Amazon Simple Storage Service

개발자 안내서

API 버전 2006-03-01

Amazon Simple Storage Service: 개발자 안내서

Copyright © 2020 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

Amazon S3이란 무엇입니까? 1
방법 1
소개 2
Amazon S3 및 이 안내서의 개요 2
Amazon S3 사용의 장점 2
Amazon S3 개념 3
버킷 3
객체 3
키 3
Regions 4
Amazon S3 데이터 일관성 모델 4
Amazon S3 기능 6
스토리지 클래스 6
버킷 정책 6
AWS Identity and Access Management 7
액세스 통제 목록 7
버전 관리 7
작업 7
Amazon S3 애플리케이션 프로그래밍 인터페이스(API) 8
REST 인터페이스 8
SOAP 인터페이스 8
Amazon S3 요금 지불 9
관련 서비스 9
요청 만들기 10
액세스 키 정보 10
AWS 계정 액세스 키 10
IAM 사용자 액세스 키 10
임시 보안 자격 증명 11
요청 엔드포인트 11
IPv6을 통해 요청 11
IPv6 시작하기 12
IAM 정책에서 IPv6 주소 사용 12
IP 주소 호환성 테스트 13
듀얼 스택 엔드포인트 사용 14
AWS SDK를 사용하여 요청하기 17
AWS 계정 또는 IAM 사용자 자격 증명 사용 18
IAM 사용자 임시 자격 증명 사용 24
연합된 사용자의 임시 자격 증명 사용 32
REST API를 사용하여 요청 42
REST API 요청에 대한 S3 호스트 이름 구성 42
가상 호스팅 방식 및 경로 방식 요청 42
듀얼 스택 엔드포인트(REST API) 43
버킷의 가상 호스팅 43
요청 리디렉션 및 REST API 49
버킷 51
버킷 생성 51
Amazon S3 콘솔 52
REST API 52
AWS SDK 52
권한 정보 53
버킷에 대한 퍼블릭 액세스 관리 53
버킷 액세스 53
가상 호스팅 방식 액세스 54
경로 방식 액세스 54

IPv6을 통해 S3 버킷에 액세스 ... 54
S3 액세스 포인트를 통해 버킷 액세스 ... 54
S3://를 사용하여 버킷 액세스 ... 55
버킷 구성 옵션 ... 55
규제 및 제한 ... 56
이름 지정 규칙 ... 57
버킷 생성 예제 ... 58
Amazon S3 콘솔 사용 ... 59
AWS SDK for Java 사용 ... 59
.NET용 AWS SDK 사용 ... 60
Ruby 버전 3용 AWS SDK 사용 ... 61
다른 AWS SDK 사용 ... 61
버킷 삭제 또는 비우기 ... 61
버킷 삭제 ... 61
버킷 비우기 ... 64
버킷에 대한 기본 암호화 ... 65
Amazon S3 기본 버킷 암호화를 설정하는 방법 ... 65
교차 계정 작업에 암호화 사용 ... 66
복제에 기본 암호화 사용 ... 66
CloudTrail 및 CloudWatch를 사용한 기본 암호화 모니터링 ... 66
추가 정보 ... 67
Transfer Acceleration ... 67
Transfer Acceleration은 왜 사용합니까? ... 67
시작하기 ... 68
Amazon S3 Transfer Acceleration을 사용하기 위한 요구 사항 ... 69
Transfer Acceleration 예 ... 69
요청자 지불 버킷 ... 74
콘솔을 사용한 구성 ... 74
REST API를 사용한 구성 ... 75
요금 세부 정보 ... 77
액세스 제어 ... 77
결제 및 사용 보고 ... 77
결제 보고서 ... 78
사용 보고서 ... 79
결제 및 사용 보고서 이해 ... 81
비용 할당 태그 사용 ... 88
액세스 포인트 ... 90
액세스 포인트 생성 ... 90
Amazon S3 액세스 포인트 이름 지정 규칙 ... 91
Virtual Private Cloud(VPC)로 제한된 액세스 포인트 생성 ... 91
액세스 포인트에 대한 퍼블릭 액세스 관리 ... 93
액세스 포인트를 사용하도록 IAM 정책 구성 ... 93
액세스 포인트 사용 ... 97
S3 작업 및 AWS 서비스와의 액세스 포인트 호환성 ... 98
모니터링 및 로깅 ... 99
예제 ... 99
규제 및 제한 ... 100
객체 ... 102
객체 키와 메타데이터 ... 102
객체 키 ... 103
객체 메타데이터 ... 105
스토리지 클래스 ... 107
자주 액세스하는 객체를 위한 스토리지 클래스 ... 108
자주 액세스하는 객체와 자주 액세스하지 않는 객체를 자동으로 최적화하는 스토리지 클래스 ... 108
자주 액세스하지 않는 객체를 위한 스토리지 클래스 ... 109
객체 아카이빙을 위한 스토리지 클래스 ... 110
Amazon S3 스토리지 클래스 비교 ... 110

객체의 스토리지 클래스 설정 ... 111
하위 리소스 ... 112
S3 버전 관리 ... 112
객체 태그 지정 ... 114
객체 태그 지정과 관련된 API 작업 ... 116
객체 태그 지정 및 부가 정보 ... 117
객체 태그 관리 ... 120
수명 주기 관리 ... 123
수명 주기 구성을 사용해야 할 때는 언제입니까? ... 123
수명 주기를 구성하는 방법 ... 124
추가 고려 사항 ... 124
수명 주기 구성의 요소 ... 130
수명 주기 구성의 예제 ... 136
수명 주기 구성 설정 ... 146
CORS(Cross-Origin Resource Sharing) ... 154
Cross-Origin 리소스 공유(CORS): 사용 사례 시나리오 ... 155
내 버킷에서 CORS를 구성하려면 어떻게 해야 합니까? ... 155
Amazon S3는 버킷에 대한 CORS 구성을 어떻게 평가합니까? ... 157
CORS 활성화 중 ... 157
CORS 문제 해결 ... 163
객체 관련 작업 ... 163
객체 가져오기 ... 164
객체 업로드 ... 172
객체 복사 ... 209
객체 키 나열 ... 220
객체 삭제 ... 225
객체에서 콘텐츠 선택 ... 243
보관된 객체의 복원 ... 247
보관된 객체의 쿼리 ... 251
스토리지 클래스 분석 ... 255
스토리지 클래스 분석 설정 방법 ... 255
스토리지 클래스 분석 ... 256
스토리지 클래스 분석 데이터를 내보내려면 어떻게 해야 합니까? ... 258
스토리지 클래스 분석 내보내기 파일 레이아웃 ... 259
Amazon S3 Analytics REST APIs ... 260
보안 ... 261
데이터 보호 ... 261
인터네트워크 개인 정보 ... 262
데이터 암호화 ... 262
ID 및 액세스 관리 ... 299
Amazon S3 리소스에 대한 액세스 관리 소개 ... 299
Amazon S3 리소스 액세스 옵션 ... 299
개요 ... 299
Amazon S3에서 요청에 권한을 부여하는 방법 ... 305
제공되는 액세스 정책 옵션 사용 지침 ... 311
예제 안내: 액세스 관리 ... 313
버킷 정책 및 사용자 정책 사용 ... 338
ACL을 사용한 액세스 관리 ... 437
퍼블릭 액세스 차단 ... 450
로깅 및 모니터링 ... 458
규정 준수 확인 ... 459
인벤토리 ... 459
복원성 ... 467
백업 암호화 ... 468
버전 관리 ... 468
객체 잠금 ... 490
인프라 보안 ... 496

구성 및 취약성 분석 ... 497
보안 모범 사례 ... 497
Amazon S3 예방 보안 모범 사례 ... 497
Amazon S3 모니터링 및 감사 모범 사례 ... 500
배치 작업 ... 502
용어 ... 502
기본 사항: 작업 ... 502
작업의 작동 방식 ... 503
매니페스트 지정 ... 503
작업 만들기 ... 504
작업 생성 요청 ... 504
작업 생성 요청 ... 505
배치 작업에 대한 권한 부여 ... 506
작업 ... 509
PUT 객체 복사 ... 510
객체 복원 시작 ... 510
Lambda 함수 호출 ... 511
PUT 객체 ACL ... 518
PUT 객체 태그 지정 ... 519
작업 관리 ... 519
작업 나열 ... 520
작업 세부 정보 보기 ... 520
태그를 사용하여 액세스 제어 및 작업 레이블 지정 ... 520
작업 우선 순위 지정 ... 521
작업 상태 ... 521
작업 실패 추적 ... 523
알림 및 로깅 ... 523
완료 보고서 ... 524
예제 ... 524
작업 태그 예제 ... 524
완료 보고서 예제 ... 529
교차 계정 복사 예제 ... 531
AWS CLI 예제 ... 536
Java 예제 ... 541
정적 웹 사이트 호스팅 ... 548
웹 사이트 엔드포인트 ... 548
웹 사이트 엔드포인트 예제 ... 549
DNS CNAME 추가 ... 549
Route 53와 함께 사용자 지정 도메인 사용 ... 550
웹 사이트 엔드포인트와 REST API 엔드포인트 간의 주요 차이점 ... 550
콘솔을 사용하여 버킷 구성 ... 550
웹 사이트 호스팅 사용 ... 551
인덱스 문서 구성 ... 551
웹 사이트 액세스에 대한 권한 설정 ... 553
(선택 사항) 웹 트래픽 로깅 구성 ... 556
(선택 사항)사용자 지정 오류 문서 구성 ... 557
(선택 사항) 리디렉션 구성 ... 559
프로그래밍 방식으로 버킷 구성 ... 564
AWS SDK for Java 사용 ... 565
.NET용 AWS SDK 사용 ... 566
PHP용 SDK 사용 ... 567
REST API 사용 ... 568
예제 안내 ... 568
정적 웹 사이트 구성 ... 569
사용자 지정 도메인을 사용하여 정적 웹 사이트 구성 ... 574
알림 ... 587
개요 ... 587

이벤트 알림을 활성화하는 방법 588
이벤트 알림 유형 및 대상 590
지원되는 이벤트 유형 590
지원되는 대상 591
객체 키 이름 필터링으로 알림 구성 591
객체 키 이름 필터링을 포함하는 유효한 알림 구성 예제 592
유효하지 않은 접두사/접미사 중첩을 포함하는 알림 구성 예제 594
대상에 이벤트 알림 메시지를 게시할 권한 부여 596
AWS Lambda 함수 호출 권한 부여 596
SNS 주제 또는 SQS 대기열로 메시지를 게시할 권한 부여 596
예제 안내: SNS 및 SQS를 사용하여 버킷 알림 추가 598
안내 요약 598
1단계: Amazon SNS 주제 만들기 599
2단계: Amazon SQS 대기열 만들기 599
3단계: 버킷에 알림 구성 추가 600
4단계: 설정 테스트 603
이벤트 메시지 구조 603
복제 607
객체 복제 유형 607
복제를 사용하는 이유 607
CRR을 사용해야 하는 경우 608
SRR을 사용해야 하는 경우 608
복제 요구 사항 608
Amazon S3가 복제하는 객체 609
복제 가능한 객체 609
복제 불가능한 객체 610
기존 객체 복제 611
관련 주제 611
복제 설정 개요 611
복제 구성 개요 612
복제에 대한 권한 설정 620
추가 복제 구성 623
S3 Replication Time Control (S3 RTC) 624
복제본 소유자 변경 626
암호화된 객체 복제 629
복제 실습 과정 633
예제 1: 동일한 계정에서 버킷 구성 633
예제 2: 서로 다른 계정에서 버킷 구성 642
예제 3: 복제본 소유자 변경 643
예제 4: 암호화된 객체 복제 647
예 5: S3 Replication Time Control (S3 RTC) 651
복제 상태 정보 654
관련 주제 655
애플리케이션 문제 해결 655
관련 주제 656
복제 추가 고려 사항 656
수명 주기 구성 및 객체 복제본 657
버전 관리 구성 및 복제 구성 657
로그 구성 및 복제 구성 657
CRR 및 대상 리전 657
복제 일시 중지 657
관련 주제 657
라우팅 요청 658
요청 리디렉션 및 REST API 658
DNS 라우팅 658
임시 요청 리디렉션 659
영구 요청 리디렉션 661

요청 리디렉션 예제 ... 661
DNS 고려 사항 ... 661
Amazon S3 성능 최적화 ... 663
성능 지침 ... 663
성능 측정 ... 664
수평 확장 ... 664
바이트 범위 가져오기 사용 ... 664
요청 재시도 ... 664
동일 리전에서 Amazon S3 및 Amazon EC2 결합 ... 664
지연 시간을 최소화하기 위해 Transfer Acceleration 사용 ... 665
최신 AWS SDK 사용 ... 665
성능 디자인 패턴 ... 665
자주 액세스하는 콘텐츠 캐싱 ... 665
지연 시간에 민감한 앱의 제한 시간 및 재시도 ... 666
수평 확장 및 요청 병렬화 ... 666
지리적으로 다른 데이터 전송 가속화 ... 667
모니터링 ... 668
모니터링 도구 ... 668
자동 도구 ... 668
수동 도구 ... 668
CloudWatch로 지표 모니터링 ... 669
지표와 차원 ... 670
Amazon S3 CloudWatch 버킷에 대한 일별 스토리지 지표 ... 670
Amazon S3 CloudWatch 요청 지표 ... 670
Amazon S3 CloudWatch 복제 지표 ... 673
Amazon S3 CloudWatch 차원 ... 674
CloudWatch 측정치 보기 ... 675
관련 리소스 ... 676
버킷에 대한 지표 구성 ... 676
CloudWatch 지표 전송은 보장되지 않음 ... 676
지표 구성 필터링 ... 677
지표 구성 추가 방법 ... 677
Amazon S3를 사용한 로깅 ... 678
AWS CloudTrail을 사용하여 API 호출 로깅 ... 679
CloudTrail의 Amazon S3 정보 ... 679
Amazon S3 서버 액세스 로그 및 CloudWatch Logs와 함께 CloudTrail 로그 사용 ... 684
예: Amazon S3 로그 파일 항목 ... 684
관련 리소스 ... 686
CloudTrail을 사용하여 Amazon S3 요청 식별 ... 686
CloudTrail가 Amazon S3에 대한 요청을 캡처하는 방식 ... 687
S3 버킷 및 객체에 대한 CloudTrail 이벤트 로깅 활성화 ... 687
CloudTrail 로그에서 Amazon S3에 대한 요청 식별 ... 687
AWS CloudTrail를 사용하여 Amazon S3 서명 버전 2 요청 식별 ... 689
CloudTrail을 사용하여 Amazon S3 객체에 대한 액세스 식별 ... 691
관련 리소스 ... 686
BitTorrent ... 694
BitTorrent 전송에 대한 요금이 부과되는 방법 ... 694
BitTorrent를 사용하여 Amazon S3에 저장된 객체 검색 ... 695
Amazon S3 및 BitTorrent를 사용하여 콘텐츠 게시 ... 695
오류 처리 ... 697
REST 오류 응답 ... 697
응답 헤더 ... 697
오류 응답 ... 698
SOAP 오류 응답 ... 698
Amazon S3 오류 모범 사례 ... 699
InternalErrors 재시도 ... 699
반복되는 SlowDown 오류에 대한 애플리케이션 튜닝 ... 699

오류 격리 ... 699
Amazon S3 문제 해결 ... 701
증상별 Amazon S3 문제 해결 ... 701
버전 관리를 사용하는 버킷에 대한 요청에 대해 HTTP 503 응답이 크게 증가 ... 701
CORS로 버킷 세트에 액세스할 때 예상하지 못한 동작 ... 701
AWS Support용 Amazon S3 요청 ID 가져오기 ... 702
HTTP를 이용한 요청 ID 가져오기 ... 702
웹 브라우저를 이용한 요청 ID 가져오기 ... 702
AWS SDK를 이용한 요청 ID 가져오기 ... 702
AWS CLI를 이용한 요청 ID 가져오기 ... 704
관련 주제 ... 704
서버 액세스 로깅 ... 705
서버 액세스 로깅 활성화 방법 ... 705
추가 로깅 고려 사항 ... 706
로그 객체 키 형식 ... 707
로그 전송 방법 ... 707
서버 로그 전송은 보장되지 않음 ... 707
버킷 로깅 상태 변경 시 일정 기간에 걸쳐 단계적으로 반영됨 ... 708
콘솔을 이용하여 로깅 활성화 ... 708
프로그래밍 방식으로 로깅 활성화 ... 708
로깅 활성화 ... 708
Log Delivery 그룹에 WRITE 및 READ_ACP 권한 부여 ... 709
예: .NET용 AWS SDK ... 709
관련 리소스 ... 711
로그 형식 ... 711
복사 작업을 위한 추가 로깅 ... 715
사용자 지정 액세 로그 정보 ... 719
확장 가능 서버 액세스 로그 형식 프로그래밍 고려 사항 ... 719
로그 파일 삭제 ... 719
관련 리소스 ... 720
Amazon S3 액세스 로그를 사용하여 Amazon S3 요청 식별 ... 720
요청에 대해 Amazon S3 액세스 로그 활성화 ... 720
요청에 대해 Amazon S3 액세스 로그 쿼리 ... 722
Amazon S3 로그 파일을 사용하여 SigV2 요청 식별 ... 724
Amazon S3 로그 파일을 사용하여 객체 액세스 식별 ... 725
관련 리소스 ... 726
AWS SDK 및 Explorer ... 727
요청 인증에서 서명 버전 지정 ... 728
Amazon S3에서 AWS 서명 버전 2 사용 중지 ... 729
서명 버전 2에서 서명 버전 4로 전환 ... 730
AWS CLI 설정 ... 733
AWS SDK for Java 사용 ... 733
Java API 구성 ... 734
Amazon S3 Java 코드 예제 테스트 ... 735
.NET용 AWS SDK 사용 ... 735
.NET API 구성 ... 735
Amazon S3 .NET 코드 예시 실행 ... 736
PHP용 AWS SDK 사용 및 PHP 예제 실행 ... 736
PHP용 AWS SDK 수준 ... 736
PHP 예제 실행 ... 737
관련 리소스 ... 737
Ruby용 AWS SDK 버전 3 사용 ... 737
Ruby API 구성 ... 737
Ruby 스크립트 예제 테스트 ... 738
AWS SDK for Python (Boto) 사용 ... 738
iOS용 및 Android용 AWS Mobile SDK 사용 ... 738
추가 정보 ... 739

AWS Amplify JavaScript 라이브러리 사용 739
추가 정보 739
부록 740
부록 A: SOAP API 사용 740
공통 SOAP API 요소 740
SOAP 요청 인증 741
SOAP를 사용한 액세스 정책 설정 741
부록 B: 요청 인증(AWS Signature 버전 2) 743
REST API를 사용하여 요청 인증 743
REST 요청 서명 및 인증 745
POST를 사용한 브라우저 기반 업로드 754
리소스 770
SQL 참조 771
SELECT 명령 771
목록 SELECT 771
FROM 절 771
WHERE 절 773
LIMIT 절(Amazon S3 Select만 해당) 774
속성 액세스 774
머리글/속성 이름의 대/소문자 구분 775
예약어를 사용자 정의 용어로 사용 775
스칼라 표현식 776
데이터 형식 776
데이터 형식 변환 776
지원되는 데이터 형식 777
연산자 777
논리 연산자 777
비교 연산자 777
패턴 일치 연산자 778
단일 연산자 778
수학 연산자 778
연산자 우선 순위 778
예약어 779
SQL 함수 782
집계 함수(Amazon S3 Select만 해당) 783
조건 함수 783
변환 함수 784
Date 함수 785
문자열 함수 790
문서 이력 794
이전 업데이트 797
AWS Glossary 810

Amazon S3이란 무엇입니까?

Amazon Simple Storage Service는 인터넷용 스토리지 서비스입니다. 이 서비스는 개발자가 더 쉽게 웹 규모 컴퓨팅 작업을 수행할 수 있도록 설계되었습니다.

Amazon S3에서 제공하는 단순한 웹 서비스 인터페이스를 사용하여 웹에서 언제 어디서나 원하는 양의 데이터를 저장하고 검색할 수 있습니다. 또한 개발자는 Amazon이 자체 웹 사이트의 글로벌 네트워크 운영에 사용하는 것과 같은 높은 확장성과 신뢰성을 갖춘 빠르고 경제적인 데이터 스토리지 인프라에 액세스할 수 있습니다. 이 서비스의 목적은 규모의 이점을 극대화하고 개발자들에게 이러한 이점을 제공하는 것입니다.

이 가이드는 버킷, 액세스 포인트 및 객체와 같은 Amazon S3의 핵심 개념과, Amazon S3 애플리케이션 프로그래밍 인터페이스(API)를 사용하여 이러한 리소스에 대한 작업 방법을 설명합니다.

방법

정보	관련 단원
일반 제품 개요 및 요금	Amazon S3
Amazon S3 빠른 실습 소개	Amazon Simple Storage Service 시작 안내서
Amazon S3 핵심 용어 및 개념에 대한 추가 정보	Amazon S3 소개 (p. 2)
버킷에 대해 어떤 작업을 할 수 있습니까?	Amazon S3 버킷을 사용한 작업 (p. 51)
액세스 포인트에 대해 어떤 작업을 할 수 있습니까?	Amazon S3 액세스 포인트를 사용한 데이터 액세스 관리 (p. 90)
객체에 대해 어떤 작업을 할 수 있습니까?	Amazon S3 객체로 작업 (p. 102)
어떻게 요청합니까?	요청 만들기 (p. 10)
리소스에 대한 액세스를 어떻게 관리합니까?	Amazon S3의 Identity and Access Management (p. 299)

Amazon S3 소개

본 Amazon Simple Storage Service(Amazon S3) 소개는 이 웹 서비스에 대한 세부 요약 정보를 제공합니다. 이 단원을 읽으면 이 제품의 혜택과 사업에의 적용 방법을 이해하게 됩니다.

주제

- Amazon S3 및 이 안내서의 개요 (p. 2)
- Amazon S3 사용의 장점 (p. 2)
- Amazon S3 개념 (p. 3)
- Amazon S3 기능 (p. 6)
- Amazon S3 애플리케이션 프로그래밍 인터페이스(API) (p. 8)
- Amazon S3 요금 지불 (p. 9)
- 관련 서비스 (p. 9)

Amazon S3 및 이 안내서의 개요

Amazon S3에서 제공하는 단순한 웹 서비스 인터페이스를 사용하여 웹에서 언제 어디서나 원하는 양의 데이터를 저장하고 검색할 수 있습니다.

이 안내서에서는 버킷을 만들고, 객체를 저장 및 검색하고, 리소스에 대한 권한을 관리하기 위해 요청을 보내는 방법에 대해 설명합니다. 이 안내서에서는 액세스 제어 및 인증 프로세스에 대해서도 설명합니다. 액세스 제어는 Amazon S3 내에서 객체 및 버킷에 액세스할 수 있는 사람과 액세스 유형(예: 읽기 및 쓰기)을 정의합니다. 인증 프로세스는 Amazon Web Services(AWS)에 액세스하려는 사용자의 자격 증명을 확인합니다.

Amazon S3 사용의 장점

Amazon S3는 의도적으로 단순성 및 견고성에 초점을 두는 최소한의 기능 세트를 사용하여 구축되었습니다. 다음은 Amazon S3을 사용할 때의 일부 장점입니다.

- 버킷 만들기 – 데이터를 저장하는 버킷을 만들고 해당 버킷의 이름을 지정합니다. 버킷은 데이터 스토리지를 위한 Amazon S3의 기본 컨테이너입니다.
- 데이터 저장 – 버킷에 데이터를 무한정으로 저장합니다. Amazon S3 버킷에 객체를 원하는 만큼 업로드할 수 있으며, 각 객체에 최대 5TB의 데이터를 포함할 수 있습니다. 각 객체는 고유한 개발자 할당 키를 사용하여 저장 및 검색합니다.
- 데이터 다운로드 – 데이터를 직접 다운로드하거나 다른 사람이 다운로드할 수 있도록 합니다. 언제든지 데이터를 직접 다운로드하거나 다른 사람이 다운로드하도록 허용할 수 있습니다.
- 권한 – 데이터를 Amazon S3 버킷으로 업로드 또는 다운로드하려는 사용자에게 액세스 권한을 부여하거나 해당 권한을 거부합니다. 3가지 유형의 사용자에게 업로드 및 다운로드 권한을 부여할 수 있습니다. 인증 메커니즘을 사용하면 데이터가 무단으로 액세스되지 않도록 보호하는 데 도움이 될 수 있습니다.
- 표준 인터페이스 – 모든 인터넷 개발 도구 키트에서 사용할 수 있도록 설계된 표준 기반 REST 및 SOAP 인터페이스를 사용합니다.

  Note

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

Amazon S3 개념

이 단원에서는 Amazon S3를 효과적으로 사용하기 위해 이해해야 할 핵심 개념 및 용어에 대해 설명합니다. 가장 먼저 접하게 될 가능성이 높은 것부터 순서대로 제공되어 있습니다.

주제

- 버킷 (p. 3)
- 객체 (p. 3)
- 키 (p. 3)
- Regions (p. 4)
- Amazon S3 데이터 일관성 모델 (p. 4)

버킷

버킷은 Amazon S3에 저장된 객체에 대한 컨테이너입니다. 모든 객체는 어떤 버킷에 포함됩니다. 예를 들어 `photos/puppy.jpg`로 명명된 객체는 미국 서부(오레곤) 리전의 `awsexamplebucket` 버킷에 저장되며 URL `https://awsexamplebucket.s3.us-west-2.amazonaws.com/photos/puppy.jpg`를 사용하여 주소를 지정할 수 있습니다.

버킷은 다음과 같은 여러 가지 용도로 사용됩니다.

- Amazon S3 네임스페이스를 최상위 수준으로 구성합니다.
- 스토리지 및 데이터 전송 요금을 담당하는 계정을 식별합니다.
- 액세스 제어에 사용됩니다.
- 사용량 보고를 위한 집계 단위로 사용됩니다.

특정 AWS 리전에서 생성되도록 버킷을 구성할 수 있습니다. 자세한 내용은 버킷 액세스 (p. 53)를 참조하십시오. 객체가 추가될 때마다 Amazon S3에서 고유한 버전 ID를 생성하고 이를 객체에 할당하도록 버킷을 구성할 수도 있습니다. 자세한 내용은 버전 관리 사용 (p. 468)을 참조하십시오.

버킷에 대한 자세한 내용은 Amazon S3 버킷을 사용한 작업 (p. 51)을 참조하십시오.

객체

객체는 Amazon S3에 저장되는 기본 개체입니다. 객체는 객체 데이터와 메타데이터로 구성됩니다. 데이터 부분은 Amazon S3에서 볼 수 없습니다. 메타데이터는 객체를 설명하는 이름-값 페어의 집합입니다. 여기에는 마지막으로 수정한 날짜와 같은 몇 가지 기본 메타데이터 및 `Content-Type` 같은 표준 HTTP 메타데이터가 포함됩니다. 객체를 저장할 때 사용자 정의 메타데이터를 지정할 수도 있습니다.

객체는 키(이름) 및 버전 ID를 통해 버킷 내에서 고유하게 식별됩니다. 자세한 내용은 키 (p. 3) 및 버전 관리 사용 (p. 468)을 참조하십시오.

키

키는 버킷 내 객체의 고유한 식별자입니다. 버킷 내 모든 객체는 정확히 하나의 키를 갖습니다. 버킷, 키 및 버전 ID의 조합은 각 객체를 고유하게 식별합니다. 따라서 Amazon S3을 "버킷 + 키 + 버전"과 객체 자체 사이의 기본 데이터 맵으로 생각할 수 있습니다. Amazon S3 내 모든 객체는 웹 서비스 엔드포인트, 버킷 이름, 키, 그리고 선택 사항인 버전의 조합을 통해 고유하게 주소를 지정할 수 있습니다. 예를 들어, `https://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl`이라는 URL에서 "`doc`"는 버킷의 이름이고 "`2006-03-01/AmazonS3.wsdl`"은 키입니다.

객체 키에 대한 자세한 내용은 객체 키를 참조하십시오.

Regions

Amazon S3에서 사용자가 만드는 버킷을 저장할 지리적 AWS 리전을 선택할 수 있습니다. 지연 시간 최적화, 비용 최소화, 규정 요구 사항 준수 등 다양한 필요에 따라 리전을 선택할 수 있습니다. 특정 리전에 저장된 객체는 사용자가 명시적으로 객체를 다른 리전으로 전송하지 않는 한 해당 리전을 벗어나지 않습니다. 예를 들어 유럽(아일랜드) 리전에 저장된 객체는 해당 리전을 벗어나지 않습니다.

> **Note**
>
> 계정에 활성화된 AWS 리전에서만 Amazon S3에 액세스하고 해당 기능을 사용할 수 있습니다.

Amazon S3 리전 및 엔드포인트 목록은 AWS 일반 참조의 리전 및 엔드포인트를 참조하십시오.

Amazon S3 데이터 일관성 모델

Amazon S3은 모든 리전의 S3 버킷에 있는 새 객체의 PUT에 대해 한 가지 주의 사항을 제시함으로써 읽기 후 쓰기 일관성을 제공합니다. 주의할 점은 객체를 생성하기 전에 키 이름에 대한 HEAD 또는 GET 요청을 하고 그 후에 바로 객체를 생성하는 경우, 최종 일관성으로 인해 후속 GET 작업에서 객체가 반환되지 않을 수 있다는 것입니다.

Amazon S3은 모든 리전의 덮어쓰기 PUT 및 DELETE에 대한 최종 일관성을 제공합니다.

단일 키에 대한 업데이트는 원자성입니다. 예를 들어 기존 키에 대해 PUT할 경우 이후의 읽기가 기존 데이터 또는 업데이트된 데이터를 반환할 수 있지만 절대로 손상된 데이터 또는 부분 데이터를 반환하지 않습니다.

Amazon S3에서는 AWS 데이터 센터 내의 여러 서버로 데이터를 복제함으로써 고가용성을 구현합니다. PUT 요청이 성공하면 데이터가 안전하게 저장됩니다. 그러나 변경 사항에 대한 정보를 Amazon S3로 복제해야 하는데 이 작업에는 일정 시간이 걸릴 수 있으므로 다음 동작을 관찰할 수 있습니다.

- 프로세스가 Amazon S3로 새 객체를 쓰고 해당 버킷 내에 바로 키를 나열합니다. 변경 사항이 완전히 전파될 때까지 객체가 목록에 나타나지 않을 수 있습니다.
- 프로세스가 기존 객체를 대체하고 바로 읽기를 시도합니다. 변경 사항이 완전히 전파될 때까지 Amazon S3에서 이전 데이터를 반환할 수 있습니다.
- 프로세스가 기존 객체를 삭제하고 바로 읽기를 시도합니다. 삭제가 완전히 전파될 때까지 Amazon S3에서 삭제된 데이터를 반환할 수 있습니다.
- 프로세스가 기존 객체를 삭제하고 해당 버킷 내에 바로 키를 나열합니다. 삭제가 완전히 전파될 때까지 Amazon S3에서 삭제된 객체를 나열할 수 있습니다.

> **Note**
>
> Amazon S3는 현재 객체 잠금을 지원하지 않습니다. 두 PUT 요청을 동시에 같은 키에 대해 실행할 경우 타임스탬프가 최신인 요청이 우선 적용됩니다. 이것이 문제가 되는 경우 객체 잠금 메커니즘을 애플리케이션에 구축해 넣어야 합니다.
> 업데이트는 키를 기반으로 하므로 여러 키에서 원자성 업데이트를 수행할 방법은 없습니다. 예를 들어, 한 키의 업데이트에 의존하는 다른 키의 업데이트를 수행할 수 없습니다. 단, 이 기능을 애플리케이션에 설계해 넣는 경우는 예외입니다.

버킷에도 유사한 일관성 모델이 적용되며 동일한 주의 사항이 있습니다. 예를 들어 버킷을 삭제하고 즉시 모든 버킷을 나열하면 목록에 Amazon S3가 여전히 표시될 수 있습니다.

다음 표에서는 최종적 일관된 읽기 및 일관된 읽기의 특징에 대해 설명합니다.

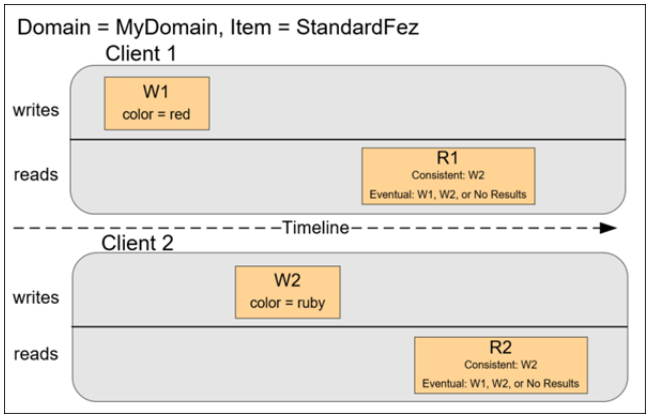
Eventually Consistent Read	Consistent Read
기한 경과 읽기 가능	기한 경과 읽기 불가능

Eventually Consistent Read	Consistent Read
가장 짧은 읽기 지연 시간	잠재적으로 더 긴 읽기 지연 시간
가장 높은 읽기 처리량	잠재적으로 더 낮은 읽기 처리량

동시 애플리케이션

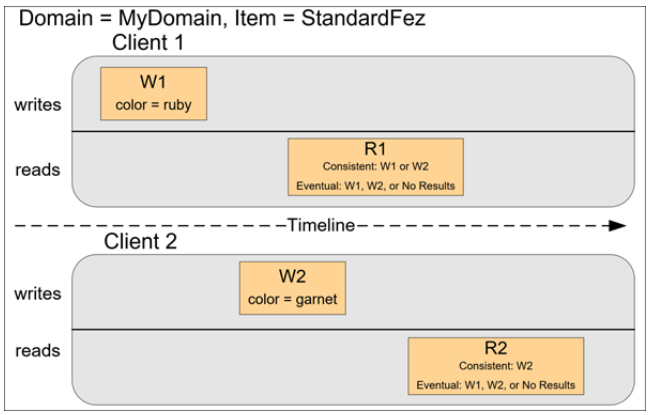
이 단원에서는 다중 클라이언트에서 같은 항목에 쓰고 있을 때의 Eventually Consistent Read 및 Consistent Read 요청에 대한 예제를 제공합니다.

이 예제에서는 W1(쓰기 1)과 W2(쓰기 2) 모두 R1(읽기 1) 및 R2(읽기 2)가 시작되기 전에 완료됩니다. Consistent Read의 경우 R1과 R2 모두 `color = ruby`를 반환합니다. 최종적 일관된 읽기의 경우 경과된 시간에 따라 R1 및 R2가 `color = red` 또는 `color = ruby`를 반환할 수 있습니다.

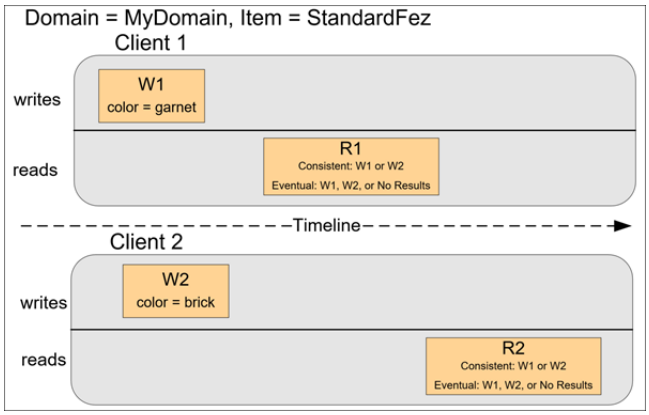


다음 예제에서 W2는 R1이 시작되기 전에 완료되지 않습니다. 따라서 R1이 Consistent Read 또는 Eventually Consistent Read에 대해 `color = ruby` 또는 `color = garnet`을 반환할 수 있습니다. 또한 경과된 시간에 따라 Eventually Consistent Read가 결과를 반환하지 않을 수 있습니다.

Consistent Read의 경우 R2가 `color = garnet`을 반환합니다. 최종적 일관된 읽기의 경우 경과된 시간에 따라 R2가 `color = ruby` 또는 `color = garnet`을 반환할 수 있습니다.



마지막 예제에서 클라이언트 2는 Amazon S3에서 W1에 대해 성공을 반환하기 전에 W2를 수행하므로 최종 값의 결과를 알 수 없습니다(`color = garnet` 또는 `color = brick`). 후속 읽기(Consistent Read 또는 Eventually Consistent Read)를 수행하면 두 값 중 아무 값이나 반환될 수 있습니다. 또한 경과된 시간에 따라 Eventually Consistent Read가 결과를 반환하지 않을 수 있습니다.



Amazon S3 기능

이 단원에서는 중요한 Amazon S3 기능에 대해 설명합니다.

주제

- 스토리지 클래스 (p. 6)
- 버킷 정책 (p. 6)
- AWS Identity and Access Management (p. 7)
- 액세스 통제 목록 (p. 7)
- 버전 관리 (p. 7)
- 작업 (p. 7)

스토리지 클래스

Amazon S3는 사용 사례에 따라 다양하게 설계된 스토리지 클래스를 제공하고 있습니다. 여기에는 자주 액세스하는 데이터의 범용 스토리지를 위한 Amazon S3 STANDARD, 수명이 길지만 액세스 빈도가 덜한 데이터를 위한 Amazon S3 STANDARD_IA 및 장기적 아카이브를 위한 S3 Glacier가 포함됩니다.

자세한 내용은 Amazon S3 스토리지 클래스 (p. 107) 단원을 참조하십시오.

버킷 정책

버킷 정책은 Amazon S3 작업, 요청자, 리소스 및 요청의 여러 측면(예: IP 주소)을 비롯한 다양한 조건을 기반으로 버킷 및 객체에 대한 중앙 집중식 액세스 제어를 제공합니다. 정책은 액세스 정책 언어로 표현되며 중앙 집중식 권한 관리를 가능하게 합니다. 버킷에 연결되는 권한은 해당 버킷의 모든 객체에 적용됩니다.

개인과 기업이 모두 버킷 정책을 사용할 수 있습니다. 기업이 Amazon S3에 등록하면 계정을 만들게 됩니다. 그 후 기업은 계정과 같아집니다. 계정은 자신과 직원이 만드는 AWS 리소스에 대해 재정적 책임을 집니다. 계정은 버킷 정책 권한을 부여하고 다양한 조건을 기반으로 직원에게 권한을 할당할 수 있습니다. 예를 들어, 계정은 사용자에게 다음과 같은 조건으로 쓰기 권한을 부여하는 정책을 만들 수 있습니다.

- 특정 S3 버킷에 대한 권한
- 계정의 기업 네트워크에서 부여
- 업무 시간 중 부여

계정은 한 사용자에 제한된 읽기 및 쓰기 권한을 부여할 수 있지만 다른 사용자가 버킷을 만들고 삭제하도록 허용할 수도 있습니다. 여러 현장 사무소에서는 계정을 통해 일일 보고서를 단일 버킷에 저장할 수 있습니다.

각 사무소는 해당 사무소의 IP 주소 범위 내에서 특정 이름 세트(예: "Nevada/*" 또는 "Utah/*")에만 쓸 수 있습니다.

개별 객체에 대해서만 권한을 추가(부여)할 수 있는 액세스 제어 목록(아래에 설명되어 있음)과 달리, 정책은 버킷 내 모든 객체 또는 객체의 하위 세트에 대해 권한을 추가 또는 거부할 수 있습니다. 계정은 요청 하나를 통해 버킷에 포함된 임의 개수의 객체에 대한 권한을 설정할 수 있습니다. 계정은 Amazon 리소스 이름(ARN) 및 기타 값에 와일드카드(정규 표현식 연산자와 유사)를 사용할 수 있습니다. 그런 다음 계정은 공통 접두사로 시작하거나 .html과 같은 지정된 확장명으로 끝나는 객체 그룹에 대한 액세스를 제어할 수 있습니다.

버킷 소유자만 정책을 버킷에 연결할 수 있습니다. 액세스 정책 언어로 작성되는 정책은 다음을 기반으로 요청을 허용 또는 거부합니다.

- Amazon S3 버킷 작업(예: `PUT ?acl`) 및 객체 작업(예: `PUT Object` 또는 `GET Object`)
- 요청자
- 정책에 지정된 조건

계정은 `GetObject`, `GetObjectVersion`, `DeleteObject` 또는 `DeleteBucket`과 같은 특정 Amazon S3 작업을 기반으로 액세스를 제어할 수 있습니다.

조건은 IP 주소, IP 주소 범위(CIDR 표기법 사용), 날짜, 사용자 에이전트, HTTP 참조자 및 전송(HTTP 및 HTTPS) 등이 될 수 있습니다.

자세한 내용은 버킷 정책 및 사용자 정책 사용 (p. 338) 단원을 참조하십시오.

AWS Identity and Access Management

AWS Identity and Access Management(IAM)를 사용하여 Amazon S3 리소스에 대한 액세스를 관리할 수 있습니다.

예를 들어 IAM을 Amazon S3와 함께 사용하여 사용자 또는 사용자 그룹이 AWS 계정에 속한 Amazon S3 버킷의 특정 부분에 대해 보유한 액세스 유형을 제어할 수 있습니다.

IAM에 대한 자세한 내용은 다음을 참조하십시오.

- AWS Identity and Access Management(IAM)
- 시작하기
- IAM 사용 설명서

액세스 통제 목록

액세스 제어 목록(ACL)을 사용하여 버킷 및 객체 각각에 대한 액세스를 제어할 수 있습니다. 자세한 내용은 ACL을 사용한 액세스 관리 (p. 437) 단원을 참조하십시오.

버전 관리

버전 관리를 사용하면 동일 버킷 내에 여러 개의 객체 버전을 보유할 수 있습니다. 자세한 내용은 객체 버전 관리 (p. 112) 단원을 참조하십시오.

작업

다음은 API를 통해 실행할 가장 일반적인 작업입니다.

일반적인 작업

- 버킷 만들기 – 객체를 저장할 고유한 버킷을 만들고 해당 버킷의 이름을 지정합니다.
- 객체 작성 – 객체를 만들거나 덮어써서 데이터를 저장합니다. 객체를 쓸 때 버킷의 네임스페이스에 고유한 키를 지정합니다. 이 시점에서 객체에 대해 원하는 액세스 제어를 지정하는 것도 좋습니다.
- 객체 읽기 – 데이터를 다시 읽습니다. HTTP 또는 BitTorrent를 통해 데이터를 다운로드할 수 있습니다.
- 객체 삭제 – 일부 데이터를 삭제합니다.
- 키 나열 – 버킷 중 하나에 포함된 키를 나열합니다. 접두사를 기반으로 키 목록을 필터링할 수 있습니다.

이러한 작업과 모든 기타 기능은 이 안내서 전체에 자세히 설명되어 있습니다.

Amazon S3 애플리케이션 프로그래밍 인터페이스 (API)

Amazon S3 아키텍처는 AWS에서 지원하는 인터페이스를 사용하여 객체를 저장 및 검색하는 프로그래밍 언어 중립적 아키텍처로 설계되었습니다.

Amazon S3는 REST 및 SOAP 인터페이스를 제공합니다. 이 둘은 비슷하지만 약간의 차이가 있습니다. 예를 들어, REST 인터페이스에서는 메타데이터가 HTTP 헤더에 반환됩니다. 본문을 제외하고 최대 4KB의 HTTP 요청만 지원되기 때문에 제공할 수 있는 메타데이터의 양이 제한됩니다.

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

REST 인터페이스

REST API는 Amazon S3에 대한 HTTP 인터페이스입니다. REST의 경우, 표준 HTTP 요청을 사용하여 버킷과 객체를 생성하고, 가져오며, 삭제합니다.

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.

SOAP 인터페이스

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

SOAP API는 Document Literal 인코딩을 사용하여 SOAP 1.1 인터페이스를 제공합니다. SOAP를 사용하는 가장 일반적인 방법은 WSDL을 다운로드하고(https://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl 참조) Apache Axis 또는 Microsoft .NET과 같은 SOAP 도구 키트를 사용하여 바인딩을 만든 다음 이 바인딩을 사용하여 Amazon S3을 호출하는 코드를 작성하는 것입니다.

Amazon S3 요금 지불

Amazon S3의 요금은 애플리케이션의 스토리지 요구 사항에 맞춰 계획을 수립할 필요가 없도록 설계되었습니다. 대부분의 스토리지 공급자는 사용자가 사전에 결정된 스토리지 및 네트워크 전송 용량을 구매하도록 합니다. 그리고 사용자가 이 용량을 초과할 경우 서비스를 중단하거나 사용자에게 높은 초과 요금을 부과합니다. 이 용량을 초과하지 않은 경우에도 해당 용량을 모두 사용한 것과 같은 요금을 지불합니다.

Amazon S3는 사용자가 실제로 사용한 만큼만 과금하며, 일체의 부대 비용이나 초과 요금이 없습니다. 이로 인해 AWS 인프라의 비용 이점을 활용하면서 사업과 함께 성장시킬 수 있는 가변 비용 서비스가 개발자에게 제공됩니다.

Amazon S3에 항목을 저장하기 전에 서비스에 등록하고 매월 말에 청구될 결제 방법을 제공해야 합니다. 서비스 사용을 시작하는 데 드는 설정 비용은 없습니다. 매월 말에 제공된 결제 방법을 통해 당월 사용에 대한 요금이 자동으로 청구됩니다.

Amazon S3 스토리지 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

관련 서비스

데이터를 Amazon S3으로 로드한 후에는 해당 데이터를 다른 AWS 서비스에 사용할 수 있습니다. 가장 자주 사용하게 될 서비스는 다음과 같습니다.

- Amazon Elastic Compute Cloud(Amazon EC2) – 이 서비스는 클라우드에서 가상 컴퓨팅 리소스를 제공합니다. 자세한 내용은 Amazon EC2 제품 세부 정보 페이지를 참조하십시오.
- Amazon EMR – 기업, 연구원, 데이터 분석가 및 개발자는 이 서비스를 사용하여 대량의 데이터를 쉽고 비용 효율적으로 처리할 수 있습니다. 이 서비스는 호스팅되는 하둡 프레임워크를 사용합니다. 하둡 프레임워크는 Amazon EC2 및 Amazon S3의 웹 스케일 인프라 상에서 실행됩니다. 자세한 내용은 Amazon EMR 제품 세부 정보 페이지를 참조하십시오.
- AWS Snowball – 이 서비스는 물리적 스토리지 디바이스를 사용함으로써 인터넷을 우회하여 AWS와 대량의 데이터를 빠르게 주고받는 서비스입니다. 각 AWS Snowball 디바이스 유형은 인터넷 속도보다 더 빠르게 데이터를 전송할 수 있습니다. 이 전송은 리전 운송업체를 통해 디바이스의 데이터를 운송하는 방식으로 이루어집니다. 자세한 내용은 AWS Snowball 제품 세부 정보 페이지를 참조하십시오.

요청 만들기

주제

- 액세스 키 정보 (p. 10)
- 요청 엔드포인트 (p. 11)
- IPv6을 통해 Amazon S3에 요청 (p. 11)
- AWS SDK를 사용하여 요청하기 (p. 17)
- REST API를 사용하여 요청 (p. 42)

Amazon S3는 REST 서비스입니다. REST API 또는 프로그래밍 작업을 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 보안 액세스 키)에서 생성됩니다. 액세스 키를 구하는 방법에 대한 자세한 내용은 AWS General Reference의 보안 자격 증명을 가져오는 방법 단원을 참조하십시오.

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

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. 이 액세스 키를 사용하여 Amazon S3에 인증된 요청을 전송할 수 있습니다.

IAM 사용자 액세스 키

회사를 대표하는 하나의 AWS 계정을 만들 수 있지만 여러 명의 직원이 회사의 AWS 리소스에 액세스해야 할 수 있습니다. 이 경우 AWS 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 계정에 개별 액세스 키를 갖는 사용자를 만들고, IAM 사용자 정책에 따라 적절한 리소스 액세스 권한을 부여할 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, AWS에서 만들고 관리합니다. 상위 계정은 사용자의 AWS 액세스 권한을 제어합니다. 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 내 사용자의 생성 및 관리에 대한 자세한 내용은 AWS Identity and Access Management 제품 세부 정보 페이지를 참조하십시오.

임시 보안 자격 증명

개별 액세스 키를 갖는 IAM 사용자를 만드는 것 외, IAM은 또한 모든 IAM 사용자에게 AWS 서비스와 리소스에 액세스하기 위한 임시 보안 자격 증명(임시 액세스 키 및 보안 토큰)을 부여할 수 있으며, AWS 외부의 시스템에서 사용자를 관리할 수도 있습니다. 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 액세스를 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 연동 사용자나 애플리케이션에 전달하기 위해 이러한 임시 보안 자격 증명을 요청할 수 있습니다. 연동 사용자를 위해 임시 보안 자격 증명을 요청할 경우 사용자 이름, 그리고 해당 임시 보안 자격 증명에 연결할 권한을 정의한 IAM 정책을 제공해야 합니다. 연동 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 S3는 요청을 거부합니다.

REST API 요청에서 임시 보안 자격 증명을 사용한 요청 서명에 대한 자세한 내용은 REST 요청 서명 및 인증 (p. 745) 단원을 참조하십시오. AWS SDK를 사용한 요청 전송에 대한 자세한 내용은 AWS SDK를 사용하여 요청하기 (p. 17) 단원을 참조하십시오.

임시 보안 자격 증명에 대한 IAM 지원에 관한 자세한 내용은 IAM 사용 설명서의 임시 보안 자격 증명 단원을 참조하십시오.

보안을 강화하기 위해 Amazon S3 리소스에 액세스할 때 MFA(멀티 팩터 인증)를 요구하도록 버킷 정책을 구성할 수 있습니다. 자세한 정보는 MFA가 필요한 버킷 정책 추가 (p. 411) 단원을 참조하십시오. Amazon S3 리소스에 액세스하려면 MFA를 거치도록 구성할 경우 이러한 리소스에 액세스하려면 항상 MFA 키로 생성된 임시 자격 증명을 제공해야 합니다. 자세한 내용은 AWS Multi-Factor Authentication 세부 정보 페이지 및 IAM 사용 설명서의 MFA 보호 API 액세스 구성 단원을 참조하십시오.

요청 엔드포인트

사전 정의된 서비스의 엔드포인트로 REST 요청을 전송합니다. 모든 AWS 서비스 및 해당 엔드포인트의 목록은 AWS General Reference의 리전 및 엔드포인트에서 확인할 수 있습니다.

IPv6을 통해 Amazon S3에 요청

Amazon Simple Storage Service(Amazon S3)는 IPv4 프로토콜 외에도 인터넷 프로토콜 버전 6(IPv6)을 사용하여 S3 버킷에 액세스하는 기능을 지원합니다. Amazon S3 듀얼 스택 엔드포인트는 IPv6 및 IPv4를 통해 S3 버킷에 요청하는 것을 지원합니다. IPv6을 통해 Amazon S3에 액세스하는 데 대한 추가 요금은 없습니다. 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

주제

- IPv6을 통해 요청하기 (p. 12)
- IAM 정책에서 IPv6 주소 사용 (p. 12)
- IP 주소 호환성 테스트 (p. 13)
- Amazon S3 듀얼 스택 엔드포인트 사용 (p. 14)

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 알고 있어야 하는 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 (p. 14) 단원을 참조하십시오.
- 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 (p. 12) 단원을 참조하십시오.
- 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 Amazon S3 서버 액세스 로그 형식 (p. 711) 및 Amazon S3 서버 액세스 로깅 (p. 705) 단원을 참조하십시오.

  Note

  로그 파일에 IPv6 주소가 있는 것과 관련하여 문제가 있는 경우, AWS Support에 문의하십시오.

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 (p. 14) 단원을 참조하십시오.

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 (p. 15)를 참조하십시오.
- AWS SDK에 대한 내용은 AWS SDK의 듀얼 스택 엔드포인트 사용 (p. 15)를 참조하십시오.
- REST API, REST API를 사용하여 듀얼 스택 엔드포인트에 요청 (p. 43) 참조.

IPv6을 통해 사용할 수 없는 기능

다음은 현재 IPv6을 통해 S3 버킷에 액세스할 때 지원되지 않는 기능입니다.

- S3 버킷에서 정적 웹 사이트 호스팅
- BitTorrent

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. IAM으로 액세스 권한을 관리하는 방법에 대한 자세한 내용은 Amazon S3의 Identity and Access Management (p. 299) 단원을 참조하십시오.

IP 주소를 필터링하는 IAM 정책은 IP 주소 조건 연산자를 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 `examplebucket`에 액세스하지 못하도록 합니다.

```
{
  "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(`54.240.143.0/24`) 및 IPv6(`2001:DB8:1234:5678::/64`) 주소 범위를 모두 허용하도록 버킷 정책의 `Condition` 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 정책이 IPv6 주소 범위를 허용하도록 업데이트해야 합니다. 기존 IPv4 주소 범위 외에도 조직의 IPv6 주소 범위를 포함하도록 IAM 정책을 업데이트하는 것이 좋습니다. IPv6 및 IPv4 모두를 통한 액세스를 허용하는 버킷 정책의 예는 특정 IP 주소 액세스 제한 (p. 409) 단원을 참조하십시오.

IAM 콘솔(https://console.aws.amazon.com/iam/)을 사용하여 IAM 사용자 정책을 검토할 수 있습니다. IAM에 대한 자세한 내용은 IAM 사용 설명서를 참조하십시오. S3 버킷 정책을 편집하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷 정책을 추가하려면 어떻게 해야 합니까? 단원을 참조하십시오.

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
```

Microsoft Windows 7 또는 Windows 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 통해 S3 버킷에 요청하는 것을 지원합니다. 이 단원에서는 듀얼 스택 엔드포인트를 사용하는 방법을 설명합니다.

주제

- Amazon S3 듀얼 스택 엔드포인트 (p. 14)
- AWS CLI의 듀얼 스택 엔드포인트 사용 (p. 15)
- AWS SDK의 듀얼 스택 엔드포인트 사용 (p. 15)
- REST API의 듀얼 스택 엔드포인트 사용 (p. 17)

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 (p. 11) 단원을 참조하십시오.

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 S3는 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 버킷 액세스 (p. 53) 단원을 참조하십시오. Amazon S3 엔드포인트 목록은 AWS General Reference의 리전 및 엔드포인트를 참조하십시오.

> **Important**
>
> 듀얼 스택 엔드포인트를 사용할 경우 Transfer Acceleration을 사용할 수 있습니다. 자세한 내용은 Amazon S3 Transfer Acceleration 시작하기 (p. 68) 단원을 참조하십시오.

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 단원에서는 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 CLI 설정 (p. 733) 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 단원을 참조하십시오.

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 Acceleration를 사용할 수 없습니다. 그러나 곧 AWS CLI를 지원할 예정입니다. 자세한 내용은 AWS Command Line Interface(AWS CLI)에서 Transfer Acceleration 사용 (p. 70) 단원을 참조하십시오.

AWS SDK의 듀얼 스택 엔드포인트 사용

이 단원에서는 AWS SDK를 사용하여 듀얼 스택 엔드포인트에 액세스하는 방법의 예제를 보여 줍니다.

AWS SDK for Java 듀얼 스택 엔드포인트 예제

다음 예제에서는 AWS SDK for Java를 사용하여 Amazon S3 클라이언트 생성 시 듀얼 스택 엔드포인트를 활성화하는 방법을 보여줍니다.

실제 Java 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
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

.NET용 AWS SDK를 사용하는 경우, 다음 예제와 같이 `AmazonS3Config` 클래스를 사용하여 듀얼 스택 엔드포인트를 사용하도록 설정합니다.

```
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.USWest2;
        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());
            }
        }
    }
}
```

객체 나열에 대한 전체 .NET 예제는 .NET용 AWS SDK를 사용한 키 나열 (p. 223) 단원을 참조하십시오.

실제 .NET 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

REST API의 듀얼 스택 엔드포인트 사용

REST API를 사용하여 듀얼 스택 엔드포인트에 요청하는 방법에 대한 자세한 내용은 REST API를 사용하여 듀얼 스택 엔드포인트에 요청 (p. 43) 단원을 참조하십시오.

AWS SDK를 사용하여 요청하기

주제

- AWS 계정 또는 IAM 사용자 자격 증명을 사용하여 요청하기 (p. 18)
- IAM 사용자의 임시 자격 증명을 사용하여 요청 (p. 24)
- 연합된 사용자의 임시 자격 증명을 사용하여 요청하기 (p. 32)

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 단원을 참조하십시오.

AWS 계정 또는 IAM 사용자 자격 증명을 사용하여 요청하기

AWS 계정 또는 IAM 사용자 보안 자격 증명을 사용하여 인증된 요청을 Amazon S3로 보낼 수 있습니다. 이 단원에서는 AWS SDK for Java, .NET용 AWS SDK 및 PHP용 AWS SDK를 사용하여 인증된 요청을 보낼 수 있는 방법의 예제를 제공합니다. 사용 가능한 AWS SDK 목록은 샘플 코드 및 라이브러리 단원을 참조하십시오.

주제

- AWS 계정 또는 IAM 사용자 자격 증명을 사용하여 요청하기 - AWS SDK for Java (p. 19)
- AWS 계정 또는 IAM 사용자 자격 증명을 사용하여 요청하기 - .NET용 AWS SDK (p. 20)
- AWS 계정 또는 IAM 사용자 자격 증명을 사용하여 요청하기 - PHP용 AWS SDK (p. 22)
- AWS 계정 또는 IAM 사용자 자격 증명을 사용하여 요청하기 - Ruby용 AWS SDK (p. 23)

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.

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.

로컬 AWS 자격 증명 파일을 생성하려면

1. AWS Management 콘솔에 로그인한 다음 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 IAM 사용자 생성에 대한 자세한 내용은 IAM 사용자 생성(콘솔) 단원을 참조하고 8단계의 지침을 따릅니다.
3. Download .csv(.csv 다운로드)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 `credentials`라는 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 계정 또는 IAM 사용자 자격 증명을 사용하여 요청하기 - AWS SDK for Java

AWS 계정 또는 IAM 사용자 자격 증명을 사용하여 Amazon S3에 인증된 요청을 보내려면 다음을 수행합니다.

- `AmazonS3ClientBuilder` 클래스를 사용하여 `AmazonS3Client` 인스턴스를 생성합니다.
- `AmazonS3Client` 메서드 중 하나를 실행하여 Amazon S3로 요청을 보냅니다. 클라이언트가 제공된 자격 증명에서 필요한 서명을 생성하여 요청에 포함합니다.

다음 예제에서는 다음과 같은 선행 작업을 수행합니다. 실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

Example

```
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;
import com.amazonaws.services.s3.model.ListObjectsRequest;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.s3.model.S3ObjectSummary;

import java.io.IOException;
import java.util.List;

public class MakingRequests {

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

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Get a list of objects in the bucket, two at a time, and
            // print the name and size of each object.
            ListObjectsRequest listRequest = new
 ListObjectsRequest().withBucketName(bucketName).withMaxKeys(2);
            ObjectListing objects = s3Client.listObjects(listRequest);
            while (true) {
                List<S3ObjectSummary> summaries = objects.getObjectSummaries();
                for (S3ObjectSummary summary : summaries) {
                    System.out.printf("Object \"%s\" retrieved with size %d\n",
 summary.getKey(), summary.getSize());
```

```
                }
                if (objects.isTruncated()) {
                    objects = s3Client.listNextBatchOfObjects(objects);
                } else {
                    break;
                }
            }
        } 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();
        }
    }
}
```

관련 리소스

- AWS SDK, CLI 및 Explorer 사용 (p. 727)

AWS 계정 또는 IAM 사용자 자격 증명을 사용하여 요청하기 - .NET용 AWS SDK

AWS 계정 또는 IAM 사용자 자격 증명을 사용하여 인증된 요청을 보내려면

- `AmazonS3Client` 클래스의 인스턴스를 만듭니다.
- `AmazonS3Client` 메서드 중 하나를 실행하여 Amazon S3로 요청을 보냅니다. 클라이언트가 사용자가 제공하는 자격 증명에서 필요로 하는 서명을 생성하여 Amazon S3로 보내는 요청에 포함시킵니다.

다음 C# 예제에서는 선행 작업을 수행하는 방법을 보여줍니다. 이 안내서의 .NET 예제 실행에 대한 자세한 내용 및 구성 파일에 자격 증명을 저장하는 방법에 대한 지침은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

Example

```
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);
            }
        }
    }
}
```

Note

보안 자격 증명을 제공하지 않고 `AmazonS3Client` 클라이언트를 만들 수 있습니다. 이 클라이언트를 사용하여 보낸 요청은 기본적으로 서명이 없는 익명 요청입니다. Amazon S3는 공개적으로 사용할 수 없는 리소스에 대해 익명 요청을 보내면 오류를 반환합니다.

사용 가능한 예제는 Amazon S3 객체로 작업 (p. 102) 및 Amazon S3 버킷을 사용한 작업 (p. 51)을 참조하십시오. AWS 계정 또는 IAM 사용자 자격 증명을 사용하여 이러한 예제를 테스트할 수 있습니다.

예를 들어, 버킷의 모든 객체 키를 나열하려면 .NET용 AWS SDK를 사용한 키 나열 (p. 223)을 참조하십시오.

관련 리소스

- AWS SDK, CLI 및 Explorer 사용 (p. 727)

AWS 계정 또는 IAM 사용자 자격 증명을 사용하여 요청하기 - PHP용 AWS SDK

이 단원에서는 AWS 계정 또는 IAM 사용자 자격 증명을 사용하여 인증된 요청을 보내기 위해 PHP용 AWS SDK 버전 3의 클래스를 사용하는 방법을 설명합니다. 이미 PHP용 AWS SDK 사용 및 PHP 예제 실행 (p. 736)의 지침에 따라 PHP용 AWS SDK가 올바르게 설치되어 있다고 가정합니다.

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

Example

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

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

$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.

사용 가능한 예제는 객체 관련 작업 (p. 163) 단원을 참조하십시오. AWS 계정 또는 IAM 사용자 자격 증명을 사용하여 이러한 예제를 테스트할 수 있습니다.

버킷의 객체 키를 나열하는 예제는 PHP용 AWS SDK를 사용한 키 나열 (p. 224)을 참조하십시오.

관련 리소스

- Amazon S3용 PHP용 AWS SDK Aws\S3\S3Client 클래스
- PHP용 AWS SDK 설명서

AWS 계정 또는 IAM 사용자 자격 증명을 사용하여 요청하기 - Ruby용 AWS SDK

Ruby용 AWS SDK 버전 3을 사용하여 Amazon S3를 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 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 (p. 18) 단원을 참조하십시오.

