클라이언트
LexerInvalidIONLiteral	SQL 표현식에 유효하지 않은 연산자가 포함되어 있습니다.	400	클라이언트
LexerInvalidLiteral	SQL 표현식에 유효하지 않은 연산자가 포함되어 있습니다.	400	클라이언트
LexerInvalidOperator	SQL 표현식에 유효하지 않은 리터럴이 포함되어 있습니다.	400	클라이언트
LikeInvalidInputs	SQL 표현식의 LIKE 절에 제공된 인수가 유효하지 않습니다.	400	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
MalformedXML	제공된 XML이 잘못된 형식이거나 게시된 스키마를 기준으로 검증되지 않았습니다. 서비스 설명서를 확인하고 다시 시도하세요.	400	클라이언트
MaxOperatorsExceeded	SQL 표현식을 구문 분석하지 못했습니다. 복잡성을 낮추세요. 예를 들어 사용되는 연산자 수를 줄입니다.	400	클라이언트
MethodNotAllowed	지정된 메서드가 이 리소스에 허용되지 않습니다.	405 Method Not Allowed	클라이언트
MissingRequiredParameter	SelectRequest 엔터티에 필수 파라미터가 없습니다. 서비스 설명서를 확인하고 다시 시도하세요.	400	클라이언트
MultipleDataSourcesUnsupported	여러 데이터 소스는 지원되지 않습니다.	400	클라이언트
NumberFormatError	숫자를 구문 분석하는 동안 오류가 발생했습니다. 이 오류는 정수의 언더플로 또는 오버플로로 인해 발생할 수 있습니다.	400	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
ObjectSerializationConflict	InputSerialization 이 둘 이상의 형식(CSV, JSON 또는 Parquet)을 지정하거나 OutputSerialization 이 둘 이상의 형식(CSV 또는 JSON)을 지정합니다. InputSerialization 및 OutputSerialization 의 경우 각각 하나의 형식만 지정할 수 있습니다.	400	클라이언트
OverMaxColumn	결과의 열 수가 허용되는 최대 열 수보다 많습니다.	400	클라이언트
OverMaxParquetBlockSize	Parquet 파일이 최대 행 그룹 크기를 초과합니다.	400	클라이언트
OverMaxRecordSize	입력 또는 결과에 있는 레코드의 길이가 1MB의 maxCharsPerRecord 제한보다 깁니다.	400	클라이언트
ParquetParsingError	Parquet 파일을 구문 분석하는 동안 오류가 발생했습니다. 파일을 확인하고 다시 시도하세요.	400	클라이언트
ParquetUnsupportedCompressionCodec	지정된 Parquet 압축 코덱은 지원되지 않습니다.	400	클라이언트
ParseAsteriskIsNotAloneInSelectList	SQL 표현식에서 점 표기법 없이 *를 사용하는 경우 SELECT 목록에 다른 표현식이 허용되지 않습니다.	400	클라이언트
ParseCannotMixSqbandWildcardInSelectList	SQL 표현식의 SELECT 목록에 있는 동일한 표현식에서 [] 및 *를 혼용할 수 없습니다.	400	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
ParseCastArity	SQL 표현식 CAST의 인수 개수가 잘못되었습니다.	400	클라이언트
ParseEmptySelect	SQL 표현식에 빈 SELECT 절이 포함되어 있습니다.	400	클라이언트
ParseExpected2TokenTypes	SQL 표현식에서 예상되는 토큰을 찾을 수 없습니다.	400	클라이언트
ParseExpectedArgumentDelimiter	SQL 표현식에서 예상되는 인수 구분 기호를 찾을 수 없습니다.	400	클라이언트
ParseExpectedDatePart	SQL 표현식에서 예상되는 날짜 부분을 찾을 수 없습니다.	400	클라이언트
ParseExpectedExpression	예상되는 SQL 표현식을 찾을 수 없습니다.	400	클라이언트
ParseExpectedIdentifierForAlias	SQL 표현식에서 별칭에 예상되는 식별자를 찾을 수 없습니다.	400	클라이언트
ParseExpectedIdentifierForAt	SQL 표현식에서 AT 이름에 예상되는 식별자를 찾을 수 없습니다.	400	클라이언트
ParseExpectedIdentifierForGroupName	GROUP은 SQL 정규식에서 지원되지 않습니다.	400	클라이언트
ParseExpectedKeyword	SQL 표현식에서 예상되는 키워드를 찾을 수 없습니다.	400	클라이언트
ParseExpectedLeftParenAfterCast	SQL 표현식에서 CAST 이후 예상되는 왼쪽 괄호를 찾을 수 없습니다.	400	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
ParseExpectedLeftParenBuiltinFunctionCall	SQL 표현식에서 예상되는 왼쪽 괄호를 찾을 수 없습니다.	400	클라이언트
ParseExpectedLeftParenValueConstructor	SQL 표현식에서 예상되는 왼쪽 괄호를 찾을 수 없습니다.	400	클라이언트
ParseExpectedMember	SQL 표현식에 지원되지 않는 MEMBER 사용이 포함되어 있습니다.	400	클라이언트
ParseExpectedNumber	SQL 표현식에서 예상되는 숫자를 찾을 수 없습니다.	400	클라이언트
ParseExpectedRightParenBuiltinFunctionCall	SQL 표현식에서 예상되는 오른쪽 괄호 문자를 찾을 수 없습니다.	400	클라이언트
ParseExpectedTokenType	SQL 표현식에서 예상되는 토큰을 찾을 수 없습니다.	400	클라이언트
ParseExpectedTypeName	SQL 표현식에서 예상되는 유형 이름을 찾을 수 없습니다.	400	클라이언트
ParseExpectedWhenClause	SQL 표현식에서 예상되는 WHEN 절을 찾을 수 없습니다. CASE는 지원되지 않습니다.	400	클라이언트
ParseInvalidContextForWildcardInSelectList	SQL 표현식의 SELECT 목록에서 *를 사용하는 것은 유효하지 않습니다.	400	클라이언트
ParseInvalidPathComponent	SQL 표현식에 유효하지 않은 경로 구성 요소가 포함되어 있습니다.	400	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
ParseInvalidTypeParam	SQL 표현식에 유효하지 않은 파라미터 값이 포함되어 있습니다.	400	클라이언트
ParseMalformedJoin	JOIN은 SQL 정규식에서 지원되지 않습니다.	400	클라이언트
ParseMissingIdentifierAt	SQL 표현식에서 @ 기호 뒤에 예상되는 식별자를 찾을 수 없습니다.	400	클라이언트
ParseNonUnaryAggregateFunctionCall	SQL 표현식의 집계 함수에는 하나의 인수만 지원됩니다.	400	클라이언트
ParseSelectMissingFrom	SQL 표현식에서 SELECT 목록 뒤에 FROM이 누락되어 있습니다.	400	클라이언트
ParseUnexpectedKeyword	SQL 표현식에 예상치 못한 키워드가 포함되어 있습니다.	400	클라이언트
ParseUnexpectedOperator	SQL 표현식에 예상치 못한 연산자가 포함되어 있습니다.	400	클라이언트
ParseUnexpectedTerm	SQL 표현식에 예상치 못한 항목이 포함되어 있습니다.	400	클라이언트
ParseUnexpectedToken	SQL 표현식에 예상치 못한 토큰이 포함되어 있습니다.	400	클라이언트
ParseUnknownOperator	SQL 표현식에 유효하지 않은 연산자가 포함되어 있습니다.	400	클라이언트
ParseUnsupportedAlias	SQL 표현식에 지원되지 않는 ALIAS 사용이 포함되어 있습니다.	400	클라이언트
ParseUnsupportedCallWithStar	SQL 표현식에는 (*)를 파라미터로 사용하는 COUNT만 지원됩니다.	400	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
ParseUnsupportedCase	SQL 표현식에 지원되지 않는 CASE 사용이 포함되어 있습니다.	400	클라이언트
ParseUnsupportedCaseClause	SQL 표현식에 지원되지 않는 CASE 사용이 포함되어 있습니다.	400	클라이언트
ParseUnsupportedLiteralsGroupBy	SQL 표현식에 지원되지 않는 GROUP BY 사용이 포함되어 있습니다.	400	클라이언트
ParseUnsupportedSelect	SQL 표현식에 지원되지 않는 SELECT 사용이 포함되어 있습니다.	400	클라이언트
ParseUnsupportedSyntax	SQL 표현식에 지원되지 않는 구문이 포함되어 있습니다.	400	클라이언트
ParseUnsupportedToken	SQL 표현식에 지원되지 않는 토큰이 포함되어 있습니다.	400	클라이언트
TruncatedInput	객체 압축 해제가 실패했습니다. 객체가 요청에 지정된 형식을 사용하여 제대로 압축되었는지 확인하세요.	400	클라이언트
UnauthorizedAccess	이 작업을 수행할 수 있는 권한이 없습니다.	401	클라이언트
UnrecognizedFormatException	유효하지 않은 레코드 유형이 발견되었습니다.	400	클라이언트
UnsupportedFunction	지원되지 않는 SQL 함수가 발견되었습니다.	400	클라이언트
UnsupportedParquetType	지정된 Parquet 유형은 지원되지 않습니다.	400	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
UnsupportedRangeHeader	이 작업에는 범위 헤더가 지원되지 않습니다.	400	클라이언트
UnsupportedScanRangeInput	이 유형의 객체에는 스캔 범위 쿼리가 지원되지 않습니다.	400	클라이언트
UnsupportedSqlOperation	지원되지 않는 SQL 작업이 발견되었습니다.	400	클라이언트
UnsupportedSqlStructure	지원되지 않는 SQL 구조가 발견되었습니다. SQL 참조를 확인하세요.	400	클라이언트
UnsupportedStorageClass	지원되지 않는 스토리지 클래스가 발견되었습니다. STANDARD, STANDARD_IA 및 ONEZONE_IA 스토리지 클래스만 지원됩니다.	400	클라이언트
UnsupportedSyntax	유효하지 않은 구문이 발견되었습니다.	400	클라이언트
UnsupportedTypeForQuerying	쿼리에 지원되지 않는 비교 유형이 포함되어 있습니다(예: Parquet INT96 열 유형이 0보다 큰지 확인).	400	클라이언트
ValueParseFailure	SQL 표현식에서 타임스탬프 구문 분석 실패가 발생했습니다.	400	클라이언트

복제 관련 오류 코드 목록

다음 표에는 Replication 작업이 반환할 수 있는 특수 오류가 나와 있습니다. 일반적인 Amazon S3 오류에 대한 자세한 내용 및 오류 코드 목록은 [오류 응답](#) 섹션을 참조하세요.

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
InvalidArgument	이 오류는 다음과 같은 이유로 발생할 수 있습니다. • <code><Account></code> 요소가 비어 있습니다. 유효한 계정 ID가 포함되어야 합니다. • <code><Account></code> 요소에 지정된 AWS 계정은 대상 버킷 소유자와 일치해야 합니다. • <code>ReplicationTime-Status</code>에는 값이 포함되어야 합니다. • <code>ReplicationTime-ReplicationTimeValue</code>에는 값이 포함되어야 합니다. • <code>Replication-ReplicationTimeValue-Minutes</code> 값은 15여야 합니다. • <code>ReplicationMetrics</code>는 <code>Status</code>를 포함해야 합니다. • <code>ReplicationMetrics</code>는 <code>EventThreshold</code>를 포함해야 합니다. • <code>EventThreshold-ReplicationTimeValue-Minutes</code> 값은 15여야 합니다.	400	클라이언트

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
	Rule ID에는 비ASCII 문자가 포함되면 안 됩니다.		
InvalidRequest	이 오류는 다음과 같은 이유로 발생할 수 있습니다. <AccessControlTranslation> 의 <Owner>에 값이 있으므로 <Account> 요소가 지정되어야 합니다. <Account> 요소가 비어 있습니다. 유효한 계정 ID가 포함되어야 합니다. - 복제 destination에는 ReplicationTime 및 Metrics가 모두 포함되거나 모두 포함되지 않아야 합니다. - ReplicationTime 과 ReplicationMetrics 는 동일한 상태여야 합니다. - 이 AWS 리전에는 S3 Replication Time Control(S3 RTC)이 지원되지 않습니다.	400	클라이언트
ReplicationConfigurationNotFoundError	이 버킷에는 복제 구성이 없습니다.	404 Not Found(404 찾을 수 없음)	클라이언트

태그 지정 관련 오류 코드 목록

다음 표에는 Storage Lens 그룹에 대해 TagResource, UntagResource 및 ListTagsForResource 작업이 반환할 수 있는 특수 오류가 나와 있습니다. Amazon S3 오류에 대한 일반적인 내용 및 오류 코드 목록은 [오류 응답](#) 섹션을 참조하세요.

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
InvalidRequest	리소스 ARN의 AWS 리전이 이 요청에 지정된 리전과 일치하지 않습니다. 리소스 ARN의 AWS 계정이 이 요청에 지정된 계정 ID와 일치하지 않습니다. resourceArn의 AWS 파티션이 유효하지 않습니다.	400 Bad Request	지원되지 않음
InvalidTag	이 요청에 유효하지 않은 태그 키 또는 값이 포함되어 있습니다. 유효한 문자에는 [a-zA-Z+-=._:/] 가 포함됩니다. 태그 키는 최대 128자입니다. 태그 값은 최대 256자입니다. 요청에 중복된 태그 키가 있습니다. 사용자 정의 태그 키는 aws:로 시작할 수 없습니다.	400 Bad Request	지원되지 않음
NoSuchResource	지정된 리소스가 존재하지 않습니다.	404 Not Found	지원되지 않음
TagPolicyException	태그 정책이 다음 태그 키에 지정된 값을 허용하지 않습니다.	400 Bad Request	지원되지 않음
TooManyTags	태그 수가 태그 제한 50개를 초과합니다.	400 Bad Request	지원되지 않음

Amazon S3 주석 관련 오류 코드 목록

다음 표에는 주석 작업(PutObjectAnnotation, GetObjectAnnotation, ListObjectAnnotations, DeleteObjectAnnotation, UpdateBucketMetadataAnnotationTableConfiguration) 및 주석 관련 CopyObject 동작이 반환할 수 있는 특수 오류가 나와 있습니다. 일반적인 Amazon S3 오류에 대한 자세한 내용 및 오류 코드 목록은 [오류 응답](#) 섹션을 참조하세요.

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
AccessDenied	액세스가 거부되었습니다. 호출자에게 요청된 주석 작업을 수행할 권한이 없거나 객체가 COMPLIANCE 모드에서 Object Lock으로 보호됩니다.	403 Forbidden	지원되지 않음
AnnotationLimitExceeded	객체 주석 수가 제한(1,000)을 초과합니다.	400 Bad Request	지원되지 않음
AnnotationNameTooLong	주석 이름이 너무 깁니다. 주석 이름은 길이가 최대 512바이트까지 가능합니다.	400 Bad Request	지원되지 않음
AnnotationRequestConflict	현재 이 주석에 대해 서로 충돌하는 작업이 진행 중입니다. 이는 고차리전 복사 중에 소스 주석이 삭제될 때 발생할 수 있습니다.	409 Conflict	지원되지 않음
EntityTooSmall	제안된 업로드가 최소 허용 크기보다 작습니다. 주석 페이로드는 1바이트 이상이어야 합니다.	400 Bad Request	지원되지 않음
InvalidAnnotationName	주석 이름에 잘못된 문자가 포함되어 있거나 이름이 예약된 접두사(aws 또는 s3, 대/소문자를 구분하지 않음)로 시작합니다. 주석 이름에는 유니코드 문자, 숫자, 밑줄(_),	400 Bad Request	지원되지 않음

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
InvalidArgument	마침표(.), 하이픈(-)만 포함될 수 있습니다. 주석 지시문이 유효하지 않습니다. 유효 값은 COPY 및 EXCLUDE입니다. ListObjectAnnotations 의 경우 max-annotation-results 는 1에서 1000 사이여야 합니다.	400 Bad Request	지원되지 않음
InvalidPrefix	제공한 주석 접두사가 유효하지 않습니다.	400 Bad Request	지원되지 않음
InvalidRequest	이 오류는 다음과 같은 이유로 발생할 수 있습니다. 객체가 주석을 지원하지 않는 SSE-C로 암호화되어 있습니다. 소스 객체에 대상이 지원되지 않는 리전에 있기 때문에 대상에 복사할 수 없는 주석이 있습니다. 주석 테이블 구성 요청이 유효하지 않습니다(예: 암호화를 지정하는 동안 테이블 비활성화, 이미 활성화된 테이블 활성화, CREATING 상태의 테이블 업데이트 또는 빈 요청 제출).	400 Bad Request	지원되지 않음
MethodNotAllowed	지정된 메서드가 이 리소스에 허용되지 않습니다. 예를 들어, 이 오류는 요청이 삭제 마커를 대상으로 할 때 발생합니다.	405 Method Not Allowed	지원되지 않음
NoSuchAnnotation	지정된 주석이 없습니다.	404 Not Found	지원되지 않음

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
NoSuchKey	지정된 키가 없습니다.	404 Not Found	지원되지 않음
NoSuchVersion	요청에 지정된 버전 ID가 기존 버전과 일치하지 않습니다.	404 Not Found	지원되지 않음
NotImplemented	Amazon S3 디렉터리 버킷에는 주석이 지원되지 않습니다.	501 Not Implemented	지원되지 않음
OperationAborted	현재 이 리소스에 대해 서로 충돌하는 조건부 작업이 진행 중입니다. 이는 동시 수정 중에 또는 교차리전 복사 중에 소스 주석 수 또는 ETag가 변경될 때 발생할 수 있습니다. 다시 시도하세요.	409 Conflict	지원되지 않음
PreconditionFailed	지정한 사전 조건 중 하나 이상이 충족되지 않았습니다. x-amz-object-if-match ETag가 현재 객체 ETag와 일치하지 않습니다.	412 Precondition Failed	지원되지 않음
UnsupportedMediaType	요청 페이로드에 지원되지 않는 미디어 유형이 포함되어 있습니다. 주석 페이로드는 UTF-8로 인코딩된 유효한 텍스트여야 합니다.	415 Unsupported Media Type	지원되지 않음
UserAnnotationNameMustBeSpecified	사용자 주석 이름을 지정해야 합니다.	400 Bad Request	지원되지 않음

Amazon S3 on Outposts 오류 코드 목록

다음 표에는 Amazon S3 on Outposts 작업이 반환할 수 있는 특수 오류가 나와 있습니다. 일반적인 Amazon S3 오류에 대한 자세한 내용 및 오류 코드 목록은 [오류 응답](#) 섹션을 참조하세요.

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
BadRequest	이전 삭제 시도로 인해 버킷이 전환 상태입니다. 나중에 다시 시도해 주세요.	400 Bad Request	지원되지 않음
InvalidRequest	이 오류는 다음과 같은 이유로 발생할 수 있습니다. - Amazon VPC 구성이 필요합니다. - S3 on Outposts 액세스 포인트에서는 퍼블릭 액세스가 허용되지 않습니다.	400 Bad Request	클라이언트
InvalidOutpostState	요청이 Outpost의 현재 상태에 대해 유효하지 않습니다.	409 충돌	지원되지 않음
InvalidRequest	액세스 포인트가 삭제할 수 있는 상태가 아닙니다.	400 Bad Request	지원되지 않음
NoSuchOutpost	지정된 Outpost가 존재하지 않습니다.	404 Not Found(404 찾을 수 없음)	지원되지 않음
UnsupportedOperation	지정된 작업이 지원되지 않습니다.	404 Not Found(404 찾을 수 없음)	지원되지 않음
InsufficientCapacity	용량 부족.	507 스토리지 부족	지원되지 않음

Amazon S3 Storage Lens 오류 코드 목록

다음 표에는 Amazon S3 Storage Lens 작업에서 반환할 수 있는 특수 오류가 나와 있습니다. Amazon S3 오류에 대한 일반적인 내용 및 오류 코드 목록은 [오류 응답](#) 섹션을 참조하세요.

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
AccessDenied	이 리전은 S3 Storage Lens의 홈 리전으로 지원되지 않습니다.	403 Forbidden	지원되지 않음
AccountNotAuthorized	이 계정은 AWS Organizations를 사용할 권한이 없습니다. 관리 계정 또는 위임된 관리자 계정을 사용하세요.	403 Forbidden	지원되지 않음
ActivityMetricsMustEnabled	활동 지표를 활성화해야 합니다.	400 Bad Request	지원되지 않음
AWSOrganizationsNotInUseException	이 계정은 조직의 일부가 아닙니다.	403 Forbidden	지원되지 않음
DefaultConfigurationDeleteForbidden	기본 구성은 삭제할 수 없습니다.	403 Forbidden	지원되지 않음
DuplicateStorageLensGroupARN	이 구성에는 Storage Lens 그룹 ARN이 동일한 항목이 두 개 이상 있습니다.	400 Bad Request	지원되지 않음
EmptyExcludeContainer	이 오류는 다음과 같은 이유로 발생할 수 있습니다. - 제외 컨테이너는 비어 있을 수 없습니다.	400 Bad Request	지원되지 않음

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
	제외 컨테이너에는 버킷이 0개일 수 없습니다. 제외 컨테이너에는 리전이 0개일 수 없습니다.		
EmptyExcludeElement	Storage Lens 그룹을 제외 요소와 함께 지정해야 합니다.	400 Bad Request	지원되지 않음
EmptyIncludeContainer	이 오류는 다음과 같은 이유로 발생할 수 있습니다. - 포함 컨테이너는 비어 있을 수 없습니다. - 포함 컨테이너에는 버킷이 0개일 수 없습니다. - 포함 컨테이너에는 리전이 0개일 수 없습니다.	400 Bad Request	지원되지 않음
InvalidAWSOrgArn	구성에 잘못된 형식의 AWS Organizations ARN이 있습니다.	400 Bad Request	지원되지 않음
EmptyIncludeElement	Storage Lens 그룹을 포함 요소와 함께 지정해야 합니다.	400 Bad Request	지원되지 않음
InvalidBucketFilter	조직 수준 구성은 버킷 필터를 지원하지 않습니다.	400 Bad Request	지원되지 않음
InvalidConfigId	구성 ID가 유효하지 않습니다.	400 Bad Request	지원되지 않음
InvalidDestination	S3 버킷 ARN이 잘못된 형식입니다.	400 Bad Request	지원되지 않음

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
InvalidEncryptionMethod	암호화 방법을 하나만 지정할 수 있습니다.	400 Bad Request	지원되지 않음
InvalidFilterForDefaultConfiguration	기본 구성에는 필터가 포함되면 안 됩니다.	400 Bad Request	지원되지 않음
InvalidIncludeExcludeContainers	구성에서 포함 컨테이너 또는 제외 컨테이너를 지정할 수 있습니다. 한 구성에서 둘 다 지정할 수는 없습니다.	400 Bad Request	지원되지 않음
InvalidIncludeExcludeElements	포함 또는 제외 요소는 하나만 허용됩니다. 포함 또는 제외 요소가 하나 이상 있어야 합니다.	400 Bad Request	지원되지 않음
InvalidKMSEncryptionKeyId	KMS 키 ID ARN이 유효하지 않습니다.	400 Bad Request	지원되지 않음
InvalidMaximumPrefixDepth	MaxDepth는 [1,10] 범위 내에 있어야 합니다.	400 Bad Request	지원되지 않음
InvalidMinimumStorageBytesPercentage	MinStorageBytesPercentage 는 [1.00,100.00] 범위 내에 있어야 합니다.	400 Bad Request	지원되지 않음
InvalidOrganizationARN	구성이 AWS Organizations ARN이 유효하지 않습니다.	400 Bad Request	지원되지 않음
InvalidOrganizationForDefaultConfiguration	기본 구성은 조직 수준 지표를 지원하지 않습니다.	400 Bad Request	지원되지 않음
InvalidRegionForDefaultConfiguration	지정된 리전은 기본 구성에서 지원되지 않습니다.	400 Bad Request	지원되지 않음

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
InvalidRegionName	리전 이름이 유효하지 않습니다.	400 Bad Request	지원되지 않음
InvalidStorageLensArn	S3 Storage Lens ARN은 입력에 필요하지 않습니다.	400 Bad Request	지원되지 않음
InvalidStorageLensGroupARN	이 Storage Lens 그룹 ARN이 유효하지 않거나 계정의 Storage Lens 그룹만 허용됩니다. 또한 Storage Lens 그룹 ARN 구조 <code>arn::s3:::storage-lens-group/</code> 을 따르고 64자 제한을 준수해야 합니다. Storage Lens 그룹 이름에는 <code>a~z</code> , <code>A~Z</code> , <code>0~9</code> , 하이픈(-) 및 밑줄(_)만 포함될 수 있습니다.	400 Bad Request	지원되지 않음
MissingAccountLevelActivityMetrics	활동 지표가 버킷 수준에서 활성화된 경우 활동 지표를 계정 수준에서 활성화해야 합니다.	400 Bad Request	지원되지 않음
MissingBucketLevelActivityMetrics	활동 지표가 계정 수준에서 활성화된 경우 활동 지표를 버킷 수준에서 활성화해야 합니다.	400 Bad Request	지원되지 않음
MissingEncryptionMethod	암호화 방법은 비어 있을 수 없습니다. 또는 SSE-KMS를 지정합니다..SSE-S3	400 Bad Request	지원되지 않음
MissingPrefixLevelStorageMetrics	접두사 수준이 활성화된 경우 접두사 수준의 스토리지 지표는 필수입니다.	400 Bad Request	지원되지 않음
OrganizationAccessDenied	이 계정은 AWS Organizations를 추가할 권한이 없습니다.	403 금지됨	지원되지 않음

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
OrgConfigurationNotSupported	지정된 리전은 구성에서 AWS Organizations를 지원하지 않습니다.	403 금지됨	지원되지 않음
ServiceNotEnabledForOrg	조직에 대해 S3 Storage Lens 서비스 연결 역할이 활성화되지 않았습니다.	403 금지됨	지원되지 않음
StorageMetricsMustEnabled	접두사 수준 스토리지 지표를 활성화해야 합니다.	400 Bad Request	지원되지 않음
TooManyBuckets	버킷 컨테이너에는 버킷이 50개를 초과할 수 없습니다.	400 Bad Request	지원되지 않음
TooManyRegions	리전 컨테이너에는 리전이 50개를 초과할 수 없습니다.	400 Bad Request	지원되지 않음
TooManyStorageLensGroups	Storage Lens 대시보드에는 Storage Lens 그룹을 50개 이상 연결할 수 없습니다.	400 Bad Request	지원되지 않음

다음 표에는 S3 Storage Lens 그룹 작업이 반환할 수 있는 특수 오류가 나와 있습니다. Amazon S3 오류에 대한 일반적인 내용 및 오류 코드 목록은 [오류 응답](#) 섹션을 참조하세요.

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
AccessDenied	Storage Lens 그룹 작업을 수행할 권한이 없습니다. 이 리전은 S3 Storage Lens 그룹의 홈 리전으로 지원되지 않습니다.	403 Forbidden	지원되지 않음
ConfigurationAlreadyExists	지정된 구성이 이미 있습니다.	409 Conflict	지원되지 않음

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
DuplicateElement	태그는 고유해야 합니다. And 논리 연산자에 중복 태그 키가 포함되어 있습니다. Or 논리 연산자에 중복 태그가 포함되어 있습니다. 논리 연산자에 중복 접두사 또는 접미사가 포함되어 있습니다.	400 Bad Request	지원되지 않음
InvalidAge	DaysLessThan 및 DaysGreaterThan 은 양수여야 합니다.	400 Bad Request	지원되지 않음
InvalidFilter	필터에는 And, Or, MatchAnyTag, MatchAnyPrefix, MatchAnySuffix, MatchObjectAge, MatchObjectSize 요소 중 하나가 포함되어야 합니다.	400 Bad Request	지원되지 않음
InvalidLogicalOperator	논리 연산자 And 또는 Or에는 두 개 이상의 하위 요소가 있어야 합니다.	400 Bad Request	지원되지 않음
InvalidMatchAnyPrefix	MatchAnyPrefix 파라미터는 비어 있을 수 없습니다.	400 Bad Request	지원되지 않음
InvalidMatchAnySuffix	MatchAnySuffix 파라미터는 비어 있을 수 없습니다.	400 Bad Request	지원되지 않음
InvalidMatchAnyTag	MatchAnyTag 파라미터는 비어 있을 수 없습니다.	400 Bad Request	지원되지 않음
InvalidMatchObjectAge	MatchObjectAge 파라미터는 비어 있을 수 없습니다.	400 Bad Request	지원되지 않음
InvalidMatchObjectSize	MatchObjectSize 파라미터는 비어 있을 수 없습니다.	400 Bad Request	지원되지 않음

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
InvalidName	Storage Lens 그룹 이름 파라미터는 1~64자여야 합니다. Storage Lens 그룹 Name 파라미터는 <code>^[a-zA-Z0-9\-_]+\$</code> 패턴을 사용해야 합니다.	400 Bad Request	지원되지 않음
InvalidNumericCombination	이 객체 수명 또는 객체 크기 조합은 유효하지 않습니다.	400 Bad Request	지원되지 않음
InvalidPrefix	접두사의 최대 길이는 1,024자입니다. 접두사 문자열은 비어 있을 수 없습니다.	400 Bad Request	지원되지 않음
InvalidSize	BytesLessThan 및 BytesGreaterThan 은 양수여야 합니다. 최대 객체 크기는 50TB를 초과할 수 없습니다. 최소 객체 크기는 50TB보다 크거나 같을 수 없습니다.	400 Bad Request	지원되지 않음
InvalidSuffix	접미사의 최대 길이는 1,024자입니다. 접미사 문자열은 비어 있을 수 없습니다.	400 Bad Request	지원되지 않음
InvalidTag	객체 태그 키는 128자를 초과할 수 없습니다. 객체 태그 키 문자열은 null이거나 비어 있을 수 없습니다. 태그 값의 최대 길이는 256자입니다. 객체 태그 키에 유효하지 않은 문자가 포함되어 있습니다. 객체 태그 키에는 a~z, A~Z, 0~9, 공백 및 다음 문자만 포함되어야 합니다. <code>^(_\.:\/+=\-\@]*)*\$</code> .	400 Bad Request	지원되지 않음

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
MismatchedName	요청에 지정된 이름이 Storage Lens 그룹 이름과 일치하지 않습니다.	400 Bad Request	지원되지 않음
TooManyConfigurations	Storage Lens 그룹 구성을 허용된 50개보다 많이 생성하려고 했습니다.	400 Bad Request	지원되지 않음
TooManyElements	Element가 논리 연산자 내에서 허용되는 최대 요소 수를 초과합니다. 접두사, 접미사 또는 태그가 10개만 허용됩니다.	400 Bad Request	지원되지 않음

Amazon S3 Object Lambda 오류 코드 목록

다음 표에는 S3 객체 Lambda가 반환할 수 있는 특수 오류가 나와 있습니다. 일반적인 Amazon S3 오류에 대한 자세한 내용 및 오류 코드 목록은 [오류 응답](#) 섹션을 참조하세요.