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::Resource` 클래스의 인스턴스를 만듭니다.
2. `Aws::S3::Resource`의 `bucket` 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 이를 포함시킵니다.
3. 객체 키 이름 배열을 터미널에 인쇄합니다.

Example

```
# This snippet example does the following:
# Creates an instance of the Aws::S3::Resource class.
# Makes a request to Amazon S3 by enumerating objects in a bucket using the bucket method
 of Aws::S3::Resource.
# The client generates the necessary signature value from the credentials in the AWS
 credentials file on your computer,
# and includes it in the request it sends to Amazon S3.
# Prints the array of object key names to the terminal.
# The credentials that are used for this example come from a local AWS credentials file on
 the computer that is running this application.
# The credentials are for an IAM user who can list objects in the bucket that the user
 specifies when they run the application.

# Use the Amazon S3 modularized gem for version 3 of the AWS Ruby SDK.
require 'aws-sdk-s3'

# Get an Amazon S3 resource.
s3 = Aws::S3::Resource.new(region: 'us-west-2')

# Create an array of up to the first 100 object keynames in the bucket.
bucket = s3.bucket('example_bucket').objects.collect(&:key)

# Print the array to the terminal.
puts bucket
```

로컬 AWS 자격 증명 파일을 가지고 있지 않더라도 여전히 `Aws::S3::Resource` 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 S3는 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. 이 예제에서 사용된 자격 증명은 이 애플리케이션을 실행 중인 컴퓨터의 로컬 AWS 자격 증명 파일에서 나온 것입니다. 자격 증명은 애플리케이션 실행 시 사용자가 지정한 버킷에 객체를 나열할 수 있는 사용자를 위한 것입니다.

```
# This snippet example does the following:
# Creates an instance of the Aws::S3::Resource class.
# Makes a request to Amazon S3 by enumerating objects in a bucket using the bucket method
 of Aws::S3::Resource.
```

```
# The client generates the necessary signature value from the credentials in the AWS
 credentials file on your computer,
# and includes it in the request it sends to Amazon S3.
# Prints the array of object key names to the terminal.
# The credentials that are used for this example come from a local AWS credentials file on
 the computer that is running this application.
# The credentials are for an IAM user who can list objects in the bucket that the user
 specifies when they run the application.

# Use the Amazon S3 modularized gem for version 3 of the AWS Ruby SDK.
require 'aws-sdk-s3'

# Usage: ruby auth_request_test.rb OPERATION BUCKET
# Currently only the list operation is supported

# The operation to perform on the bucket.
operation = 'list' # default
operation = ARGV[0] if (ARGV.length > 0)

if ARGV.length > 1
  bucket_name = ARGV[1]
else
  exit 1
end

# Get an Amazon S3 resource.
s3 = Aws::S3::Resource.new(region: 'us-west-2')

# Get the bucket by name.
bucket = s3.bucket(bucket_name)

case operation

when 'list'
  if bucket.exists?
    # Enumerate the bucket contents and object etags.
    puts "Contents of '%s':" % bucket_name
    puts '  Name => GUID'

    bucket.objects.limit(50).each do |obj|
      puts "  #{obj.key} => #{obj.etag}"
    end
  else
    puts "The bucket '%s' does not exist!" % bucket_name
  end

else
  puts "Unknown operation: '%s'! Only list is supported." % operation
end
```

IAM 사용자의 임시 자격 증명을 사용하여 요청

주제

- IAM 사용자 임시 자격 증명을 사용하여 요청 - AWS SDK for Java (p. 25)
- IAM 사용자 임시 자격 증명을 사용하여 요청 - .NET용 AWS SDK (p. 26)
- AWS 계정 또는 IAM 사용자 임시 자격 증명을 사용하여 요청 - PHP용 AWS SDK (p. 28)
- IAM 사용자 임시 자격 증명을 사용하여 요청 - Ruby용 AWS SDK (p. 30)

AWS 계정 또는 IAM 사용자는 임시 보안 자격 증명을 요청한 후 이 자격 증명을 사용하여 Amazon S3에 인증된 요청을 전송할 수 있습니다. 이 단원에서는 Amazon S3 요청 인증에 사용하기 위해 AWS SDK for Java, .NET 및 PHP로 임시 보안 자격 증명을 구하는 방법에 대한 예를 제공합니다.

IAM 사용자 임시 자격 증명을 사용하여 요청 - AWS SDK for Java

IAM 사용자 또는 AWS 계정은 AWS SDK for Java을 사용하여 임시 보안 자격 증명을 요청(요청 만들기 (p. 10) 참조)한 후 이 자격 증명을 사용하여 Amazon S3에 액세스할 수 있습니다. 세션의 지정된 유효 기간이 만료된 후 이 자격 증명도 만료됩니다. 임시 보안 자격 증명을 사용하려면 다음을 수행합니다.

1. `AWSSecurityTokenService` 클래스의 인스턴스를 만듭니다. 자격 증명 제공에 대한 자세한 내용은 AWS SDK, CLI 및 Explorer 사용 (p. 727) 단원을 참조하십시오.
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 에서 오류를 반환합니다.

   Note

   AWS 계정 보안 자격 증명을 사용하여 구한 임시 자격 증명의 유효 기간은 1시간입니다. IAM 사용자의 자격 증명을 사용하여 세션을 요청할 경우에만 세션 유효 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 IAM 사용자와 관리자 그룹 처음 만들기 단원을 참조하십시오.

실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
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();
        }
    }
}
```

관련 리소스

- AWS SDK, CLI 및 Explorer 사용 (p. 727)

IAM 사용자 임시 자격 증명을 사용하여 요청 - .NET용 AWS SDK

IAM 사용자 또는 AWS 계정은 .NET용 AWS SDK를 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. 임시 보안 자격 증명을 가져오고 에 액세스하려면 다음을 수행합니다.

1. AWS Security Token Service 클라이언트의 인스턴스 `AmazonSecurityTokenServiceClient`를 만듭니다. 자격 증명 제공에 대한 자세한 내용은 AWS SDK, CLI 및 Explorer 사용 (p. 727) 단원을 참조하십시오.

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 구한 임시 보안 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. 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 IAM 사용자와 관리자 그룹 처음 만들기 단원을 참조하십시오. 요청 생성에 대한 자세한 내용은 요청 만들기 (p. 10) 단원을 참조하십시오.

실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

```
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;
            }
        }
    }
}
```

관련 리소스

- AWS SDK, CLI 및 Explorer 사용 (p. 727)

AWS 계정 또는 IAM 사용자 임시 자격 증명을 사용하여 요청 - PHP용 AWS SDK

이 주제에서는 PHP용 AWS SDK 버전 3의 클래스를 사용하여 임시 보안 자격 증명을 요청한 후 이러한 자격 증명을 사용하여 Amazon S3에 액세스하는 방법을 설명합니다. 이미 PHP용 AWS SDK 사용 및 PHP 예제 실행 (p. 736)의 지침에 따라 PHP용 AWS SDK가 올바르게 설치되어 있다고 가정합니다.

IAM 사용자 또는 AWS 계정은 PHP용 AWS SDK 버전 3을 사용하여 임시 보안 자격 증명을 요청할 수 있습니다. 그런 다음 얻은 임시 보안 자격 증명을 사용하여 에 액세스할 수 있습니다. 세션의 유효 기간이 만료되면 이 자격 증명도 만료됩니다. 세션은 기본적으로 1시간 동안 지속됩니다. 사용자 자격 증명을 사용할 경우 임시 보안 자격 증명을 요청할 때 기간(1~36시간)을 지정할 수 있습니다. 임시 보안 자격 증명에 대한 자세한

내용은 IAM 사용 설명서의 임시 보안 자격 증명 단원을 참조하십시오. 요청 생성에 대한 자세한 내용은 요청 만들기 (p. 10) 단원을 참조하십시오.

> **Note**
>
> AWS 계정 보안 자격 증명을 사용하여 구한 임시 보안 자격 증명의 유효 기간은 1시간입니다. IAM 사용자의 자격 증명을 사용하여 세션을 요청할 경우에만 세션 유효 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. 이 가이드의 PHP 예제 실행에 대한 자세한 내용은 PHP 예제 실행 (p. 737) 단원을 참조하십시오.

IAM 사용자 자격 증명을 사용하여 예제를 테스트하려면 AWS 계정에서 IAM 사용자를 생성해야 합니다. IAM 사용자를 만드는 방법에 대한 내용은 IAM 사용 설명서의 IAM 사용자와 관리자 그룹 처음 만들기 단원을 참조하십시오. IAM 사용자 자격 증명을 사용하여 세션을 요청할 경우 세션 유효 기간을 설정하는 방법은 연동 사용자의 임시 자격 증명을 사용하여 요청하기 - PHP용 AWS SDK (p. 37) 단원을 참조하십시오.

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

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

$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;
}
```

관련 리소스

- Amazon S3용 PHP용 AWS SDK Aws\S3\S3Client 클래스
- PHP용 AWS SDK 설명서

IAM 사용자 임시 자격 증명을 사용하여 요청 - Ruby용 AWS SDK

IAM 사용자 또는 AWS 계정은 Ruby용 AWS SDK를 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. 사용자 자격 증명을 사용할 경우 임시 보안 자격 증명을 요청할 때 기간(1~36시간)을 지정할 수 있습니다. 임시 보안 자격 증명 요청에 대한 자세한 내용은 요청 만들기 (p. 10) 단원을 참조하십시오.

> **Note**
>
> AWS 계정 보안 자격 증명을 사용하여 구한 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 증명이 있어야 합니다.

```
# This snippet example does the following:
# The following Ruby example creates a temporary user to list the items in a specified
 bucket
# for one hour. To use this example, you must have AWS credentials that have the necessary
# permissions to create new AWS Security Token Service (AWS STS) clients, and list Amazon
 S3 buckets using temporary security credentials
# using your AWS account security credentials, the temporary security credentials are valid
 for only one hour. You can
# specify session duration only if you use &IAM; user credentials to request a session.

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

USAGE = <<DOC

Usage: assumerole_create_bucket_policy.rb -b BUCKET -u USER [-r REGION] [-d] [-h]

  Assumes a role for USER to list items in BUCKET for one hour.

  BUCKET is required and must already exist.

  USER is required and if not found, is created.

  If REGION is not supplied, defaults to us-west-2.

  -d gives you extra (debugging) information.

  -h displays this message and quits.

DOC

def print_debug(debug, s)
  if debug
    puts s
  end
end

# Get the user if they exist, otherwise create them
def get_user(region, user_name, debug)
```

```
  iam = Aws::IAM::Resource.new(region: region)

  # See if user exists
  user = iam.user(user_name)

  # If user does not exist, create them
  if user == nil
    user = iam.create_user(user_name: user_name)
    iam.wait_until(:user_exists, user_name: user_name)
    print_debug(debug, "Created new user #{user_name}")
  else
    print_debug(debug, "Found user #{user_name} in region #{region}")
  end

  user
end

# main
region = 'us-west-2'
user_name = ''
bucket_name = ''

i = 0

while i &lt; ARGV.length
  case ARGV[i]

    when '-b'
      i += 1
      bucket_name = ARGV[i]

    when '-u'
      i += 1
      user_name = ARGV[i]

    when '-r'
      i += 1

      region = ARGV[i]

    when '-h'
      puts USAGE
      exit 0

    else
      puts 'Unrecognized option: ' + ARGV[i]
      puts USAGE
      exit 1

  end

  i += 1
end

if bucket_name == ''
  puts 'You must supply a bucket name'
  puts USAGE
  exit 1
end

if user_name == ''
  puts 'You must supply a user name'
  puts USAGE
  exit 1
end
```

```
# Create a new Amazon STS client and get temporary credentials. This uses a role that was
 already created.
begin
  creds = Aws::AssumeRoleCredentials.new(
    client: Aws::STS::Client.new(region: region),
    role_arn: "arn:aws:iam::111122223333:role/assumedrolelist",
    role_session_name: "assumerole-s3-list"
  )

  # Create an Amazon S3 resource with temporary credentials.
  s3 = Aws::S3::Resource.new(region: region, credentials: creds)

  puts "Contents of '%s':" % bucket_name
  puts '  Name => GUID'

  s3.bucket(bucket_name).objects.limit(50).each do |obj|
    puts "  #{obj.key} => #{obj.etag}"
  end
rescue StandardError => ex
  puts 'Caught exception accessing bucket ' + bucket_name + ':'
  puts ex.message
end
```

연합된 사용자의 임시 자격 증명을 사용하여 요청하기

임시 보안 자격 증명을 요청하고 이를 AWS 리소스에 액세스해야 하는 연합된 사용자 또는 애플리케이션에 제공할 수 있습니다. 이 단원에서는 AWS SDK를 사용하여 연동 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 단원을 참조하십시오.

> **Note**
>
> AWS 계정과 IAM 사용자 모두 연동 사용자에 대해 임시 보안 자격 증명을 요청할 수 있습니다. 그러나 보안 강화를 위해 필요한 권한이 있는 IAM 사용자만 이러한 임시 자격 증명을 요청해야 연동 사용자가 최대한 요청하는 IAM 사용자의 권한만 얻을 수 있게 됩니다. 일부 애플리케이션에서는 연동 사용자 및 애플리케이션에 임시 보안 자격 증명을 부여하기 위한 목적으로만 특정 권한이 있는 사용자를 만드는 것이 적합할 수도 있습니다.

연동 사용자의 임시 자격 증명을 사용하여 요청하기 - AWS SDK for Java

AWS 리소스에 액세스하기 위해 인증된 요청을 보낼 수 있도록 연동 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**
>
> 연동 사용자 및 애플리케이션에 대한 임시 보안 자격 증명을 요청할 때 보안 강화를 위해 필요한 액세스 권한만 가진 전용 IAM 사용자를 사용할 수 있습니다. 임시 사용자는 임시 보안 자격 증명을 요청한 IAM 사용자보다 더 많은 권한을 가질 수 없습니다. 자세한 내용은 AWS Identity and Access Management FAQ를 참조하십시오.

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

- `AWSSecurityTokenServiceClient` 클래스의 인스턴스를 만듭니다. 자격 증명 제공에 대한 자세한 내용은 AWS SDK for Java 사용 (p. 733) 단원을 참조하십시오.
- 보안 토큰 서비스(STS) 클라이언트의 `getFederationToken()` 메서드를 호출하여 세션을 시작합니다. 임시 자격 증명에 연결하려는 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 먼저 연동 사용자를 위해 2시간짜리 세션에 대한 임시 보안 자격 증명을 가져오고 이 자격 증명을 사용하여 인증된 요청을 Amazon S3로 보냅니다. 이 예제를 실행하려면 사용자가 임시 보안 자격 증명을 요청하고 AWS 리소스를 나열하도록 허용하는 연결된 정책을 사용해 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 IAM 사용자와 관리자 그룹 처음 만들기 단원을 참조하십시오.

IAM 사용자를 생성한 후 이전 정책을 연결하면 다음 예제를 실행할 수 있습니다. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
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();
        }
    }
}
```

관련 리소스

- AWS SDK, CLI 및 Explorer 사용 (p. 727)

연동 사용자의 임시 자격 증명을 사용하여 요청하기 - .NET용 AWS SDK

AWS 리소스에 액세스하기 위해 인증된 요청을 보낼 수 있도록 연동 사용자 및 애플리케이션에 대해 임시 보안 자격 증명을 제공할 수 있습니다. 이러한 임시 자격 증명을 요청할 때는 사용자 이름과 부여할 리소스 권

한을 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 (p. 10) 단원을 참조하십시오.

> **Note**
>
> 연동 사용자 및 애플리케이션에 대한 임시 보안 자격 증명을 요청할 때 보안 강화를 위해 필요한 액세스 권한만 가진 전용 IAM 사용자를 사용하는 것이 좋습니다. 임시 사용자는 임시 보안 자격 증명을 요청한 IAM 사용자보다 더 많은 권한을 가질 수 없습니다. 자세한 내용은 AWS Identity and Access Management FAQ를 참조하십시오.

다음을 수행합니다.

- AWS Security Token Service 클라이언트의 인스턴스인 `AmazonSecurityTokenServiceClient` 클래스를 만듭니다. 자격 증명 제공에 대한 자세한 내용은 .NET용 AWS SDK 사용 (p. 735) 단원을 참조하십시오.
- STS 클라이언트의 `GetFederationToken` 메서드를 호출하여 세션을 시작합니다. 임시 자격 증명에 연결하려는 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 에 요청을 보냅니다. 만료된 자격 증명을 사용하여 요청을 보낼 경우 Amazon S3에서 오류를 반환합니다.

Example

다음 C# 예제는 지정된 버킷의 키를 나열합니다. 이 예제에서는 먼저 연동 사용자(User1)를 위해 2시간짜리 세션에 대한 임시 보안 자격 증명을 가져오고 이 자격 증명을 사용하여 인증된 요청을 Amazon S3로 보냅니다.

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

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

  이 정책은 IAM 사용자에게 임시 보안 자격 증명을 요청하고, AWS 리소스를 목록을 조회할 수만 있는 액세스 권한을 허용합니다. IAM 사용자를 만드는 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 사용자와 관리자 그룹 처음 만들기 단원을 참조하십시오.
- IAM 사용자 보안 자격 증명을 사용하여 다음 예제를 테스트합니다. 이 예제에서는 임시 보안 자격 증명을 사용하여 인증된 요청을 Amazon S3로 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. 실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736)를 참조하십시오.

```
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"":""Stmt1311212314284"",
                   ""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;
        }
    }
}
```

관련 리소스

- AWS SDK, CLI 및 Explorer 사용 (p. 727)

연동 사용자의 임시 자격 증명을 사용하여 요청하기 - PHP용 AWS SDK

이 주제에서는 PHP용 AWS SDK 버전 3의 클래스를 사용하여 연동 사용자 및 애플리케이션에 대한 임시 보안 자격 증명을 요청하고 이를 사용하여 Amazon S3에 저장된 리소스에 액세스하는 방법을 설명합니다. 이

미 PHP용 AWS SDK 사용 및 PHP 예제 실행 (p. 736)의 지침에 따라 PHP용 AWS SDK가 올바르게 설치되어 있다고 가정합니다.

AWS 리소스에 액세스하기 위해 인증된 요청을 보낼 수 있도록 연동 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 단원을 참조하십시오. 연동 사용자와 애플리케이션에 임시 보안 자격 증명 제공에 대한 자세한 내용은 요청 만들기 (p. 10) 단원을 참조하십시오.

연동 사용자 및 애플리케이션에 대한 임시 보안 자격 증명을 요청할 때 보안 강화를 위해 필요한 액세스 권한만 가진 전용 IAM 사용자를 사용하는 것이 좋습니다. 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 FAQ를 참조하십시오.

이 가이드의 PHP 예제 실행에 대한 자세한 내용은 PHP 예제 실행 (p. 737) 단원을 참조하십시오.

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 S3로 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 연동 사용자에게 최소한의 애플리케이션별 권한만 부여하도록 이 IAM 사용자의 액세스 권한을 제한할 수도 있습니다. 이 예제는 특정 버킷의 객체만 나열합니다. 다음과 같은 정책을 연결하여 사용자를 생성합니다.

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

이 정책은 IAM 사용자에게 임시 보안 자격 증명을 요청하고, AWS 리소스를 목록을 조회할 수만 있는 액세스 권한을 허용합니다. IAM 사용자를 만드는 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 사용자와 관리자 그룹 처음 만들기 단원을 참조하십시오.

이제 IAM 사용자 보안 자격 증명을 사용하여 다음 예제를 테스트할 수 있습니다. 이 예제에서는 임시 보안 자격 증명을 사용하여 인증된 요청을 Amazon S3로 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 `YourBucketName`을 자신의 기존 버킷 이름으로 대체합니다.

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

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

$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;
}
```

관련 리소스

- Amazon S3용 PHP용 AWS SDK Aws\S3\S3Client 클래스
- PHP용 AWS SDK 설명서

연동 사용자의 임시 자격 증명을 사용하여 요청하기 - Ruby용 AWS SDK

AWS 리소스에 액세스하기 위해 인증된 요청을 보낼 수 있도록 연동 사용자 및 애플리케이션에 대해 임시 보안 자격 증명을 제공할 수 있습니다. IAM 서비스로부터 임시 자격 증명을 요청할 경우 사용자 이름과, 부여할

리소스 권한을 정의한 IAM 정책을 제공해야 합니다. 세션은 기본적으로 1시간 동안 지속됩니다. 하지만 IAM 사용자 자격 증명을 사용하여 임시 자격 증명을 요청하면, 연동 사용자 및 애플리케이션에 대해 임시 보안 자격 증명을 요청할 때 다른 기간 값을 명시적으로 설정할 수 있습니다. 연동 사용자와 애플리케이션의 임시 보안 자격 증명에 대한 자세한 내용은 요청 만들기 (p. 10) 단원을 참조하십시오.

> **Note**
>
> 연동 사용자 및 애플리케이션에 대한 임시 보안 자격 증명을 요청할 때 보안 강화를 위해 필요한 액세스 권한만 가진 전용 IAM 사용자를 사용할 수 있습니다. 임시 사용자는 임시 보안 자격 증명을 요청한 IAM 사용자보다 더 많은 권한을 가질 수 없습니다. 자세한 내용은 AWS Identity and Access Management FAQ를 참조하십시오.

Example

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

```
require 'aws-sdk-s3'
require 'aws-sdk-iam'

USAGE = <<DOC

Usage: ruby auth_federation_token_request_test.rb -b BUCKET -u USER [-r REGION] [-d] [-h]

  Creates a federated policy for USER to list items in BUCKET for one hour.

  BUCKET is required and must already exist.

  USER is required and if not found, is created.

  If REGION is not supplied, defaults to us-west-2.

  -d gives you extra (debugging) information.

  -h displays this message and quits.

DOC

def print_debug(debug, s)
  if debug
    puts s
  end
end

# Get the user if they exist, otherwise create them
def get_user(region, user_name, debug)
  iam = Aws::IAM::Client.new(region: 'us-west-2')

  # See if user exists
  user = iam.user(user_name)

  # If user does not exist, create them
  if user == nil
    user = iam.create_user(user_name: user_name)
    iam.wait_until(:user_exists, user_name: user_name)
    print_debug(debug, "Created new user #{user_name}")
  else
    print_debug(debug, "Found user #{user_name} in region #{region}")
  end

 user
end

# main
region = 'us-west-2'
```

```
user_name = ''
bucket_name = ''

i = 0

while i < ARGV.length
  case ARGV[i]

    when '-b'
      i += 1
      bucket_name = ARGV[i]

    when '-u'
      i += 1
      user_name = ARGV[i]

    when '-r'
      i += 1
      region = ARGV[i]

    when '-h'
      puts USAGE
      exit 0

    else
      puts 'Unrecognized option: ' + ARGV[i]
      puts USAGE
      exit 1

  end

  i += 1
end

if bucket_name == ''
  puts 'You must supply a bucket name'
  puts USAGE
  exit 1
end

if user_name == ''
  puts 'You must supply a user name'
  puts USAGE
  exit 1
end

# Create a new STS client and get temporary credentials.
sts = Aws::STS::Client.new(region: region)

creds = sts.get_federation_token({
  duration_seconds: 3600,
  name: user_name,
  policy: "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":\"Stmt1\",\"Effect\":\"Allow
\",\"Action\":\"s3:ListBucket\",\"Resource\":\"arn:aws:s3:::#{bucket_name}\"}]}",
})

# Create an Amazon S3 resource with temporary credentials.
s3 = Aws::S3::Resource.new(region: region, credentials: creds)

puts "Contents of '%s':" % bucket_name
puts '  Name => GUID'

begin
  s3.bucket(bucket_name).objects.limit(50).each do |obj|
    puts "  #{obj.key} => #{obj.etag}"
  end
```

```
rescue StandardError => ex
  puts 'Caught exception accessing bucket ' + bucket_name + ':'
  puts ex.message
end
```

REST API를 사용하여 요청

이 단원에는 REST API를 사용하여 Amazon S3 엔드포인트에 요청하는 방법에 대한 정보가 포함되어 있습니다. Amazon S3 엔드포인트 목록은 AWS General Reference의 리전 및 엔드포인트를 참조하십시오.

주제

- REST API 요청에 대한 S3 호스트 이름 구성 (p. 42)
- 가상 호스팅 방식 및 경로 방식 요청 (p. 42)
- REST API를 사용하여 듀얼 스택 엔드포인트에 요청 (p. 43)
- 버킷의 가상 호스팅 (p. 43)
- 요청 리디렉션 및 REST API (p. 49)

REST API 요청에 대한 S3 호스트 이름 구성

Amazon S3 엔드포인트는 아래에 나타낸 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 AWS 일반 참조의 Amazon S3 리전 및 엔드포인트를 참조하십시오.

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 버킷의 가상 호스팅 (p. 43) 단원을 참조하십시오.

Example 가상 호스팅 방식 요청

다음은 미국 서부(오레곤) 리전의 `examplebucket` 버킷에서 `puppy.jpg` 파일을 삭제하기 위한 가상 호스팅 방식의 요청에 대한 예제입니다. 가상 호스팅 방식 요청에 대한 자세한 내용은 가상 호스팅 방식 요청 (p. 44)을 참조하십시오.

```
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
```

Example 경로 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 S3에서는 모든 리전에서 가상 호스팅 스타일 액세스와 경로 스타일 액세스를 지원하지만 변경될 예정입니다(다음 중요 참조).

경로 방식 요청에 대한 자세한 내용은 경로 방식 요청 (p. 44) 단원을 참조하십시오.

> **Important**
>
> 2020년 9월 30일 이후에 생성된 버킷은 가상 호스팅 방식의 요청만 지원합니다. 이 날짜 당일 또는 이전에 생성된 버킷에 대해서는 경로 방식의 요청이 계속 지원됩니다. 자세한 내용은 Amazon S3 Path Deprecation Plan - The Rest of the Story를 참조하십시오.

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.

Example 가상 호스팅 방식 듀얼 스택 엔드포인트 요청

미국 서부(오레곤) 리전의 `examplebucket` 버킷에서 `puppy.jpg` 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
```

Example 경로 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 (p. 14)을 참조하십시오.

버킷의 가상 호스팅

가상 호스팅은 한 웹 서버에 있는 여러 웹 사이트에 서비스를 제공하는 기능입니다. 사이트를 구분하는 한 가지 방법은 URI의 경로 이름 부분 대신 요청의 명백한 호스트 이름을 사용하는 것입니다. 일반적인 Amazon S3 REST 요청은 요청 -URI 경로의 슬래시로 구분된 구성 요소를 사용하여 버킷을 지정합니다. 또는 Amazon S3 가상 호스팅을 통해 HTTP `Host` 헤더를 사용하여 REST API 호출 버킷을 처리할 수 있습니다. 실제로 Amazon S3는 `Host`의 의미를 `https://bucketname.s3.Region.amazonaws.com`에서 대부분의 버킷이 제한된 요청 유형에 대해 자동으로 액세스가 가능하다고 해석합니다(Amazon S3 리전 및 엔드포인트의 전체 목록은 AWS 일반 참조의 Amazon S3 리전 및 엔드포인트를 참조하십시오. 참조).

가상 호스팅에는 다른 이점도 있습니다. 등록된 도메인 이름 뒤에 버킷을 이름 지정하고 해당 이름을 Amazon S3용 DNS 별칭으로 만들어 Amazon S3 리소스의 URL을 `https://my.bucketname.com/`처럼 완전하게 사용자 지정할 수 있습니다. 또한 버킷의 가상 서버에 있는 "루트 디렉터리"에 게시할 수도 있습니다. 이 기능은 기존의 많은 애플리케이션이 이 표준 위치에서 파일을 검색하기 때문에 중요할 수 있습니다. 예를 들어, `favicon.ico`, `robots.txt`, `crossdomain.xml`은 모두 루트에서 검색될 수 있습니다.

주제

- 경로 방식 요청 (p. 44)
- 가상 호스팅 방식 요청 (p. 44)
- HTTP 호스트 헤더 버킷 사양 (p. 44)
- 예제 (p. 45)
- Amazon S3 URL을 CNAME으로 사용자 지정 (p. 46)
- 제한 사항 (p. 47)
- 이전 버전과의 호환성 (p. 47)

경로 방식 요청

현재 Amazon S3에서는 모든 리전에서 가상 호스팅 스타일 액세스와 경로 스타일 액세스를 지원하지만 변경될 예정입니다(다음 중요 참조).

Amazon S3에서 경로 스타일 URL은 다음 형식을 따릅니다.

```
https://s3.Region.amazonaws.com/bucket-name/key name
```

예를 들어 미국 서부(오레곤) 리전에서 이름이 `mybucket`인 버킷을 생성하고 해당 버킷의 `puppy.jpg` 객체에 액세스하려는 경우 다음 경로 스타일 URL을 사용할 수 있습니다.

```
https://s3.us-west-2.amazonaws.com/mybucket/puppy.jpg
```

> **Important**
>
> 2020년 9월 30일 이후에 생성된 버킷은 가상 호스팅 방식의 요청만 지원합니다. 이 날짜 당일 또는 이전에 생성된 버킷에 대해서는 경로 방식의 요청이 계속 지원됩니다. 자세한 내용은 Amazon S3 Path Deprecation Plan - The Rest of the Story를 참조하십시오.

가상 호스팅 방식 요청

가상 호스팅 방식의 URL에서 버킷 이름은 URL에서 도메인 이름의 일부입니다.

Amazon S3 가상 호스팅 스타일 URL은 다음 형식을 따릅니다.

```
https://bucket-name.s3.Region.amazonaws.com/key name
```

이 예제에서 `my-bucket`은 버킷 이름이고, 미국 서부(오레곤)는 리전이며, `puppy.png`는 키 이름입니다.

```
https://my-bucket.s3.us-west-2.amazonaws.com/puppy.png
```

HTTP 호스트 헤더 버킷 사양

`GET` 요청이 SSL 엔드포인트를 사용하지 않는 한 HTTP `Host` 헤더를 사용하여 요청에 대한 버킷을 지정할 수 있습니다. REST 요청의 `Host` 헤더는 다음과 같이 해석됩니다.

- `Host` 헤더가 생략되었거나 그 값이 `s3.Region.amazonaws.com`이면, 요청 버킷은 요청-URI의 슬래시로 구분된 첫 번째 구성 요소가 되고 요청 키는 요청-URI의 나머지 부분이 됩니다. 이는 일반적인 메서드로 이 단원의 첫 번째 예제와 두 번째 예제에 나와 있습니다. 호스트 헤더를 생략하는 것은 HTTP 1.0 요청에만 유효합니다.
- 또는 `Host` 헤더 값이 `.s3.Region.amazonaws.com`으로 끝나는 경우 버킷 이름이 `Host` 헤더 값에서 `.s3.Region.amazonaws.com`까지의 선행 구성 요소가 됩니다. 요청에 대한 키는 요청-URI입니다. 이 해석은 이 단원의 세 번째와 네 번째 예제에 나와 있는 것처럼 버킷을 `.s3.Region.amazonaws.com`의 하위 도메인으로 표시합니다.
- 그렇지 않은 경우 요청에 대한 버킷은 `Host` 헤더의 소문자 값이며 요청에 대한 키는 요청-URI입니다. 이 해석은 버킷 이름과 같은 DNS 이름을 등록하고 이 이름을 Amazon S3용 CNAME 별칭으로 구성한 경우에 유용합니다. 도메인 이름 등록 및 DNS 구성 절차는 본 가이드에서 다루지 않지만, 이 단원의 마지막 예제에 나와 있습니다.

예제

이 단원은 예제 URL 및 요청을 제공합니다.

Example 경로 방식

이 예에서는 다음을 사용합니다.

- 버킷 이름 - `awsexamplebucket.net`
- 리전 - 미국 동부(버지니아 북부)
- 키 이름 - `homepage.html`

URL은 다음과 같습니다.

```
https://s3.us-east-1.amazonaws.com/awsexamplebucket.net/homepage.html
```

요청은 다음과 같습니다.

```
GET /awsexamplebucket.net/homepage.html HTTP/1.1
Host: s3.us-east-1.amazonaws.com
```

HTTP 1.0을 사용한 요청 및 `host` 헤더 생략은 다음과 같습니다.

```
GET /awsexamplebucket.net/homepage.html HTTP/1.0
```

DNS를 준수하는 이름에 대한 자세한 내용은 제한 사항 (p. 47)을 참조하십시오. 키에 대한 자세한 내용은 키 (p. 3)를 참조하십시오.

Example 가상 호스팅 방식

이 예에서는 다음을 사용합니다.

- 버킷 이름 - `awsexamplebucket.eu`
- 리전 - 유럽(아일랜드)
- 키 이름 - `homepage.html`

URL은 다음과 같습니다.

```
https://awsexamplebucket.eu.s3.eu-west-1.amazonaws.com/homepage.html
```

요청은 다음과 같습니다.

```
GET /homepage.html HTTP/1.1
Host: awsexamplebucket.eu.s3.eu-west-1.amazonaws.com
```

Example CNAME 메서드

이 메서드를 사용하려면 DNS 이름을 *bucketname*.s3.us-east-1.amazonaws.com용 CNAME 별칭으로 구성해야 합니다. 자세한 내용은 Amazon S3 URL을 CNAME으로 사용자 지정 (p. 46) 단원을 참조하십시오. 이 예에서는 다음을 사용합니다.

- 버킷 이름 - awsexamplebucket.net
- 키 이름 - homepage.html

URL은 다음과 같습니다.

```
https://www.awsexamplebucket.net/homepage.html
```

예제는 다음과 같습니다.

```
GET /homepage.html HTTP/1.1
Host: www.awsexamplebucket.net
```

Amazon S3 URL을 CNAME으로 사용자 지정

필요에 따라 웹 사이트 또는 서비스에 s3.*Region*.amazonaws.com이 표시되지 않게 할 수도 있습니다. 예를 들어 Amazon S3에 웹 사이트 이미지를 호스팅할 경우 https://awsexamplebucket-images.s3.us-east-1.amazonaws.com/ 대신 https://images.awsexamplebucket.net/을 선호할 수 있습니다. DNS를 준수하는 이름의 버킷은 https://*BucketName*.s3.*Region*.amazonaws.com/[*Filename*]과 같이 참조될 수 있습니다 (예: https://images.awsexamplebucket.net.s3.us-east-1.amazonaws.com/mydog.jpg). CNAME을 사용하여 images.awsexamplebucket.net을 Amazon S3 호스트 이름으로 매핑함으로써 이전 URL이 https://images.awsexamplebucket.net/mydog.jpg가 될 수 있습니다.

버킷 이름은 CNAME과 동일해야 합니다. 예를 들어 images.awsexamplebucket.net을 images.awsexamplebucket.net.s3.amazonaws.com으로 매핑하기 위한 CNAME을 만드는 경우 https://images.awsexamplebucket.net/filename과 https://images.awsexamplebucket.net.s3.us-east-1.amazonaws.com/filename은 모두 동일합니다.

CNAME DNS 레코드는 도메인 이름으로 적절한 가상 호스팅–방식 호스트 이름의 별칭을 사용해야 합니다. 예를 들어 버킷 이름과 도메인 이름이 images.awsexamplebucket.net이고 버킷이 미국 동부(버지니아 북부) 리전에 있는 경우, CNAME 레코드의 별칭은 images.awsexamplebucket.net.s3.us-east-1.amazonaws.com으로 지정되어야 합니다.

```
images.awsexamplebucket.net CNAME    images.awsexamplebucket.net.s3.us-
east-1.amazonaws.com.
```

Amazon S3는 호스트 이름을 사용하여 버킷 이름을 확인합니다. 따라서 CNAME과 버킷 이름은 동일해야 합니다. 예를 들어, www.example.com을 www.example.com.s3.us-east-1.amazonaws.com. https://www.example.com을 액세스하는 경우, Amazon S3는 다음과 같은 요청을 받습니다.

Example

```
GET / HTTP/1.1
Host: www.example.com
Date: date
```

```
Authorization: signatureValue
```

Amazon S3는 원래 호스트 이름 `www.example.com`만 보고 요청을 해결하는 데 사용한 CNAME 매핑을 알지 못합니다.

모든 Amazon S3 엔드포인트는 CNAME에서 사용할 수 있습니다. 예를 들어, `s3.ap-southeast-1.amazonaws.com`은 CNAME에서 사용될 수 있습니다. 엔드포인트에 대한 자세한 내용은 요청 엔드포인트 (p. 11)를 참조하십시오.

CNAME을 사용하여 호스트 이름과 Amazon S3 버킷을 연결하려면

1. 제어하는 도메인에 속한 호스트 이름을 선택합니다.

   이 예제는 `images` 도메인의 `awsexamplebucket.net` 하위 도메인을 사용합니다.
2. 호스트 이름과 일치하는 버킷을 만듭니다.

   이 예제에서 호스트 및 버킷 이름은 `images.awsexamplebucket.net`입니다. 버킷 이름은 호스트 이름과 정확하게 일치해야 합니다.
3. Amazon S3 버킷에 대한 별칭으로 호스트 이름을 정의하는 CNAME 레코드를 만듭니다.

   예:

   ```
   images.awsexamplebucket.net CNAME images.awsexamplebucket.net.s3.us-west-2.amazonaws.com
   ```

   > **Important**
   >
   > 요청 라우팅으로 인해, CNAME 레코드는 앞의 예제에 나와 있는 것처럼 정확하게 정의해야 합니다. 그렇지 않은 경우 제대로 작동하는 것처럼 보일 수 있으나 결국 예기치 않은 동작이 발생합니다.

   DNS를 구성하는 절차는 해당 DNS 서버나 DNS 공급자에 따라 다릅니다. 자세한 내용은 서버 설명서를 참조하거나, 제공자에게 문의하십시오.

제한 사항

SSL

가상 호스팅 URL은 비 SSL(HTTP) 요청에서만 지원됩니다. SSL과 함께 가상 호스팅 방식의 버킷을 사용할 경우, SSL 와일드카드 인증서는 마침표를 포함하지 않는 버킷에만 매핑됩니다. 이 문제를 해결하려면 HTTP를 사용하거나, 인증서 확인 로직을 직접 작성합니다.

SOAP

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

이전 버전과의 호환성

레거시 엔드포인트

일부 리전에서 레거시 엔드포인트를 지원합니다. 서버 액세스 로그 또는 CloudTrail 로그에 이러한 엔드포인트가 표시될 수 있습니다. 자세한 내용은 아래 정보를 검토하십시오. Amazon S3 리전 및 엔드포인트의 전체 목록은 AWS 일반 참조의 Amazon S3 리전 및 엔드포인트를 참조하십시오.

> **Important**
>
> 로그에 레거시 엔드포인트가 표시될 수도 있지만 항상 표준 엔드포인트 구문을 사용하여 버킷에 액세스하는 것이 좋습니다.

Amazon S3 가상 호스팅 스타일 URL은 다음 형식을 따릅니다.

```
https://bucket-name.s3.Region.amazonaws.com/key name
```

Amazon S3에서 경로 스타일 URL은 다음 형식을 따릅니다.

```
https://s3.Region.amazonaws.com/bucket-name/key name
```

s3 리전

일부 이전 Amazon S3 리전에서는 S3와 리전 사이에 점이 있는 엔드포인트(예: `S3#us-west-2`) 대신 대시가 있는 엔드포인트(예: `S3.us-west-2`)를 지원합니다. 버킷이 이러한 리전 중 하나에 있는 경우, 서버 액세스 로그 또는 CloudTrail 로그에 다음과 같은 엔드포인트 형식이 표시될 수 있습니다.

```
https://bucket-name.s3-Region.amazonaws.com
```

이 예제에서 버킷 이름은 `my-bucket`이고 리전은 미국 서부(오레곤)입니다.

```
https://my-bucket.s3-us-west-2.amazonaws.com
```

레거시 전역 엔드포인트

일부 리전에서는 레거시 전역 엔드포인트를 사용하여 리전별 엔드포인트를 지정하지 않는 요청을 구성할 수 있습니다. 레거시 전역 엔드포인트 지점은 다음과 같습니다.

```
bucket-name.s3.amazonaws.com
```

레거시 전역 엔드포인트를 사용하는 요청이 서버 액세스 로그 또는 CloudTrail 로그에 표시될 수 있습니다. 이 예제에서 버킷 이름은 `my-bucket`이며 레거시 전역 엔드포인트가 표시됩니다.

```
https://my-bucket.amazonaws.com
```

미국 동부(버지니아 북부)에 대한 가상 호스팅 방식 요청

레거시 전역 엔드포인트를 사용한 요청은 기본적으로 미국 동부(버지니아 북부)로 이동합니다. 따라서 레거시 전역 엔드포인트는 미국 동부(버지니아 북부)에 대한 리전 엔드포인트 대신 사용되기도 합니다. 미국 동부(버지니아 북부)에 버킷을 만들고 전역 엔드포인트를 사용하는 경우 Amazon S3는 기본적으로 이 리전으로 요청을 라우팅합니다.

다른 리전에 대한 가상 호스팅 방식 요청

또한 레거시 전역 엔드포인트는 지원되는 다른 리전의 가상 호스팅 방식 요청에도 사용됩니다. 2019년 3월 20일 이전에 시작된 리전에 버킷을 만들고 레거시 전역 엔드포인트를 사용하면 Amazon S3는 DNS를 업데이트하여 요청을 올바른 위치로 다시 라우팅하며, 이로 인해 시간이 걸릴 수 있습니다. 동시에, 기본 규칙이 적용되어 가상 호스팅 방식 요청이 미국 동부(버지니아 북부) 리전으로 라우팅되면 Amazon S3는 HTTP 307을 통해 이를 올바른 리전으로 리디렉션합니다. 2019년 3월 20일 이후에 시작된 리전의 S3 버킷인 경우 DNS는 버킷이 있는 AWS 리전으로 요청을 직접 라우팅하지 않습니다. 대신, HTTP 400 Bad Request 오류를 반환합니다. 자세한 내용은 요청 리디렉션 및 REST API (p. 658) 단원을 참조하십시오.

경로 방식 요청

미국 동부(버지니아 북부) 리전에서는, 레거시 전역 엔드포인트를 경로 방식 요청에 사용할 수 있습니다.

다른 모든 리전에서 경로 방식 구문을 사용할 경우 버킷에 액세스할 때 리전별 엔드포인트를 사용해야 합니다. 버킷이 상주하는 리전의 엔드포인트와 다른 엔드포인트 또는 레거시 전역 엔드포인트가 있는 버킷에 액세스하려고 하면 HTTP 응답 코드 307 임시 리디렉션 오류와 리소스에 대한 올바른 URI

를 나타내는 메시지가 표시됩니다. 예를 들어 미국 서부(오레곤) 리전에 생성된 버킷에 `https://s3.amazonaws.com/bucket-name`을 사용하는 경우 HTTP 307 임시 리디렉션 오류가 표시됩니다.

요청 리디렉션 및 REST API

주제

- 리디렉션 및 HTTP 사용자 에이전트 (p. 49)
- 리디렉션 및 100-Continue (p. 49)
- 리디렉션 예제 (p. 49)

이 단원에서는 Amazon S3 REST API를 사용하여 HTTP 리디렉션을 처리하는 방법에 대해 설명합니다. Amazon S3 리디렉션에 대한 일반적인 정보는 Amazon Simple Storage Service API Reference의 요청 리디렉션 및 REST API (p. 658)를 참조하십시오.

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 메서드가 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. 애플리케이션에서 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 8.2.3을 참조하십시오.

> **Note**
>
> RFC 2616에 따르면 알 수 없는 HTTP 서버에 `Expect: Continue`를 사용할 때 요청 본문을 보내기 전에 무한정으로 기다리지 않아야 합니다. 이는 일부 HTTP 서버에서 100-Continue가 인식되지 않기 때문입니다. 그러나 Amazon S3에서는 요청에 `Expect: Continue`가 포함되어 있는 경우 이를 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
```

클라이언트에서 요청 본문 전송을 진행해야 함을 나타내는 100-Continue를 Amazon S3에서 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
```

Amazon S3 버킷을 사용한 작업

Amazon S3에 데이터(사진, 동영상, 문서 등)를 업로드하려면 우선 하나의 AWS 리전에 S3 버킷을 만들어야 합니다. 그런 다음 이 버킷에 객체를 얼마든지 업로드할 수 있습니다.

구현 측면에서 버킷과 객체는 리소스이고, Amazon S3는 이를 관리하기 위한 API를 제공합니다. 예를 들어 Amazon S3 API를 사용하여 버킷을 만들고 객체를 업로드할 수 있습니다. 이러한 작업은 Amazon S3 콘솔을 사용하여 수행할 수도 있습니다. 이 콘솔은 Amazon S3 API를 사용하여 Amazon S3에 요청을 보냅니다.

이 단원에서는 버킷을 사용을 사용하는 방법에 대해 설명합니다. 객체 작업에 대한 자세한 내용은 Amazon S3 객체로 작업 (p. 102)을 참조하십시오.

Amazon S3 버킷 이름은 전역 수준에서 고유하며, 네임스페이스는 모든 AWS 계정이 공유합니다. 따라서 버킷이 생성된 후에는 해당 버킷이 삭제될 때까지 어떤 AWS 리전에서도 다른 AWS 계정이 해당 버킷 이름을 사용할 수 없습니다. 가용성 또는 보안 확인 목적을 위해 특정 버킷 명명 규칙에 의존하면 안 됩니다. 버킷 이름 지정 지침은 버킷 규제 및 제한 (p. 56)을 참조하십시오.

Amazon S3은 사용자가 지정한 리전에 버킷을 만듭니다. 지연 시간을 최적화하고, 비용을 최소화하며, 규제 요건을 해결하려면 지리적으로 가까운 AWS 리전을 선택합니다. 예를 들어 유럽에 거주할 경우 유럽(아일랜드) 또는 유럽(프랑크푸르트) 리전에서 버킷을 생성하는 것이 유리할 수 있습니다. Amazon S3 리전 목록은 AWS 일반 참조의 리전 및 엔드포인트를 참조하십시오.

> **Note**
>
> 특정 AWS 리전에서 만든 버킷에 속하는 객체는 명시적으로 다른 리전으로 보내지 않는 한 해당 리전을 벗어나지 않습니다. 예를 들어 유럽(아일랜드) 리전에 저장된 객체는 해당 리전을 벗어나지 않습니다.

주제

- 버킷 생성 (p. 51)
- 버킷에 대한 퍼블릭 액세스 관리 (p. 53)
- 버킷 액세스 (p. 53)
- 버킷 구성 옵션 (p. 55)
- 버킷 규제 및 제한 (p. 56)
- 버킷 생성 예제 (p. 58)
- 버킷 삭제 또는 비우기 (p. 61)
- S3 버킷에 대한 Amazon S3 기본 암호화 (p. 65)
- Amazon S3 Transfer Acceleration (p. 67)
- 요청자 지불 버킷 (p. 74)
- 버킷 및 액세스 제어 (p. 77)
- S3 버킷에 대한 결제 및 보고 (p. 77)

버킷 생성

Amazon S3는 버킷을 만들고 관리할 수 있는 API를 제공합니다. 기본적으로 AWS 계정 각각에 대해 최대 100개의 버킷을 만들 수 있습니다. 버킷이 더 필요한 경우 서비스 한도 증가를 제출하여 계정버킷 한도 를 최대 1,000개의 버킷 으로 늘릴 수 있습니다. 버킷 한도 상승을 제출하는 방법은 AWS 일반 참조의 AWS 서비스 제한을 참조하십시오. 버킷에 저장할 수 있는 객체 수에는 제한이 없습니다.

버킷을 만들려면 버킷의 이름과 버킷을 만들려는 AWS 리전을 제공해야 합니다. 버킷 이름 지정에 대한 자세한 내용은 버킷 이름 지정 규칙 (p. 57)을 참조하십시오.

아래 나열된 방법 중 하나를 사용하여 버킷을 만들 수 있습니다. 예제는 버킷 생성 예제 (p. 58)을 참조하십시오.

Amazon S3 콘솔

Amazon S3 콘솔을 사용하여 버킷을 만들 수 있습니다. 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 버킷 생성을 참조하십시오.

REST API

REST API를 사용하여 버킷을 생성하는 방법은 요청 인증을 위한 코드를 작성해야 하므로 번거로울 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API Reference의 PUT 버킷을 참조하십시오. 대신 AWS Management 콘솔 또는 AWS SDK를 사용하는 것이 좋습니다.

AWS SDK

AWS SDK를 사용하여 버킷을 만드는 경우, 먼저 클라이언트를 만든 다음 이 클라이언트를 사용하여 버킷 생성 요청을 보냅니다. 클라이언트 또는 버킷을 만들 때 리전을 지정하지 않으면 Amazon S3는 기본 리전인 미국 동부(버지니아 북부)을 사용합니다. 또한 특정 리전을 지정할 수도 있습니다. 사용 가능한 AWS 리전 목록은 AWS General Reference의 리전 및 엔드포인트를 참조하십시오. AWS 리전 활성화 또는 비활성화에 대한 자세한 내용은 AWS General Reference의 AWS 리전 관리를 참조하십시오.

레거시 전역 엔드포인트

모범 사례로서, 동일한 리전에 클라이언트 및 버킷을 만들어야 합니다. 하지만 기본적으로 또는 의도적으로 미국 동부(버지니아 북부) 리전에 클라이언트를 만들면 Amazon S3는 레거시 전역 엔드포인트를 사용하여 클라이언트와 통신할 수 있습니다.

```
s3.amazonaws.com
```

따라서 미국 동부(버지니아 북부)의 클라이언트를 사용하여 2019년 3월 20일 이전에 시작된 모든 리전에 버킷을 만들 수 있습니다. 하지만 2019년 3월 20일 이후에 시작된 리전에 버킷을 만드는 경우 클라이언트 및 버킷이 동일한 리전에 있어야 합니다. 자세한 내용은 레거시 엔드포인트 (p. 47) 단원을 참조하십시오.

클라이언트 만들기

클라이언트를 만들 때 클라이언트를 생성할 AWS 리전을 지정할 수 있습니다. 리전을 지정하지 않으면 Amazon S3는 기본 리전인 미국 동부(버지니아 북부)에 버킷을 만듭니다. 클라이언트를 만들어 듀얼 스택 엔드포인트에 액세스하려면 AWS 리전을 지정해야 합니다. 자세한 내용은 듀얼 스택 엔드포인트 (p. 14) 단원을 참조하십시오.

클라이언트를 만들면 리전은 리전별 엔드포인트로 매핑됩니다. 클라이언트는 이 엔드포인트를 사용하여 Amazon S3와 통신합니다.

```
s3.<region>.amazonaws.com
```

예를 들어, eu-west-1 리전을 지정하여 클라이언트를 만들면 다음과 같이 리전별 엔드포인트로 매핑됩니다.

```
s3.eu-west-1.amazonaws.com
```

버킷 생성

버킷을 만들 때 리전을 지정하지 않으면 Amazon S3는 미국 동부(버지니아 북부) 리전에 버킷을 만듭니다. 따라서 특정 리전에 버킷을 만들어야 한다면 버킷을 만들 때 리전을 지정해야 합니다.

2020년 9월 30일 이후에 생성된 버킷은 가상 호스팅 방식의 요청만 지원합니다. 이 날짜 당일 또는 이전에 생성된 버킷에 대해서는 경로 방식의 요청이 계속 지원됩니다. 자세한 내용은 Amazon S3 Path Deprecation Plan - The Rest of the Story를 참조하십시오.

권한 정보

AWS 계정 루트 자격 증명을 사용하여 버킷을 만들고 기타 Amazon S3 작업을 수행할 수 있습니다. 하지만 AWS는 AWS 계정의 루트 자격 증명을 사용하여 버킷 생성과 같은 요청을 하지 않을 것을 권장합니다. 대신 IAM 사용자를 만들고 이 사용자에게 모든 액세스 권한을 부여합니다(기본적으로 사용자는 권한이 없음). 이러한 사용자를 관리자 사용자라고 합니다. 자체 계정의 루트 자격 증명 대신 관리자 사용자 자격 증명을 사용하여 AWS와 연동하면서 버킷 생성, 사용자 생성, 권한 부여 등의 작업을 수행할 수 있습니다.

자세한 내용은 루트 계정 자격 증명 대 IAM 사용자 자격 증명(AWS 일반 참조)과 IAM 모범 사례(IAM 사용 설명서)를 참조하십시오.

리소스를 만든 AWS 계정은 해당 리소스의 소유자가 됩니다. 예를 들어, 자신의 AWS 계정에 IAM 사용자를 만들고 버킷을 만들 수 있는 사용자 권한을 부여하면, 이 사용자가 버킷을 만들 수 있습니다. 하지만 사용자는 버킷을 소유하지 않으며, 이 사용자가 속한 AWS 계정에서 버킷을 소유합니다. 사용자가 다른 버킷 작업을 수행하려면 리소스 소유자로부터 추가 권한을 받아야 합니다. Amazon S3 리소스 권한 관리에 대한 자세한 내용은 Amazon S3의 Identity and Access Management (p. 299) 단원을 참조하십시오.

버킷에 대한 퍼블릭 액세스 관리

ACL(액세스 제어 목록), 버킷 정책 또는 둘 다를 통해 버킷 및 객체에 퍼블릭 액세스 권한이 부여됩니다. Amazon S3 리소스에 대한 퍼블릭 액세스를 관리할 수 있도록 Amazon S3는 퍼블릭 액세스 차단 설정을 제공합니다. Amazon S3 퍼블릭 액세스 차단 설정은 ACL 및 버킷 정책을 재정의하여 이러한 리소스에 대한 퍼블릭 액세스에 균일한 제한을 적용할 수 있도록 합니다. 개별 버킷 또는 계정의 모든 버킷에 퍼블릭 액세스 차단 설정을 적용할 수 있습니다.

모든 Amazon S3 버킷 및 객체의 퍼블릭 액세스가 차단되도록 하려면 계정에 대해 퍼블릭 액세스 차단 설정 4개를 모두 켜는 것이 좋습니다. 이러한 설정은 모든 현재 및 미래 버킷에 대한 퍼블릭 액세스를 차단합니다.

이 설정을 적용하기 전에 퍼블릭 액세스 없이 애플리케이션이 올바르게 작동하는지 확인합니다. 버킷 또는 객체에 대한 특정 수준의 퍼블릭 액세스가 필요한 경우(예를 들어 Amazon S3 정적 웹 사이트 호스팅 (p. 548)에 설명된 대로 정적 웹 사이트를 호스팅하는 경우), 스토리지 사용 사례에 적합하도록 개별 설정을 사용자 지정할 수 있습니다. 자세한 내용은 Amazon S3 퍼블릭 액세스 차단 사용 (p. 450) 단원을 참조하십시오.

버킷 액세스

Amazon S3 콘솔을 사용하여 버킷에 액세스할 수 있습니다. 콘솔 UI를 사용하여 코드를 작성하지 않고 거의 모든 버킷 작업을 수행할 수 있습니다.

프로그래밍 방식으로 버킷에 액세스할 경우 Amazon S3는 RESTful 아키텍처를 지원하며, 이 아키텍처에서 버킷과 객체는 리소스이고 각각 고유한 리소스 URI를 갖습니다.

Amazon S3는 버킷 액세스에 대해 가상 호스팅 방식과 경로 방식 URL을 모두 지원합니다. 버킷은 경로 방식과 가상 호스팅 방식의 URL을 사용하여 액세스할 수 있기 때문에 DNS를 준수하는 버킷 이름으로 버킷을 만드는 것이 좋습니다. 자세한 내용은 버킷 규제 및 제한 (p. 56) 단원을 참조하십시오.

> **Note**
>
> 가상 호스팅 스타일 및 경로 스타일 요청은 S3 점(.) 리전 엔드포인트 구조(`s3.Region`)를 사용합니다(예: `https://my-bucket.s3.us-west-2.amazonaws.com`). 그러나 일부 이전 Amazon S3 리전은 S3 대시(-) 리전 엔드포인트 `s3-Region`도 지원합니다(예: `https://my-bucket.s3-us-west-2.amazonaws.com`). 버킷이 이러한 리전 중 하나에 있는 경우 서버 액세스 로그 또는 CloudTrail 로그에 `s3-Region` 엔드포인트가 표시될 수 있습니다. 요청에 이 엔드포인트 구조를 사용하지 않는 것이 좋습니다.

가상 호스팅 방식 액세스

가상 호스팅 방식의 요청에서 버킷 이름은 URL에서 도메인 이름의 일부입니다.

Amazon S3 가상 호스팅 스타일 URL은 다음 형식을 따릅니다.

```
https://bucket-name.s3.Region.amazonaws.com/key name
```

이 예제에서 `my-bucket`은 버킷 이름이고, 미국 서부(오레곤)는 리전이며, `puppy.png`는 키 이름입니다.

```
https://my-bucket.s3.us-west-2.amazonaws.com/puppy.png
```

가상 호스팅 방식 액세스에 대한 자세한 내용은 가상 호스팅 방식 요청 (p. 44) 단원을 참조하십시오.

경로 방식 액세스

Amazon S3에서 경로 스타일 URL은 다음 형식을 따릅니다.

```
https://s3.Region.amazonaws.com/bucket-name/key name
```

예를 들어 미국 서부(오레곤) 리전에서 이름이 `mybucket`인 버킷을 생성하고 해당 버킷의 `puppy.jpg` 객체에 액세스하려는 경우 다음 경로 스타일 URL을 사용할 수 있습니다.

```
https://s3.us-west-2.amazonaws.com/mybucket/puppy.jpg
```

자세한 내용은 경로 방식 요청 (p. 44) 단원을 참조하십시오.

> **Important**
>
> 2020년 9월 30일 이후에 생성된 버킷은 가상 호스팅 방식의 요청만 지원합니다. 이 날짜 당일 또는 이전에 생성된 버킷에 대해서는 경로 방식의 요청이 계속 지원됩니다. 자세한 내용은 Amazon S3 Path Deprecation Plan - The Rest of the Story를 참조하십시오.

IPv6을 통해 S3 버킷에 액세스

Amazon S3에는 인터넷 프로토콜 버전 6(IPv6) 및 IPv4 모두를 통해 S3 버킷에 요청하는 것을 지원하는 여러 개의 듀얼 스택 엔드포인트가 있습니다. 자세한 내용은 IPv6을 통해 요청 (p. 11) 단원을 참조하십시오.

S3 액세스 포인트를 통해 버킷 액세스

버킷에 직접 액세스하는 것 외에도 S3 액세스 포인트를 통해 버킷에 액세스할 수 있습니다. S3 액세스 포인트에 대한 자세한 내용은 Amazon S3 액세스 포인트를 사용한 데이터 액세스 관리 (p. 90) 단원을 참조하십시오.

S3 액세스 포인트는 가상 호스트 스타일의 주소 지정만 지원합니다. 액세스 포인트를 통해 버킷의 주소를 지정하려면 다음 형식을 사용합니다.

```
https://AccessPointName-AccountId.s3-accesspoint.region.amazonaws.com.
```

> **Note**
>
> - 액세스 포인트 이름에 대시(-) 문자가 포함된 경우 URL에 대시를 포함시키고 계정 ID 앞에 다른 대시를 삽입합니다. 예를 들어 `us-west-2` 리전의 `123456789012` 계정에서 소유한 `finance-docs`라는 액세스 포인트를 사용하려면 해당 URL은 `https://finance-docs-123456789012.s3-accesspoint.us-west-2.amazonaws.com`입니다.

- S3 액세스 포인트는 HTTP에 의한 액세스를 지원하지 않으며 HTTPS에 의한 보안 액세스만 지원합니다.

S3://를 사용하여 버킷 액세스

일부 AWS 서비스에서는 `S3://bucket`을 사용하여 Amazon S3 버킷을 지정해야 합니다. 올바른 형식이 아래에 나와 있습니다. 이 형식을 사용할 경우 버킷 이름에 리전이 포함되지 않습니다.

```
S3://bucket-name/key-name
```

예를 들어 앞의 경로 방식 단원에서 설명한 샘플 버킷을 사용하는 경우 다음과 같습니다.

```
S3://mybucket/puppy.jpg
```

버킷 구성 옵션

Amazon S3는 버킷을 구성하기 위한 다양한 옵션을 지원합니다. 예를 들어 웹 사이트 호스팅용으로 버킷을 구성하고, 버킷의 객체 수명 주기를 관리하는 구성을 추가하며, 버킷에 대한 모든 액세스를 로그로 기록하도록 버킷을 구성할 수 있습니다. Amazon S3는 버킷 구성 정보를 저장 및 관리할 수 있는 하위 리소스를 지원합니다. Amazon S3 API를 사용하여 이러한 하위 리소스를 만들고 관리할 수 있습니다. 하지만 콘솔이나 AWS SDK를 사용할 수도 있습니다.

> **Note**
>
> 또한 객체 수준 구성도 제공됩니다. 예를 들어, 객체에 고유한 ACL(액세스 통제 목록)을 구성하여 객체 수준 권한을 구성할 수 있습니다.

이들 리소스는 특정 버킷이나 객체의 컨텍스트에 존재하므로 하위 리소스라고 합니다. 다음 표에는 버킷별 구성을 관리할 수 있는 하위 리소스가 나와 있습니다.

하위 리소스	설명
cors(cross-origin 리소스 공유)	cross-origin 요청을 허용하도록 버킷을 구성할 수 있습니다. 자세한 내용은 Cross-Origin 리소스 공유 활성화를 참조하십시오.
이벤트 알림	지정한 버킷 이벤트의 알림을 받도록 버킷을 설정할 수 있습니다. 자세한 내용은 Amazon S3 이벤트 알림 구성 (p. 587) 단원을 참조하십시오.
수명 주기	버킷의 객체에 대해 명확한 수명 주기를 정의한 수명 주기 규칙을 정의할 수 있습니다. 예를 들어, 생성 후 1년이 지난 객체를 보관하거나, 10년이 지난 객체를 삭제하는 규칙을 정의할 수 있습니다. 자세한 내용은 객체 수명 주기 관리를 참조하십시오.
location	버킷을 만들려면 Amazon S3가 버킷을 생성할 AWS 리전을 지정해야 합니다. Amazon S3는 이 정보를 location 하위 리소스에 저장하고 이 정보를 검색하기 위한 API를 제공합니다.
logging	로깅을 사용하여 버킷에 대한 액세스 요청을 추적할 수 있습니다. 각 액세스 로그 레코드는 한 액세스 요청에 대한 세부 정보(요청자, 버킷 이름, 요청 시간, 요청 작업, 응답 상태, 오류 코드 등)를 제공합니다. 액세스 로그 정보는 보안 및 액세스 감사에 유용할 수 있습니다. 또한 고객층을 파악하고 Amazon S3 결제 요금을 확인할 수 있습니다.

하위 리소스	설명
	자세한 내용은 Amazon S3 서버 액세스 로깅 (p. 705) 단원을 참조하십시오.
객체 잠금	Amazon S3 객체 잠금을 사용하려면 버킷에 대해 활성화해야 합니다. 버킷에 있는 새 객체에 적용할 기본 보관 모드와 기간을 선택적으로 구성할 수도 있습니다. 자세한 내용은 버킷 구성 (p. 492) 단원을 참조하십시오.
정책 및 ACL(액세스 통제 목록)	모든 리소스(버킷과 객체)는 기본적으로 비공개입니다. Amazon S3는 버킷 수준 권한을 부여하고 관리할 수 있도록 버킷 정책과 ACL(액세스 제어 목록) 옵션을 모두 지원합니다. Amazon S3는 권한 정보를 policy 및 acl 하위 리소스에 저장합니다. 자세한 내용은 Amazon S3의 Identity and Access Management (p. 299) 단원을 참조하십시오.
복제	복제는 서로 다른 AWS 리전 또는 동일한 AWS 리전의 버킷 간에 객체를 비동기식으로 자동 복사하는 것을 말합니다. 자세한 내용은 복제 (p. 607) 단원을 참조하십시오.
requestPayment	기본적으로 버킷에서의 다운로드에 대한 요금은 해당 버킷을 만든 AWS 계정(버킷 소유자)이 지불합니다. 버킷 소유자는 이 하위 리소스를 사용하여, 다운로드를 요청하는 사용자에게 다운로드 요금이 부과되도록 지정할 수 있습니다. Amazon S3는 이 하위 리소스를 관리하기 위한 API를 제공합니다. 자세한 내용은 요청자 지불 버킷 (p. 74) 단원을 참조하십시오.
tagging	비용 할당 태그를 버킷에 추가하여 AWS 비용을 분류하고 추적할 수 있습니다. 버킷의 태그를 저장 및 관리하기 위해 Amazon S3는 tagging 하위 리소스를 제공합니다. AWS는 버킷에 적용된 태그를 사용하여 사용 내역 및 비용을 태그별로 집계한 비용 할당 보고서를 만듭니다. 자세한 내용은 S3 버킷에 대한 결제 및 보고 (p. 77) 단원을 참조하십시오.
전송 속도 향상	Transfer Acceleration은 거리가 먼 클라이언트와 S3 버킷 간에 파일을 빠르고, 쉽고, 안전하게 전송할 수 있게 해줍니다. Transfer Acceleration은 전 세계적으로 분산되어 있는 Amazon CloudFront의 엣지 로케이션을 활용합니다. 자세한 내용은 Amazon S3 Transfer Acceleration (p. 67) 단원을 참조하십시오.
버전 관리	버전 관리는 실수로 덮어쓰거나 삭제하는 경우 복구할 수 있도록 해주는 기능입니다. 실수로 객체를 잘못 삭제하거나 덮어쓰는 경우 이를 복구할 수 있도록 하기 위해 버전 관리 기능을 사용하는 것이 좋습니다. 자세한 내용은 버전 관리 사용 (p. 468) 단원을 참조하십시오.
웹 사이트	버킷을 정적 웹 사이트 호스팅용으로 구성할 수 있습니다. Amazon S3는 website 하위 리소스를 만들어 이 구성을 저장합니다. 자세한 내용은 Amazon S3에서 정적 웹 사이트 호스팅을 참조하십시오.

버킷 규제 및 제한

버킷은 해당 버킷을 생성한 AWS 계정의 소유입니다. 버킷 소유권은 양도할 수 없습니다. 버킷을 만들 때 버킷을 생성할 리전을 선택합니다. 버킷을 만든 후에는 리전을 변경할 수 없습니다.

기본적으로 AWS 계정 각각에 대해 최대 100개의 버킷을 만들 수 있습니다. 추가 버킷이 필요할 경우 서비스 한도 증가를 제출하여 계정 버킷 한도를 최대 1,000 버킷으로 늘릴 수 있습니다. 많은 버킷을 사용하든 혹은 몇 개만 사용하든 성능에는 차이가 없습니다. 버킷 한도를 늘리는 방법은 AWS 일반 참조의 AWS Service Limits를 참조하십시오.

버킷 이름 재사용

버킷을 비웠으면 버킷을 삭제할 수 있습니다. 버킷을 삭제한 후 버킷 이름은 다시 사용할 수 있게 됩니다. 하지만 버킷을 삭제한 후 여러 가지 이유로 버킷 이름을 다시 사용하지 못할 수도 있습니다. 예를 들어 버킷을 삭제하고 버킷 이름을 다시 사용할 수 있게 되면 다른 계정에서 해당 이름으로 된 버킷을 만들 수 있습니다. 또한 삭제된 버킷의 이름을 다시 사용할 수 있을 때까지 약간의 시간이 걸릴 수 있습니다. 동일한 버킷 이름을 사용하려 한다면 버킷을 삭제하지 않는 것이 좋습니다.

객체 및 버킷

버킷에 저장할 수 있는 객체 수에는 제한이 없습니다. 모든 객체를 하나의 버킷에 저장하거나, 여러 버킷에 저장할 수 있습니다. 하지만 다른 버킷 내에 버킷을 만들 수는 없습니다.

버킷 작업

Amazon S3의 고가용성 설계는 get, put, list, delete 작업에 중점을 두고 있습니다. 버킷 작업은 중앙의 전역 리소스 공간에 영향을 주기 때문에 애플리케이션의 고가용성 코드 경로에 버킷을 만들거나 삭제하는 것은 좋지 않습니다. 자주 실행하지 않는 별도의 초기화 루틴이나 설정 루틴에서 버킷을 만들거나 삭제하는 것이 좋습니다.

버킷 이름 지정 및 자동으로 생성된 버킷

애플리케이션에서 자동으로 버킷을 생성할 경우, 이름 충돌 가능성이 낮은 버킷 이름 지정 체계를 선택합니다. 버킷 이름이 이미 사용 중이면 애플리케이션 로직에서 다른 버킷 이름을 선택합니다.

버킷 이름 지정 규칙

S3 버킷을 만든 후에는 버킷 이름을 변경할 수 없으므로 신중하게 이름을 선택하십시오.

> **Important**
>
> 2018년 3월 1일에 미국 동부(버지니아 북부) 리전에 있는 S3 버킷의 이름 지정 규칙이 전 세계 다른 모든 AWS 리전에서 사용하고 있는 이름 지정 규칙과 일치하도록 업데이트되었습니다. Amazon S3에서 대문자나 밑줄이 포함된 버킷 이름을 더 이상 생성할 수 없습니다. 이 변경 사항에 따라 각 버킷을 `https://myawsbucket.s3.us-east-1.amazonaws.com`과 같이 가상 호스트 스타일 주소 지정을 사용하여 주소 지정할 수 있게 됩니다. 사용자는 기존 버킷 생성 절차를 검토하여 이러한 DNS 형식의 이름 지정 규칙을 따르고 있는지 확인하는 것이 좋습니다.

규칙

다음은 모든 AWS 리전에서 S3 버킷의 이름을 지정하기 위한 규칙입니다.

- Amazon S3에 있는 모든 기존의 버킷 이름은 고유해야 합니다.
- 버킷 이름은 DNS 이름 지정 규칙을 준수해야 합니다.
- 버킷 이름은 3자 이상, 63자 이하여야 합니다.
- 버킷 이름에 대문자나 밑줄을 사용할 수 없습니다.
- 버킷 이름은 소문자나 숫자로 시작해야 합니다.
- 버킷 이름은 한 개 이상으로 구성된 일련의 레이블이어야 합니다. 각 레이블은 마침표(.)로 구분합니다. 버킷 이름은 소문자, 숫자, 하이픈(-)을 포함할 수 있습니다. 각 레이블은 소문자나 숫자로 시작하고 끝나야 합니다.

- 버킷 이름은 IP 주소 형식(예: 192.168.5.4)을 사용하지 않습니다.
- Secure Sockets Layer(SSL)와 함께 가상 호스팅 방식의 버킷을 사용할 경우, SSL 와일드카드 인증서는 마침표가 포함되지 않은 버킷에만 연결됩니다. 이 문제를 해결하려면 HTTP를 사용하거나, 인증서 확인 로직을 직접 작성합니다. 가상 호스팅 방식의 버킷을 사용할 경우, 버킷 이름에 마침표(".")를 사용하지 않는 것이 좋습니다.

DNS 형식이 아닌 레거시 버킷 이름

2018년 3월 1일을 시작으로, DNS 형식의 이름을 요구하도록 미국 동부(버지니아 북부) 리전에 있는 S3 버킷의 이름 지정 규칙이 업데이트되었습니다.

이전에 미국 동부(버지니아 북부) 리전은 버킷 이름 지정에 유연한 기준을 적용했으며 따라서 DNS 형식이 아닌 버킷 이름이 사용될 수 있었습니다. 예를 들어 `MyAWSbucket`은 대문자를 포함하고 있지만 유효한 버킷 이름이었습니다. 이 버킷에 가상 호스팅 방식의 요청(`https://MyAWSbucket.s3.us-east-1.amazonaws.com/yourobject`)을 사용하여 액세스할 경우, URL은 `MyAWSbucket` 버킷이 아닌 `myawsbucket` 버킷으로 변환됩니다. 이에 대한 응답으로 Amazon S3는 "bucket not found(버킷을 찾을 수 없음)"이라는 오류를 반환합니다. 버킷에 대한 가상 호스팅 방식의 액세스에 대한 자세한 내용은 버킷의 가상 호스팅 (p. 43) 단원을 참조하십시오.

미국 동부(버지니아 북부) 리전의 버킷 이름에 대한 레거시 규칙에서는 최대 255자의 버킷 이름을 허용했으며, 버킷 이름에 대문자, 소문자, 숫자, 마침표(.), 하이픈(-), 밑줄(_)을 조합하여 사용할 수 있었습니다.

Amazon S3 Transfer Acceleration에 사용되는 버킷의 이름은 DNS를 따라야 하며 마침표(".")를 포함할 수 없습니다. Transfer Acceleration에 대한 자세한 내용은 Amazon S3 Transfer Acceleration (p. 67) 단원을 참조하십시오.

버킷 생성 예제

주제

- Amazon S3 콘솔 사용 (p. 59)
- AWS SDK for Java 사용 (p. 59)
- .NET용 AWS SDK 사용 (p. 60)
- Ruby 버전 3용 AWS SDK 사용 (p. 61)
- 다른 AWS SDK 사용 (p. 61)

다음 코드 예제에서는 Java, .NET 및 Ruby용 AWS SDK를 사용하여 프로그래밍 방식으로 버킷을 생성합니다. 이 코드 예제는 다음 작업을 수행합니다.

- 버킷이 아직 없는 경우 버킷을 생성합니다. 이 예제에서는 다음 작업을 수행하여 버킷을 생성합니다.
  - AWS 리전(이 예제에서는 `s3.eu-west-1` 리전 사용)을 명시적으로 지정하여 클라이언트를 생성합니다. 이렇게 생성한 클라이언트는 `s3.eu-west-1.amazonaws.com` 엔드포인트를 사용하여 Amazon S3와 통신합니다. 다른 AWS 리전을 지정할 수도 있습니다. AWS 리전 목록은 AWS 일반 참조의 리전 및 엔드포인트를 참조하십시오.
  - 버킷 이름만 지정하여 버킷 생성 요청을 보냅니다. 이 버킷 생성 요청에서는 다른 AWS 리전을 지정하지 않습니다. 클라이언트가 클라이언트 생성 시 지정한 리전에서 버킷을 생성하는 요청을 Amazon S3로 보냅니다. 버킷을 만든 후에는 버킷 리전을 변경할 수 없습니다.

    Note

    버킷 생성 요청에 클라이언트 생성 시 지정한 리전과 다른 AWS 리전을 명시적으로 지정할 경우, 오류가 발생할 수 있습니다. 자세한 내용은 버킷 생성 (p. 51) 단원을 참조하십시오.

SDK 라이브러리는 버킷 PUT 요청을 Amazon S3에 보내 버킷을 생성합니다. 자세한 내용은 PUT Bucket 단원을 참조하십시오.

- 버킷의 위치에 대한 정보를 가져옵니다. Amazon S3는 버킷과 연결된 location 하위 리소스에 버킷 위치 정보를 저장합니다. SDK 라이브러리는 GET 버킷 위치 요청(GET 버킷 위치 참조)을 보내 이 정보를 가져옵니다.

Amazon S3 콘솔 사용

Amazon S3 콘솔을 사용하여 버킷을 생성하려면 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷을 생성하려면 어떻게 해야 합니까? 단원을 참조하십시오.

AWS SDK for Java 사용

Example

이 예제에서는 AWS SDK for Java를 사용하여 Amazon S3 버킷을 생성하는 방법을 보여줍니다. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
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;
import com.amazonaws.services.s3.model.CreateBucketRequest;
import com.amazonaws.services.s3.model.GetBucketLocationRequest;

import java.io.IOException;

public class CreateBucket {

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

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            if (!s3Client.doesBucketExistV2(bucketName)) {
                // Because the CreateBucketRequest object doesn't specify a region, the
                // bucket is created in the region specified in the client.
                s3Client.createBucket(new CreateBucketRequest(bucketName));

                // Verify that the bucket was created by retrieving it and checking its
 location.
                String bucketLocation = s3Client.getBucketLocation(new
 GetBucketLocationRequest(bucketName));
                System.out.println("Bucket location: " + bucketLocation);
            }
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and 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();
        }
    }
}
```

.NET용 AWS SDK 사용

실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

Example

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

namespace Amazon.DocSamples.S3
{
    class CreateBucketTest
    {
        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()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            CreateBucketAsync().Wait();
        }

        static async Task CreateBucketAsync()
        {
            try
            {
                if (!(await AmazonS3Util.DoesS3BucketExistAsync(s3Client, bucketName)))
                {
                    var putBucketRequest = new PutBucketRequest
                    {
                        BucketName = bucketName,
                        UseClientRegion = true
                    };

                    PutBucketResponse putBucketResponse = await
 s3Client.PutBucketAsync(putBucketRequest);
                }
                // Retrieve the bucket location.
                string bucketLocation = await FindBucketLocationAsync(s3Client);
            }
            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);
```

```
            }
        }
        static async Task<string> FindBucketLocationAsync(IAmazonS3 client)
        {
            string bucketLocation;
            var request = new GetBucketLocationRequest()
            {
                BucketName = bucketName
            };
            GetBucketLocationResponse response = await
 client.GetBucketLocationAsync(request);
            bucketLocation = response.Location.ToString();
            return bucketLocation;
        }
    }
}
```

Ruby 버전 3용 AWS SDK 사용

실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Ruby용 AWS SDK 버전 3 사용 (p. 737) 단원을 참조하십시오.

Example

```
require 'aws-sdk-s3'

s3 = Aws::S3::Client.new(region: 'us-west-2')
s3.create_bucket(bucket: 'bucket-name')
```

다른 AWS SDK 사용

다른 AWS SDK 사용에 대한 자세한 내용은 Sample Code and Libraries 단원을 참조하십시오.

버킷 삭제 또는 비우기

일부 상황에서는 객체가 포함된 버킷을 삭제하거나 비워야 할 수 있습니다. 이 단원에서는 버전 관리 미사용 버킷에서 객체를 삭제하는 방법 및 버전 관리가 활성화된 버킷에서 객체 버전 및 삭제 마커를 삭제하는 방법을 설명합니다. 버전 관리에 대한 자세한 내용은 버전 관리 사용 (p. 468) 단원을 참조하십시오. 경우에 따라서는 버킷을 삭제하는 대신 비워야 할 수도 있습니다. 이 단원에서는 객체가 포함된 버킷을 삭제하거나 비울 수 있는 여러 가지 옵션을 설명합니다.

주제

- 버킷 삭제 (p. 61)
- 버킷 비우기 (p. 64)

버킷 삭제

AWS SDK를 사용하여 프로그래밍 방식으로 버킷과 그 내용을 삭제할 수 있습니다. 버킷의 수명 주기 구성을 사용하여 내용을 비운 후 버킷을 삭제할 수도 있습니다. Amazon S3 콘솔과 AWS CLI를 사용하는 등의 추가 옵션도 있지만 이러한 방법에는 버킷에 포함된 객체 수 및 버킷의 버전 관리 상태에 따라 제한이 있습니다.

주제

- 버킷 삭제: Amazon S3 콘솔 사용 (p. 62)
- 버킷 삭제: AWS CLI 사용 (p. 62)
- 버킷 삭제: AWS SDKs 사용 (p. 62)

버킷 삭제: Amazon S3 콘솔 사용

Amazon S3 콘솔을 사용하여 비어 있거나 비어 있지 않은 버킷을 삭제할 수 있습니다. Amazon S3 콘솔을 사용하여 버킷을 삭제하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷을 삭제하려면 어떻게 해야 합니까? 단원을 참조하십시오.

버킷 삭제: AWS CLI 사용

버킷의 버전 관리가 활성화되지 않은 경우에만 AWS CLI를 사용하여 객체가 포함된 버킷을 삭제할 수 있습니다. 버킷의 버전 관리가 활성화되지 않은 경우 `rb`(버킷 제거) AWS CLI 명령을 `--force` 파라미터와 함께 사용하여 비어 있지 않은 버킷을 제거할 수 있습니다. 이 명령은 먼저 모든 객체를 삭제한 후 버킷을 삭제합니다.

```
$ aws s3 rb s3://bucket-name --force
```

자세한 정보는 AWS Command Line Interface 사용 설명서의 AWS 명령줄 인터페이스와 함께 상위 수준 S3 명령 사용을 참조하십시오.

버킷 삭제: AWS SDKs 사용

AWS SDK를 사용하여 버킷을 삭제할 수 있습니다. 다음 단원에서는 Java 및 .NET용 AWS SDK를 사용하여 버킷을 삭제하는 방법의 예를 제시합니다. 먼저 코드가 버킷의 객체를 삭제한 후 버킷을 삭제합니다. 그 밖의 AWS SDK에 대한 자세한 정보는 Amazon Web Services용 도구를 참조하십시오.

AWS SDK for Java 제품을 사용하여 버킷 삭제

다음 Java 예제는 객체가 포함된 버킷을 삭제합니다. 모든 객체를 삭제한 후 버킷을 삭제합니다. 이 예제는 버전 관리가 활성화되거나 활성화되지 않은 버킷에 적용됩니다.

> **Note**
>
> 버전 관리를 사용하지 않는 버킷의 경우 모든 객체를 직접 삭제한 다음 버킷을 삭제할 수 있습니다. 버전 관리를 사용하는 버킷의 경우 버킷을 삭제하기 전에 객체 버전을 모두 삭제해야 합니다.

실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
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;
import com.amazonaws.services.s3.model.*;

import java.util.Iterator;

public class DeleteBucket {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
```

```
        String bucketName = "*** Bucket name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Delete all objects from the bucket. This is sufficient
            // for unversioned buckets. For versioned buckets, when you attempt to delete
objects, Amazon S3 inserts
            // delete markers for all objects, but doesn't delete the object versions.
            // To delete objects from versioned buckets, delete all of the object versions
before deleting
            // the bucket (see below for an example).
            ObjectListing objectListing = s3Client.listObjects(bucketName);
            while (true) {
                Iterator<S3ObjectSummary> objIter =
objectListing.getObjectSummaries().iterator();
                while (objIter.hasNext()) {
                    s3Client.deleteObject(bucketName, objIter.next().getKey());
                }

                // If the bucket contains many objects, the listObjects() call
                // might not return all of the objects in the first listing. Check to
                // see whether the listing was truncated. If so, retrieve the next page of
objects
                // and delete them.
                if (objectListing.isTruncated()) {
                    objectListing = s3Client.listNextBatchOfObjects(objectListing);
                } else {
                    break;
                }
            }

            // Delete all object versions (required for versioned buckets).
            VersionListing versionList = s3Client.listVersions(new
ListVersionsRequest().withBucketName(bucketName));
            while (true) {
                Iterator<S3VersionSummary> versionIter =
versionList.getVersionSummaries().iterator();
                while (versionIter.hasNext()) {
                    S3VersionSummary vs = versionIter.next();
                    s3Client.deleteVersion(bucketName, vs.getKey(), vs.getVersionId());
                }

                if (versionList.isTruncated()) {
                    versionList = s3Client.listNextBatchOfVersions(versionList);
                } else {
                    break;
                }
            }

            // After all objects and object versions are deleted, delete the bucket.
            s3Client.deleteBucket(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();
        }
    }
}
```

버킷 비우기

AWS SDK를 사용하여 프로그래밍 방식으로 버킷의 내용을 비워 모든 내용을 삭제하되 버킷은 유지할 수 있습니다. 객체 만료를 지정하도록 버킷의 수명 주기 구성을 지정하여 Amazon S3에서 객체를 삭제할 수 있습니다. Amazon S3 콘솔과 AWS CLI를 사용하는 등의 추가 옵션도 있지만 이 방법에는 버킷에 포함된 객체 수 및 버킷의 버전 관리 상태에 따라 제한이 있습니다.

주제

- 버킷 비우기: Amazon S3 콘솔 사용 (p. 64)
- 버킷 비우기: AWS CLI 사용 (p. 64)
- 버킷 비우기: 수명 주기 구성 사용 (p. 64)
- 버킷 비우기: AWS SDKs 사용 (p. 65)

버킷 비우기: Amazon S3 콘솔 사용

Amazon S3 콘솔을 사용하여 버킷을 비우는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷을 비우려면 어떻게 해야 합니까? 단원을 참조하십시오.

버킷 비우기: AWS CLI 사용

버킷의 버전 관리가 활성화되지 않은 경우에만 AWS CLI를 사용하여 버킷을 비울 수 있습니다. 버킷의 버전 관리가 활성화되지 않은 경우 `rm`(제거) AWS CLI 명령을 `--recursive` 파라미터와 함께 사용하여 버킷을 비우거나 특정 키 이름 접두사를 가진 객체의 하위 집합을 제거할 수 있습니다.

다음 `rm` 명령은 키 이름 접두사 `doc`가 있는 객체(예: `doc/doc1` 및 `doc/doc2`)를 제거합니다.

```
$ aws s3 rm s3://bucket-name/doc --recursive
```

다음 명령을 사용하여 접두사를 지정하지 않고 모든 객체를 제거합니다.

```
$ aws s3 rm s3://bucket-name --recursive
```

자세한 정보는 AWS Command Line Interface 사용 설명서의 AWS 명령줄 인터페이스와 함께 상위 수준 S3 명령 사용을 참조하십시오.

> **Note**
>
> 버전 관리가 활성화된 버킷에서 객체를 제거할 수 없습니다. 이 명령을 사용하면 객체를 삭제할 때 Amazon S3가 삭제 마커를 추가합니다. 버전 관리에 대한 자세한 내용은 버전 관리 사용 (p. 468) 단원을 참조하십시오.

버킷 비우기: 수명 주기 구성 사용

버킷의 수명 주기를 구성하여 객체 만료를 지정하고 Amazon S3가 만료된 객체를 삭제하도록 요청합니다. 수명 주기 구성 규칙을 추가하여 모든 객체 또는 특정 키 이름 접두사를 가진 객체의 하위 집합에 대해 만료를 지정할 수 있습니다. 예를 들어, 버킷의 모든 객체를 제거하려는 경우 생성 하루 후에 객체가 만료되도록 수명 주기 규칙을 설정할 수 있습니다.

버킷의 버전 관리가 활성화된 경우 최신이 아닌 객체가 만료되도록 규칙을 구성할 수도 있습니다. 버전 관리가 활성화된 버킷의 콘텐츠를 완전히 비우려면 버킷에 있는 최신 객체와 최신이 아닌 객체 모두에 만료 정책을 구성해야 합니다.

자세한 내용은 객체 수명 주기 관리 (p. 123) 및 객체 만료 이해 (p. 129) 단원을 참조하십시오.

버킷 비우기: AWS SDKs 사용

AWS SDK를 사용하여 버킷을 비우거나 특정 키 이름 접두사를 가진 객체의 하위 집합을 제거할 수 있습니다.

AWS SDK for Java 제품을 사용하여 버킷을 비우는 방법의 예는 AWS SDK for Java 제품을 사용하여 버킷 삭제 (p. 62) 단원을 참조하십시오. 버킷의 버전 관리가 활성화되었는지 여부와 관계 없이 코드가 모든 객체를 삭제한 후 버킷을 삭제합니다. 버킷을 비우기만 하려면 버킷을 삭제하는 문을 제거해야 합니다.

그 밖의 AWS SDK 사용에 대한 자세한 정보는 Amazon Web Services용 도구를 참조하십시오.

S3 버킷에 대한 Amazon S3 기본 암호화

Amazon S3 기본 암호화를 사용하면 S3 버킷의 기본 암호화 동작을 설정할 수 있습니다. 버킷에 저장되는 모든 새 객체를 암호화하도록 버킷에 대한 기본 암호화를 설정할 수 있습니다. Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) 또는 AWS Key Management Service(AWS KMS)에 저장된 고객 마스터 키(CMK)를 사용한 서버 측 암호화로 객체를 암호화합니다.

서버 측 암호화를 사용하는 경우 Amazon S3에서는 객체를 디스크에 저장하기 전에 암호화하고 객체를 다운로드할 때 암호 해독합니다. 서버 측 암호화를 사용한 데이터 보호 및 암호화 키 관리에 대한 자세한 내용은 서버 측 암호화를 사용하여 데이터 보호 (p. 263) 단원을 참조하십시오.

주제

- S3 버킷에 대한 Amazon S3 기본 암호화를 설정하려면 어떻게 해야 합니까? (p. 65)
- 교차 계정 작업에 암호화 사용 (p. 66)
- 복제에 기본 암호화 사용 (p. 66)
- CloudTrail 및 CloudWatch를 사용한 기본 암호화 모니터링 (p. 66)
- 추가 정보 (p. 67)

S3 버킷에 대한 Amazon S3 기본 암호화를 설정하려면 어떻게 해야 합니까?

이 단원에서는 Amazon S3 기본 암호화를 설정하는 방법을 설명합니다. AWS SDK, Amazon S3 REST API, AWS Command Line Interface(AWS CLI) 또는 Amazon S3 콘솔을 사용하여 기본 암호화를 활성화할 수 있습니다. S3 버킷에 대한 기본 암호화를 설정하는 가장 간편한 방법은 을 사용하는 것입니다.

버킷에서 기본 암호화를 설정하려면 다음 방법 중 하나를 사용합니다.

- Amazon S3 콘솔을 사용합니다. 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷의 기본 암호화를 활성화하려면 어떻게 해야 합니까? 단원을 참조하십시오.
- REST API PUT 버킷 암호화 작업을 사용하여 기본 암호화를 활성화하고 사용할 서버 측 암호화 유형(SSE-S3 또는 SSE-KMS)을 설정합니다.
- AWS CLI 및 AWS SDK를 사용합니다. 자세한 내용은 AWS SDK, CLI 및 Explorer 사용 (p. 727) 단원을 참조하십시오.

버킷에 대한 기본 암호화를 활성화한 후에는 다음 암호화 동작이 적용됩니다.

- 기본 암호화가 활성화되기 전에 버킷에 있었던 객체의 암호화는 변경되지 않습니다.
- 기본 암호화를 활성화한 후 객체를 업로드하는 경우:

  - `PUT` 요청 헤더에 암호화 정보가 포함되지 않는 경우 Amazon S3는 버킷의 기본 암호화 설정을 사용하여 객체를 암호화합니다.
  - `PUT` 요청 헤더에 암호화 정보가 포함되는 경우 Amazon S3는 객체를 Amazon S3에 저장하기 전에 `PUT` 요청의 암호화 정보를 사용하여 객체를 암호화합니다.
- 기본 암호화 구성에 대해 SSE-KMS 옵션을 사용할 경우 AWS KMS에 대한 RPS(초당 요청 수) 제한이 적용됩니다. AWS KMS 제한과 한도 증가를 요청하는 방법에 대한 자세한 내용은 AWS KMS 제한 단원 참조하십시오.

단일 요청으로 기존 Amazon S3 객체를 암호화하기 위해 Amazon S3 Batch 작업을 사용할 수 있습니다. 작업할 객체 목록을 Amazon S3 Batch 작업에 제공하고, Amazon S3 Batch 작업은 각각의 API를 호출하여 지정된 작업을 수행합니다. 복사 작업을 사용하여 암호화되지 않은 기존 객체를 복사하고 암호화된 새로운 객체를 동일한 버킷에 작성할 수 있습니다. 단일 Amazon S3 Batch 작업으로 엑사바이트 규모의 데이터가 포함된 수십억 개의 객체에서 지정된 작업을 수행할 수 있습니다.

> **Note**
>
> SSE-KMS를 사용하는 기본 버킷 암호화가 있는 Amazon S3 버킷은 서버 액세스 로깅 (p. 705)의 대상 버킷으로 사용할 수 없습니다. 서버 액세스 로그 대상 버킷에는 SSE-S3 기본 암호화만 지원됩니다.

교차 계정 작업에 암호화 사용

교차 계정 작업에 암호화를 사용하는 경우 다음 사항에 유의하십시오.

- aws/s3 AWS 관리형 CMK는 요청 시간에 CMK ARN 또는 별칭이 제공되지도 않고 버킷의 기본 암호화 구성을 통해서도 제공되지 않는 경우에 사용됩니다.
- 자체 CMK를 지정하는 경우 정규화된 CMK 키 ARN을 사용해야 합니다. CMK 별칭을 사용하는 경우 KMS가 요청자의 계정 내에서 키를 확인한다는 점에 유의하십시오. 이로 인해 버킷 관리자가 아닌 요청자에 속한 CMK로 데이터가 암호화될 수 있습니다.
- 사용자(요청자)에게 `Encrypt` 권한이 부여된 키를 지정해야 합니다. 자세한 내용은 키 사용자가 암호화 작업에 CMK를 사용하도록 허용을 참조하십시오.

복제에 기본 암호화 사용

복제 대상 버킷에 대한 기본 암호화를 활성화한 후에는 다음 암호화 동작이 적용됩니다.

- 원본 버킷의 객체가 암호화되지 않은 경우 대상 버킷의 복제본 객체는 대상 버킷의 기본 암호화 설정을 사용하여 암호화됩니다. 결과적으로 원본 객체의 `ETag`는 복제본 객체의 `ETag`와 다릅니다. 이러한 차이점을 수용하도록 `ETag`를 사용하는 애플리케이션을 업데이트해야 합니다.

- 원본 버킷의 객체가 SSE-S3 또는 SSE-KMS를 사용하여 암호화되는 경우 대상 버킷의 복제본 객체는 원본 객체 암호화와 동일한 암호화를 사용합니다. 대상 버킷의 기본 암호화 설정은 사용되지 않습니다.

SSE-KMS와 함께 기본 암호화를 사용하는 방법에 대한 자세한 내용은 암호화된 객체 복제 (p. 629) 단원을 참조하십시오.

CloudTrail 및 CloudWatch를 사용한 기본 암호화 모니터링

AWS CloudTrail 이벤트를 통해 기본 암호화 구성 요청을 추적할 수 있습니다. CloudTrail 로그에 사용되는 API 이벤트 이름은 `PutBucketEncryption`, `GetBucketEncryption` 및 `DeleteBucketEncryption`입

니다. S3 버킷 레벨 작업을 이벤트 유형으로 사용하여 를 생성할 수도 있습니다. CloudTrail 이벤트에 대한 자세한 내용은 CloudTrail 데이터 이벤트로 S3 버킷에 대해 객체 수준 로깅을 활성화하려면 어떻게 합니까?를 참조하십시오.

CloudTrail 로그를 객체 레벨 Amazon S3 작업에 사용하여 Amazon S3에 대한 `PUT` 및 `POST` 요청을 추적하고 수신 `PUT` 요청에 암호화 헤더가 없는 경우 객체를 암호화하는 데 기본 암호화가 사용되는지 여부를 확인할 수 있습니다.

Amazon S3가 기본 암호화 설정을 사용하여 객체를 암호화하는 경우 로그에는 `"SSEApplied":"Default_SSE_S3" or "SSEApplied":"Default_SSE_KMS"` 필드가 이름/값 페어로 포함됩니다.

Amazon S3가 `PUT` 암호화 헤더를 사용하여 객체를 암호화하는 경우 로그에는 `"SSEApplied":"SSE_S3", "SSEApplied":"SSE_KMS` 또는 `"SSEApplied":"SSE_C"` 필드가 이름/값 페어로 포함됩니다. 멀티파트 업로드의 경우 이 정보가 `InitiateMultipartUpload` API 요청에 포함됩니다. CloudTrail 및 CloudWatch에 대한 자세한 내용은 Amazon S3 모니터링 (p. 668) 단원을 참조하십시오.

추가 정보

- PUT 버킷 암호화
- DELETE 버킷 암호화
- GET 버킷 암호화

Amazon S3 Transfer Acceleration

Amazon S3 Transfer Acceleration은 거리가 먼 클라이언트와 S3 버킷 간에 파일을 빠르고, 쉽고, 안전하게 전송할 수 있게 해줍니다. Transfer Acceleration은 전 세계적으로 분산되어 있는 Amazon CloudFront의 엣지 로케이션을 활용합니다. 엣지 로케이션에 도착한 데이터는 최적화된 네트워크 경로를 통해 Amazon S3로 라우팅됩니다.

Transfer Acceleration을 사용할 경우 데이터 전송 요금이 추가로 적용될 수 있습니다. 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

주제

- Amazon S3 Transfer Acceleration을 사용하는 이유는 무엇입니까? (p. 67)
- Amazon S3 Transfer Acceleration 시작하기 (p. 68)
- Amazon S3 Transfer Acceleration을 사용하기 위한 요구 사항 (p. 69)
- Amazon S3 Transfer Acceleration 예 (p. 69)

Amazon S3 Transfer Acceleration을 사용하는 이유는 무엇입니까?

버킷에서 Transfer Acceleration을 사용하는 이유는 다음과 같이 다양합니다.

- 전 세계 각지에서 중앙의 버킷으로 업로드하는 고객이 있을 경우
- 전 세계에 정기적으로 수 기가바이트에서 수 테라바이트의 데이터를 전송할 경우
- Amazon S3에 업로드할 경우 인터넷을 통해 사용 가능한 대역폭을 충분히 활용하지 못할 수 있습니다.

언제 Transfer Acceleration을 사용하는지에 대한 자세한 내용은 Amazon S3 FAQ를 참조하십시오.

Amazon S3 Transfer Acceleration 속도 비교 도구 사용

Amazon S3 Transfer Acceleration 속도 비교 도구를 사용하면 Amazon S3 리전에서 속도를 높인 경우와 그렇지 않은 경우의 업로드 속도를 비교할 수 있습니다. 속도 비교 도구는 멀티파트 업로드를 사용하여 브라우저에서 Transfer Acceleration을 사용하거나 사용하지 않고 여러 Amazon S3 리전으로 파일을 전송합니다.

속도 비교 도구는 다음과 같은 방법으로 이용할 수 있습니다.

- 다음 URL을 브라우저 창에 복사합니다. *region*을 현재 사용하는 리전으로 바꾸고(예: us-west-2), *yourBucketName*을 평가하려는 버킷의 이름으로 바꿉니다.

  ```
  http://s3-accelerate-speedtest.s3-accelerate.amazonaws.com/en/accelerate-speed-comparsion.html?region=region&origBucketName=yourBucketName
  ```

  Amazon S3가 지원하는 리전 목록은 Amazon Web Services 일반 참조의 리전 및 엔드포인트를 참조하십시오.
- Amazon S3 콘솔을 사용합니다. 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 Transfer Acceleration 사용을 참조하십시오.

Amazon S3 Transfer Acceleration 시작하기

Amazon S3 Transfer Acceleration 사용을 시작하려면 다음 단계를 수행합니다.

1. 버킷에서 Transfer Acceleration 설정 – 전송 속도 향상 기능을 사용하려는 경우 버킷 이름이 DNS 이름 지정 요구 사항을 준수해야 하며 마침표(".")를 포함하면 안 됩니다.

   다음과 같은 방법으로 버킷에서 Transfer Acceleration을 설정할 수 있습니다.
   - Amazon S3 콘솔을 사용합니다. 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 Transfer Acceleration 설정 단원을 참조하십시오.
   - REST API PUT Bucket accelerate 작업을 사용합니다.
   - AWS CLI와 AWS SDK를 사용합니다. 자세한 내용은 AWS SDK, CLI 및 Explorer 사용 (p. 727) 단원을 참조하십시오.
2. 다음과 같은 s3 가속화 엔드포인트 도메인 이름 중 하나를 사용하여 가속 설정된 버킷과 데이터를 주고받을 수 있습니다.
   - *bucketname*.s3-accelerate.amazonaws.com 가속 설정된 버킷에 액세스합니다.
   - *bucketname*.s3-accelerate.dualstack.amazonaws.com 가속 설정된 버킷에 IPv6을 통해 액세스합니다. Amazon S3 듀얼 스택 엔드포인트는 IPv6 및 IPv4를 통한 S3 버킷 요청을 지원합니다. Transfer Acceleration 듀얼 스택 엔드포인트는 가상 호스팅 방식의 엔드포인트 이름만 사용합니다. 자세한 내용은 IPv6을 통해 요청하기 (p. 12) 및 Amazon S3 듀얼 스택 엔드포인트 사용 (p. 14) 단원을 참조하십시오.

     Important

     현재 듀얼 스택 가속 엔드포인트에 대한 지원은 AWS Java SDK에서만 가능합니다. AWS CLI 및 기타 AWS SDK를 곧 지원할 예정입니다.

   Note

   가속 엔드포인트뿐 아니라 일반 엔드포인트도 계속 사용할 수 있습니다.

   Transfer Acceleration를 설정한 후에는 Amazon S3 PUT 객체 및 GET 객체 요청을 s3-accelerate 엔드포인트 도메인으로 지정할 수 있습니다. 예를 들어 현재 PUT 요청에 호스트 이름이 mybucket.s3.us-east-1.amazonaws.com인 PUT Object를 사용하는 REST API 애플리케이션이 있다고 가정해 보겠습니다. PUT 속도를 높이려면 요청에서 호스트 이름을 mybucket.s3-accelerate.amazonaws.com으

로 변경만 하면 됩니다. 일반 업로드 속도를 다시 사용하려면 이름을 다시 mybucket.s3.us-east-1.amazonaws.com으로 바꿔 주면 됩니다.

Transfer Acceleration 설정 후 성능 이점이 느껴질 때까지 최대 20분이 걸릴 수 있습니다. 그러나 Transfer Acceleration 설정 즉시 가속 엔드포인트를 사용할 수 있습니다.

AWS CLI, AWS SDK 및 Amazon S3와 데이터를 주고받는 다른 도구에서 가속 엔드포인트를 사용할 수 있습니다. AWS SDK를 사용할 경우 지원되는 몇 가지 언어에서는 가속 엔드포인트 클라이언트 구성 플래그를 사용하므로, Transfer Acceleration의 엔드포인트를 *bucketname*.s3-accelerate.amazonaws.com으로 명시적으로 설정할 필요가 없습니다. 가속 엔드포인트 클라이언트 구성 플래그를 사용하는 방법의 예를 보려면 Amazon S3 Transfer Acceleration 예 (p. 69) 단원을 참조하십시오.

전송 가속 엔드포인트에서 모든 Amazon S3 작업을 사용할 수 있습니다. 단, GET Service(버킷 나열), PUT Bucket(버킷 생성), DELETE Bucket 작업은 예외입니다. 또한 Amazon S3 Transfer Acceleration은 PUT Object - Copy를 사용하는 교차 리전 복사를 지원하지 않습니다.

Amazon S3 Transfer Acceleration을 사용하기 위한 요구 사항

다음은 S3 버킷에서 Transfer Acceleration을 사용하기 위한 요구 사항입니다.

- Transfer Acceleration은 가상 방식 요청에서만 지원됩니다. 가상 방식 요청에 대한 자세한 내용은 REST API를 사용하여 요청 (p. 42) 단원을 참조하십시오.
- Transfer Acceleration에 사용되는 버킷의 이름은 DNS를 따라야 하며 마침표(".")를 포함할 수 없습니다.
- 버킷에서 Transfer Acceleration을 설정해야 합니다. 버킷에서 Transfer Acceleration을 활성화하면 버킷으로의 데이터 전송 속도가 증가하기까지 최대 20분이 걸릴 수 있습니다.
- Transfer Acceleration 설정된 버킷에 액세스하려면 엔드포인트 *bucketname*`.s3-accelerate.amazonaws.com`을 사용해야 합니다. 또는 듀얼 스택 엔드포인트 *bucketname*`.s3-accelerate.dualstack.amazonaws.com`을 사용하여 IPv6을 통해 해당 버킷에 연결해야 합니다.
- 전송 가속 상태를 설정하려면 버킷 소유자여야 합니다. 버킷 소유자는 다른 사용자에게 버킷에 가속 상태를 설정할 수 있는 권한을 할당할 수 있습니다. `s3:PutAccelerateConfiguration` 권한은 사용자가 버킷에서 Transfer Acceleration을 활성화 또는 비활성화할 수 있도록 허용합니다. `s3:GetAccelerateConfiguration` 권한은 사용자가 버킷의 Transfer Acceleration 상태(`Enabled` 또는 `Suspended`.)를 반환할 수 있도록 허용합니다. 이러한 권한에 대한 자세한 내용은 — 버킷 하위 리소스 작업 예제 (p. 344) 및 Amazon S3의 Identity and Access Management (p. 299) 단원을 참조하십시오.

추가 정보

- GET Bucket 가속
- PUT Bucket 가속

Amazon S3 Transfer Acceleration 예

이 단원에서는 버킷에서 Amazon S3 Transfer Acceleration을 설정하는 방법과 설정된 버킷에 대해 가속 엔드포인트를 사용하는 방법의 예를 설명합니다. 일부 AWS SDK 지원 언어(예: Java, .NET)는 가속 엔드포인트 클라이언트 구성 플래그를 사용하므로, Transfer Acceleration의 엔드포인트를 *bucketname*.s3-accelerate.amazonaws.com으로 명시적으로 설정할 필요가 없습니다. For more information about Transfer Acceleration, see Amazon S3 Transfer Acceleration (p. 67).

주제

- Amazon S3 콘솔 사용 (p. 70)

- AWS Command Line Interface(AWS CLI)에서 Transfer Acceleration 사용 (p. 70)
- AWS SDK for Java에서 Transfer Acceleration 사용 (p. 71)
- .NET용 AWS SDK의 Transfer Acceleration 사용 (p. 72)
- JavaScript용 AWS SDK의 Transfer Acceleration 사용 (p. 73)
- AWS SDK for Python (Boto)에서 Transfer Acceleration 사용 (p. 73)
- 다른 AWS SDK 사용 (p. 73)

Amazon S3 콘솔 사용

Amazon S3 콘솔을 사용하여 버킷에서 Transfer Acceleration을 설정하는 방법은 Amazon Simple Storage Service 콘솔 사용 설명서의 Transfer Acceleration 설정 단원을 참조하십시오.

AWS Command Line Interface(AWS CLI)에서 Transfer Acceleration 사용

이 단원에서는 Transfer Acceleration에 사용되는 AWS CLI 명령의 예를 설명합니다. AWS CLI를 설치하는 지침은 AWS CLI 설정 (p. 733) 단원을 참조하십시오.

AWS CLI를 사용하여 버킷에 Transfer Acceleration 설정

AWS CLI put-bucket-accelerate-configuration 명령을 사용하여 버킷에 Transfer Acceleration을 설정하거나 중지할 수 있습니다. 다음은 버킷에서 Transfer Acceleration을 사용하기 위해 `Status=Enabled`를 설정하는 예제입니다. Transfer Acceleration을 중지하려면 `Status=Suspended`를 사용합니다.

Example

```
$ aws s3api put-bucket-accelerate-configuration --bucket bucketname --accelerate-
configuration Status=Enabled
```

AWS CLI에서 Transfer Acceleration 사용

AWS Config 파일의 프로필에서 `use_accelerate_endpoint` 구성 값을 `true`로 설정하면 s3 및 s3api AWS CLI 명령을 통해 이루어진 Amazon S3 요청이 모두 가속 엔드포인트 `s3-accelerate.amazonaws.com`으로 지정됩니다. 가속 엔드포인트를 사용하려면 버킷에서 Transfer Acceleration을 사용하도록 설정해야 합니다.

모든 요청은 가상 방식의 버킷 주소 지정 `my-bucket.s3-accelerate.amazonaws.com`을 사용하여 전송됩니다. `ListBuckets`, `CreateBucket`, `DeleteBucket` 요청은 엔드포인트에서 해당 작업을 지원하지 않기 때문에 가속 엔드포인트로 전송되지 않습니다. `use_accelerate_endpoint`에 대한 자세한 내용은 AWS CLI S3 구성을 참조하십시오.

다음 예는 기본 프로파일에서 `use_accelerate_endpoint`를 `true`로 설정합니다.

Example

```
$ aws configure set default.s3.use_accelerate_endpoint true
```

일부 AWS CLI 명령에는 가속 엔드포인트를 사용하고 다른 명령에는 사용하지 않으려면 다음 방법 중 하나를 사용합니다.

- s3 or s3api 명령에 대해 `--endpoint-url` 파라미터를 `https://s3-accelerate.amazonaws.com` 또는 `http://s3-accelerate.amazonaws.com`으로 설정하여 명령별로 가속 엔드포인트를 사용할 수 있습니다.

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

Transfer Acceleration 설정 버킷에 객체를 업로드하는 AWS CLI 예제

다음은 가속 엔드포인트를 사용하도록 구성된 기본 프로파일을 사용하여 Transfer Acceleration이 설정된 버킷에 파일을 업로드하는 예제입니다.

Example

```
$ aws s3 cp file.txt s3://bucketname/keyname --region region
```

다음은 `--endpoint-url` 파라미터를 사용하여 가속 엔드포인트를 지정함으로써 Transfer Acceleration이 설정된 버킷에 파일을 업로드하는 예제입니다.

Example

```
$ aws configure set s3.addressing_style virtual
$ aws s3 cp file.txt s3://bucketname/keyname --region region --endpoint-url http://s3-
accelerate.amazonaws.com
```

AWS SDK for Java에서 Transfer Acceleration 사용

Example

다음은 가속 엔드포인트를 사용하여 Amazon S3에 객체를 업로드하는 방법을 보여 주는 예제입니다. 이 예제는 다음을 수행합니다.

- 가속 엔드포인트를 사용하도록 구성된 `AmazonS3Client`를 만듭니다. 클라이언트가 액세스하는 모든 버킷에 Transfer Acceleration이 설정되어 있어야 합니다.
- 지정된 버킷에 대해 Transfer Acceleration을 설정합니다. 이 단계는 지정한 버킷에 아직 Transfer Acceleration이 설정되지 않은 경우에만 필요합니다.
- 지정된 버킷에 Transfer Acceleration이 설정되었는지 확인합니다.
- 버킷의 가속 엔드포인트를 사용하여 지정된 버킷에 새 객체를 업로드합니다.

Transfer Acceleration 사용에 관한 자세한 내용은 Amazon S3 Transfer Acceleration 시작하기 (p. 68) 단원을 참조하십시오. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
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;
import com.amazonaws.services.s3.model.BucketAccelerateConfiguration;
import com.amazonaws.services.s3.model.BucketAccelerateStatus;
import com.amazonaws.services.s3.model.GetBucketAccelerateConfigurationRequest;
import com.amazonaws.services.s3.model.SetBucketAccelerateConfigurationRequest;

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

```
        String keyName = "*** Key name ***";

        try {
            // Create an Amazon S3 client that is configured to use the accelerate
 endpoint.
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .enableAccelerateMode()
                    .build();

            // Enable Transfer Acceleration for the specified bucket.
            s3Client.setBucketAccelerateConfiguration(
                    new SetBucketAccelerateConfigurationRequest(bucketName,
                            new BucketAccelerateConfiguration(
                                    BucketAccelerateStatus.Enabled)));

            // Verify that transfer acceleration is enabled for the bucket.
            String accelerateStatus = s3Client.getBucketAccelerateConfiguration(
                    new GetBucketAccelerateConfigurationRequest(bucketName))
                    .getStatus();
            System.out.println("Bucket accelerate status: " + accelerateStatus);

            // Upload a new object using the accelerate endpoint.
            s3Client.putObject(bucketName, keyName, "Test object for transfer
 acceleration");
            System.out.println("Object \"" + keyName + "\" uploaded with transfer
 acceleration.");
        } 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();
        }
    }
}
```

.NET용 AWS SDK의 Transfer Acceleration 사용

다음은 .NET용 AWS SDK를 사용하여 버킷에 Transfer Acceleration을 설정하는 방법을 보여 주는 예제입니다. 실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

Example

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

namespace Amazon.DocSamples.S3
{
    class TransferAccelerationTest
    {
        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()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            EnableAccelerationAsync().Wait();
        }

        static async Task EnableAccelerationAsync()
        {
                try
                {
                    var putRequest = new PutBucketAccelerateConfigurationRequest
                    {
                        BucketName = bucketName,
                        AccelerateConfiguration = new AccelerateConfiguration
                        {
                            Status = BucketAccelerateStatus.Enabled
                        }
                    };
                    await s3Client.PutBucketAccelerateConfigurationAsync(putRequest);

                    var getRequest = new GetBucketAccelerateConfigurationRequest
                    {
                        BucketName = bucketName
                    };
                    var response = await
 s3Client.GetBucketAccelerateConfigurationAsync(getRequest);

                    Console.WriteLine("Acceleration state = '{0}' ", response.Status);
                }
                catch (AmazonS3Exception amazonS3Exception)
                {
                    Console.WriteLine(
                        "Error occurred. Message:'{0}' when setting transfer acceleration",
                        amazonS3Exception.Message);
                }
        }
    }
}
```

Transfer Acceleration이 설정된 버킷에 객체를 업로드할 경우, 다음과 같이 클라이언트를 만들 때 가속 엔드포인트 사용을 지정합니다.

```
var client = new AmazonS3Client(new AmazonS3Config
            {
                RegionEndpoint = TestRegionEndpoint,
                UseAccelerateEndpoint = true
            }
```

JavaScript용 AWS SDK의 Transfer Acceleration 사용

JavaScript용 AWS SDK를 사용하여 Transfer Acceleration을 활성화하는 예제는 JavaScript용 AWS SDK API 참조의 putBucketAccelerateConfiguration 작업 호출을 참조하십시오.

AWS SDK for Python (Boto)에서 Transfer Acceleration 사용

SDK for Python을 사용하여 Transfer Acceleration을 활성화하는 방법은 AWS SDK for Python (Boto 3) API Reference의 put_bucket_accelerate_configuration을 참조하십시오.

다른 AWS SDK 사용

다른 AWS SDK 사용에 대한 자세한 내용은 Sample Code and Libraries 단원을 참조하십시오.

요청자 지불 버킷

주제

- Amazon S3 콘솔을 사용한 요청자 지불 구성 (p. 74)
- REST API를 사용한 요청자 지불 구성 (p. 75)
- 요금 세부 정보 (p. 77)

일반적으로 버킷과 연결된 모든 Amazon S3 스토리지 및 데이터 전송 비용은 버킷 소유자가 지불합니다. 하지만 버킷 소유자는 버킷을 요청자 지불 버킷으로 구성할 수 있습니다. 요청자 지불 버킷은 버킷 소유자 대신 요청자가 데이터 다운로드 및 요청 비용을 지불합니다. 데이터 저장 비용은 항상 버킷 소유자가 지불합니다.

데이터를 공유하려 하지만 다른 사람이 데이터를 액세스하는 것에 대해 요금이 발생하는 것은 원치 않을 경우, 일반적으로 요청자 지불로 버킷을 구성합니다. 예를 들어, 우편번호부, 참조 데이터, 지역 관련 정보, 웹 크롤링 데이터 등과 같이 대량의 데이터 세트를 만들 경우 요청자 지불 버킷을 사용할 수 있습니다

> **Important**
>
> 버킷에 요청자 지불을 사용하도록 설정하면 버킷에 대한 익명 액세스가 허용되지 않습니다.

요청자 지불 버킷에 대한 모든 요청을 인증해야 합니다. Amazon S3는 요청 인증을 통해 요청자 지불 버킷을 사용하는 요청자를 식별하고 요금을 부과할 수 있습니다.

요청자가 요청에 앞서 AWS Identity and Access Management(IAM) 역할을 맡을 때는 해당 역할이 속한 계정에 요청에 대한 요금이 부과됩니다. IAM 역할에 대한 자세한 정보는 IAM 사용 설명서의 IAM 역할을 참조하십시오.

버킷을 요청자 지불 버킷으로 구성하면 요청자가 `x-amz-request-payer`를 요청의 헤더에 포함하거나 (POST, GET 및 HEAD 요청의 경우) 파라미터로 포함하여(REST 요청의 경우) 요청 및 데이터 다운로드에 대한 요금 부과를 인지했음을 표시해야 합니다.

요청자 지불 버킷은 다음을 지원하지 않습니다.

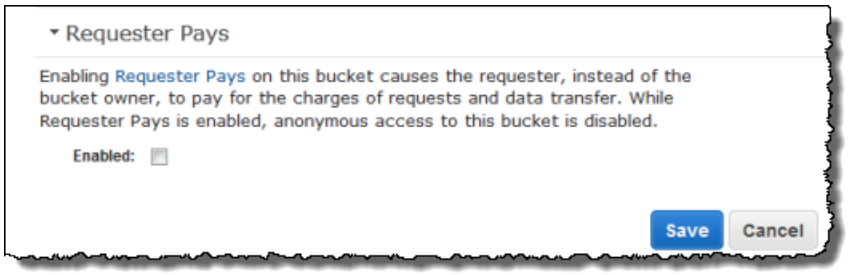
- 익명 요청
- BitTorrent
- SOAP 요청
- 요청자 지불 버킷을 사용할 경우 대상 버킷에 대해 최종 사용자 로깅을 설정할 수 없으며, 그 반대의 경우도 마찬가지입니다. 하지만 대상 버킷이 요청자 지불 버킷이 아닌 경우에는 요청자 지불 버킷에 대해 최종 사용자 로깅을 설정할 수 있습니다.

Amazon S3 콘솔을 사용한 요청자 지불 구성

Amazon S3 콘솔을 사용하여 요청자 지불에 대한 버킷을 구성할 수 있습니다.

요청자 지불에 대한 버킷을 구성하려면

1. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. Buckets 목록에서 버킷 이름 왼쪽의 세부 정보 아이콘과 속성를 차례로 클릭하면 버킷 속성이 표시됩니다.
3. 속성 창에서 요청자 지불을 클릭합니다.
4. 활성 확인란을 선택합니다.



REST API를 사용한 요청자 지불 구성

주제

- requestPayment 버킷 구성 설정 (p. 75)
- requestPayment 구성 검색 (p. 76)
- 요청자 지불 버킷의 객체 다운로드 (p. 76)

requestPayment 버킷 구성 설정

버킷 소유자만 버킷의 `RequestPaymentConfiguration.payer` 구성 값을 `BucketOwner`(기본값) 또는 `Requester`로 설정할 수 있습니다. `requestPayment` 리소스 설정은 선택 사항입니다. 기본적으로 버킷은 요청자 지불 버킷이 아닙니다.

요청자 지불 버킷을 일반 버킷으로 되돌리려면 `BucketOwner` 값을 사용합니다. 일반적으로 데이터를 Amazon S3 버킷에 업로드할 때는 `BucketOwner`를 사용하고, 버킷에 객체를 게시하기 전에 이 값을 `Requester`로 설정합니다.

requestPayment를 설정하려면

- `PUT` 요청을 사용하여 지정된 버킷에 대해 `Payer` 값을 `Requester`로 설정합니다.

```
PUT ?requestPayment HTTP/1.1
Host: [BucketName].s3.amazonaws.com
Content-Length: 173
Date: Wed, 01 Mar 2009 12:00:00 GMT
Authorization: AWS [Signature]

<RequestPaymentConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Payer>Requester</Payer>
</RequestPaymentConfiguration>
```

요청이 성공하면 Amazon S3는 다음과 비슷한 응답을 반환합니다.

```
HTTP/1.1 200 OK
x-amz-id-2: [id]
x-amz-request-id: [request_id]
Date: Wed, 01 Mar 2009 12:00:00 GMT
Content-Length: 0
Connection: close
Server: AmazonS3
x-amz-request-charged:requester
```

요청자 지불은 버킷 수준에서만 설정할 수 있으며, 버킷의 특정 객체에 대해서는 요청자 지불을 설정할 수 없습니다.

언제든지 버킷을 `BucketOwner` 또는 `Requester`로 구성할 수 있습니다. 하지만 새로운 구성 값이 적용되기까지 몇 분 정도 지연이 있을 수 있습니다.

Note

미리 서명된 URL을 제공하는 버킷 소유자는 버킷을 요청자 지불로 구성하기 전에 신중하게 고려해야 합니다. 특히 수명 주기가 매우 긴 URL의 경우에는 더욱 신중해야 합니다. 요청자가 버킷 소유자의 자격 증명을 사용하는 미리 서명된 URL을 사용할 때마다 버킷 소유자에게 요금이 부과됩니다.

requestPayment 구성 검색

`Payer` 리소스를 요청하여 버킷에 설정된 `requestPayment` 값을 확인할 수 있습니다.

requestPayment 리소스를 반환하려면

- 다음 요청에서와 같이 GET 요청을 사용하여 `requestPayment` 리소스를 확인합니다.

```
GET ?requestPayment HTTP/1.1
Host: [BucketName].s3.amazonaws.com
Date: Wed, 01 Mar 2009 12:00:00 GMT
Authorization: AWS [Signature]
```

요청이 성공하면 Amazon S3는 다음과 비슷한 응답을 반환합니다.

```
HTTP/1.1 200 OK
x-amz-id-2: [id]
x-amz-request-id: [request_id]
Date: Wed, 01 Mar 2009 12:00:00 GMT
Content-Type: [type]
Content-Length: [length]
Connection: close
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<RequestPaymentConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Payer>Requester</Payer>
</RequestPaymentConfiguration>
```

이 응답은 `payer` 값이 `Requester`로 설정되었음을 보여 줍니다.

요청자 지불 버킷의 객체 다운로드

요청자 지불 버킷에서 데이터를 다운로드하면 요청자가 요금을 지불하기 때문에, 요청자는 특수 파라미터인 `x-amz-request-payer`를 포함시킴으로써 요청자 본인에게 해당 다운로드에 대한 요금이 부과됨을 인지했다고 밝혀야 합니다. 요청자 지불 버킷의 객체를 액세스하려면 요청에 다음 중 하나가 포함되어야 합니다.

- GET, HEAD 및 POST 요청의 경우, 헤더에 `x-amz-request-payer : requester`를 포함합니다.
- 서명된 URL의 경우, 요청에 `x-amz-request-payer=requester`를 포함합니다.

요청이 성공하고 요청자에게 요금이 부과될 경우 응답에 `x-amz-request-charged:requester` 헤더가 포함됩니다. 요청에 `x-amz-request-payer`가 없을 경우 Amazon S3는 403 오류를 반환하고 버킷 소유자에게 요청에 대한 요금을 부과합니다.

Note

버킷 소유자는 요청에 `x-amz-request-payer`를 추가할 필요가 없습니다.

서명 계산에 `x-amz-request-payer`와 해당 값을 포함시켰는지 확인하십시오. 자세한 내용은 CanonicalizedAmzHeaders 요소 구성 (p. 748)을 참조하십시오.

요청자 지불 버킷에서 객체를 다운로드하려면

- 다음 요청에서와 같이 `GET` 요청을 사용하여 요청자 지불 버킷에서 객체를 다운로드합니다.