비GetObject 요청 동안 지원 액세스 포인트에서 수신한 오류 응답은 변경되지 않은 상태로 호출자에게 전송됩니다.

오류 코드	설명	HTTP 상태 코드	
LambdaInvalidResponse	WriteGetObjectResponse 가 AWS Lambda에 ValidationError 로 응답할 때 원래 호출자에게 반환됩니다. 자세한 내용은 ValidationError 메시지를 참조하세요. ValidationError 의 모든 경우에서 LambdaInvalidResponse 오류가 발생하는 것은 아닙니다.	400 Bad Request	

오류 코드	설명	HTTP 상태 코드	
LambdaInvocationFailed	Lambda 함수 간접 호출이 실패했습니다. S3 객체 Lambda가 구성된 Lambda 함수를 간접적으로 호출할 수 없는 경우 호출자에게 다음 오류가 표시될 수 있습니다. 오류 메시지에에는 함수를 호출할 때 AWS Lambda 서비스에서 반환한 최종 오류에 대한 세부 정보(예: 상태 코드, 오류 코드, 오류 메시지, 요청 ID)가 포함될 수 있습니다.	400 Bad Request	
LambdaNotFound	AWS Lambda 함수를 찾을 수 없습니다. 간접 호출 시도 시 구성된 Lambda 함수, 버전 또는 별칭을 찾을 수 없습니다. S3 객체 Lambda 액세스 포인트 구성이 올바른 Lambda 함수 ARN을 가리키는지 확인하세요. 오류 메시지에에는 함수를 호출할 때 AWS Lambda 서비스에서 반환한 최종 오류에 대한 세부 정보(예: 상태 코드, 오류 코드, 오류 메시지, 요청 ID)가 포함될 수 있습니다.	404 Not Found	

오류 코드	설명	HTTP 상태 코드	
LambdaPermissionError	호출자에게 Lambda 함수를 간접적으로 호출할 권한이 없습니다. 호출자에게 Lambda 함수를 간접적으로 호출할 권한이 있어야 합니다. 호출자에게 연결된 정책을 검토하여 구성된 함수에 <code>lambda:Invoke</code> 를 사용할 수 있도록 허용되었는지 확인하세요. 오류 메시지에는 함수를 호출할 때 AWS Lambda 서비스에서 반환한 최종 오류에 대한 세부 정보(예: 상태 코드, 오류 코드, 오류 메시지, 요청 ID)가 포함될 수 있습니다.	403 Forbidden	
LambdaResponseNotReceived	Lambda 함수가 <code>WriteGetObjectResponse</code> 를 직접적으로 호출하지 않고 종료되었습니다. <code>GetObject</code> 응답 데이터는 Lambda 함수가 <code>WriteGetObjectResponse</code> API 작업을 직접적으로 호출하여 제공합니다. 함수가 정상적으로 종료되었음에도 이 API 작업을 직접적으로 호출하지 못한 이유는 함수에 대한 Amazon CloudWatch 로그를 통해 더 자세히 파악할 수 있습니다.	500 Internal Service Error	

오류 코드	설명	HTTP 상태 코드	
LambdaRuntimeError	Lambda 함수가 실행 중에 실패했습니다. Lambda 함수에서 명시적 오류가 수신되었습니다. 실패에 대한 자세한 내용은 AWS CloudFormation 로그를 확인하세요.	500 Internal Service Error	
LambdaTimeout	Lambda 함수가 허용된 시간 내에 응답하지 않았습니다. Lambda 함수가 60초 이내에 WriteGetObjectResponse 에 대한 직접 호출을 완료하지 못했습니다.	500 Internal Service Error	
SlowDown	AWS Lambda 관련 작업에 대한 요청 속도를 줄이세요. 함수 간접 호출이 AWS Lambda에 의해 제한되었습니다. 구성된 동시성 제한에 도달한 것일 수 있습니다. 자세한 내용은 AWS Lambda 개발자 안내서의 Lambda 함수의 동시성 관리를 참조하세요. 오류 메시지에는 함수를 호출할 때 AWS Lambda 서비스에서 반환한 최종 오류에 대한 세부 정보(예: 상태 코드, 오류 코드, 오류 메시지, 요청 ID)가 포함될 수 있습니다.	503 Slow Down	

오류 코드	설명	HTTP 상태 코드	
ValidationError	WriteGetObjectResponse API 작업에서 검증 오류가 반환될 수 있으며 이는 여러 가지 이유로 발생할 수 있습니다. 자세한 내용은 오류 메시지를 참조하세요.	400 Bad Request	

Amazon S3 비동기 오류 코드 목록

다음 표에는 비동기 요청이 반환할 수 있는 특수 오류가 나와 있습니다. 일반적인 Amazon S3 오류에 대한 자세한 내용 및 오류 코드 목록은 [오류 응답](#) 섹션을 참조하세요.

이러한 오류는 DescribeMultiRegionAccessPointOperation 사용 등 비동기 요청의 상태를 쿼리할 때 반환됩니다. 이러한 요청은 비동기식이므로 이러한 오류의 상태 코드는 모두 200 OK입니다.

오류 코드	설명	HTTP 상태 코드	
AccessDenied	액세스가 거부되었습니다.	200 OK	
InternalErrors	내부 서버 오류가 발생했습니다.	200 OK	
MalformedPolicy	지정된 정책 구문이 유효하지 않습니다.	200 OK	
MultiRegionAccessPointAlreadyOwnedByYou	이름이 동일한 다중 리전 액세스 포인트가 이미 존재합니다.	200 OK	
MultiRegionAccessPointModifiedByAnotherRequest	다른 요청이 지정된 리소스를 수정하고 있기 때문에 작업이 실패했습니다. 이전 요청이 완료된 후 요청을 다시 제출해 보세요.	200 OK	

오류 코드	설명	HTTP 상태 코드	
MultiRegionAccessPointNotReady	지정된 다중 리전 액세스 포인트를 업데이트할 준비가 되지 않았습니다.	200 OK	
MultiRegionAccessPointSameBucketRegion	다중 리전 액세스 포인트를 생성하는데 사용되는 버킷은 동일한 리전에 있을 수 없습니다.	200 OK	
MultiRegionAccessPointUnsupportedRegion	다중 리전 액세스 포인트를 생성하기 위해 제공된 버킷 중 하나가 지원되지 않는 리전에 있습니다.	200 OK	
NoSuchBucket	지정된 버킷이 존재하지 않습니다.	200 OK	
NoSuchMultiRegionAccessPoint	지정된 다중 리전 액세스 포인트가 존재하지 않습니다.	200 OK	

Amazon S3 Access Grants 오류 코드 목록

다음 표에는 S3 Access Grants 요청이 반환할 수 있는 특수 오류가 나와 있습니다. 일반적인 Amazon S3 오류에 대한 자세한 내용 및 오류 코드 목록은 [오류 응답](#) 섹션을 참조하세요.

오류 코드	설명	HTTP 상태 코드	
AccessGrantAlreadyExists	지정된 액세스 권한이 이미 있습니다.	409	
AccessGrantsInstanceAlreadyExists	Access Grants 인스턴스가 이미 있습니다.	409	
AccessGrantsInstanceNotEmptyError	Access Grants 인스턴스를 삭제하기 전에 위치를 정리하세요.	400	
AccessGrantsInstanceNotExistsError	Access Grants 인스턴스가 존재하지 않습니다.	404	

오류 코드	설명	HTTP 상태 코드	
AccessGrantsInstanceResourcePolicyNotExists	Access Grants 인스턴스 리소스 정책이 없습니다.	404	
AccessGrantsLocationAlreadyExistsError	지정된 Access Grants 위치가 이미 존재합니다.	409	
AccessGrantsLocationNotEmptyError	Access Grants 위치를 삭제하기 전에 액세스 권한 부여를 정리하세요.	400	
AccessGrantsLocationsQuotaExceededError	Access Grants 위치 할당량을 초과했습니다. Access Grants 위치 할당량: <i><value></i> . 증가가 필요한 경우 S3에 문의하세요.	409	
AccessGrantsQuotaExceededError	액세스 권한 부여 할당량을 초과했습니다. 액세스 권한 부여 할당량: <i><value></i> . 증가가 필요한 경우 S3에 문의하세요.	409	
InvalidTag	요청에 중복된 태그 키가 있습니다. 중복 태그 키를 제거하고 다시 시도하세요.	400	
InvalidAccessGrant	지정된 액세스 권한 부여가 유효하지 않습니다.	400	
InvalidAccessGrantLocation	지정된 Access Grants 위치가 유효하지 않습니다.	400	
InvalidIamRole	지정된 IAM 역할이 유효하지 않습니다.	400	
InvalidIdentityCenterInstance	지정된 자격 증명 센터 인스턴스가 유효하지 않습니다.	400	

오류 코드	설명	HTTP 상태 코드	
InvalidResourcePolicy	지정된 리소스가 정책이 유효하지 않습니다.	400	
InvalidResourcePolicy	지정된 리소스가 정책이 유효하지 않습니다.	400	
InvalidTag	이 요청에 유효하지 않은 태그 키 또는 값이 포함되어 있습니다. 유효한 문자에는 [a~z A~Z + - = . _ : /]가 포함됩니다. 태그 키는 최대 128자입니다. 태그 값은 최대 256자입니다.	400	
NoSuchAccessGrantError	지정된 액세스 권한 부여가 없습니다.	404	
NoSuchAccessGrantsLocationError	지정된 Access Grants 위치가 존재하지 않습니다.	404	
AccessDenied	요청된 S3 접두사: <### ##> 에 대한 <### ##> 권한이 없습니다.	403 Forbidden	
StsNotAuthorizedError	GetDataAccess 작업을 직접적으로 호출할 때 오류(StsNotAuthorizedError)가 발생했습니다. 사용자: access-grants.s3.amazonaws.com 은 리소스: <IAM ## ARN> 에서 sts:AssumeRole 을 수행할 권한이 없습니다.	403	
StsPackedPolicyTooLargeError	GetDataAccess 작업을 직접적으로 호출할 때 오류(StsPackedPolicyTooLargeError)가 발생했습니다. 직렬화된 토큰이 세션에 비해 너무 큼니다.	400	

오류 코드	설명	HTTP 상태 코드	
StsValidationError	오류 메시지는 검증 오류에 따라 달라집니다.	400	
InvalidTags	태그 키는 시스템 태그에 예약된 접두사 AWS로 시작할 수 없습니다."	400	
TooManyTags	태그 수가 태그 제한 50개를 초과합니다. 일부 태그를 제거하고 다시 시도하세요.	400	

Amazon FSx 관련 오류 코드 목록

다음 표에는 Amazon S3 액세스 포인트를 통한 Amazon FSx에 대한 요청이 반환할 수 있는 특수 오류가 나와 있습니다. 일반적인 Amazon S3 오류에 대한 자세한 내용 및 오류 코드 목록은 [오류 응답](#) 섹션을 참조하세요.

오류 코드	설명	HTTP 상태 코드	
AccessDenied	이 오류는 다음과 같은 이유로 발생할 수 있습니다. - 호출자가 지정된 S3 작업을 수행할 권한이 없습니다. - 호출자가 지정된 키의 Amazon FSx 리소스에 액세스할 권한이 없습니다.	403 Forbidden	
InsufficientCapacity	파일 시스템의 최대 스토리지 용량에 도달했습니다.	507 Insufficient Capacity	

오류 코드	설명	HTTP 상태 코드
InvalidKey	지정된 키가 유효하지 않습니다.	400 Bad Request
SlowDown	요청 속도를 낮추세요.	503 Slow Down

Amazon S3 Tables 오류 코드 목록

다음 표에는 Amazon S3 Tables 작업이 반환할 수 있는 특수 오류가 나와 있습니다. 일반적인 Amazon S3 오류에 대한 자세한 내용 및 오류 코드 목록은 [오류 응답](#) 섹션을 참조하세요.

오류 코드	설명	HTTP 상태 코드
AccessDeniedException	필요한 권한이 없으므로 작업을 수행할 수 없습니다.	403 Forbidden
BadRequestException	요청이 유효하지 않거나 잘못된 형식입니다.	400 Bad Request
ConflictException	이전 쓰기와 충돌하여 요청이 실패했습니다. 요청을 재시도할 수 있습니다.	409 Conflict
ForbiddenException	호출자는 요청을 만들 권한이 없습니다.	403 Forbidden
InternalServerErrorException	내부 서버 오류로 인해 요청이 실패했습니다.	500 Internal Server Error
NotFoundException	지정된 리소스를 찾을 수 없으므로 요청이 거부되었습니다.	404 Not Found

오류 코드	설명	HTTP 상태 코드
TooManyRequestsException	초당 요청 수 제한이 초과되었습니다.	429 Too Many Requests

Amazon S3 오류 모범 사례

많은 오류 응답에는 프로그래밍 오류를 진단하는 개발자가 판독하고 이해하기 위한 체계적 데이터가 추가로 포함되어 있습니다. 예를 들어, REST PUT 요청에 서버에서 계산된 다이제스트와 일치하지 않는 Content-MD5 헤더를 보내면 BadDigest 오류가 반환됩니다. 오류 응답에는 계산한 다이제스트와 예상한 다이제스트와 같은 세부적인 요소도 포함합니다. 개발 단계 이 정보를 사용하여 오류를 진단할 수 있습니다. 프로덕션 단계에서 모범적인 프로그램은 오류 로그에 이 내용을 포함할 수 있습니다.

Amazon S3과 함께 사용할 애플리케이션을 설계할 경우, Amazon S3 오류를 적절하게 처리하는 것이 중요합니다. 이 단원에서는 애플리케이션 설계 시 고려할 문제에 대해 설명합니다.

InternalErrors 재시도

내부 오류는 Amazon S3 환경 내에서 발생하는 오류입니다.

InternalError 응답을 받은 요청은 처리되지 않을 수 있습니다. 예를 들어, PUT 요청이 InternalError를 반환하는 경우, 후속 GET은 이전 값 또는 업데이트된 값을 검색할 수 있습니다.

Amazon S3이 InternalError 응답을 반환하는 경우, 요청을 재시도하세요.

반복되는 SlowDown 오류에 대한 애플리케이션 튜닝

분산 시스템과 마찬가지로 S3는 의도적이거나 의도되지 않은 리소스 초과 사용을 감지하여 이에 대응하는 보호 메커니즘이 있습니다. SlowDown 오류는 이러한 메커니즘 중 하나가 높은 빈도로 요청될 때 발생할 수 있습니다. 요청 빈도를 줄이면 이 유형의 오류가 감소하거나 제거됩니다. 일반적으로 대부분의 사용자는 이러한 오류를 자주 겪지 않습니다. 하지만 이러한 오류에 대한 자세한 내용을 알고 싶거나, SlowDown 오류가 자주 발생하거나 예기치 않은 경우에 발생할 경우 [Amazon S3 개발자 포럼](https://aws.amazon.com/premiumsupport/)에 글을 게시하거나 지원 <https://aws.amazon.com/premiumsupport/>에 가입하세요.

오류 격리

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료 (EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

Amazon S3은 SOAP 및 REST API에서 사용하는 오류 코드 세트를 제공합니다. SOAP API는 표준 Amazon S3 오류 코드를 반환합니다. REST API는 표준 HTTP 서버와 같이 설계되고 기존 HTTP 클라이언트와 연동합니다(예: 브라우저, HTTP 클라이언트 라이브러리, 프록시, 캐시 등). HTTP 클라이언트가 오류를 적절하게 처리할 수 있도록 하기 위해 각 Amazon S3 오류를 HTTP 상태 코드에 매핑합니다.

HTTP 상태 코드는 Amazon S3 오류 코드보다 이해하기 어렵고, 오류에 관한 정보가 부족합니다. 예를 들어, NoSuchKey 및 NoSuchBucket Amazon S3 오류는 HTTP 404 Not Found 상태 코드에 매핑됩니다.

HTTP 상태 코드에는 오류 정보가 부족하지만 HTTP를 인식하는 클라이언트(Amazon S3 API 제외)는 일반적으로 오류를 제대로 처리합니다.

따라서 오류를 처리하거나 Amazon S3 오류를 최종 사용자에게 보고할 경우, HTTP 상태 코드 대신, 오류에 관한 많은 정보를 포함하는 Amazon S3 오류 코드를 사용하세요. 또한 애플리케이션을 디버깅할 경우, 사람이 읽을 수 있는 XML 오류 응답의 <Details> 요소를 참조해야 합니다.

AWS 용어집

최신 AWS 용어는 AWS 용어집 참조서의 [AWS 용어집](#)을 참조하세요.

Amazon S3 리소스

다음은 이 서비스를 이용할 때 유용하게 참조할 수 있는 관련 리소스가 나와 있는 표입니다.

리소스	설명
Amazon Simple Storage Service 사용 설명서.	이 시작 안내서에서는 간단한 사용 사례를 기반으로 빠른 서비스 자습서를 제공합니다.
Amazon Simple Storage Service 사용 설명서.	이 개발자 안내서에서는 Amazon S3 작업을 사용하여 태스크를 수행하는 방법을 설명합니다.
Amazon S3 기술 FAQ	FAQ는 이 제품에 대해 개발자들이 가장 많이 질문한 상위 20개 내용을 소개합니다.
Amazon S3 릴리스 정보	출시 정보에서는 최신 릴리스에 대한 수준 높은 개요를 제공합니다. 특히 새로운 기능, 상관 관계 및 알려진 문제점에 대해 다룹니다.
Amazon Web Services용 도구	AWS SDK 및 도구를 사용하여 혁신적인 애플리케이션을 구현하는 데 도움이 되는 설명서, 코드 샘플, 릴리스 정보 및 기타 정보를 이 한 곳에서 찾을 수 있습니다.
AWS Management Console	이 콘솔을 사용하면 프로그래밍 없이 Amazon S3 기능 중 대부분을 수행할 수 있습니다.
토론 포럼	Amazon Web Services와 관련된 기술적 질문에 대해 토론할 수 있는 개발자를 위한 커뮤니티 기반 포럼입니다.
지원 Center	AWS Technical Support 홈 페이지입니다. 이 페이지에서 개발자 포럼, 기술 FAQ, 서비스 상태 페이지, Premium Support에 액세스할 수 있습니다.
지원	지원에 대한 정보를 제공하는 기본 웹페이지로, 일대일의 신속한 응대 지원 채널을 통해 AWS 인프라 서비스에서 애플리케이션을 빌드하고 실행하도록 지원합니다.

리소스	설명
Amazon S3 제품 정보	Amazon S3에 대한 정보를 제공하는 기본 웹 페이지입니다.
문의처	AWS 결제, 계정, 이벤트, 침해 등에 대해 문의할 수 있는 중앙 연락 지점입니다.
사용 조건	Amazon.com에서의 저작권 및 상표 사용과 기타 주제에 대한 세부 정보입니다.

부록

주제

- [부록: SelectObjectContent 응답](#)
- [부록: OPTIONS 객체](#)
- [부록: SOAP API](#)
- [부록: 요청 인증\(AWS 서명 버전 2\)](#)
- [부록: 수명 주기 구성 API\(사용되지 않음\)](#)

부록: SelectObjectContent 응답

설명

Amazon S3 Select 작업은 간단한 구조화된 쿼리 언어(SQL) 문을 기반으로 Amazon S3 객체의 콘텐츠를 필터링합니다. 이 작업의 응답 크기를 알 수 없으므로, Amazon S3 Select는 응답을 일련의 메시지로 스트리밍하고 응답에 값이 **chunked**인 Transfer-Encoding 헤더를 포함합니다.

Amazon S3 Select에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [객체에서 콘텐츠 선택](#)을 참조하세요.

Amazon S3 Select와 함께 SQL을 사용하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [Amazon S3 Select 및 Amazon Glacier Select용 SQL 참조](#)를 참조하세요.

응답

Amazon S3 Select 작업이 성공하면 200 OK 상태 코드가 반환됩니다.

응답 헤더

이 작업의 구현에서는 대부분의 응답에 공통되는 응답 헤더만 사용합니다. 자세한 내용은 [공통 응답 헤더](#)를 참조하세요.

응답 본문

Amazon S3 Select 응답 크기를 알 수 없으므로, Amazon S3는 응답을 일련의 메시지로 스트리밍하고 응답에 값이 **chunked**인 Transfer-Encoding 헤더를 포함합니다. 다음 예제는 최상위 수준의 응답 형식을 보여줍니다.

```
<Message 1>
<Message 2>
<Message 3>
.....
<Message n>
```

각 메시지는 두 섹션(서문 및 데이터)으로 구성됩니다. 서문 섹션은 1) 메시지의 총 바이트 길이 및 2) 모든 헤더의 총 바이트 길이로 구성됩니다. 데이터 섹션은 1) 헤더 및 2) 페이로드로 구성됩니다.

각 섹션은 4바이트 빅 엔디안 정수 체크섬(CRC)으로 끝납니다. Amazon S3 Select는 CRC32(GZIP CRC32라고도 함)를 사용하여 두 CRC를 모두 계산합니다. CRC32에 대한 자세한 내용은 [GZIP file format specification version 4.3](#)을 참조하세요.

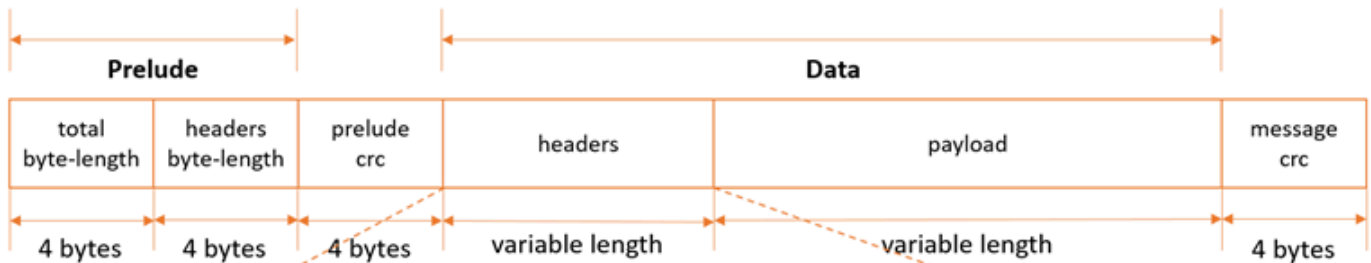
총 메시지 오버헤드(서문과 두 체크섬 포함)는 16바이트입니다.

Note

메시지 내의 모든 integer 값은 네트워크 바이트 순서 또는 빅 엔디안 순서입니다.

다음 다이어그램은 메시지와 헤더를 구성하는 구성 요소를 보여줍니다. 메시지별로 여러 개의 헤더가 있습니다.

Message:



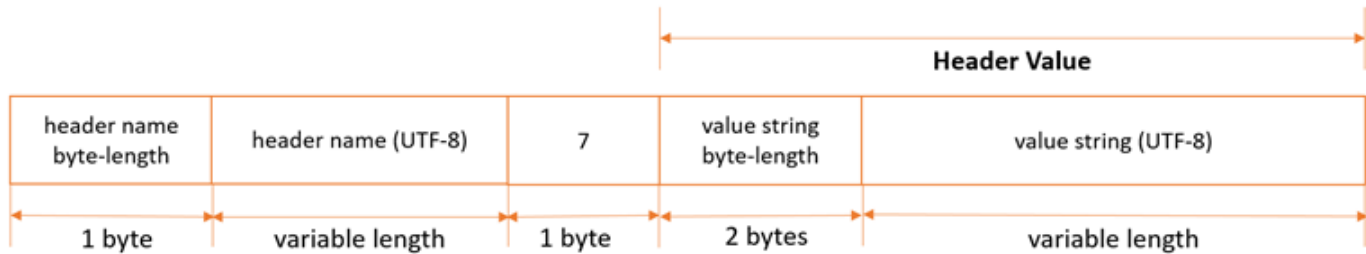
Headers (multiple headers per message):



Note

Amazon S3 Select의 경우 헤더 값 유형은 항상 7(type=String)입니다. 이 유형의 경우 헤더 값은 두 가지 구성 요소, 즉 2바이트 빅 엔디안 정수 길이와 해당 바이트 길이의 UTF-8 문자열로 이루어집니다. 다음 다이어그램은 Amazon S3 Select 헤더를 구성하는 구성 요소를 보여줍니다.

Amazon S3 Select Headers (type=String):



페이로드 바이트 길이 계산(다음 두 계산은 동일):

- $\text{payload\_length} = \text{total\_length} - \text{header\_length} - \text{sizeof}(\text{total\_length}) - \text{sizeof}(\text{header\_length}) - \text{sizeof}(\text{prelude\_crc}) - \text{sizeof}(\text{message\_crc})$
- $\text{payload\_length} = \text{total\_length} - \text{header\_length} - 16$

각 메시지는 다음 구성 요소를 포함합니다.

- 서문: 항상 8바이트 고정 크기이며 두 필드가 각각 4바이트를 차지합니다.
 - 처음 4바이트: 총 바이트 길이: 전체 메시지의 빅 엔디안 정수 바이트 길이입니다(4바이트 총 길이 필드 자체를 포함).
 - 두 번째 4바이트: 헤더 바이트 길이: 메시지 헤더 부분의 빅 엔디안 정수 바이트 길이입니다(헤더 길이 필드 자체는 제외).
- 서문 CRC: 메시지 서문 부분의 4바이트 빅 엔디안 정수 체크섬(CRC)입니다(CRC 자체는 제외). 서문에는 메시지 CRC(아래 참조)와 분리된 CRC가 있어서 비정상적인 버퍼링 동작을 일으키지 않으면서 손상된 바이트 길이 정보를 즉시 감지할 수 있습니다.
- 헤더: 메시지 유형, 페이로드 형식 등 메시지에 대한 정보를 포함하는 메타데이터 세트입니다. 메시지는 여러 헤더가 있을 수 있으므로 메시지의 이 부분은 메시지 유형에 따라 바이트 길이가 다를 수 있습니다. 헤더는 키-값 페어이며, 키와 값은 모두 UTF-8 문자열입니다. 헤더는 메시지의 헤더 부분에 순서에 상관없이 표시되며 특정 헤더 유형은 한 번만 표시될 수 있습니다.

Amazon S3 Select의 경우 다음은 메시지 유형에 따른 헤더 이름 목록 및 유효한 값 세트입니다.

- MessageType 헤더:
 - 헤더 이름 => ":message-type"
 - 유효한 헤더 값 => "error", "event"
- EventType 헤더:

- 헤더 이름 => ":event-type"
- 유효한 헤더 값 => "Records", "Cont", "Progress", "Stats", "End"
- ErrorCode 헤더:
 - 헤더 이름 => ":error-code"
 - 유효한 헤더 값 => [SELECT 객체 콘텐츠 오류 코드 목록](#) 섹션의 테이블에 있는 오류 코드입니다.
- ErrorMessage 헤더:
 - 헤더 이름 => ":error-message"
 - 유효한 헤더 값 => 요청 수준 오류를 진단하는 데 도움이 되도록 서비스에서 반환하는 오류 메시지입니다.
- 페이로드: 무엇이든 가능합니다.
- 메시지 CRC: 메시지 시작부터 체크섬 시작까지 4바이트 빅 엔디안 정수 체크섬(CRC), 즉 메시지 CRC 자체를 제외한 메시지의 모든 항목입니다.

각 헤더는 다음 구성 요소를 포함합니다. 메시지별로 여러 개의 헤더가 있을 수 있습니다.

- 헤더 이름 바이트 길이: 헤더 이름의 바이트 길이입니다.
- 헤더 이름: 헤더 유형을 나타내는 헤더 이름입니다. 유효한 값: ":message-type" ":event-type" ":error-code" ":error-message"
- 헤더 값 유형: 헤더 값 유형을 나타내는 열거형입니다. Amazon S3 Select의 경우 항상 7입니다.
- 값 문자열 바이트 길이: (Amazon S3 Select의 경우) 헤더 값 문자열의 바이트 길이입니다.
- 헤더 값 문자열: (Amazon S3 Select의 경우) 헤더 문자열의 값입니다. 이 필드의 유효한 값은 헤더 유형에 따라 달라집니다. 각 헤더 유형 및 메시지 유형에 대한 유효한 값은 아래 섹션을 참조하세요.

Amazon S3 Select의 경우 응답은 다음 유형의 메시지일 수 있습니다.

- 레코드 메시지: 단일 레코드, 부분 레코드 또는 여러 레코드를 포함할 수 있습니다. 결과의 크기에 따라 응답에 이러한 메시지가 하나 이상이 포함될 수 있습니다.
- 연속 메시지: Amazon S3는 TCP 연결을 유지하기 위해 이 메시지를 주기적으로 전송합니다. 이러한 메시지는 응답에 무작위로 표시됩니다. 클라이언트는 메시지 유형을 감지하고 그에 따라 처리해야 합니다.
- 진행 상황 메시지: Amazon S3는 요청 시 이 메시지를 주기적으로 전송합니다. 여기에는 시작되었지만 아직 완료되지 않은 쿼리의 진행 상황에 대한 정보가 포함되어 있습니다.

- 통계 메시지: Amazon S3는 요청 종료 시 이 메시지를 전송합니다. 여기에는 쿼리에 대한 통계가 포함되어 있습니다.
- 종료 메시지: 요청이 완료되었으며 더 이상 메시지가 전송되지 않음을 나타냅니다. 클라이언트가 End 메시지를 수신할 때까지는 요청이 완료되었다고 가정해서는 안 됩니다.
- RequestLevelError 메시지: Amazon S3는 어떤 이유로든 요청이 실패한 경우 이 메시지를 전송합니다. 실패에 대한 오류 코드 및 오류 메시지가 포함되어 있습니다. Amazon S3는 RequestLevelError 메시지를 전송하는 경우 End 메시지를 전송하지 않습니다.