```
GET / [destinationObject] HTTP/1.1
Host: [BucketName].s3.amazonaws.com
x-amz-request-payer : requester
Date: Wed, 01 Mar 2009 12:00:00 GMT
Authorization: AWS [Signature]
```

GET 요청이 성공하고 요청자에게 요금이 부과되면 응답에 `x-amz-request-charged:requester`가 포함됩니다.

Amazon S3는 요청자 지불 버킷에서 객체를 받으려는 요청에 대해 `Access Denied` 오류를 반환할 수 있습니다. 자세한 내용은 오류 응답을 참조하십시오.

요금 세부 정보

성공한 요청자 지불 요청에 대한 요금 부과는 간단합니다. 요청자가 데이터 전송 및 요청에 대한 요금을 지불하고, 버킷 소유자는 데이터 저장 요금을 지불합니다. 하지만 다음과 같은 경우에는 버킷 소유자에게 요청에 대한 요금이 부과됩니다.

- 요청자가 `x-amz-request-payer` 파라미터를 헤더에 포함하지 않은 경우(GET, HEAD 또는 POST) 또는 요청에 파라미터로 포함하지 않은 경우(REST)(HTTP 코드 403)
- 인증 요청이 실패한 경우(HTTP 코드403)
- 요청이 익명인 경우(HTTP code 403)
- 요청이 SOAP 요청인 경우

버킷 및 액세스 제어

각 버킷에는 연결된 액세스 제어 정책이 있습니다. 이 정책은 버킷 내에서의 객체 생성, 삭제, 열거를 관리합니다. 자세한 내용은 Amazon S3의 Identity and Access Management (p. 299) 단원을 참조하십시오.

S3 버킷에 대한 결제 및 보고

Amazon Simple Storage Service(Amazon S3)를 사용하는 경우 사용자는 선불 요금을 지불하거나 얼마나 많은 콘텐츠를 보유할지 약정하지 않아도 됩니다. 기타 Amazon Web Services(AWS) 서비스와 마찬가지로 사용할 때, 사용한 내역에 대해서만 요금을 지불합니다.

AWS는 Amazon S3에 대해 다음과 같은 보고서를 제공합니다.

- 결제 보고서 – 여러 개의 보고서를 통해 Amazon S3을 포함해 사용 중인 AWS 서비스에 대한 모든 활동을 세부적으로 보여 줍니다. 버킷이 요청자 지불 버킷으로 생성되지 않은 한, AWS는 항상 S3 버킷의 소유자에게 Amazon S3 요금을 적용합니다. 요청자 지불에 대한 자세한 내용은 요청자 지불 버킷 (p. 74) 단원을 참조하십시오. 결제 보고서에 대한 자세한 내용은 Amazon S3에 대한 AWS 결제 보고서 (p. 78) 단원을 참조하십시오.
- 사용 보고서 – 특정 서비스에 대한 활동을 시간, 일 또는 월 기준으로 요약한 보고서입니다. 포함시킬 사용 유형과 작업을 선택할 수 있습니다. 데이터가 축적되는 방식을 선택할 수도 있습니다. 자세한 내용은 Amazon S3에 대한 AWS 사용 보고서 (p. 79) 단원을 참조하십시오.

Amazon S3에 대한 결제 및 사용 보고는 다음 주제를 참조하십시오.

주제

- Amazon S3에 대한 AWS 결제 보고서 (p. 78)
- Amazon S3에 대한 AWS 사용 보고서 (p. 79)
- Amazon S3에 대한 AWS 결제 및 사용 보고서 이해 (p. 81)
- 비용 할당 S3 버킷 태그 사용 (p. 88)

Amazon S3에 대한 AWS 결제 보고서

AWS에서 보내는 월 청구서에는 AWS 서비스 및 기능에 따라 사용 정보와 비용이 구분되어 있습니다. 월별 보고서, 비용 할당 보고서, 세부 결제 보고서 등 몇 가지 AWS 결제 보고서가 제공되고 있습니다. 결제 보고서 확인 방법에 대한 자세한 내용은 AWS Billing and Cost Management 사용 설명서의 청구서 보기 단원을 참조하십시오.

결제 보고서보다 Amazon S3 스토리지 사용에 대해 세부 정보를 제공하는 사용 보고서를 다운로드할 수도 있습니다. 자세한 내용은 Amazon S3에 대한 AWS 사용 보고서 (p. 79) 단원을 참조하십시오.

다음 표에는 Amazon S3 사용과 관련된 요금이 나와 있습니다.

Amazon S3 사용 요금

요금	설명
스토리지	S3 버킷에서의 객체 저장에 대한 요금이 부과됩니다. 부과되는 요금은 STANDARD, INTELLIGENT_TIERING, STANDARD_IA(자주 액세스하지 않는 경우의 IA), ONEZONE_IA, S3 Glacier, S3 Glacier Deep Archive 또는 Reduced Redundancy Storage(RRS) 같은 스토리지 클래스와 객체의 크기, 당월에 객체가 저장된 기간에 따라 다릅니다. 스토리지 클래스에 대한 자세한 정보는 Amazon S3 스토리지 클래스 (p. 107)를 참조하십시오.
모니터링 및 자동화	INTELLIGENT_TIERING 스토리지 클래스에 저장된 객체별 월 모니터링 및 자동화 요금을 지불하여 액세스 패턴을 모니터링하고 INTELLIGENT_TIERING의 액세스 티어 사이에서 객체를 이동합니다.
요청	S3 버킷 및 객체에 대한 요청(예를 들어, GET 요청)에 요금이 부과됩니다. 여기에는 수명 주기 요청이 포함됩니다. 요청에 대한 요금은 수행 중인 요청의 종류에 따라 다릅니다. 요청 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.
검색	STANDARD_IA, ONEZONE_IA, S3 Glacier 및 S3 Glacier Deep Archive 스토리지에 저장된 객체 검색에 대한 요금이 부과됩니다.
조기 삭제	최소 스토리지 약정 기간이 지나기 전에 INTELLIGENT_TIERING, STANDARD_IA, ONEZONE_IA, S3 Glacier 또는 S3 Glacier Deep Archive 스토리지에 저장된 객체를 삭제하는 경우에는 해당 객체에 대한 조기 삭제 요금이 부과됩니다.

요금	설명
스토리지 관리	계정의 버킷에서 활성화된 스토리지 관리 기능(Amazon S3 인벤토리 분석 및 객체 태그 지정)에 대한 요금이 부과됩니다.
대역폭	다음을 제외하고 Amazon S3에서 들어오고 나가는 모든 대역폭에 대해 요금이 부과됩니다. • 인터넷에서 전송되는 데이터 • 인스턴스가 S3 버킷과 동일한 AWS 리전에 있을 때 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스로 전송되는 데이터 • Amazon CloudFront(CloudFront)로 전송되는 데이터 또한 Amazon S3 Transfer Acceleration을 사용하여 전송된 모든 데이터에 대해 요금이 부과됩니다.

저장, 데이터 전송 및 서비스에 대한 Amazon S3 사용 요금의 세부 정보는 Amazon S3 요금 및 Amazon S3 FAQ 단원을 참조하십시오.

Amazon S3에 대한 결제 및 사용 보고서에서 사용되는 코드와 약어에 대한 내용은 Amazon S3에 대한 AWS 결제 및 사용 보고서 이해 (p. 81) 단원을 참조하십시오.

추가 정보

- Amazon S3에 대한 AWS 사용 보고서 (p. 79)
- 비용 할당 S3 버킷 태그 사용 (p. 88)
- AWS 결제 및 비용 관리
- Amazon S3 요금
- Amazon S3 FAQ
- S3 Glacier 요금

Amazon S3에 대한 AWS 사용 보고서

Amazon S3 스토리지 사용에 대해 자세히 알아보려면 동적으로 생성된 AWS 사용 보고서를 다운로드합니다. 포함시킬 사용 유형, 작업 및 기간을 선택할 수 있습니다. 데이터가 축적되는 방식을 선택할 수도 있습니다.

사용 보고서를 다운로드하면 사용 데이터를 시간, 일 또는 월 기준으로 집계할 수 있습니다. Amazon S3 사용 보고서는 아시아 태평양(시드니) 지역 외부로 전송된 데이터의 양 같은 사용 유형 및 AWS 리전에 따른 작업을 나열합니다.

Amazon S3 사용 보고서에는 다음 정보가 포함됩니다.

- 서비스 – `Amazon Simple Storage Service`
- 작업 – 버킷 또는 객체에서 수행되는 작업입니다. Amazon S3 작업에 대한 자세한 설명은 사용 보고서의 추적 작업 (p. 87) 단원을 참조하십시오.
- UsageType – 다음 값 중 하나입니다.
  - 스토리지의 유형을 식별하는 코드

- 요청의 유형을 식별하는 코드
- 검색의 유형을 식별하는 코드
- 데이터 전송의 유형을 식별하는 코드입니다.
- INTELLIGENT_TIERING, STANDARD_IA, ONEZONE_IA, S3 Glacier 또는 S3 Glacier Deep Archive 스토리지에서 조기 삭제를 식별하는 코드
- `StorageObjectCount` -- 지정된 버킷 내에 저장되는 객체의 수

Amazon S3 사용 유형에 대한 자세한 설명은 Amazon S3에 대한 AWS 결제 및 사용 보고서 이해 (p. 81) 단원을 참조하십시오.

- 리소스 – 나열된 사용과 관련된 버킷의 이름입니다.
- StartTime – 사용이 적용되는 날짜의 시작 시간으로, UTC(협정 세계시)를 따릅니다.
- EndTime – 사용이 적용되는 날짜의 종료 시간으로, UTC(협정 세계시)를 따릅니다.
- UsageValue – 다음 볼륨 값 중 하나입니다.
  - 지정된 기간 동안 요청 수
  - 전송된 데이터의 양(바이트 단위)
  - 바이트/시간(지정된 시간에 저장된 바이트 수)으로 저장되는 데이터 양
  - S3 Glacier Deep Archive, S3 Glacier, STANDARD_IA 또는 ONEZONE_IA 스토리지에서 복원과 관련된 데이터 양(바이트 단위)

  Tip

  Amazon S3가 객체에 대해 수신하는 모든 요청에 대한 세부 정보를 보려면 버킷에 대한 서버 액세스 로깅을 활성화하십시오. 자세한 내용은 Amazon S3 서버 액세스 로깅 (p. 705) 단원을 참조하십시오.

사용 보고서를 XML 또는 CSV(쉼표로 구분된 값) 파일로 다운로드할 수 있습니다. 다음은 스프레드시트 애플리케이션에 열려 있는 CSV 사용 보고서의 예입니다.

Service	Operation	UsageType	Resource	StartTime	EndTime	UsageValue
AmazonS3	HeadBucket	USW2-C3DataTransfer-Out-Bytes	admin-created3	6/1/2017 0:00	7/1/2017 0:00	15309
AmazonS3	PutObject	USW2-C3DataTransfer-In-Bytes	admin-created3	6/1/2017 0:00	7/1/2017 0:00	19062
AmazonS3	HeadBucket	USW2-Requests-Tier2	admin-created3	6/1/2017 0:00	7/1/2017 0:00	68
AmazonS3	PutObjectForRepl	USW1-Requests-SIA-Tier1	ca-example-bucket	6/1/2017 0:00	7/1/2017 0:00	178294
AmazonS3	PutObjectForRepl	USW1-USW2-AWS-In-Bytes	ca-example-bucket	6/1/2017 0:00	7/1/2017 0:00	387929083
AmazonS3	GetObjectForRepl	USW2-Requests-NoCharge	admin-created3	6/1/2017 0:00	7/1/2017 0:00	108
AmazonS3	GetObjectForRepl	USW2-USW1-AWS-Out-Bytes	my-test-bucket-bash	6/1/2017 0:00	7/1/2017 0:00	387910021

사용 보고서에 대한 자세한 내용은 Amazon S3에 대한 AWS 결제 및 사용 보고서 이해 (p. 81) 단원을 참조하십시오.

AWS 사용 보고서 다운로드

사용 보고서를 .xml 또는 .csv 파일로 다운로드할 수 있습니다.

사용 보고서를 다운로드하려면

1. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 제목 표시줄에서 AWS Identity and Access Management(IAM) 사용자 이름을 클릭하고 My Billing Dashboard(내 결제 대시보드)를 선택합니다.
3. 탐색 창에서 AWS 비용 및 사용 보고서를 선택합니다.
4. 기타 보고서에서 AWS 사용 보고서를 선택합니다.

5. 서비스:에서 Amazon Simple Storage Service를 선택합니다.
6. 사용 보고서 다운로드에서 다음 설정을 선택합니다.
   - 사용 유형 – Amazon S3 사용 유형에 대한 자세한 설명은 Amazon S3에 대한 AWS 결제 및 사용 보고서 이해 (p. 81) 단원을 참조하십시오.
   - 작업 – Amazon S3 작업에 대한 자세한 설명은 사용 보고서의 추적 작업 (p. 87) 단원을 참조하십시오.
   - 기간 – 보고서에서 평가하고자 하는 기간입니다.
   - 보고서 세부 수준 – 보고서에 시간, 일 또는 월 기준으로 소계를 포함할지 여부를 선택합니다.
7. 보고서의 형식을 선택하려면 해당 형식에 대한 다운로드를 선택한 다음, 프롬프트의 메시지에 따라 보고서를 확인 또는 저장합니다.

추가 정보

- Amazon S3에 대한 AWS 결제 및 사용 보고서 이해 (p. 81)
- Amazon S3에 대한 AWS 결제 보고서 (p. 78)

Amazon S3에 대한 AWS 결제 및 사용 보고서 이해

Amazon S3 결제 및 사용 보고서는 코드와 약어를 사용합니다. 예를 들어, 아래 표에 나와 있는 사용 보고서의 경우 ##이 다음 약어 중 하나로 바뀝니다.

- APE1: 아시아 태평양(홍콩)
- APN1: 아시아 태평양(도쿄)
- APN2: 아시아 태평양(서울)
- APS1: 아시아 태평양(싱가포르)
- APS2: 아시아 태평양(시드니)
- APS3: 아시아 태평양(뭄바이)
- CAN1: 캐나다(중부)
- EUN1: 유럽(스톡홀름)
- EUC1: 유럽(프랑크푸르트)
- EU: 유럽(아일랜드)
- EUW2: 유럽(런던)
- EUW3: 유럽(파리)
- MES1: 중동(바레인)
- SAE1: 남아메리카(상파울루)
- UGW1: AWS GovCloud (US-West)
- UGE1: AWS GovCloud(US-East)
- USE1 (or no prefix): 미국 동부(버지니아 북부)
- USE2: 미국 동부(오하이오)
- USW1: 미국 서부(캘리포니아 북부 지역)
- USW2: 미국 서부(오레곤)

AWS 리전별 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

다음 표의 첫째 열에는 결제 및 사용 보고서에 나타나는 사용 유형이 나열됩니다.

사용 유형

사용 유형	단위	Granularity	설명
region1-*region2*-AWS-In-Bytes	바이트	시간당	AWS Region2에서 AWS Region1으로 전송된 데이터의 양
region1-*region2*-AWS-Out-Bytes	바이트	시간당	AWS Region1에서 AWS Region2로 전송된 데이터의 양
region-BatchOperations-Jobs	개수	시간당	수행된 Amazon S3 Batch Operations 작업의 수.
region-BatchOperations-Objects	개수	시간당	Amazon S3 Batch Operations에서 수행한 객체 작업의 수.
region-DataTransfer-In-Bytes	바이트	시간당	인터넷에서 Amazon S3로 전송된 데이터의 양
region-DataTransfer-Out-Bytes	바이트	시간당	Amazon S3에서 인터넷으로 전송된 데이터의 양[1]
region-C3DataTransfer-In-Bytes	바이트	시간당	같은 AWS 리전 내에서 Amazon EC2에서 Amazon S3로 전송된 데이터의 양
region-C3DataTransfer-Out-Bytes	바이트	시간당	같은 AWS 리전 내에서 Amazon S3에서 Amazon EC2로 전송된 데이터의 양
region-S3G-DataTransfer-In-Bytes	바이트	시간당	S3 Glacier 또는 S3 Glacier Deep Archive 스토리지에서 객체를 복원하기 위해 Amazon S3으로 전송된 데이터의 양
region-S3G-DataTransfer-Out-Bytes	바이트	시간당	S3 Glacier 또는 S3 Glacier Deep Archive 스토리지로 객체를 전환하기 위해 Amazon S3에서 전송된 데이터의 양
region-DataTransfer-Regional-Bytes	바이트	시간당	같은 AWS 리전 내에서 Amazon S3에서 AWS 리소스로 전송된 데이터의 양
StorageObjectCount	개수	일별	지정된 버킷 내에 저장되는 객체의 수
region-CloudFront-In-Bytes	바이트	시간당	CloudFront 배포에서 AWS 리전으로 전송된 데이터의 양
region-CloudFront-Out-Bytes	바이트	시간당	AWS 리전에서 CloudFront 배포로 전송된 데이터의 양
region-EarlyDelete-ByteHrs	바이트/시간[2]	시간당	최소 90일 약정 종료 전 S3 Glacier 스토리지에서 삭제

사용 유형	단위	Granularity	설명
			된 객체에 대해 비례 할당으로 계산된 스토리지 사용량[3]
region-EarlyDelete-GDA	바이트/시간[2]	시간당	최소 180일 약정 종료 전 S3 Glacier Deep Archive 스토리지에서 삭제된 객체에 대해 비례 할당으로 계산된 스토리지 사용량[3]
region-EarlyDelete-SIA	바이트/시간	시간당	최소 30일 약정이 끝나기 전에 STANDARD_IA에서 삭제된 객체에 대해 비례 할당으로 계산된 스토리지 사용량[4]
region-EarlyDelete-ZIA	바이트/시간	시간당	최소 30일 약정이 끝나기 전에 ONEZONE_IA에서 삭제된 객체에 대해 비례 할당으로 계산된 스토리지 사용량[4]
region-EarlyDelete-SIA-SmObjects	바이트/시간	시간당	최소 30일 약정이 끝나기 전에 STANDARD_IA에서 삭제된 작은 객체(128KB 미만)에 대해 비례 할당으로 계산된 스토리지 사용량[4]
region-EarlyDelete-ZIA-SmObjects	바이트/시간	시간당	최소 30일 약정이 끝나기 전에 ONEZONE_IA에서 삭제된 작은 객체(128KB 미만)에 대해 비례 할당으로 계산된 스토리지 사용량[4]
region-Inventory-ObjectsListed	객체	시간당	인벤토리 목록에서 객체 그룹(객체가 버킷 또는 접두사별로 그룹화)에 나열된 객체의 수
region-Requests-S3 Glacier-Tier1	개수	시간당	S3 Glacier 객체에 대한 PUT, COPY, POST, InitiateMultipartUpload, UploadPart 또는 CompleteMultipartUpload 요청의 수
region-Requests-S3 Glacier-Tier2	개수	시간당	S3 Glacier 객체에 대해 나열되지 않은 GET 및 기타 모든 요청의 수
region-Requests-SIA-Tier1	개수	시간당	STANDARD_IA 객체에 대한 PUT, COPY, POST 또는 LIST 요청의 수
region-Requests-ZIA-Tier1	개수	시간당	ONEZONE_IA 객체에 대한 PUT, COPY, POST 또는 LIST 요청의 수

사용 유형	단위	Granularity	설명
region-Requests-SIA-Tier2	개수	시간당	STANDARD_IA 객체에 대한 GET 및 기타 모든 비 SIA-Tier1 요청의 수
region-Requests-ZIA-Tier2	개수	시간당	ONEZONE_IA 객체에 대한 GET 및 기타 모든 비 ZIA-Tier1 요청의 수
region-Requests-Tier1	개수	시간당	STANDARD, RRS 및 태그에서 PUT, COPY, POST 또는 LIST 요청의 수
region-Requests-Tier2	개수	시간당	GET 및 기타 모든 비 SIA-Tier1 요청의 수
region-Requests-Tier3	개수	시간당	S3 Glacier 또는 S3 Glacier Deep Archive 및 표준 S3 Glacier 복원 요청에 대한 수명 주기 요청의 수
region-Requests-Tier4	개수	시간당	INTELLIGENT_TIERING, STANDARD_IA 또는 ONEZONE_IA 스토리지로의 수명 주기 전환의 수
region-Requests-Tier5	개수	시간당	대량 S3 Glacier 복원 요청의 수
region-Requests-GDA-Tier1	개수	시간당	DEEP Archive 객체에 대한 PUT, COPY, POST, InitiateMultipartUpload, UploadPart 또는 CompleteMultipartUpload 요청 수
region-Requests-GDA-Tier2	개수	시간당	GET, HEAD, and LIST 요청 건수
region-Requests-GDA-Tier3	개수	시간당	S3 Glacier Deep Archive 표준 복원 요청의 수
region-Requests-GDA-Tier5	개수	시간당	대량 S3 Glacier Deep Archive 복원 요청의 수
region-Requests-Tier6	개수	시간당	신속 S3 Glacier 복원 요청 수
region-Bulk-Retrieval-Bytes	바이트	시간당	대량 S3 Glacier 또는 S3 Glacier Deep Archive 요청을 통해 검색된 데이터의 바이트 수
region-Requests-INT-Tier1	개수	시간당	INTELLIGENT_TIERING 객체에 대한 PUT, COPY, POST 또는 LIST 요청의 수

사용 유형	단위	Granularity	설명
region-Requests-INT-Tier2	개수	시간당	INTELLIGENT_TIERING 객체에 대한 GET 및 기타 모든 비-Tier1 요청의 수
region-Select-Returned-INT-Bytes	바이트	시간당	INTELLIGENT_TIERING 스토리지에서 Select 요청을 통해 반환된 데이터의 바이트 수
region-Select-Scanned-INT-Bytes	바이트	시간당	INTELLIGENT_TIERING 스토리지에서 Select 요청을 통해 스캔한 데이터의 바이트 수
region-EarlyDelete-INT	바이트/시간	시간당	최소 30일 약정이 끝나기 전에 INTELLIGENT_TIERING 에서 삭제된 객체에 대해 비례 할당으로 계산된 스토리지 사용량
region-Monitoring-Automation-INT	객체	시간당	INTELLIGENT_TIERING 스토리지 클래스에서 모니터링되고 자동으로 계층화된 고유한 객체 수
region-Expedited-Retrieval-Bytes	바이트	시간당	신속 S3 Glacier 요청을 통해 검색된 데이터의 바이트 수
region-Standard-Retrieval-Bytes	바이트	시간당	표준 S3 Glacier 또는 S3 Glacier Deep Archive 요청을 통해 검색된 데이터의 바이트 수
region-Retrieval-SIA	바이트	시간당	STANDARD_IA 스토리지에서 검색된 데이터의 바이트 수
region-Retrieval-ZIA	바이트	시간당	ONEZONE_IA 스토리지에서 검색된 데이터의 바이트 수
region-StorageAnalytics-ObjCount	객체	시간당	스토리지 분석을 통해 추적한 각 객체 그룹(객체가 버킷 또는 접두사별로 그룹화)의 고유한 객체 수
region-Select-Scanned-Bytes	바이트	시간당	STANDARD 스토리지에서 Select 요청을 통해 스캔된 데이터의 바이트 수
region-Select-Scanned-SIA-Bytes	바이트	시간당	STANDARD_IA 스토리지에서 Select 요청을 통해 스캔된 데이터의 바이트 수
region-Select-Scanned-ZIA-Bytes	바이트	시간당	ONEZONE_IA 스토리지에서 Select 요청을 통해 스캔된 데이터의 바이트 수

사용 유형	단위	Granularity	설명
region-Select-Returned-Bytes	바이트	시간당	STANDARD 스토리지에서 Select 요청을 통해 반환된 데이터의 바이트 수
region-Select-Returned-SIA-Bytes	바이트	시간당	STANDARD_IA 스토리지에서 Select 요청을 통해 반환된 데이터의 바이트 수
region-Select-Returned-ZIA-Bytes	바이트	시간당	ONEZONE_IA 스토리지에서 Select 요청을 통해 반환된 데이터의 바이트 수
region-TagStorage-TagHrs	Tag-Hours	일별	시간별로 보고된 버킷의 모든 객체에 대한 총 태그 수
region-TimedStorage-ByteHrs	바이트/시간	일별	STANDARD 스토리지에 데이터가 저장된 바이트/시간 수
region-TimedStorage-S3 GlacierByteHrs	바이트/시간	일별	S3 Glacier 스토리지에 데이터가 저장된 바이트/시간 수
region-TimedStorage-GlacierStaging	바이트/시간	일별	S3 Glacier 스테이징 스토리지에 데이터가 저장된 바이트/시간 수
region-TimedStorage-GDA-ByteHrs	바이트/시간	일별	S3 Glacier Deep Archive 스토리지에 데이터가 저장된 바이트/시간 수
region-TimedStorage-GDA-Staging	바이트/시간	일별	S3 Glacier Deep Archive 스테이징 스토리지에 데이터가 저장된 바이트/시간 수
region-TimedStorage-INT-FA-ByteHrs	바이트/시간	일별	INTELLIGENT_TIERING 스토리지의 빈번한 액세스 티어에 데이터가 저장되었던 바이트-시간 수입니다.
region-TimedStorage-INT-IA-ByteHrs	바이트/시간	일별	INTELLIGENT_TIERING 스토리지의 빈번하지 않은 액세스 티어에 데이터가 저장되었던 바이트-시간 수입니다.
region-TimedStorage-RRS-ByteHrsb	바이트/시간	일별	RRS(Reduced Redundancy Storage) 스토리지에 데이터가 저장된 바이트/시간 수
region-TimedStorage-SIA-ByteHrs	바이트/시간	일별	STANDARD_IA 스토리지에 데이터가 저장된 바이트/시간 수
region-TimedStorage-ZIA-ByteHrs	바이트/시간	일별	ONEZONE_IA 스토리지에 데이터가 저장된 바이트/시간 수

사용 유형	단위	Granularity	설명
region-TimedStorage-SIA-SmObjects	바이트/시간	일별	STANDARD_IA 스토리지에 작은 객체(128KB 미만)가 저장된 바이트/시간 수
region-TimedStorage-ZIA-SmObjects	바이트/시간	일별	ONEZONE_IA 스토리지에 작은 객체(128KB 미만)가 저장된 바이트/시간 수

참고:

1. 완료 전에 전송을 종료하면 전송되는 데이터의 양이 애플리케이션이 수신하는 데이터의 양을 초과할 수 있습니다. 전송 종료 요청을 동시에 실행할 수 없고 일부 데이터가 종료 요청의 전송 실행 대기 중인 경우에 이와 같은 불일치가 발생할 수 있습니다. 전송 중인 이 데이터는 "아웃바운드" 전송된 데이터로 청구됩니다.
2. 바이트/시간 단위에 대한 자세한 내용은 사용 바이트/시간을 GB/월 청구로 변환 (p. 87) 단원을 참조하십시오.
3. S3 Glacier 또는 S3 Glacier Deep Archive 스토리지 클래스에 보관된 객체는 S3 Glacier의 경우 90일, S3 Glacier Deep Archive의 경우 180일인 최소 스토리지 약정 기간이 경과하기 전에 삭제되거나 덮어쓰거나 다른 스토리지 클래스로 이전되면 남은 기간(일) 동안 기가바이트당 요금이 비례 할당으로 계산됩니다.
4. INTELLIGENT_TIERING, STANDARD_IA 또는 ONEZONE_IA 스토리지에 보관되는 객체의 경우에는 30일 이전에 삭제 또는 덮어쓰기 되거나 다른 스토리지 클래스로 이전되면 남은 기간에 대해 기가바이트별로 요금이 일할 계산됩니다.
5. STANDARD_IA 또는 ONEZONE_IA 스토리지에 보관되는 작은 객체(128KB 미만)의 경우에는 30일 이전에 삭제 또는 덮어쓰기 되거나 다른 스토리지 클래스로 이전되면 남은 기간에 대해 기가바이트별로 요금이 일할 계산됩니다.
6. INTELLIGENT_TIERING 스토리지 클래스에는 객체에 대해 청구 가능한 최소 객체 크기는 없지만 128KB 보다 작은 객체는 자동 계층화에 적합하지 않으며 항상 INTELLIGENT_TIERING 빈번한 액세스 티어에 대한 요금으로 청구됩니다.

사용 보고서의 추적 작업

작업은 지정된 사용 유형에 따라 AWS 객체 또는 버킷에 대해 취해진 조치를 설명합니다. 작업은 `PutObject` 또는 `ListBucket` 같이 이름에 설명이 포함된 코드로 표시가 됩니다. 이러한 코드를 사용하여 버킷에 대한 어떤 조치가 특정한 사용 유형을 생성했는지 확인합니다. 사용 보고서를 생성할 때 선택에 따라 보고서에 모든 작업 또는 GetObject 같은 특정 작업을 포함시킬 수 있습니다.

사용 바이트/시간을 GB/월 청구로 변환

매달 요금이 청구되는 스토리지 볼륨은 한달 동안 사용한 평균 스토리지 양에 따라 결정됩니다. AWS 계정으로 만든 버킷에 저장된 모든 객체 데이터와 메타데이터에 대해 요금이 청구됩니다. 메타데이터에 대한 자세한 정보는 객체 키와 메타데이터 (p. 102) 단원을 참조하십시오.

매달 말에 총 합계가 되는 TimedStorage-ByteHrs으로 스토리지 사용을 측정하여 월별 요금을 계산합니다. 사용 보고서는 바이트/시간 단위로 스토리지 사용을 보고하고, 결제 보고서는 GB/월 단위로 스토리지 사용을 보고합니다. 사용 보고서와 결제 보고서를 연결하려면 바이트/시간을 GB/월로 변환해야 합니다.

예를 들어, 100GB(107,374,182,400바이트)의 STANDARD Amazon S3 스토리지 데이터를 3월의 첫 15일 동안 버킷에 저장하고 100TB(109,951,162,777,600바이트)의 STANDARD Amazon S3 스토리지 데이터를 3월의 나머지 16일 동안 저장하면 사용량은 42,259,901,212,262,400바이트/시간이 됩니다.

먼저, 다음과 같이 총 바이트/시간 사용량을 계산합니다.

```
[107,374,182,400 bytes x 15 days x (24 hours/day)]
    + [109,951,162,777,600 bytes x 16 days x (24 hours/day)]
    = 42,259,901,212,262,400 byte-hours
```

그런 다음, 아래와 같이 바이트/시간을 GB/월로 변환합니다.

```
42,259,901,212,262,400 byte-hours/1,073,741,824 bytes per GB/24 hours per day
     /31 days in March
     =52,900 GB-Months
```

추가 정보

- Amazon S3에 대한 AWS 사용 보고서 (p. 79)
- Amazon S3에 대한 AWS 결제 보고서 (p. 78)
- Amazon S3 요금
- Amazon S3 FAQ
- S3 Glacier 요금
- S3 Glacier FAQ

비용 할당 S3 버킷 태그 사용

개별 프로젝트나 프로젝트 그룹에 대해 스토리지 비용이나 기타 기준을 추적하려면 비용 할당 태그를 사용하여 Amazon S3 버킷에 레이블을 지정합니다. 비용 할당 태그는 S3 버킷과 연관된 키-값 페어입니다. 비용 할당 태그를 활성화하고 나면 AWS가 이 태그를 사용하여 비용 할당 보고서에서 리소스 비용을 구성합니다. 비용 할당 태그는 버킷에 레이블을 지정할 때만 사용할 수 있습니다. 객체에 레이블을 지정할 때 사용하는 태그에 대한 자세한 내용은 객체 태그 지정 (p. 114) 단원을 참조하십시오.

비용 할당 보고서에는 내 계정에 대해 제품 카테고리 및 AWS Identity and Access Management(IAM) 사용자별 AWS 사용이 나열됩니다. 이 보고서에는 세부 결제 보고서(Amazon S3에 대한 AWS 결제 및 사용 보고서 이해 (p. 81) 참조)와 동일한 행 항목 및 태그 키에 대한 추가 열이 포함되어 있습니다.

AWS는 AWS 생성 태그와 사용자 정의 태그라는 두 가지 유형의 비용 할당 태그를 제공합니다. AWS는 Amazon S3 CreateBucket 이벤트 이후에 AWS에서 생성된 `createdBy` 태그를 정의, 생성 및 적용합니다. 사용자 정의 태그를 정의 및 생성하고 S3 버킷에 적용합니다.

결제 및 비용 관리 콘솔에서 개별적으로 두 유형의 태그를 모두 활성화해야만 결제 보고서에 표시가 됩니다. AWS 생성 태그에 대한 자세한 내용은 AWS 생성 비용 할당 태그를 참조하십시오. 태그 활성화에 대한 자세한 내용은 AWS Billing and Cost Management 사용 설명서의 비용 할당 태그 사용을 참조하십시오.

사용자 정의 비용 할당 태그는 다음과 같은 요소로 구성되어 있습니다.

- 태그 키. 태그 키는 태그의 이름입니다. 예를 들어, project/Trinity 태그에서 프로젝트가 키입니다. 태그 키는 대/소문자 구분 문자열로 유니코드 문자 1 ~ 128자를 포함할 수 있습니다.

- 태그 값. 태그 값은 필수 문자열입니다. 예를 들어, project/Trinity 태그에서 Trinity가 값입니다. 태그 값은 대/소문자 구분 문자열로 유니코드 문자 0 ~ 256자를 포함할 수 있습니다.

사용자 정의 태그에서 허용되는 문자와 기타 제한에 대한 세부 정보는 AWS Billing and Cost Management 사용 설명서의 사용자 정의 태그 제한을 참조하십시오.

각 S3 버킷은 태그 세트를 가지고 있습니다. 태그 세트에는 해당 버킷에 할당된 모든 태그가 포함되어 있습니다. 태그 세트는 최대 50개의 태그를 포함하거나 비어 있을 수 있습니다. 키는 태그 세트 내에서 고유해야

하지만, 태그의 값은 반드시 고유할 필요가 없습니다. 예를 들어, project/Trinity라는 태그 세트와 cost-center/Trinity라는 태그 세트에 동일한 값을 사용할 수 있습니다.

버킷 내에 기존 태그와 동일한 키를 가진 태그를 추가하면 새 값이 기존 값을 덮어씁니다.

AWS는 태그에 의미론적 의미를 적용하지 않습니다. 태그는 엄격히 문자열로 해석됩니다.

태그를 추가하거나, 나열하거나, 편집하거나, 삭제하려면 Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 또는 Amazon S3 API를 사용합니다.

태그 생성에 대한 자세한 내용은 해당 주제를 참조하십시오.

- 콘솔에서 태그를 생성하는 방법은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷에 대한 속성을 보려면? 단원을 참조하십시오.
- Amazon S3 API를 사용하여 태그를 생성하는 방법은 Amazon Simple Storage Service API Reference의 PUT 버킷 태그 지정 단원을 참조하십시오.
- AWS CLI를 사용하여 태그를 생성하는 방법은 AWS CLI Command Reference의 put-bucket-tagging을 참조하십시오.

사용자 정의 태그에 대한 자세한 내용은 AWS Billing and Cost Management 사용 설명서의 사용자 정의 비용 할당 태그를 참조하십시오.

추가 정보

- AWS Billing and Cost Management 사용 설명서의 비용 할당 태그 사용
- Amazon S3에 대한 AWS 결제 및 사용 보고서 이해 (p. 81)
- Amazon S3에 대한 AWS 결제 보고서 (p. 78)

Amazon S3 액세스 포인트를 사용한 데이터 액세스 관리

Amazon S3 액세스 포인트는 S3의 공유 데이터 세트에 대한 대규모 데이터 액세스 관리를 간소화합니다. 액세스 포인트는 `GetObject` 및 `PutObject` 같은 S3 객체 작업을 수행하는 데 사용할 수 있는 버킷에 연결된 네트워크 엔드포인트입니다. 각 액세스 포인트에는 해당 액세스 포인트를 통해 이루어진 모든 요청에 대해 S3가 적용하는 고유한 권한 및 네트워크 제어가 있습니다. 각 액세스 포인트는 기본 버킷에 연결된 버킷 정책과 함께 작동하는 사용자 지정 액세스 포인트 정책을 적용합니다. Virtual Private Cloud(VPC)의 요청만 수락하도록 액세스 포인트를 구성하여 프라이빗 네트워크에 대한 Amazon S3 데이터 액세스를 제한할 수 있습니다. 또한 각 액세스 포인트에 대해 사용자 지정 퍼블릭 액세스 차단 설정을 구성할 수 있습니다.

Note

- 액세스 포인트를 사용하면 객체에 대한 작업만 수행할 수 있습니다. 액세스 포인트를 사용하여 버킷 수정 또는 삭제와 같은 다른 Amazon S3 작업을 수행할 수 없습니다. 액세스 포인트를 지원하는 전체 S3 작업 목록은 S3 작업 및 AWS 서비스와의 액세스 포인트 호환성 (p. 98) 단원을 참조하십시오.
- 액세스 포인트는 전부는 아니지만 일부 AWS 서비스 및 기능을 사용하여 작업합니다. 예를 들어, 교차 리전 복제가 액세스 포인트를 통해 작동하도록 구성할 수 없습니다. S3 액세스 포인트와 호환되는 전체 AWS 서비스 목록은 S3 작업 및 AWS 서비스와의 액세스 포인트 호환성 (p. 98) 단원을 참조하십시오.

이 단원에서는 Amazon S3 액세스 포인트로 작업하는 방법에 대해 설명합니다. 버킷 작업에 대한 자세한 내용은 Amazon S3 버킷을 사용한 작업 (p. 51) 단원을 참조하십시오. 객체 작업에 대한 자세한 내용은 Amazon S3 객체로 작업 (p. 102)을 참조하십시오.

주제

- 액세스 포인트 생성 (p. 90)
- 액세스 포인트 사용 (p. 97)
- 액세스 포인트 규제 및 제한 (p. 100)

액세스 포인트 생성

Amazon S3에서는 액세스 포인트를 생성하고 관리하는 기능을 제공합니다. 기본적으로 각 AWS 계정에 대해 리전당 최대 1,000개의 액세스 포인트를 생성할 수 있습니다. 단일 리전에서 단일 계정에 대해 1,000개 이상의 액세스 포인트가 필요한 경우 서비스 할당량 증가를 요청할 수 있습니다. 서비스 할당량 및 증가 요청에 대한 자세한 내용은 AWS General Reference의 AWS 서비스 할당량을 참조하십시오.

AWS Management 콘솔, AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하여 S3 액세스 포인트를 생성할 수 있습니다. 다음 예제에서는 Java용 AWS CLI 및 AWS SDK를 사용하여 액세스 포인트를 생성하는 방법을 보여줍니다.

AWS Management 콘솔을 사용하여 액세스 포인트를 생성하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 Amazon S3 액세스 포인트 소개를 참조하십시오. REST API를 사용하여 액세스 포인트를 생성하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service API Reference의 CreateAccessPoint를 참조하십시오.

Note

사용자가 액세스 포인트를 사용할 수 있도록 액세스 포인트 이름을 공개할 수 있으므로 액세스 포인트 이름에 중요한 정보를 포함하지 않는 것이 좋습니다.

Amazon S3 액세스 포인트 이름 지정 규칙

액세스 포인트 이름은 다음과 같은 조건을 만족해야 합니다.

- 단일 AWS 계정 및 리전 내에서 고유해야 함
- DNS 이름 지정 제한을 준수해야 함
- 숫자 또는 소문자로 시작해야 함
- 3~50자 이내여야 함
- 대시로 시작하거나 끝낼 수 없음
- 밑줄, 대문자 또는 마침표를 포함할 수 없음

Example

예: 액세스 포인트 생성

다음 예제에서는 계정 `123456789012`의 버킷 `example-bucket`에 이름이 `example-ap`인 액세스 포인트를 생성하는 방법을 보여줍니다. 액세스 포인트를 만들려면 액세스 포인트 이름, 액세스 포인트를 연결할 버킷의 이름, 버킷을 소유한 AWS 계정의 계정 ID를 지정하여 Amazon S3에 요청을 보냅니다. 이름 지정 규칙에 대한 자세한 내용은 the section called “Amazon S3 액세스 포인트 이름 지정 규칙” (p. 91) 단원을 참조하십시오.

AWS CLI

```
aws s3control create-access-point --name example-ap --account-id 123456789012 --bucket
 example-bucket
```

Virtual Private Cloud(VPC)로 제한된 액세스 포인트 생성

액세스 포인트를 생성할 때 인터넷에서 액세스 포인트를 액세스할 수 있도록 선택하거나 해당 액세스 포인트를 통해 이루어진 모든 요청이 특정 Virtual Private Cloud(VPC)에서 시작되도록 지정할 수 있습니다. 인터넷에서 액세스할 수 있는 액세스 포인트의 네트워크 오리진은 `Internet`입니다. 액세스 포인트, 기본 버킷 및 관련 리소스(예: 요청된 객체)에 대한 기타 액세스 제한에 따라 인터넷 어디에서나 사용할 수 있습니다. 지정된 VPC에서만 액세스할 수 있는 액세스 포인트의 네트워크 오리진은 `VPC`이고, Amazon S3는 해당 VPC에서 시작되지 않은 액세스 포인트에 대한 모든 요청을 거부합니다.

> **Important**
>
> 액세스 포인트를 생성할 때만 액세스 포인트의 네트워크 오리진을 지정할 수 있습니다. 액세스 포인트를 생성한 후에는 네트워크 오리진을 변경할 수 없습니다.

액세스 포인트를 VPC 전용 액세스로 제한하려면 액세스 포인트 생성 요청에 `VpcConfiguration` 파라미터를 포함합니다. `VpcConfiguration` 파라미터에서 액세스 포인트를 사용할 수 있도록 할 VPC ID를 지정합니다. Amazon S3는 액세스 포인트를 통해 이루어진 요청이 해당 VPC에서 시작되지 않는 한 거부합니다.

AWS CLI, AWS SDK 또는 REST API를 사용하여 액세스 포인트의 네트워크 오리진을 검색할 수 있습니다. 액세스 포인트에 VPC 구성이 지정되어 있는 경우 네트워크 오리진은 `VPC`입니다. 그렇지 않으면 액세스 포인트의 네트워크 오리진은 `Internet`입니다.

Example

예: VPC 액세스로 제한된 액세스 포인트 생성

다음 예제에서는 VPC `vpc-1a2b3c`에서만 액세스하도록 허용하는 계정 `123456789012`의 버킷 `example-bucket`에 이름이 `example-vpc-ap`인 액세스 포인트를 생성합니다. 그런 다음 새 액세스 포인트의 네트워크 오리진이 `VPC`인지 확인합니다.

AWS CLI

```
aws s3control create-access-point --name example-vpc-ap --account-id 123456789012 --
bucket example-bucket --vpc-configuration VpcId=vpc-1a2b3c
```

```
aws s3control get-access-point --name example-vpc-ap --account-id 123456789012

{
    "Name": "example-vpc-ap",
    "Bucket": "example-bucket",
    "NetworkOrigin": "VPC",
    "VpcConfiguration": {
        "VpcId": "vpc-1a2b3c"
    },
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2019-11-27T00:00:00Z"
}
```

VPC와 함께 액세스 포인트를 사용하려면 VPC 엔드포인트에 대한 액세스 정책을 수정해야 합니다. VPC 엔드포인트를 사용하면 VPC에서 Amazon S3로 트래픽이 흐를 수 있습니다. VPC 내의 리소스가 S3와 상호 작용할 수 있는 방법을 제어하는 액세스 제어 정책이 있습니다. VPC에서 S3로의 요청은 VPC 엔드포인트 정책이 액세스 포인트와 기본 버킷 모두에 대한 액세스 권한을 부여하는 경우에만 액세스 포인트를 통해 성공합니다.

다음 예제 정책 문은 이름이 `example-bucket`인 버킷과 이름이 `example-vpc-ap`인 액세스 포인트에 `GetObject` 호출을 허용하도록 VPC 엔드포인트를 구성합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Principal": "*",
        "Action": [
            "s3:GetObject"
        ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:s3:::example-bucket/*",
            "arn:aws:s3:us-west-2:123456789012:accesspoint/example-vpc-ap/object/*"
        ]
    }]
}
```

Note

이 예제의 `"Resource"` 선언은 Amazon 리소스 이름(ARN)을 사용하여 액세스 포인트를 지정합니다. 액세스 포인트 ARN에 대한 자세한 내용은 액세스 포인트 사용 (p. 97) 단원을 참조하십시오.

VPC 엔드포인트 정책에 대한 자세한 내용은 Amazon Virtual Private Cloud 사용 설명서의 Amazon S3에 대한 엔드포인트 정책 사용을 참조하십시오.

액세스 포인트에 대한 퍼블릭 액세스 관리

Amazon S3 액세스 포인트는 각 액세스 포인트에 대해 독립적인 퍼블릭 액세스 차단 설정을 지원합니다. 액세스 포인트를 생성할 때 해당 액세스 포인트에 적용되는 퍼블릭 액세스 차단 설정을 지정할 수 있습니다. 액세스 포인트를 통해 이루어진 모든 요청에 대해 Amazon S3는 해당 액세스 포인트, 기본 버킷 및 버킷 소유자 계정에 대한 퍼블릭 액세스 차단 설정을 평가합니다. 이러한 설정 중 하나가 요청이 차단되어야 함을 나타내는 경우, Amazon S3는 요청을 거부합니다.

S3 퍼블릭 액세스 차단 기능에 대한 자세한 내용은 Amazon S3 퍼블릭 액세스 차단 사용 (p. 450) 단원을 참조하십시오.

Important

- 액세스 포인트에는 기본적으로 모든 퍼블릭 액세스 차단 설정이 활성화되어 있습니다. 액세스 포인트에 적용하지 않으려는 모든 설정을 명시적으로 비활성화해야 합니다.
- Amazon S3에서는 현재 액세스 포인트가 생성된 후 액세스 포인트의 퍼블릭 액세스 차단 설정을 변경하도록 지원하지 않습니다.

Example

예: 사용자 지정 퍼블릭 액세스 차단 설정을 사용하여 액세스 포인트 생성

이 예제에서는 기본값이 아닌 퍼블릭 액세스 차단 설정을 사용하여 계정 `123456789012`의 버킷 `example-bucket`에 이름이 `example-ap`인 액세스 포인트를 생성합니다. 그런 다음 새 액세스 포인트의 구성을 검색하여 퍼블릭 액세스 차단 설정을 확인합니다.

AWS CLI

```
aws s3control create-access-point --name example-ap --account-id
 123456789012 --bucket example-bucket --public-access-block-configuration
 BlockPublicAcls=false,IgnorePublicAcls=false,BlockPublicPolicy=true,RestrictPublicBuckets=true
```

```
aws s3control get-access-point --name example-ap --account-id 123456789012

{
    "Name": "example-ap",
    "Bucket": "example-bucket",
    "NetworkOrigin": "Internet",
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": false,
        "IgnorePublicAcls": false,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2019-11-27T00:00:00Z"
}
```

액세스 포인트를 사용하도록 IAM 정책 구성

Amazon S3 액세스 포인트는 리소스, 사용자 또는 기타 조건별로 액세스 포인트 사용을 제어할 수 있는 AWS Identity and Access Management(IAM) 리소스 정책을 지원합니다. 애플리케이션 또는 사용자가 액세스 포인트를 통해 객체에 액세스할 수 있으려면 액세스 포인트와 기본 버킷 모두에서 요청을 허용해야 합니다.

Important

버킷에 S3 액세스 포인트를 추가해도 기존 버킷 이름이나 ARN을 통해 액세스할 때 버킷의 동작은 변경되지 않습니다. 버킷에 대한 모든 기존 작업은 이전과 같이 계속 작동합니다. 액세스 포인트 정책에 포함시키는 제한은 해당 액세스 포인트를 통해 이루어진 요청에만 적용됩니다.

조건 키

S3 액세스 포인트는 리소스에 대한 액세스를 제어하는 IAM 정책에서 사용할 수 있는 세 가지 새로운 조건 키를 도입합니다.

s3:DataAccessPointArn

액세스 포인트 ARN과 일치시키는 데 사용할 수 있는 문자열입니다. 다음 예제에서는 리전 `us-west-2`의 AWS 계정 `123456789012`에 대한 모든 액세스 포인트와 일치합니다.

```
"Condition" : {
    "StringLike": {
        "s3:DataAccessPointArn": "arn:aws:s3:us-west-2:123456789012:accesspoint/*"
    }
}
```

s3:DataAccessPointAccount

액세스 포인트 소유자의 계정 ID와 일치시키는 데 사용할 수 있는 문자열 연산자입니다. 다음 예제에서는 AWS 계정 `123456789012`에서 소유한 모든 액세스 포인트와 일치합니다.

```
"Condition" : {
    "StringEquals": {
        "s3:DataAccessPointAccount": "123456789012"
    }
}
```

s3:AccessPointNetworkOrigin

네트워크 오리진 `Internet` 또는 `VPC`와 일치시키는 데 사용할 수 있는 문자열 연산자입니다. 다음 예제에서는 오리진이 VPC인 액세스 포인트와만 일치합니다.

```
"Condition" : {
    "StringEquals": {
        "s3:AccessPointNetworkOrigin": "VPC"
    }
}
```

Amazon S3에서 조건 키 사용에 대한 자세한 내용은 Amazon S3에 사용되는 작업, 리소스 및 조건 키 (p. 356) 단원을 참조하십시오.

액세스 포인트에 액세스 제어 위임

버킷에 대한 액세스 제어를 버킷의 액세스 포인트에 위임할 수 있습니다. 다음 예제 버킷 정책은 버킷 소유자의 계정이 소유한 모든 액세스 포인트에 대한 전체 액세스를 허용합니다. 따라서 이 버킷에 대한 모든 액세스는 해당 액세스 포인트에 연결된 정책에 의해 제어됩니다. 버킷에 직접 액세스할 필요가 없는 모든 사용 사례에 대해 이 방법으로 버킷을 구성하는 것이 좋습니다.

Example 액세스 포인트에 액세스 제어를 위임하는 버킷 정책

```
{
    "Version": "2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS": "*" },
        "Action" : "*",
```

```
        "Resource" : [ "Bucket ARN", "Bucket ARN/*"],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointAccount" : "Bucket owner's account ID" }
        }
    }]
}
```

액세스 포인트 정책 예제

다음 예제에서는 액세스 포인트를 통해 이루어진 요청을 제어하기 위해 IAM 정책을 만드는 방법을 보여줍니다.

Example 액세스 포인트 정책 권한 부여

다음 액세스 포인트 정책은 계정 `123456789012`의 IAM 사용자 `Alice`에게 액세스 포인트 `my-access-point`를 통해 접두사가 `Alice/`가 `GET` 및 `PUT` 객체에 대한 권한을 부여합니다.

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Alice"
        },
        "Action": ["s3:GetObject", "s3:PutObject"],
        "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/my-access-point/object/
Alice/*"
    }]
}
```

> **Note**
>
> 액세스 포인트 정책이 `Alice`에 대한 액세스 권한을 효과적으로 부여하려면 기본 버킷에서도 `Alice`에 대해 동일한 액세스를 허용해야 합니다. 다음 버킷 정책은 필요한 권한을 Alice에게 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Alice"
        },
        "Action": ["s3:GetObject", "s3:PutObject"],
        "Resource": "arn:aws:s3:::example-bucket/Alice/*"
    }]
}
```

Example 태그 조건이 있는 액세스 포인트 정책

다음 액세스 포인트 정책은 계정 `123456789012`의 IAM 사용자 `Bob`에게 태그 키 `data`가 `finance` 값으로 설정된 액세스 포인트 `my-access-point`를 통해 `GET` 및 `PUT` 객체에 대한 권한을 부여합니다.

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

```
        "Principal" : {
            "AWS": "arn:aws:iam::123456789012:user/Bob"
        },
        "Action":["s3:GetObject", "s3:PutObject"],
        "Resource" : "arn:aws:s3:us-west-2:123456789012:accesspoint/my-access-point/object/
*",
        "Condition" : {
            "StringEquals": {
                "s3:ExistingObjectTag/data": "finance"
            }
        }
    }]
}
```

Note

액세스 포인트 정책이 `Bob`에 대한 액세스 권한을 효과적으로 부여하려면 기본 버킷에서도 `Bob`에 대해 동일한 액세스를 허용해야 합니다. 다음 버킷 정책은 필요한 권한을 `Bob`에게 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Bob"
        },
        "Action": ["s3:GetObject", "s3:PutObject"],
        "Resource": "arn:aws:s3:::example-bucket/*",
        "Condition" : {
            "StringEquals": {
                "s3:ExistingObjectTag/data": "finance"
            }
        }
    }]
}
```

Example 서비스 제어 정책

다음 서비스 제어 정책에서는 VPC 네트워크 오리진을 사용하여 모든 새 액세스 포인트를 생성해야 합니다. 이 정책을 적용하면 조직의 사용자가 인터넷에서 액세스할 수 있는 새 액세스 포인트를 생성할 수 없습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Deny",
        "Action": "s3:CreateAccessPoint",
        "Resource": "*",
        "Condition": {
            "StringNotEquals": {
                "s3:AccessPointNetworkOrigin": "VPC"
            }
        }
    }]
}
```

Example S3 작업을 VPC 네트워크 오리진으로 제한하는 버킷 정책

다음 버킷 정책은 버킷 `examplebucket`에 대한 모든 S3 객체 작업에 대한 액세스를 네트워크 오리진이 VPC인 액세스 포인트로 제한합니다.

Important

이 예제와 같은 설명을 사용하기 전에 교차 리전 복제와 같이 액세스 포인트에서 지원하지 않는 기능을 사용할 필요가 없습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:BypassGovernanceRetention",
                "s3:DeleteObject",
                "s3:DeleteObjectTagging",
                "s3:DeleteObjectVersion",
                "s3:DeleteObjectVersionTagging",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectLegalHold",
                "s3:GetObjectRetention",
                "s3:GetObjectTagging",
                "s3:GetObjectVersion",
                "s3:GetObjectVersionAcl",
                "s3:GetObjectVersionTagging",
                "s3:ListMultipartUploadParts",
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectLegalHold",
                "s3:PutObjectRetention",
                "s3:PutObjectTagging",
                "s3:PutObjectVersionAcl",
                "s3:PutObjectVersionTagging",
                "s3:RestoreObject"
            ],
            "Resource": "arn:aws:s3:::examplebucket/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:AccessPointNetworkOrigin": "VPC"
                }
            }
        }
    ]
}
```

액세스 포인트 사용

AWS Management 콘솔, AWS CLI, AWS SDK 또는 S3 REST API를 사용하는 액세스 포인트로 Amazon S3 버킷의 객체에 액세스할 수 있습니다.

액세스 포인트에 Amazon 리소스 이름(ARN)이 있습니다. 액세스 포인트 ARN은 버킷 ARN과 비슷하지만 명시적으로 입력되고 액세스 포인트의 리전 및 액세스 포인트 소유자의 AWS 계정 ID를 인코딩합니다. ARN에 대한 자세한 내용은 AWS General Reference의 Amazon 리소스 이름(ARN)을 참조하십시오.

액세스 포인트 ARN은 `arn:aws:s3:region:account-id:accesspoint/resource` 형식을 사용합니다. 예:

- arn:aws:s3:us-west-2:123456789012:accesspoint/test는 리전 `us-west-2`의 계정 `123456789012`에서 소유한 이름이 `test`인 액세스 포인트를 나타냅니다.

- arn:aws:s3:us-west-2:123456789012:accesspoint/*는 리전 `us-west-2`의 계정 `123456789012`에 있는 모든 액세스 포인트를 나타냅니다.

액세스 포인트를 통해 액세스하는 객체의 ARN은 `arn:aws:s3:`*`region`*`:`*`account-id`*`:accesspoint/`*`access-point-name`*`/object/`*`resource` 형식을 사용합니다. 예:

- arn:aws:s3:us-west-2:123456789012:accesspoint/test/object/unit-01은 리전 `us-west-2`의 계정 `123456789012`에서 소유한 이름이 `test`인 액세스 포인트를 통해 액세스한 객체 `unit-01`을 나타냅니다.
- arn:aws:s3:us-west-2:123456789012:accesspoint/test/object/*는 리전 `us-west-2`의 계정 `123456789012`에서 액세스 포인트 `test`에 대한 모든 객체를 나타냅니다.
- arn:aws:s3:us-west-2:123456789012:accesspoint/test/object/unit-01/finance/*는 리전 `us-west-2`의 계정 `123456789012`에서 액세스 포인트 `test`에 대한 접두사 `unit-01/finance/`이 있는 모든 객체를 나타냅니다.

S3 작업 및 AWS 서비스와의 액세스 포인트 호환성

S3 액세스 포인트는 S3 작업 및 기타 AWS 서비스의 하위 집합과 호환됩니다. 다음 단원에는 호환되는 서비스 및 S3 작업이 나열되어 있습니다.

AWS 서비스

AWS CloudFormation과 함께 S3 액세스 포인트를 사용할 수 있습니다.

AWS CloudFormation에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS CloudFormation이란 무엇입니까?를 참조하십시오.

S3 작업

액세스 포인트를 사용하여 다음 Amazon S3 API 하위 집합을 사용하는 버킷에 액세스할 수 있습니다.

- `AbortMultipartUpload`
- `CompleteMultipartUpload`
- `CreateMultipartUpload`
- `DeleteObject`
- `DeleteObjectTagging`
- `GetObject`
- `GetObjectAcl`
- `GetObjectLegalHold`
- `GetObjectRetention`
- `GetObjectTagging`
- `HeadObject`
- `ListMultipartUploads`
- `ListObjectsV2`
- `ListParts`
- `PutObject`
- `PutObjectLegalHold`
- `PutObjectRetention`
- `PutObjectAcl`
- `PutObjectTagging`
- `RestoreObject`

- UploadPart

모니터링 및 로깅

Amazon S3는 액세스 포인트를 통해 이루어진 요청과 액세스 포인트를 관리하는 API에 대한 요청(예: CreateAccessPoint 및 GetAccessPointPolicy)을 로그로 기록합니다.

액세스 포인트를 통해 이루어진 Amazon S3에 대한 요청은 액세스 포인트의 호스트 이름과 함께 S3 서버 액세스 로그 및 AWS CloudTrail 로그에 표시됩니다. 액세스 포인트의 호스트 이름은 *access_point_name*-*account_id*.s3-accesspoint.*Region*.amazonaws.com 형식을 사용합니다. 예를 들어 다음과 같은 버킷 및 액세스 포인트 구성이 있다고 가정합니다.

- us-west-2 리전에서 객체 my-image.jpg가 포함된 이름이 my-bucket인 버킷
- my-bucket과 연결된 이름이 my-bucket-ap인 액세스 포인트
- AWS 계정 ID는 123456789012입니다.

버킷을 통해 my-image.jpg를 직접 검색하기 위해 이루어진 요청은 호스트 이름이 my-bucket.s3.us-west-2.amazonaws.com인 로그에 표시됩니다. 대신 액세스 포인트를 통해 요청한 경우 Amazon S3는 동일한 객체를 검색하지만 호스트 이름 my-bucket-ap-123456789012.s3-accesspoint.us-west-2.amazonaws.com을 사용하여 요청을 로그에 기록합니다.

S3 서버 액세스 로그에 대한 자세한 내용은 Amazon S3 서버 액세스 로깅 (p. 705) 단원을 참조하십시오. AWS CloudTrail에 대한 자세한 내용은 AWS CloudTrail User Guide의 AWS CloudTrail이란 무엇입니까?를 참조하십시오.

> **Note**
>
> S3 액세스 포인트는 현재 Amazon CloudWatch 지표와 호환되지 않습니다.

예제

Example

예: 액세스 포인트를 통해 객체 요청

다음 예제에서는 리전 us-west-2의 계정 ID 123456789012에서 소유한 액세스 포인트 prod를 통해 객체 my-image.jpg를 요청하는 방법을 보여주며 다운로드한 파일을 download.jpg로 저장합니다.

AWS CLI

```
aws s3api get-object --key my-image.jpg --bucket arn:aws:s3:us-
west-2:123456789012:accesspoint/prod download.jpg
```

Example

예: 액세스 포인트를 통해 객체 업로드

다음 예제에서는 리전 us-west-2의 계정 ID 123456789012에서 소유한 액세스 포인트 prod를 통해 객체 my-image.jpg를 업로드하는 방법을 보여줍니다.

AWS CLI

```
aws s3api put-object --bucket arn:aws:s3:us-west-2:123456789012:accesspoint/prod --key
 my-image.jpg --body my-image.jpg
```

Example

예: 액세스 포인트를 통해 객체 삭제

다음 예제에서는 리전 us-west-2의 계정 ID 123456789012에서 소유한 액세스 포인트 prod를 통해 객체 my-image.jpg를 삭제하는 방법을 보여줍니다.

AWS CLI

```
aws s3api delete-object --bucket arn:aws:s3:us-west-2:123456789012:accesspoint/prod --
key my-image.jpg
```

Example

예: 액세스 포인트를 통해 객체 나열

다음 예제에서는 리전 us-west-2의 계정 ID 123456789012에서 소유한 액세스 포인트 prod를 통해 객체를 나열하는 방법을 보여줍니다.

AWS CLI

```
aws s3api list-objects-v2 --bucket arn:aws:s3:us-west-2:123456789012:accesspoint/prod
```

Example

예: 액세스 포인트를 통해 객체에 태그 세트 추가

다음 예제에서는 리전 us-west-2의 계정 ID 123456789012에서 소유한 액세스 포인트 prod를 통해 기존 객체 my-image.jpg에 태그 세트를 추가하는 방법을 보여줍니다.

AWS CLI

```
aws s3api put-object-tagging --bucket arn:aws:s3:us-west-2:123456789012:accesspoint/
prod --key my-image.jpg --tagging TagSet=[{Key="finance",Value="true"}]
```

Example

예: ACL을 사용하여 액세스 포인트를 통해 액세스 권한 부여

다음 예제에서는 리전 us-west-2의 계정 ID 123456789012에서 소유한 액세스 포인트 prod를 통해 기존 객체 my-image.jpg에 ACL을 적용하는 방법을 보여줍니다.

AWS CLI

```
aws s3api put-object-acl --bucket arn:aws:s3:us-west-2:123456789012:accesspoint/prod --
key my-image.jpg --acl private
```

액세스 포인트 규제 및 제한

Amazon S3 액세스 포인트에는 다음과 같은 규제 및 제한이 있습니다.

- 소유한 버킷에 대해서만 액세스 포인트를 생성할 수 있습니다.

- 각 액세스 포인트는 정확히 하나의 버킷과 연결되어 있으며, 이 버킷은 액세스 포인트를 생성할 때 지정해야 합니다. 액세스 포인트를 생성한 후에는 다른 버킷과 연결할 수 없습니다. 그러나 액세스 포인트를 삭제한 다음 다른 버킷과 연결된 동일한 이름을 가진 다른 액세스 포인트를 생성할 수 있습니다.
- 액세스 포인트를 생성한 후에는 해당 Virtual Private Cloud(VPC) 구성을 변경할 수 없습니다.
- 액세스 포인트 정책은 크기가 20KB로 제한됩니다.
- AWS 계정에 대해 리전당 최대 1,000개의 액세스 포인트를 생성할 수 있습니다. 단일 리전에서 단일 계정에 대해 1,000개 이상의 액세스 포인트가 필요한 경우 서비스 할당량 증가를 요청할 수 있습니다. 서비스 할당량 및 증가 요청에 대한 자세한 내용은 AWS General Reference의 AWS 서비스 할당량을 참조하십시오.
- 액세스 포인트를 S3 교차 리전 복제의 대상으로 사용할 수 없습니다. 복제에 대한 자세한 내용은 복제 (p. 607) 단원을 참조하십시오.
- 가상 호스트 스타일의 URL을 사용해야만 액세스 포인트의 주소를 지정할 수 있습니다. 가상 호스트 스타일 주소 지정에 대한 자세한 내용은 버킷 액세스를 참조하십시오.
- 액세스 포인트 기능을 제어하는 API(예: `PutAccessPoint` 및 `GetAccessPointPolicy`)는 교차 계정 호출을 지원하지 않습니다.
- REST API를 사용하여 액세스 포인트에 요청할 때는 AWS 서명 버전 4를 사용해야 합니다. 요청 인증에 대한 자세한 내용은 Amazon Simple Storage Service API Reference의 요청 인증(AWS 서명 버전 4)을 참조하십시오.
- 액세스 포인트는 HTTPS를 통한 액세스만 지원합니다.
- 액세스 포인트는 익명 액세스를 지원하지 않습니다.

Amazon S3 객체로 작업

Amazon S3는 최대한 많은 객체를 저장할 수 있도록 설계된 단순한 키-값의 스토어입니다. 이러한 객체는 하나 이상의 버킷에 저장되며 각 객체의 크기는 5TB까지 가능합니다. 객체는 다음과 같은 요소로 구성됩니다.

- 키 – 객체에 지정한 이름입니다. 객체 키를 사용하여 객체를 검색합니다.

  자세한 내용은 객체 키와 메타데이터 (p. 102)를 참조하십시오.
- 버전 ID – 버킷 내에서 키와 버전 ID를 사용하여 각 객체를 고유하게 식별할 수 있습니다.

  버전 ID는 버킷에 객체를 추가할 때 Amazon S3가 생성하는 문자열입니다. 자세한 내용은 객체 버전 관리 (p. 112) 단원을 참조하십시오.
- 값 – 저장하는 콘텐츠입니다.

  객체 값은 임의의 바이트 시퀀스입니다. 객체 크기는 0TB 이상 5TB까지 가능합니다. 자세한 내용은 객체 업로드 (p. 172) 단원을 참조하십시오.
- 메타데이터 – 객체 관련 정보를 저장하기 위한 이름-값 페어의 집합입니다.

  Amazon S3의 객체에 사용자 정의 메타데이터라고 하는 메타데이터를 지정할 수 있습니다. Amazon S3는 이러한 객체의 관리에 사용되는 시스템 메타데이터를 객체에 지정합니다. 자세한 내용은 객체 키와 메타데이터 (p. 102)를 참조하십시오.
- 하위 리소스 – Amazon S3는 하위 리소스 메커니즘을 사용하여 객체 관련 추가 정보를 저장합니다.

  하위 리소스는 객체에 종속되므로 항상 객체, 버킷 등의 다른 항목과 연결됩니다. 자세한 내용은 객체 하위 리소스 (p. 112) 단원을 참조하십시오.
- 액세스 제어 정보 – Amazon S3에 저장한 객체에 대한 액세스를 제어할 수 있습니다..

  Amazon S3은 ACL(액세스 제어 목록), 버킷 정책 등의 리소스 기반 액세스 제어와 사용자 기반 액세스 제어를 모두 지원합니다. 자세한 내용은 Amazon S3의 Identity and Access Management (p. 299) 단원을 참조하십시오.

객체 관련 작업에 대한 자세한 내용은 다음 단원을 참조하십시오. Amazon S3 리소스(예: 버킷 및 객체)는 기본적으로 비공개입니다. 명시적으로 권한을 부여해야 다른 사용자가 이러한 리소스에 액세스할 수 있습니다. 예를 들어, 웹 사이트의 Amazon S3 버킷에 저장된 동영상이나 사진을 공유하고 싶으면 객체를 공개하거나 웹 사이트의 미리 서명된 URL을 사용해야 합니다. 객체 공유에 대한 자세한 내용은 다른 사용자와 객체 공유 (p. 170) 단원을 참조하십시오.

주제

- 객체 키와 메타데이터 (p. 102)
- Amazon S3 스토리지 클래스 (p. 107)
- 객체 하위 리소스 (p. 112)
- 객체 버전 관리 (p. 112)
- 객체 태그 지정 (p. 114)
- 객체 수명 주기 관리 (p. 123)
- CORS(Cross-Origin Resource Sharing) (p. 154)
- 객체 관련 작업 (p. 163)

객체 키와 메타데이터

각 Amazon S3 객체는 데이터와 키, 메타데이터로 구성됩니다. 객체 키(또는 키 이름)는 버킷 내 객체를 고유하게 식별합니다. 객체 메타데이터는 이름-값 페어의 집합입니다. 객체를 업로드할 때 객체 메타데이터를 설

정할 수 있습니다. 객체를 업로드한 후에는 객체 메타데이터를 변경할 수 없습니다. 객체 메타데이터를 수정할 수 있는 유일한 방법은 객체 복사본을 만든 후 메타데이터를 설정하는 것입니다.

주제

- 객체 키 (p. 103)
- 객체 메타데이터 (p. 105)

객체 키

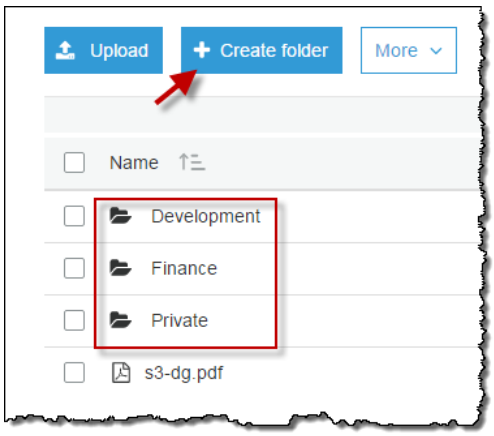
객체를 만들 때 버킷 내 각 객체의 고유한 식별자로 키 이름을 지정합니다. 예를 들어, Amazon S3 콘솔에서 버킷을 강조 표시하면 버킷 내 객체의 목록이 표시됩니다. 이러한 이름이 객체 키입니다. 키 이름은 최대 1,024바이트 길이와 UTF-8 인코딩을 사용하는 유니코드 문자의 시퀀스입니다.

Amazon S3 데이터 모델은 플랫 구조입니다. 버킷을 만들면 이 버킷에 객체가 저장됩니다. 하위 버킷 또는 하위 폴더의 계층 구조는 없습니다. 그러나 Amazon S3 콘솔과 같이 키 이름 접두사 및 구분 기호를 사용하여 논리적인 계층 구조를 유추할 수 있습니다. Amazon S3 콘솔은 폴더 개념을 지원합니다.

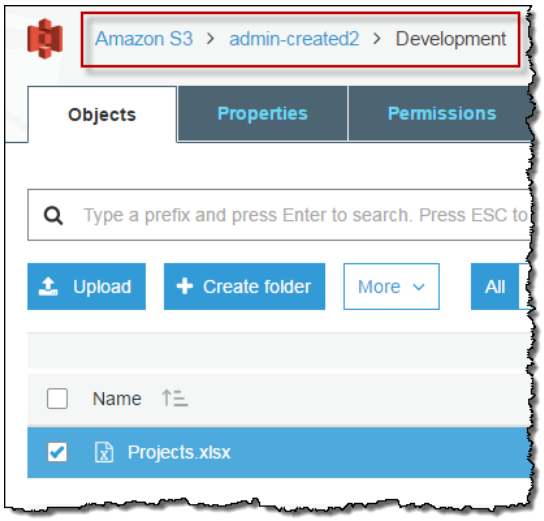
버킷(`admin-created`)에 다음과 같은 객체 키를 가진 4개의 객체가 있다고 가정해 보겠습니다.

```
Development/Projects.xls

Finance/statement1.pdf

Private/taxdocument.pdf

s3-dg.pdf
```

콘솔은 키 이름 접두사(`Development/`, `Finance/` 및 `Private/`) 및 구분 기호('/')를 사용하여 다음과 같은 폴더 구조를 표현합니다.



`s3-dg.pdf` 키에는 접두사가 없으므로 이 객체는 버킷의 루트 수준에 표시됩니다. `Development/` 폴더를 열면 그 안에 `Projects.xlsx` 객체가 표시됩니다.



Note

Amazon S3는 버킷과 객체를 지원하며 계층 구조가 없습니다. 그러나 객체 키 이름에 접두사와 구분 기호를 사용하면 Amazon S3 콘솔과 AWS SDK에서 계층 구조를 추론하고 폴더 개념을 도입할 수 있습니다.

객체 키 명명 지침

객체 키 이름에 임의의 UTF-8 문자를 사용할 수 있습니다. 하지만 특정 문자는 키 이름에 사용하면 일부 애플리케이션 또는 프로토콜에 문제가 발생할 수도 있습니다. 다음 지침에 따르면 DNS, 웹 안전 문자, XML 파싱 프로그램 및 기타 API의 호환성을 극대화할 수 있습니다.

사용 가능 문자

다음 문자 집합은 일반적으로 키 이름으로 사용해도 문제가 되지 않습니다.

영숫자	• 0~9 • a-z • A-Z
특수 문자	• ! • - • _ • . • * • ' • (•)

다음은 유효한 객체 키 이름의 예입니다.

- `4my-organization`
- `my.great_photos-2014/jan/myvacation.jpg`
- `videos/2014/birthday/video1.wmv`

> **Important**
>
> 객체 키 이름이 마침표(.) 또는 2개의 마침표(..)로 끝나는 경우 Amazon S3 콘솔을 사용하여 객체를 다운로드할 수 없습니다. 키 이름이 "." 또는 ".."로 끝나는 객체를 다운로드하려면 AWS Command Line Interface(AWS CLI), AWS SDK 또는 REST API를 사용해야 합니다.

특별한 처리가 필요한 문자

키 이름에서 다음 문자는 추가 코드 처리가 필요할 수도 있으며 URL 인코딩되거나 HEX로 참조해야 할 수 있습니다. 이러한 문자 중 일부는 인쇄가 되지 않으며 브라우저에서 처리하지 못할 수 있으므로 특별한 처리가 필요합니다.

- 앰퍼샌드("&")
- 달러("$")
- ASCII 문자 범위 16진수 00~1F(10진수: 0~31) 및 7F(10진수: 127)
- 'At' 기호("@")
- 등호("=")
- 세미콜론(";")
- 콜론(":")
- 더하기("+")
- 공백 – 경우에 따라 중요한 의미가 있는 공백의 순서가 사라질 수 있음(특히 공백이 여러 개 있는 경우)
- 쉼표(",")
- 물음표("?")

피해야 하는 문자

모든 애플리케이션 간 일관성을 유지하기 위해 상당한 특수 처리가 필요하므로 다음과 같은 문자는 키 이름에서 사용하지 않는 것이 좋습니다.

- 백슬래시("\")
- 왼쪽 중괄호("{")
- 인쇄되지 않는 ASCII 문자(128~255 10진수)
- 캐럿("^")
- 오른쪽 중괄호("}")
- 백분율 문자("%")
- 억음 악센트 기호("`")
- 오른쪽 대괄호("]")
- 인용 부호
- '보다 큼' 기호(">")
- 왼쪽 대괄호("[")
- 물결표("~")
- '보다 작은' 기호("<")
- '파운드' 문자("#")
- 세로 막대/파이프("|")

객체 메타데이터

시스템 메타데이터 및 사용자 정의 메타데이터라는 두 종류의 메타데이터가 있습니다.

시스템 정의 객체 메타데이터

버킷에 저장된 각 객체에 대해 Amazon S3는 시스템 메타데이터의 조합을 유지합니다. Amazon S3는 필요할 경우 이 시스템 메타데이터를 처리합니다. 예를 들어 Amazon S3는 객체 생성일과 크기 메타데이터를 유지하며 객체 관리의 일환으로 이 정보를 사용합니다.

시스템 메타데이터에는 다음 2가지 카테고리가 있습니다.

1. 객체 생성 날짜와 같은 메타데이터는 시스템에서 제어하며 Amazon S3만 값을 수정할 수 있습니다.
2. 그 외, 객체에 대해 구성된 스토리지 클래스, 객체의 서버 측 암호화 사용 여부와 같은 시스템 메타데이터는 사용자가 값을 제어할 수 있는 시스템 메타데이터의 예입니다. 웹 사이트로 구성된 버킷에서는 다른 페이지 또는 외부 URL로 페이지 요청을 리디렉션해야 할 때가 있습니다. 이 경우 웹 페이지는 버킷의 객체가 됩니다. Amazon S3는 페이지 리디렉션 값을 시스템 메타데이터로 저장하며, 그 값은 사용자가 제어합니다.

   객체를 만들 때 이러한 시스템 메타데이터 항목의 값을 구성하고, 언제든지 필요할 때마다 값을 업데이트할 수 있습니다. 스토리지 클래스에 대한 자세한 정보는 Amazon S3 스토리지 클래스 (p. 107)를 참조하십시오.

   서버 측 암호화에 대한 자세한 정보는 암호화를 사용하여 데이터 보호 (p. 262) 단원을 참조하십시오.

다음 표에 시스템 정의 메타데이터의 목록과 사용자의 업데이트 여부가 정리되어 있습니다.

이름	설명	사용자의 값 수정 여부
날짜	현재 날짜 및 시간.	아니요
Content-Length	객체 크기(바이트 단위)	아니요
Content-Type	객체 유형	예
Last-Modified	객체 생성일 또는 최종 수정일 중 최근 날짜	아니요
Content-MD5	객체의 base64 인코딩 128비트 MD5 DIGEST	아니요
x-amz-server-side-encryption	객체에 대한 서버 측 암호화 활성화 여부 및 해당 암호화의 AWS Key Management Service(AWS KMS) 또는 Amazon S3 관리형 암호화(SSE-S3) 여부를 나타냅니다. 자세한 내용은 서버 측 암호화를 사용하여 데이터 보호 (p. 263) 단원을 참조하십시오.	예
x-amz-version-id	객체 버전. 버전 관리를 사용하는 버킷의 경우 Amazon S3는 버킷에 추가된 객체에 버전 번호를 지정합니다. 자세한 내용은 버전 관리 사용 (p. 468) 단원을 참조하십시오.	아니요
x-amz-delete-marker	버전 관리를 사용하는 버킷의 경우 이 마커는 객체가 삭제 마커인지 여부를 나타냅니다.	아니요
x-amz-storage-class	객체 저장에 사용된 스토리지 클래스. 자세한 내용은 Amazon S3 스토리지 클래스 (p. 107) 단원을 참조하십시오.	예
x-amz-website-redirect-location	관련 객체에 대한 요청을 동일한 버킷의 다른 객체 또는 외부 URL로 리디렉션합니다. 자세한 내용은 (선택 사항) 웹 페이지 리디렉션 구성 (p. 559) 단원을 참조하십시오.	예

이름	설명	사용자의 값 수정 여부
x-amz-server-side-encryption-aws-kms-key-id	x-amz-server-side-encryption이 표시되고 `aws:kms` 값이 지정된 경우 이 값은 객체에 사용된 AWS KMS 대칭 고객 마스터 키(CMK)의 ID를 나타냅니다.	예
x-amz-server-side-encryption-customer-algorithm	고객 제공 암호화 키(SSE-C)를 사용하는 서버 측 암호화의 활성화 여부를 나타냅니다. 자세한 내용은 고객 제공 암호화 키(SSE-C)로 서버 측 암호화를 사용하여 데이터 보호 (p. 278) 단원을 참조하십시오.	예

사용자 정의 객체 메타데이터

객체를 업로드할 때 객체에 메타데이터를 지정할 수 있습니다. 객체를 생성하기 위해 PUT 또는 POST 요청을 전송할 때 필요할 경우 이름-값(키-값)의 페어로 이 정보를 제공할 수 있습니다. REST API를 사용하여 객체를 업로드할 때 선택 사항으로 제공되는 사용자 정의 메타데이터의 이름은 다른 HTTP 헤더와 구분할 수 있도록 "x-amz-meta-"로 시작해야 합니다. REST API를 사용하여 객체를 검색하면 이 접두사가 반환됩니다. SOAP API를 사용하여 객체를 업로드할 때는 접두사가 필요 없습니다. SOAP API를 사용하여 객체를 검색하면 객체 업로드 시 사용한 API에 관계없이 접두사가 제거됩니다.

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

REST API를 통해 메타데이터를 검색할 때 Amazon S3는 동일한 이름(대소문자 무시)을 가진 헤더를 쉼표로 구분되는 목록으로 결합합니다. 일부 메타데이터에 인쇄되지 않는 문자가 포함될 경우 반환되지 않습니다. 대신 인쇄할 수 없는 메타데이터 항목의 수를 나타내는 값과 함께 `x-amz-missing-meta` 헤더가 반환됩니다.

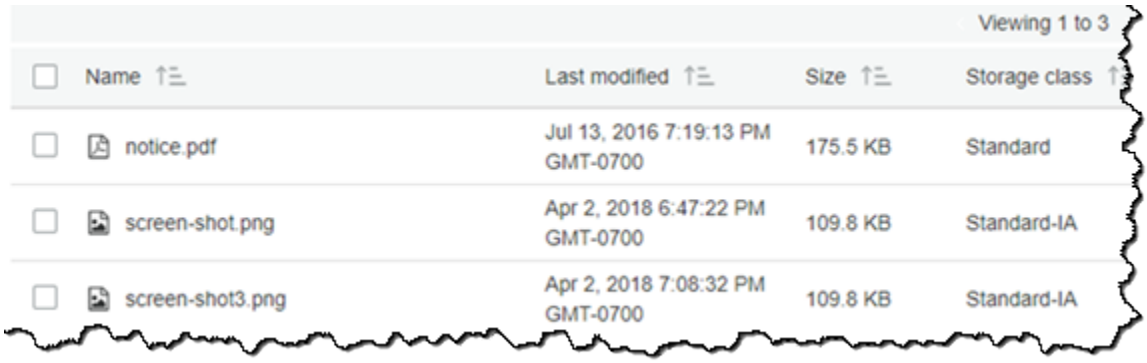
사용자 정의 메타데이터는 키-값 페어의 집합입니다. Amazon S3는 사용자 정의 메타데이터 키를 소문자로 저장합니다. 각 키 값 페어는 REST를 사용하는 경우 US-ASCII, SOAP를 사용하거나 POST로 브라우저 기반 업로드 수행 시 UTF-8을 준수해야 합니다.

> **Note**
>
> PUT 요청 헤더는 크기가 8KB 이하여야 합니다. PUT 요청 헤더에 포함되는 사용자 정의 메타데이터의 크기는 2KB 이하여야 합니다. 사용자 정의 메타데이터의 크기는 UTF-8로 인코딩된 각 키와 값의 바이트 수를 더하여 계산합니다.

객체를 업로드한 후 이에 메타데이터를 추가하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 객체에 메타데이터를 추가하려면 어떻게 해야 합니까? 단원을 참조하십시오.

Amazon S3 스토리지 클래스

Amazon S3의 각 객체에는 연결된 스토리지 클래스가 있습니다. 예를 들어 S3 버킷의 객체를 나열하면 콘솔은 나열된 모든 객체의 스토리지 클래스를 표시합니다.



Amazon S3는 저장된 객체에 대해 다양한 스토리지 클래스를 제공합니다. 사용 사례 시나리오 및 성능 액세스 요구 사항에 따라 클래스를 선택합니다. 모든 스토리지 클래스는 내구성이 뛰어납니다.

주제

- 자주 액세스하는 객체를 위한 스토리지 클래스 (p. 108)
- 자주 액세스하는 객체와 자주 액세스하지 않는 객체를 자동으로 최적화하는 스토리지 클래스 (p. 108)
- 자주 액세스하지 않는 객체를 위한 스토리지 클래스 (p. 109)
- 객체 아카이빙을 위한 스토리지 클래스 (p. 110)
- Amazon S3 스토리지 클래스 비교 (p. 110)
- 객체의 스토리지 클래스 설정 (p. 111)

자주 액세스하는 객체를 위한 스토리지 클래스

성능에 민감한 사용 사례(밀리초 액세스 시간을 필요로 하는 사례)와 자주 액세스되는 데이터를 위해 Amazon S3는 다음과 같은 스토리지 클래스를 제공합니다.

- S3 Standard — 기본 스토리지 클래스입니다. 객체를 업로드할 때 스토리지 클래스를 지정하지 않으면 Amazon S3가 S3 Standard 스토리지 클래스를 할당합니다.
- Reduced Redundancy — RRS(Reduced Redundancy Storage) 스토리지 클래스는 S3 Standard 스토리지 클래스보다 더 적은 중복성으로 저장될 수 있으며 중요하지 않고 재현 가능한 데이터용으로 설계되었습니다.

  > **Important**
  >
  > 이 스토리지 클래스는 사용하지 않는 것이 좋습니다. S3 Standard 스토리지 클래스는 비용 대비 효과가 더 좋습니다.

  내구성 측면에서 RRS 객체는 객체의 연평균 예측 손실율이 0.01%입니다. RRS 객체가 손실되었는데 그 객체에 대해 요청한 경우 Amazon S3가 405 오류를 반환합니다.

자주 액세스하는 객체와 자주 액세스하지 않는 객체를 자동으로 최적화하는 스토리지 클래스

S3 Intelligent-Tiering 스토리지 클래스는 성능 영향 또는 운영 오버헤드 없이 가장 비용 효율적인 스토리지 액세스 계층으로 데이터를 자동으로 이동하여 스토리지 비용을 최적화하도록 설계되었습니다. S3 Intelligent-Tiering은 액세스 패턴이 변경될 때 자주 액세스하는 계층과 저렴한 비용의 자주 액세스하지 않는 계층 간에 세분화된 객체 수준의 데이터를 이동함으로써 자동 비용 절감 효과를 제공합니다. Intelligent-Tiering 스토리지 클래스는 액세스 패턴을 알 수 없거나 예측할 수 없어 수명이 긴 데이터에 대해 스토리지 비용을 자동으로 최적화하려는 경우 이상적입니다.

S3 Intelligent-Tiering 스토리지 클래스는 두 액세스 계층에 객체를 저장합니다. 한 계층은 자주 액세스하는 데이터에 최적화되어 있으며 비용이 저렴한 다른 계층은 자주 액세스하지 않는 데이터에 최적화되어 있습니다. Amazon S3는 객체당 소액의 월별 모니터링 및 자동화 요금으로 S3 Intelligent-Tiering 스토리지 클래스에서 객체의 액세스 패턴을 모니터링하고 연속 30일 동안 액세스하지 않은 객체를 자주 액세스하지 않는 액세스 계층으로 이동합니다.

S3 Intelligent-Tiering 스토리지 클래스를 사용할 때 검색 요금은 없습니다. 빈번하지 않은 액세스 계층의 객체에 액세스하면 이 객체는 자동으로 빈번한 액세스 계층으로 다시 이동합니다. 객체가 S3 Intelligent-Tiering 스토리지 클래스 내 액세스 계층 간에 이동될 때는 계층화 요금이 추가로 적용되지 않습니다.

> **Note**
>
> S3 Intelligent-Tiering 스토리지 클래스는 사용자가 30일 이상 저장하려는 128KB 이상의 객체에 적합합니다. 객체의 크기가 128KB 미만이면 자동 계층화를 사용할 수 없습니다. 더 작은 객체도 저장할 수 있지만 항상 S3 Intelligent-Tiering 스토리지 클래스에서 자주 액세스하는 계층 요금으로 청구됩니다.
> 최소 스토리지 기간인 30일이 끝나기 전에 객체를 삭제하면 30일 요금이 부과됩니다. 요금 정보는 Amazon S3 요금을 참조하십시오.

자주 액세스하지 않는 객체를 위한 스토리지 클래스

S3 Standard_IA 및 S3 One Zone-IA 스토리지 클래스는 수명이 길고 자주 액세스하지 않는 데이터용으로 설계되었습니다. 여기서 IA는 자주 액세스하지 않음을 의미합니다. S3 Standard-IA 및 S3 One Zone-IA 객체는 밀리초 액세스에 사용 가능합니다(S3 Standard 스토리지 클래스와 유사함). Amazon S3는 이러한 객체에 대한 검색 요금을 부과하므로 이러한 객체는 자주 액세스되지 않는 데이터에 가장 적합합니다. 요금 정보는 Amazon S3 요금을 참조하십시오.

예를 들어, S3 Standard-IA 및 S3 One Zone-IA 스토리지 클래스를 선택할 수 있습니다.

- 백업을 저장하는 경우.
- 자주 액세스되지는 않지만 그래도 밀리초 액세스가 필요한 오래된 데이터의 경우. 예를 들어, 데이터를 업로드할 때 S3 Standard 스토리지 클래스를 선택하고 수명 주기 구성을 사용하여 Amazon S3에 객체를 S3 Standard-IA 또는 S3 One Zone-IA 클래스로 전환하도록 지시할 수 있습니다.

  수명 주기 관리에 대한 자세한 내용은 객체 수명 주기 관리 (p. 123)를 참조하십시오.

> **Note**
>
> S3 Standard-IA 및 S3 One Zone-IA 스토리지 클래스는 사용자가 30일 이상 저장하려는 128KB 이상의 객체에 적합합니다. 객체가 128KB 미만이더라도 Amazon S3는 128KB에 대한 요금을 부과합니다. 최소 스토리지 기간인 30일이 끝나기 전에 객체를 삭제하면 30일 요금이 부과됩니다. 요금 정보는 Amazon S3 요금을 참조하십시오.

이들 스토리지 클래스는 다음과 같은 차이가 있습니다.

- S3 Standard-IA — Amazon S3가 객체 데이터를 지리적으로 분리된 여러 개의 가용 영역에 중복 저장합니다(S3 Standard 스토리지 클래스와 유사함). S3 Standard-IA 객체는 가용 영역의 손실에 대한 복원성이 있습니다. 이 스토리지 클래스는 S3 One Zone-IA 클래스보다 뛰어난 가용성 및 복원성을 제공합니다.
- S3 One Zone-IA — Amazon S3가 객체 데이터를 한 개의 가용 영역에만 저장하므로 S3 Standard-IA보다 비용이 더 저렴합니다. 그러나 데이터는 지진 및 홍수와 같은 재해에 의한 가용 영역의 물리적 손실에 대해서는 복원성이 없습니다. S3 One Zone-IA 스토리지 클래스는 Standard-IA만큼 내구성이 있지만 가용성과 복원성은 더 낮습니다. 스토리지 클래스의 내구성과 가용성을 비교하려면 이 단원 끝부분의 내구성 및 가용성 표를 참조하십시오. 요금에 대한 내용은 Amazon S3 요금을 참조하십시오.

다음과 같이 하는 것이 좋습니다.

- S3 Standard-IA — 기본 데이터나 다시 생성할 수 없는 데이터의 복사본에만 사용합니다.
- S3 One Zone-IA — 가용 영역에 장애 발생 시 데이터를 다시 생성할 수 있는 경우, 그리고 S3 교차 리전 복제(CRR)를 설정하는 경우 객체 복제본에 사용합니다.

객체 아카이빙을 위한 스토리지 클래스

S3 Glacier 및 S3 Glacier Deep Archive 스토리지 클래스는 저비용 데이터 아카이빙을 위해 설계되었습니다. 이러한 스토리지 클래스는 S3 Standard 스토리지 클래스와 동일한 내구성과 복원성을 제공합니다. 스토리지 클래스의 내구성과 가용성을 비교하려면 이 단원 끝부분에 있는 표를 참조하십시오.

이들 스토리지 클래스는 다음과 같은 차이가 있습니다.

- S3 Glacier—분 단위로 데이터의 일부를 검색해야 하는 아카이브에 사용합니다. S3 Glacier 스토리지 클래스에 저장된 데이터는 최소 스토리지 기간이 90일이며 신속 검색을 사용하여 최소 1~5분 이내에 액세스할 수 있습니다. 90일 최소 기간 이전에 삭제했거나 덮어썼거나 다른 스토리지 클래스로 이전한 경우, 90일 요금이 부과됩니다. 요금 정보는 Amazon S3 요금을 참조하십시오.
- S3 Glacier Deep Archive—거의 액세스할 필요가 없는 데이터를 보관할 때 사용합니다. S3 Glacier Deep Archive 스토리지 클래스에 저장된 데이터의 최소 스토리지 기간은 180일이고 기본 검색 시간은 12시간입니다. 180일 최소 기간 이전에 삭제했거나 덮어썼거나 다른 스토리지 클래스로 이전한 경우, 180일 요금이 부과됩니다. 요금 정보는 Amazon S3 요금을 참조하십시오.

  S3 Glacier Deep Archive는 AWS에서 가장 저렴한 스토리지 옵션입니다. S3 Glacier Deep Archive에 대한 스토리지 비용은 S3 Glacier 스토리지 클래스를 사용하는 것보다 저렴합니다. 48시간 이내에 데이터를 반환하는 대량 검색을 사용하여 S3 Glacier Deep Archive 검색 비용을 절감할 수 있습니다.

아카이브된 객체 복원

객체의 스토리지 클래스 설정 (p. 111) 섹션에서 설명한 것처럼 다른 스토리지 클래스와 동일한 방법으로 객체의 스토리지 클래스를 S3 Glacier 또는 S3 Glacier Deep Archive로 설정할 수 있습니다. 그러나 S3 Glacier 및 S3 Glacier Deep Archive 객체에는 실시간으로 액세스할 수 없습니다. S3 Glacier 및 S3 Glacier Deep Archive 객체에 액세스하려면 먼저 해당 객체를 복원해야 합니다. S3 Standard, RRS, S3 Standard-IA, S3 One Zone-IA 및 S3 Intelligent-Tiering 객체는 언제든지 액세스할 수 있습니다. 아카이브된 객체 복원에 대한 자세한 내용은 보관된 객체의 복원 (p. 247) 항목을 참조하십시오.

> **Important**
>
> S3 Glacier 또는 S3 Glacier Deep Archive 스토리지 클래스를 선택하면 객체가 Amazon S3에 남아 있습니다. 별도의 Amazon S3 Glacier 서비스를 통해 직접 액세스할 수 없습니다.

Amazon S3 Glacier 서비스에 대한 자세한 내용은 Amazon S3 Glacier 개발자 안내서 항목을 참조하십시오.

Amazon S3 스토리지 클래스 비교

다음 표는 스토리지 클래스를 비교합니다.

Storage Class	Designed for	Durability (designed for)	Availability (designed for)	Availability Zones	Min storage duration	Min billable object size	Other Considerations
STANDARD	Frequently accessed data	99.999999999%	99.99%	>= 3	None	None	None
STANDARD_IA	Long-lived, infrequently accessed data	99.999999999%	99.9%	>= 3	30 days	128 KB	Per GB retrieval fees apply.
INTELLIGENT_TIERING	Long-lived data with changing or unknown access patterns	99.999999999%	99.9%	>= 3	30 days	None	Monitoring and automation fees per object apply. No retrieval fees.
ONEZONE_IA	Long-lived, infrequently accessed, non-critical data	99.999999999%	99.5%	1	30 days	128 KB	Per GB retrieval fees apply. Not resilient to the loss of the Availability Zone.
GLACIER	Long-term data archiving with retrieval times ranging from minutes to hours	99.999999999%	99.99% (after you restore objects)	>= 3	90 days	None	Per GB retrieval fees apply. You must first restore archived objects before you can access them. For more information, see Restoring Archived Objects.
DEEP_ARCHIVE	Archiving rarely accessed data with a default retrieval time of 12 hours	99.999999999%	99.99% (after you restore objects)	>= 3	180 days	None	Per GB retrieval fees apply. You must first restore archived objects before you can access them. For more information, see Restoring Archived Objects.
RRS (Not recommended)	Frequently accessed, non-critical data	99.99%	99.99%	>= 3	None	None	None

S3 One Zone-IA를 제외한 모든 스토리지 클래스는 단일 가용 영역에서의 데이터 완전 손실과 다른 가용 영역에서의 부분 손실에 대해 동시 복원성을 갖도록 설계되어 있습니다.

애플리케이션 시나리오의 성능 요구 사항과 더불어, 가격을 고려하십시오. 스토리지 클래스의 요금에 대해서는 Amazon S3 요금을 참조하십시오.

객체의 스토리지 클래스 설정

Amazon S3 API는 객체의 스토리지 클래스에 대해 다음과 같은 설정(또는 업데이트)을 지원합니다.

- 새 객체를 생성할 때 그 객체의 스토리지 클래스를 지정할 수 있습니다. 예를 들어, PUT Object, POST Object 및 멀티파트 업로드 시작 API를 사용하여 객체를 생성할 때, `x-amz-storage-class` 요청 헤더를 추가하여 스토리지 클래스를 지정하십시오. 이 헤더를 추가하지 않으면 Amazon S3는 기본 스토리지 클래스인 Standard를 사용합니다.
- 또한 PUT Object - Copy API를 사용하여 객체의 복사본을 만들어 Amazon S3에 이미 저장된 객체의 스토리지 클래스를 다른 스토리지 클래스로 변경할 수도 있습니다. 그러나 S3 Glacier 또는 S3 Glacier Deep Archive 스토리지 클래스에 저장된 객체를 복사하려면 PUT Object - Copy를 사용할 수 없습니다.

  그 객체를 동일한 키 이름으로 동일한 버킷에 복사한 후 다음과 같이 요청 헤더를 지정하십시오.
  - `x-amz-metadata-directive` 헤더를 COPY로 설정합니다.
  - `x-amz-storage-class`를 사용하려는 스토리지 클래스로 설정합니다.

  버전 관리를 사용하는 버킷에서는 특정 객체 버전의 스토리지 클래스를 변경할 수 없습니다. 객체를 복사할 때 Amazon S3에서는 여기에 새 버전 ID를 부여합니다.
- 버킷에 S3 수명 주기 구성을 추가하여 Amazon S3가 객체의 스토리지 클래스를 변경하도록 지정할 수 있습니다. 자세한 내용은 객체 수명 주기 관리 (p. 123) 단원을 참조하십시오.
- 복제 구성을 설정할 때 복제된 객체의 스토리지 클래스를 다른 스토리지 클래스로 설정할 수 있습니다. 그러나 S3 Glacier 또는 S3 Glacier Deep Archive 스토리지 클래스에 저장된 객체는 복제할 수 없습니다. 자세한 내용은 복제 구성 개요 (p. 612) 단원을 참조하십시오.

객체 스토리지 클래스를 생성하고 업데이트하려면 Amazon S3 콘솔, AWS SDK 또는 AWS Command Line Interface(AWS CLI)를 사용합니다. 이들 모두는 Amazon S3 API를 사용하여 Amazon S3에 요청을 보냅니다.

액세스 정책 권한을 특정 스토리지 클래스로 제한

Amazon S3 작업에 대한 액세스 정책 권한을 부여할 때는 `s3:x-amz-storage-class` 조건 키를 사용하여 업로드된 객체를 저장할 때 사용할 스토리지 클래스를 제한할 수 있습니다. 예를 들어 `s3:PUTObject` 권한

을 부여하면 객체 업로드를 특정 스토리지 클래스로 제한할 수 있습니다. 정책에 대한 예는 예 5: 특정 스토리지 클래스를 이용한 객체 업로드 제한 (p. 351)을 참조하십시오. 정책의 조건 및 Amazon S3 조건 키의 전체 목록을 사용하는 방법에 대한 자세한 내용은 다음을 참조하십시오.

- Amazon S3에 사용되는 작업, 리소스 및 조건 키 (p. 356)
- Amazon S3 조건 키 (p. 345)

객체 하위 리소스

Amazon S3는 버킷 및 객체에 연결된 하위 리소스의 집합을 정의합니다. 하위 리소스는 객체에 종속됩니다. 즉, 하위 리소스는 자체적으로 존재하지 않습니다. 항상 객체나 버킷 등의 다른 항목과 연결됩니다.

다음 표에 Amazon S3 객체에 연결된 하위 리소스의 목록이 나와 있습니다.

하위 리소스	설명
acl	피부여자 및 부여된 권한을 식별하는 권한 부여 목록이 포함됩니다. 객체를 만들 때 `acl`은 객체에 대한 모든 제어 권한을 보유한 객체 소유자를 식별합니다. 객체 ACL을 검색하거나 업데이트된 권한 부여 목록으로 교체할 수 있습니다. ACL을 업데이트하려면 기존 ACL을 교체해야 합니다. ACL에 대한 자세한 내용은 ACL을 사용한 액세스 관리 (p. 437) 단원을 참조하십시오.
Torrent	Amazon S3는 BitTorrent 프로토콜을 지원합니다. Amazon S3는 `torrent` 하위 리소스를 사용하여 특정 객체에 연결된 Torrent 파일을 반환합니다. Torrent 파일을 검색하려면 GET 요청에 `torrent` 하위 리소스를 지정합니다. Amazon S3가 Torrent 파일을 만들어 반환합니다. 사용자는 `torrent` 하위 리소스의 검색만 가능하며, `torrent` 하위 리소스를 생성, 업데이트 또는 삭제할 수 없습니다. 자세한 내용은 Amazon S3에서 BitTorrent 사용 (p. 694) 단원을 참조하십시오. Note Amazon S3는 2016년 5월 30일 이후 시작된 AWS 리전에서 BitTorrent 프로토콜을 지원하지 않습니다.

객체 버전 관리

Amazon S3 버전 관리를 통해 한 버킷 내에 여러 개의 객체 버전을 유지할 수 있습니다. 예를 들어, `my-image.jpg`(버전 111111) 및 `my-image.jpg`(버전 222222)를 단일 버킷에 저장할 수 있습니다. S3 버전 관리는 의도하지 않은 덮어쓰기 및 삭제 결과에 영향 받지 않도록 사용자를 보호합니다. 또한 버전 관리를 사용해 객체를 아카이브하여 이전 버전에 액세스할 수 있습니다.

> **Note**
>
> SOAP API는 S3 버전 관리를 지원하지 않습니다. HTTP를 통한 SOAP 지원은 중단되었지만 HTTPS를 통해 계속해서 사용할 수 있습니다. Amazon S3의 새로운 기능들은 SOAP를 지원하지 않습니다.

데이터 보존 접근 방식을 사용자 지정하고 스토리지 비용을 관리하기 위해 객체 수명 주기 관리 (p. 123)(으)로 객체 버전 관리를 사용하십시오. AWS Management 콘솔을 사용하여 S3 수명 주기 정책을 생성하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷에 대한 수명 주기 정책을 생성하려면 어떻게 해야 합니까?를 참조하십시오.

버전이 지정되지 않은 버킷에 객체 만료 수명 주기 정책이 있고 버전 관리를 사용할 때 같은 영구 삭제 행동을 유지하고자 하는 경우에는 비 최신 버전의 만료 정책을 추가해야 합니다. 비 최신 버전의 만료 수명 주기

정책은 버전 관리를 사용하는 버킷에서 비 최신 객체 버전의 삭제를 관리합니다. (버전 관리가 활성화된 버킷은 하나의 현재 객체 버전과 0개 이상의 최신이 아닌 객체 버전을 유지합니다.)

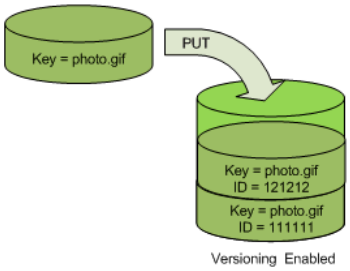
버킷에서 명시적으로 S3 버전 관리를 활성화해야 합니다. 기본적으로 S3 버전 관리는 비활성화되어 있습니다. 버전 관리 활성화 여부에 관계없이 버킷의 각 객체에는 버전 ID가 부여됩니다. 버전 관리를 활성화하지 않으면 Amazon S3가 버전 ID의 값을 null로 설정합니다. S3 버전 관리가 활성화되면 객체의 버전 ID 값을 Amazon S3에 할당합니다. 이 값은 동일한 키의 다른 버전과 구별됩니다.

버전 관리 활성화 및 일시 중지는 버킷 수준에서 이루어집니다. 기존 버킷에서 버전 관리를 활성화하면 버킷에 이미 저장된 객체는 변경되지 않습니다. 버전 ID(null), 콘텐츠, 그리고 권한은 동일하게 유지됩니다. 버킷에 대한 S3 버전 관리를 활성화하면 버킷에 추가된 각 객체가 동일한 키의 다른 버전과 구별되는 버전 ID를 가져옵니다.

Amazon S3는 버전 ID만 생성하며 편집할 수 없습니다. 버전 ID는 유니코드, UTF-8 인코딩, URL 지원, 불투명 문자열이며 길이가 1,024바이트를 넘지 않습니다. 예를 들면, `3/L4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo`입니다.

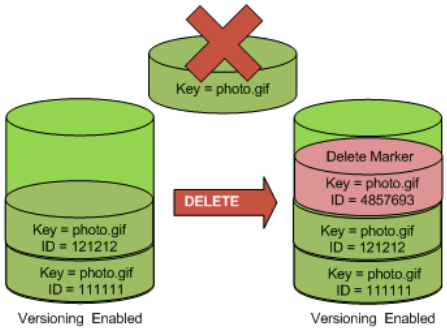
> **Note**
>
> 간단히 설명하기 위해 모든 예제에서는 더 짧은 ID를 사용합니다.

버전 관리를 사용하는 버킷에 객체를 `PUT`할 때 비 최신 버전은 덮어쓰지 않습니다. 다음 그림은 동일한 이름의 객체를 이미 보유하고 있는 버킷에 새 버전의 `photo.gif`를 `PUT`할 때 원래 객체(ID = 111111)는 버킷에 계속 남아 있으며, Amazon S3가 새 버전 ID(121212)를 만들어 버킷에 추가하는 과정을 보여줍니다.

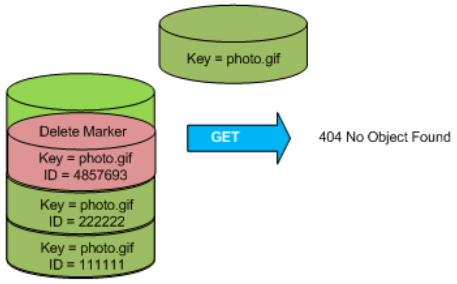


이 기능으로 실수로 객체를 덮어쓰거나 삭제하는 것을 방지하고, 객체의 이전 버전을 검색할 수 있습니다.

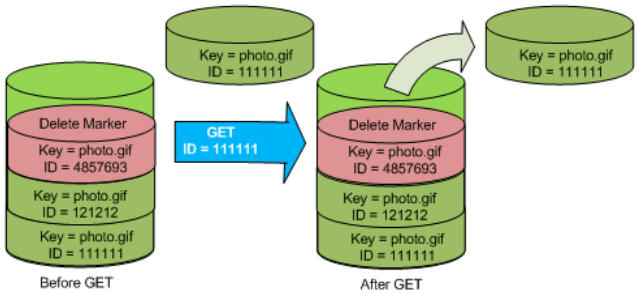
객체를 `DELETE`하면 다음 그림과 같이 버킷에 모든 버전이 계속 유지되고 Amazon S3가 삭제 마커를 삽입합니다.



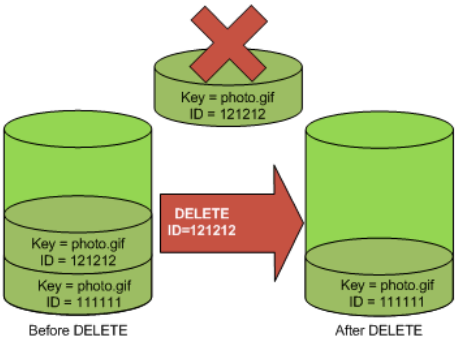
삭제 마커가 객체의 최신 버전이 됩니다. 기본적으로 `GET` 요청은 가장 최근에 저장된 버전을 검색합니다. 최신 버전이 삭제 마커일 때 간단한 `GET Object` 요청을 수행하면 다음 그림과 같이 `404 Not Found` 오류가 반환됩니다.



하지만 버전 ID를 지정하여 최신 버전이 아닌 객체에 대해 `GET`을 실행할 수 있습니다. 다음 그림과 같이 특정 객체 버전(111111)에 대해 `GET`을 실행하면 Amazon S3가 최신 버전은 아니지만 해당 버전의 객체를 반환합니다.



삭제하고자 하는 버전을 지정하여 객체를 영구적으로 삭제할 수 있습니다. Amazon S3 버킷 소유자만이 영구적으로 버전을 삭제할 수 있습니다. 다음 그림은 `DELETE versionId`에 의해 버킷에서 객체가 영구 삭제되고, Amazon S3가 삭제 마커를 삽입하지 않음을 보여줍니다.



MFA(멀티 팩터 인증) Delete이 활성화되도록 버킷을 구성하여 보안을 강화할 수 있습니다. 이렇게 하는 경우 버킷 소유자는 버전을 삭제하거나 버킷의 버전 관리 상태를 변경하는 모든 요청에 두 가지 형식의 인증을 포함해야 합니다. 자세한 내용은 MFA Delete (p. 470) 단원을 참조하십시오.

> **Important**
>
> S3 버전 관리가 활성화된 버킷에 대한 Amazon S3 PUT 또는 DELETE 객체 요청에서 받은 HTTP 503 속도 저하 응답의 횟수가 크게 증가한다면 버전이 무수히 많은 객체가 하나 이상 버킷에 있을 수 있습니다. 자세한 내용은 Amazon S3 문제 해결 (p. 701) 단원을 참조하십시오.

자세한 내용은 버전 관리 사용 (p. 468)을 참조하십시오.

객체 태그 지정

객체 태그 지정을 이용하여 스토리지를 분류합니다. 각 태그는 키-값 페어입니다. 태그 지정에 대한 다음 예시를 참고하십시오.

- 객체에는 보호 대상 건강 정보(PHI) 데이터가 포함되어 있다고 가정합시다. 다음과 같이 키-값 페어를 사용하여 객체에 태그를 지정할 수 있습니다.

```
PHI=True
```

또는

```
Classification=PHI
```

- S3 버킷에 프로젝트 파일을 저장한다고 가정합시다. 아래와 같이 `Project`라는 키와 값으로 이 객체들에 태그를 지정할 수 있습니다.

```
Project=Blue
```

- 다음과 같이 한 객체에 태그를 여러 개 덧붙일 수 있습니다.

```
Project=x
Classification=confidential
```

새 객체를 업로드할 때 태그를 덧붙이거나 기존 객체에 태그를 덧붙일 수 있습니다. 다음을 참조하십시오.

- 한 개체에 태그를 최대 10개까지 연결할 수 있습니다. 각 객체에 연결된 태그에는 고유한 태그 키가 있어야 합니다.
- 태그 키의 최대 길이는 128개 유니코드 문자이며, 태그 값의 최대 길이는 256개 유니코드 문자입니다.
- 키와 값은 대/소문자를 구분합니다.
- 태그 제한에 대한 자세한 내용은 사용자 정의 태그 제한 단원을 참조하십시오.

객체 키 이름 접두사를 통해 스토리지를 분류할 수도 있습니다. 그러나 접두사 기반의 분류는 1차원적입니다. 다음 객체 키 이름을 참고하십시오.

```
photos/photo1.jpg
project/projectx/document.pdf
project/projecty/document2.pdf
```

이 키 이름들에는 `photos/`, `project/projectx/` 및 `project/projecty/`라는 접두사가 있습니다. 이 접두사들을 사용하여 1차원 분류를 할 수 있습니다. 즉 하나의 접두사 아래에 있는 모든 것이 하나의 범주가 됩니다. 예를 들어, `project/projectx` 접두사는 프로젝트 x와 관련된 모든 문서를 식별합니다.

태그 지정을 통해 이제 또 다른 차원을 갖게 되었습니다. 프로젝트 x 범주에 속한 photo1을 원하는 경우, 그에 따라 객체에 태그를 지정할 수 있습니다. 태그 지정은 데이터 분류 뿐만 아니라 다음과 같은 이점도 제공합니다.

- 객체 태그를 통해 권한에 대한 액세스 제어를 세부적으로 수행할 수 있습니다. 예를 들어, IAM 사용자에게 특정 태그가 있는 객체만 읽을 수 있는 권한을 부여할 수 있습니다.
- 객체 태그를 통해 수명 주기 규칙 내에서 키 이름 접두사뿐만 아니라 태그 기반 필터를 지정할 수 있는 세분화된 객체 수명 주기 관리가 가능합니다.
- Amazon S3 분석 기능을 사용하는 경우 분석을 위해 객체를 그룹화하도록 객체 태그, 키 이름 접두사 또는 접두사와 태그 둘 다를 기준으로 필터를 구성할 수 있습니다.
- 또한 Amazon CloudWatch 지표를 사용자 지정하여 특정 태그 필터를 통해 정보를 표시할 수도 있습니다. 다음 단원들에서 세부 정보가 제공됩니다.

> **Important**
>
> 태그를 사용하여 기밀 데이터(예: 개인 식별 정보(PII) 또는 보호 대상 건강 정보(PHI))가 포함된 객체를 라벨링하는 것이 허용됩니다. 하지만 태그 자체에는 기밀 정보가 포함되어서는 안 됩니다.

단일 요청으로 하나 이상의 Amazon S3 객체에 객체 태그 집합을 추가하기 위해 Amazon S3 배치 작업을 사용할 수 있습니다. 작업할 객체 목록을 Amazon S3 Batch 작업에 제공합니다. Amazon S3 Batch 작업은 각각의 API를 호출하여 지정된 작업을 수행합니다. 단일 Amazon S3 Batch 작업으로 엑사바이트 규모의 데이터가 포함된 수십억 개의 객체에서 지정된 작업을 수행할 수 있습니다.

Amazon S3 Batch 작업은 진행 상황을 추적하고 알림을 보내며 모든 작업에 대한 자세한 완료 보고서를 저장하여 감사 가능한 종합 관리형 서버리스 환경을 제공합니다. AWS Management 콘솔, AWS CLI, AWS SDK 또는 REST API를 통해 Amazon S3 Batch 작업을 사용할 수 있습니다. 자세한 내용은 the section called “기본 사항: 작업” (p. 502) 단원을 참조하십시오.

객체 태그 지정과 관련된 API 작업

Amazon S3는 특히 객체 태그 지정을 위한 다음 API 작업을 지원합니다.

객체 API 작업

- PUT Object 태그 지정 – 객체에 있는 태그를 교체합니다. 요청 본문에서 태그를 지정합니다. 이 API를 사용하는, 서로 뚜렷이 구분되는 두 가지 객체 태그 관리 시나리오가 있습니다.
  - 객체에는 태그가 없는 경우 – 이 API를 사용하여 객체에 태그 세트를 붙일 수 있습니다(객체에는 이전에 붙인 태그가 없음).
  - 객체에 기존 태그 세트가 있는 경우 – 기존 태그 세트를 수정하려면 기존 태그 세트를 조회하여 클라이언트 측에서 해당 세트를 수정한 다음, 이 API를 사용하여 태그 세트를 교체해야 합니다.

    > **Note**
    >
    > 빈 태그 세트를 사용하여 이 요청을 전송하는 경우, Amazon S3는 객체에 있는 기존 태그 세트를 삭제합니다. 이 방법을 사용하면 티어 1 요청(PUT)에 대한 요금이 부과됩니다. 자세한 내용은 Amazon S3 요금을 참조하십시오.
    > DELETE Object 태그 지정 요청에는 요금이 부과되지 않고 동일한 결과를 얻으므로 더 유용합니다.
- GET Object 태그 지정 – 객체에 연결된 태그 세트를 반환합니다. Amazon S3는 응답 본문의 객체 태그를 반환합니다.
- DELETE Object 태그 지정 – 객체에 연결된 태그 세트를 삭제합니다.

태그 지정을 지원하는 기타 API 작업

- PUT Object 및 멀티파트 업로드 시작 – 객체를 생성할 때 태그를 지정할 수 있습니다. `x-amz-tagging` 요청 헤더를 사용하여 태그를 지정합니다.
- GET Object – 헤더 응답 크기가 8킬로바이트로 제한되어 있기 때문에 Amazon S3는 태그 세트를 반환하는 대신에 `x-amz-tag-count` 헤더의 객체 태그 개수를 반환합니다(요청자에게 태그를 읽을 수 있는 권한이 있는 경우에 한함). 태그를 보고 싶은 경우, GET Object 태그 지정 API 작업에 대해 또 다른 요청을 합니다.
- POST Object – POST 요청에서 태그를 지정할 수 있습니다.

  요청 시 태그가 8킬로바이트 HTTP 요청 헤더 크기 제한을 초과하지 않는 한, `PUT Object` API를 사용하여 태그가 있는 객체를 만들 수 있습니다. 지정하는 태그가 헤더 크기 제한을 초과하는 경우, 태그를 본문에 포함시키는 이 POST 방법을 사용할 수 있습니다.

  PUT Object - Copy – 요청 시 `x-amz-tagging-directive`를 지정하여 Amazon S3가 태그를 복사(기본 동작)하거나 요청 시 제공되는 새로운 태그 세트로 태그를 교체하도록 지시할 수 있습니다.

다음을 참조하십시오.

- 태그 지정은 최종 일관성 모델을 따릅니다. 즉 객체에 태그를 붙인 직후에 태그를 조회하고자 하는 경우, 객체에 붙은 이전 태그(있는 경우)를 조회할 수 있습니다. 그러나 후속 호출은 업데이트된 태그를 제공할 수 있습니다.

객체 태그 지정 및 부가 정보

이 단원에서는 객체 태그 지정이 다른 구성과 어떤 관계를 맺는지에 대해 설명합니다.

객체 태그 지정 및 수명 주기 관리

버킷 수명 주기 구성에서 필터를 지정하여 규칙을 적용할 객체의 하위 집합을 선택할 수 있습니다. 키 이름 접두사, 객체 태그 또는 이 두 가지를 바탕으로 필터를 지정할 수 있습니다.

Amazon S3 버킷에 사진(원본 및 마감 형식)을 저장한다고 가정합시다. 이 객체들에 다음과 같이 태그를 지정할 수 있습니다.

```
phototype=raw
or
phototype=finished
```

원본 사진은 생성한 후 언젠가 S3 Glacier에 보관하는 것을 고려해볼 수 있습니다. 특정 태그(`photos/`)를 지닌 키 이름 접두사(`phototype=raw`)가 있는 객체의 하위 집합을 식별하는 필터를 사용하여 수명 주기 규칙을 구성할 수 있습니다.

자세한 내용은 객체 수명 주기 관리 (p. 123) 단원을 참조하십시오.

객체 태그 지정 및 복제

버킷에서 복제를 구성한 경우, 태그를 읽을 수 있는 권한을 부여하면 Amazon S3는 태그를 복제합니다. 자세한 내용은 복제 설정 개요 (p. 611) 단원을 참조하십시오.

객체 태그 지정 및 액세스 제어 정책

또한 권한 정책(버킷 및 사용자 정책)을 사용하여 객체 태그 지정과 관련된 권한을 관리할 수 있습니다. 정책 작업에 대해서는 다음 주제를 참조하십시오.

- — 객체 작업 예제 (p. 343)
- — 버킷 작업 예제 (p. 343)

객체 태그를 통해 권한 관리에 대한 액세스 제어를 세부적으로 수행할 수 있습니다. 객체 태그를 바탕으로 조건적인 권한을 부여할 수 있습니다. Amazon S3에서는 객체 태그를 기반으로 조건적 권한을 부여하는 데 사용할 수 있는 다음과 같은 조건 키를 지원합니다.

- `s3:ExistingObjectTag/<tag-key>` – 이 조건 키를 사용하여 기존 객체 태그에 특정 태그 키 및 값이 있다는 것을 확인합니다.

  Note

  `PUT Object` 및 `DELETE Object` 작업에 대한 권한을 부여할 때 이 조건 키는 지원되지 않습니다. 즉 기존 태그에 기반을 둔 객체를 삭제 또는 덮어쓸 수 있는 권한을 사용자에게 부여하거나 거부할 수 있는 정책을 생성할 수 없습니다.
- `s3:RequestObjectTagKeys` – 이 조건을 사용하여 객체에 대해 허용하고자 하는 태그 키를 제한합니다. 이것은 PutObjectTagging 및 PutObject, 그리고 POST 객체 요청을 사용하여 객체에 태그를 붙일 때 유용합니다.

- `s3:RequestObjectTag/<tag-key>` – 이 조건을 사용하여 객체에 대해 허용하고자 하는 태그 키 및 값을 제한합니다. 이것은 PutObjectTagging 및 PutObject, 그리고 POST Bucket 요청을 사용하여 객체에 태그를 붙일 때 유용합니다.

Amazon S3 서비스별 조건 키의 전체 목록은 Amazon S3 조건 키 (p. 345) 단원을 참조하십시오. 다음 권한 정책들은 객체 태그 지정을 통해 어떻게 세부적인 액세스 권한 관리가 가능한지 보여줍니다.

Example 1: 사용자가 특정 태그가 있는 객체만을 읽도록 허용

다음 권한 정책은 객체를 읽을 수 있는 권한을 사용자에게 부여하지만, 해당 조건으로 인해 읽기 권한은 다음과 같은 특정 태그 키 및 값이 있는 객체로 제한됩니다.

```
security : public
```

이 정책은 Amazon S3 조건 키 `s3:ExistingObjectTag/<tag-key>`를 사용하여 키 및 값을 지정합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect":      "Allow",
      "Action":      "s3:GetObject",
      "Resource":     "arn:aws:s3:::examplebucket/*",
      "Principal":    "*",
      "Condition": {  "StringEquals": {"s3:ExistingObjectTag/security": "public" } }
    }
  ]
}
```

Example 2: 사용자가 객체 태그를 붙일 수 있게 하되, 허용되는 태그 키에 제한을 둠

다음 권한 정책은 사용자에게 `s3:PutObjectTagging` 작업을 수행할 수 있는 권한을 부여합니다. 이에 따라 사용자는 기존 객체에 태그를 붙일 수 있습니다. 이 조건은 사용자가 사용하도록 허용된 태그 키를 제한합니다. 이 조건은 `s3:RequestObjectTagKeys` 조건 키를 사용하여 태그 키 세트를 지정합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::examplebucket/*"
      ],
      "Principal":{
        "CanonicalUser":[
            "64-digit-alphanumeric-value"
         ]
       },
      "Condition": {
        "ForAllValues:StringLike": {
          "s3:RequestObjectTagKeys": [
            "Owner",
            "CreationDate"
          ]
        }
      }
    }
```

```
  ]
}
```

이 정책은 요청에 지정된 경우, 태그 세트에 지정된 키가 있는지 확인합니다. 사용자는 `PutObjectTagging`에 빈 태그 세트를 전송할 수 있습니다. 이는 이 정책에 의해 허용됩니다(요청의 빈 태그 세트는 객체의 기존 태그를 제거함). 사용자가 태그 세트를 제거하지 못하도록 하려면 다른 조건을 추가하여 사용자가 최소한 1개의 값을 제공하도록 할 수 있습니다. 조건 내의 `ForAnyValue`는 지정된 값 중 최소한 1개가 요청에 반드시 있도록 보장합니다.

```
{

  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::examplebucket/*"
      ],
      "Principal":{
        "AWS":[
            "arn:aws:iam::account-number-without-hyphens:user/username"
         ]
       },
      "Condition": {
        "ForAllValues:StringLike": {
          "s3:RequestObjectTagKeys": [
            "Owner",
            "CreationDate"
          ]
        },
        "ForAnyValue:StringLike": {
          "s3:RequestObjectTagKeys": [
            "Owner",
            "CreationDate"
          ]
        }
      }
    }
  ]
}
```

자세한 내용은 IAM 사용 설명서의 여러 키 값을 테스트하는 조건 생성(설정 작업) 단원을 참조하십시오.

Example 3: 사용자가 특정 태그 키 및 값이 포함된 객체 태그를 붙일 수 있게 허용

다음 사용자 정책은 사용자에게 `s3:PutObjectTagging` 작업을 수행할 수 있는 권한을 부여합니다. 이에 따라 사용자는 기존 객체에 태그를 붙일 수 있습니다. 이 조건은 사용자에게 `Project`로 설정된 값을 지닌 특정 태그(X)를 포함하도록 요구합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::examplebucket/*"
```

```
      ],
      "Principal":{
        "AWS":[
            "arn:aws:iam::account-number-without-hyphens:user/username"
         ]
       },
      "Condition": {
        "StringEquals": {
          "s3:RequestObjectTag/Project": "X"
        }
      }
    }
  ]
}
```

관련 주제

객체 태그 관리 (p. 120)

객체 태그 관리

이 단원에서는 Java용 AWS SDK 또는 Amazon S3 콘솔을 사용하여 프로그램 방식으로 객체 태그를 추가하는 방법에 대해 설명합니다.

주제

- 콘솔을 사용한 객체 태그 관리 (p. 120)
- AWS SDK for Java를 사용하여 태그 관리 (p. 120)
- .NET용 AWS SDK를 사용한 태그 관리 (p. 121)

콘솔을 사용한 객체 태그 관리

새 객체를 업로드할 때 또는 기존 객체에 태그를 붙일 때에 Amazon S3 콘솔을 사용하여 태그를 붙일 수 있습니다. Amazon S3 콘솔을 사용하여 객체에 태그를 추가하는 방법에 대한 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 객체 태그 추가 단원을 참조하십시오.

AWS SDK for Java를 사용하여 태그 관리

다음 예제는 AWS SDK for Java를 사용하여 새 객체에 대한 태그를 설정하고 기존 객체에 대한 태그를 가져오거나 대체하는 방법을 보여줍니다. 객체 태그 지정에 대한 자세한 정보는 객체 태그 지정 (p. 114) 단원을 참조하십시오. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
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;
import com.amazonaws.services.s3.model.*;

import java.io.File;
import java.util.ArrayList;
import java.util.List;

public class ManagingObjectTags {

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

```
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Object key ***";
        String filePath = "*** File path ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Create an object, add two new tags, and upload the object to Amazon S3.
            PutObjectRequest putRequest = new PutObjectRequest(bucketName, keyName, new
 File(filePath));
            List<Tag> tags = new ArrayList<Tag>();
            tags.add(new Tag("Tag 1", "This is tag 1"));
            tags.add(new Tag("Tag 2", "This is tag 2"));
            putRequest.setTagging(new ObjectTagging(tags));
            PutObjectResult putResult = s3Client.putObject(putRequest);

            // Retrieve the object's tags.
            GetObjectTaggingRequest getTaggingRequest = new
 GetObjectTaggingRequest(bucketName, keyName);
            GetObjectTaggingResult getTagsResult =
 s3Client.getObjectTagging(getTaggingRequest);

            // Replace the object's tags with two new tags.
            List<Tag> newTags = new ArrayList<Tag>();
            newTags.add(new Tag("Tag 3", "This is tag 3"));
            newTags.add(new Tag("Tag 4", "This is tag 4"));
            s3Client.setObjectTagging(new SetObjectTaggingRequest(bucketName, keyName, new
 ObjectTagging(newTags)));
        } 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();
        }
    }
}
```

.NET용 AWS SDK를 사용한 태그 관리

다음 예제는 .NET용 AWS SDK를 사용하여 새 객체에 대한 태그를 설정하고 기존 객체에 대한 태그를 가져오거나 대체하는 방법을 보여줍니다. 객체 태그 지정에 대한 자세한 정보는 객체 태그 지정 (p. 114) 단원을 참조하십시오.

실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

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

namespace Amazon.DocSamples.S3
{
```

```
    public class ObjectTagsTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** key name for the new object ***";
        private const string filePath = @"*** file path ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            PutObjectWithTagsTestAsync().Wait();
        }

        static async Task PutObjectWithTagsTestAsync()
        {
            try
            {
                // 1. Put 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);
                // 2. Retrieve the object's tags.
                GetObjectTaggingRequest getTagsRequest = new GetObjectTaggingRequest
                {
                    BucketName = bucketName,
                    Key = keyName
                };

                GetObjectTaggingResponse objectTags = await
client.GetObjectTaggingAsync(getTagsRequest);
                for (int i = 0; i < objectTags.Tagging.Count; i++)
                    Console.WriteLine("Key: {0}, Value: {1}", objectTags.Tagging[i].Key,
objectTags.Tagging[i].Value);


                // 3. Replace the tagset.