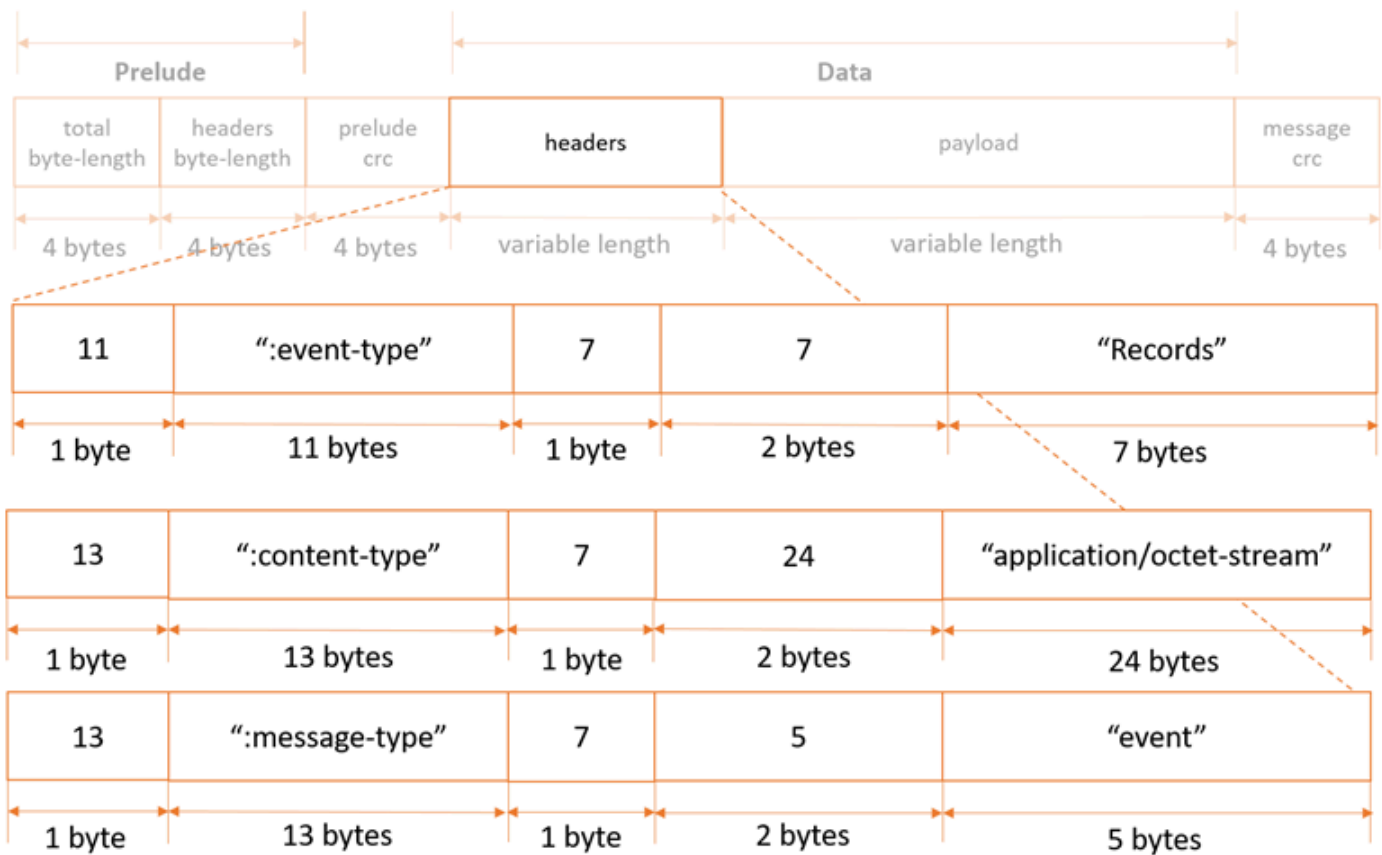
다음 섹션에서는 각 메시지 유형의 구조를 자세히 설명합니다.

이 프로토콜을 사용하는 샘플 코드 및 유닛 테스트는 GitHub 웹 사이트의 [AWS C 이벤트 스트림](#)을 참조하세요.

레코드 메시지

헤더 사양

레코드 메시지에는 다음과 같이 세 개의 헤더가 포함됩니다.



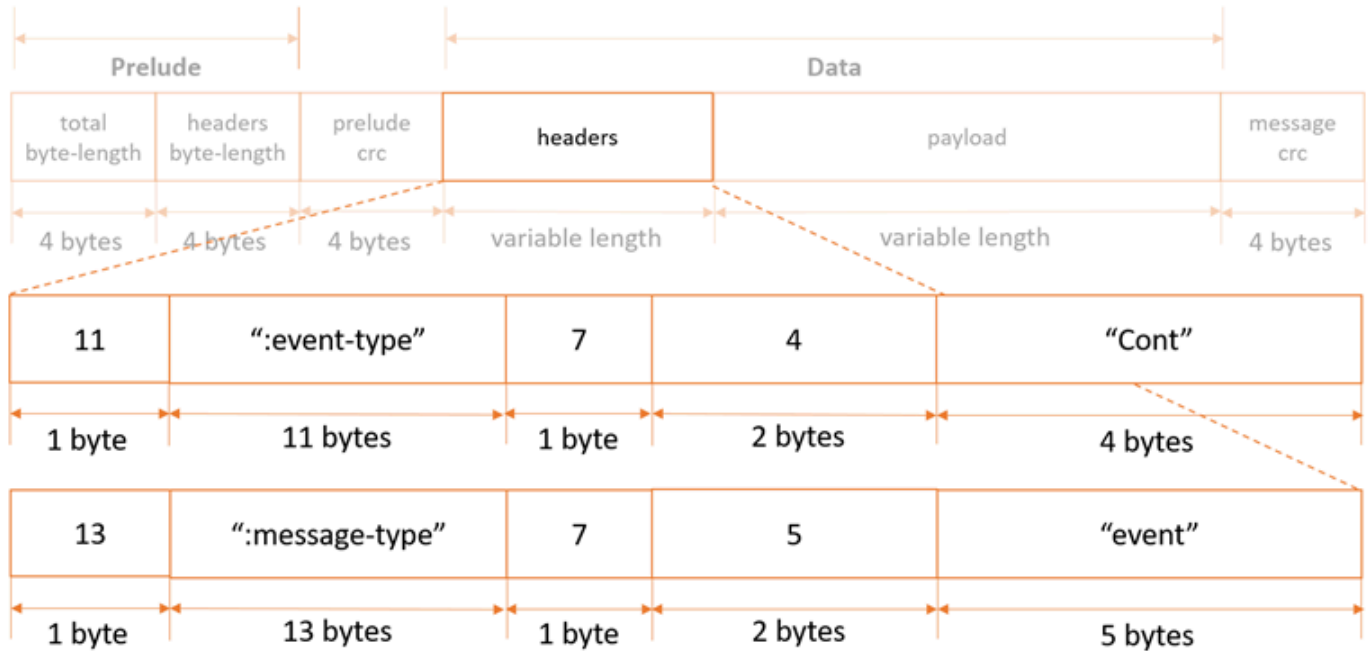
페이로드 사양

레코드 메시지 페이로드에는 단일 레코드, 부분 레코드 또는 여러 레코드가 포함될 수 있습니다.

연속 메시지

헤더 사양

연속 메시지에는 다음과 같이 두 개의 헤더가 포함됩니다.



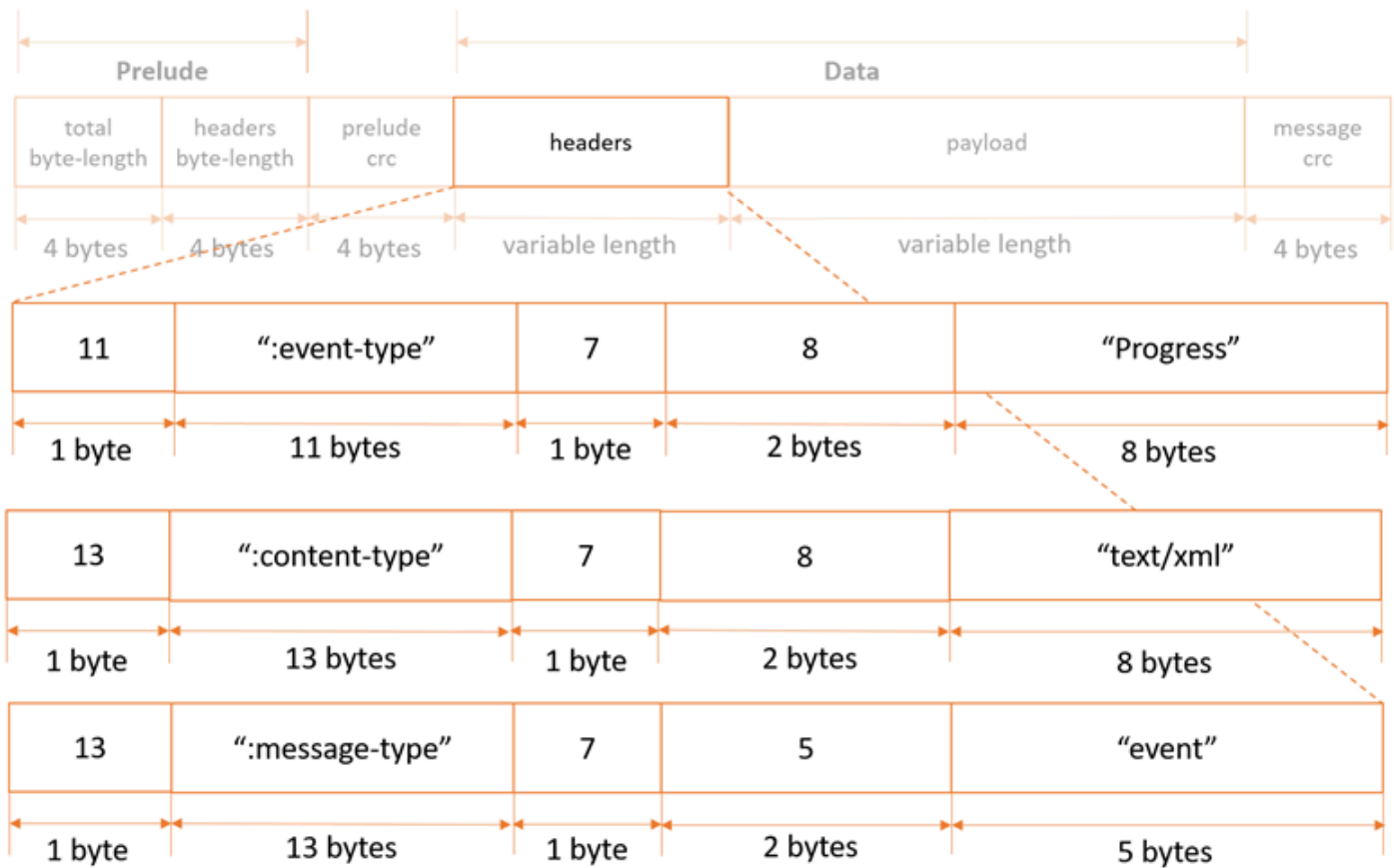
페이로드 사양

연속 메시지에는 페이로드가 없습니다.

진행 상황 메시지

헤더 사양

진행 상황 메시지에는 다음과 같이 세 개의 헤더가 포함됩니다.



페이로드 사양

진행 상황 메시지 페이로드는 요청의 진행 상황에 대한 정보가 포함된 XML 문서입니다.

- BytesScanned => 압축 해제 전에 처리된 바이트 수입니다(파일이 압축된 경우).
- BytesProcessed => 압축 해제 후에 처리된 바이트 수입니다(파일이 압축된 경우).
- BytesReturned => Amazon S3에서 반환한 레코드 페이로드 데이터의 현재 바이트 수입니다.

압축되지 않은 파일의 경우 BytesScanned와 BytesProcessed는 동일합니다.

예제:

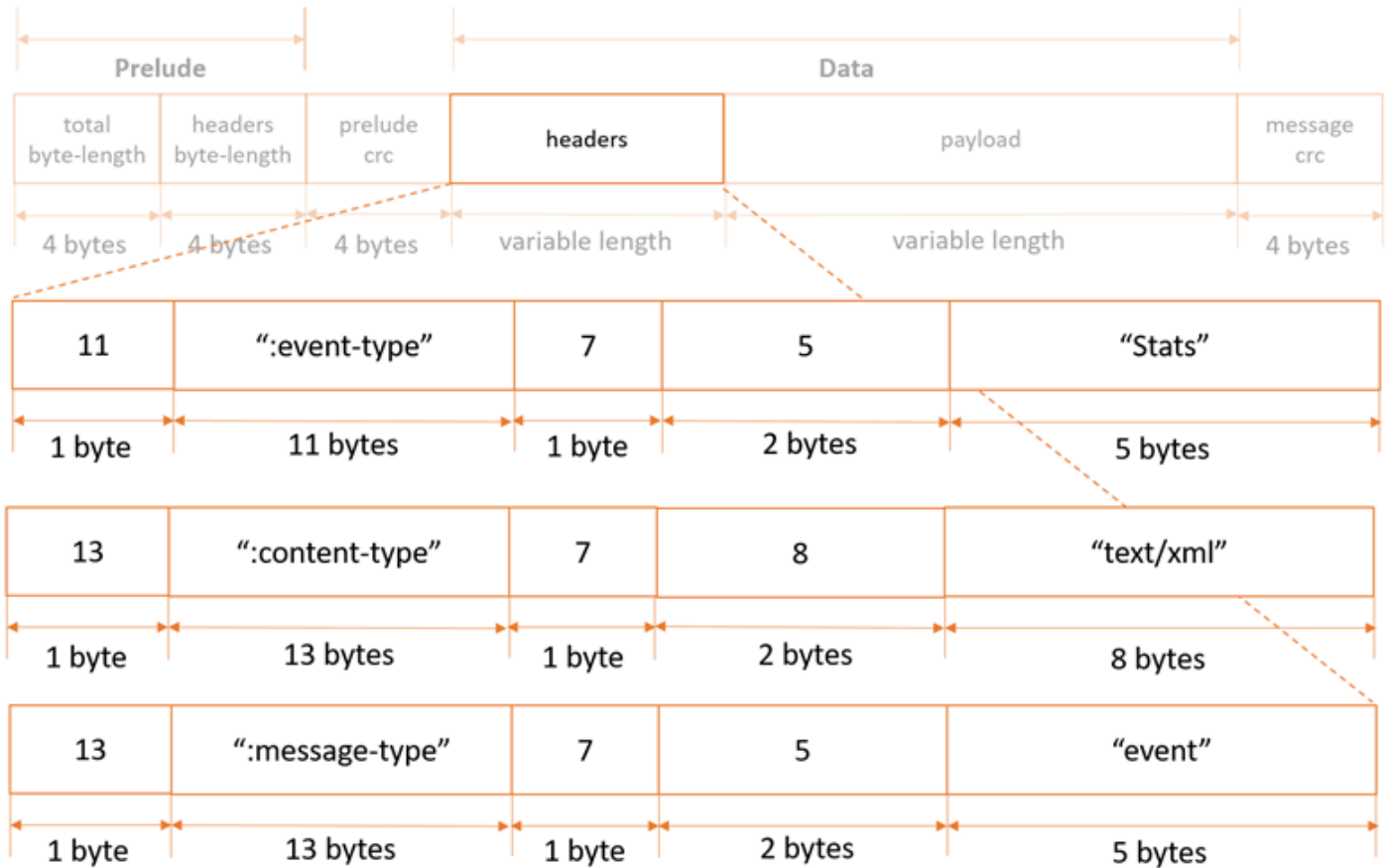
```
<?xml version="1.0" encoding="UTF-8"?>
<Progress>
  <BytesScanned>512</BytesScanned>
  <BytesProcessed>1024</BytesProcessed>
  <BytesReturned>1024</BytesReturned>
```

</Progress>

통계 메시지

헤더 사양

통계 메시지는 다음과 같이 세 개의 헤더가 포함됩니다.



페이로드 사양

통계 메시지 페이로드는 처리가 완료된 요청의 통계에 대한 정보가 포함된 XML 문서입니다.

- BytesScanned => 압축 해제 전에 처리된 바이트 수입니다(파일이 압축된 경우).
- BytesProcessed => 압축 해제 후에 처리된 바이트 수입니다(파일이 압축된 경우).
- BytesReturned => Amazon S3에서 반환한 레코드 페이로드 데이터의 총 바이트 수입니다.

압축되지 않은 파일의 경우 BytesScanned와 BytesProcessed는 동일합니다.

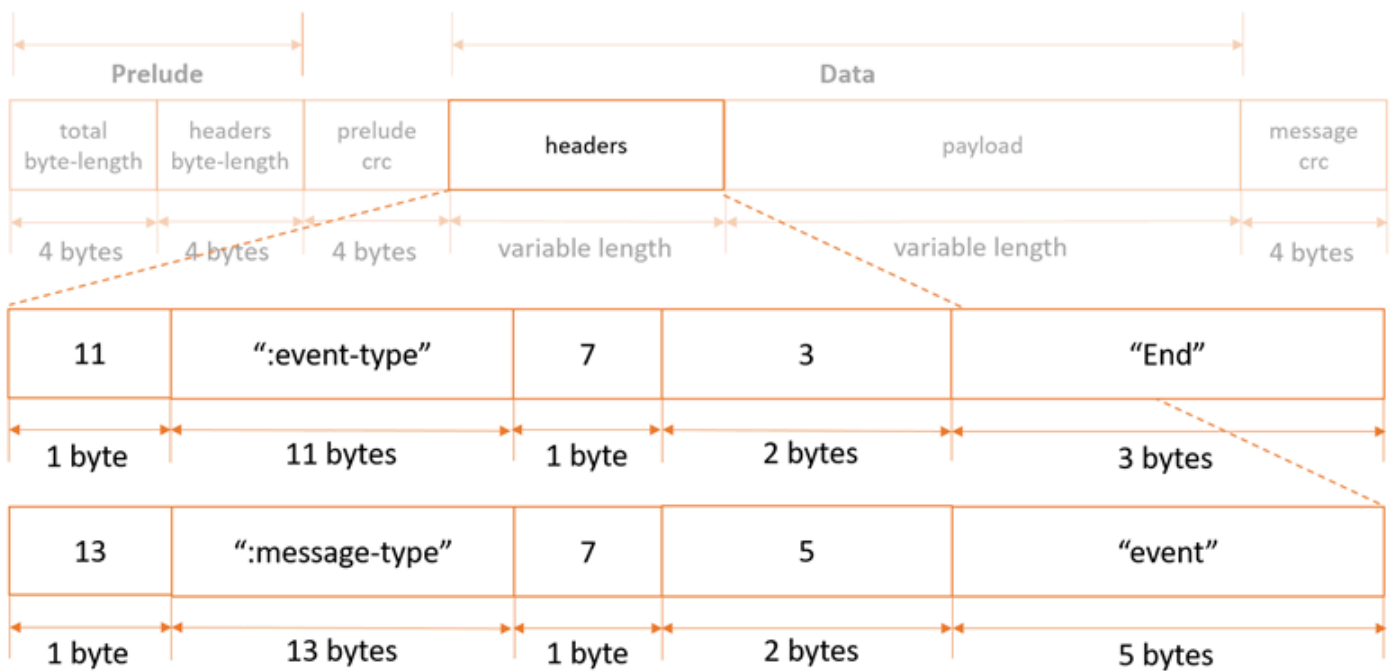
예제:

```
<?xml version="1.0" encoding="UTF-8"?>
<Stats>
  <BytesScanned>512</BytesScanned>
  <BytesProcessed>1024</BytesProcessed>
  <BytesReturned>1024</BytesReturned>
</Stats>
```

종료 메시지

헤더 사양

종료 메시지에는 다음과 같이 두 개의 헤더가 포함됩니다.



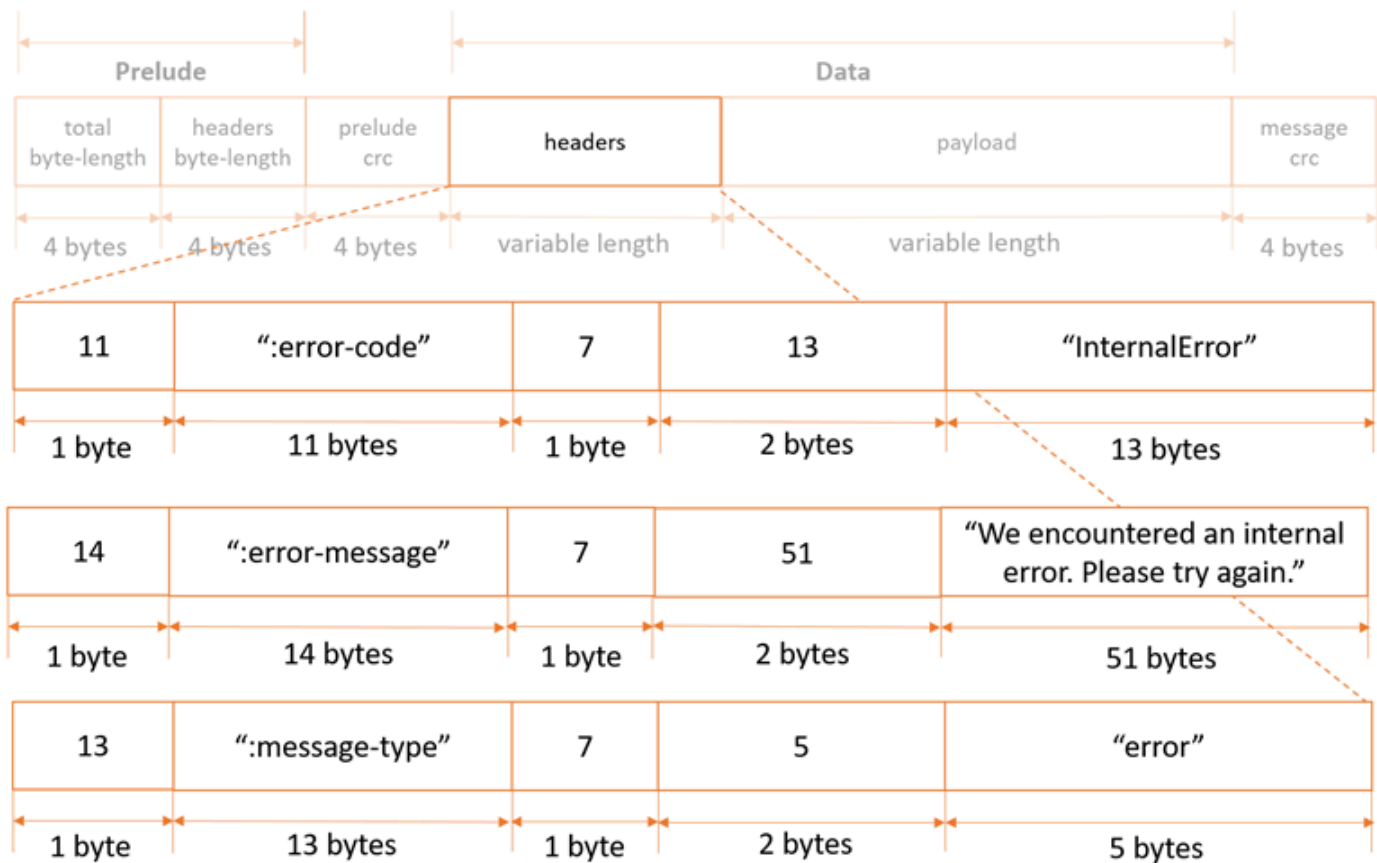
페이로드 사양

종료 메시지에는 페이로드가 없습니다.

요청 수준 오류 메시지

헤더 사양

요청 수준 오류 메시지에는 다음과 같이 세 개의 헤더가 포함됩니다.



가능한 오류 코드 및 오류 메시지 목록은 [SELECT 객체 콘텐츠 오류 코드 목록](#) 섹션을 참조하세요.

페이로드 사양

요청 수준 오류 메시지는 페이로드가 없습니다.

관련 리소스

- [SelectObjectContent](#)
- [GetObject](#)
- [GetBucketLifecycleConfiguration](#)
- [PutBucketLifecycleConfiguration](#)

부록: OPTIONS 객체

설명

브라우저는 이 사전 요청을 Amazon S3에 전송하여 특정 오리진, HTTP 메서드 및 헤더를 사용하여 실제 요청을 전송할 수 있는지 확인할 수 있습니다.

Amazon S3는 버킷에 cors 하위 리소스를 추가할 수 있도록 하여 교차 오리진 리소스 공유(CORS)를 지원합니다. 브라우저가 이 사전 요청을 전송하면 Amazon S3는 cors 구성에 정의된 규칙을 평가하여 응답합니다.

버킷에서 cors가 활성화되지 않은 경우 Amazon S3는 403 Forbidden 응답을 반환합니다.

CORS에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [교차 오리진 리소스 공유 활성화](#)를 참조하세요.

요청

구문

```
OPTIONS /ObjectName HTTP/1.1
Host: BucketName.s3.amazonaws.com
Origin: Origin
Access-Control-Request-Method: HTTPMethod
Access-Control-Request-Headers: RequestHeader
```

요청 파라미터

이 작업은 특정 요청 파라미터를 도입하지 않지만 실제 요청에 필요한 요청 파라미터를 포함할 수 있습니다.

요청 헤더

이름	설명	필수
Origin	Amazon S3에 대한 교차 오리진 요청의 오리진을 식별합니다. 예를 들면 http://www.example.com입니다. 유형: 문자열	예

이름	설명	필수
	기본값: None	
Access-Control-Request-Method	실제 요청에 사용할 HTTP 메서드를 식별합니다. 유형: 문자열 기본값: None	예
Access-Control-Request-Headers	실제 요청에서 전송할 HTTP 헤더의 쉼표로 구분된 목록입니다. 예를 들어 서버 측 암호화를 사용하여 객체를 넣으려면 이 사전 요청은 요청에 x-amz-server-side-encryption 헤더를 포함할 수 있는지 여부를 결정합니다. 유형: 문자열 기본값: None	아니요

요청 요소

이 작업의 구현에서는 요청 요소를 사용하지 않습니다.

응답

응답 헤더

헤더	설명
Access-Control-Allow-Origin	요청에서 전송한 오리진입니다. 요청의 오리진이 허용되지 않는 경우 Amazon S3는 응답에 이 헤더를 포함하지 않습니다. 유형: 문자열
Access-Control-Max-Age	사전 요청의 결과를 캐시할 수 있는 시간(초 단위)입니다.

헤더	설명
	유형: 문자열
Access-Control-Allow-Methods	원래 요청에서 전송된 HTTP 메서드입니다. 요청의 메서드가 허용되지 않는 경우 Amazon S3는 응답에 이 헤더를 포함하지 않습니다. 유형: 문자열
Access-Control-Allow-Headers	브라우저가 실제 요청에서 전송할 수 있는 HTTP 헤더의 쉼표로 구분된 목록입니다. 요청된 헤더가 하나라도 허용되지 않는 경우 Amazon S3는 응답에 해당 헤더를 포함하지 않으며 응답에는 Access-Control-Expose-Headers가 있는 헤더도 포함하지 않습니다. 유형: 문자열
Access-Control-Expose-Headers	HTTP 헤더의 쉼표로 구분된 목록입니다. 이 헤더는 실제 요청에 대한 응답에 포함된 이러한 헤더에 대한 액세스 권한을 JavaScript 클라이언트에 제공합니다. 유형: 문자열

응답 요소

이 작업의 구현은 응답 요소를 반환하지 않습니다.

예제

예: **cors** 활성화된 버킷에 사전 OPTIONS 요청 전송

브라우저는 이 사전 요청을 Amazon S3로 전송하여 `http://www.example.com` 오리진에서 `examplebucket`이라는 Amazon S3 버킷으로 실제 PUT 요청을 전송할 수 있는지 확인할 수 있습니다.

샘플 요청

```
OPTIONS /exampleobject HTTP/1.1
```

```
Host: examplebucket.s3.amazonaws.com
Origin: http://www.example.com
Access-Control-Request-Method: PUT
```

샘플 응답

```
HTTP/1.1 200 OK
x-amz-id-2: 6SvaESv3VULYPLik5LL171SPPtSnBvDdGmnk1X1HfU17uS2m1DF6td6KWKNjYMXZ
x-amz-request-id: BDC4B83DF5096BBE
Date: Wed, 21 Aug 2012 23:09:55 GMT
Etag: "1f1a1af1f111111111111111c11aed1da1"
Access-Control-Allow-Origin: http://www.example.com
Access-Control-Allow-Methods: PUT
Access-Control-Expose-Headers: x-amz-request-id
Content-Length: 0
Server: AmazonS3
```

관련 리소스

- [GetBucketCors](#)
- [DeleteBucketCors](#)
- [PutBucketCors](#)

부록: SOAP API

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료 (EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

이 섹션에서는 서비스, 버킷 및 객체 작업과 관련된 SOAP API에 대해 설명합니다. 인증된 SOAP 요청과 익명 SOAP 요청 모두 SSL을 사용하여 Amazon S3로 전송해야 합니다. HTTP로 SOAP 요청을 전송하면 Amazon S3가 오류를 반환합니다.

최신 Amazon S3 WSDL은 docs.aws.amazon.com/2006-03-01/AmazonS3.wSDL에서 확인할 수 있습니다.

주제

- [서비스에 대한 작업\(SOAP API\)](#)
- [버킷에 대한 작업\(SOAP API\)](#)
- [객체에 대한 작업\(SOAP API\)](#)
- [SOAP 요청 인증](#)
- [SOAP를 사용한 액세스 정책 설정](#)
- [공통 요소](#)
- [SOAP 오류 응답](#)

서비스에 대한 작업(SOAP API)

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료 (EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

이 섹션에서는 Amazon S3 서비스에서 수행할 수 있는 작업을 설명합니다.

주제

- [ListAllMyBuckets\(SOAP API\)](#)

ListAllMyBuckets(SOAP API)

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료 (EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

ListAllMyBuckets 작업은 요청의 발신자가 소유한 모든 버킷의 목록을 반환합니다.

Example

Sample Request

```
<ListAllMyBuckets xmlns="https://doc.s3.amazonaws.com/2006-03-01">
```

```
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE= </Signature>
</ListAllMyBuckets>
```

Sample Response

```
<ListAllMyBucketsResult xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <Owner>
    <ID>bcaf1ffd86f41161ca5fb16fd081034f</ID>
    <DisplayName>webfile</DisplayName>
  </Owner>
  <Buckets>
    <Bucket>
      <Name>quotes</Name>
      <CreationDate>2006-02-03T16:45:09.000Z</CreationDate>
    </Bucket>
    <Bucket>
      <Name>samples</Name>
      <CreationDate>2006-02-03T16:41:58.000Z</CreationDate>
    </Bucket>
  </Buckets>
</ListAllMyBucketsResult>
```

응답 본문

- Owner :

이 필드는 Amazon S3가 인증 및 액세스 제어를 위해 자격 증명을 표시하는 데 사용하는 정보를 제공합니다. ID는 요청한 개발자의 고유하고 영구적인 식별자입니다. DisplayName은 요청한 개발자를 나타내는 사람이 읽을 수 있는 이름입니다. 이 이름은 고유하지 않으며 시간이 지남에 따라 변경될 수 있습니다. DisplayName을 포럼 이름과 일치시키는 것이 좋습니다.

- Name :

버킷의 이름입니다. 버킷 중 하나가 최근에 삭제된 경우 삭제된 버킷의 이름이 일정 기간 이 목록에 남아 있을 수 있습니다.

- CreationDate :

버킷이 생성된 시간입니다.

액세스 통제

유효한 AWS 액세스 키 ID로 인증해야 합니다. 익명 요청은 버킷을 나열할 수 없으며 자신이 소유한 버킷만 나열할 수 있습니다.

버킷에 대한 작업(SOAP API)

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료(EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

이 섹션에서는 Amazon S3 버킷에서 수행할 수 있는 작업에 대해 설명합니다.

주제

- [CreateBucket\(SOAP API\)](#)
- [DeleteBucket\(SOAP API\)](#)
- [ListBucket\(SOAP API\)](#)
- [GetBucketAccessControlPolicy\(SOAP API\)](#)
- [SetBucketAccessControlPolicy\(SOAP API\)](#)
- [GetBucketLoggingStatus\(SOAP API\)](#)
- [SetBucketLoggingStatus\(SOAP API\)](#)

CreateBucket(SOAP API)

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료(EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

CreateBucket 작업은 버킷을 생성합니다. 모든 문자열이 허용되는 버킷 이름은 아닙니다. 버킷 이름 지정 제한 사항에 대한 자세한 내용은 [Amazon S3 버킷 작업](#)을 참조하세요.

Note

버킷 이름이 존재하는지 확인하려면 ListBucket을 사용하고 MaxKeys를 0으로 설정합니다. NoSuchBucket 응답은 버킷을 사용할 수 있음을 나타내고, AccessDenied 응답은 다른 사람이 버킷을 소유하고 있음을 나타내고, Success 응답은 버킷을 소유하고 있거나 버킷에 액세스할 권한이 있음을 나타냅니다.

Example'quotes'라는 버킷 생성**Sample Request**

```
<CreateBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</CreateBucket>
```

Sample Response

```
<CreateBucketResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <CreateBucketResponse>
    <Bucket>quotes</Bucket>
  </CreateBucketResponse>
</CreateBucketResponse>
```

요소

- **Bucket**: 생성하려는 버킷의 이름입니다.
- **AccessControlList**: 새 버킷에 대한 액세스 제어 목록입니다. 이 요소는 선택 사항입니다. 제공하지 않으면 버킷이 요청자에게 FULL_CONTROL 액세스 권한을 부여하는 액세스 정책으로 생성됩니다.

액세스 통제

유효한 AWS 액세스 키 ID로 인증해야 합니다. 익명 요청은 버킷을 생성할 수 없습니다.

관련 리소스

- [ListBucket\(SOAP API\)](#)

DeleteBucket(SOAP API)

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료 (EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

DeleteBucket 작업은 버킷을 삭제합니다. 버킷 자체를 삭제하려면 먼저 버킷의 모든 객체를 삭제해야 합니다.

Example

이 예제에서는 'quotes' 버킷을 삭제합니다.

Sample Request

```
<DeleteBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AWSAccessKeyId> AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</DeleteBucket>
```

Sample Response

```
<DeleteBucketResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <DeleteBucketResponse>
    <Code>204</Code>
    <Description>No Content</Description>
  </DeleteBucketResponse>
</DeleteBucketResponse>
```

요소

- **Bucket**: 삭제할 버킷의 이름입니다.

액세스 통제

버킷의 액세스 제어 정책에 관계없이 버킷 소유자만 버킷을 삭제할 수 있습니다.

ListBucket(SOAP API)

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료 (EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

ListBucket 작업은 버킷의 일부 항목에 대한 정보를 반환합니다.

목록 작업에 대한 일반적인 소개는 [객체 키 나열](#)을 참조하세요.

요청

이 예제에서는 'quotes' 버킷에서 접두사 'notes'가 있는 키를 최대 1,000개까지 나열합니다.

구문

```
<ListBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Prefix>notes/</Prefix>
  <Delimiter></Delimiter>
  <MaxKeys>1000</MaxKeys>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListBucket>
```

파라미터

이름	설명	필수
prefix	응답을 표시된 접두사로 시작하는 키로 제한합니다. 접두사를 사용하여 파일 시스템이 폴더를 사용하는 방식과 유사하게 버킷을 다른 키 세트로 분리할 수 있습니다. 유형: 문자열	아니요

이름	설명	필수
	기본값: None	
marker	버킷에서 나열을 시작할 위치를 나타냅니다. 이 목록에는 문자순으로 마커 뒤에 오는 키만 포함됩니다. 이는 페이지 매김에 편리합니다. 결과의 다음 페이지를 가져오려면 현재 페이지의 마지막 키를 마커로 사용합니다. 유형: 문자열 기본값: None	아니요
max-keys	응답 본문에서 보려는 최대 키 수입니다. 서버는 이보다 적은 수의 키를 반환할 수는 있지만 더 많은 키를 반환하지는 않습니다. 유형: 문자열 기본값: None	아니요
delimiter	접두사와 첫 번째 구분 기호 사이에 동일한 문자열이 포함된 키를 CommonPrefixes 모음의 단일 결과 요소로 롤업합니다. 이러한 롤업 키는 응답의 다른 곳에서 반환되지 않습니다. 유형: 문자열 기본값: None	아니요

성공 응답

이 응답은 버킷에 다음 키가 포함되어 있다고 가정합니다.

```
notes/todos.txt
notes/2005-05-23/customer_mtg_notes.txt
notes/2005-05-23/phone_notes.txt
notes/2005-05-28/sales_notes.txt
```

구문

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
```

```

<Name>backups</Name>
<Prefix>notes/</Prefix>
<MaxKeys>1000</MaxKeys>
<Delimiter>/</Delimiter>
<IsTruncated>>false</IsTruncated>
<Contents>
  <Key>notes/todos.txt</Key>
  <LastModified>2006-01-01T12:00:00.000Z</LastModified>
  <ETag>"828ef3fd9a96f00ad9f27c383fc9ac7f"</ETag>
  <Size>5126</Size>
  <StorageClass>STANDARD</StorageClass>
  <Owner>
    <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
    <DisplayName>webfile</DisplayName>
  </Owner>
  <StorageClass>STANDARD</StorageClass>
</Contents>
<CommonPrefixes>
  <Prefix>notes/2005-05-23/</Prefix>
</CommonPrefixes>
<CommonPrefixes>
  <Prefix>notes/2005-05-28/</Prefix>
</CommonPrefixes>
</ListBucketResult>

```

보다시피 응답의 많은 필드는 요청 파라미터를 그대로 반영합니다. 새로운 정보를 포함할 수 있는 유일한 응답 요소는 IsTruncated, Contents, CommonPrefixes뿐입니다.

응답 요소

이름	설명
Contents	반환된 각 객체에 대한 메타데이터입니다. 유형: XML 메타데이터 상위 항목: ListBucketResult
CommonPrefixes	delimiter 를 지정하는 경우에만 응답에 CommonPrefixes 가 포함될 수 있습니다. 그러면 CommonPrefixes 에 Prefix와 delimiter 로 지정된 문자열의 다음번 발생 지점 사이에 있는 모든 키(있는 경우)가 포함됩니다. 실제로 CommonPrefixes 는 Prefix로 지정된 디렉터리의 하위 디렉터리처

이름	설명
	럼 작동하는 키를 나열합니다. 예를 들어 prefix가 notes/이고 delimiter가 슬래시(/)인 경우 notes/summer/july 에서 공통 접두사는 notes/summer/ 입니다. 유형: 문자열 상위 항목: ListBucketResult
Delimiter	접두사와 첫 번째 구분 기호 사이에 동일한 문자열이 포함된 키를 CommonPrefixes 모음의 단일 결과 요소로 롤업합니다. 이러한 롤업 키는 응답의 다른 곳에서 반환되지 않습니다. 유형: 문자열 상위 항목: ListBucketResult
IsTruncated	모든 결과가 반환되었는지(true) 또는 반환되지 않았는지(false) 지정합니다. 결과가 MaxKeys로 지정된 수를 초과하는 경우 모든 결과가 반환되지 않을 수 있습니다. 유형: 문자열 상위 항목: 부울
Marker	버킷에서 나열을 시작할 위치를 나타냅니다. 유형: 문자열 상위 항목: ListBucketResult
MaxKeys	응답 본문에서 반환되는 최대 키 수입니다. 유형: 문자열 상위 항목: ListBucketResult

이름	설명
Name	버킷의 이름입니다. 유형: 문자열 상위 항목: ListBucketResult
Prefix	표시된 접두사로 시작하는 키입니다. 유형: 문자열 상위 항목: ListBucketResult

응답 본문

목록 응답에 대한 자세한 내용은 [키 나열 응답](#)을 참조하세요.

액세스 통제

버킷의 키를 나열하려면 버킷에 대한 READ 액세스 권한이 부여되어야 합니다.

GetBucketAccessControlPolicy(SOAP API)

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료 (EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

GetBucketAccessControlPolicy 작업은 버킷에 대한 액세스 제어 정책을 가져옵니다.

Example

이 예제에서는 'quotes' 버킷에 대한 액세스 제어 정책을 검색합니다.

Sample Request

```
<GetBucketAccessControlPolicy xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
```

```
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetBucketAccessControlPolicy>
```

Sample Response

```
<AccessControlPolicy>
  <Owner>
    <ID>a9a7b886d6fd2441bf9b1c61be666e9</ID>
    <DisplayName>chriscustomer</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6fd2441bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

응답 본문

응답에는 버킷에 대한 액세스 제어 정책이 포함됩니다. 이 응답에 대한 설명은 [SOAP 액세스 정책](#)을 참조하세요.

액세스 통제

버킷에 대한 액세스 제어 정책을 검색하려면 버킷에 대한 READ_ACP 권한이 있어야 합니다.

SetBucketAccessControlPolicy(SOAP API)

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료(EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

SetBucketAccessControlPolicy 작업은 기존 버킷에 대한 액세스 제어 정책을 설정합니다. 작업이 성공하면 버킷에 대한 이전 액세스 제어 정책이 지정된 액세스 제어 정책으로 완전히 대체됩니다.

Example

지정된 사용자(일반적으로 소유자)에게 'quotes' 버킷에 대한 FULL_CONTROL 액세스 권한을 부여합니다.

Sample Request

```
<SetBucketAccessControlPolicy xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b8863000e241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE= </Signature>
</SetBucketAccessControlPolicy >
```

Sample Response

```
<GetBucketAccessControlPolicyResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <GetBucketAccessControlPolicyResponse>
    <Code>200</Code>
    <Description>OK</Description>
  </GetBucketAccessControlPolicyResponse>
</GetBucketAccessControlPolicyResponse>
```

액세스 통제

버킷에 대한 액세스 제어 정책을 설정하려면 버킷에 대한 WRITE_ACP 권한이 있어야 합니다.

GetBucketLoggingStatus(SOAP API)

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료 (EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

GetBucketLoggingStatus는 기존 버킷의 로깅 상태를 검색합니다.

이 기능에 대한 일반적인 소개는 [서버 로그](#)를 참조하세요.

Example

Sample Request

```
<?xml version="1.0" encoding="utf-8"?>
  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
    <soap:Body>
      <GetBucketLoggingStatus xmlns="https://doc.s3.amazonaws.com/2006-03-01">
        <Bucket>mybucket</Bucket>
        <AWSAccessKeyId>YOUR_AWS_ACCESS_KEY_ID</AWSAccessKeyId>
        <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
        <Signature>YOUR_SIGNATURE_HERE</Signature>
      </GetBucketLoggingStatus>
    </soap:Body>
  </soap:Envelope>
```

Sample Response

```
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" >
    <soapenv:Header>
    </soapenv:Header>
    <soapenv:Body>
      <GetBucketLoggingStatusResponse xmlns="https://s3.amazonaws.com/
doc/2006-03-01">
```

```

    <GetBucketLoggingStatusResponse>
      <LoggingEnabled>
        <TargetBucket>mylogs</TargetBucket>
        <TargetPrefix>mybucket-access_log-</TargetPrefix>
      </LoggingEnabled>
    </GetBucketLoggingStatusResponse>
  </GetBucketLoggingStatusResponse>
</soapenv:Body>
</soapenv:Envelope>

```

액세스 통제

버킷 소유자만 이 작업을 간접적으로 호출할 수 있습니다.

SetBucketLoggingStatus(SOAP API)

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료 (EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

SetBucketLoggingStatus 작업은 기존 버킷의 로깅 상태를 업데이트합니다.

이 기능에 대한 일반적인 소개는 [서버 로그](#)를 참조하세요.

Example

이 샘플 요청은 'mybucket' 버킷에 대한 서버 액세스 로깅을 활성화하고 접두사 'access_log-' 아래의 'mylogs'로 전달되도록 로그를 구성합니다.

Sample Request

```

<?xml version="1.0" encoding="utf-8"?>
  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
    www.w3.org/2001/XMLSchema">
    <soap:Body>
      <SetBucketLoggingStatus xmlns="https://doc.s3.amazonaws.com/2006-03-01">
        <Bucket>myBucket</Bucket>
        <AWSAccessKeyId>YOUR_AWS_ACCESS_KEY_ID</AWSAccessKeyId>
      </SetBucketLoggingStatus>
    </soap:Body>
  </soap:Envelope>

```



```

<Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
<Signature>YOUR_SIGNATURE_HERE</Signature>
<BucketLoggingStatus>
  <LoggingEnabled>
    <TargetBucket>mylogs</TargetBucket>
    <TargetPrefix>mybucket-access_log-</TargetPrefix>
  </LoggingEnabled>
</BucketLoggingStatus>
</SetBucketLoggingStatus>
</soap:Body>
:</soap:Envelope>

```

Sample Response

```

<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" >
    <soapenv:Header>
    </soapenv:Header>
    <soapenv:Body>
      <SetBucketLoggingStatusResponse xmlns="https://s3.amazonaws.com/
doc/2006-03-01"/>
    </soapenv:Body>
  </soapenv:Envelope>

```

액세스 통제

버킷 소유자만 이 작업을 간접적으로 호출할 수 있습니다.

객체에 대한 작업(SOAP API)

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료 (EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

이 섹션에서는 Amazon S3 객체에서 수행할 수 있는 작업에 대해 설명합니다.

주제

- [PutObjectInline\(SOAP API\)](#)
- [PutObject\(SOAP API\)](#)
- [CopyObject\(SOAP API\)](#)
- [GetObject\(SOAP API\)](#)
- [GetObjectExtended\(SOAP API\)](#)
- [DeleteObject\(SOAP API\)](#)
- [GetObjectAccessControlPolicy\(SOAP API\)](#)
- [SetObjectAccessControlPolicy\(SOAP API\)](#)

PutObjectInline(SOAP API)

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료 (EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

PutObjectInline 작업은 버킷에 객체를 추가합니다. 객체에 대한 데이터는 SOAP 메시지의 본문에 제공됩니다.

버킷에 객체가 이미 있는 경우 새 객체가 기존 객체를 덮어씁니다. Amazon S3는 마지막 쓰기 요청을 저장하기 때문입니다. 하지만 Amazon S3는 분산 시스템입니다. Amazon S3가 거의 동시에 동일한 객체에 대해 여러 개의 쓰기 요청을 수신하면 최종적으로 한 요청만 성공하더라도 모든 객체가 저장될 수 있습니다. Amazon S3는 객체 잠금을 제공하지 않습니다. 이 기능이 필요한 경우 애플리케이션 계층에 구축해야 합니다.

객체가 네트워크를 통해 손상되지 않도록 하려면 객체의 MD5를 계산하여 Amazon S3에 PUT하고 반환된 Etag를 계산된 MD5 값과 비교할 수 있습니다.

PutObjectInline은 대형 객체에 사용하기에 적합하지 않습니다. 시스템은 이 작업을 1MB 이하의 객체에 사용하도록 제한합니다. 데이터 파라미터가 1MB보다 큰 객체를 인코딩하는 경우 PutObjectInline은 InlineDataTooLargeError 상태 코드와 함께 실패합니다. 대형 객체를 업로드하려면 대신 비 인라인 PutObject API 또는 REST API를 사용하는 것이 좋습니다.

Example

이 예제에서는 일부 텍스트 및 메타데이터를 'quotes' 버킷의 'Nelson' 객체에 쓰고, 사용자(일반적으로 소유자)에게 객체에 대한 FULL_CONTROL 액세스 권한을 부여하고, 익명 사용자가 객체를 읽을 수 있도록 설정합니다.

Sample Request

```
<PutObjectInline xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
    <Name>family</Name>
    <Value>Muntz</Value>
  </Metadata>
  <Data>aGEtaGE=</Data>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6fde241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObjectInline>
```

Sample Response

```
<PutObjectInlineResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectInlineResponse>
    <ETag>&quot;828ef3fd96f00ad9f27c383fc9ac7f&quot;</ETag>
    <LastModified>2006-01-01T12:00:00.000Z</lastModified>
  </PutObjectInlineResponse>
</PutObjectInlineResponse>
```

요소

- **Bucket:** 객체를 추가할 버킷입니다.
- **Key:** 객체에 할당할 키입니다.
- **Metadata:** 메타데이터 요소에서 이름-값 메타데이터 쌍을 제공할 수 있습니다. 이는 객체와 함께 저장됩니다.
- **Data:** 데이터의 Base64로 인코딩된 형식입니다.
- **ContentLength:** 데이터의 바이트 단위 길이입니다.
- **AccessControlList:** 리소스에 대한 액세스 제어 목록입니다. 이 요소는 선택 사항입니다. 이 요소를 생략하면 요청자에게 객체에 대한 FULL_CONTROL 액세스 권한이 부여됩니다. 객체가 이미 있는 경우 기존 액세스 제어 정책이 대체됩니다.

응답

- **ETag:** 엔터티 태그는 GetObjectExtended를 사용하여 객체의 조건부 가져오기를 수행하는 데 사용할 수 있는 객체의 MD5 해시입니다. ETag는 객체의 콘텐츠에 대한 변경 사항만 반영하고 메타데이터에 대한 변경을 반영하지 않습니다.
- **LastModified:** 저장된 객체의 Amazon S3 타임스탬프입니다.

액세스 통제

객체를 버킷에 넣으려면 버킷에 대한 WRITE 액세스 권한이 있어야 합니다.

관련 리소스

- [PutObject\(SOAP API\)](#)

- [CopyObject\(SOAP API\)](#)

PutObject(SOAP API)

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료 (EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

PutObject 작업은 버킷에 객체를 추가합니다. 객체에 대한 데이터는 DIME 첨부 파일로 첨부됩니다.

객체가 네트워크를 통해 손상되지 않도록 하려면 객체의 MD5를 계산하여 Amazon S3에 PUT하고 반환된 Etag를 계산된 MD5 값과 비교할 수 있습니다.

버킷에 객체가 이미 있는 경우 새 객체가 기존 객체를 덮어씁니다. Amazon S3는 마지막 쓰기 요청을 저장하기 때문입니다. 하지만 Amazon S3는 분산 시스템입니다. Amazon S3가 거의 동시에 동일한 객체에 대해 여러 개의 쓰기 요청을 수신하면 최종적으로 한 요청만 성공하더라도 모든 객체가 저장될 수 있습니다. Amazon S3는 객체 잠금을 제공하지 않습니다. 이 기능이 필요한 경우 애플리케이션 계층에 구축해야 합니다.

Example

이 예제에서는 일부 데이터 및 메타데이터를 'quotes' 버킷의 'Nelson' 객체에 넣고, 사용자(일반적으로 소유자)에게 객체에 대한 FULL_CONTROL 액세스 권한을 부여하고, 익명 사용자가 객체를 읽을 수 있도록 설정합니다. 이 샘플에서는 실제 첨부 파일이 표시되지 않습니다.

Sample Request

```
<PutObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
    <Name>family</Name>
    <Value>Muntz</Value>
  </Metadata>
```

```

<ContentLength>5</ContentLength>
<AccessControlList>
  <Grant>
    <Grantee xsi:type="CanonicalUser">
      <ID>a9a7b886d6241bf9b1c61be666e9</ID>
      <DisplayName>chriscustomer</DisplayName>
    </Grantee>
    <Permission>FULL_CONTROL</Permission>
  </Grant>
  <Grant>
    <Grantee xsi:type="Group">
      <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
    </Grantee>
    <Permission>READ</Permission>
  </Grant>
</AccessControlList>
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2007-05-11T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObject>

```

Sample Response

```

<PutObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2006-03-01T12:00:00.183Z</LastModified>
  </PutObjectResponse>
</PutObjectResponse>

```

요소

- **Bucket**: 객체를 추가할 버킷입니다.
- **Key**: 객체에 할당할 키입니다.
- **Metadata**: 메타데이터 요소에서 이름-값 메타데이터 쌍을 제공할 수 있습니다. 이는 객체와 함께 저장됩니다.
- **ContentLength**: 데이터의 바이트 단위 길이입니다.
- **AccessControlList**: 리소스에 대한 액세스 제어 목록입니다. 이 요소는 선택 사항입니다. 이 요소를 생략하면 요청자에게 객체에 대한 FULL_CONTROL 액세스 권한이 부여됩니다. 객체가 이미 있는 경우 기존 액세스 제어 정책이 대체됩니다.

응답

- **ETag**: 엔터티 태그는 `GetObjectExtended`를 사용하여 객체의 조건부 가져오기를 수행하는 데 사용할 수 있는 객체의 MD5 해시입니다. ETag는 객체의 콘텐츠에 대한 변경 사항만 반영하고 메타데이터에 대한 변경을 반영하지 않습니다.
- **LastModified**: 저장된 객체의 Amazon S3 타임스탬프입니다.

액세스 통제

객체를 버킷에 넣으려면 버킷에 대한 `WRITE` 액세스 권한이 있어야 합니다.

관련 리소스

- [CopyObject\(SOAP API\)](#)

CopyObject(SOAP API)

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료(EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

설명

소스 객체의 키 및 버킷과 대상의 키 및 버킷을 지정하면 `CopyObject` 작업이 객체의 복사본을 생성합니다.

객체를 복사할 때 모든 메타데이터를 보존(기본값)하거나 새 메타데이터를 지정할 수 있습니다. 그러나 ACL은 보존되지 않으며 요청을 하는 사용자에게 대해 `private`으로 설정됩니다. 기본 ACL 설정을 재정의하려면 복사 요청을 생성할 때 새 ACL을 지정합니다. 자세한 내용은 [ACL 사용](#)을 참조하십시오.

모든 복사 요청은 인증되어야 합니다. 또한 소스 객체에 대한 읽기 액세스 권한과 대상 버킷에 대한 쓰기 액세스 권한이 있어야 합니다. 자세한 내용은 [인증 액세스 사용](#)을 참조하세요.

Etag가 일치하는지 여부 또는 객체가 지정된 날짜 이전 또는 이후에 수정되었는지 여부와 같은 특정 조건에서만 객체를 복사하려면 요청 파라미터 `CopySourceIfUnmodifiedSince`, `CopyIfUnmodifiedSince`, `CopySourceIfMatch` 또는 `CopySourceIfNoneMatch`를 사용합니다.

 Note

대형 객체를 복사하기 위해 SOAP 스택 소켓 제한 시간을 구성해야 할 수 있습니다.

구문 요청

```
<CopyObject xmlns="http://bucket_name.s3.amazonaws.com/2006-03-01">
  <SourceBucket>source_bucket</SourceBucket>
  <SourceObject>source_object</SourceObject>
  <DestinationBucket>destination_bucket</DestinationBucket>
  <DestinationObject>destination_object</DestinationObject>
  <MetadataDirective>{REPLACE | COPY}</MetadataDirective>
  <Metadata>
    <Name>metadata_name</Name>
    <Value>metadata_value</Value>
  </Metadata>
  ...
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="user_type">
        <ID>user_id</ID>
        <DisplayName>display_name</DisplayName>
      </Grantee>
      <Permission>permission</Permission>
    </Grant>
    ...
  </AccessControlList>
  <CopySourceIfMatch>etag</CopySourceIfMatch>
  <CopySourceIfNoneMatch>etag</CopySourceIfNoneMatch>
  <CopySourceIfModifiedSince>date_time</CopySourceIfModifiedSince>
  <CopySourceIfUnmodifiedSince>date_time</CopySourceIfUnmodifiedSince>
  <AWSAccessKeyId>AWSAccessKeyId</AWSAccessKeyId>
  <Timestamp>TimeStamp</Timestamp>
  <Signature>Signature</Signature>
</CopyObject>
```


요청 파라미터

이름	설명	필수
SourceBucket	소스 버킷의 이름입니다. 유형: 문자열 기본값: None 제약 조건: 유효한 소스 버킷.	예
SourceKey	소스 객체의 키 이름입니다. 유형: 문자열 기본값: None 제약 조건: READ 액세스 권한이 있는 유효한 소스 객체의 키.	예
DestinationBucket	대상 버킷의 이름입니다. 유형: 문자열 기본값: None 제약 조건: 대상 버킷에 대한 WRITE 액세스 권한이 있어야 합니다.	예
DestinationKey	대상 객체의 키입니다. 유형: 문자열 기본값: None 제약 조건: 대상 버킷에 대한 WRITE 액세스 권한이 있어야 합니다.	예
MetadataDirective	메타데이터가 소스 객체에서 복사되는지 아니면 요청에 제공된 메타데이터로 대체되는지 지정합니다.	아니요

이름	설명	필수
	유형: 문자열 기본값: COPY 유효한 값: COPY REPLACE 제약 조건: COPY 또는 REPLACE 이외의 값은 즉시 오류가 발생합니다. MetadataDirective 헤더를 지정하고 해당 값을 REPLACE로 설정하지 않으면 객체를 자체로 복사할 수 없습니다.	
Metadata	객체에 설정할 메타데이터 이름-값 쌍을 지정합니다. MetadataDirective가 COPY로 설정된 경우 모든 메타데이터가 무시됩니다. 유형: 문자열 기본값: None 제약 조건: 없음.	아니요
AccessControlList	이메일 주소 또는 정식 사용자 ID를 기준으로 사용자에게 액세스 권한을 부여합니다. 유형: 문자열 기본값: None 제약 조건: 없음	아니요

이름	설명	필수
CopySourceIfMatch	객체의 엔터티 태그(ETag)가 지정된 태그와 일치하면 객체를 복사하고, 그렇지 않으면 PreconditionFailed를 반환합니다. 유형: 문자열 기본값: None 제약 조건: 없음. Etag가 일치하지 않으면 객체가 복사되지 않습니다.	아니요
CopySourceIfNoneMatch	객체의 엔터티 태그(ETag)가 지정된 Etag와 다르면 객체를 복사하고, 그렇지 않으면 오류를 반환합니다. 유형: 문자열 기본값: None 제약 조건: 없음.	아니요
CopySourceIfUnmodifiedSince	객체가 지정된 시간 이후에 수정되지 않았으면 객체를 복사하고, 그렇지 않으면 PreconditionFailed를 반환합니다. 유형: 날짜/시간 기본값: None	아니요
CopySourceIfModifiedSince	객체가 지정된 시간 이후에 수정되었으면 객체를 복사하고, 그렇지 않으면 오류를 반환합니다. 유형: 날짜/시간 기본값: None	아니요

Response Syntax

```
<CopyObjectResponse xmlns="http://bucket_name.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"etag"</ETag>
    <LastModified>timestamp</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

Response Elements

다음은 응답 요소의 목록입니다.

Note

SOAP API는 추가 공백을 반환하지 않습니다. 추가 공백은 REST API에서만 반환됩니다.

이름	설명
Etag	새 객체의 etag를 반환합니다. ETag는 객체의 콘텐츠에 대한 변경 사항만 반영하고 메타데이터에 대한 변경을 반영하지 않습니다. 유형: 문자열 상위 항목: CopyObjectResult
LastModified	객체가 마지막으로 수정된 날짜를 반환합니다. 유형: 문자열 상위 항목: CopyObjectResult

일반 응답 요소에 대한 자세한 내용은 [REST 오류 응답 헤더 사용](#)을 참조하세요.

특수 오류

이 작업에는 특수 오류가 없습니다. Amazon S3 오류에 관한 일반 정보는 [오류 코드 목록](#) 섹션을 참조하세요.

예제

이 예제는 메타데이터를 유지하면서 flotsam 버킷의 pacific 객체를 jetsam 버킷의 atlantic 객체로 복사합니다.

샘플 요청

```
<CopyObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <SourceBucket>pacific</SourceBucket>
  <SourceObject>flotsam</SourceObject>
  <DestinationBucket>atlantic</DestinationBucket>
  <DestinationObject>jetsam</DestinationObject>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2008-02-18T13:54:10.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbq7RrtSFmw=</Signature>
</CopyObject>
```

샘플 응답

```
<CopyObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2008-02-18T13:54:10.183Z</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

이 예제에서는 wonderland 버킷의 'tweedledee' 객체를 wonderland 버킷의 'tweedledum' 객체로 복사하여 해당 메타데이터를 대체합니다.