                Tagging newTagSet = new Tagging();
                newTagSet.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);

                // 4. Retrieve the object's tags.
                GetObjectTaggingRequest getTagsRequest2 = new GetObjectTaggingRequest();
```

```
                getTagsRequest2.BucketName = bucketName;
                getTagsRequest2.Key = keyName;
                GetObjectTaggingResponse objectTags2 = await
client.GetObjectTaggingAsync(getTagsRequest2);
                for (int i = 0; i < objectTags2.Tagging.Count; i++)
                    Console.WriteLine("Key: {0}, Value: {1}", objectTags2.Tagging[i].Key,
objectTags2.Tagging[i].Value);

            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine(
                        "Error encountered ***. Message:'{0}' when writing an object"
                        , e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine(
                    "Encountered an error. Message:'{0}' when writing an object"
                    , e.Message);
            }
        }
    }
}
```

객체 수명 주기 관리

수명 주기 동안 객체가 비용 효율적으로 저장되도록 관리하려면 해당 Amazon S3 수명 주기를 구성합니다. S3 수명 주기 구성은 Amazon S3가 객체 그룹에 적용되는 작업을 정의하는 일련의 규칙입니다. 다음과 같은 두 가지 유형의 작업이 있습니다.

- 전환 작업 — 객체가 다른 스토리지 클래스로 전환할 시기를 정의합니다. 예를 들어, 생성 후 30일이 지나면 객체를 S3 Standard-IA 스토리지 클래스로 전환하거나 생성 후 1년이 지나면 객체를 S3 Glacier 스토리지 클래스에 아카이브하도록 선택할 수 있습니다.

  수명 주기 이전 요청과 관련된 비용이 있습니다. 요금 정보는 Amazon S3 요금을 참조하십시오.

- 만료 작업 — 객체가 만료되는 시기를 정의합니다. Amazon S3은 만료된 객체를 자동으로 삭제합니다.

  수명 주기 만료 비용은 선택한 객체 만료 시점에 따라 달라집니다. 자세한 내용은 객체 만료 이해 (p. 129) 단원을 참조하십시오.

S3 수명 주기 규칙에 대한 자세한 내용은 수명 주기 구성의 요소 (p. 130) 단원을 참조하십시오.

수명 주기 구성을 사용해야 할 때는 언제입니까?

수명 주기가 명확한 객체에 대해 S3 수명 주기 구성 규칙을 정의합니다. 다음 예를 참조하십시오.

- 버킷에 주기적으로 로그를 업로드할 경우 애플리케이션에서는 이것을 한 주나 한 달 동안 필요로 할 수 있습니다. 그 이후에는 사용자가 이것을 삭제하고 싶을 것입니다.
- 일부 문서는 제한된 기간 동안 자주 액세스됩니다. 그 이후에는 문서가 가끔 액세스됩니다. 어느 시점이 되면 이러한 문서에 실시간으로 액세스할 필요가 없지만 조직 또는 규정에서 특정 기간 동안 해당 문서를 보관할 것을 요구할 수도 있습니다. 그 이후에는 사용자가 문서를 삭제할 수 있습니다.
- 어떤 유형의 데이터는 주로 보관 목적으로 Amazon S3에 업로드할 수도 있습니다. 예를 들어, 디지털 미디어, 금융 및 의료 기록, 가공되지 않은 유전체 염기서열 데이터, 장기 데이터베이스 백업 파일, 그리고 규제 준수를 위해 보존해야 하는 데이터 등을 보관할 것입니다.

S3 수명 주기 구성 규칙을 사용하면 Amazon S3가 객체를 더 저렴한 스토리지 클래스로 전환하거나 아카이브하거나 삭제하도록 지시할 수 있습니다.

수명 주기를 구성하는 방법

S3 수명 주기 구성(XML 파일)은 객체의 수명 동안 해당 객체에 대해 Amazon S3가 수행하도록 할 사전 정의된 작업이 포함된 일련의 규칙으로 이루어져 있습니다.

Amazon S3가 버킷에서 수명 주기 구성을 관리하기 위한 일련의 API 작업을 제공합니다. Amazon S3는 버킷에 연결된 수명 주기 하위 리소스로 구성을 저장합니다. 세부 정보는 다음을 참조하십시오.

PUT Bucket lifecycle

GET Bucket lifecycle

DELETE Bucket lifecycle

수명 주기는 Amazon S3 콘솔을 사용하여 또는 AWS SDK 래퍼 라이브러리를 사용해 프로그래밍 방식으로 구성할 수도 있습니다. 필요한 경우 직접 REST API를 호출할 수도 있습니다. 자세한 내용은 버킷에서 수명 주기 구성 설정 (p. 146) 단원을 참조하십시오.

자세한 내용은 다음 항목을 참조하십시오.

- 수명 주기 구성에 관한 추가 고려 사항 (p. 124)
- 수명 주기 구성의 요소 (p. 130)
- 수명 주기 구성의 예제 (p. 136)
- 버킷에서 수명 주기 구성 설정 (p. 146)

수명 주기 구성에 관한 추가 고려 사항

객체의 수명 주기를 구성할 때는 객체의 전환과 만료 날짜의 설정, 그리고 기타 객체 구성들에 관한 다음 지침을 이해해야 합니다.

주제

- Amazon S3 수명 주기를 사용하여 객체 이전 (p. 124)
- 객체 만료 이해 (p. 129)
- 수명 주기 및 기타 버킷 구성 (p. 129)

Amazon S3 수명 주기를 사용하여 객체 이전

S3 수명 주기 구성에 규칙을 추가하여 Amazon S3가 객체를 다른 Amazon S3 스토리지 클래스로 전환하도록 지시할 수 있습니다. 다음 예를 참조하십시오.

- 객체에 자주 액세스하지 않는 경우 해당 객체를 S3 Standard-IA 스토리지 클래스로 전환할 수 있습니다.
- 실시간으로 액세스할 필요가 없는 객체를 S3 Glacier 스토리지 클래스에 보관할 수 있습니다.

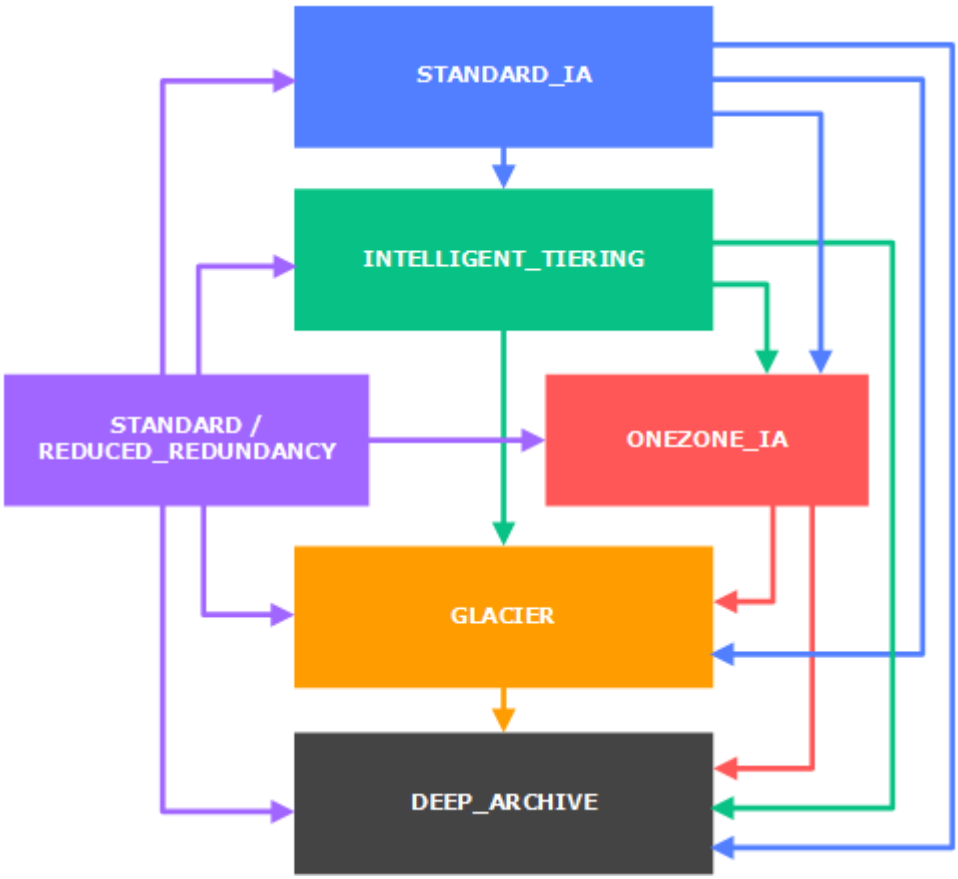
다음 섹션에서는 지원되는 전환 작업, 관련 제한 사항 및 S3 Glacier 스토리지 클래스로의 전환에 대해 설명합니다.

지원되는 전환 작업 및 관련 제한 사항

S3 수명 주기 구성에서 객체를 한 스토리지 클래스에서 다른 스토리지 클래스로 전환하여 스토리지 비용을 절약하도록 규칙을 정의할 수 있습니다. 객체의 액세스 패턴을 모르거나 시간이 지남에 따라 액세스 패턴이

변경되면 자동 비용 절감을 위해 객체를 S3 Intelligent-Tiering 스토리지 클래스로 전환할 수 있습니다. 스토리지 클래스에 대한 자세한 내용은 Amazon S3 스토리지 클래스 (p. 107) 단원을 참조하십시오.

Amazon S3는 다음 다이어그램과 같이 스토리지 클래스간 전환을 위한 폭포형(Waterfall) 모델을 지원합니다.



지원되는 수명 주기 전환

Amazon S3는 S3 수명 주기 구성을 사용하여 스토리지 클래스 간에 다음과 같은 수명 주기 전환을 지원합니다.

다음과 같이 전환할 수 있습니다.

- S3 Standard 스토리지 클래스에서 다른 스토리지 클래스로 전환할 수 있습니다.
- 모든 스토리지 클래스에서 S3 Glacier 또는 S3 Glacier Deep Archive 스토리지 클래스로 전환할 수 있습니다.
- S3 Standard-IA 스토리지 클래스에서 S3 Intelligent-Tiering 또는 S3 One Zone-IA 스토리지 클래스로 전환할 수 있습니다.
- S3 Intelligent-Tiering 스토리지 클래스에서 S3 One Zone-IA 스토리지 클래스로 전환할 수 있습니다.
- S3 Glacier 스토리지 클래스에서 S3 Glacier Deep Archive 스토리지 클래스로 전환할 수 있습니다.

지원되지 않는 수명 주기 전환

Amazon S3는 다음 수명 주기 전환을 지원하지 않습니다.

다음과 같이 전환할 수 없습니다.

- 임의의 스토리지 클래스에서 S3 Standard 스토리지 클래스로 전환할 수 없습니다.
- 임의의 스토리지 클래스에서 Reduced Redundancy 스토리지 클래스로 전환할 수 없습니다.
- S3 Intelligent-Tiering 스토리지 클래스에서 S3 Standard-IA 스토리지 클래스로 전환할 수 없습니다.
- S3 One Zone-IA 스토리지 클래스에서 S3 Standard-IA 또는 S3 Intelligent-Tiering 스토리지 클래스로 전환할 수 없습니다.

Constraints

수명 주기 스토리지 클래스 전환에는 다음과 같은 제약이 있습니다.

객체 크기 및 S3 Standard 또는 S3 Standard-IA에서 S3 Intelligent-Tiering, S3 Standard-IA 또는 S3 One Zone-IA로의 전환

객체를 S3 Standard 또는 S3 Standard-IA 스토리지 클래스에서 S3 Intelligent-Tiering, S3 Standard-IA 또는 S3 One Zone-IA 스토리지 클래스로 전환하는 경우 다음과 같은 객체 크기 제약이 적용됩니다.

- 크기가 큰 객체 - 다음과 같은 전환의 경우, 크기가 큰 객체 전환 시 비용적인 이점이 있습니다.
  - S3 Standard 또는 S3 Standard-IA 스토리지 클래스에서 S3 Intelligent-Tiering로의 전환.
  - S3 Standard 스토리지 클래스에서 S3 Standard-IA 또는 S3 One Zone-IA로의 전환.
- 128KB보다 작은 객체 - 다음과 같은 전환의 경우, 128KB보다 작은 객체는 비용 효율적이지 않으므로 Amazon S3는 이러한 객체를 전환하지 않습니다.
  - S3 Standard 또는 S3 Standard-IA 스토리지 클래스에서 S3 Intelligent-Tiering로의 전환.
  - S3 Standard 스토리지 클래스에서 S3 Standard-IA 또는 S3 One Zone-IA로의 전환.

S3 Standard 또는 S3 Standard-IA에서 S3 Standard-IA 또는 S3 One Zone-IA로의 전환을 위한 최소 일수

객체를 S3 Standard 또는 S3 Standard-IA 스토리지 클래스에서 S3 Standard-IA 또는 S3 One Zone-IA 스토리지 클래스로 전환하기 전에 객체를 S3 Standard 스토리지 클래스에 30일 이상 보관해야 합니다. 예를 들어, 생성 후 1일이 지난 객체를 S3 Standard-IA 스토리지 클래스로 전환하는 수명 주기 규칙을 생성할 수 없습니다. 새 객체는 더 자주 액세스되거나 S3 Standard-IA 또는 S3 One Zone-IA 스토리지에 적합해지기 전에 삭제되는 경우가 종종 있기 때문에 Amazon S3는 최초 30일 동안에는 객체를 전환하지 않습니다.

마찬가지로, 최신이 아닌 객체를 전환할 경우(버전이 지정된 버킷에서) 30일 이상의 최신이 아닌 객체만 S3 Standard-IA 또는 S3 One Zone-IA 스토리지로 전환할 수 있습니다.

S3 Intelligent-Tiering, S3 Standard-IA, 및 S3 One Zone-IA에 대한 최소 30일 스토리지 요금

S3 Intelligent-Tiering, S3 Standard-IA 및 S3 One Zone-IA 스토리지 클래스의 스토리지 요금은 최소 30일을 기준으로 하고 있습니다. 따라서 S3 Glacier 또는 S3 Glacier Deep Archive 전환이 S3 Intelligent-Tiering, S3 Standard-IA 또는 S3 One Zone-IA 전환 후 30일이 경과하기 전에 수행될 경우 S3 Intelligent-Tiering, S3 Standard-IA 또는 S3 One Zone-IA 전환과 S3 Glacier 또는 S3 Glacier Deep Archive 전환 모두에 대해 하나의 수명 주기 규칙을 지정할 수는 없습니다.

S3 Standard-IA 스토리지에서 S3 One Zone-IA 또는 S3 Intelligent-Tiering 스토리지로의 전환을 지정할 때에도 똑같이 최소 30일이 적용됩니다. 이를 달성하기 위해 두 가지 규칙을 지정할 수 있지만 사용자는 최소 스토리지 요금을 지불하게 됩니다. 비용 고려 사항에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

객체의 전체 수명 주기 관리

이러한 S3 수명 주기 작업을 결합하여 객체의 전체 수명 주기를 관리할 수 있습니다. 예를 들어, 생성하는 객체의 수명 주기가 명확하게 정의되어 있다고 가정해봅시다. 처음 30일 동안에는 객체에 자주 액세스합니다. 그 후, 최대 90일 동안에는 객체에 가끔 액세스합니다. 그 다음에는 객체가 더 이상 필요하지 않아서 보관하거나 삭제하기로 선택할 것입니다.

이 시나리오에서는 S3 Intelligent-Tiering, S3 Standard-IA 또는 S3 One Zone-IA 스토리지로의 최초 전환 작업과 아카이브를 위한 S3 Glacier 스토리지로의 다른 전환 작업, 그리고 만료 작업을 지정하는 S3 수명 주기

규칙을 생성할 수 있습니다. 한 스토리지 클래스에서 다른 스토리지 클래스로 객체를 이동하면 스토리지 비용이 절약됩니다. 비용 고려 사항에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

S3 Glacier 및 S3 Glacier Deep Archive 스토리지 클래스로 전환(객체 보관)

S3 수명 주기 구성을 사용하여 아카이브를 위해 객체를 S3 Glacier 또는 S3 Glacier Deep Archive 스토리지 클래스로 전환할 수 있습니다. S3 Glacier 또는 S3 Glacier Deep Archive 스토리지 클래스를 선택하면 객체가 Amazon S3에 남아 있습니다. 별도의 Amazon S3 Glacier 서비스를 통해 직접 액세스할 수 없습니다.

객체를 보관하기 전에 이와 관련된 고려 사항에 대해 설명한 다음 단원들을 살펴보십시오.

일반적인 고려 사항

다음은 객체를 보관하기 전에 고려할 일반적인 사항들입니다.

- 암호화된 객체는 스토리지 클래스 전환 프로세스 전체에서 암호화된 상태로 유지됩니다.
- S3 Glacier 또는 S3 Glacier Deep Archive 스토리지 클래스에 저장된 객체는 실시간으로 사용할 수 없습니다.

  보관된 객체는 Amazon S3 객체이지만 보관된 객체에 액세스하려면 먼저 해당 객체의 임시 복사본을 복원해야 합니다. 복원된 객체 사본은 요청자가 복원 요청에서 지정한 기간 동안만 사용할 수 있습니다. 그 이후에 Amazon S3은 임시 복사본을 삭제하고 해당 객체가 Amazon S3 Glacier에 아카이브된 상태로 유지됩니다.

  Amazon S3 콘솔을 사용하거나 코드에 AWS SDK 래퍼 라이브러리 또는 Amazon S3 REST API를 사용하여 프로그래밍 방식으로 객체를 복원할 수 있습니다. 자세한 내용은 보관된 객체의 복원 (p. 247) 단원을 참조하십시오.
- S3 Glacier 스토리지 클래스에 저장된 객체는 S3 Glacier Deep Archive 스토리지 클래스로만 전환할 수 있습니다.

  S3 수명 주기 구성 규칙을 사용하여 객체의 스토리지 클래스를 S3 Glacier에서 S3 Glacier Deep Archive 스토리지 클래스로만 변환할 수 있습니다. S3 Glacier에 저장된 객체의 스토리지 클래스를 S3 Glacier Deep Archive가 아닌 스토리지 클래스로 변경하려면 복원 작업을 사용하여 객체의 임시 복사본을 만들어야 합니다. 그런 다음 복사 작업을 사용하여 S3 Standard, S3 Intelligent-Tiering, S3 Standard-IA, S3 One Zone-IA 또는 Reduced Redundancy를 스토리지 클래스로 지정하여 객체를 덮어씁니다.
- 객체를 S3 Glacier Deep Archive 스토리지 클래스로 전환할 때는 한 방향으로만 이동할 수 있습니다.

  S3 수명 주기 구성 규칙을 사용하여 객체의 스토리지 클래스를 S3 Glacier Deep Archive에서 다른 스토리지 클래스로 변환할 수는 없습니다. 아카이빙된 객체의 스토리지 클래스를 다른 스토리지 클래스로 변경하고자 하는 경우, 먼저 복원 작업을 통해 객체의 임시 사본을 만들어야 합니다. 그런 다음 복사 작업을 사용하여 S3 Standard, S3 Intelligent-Tiering, S3 Standard-IA, S3 One Zone-IA, S3 Glacier 또는 Reduced Redundancy를 스토리지 클래스로 지정하여 객체를 덮어씁니다.
- S3 Glacier 및 S3 Glacier Deep Archive 스토리지 클래스에 저장된 객체는 표시되며 Amazon S3을 통해서만 사용할 수 있습니다. 별도의 Amazon S3 Glacier 서비스를 통해 사용할 수는 없습니다.

  이들은 Amazon S3 객체이므로 Amazon S3 콘솔 또는 Amazon S3 API를 통해서만 액세스할 수 있습니다. 별도의 Amazon S3 Glacier 콘솔 또는 Amazon S3 Glacier API를 통해서는 아카이브에 보관된 객체에 액세스할 수 없습니다.

비용 고려 사항

자주 액세스하지 않는 데이터를 수개월 또는 수년 동안 보관할 계획이라면 S3 Glacier 및 S3 Glacier Deep Archive 스토리지 클래스로 스토리지 비용을 절감할 수 있습니다. 그러나 S3 Glacier 또는 S3 Glacier Deep Archive 스토리지 클래스가 적합한지 확인하려면 다음을 고려하십시오.

- 스토리지 오버헤드 요금 – S3 Glacier 또는 S3 Glacier Deep Archive 스토리지 클래스로 객체를 전환할 때 각 객체에 객체 관리를 위한 메타데이터용으로 일정 용량의 스토리지가 추가됩니다.

  - S3 Glacier 또는 S3 Glacier Deep Archive에 보관된 각 객체에 대해 Amazon S3은 객체 이름 및 기타 메타데이터용으로 8KB의 스토리지를 사용합니다. Amazon S3은 이 메타데이터를 저장하므로 Amazon S3 API를 사용하여 보관된 객체의 실시간 목록을 확인할 수 있습니다. 자세한 내용은 GET Bucket (List Objects) 단원을 참조하십시오. 이 추가 스토리지에 대해 Amazon S3 Standard 요금이 청구됩니다.
  - S3 Glacier 또는 S3 Glacier Deep Archive에 보관되는 각 객체에 대해 Amazon S3는 인덱스 및 관련 메타데이터에 대해 32KB의 스토리지를 추가합니다. 이 추가 데이터는 객체를 식별하고 복원하는 데 필요합니다. 이 추가 스토리지에 대해 S3 Glacier 또는 S3 Glacier Deep Archive 요금이 청구됩니다.

  작은 크기의 객체를 보관할 경우 이러한 스토리지 요금을 고려해야 합니다. 또한 크기가 작은 여러 객체를 더 적은 수의 크기가 큰 객체로 통합하면 오버헤드 비용을 줄일 수 있습니다.
- 객체의 보관 일수 — S3 Glacier 및 S3 Glacier Deep Archive는 장기간 보관 솔루션입니다. 최소 스토리지 기간은 S3 Glacier 스토리지 클래스의 경우 90일, S3 Glacier Deep Archive의 경우 180일입니다. 삭제한 객체가 최소 스토리지 기간보다 오래 보관되는 경우 Amazon S3 Glacier에 보관된 데이터를 삭제하는 것은 무료입니다. 최소 기간 내에 보관된 객체를 삭제하거나 덮어쓸 경우 Amazon S3은 비례 할당으로 계산된 조기 삭제 요금을 부과합니다.
- S3 Glacier 및 S3 Glacier Deep Archive 전환 요청 요금 — S3 Glacier 또는 S3 Glacier Deep Archive 스토리지 클래스로 전환하는 각 객체는 하나의 전환 요청을 구성합니다. 그러한 각 요청에 대해 비용이 부과됩니다. 많은 수의 객체를 이전할 경우 요청 요금을 고려해야 합니다. 스몰 객체를 아카이브 중인 경우 많은 스몰 객체를 적은 수의 라지 객체로 집계하여 변환 요청 비용을 절감할 것을 고려해 보십시오.
- S3 Glacier 및 S3 Glacier Deep Archive 데이터 복원 요금 — S3 Glacier 및 S3 Glacier Deep Archive는 자주 액세스하지 않는 데이터의 장기 아카이브를 위해 설계되었습니다. 데이터 복원 요금에 대한 자세한 내용은 Amazon S3 FAQ에서 How Amazon S3 Glacier에서 데이터를 검색하는 데 드는 비용은 얼마인가요? 항목을 참조하십시오. Amazon S3 Glacier에서 데이터 복원 방법에 대한 자세한 내용은 보관된 객체의 복원 (p. 247) 항목을 참조하십시오.

S3 수명 주기 관리를 사용하여 Amazon S3 Glacier에 객체를 아카이브할 경우 Amazon S3는 이러한 객체를 비동기 방식으로 전환합니다. 수명 주기 구성 규칙의 전환 날짜와 실제 전환 작업 수행 날짜 간에 약간의 지연이 발생할 수 있습니다. 그러나 Amazon S3 Glacier 요금은 규칙에 지정된 이전 날짜를 기준으로 청구됩니다. 자세한 내용은 Amazon S3 FAQ의 Amazon S3 Glacier 단원을 참조하십시오.

Amazon S3 제품 세부 정보 페이지에 자세한 요금 정보와 Amazon S3 객체 보관의 요금 계산 예제가 나와 있습니다. 자세한 내용은 다음 항목을 참조하십시오.

- Amazon S3 Glacier에 보관된 Amazon S3 객체에 대한 나의 스토리지 요금은 어떻게 계산됩니까?
- Amazon S3 Glacier에서 3개월 미만의 객체를 삭제하면 요금이 어떻게 청구됩니까?
- Amazon S3 Glacier에서 데이터를 검색하는 데 드는 비용은 얼마인가요?
- 다양한 스토리지 클래스의 스토리지 비용에 대한 Amazon S3 요금입니다.

보관된 객체의 복원

보관된 객체에는 실시간으로 액세스할 수 없습니다. 먼저 복원 요청을 시작한 후 요청에 지정된 기간 동안 객체의 임시 사본을 사용할 수 있게 될 때까지 기다려야 합니다. 복원된 객체의 임시 복사본을 받으면 객체의 스토리지 클래스는 S3 Glacier 또는 S3 Glacier Deep Archive에 유지됩니다. HEAD 객체 또는 GET 객체 API 작업 요청은 S3 Glacier 또는 S3 Glacier Deep Archive 스토리지 클래스로 반환됩니다.

> **Note**
>
> 아카이브를 복원하면 아카이브(S3 Glacier 또는 S3 Glacier Deep Archive 요금) 및 임시로 복원한 복사본(Reduced Redundancy 스토리지 요금) 둘 다에 대해 요금이 청구됩니다. 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

프로그래밍 방식으로 또는 Amazon S3 콘솔을 사용하여 객체 사본을 복원할 수 있습니다. Amazon S3는 각 객체에 대해 한 번에 하나의 복원 요청만 처리합니다. 자세한 내용은 보관된 객체의 복원 (p. 247) 단원을 참조하십시오.

객체 만료 이해

객체가 수명 주기의 끝에 도달한 경우, Amazon S3에서는 제거를 위해 이를 대기열에 넣고 비동기 방식으로 제거합니다. 만료 날짜와 Amazon S3에서 객체를 제거하는 날짜 사이의 지연이 있을 수 있습니다. 만료된 객체와 관련된 스토리지 시간에 대해서는 요금이 청구되지 않습니다.

객체가 만료되도록 예약된 시기를 알아보려면 HEAD Object 또는 GET Object API 작업을 사용합니다. 이러한 API 작업은 이 정보를 제공하는 응답 헤더를 반환합니다.

S3 Intelligent-Tiering, S3 Standard-IA 또는 S3 One Zone-IA 스토리지에 30일 미만의 기간 동안 저장되어 있었던 객체를 만료 처리하는 S3 수명 주기 만료 규칙을 생성할 경우 30일에 해당하는 요금이 청구됩니다. S3 Glacier 스토리지에 90일 미만의 기간 동안 저장되어 있었던 객체를 만료 처리하는 수명 주기 만료 규칙을 생성할 경우 90일에 해당하는 요금이 청구됩니다. S3 Glacier Deep Archive 스토리지에 180일 미만의 기간 동안 저장되어 있었던 객체를 만료 처리하는 수명 주기 만료 규칙을 생성할 경우 180일에 해당하는 요금이 청구됩니다. 자세한 내용은 Amazon S3 요금을 참조하십시오.

수명 주기 및 기타 버킷 구성

S3 수명 주기 구성 외에 다른 구성도 버킷에 연결할 수 있습니다. 이 단원에서는 S3 수명 주기 구성과 다른 버킷 구성과의 관계를 설명합니다.

수명 주기 및 버전 관리

버전 관리를 사용하지 않는 버킷과 버전 관리가 활성화된 버킷에 S3 수명 주기 구성을 추가할 수 있습니다. 자세한 내용은 객체 버전 관리 (p. 112) 단원을 참조하십시오.

버전 관리가 활성화된 버킷은 하나의 현재 객체 버전과 0개 이상의 최신이 아닌 객체 버전을 유지합니다. 최신 객체 버전과 최신이 아닌 객체 버전에 대해 별도의 수명 주기 규칙을 정의할 수 있습니다.

자세한 내용은 수명 주기 구성의 요소 (p. 130) 단원을 참조하십시오. S3 버전 관리에 대한 자세한 내용은 객체 버전 관리 (p. 112) 단원을 참조하십시오.

MFA 활성화 버킷에 대한 수명 주기 구성

MFA(Multi-Factor Authentication) 활성화 버킷에 대한 수명 주기 구성은 지원되지 않습니다.

수명 주기 및 로깅

AWS CloudTrail 객체 수준 로깅으로는 Amazon S3 수명 주기 작업이 캡처되지 않습니다. CloudTrail이 외부 Amazon S3 엔드포인트에 대한 API 요청을 캡처하고, 내부 Amazon S3 엔드포인트를 사용하여 S3 수명 주기 작업이 수행되기 때문입니다. S3 버킷에서 Amazon S3 서버 액세스 로그를 활성화하여 다른 스토리지 클래스로의 객체 전환 및 영구적 삭제나 논리적 삭제를 유발하는 객체 만료와 같은 S3 수명 주기 관련 작업을 캡처할 수 있습니다. 자세한 내용은 서버 액세스 로깅 (p. 705) 단원을 참조하십시오.

버킷에서 로깅을 활성화한 경우 Amazon S3 서버 액세스 로그가 다음 작업의 결과를 보고합니다.

작업 로그	설명
`S3.EXPIRE.OBJECT`	수명 주기 만료 작업으로 인해 Amazon S3이 영구적으로 객체를 삭제합니다.
`S3.CREATE.DELETEMARKER`	Amazon S3가 현재 버전을 논리적으로 삭제하고, 버전 관리가 활성화된 버킷에서 삭제 마커를 추가합니다.
`S3.TRANSITION_SIA.OBJECT`	Amazon S3가 S3 Standard-IA 스토리지 클래스로 객체를 전환합니다.

작업 로그	설명
`S3.TRANSITION_ZIA.OBJECT`	Amazon S3가 S3 One Zone-IA 스토리지 클래스로 객체를 전환합니다.
`S3.TRANSITION_INT.OBJECT`	Amazon S3이 Intelligent-Tiering 스토리지 클래스로 객체를 이전합니다.
`S3.TRANSITION.OBJECT`	Amazon S3이 S3 Glacier 스토리지 클래스로 객체 전환을 시작합니다.
`S3.TRANSITION_GDA.OBJECT`	Amazon S3이 S3 Glacier S3 Glacier Deep Archive 스토리지 클래스로 객체 전환을 시작합니다.
`S3.DELETE.UPLOAD`	Amazon S3이 불완전한 멀티파트 업로드를 중단합니다.

> **Note**
>
> Amazon S3 서버 액세스 로그 레코드는 일반적으로 최대한 전달되며, 모든 Amazon S3 요청의 완벽한 기록에 사용할 수는 없습니다.

추가 정보

- 수명 주기 구성의 요소 (p. 130)
- S3 Glacier 및 S3 Glacier Deep Archive 스토리지 클래스로 전환(객체 보관) (p. 127)
- 버킷에서 수명 주기 구성 설정 (p. 146)

수명 주기 구성의 요소

주제

- ID 요소 (p. 131)
- 상태 요소 (p. 131)
- 필터 요소 (p. 131)
- 수명 주기 작업을 설명할 요소 (p. 133)

S3 수명 주기 구성을 하나 이상의 수명 주기 규칙으로 이루어진 XML로 지정합니다.

```
<LifecycleConfiguration>
    <Rule>
         ...
    </Rule>
    <Rule>
         ...
    </Rule>
</LifecycleConfiguration>
```

각 규칙은 다음과 같은 요소로 구성됩니다.

- 규칙 메타데이터에는 규칙 ID와 규칙의 활성화 여부를 나타내는 상태가 포함됩니다. 규칙이 비활성화되면 Amazon S3는 규칙에 지정된 작업을 수행하지 않습니다.
- 규칙이 적용되는 객체를 식별하는 필터. 객체 키 접두사, 하나 이상의 객체 태그 또는 이 두 가지를 모두 사용하여 필터를 지정할 수 있습니다.

- Amazon S3가 객체 수명 주기 상의 특정 날짜 또는 기간 동안 지정된 작업을 수행하기를 원하는 하나 이상의 전환 또는 만료 작업.

다음 단원에서는 S3 수명 주기 구성의 XML 요소에 대해 설명합니다. 구성에 대한 예시는 수명 주기 구성의 예제 (p. 136) 단원을 참조하십시오.

ID 요소

S3 수명 주기 구성에는 규칙을 최대 1,000개까지 포함할 수 있습니다. <ID> 요소는 각 규칙을 고유하게 식별합니다. ID 길이는 255자로 제한됩니다.

상태 요소

<Status> 요소 값은 Enabled 또는 Disabled입니다. 규칙이 비활성화되면 Amazon S3는 규칙에 정의된 작업을 수행하지 않습니다.

필터 요소

수명 주기 규칙은 수명 주기 규칙에서 지정하는 <Filter> 요소에 기반하여 버킷 내 객체의 모든 하위 집합 또는 하나의 하위 집합에 적용될 수 있습니다.

객체들을 키 접두사, 객체 태그 또는 이 둘의 조합으로 필터링할 수 있습니다(조합하는 경우, Amazon S3는 논리 AND를 사용하여 필터를 조합합니다). 다음 예제를 고려하십시오.

- 키 접두사를 사용하여 필터 지정 – 이 예제는 키 이름 접두사(`logs/`)에 기반하여 객체의 하위 집합에 적용되는 S3 수명 주기 규칙을 보여 줍니다. 예를 들어, 이 수명 주기 규칙은 객체 `logs/mylog.txt`, `logs/temp1.txt` 및 `logs/test.txt`에 적용됩니다. 이 규칙은 객체 `example.jpg`에는 적용되지 않습니다.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
           <Prefix>logs/</Prefix>
        </Filter>
        transition/expiration actions.
         ...
    </Rule>
    ...
</LifecycleConfiguration>
```

다른 키 이름 접두사에 기반하여 객체의 하위 집합에 수명 주기 작업을 적용하려면 별도의 규칙을 지정하십시오. 각각의 규칙에서 접두사 기반 필터를 지정하십시오. 예를 들어, 키 접두사 `projectA/`와 `projectB/`를 가진 객체의 수명 주기 작업을 설명하려면 아래와 같이 두 가지 규칙을 지정합니다.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
           <Prefix>projectA/</Prefix>
        </Filter>
        transition/expiration actions.
         ...
    </Rule>

    <Rule>
        <Filter>
           <Prefix>projectB/</Prefix>
        </Filter>
        transition/expiration actions.
         ...
```

```
    </Rule>
</LifecycleConfiguration>
```

객체 키에 대한 자세한 내용은 객체 키 (p. 103) 단원을 참조하십시오.

- 객체 태그에 기반한 필터 지정 – 다음 예제에서 수명 주기 규칙은 태그(*key*)와 값(*value*)에 기반하여 필터를 지정합니다. 그러면 규칙은 특정 태그를 가진 객체의 하위 집합에만 적용됩니다.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
           <Tag>
              <Key>key</Key>
              <Value>value</Value>
           </Tag>
        </Filter>
        transition/expiration actions.
        ...
    </Rule>
</LifecycleConfiguration>
```

여러 개의 태그를 기반으로 필터를 지정할 수 있습니다. 태그는 다음 예제에서 보듯 <AND> 요소로 감싸야 합니다. 규칙은 Amazon S3에게 2개의 태그(특정 태그 키와 값)를 가진 객체에서 수명 주기 작업을 수행하도록 지시합니다.

```
<LifecycleConfiguration>
    <Rule>
      <Filter>
         <And>
            <Tag>
               <Key>key1</Key>
               <Value>value1</Value>
            </Tag>
            <Tag>
               <Key>key2</Key>
               <Value>value2</Value>
            </Tag>
             ...
          </And>
      </Filter>
      transition/expiration actions.
    </Rule>
</Lifecycle>
```

수명 주기 규칙은 지정된 태그 2개가 모두 있는 객체에 적용됩니다. Amazon S3은 논리 AND를 수행합니다. 다음을 참조하십시오.

- 각 태그는 키 및 값과 정확히 일치해야 합니다.
- 이 규칙은 규칙에서 지정된 태그가 있는 객체의 하위 집합에 적용됩니다. 객체에 추가 태그가 지정되어 있더라도 상관없습니다.

  Note

  필터에서 복수의 태그를 지정할 경우, 각각의 태그 키가 고유해야 합니다.

- 접두사와 하나 이상의 태그에 기반한 필터 지정 – 수명 주기 규칙에서 키 접두사 및 하나 이상의 태그에 기반하여 필터를 지정할 수 있습니다. 이 경우에도 아래와 같이 이 모두를 <AND> 요소로 감싸야 합니다.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
          <And>
```

```
            <Prefix>key-prefix</Prefix>
            <Tag>
               <Key>key1</Key>
               <Value>value1</Value>
            </Tag>
            <Tag>
               <Key>key2</Key>
               <Value>value2</Value>
            </Tag>
             ...
          </And>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions.
    </Rule>
</LifecycleConfiguration>
```

Amazon S3는 논리 AND를 사용하여 이 필터들을 조합합니다. 즉, 규칙은 특정 키 접두사와 특정 태그를 가진 객체의 하위 집합에 적용됩니다. 필터는 단 1개의 접두사와 0개 이상의 태그를 가질 수 있습니다.

- 빈 필터를 지정할 수도 있는데, 이 경우에 규칙은 버킷 내 모든 객체에 적용됩니다.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions.
    </Rule>
</LifecycleConfiguration>
```

수명 주기 작업을 설명할 요소

S3 수명 주기 규칙에 사전 정의된 다음 작업 중 하나 이상을 지정하여 객체의 수명 주기에 따라 Amazon S3이 특정 작업을 수행하도록 지시할 수 있습니다. 이러한 작업의 효과는 버킷의 버전 관리 상태에 따라 달라집니다.

- 전환 작업 요소 – 한 스토리지 클래스에서 다른 스토리지 클래스로 객체를 전환하려면 작업을 지정합니다. 객체 전환에 대한 자세한 내용은 지원되는 전환 작업 및 관련 제한 사항 (p. 124) 단원을 참조하십시오. 객체의 수명 주기에서 지정된 날짜 또는 기간이 도래하면 Amazon S3가 전환을 수행합니다.

  버전이 지정된 버킷(버전 관리를 사용하거나 버전 관리가 일시 중지된 버킷)의 경우 `Transition` 작업은 현재 객체 버전에 적용됩니다. 비 최신 버전을 관리하기 위해 Amazon S3에서 `NoncurrentVersionTransition` 작업을 정의합니다(아래 설명 참조).

- 만료 작업 요소 – `Expiration` 작업은 규칙에 지정된 객체를 만료시키고 Amazon S3 스토리지 클래스의 적격 객체에 적용합니다. 스토리지 클래스에 대한 자세한 정보는 Amazon S3 스토리지 클래스 (p. 107) 단원을 참조하십시오. Amazon S3는 만료된 모든 객체를 사용할 수 없게 만들고, 버킷의 버전 관리 상태에 따라 객체의 영구 삭제 여부가 결정됩니다.

  > **Important**
  >
  > 객체 만료 수명 주기 정책은 불완전 멀티파트 업로드업로드를 제거하지 않습니다. 불완전 멀티파트 업로드업로드를 제거하려면 이 단원 뒷부분에서 설명하는 AbortIncompleteMultipartUpload 수명 주기 구성 작업을 사용해야 합니다.

  - 버전 관리를 사용하지 않는 버킷 – `Expiration` 작업을 수행하면 Amazon S3에서는 객체를 영구적으로 제거합니다.

- 버전이 지정된 버킷 – 버전이 지정된 버킷(즉, 버전 관리를 사용하거나 버전 관리가 일시 중지된 버킷)의 경우, Amazon S3의 `expiration` 작업 처리 방식의 지침이 되는 몇 가지 고려 사항이 있습니다. 자세한 내용은 버전 관리 사용 (p. 468) 단원을 참조하십시오. 버전 관리 상태에 상관없이 다음 사항이 적용됩니다.
  - `Expiration` 작업은 최신 버전에만 적용됩니다(비 최신 버전에는 영향이 없음).
  - 객체 버전이 1개 이상이고 삭제 마커가 최신 버전이면 Amazon S3는 작업을 수행하지 않습니다.
  - 최신 객체 버전이 유일한 객체 버전이면서 삭제 마커인 경우(만료된 객체 삭제 마커라고도 하는데, 이 경우 모든 객체 버전이 삭제되고 하나의 삭제 마커만 남게 됨), Amazon S3는 만료된 객체 삭제 마커를 제거합니다. 만료 작업을 사용하여 Amazon S3에게 만료된 객체 삭제 마커를 제거하도록 지시할 수도 있습니다. 문제 해결 예는 예제 7: 만료된 객체 삭제 마커의 제거 (p. 144) 단원을 참조하십시오.

  Amazon S3를 설정하여 만료를 관리할 때는 다음 사항도 고려하십시오.

  - 버전 관리를 사용하는 버킷

    최신 객체 버전이 삭제 마커가 아닌 경우 Amazon S3는 고유한 버전 ID를 갖는 삭제 마커를 추가합니다. 이를 통해 최신 버전이 최신이 아닌 버전이 되고 삭제 마커가 최신 버전이 됩니다.
  - 버전 관리가 일시 중지된 버킷

    버전 관리가 일시 중지된 버킷에서는 만료 작업으로 인해 Amazon S3는 버전 ID가 null인 삭제 마커를 생성합니다. 삭제 마커는 버전 계층 구조에서 null 버전 ID로 모든 객체 버전을 대체함으로써 결과적으로 객체를 삭제합니다.

또한 Amazon S3에서는 버전이 지정된 버킷(즉, 버전 관리를 사용하거나 버전 관리가 일시 중지된 버킷)에서 비 최신 객체 버전을 관리하는 데 사용할 수 있는 다음 작업을 제공합니다.

- NoncurrentVersionTransition 작업 요소 – 이 작업을 사용하면 Amazon S3에서 객체를 지정된 스토리지 클래스로 이전하기 전 현재 스토리지 클래스에 유지할 기간(객체가 비 최신 버전이 된 시간부터)을 지정할 수 있습니다. 객체 전환에 대한 자세한 내용은 지원되는 전환 작업 및 관련 제한 사항 (p. 124) 단원을 참조하십시오.
- NoncurrentVersionExpiration 작업 요소 – 이 작업을 사용하면 Amazon S3가 객체를 영구 삭제하기 전 비 최신 버전의 객체를 유지할 기간(객체가 비 최신 버전이 된 시간부터)을 지정할 수 있습니다. 영구 삭제된 객체는 다시 복구할 수 없습니다.

  최신 버전이 아닌 객체를 이렇게 지연 제거하면 실수로 인한 삭제 또는 덮어쓰기로부터 복구해야 하는 경우에 유용합니다. 예를 들어, 비 최신 버전이 되고 5일 이후에 비 최신 버전을 삭제하도록 만료 규칙을 구성할 수 있습니다. 예를 들어, 2014년 1월 1일 오전 10시 30분(UTC)에 `photo.gif`(버전 ID 111111)라고 하는 객체를 생성한 경우, 2014년 1월 2일 오전 11시 30분에 실수로 `photo.gif`(버전 ID 111111)를 삭제하면 새 버전 ID(예: 버전 ID 4857693)와 함께 삭제 마커가 생성됩니다. 이제 5일 동안 원래 버전의 `photo.gif`(버전 ID 111111)를 복구할 수 있으며, 5일이 지나면 영구 삭제됩니다. 비 최신 버전이 된 지 5일 후인 2014년 1월 8일 0시에 만료 수명 주기 규칙이 실행되어 `photo.gif`(버전 ID 111111)를 영구 삭제합니다.

  > **Important**
  >
  > 객체 만료 수명 주기 정책은 불완전 멀티파트 업로드업로드를 제거하지 않습니다. 불완전 멀티파트 업로드업로드를 제거하려면 이 단원 뒷부분에서 설명하는 AbortIncompleteMultipartUpload 수명 주기 구성 작업을 사용해야 합니다.

전환 및 만료 작업뿐만 아니라 다음과 같은 수명 주기 구성 작업을 사용함으로써 Amazon S3에게 불완전 멀티파트 업로드업로드를 중단하도록 지시할 수 있습니다.

- AbortIncompleteMultipartUpload 작업 요소 – 이 요소를 사용해 멀티파트 업로드가 진행 상태에 있길 원하는 최대 시간(일수)를 설정합니다. 해당 멀티파트 업로드(수명 주기 규칙에서 지정된 키 이름 `prefix`에 의해 결정)가 사전 지정된 기간 내에 성공적으로 완료되지 않으면, Amazon S3은 불완전 멀티파트 업로드를

중단합니다. 자세한 내용은 버킷 수명 주기 정책을 사용한 미완료 멀티파트 업로드 중단 (p. 180) 단원을 참조하십시오.

Note

객체 태그에 기반하여 필터를 지정하는 규칙에서는 이 수명 주기 작업을 지정할 수 없습니다.

- ExpiredObjectDeleteMarker 작업 요소 – 버전 관리를 사용하는 버킷에서 비 최신 버전이 없는 삭제 마커는 만료된 객체 삭제 마커라고 부릅니다. 이 수명 주기 작업을 사용하여 S3에게 만료된 객체 삭제 마커를 제거하도록 지시할 수 있습니다. 문제 해결 예는 예제 7: 만료된 객체 삭제 마커의 제거 (p. 144) 단원을 참조하십시오.

  Note

  객체 태그에 기반하여 필터를 지정하는 규칙에서는 이 수명 주기 작업을 지정할 수 없습니다.

Amazon S3가 객체가 비 최신 버전이 된 기간을 계산하는 방법

버전 관리를 사용하는 버킷에는 항상 하나의 최신 버전 객체와 함께 0 또는 비 최신 버전의 여러 객체 버전이 존재할 수 있습니다. 새로 객체를 업로드할 때마다 새로 업로드된 후임 버전이 최신 버전이 되고, 기존의 최신 버전은 비 최신 버전이 됩니다. 객체가 비 최신 버전이 된 일수를 계산하기 위해 Amazon S3는 후임 객체가 생성된 날짜를 확인합니다. Amazon S3는 후임 객체의 생성 일수를 사용하여 객체가 비 최신 버전이 된 일수를 계산합니다.

수명 주기 구성을 사용한 이전 버전의 객체 복원

이전 버전 복원 (p. 486) 항목의 설명과 같이 다음 두 가지 방법 중 하나를 사용하여 이전 버전의 객체를 복구할 수 있습니다.

1. 비 최신 버전의 객체를 동일한 버킷으로 복사합니다. 복사된 객체는 해당 객체의 최신 버전이 되고 모든 객체 버전은 유지됩니다.
2. 객체의 최신 버전을 영구 삭제합니다. 최신 객체 버전을 삭제하는 것은 실제로 해당 객체의 비 최신 버전을 최신 버전으로 만드는 것과 같습니다.

버전 관리를 사용하는 버킷에 S3 수명 주기 구성 규칙을 사용할 경우 첫 번째 방법을 사용하는 것이 모범 사례입니다.
Amazon S3의 최종 일관성에 따라 영구 삭제한 최신 버전은 이 변경 사항이 반영될 때까지 사라지지 않을 수 있습니다(Amazon S3이 이 삭제를 인식하지 못할 수 있음). 그 동안 비 최신 버전 객체를 만료시키도록 구성된 수명 주기 규칙에 의해 복구하려는 객체를 포함하여 비 최신 버전 객체가 영구 제거될 수 있습니다. 따라서 첫 번째 방법에서 권장하는 대로 이전 버전을 복사하는 것이 더 안전합니다.

수명 주기 규칙: 객체 기간 기반

객체 생성 또는 수정을 기준으로 Amazon S3가 작업을 수행할 수 있는 일수를 지정할 수 있습니다.

S3 수명 주기 구성에서 `Transition` 및 `Expiration` 작업의 일 수를 지정할 경우, 다음에 유의하십시오.

- 작업이 이루어질 객체 생성 이후의 일수입니다.
- Amazon S3는 객체 생성 시간을 기준으로 규칙에 지정된 일수를 계산하며, 익일 자정으로 계산된 시간을 상향 조정합니다. 예를 들어, 객체가 2014년 1월 15일 오전 10시 30분에 생성되고, 이전 규칙에 3일을 지정한 경우 객체의 이전일은 2014년 1월 19일 0시가 됩니다.

Note

Amazon S3는 각 객체에 대해 최종 수정일만 유지합니다. Amazon S3 콘솔은 객체의 속성 창에 최종 수정일을 표시합니다. 새 객체를 처음 생성할 때 이 날짜는 객체가 생성된 날짜가 되고, 객체를 변경하면 날짜도 이에 따라 바뀝니다. 따라서 생성일은 최종 수정일과 같은 날짜가 됩니다.

수명 주기 구성에서 `NoncurrentVersionTransition` 및 `NoncurrentVersionExpiration` 작업의 일수를 지정할 경우 다음에 유의하십시오.

- Amazon S3에서 지정된 객체에 대해 작업을 수행할 기간으로 객체 버전이 비 최신 버전이 된 이후의 일수(즉, 객체를 덮어쓰거나 삭제한 후 경과한 일수)를 지정합니다.
- Amazon S3는 후임 버전의 새 객체가 생성된 시간을 기준으로 규칙에 지정된 일수를 계산하며, 익일 자정으로 계산된 시간을 상향 조정합니다. 예를 들어 버킷에서 2014년 1월 1일 오전 10:30 UTC에 생성된 객체가 현재 버전이라고 가정합시다. 현재 버전을 대체한 새 버전이 2014년 1월 15일 오전 10:30 UTC에 생성되었고 전환 규칙에 3일을 지정했다면, 해당 객체의 전환 날짜는 2014년 1월 19일 00:00 UTC로 계산됩니다.

수명 주기 규칙: 지정된 날짜 기반

S3 수명 주기 규칙에서 작업을 지정할 때 Amazon S3에서 조치를 취하기를 원하는 날짜를 지정할 수 있습니다. 특정 날짜가 되면 Amazon S3이 필터 기준에 부합하는 모든 객체에 작업을 적용합니다.

과거 날짜를 사용하여 S3 수명 주기 작업을 지정할 경우 기준에 부합하는 모든 객체가 해당 수명 주기 작업에 대해 즉시 적격 상태가 됩니다.

> **Important**
>
> 날짜 기반 작업은 일회성 작업이 아닙니다. Amazon S3은 날짜가 경과한 이후에도 해당 규칙 상태가 `Enabled`로 유지되는 한 날짜 기반 작업을 계속 적용합니다.
> 예를 들어, 날짜 기반 `Expiration` 작업을 지정하여 모든 객체를 삭제한다고 가정해 봅시다(규칙에 지정된 필터가 없는 것으로 가정). 지정된 날짜에 Amazon S3은 버킷 내 모든 객체를 만료시킵니다. 또한 S3는 사용자가 버킷에서 만드는 새 객체도 모두 만료시킵니다. 수명 주기 작업을 중지하려면 작업을 수명 주기 구성에서 제거하거나, 규칙을 해제하거나, 규칙을 수명 주기 구성에서 삭제해야 합니다.

날짜 값은 ISO 8601 형식을 준수해야 합니다. 시간은 항상 자정(UTC)입니다.

> **Note**
>
> Amazon S3 콘솔을 사용하여 날짜 기반 수명 주기 규칙을 만들 수는 없지만 그러한 규칙을 조회, 비활성화 또는 삭제할 수는 있습니다.

수명 주기 구성의 예제

이 단원에서는 S3 수명 주기 구성의 예제를 제공합니다. 각 예제에서는 예제 시나리오별로 XML을 지정하는 방법을 보여줍니다.

주제

- 예 1: 필터 지정 (p. 136)
- 예 2: 수명 주기 규칙 비활성화 (p. 138)
- 예 3: 객체의 수명 주기 동안 스토리지 클래스 티어 다운 (p. 139)
- 예 4: 여러 규칙 지정 (p. 139)
- 예 5: 중복 필터, 서로 충돌하는 수명 주기 작업 및 Amazon S3가 수행하는 작업 (p. 140)
- 예 6: 버전 관리를 사용하는 버킷에 대한 수명 주기 규칙 지정 (p. 143)
- 예제 7: 만료된 객체 삭제 마커의 제거 (p. 144)
- 예 8: 멀티파트 업로드 중단을 위한 수명 주기 구성 (p. 145)

예 1: 필터 지정

각 S3 수명 주기 규칙에는 수명 주기 규칙이 적용되는 버킷 내 객체의 하위 집합을 식별하는 데 사용할 수 있는 필터가 포함되어 있습니다. 다음 S3 수명 주기 구성은 필터를 지정하는 방법에 대한 예제입니다.

- 이 수명 주기 구성 규칙에서 필터는 키 접두사(`tax/`)를 지정합니다. 따라서 이 규칙은 `tax/doc1.txt` 및 `tax/doc2.txt`와 같은 키 이름 접두사 `tax/`가 있는 객체에 적용됩니다.

  이 규칙은 Amazon S3에게 다음을 수행하도록 지시하는 두 가지 작업을 지정합니다.
  - 생성 후 365일(1년)이 지난 객체를 S3 Glacier 스토리지 클래스로 전환합니다.
  - 생성 후 3,650일(10년)이 지난 객체를 삭제합니다(`Expiration` 작업).

```
<LifecycleConfiguration>
  <Rule>
    <ID>Transition and Expiration Rule</ID>
    <Filter>
       <Prefix>tax/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>365</Days>
      <StorageClass>S3 Glacier</StorageClass>
    </Transition>
    <Expiration>
      <Days>3650</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

  생성 후 객체 기간을 일수로 지정하는 대신 각 작업에 날짜를 지정할 수 있습니다. 하지만 동일한 규칙에서 `Date`와 `Days`를 모두 사용할 수는 없습니다.

- 수명 주기 규칙을 버킷의 모든 객체에 적용하려면 빈 접두사를 지정합니다. 다음 구성에서 이 규칙은 생성 후 0일이 지난 객체를 S3 Glacier 스토리지 클래스로 이전하도록 Amazon S3에 요청하는 `Transition` 작업을 지정합니다. 이 경우 객체는 생성 후 당일 자정(UTC)이 되면 Amazon S3 Glacier에 보관될 수 있습니다.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Archive all object same-day upon creation</ID>
    <Filter>
      <Prefix></Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>S3 Glacier</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

- 0개 이상의 키 이름 접두사, 0개 이상의 객체 태그를 필터에 지정할 수 있습니다. 다음 예제 코드에서는 키 접두사 `tax/`가 있는 객체의 하위 집합과 특정 키 및 값을 가진 태그 두 개가 있는 객체에 수명 주기 규칙을 적용합니다. 하나 이상의 필터를 지정하는 경우, 예시에서와 같이 AND를 포함해야 한다는 점에 유의하십시오(Amazon S3는 논리적 AND를 적용하여 지정된 필터 조건을 결합함).

```
...
<Filter>
   <And>
      <Prefix>tax/</Prefix>
      <Tag>
         <Key>key1</Key>
         <Value>value1</Value>
      </Tag>
      <Tag>
         <Key>key2</Key>
```

```
            <Value>value2</Value>
         </Tag>
      </And>
</Filter>
...
```

- 태그에만 기반을 두어 객체를 필터링할 수 있습니다. 예를 들어, 아래의 수명 주기 규칙은 지정된 태그가 두 개인 객체에 적용됩니다(접두사는 지정하지 않음).

```
...
<Filter>
   <And>
      <Tag>
         <Key>key1</Key>
         <Value>value1</Value>
      </Tag>
      <Tag>
         <Key>key2</Key>
         <Value>value2</Value>
      </Tag>
    </And>
</Filter>
...
```

Important

S3 수명 주기 구성에 규칙이 여러 개인 경우 객체는 여러 수명 주기 작업을 수행할 수 있습니다. 이러한 경우 Amazon S3는 다음과 같은 일반적인 규칙을 따릅니다.

- 영구 삭제는 전환에 우선합니다.
- 전환은 삭제 마커 생성에 우선합니다.
- 객체에서 S3 Glacier 및 S3 Standard-IA(또는 S3 One Zone-IA) 전환을 모두 사용할 수 있는 경우 Amazon S3는 S3 Glacier 전환을 선택합니다.

예제는 예 5: 중복 필터, 서로 충돌하는 수명 주기 작업 및 Amazon S3가 수행하는 작업 (p. 140)을 참조하십시오.

예 2: 수명 주기 규칙 비활성화

수명 주기 규칙을 일시적으로 비활성화할 수 있습니다. 다음 수명 주기 구성에는 두 가지 규칙을 지정합니다.

- 규칙 1은 Amazon S3에게 객체 생성 직후 `logs/` 접두사가 있는 객체를 S3 Glacier 스토리지 클래스로 전환하도록 지시합니다.
- 규칙 2는 Amazon S3에게 객체 생성 직후 `documents/` 접두사가 있는 객체를 S3 Glacier 스토리지 클래스로 전환하도록 지시합니다.

정책에서 규칙 1은 활성화되고 규칙 2는 비활성화됩니다. Amazon S3는 비활성화된 규칙에 대해서는 아무 작업도 수행하지 않습니다.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule1</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
```

```
      <Days>0</Days>
      <StorageClass>S3 Glacier</StorageClass>
    </Transition>
  </Rule>
  <Rule>
    <ID>Rule2</ID>
    <Prefix>documents/</Prefix>
    <Status>Disabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>S3 Glacier</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

예 3: 객체의 수명 주기 동안 스토리지 클래스 티어 다운

이 예제에서는 수명 주기 구성을 사용하여 수명 주기 동안 객체의 스토리지 클래스를 계층 다운합니다. 티어 다운을 사용하면 스토리지 비용을 줄일 수 있습니다. 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

다음 S3 수명 주기 구성에는 키 이름 접두사가 `logs/`인 객체에 적용되는 규칙을 지정합니다. 이 규칙은 다음 작업을 지정합니다.

- 두 전환 작업:
  - 생성 후 30일이 지난 객체를 S3 Standard-IA 스토리지 클래스로 전환합니다.
  - 생성 후 90일이 지난 객체를 S3 Glacier 스토리지 클래스로 전환합니다.
- 생성 후 1년이 지난 객체를 삭제하도록 Amazon S3에 지시하는 만료 작업 1개.

```
<LifecycleConfiguration>
  <Rule>
    <ID>example-id</ID>
    <Filter>
       <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>STANDARD_IA</StorageClass>
    </Transition>
    <Transition>
      <Days>90</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

> **Note**
>
> 모든 작업이 필터로 식별된 동일한 객체 집합에 적용되는 경우 한 개의 규칙으로 모든 수명 주기 작업을 설명할 수 있습니다. 또는 각각 다른 필터를 지정하는 여러 규칙을 추가할 수 있습니다.

예 4: 여러 규칙 지정

객체마다 다른 수명 주기 작업을 원할 경우 여러 규칙을 지정할 수 있습니다. 다음 수명 주기 구성에는 두 가지 규칙이 있습니다.

- 규칙 1은 키 이름 접두사가 `classA/`인 객체에 적용되며, Amazon S3에게 생성 후 1년이 지난 객체를 S3 Glacier 스토리지 클래스로 전환하고 생성 후 10년이 지난 객체를 만료하도록 지시합니다.
- 규칙 2는 키 이름 접두사가 `classB/`인 객체에 적용되며, 생성 후 90일이 지난 객체를 S3 Standard-IA 스토리지 클래스로 전환하고 생성 후 1년이 지난 객체를 삭제하도록 Amazon S3에 지시합니다.

```
<LifecycleConfiguration>
    <Rule>
        <ID>ClassADocRule</ID>
        <Filter>
           <Prefix>classA/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
           <Days>365</Days>
           <StorageClass>GLACIER</StorageClass>
        </Transition>
        <Expiration>
             <Days>3650</Days>
        </Expiration>
    </Rule>
    <Rule>
        <ID>ClassBDocRule</ID>
        <Filter>
            <Prefix>classB/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
           <Days>90</Days>
           <StorageClass>STANDARD_IA</StorageClass>
        </Transition>
        <Expiration>
             <Days>365</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

예 5: 중복 필터, 서로 충돌하는 수명 주기 작업 및 Amazon S3가 수행하는 작업

S3 수명 주기 구성을 지정하여 중복 접두사 또는 작업을 지정할 수 있습니다.

일반적으로 Amazon S3 수명 주기는 비용을 최적화합니다. 예를 들어, 두 만료 정책이 겹치는 경우 데이터가 예상보다 오래 저장되지 않도록 더 짧은 만료 정책이 적용됩니다.

마찬가지로, 두 전환 정책이 겹치는 경우 S3 수명 주기는 객체를 더 저렴한 비용의 스토리지 클래스로 전환합니다. 두 경우 모두 S3 수명 주기는 가장 비용이 적게 드는 경로를 선택하려고 시도합니다. 이 일반적인 규칙의 예외로는 S3 Intelligent-Tiering 스토리지 클래스가 있습니다. S3 수명 주기에서는 S3 Glacier 및 S3 Glacier Deep Archive 스토리지 클래스를 제외한 모든 스토리지 클래스에 비해 S3 Intelligent-Tiering을 선호합니다.

다음 예시에서는 Amazon S3가 어떤 방식으로 잠재적인 충돌을 해결하는지 보여줍니다.

Example 1: 중복 접두사(충돌 없음)

다음에서 예로 든 구성에는 중복 접두사를 지정하는 규칙이 두 개 있습니다.

- 첫 번째 규칙은 버킷의 모든 객체를 의미하는 빈 필터를 지정합니다.
- 두 번째 규칙은 버킷에서 객체의 하위 집합만을 의미하는 `logs/`라는 키 이름 접두사를 지정합니다.

규칙 1은 생성 후 1년이 지난 모든 객체를 삭제하도록 Amazon S3에 요청합니다. 규칙 2는 생성 후 30일이 지난 객체의 하위 집합을 S3 Standard-IA 스토리지 클래스로 전환하도록 Amazon S3에 요청합니다.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
    </Filter>
    <Status>Enabled</Status>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
  <Rule>
    <ID>Rule 2</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA<StorageClass>
      <Days>30</Days>
    </Transition>
   </Rule>
</LifecycleConfiguration>
```

Example 2: 서로 충돌하는 수명 주기 작업

이 구성 예시에는 객체 수명 주기 중 같은 시각에 동일한 객체 집합에 대해 두 가지 작업을 수행하도록 Amazon S3에게 지시하는 규칙이 두 개 있습니다.

- 두 규칙은 동일한 키 이름 접두사를 지정하므로, 두 규칙은 동일한 객체 집합에 적용됩니다.
- 두 규칙은, 규칙이 적용될 때 생성 후 365일이 지난 동일한 객체를 지정합니다.
- 한 규칙은 객체를 S3 Standard-IA 스토리지 클래스로 전환하도록 Amazon S3에 지시하고, 이와 동시에 다른 규칙은 객체를 만료 처리하도록 Amazon S3에 지시합니다.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
  <Rule>
    <ID>Rule 2</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA<StorageClass>
      <Days>365</Days>
    </Transition>
   </Rule>
</LifecycleConfiguration>
```

이 경우, 객체가 만료(제거)되기를 원하는 것이므로 스토리지 클래스를 변경하는 것은 의미가 없습니다. 따라서 Amazon S3는 이 객체들에 대해 만료 작업만 선택합니다.

Example 3: 중복 접두사로 인해 서로 충돌하는 수명 주기 작업

이 예제에서 구성에는 다음과 같이 중복 접두사를 지정하는 두 가지 규칙이 있습니다.

- 규칙 1은 빈 접두사(모든 객체를 의미)를 지정합니다.
- 규칙 2는 모든 객체의 하위 집합을 식별하는 키 이름 접두사(`logs/`)를 지정합니다.

키 이름 접두사가 `logs/`인 객체의 하위 집합에는 두 규칙의 수명 주기 작업이 적용됩니다. 한 규칙은 Amazon S3에게 생성 후 10일이 지난 객체를 이전하도록 지시하고, 다른 규칙은 Amazon S3에게 생성 후 365일이 지난 객체를 이전하도록 지시합니다.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
      <Prefix></Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA<StorageClass>
      <Days>10</Days>
    </Transition>
  </Rule>
  <Rule>
    <ID>Rule 2</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA<StorageClass>
      <Days>365</Days>
    </Transition>
   </Rule>
</LifecycleConfiguration>
```

이 경우, Amazon S3는 생성 후 10일 지난 객체를 이전하는 쪽을 선택합니다.

Example 4: 태그 기반 필터링과 그로 인해 서로 충돌하는 수명 주기 작업

다음과 같이 각각 태그 필터를 지정하는 두 가지 규칙을 가진 S3 수명 주기 정책이 있다고 가정합니다.

- 규칙 1은 태그 기반 필터(`tag1/value1`)를 지정합니다. 이 규칙은 Amazon S3에게 생성 후 365일이 지난 객체를 S3 Glacier 스토리지 클래스로 전환하도록 지시합니다.
- 규칙 2는 태그 기반 필터(`tag2/value2`)를 지정합니다. 이 규칙은 Amazon S3에게 생성 후 14일이 지난 객체를 만료 처리하도록 지시합니다.

수명 주기 구성은 다음과 같습니다.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
      <Tag>
         <Key>tag1</Key>
```

```
          <Value>value1</Value>
        </Tag>
      </Filter>
      <Status>Enabled</Status>
      <Transition>
        <StorageClass>GLACIER<StorageClass>
        <Days>365</Days>
      </Transition>
    </Rule>
    <Rule>
      <ID>Rule 2</ID>
      <Filter>
        <Tag>
           <Key>tag2</Key>
           <Value>value1</Value>
        </Tag>
      </Filter>
      <Status>Enabled</Status>
      <Expiration>
        <Days>14</Days>
      </Expiration>
     </Rule>
</LifecycleConfiguration>
```

정책은 좋습니다. 그러나 두 태그가 모두 있는 객체가 있는 경우, S3는 어떤 작업을 할지 결정해야 합니다. 즉, 두 규칙 모두 한 객체에 적용되므로 결과적으로 서로 충돌하는 작업을 수행하도록 Amazon S3에 지시하는 셈이 됩니다. 이 경우, Amazon S3는 생성 후 14일 지난 객체를 만료 처리합니다. 그 객체가 제거되면, 이에 따라 전환 작업은 실행되지 않습니다.

예 6: 버전 관리를 사용하는 버킷에 대한 수명 주기 규칙 지정

버전 관리를 사용하는 버킷이 있다고 가정합니다. 즉, 각 객체에 대해 현재 버전과 0 버전 이상의 여러 버전이 있음을 의미합니다. 기록을 1년 동안 유지한 후 최신이 아닌 버전을 삭제하고자 합니다. For more information about S3 버전 관리, see 객체 버전 관리 (p. 112).

또한 비 최신 버전(실시간 액세스가 필요 없는 콜드 데이터라고 간주)이 된 후 30일이 지나면 비 최신 버전을 S3 Glacier로 이동하여 스토리지 비용을 절감하려고 합니다. 또한 생성 후 90일이 지나면 최신 버전에 대한 액세스 빈도가 감소할 것으로 예상하여 해당 객체를 S3 Standard-IA 스토리지 클래스로 이동하도록 선택할 수 있습니다.

```
<LifecycleConfiguration>
    <Rule>
        <ID>sample-rule</ID>
        <Filter>
           <Prefix></Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
           <Days>90</Days>
           <StorageClass>STANDARD_IA</StorageClass>
        </Transition>
        <NoncurrentVersionTransition>
            <NoncurrentDays>30</NoncurrentDays>
            <StorageClass>S3 Glacier</StorageClass>
        </NoncurrentVersionTransition>
       <NoncurrentVersionExpiration>
            <NoncurrentDays>365</NoncurrentDays>
       </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

예제 7: 만료된 객체 삭제 마커의 제거

버전 관리를 사용하는 버킷에는 최신 버전의 객체 1개와 각 객체에 대한 0개 이상의 비 최신 버전이 존재합니다. 객체를 삭제할 때는 다음 사항에 유의하십시오.

- 삭제 요청에서 버전 ID를 지정하지 않는 경우, Amazon S3에서는 그 객체를 삭제하는 대신 삭제 마커를 추가합니다. 최신 객체 버전이 최신이 아닌 버전이 된 후에는 삭제 마커가 최신 버전이 됩니다.
- 삭제 요청에서 버전 ID를 지정하는 경우, Amazon S3에서는 그 객체 버전을 영구적으로 삭제합니다(삭제 마커는 생성되지 않습니다).
- 비 최신 버전이 없는 삭제 마커는 만료된 객체 삭제 마커라고 부릅니다.

이 예제에서는 버킷에 만료된 객체 삭제 마커를 생성할 수 있는 시나리오, 그리고 S3 수명 주기 구성을 사용해 만료된 객체 삭제 마커를 제거하도록 Amazon S3에 지시하는 방법을 보여줍니다.

다음과 같이 최신이 아닌 버전이 30일 후에 제거되도록 `NoncurrentVersionExpiration` 작업을 지정하는 수명 주기 정책을 작성한다고 가정합니다.

```
<LifecycleConfiguration>
    <Rule>
        ...
        <NoncurrentVersionExpiration>
            <NoncurrentDays>30</NoncurrentDays>
        </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

`NoncurrentVersionExpiration` 작업은 현재 객체 버전에는 적용되지 않습니다. 비 현재 버전만 제거합니다.

최신 객체 버전의 경우 최신 객체 버전이 잘 정의된 수명 주기를 따르는지 여부에 따라 수명 주기를 관리할 수 있는 옵션이 다음과 같이 제공됩니다.

- 객체의 최신 버전은 잘 정의된 수명 주기를 따릅니다.

  이 경우 `Expiration` 작업과 함께 수명 주기 정책을 사용하여 다음 예제와 같이 최신 버전을 제거하도록 Amazon S3에 지시할 수 있습니다.

  ```
  <LifecycleConfiguration>
      <Rule>
          ...
          <Expiration>
             <Days>60</Days>
          </Expiration>
          <NoncurrentVersionExpiration>
              <NoncurrentDays>30</NoncurrentDays>
          </NoncurrentVersionExpiration>
      </Rule>
  </LifecycleConfiguration>
  ```

  Amazon S3는 최신 버전의 객체 각각에 삭제 마커를 추가함으로써 생성된 지 60일이 지난 최신 버전을 제거합니다. 이를 통해 최신 버전이 최신이 아닌 버전이 되고 삭제 마커가 최신 버전이 됩니다. 자세한 내용은 버전 관리 사용 (p. 468) 단원을 참조하십시오.

  Note

  이 규칙은 버전이 지정된 버킷에서 자동으로 `ExpiredObjectDeleteMarker` 정리를 수행하므로 `ExpiredObjectDeleteMarker` 태그를 포함할 필요가 없습니다.

동일한 수명 주기 구성의 `NoncurrentVersionExpiration` 작업은 최신 버전이 아닌 30일이 지난 객체를 제거합니다. 따라서 이 예제에서는 객체 생성 후 90일이 지나면 모든 객체 버전이 영구적으로 제거됩니다. 만료된 객체 삭제 마커가 있으면 Amazon S3는 만료된 객체 삭제 마커를 자동 감지하여 제거합니다.

- 객체의 최신 버전은 잘 정의된 수명 주기를 따르지 않습니다.

  이 경우 그 객체들이 필요 없는 경우 1개 이상의 비 최신 버전에 삭제 마커를 생성하여 수동으로 제거할 수 있습니다. 수명 주기 구성에 `NoncurrentVersionExpiration` 작업을 사용하여 최신이 아닌 버전을 모두 제거하면 만료된 객체 삭제 마커가 생깁니다.

  특히 이러한 시나리오의 경우 Amazon S3 수명 주기 구성이 `Expiration` 작업을 제공하므로, 여기서 만료된 객체 삭제 마커를 제거하도록 Amazon S3에 요청할 수 있습니다.

```
<LifecycleConfiguration>
    <Rule>
       <ID>Rule 1</ID>
        <Filter>
          <Prefix>logs/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Expiration>
           <ExpiredObjectDeleteMarker>true</ExpiredObjectDeleteMarker>
        </Expiration>
        <NoncurrentVersionExpiration>
            <NoncurrentDays>30</NoncurrentDays>
        </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

`Expiration` 작업에서 `ExpiredObjectDeleteMarker` 요소를 true로 설정함으로써 Amazon S3에게 만료된 객체 삭제 마커를 제거하도록 지시합니다.

> **Note**
>
> `ExpiredObjectDeleteMarker` 수명 주기 작업을 지정하면 이 규칙은 태그 기반 필터를 지정할 수 없습니다.

예 8: 멀티파트 업로드 중단을 위한 수명 주기 구성

멀티파트 업로드 API를 사용하면 대형 객체를 여러 부분으로 나누어 업로드할 수 있습니다. 멀티파트 업로드에 대한 자세한 내용은 멀티파트 업로드 개요 (p. 178) 단원을 참조하십시오.

멀티파트 업로드 시작 후 지정된 일 수 내에 완료되지 않은 경우 S3 수명 주기 구성을 사용하면 미완료 멀티파트 업로드(규칙에 지정된 키 이름 접두사로 식별됨)를 중단하도록 Amazon S3에 지시할 수 있습니다. Amazon S3는 멀티파트 업로드를 중단할 때 멀티파트 업로드와 관련된 모든 부분을 삭제합니다. 이를 통해 미완료된 멀티파트 업로드는 물론 Amazon S3에 저장된 부분들이 없도록 보장하므로, 이 부분들에 대해 스토리지 비용을 지불할 필요가 없습니다.

> **Note**
>
> `AbortIncompleteMultipartUpload` 수명 주기 작업을 지정하면 이 규칙은 태그 기반 필터를 지정할 수 없습니다.

다음은 `AbortIncompleteMultipartUpload` 작업으로 규칙을 지정하는 S3 수명 주기 구성의 예제입니다. 이 작업은 시작 후 7일이 지나면 미완료된 멀티파트 업로드 중단하도록 Amazon S3에 요청합니다.

```
<LifecycleConfiguration>
    <Rule>
        <ID>sample-rule</ID>
        <Filter>
```

```
            <Prefix>SomeKeyPrefix/</Prefix>
        </Filter>
        <Status>rule-status</Status>
        <AbortIncompleteMultipartUpload>
          <DaysAfterInitiation>7</DaysAfterInitiation>
        </AbortIncompleteMultipartUpload>
    </Rule>
</LifecycleConfiguration>
```

버킷에서 수명 주기 구성 설정

주제

- Amazon S3 콘솔을 사용하여 객체 수명 주기 관리 (p. 146)
- AWS CLI를 사용하여 수명 주기 구성 설정 (p. 146)
- AWS SDK for Java를 사용하여 객체 수명 주기 관리 (p. 149)
- .NET용 AWS SDK를 사용하여 객체 수명 주기 관리 (p. 151)
- Ruby용 AWS SDK를 사용하여 객체 수명 주기 관리 (p. 154)
- REST API를 사용하여 객체 수명 주기 관리 (p. 154)

이 단원에서는 AWS SDK를 사용한 프로그래밍 방식으로, 또는 Amazon S3콘솔이나 AWS CLI를 사용하여 버킷에서 S3 수명 주기 구성을 설정하는 방법을 설명합니다. 다음을 참조하십시오.

- 버킷에 S3 수명 주기 구성을 추가할 때 일반적으로 새로운 수명 주기 또는 업데이트된 수명 주기 구성이 모든 Amazon S3 시스템에 완전히 전파될 때까지 약간의 지연 시간이 있습니다. 구성이 완전히 적용되려면 몇 분간 기다려야 합니다. 이러한 지연은 S3 수명 주기 구성을 삭제할 때도 발생할 수 있습니다.
- 수명 주기 규칙을 비활성화 또는 삭제할 때, 약간의 지연 후 Amazon S3이 새로운 객체에 대한 삭제 또는 전환 예약을 중단합니다. 이미 예약된 객체는 모두 예약이 취소되고 삭제 또는 전환되지 않습니다.
- 버킷에 수명 주기 구성을 추가할 경우 구성 규칙이 기존 객체는 물론 나중에 추가하는 객체에도 적용됩니다. 예를 들어 특정 접두사가 있는 객체를 생성 후 30일 경과 시 만료시키는 만료 작업을 포함하는 수명 주기 구성 규칙을 오늘 추가할 경우, Amazon S3은 30일 이상 경과한 모든 기존 객체를 삭제 대기열에 넣습니다.
- 수명 주기 구성 규칙이 충족되는 시점과 규칙 충족에 따라 트리거된 조치가 실행되는 시점 사이에는 지연이 있을 수 있습니다. 그러나 결제 변경은 조치가 실행되지 않더라도 수명 주기 구성 규칙이 충족되는 즉시 변경됩니다. 예를 들어, 객체 만료 시간 이후에는 해당 객체가 즉시 삭제되지 않아도 스토리지 비용이 청구되지 않습니다. 또 다른 예는 객체가 S3 Glacier 스토리지 클래스로 즉시 전환되지 않은 경우에도 객체 전환 시간이 경과하는 순간부터 Amazon S3 Glacier 스토리지 요금이 부과된다는 것입니다. S3 Intelligent Tiering 스토리지 클래스로의 수명 주기 전환은 예외이며 객체가 S3 Intelligent-Tiering 스토리지 클래스로 전환될 때까지 결제 변경이 발생하지 않습니다.

S3 수명 주기 구성에 대한 자세한 내용은 객체 수명 주기 관리 (p. 123) 단원을 참조하십시오.

Amazon S3 콘솔을 사용하여 객체 수명 주기 관리

Amazon S3 콘솔을 사용하여 버킷에서 S3 수명 주기 규칙을 지정할 수 있습니다.

AWS Management 콘솔을 사용하여 S3 수명 주기 규칙을 설정하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷에 대한 수명 주기 정책을 생성하려면 어떻게 해야 합니까?를 참조하십시오.

AWS CLI를 사용하여 수명 주기 구성 설정

다음 AWS CLI 명령을 사용하여 S3 수명 주기 구성을 관리할 수 있습니다.

- `put-bucket-lifecycle-configuration`

- `get-bucket-lifecycle-configuration`
- `delete-bucket-lifecycle`

AWS CLI를 설치하는 지침은 AWS CLI 설정 (p. 733) 단원을 참조하십시오.

Amazon S3 수명 주기 구성은 XML 파일입니다. 그러나 AWS CLI를 사용할 때는 XML을 지정할 수 없으며 대신 JSON을 지정해야 합니다. 다음은 XML 수명 주기 구성과 AWS CLI 명령에서 지정할 수 있는 그에 상당하는 JSON의 예입니다.

- 다음 예제 S3 수명 주기 구성을 살펴보십시오.

```
<LifecycleConfiguration>
    <Rule>
        <ID>ExampleRule</ID>
        <Filter>
           <Prefix>documents/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
           <Days>365</Days>
           <StorageClass>S3 Glacier</StorageClass>
        </Transition>
        <Expiration>
             <Days>3650</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

  XML에 상당하는 JSON이 표시됩니다.

```
{
    "Rules": [
        {
            "Filter": {
                "Prefix": "documents/"
            },
            "Status": "Enabled",
            "Transitions": [
                {
                    "Days": 365,
                    "StorageClass": "S3 Glacier"
                }
            ],
            "Expiration": {
                "Days": 3650
            },
            "ID": "ExampleRule"
        }
    ]
}
```

- 다음 예제 S3 수명 주기 구성을 살펴보십시오.

```
<LifecycleConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Rule>
        <ID>id-1</ID>
        <Expiration>
            <Days>1</Days>
        </Expiration>
        <Filter>
            <And>
```

```
                <Prefix>myprefix</Prefix>
                <Tag>
                    <Key>mytagkey1</Key>
                    <Value>mytagvalue1</Value>
                </Tag>
                <Tag>
                    <Key>mytagkey2</Key>
                    <Value>mytagvalue2</Value>
                </Tag>
            </And>
        </Filter>
        <Status>Enabled</Status>
    </Rule>
</LifecycleConfiguration>
```

XML에 상당하는 JSON이 표시됩니다.

```
{
    "Rules": [
        {
            "ID": "id-1",
            "Filter": {
                "And": {
                    "Prefix": "myprefix",
                    "Tags": [
                        {
                            "Value": "mytagvalue1",
                            "Key": "mytagkey1"
                        },
                        {
                            "Value": "mytagvalue2",
                            "Key": "mytagkey2"
                        }
                    ]
                }
            },
            "Status": "Enabled",
            "Expiration": {
                "Days": 1
            }
        }
    ]
}
```

다음과 같이 `put-bucket-lifecycle-configuration`을 테스트할 수 있습니다.

구성을 테스트하려면

1. 파일(`lifecycle.json`)에 JSON 수명 주기 구성을 저장합니다.
2. 다음 AWS CLI 명령을 실행하여 버킷에서 수명 주기 구성을 설정합니다.

```
$ aws s3api put-bucket-lifecycle-configuration  \
--bucket bucketname  \
--lifecycle-configuration file://lifecycle.json
```

3. 확인하려면 다음과 같이 `get-bucket-lifecycle-configuration` AWS CLI 명령을 사용해 S3 수명 주기 구성을 가져옵니다.

```
$ aws s3api get-bucket-lifecycle-configuration  \
--bucket bucketname
```

4. S3 수명 주기 구성을 삭제하려면 다음과 같이 `delete-bucket-lifecycle` AWS CLI 명령을 사용합니다.

```
aws s3api delete-bucket-lifecycle \
--bucket bucketname
```

AWS SDK for Java를 사용하여 객체 수명 주기 관리

AWS SDK for Java를 사용하여 버킷의 S3 수명 주기 구성을 관리할 수 있습니다. S3 수명 주기 구성 관리에 대한 자세한 내용은 객체 수명 주기 관리 (p. 123) 단원을 참조하십시오.

> **Note**
>
> 버킷에 S3 수명 주기구성을 추가하면 Amazon S3이 버킷의 현재 수명 주기 구성(있는 경우)을 대체합니다. 구성을 업데이트하려면 검색한 후 원하는 대로 변경한 다음 수정한 구성을 버킷에 추가합니다.

Example

다음 예제는 AWS SDK for Java를 사용하여 버킷의 수명 주기 구성을 추가, 업데이트 및 삭제하는 방법을 보여줍니다. 이 예제는 다음을 수행합니다.

- 버킷에 수명 주기 구성을 추가합니다.
- 수명 주기 구성을 검색하고 다른 규칙을 추가하여 구성을 업데이트합니다.
- 버킷에 수정된 수명 주기 구성을 추가합니다. Amazon S3이 기존 구성을 대체합니다.
- 구성을 다시 검색한 다음 규칙 수를 인쇄하여 올바른 개수의 규칙이 있는지 확인합니다.
- 수명 주기 구성을 삭제한 다음 다시 검색을 시도하여 삭제되었는지 확인합니다.

실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
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;
import com.amazonaws.services.s3.model.BucketLifecycleConfiguration;
import com.amazonaws.services.s3.model.BucketLifecycleConfiguration.Transition;
import com.amazonaws.services.s3.model.StorageClass;
import com.amazonaws.services.s3.model.Tag;
import com.amazonaws.services.s3.model.lifecycle.LifecycleAndOperator;
import com.amazonaws.services.s3.model.lifecycle.LifecycleFilter;
import com.amazonaws.services.s3.model.lifecycle.LifecyclePrefixPredicate;
import com.amazonaws.services.s3.model.lifecycle.LifecycleTagPredicate;

import java.io.IOException;
import java.util.Arrays;

public class LifecycleConfiguration {

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

        // Create a rule to archive objects with the "glacierobjects/" prefix to Glacier
 immediately.
```

```
        BucketLifecycleConfiguration.Rule rule1 = new BucketLifecycleConfiguration.Rule()
                .withId("Archive immediately rule")
                .withFilter(new LifecycleFilter(new
LifecyclePrefixPredicate("glacierobjects/")))
                .addTransition(new
Transition().withDays(0).withStorageClass(StorageClass.Glacier))
                .withStatus(BucketLifecycleConfiguration.ENABLED);

        // Create a rule to transition objects to the Standard-Infrequent Access storage
class
        // after 30 days, then to Glacier after 365 days. Amazon S3 will delete the objects
after 3650 days.
        // The rule applies to all objects with the tag "archive" set to "true".
        BucketLifecycleConfiguration.Rule rule2 = new BucketLifecycleConfiguration.Rule()
                .withId("Archive and then delete rule")
                .withFilter(new LifecycleFilter(new LifecycleTagPredicate(new
Tag("archive", "true"))))
                .addTransition(new
Transition().withDays(30).withStorageClass(StorageClass.StandardInfrequentAccess))
                .addTransition(new
Transition().withDays(365).withStorageClass(StorageClass.Glacier))
                .withExpirationInDays(3650)
                .withStatus(BucketLifecycleConfiguration.ENABLED);

        // Add the rules to a new BucketLifecycleConfiguration.
        BucketLifecycleConfiguration configuration = new BucketLifecycleConfiguration()
                .withRules(Arrays.asList(rule1, rule2));

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Save the configuration.
            s3Client.setBucketLifecycleConfiguration(bucketName, configuration);

            // Retrieve the configuration.
            configuration = s3Client.getBucketLifecycleConfiguration(bucketName);

            // Add a new rule with both a prefix predicate and a tag predicate.
            configuration.getRules().add(new
BucketLifecycleConfiguration.Rule().withId("NewRule")
                    .withFilter(new LifecycleFilter(new LifecycleAndOperator(
                            Arrays.asList(new LifecyclePrefixPredicate("YearlyDocuments/"),
                                    new LifecycleTagPredicate(new Tag("expire_after",
"ten_years"))))))
                    .withExpirationInDays(3650)
                    .withStatus(BucketLifecycleConfiguration.ENABLED));

            // Save the configuration.
            s3Client.setBucketLifecycleConfiguration(bucketName, configuration);

            // Retrieve the configuration.
            configuration = s3Client.getBucketLifecycleConfiguration(bucketName);

            // Verify that the configuration now has three rules.
            configuration = s3Client.getBucketLifecycleConfiguration(bucketName);
            System.out.println("Expected # of rules = 3; found: " +
configuration.getRules().size());

            // Delete the configuration.
            s3Client.deleteBucketLifecycleConfiguration(bucketName);

            // Verify that the configuration has been deleted by attempting to retrieve it.
            configuration = s3Client.getBucketLifecycleConfiguration(bucketName);
```

```
            String s = (configuration == null) ? "No configuration found." : "Configuration
 found.";
            System.out.println(s);
        } 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();
        }
    }
}
```

.NET용 AWS SDK를 사용하여 객체 수명 주기 관리

.NET용 AWS SDK를 사용하여 버킷에서 S3 수명 주기 구성을 관리할 수 있습니다. 수명 주기 구성 관리에 대한 자세한 내용은 객체 수명 주기 관리 (p. 123) 단원을 참조하십시오.

> **Note**
>
> 수명 주기 구성을 추가하면 Amazon S3이 지정된 버킷에서 기존 구성을 대체합니다. 구성을 업데이트하려면 먼저 수명 주기 구성을 검색하여 변경한 후, 수정한 수명 주기 구성을 버킷에 추가해야 합니다.

Example .NET 코드 예

다음 예제는 .NET용 AWS SDK를 사용하여 버킷의 수명 주기 구성을 추가, 업데이트 및 삭제하는 방법을 보여줍니다. 이 코드 예제에서는 다음 작업을 수행합니다.

- 버킷에 수명 주기 구성을 추가합니다.
- 수명 주기 구성을 검색하고 다른 규칙을 추가하여 구성을 업데이트합니다.
- 버킷에 수정된 수명 주기 구성을 추가합니다. Amazon S3이 기존 수명 주기 구성을 대체합니다.
- 구성을 다시 검색한 다음 구성의 규칙 수를 인쇄하여 구성을 확인합니다.
- 수명 주기 구성을 삭제하고 삭제되었는지 확인합니다.

실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

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

namespace Amazon.DocSamples.S3
{
    class LifecycleTest
    {
        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()
        {
```

```
            client = new AmazonS3Client(bucketRegion);
            AddUpdateDeleteLifecycleConfigAsync().Wait();
        }

        private static async Task AddUpdateDeleteLifecycleConfigAsync()
        {
            try
            {
                var lifeCycleConfiguration = new LifecycleConfiguration()
                {
                    Rules = new List<LifecycleRule>
                        {
                            new LifecycleRule
                            {
                                 Id = "Archive immediately rule",
                                 Filter = new LifecycleFilter()
                                 {
                                     LifecycleFilterPredicate = new
LifecyclePrefixPredicate()
                                     {
                                         Prefix = "glacierobjects/"
                                     }
                                 },
                                 Status = LifecycleRuleStatus.Enabled,
                                 Transitions = new List<LifecycleTransition>
                                 {
                                      new LifecycleTransition
                                      {
                                           Days = 0,
                                           StorageClass = S3StorageClass.Glacier
                                      }
                                  },
                            },
                            new LifecycleRule
                            {
                                 Id = "Archive and then delete rule",
                                  Filter = new LifecycleFilter()
                                 {
                                     LifecycleFilterPredicate = new
LifecyclePrefixPredicate()
                                     {
                                         Prefix = "projectdocs/"
                                     }
                                 },
                                 Status = LifecycleRuleStatus.Enabled,
                                 Transitions = new List<LifecycleTransition>
                                 {
                                      new LifecycleTransition
                                      {
                                           Days = 30,
                                           StorageClass =
S3StorageClass.StandardInfrequentAccess
                                      },
                                      new LifecycleTransition
                                      {
                                        Days = 365,
                                        StorageClass = S3StorageClass.Glacier
                                      }
                                 },
                                 Expiration = new LifecycleRuleExpiration()
                                 {
                                       Days = 3650
                                 }
                            }
                        }
                };
```

```
                // Add the configuration to the bucket.
                await AddExampleLifecycleConfigAsync(client, lifeCycleConfiguration);

                // Retrieve an existing configuration.
                lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);

                // Add a new rule.
                lifeCycleConfiguration.Rules.Add(new LifecycleRule
                {
                    Id = "NewRule",
                    Filter = new LifecycleFilter()
                    {
                        LifecycleFilterPredicate = new LifecyclePrefixPredicate()
                        {
                            Prefix = "YearlyDocuments/"
                        }
                    },
                    Expiration = new LifecycleRuleExpiration()
                    {
                        Days = 3650
                    }
                });

                // Add the configuration to the bucket.
                await AddExampleLifecycleConfigAsync(client, lifeCycleConfiguration);

                // Verify that there are now three rules.
                lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);
                Console.WriteLine("Expected # of rulest=3; found:{0}",
lifeCycleConfiguration.Rules.Count);

                // Delete the configuration.
                await RemoveLifecycleConfigAsync(client);

                // Retrieve a nonexistent configuration.
                lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);

            }
            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);
            }
        }

        static async Task AddExampleLifecycleConfigAsync(IAmazonS3 client,
LifecycleConfiguration configuration)
        {

            PutLifecycleConfigurationRequest request = new PutLifecycleConfigurationRequest
            {
                BucketName = bucketName,
                Configuration = configuration
            };
            var response = await client.PutLifecycleConfigurationAsync(request);
        }

        static async Task<LifecycleConfiguration> RetrieveLifecycleConfigAsync(IAmazonS3
client)
        {
```

```
            GetLifecycleConfigurationRequest request = new GetLifecycleConfigurationRequest
            {
                BucketName = bucketName
            };
            var response = await client.GetLifecycleConfigurationAsync(request);
            var configuration = response.Configuration;
            return configuration;
        }

        static async Task RemoveLifecycleConfigAsync(IAmazonS3 client)
        {
            DeleteLifecycleConfigurationRequest request = new
 DeleteLifecycleConfigurationRequest
            {
                BucketName = bucketName
            };
            await client.DeleteLifecycleConfigurationAsync(request);
        }
    }
}
```

Ruby용 AWS SDK를 사용하여 객체 수명 주기 관리

Ruby용 AWS SDK를 사용하여 버킷에서 AWS::S3::BucketLifecycleConfiguration 클래스를 통해 S3 수명 주기 구성을 관리할 수 있습니다. Amazon S3에서 Ruby용 AWS SDK를 사용하는 방법에 대한 자세한 내용은 Ruby용 AWS SDK 버전 3 사용 (p. 737) 단원을 참조하십시오. 수명 주기 구성 관리에 대한 자세한 내용은 객체 수명 주기 관리 (p. 123) 단원을 참조하십시오.

REST API를 사용하여 객체 수명 주기 관리

AWS Management 콘솔을 사용하여 버킷에서 S3 수명 주기 구성을 설정할 수 있습니다. 하지만 애플리케이션에서 요구할 경우 REST 요청을 직접 전송할 수도 있습니다. Amazon Simple Storage Service API Reference의 다음 단원에서는 S3 수명 주기 구성과 관련된 REST API를 설명합니다.

- PUT Bucket lifecycle
- GET Bucket lifecycle
- DELETE Bucket lifecycle

CORS(Cross-Origin Resource Sharing)

CORS(Cross-origin 리소스 공유)는 한 도메인에서 로드되어 다른 도메인에 있는 리소스와 상호 작용하는 클라이언트 웹 애플리케이션에 대한 방법을 정의합니다. CORS 지원을 통해 Amazon S3로 다양한 기능의 클라이언트 측 웹 애플리케이션을 구축하고, Amazon S3 리소스에 대한 cross-origin 액세스를 선택적으로 허용할 수 있습니다.

이 단원에서는 CORS 개요를 다룹니다. 하위 주제에서는 Amazon S3 콘솔을 사용하거나 Amazon S3 REST API 및 AWS SDK를 사용하는 프로그래밍 방식으로 여 CORS를 활성화하는 방법을 설명합니다.

주제

- Cross-Origin 리소스 공유(CORS): 사용 사례 시나리오 (p. 155)
- 내 버킷에서 CORS를 구성하려면 어떻게 해야 합니까? (p. 155)
- Amazon S3는 버킷에 대한 CORS 구성을 어떻게 평가합니까? (p. 157)
- Cross-Origin 리소스 공유(CORS) 활성화 (p. 157)
- CORS 문제 해결 (p. 163)

Cross-Origin 리소스 공유(CORS): 사용 사례 시나리오

다음은 CORS 사용에 대한 예제 시나리오입니다.

시나리오 1

Amazon S3 정적 웹 사이트 호스팅 (p. 548) 단원에서 설명한 대로 `website`라는 이름의 Amazon S3 버킷에서 웹 사이트를 호스팅한다고 가정해 보겠습니다. 사용자는 웹 사이트 엔드포인트를 로드합니다.

```
http://website.s3-website.us-east-1.amazonaws.com
```

이제 이 버킷에 저장된 웹 페이지의 JavaScript를 사용하여, `website.s3.us-east-1.amazonaws.com` 버킷에 대한 Amazon S3 API 엔드포인트를 사용하여 같은 버킷에 대해 GET 및 PUT 요청 인증을 가능하게 하려고 합니다. 일반적으로 브라우저에서 이러한 요청을 허용하지 못하도록 JavaScript를 차단하지만, CORS를 사용하여 `website.s3-website.us-east-1.amazonaws.com`으로부터의 cross-origin 요청을 활성화하도록 버킷을 구성할 수 있습니다.

시나리오 2

S3 버킷에서 웹 글꼴을 호스팅한다고 가정해 보겠습니다. 브라우저는 웹 글꼴을 불러오기 위해 CORS 검사(preflight check라고도 함)가 필요하므로, 이 웹 글꼴을 호스팅하는 버킷이 이러한 요청을 하는 오리진을 허용하도록 구성합니다.

내 버킷에서 CORS를 구성하려면 어떻게 해야 합니까?

cross-origin 요청을 허용하도록 버킷을 구성하려면 CORS 구성, 즉 버킷에 대한 액세스를 허용할 오리진과 각 오리진에 대해 지원할 작업(HTTP 메서드)을 식별하는 규칙과 기타 작업별 정보를 포함하는 XML 문서를 만듭니다.

구성당 최대 100개의 규칙을 추가할 수 있습니다. 프로그래밍 방식으로 또는 Amazon S3 콘솔을 사용해 XML 문서를 버킷에 대한 `cors` 하위 원본으로 추가합니다. 자세한 내용은 Cross-Origin 리소스 공유(CORS) 활성화 (p. 157) 단원을 참조하십시오.

Amazon S3 웹 사이트 엔드포인트를 사용하여 웹 사이트에 액세스하는 대신, `example1.com`과 같은 자체 도메인을 사용하여 콘텐츠를 서비스할 수 있습니다. 자체 도메인 사용에 대한 자세한 내용은 Route 53에 등록된 사용자 지정 도메인을 사용하여 정적 웹 사이트 구성 (p. 574) 단원을 참조하십시오. 다음 예시 `cors` 구성에는 `CORSRule` 요소로 지정된 규칙이 3개 있습니다.

- 첫 번째 규칙은 `http://www.example1.com` 오리진에서 cross-origin PUT, POST, DELETE 요청을 허용합니다. 이 규칙은 또한 `Access-Control-Request-Headers` 헤더를 통해 preflight OPTIONS 요청의 모든 헤더를 허용합니다. 는 preflight OPTIONS 요청에 대한 응답으로 요청된 헤더를 반환합니다.
- 두 번째 규칙은 첫 번째 규칙과 같이 cross-origin 요청을 허용하지만 다른 오리진인 `http://www.example2.com`에 적용됩니다.
- 세 번째 규칙은 모든 오리진에서 cross-origin GET 요청을 허용합니다. `*` 와일드카드 문자는 모든 오리진을 나타냅니다.

```
<CORSConfiguration>
 <CORSRule>
   <AllowedOrigin>http://www.example1.com</AllowedOrigin>

   <AllowedMethod>PUT</AllowedMethod>
   <AllowedMethod>POST</AllowedMethod>
   <AllowedMethod>DELETE</AllowedMethod>

   <AllowedHeader>*</AllowedHeader>
 </CORSRule>
```

```
 <CORSRule>
   <AllowedOrigin>http://www.example2.com</AllowedOrigin>

   <AllowedMethod>PUT</AllowedMethod>
   <AllowedMethod>POST</AllowedMethod>
   <AllowedMethod>DELETE</AllowedMethod>

   <AllowedHeader>*</AllowedHeader>
 </CORSRule>
 <CORSRule>
   <AllowedOrigin>*</AllowedOrigin>
   <AllowedMethod>GET</AllowedMethod>
 </CORSRule>
</CORSConfiguration>
```

CORS 구성은 다음에서 볼 수 있듯이 선택적인 구성 파라미터도 허용합니다. 이 예제에서 CORS 구성은 `http://www.example.com` 오리진으로부터의 cross-origin PUT, POST 및 DELETE 요청을 허용합니다.

```
<CORSConfiguration>
 <CORSRule>
   <AllowedOrigin>http://www.example.com</AllowedOrigin>
   <AllowedMethod>PUT</AllowedMethod>
   <AllowedMethod>POST</AllowedMethod>
   <AllowedMethod>DELETE</AllowedMethod>
   <AllowedHeader>*</AllowedHeader>
  <MaxAgeSeconds>3000</MaxAgeSeconds>
  <ExposeHeader>x-amz-server-side-encryption</
ExposeHeader>
  <ExposeHeader>x-amz-request-id</
ExposeHeader>
  <ExposeHeader>x-amz-id-2</ExposeHeader>
 </CORSRule>
</CORSConfiguration>
```

이전 구성에서 `CORSRule` 요소는 다음의 선택적 요소를 포함합니다.

- `MaxAgeSeconds`–브라우저가 지정된 리소스에 대한 Amazon S3 preflight OPTIONS 요청 응답을 캐시하는 시간을 초 단위(이 예에서는 3000)로 지정합니다. 원래 요청이 반복될 경우, 브라우저는 이 응답을 캐시함으로써 에 preflight 요청을 보낼 필요가 없습니다.
- `ExposeHeader`–고객이 자체 애플리케이션(예: JavaScript `XMLHttpRequest` 객체)에서 액세스할 수 있는 응답 헤더(이 예에서는 `x-amz-server-side-encryption`, `x-amz-request-id`, `x-amz-id-2`)를 식별합니다.

AllowedMethod 요소

CORS 구성에서 `AllowedMethod` 요소에 대한 다음 값을 지정할 수 있습니다.

- GET
- PUT
- POST
- DELETE
- HEAD

AllowedOrigin 요소

`AllowedOrigin` 요소에 교차 도메인 요청을 허용하려는 오리진을 지정합니다(예: `http://www.example.com`). 오리진 문자열에는 `*` 와일드 문자 한 개만 포함할 수 있습니다(예: `http://`

`*.example.com`). `*`를 모든 오리진이 교차 오리진 요청을 보낼 수 있는 오리진으로 지정할 수도 있습니다. 또한 `https`를 지정하여 안전한 오리진만 허용할 수도 있습니다.

AllowedHeader 요소

`AllowedHeader` 요소는 `Access-Control-Request-Headers` 헤더를 통해 preflight 요청에 허용되는 헤더를 지정합니다. `Access-Control-Request-Headers` 헤더의 각 헤더 이름은 규칙의 해당 항목과 일치해야 합니다. Amazon S3는 요청된 응답에서 허용된 헤더만을 보냅니다. Amazon S3에 대한 요청에 사용할 수 있는 헤더의 예는 Amazon Simple Storage Service API Reference 가이드의 일반 요청 헤더를 참조하십시오.

규칙의 각 AllowedHeader 문자열에는 최대 한 개의 * 와일드카드 문자를 포함할 수 있습니다. 예를 들어, `<AllowedHeader>x-amz-*</AllowedHeader>`는 모든 Amazon별 헤더를 허용합니다.

ExposeHeader 요소

각 `ExposeHeader` 요소는 응답에서 고객이 해당 애플리케이션(예: JavaScript `XMLHttpRequest` 객체)으로부터 액세스할 수 있도록 하려는 헤더를 식별합니다. 일반적인 Amazon S3 응답 헤더는 Amazon Simple Storage Service API Reference 가이드의 일반 응답 헤더 단원을 참조하십시오.

MaxAgeSeconds 요소

`MaxAgeSeconds` 요소는 브라우저가 리소스, HTTP 메서드, 오리진으로 식별되는 preflight 요청에 대한 응답을 캐시할 수 있는 시간(초)을 지정합니다.

Amazon S3는 버킷에 대한 CORS 구성을 어떻게 평가합니까?

Amazon S3는 브라우저에서 preflight 요청을 받으면 버킷에 대한 CORS 구성을 평가하여, 수신된 브라우저 요청과 일치하는 첫 번째 `CORSRule` 규칙을 사용하여 cross-origin 요청을 허용합니다. 규칙이 일치하려면 다음 조건이 충족되어야 합니다.

- 요청의 `Origin` 헤더가 `AllowedOrigin` 요소와 일치해야 합니다.
- 요청 메서드(예: GET 또는 PUT) 또는 `Access-Control-Request-Method` 헤더(preflight `OPTIONS` 요청의 경우)가 `AllowedMethod` 요소 중 하나와 일치해야 합니다.
- preflight 요청의 `Access-Control-Request-Headers` 헤더에 나열된 모든 헤더가 `AllowedHeader` 요소와 일치해야 합니다.

> **Note**
>
> 버킷에 대해 CORS를 허용하면 ACL과 정책이 계속 적용됩니다.

Cross-Origin 리소스 공유(CORS) 활성화

AWS Management 콘솔, REST API 또는 AWS SDK를 사용하여 버킷에 CORS 구성을 설정함으로써 cross-origin 리소스 공유를 활성화할 수 있습니다.

주제

- AWS Management 콘솔을 사용하여 Cross-Origin 리소스 공유(CORS) 활성화 (p. 158)
- AWS SDK for Java을 사용하여 Cross-Origin 리소스 공유(CORS) 활성화 (p. 158)
- .NET용 AWS SDK을 사용하여 Cross-Origin 리소스 공유(CORS) 활성화 (p. 160)

- REST API를 사용하여 Cross-Origin 리소스 공유(CORS) 활성화 (p. 163)

AWS Management 콘솔을 사용하여 Cross-Origin 리소스 공유(CORS) 활성화

AWS Management 콘솔을 사용하여 버킷에 CORS 구성을 설정할 수 있습니다. 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 CORS와의 교차 도메인 리소스 공유를 허용하려면 어떻게 해야 합니까? 단원을 참조하십시오.

AWS SDK for Java을 사용하여 Cross-Origin 리소스 공유(CORS) 활성화

AWS SDK for Java를 사용하여 버킷에 대한 cross-origin 리소스 공유(CORS)를 관리할 수 있습니다. CORS에 대한 자세한 내용은 CORS(Cross-Origin Resource Sharing) (p. 154) 단원을 참조하십시오.

Example

다음 예제는:

- CORS 구성을 만들고 버킷에 구성 설정
- 구성을 검색하고 규칙을 추가하여 구성 수정
- 버킷에 수정된 구성 추가
- 구성 삭제

실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
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;
import com.amazonaws.services.s3.model.BucketCrossOriginConfiguration;
import com.amazonaws.services.s3.model.CORSRule;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class CORS {

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

        // Create two CORS rules.
        List<CORSRule.AllowedMethods> rule1AM = new ArrayList<CORSRule.AllowedMethods>();
        rule1AM.add(CORSRule.AllowedMethods.PUT);
        rule1AM.add(CORSRule.AllowedMethods.POST);
        rule1AM.add(CORSRule.AllowedMethods.DELETE);
        CORSRule rule1 = new CORSRule().withId("CORSRule1").withAllowedMethods(rule1AM)
                .withAllowedOrigins(Arrays.asList("http://*.example.com"));

        List<CORSRule.AllowedMethods> rule2AM = new ArrayList<CORSRule.AllowedMethods>();
```

```
        rule2AM.add(CORSRule.AllowedMethods.GET);
        CORSRule rule2 = new CORSRule().withId("CORSRule2").withAllowedMethods(rule2AM)
                .withAllowedOrigins(Arrays.asList("*")).withMaxAgeSeconds(3000)
                .withExposedHeaders(Arrays.asList("x-amz-server-side-encryption"));

        List<CORSRule> rules = new ArrayList<CORSRule>();
        rules.add(rule1);
        rules.add(rule2);

        // Add the rules to a new CORS configuration.
        BucketCrossOriginConfiguration configuration = new
BucketCrossOriginConfiguration();
        configuration.setRules(rules);

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Add the configuration to the bucket.
            s3Client.setBucketCrossOriginConfiguration(bucketName, configuration);

            // Retrieve and display the configuration.
            configuration = s3Client.getBucketCrossOriginConfiguration(bucketName);
            printCORSConfiguration(configuration);

            // Add another new rule.
            List<CORSRule.AllowedMethods> rule3AM = new
ArrayList<CORSRule.AllowedMethods>();
            rule3AM.add(CORSRule.AllowedMethods.HEAD);
            CORSRule rule3 = new CORSRule().withId("CORSRule3").withAllowedMethods(rule3AM)
                    .withAllowedOrigins(Arrays.asList("http://www.example.com"));

            rules = configuration.getRules();
            rules.add(rule3);
            configuration.setRules(rules);
            s3Client.setBucketCrossOriginConfiguration(bucketName, configuration);

            // Verify that the new rule was added by checking the number of rules in the
configuration.
            configuration = s3Client.getBucketCrossOriginConfiguration(bucketName);
            System.out.println("Expected # of rules = 3, found " +
configuration.getRules().size());

            // Delete the configuration.
            s3Client.deleteBucketCrossOriginConfiguration(bucketName);
            System.out.println("Removed CORS configuration.");

            // Retrieve and display the configuration to verify that it was
            // successfully deleted.
            configuration = s3Client.getBucketCrossOriginConfiguration(bucketName);
            printCORSConfiguration(configuration);
        } 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();
        }
    }

    private static void printCORSConfiguration(BucketCrossOriginConfiguration
configuration) {
```

```
        if (configuration == null) {
            System.out.println("Configuration is null.");
        } else {
            System.out.println("Configuration has " + configuration.getRules().size() + "
 rules\n");

            for (CORSRule rule : configuration.getRules()) {
                System.out.println("Rule ID: " + rule.getId());
                System.out.println("MaxAgeSeconds: " + rule.getMaxAgeSeconds());
                System.out.println("AllowedMethod: " + rule.getAllowedMethods());
                System.out.println("AllowedOrigins: " + rule.getAllowedOrigins());
                System.out.println("AllowedHeaders: " + rule.getAllowedHeaders());
                System.out.println("ExposeHeader: " + rule.getExposedHeaders());
                System.out.println();
            }
        }
    }
}
```

.NET용 AWS SDK을 사용하여 Cross-Origin 리소스 공유(CORS) 활성화

버킷에 대한 cross-origin 리소스 공유(CORS)를 관리하려면 .NET용 AWS SDK를 사용할 수 있습니다. CORS에 대한 자세한 내용은 CORS(Cross-Origin Resource Sharing) (p. 154) 단원을 참조하십시오.

Example

다음 C# 코드는:

- CORS 구성을 만들고 버킷에 구성 설정
- 구성을 검색하고 규칙을 추가하여 구성 수정
- 버킷에 수정된 구성 추가
- 구성 삭제

실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

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

namespace Amazon.DocSamples.S3
{
    class CORSTest
    {
        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()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            CORSConfigTestAsync().Wait();
        }
```

```
        private static async Task CORSConfigTestAsync()
        {
            try
            {
                // Create a new configuration request and add two rules
                CORSConfiguration configuration = new CORSConfiguration
                {
                    Rules = new System.Collections.Generic.List<CORSRule>
                        {
                          new CORSRule
                          {
                            Id = "CORSRule1",
                            AllowedMethods = new List<string> {"PUT", "POST", "DELETE"},
                            AllowedOrigins = new List<string> {"http://*.example.com"}
                          },
                          new CORSRule
                          {
                            Id = "CORSRule2",
                            AllowedMethods = new List<string> {"GET"},
                            AllowedOrigins = new List<string> {"*"},
                            MaxAgeSeconds = 3000,
                            ExposeHeaders = new List<string> {"x-amz-server-side-
encryption"}
                          }
                        }
                };

                // Add the configuration to the bucket.
                await PutCORSConfigurationAsync(configuration);

                // Retrieve an existing configuration.
                configuration = await RetrieveCORSConfigurationAsync();

                // Add a new rule.
                configuration.Rules.Add(new CORSRule
                {
                    Id = "CORSRule3",
                    AllowedMethods = new List<string> { "HEAD" },
                    AllowedOrigins = new List<string> { "http://www.example.com" }
                });

                // Add the configuration to the bucket.
                await PutCORSConfigurationAsync(configuration);

                // Verify that there are now three rules.
                configuration = await RetrieveCORSConfigurationAsync();
                Console.WriteLine();
                Console.WriteLine("Expected # of rulest=3; found:{0}",
 configuration.Rules.Count);
                Console.WriteLine();
                Console.WriteLine("Pause before configuration delete. To continue, click
 Enter...");
                Console.ReadKey();

                // Delete the configuration.
                await DeleteCORSConfigurationAsync();

                // Retrieve a nonexistent configuration.
                configuration = await RetrieveCORSConfigurationAsync();
            }
            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);
            }
        }

        static async Task PutCORSConfigurationAsync(CORSConfiguration configuration)
        {

            PutCORSConfigurationRequest request = new PutCORSConfigurationRequest
            {
                BucketName = bucketName,
                Configuration = configuration
            };

            var response = await s3Client.PutCORSConfigurationAsync(request);
        }

        static async Task<CORSConfiguration> RetrieveCORSConfigurationAsync()
        {
            GetCORSConfigurationRequest request = new GetCORSConfigurationRequest
            {
                BucketName = bucketName

            };
            var response = await s3Client.GetCORSConfigurationAsync(request);
            var configuration = response.Configuration;
            PrintCORSRules(configuration);
            return configuration;
        }

        static async Task DeleteCORSConfigurationAsync()
        {
            DeleteCORSConfigurationRequest request = new DeleteCORSConfigurationRequest
            {
                BucketName = bucketName
            };
            await s3Client.DeleteCORSConfigurationAsync(request);
        }

        static void PrintCORSRules(CORSConfiguration configuration)
        {
            Console.WriteLine();

            if (configuration == null)
            {
                Console.WriteLine("\nConfiguration is null");
                return;
            }

            Console.WriteLine("Configuration has {0} rules:", configuration.Rules.Count);
            foreach (CORSRule rule in configuration.Rules)
            {
                Console.WriteLine("Rule ID: {0}", rule.Id);
                Console.WriteLine("MaxAgeSeconds: {0}", rule.MaxAgeSeconds);
                Console.WriteLine("AllowedMethod: {0}", string.Join(", ",
rule.AllowedMethods.ToArray()));
                Console.WriteLine("AllowedOrigins: {0}", string.Join(", ",
rule.AllowedOrigins.ToArray()));
                Console.WriteLine("AllowedHeaders: {0}", string.Join(", ",
rule.AllowedHeaders.ToArray()));
                Console.WriteLine("ExposeHeader: {0}", string.Join(", ",
rule.ExposeHeaders.ToArray()));
            }
        }
    }
```

```
}
```

REST API를 사용하여 Cross-Origin 리소스 공유(CORS) 활성화

버킷에 CORS 구성을 설정하려면 AWS Management 콘솔을 사용할 수 있습니다. 하지만 애플리케이션에서 요구할 경우 REST 요청을 직접 전송할 수도 있습니다. 다음 Amazon Simple Storage Service API Reference의 단원에서는 CORS 구성과 관련된 REST API 작업을 설명합니다.

- PUT Bucket cors
- GET Bucket cors
- DELETE Bucket cors
- OPTIONS 객체

CORS 문제 해결

CORS 구성으로 설정된 버킷에 액세스하는 중에 예상치 못한 동작이 발생할 경우, 다음 단계에 따라 문제를 해결해 보십시오.

1. CORS 구성이 버킷에 설정되어 있는지 확인합니다.

   지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 버킷 권한 편집을 참조하십시오. CORS 구성이 설정된 경우 콘솔에서 속성 버킷의 권한 섹션에 Edit CORS Configuration(CORS 구성 편집) 링크가 표시됩니다.
2. 원하는 도구를 사용하여 완료 요청 및 응답을 캡처합니다. Amazon S3가 수신하는 각 요청에 대해 다음과 같이 해당 요청의 데이터와 일치하는 CORS 규칙이 있어야 합니다.
   a. 요청에 Origin 헤더가 있는 확인합니다.

      헤더가 없으면 Amazon S3는 요청을 cross-origin 요청으로 처리하지 않고, 응답에 CORS 응답 헤더를 보내지 않습니다.
   b. 요청의 Origin 헤더가 지정된 `AllowedOrigin`의 `CORSRule` 요소 중 최소 하나와 일치하는지 확인합니다.

      Origin 요청 헤더의 체계, 호스트, 포트 값이 `AllowedOrigin`의 `CORSRule` 요소와 일치해야 합니다. 예를 들어 `CORSRule`을 설정하여 `http://www.example.com` 오리진을 허용한 경우, 요청의 `https://www.example.com` 오리진과 `http://www.example.com:80` 오리진 모두가 해당 구성의 허용되는 오리진과 일치하지 않는 것입니다.
   c. 요청의 메서드(preflight 요청에서는 `Access-Control-Request-Method`에 지정된 메서드)가 동일한 `AllowedMethod`의 `CORSRule` 요소 중 하나인지 확인합니다.
   d. preflight 요청의 경우 요청에 `Access-Control-Request-Headers` 헤더가 있을 경우, `CORSRule` 헤더에 각 값에 대한 `AllowedHeader` 항목이 `Access-Control-Request-Headers`에 포함되었는지 확인합니다.

객체 관련 작업

Amazon S3는 객체를 저장, 검색 및 삭제할 수 있는 기능을 제공합니다. 전체 객체 또는 객체 일부만 검색할 수 있습니다. 버킷에서 S3 버전 관리를 활성화한 경우 특정 버전의 객체를 검색할 수 있습니다. 또한 객체에 연결된 하위 리소스를 검색하고 필요할 경우 업데이트할 수 있습니다. 기존 객체를 복사할 수 있습니다. 객체 크기에 따라 업로드 및 복사 시 다음과 같은 사항을 고려해야 합니다.

- 객체 업로드–단일 작업으로 최대 5GB 크기의 객체를 업로드할 수 있습니다. 5GB보다 큰 객체는 멀티파트 업로드 API를 사용해야 합니다.

멀티파트 업로드 API를 사용하면 최대 5TB의 객체를 업로드할 수 있습니다. 자세한 내용은 멀티파트 업로드 API를 사용한 객체 업로드 (p. 177) 단원을 참조하십시오.

- 객체 복사–복사 작업은 Amazon S3에 저장되어 있는 객체의 복사본을 만듭니다.

  객체 업로드 - 단일 원자성 작업으로 5GB 크기 이하의 객체에 대해 복사본을 만들 수 있습니다. 5GB보다 큰 객체를 복사하려면 멀티파트 업로드 API를 사용해야 합니다. 자세한 내용은 객체 복사 (p. 209) 단원을 참조하십시오.

REST API(REST API를 사용하여 요청 (p. 42) 참조)를 사용하여 객체 작업을 수행하거나 다음 AWS SDK 라이브러리 중 하나를 사용할 수 있습니다.

- AWS SDK for Java
- .NET용 AWS SDK
- PHP용 AWS SDK

이 라이브러리는 높은 수준의 추상화를 제공하여 쉽게 객체 관련 작업을 수행할 수 있습니다. 그러나 필요할 경우 애플리케이션에서 바로 REST API를 사용할 수 있습니다.

객체 가져오기

주제

- 관련 리소스 (p. 165)
- AWS SDK for Java를 사용하여 객체 가져오기 (p. 165)
- .NET용 AWS SDK를 사용하여 객체 가져오기 (p. 167)
- PHP용 AWS SDK를 사용하여 객체 가져오기 (p. 168)
- REST API를 사용하여 객체 가져오기 (p. 169)
- 다른 사용자와 객체 공유 (p. 170)

Amazon S3에서 바로 객체를 검색할 수 있으며, 객체 검색 시 다음과 같은 옵션을 선택할 수 있습니다.

- 전체 객체 검색–단일 GET 작업으로 Amazon S3에 저장된 전체 객체를 반환할 수 있습니다.
- 부분 객체 검색–GET 요청에 `Range` HTTP 헤더를 사용하여 Amazon S3에 저장된 객체에서 특정 바이트 범위를 검색할 수 있습니다.

  애플리케이션이 준비만 되면 언제든지 객체의 다른 부분을 계속 가져올 수 있습니다. 이어서 받기 가능한 이 다운로드는 객체 데이터의 일부분만 필요할 때 유용하게 사용할 수 있습니다. 또한 네트워크 연결이 갑자기 끊어질 경우에 대비해야 할 때도 유용합니다.

  > **Note**
  >
  > Amazon S3는 GET 요청에 따른 데이터의 여러 범위 검색을 지원하지 않습니다.

객체를 검색하면 응답 헤더에 그 메타데이터가 반환됩니다. GET 응답에 반환된 특정 응답 헤더 값을 무시해야 할 수 있습니다. 예를 들어, GET 요청에서 `Content-Disposition` 응답 헤더 값을 무시할 수 있습니다. REST GET Object API(GET Object 참조)를 사용하면 GET 요청에 쿼리 문자열 파라미터를 지정하여 이 값을 무시할 수 있습니다.

Java, .NET 및 PHP용 AWS SDK 또한 GET 요청에서 이러한 응답 헤더를 위한 값을 지정하는 데 사용할 수 있는 필수 객체를 제공합니다.

서버 측 암호화를 사용하여 암호화된 상태로 저장된 객체를 검색하려면 적절한 요청 헤더를 제공해야 합니다. 자세한 내용은 암호화를 사용하여 데이터 보호 (p. 262) 단원을 참조하십시오.

관련 리소스

- AWS SDK, CLI 및 Explorer 사용 (p. 727)

AWS SDK for Java를 사용하여 객체 가져오기

AWS SDK for Java를 통해 객체를 다운로드하면 Amazon S3가 모든 객체의 메타데이터와 객체의 콘텐츠를 읽기 위한 입력 스트림을 반환합니다.

객체를 검색하려면 다음을 수행합니다.

- 요청에 버킷 이름과 객체 키를 제공하여 `AmazonS3Client.getObject()` 메서드를 실행합니다.
- `S3Object` 인스턴스 메서드 중 하나를 실행하여 입력 스트림을 처리합니다.

> **Note**
>
> 모든 데이터를 읽거나 입력 스트림을 닫을 때까지 네트워크 연결이 열린 상태로 유지됩니다. 따라서 최대한 빨리 스트림의 콘텐츠를 읽는 것이 좋습니다.

다음은 사용할 수 있는 몇 가지 변형 형태입니다.

- 전체 객체를 읽는 대신, 요청에서 원하는 바이트 범위를 지정하여 객체 데이터의 일부만 읽을 수 있습니다.
- 객체 가져오기 (p. 164) 객체를 사용하고 해당 요청 속성을 설정하여 필요할 경우 응답 헤더 값을 무시할 수 있습니다(`ResponseHeaderOverrides` 참조). 예를 들어 이 기능을 사용하여 객체 키 이름과 다른 파일 이름을 가진 파일로 다운로드해야 할 객체를 지정할 수 있습니다.

다음 예제에서는 세 가지 방법으로 Amazon S3 버킷에서 객체를 검색합니다. 먼저 전체 객체로, 그 다음은 객체의 바이트 범위로, 그 다음은 무시된 응답 헤더 값을 가진 전체 객체로 검색합니다. Amazon S3에서 객체 가져오기에 대한 자세한 내용은 GET Object를 참조하십시오. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
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;
import com.amazonaws.services.s3.model.GetObjectRequest;
import com.amazonaws.services.s3.model.ResponseHeaderOverrides;
import com.amazonaws.services.s3.model.S3Object;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

public class GetObject {

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

        S3Object fullObject = null, objectPortion = null, headerOverrideObject = null;
        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
```

```
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();

            // Get an object and print its contents.
            System.out.println("Downloading an object");
            fullObject = s3Client.getObject(new GetObjectRequest(bucketName, key));
            System.out.println("Content-Type: " +
 fullObject.getObjectMetadata().getContentType());
            System.out.println("Content: ");
            displayTextInputStream(fullObject.getObjectContent());

            // Get a range of bytes from an object and print the bytes.
            GetObjectRequest rangeObjectRequest = new GetObjectRequest(bucketName, key)
                    .withRange(0, 9);
            objectPortion = s3Client.getObject(rangeObjectRequest);
            System.out.println("Printing bytes retrieved.");
            displayTextInputStream(objectPortion.getObjectContent());

            // Get an entire object, overriding the specified response headers, and print
 the object's content.
            ResponseHeaderOverrides headerOverrides = new ResponseHeaderOverrides()
                    .withCacheControl("No-cache")
                    .withContentDisposition("attachment; filename=example.txt");
            GetObjectRequest getObjectRequestHeaderOverride = new
 GetObjectRequest(bucketName, key)
                    .withResponseHeaders(headerOverrides);
            headerOverrideObject = s3Client.getObject(getObjectRequestHeaderOverride);
            displayTextInputStream(headerOverrideObject.getObjectContent());
        } 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();
        } finally {
            // To ensure that the network connection doesn't remain open, close any open
 input streams.
            if (fullObject != null) {
                fullObject.close();
            }
            if (objectPortion != null) {
                objectPortion.close();
            }
            if (headerOverrideObject != null) {
                headerOverrideObject.close();
            }
        }
    }

    private static void displayTextInputStream(InputStream input) throws IOException {
        // Read the text input stream one line at a time and display each line.
        BufferedReader reader = new BufferedReader(new InputStreamReader(input));
        String line = null;
        while ((line = reader.readLine()) != null) {
            System.out.println(line);
        }
        System.out.println();
    }
}
```

.NET용 AWS SDK를 사용하여 객체 가져오기

객체를 다운로드하면 모든 객체의 메타데이터와 콘텐츠를 읽기 위한 스트림을 가져오게 됩니다. Amazon S3에서 바로 데이터가 스트리밍되어 모든 데이터를 읽거나 입력 스트림을 닫을 때까지 네트워크 연결을 열어두어야 하므로 최대한 빨리 콘텐츠를 읽어야 합니다. 객체를 가져오려면 다음을 수행합니다.

- 요청에 버킷 이름과 객체 키를 제공하여 `getObject` 메서드를 실행합니다.
- `GetObjectResponse` 메서드 중 하나를 실행하여 스트림을 처리합니다.

다음은 사용할 수 있는 몇 가지 변형 형태입니다.

- 전체 객체를 읽는 대신, 다음 C# 예제에서와 같이 요청에 바이트 범위를 지정하여 객체 데이터의 일부만 읽을 수 있습니다.

  Example

```
GetObjectRequest request = new GetObjectRequest
{
    BucketName = bucketName,
    Key = keyName,
    ByteRange = new ByteRange(0, 10)
};
```

- 객체를 검색할 때 필요할 경우 객체 가져오기 (p. 164) 객체를 사용하고 해당 요청 속성을 설정하여 필요할 경우 응답 헤더 값을 무시할 수 있습니다(`ResponseHeaderOverrides` 참조). 다음 C# 코드 예제에서는 이를 수행하는 방법을 보여 줍니다. 예를 들어 이 기능을 사용하여 객체 키 이름과 다른 파일 이름을 가진 파일로 다운로드해야 할 객체를 지정할 수 있습니다.

  Example

```
GetObjectRequest request = new GetObjectRequest
{
    BucketName = bucketName,
    Key = keyName
};

ResponseHeaderOverrides responseHeaders = new ResponseHeaderOverrides();
responseHeaders.CacheControl = "No-cache";
responseHeaders.ContentDisposition = "attachment; filename=testing.txt";

request.ResponseHeaderOverrides = responseHeaders;
```

Example

다음 C# 코드 예제는 Amazon S3 버킷에서 객체를 검색합니다. 이 예제는 `GetObjectResponse.ResponseStream` 속성을 사용하여 응답으로 전송된 객체 데이터를 읽습니다. 이 예제는 또한 `GetObjectResponse.Metadata` 모음을 사용하여 객체 메타데이터를 읽는 방법을 보여줍니다. 검색한 객체에 `x-amz-meta-title` 메타데이터가 있을 경우 코드는 메타데이터 값을 인쇄합니다.