샘플 요청

```
<CopyObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <SourceBucket>wonderland</SourceBucket>
  <SourceObject>tweedledee</SourceObject>
  <DestinationBucket>wonderland</DestinationBucket>
  <DestinationObject>tweedledum</DestinationObject>
  <MetadataDirective>REPLACE</MetadataDirective>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
```

```
<Name>relationship</Name>
<Value>twins</Value>
</Metadata>
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2008-02-18T13:54:10.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbq7RrtSFmw=</Signature>
</CopyObject>
```

샘플 응답

```
<CopyObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"828ef3fd96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2008-02-18T13:54:10.183Z</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

관련 리소스

- [PutObject\(SOAP API\)](#)
- [PutObjectInline\(SOAP API\)](#)

GetObject(SOAP API)

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료 (EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

GetObject 작업은 객체의 현재 버전을 반환합니다. 삭제 마커가 현재 버전인 객체에 대해 GetObject를 시도하면 S3가 404 오류를 반환합니다. 이 SOAP API를 사용하여 객체의 지정된 버전을 검색할 수 없습니다. 이를 위해서는 REST API를 사용합니다. 자세한 내용은 [버전 관리](#)를 참조하십시오. 추가 옵션을 선택하려면 [GetObjectExtended\(SOAP API\)](#) 작업을 사용합니다.

Note

값이 "soap"인 객체 키 이름은 [가상 호스팅 스타일 요청](#)에 지원되지 않습니다. "soap"이 사용되는 객체 키 이름 값의 경우 [경로 스타일 URL](#)을 대신 사용해야 합니다.

Example

이 예제에서는 'quotes' 버킷에서 'Nelson' 객체를 가져옵니다.

Sample Request

```
<GetObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>
```

Sample Response

```
<GetObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
    </Status>
    <Metadata>
      <Name>Content-Type</Name>
      <Value>text/plain</Value>
    </Metadata>
    <Metadata>
      <Name>family</Name>
      <Value>Muntz</Value>
    </Metadata>
    <Data>aGEtaGE=</Data>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
  </GetObjectResponse>
</GetObjectResponse>
```

요소

- **Bucket:** 객체를 검색할 버킷입니다.

- Key: 객체를 식별하는 키입니다.
- GetMetadata: true인 경우 메타데이터가 객체와 함께 반환됩니다.
- GetData: true인 경우 객체 데이터가 반환됩니다.
- InlineData: true인 경우 데이터가 응답의 SOAP 본문의 일부로 Base64로 인코딩된 상태로 반환됩니다. false인 경우 데이터가 SOAP 첨부 파일로 반환됩니다. InlineData 옵션은 대형 객체에 사용하기에 적합하지 않습니다. 시스템은 이 작업을 1MB 이하의 데이터에 사용하도록 제한합니다. InlineData 플래그가 설정된 GetObject 요청은 결과 데이터 파라미터의 인코딩 후 크기가 1MB를 초과하게 될 경우 InlineDataTooLargeError 상태 코드와 함께 실패합니다. 대형 객체를 다운로드하려면 InlineData 플래그를 설정하지 않고 GetObject를 직접적으로 호출하거나 대신 REST API를 사용하는 것이 좋습니다.

반환된 요소

- Metadata: 객체와 함께 저장된 이름-값 쌍 메타데이터입니다.
- Data: 요청에서 InlineData가 true인 경우 여기에는 Base64로 인코딩된 객체 데이터가 포함됩니다.
- LastModified: 객체가 Amazon S3에 저장된 시간입니다.
- ETag: 객체의 엔터티 태그입니다. 조건부 가져오기를 수행하는 데 사용할 수 있는 객체의 해시입니다. ETag는 객체의 콘텐츠에 대한 변경 사항만 반영하고 메타데이터에 대한 변경을 반영하지 않습니다.

액세스 통제

객체에 대한 READ 액세스 권한이 부여된 경우에만 객체를 읽을 수 있습니다.

SOAP 청크 다운로드 및 재개 가능 다운로드

GET 유연성을 제공하기 위해 Amazon S3는 청크 다운로드와 재개 가능한 다운로드를 지원합니다.

다음 중에서 선택합니다.

- 대형 객체 다운로드의 경우 객체를 작은 청크로 분할해야 할 수 있습니다. 자세한 내용은 [범위 GET](#) 섹션을 참조하세요.
- GET 작업이 실패하는 경우 전체 파일 대신 나머지 파일을 다운로드하도록 애플리케이션을 설계할 수 있습니다. 자세한 내용은 [REST GET 오류 복구](#) 섹션을 참조하세요.

범위 GET

일부 클라이언트의 경우 대용량 다운로드를 더 작은 다운로드로 분할할 수 있습니다. GET을 더 작은 단위로 분할하려면 범위를 사용합니다.

GET을 더 작은 단위로 분할하려면 먼저 크기를 결정해야 합니다. 예를 들어 다음 요청은 bigfile 객체의 크기를 가져옵니다.

```
<ListBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
  <Prefix>bigfile</Prefix>
  <MaxKeys>1</MaxKeys>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListBucket>
```

Amazon S3는 다음 응답을 반환합니다.

```
<ListBucketResult xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <Name>quotes</Name>
  <Prefix>N</Prefix>
  <MaxKeys>1</MaxKeys>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>bigfile</Key>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <Size>2023276</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>bcaflfffd86f41161ca5fb16fd081034f</ID>
      <DisplayName>bigfile</DisplayName>
    </Owner>
  </Contents>
</ListBucketResult>
```

다음은 bigfile 객체에서 첫 번째 메가바이트를 다운로드하는 요청입니다.

```
<GetObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
  <Key>bigfile</Key>
```

```

<GetMetadata>true</GetMetadata>
<GetData>true</GetData>
<InlineData>true</InlineData>
<ByteRangeStart>0</ByteRangeStart>
<ByteRangeEnd>1048576</ByteRangeEnd>
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>

```

Amazon S3는 파일의 첫 번째 메가바이트와 파일의 Etag를 반환합니다.

```

<GetObjectResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
    </Status>
    <Metadata>
      <Name>Content-Type</Name>
      <Value>text/plain</Value>
    </Metadata>
    <Metadata>
      <Name>family</Name>
      <Value>Muntz</Value>
    </Metadata>
    <Data>--first megabyte of bigfile--</Data>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fd96f00ad9f27c383fc9ac7f"</ETag>
  </GetObjectResponse>
</GetObjectResponse>

```

이전 부분이 다운로드된 이후 파일이 변경되지 않았는지 확인하려면 IfMatch 요소를 지정합니다. IfMatch 요소가 필수는 아니지만 변경 가능성이 있는 콘텐츠에는 권장됩니다.

다음은 IfMatch 요청 헤더를 사용하여 파일의 나머지 부분을 가져오는 요청입니다.

```

<GetObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
  <Key>bigfile</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>

```

```

<ByteRangeStart>10485761</ByteRangeStart>
<ByteRangeEnd>2023276</ByteRangeEnd>
<IfMatch>"828ef3fdfa96f00ad9f27c383fc9ac7f"</IfMatch>
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>

```

Amazon S3는 다음 응답과 함께 파일의 나머지 부분을 반환합니다.

```

<GetObjectResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
    </Status>
    <Metadata>
      <Name>Content-Type</Name>
      <Value>text/plain</Value>
    </Metadata>
    <Metadata>
      <Name>family</Name>
      <Value>>Muntz</Value>
    </Metadata>
    <Data>--remainder of bigfile--</Data>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
  </GetObjectResponse>
</GetObjectResponse>

```

버전이 지정된 GetObject

다음 요청은 버킷에 있는 객체의 지정된 버전을 반환합니다.

```

<GetObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>

```

```
</GetObject>
```

샘플 응답

```
<GetObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
    </Status>
    <Metadata>
      <Name>Content-Type</Name>
      <Value>text/plain</Value>
    </Metadata>
    <Metadata>
      <Name>family</Name>
      <Value>Muntz</Value>
    </Metadata>
    <Data>aGEtaGE=</Data>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
  </GetObjectResponse>
</GetObjectResponse>
```

REST GET 오류 복구

객체 GET이 실패하면 다운로드할 범위를 지정하여 파일의 나머지 부분을 가져올 수 있습니다. 이렇게 하려면 ListBucket을 사용하여 객체의 크기를 가져오고 파일의 나머지 부분에서 범위 GET을 수행해야 합니다. 자세한 내용은 [GetObjectExtended\(SOAP API\)](#) 섹션을 참조하세요.

관련 리소스

[객체에 대한 작업\(SOAP API\)](#)

GetObjectExtended(SOAP API)

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료 (EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

GetObjectExtended는 HTTP GET 헤더(<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html> 참조)에서 제공하는 기능을 대부분 수행하는 데 사용할 수 있는 다음과 같은 추가 요소를 지원한다는 점을 제외하면 [GetObject\(SOAP API\)](#)와 동일합니다.

GetObjectExtended는 GetObject에서 지원하는 것 외에도 다음 요소를 지원합니다.

- **ByteRangeStart, ByteRangeEnd**: 이러한 요소는 객체 데이터의 일부만 검색하도록 지정합니다. 또한 HTTP 바이트 범위의 동작을 따릅니다(<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35> 참조).
- **IfModifiedSince**: 객체의 타임스탬프가 지정된 타임스탬프보다 늦은 경우에만 객체를 반환합니다(<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.25>).
- **IfUnmodifiedSince**: 객체의 타임스탬프가 지정된 타임스탬프보다 빠르거나 같은 경우에만 객체를 반환합니다(<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.28> 참조).
- **IfMatch**: ETag가 제공된 태그와 일치하는 경우에만 객체를 반환합니다(<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.24> 참조).
- **IfNoneMatch**: ETag가 제공된 태그와 일치하지 않는 경우에만 객체를 반환합니다(<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.26> 참조).
- **ReturnCompleteObjectOnConditionFailure**: ReturnCompleteObjectOnConditionFailure: true인 경우 요청에 범위 요소와 IfUnmodifiedSince/IfMatch 요소 중 하나 또는 둘 다 포함되고 조건이 실패하면 오류가 아닌 전체 객체를 반환합니다. 그러면 If-Range 기능이 활성화됩니다(<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.27> 참조).

DeleteObject(SOAP API)

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료(EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

DeleteObject 작업은 Amazon S3에서 지정된 객체를 제거합니다. 일단 삭제된 객체를 복원하거나 삭제 취소하는 방법은 없습니다.

Note

존재하지 않는 객체를 삭제하면 Amazon S3는 성공을 반환합니다(오류 메시지가 아님).

Example

이 예제에서는 'quotes' 버킷에서 'Nelson' 객체를 삭제합니다.

Sample Request

```
<DeleteObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <AWSAccessKeyId> AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</DeleteObject>
```

Sample Response

```
<DeleteObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <DeleteObjectResponse>
    <Code>200</Code>
    <Description>OK</Description>
  </DeleteObjectResponse>
</DeleteObjectResponse>
```

요소

- Bucket: 객체를 보관하는 버킷입니다.
- Key: 객체를 식별하는 키입니다.

액세스 통제

객체의 소유자 또는 객체에 부여된 권한에 관계없이 버킷에 대한 WRITE 액세스 권한이 있는 경우에만 객체를 삭제할 수 있습니다.

GetObjectAccessControlPolicy(SOAP API)

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료 (EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

GetObjectAccessControlPolicy 작업은 객체에 대한 액세스 제어 정책을 가져옵니다.

Example

이 예제에서는 'quotes' 버킷에서 'Nelson' 객체에 대한 액세스 제어 정책을 검색합니다.

Sample Request

```
<GetObjectAccessControlPolicy xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObjectAccessControlPolicy>
```

Sample Response

```
<AccessControlPolicy>
  <Owner>
    <ID>a9a7b886d6fd24a541bf9b1c61be666e9</ID>
    <DisplayName>chriscustomer</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b841bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

응답 본문

응답에는 버킷에 대한 액세스 제어 정책이 포함됩니다. 이 응답에 대한 설명은 [SOAP 액세스 정책](#)을 참조하세요.

액세스 통제

객체에 대한 액세스 제어 정책을 검색하려면 객체에 대한 READ_ACP 권한이 있어야 합니다.

SetObjectAccessControlPolicy(SOAP API)

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료(EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

SetObjectAccessControlPolicy 작업은 기존 객체에 대한 액세스 제어 정책을 설정합니다. 작업이 성공하면 객체에 대한 이전 액세스 제어 정책이 지정된 액세스 제어 정책으로 완전히 대체됩니다.

Example

이 예제에서는 지정된 사용자(일반적으로 소유자)에게 'quotes' 버킷의 'Nelson' 객체에 대한 FULL_CONTROL 액세스 권한을 부여합니다.

Sample Request

```
<SetObjectAccessControlPolicy xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
</SetObjectAccessControlPolicy>
```



```
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</SetObjectAccessControlPolicy>
```

Sample Response

```
<SetObjectAccessControlPolicyResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <SetObjectAccessControlPolicyResponse>
    <Code>200</Code>
    <Description>OK</Description>
  </SetObjectAccessControlPolicyResponse>
</SetObjectAccessControlPolicyResponse>
```

키

액세스 통제

버킷에 대한 액세스 제어 정책을 설정하려면 객체에 대한 WRITE_ACP 권한이 있어야 합니다.

SOAP 요청 인증

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료 (EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

모든 비익명 요청에는 요청하는 보안 주체의 자격 증명을 설정하기 위한 인증 정보가 반드시 포함되어야 합니다. SOAP에서 인증 정보는 다음과 같은 SOAP 요청 요소로 표현됩니다.

- AWS 액세스 키 ID

Note

인증된 SOAP 요청 시, 임시 보안 자격 증명은 지원되지 않습니다. 자격 증명 유형에 대한 자세한 정보는 [요청 만들기](#)를 참조하십시오.

- **Timestamp:** 반드시 2009-01-01T12:00:00.000Z와 같은 협정 세계시(그리니치 표준시) 시간 대의 날짜/시간(<http://www.w3.org/TR/xmlschema-2/#dateTime> 참조)으로 설정해야 합니다. 타임스탬프가 Amazon S3 서버와 15분 이상 차이가 나면 권한 부여가 실패합니다.

- **Signature:** AWS 비밀 액세스 키를 키로 사용하는 "AmazonS3" + OPERATION + 타임스탬프로 연속된 값의 RFC 2104 HMAC-SHA1 다이제스트(<http://www.ietf.org/rfc/rfc2104.txt> 참조). 예를 들어, 다음 CreateBucket 샘플 요청에서 서명 요소에는 "AmazonS3CreateBucket2009-01-01T12:00:00.000Z" 값의 HMAC-SHA1 다이제스트가 포함됩니다.

예를 들어, 다음 CreateBucket 샘플 요청에서 서명 요소에는 "AmazonS3CreateBucket2009-01-01T12:00:00.000Z" 값의 HMAC-SHA1 다이제스트가 포함됩니다.

Example

```
<CreateBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Acl>private</Acl>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2009-01-01T12:00:00.000Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</CreateBucket>
```

Note

인증된 SOAP 요청 및 익명 SOAP 요청은 SSL을 사용하여 Amazon S3로 보내야 합니다. HTTP로 SOAP 요청을 전송하면 Amazon S3가 오류를 반환합니다.

Important

시간 정밀도의 유효 자릿수에 대한 해석이 다양하기 때문에 .NET 사용자는 너무 자세한 타임스탬프를 Amazon S3로 보내지 않도록 합니다. 밀리초 단위만 사용해 수동으로 DateTime 객체를 생성하면 됩니다.

SOAP를 사용한 액세스 정책 설정

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료(EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

액세스 제어는 "AccessControlList" 요소를 CreateBucket, PutObjectInline 또는 PutObject에 대한 요청에 포함하여 버킷 또는 객체가 기록될 때 설정할 수 있습니다. AccessControlList 요소는 [Amazon S3의 자격 증명 및 액세스 관리](#)에 설명되어 있습니다. 이러한 작업으로 지정된 액세스 통제 목록이 없는 경우, 리소스는 요청자에게 FULL_CONTROL 액세스를 제공하는 기본 액세스 정책으로 생성됩니다(이는 요청이 이미 존재하는 객체에 대한 PutObjectInline 또는 PutObject 요청인 경우에도 마찬가지임).

다음은 데이터를 객체에 쓰는 요청으로, 익명 보안 주체가 객체를 읽기 가능하도록 만들며 특정 사용자에게 버킷에 대한 FULL_CONTROL 권한을 부여합니다. (대다수 개발자는 버킷에 스스로 FULL_CONTROL 접근을 부여하는 것을 희망합니다.)

Example

다음은 데이터를 객체에 쓰는 요청으로, 익명 보안 주체가 객체를 읽기 가능하도록 만듭니다.

Sample Request

```
<PutObjectInline xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Data>aGEtaGE=</Data>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
```

```
<Timestamp>2009-03-01T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObjectInline>
```

Sample Response

```
<PutObjectInlineResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectInlineResponse>
    <ETag>"828ef3fd96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2009-01-01T12:00:00.000Z</LastModified>
  </PutObjectInlineResponse>
</PutObjectInlineResponse>
```

GetBucketAccessControlPolicy, GetObjectAccessControlPolicy, SetBucketAccessControlPolicy 및 SetObjectAccessControlPolicy 메서드를 사용하여 액세스 제어 정책을 읽거나 기존 버킷 또는 객체에 대해 이를 설정할 수 있습니다. 자세한 내용은 이러한 메서드에 관한 세부적인 설명을 참조하십시오.

공통 요소

다음 권한 부여 관련 요소를 SOAP 요청에 포함할 수 있습니다.

- **AWSSessionToken**: 요청자의 AWS 액세스 키 ID
- **Timestamp**: 시스템의 현재 시간
- **Signature**: 요청에 대한 서명

SOAP 오류 응답

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료 (EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

SOAP에서 오류 결과는 HTTP 응답 코드 500과 함께 SOAP 오류로 클라이언트에 반환됩니다. SOAP 오류가 반환되지 않으면 요청이 성공한 것입니다. Amazon S3 SOAP 오류 코드는 Amazon S3 전용 오류 코드와 연결된 표준 SOAP 1.1 오류 코드('Server' 또는 'Client')로 구성되어 있습니다(예: "Server.InternalError" 또는 "Client.NoSuchBucket"). SOAP 오류 문자열 요소에는 사람이 읽을 수 있는

일반적인 오류 메시지가 영어로 담겨 있습니다. 마지막으로 SOAP 오류 세부 요소에는 오류와 관련된 기타 정보가 포함됩니다.

예를 들어, 존재하지 않는 "Fred" 객체를 제거하려고 시도하면 SOAP 응답의 본문에 "NoSuchKey" SOAP 오류가 포함됩니다.

다음 예제는 샘플 SOAP 오류 응답을 보여줍니다.

```
<soapenv:Body>
  <soapenv:Fault>
    <Faultcode>soapenv:Client.NoSuchKey</Faultcode>
    <Faultstring>The specified key does not exist.</Faultstring>
    <Detail>
      <Key>Fred</Key>
    </Detail>
  </soapenv:Fault>
</soapenv:Body>
```

다음 표에서는 SOAP 오류 응답 요소를 설명합니다.

이름	설명
Detail	오류와 관련된 키를 위한 컨테이너 유형: 컨테이너 상위 항목: Body.Fault
Fault	오류 정보를 위한 컨테이너입니다. 유형: 컨테이너 상위 항목: Body
Faultcode	오류 코드는 오류 상태를 고유하게 식별하는 문자열입니다. 이 코드는 오류를 감지하여 유형별로 처리하는 프로그램에서 판독하고 이해하기 위한 것입니다. 자세한 내용은 오류 코드 목록 을 참조하세요. 유형: 문자열 상위 항목: Body.Fault

이름	설명
Faultstring	오류 문자열에는 오류 상태에 관한 일반적인 설명(영문)이 포함되어 있습니다. 이는 기계가 아닌 사람을 위한 것입니다. 간단한 프로그램은 자체적으로 확인되지 않거나 어떻게 처리해야 할지 모르는 오류 상태가 발생하면 최종 사용자에게 직접 메시지를 표시합니다. 좀 더 포괄적인 오류 처리 기능을 갖고 있고 적절히 국제화된 고급 프로그램은 오류 문자열을 무시할 가능성이 큼니다. 유형: 문자열 상위 항목: Body.Fault
Key	오류와 관련된 키를 식별 유형: 문자열 상위 항목: Body.Fault

부록: 요청 인증(AWS 서명 버전 2)

Important

이 섹션에서는 AWS 서명 버전 2를 사용하여 요청을 인증하는 방법을 설명합니다. 서명 버전 2가 꺼지고(사용되지 않음), Amazon S3는 서명 버전 4를 사용하여 서명된 API 요청만 수락합니다. 자세한 정보는 [AWS Amazon S3에서 서명 버전 2 사용 중지\(사용되지 않음\)](#) 섹션을 참조하세요.

서명 버전 4는 모든 AWS 리전에서 지원되며, 새 리전을 지원하는 유일한 버전입니다. 자세한 내용은 Amazon Simple Storage Service API 참조의 [요청 인증\(AWS 서명 버전 4\)](#)을 참조하세요.

Amazon S3는 요청에 서명하는 데 사용된 API 서명 버전을 식별할 수 있는 기능을 제공합니다. 시그니처 버전 2 서명을 활용하는 워크플로가 있는지 확인하고 시그니처 버전 4를 사용하도록 업그레이드하여 비즈니스에 미치는 영향을 차단해야 합니다.

- CloudTrail 이벤트 로그(권장 옵션)를 사용하는 경우 [CloudTrail을 사용하여 Amazon S3 서명 버전 2 요청 식별](#)에서 이러한 요청을 쿼리하고 식별하는 방법을 참조하세요.
- Amazon S3 서버 액세스 로그를 사용하는 경우 [Amazon S3 서버 액세스 로그를 사용하여 요청 식별](#)을 참조하세요.

주제

- [REST API를 사용하여 요청 인증\(AWS 서명 버전 2\)](#)
- [REST 요청 서명 및 인증\(AWS 서명 버전 2\)](#)
- [POST를 사용한 브라우저 기반 업로드\(AWS 서명 버전 2\)](#)

REST API를 사용하여 요청 인증(AWS 서명 버전 2)

REST를 사용하여 Amazon S3에 액세스할 때는 요청이 인증될 수 있도록 요청에 다음 항목을 제공해야 합니다.

요청 요소

- AWS 액세스 키 ID – 각 요청에 요청을 보내기 위해 사용하는 자격 증명의 액세스 키 ID를 포함해야 합니다.
- 서명 – 각 요청에 유효한 요청 서명을 포함하지 않으면 요청이 거부됩니다.

요청 서명은 사용자 및 AWS에게만 알려지는 공유 암호인 보안 액세스 키를 사용하여 계산됩니다.

- 타임스탬프 – 각 요청에 요청이 생성된 날짜 및 시간을 UTC 문자열로 표기하여 포함해야 합니다.
- 날짜 – 각 요청에 요청의 타임스탬프를 포함해야 합니다.

사용하는 API 작업에 따라 타임스탬프 대신 또는 타임스탬프 외에 요청의 만료 날짜 및 시간을 제공할 수 있습니다. 필요한 사항을 확인하려면 특정 작업에 대한 인증 주제를 참조하십시오.

다음은 Amazon S3에 대한 요청을 인증하는 데 사용되는 일반적인 단계입니다. 사용자에게 필요한 보안 자격 증명, 액세스 키 ID 및 보안 액세스 키가 있다고 가정됩니다.

You**1 Create a request:****Request**

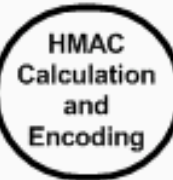
AccessKeyId = ...
 Action = ...
 Timestamp = ...
 ParameterA = ...

2 Create an HMAC-SHA1 signature:

String based on
request contents

+

Your Secret Access Key



Your Signature

3 Send the request and signature to AWS:**Request**

AccessKeyId = ...
 Action = ...
 Timestamp = ...
 ParameterA = ...



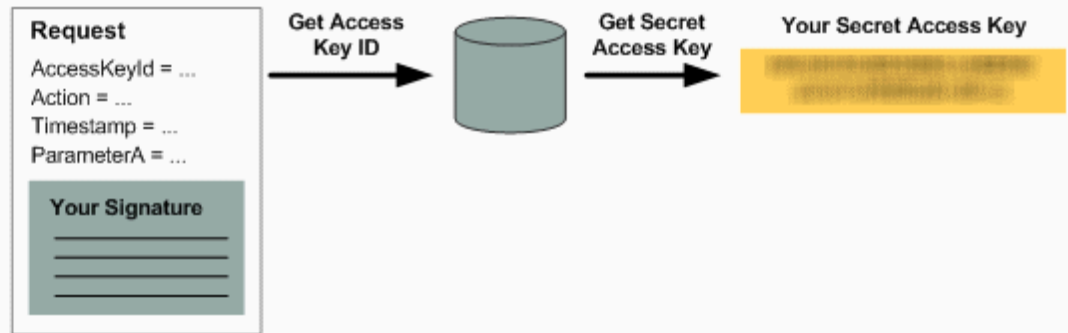
AWS

Your Signature

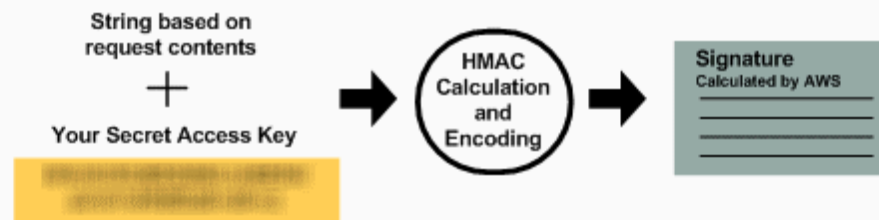
- 1 AWS에 대한 요청을 구성합니다.
- 2 보안 액세스 키를 사용하여 서명을 계산합니다.
- 3 Amazon S3에 요청을 보냅니다. 요청에 액세스 키 ID 및 서명을 포함합니다. Amazon S3에서 다음 3가지 단계를 수행합니다.

AWS

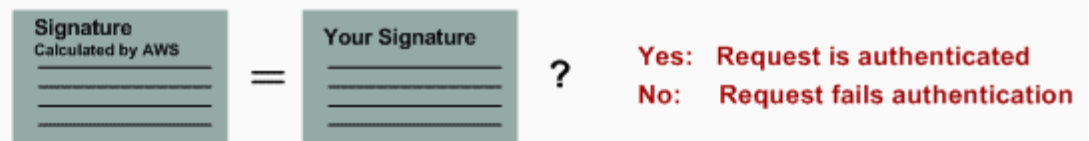
4 Retrieve your Secret Access Key:



5 Create an HMAC-SHA1 signature:



6 Compare the two signatures:



4 Amazon S3에서 액세스 키 ID를 사용하여 보안 액세스 키를 조회합니다.

5 Amazon S3에서 요청을 통해 전송된 서명을 계산하는 데 사용된 것과 같은 알고리즘을 사용하여 요청 데이터 및 보안 액세스 키에서 서명을 계산합니다.

6 Amazon S3에서 생성된 서명이 사용자가 요청을 통해 보낸 서명과 일치하는 경우 요청이 인증된 것으로 간주됩니다. 서명이 일치하지 않는 경우 요청이 삭제되고 Amazon S3에서 오류 응답을 반환합니다.

세부 인증 정보

REST 인증에 대한 세부 정보는 [REST 요청 서명 및 인증\(AWS 서명 버전 2\)](#)을 참조하십시오.

REST 요청 서명 및 인증(AWS 서명 버전 2)

주제

- [임시 보안 자격 증명 사용](#)
- [Authentication 헤더](#)
- [서명에 대한 요청 정규화](#)
- [CanonicalizedResource 요소 구성](#)
- [CanonicalizedAmzHeaders 요소 구성](#)
- [위치 및 이름 지정된 HTTP 헤더 StringToSign 요소](#)
- [타임스탬프 요구 사항](#)
- [인증 예제](#)
- [REST 요청 서명 문제](#)
- [쿼리 문자열 요청 인증 대체 방법](#)

Note

이 항목에서는 서명 버전 2를 사용한 요청 인증을 설명합니다. Amazon S3가 이제 최신 서명 버전 4를 지원합니다. 최신 서명 버전은 모든 리전에서 지원되며 2014년 1월 30일 이후에는 모든 새 리전에서 서명 버전 4만 지원합니다. 자세한 내용은 Amazon Simple Storage Service API 참조의 [요청 인증\(AWS 서명 버전 4\)](#)을 참조하세요.

인증은 시스템에 대한 자격 증명을 입증하는 과정입니다. 자격 증명은 Amazon S3 액세스 제어 결정에서 중요한 요소입니다. 요청은 요청자의 자격 증명에 따라 부분적으로 허용 또는 거부됩니다. 예를 들어, 버킷을 만들 수 있는 권한은 등록된 개발자용으로 예약되어 있고(기본 설정) 버킷의 객체를 만들 수 있는 권한은 해당 버킷 소유자용으로 예약되어 있습니다. 개발자는 이러한 권한을 호출하는 요청을 생성하여 요청을 인증함으로써 시스템에 자격 증명을 입증해야 합니다. 이 단원에서 그 방법을 설명합니다.

Note

이 단원의 내용은 HTTP POST에 적용되지 않습니다. 자세한 내용은 [POST를 사용한 브라우저 기반 업로드\(AWS 서명 버전 2\)](#) 섹션을 참조하세요.

Amazon S3 REST API는 키 참조 HMAC(Hash Message Authentication Code)를 기반으로 하는 사용자 지정 HTTP 체계를 인증에 사용합니다. 요청을 인증하려면 먼저 요청의 선택된 요소를 연결하여 문자열을 구성합니다. 그 다음 AWS 비밀 액세스 키를 사용하여 해당 문자열의 HMAC를 계산합니다. 이 프로세스를 약식으로 "요청 서명"이라고 부르며, 이 프로세스는 실제 서명의 보안 속성을 시뮬레이션하기 때문에 HMAC 알고리즘의 결과를 서명이라고 부릅니다. 마지막으로 이 단원에서 설명된 구문을 사용하여 이 서명을 요청의 파라미터로 추가합니다.

인증된 요청을 받으면 시스템은 요청된 AWS 비밀 액세스 키를 가져와 동일한 방식으로 사용하여 받은 메시지에 대한 서명을 계산합니다. 그런 다음 계산한 서명을 요청자가 제시한 서명과 비교합니다. 두 서명이 일치하면 요청자가 AWS 비밀 액세스 키에 대한 액세스 권한을 가지고 있는 것으로 간주되어 키가 발행된 보안 주체의 권한으로 동작합니다. 두 서명이 일치하지 않으면 요청이 삭제되고 시스템에서 오류 메시지를 반환합니다.

Example인증된 Amazon S3 REST 요청

```
GET /photos/puppy.jpg HTTP/1.1
Host: awsexamplebucket1.us-west-1.s3.amazonaws.com
Date: Tue, 27 Mar 2007 19:36:42 +0000
```

```
Authorization: AWS AKIAIOSFODNN7EXAMPLE:
qgk2+6Sv9/oM7G3qLEjTH1a1l1g=
```

임시 보안 자격 증명 사용

임시 보안 자격 증명을 사용하여 요청을 서명할 경우([요청 만들기](#) 참조), x-amz-security-token 헤더를 추가하여 요청에 해당 보안 토큰을 포함해야 합니다.

AWS Security Token Service API를 사용하여 임시 보안 자격 증명을 받을 경우에는 응답에 임시 보안 자격 증명과 세션 토큰이 포함됩니다. Amazon S3로 요청을 보낼 때 x-amz-security-token 헤더에 세션 토큰 값을 제공합니다. IAM에서 제공하는 AWS Security Token Service API에 대한 자세한 내용은 AWS Security Token Service API 참조 가이드의 [작업](#)을 참조하세요.

Authentication 헤더

Amazon S3 REST API는 표준 HTTP Authorization 헤더를 사용하여 인증 정보를 전달합니다. 표준 헤더의 이름은 아쉽게도 인증이 아닌 인증 정보만을 나타냅니다. Amazon S3 인증 체계에서 Authorization 헤더는 다음 형식을 갖습니다.

```
Authorization: AWS AWSAccessKeyId:Signature
```

개발자는 등록 시 AWS 액세스 키 ID와 AWS 비밀 액세스 키를 발급받습니다. 요청 인증에 대해 `AWSSignatureVersion` 요소는 서명을 계산할 때 사용된 액세스 키 ID를 식별하고, 요청을 하는 개발자를 간접적으로 식별합니다.

Signature 요소는 요청의 선택된 요소에 대한 RFC 2104 HMAC-SHA1이므로 Authorization 헤더의 Signature 부분은 요청마다 다양합니다. 시스템에서 계산한 요청 서명이 요청에 포함된 Signature와 일치하면 요청자는 AWS 비밀 액세스 키를 소유한 것으로 인증됩니다. 그런 다음 요청은 키가 발급되었던 개발자의 자격 증명과 권한으로 처리됩니다.

다음은 Authorization 요청 헤더의 생성을 보여 주는 구문의 예입니다. (이 예에서 `\n`은 줄 바꿈이라고 하는 유니코드 코드 포인트 `U+000A`를 의미합니다).

```
Authorization = "AWS" + " " + AWSSignatureVersion + ":" + Signature;

Signature = Base64( HMAC-SHA1( UTF-8-Encoding-Of(YourSecretAccessKey), UTF-8-Encoding-Of( StringToSign ) ) );

StringToSign = HTTP-Verb + "\n" +
  Content-MD5 + "\n" +
  Content-Type + "\n" +
  Date + "\n" +
  CanonicalizedAmzHeaders +
  CanonicalizedResource;

CanonicalizedResource = [ "/" + Bucket ] +
  <HTTP-Request-URI, from the protocol name up to the query string> +
  [ subresource, if present. For example "?acl", "?location", or "?logging" ];

CanonicalizedAmzHeaders = <described below>
```

HMAC-SHA1은 [메시지 인증에 대한 RFC 2104 - 키-해싱](#)으로 정의되는 알고리즘입니다. 알고리즘은 키와 메시지로 구성된 2바이트 문자열을 입력으로 사용합니다. Amazon S3 요청 인증의 경우, AWS 비밀 액세스 키(`YourSecretAccessKey`)를 키로 사용하고 `StringToSign`의 UTF-8 인코딩을 메시지로 사용합니다. 또한 HMAC-SHA1의 출력은 다이제스트라고 하는 1바이트 문자열입니다. Signature 요청 파라미터는 Base64 인코딩 다이제스트로 구성됩니다.

서명에 대한 요청 정규화

시스템에서 인증된 요청을 받으면 계산된 이 요청 서명을, `StringToSign`의 요청에서 제공된 서명과 비교한다는 것을 기억하십시오. 이러한 이유로 인해 Amazon S3가 사용한 방법과 동일한 방법으로 서

명을 계산해야 합니다. 서명 정규화를 위해 요청을 합의된 형식에 추가(put)하는 프로세스를 호출합니다.

CanonicalizedResource 요소 구성

CanonicalizedResource는 요청 대상인 Amazon S3 리소스를 나타냅니다. REST 요청의 경우 다음과 같이 작성합니다.

1. 빈 문자열("")로 시작합니다.
2. 요청이 HTTP 호스트 헤더(가상 호스팅 방식)를 사용하여 버킷을 지정할 경우, 버킷 이름 앞에 "/"를 붙입니다(예: "/bucketname"). 경로 방식 요청 및 버킷을 다루지 않는 요청에는 해당되지 않습니다. 가상 호스팅 방식 요청에 대한 자세한 내용은 [버킷의 가상 호스팅](#)을 참조하세요.

가상 호스팅 방식 요청 "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg"의 경우 CanonicalizedResource는 "/awsexamplebucket1"입니다.

경로 방식 요청 "https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg"의 경우 CanonicalizedResource는 ""입니다.

3. 디코딩되지 않는 HTTP 요청 URI의 경로 부분을 쿼리 문자열 직전까지 추가합니다.

가상 호스팅 방식 요청 "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg"의 경우 CanonicalizedResource는 "/awsexamplebucket1/photos/puppy.jpg"입니다.

경로 방식 요청 "https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg"의 경우 CanonicalizedResource는 "/awsexamplebucket1/photos/puppy.jpg"입니다. 이 시점에서 CanonicalizedResource는 가상 호스팅 방식과 경로 방식 요청에서 동일합니다.

[GET Service](#)와 같이 버킷을 다루지 않는 요청의 경우, "/"를 추가합니다.

4. 요청이 ?versioning, ?location, ?acl, ?lifecycle 또는 ?versionid와(과) 같은 하위 리소스를 다루는 경우, 하위 리소스와 하위 리소스의 값(있을 경우) 그리고 물음표를 추가합니다. 하위 리소스가 여러 개일 경우, 하위 리소스 이름에 따라 문자순으로 정렬되고 '&'로 구분됩니다(예: ?acl&versionId=*value*).

CanonicalizedResource 요소를 작성할 때 포함되어야 하는 하위 리소스는 acl, lifecycle, location, logging, notification, partNumber, policy, requestPayment, uploadId, uploads, versionId, versioning, versions, website입니다.

요청이 응답 헤더 값을 재정의하는 쿼리 문자열 파라미터를 지정할 경우([Get Object](#) 참조), 쿼리 문자열 파라미터와 그 값을 추가합니다. 서명 시에는 이러한 값을 인코딩하지 않지만, 요청을 할 때에

는 이러한 파라미터 값을 인코딩해야 합니다. GET 요청의 쿼리 문자열 파라미터에는 response-content-type, response-content-language, response-expires, response-cache-control, response-content-disposition, response-content-encoding이 포함됩니다.

다중 객체 삭제 요청에 대한 CanonicalizedResource를 생성할 경우에는 delete 쿼리 문자열 파라미터가 포함되어야 합니다.

HTTP 요청 URI에서 파생된 CanonicalizedResource의 요소는 URL 인코딩 메타 문자를 포함하여 HTTP 요청에 표시된 그대로 서명되어야 합니다.

CanonicalizedResource는 HTTP 요청-URI와는 다를 수 있습니다. 특히, 요청이 HTTP Host 헤더를 사용하여 버킷을 지정할 경우에는 HTTP 요청-URI에 버킷이 표시되지 않습니다. 하지만 CanonicalizedResource에는 계속 버킷이 포함됩니다. 쿼리 문자열 파라미터는 요청-URI에 표시될 수 있지만 CanonicalizedResource에는 포함되지 않습니다. 자세한 내용은 [버킷의 가상 호스팅](#)을 참조하세요.

CanonicalizedAmzHeaders 요소 구성

StringToSign의 CanonicalizedAmzHeaders 부분을 작성하려면 'x-amz-'(대/소문자 구분 없는 비교 사용)로 시작하는 모든 HTTP 요청 헤더를 선택한 후 다음 프로세스를 사용합니다.

1. 각 HTTP 헤더 이름을 소문자로 변환합니다. 예를 들어, 'X-Amz-Date'는 'x-amz-date'가 됩니다.
2. 헤더 컬렉션을 헤더 이름에 따라 문자순으로 정렬합니다.
3. 이름이 같은 헤더 필드를 RFC 2616, 단원 4.2에 지정된 대로 값 사이 공백 없이 하나의 "헤더 이름: 쉼표로 구분된 값 목록" 페어로 통합합니다. 예를 들어, 두 개의 메타데이터 헤더인 'x-amz-meta-username: fred'와 'x-amz-meta-username: barney'는 하나의 'x-amz-meta-username: fred,barney' 헤더로 통합됩니다.
4. RFC 2616, 단원 4.2의 허용에 따라 여러 줄을 차지하는 긴 헤더는 폴딩 공백(줄 바꿈 포함)을 하나의 공백으로 바꿔서 펼칩니다(언폴딩).
5. 헤더에서 콜론 앞뒤의 공백을 자릅니다. 예를 들어, 'x-amz-meta-username: fred,barney'는 'x-amz-meta-username:fred,barney'가 됩니다.
6. 마지막으로 줄 바꿈 문자(U+000A)를 결과 목록의 정규화된 각 헤더에 추가합니다. 이 목록의 모든 헤더를 하나의 문자열로 연결하여 CanonicalizedResource 요소를 작성합니다.

위치 및 이름 지정된 HTTP 헤더 StringToSign 요소

StringToSign의 처음 몇 개의 헤더 요소(Content-Type, Date, Content-MD5)는 위치와 관련된 것입니다. StringToSign은 이러한 헤더 이름을 포함하지 않고 요청의 값만 포함합니다. 반면에 'x-amz-' 요소에는 이름이 지정됩니다. 헤더 이름과 헤더 값은 모두 StringToSign에 표시됩니다.

StringToSign 정의에서 호출하는 위치 헤더가 요청에 없으면(예를 들어, Content-Type 또는 Content-MD5가 PUT 요청의 경우 선택 사항이고, GET 요청의 경우 필요 없을 경우), 해당 위치를 빈 문자열("")로 대체합니다.

타임스탬프 요구 사항

HTTP Date 헤더 또는 x-amz-date 대체 헤더를 사용하는 유효한 타임스탬프는 인증된 요청에 필수입니다. 또한 인증된 요청에 포함된 클라이언트 타임스탬프는 요청 수신 시 Amazon S3 시스템 시간이 15분 이내여야 합니다. 그렇지 않을 경우 RequestTimeTooSkewed 오류 코드와 함께 요청이 실패합니다. 이렇게 제한하는 목적은 악의적으로 가로챈 요청을 재실행할 수 있는 가능성을 제한하기 위한 것입니다. 도청에 대한 보안을 강화하려면 인증된 사용자에게 대해 HTTPS 전송을 사용합니다.

Note

요청 날짜에 대한 검증 제약 조건은 쿼리 문자열 인증을 사용하지 않는 인증된 요청에만 적용됩니다. 자세한 내용은 [쿼리 문자열 요청 인증 대체 방법](#) 섹션을 참조하세요.

일부 HTTP 클라이언트 라이브러리는 요청에 대한 Date 헤더 설정 가능 여부를 표시하지 않습니다. 정규화된 헤더에 'Date' 헤더 값을 포함하는 데 문제가 있을 경우, 대신 'x-amz-date' 헤더를 사용하여 요청에 대한 타임스탬프를 설정할 수 있습니다. x-amz-date 헤더의 값은 RFC 2616 형식 (<http://www.ietf.org/rfc/rfc2616.txt>) 중 하나여야 합니다. x-amz-date 헤더가 요청에 있으면 시스템에서 요청 서명을 계산할 때 모든 Date 헤더를 무시합니다. 따라서 x-amz-date 헤더를 포함할 경우에는 Date를 작성할 때 StringToSign에 빈 문자열을 사용하십시오. 다음 단원의 예제를 참조하십시오.

인증 예제

이 단원의 예제에서는 다음 표의 (작동하지 않는) 자격 증명을 사용합니다.

파라미터	값
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE

파라미터	값
AWSecret AccessKey	wJa1rXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

StringToSign 예제에서 형식은 크게 중요하지 않으며 \n은 일반적으로 줄 바꿈이라고 하는 유니코드 코드 포인트 U+000A를 의미합니다. 또한 이 예제는 "+0000"을 사용하여 표준 시간대를 지정합니다. 대신 "GMT"를 사용하여 표준 시간대를 지정할 수 있지만 예제에 나와 있는 서명이 달라집니다.

객체 GET

이 예제는 awsexamplebucket1 버킷에서 객체를 가져옵니다.

요청	StringToSign
<pre>GET /photos/puppy.jpg HTTP/1.1 Host: awsexamplebucket1.us- west-1.s3.amazonaws.com Date: Tue, 27 Mar 2007 19:36:42 +0000 Authorization: AWS AKIAIOSF0 DNN7EXAMPLE: qgk2+6Sv9/oM7G3qLEjTH1a1l1g=</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:36:42 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

CanonicalizedResource는 버킷 이름을 포함하지만 HTTP 요청-URI는 그렇지 않습니다. 버킷은 호스트 헤더가 지정합니다.

Note

다음 Python 스크립트는 제공된 파라미터를 사용하여 선행 서명을 계산합니다. 이 스크립트를 사용해 자체 서명을 구성하여 키 및 StringToSign을 적절하게 바꿀 수 있습니다.

```
import base64
import hmac
from hashlib import sha1

access_key = 'AKIAIOSF0DNN7EXAMPLE'.encode("UTF-8")
```

```
secret_key = 'wJa1rXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY'.encode("UTF-8")

string_to_sign = 'GET\n\n\nTue, 27 Mar 2007 19:36:42 +0000\n/awsexamplebucket1/
photos/puppy.jpg'.encode("UTF-8")
signature = base64.b64encode(
    hmac.new(
        secret_key, string_to_sign, sha1
    ).digest()
).strip()

print(f"AWS {access_key.decode()}:{signature.decode()}")
```

객체 PUT

이 예제는 awsexamplebucket1 버킷으로 객체를 가져옵니다.

요청	StringToSign
<pre>PUT /photos/puppy.jpg HTTP/1.1 Content-Type: image/jpeg Content-Length: 94328 Host: awsexamplebucket1.s3.us-west-1.amazonaws.com Date: Tue, 27 Mar 2007 21:15:45 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE: iqRzw+ileNPu1fhspnRs8n0jjIA=</pre>	<pre>PUT\n \n image/jpeg\n Tue, 27 Mar 2007 21:15:45 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

요청과 StringToSign의 Content-Type 헤더에 유의합니다. 또한 Content-MD5는 요청에 나타나지 않기 때문에 StringToSign에서 공백이어야 합니다.

List

이 예제는 awsexamplebucket1 버킷의 내용을 나열합니다.

요청	StringToSign
<pre>GET /?prefix=photos&max-keys=50&marker=puppy HTTP/1.1 User-Agent: Mozilla/5.0 Host: awsexamplebucket1.s3.us-west-1.amazo naws.com Date: Tue, 27 Mar 2007 19:42:41 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE: m0WP8eCtspQl5Ahe6L1SozdX9YA=</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:42:41 +0000\n /awsexamplebucket1/</pre>

CanonicalizedResource 뒤에 슬래시가 오고, 쿼리 문자열 파라미터가 없다는 점에 유의합니다.

가져오기

이 예제는 'awsexamplebucket1' 버킷에 대한 액세스 제어 정책 하위 리소스를 가져옵니다.

요청	StringToSign
<pre>GET /?acl HTTP/1.1 Host: awsexamplebucket1.s3.us-west-1.amazo naws.com Date: Tue, 27 Mar 2007 19:44:46 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE: 82ZHiFIjc+WbcwFKGUVESpPn+0=</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:44:46 +0000\n /awsexamplebucket1/?acl</pre>

CanonicalizedResource에 하위 리소스 쿼리 문자열 파라미터가 어떻게 포함되는지를 살펴보십시오.

삭제

이 예제는 경로 방식 및 Date 대체 헤더를 사용하여 'awsexamplebucket1' 버킷에서 객체를 제거합니다.

요청	StringToSign
<pre>DELETE /awsexamplebucket1/photos/puppy.jpg HTTP/1.1 User-Agent: dotnet Host: s3.us-west-1.amazonaws.com Date: Tue, 27 Mar 2007 21:20:27 +0000 x-amz-date: Tue, 27 Mar 2007 21:20:26 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE:XbyTlbQdu9Xw5o8P4iMwPktxQd8=</pre>	<pre>DELETE\n \n \n Tue, 27 Mar 2007 21:20:26 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

클라이언트 라이브러리로 인해 날짜를 설정할 수 없어서 'x-amz-date' 메서드를 대신 사용하여 어떻게 날짜를 지정했는지 살펴보십시오. 이 경우 x-amz-date는 Date 헤더보다 우선 적용됩니다. 따라서 서명의 날짜 항목에 x-amz-date 헤더 값이 포함되어야 합니다.

업로드

이 예제는 메타데이터와 함께 CNAME 방식의 가상 호스팅 버킷에 객체를 업로드합니다.

요청	StringToSign
<pre>PUT /db-backup.dat.gz HTTP/1.1 User-Agent: curl/7.15.5 Host: static.example.com:8080 Date: Tue, 27 Mar 2007 21:06:08 +0000 x-amz-acl: public-read content-type: application/x-download Content-MD5: 4gJE4saaMU4BqNR0kLY+lw== X-Amz-Meta-ReviewedBy: joe@example.com X-Amz-Meta-ReviewedBy: jane@example.com X-Amz-Meta-FileChecksum: 0x02661779 X-Amz-Meta-ChecksumAlgorithm: crc32 Content-Disposition: attachment; filename=database.dat Content-Encoding: gzip Content-Length: 5913339</pre>	<pre>PUT\n 4gJE4saaMU4BqNR0kLY+lw==\n application/x-download\n Tue, 27 Mar 2007 21:06:08 +0000\n x-amz-acl:public-read\n x-amz-meta-checksumalgorithm:crc32\n x-amz-meta-filechecksum:0x02661779\n x-amz-meta-reviewedby:joe@example.com,jane@example.com\n /static.example.com/db-backup.dat.gz</pre>

요청	StringToSign
<pre>Authorization: AWS AKIAIOSFODNN7EXAMPLE LE: jtBQa0Aq+DkULFI8qrpwIjGEx0E=</pre>	

'x-amz-' 헤더가 어떻게 정렬되고, 추가 공백이 잘렸으며, 소문자로 변환되었는지 살펴보십시오. 또한 이름이 같은 여러 헤더가 값을 구분하는 쉼표를 사용하여 어떻게 결합되었는지도 살펴보십시오.

Content-Type 및 Content-MD5 HTTP 엔터티 헤더만 StringToSign에 표시되는 것에 유의합니다. 다른 Content-* 엔터티 헤더는 표시되지 않습니다.

즉, CanonicalizedResource에는 버킷 이름이 포함되지만, HTTP 요청-URI에는 포함되지 않습니다. 버킷은 호스트 헤더가 지정합니다.

내 버킷 모두 나열하기

요청	StringToSign
<pre>GET / HTTP/1.1 Host: s3.us-west-1.amazonaws.com Date: Wed, 28 Mar 2007 01:29:59 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE:qGdzdE RIC03wnaRNKh60qZehG9s=</pre>	<pre>GET\n \n \n Wed, 28 Mar 2007 01:29:59 +0000\n /</pre>

유니코드 키

요청	StringToSign
<pre>GET /dictionary/fran%C3%A7ais/pr %c3%a9f%c3%a8re HTTP/1.1 Host: s3.us-west-1.amazonaws.com Date: Wed, 28 Mar 2007 01:49:49 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE LE:DNEZGsoieTZ92F3bUfSPQcbGm1M=</pre>	<pre>GET\n \n \n Wed, 28 Mar 2007 01:49:49 +0000\n /dictionary/fran%C3%A7ais/pr %c3%a9f%c3%a8re</pre>

Note

요청-URI에서 파생된 StringToSign의 요소는 URL-인코딩 및 대문자를 포함하여 그대로 처리됩니다.

REST 요청 서명 문제

REST 요청 인증이 실패하면 시스템은 XML 오류 문서로 요청에 응답합니다. 이 오류 문서에 포함된 정보를 통해 개발자는 문제를 진단할 수 있습니다. 특히, StringToSign 오류 문서의 SignatureDoesNotMatch 요소는 시스템에서 사용하는 정확한 요청 정규화를 정확히 알려 줍니다.

일부 도구 키트는 PUT 과정에서 Content-Type 헤더와 같은 헤더 삽입을 자동으로 수행합니다. 대부분의 경우 삽입된 헤더의 값은 그대로 유지되므로 Ethereum 또는 tcpmon과 같은 도구를 사용하여 누락된 헤더를 발견할 수 있습니다.

쿼리 문자열 요청 인증 대체 방법

Authorization HTTP 헤더를 사용하는 대신 쿼리 문자열 파라미터로 필요한 정보를 전달하여 특정 유형의 요청을 인증할 수 있습니다. 이 방법은 요청을 프록시하지 않고 타사 브라우저에서 Amazon S3 비공개 데이터에 직접 액세스할 수 있도록 할 경우에 유용합니다. 방법은 "미리 서명된" 요청을 작성한 후 이를 최종 사용자의 브라우저가 검색할 수 있는 URL로 인코딩하는 것입니다. 또한 만료 시간을 지정하여 미리 서명된 요청을 제한할 수 있습니다.

쿼리 파라미터를 사용하여 요청을 인증하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 [요청 인증: 쿼리 파라미터 사용\(AWS 서명 버전 4\)](#)을 참조하세요. AWS SDK를 사용하여 미리 서명된 URL을 생성하는 예제는 [미리 서명된 URL을 사용하여 객체 공유](#)를 참조하세요.

서명 생성

다음은 쿼리 문자열 인증 Amazon S3 REST 요청의 예제입니다.

```
GET /photos/puppy.jpg
?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Expires=1141889120&Signature=vjbyPxybdZaNmGa%2ByT272YEAiv4%3D HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
Date: Mon, 26 Mar 2007 19:37:58 +0000
```

쿼리 문자열 요청 인증 메서드는 특별한 HTTP 헤더를 필요로 하지 않습니다. 대신에 필수 인증 요소는 쿼리 문자열 파라미터로 지정됩니다.

쿼리 문자열 파라미터 이름	값 예	설명
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE	사용자의 AWS 액세스 키 ID입니다. 요청에 서명하는 데 사용되는 AWS 비밀 액세스 키를 지정하고, 요청을 하는 개발자의 자격 증명을 간접적으로 지정합니다.
Expires	1141889120	서명이 만료되는 시간으로, epoch(1970년 1월 1일 00:00:00 UTC) 이후 경과 시간(초)으로 지정됩니다. 이 시간이 지나면(서버 기준) 요청이 거절됩니다.
Signature	vjbyPxybdZaNmGa%2ByT272YEAiv4%3D	URL 인코딩으로, StringToSign의 HMAC-SHA1의 Base64 인코딩입니다.

쿼리 문자열 요청 인증 메서드는 일반적인 메서드와 약간 다르며, Signature 요청 파라미터 및 StringToSign 요소의 형식도 다릅니다. 다음은 쿼리 문자열 인증 메서드를 보여 주는 구문의 예입니다.

```
Signature = URL-Encode( Base64( HMAC-SHA1( YourSecretAccessKey, UTF-8-Encoding-Of( StringToSign ) ) ) );
```

```
StringToSign = HTTP-VERB + "\n" +
    Content-MD5 + "\n" +
    Content-Type + "\n" +
    Expires + "\n" +
    CanonicalizedAmzHeaders +
    CanonicalizedResource;
```

YourSecretAccessKey는 Amazon 웹 서비스 개발자로 등록할 때 Amazon에서 할당한 AWS 비밀 액세스 키 ID입니다. Signature가 쿼리 문자열에 배치될 수 있도록 URL 인코딩된 것에 유의하십시오. 또한 StringToSign에서 HTTP Date 위치 요소가 Expires로 교체되었음에 유의합니다. CanonicalizedAmzHeaders와 CanonicalizedResource는 동일합니다.

Note

쿼리 문자열 인증 메서드에서 서명할 문자열을 계산할 때 Date 또는 x-amz-date request 헤더를 사용하지 않습니다.

쿼리 문자열 요청 인증

요청	StringToSign
<pre>GET /photos/puppy.jpg?AWSAccess KeyId=AKIAIOSFODNN7EXAMPLE& Signature=NpgCjnDzrM%2BWFzo ENXmpNDUsSn8%3D& Expires=1175139620 HTTP/1.1 Host: awsexamplebucket1.s3.us-wes t-1.amazonaws.com</pre>	<pre>GET\n \n \n 1175139620\n /awsexamplebucket1/photos/puppy.jpg</pre>

브라우저가 GET 요청을 할 때 Content-MD5 또는 Content-Type 헤더를 제공하지 않으며 어떠한 x-amz- 헤더도 설정하지 않으며 따라서 StringToSign의 부분은 공백으로 남는다고 가정합니다.

Base64 인코딩 사용

HMAC 요청 서명은 Base64 인코딩이어야 합니다. Base64 인코딩은 서명을 요청에 연결할 수 있는 간단한 ASCII 문자열로 변환할 수 있습니다. 더하기(+), 슬래시(/), 등호(=)와 같이 서명 문자열에 나타날 수 있는 문자는 URI에 사용할 경우 인코딩해야 합니다. 예를 들어, 인증 코드에 더하기(+) 기호가 포함될 경우, 요청에서 %2B로 인코딩해야 합니다. 슬래시는 %2F, 등호는 %3D로 인코딩합니다.

Base64 방식의 인코딩 예는 Amazon S3 [인증 예제](#)를 참조하십시오.

POST를 사용한 브라우저 기반 업로드(AWS 서명 버전 2)

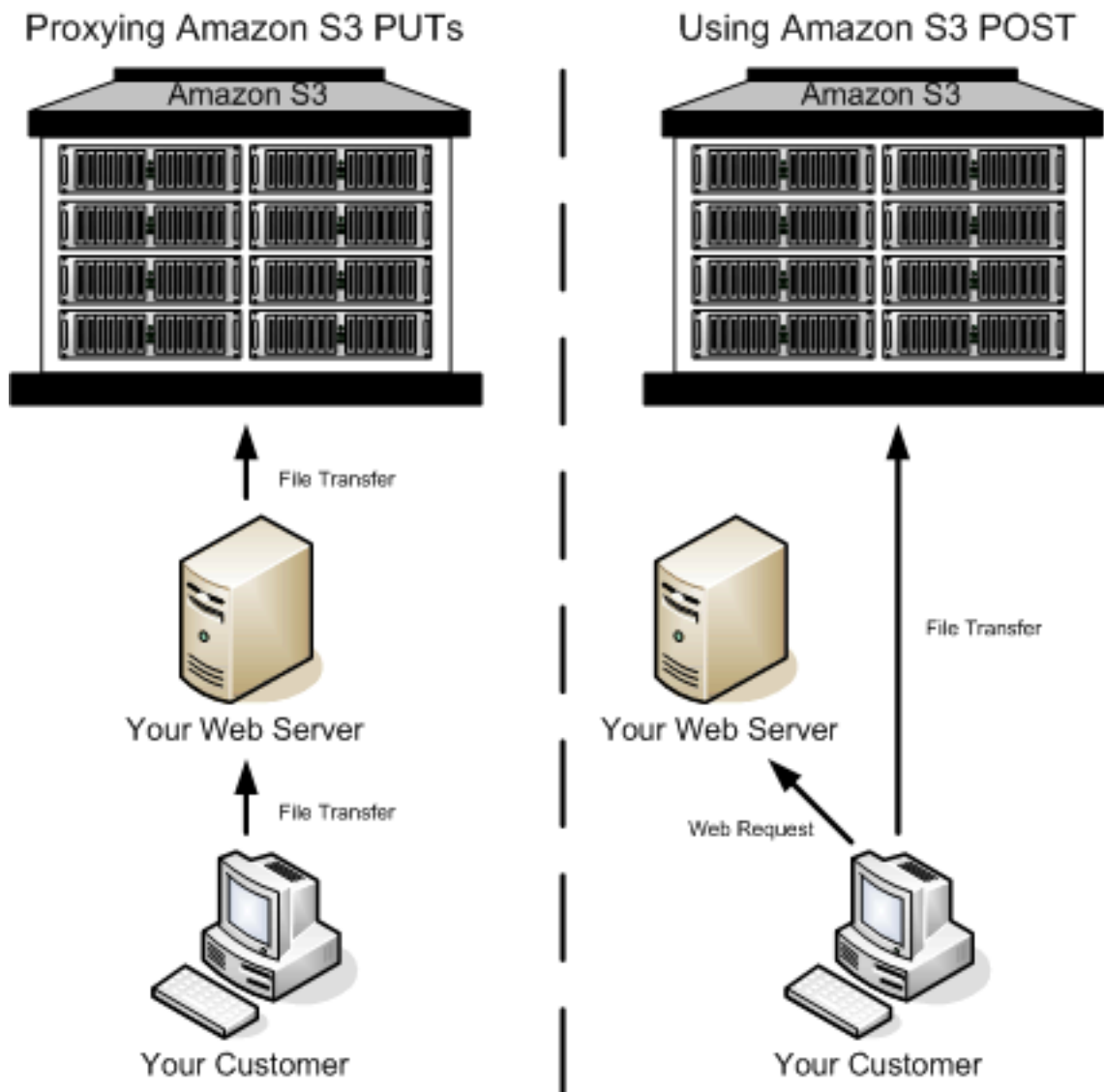
Amazon S3은 POST를 지원하므로 사용자가 곧바로 Amazon S3에 콘텐츠를 업로드할 수 있습니다. POST는 사용자가 Amazon S3에 데이터를 업로드하여 저장하는 애플리케이션에서 업로드를 간소화하고 업로드 지연 시간을 줄이고 비용을 절약하도록 설계되었습니다.

Note

이 섹션에서 다루는 요청 인증은 AWS 서비스에 대한 API 인바운드 요청을 인증하는 프로토콜, AWS 서명 버전 2 기반입니다.

Amazon S3는 이제 모든 AWS 리전에서 AWS 서비스에 대한 인바운드 API 요청을 인증하기 위한 프로토콜인 서명 버전 4를 지원합니다. 현재 2014년 1월 30일 이전에 생성된 AWS 리전이라고 해도 이전 프로토콜인 서명 버전 2에 대한 지원은 계속됩니다. 하지만 2014년 1월 30일 이후에 새롭게 생성된 리전은 모두 서명 버전 4만 지원하며, 이에 따라 새로운 리전에 대한 모든 요청은 서명 버전 4를 이용해야 합니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 [POST를 사용하여 브라우저 기반 업로드에서 요청 인증\(AWS 서명 버전 4\)](#)을 참조하세요.

다음 그림은 Amazon S3 POST를 사용한 업로드를 보여 줍니다.



POST를 사용한 업로드

- 1 사용자가 웹 브라우저를 열고 웹 페이지에 액세스합니다.
- 2 웹 페이지는 사용자가 Amazon S3에 콘텐츠를 업로드하는 데 필요한 모든 정보가 들어 있는 HTTP 양식을 포함합니다.
- 3 사용자가 곧바로 Amazon S3에 콘텐츠를 업로드합니다.

Note

쿼리 문자열 인증은 POST에서 지원되지 않습니다.

HTML 양식(AWS 서명 버전 2)

주제

- [HTML 양식 인코딩](#)
- [HTML 양식 선언](#)
- [HTML 양식 필드](#)
- [정책 구성](#)
- [서명 생성](#)
- [리디렉션](#)

Amazon S3와 통신할 때 일반적으로 REST 또는 SOAP API를 사용하여 put, get, delete 등의 작업을 수행합니다. SOAP API를 처리하거나 REST PUT 요청을 만들 수 없는 브라우저의 경우, POST를 사용하여 브라우저를 통해 곧바로 Amazon S3에 데이터를 업로드할 수 있습니다.

Note

Amazon S3용 SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료 (EOL)됩니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

사용자가 브라우저를 사용하여 Amazon S3에 콘텐츠를 업로드하려면 HTML 양식을 사용해야 합니다. HTML 양식은 양식 선언과 양식 필드로 구성됩니다. 양식 선언은 요청에 대한 상위 수준 정보를 포함합니다. 양식 필드는 요청에 대한 세부 정보는 물론 요청 인증에 사용되는 정책을 포함하며 요청이 사용자가 지정하는 조건에 부합한다는 것을 보장합니다.

Note

양식 데이터 및 경계(파일의 내용 제외)는 20KB를 초과할 수 없습니다.

이 단원에서는 HTML 양식을 사용하는 방법을 알아봅니다.

HTML 양식 인코딩

양식과 정책은 UTF-8로 인코딩되어야 합니다. HTML 헤딩에 혹은 요청 헤더로 인코딩을 지정하여 양식에 UTF-8 인코딩을 적용할 수 있습니다.

Note

HTML 양식 선언에서 쿼리 문자열 인증 파라미터는 허용되지 않습니다.

다음은 HTML 헤딩의 UTF-8 인코딩 예제입니다.

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

다음은 요청 헤더의 UTF-8 인코딩 예제입니다.

```
Content-Type: text/html; charset=UTF-8
```

HTML 양식 선언

양식 선언은 작업, 메서드, 엔클로저 유형 등 3개의 구성 요소를 포함합니다. 이들 값 중 하나라도 잘못 설정되는 경우 요청이 실패합니다.

이 작업은 요청을 처리하는 URL을 지정하는 것이므로 반드시 버킷의 URL로 설정되어야 합니다. 예를 들어 버킷의 이름이 `awsexamplebucket1`이고 리전이 미국 서부(캘리포니아 북부)인 경우 URL은 `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/`입니다.

Note

키 이름은 양식 필드에 지정됩니다.

메서드는 POST이어야 합니다.

엔클로저 유형(`enctype`)은 반드시 지정되어야 하며 파일 업로드와 텍스트 영역 업로드 모두에 대해 `multipart/form-data`로 설정되어야 합니다. 자세한 내용은 [RFC 1867](#)을 참조하십시오.

Example

다음은 버킷 "awsexamplebucket1"에 대한 양식 선언 예제입니다.

```
<form action="https://awsexamplebucket1.s3.us-west-1.amazonaws.com/" method="post"
  enctype="multipart/form-data">
```

HTML 양식 필드

다음 표에서는 HTML 양식 내에서 사용할 수 있는 필드를 설명합니다.

Note

변수 `${filename}`은 사용자가 제공한 파일의 이름으로 자동 대체되며 모든 양식 필드가 이 변수를 인식합니다. 브라우저나 클라이언트에 파일의 전체 또는 부분 경로가 지정되어 있는 경우 슬래시(/) 또는 백슬래시(\) 뒤에 오는 텍스트만 사용됩니다. 예를 들면, "C:\Program Files\directory1\file.txt"는 "file.txt"로 해석됩니다. 파일이나 파일 이름이 제공되는 경우 변수가 빈 문자열로 대체됩니다.

필드 이름	설명	필수
AWSAccessKeyId	정책에 속하는 일련의 제약 조건을 충족하는 요청에 대해 익명 사용자 액세스를 허용하는 버킷 소유자의 AWS 액세스 키 ID입니다. 요청이 정책 문서를 포함하는 경우 이 필드는 필수입니다.	조건
acl	Amazon S3 ACL(액세스 제어 목록)입니다. 잘못된 액세스 통제 목록이 지정되는 경우 오류가 발생합니다. 유형: 문자열 기본: 프라이빗	아니요

필드 이름	설명	필수
	유효한 값: private public-read public-read-write aws-exec-read authenticated-read bucket-owner-read bucket-owner-full-control	
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	REST 지정 헤더입니다. 자세한 내용은 PUT Object 단원을 참조하십시오.	아니요
key	업로드된 키의 이름입니다. 사용자가 제공한 파일 이름을 사용하려면 <code>\${filename}</code> 변수를 사용하십시오. 예를 들어, 사용자 Betty가 파일 lolcatz.jpg를 업로드하고 경로가 /user/betty/\${filename}으로 지정돼 있는 경우, 파일은 /user/betty/lolcatz.jpg로 저장됩니다. 자세한 내용은 객체 메타데이터 작업 을 참조하십시오.	예
policy	요청에서 허용되는 항목을 설명하는 보안 정책입니다. 보안 정책이 없는 요청은 익명으로 간주되며 공개적으로 쓰기 가능한 버킷에서만 요청이 허용됩니다.	아니요

필드 이름	설명	필수
success_action_redirect, redirect	업로드 완료 시 클라이언트가 리디렉션되는 URL입니다. Amazon S3은 해당 URL에 버킷, 키 및 etag 값을 쿼리 문자열 파라미터로 추가합니다. success_action_redirect를 지정하지 않을 경우 Amazon S3은 success_action_status 필드에 지정된 빈 문서 유형을 반환합니다. Amazon S3이 URL을 해석할 수 없는 경우 필드는 무시됩니다. 업로드가 실패하는 경우 Amazon S3이 오류를 표시하고 사용자는 어떤 URL로도 리디렉션되지 않습니다. 자세한 내용은 리디렉션 단원을 참조하십시오.  Note 리디렉션 필드 이름을 더 이상 사용하지 않는 경우 해당 리디렉션 필드 이름에 대한 지원이 조만간 제거됩니다.	아니요

필드 이름	설명	필수
success_action_status	success_action_redirect를 지정하지 않을 경우 업로드 완료 시 클라이언트에 반환되는 상태 코드입니다. 유효한 값은 200, 201 또는 204입니다(기본값). 200 또는 204로 값을 설정하는 경우 Amazon S3은 상태 코드가 200 또는 204인 빈 문서를 반환합니다. 201로 값을 설정하는 경우 Amazon S3은 상태 코드가 201인 XML 문서를 반환합니다. XML 문서의 내용에 대한 자세한 내용은 POST Object 단원을 참조하십시오. 값이 설정되지 않거나 유효한 값으로 설정되지 않은 경우 Amazon S3은 상태 코드가 204인 빈 문서를 반환합니다.  Note 일부 버전의 Adobe Flash Player는 본문이 비어 있는 HTTP 응답을 올바르게 처리하지 못합니다. Adobe Flash를 통한 업로드를 지원하려면 success_action_status 를 201로 설정하는 것이 바람직합니다.	아니요

필드 이름	설명	필수
signature	제공된 AWSSignatureVersion에 해당하는 보안 액세스 키를 사용하여 생성하는 HMAC 서명입니다. 요청과 함께 정책 문서가 포함되는 경우 이 필드는 필수입니다. 자세한 내용은 Amazon S3의 자격 증명 및 액세스 관리를 참조하세요.	조건부
x-amz-security-token	세션 자격 증명이 사용하는 보안 토큰 요청이 Amazon DevPay를 사용하는 경우 제품 토큰과 사용자 토큰에 하나씩 2개의 x-amz-security-token 양식 필드가 요청에 필요합니다. 요청이 세션 자격 증명을 사용 중인 경우 이 요청에는 x-amz-security-token 양식이 필요합니다. 자세한 내용은 IAM 사용 설명서의 임시 보안 자격 증명을 참조하세요.	아니요
x-amz-meta- 접두사가 붙는 기타 필드 이름	사용자 지정 메타데이터입니다. Amazon S3은 이 데이터를 확인하거나 사용하지 않습니다. 자세한 내용은 PUT Object 단원을 참조하십시오.	아니요
file	파일 또는 텍스트 콘텐츠입니다. 파일이나 콘텐츠는 양식의 마지막 필드이어야 합니다. 그 다음 필드는 모두 무시됩니다. 한 번에 파일 하나만 업로드할 수 있습니다.	예

정책 구성

주제

- [Expiration](#)
- [조건](#)
- [조건 매칭](#)
- [문자 이스케이프](#)

정책은 요청이 충족해야 하는 조건을 지정하고 콘텐츠를 인증하는 데 사용되는 UTF-8 및 Base64로 인코딩된 JSON 문서입니다. 정책 문서를 디자인하는 방법에 따라 모든 업로드에 대하여 업로드 단위나 사용자 단위로 혹은 필요에 맞는 다른 디자인에 따라 문서를 사용할 수 있습니다.

Note

정책 문서가 선택 사항이긴 하지만, 버킷을 공개적으로 읽기 가능하도록 설정하는 것보다는 정책 문서를 적극 권장합니다.

다음은 정책 문서의 예입니다.

```
{ "expiration": "2007-12-01T12:00:00.000Z",  
  "conditions": [  
    {"acl": "public-read" },  
    {"bucket": "awsexamplebucket1" },  
    ["starts-with", "$key", "user/eric/"],  
  ]  
}
```

정책 문서는 만료 및 조건을 포함합니다.

Expiration

만료 요소는 ISO 8601 UTC 데이터 형식으로 된 정책의 만료 날짜를 지정합니다. 예를 들어, "2007-12-01T12:00:00.000Z"는 해당 정책이 2007년 12월 1일 자정(UTC)이 지나면 유효하지 않다는 것을 명시합니다. 만료는 정책에 필수 사항입니다.

조건

정책 문서에 포함되는 조건은 업로드된 객체의 내용에 대한 유효성 검증에 사용됩니다. 양식에 지정하는 각 양식 필드(x-ignore- 접두사가 오는 AWSAccessKeyId, 서명, 파일, 정책 및 필드 이름 제외)는 조건 목록에 포함되어야 합니다.

Note

이름이 같은 필드가 여러 개 있는 경우 해당 값을 쉼표로 구분해야 합니다. 예를 들어, "x-amz-meta-tag"라 명명된 필드가 2개이고 첫 번째 필드 값이 "Ninja", 두 번째 필드 값이 "Stallman"이라면 정책 이름을 Ninja,Stallman이라 설정할 것입니다.

이 양식에 들어 있는 모든 변수가 확장돼야 정책의 유효성이 검증됩니다. 따라서 이렇게 확장된 필드에 대하여 모든 조건을 대조해야 합니다. 예를 들어, 키 필드를 user/betty/\${filename}으로 설정했다면 ["starts-with", "\$key", "user/betty/"]와 같은 정책을 사용할 수 있습니다. ["starts-with", "\$key", "user/betty/\${filename}"]는 입력할 수 없습니다. 자세한 내용은 [조건 매칭](#) 섹션을 참조하세요.

다음 표에서는 정책 문서 조건을 설명합니다.

요소 이름	설명
acl	ACL이 충족해야 하는 조건을 지정합니다. 하드 매칭 및 starts-with 를 지원합니다.
content-length-range	업로드되는 콘텐츠에 대하여 허용 가능한 최소 및 최대 크기를 지정합니다. 범위 매칭을 지원합니다.

요소 이름	설명
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	REST 지정 헤더입니다. 하드 매칭 및 starts-with 를 지원합니다.
키	업로드된 키의 이름입니다. 하드 매칭 및 starts-with 를 지원합니다.
success_action_redirect, redirect	업로드 완료 시 클라이언트가 리디렉션되는 URL입니다. 하드 매칭 및 starts-with 를 지원합니다.
success_action_status	success_action_redirect를 지정하지 않을 경우 업로드 완료 시 클라이언트에 반환되는 상태 코드입니다. 하드 매칭을 지원합니다.
x-amz-security-token	Amazon DevPay 보안 토큰입니다. Amazon DevPay를 사용하는 각 요청에는 제품 토큰과 사용자 토큰에 하나씩 2개의 x-amz-security-token 양식 필드가 필요합니다. 따라서 해당 값을 쉼표로 구분해야 합니다. 예를 들어, 사용자 토큰이 ew91dHViZQ== 이고 제품 토큰이 b0hnNVNKWVJIQTA= 인 경우 정책 항목을 { "x-amz-security-token": "ew91dHViZQ==,b0hnNVNKWVJIQTA=" } 로 설정합니다.
x-amz-meta- 접두사가 붙는 기타 필드 이름	사용자 지정 메타데이터입니다. 하드 매칭 및 starts-with 를 지원합니다.

Note

도구 키트로 필드를 추가하는 경우(예: Flash에 의한 파일 이름 추가 등) 정책 문서에 해당 필드를 추가해야 합니다. 이 기능을 제어할 수 있는 경우 해당 필드에 x-ignore- 접두사를 붙여야 Amazon S3이 해당 기능을 무시하고 나중 버전의 기능에 영향을 주지 않습니다.

조건 매칭

다음 표에서는 조건 매칭 유형을 설명합니다. 양식에 지정하는 양식 필드마다 조건을 하나씩 지정해야 하지만 양식 필드 하나에 여러 조건을 지정하면 좀 더 복잡한 매칭 조건을 만들 수 있습니다.

Condition	설명
완전 매치	완전 매치로 필드가 특정 값과 일치하는지 확인합니다. 이 예제는 ACL이 public-read로 설정되어야 한다는 것을 나타냅니다. <pre>{"acl": "public-read" }</pre> 이 예제를 통해 ACL이 public-read로 설정되어야 함을 알 수 있습니다. <pre>["eq", "\$acl", "public-read"]</pre>
Starts With(다음으로 시작)	값이 특정 값으로 시작해야 하는 경우 starts-with를 사용합니다. 이 예제는 키가 user/betty로 시작해야 한다는 것을 나타냅니다. <pre>["starts-with", "\$key", "user/betty/"]</pre>
내용 무관 매칭	필드에 들어 있는 모든 내용을 허용하도록 구성하려면 값이 비어 있는 starts-with를 사용합니다. 이 예제에서는 어떤 success_action_redirect도 허용됩니다. <pre>["starts-with", "\$success_action_redirect", ""]</pre>

Condition	설명
범위 지정	범위를 허용하는 필드의 경우 상위 및 하위 범위를 쉼표로 구분하십시오. 이 예제에서는 1~10MB의 파일 크기가 허용됩니다. <pre>["content-length-range", 1048579, 10485760]</pre>

문자 이스케이프

다음 표에서는 정책 문서 내에서 이스케이프되어야 하는 문자들을 설명합니다.

이스케이프 시퀀스	설명
\\	백슬래시
\\\$	달러 기호
\\b	백스페이스
\\f	용지 공급
\\n	줄 바꿈
\\r	캐리지 리턴
\\t	가로 탭
\\v	세로 탭
\\uXXXX	모든 유니코드 문자

서명 생성

단계	설명
1	UTF-8을 이용하여 정책을 인코딩합니다.
2	Base64를 사용하여 해당 UTF-8을 인코딩합니다.
3	HMAC SHA-1을 사용하여 보안 액세스 키가 있는 정책에 서명합니다.
4	Base64를 사용하여 SHA-1 서명을 인코딩합니다.

인증에 대한 일반적인 내용은 [Amazon S3의 자격 증명 및 액세스 관리](#)를 참조하세요.

리디렉션

이 단원에서는 리디렉션 처리 방법을 설명합니다.

일반 리디렉션

POST 요청이 완료되면 사용자가 `success_action_redirect` 필드에 직접 지정한 위치로 리디렉션됩니다. Amazon S3이 URL을 해석할 수 없는 경우 `success_action_redirect` 필드는 무시됩니다.

`success_action_redirect`를 지정하지 않을 경우 Amazon S3은 `success_action_status` 필드에 지정된 빈 문서 유형을 반환합니다.

POST 요청이 실패하는 경우 Amazon S3이 오류를 표시하고 리디렉션을 제공하지 않습니다.

서전 업로드 리디렉션

<CreateBucketConfiguration>을 사용하여 버킷을 만든 경우 최종 사용자의 경우 리디렉션이 필요할 수 있습니다. 이렇게 되는 경우 일부 브라우저에서 리디렉션이 잘못 처리될 수도 있습니다. 이러한 경우는 비교적 드물게 일어나지만 버킷을 만든 직후 발생할 가능성이 가장 높습니다.

업로드 예제(AWS 서명 버전 2)

주제

- [파일 업로드](#)
- [텍스트 영역 업로드](#)

Note

이 섹션에서 다루는 요청 인증은 AWS 서비스에 대한 API 인바운드 요청을 인증하는 프로토콜, AWS 서명 버전 2 기반입니다.

Amazon S3는 이제 모든 AWS 리전에서 AWS 서비스에 대한 인바운드 API 요청을 인증하기 위한 프로토콜인 서명 버전 4를 지원합니다. 현재 2014년 1월 30일 이전에 생성된 AWS 리전이라고 해도 이전 프로토콜인 서명 버전 2에 대한 지원은 계속됩니다. 하지만 2014년 1월 30일 이후에 새롭게 생성된 리전은 모두 서명 버전 4만 지원하며, 이에 따라 새로운 리전에 대한 모든 요청은 서명 버전 4를 이용해야 합니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 [예제: HTTP POST를 사용하여 브라우저 기반 업로드\(AWS 서명 버전 4 사용\)](#)를 참조하세요.

파일 업로드

이 예제에서는 파일 첨부 업로드에 사용할 수 있는 정책 및 양식의 생성 절차를 처음부터 끝까지 보여 줍니다.

정책 및 양식 생성

다음 정책은 awsexamplebucket1 버킷의 Amazon S3에 대한 업로드를 지원합니다.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "awsexamplebucket1"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html"},
    ["starts-with", "$Content-Type", "image/"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""]
  ]
}
```

이 정책에 대한 요구 사항은 다음과 같습니다.

- 업로드가 2007년 12월 1일 12:00 UTC 이전에 발생해야 합니다.
- 콘텐츠는 `awsexamplebucket1` 버킷에 업로드되어야 합니다.
- 키가 `"user/eric/"`로 시작해야 합니다.
- ACL이 `public-read`로 설정되었습니다.
- `success_action_redirect`가 `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html`로 설정되어 있습니다.
- 객체가 이미지 파일입니다.
- `x-amz-meta-uuid` 태그가 `14365123651274`로 설정되어야 합니다.
- `x-amz-meta-tag`는 아무 값도 포함할 수 없습니다.

다음은 이 정책이 Base64로 인코딩된 버전입니다.

eyJ0eXhwaXJhdGlubiI6ICIyMDA3LTExVDEyOjAwMjAwLjAwMFoiLAogICJjb25kaXRpb25zIjoqWwogICAgeyJidWN

자격 증명을 사용하여 예컨대 앞서 다른 정책 문서의 서명인 0RavWzkygo6QX9caELEqKi9kDbU=와 같은 서명을 만듭니다.

다음 양식은 이 정책을 사용하는 amzn-s3-demo-bucket 버킷에 대한 POST 요청을 지원합니다.

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
    ...
    <form action="https://amzn-s3-demo-bucket.s3.us-west-1.amazonaws.com/" method="post"
    enctype="multipart/form-data">
      Key to upload: <input type="input" name="key" value="user/eric/" /><br />
      <input type="hidden" name="acl" value="public-read" />
      <input type="hidden" name="success_action_redirect" value="https://
awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html" />
      Content-Type: <input type="input" name="Content-Type" value="image/jpeg" /><br />
      <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
      Tags for File: <input type="input" name="x-amz-meta-tag" value="" /><br />
```

```



```

예제 요청

이 요청은 업로드된 이미지가 117,108바이트라 가정하며 이때 이미지 데이터는 포함되지 않습니다.

```

POST / HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.10) Gecko/20071115
  Firefox/2.0.0.10
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=9431149156168
Content-Length: 118698

--9431149156168
Content-Disposition: form-data; name="key"

user/eric/MyPicture.jpg
--9431149156168
Content-Disposition: form-data; name="acl"

public-read
--9431149156168
Content-Disposition: form-data; name="success_action_redirect"

https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html
--9431149156168
Content-Disposition: form-data; name="Content-Type"

image/jpeg

```

```
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-uuid"

14365123651274
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-tag"

Some, Tag, For, Picture
--9431149156168
Content-Disposition: form-data; name="AWSAccessKeyId"

AKIAIOSFODNN7EXAMPLE
--9431149156168
Content-Disposition: form-data; name="Policy"

eyJhZXBwaXJhdGlvbiI6ICIyMDA3LTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWN
--9431149156168
Content-Disposition: form-data; name="Signature"

0RavWzkygo6QX9caELEqKi9kDbU=
--9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename.jpg"
Content-Type: image/jpeg

...file content...
--9431149156168
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
--9431149156168--
```

샘플 응답

```
HTTP/1.1 303 Redirect
x-amz-request-id: 1AEE782442F35865
x-amz-id-2: cxzFLJRatFHy+NGtaDFRR8YvI9BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh
Content-Type: application/xml
Date: Wed, 14 Nov 2007 21:21:33 GMT
Connection: close
Location: https://awsexamplebucket1.s3.us-west-1.amazonaws.com/
successful_upload.html?bucket=awsexamplebucket1&key=user/eric/
MyPicture.jpg&etag="39d459dfbc0faabbb5e179358dfb94c3&quot;
Server: AmazonS3
```

텍스트 영역 업로드

주제

- [정책 및 양식 생성](#)
- [예제 요청](#)
- [샘플 응답](#)

다음 예제는 텍스트 영역의 업로드를 위한 정책 및 양식의 생성 절차를 처음부터 끝까지 보여 줍니다. 텍스트 영역의 업로드는 블로그 게시물과 같은 사용자 제작 콘텐츠의 제출에 유용합니다.

정책 및 양식 생성

다음 정책은 awsexamplebucket1 버킷의 Amazon S3에 대한 텍스트 영역 업로드를 지원합니다.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "awsexamplebucket1"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html"},
    ["eq", "$Content-Type", "text/html"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""]
  ]
}
```

이 정책에 대한 요구 사항은 다음과 같습니다.

- 업로드가 2007년 12월 1일 12:00 GMT 이전에 발생해야 합니다.
- 콘텐츠는 awsexamplebucket1 버킷에 업로드되어야 합니다.
- 키가 "user/eric/"로 시작해야 합니다.
- ACL이 public-read로 설정되었습니다.
- success_action_redirect가 https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html로 설정되어 있습니다.
- 객체가 HTML 텍스트입니다.
- x-amz-meta-uuid 태그가 14365123651274로 설정되어야 합니다.
- x-amz-meta-tag는 아무 값도 포함할 수 없습니다.


```
</html>
```

예제 요청

이 요청은 업로드된 이미지가 117,108바이트라 가정하며 이때 이미지 데이터는 포함되지 않습니다.

```
POST / HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.10) Gecko/20071115
  Firefox/2.0.0.10
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=178521717625888
Content-Length: 118635

-178521717625888
Content-Disposition: form-data; name="key"

ser/eric/NewEntry.html
--178521717625888
Content-Disposition: form-data; name="acl"

public-read
--178521717625888
Content-Disposition: form-data; name="success_action_redirect"

https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html
--178521717625888
Content-Disposition: form-data; name="Content-Type"

text/html
--178521717625888
Content-Disposition: form-data; name="x-amz-meta-uuid"

14365123651274
--178521717625888
Content-Disposition: form-data; name="x-amz-meta-tag"

Interesting Post
```

```
--178521717625888
Content-Disposition: form-data; name="AWSAccessKeyId"

AKIAIOSFODNN7EXAMPLE
--178521717625888
Content-Disposition: form-data; name="Policy"
eyJhZlZxhwaXJhdGlvbiI6IClYMDA3LTExLTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgcyJidWN
--178521717625888
Content-Disposition: form-data; name="Signature"

qA7FWXKq6VvU68lI9KdveT1cWgF=
--178521717625888
Content-Disposition: form-data; name="file"

...content goes here...
--178521717625888
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
--178521717625888--
```

샘플 응답

```
HTTP/1.1 303 Redirect
x-amz-request-id: 1AEE782442F35865
x-amz-id-2: cxzFLJRatFHy+NGtaDFRR8YvI9BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh
Content-Type: application/xml
Date: Wed, 14 Nov 2007 21:21:33 GMT
Connection: close
Location: https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html?
bucket=awsexamplebucket1&key=user/eric/
NewEntry.html&etag=40c3271af26b7f1672e41b8a274d28d4
Server: AmazonS3
```

Adobe Flash를 사용한 POST(AWS 서명 버전 2)

이 단원에서는 Adobe Flash에서 POST를 사용하는 방법을 설명합니다.

Adobe Flash Player 보안

기본적으로 Adobe Flash Player 보안 모델에서 Adobe Flash Player는 SWF 파일이 표시되는 도메인 밖에 있는 서버에 네트워크를 통하여 연결할 수 없습니다.

이 기본값을 다시 정의하려면 POST 업로드가 허용되는 버킷에 공개적으로 읽기 가능한 crossdomain.xml 파일을 업로드해야 합니다. 다음은 예제 crossdomain.xml 파일입니다.

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM
"http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
<allow-access-from domain="*" secure="false" />
</cross-domain-policy>
```

Note

Adobe Flash 보안 모델에 대한 자세한 내용은 Adobe 웹 사이트를 참조하십시오. 버킷에 crossdomain.xml 파일을 추가하면 Adobe Flash Player를 통해 버킷 안의 crossdomain.xml 파일에 연결할 수 있습니다. 단, 실제 Amazon S3 버킷에는 액세스할 수 없습니다.

Adobe Flash 고려 사항

Adobe Flash의 FileReference API는 POST 요청에 Filename 양식 필드를 추가합니다. FileReference API 작업을 사용하여 Amazon S3에 업로드하는 Adobe Flash 애플리케이션을 작성하는 경우 정책에 다음 조건을 포함시키십시오.

```
['starts-with', '$Filename', '']
```

일부 버전의 Adobe Flash Player는 본문이 비어 있는 HTTP 응답을 올바르게 처리하지 못합니다. 본문이 비어 있지 않은 응답을 반환하도록 POST를 구성하려면 success_action_status을 201로 설정하십시오. 그러면 Amazon S3이 상태 코드가 201인 XML 문서를 반환합니다. XML 문서의 내용에 대한 자세한 내용은 [POST Object](#) 단원을 참조하십시오. 양식 필드에 대한 자세한 내용은 [HTML 양식 필드](#) 단원을 참조하십시오.

부록: 수명 주기 구성 API(사용되지 않음)

버킷 수명 주기 구성은 객체 태그를 기반으로 하는 필터를 지원하도록 업데이트됩니다. 즉, 이제 키 이름 접두사, 하나 이상의 객체 태그 또는 둘 다를 지정하는 규칙을 지정하여 규칙이 적용되는 객체의 하위 집합을 선택할 수 있습니다. 그에 따라 API가 업데이트되었습니다. 다음 주제에서는 이전 버전과의 호환성을 위해 PUT 및 GET 버킷 수명 주기 작업의 이전 버전에 대해 설명합니다.

주제

- [PUT 버킷 수명 주기\(사용되지 않음\)](#)
- [GET 버킷 수명 주기\(사용되지 않음\)](#)

PUT 버킷 수명 주기(사용되지 않음)

설명

Important

이 API의 업데이트된 버전은 [PutBucketLifecycleConfiguration](#)을 참조하세요. 이 버전은 사용 중단되었습니다. 기존 수명 주기 구성은 작동합니다. 새 수명 주기 구성의 경우 업데이트된 API를 사용하세요.

버킷에 새로운 수명 주기 구성을 생성하거나 기존 수명 주기 구성을 대체합니다. 수명 주기 구성에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [객체 수명 주기 관리](#)를 참조하세요.

권한

기본적으로 버킷, 객체 및 관련 하위 리소스(예: 수명 주기 구성 및 웹 사이트 구성)를 비롯한 모든 Amazon S3 리소스는 비공개입니다. 리소스 소유자(리소스를 생성한 AWS 계정)만 해당 리소스에 액세스할 수 있습니다. 리소스 소유자는 선택적으로 액세스 정책을 작성하여 다른 사람에게 액세스 권한을 부여할 수 있습니다. 이 작업을 수행하려면 사용자가 s3:PutLifecycleConfiguration 권한을 얻어야 합니다.

권한을 명시적으로 거부할 수도 있습니다. 명시적 거부는 다른 모든 권한에 우선합니다. 사용자 또는 계정이 버킷에서 객체를 제거하거나 삭제하지 못하도록 하려면 다음 작업에 대한 권한을 거부해야 합니다.

- s3:DeleteObject
- s3:DeleteObjectVersion
- s3:PutLifecycleConfiguration

권한에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [Amazon S3 리소스에 대한 액세스 권한 관리](#)를 참조하세요.

요청

구문

```
PUT /?lifecycle HTTP/1.1
```

```
Host: bucketname.s3.amazonaws.com
Content-Length: length
Date: date
Authorization: authorization string
Content-MD5: MD5
```

Lifecycle configuration in the request body

권한 부여 문자열에 대한 자세한 내용은 [요청 인증\(AWS 서명 버전 4\)](#) 섹션을 참조하세요.

요청 파라미터

이 작업 구현에서는 요청 파라미터를 사용하지 않습니다.

요청 헤더

이름	설명	필수
Content-MD5	데이터의 Base64로 인코딩된 128비트 MD5 다 이제스트입니다. 이 헤더를 메시지 무결성 검사로 사용하여 요청 본문이 전송 중에 손상되지 않았는지 확인해야 합니다. 자세한 내용은 RFC 1864를 참조하세요. 유형: 문자열 기본값: None	예

Request Body

요청에서 요청 본문에 수명 주기 구성을 지정합니다. 수명 주기 구성은 XML로 지정됩니다. 다음은 기본 수명 주기 구성의 예입니다. 이 예제는 규칙 하나를 지정합니다. 규칙의 Prefix는 규칙이 적용되는 객체를 식별합니다. 또한 규칙은 두 가지 작업(Transition 및 Expiration)을 지정합니다. 각 작업은 Amazon S3가 해당 작업을 수행해야 하는 타임라인을 지정합니다. Status는 규칙이 활성화 또는 비활성화되었는지 표시합니다.

```
<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Prefix>key-prefix</Prefix>
```

```

    <Status>rule-status</Status>
    <Transition>
      <Date>value</Date>
      <StorageClass>storage class</StorageClass>
    </Transition>
    <Expiration>
      <Days>value</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>

```

버킷의 상태가 버전 관리 활성화 또는 버전 관리 일시 중지인 경우, 동일 객체에 대해 여러 버전을 가질 수 있습니다. 즉, 1개의 현재 버전과 0개 이상의 비 현재 버전이 존재할 수 있습니다. 다음 수명 주기 구성은 최신이 아닌 객체 버전과 관련된 작업(NoncurrentVersionTransition, NoncurrentVersionExpiration)을 지정합니다.

```

<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Prefix>key-prefix</Prefix>
    <Status>rule-status</Status>
    <NoncurrentVersionTransition>
      <NoncurrentDays>value</NoncurrentDays>
      <StorageClass>storage class</StorageClass>
    </NoncurrentVersionTransition>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>value</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
</LifecycleConfiguration>

```

멀티파트 업로드 API를 사용하면 대형 객체를 여러 부분으로 나누어 업로드할 수 있습니다. 멀티파트 업로드에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [멀티파트 업로드 개요](#)를 참조하세요. 수명 주기 구성을 사용하면, 멀티파트 업로드가 지정된 일수 내에 완료되지 않은 경우 미완료 멀티파트 업로드(규칙에 지정된 키 이름 접두사로 식별됨)를 취소하도록 Amazon S3에 지시할 수 있습니다. Amazon S3는 멀티파트 업로드를 취소할 때 멀티파트 업로드와 관련된 모든 파트를 삭제합니다. 이렇게 하면 Amazon S3에 일부 파트가 저장된 상태로 남아 있는 미완료 멀티파트 업로드가 발생하지 않으므로 이러한 파트에 대한 스토리지 비용을 지불할 필요가 없습니다. 다음은 AbortIncompleteMultipartUpload 작업으로 규칙을 지정하는 수명 주기 구성의 예시입니다. 이 작업은 시작 후 7일이 지나면 미완료 멀티파트 업로드를 취소하도록 Amazon S3에 지시합니다.

```
<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Prefix>SomeKeyPrefix</Prefix>
    <Status>rule-status</Status>
    <AbortIncompleteMultipartUpload>
      <DaysAfterInitiation>7</DaysAfterInitiation>
    </AbortIncompleteMultipartUpload>
  </Rule>
</LifecycleConfiguration>
```

다음 표에서는 수명 주기 구성의 XML 요소에 대해 설명합니다.

이름	설명	필수
AbortIncompleteMultipartUpload	미완료 멀티파트 업로드가 중단 작업의 대상이 되는 시점을 지정하기 위한 컨테이너입니다. 하위: DaysAfterInitiation 유형: 컨테이너 상위 항목: Rule	규칙에 다른 작업이 지정되지 않은 경우, 예
Date	Amazon S3가 작업을 수행할 날짜입니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 수명 주기 규칙: 지정된 날짜 기반을 참조하세요. 날짜 값은 ISO 8601 형식을 준수해야 합니다. 시간은 항상 자정(UTC)입니다. 유형: 문자열 상위 항목: Expiration 또는 Transition	Days 및 ExpiredObjectDeleteMarker가 없는 경우, 예
Days		Date 및 ExpiredObjectDeleteMarker

이름	설명	필수
	객체 생성 후 특정 규칙 작업이 적용되기까지의 일수를 지정합니다. 유형: Transition 과 함께 사용할 경우 음이 아닌 정수, Expiration 과 함께 사용할 경우 양의 정수 상위 항목: Expiration , Transition	jectDeleteMarker 가 없는 경우, 예
DaysAfterInitiation	멀티파트 업로드를 시작한 후 해당 멀티파트 업로드가 완료되어야 하는 시점까지의 일수를 지정합니다. 지정된 일수까지 완료되지 않으면 중단 작업을 수행할 수 있으며 Amazon S3는 미완료 멀티파트 업로드를 취소합니다. 유형: 양의 정수 상위 항목: AbortIncompleteMultipartUpload	상위 태그가 지정된 경우, 예

이름	설명	필수
Expiration	이 작업은 Amazon S3가 객체 수명에서 적절한 만료 작업을 수행해야 하는 기간을 지정합니다. Amazon S3가 수행하는 작업은 버킷에서 버전 관리가 활성화되었는지 여부에 따라 달라집니다. - 버킷에서 버전 관리가 활성화되지 않은 경우 Amazon S3는 객체의 유일한 복사본을 영구적으로 삭제합니다. - 버킷에서 버전 관리가 활성화(또는 일시 중지)된 경우, 해당 작업은 객체의 현재 버전에만 적용됩니다. 버전 관리를 사용하는 버킷에는 1개의 현재 버전과 0개 이상의 비 현재 버전 등 동일 객체에 대해 여러 버전이 존재할 수 있습니다. Amazon S3는 현재 객체 버전을 삭제하는 대신 새로운 현재 버전으로 삭제 마커를 추가하여 현재 버전을 비 현재 버전으로 전환합니다.  Important 버킷의 상태가 버전 관리 일시 중지인 경우 Amazon S3는 버전 ID가 null인 삭제 마커를 생성합니다. 버전 ID가 null인 버전이 있는 경우 Amazon S3는 해당 버전을 덮어씁니다.	Rule에 다른 작업이 없는 경우, 예

이름	설명	필수
	 Note 최신이 아닌 객체의 만료를 설정하려면 NoncurrentVersionExpiration 작업을 사용합니다. 유형: 컨테이너 하위 항목: Days 또는 Date 상위 항목: Rule	
ID	규칙의 고유 식별자입니다. 값은 255자보다 길 수 없습니다. 유형: 문자열 상위 항목: Rule	아니요
LifecycleConfiguration	수명 주기 규칙을 위한 컨테이너입니다. 규칙을 최대 1,000개까지 추가할 수 있습니다. 유형: 컨테이너 하위 항목: Rule 상위 항목: 없음	예

이름	설명	필수
ExpiredObjectDeleteMarker	버전이 지정된 버킷(버전 관리가 활성화된 버킷 또는 버전 관리가 일시 중지된 버킷)에서 수명 주기 구성에 이 요소를 추가하여 만료된 객체 삭제 마커를 삭제하도록 Amazon S3에 지시할 수 있습니다. 예제는 Amazon Simple Storage Service 사용 설명서의 예제 8: 만료된 객체 삭제 마커 제거를 참조하세요. 버전이 지정되지 않은 버킷에는 이 요소를 추가하지 마세요. 해당 유형의 버킷은 삭제 마커를 포함할 수 없습니다. 유형: 문자열 유효한 값: true false(값 false는 허용되지만 no-op이므로 Amazon S3가 조치를 취하지 않음) 상위 항목: Expiration	Date 및 Days가 없는 경우, 예
NoncurrentDays	Amazon S3가 관련 작업을 수행하기 전에 객체가 비 현재 상태인 일수를 지정합니다. 최신이 아닌 일수 계산에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 Amazon S3이 최신이 아닌 객체 버전이 된 기간을 계산하는 방법을 참조하세요. 유형: NoncurrentVersionTransition과 함께 사용할 경우 음이 아닌 정수, NoncurrentVersionExpiration과 함께 사용할 경우 양의 정수 상위 항목: NoncurrentVersionExpiration 또는 NoncurrentVersionTransition	예

이름	설명	필수
NoncurrentVersionExpiration	최신이 아닌 객체 버전이 만료되는 시기를 지정합니다. 만료 시 Amazon S3는 최신이 아닌 객체 버전을 영구적으로 삭제합니다. 버전 관리가 활성화(또는 일시 중지)된 버킷에서 이 수명 주기 구성 작업을 설정하여 객체 수명의 특정 기간에 최신이 아닌 객체 버전을 삭제하도록 Amazon S3에 지시합니다. 유형: 컨테이너 하위 항목: NoncurrentDays 상위 항목: Rule	Rule에 다른 작업이 없는 경우, 예
NoncurrentVersionTransition	최신이 아닌 객체가 STANDARD_IA , ONEZONE_IA 또는 스토리지 클래스로 전환되는 시기를 설명하는 전환 규칙을 위한 컨테이너입니다. 버킷에서 버전 관리가 활성화(또는 일시 중지)된 경우, 이 작업을 설정하여 객체 수명의 특정 기간에 최신이 아닌 객체 버전을 전환하도록 Amazon S3에 지시할 수 있습니다. 유형: 컨테이너 하위 항목: NoncurrentDays 및 StorageClass 상위 항목: Rule	Rule에 다른 작업이 없는 경우, 예

이름	설명	필수
Prefix	규칙이 적용되는 하나 이상의 객체를 식별하는 객체 키 접두사입니다. 유형: 문자열 상위 항목: Rule	예
Rule	수명 주기 규칙을 위한 컨테이너입니다. 수명 주기 구성에는 최대 1,000개의 규칙이 포함될 수 있습니다. 유형: 컨테이너 상위 항목: LifecycleConfiguration	예
Status	활성화된 경우 Amazon S3는 예약된 대로 규칙을 실행합니다. 비활성화된 경우 Amazon S3는 규칙을 무시합니다. 유형: 문자열 상위 항목: Rule 유효한 값: Enabled, Disabled	예
StorageClass	객체를 전환하려는 Amazon S3 스토리지 클래스를 지정합니다. 유형: 문자열 상위 항목: Transition 및 NoncurrentVersionTransition 유효한 값: STANDARD_IA ONEZONE_IA GLACIER	예 이 요소는 상위 항목 중 하나 또는 둘 다를 지정하는 경우에만 필수입니다.

이름	설명	필수
Transition	이 작업은 Amazon S3가 객체 수명에서 객체를 STANDARD_IA , ONEZONE_IA 또는 스토리지 클래스로 전환해야 하는 기간을 지정합니다. 이 작업이 적용되면 Amazon S3가 수행하는 작업은 버킷의 버전 관리 활성화 여부에 따라 달라집니다. - 버킷에서 버전 관리가 활성화된 적이 없는 경우 Amazon S3는 객체의 유일한 복사본을 지정된 스토리지 클래스로 전환합니다. - 버킷에 버전 관리가 활성화(또는 일시 중지)된 경우, Amazon S3는 규칙에서 식별된 객체의 현재 버전만 전환합니다.  Note 버전 관리가 활성화된 버킷에는 동일 객체의 여러 버전이 존재할 수 있습니다. 이 작업은 최신이 아닌 객체 버전에는 영향을 주지 않습니다. 최신이 아닌 객체를 전환하려면 NoncurrentVersionTransition 작업을 사용해야 합니다. 유형: 컨테이너 하위 항목: Days 또는 Date, StorageClass 상위 항목: Rule	Rule에 다른 작업이 없는 경우, 예

응답

응답 헤더

이 작업의 구현에서는 대부분의 응답에 공통되는 응답 헤더만 사용합니다. 자세한 내용은 [공통 응답 헤더](#)를 참조하세요.

응답 요소

이 작업의 구현은 응답 요소를 반환하지 않습니다.

특수 오류

이 작업의 구현은 특수 오류를 반환하지 않습니다. 일반적인 Amazon S3 오류에 대한 자세한 내용 및 오류 코드 목록은 [오류 응답](#)을 참조하세요.

예제

예제 1: 버전 관리가 활성화되지 않은 버킷에 수명 주기 구성 추가

다음 수명 주기 구성에서는 작업마다 하나씩 두 개의 규칙을 지정합니다.

- Transition 작업은 'documents/' 접두사가 있는 객체를 생성 후 30일이 지나면 스토리지 클래스로 전환하도록 Amazon S3에 지시합니다.
- Expiration 작업은 'logs/' 접두사가 있는 객체를 생성 후 365일이 지나면 삭제하도록 Amazon S3에 지시합니다.

```
<LifecycleConfiguration>
  <Rule>
    <ID>id1</ID>
    <Prefix>documents/</Prefix>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
  <Rule>
    <ID>id2</ID>
    <Prefix>logs/</Prefix>
```

```

<Status>Enabled</Status>
<Expiration>
  <Days>365</Days>
</Expiration>
</Rule>
</LifecycleConfiguration>

```

다음은 위의 수명 주기 구성을 examplebucket 버킷에 추가하는 샘플 PUT `/?lifecycle` 요청입니다.

```

PUT /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Wed, 14 May 2014 02:11:21 GMT
Content-MD5: q6yJDlIkBaGGfb3QLY69A==
Authorization: authorization string
Content-Length: 415

<LifecycleConfiguration>
  <Rule>
    <ID>id1</ID>
    <Prefix>documents/</Prefix>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
  <Rule>
    <ID>id2</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>

```

다음은 응답 예입니다.

```

HTTP/1.1 200 OK
x-amz-id-2: r+qR7+nhXtJDDIJ0JJYcd+1j5nM/rUFiiiZ/fNbD0sd3JUE8NWMLNHXmvPfwMpdC
x-amz-request-id: 9E26D08072A8EF9E

```

```
Date: Wed, 14 May 2014 02:11:22 GMT
Content-Length: 0
Server: AmazonS3
```

예제 2: 버전 관리가 활성화된 버킷에 수명 주기 구성 추가

다음 수명 주기 구성은 Amazon S3가 수행할 작업마다 하나씩 두 개의 규칙을 지정합니다. 버킷에서 버전 관리가 활성화 또는 일시 중지된 경우 다음 작업을 지정합니다.

- `NoncurrentVersionExpiration` 작업은 'logs/' 접두사가 있는 객체가 최신이 아닌 상태가 된 후 100일이 지나면 객체의 비 현재 버전을 만료하도록 Amazon S3에 지시합니다.
- `NoncurrentVersionTransition` 작업은 'documents/' 접두사가 있는 객체가 최신이 아닌 상태가 된 후 30일이 지나면 객체의 비 현재 버전을 스토리지 클래스로 전환하도록 Amazon S3에 지시합니다.

```
<LifecycleConfiguration>
  <Rule>
    <ID>DeleteAfterBecomingNonCurrent</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>100</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
  <Rule>
    <ID>TransitionAfterBecomingNonCurrent</ID>
    <Prefix>documents/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionTransition>
      <NoncurrentDays>30</NoncurrentDays>
      <StorageClass>GLACIER</StorageClass>
    </NoncurrentVersionTransition>
  </Rule>
</LifecycleConfiguration>
```

다음은 위의 수명 주기 구성을 examplebucket 버킷에 추가하는 샘플 PUT `/?lifecycle` 요청입니다.

```
PUT /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
```

```
x-amz-date: Wed, 14 May 2014 02:21:48 GMT
Content-MD5: 96rxH9mDqVnKkaZDddgnw==
Authorization: authorization string
Content-Length: 598

<LifecycleConfiguration>
  <Rule>
    <ID>DeleteAfterBecomingNonCurrent</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>1</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
  <Rule>
    <ID>TransitionSoonAfterBecomingNonCurrent</ID>
    <Prefix>documents/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionTransition>
      <NoncurrentDays>0</NoncurrentDays>
      <StorageClass>GLACIER</StorageClass>
    </NoncurrentVersionTransition>
  </Rule>
</LifecycleConfiguration>
```

다음은 응답 예입니다.

```
HTTP/1.1 200 OK
x-amz-id-2: aXQ+KbIrmMmo0//3bMdDTw/CnjArwe+J49Hf+j44yRb/VmbIkgI05A+PT98Cp/6k07hf
+LD2mY=
x-amz-request-id: 02D7EC4C10381EB1
Date: Wed, 14 May 2014 02:21:50 GMT
Content-Length: 0
Server: AmazonS3
```

추가 예제

객체를 STANDARD_IA 또는 ONEZONE_IA와 같은 스토리지 클래스로 전환하는 추가 예제는 [수명 주기 구성 예제](#)를 참조하세요.

관련 리소스

- [GetBucketLifecycleConfiguration](#)

- [POST Object restore](#)
- 기본적으로 리소스 소유자, 이 경우 버킷을 생성한 AWS 계정인 버킷 소유자는 모든 작업을 수행할 수 있습니다. 리소스 소유자는 다른 사용자에게 작업을 수행할 수 있는 권한을 부여할 수도 있습니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 다음 주제를 참조하세요.
 - [정책에서 권한 지정](#)
 - [Amazon S3 리소스에 대한 액세스 권한 관리](#)

GET 버킷 수명 주기(사용되지 않음)

설명

Important

이 API의 업데이트된 버전은 [GetBucketLifecycleConfiguration](#)을 참조하세요. <filter> 요소를 사용하여 버킷 수명 주기를 구성한 경우 이 항목의 업데이트된 버전이 표시됩니다. 이 항목은 이전 버전과의 호환성을 위해 제공됩니다.

버킷에 설정된 lifecycle 구성 정보를 반환합니다. 수명 주기 구성에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [객체 수명 주기 관리](#)를 참조하세요.

이 작업을 사용하려면 s3:GetLifecycleConfiguration 작업을 수행할 수 있는 권한이 있어야 합니다. 버킷 소유자는 기본적으로 이 권한을 보유합니다. 버킷 소유자는 이 권한을 다른 사용자에게 부여할 수 있습니다. 권한에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [Amazon S3 리소스에 대한 액세스 권한 관리](#)를 참조하세요.

요청

구문

```
GET /?lifecycle HTTP/1.1
Host: bucketname.s3.amazonaws.com
Date: date
Authorization: authorization string (see ## ##\(AWS ## ## 4\))
```

요청 파라미터

이 작업 구현에서는 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업의 구현에서는 모든 작업에 공통되는 요청 헤더만 사용합니다. 자세한 내용은 [공통 요청 헤더](#)를 참조하세요.

요청 요소

이 작업의 구현에서는 요청 요소를 사용하지 않습니다.

응답

응답 헤더

이 작업의 구현에서는 대부분의 응답에 공통되는 응답 헤더만 사용합니다. 자세한 내용은 [공통 응답 헤더](#)를 참조하세요.

응답 요소

이 GET의 구현은 다음 응답 요소를 반환합니다.

이름	설명	필수
AbortIncompleteMultipartUpload	미완료 멀티파트 업로드가 중단 작업의 대상이 되는 시점을 지정하기 위한 컨테이너입니다. 하위: DaysAfterInitiation 유형: 컨테이너 상위 항목: Rule	규칙에 다른 작업이 지정되지 않은 경우, 예
Date	Amazon S3가 작업을 수행할 날짜입니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 수명 주기 규칙: 지정된 날짜 기반 을 참조하세요. 날짜 값은 ISO 8601 형식을 준수해야 합니다. 시간은 항상 자정(UTC)입니다. 유형: 문자열 상위 항목: Expiration 또는 Transition	Days 및 ExpiredObjectDeleteMarker 가 없는 경우, 예
Days	객체 생성 후 특정 규칙 작업이 적용되기까지의 일수를 지정합니다. 객체의 자격 시간은 생성 시간 + 일수로 계산됩니다(결과 시간은 익일 자정 UTC로 올림됨).	Date 및 ExpiredObjectDeleteMarker 가

이름	설명	필수
	유형: Transition 과 함께 사용할 경우 음이 아닌 정수, Expiration 과 함께 사용할 경우 양의 정수. 상위 항목: Transition 또는 Expiration	없는 경우, 예
DaysAfterInitiation	멀티파트 업로드를 시작한 후 해당 멀티파트 업로드가 완료되어야 하는 시점까지의 일수를 지정합니다. 지정된 일수까지 완료되지 않으면 중단 작업을 수행할 수 있으며 Amazon S3는 미완료 멀티파트 업로드를 취소합니다. 유형: 양의 정수 상위 항목: AbortIncompleteMultipartUpload	Date가 없는 경우, 예

이름	설명	필수
Expiration	이 작업은 Amazon S3가 객체 수명에서 적절한 만료 작업을 수행해야 하는 기간을 지정합니다. 만료 작업은 하위 Date 또는 Days 요소에 지정된 기간에 따라 자격을 충족한 객체에서만 발생합니다. Amazon S3가 수행하는 작업은 버킷에서 버전 관리가 활성화되었는지 여부에 따라 달라집니다. - 버킷에서 버전 관리가 활성화되지 않은 경우 Amazon S3는 객체의 유일한 복사본을 영구적으로 삭제합니다. - 버킷에서 버전 관리가 활성화(또는 일시 중지)된 경우 해당 작업은 객체의 현재 버전에만 적용됩니다. 버전 관리가 활성화 또는 일시 중지된 버킷은 동일 객체에 대해 여러 버전을 가질 수 있습니다. 즉, 1개의 현재 버전과 0개 이상의 비 현재 버전이 존재할 수 있습니다. Amazon S3는 현재 객체 버전을 삭제하는 대신 새로운 현재 버전으로 삭제 마커를 추가하여 현재 버전을 비 현재 버전으로 전환합니다.  Important 버킷의 상태가 버전 관리 일시 중지인 경우 Amazon S3는 버전 ID가 null인 삭제 마커를 생성합니다. 버전 ID가 null인 버전이 있는 경우 Amazon S3는 해당 버전을 덮어씁니다.	상위 태그가 지정된 경우, 예

이름	설명	필수
	 Note 최신이 아닌 객체의 만료를 설정하려면 NoncurrentVersionExpiration 작업을 사용해야 합니다. 유형: 컨테이너 하위 항목: Days 또는 Date 상위 항목: Rule	
ID	규칙의 고유 식별자입니다. 값은 255자보다 길 수 없습니다. 유형: 문자열 상위 항목: Rule	아니요
LifecycleConfiguration	수명 주기 규칙을 위한 컨테이너입니다. 규칙을 최대 1,000개까지 추가할 수 있습니다. 유형: 컨테이너 하위 항목: Rule 상위 항목: 없음	예

이름	설명	필수
ExpiredObjectDeleteMarker	버전이 지정된 버킷(버전 관리가 활성화된 버킷 또는 버전 관리가 일시 중지된 버킷)에서 이 요소는 Amazon S3가 버킷에서 만료된 객체 삭제 마커를 삭제할지 여부를 나타냅니다. 예제는 Amazon Simple Storage Service 사용 설명서의 예제 8: 만료된 객체 삭제 마커를 제거하기 위한 만료 작업 지정을 참조하세요. 유형: 문자열 유효한 값: true false(값 false는 허용되지만 no-op이며, Amazon S3는 값이 false인 경우 조치를 취하지 않음) 상위 항목: Expiration	Date 및 Days가 없는 경우, 예
NoncurrentDays	Amazon S3가 관련 작업을 수행하기 전에 객체가 최신이 아닌 상태를 유지하는 기간(일수)을 지정합니다. 최신이 아닌 일수 계산에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 일수 기반 수명 주기 규칙을 참조하세요. 유형: NoncurrentVersionTransition과 함께 사용할 경우 음이 아닌 정수, NoncurrentVersionExpiration과 함께 사용할 경우 양의 정수 상위 항목: NoncurrentVersionExpiration 또는 NoncurrentVersionTransition	상위 항목이 있는 경우에만 예

이름	설명	필수
NoncurrentVersionExpiration	최신이 아닌 객체 버전이 만료되는 시기를 지정합니다. 만료 시 Amazon S3는 최신이 아닌 객체 버전을 영구적으로 삭제합니다. 버전 관리가 활성화(또는 일시 중단)된 버킷에서 이 수명 주기 구성 작업을 설정하여 객체 수명의 특정 기간에 최신이 아닌 객체 버전을 삭제하도록 Amazon S3에 지시합니다. 유형: 컨테이너 하위 항목: NoncurrentDays 상위 항목: Rule	Rule에 다른 작업이 없는 경우, 예
NoncurrentVersionTransition	최신이 아닌 객체가 STANDARD_IA , ONEZONE_IA 또는 스토리지 클래스로 전환되는 시기를 설명하는 전환 규칙을 위한 컨테이너입니다. 버킷에서 버전 관리가 활성화(또는 일시 중지)된 경우, 이 작업을 설정하여 객체 수명의 특정 기간에 최신이 아닌 객체 버전을 스토리지 클래스로 전환하도록 Amazon S3에 지시할 수 있습니다. 유형: 컨테이너 하위 항목: NoncurrentDays 및 StorageClass 상위 항목: Rule	Rule에 다른 작업이 없는 경우, 예

이름	설명	필수
Prefix	규칙이 적용되는 하나 이상의 객체를 식별하는 객체 키 접두사입니다. 유형: 문자열 상위 항목: Rule	예
Rule	수명 주기 규칙을 위한 컨테이너입니다. 유형: 컨테이너 상위 항목: LifecycleConfiguration	예
Status	활성화된 경우 Amazon S3는 예약된 대로 규칙을 실행합니다. 비활성화된 경우 Amazon S3는 규칙을 무시합니다. 유형: 문자열 상위 항목: Rule 유효한 값: Enabled 또는 Disabled	예
StorageClass	객체를 전환하려는 Amazon S3 스토리지 클래스를 지정합니다. 유형: 문자열 상위 항목: Transition 및 NoncurrentVersionTransition 유효한 값: STANDARD_IA ONEZONE_IA	예

이름	설명	필수
Transition	이 작업은 Amazon S3가 객체 수명에서 객체를 STANDARD_IA , ONEZONE_IA 또는 스토리지 클래스로 전환해야 하는 기간을 지정합니다. 이 작업이 적용되면 Amazon S3가 수행하는 작업은 버킷의 버전 관리 활성화 여부에 따라 달라집니다. - 버킷에서 버전 관리가 활성화된 적이 없는 경우 Amazon S3는 객체의 유일한 복사본을 지정된 스토리지 클래스로 전환합니다. - 버킷에 버전 관리가 활성화(또는 일시 중지)된 경우, Amazon S3는 규칙에서 식별된 객체의 현재 버전만 전환합니다.  Note 버전 관리가 활성화 또는 일시 중지된 버킷에는 동일 객체의 여러 버전이 포함될 수 있습니다. 이 작업은 최신이 아닌 객체 버전에는 영향을 주지 않습니다. 최신이 아닌 객체를 전환하려면 NoncurrentVersionTransition 작업을 사용해야 합니다. 유형: 컨테이너 하위 항목: Days 또는 Date, StorageClass	Rule에 다른 작업이 없는 경우, 예

이름	설명	필수
	상위 항목: Rule	

특수 오류

오류 코드	설명	HTTP 상태 코드	SOAP 오류 코드 접두사
NoSuchLifecycleConfiguration	수명 주기 구성이 없습니다.	404 Not Found(404 찾을 수 없음)	클라이언트

일반적인 Amazon S3 오류에 대한 자세한 내용 및 오류 코드 목록은 [오류 응답](#) 섹션을 참조하세요.

예제

예제 1: 수명 주기 하위 리소스 검색

이 예제는 지정된 버킷에서 lifecycle 하위 리소스를 검색하기 위한 GET 요청 및 반환된 수명 주기 구성이 포함된 예시 응답입니다.

샘플 요청

```
GET /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Thu, 15 Nov 2012 00:17:21 GMT
Authorization: signatureValue
```

샘플 응답

```
HTTP/1.1 200 OK
x-amz-id-2: ITnGT1y4RyTmXa3rPi4hklTXouTf0hccUjo0iCPjz6FnfIutBj3M7fPG1W02SEWp
x-amz-request-id: 51991C342C575321
Date: Thu, 15 Nov 2012 00:17:23 GMT
Server: AmazonS3
Content-Length: 358
```

```
<?xml version="1.0" encoding="UTF-8"?>
<LifecycleConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Rule>
    <ID>Archive and then delete rule</ID>
    <Prefix>projectdocs</Prefix>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>STANDARD_IA</StorageClass>
    </Transition>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
    <Expiration>
      <Days>3650</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

관련 리소스

- [PutBucketLifecycleConfiguration](#)
- [DeleteBucketLifecycle](#)

문서 기록