실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

```
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.IO;
```

```
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class GetObjectTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** object key ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            ReadObjectDataAsync().Wait();
        }

        static async Task ReadObjectDataAsync()
        {
            string responseBody = "";
            try
            {
                GetObjectRequest request = new GetObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName
                };
                using (GetObjectResponse response = await client.GetObjectAsync(request))
                using (Stream responseStream = response.ResponseStream)
                using (StreamReader reader = new StreamReader(responseStream))
                {
                    string title = response.Metadata["x-amz-meta-title"]; // Assume you
 have "title" as medata added to the object.
                    string contentType = response.Headers["Content-Type"];
                    Console.WriteLine("Object metadata, Title: {0}", title);
                    Console.WriteLine("Content type: {0}", contentType);

                    responseBody = reader.ReadToEnd(); // Now you process the response
 body.
                }
            }
            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);
            }
        }
    }
}
```

PHP용 AWS SDK를 사용하여 객체 가져오기

이 주제에서는 PHP용 AWS SDK의 클래스를 사용하여 Amazon S3 객체를 검색하는 방법을 설명합니다. 전체 객체를 검색하거나 객체의 바이트 범위를 검색할 수 있습니다. 이미 PHP용 AWS SDK 사용 및 PHP 예제 실행 (p. 736)의 지침에 따라 PHP용 AWS SDK가 올바르게 설치되어 있다고 가정합니다.

다음 PHP 코드 예제에서와 같이 `ResponseContentType` 메서드에 응답 키인 `ResponseContentLanguage`, `ResponseContentDisposition`, `ResponseCacheControl`,

`ResponseExpires` 및 `getObject()`를 추가하여 필요할 경우 객체를 검색할 때 응답 헤더 값을 무시할 수 있습니다.

Example

```
$result = $s3->getObject([
    'Bucket'                     => $bucket,
    'Key'                        => $keyname,
    'ResponseContentType'        => 'text/plain',
    'ResponseContentLanguage'    => 'en-US',
    'ResponseContentDisposition' => 'attachment; filename=testing.txt',
    'ResponseCacheControl'       => 'No-cache',
    'ResponseExpires'            => gmdate(DATE_RFC2822, time() + 3600),
]);
```

객체 검색에 대한 자세한 내용은 객체 가져오기 (p. 164) 단원을 참조하십시오.

다음 PHP 예제는 객체를 검색한 후 브라우저에 객체 콘텐츠를 표시합니다. 이 예제에서는 `getObject()` 메서드를 사용하는 방법을 보여 줍니다. 이 가이드의 PHP 예제 실행에 대한 자세한 내용은 PHP 예제 실행 (p. 737) 단원을 참조하십시오.

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

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

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

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

try {
    // Get the object.
    $result = $s3->getObject([
        'Bucket' => $bucket,
        'Key'    => $keyname
    ]);

    // Display the object in the browser.
    header("Content-Type: {$result['ContentType']}");
    echo $result['Body'];
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

관련 리소스

- Amazon S3용 PHP용 AWS SDK Aws\S3\S3Client 클래스
- PHP용 AWS SDK 설명서

REST API를 사용하여 객체 가져오기

AWS SDK를 사용하여 버킷에서 객체 키를 가져올 수 있습니다. 하지만 애플리케이션에서 요구할 경우 REST 요청을 직접 전송할 수 있습니다. GET 요청을 전송하여 객체 키를 검색할 수 있습니다. 요청 및 응답 형식에 대한 자세한 내용은 Get Object 단원을 참조하십시오.

다른 사용자와 객체 공유

주제

- Visual Studio용 AWS Explorer를 사용하여 미리 서명된 객체 URL 생성 (p. 170)
- AWS SDK for Java를 사용하여 미리 서명된 객체 URL 생성 (p. 170)
- .NET용 AWS SDK를 사용하여 미리 서명된 객체 URL 생성 (p. 171)

- AWS CLI를 사용하여 미리 서명된 객체 URL 생성

모든 객체는 기본적으로 비공개이며, 객체 소유자만 이러한 객체에 액세스할 수 있습니다. 그러나 객체 소유자는 필요할 경우 자신의 보안 자격 증명을 사용하여 일정 기간 동안 객체 다운로드를 허가하는 미리 서명된 URL을 만들어 다른 사용자와 객체를 공유할 수 있습니다.

객체에 대해 미리 서명된 URL을 만들 때 보안 자격 증명을 제공하고 버킷 이름, 객체 키, HTTP 메서드(GET으로 객체 다운로드) 및 만료 날짜와 시간을 지정해야 합니다. 사전 서명된 URL은 지정된 기간 동안만 유효합니다.

미리 서명된 URL을 받은 사용자는 누구나 객체에 액세스할 수 있습니다. 예를 들어, 버킷에 동영상이 있고 버킷과 객체 모두 비공개인 경우 미리 서명된 URL을 만들어 다른 사용자와 동영상을 공유할 수 있습니다.

Note

- 유효한 보안 자격 증명을 가진 사용자는 누구나 미리 서명된 URL을 만들 수 있습니다. 단, 미리 서명된 URL에서 제공하려는 작업을 수행할 권한이 있는 사용자가 생성해야 이 URL을 통해 성공적으로 객체에 액세스할 수 있습니다.
- 미리 서명된 URL을 생성하는 데 사용할 수 있는 자격 증명에는 다음 사항이 포함됩니다.
  - IAM 인스턴스 프로파일: 최대 6시간 동안 유효함
  - AWS Security Token Service: AWS 계정 루트 사용자 또는 IAM 사용자의 자격 증명과 같은 영구 자격 증명을 통해 서명된 경우 최대 36시간 동안 유효함
  - IAM 사용자: AWS 서명 버전 4를 사용할 경우 최대 7일 동안 유효함

    최대 7일 동안 유효한 미리 서명된 URL을 생성하려면 먼저 사용 중인 SDK에 IAM 사용자 자격 증명(액세스 키 및 보안 액세스 키)을 지정합니다. 그런 다음 AWS 서명 버전 4를 사용하여 미리 서명된 URL을 생성합니다.
- 임시 토큰을 사용하여 미리 서명된 URL을 생성하면 이 URL이 토큰 만료보다 이후의 만료 시간으로 생성된 경우에도 토큰이 만료되면 이 URL도 만료됩니다.

AWS SDK for Java, .NET, 루비, PHP, Node.js 및 파이썬을 사용하여 프로그래밍 방식으로 미리 서명된 URL을 생성할 수 있습니다.

Visual Studio용 AWS Explorer를 사용하여 미리 서명된 객체 URL 생성

Visual Studio를 사용할 경우 코드를 작성할 필요 없이 Visual Studio용 AWS Explorer로 미리 서명된 객체 URL을 만들 수 있습니다. 이 URL만 있으면 누구나 객체를 다운로드할 수 있습니다. 자세한 내용은 AWS Explorer에서 Amazon S3 사용하기 단원을 참조하십시오.

AWS Explorer를 설치하는 방법에 대한 자세한 내용은 AWS SDK, CLI 및 Explorer 사용 (p. 727) 단원을 참조하십시오.

AWS SDK for Java를 사용하여 미리 서명된 객체 URL 생성

Example

다음 예제에서는 S3 버킷에서 객체를 검색할 수 있도록 다른 사용자에게 제공할 미리 서명된 URL을 생성합니다. 자세한 내용은 다른 사용자와 객체 공유 (p. 170) 단원을 참조하십시오.

실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.HttpMethod;
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;
import com.amazonaws.services.s3.model.GeneratePresignedUrlRequest;

import java.io.IOException;
import java.net.URL;

public class GeneratePresignedURL {

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

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();

            // Set the presigned URL to expire after one hour.
            java.util.Date expiration = new java.util.Date();
            long expTimeMillis = expiration.getTime();
            expTimeMillis += 1000 * 60 * 60;
            expiration.setTime(expTimeMillis);

            // Generate the presigned URL.
            System.out.println("Generating pre-signed URL.");
            GeneratePresignedUrlRequest generatePresignedUrlRequest =
                    new GeneratePresignedUrlRequest(bucketName, objectKey)
                            .withMethod(HttpMethod.GET)
                            .withExpiration(expiration);
            URL url = s3Client.generatePresignedUrl(generatePresignedUrlRequest);

            System.out.println("Pre-Signed URL: " + url.toString());
        } 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();
        }
    }
}
```

.NET용 AWS SDK를 사용하여 미리 서명된 객체 URL 생성

Example

다음 예제는 객체를 검색할 수 있도록 다른 사용자에게 제공할 미리 서명된 URL을 생성합니다. 자세한 내용은 다른 사용자와 객체 공유 (p. 170) 단원을 참조하십시오.

실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736)를 참조하십시오.

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

namespace Amazon.DocSamples.S3
{
    class GenPresignedURLTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string objectKey = "*** object key ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            string urlString = GeneratePreSignedURL();
        }
        static string GeneratePreSignedURL()
        {
            string urlString = "";
            try
            {
                GetPreSignedUrlRequest request1 = new GetPreSignedUrlRequest
                {
                    BucketName = bucketName,
                    Key = objectKey,
                    Expires = DateTime.Now.AddMinutes(5)
                };
                urlString = s3Client.GetPreSignedURL(request1);
            }
            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);
            }
            return urlString;
        }
    }
}
```

객체 업로드

Amazon S3는 업로드하는 데이터의 크기에 따라 다음과 같은 옵션을 제공합니다.

- 단일 작업으로 객체 업로드 - 단일 PUT 작업으로 최대 5GB 크기의 객체를 업로드할 수 있습니다.

  자세한 내용은 단일 작업으로 객체 업로드 (p. 173) 단원을 참조하십시오.
- 파트로 나누어 객체 업로드 - 멀티파트 업로드 API를 사용하여 최대 5TB의 대형 객체를 업로드할 수 있습니다.

멀티파트 업로드 API는 대용량 객체의 업로드 경험을 개선하기 위해 디자인되었습니다. 객체를 파트별로 업로드할 수 있습니다. 이러한 객체 파트는 임의의 순서로 독립적으로, 그리고 병렬적으로 업로드할 수 있습니다. 크기가 5MB에서 5TB까지인 객체에 대해 멀티파트 업로드를 사용할 수 있습니다. 자세한 내용은 멀티파트 업로드 API를 사용한 객체 업로드 (p. 177) 단원을 참조하십시오.

다음 방법으로 멀티파트 업로딩을 사용하는 것이 좋습니다.

- 안정적인 높은 대역폭 네트워크를 통해 큰 객체를 업로드하는 경우, 멀티파트 업로딩을 사용하여 멀티 스레드 성능을 위해 여러 객체 파트를 동시에 업로드함으로써 대역폭 사용을 극대화합니다.
- 불규칙한 네트워크를 통해 업로드하는 경우, 멀티파트 업로딩을 사용하여 업로드가 다시 시작되는 것을 방지하여 네트워크 오류에 대한 복원력을 높입니다. 멀티파트 업로딩을 사용하는 경우, 업로드 중에 중단된 파트만 다시 업로드해야 합니다. 객체 업로드를 처음부터 다시 시작하지 않아도 됩니다.

멀티파트 업로드에 대한 자세한 내용은 멀티파트 업로드 개요 (p. 178) 단원을 참조하십시오.

주제

- 단일 작업으로 객체 업로드 (p. 173)
- 멀티파트 업로드 API를 사용한 객체 업로드 (p. 177)
- 미리 서명된 URL을 사용하여 객체 업로드 (p. 205)

객체를 업로드할 때 원할 경우 Amazon S3가 객체를 암호화하여 디스크에 저장하고, 객체를 다운로드할 때 복호화하도록 요청할 수 있습니다. 자세한 내용은 암호화를 사용하여 데이터 보호 (p. 262) 단원을 참조하십시오.

관련 주제

AWS SDK, CLI 및 Explorer 사용 (p. 727)

단일 작업으로 객체 업로드

주제

- AWS SDK for Java를 사용하여 객체 업로드 (p. 173)
- .NET용 AWS SDK를 사용하여 객체 업로드 (p. 174)
- PHP용 AWS SDK를 사용하여 객체 업로드 (p. 176)
- Ruby용 AWS SDK를 사용하여 객체 업로드 (p. 176)
- REST API를 사용하여 객체 업로드 (p. 177)
- CLI를 사용하여 객체 업로드 (p. 177)

AWS SDK를 사용하여 객체를 업로드할 수 있습니다. SDK는 데이터를 간편하게 업로드할 수 있는 래퍼 라이브러리를 제공합니다. 그러나 필요할 경우 애플리케이션에서 바로 REST API를 사용할 수 있습니다. AWS Command Line Interface도 사용할 수 있습니다.

AWS SDK for Java를 사용하여 객체 업로드

Example

다음 예제에서는 객체 두 개를 만듭니다. 첫 번째 객체는 텍스트 문자열을 데이터로 갖고 있고, 두 번째 객체는 파일입니다. 이 예제에서는 `AmazonS3Client.putObject()`에 대한 호출로 직접 버킷 이름, 객체 키 및 텍스트 데이터를 지정하여 첫 번째 객체를 만듭니다. 이 예제에서는 버킷 이름, 객체 키 및 파일 경로를 지정하는 `PutObjectRequest`를 사용하여 두 번째 객체를 만듭니다. `PutObjectRequest`는 `ContentType` 헤더 및 제목 메타데이터도 지정합니다.

실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectMetadata;
import com.amazonaws.services.s3.model.PutObjectRequest;

import java.io.File;
import java.io.IOException;

public class UploadObject {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String stringObjKeyName = "*** String object key name ***";
        String fileObjKeyName = "*** File object key name ***";
        String fileName = "*** Path to file to upload ***";

        try {
            //This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .build();

            // Upload a text string as a new object.
            s3Client.putObject(bucketName, stringObjKeyName, "Uploaded String Object");

            // Upload a file as a new object with ContentType and title specified.
            PutObjectRequest request = new PutObjectRequest(bucketName, fileObjKeyName, new
 File(fileName));
            ObjectMetadata metadata = new ObjectMetadata();
            metadata.setContentType("plain/text");
            metadata.addUserMetadata("title", "someTitle");
            request.setMetadata(metadata);
            s3Client.putObject(request);
        } 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();
        }
    }
}
```

.NET용 AWS SDK를 사용하여 객체 업로드

Example

다음 C# 코드 예제에서는 `PutObjectRequest` 요청 두 개로 객체 두 개를 만듭니다.

- 첫 번째 `PutObjectRequest` 요청은 텍스트 문자열을 샘플 객체 데이터로 저장합니다. 또한 버킷과 객체 키 이름도 지정합니다.

- 두 번째 `PutObjectRequest` 요청은 파일 이름을 지정하여 파일을 업로드합니다. 이 요청은 `ContentType` 헤더와 객체 메타데이터(제목)도 지정합니다.

실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

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

namespace Amazon.DocSamples.S3
{
    class UploadObjectTest
    {
        private const string bucketName = "*** bucket name ***";
        // For simplicity the example creates two objects from the same file.
        // You specify key names for these objects.
        private const string keyName1 = "*** key name for first object created ***";
        private const string keyName2 = "*** key name for second object created ***";
        private const string filePath = @"*** file path ***";
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.EUWest1;

        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            WritingAnObjectAsync().Wait();
        }

        static async Task WritingAnObjectAsync()
        {
            try
            {
                // 1. Put object-specify only key name for the new object.
                var putRequest1 = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName1,
                    ContentBody = "sample text"
                };

                PutObjectResponse response1 = await client.PutObjectAsync(putRequest1);

                // 2. Put the object-set ContentType and add metadata.
                var putRequest2 = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName2,
                    FilePath = filePath,
                    ContentType = "text/plain"
                };
                putRequest2.Metadata.Add("x-amz-meta-title", "someTitle");
            }
            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);
            }
        }
    }
}
```

PHP용 AWS SDK를 사용하여 객체 업로드

이 단원에서는 PHP용 AWS SDK의 클래스를 사용하여 최대 5GB 크기의 객체를 업로드하는 방법을 설명합니다. 5GB보다 더 큰 파일은 멀티파트 업로드 API를 사용해야 합니다. 자세한 내용은 멀티파트 업로드 API를 사용한 객체 업로드 (p. 177) 단원을 참조하십시오.

이 주제에서는 이미 PHP용 AWS SDK 사용 및 PHP 예제 실행 (p. 736)의 지침에 따라 PHP용 AWS SDK가 올바르게 설치되어 있다고 가정합니다.

Example 데이터를 업로드하여 Amazon S3 버킷에 객체 생성

다음 PHP 예제는 `putObject()` 메서드를 사용한 데이터 업로드를 통해 지정된 버킷에 객체를 생성합니다. 이 가이드의 PHP 예제 실행에 대한 자세한 내용은 PHP 예제 실행 (p. 737) 단원을 참조하십시오.

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

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

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

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

try {
    // Upload data.
    $result = $s3->putObject([
        'Bucket' => $bucket,
        'Key'    => $keyname,
        'Body'   => 'Hello, world!',
        'ACL'    => 'public-read'
    ]);

    // Print the URL to the object.
    echo $result['ObjectURL'] . PHP_EOL;
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

관련 리소스

- Amazon S3용 PHP용 AWS SDK Aws\S3\S3Client 클래스
- PHP용 AWS SDK 설명서

Ruby용 AWS SDK를 사용하여 객체 업로드

Ruby용 AWS SDK 버전 3은 Amazon S3로 객체를 업로드하는 2가지 방법을 제공합니다. 첫 번째 방법은 간편하게 디스크에서 모든 크기의 파일을 업로드할 수 있는 관리형 파일 업로더를 사용합니다. 관리형 파일 업로더 방법을 사용하려면:

1. `Aws::S3::Resource` 클래스의 인스턴스를 만듭니다.
2. 버킷 이름과 키로 대상 객체를 참조합니다. 객체가 버킷에 상주하며 각 객체를 식별하는 고유의 키가 있습니다.
3. 객체의 `#upload_file`을 호출합니다.

Example

```
require 'aws-sdk-s3'

s3 = Aws::S3::Resource.new(region:'us-west-2')
obj = s3.bucket('bucket-name').object('key')
obj.upload_file('/path/to/source/file')
```

Ruby용 AWS SDK 버전 3에서 객체를 업로드하는 두 번째 방법은 `Aws::S3::Object`의 `#put` 메서드를 사용하는 것입니다. 객체가 문자열이거나 디스크에 저장된 파일이 아닌 I/O 객체인 경우 이 방법이 유용합니다. 이 방법을 사용하려면:

1. `Aws::S3::Resource` 클래스의 인스턴스를 만듭니다.
2. 버킷 이름과 키로 대상 객체를 참조합니다.
3. 문자열 또는 I/O 객체를 전달하는 `#put`을 호출합니다.

Example

```
require 'aws-sdk-s3'

s3 = Aws::S3::Resource.new(region:'us-west-2')
obj = s3.bucket('bucket-name').object('key')

# I/O object
File.open('/path/to/source.file', 'rb') do |file|
  obj.put(body: file)
end
```

REST API를 사용하여 객체 업로드

AWS SDK를 사용하여 객체를 업로드할 수 있습니다. 하지만 애플리케이션에서 요구할 경우 REST 요청을 직접 전송할 수 있습니다. PUT 요청을 전송하여 단일 작업에서 데이터를 업로드할 수 있습니다. 자세한 내용은 PUT Object 단원을 참조하십시오.

CLI를 사용하여 객체 업로드

AWS SDK를 사용하여 객체를 업로드할 수 있습니다. 하지만 애플리케이션에서 요구할 경우 AWS Command Line Interface 요청을 직접 전송할 수 있습니다. PUT 요청을 전송하여 단일 작업에서 데이터를 업로드할 수 있습니다. 자세한 내용은 AWS CLI Command Reference 안내서의 PUT Object를 참조하십시오.

멀티파트 업로드 API를 사용한 객체 업로드

주제

- 멀티파트 업로드 개요 (p. 178)
- 멀티파트 업로드용 AWS Java SDK(상위 수준 API) 사용 (p. 183)
- 멀티파트 업로드용 AWS Java SDK(하위 수준 API) 사용 (p. 187)
- 멀티파트 업로드에 .NET용 AWS SDK(상위 수준 API) 사용 (p. 191)

- 멀티파트 업로드에 .NET용 AWS SDK(하위 수준 API) 사용 (p. 197)
- 멀티파트 업로드를 위해 AWS PHP SDK 사용 (p. 201)
- 멀티파트 업로드용 AWS PHP SDK(하위 수준 API) 사용 (p. 202)
- 멀티파트 업로드를 위한 Ruby용 AWS SDK 사용 (p. 204)
- 멀티파트 업로드를 위해 REST API 사용 (p. 205)

- 멀티파트 업로드를 위한 AWS SDK for JavaScript in Node.js 사용

멀티파트 업로드를 사용하면 단일 객체를 여러 파트의 집합으로 업로드할 수 있습니다. 각 파트는 객체 데이터의 연속적인 부분입니다. 이러한 객체 파트는 독립적으로 그리고 임의의 순서로 업로드할 수 있습니다. 파트의 전송이 실패할 경우 다른 파트에 영향을 주지 않고도 해당 파트를 재전송할 수 있습니다. 객체의 모든 파트가 업로드되면 Amazon S3에서 이들 파트를 수집하여 객체를 생성합니다. 일반적으로 객체 크기가 100MB에 근접할 경우, 단일 작업에서 객체를 업로드하는 대신 멀티파트 업로드 사용을 고려해 봐야 합니다.

멀티파트 업로드 사용은 다음 이점을 제공합니다.

- 개선된 처리량 - 파트를 병렬적으로 업로드하여 처리량을 개선할 수 있습니다.
- 네트워크 문제로부터 빠른 복구 - 더 작아진 파트 크기는 네트워크 오류로 인해 실패한 업로드 재시작의 영향을 최소화합니다.
- 객체 업로드 일시 중지 및 재개 – 객체 파트를 장시간에 걸쳐 업로드할 수 있습니다. 일단 멀티파트 업로드가 시작되면 제한 시간이 없습니다. 멀티파트 업로드를 명시적으로 완료하거나 중단해야 합니다.
- 최종 객체 크기를 알기 전에 업로드를 시작 – 객체를 생성하는 동안 업로드할 수 있습니다.

자세한 내용은 멀티파트 업로드 개요 (p. 178) 단원을 참조하십시오.

멀티파트 업로드 개요

주제

- 동시 멀티파트 업로드 작업 (p. 179)
- 멀티파트 업로드 및 요금 (p. 180)
- 버킷 수명 주기 정책을 사용한 미완료 멀티파트 업로드 중단 (p. 180)
- Amazon S3 멀티파트 업로드 제한 (p. 181)
- 멀티파트 업로드를 위한 API 지원 (p. 181)
- 멀티파트 업로드 API 및 권한 (p. 182)

멀티파트 업로드 API를 사용하면 대형 객체를 여러 개로 나누어 업로드할 수 있습니다. 이 API를 사용하여 새 대형 객체를 업로드하거나 기존 객체의 복사본을 만들 수 있습니다(객체 관련 작업 (p. 163) 참조).

멀티파트 업로드는 3단계 프로세스입니다. 업로드를 시작한 후 각 부분별로 객체를 업로드하여 모두 업로드되면 멀티파트 업로드를 완료합니다. Amazon S3에 멀티파트 업로드 완료 요청이 전송되면 버킷의 다른 객체와 같이 이 객체에 액세스할 수 있도록 업로드된 각 부분을 모아 완전한 객체를 구성합니다.

진행 중인 모든 멀티파트 업로드 작업의 목록 또는 특정 멀티파트 업로드에 대해 업로드한 부분의 목록을 확인할 수 있습니다. 다음 단원에서 이러한 각 작업에 대해 자세히 설명합니다.

멀티파트 업로드 시작

멀티파트 업로드 시작 요청을 전송하면 Amazon S3가 멀티파트 업로드에 대한 고유 식별자인 업로드 ID와 함께 응답을 반환합니다. 파트 업로드, 부분 목록 확인, 업로드 완료 또는 업로드 중단 요청 시 항상 이 업로드 ID를 포함해야 합니다. 업로드할 객체에 메타데이터를 제공하려는 경우 멀티파트 업로드 시작 요청에서 메타데이터를 제공해야 합니다.

파트 업로드

부분을 업로드할 때 업로드 ID와 함께 부분 번호를 지정해야 합니다. 1부터 10,000까지 부분 번호를 지정할 수 있습니다. 부분 번호를 사용하여 업로드하는 객체에서 각 부분과 그 위치를 고유하게 식별합니다. 부분 번호는 굳이 연속 시퀀스로 선택할 필요가 없습니다(예를 들면 1, 5 및 14를 선택해도 됩니다). 이전에 업로드한 부분과 동일한 부분 번호로 새 부분을 업로드할 경우 이전에 업로드한 부분을 덮어쓰게 됩니다. 부분을 업로드할 때마다 Amazon S3는 그 응답으로 ETag 헤더를 반환합니다. 각 파트 업로드에 대해 부분 번호와 ETag 값을 기록해야 합니다. 이후 멀티파트 업로드를 완료하기 위한 요청에 이러한 값을 포함해야 하기 때문입니다.

> **Note**
>
> 멀티파트 업로드를 시작하여 하나 이상의 부분을 업로드한 후 멀티파트 업로드를 완료하거나 중단해야 업로드된 부분의 스토리지 비용이 청구되지 않습니다. 멀티파트 업로드를 완료 또는 중단한 후에만 Amazon S3가 부분 스토리지를 비우고 부분 스토리지에 대한 비용 청구를 중단합니다.

멀티파트 업로드 완료(또는 중단)

멀티파트 업로드를 완료하면 Amazon S3는 부분 번호를 바탕으로 오름차순으로 각 부분을 결합하여 객체를 완성합니다. 멀티파트 업로드 시작 요청에서 객체 메타데이터가 제공된 경우 Amazon S3는 객체에 해당 메타데이터를 포함시킵니다. 성공적으로 완료 요청이 수행되면 부분은 더 이상 존재하지 않습니다. 멀티파트 업로드 완료 요청에는 업로드 ID와 각 부분 번호 및 해당 ETag 값의 목록이 포함되어야 합니다. Amazon S3 응답에는 결합된 객체 데이터를 고유하게 식별하는 ETag가 포함됩니다. 이 ETag가 객체 데이터의 MD5 해시가 될 필요는 없습니다. 필요할 경우 멀티파트 업로드를 중단할 수 있습니다. 멀티파트 업로드를 중단한 후에는 해당 업로드 ID로 다시 부분을 업로드할 수 없습니다. 중단된 멀티파트 업로드의 부분이 소비하는 모든 스토리지는 다시 비워집니다. 파트 업로드가 진행 중일 때 멀티파트 업로드를 중단해도 이 업로드가 계속 수행될 수 있습니다. 따라서 각 부분에서 사용하고 있는 스토리지를 모두 비우려면 모든 파트 업로드가 완료된 후에 멀티파트 업로드를 중단해야 합니다.

멀티파트 업로드 나열

특정 멀티파트 업로드 또는 진행 중인 모든 멀티파트 업로드에 대해 부분 목록을 확인할 수 있습니다. 파트 목록 조회 작업은 특정 멀티파트 업로드에 대해 업로드한 부분의 정보를 반환합니다. 각 파트 목록 조회 요청에 대해 Amazon S3는 특정 멀티파트 업로드에서 최대 1,000개의 부분에 대해 부분 정보를 반환합니다. 멀티파트 업로드에서 1,000개 이상의 부분이 있을 경우 모든 부분을 검색하려면 파트 목록 조회 요청을 여러 번 반복해야 합니다. 반환된 부분 목록에는 업로드가 완료되지 않은 부분은 포함되지 않습니다. 멀티파트 업로드 나열 작업을 사용하여 진행 중인 멀티파트 업로드의 목록을 확인할 수 있습니다. 진행 중인 멀티파트 업로드는 시작했지만 아직 완료 또는 중단하지 않은 업로드입니다. 각 요청은 최대 1,000개의 멀티파트 업로드를 반환합니다. 진행 중인 멀티파트 업로드가 1,000개 이상일 경우 남은 멀티파트 업로드를 모두 검색하려면 추가 요청을 전송해야 합니다. 반환된 목록은 확인을 위해서만 사용합니다. 멀티파트 업로드 완료 요청을 전송할 때 이 나열 결과를 사용할 수 없습니다. 대신 부분을 업로드할 때 지정한 부분 번호 및 Amazon S3가 반환한 해당 ETag 값의 목록을 보관해야 합니다.

동시 멀티파트 업로드 작업

분산 개발 환경에서는 애플리케이션에서 한 객체에 대해 동시에 여러 업데이트를 시작할 수 있습니다. 애플리케이션은 동일한 객체 키를 사용하여 여러 멀티파트 업로드를 시작할 수 있습니다. 이러한 각 업로드에 대해 애플리케이션은 부분을 업로드한 다음 Amazon S3가 객체를 생성하도록 업로드 완료 요청을 전송할 수 있습니다. 버킷에서 버전 관리를 사용할 경우 멀티파트 업로드를 완료하면 항상 새 버전이 생성됩니다. 버전 관리를 사용하지 않은 버킷의 경우 멀티파트 업로드의 시작 및 완료 중간에 전송된 다른 요청을 우선 수행할 수 있습니다.

> **Note**
>
> 멀티파트 업로드의 시작 및 완료 중간에 전송된 다른 요청을 우선 수행할 수 있습니다. 예를 들어, 특정 키를 사용하여 멀티파트 업로드를 시작한 후 업로드가 완료되기 전에 다른 작업에서 해당 키를 삭제한 경우 객체가 없어도 멀티파트 업로드 완료 요청에 대해 객체 생성에 성공했다고 응답할 수 있습니다.

멀티파트 업로드 및 요금

멀티파트 업로드가 시작되면 Amazon S3는 업로드가 완료되거나 중단될 때까지 모든 부분을 계속 보존합니다. 수명 주기가 끝날 때까지 이 멀티파트 업로드와 관련 부분의 모든 스토리지, 대역폭 및 요청에 대해 비용이 청구됩니다. 멀티파트 업로드를 중단하면 Amazon S3가 업로드 결과 및 업로드된 모든 부분을 삭제하므로 더 이상 비용이 청구되지 않습니다. 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

버킷 수명 주기 정책을 사용한 미완료 멀티파트 업로드 중단

멀티파트 업로드를 개시한 후에 부분 업로드를 시작합니다. Amazon S3는 이 부분들을 저장하지만, 그 부분들을 모두 업로드하고 멀티파트 업로드를 완료하라는 `successful` 요청을 보낸 후에만 부분들로부터 객체를 생성합니다(멀티파트 업로드를 완료하라는 요청이 성공했는지 확인해야 합니다). 완전한 멀티파트 업로드 요청을 수신하자마자 Amazon S3는 부분들을 결합해 객체를 생성합니다.

완전한 멀티파트 업로드 요청을 성공적으로 전송하지 못하면, Amazon S3는 부분들을 결합하지 않고 어떤 객체도 생성하지 않을 것입니다. 따라서 부분들은 Amazon S3에 그대로 남게 되고 Amazon S3에 저장된 부분들에 대해서는 요금이 청구됩니다. 모범 사례로, 저장 비용을 최소화하도록 수명 주기 규칙(`AbortIncompleteMultipartUpload` 작업 사용)을 구성할 것을 권장합니다.

이제 Amazon S3는 시작된 후 지정 일수 내에 완료되지 않은 멀티파트 업로드를 중단하도록 Amazon S3에게 지시하는 데 사용할 수 있는 버킷 수명 주기 규칙을 지원합니다. 멀티파트 업로드가 시간 주기 내에 완료되지 않은 경우, 중단 작업을 할 수 있는 조건이 되어 Amazon S3는 멀티파트 업로드를 중단합니다(그리고 멀티파트 업로드에 연결된 부분들을 삭제합니다).

다음은 `AbortIncompleteMultipartUpload` 작업으로 규칙을 지정하는 수명 주기 구성의 예시입니다.

```
<LifecycleConfiguration>
    <Rule>
        <ID>sample-rule</ID>
        <Prefix></Prefix>
        <Status>Enabled</Status>
        <AbortIncompleteMultipartUpload>
          <DaysAfterInitiation>7</DaysAfterInitiation>
        </AbortIncompleteMultipartUpload>
    </Rule>
</LifecycleConfiguration>
```

이 예제에서 규칙은 `Prefix` 요소의 값(객체 키 이름 접두사)을 지정하지 않습니다. 따라서 멀티파트 업로드를 시작한 버킷의 모든 객체에 적용됩니다. 시작되고 7일 이내에 완료되지 않은 멀티파트 업로드는 중단 작업을 수행할 수 있습니다. 중단 작업은 완료된 멀티파트 업로드에 영향을 주지 않습니다.

버킷 수명 주기 구성에 대한 자세한 내용은 객체 수명 주기 관리 (p. 123) 단원을 참조하십시오.

> **Note**
>
> 멀티파트 업로드가 규칙에 지정된 일수 내에 완료되면, `AbortIncompleteMultipartUpload` 수명 주기 작업은 적용되지 않습니다(즉 Amazon S3는 어떤 작업도 하지 않습니다). 또한 이 작업은 객체에 적용되지 않고, 어떤 객체도 이 수명 주기 작업에 의해 삭제되지 않습니다.

다음 `put-bucket-lifecycle` CLI 명령은 지정된 버킷에 수명 주기 구성을 추가합니다.

```
$ aws s3api put-bucket-lifecycle  \
        --bucket bucketname  \
        --lifecycle-configuration filename-containing-lifecycle-configuration
```

CLI 명령을 테스트하려면 다음과 같이 합니다.

1. AWS CLI를 설치합니다. 지침은 AWS CLI 설정 (p. 733) 단원을 참조하십시오.

2. 파일(lifecycle.json)에 다음과 같은 수명 주기 구성 예시를 저장합니다. 예시 구성에서는 빈 접두사를 지정하므로 버킷에 있는 모든 객체에 적용됩니다. 접두사를 지정하여 정책을 객체의 하위 집합으로 제한할 수 있습니다.

```
{
    "Rules": [
        {
            "ID": "Test Rule",
            "Status": "Enabled",
            "Prefix": "",
            "AbortIncompleteMultipartUpload": {
                "DaysAfterInitiation": 7
            }
        }
    ]
}
```

3. 다음 CLI 명령을 사용하여 버킷에서 수명 주기 구성을 설정합니다.

```
aws s3api put-bucket-lifecycle   \
--bucket bucketname  \
--lifecycle-configuration file://lifecycle.json
```

4. 확인하려면 `get-bucket-lifecycle` CLI 명령을 사용해 수명 주기 구성을 조회합니다.

```
aws s3api get-bucket-lifecycle  \
--bucket bucketname
```

5. 수명 주기 구성을 삭제하려면 `delete-bucket-lifecycle` CLI 명령을 사용합니다.

```
aws s3api delete-bucket-lifecycle \
--bucket bucketname
```

Amazon S3 멀티파트 업로드 제한

다음 표에 멀티파트 업로드의 주요 사양이 나와 있습니다. 자세한 내용은 멀티파트 업로드 개요 (p. 178) 단원을 참조하십시오.

항목	사양
최대 객체 크기	5TB
업로드당 최대 부분 개수	10,000개
부분 번호	1 ~ 10,000(포함)
부분 크기	5MB~5GB, 마지막 파트는 < 5MB일 수 있음
파트 목록 조회 요청에 대해 반환되는 최대 부분 개수	1000
멀티파트 업로드 나열 요청에서 반환되는 최대 멀티파트 업로드 개수	1000

멀티파트 업로드를 위한 API 지원

AWS SDK를 사용하여 객체를 여러 부분으로 나누어 업로드할 수 있습니다. 다음 AWS SDK 라이브러리는 멀티파트 업로드를 지원합니다.

- AWS SDK for Java
- .NET용 AWS SDK
- PHP용 AWS SDK
- Ruby용 AWS SDK
- AWS SDK for Python (Boto)
- AWS SDK for JavaScript in Node.js

이들 라이브러리는 높은 수준의 추상화를 제공하여 간편하게 객체의 멀티파트 업로드를 수행할 수 있습니다. 그러나 필요할 경우 애플리케이션에서 바로 REST API를 사용할 수 있습니다. Amazon Simple Storage Service API Reference의 다음 단원에 멀티파트 업로드를 위한 REST API가 설명되어 있습니다.

- 멀티파트 업로드 시작
- 파트 업로드
- 파트 업로드(복사)
- 멀티파트 업로드 완료
- 멀티파트 업로드 중단
- 파트 목록 조회
- 멀티파트 업로드 목록 조회

멀티파트 업로드 API 및 권한

멀티파트 업로드 작업을 수행하려면 적절한 권한이 있어야 합니다. ACL(액세스 제어 목록), 버킷 정책 또는 사용자 정책을 사용하여 각 사용자에게 이 작업을 수행할 권한을 부여할 수 있습니다. 다음 표에는 ACL, 버킷 정책 또는 사용자 정책을 사용할 때 여러 멀티파트 업로드 작업에 대한 필수 권한이 나와 있습니다.

Action	필요한 권한
멀티파트 업로드 시작	멀티파트 업로드를 시작하려면 객체에 대해 `s3:PutObject` 작업을 수행할 수 있어야 합니다. 버킷 소유자는 다른 주체가 `s3:PutObject` 작업을 수행하도록 허가할 수 있습니다.
시작한 사용자	멀티파트 업로드를 시작한 사용자를 식별하는 컨테이너 요소입니다. 시작한 사용자가 AWS 계정일 경우 이 요소는 소유자 요소와 동일한 정보를 제공합니다. 시작한 사용자가 IAM 사용자일 경우 이 요소는 사용자 ARN 및 표시 이름을 제공합니다.
파트 업로드	부분을 업로드하려면 객체에 대해 `s3:PutObject` 작업을 수행할 수 있어야 합니다. 버킷 소유자는 멀티파트 작업을 시작한 사용자가 객체의 부분을 업로드할 수 있도록 객체에 대해 `s3:PutObject` 작업을 수행할 수 있는 권한을 부여해야 합니다.
파트 업로드(복사)	부분을 업로드하려면 객체에 대해 `s3:PutObject` 작업을 수행할 수 있어야 합니다. 기존 객체의 부분을 업로드하므로 원본 객체에 대해 `s3:GetObject` 작업을 수행할 수 있어야 합니다. 버킷 소유자는 객체의 멀티파트 업로드를 시작하는 사용자에게 `s3:PutObject` 작업을 수행할 수 있는 권한을 부여해야 합니다.
멀티파트 업로드 완료	멀티파트 업로드를 완료하려면 객체에 대해 `s3:PutObject` 작업을 수행할 수 있어야 합니다. 버킷 소유자는 멀티파트 작업을 시작한 사용자가 해당 객체의 멀티파트 업로드를 완료할 수 있도록 객체에 대해 `s3:PutObject` 작업을 수행할 수 있는 권한을 부여해야 합니다.

Action	필요한 권한
멀티파트 업로드 중단	멀티파트 업로드를 중단하려면 객체에 대해 `s3:AbortMultipartUpload` 작업을 수행할 수 있어야 합니다. 기본적으로 버킷 소유자와 멀티파트 업로드를 시작한 사용자는 이 작업을 수행할 수 있어야 합니다. 시작한 사용자가 IAM 사용자인 경우 해당 사용자의 AWS 계정 또한 이 멀티파트 업로드를 중단할 수 있어야 합니다. 버킷 소유자는 이러한 기본 사용자 외 다른 주체가 객체에 대해 `s3:AbortMultipartUpload` 작업을 수행하도록 허가할 수 있습니다. 버킷 소유자는 다른 주체의 `s3:AbortMultipartUpload` 작업 수행을 거부할 수 있습니다.
파트 목록 조회	`s3:ListMultipartUploadParts` 작업을 수행하여 멀티파트 업로드의 각 부분을 나열할 수 있어야 합니다. 기본적으로 버킷 소유자에게는 버킷에 대한 멀티파트 업로드의 부분을 나열할 권한이 부여됩니다. 멀티파트 업로드를 시작한 사용자는 특정 멀티파트 업로드의 부분을 나열할 권한이 있습니다. 멀티파트 업로드를 시작한 사용자가 IAM 사용자인 경우 이 IAM 사용자의 AWS 계정 또한 해당 업로드의 부분을 나열할 권한이 있습니다. 버킷 소유자는 이러한 기본 사용자 외 다른 주체가 객체에 대해 `s3:ListMultipartUploadParts` 작업을 수행하도록 허가할 수 있습니다. 버킷 소유자는 또한 다른 주체의 `s3:ListMultipartUploadParts` 작업 수행을 거부할 수 있습니다.
멀티파트 업로드 목록 조회	버킷에 대해 `s3:ListBucketMultipartUploads` 작업을 수행하여 해당 버킷에 대해 진행 중인 멀티파트 업로드를 나열할 수 있어야 합니다. 버킷 소유자는 이러한 기본 사용자 외, 다른 주체가 버킷에 대해 `s3:ListBucketMultipartUploads` 작업을 수행하도록 허가할 수 있습니다.
AWS KMS 암호화 및 암호 해독 관련 권한	AWS Key Management Service(AWS KMS) 고객 마스터 키(CMK)를 사용하는 암호화로 멀티파트 업로드를 수행하려면 요청자에게 키의 `kms:Encrypt`, `kms:Decrypt`, `kms:ReEncrypt*`, `kms:GenerateDataKey*` 및 `kms:DescribeKey` 작업에 대한 권한이 있어야 합니다. 이러한 권한이 필요한 이유는 Amazon S3이 멀티파트 업로드를 완료하기 전에 암호화된 파일 부분에서 데이터를 암호 해독하고 읽어야 하기 때문입니다. IAM 사용자 또는 역할이 AWS KMS CMK와 동일한 AWS 계정에 있는 경우 키 정책에 대한 이러한 권한이 있어야 합니다. IAM 사용자 또는 역할이 CMK와 다른 계정에 속한 경우 키 정책과 IAM 사용자 또는 역할 모두에 대한 권한이 있어야 합니다. >

ACL 권한 및 액세스 정책의 권한 간 관계에 대한 자세한 내용은 ACL 권한과 액세스 정책 권한의 매핑 (p. 441) 단원을 참조하십시오. IAM 사용자에 대한 자세한 내용은 사용자 및 그룹 작업 단원을 참조하십시오.

멀티파트 업로드용 AWS Java SDK(상위 수준 API) 사용

주제

- 파일 업로드 (p. 184)
- 멀티파트 업로드 중단 (p. 185)
- 멀티파트 업로드 진행률 추적 (p. 186)

AWS SDK for Java는 멀티파트 업로드를 간소화하는 `TransferManager`라는 상위 수준 API를 표시합니다 (멀티파트 업로드 API를 사용한 객체 업로드 (p. 177) 참조). 파일 또는 스트림에서 데이터를 업로드할 수

있습니다. 또한 멀티파트 업로드에 사용할 부분 크기, 또는 부분을 업로드할 때 사용할 동시 스레드 수와 같은 고급 옵션을 설정할 수 있습니다. 선택적 객체 속성, 스토리지 클래스 또는 ACL도 설정할 수 있습니다. 이러한 고급 옵션은 `PutObjectRequest` 및 `TransferManagerConfiguration` 클래스를 사용하여 설정합니다.

가능한 경우, `TransferManager`는 여러 스레드를 사용하여 단일 객체의 여러 부분을 한 번에 업로드하려고 시도합니다. 고대역폭에서 대용량 콘텐츠를 처리하는 경우 처리량이 크게 증가할 수 있습니다.

파일 업로드 기능 외에도, `TransferManager` 클래스를 사용하여 진행 중인 멀티파트 업로드를 중단할 수 있습니다. 사용자가 업로드를 시작한 후 완료 또는 중단할 때까지 업로드가 진행 중인 것으로 간주됩니다. `TransferManager`는 지정된 날짜 및 시간 이전에 시작된 지정된 버킷의 진행 중인 멀티파트 업로드를 모두 중단합니다.

하위 수준 API 메서드에서 제공하는 추가 기능을 포함하여 멀티파트 업로드에 대한 자세한 내용은 멀티파트 업로드 API를 사용한 객체 업로드 (p. 177) 단원을 참조하십시오.

파일 업로드

Example

다음 예제에서는 상위 수준 멀티파트 업로드 Java API(`TransferManager` 클래스)를 사용하여 객체를 업로드하는 방법을 보여 줍니다. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
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;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
import com.amazonaws.services.s3.transfer.Upload;

import java.io.File;

public class HighLevelMultipartUpload {

    public static void main(String[] args) throws Exception {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Object key ***";
        String filePath = "*** Path for file to upload ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();
            TransferManager tm = TransferManagerBuilder.standard()
                    .withS3Client(s3Client)
                    .build();

            // TransferManager processes all transfers asynchronously,
            // so this call returns immediately.
            Upload upload = tm.upload(bucketName, keyName, new File(filePath));
            System.out.println("Object upload started");

            // Optionally, wait for the upload to finish before continuing.
            upload.waitForCompletion();
            System.out.println("Object upload complete");
        } 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();
        }
    }
}
```

멀티파트 업로드 중단

Example

다음 예제에서는 상위 수준 Java API(`TransferManager` 클래스)를 사용하여 1주일 이상 전에 특정 버킷에서 시작된 진행 중인 멀티파트 업로드를 모두 중단합니다. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
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;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;

import java.util.Date;

public class HighLevelAbortMultipartUpload {

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

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();
            TransferManager tm = TransferManagerBuilder.standard()
                    .withS3Client(s3Client)
                    .build();

            // sevenDays is the duration of seven days in milliseconds.
            long sevenDays = 1000 * 60 * 60 * 24 * 7;
            Date oneWeekAgo = new Date(System.currentTimeMillis() - sevenDays);
            tm.abortMultipartUploads(bucketName, oneWeekAgo);
        } 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();
        }
    }
}
```

멀티파트 업로드 진행률 추적

상위 수준 Java 멀티파트 업로드 API는 객체를 Amazon S3에 업로드할 때 진행률을 추적하기 위한 수신 대기 인터페이스인 `ProgressListener`를 제공합니다. 진행률 이벤트는 바이트가 전송되었음을 리스너에 주기적으로 알립니다.

다음 예제에서는 `ProgressEvent` 이벤트 구독 및 핸들러 작성 방법을 보여 줍니다.

Example

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.event.ProgressEvent;
import com.amazonaws.event.ProgressListener;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
import com.amazonaws.services.s3.transfer.Upload;

import java.io.File;

public class HighLevelTrackMultipartUpload {

    public static void main(String[] args) throws Exception {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Object key ***";
        String filePath = "*** Path to file to upload ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();
            TransferManager tm = TransferManagerBuilder.standard()
                    .withS3Client(s3Client)
                    .build();
            PutObjectRequest request = new PutObjectRequest(bucketName, keyName, new
 File(filePath));

            // To receive notifications when bytes are transferred, add a
            // ProgressListener to your request.
            request.setGeneralProgressListener(new ProgressListener() {
                public void progressChanged(ProgressEvent progressEvent) {
                    System.out.println("Transferred bytes: " +
 progressEvent.getBytesTransferred());
                }
            });
            // TransferManager processes all transfers asynchronously,
            // so this call returns immediately.
            Upload upload = tm.upload(request);

            // Optionally, you can wait for the upload to finish before continuing.
            upload.waitForCompletion();
        } 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();
        }
    }
}
```

멀티파트 업로드용 AWS Java SDK(하위 수준 API) 사용

주제

- 파일 업로드 (p. 187)
- 멀티파트 업로드 목록 조회 (p. 189)
- 멀티파트 업로드 중단 (p. 190)

AWS SDK for Java는 멀티파트 업로드용 Amazon S3 REST API와 매우 유사한 하위 수준 API를 표시합니다(멀티파트 업로드 API를 사용한 객체 업로드 (p. 177) 참조). 멀티파트 업로드를 일시 중지했다 다시 시작해야 하거나 업로드 중에 부분 크기를 변경해야 하거나 혹은 업로드 데이터 크기를 미리 확인하지 않은 경우 하위 수준 API를 사용합니다. 이러한 요구 사항이 없는 경우에는 상위 수준 API를 사용하십시오(멀티파트 업로드용 AWS Java SDK(상위 수준 API) 사용 (p. 183) 참조).

파일 업로드

다음 예제에서는 하위 수준 Java 클래스를 사용하여 파일을 업로드하는 방법을 보여 줍니다. 여기에서는 다음 단계를 수행합니다.

- `AmazonS3Client.initiateMultipartUpload()` 메서드를 사용하여 멀티파트 업로드를 시작하고, `InitiateMultipartUploadRequest` 객체를 전달합니다.
- `AmazonS3Client.initiateMultipartUpload()` 메서드가 반환하는 업로드 ID를 저장합니다. 이후의 각 멀티파트 업로드 작업에서 이 업로드 ID를 제공합니다.
- 객체의 파트를 업로드합니다. 각 파트에 대해 `AmazonS3Client.uploadPart()` 메서드를 호출합니다. `UploadPartRequest` 객체를 사용하여 파트 업로드 정보를 제공합니다.
- 각 파트에 대해 `AmazonS3Client.uploadPart()` 메서드의 응답에서 얻은 ETag를 목록에 저장합니다. ETag 값을 사용하여 멀티파트 업로드를 완료합니다.
- `AmazonS3Client.completeMultipartUpload()` 메서드를 호출하여 멀티파트 업로드를 완료합니다.

Example

실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
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;
import com.amazonaws.services.s3.model.*;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class LowLevelMultipartUpload {
```

```
   public static void main(String[] args) throws IOException {
       Regions clientRegion = Regions.DEFAULT_REGION;
       String bucketName = "*** Bucket name ***";
       String keyName = "*** Key name ***";
       String filePath = "*** Path to file to upload ***";

       File file = new File(filePath);
       long contentLength = file.length();
       long partSize = 5 * 1024 * 1024; // Set part size to 5 MB.

       try {
           AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                   .withRegion(clientRegion)
                   .withCredentials(new ProfileCredentialsProvider())
                   .build();

           // Create a list of ETag objects. You retrieve ETags for each object part
uploaded,
           // then, after each individual part has been uploaded, pass the list of ETags
to
           // the request to complete the upload.
           List<PartETag> partETags = new ArrayList<PartETag>();

           // Initiate the multipart upload.
           InitiateMultipartUploadRequest initRequest = new
InitiateMultipartUploadRequest(bucketName, keyName);
           InitiateMultipartUploadResult initResponse =
s3Client.initiateMultipartUpload(initRequest);

           // Upload the file parts.
           long filePosition = 0;
           for (int i = 1; filePosition < contentLength; i++) {
               // Because the last part could be less than 5 MB, adjust the part size as
needed.
               partSize = Math.min(partSize, (contentLength - filePosition));

               // Create the request to upload a part.
               UploadPartRequest uploadRequest = new UploadPartRequest()
                       .withBucketName(bucketName)
                       .withKey(keyName)
                       .withUploadId(initResponse.getUploadId())
                       .withPartNumber(i)
                       .withFileOffset(filePosition)
                       .withFile(file)
                       .withPartSize(partSize);

               // Upload the part and add the response's ETag to our list.
               UploadPartResult uploadResult = s3Client.uploadPart(uploadRequest);
               partETags.add(uploadResult.getPartETag());

               filePosition += partSize;
           }

           // Complete the multipart upload.
           CompleteMultipartUploadRequest compRequest = new
CompleteMultipartUploadRequest(bucketName, keyName,
                   initResponse.getUploadId(), partETags);
           s3Client.completeMultipartUpload(compRequest);
       } 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();
        }
    }
}
```

멀티파트 업로드 목록 조회

Example

다음 예제에서는 하위 수준 Java API를 사용하여 진행 중인 멀티파트 업로드 목록을 검색하는 방법을 보여줍니다.

```
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;
import com.amazonaws.services.s3.model.ListMultipartUploadsRequest;
import com.amazonaws.services.s3.model.MultipartUpload;
import com.amazonaws.services.s3.model.MultipartUploadListing;

import java.util.List;

public class ListMultipartUploads {

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

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Retrieve a list of all in-progress multipart uploads.
            ListMultipartUploadsRequest allMultipartUploadsRequest = new
 ListMultipartUploadsRequest(bucketName);
            MultipartUploadListing multipartUploadListing =
 s3Client.listMultipartUploads(allMultipartUploadsRequest);
            List<MultipartUpload> uploads = multipartUploadListing.getMultipartUploads();

            // Display information about all in-progress multipart uploads.
            System.out.println(uploads.size() + " multipart upload(s) in progress.");
            for (MultipartUpload u : uploads) {
                System.out.println("Upload in progress: Key = \"" + u.getKey() + "\", id =
 " + u.getUploadId());
            }
        } 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();
        }
    }
}
```

멀티파트 업로드 중단

`AmazonS3Client.abortMultipartUpload()` 메서드를 호출하여 진행 중인 멀티파트 업로드를 중단할 수 있습니다. 이 메서드는 에 업로드된 모든 부분을 삭제하므로 리소스가 늘어나게 됩니다. 업로드 ID, 버킷 이름 및 키 이름을 제공합니다.

Example

다음 예제에서는 하위 수준 Java API를 사용하여 멀티파트 업로드를 중단하는 방법을 보여 줍니다.

```
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;
import com.amazonaws.services.s3.model.AbortMultipartUploadRequest;
import com.amazonaws.services.s3.model.ListMultipartUploadsRequest;
import com.amazonaws.services.s3.model.MultipartUpload;
import com.amazonaws.services.s3.model.MultipartUploadListing;

import java.util.List;

public class LowLevelAbortMultipartUpload {

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

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();

            // Find all in-progress multipart uploads.
            ListMultipartUploadsRequest allMultipartUploadsRequest = new
 ListMultipartUploadsRequest(bucketName);
            MultipartUploadListing multipartUploadListing =
 s3Client.listMultipartUploads(allMultipartUploadsRequest);

            List<MultipartUpload> uploads = multipartUploadListing.getMultipartUploads();
            System.out.println("Before deletions, " + uploads.size() + " multipart uploads
 in progress.");

            // Abort each upload.
            for (MultipartUpload u : uploads) {
                System.out.println("Upload in progress: Key = \"" + u.getKey() + "\", id =
 " + u.getUploadId());
                s3Client.abortMultipartUpload(new AbortMultipartUploadRequest(bucketName,
 u.getKey(), u.getUploadId()));
                System.out.println("Upload deleted: Key = \"" + u.getKey() + "\", id = " +
 u.getUploadId());
            }

            // Verify that all in-progress multipart uploads have been aborted.
            multipartUploadListing =
 s3Client.listMultipartUploads(allMultipartUploadsRequest);
            uploads = multipartUploadListing.getMultipartUploads();
            System.out.println("After aborting uploads, " + uploads.size() + " multipart
 uploads in progress.");
        } 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();
        }
    }
}
```

Note

멀티파트 업로드를 개별적으로 중단하는 대신, 특정 시간 이전에 시작된 진행 중인 멀티파트 업로드를 모두 중단할 수 있습니다. 이 정리 작업은 시작했지만 완료 또는 중단되지 않은 멀티파트 업로드를 중단할 때 유용합니다. 자세한 내용은 멀티파트 업로드 중단 (p. 185) 단원을 참조하십시오.

멀티파트 업로드에 .NET용 AWS SDK(상위 수준 API) 사용

주제

- .NET용 AWS SDK(상위 수준 API)를 사용하여 S3 버킷에 파일 업로드 (p. 191)
- 디렉터리 업로드 (p. 193)
- .NET용 AWS SDK(상위 수준 API)를 사용하여 S3 버킷으로 멀티파트 업로드 중단 (p. 194)
- .NET용 AWS SDK(상위 수준 API)를 사용하여 S3 버킷에 대한 멀티파트 업로드 진행 상황 추적 (p. 196)

.NET용 AWS SDK는 멀티파트 업로드를 간소화하는 상위 수준 API를 공개합니다(멀티파트 업로드 API를 사용한 객체 업로드 (p. 177)참조). 파일, 디렉터리 또는 스트림에서 데이터를 업로드할 수 있습니다. Amazon S3 멀티파트 업로드에 대한 자세한 내용은 멀티파트 업로드 개요 (p. 178) 단원을 참조하십시오.

`TransferUtility` 클래스는 파일 및 디렉터리를 업로드하고, 업로드 진행 상황을 추적하고, 멀티파트 업로드를 중단하는 방법을 제공합니다.

.NET용 AWS SDK(상위 수준 API)를 사용하여 S3 버킷에 파일 업로드

S3 버킷에 파일을 업로드하려면 `TransferUtility` 클래스를 사용합니다. 파일에서 데이터를 업로드할 경우에는 객체의 키 이름을 제공해야 합니다. 제공하지 않으면 API에서는 키 이름에 해당하는 파일 이름을 사용합니다. 스트림에서 데이터를 업로드할 경우에는 객체의 키 이름을 제공해야 합니다.

고급 업로드 옵션(예: 부분 크기, 부분을 동시에 업로드할 때 스레드 수, 메타데이터, 스토리지 클래스 또는 ACL)을 설정하려면 `TransferUtilityUploadRequest` 클래스를 사용합니다.

다음 C# 예제는 Amazon S3 버킷에 파일을 여러 부분으로 업로드합니다. 다양한 `TransferUtility.Upload` 오버로드를 사용하여 파일을 업로드하는 방법을 보여줍니다. 업로드에 대해 연속적으로 수행되는 각 호출이 이전 업로드를 대체합니다. 특정 버전의 .NET용 AWS SDK와 이 예제의 호환성 및 실제 예제를 작성하여 테스트하는 방법에 대한 지침은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

```
using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.IO;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadFileMPUHighLevelAPITest
    {
        private const string bucketName = "*** provide bucket name ***";
```

```
        private const string keyName = "*** provide a name for the uploaded object ***";
        private const string filePath = "*** provide the full path name of the file to
upload ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            UploadFileAsync().Wait();
        }

        private static async Task UploadFileAsync()
        {
            try
            {
                var fileTransferUtility =
                    new TransferUtility(s3Client);

                // Option 1. Upload a file. The file name is used as the object key name.
                await fileTransferUtility.UploadAsync(filePath, bucketName);
                Console.WriteLine("Upload 1 completed");

                // Option 2. Specify object key name explicitly.
                await fileTransferUtility.UploadAsync(filePath, bucketName, keyName);
                Console.WriteLine("Upload 2 completed");

                // Option 3. Upload data from a type of System.IO.Stream.
                using (var fileToUpload =
                    new FileStream(filePath, FileMode.Open, FileAccess.Read))
                {
                    await fileTransferUtility.UploadAsync(fileToUpload,
                                                bucketName, keyName);
                }
                Console.WriteLine("Upload 3 completed");

                // Option 4. Specify advanced settings.
                var fileTransferUtilityRequest = new TransferUtilityUploadRequest
                {
                    BucketName = bucketName,
                    FilePath = filePath,
                    StorageClass = S3StorageClass.StandardInfrequentAccess,
                    PartSize = 6291456, // 6 MB.
                    Key = keyName,
                    CannedACL = S3CannedACL.PublicRead
                };
                fileTransferUtilityRequest.Metadata.Add("param1", "Value1");
                fileTransferUtilityRequest.Metadata.Add("param2", "Value2");

                await fileTransferUtility.UploadAsync(fileTransferUtilityRequest);
                Console.WriteLine("Upload 4 completed");
            }
            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);
            }

        }
    }
```

```
}
```

추가 정보

.NET용 AWS SDK

디렉터리 업로드

`TransferUtility` 클래스를 사용하여 전체 디렉터리를 업로드할 수 있습니다. 기본적으로 API는 지정된 디렉터리의 루트에서만 파일을 업로드합니다. 하지만 모든 하위 디렉터리에서 반복적으로 파일을 업로드하도록 지정할 수 있습니다.

필터링 기준에 따라 지정된 디렉터리에서 파일을 선택하려면 필터링 표현식을 지정합니다. 예를 들어 디렉터리에서 .pdf 파일만 업로드하려면 `"*.pdf"` 필터 표현식을 지정합니다.

디렉터리에서 파일을 업로드할 경우 결과 객체의 키 이름을 지정하지 않습니다. Amazon S3가 원래 파일 경로를 사용하여 키 이름을 구성합니다. 예를 들어, 다음 구조의 `c:\myfolder` 디렉터리가 있다고 가정합니다.

Example

```
C:\myfolder
      \a.txt
      \b.pdf
      \media\
             An.mp3
```

이 디렉터리를 업로드하는 경우, Amazon S3는 다음 키 이름을 사용합니다.

Example

```
a.txt
b.pdf
media/An.mp3
```

Example

다음 C# 예제는 Amazon S3 버킷에 디렉터리를 업로드합니다. 다양한 `TransferUtility.UploadDirectory` 오버로드를 사용하여 디렉터리를 업로드하는 방법을 보여줍니다. 업로드에 대해 연속적으로 수행되는 각 호출이 이전 업로드를 대체합니다. 실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

```
using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.IO;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadDirMPUHighLevelAPITest
    {
        private const string existingBucketName = "*** bucket name ***";
        private const string directoryPath = @"*** directory path ***";
        // The example uploads only .txt files.
        private const string wildCard = "*.txt";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        static void Main()
```

```
        {
            s3Client = new AmazonS3Client(bucketRegion);
            UploadDirAsync().Wait();
        }

        private static async Task UploadDirAsync()
        {
            try
            {
                var directoryTransferUtility =
                    new TransferUtility(s3Client);

                // 1. Upload a directory.
                await directoryTransferUtility.UploadDirectoryAsync(directoryPath,
                    existingBucketName);
                Console.WriteLine("Upload statement 1 completed");

                // 2. Upload only the .txt files from a directory
                //    and search recursively.
                await directoryTransferUtility.UploadDirectoryAsync(
                                                directoryPath,
                                                existingBucketName,
                                                wildCard,
                                                SearchOption.AllDirectories);
                Console.WriteLine("Upload statement 2 completed");

                // 3. The same as Step 2 and some optional configuration.
                //    Search recursively for .txt files to upload.
                var request = new TransferUtilityUploadDirectoryRequest
                {
                    BucketName = existingBucketName,
                    Directory = directoryPath,
                    SearchOption = SearchOption.AllDirectories,
                    SearchPattern = wildCard
                };

                await directoryTransferUtility.UploadDirectoryAsync(request);
                Console.WriteLine("Upload statement 3 completed");
            }
            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);
            }
        }
    }
}
```

.NET용 AWS SDK(상위 수준 API)를 사용하여 S3 버킷으로 멀티파트 업로드 중단

진행 중인 멀티파트 업로드를 중단하려면 .NET용 AWS SDK에서 `TransferUtility` 클래스를 사용합니다. `DateTime` 값을 입력합니다. 그러면 API가 지정된 날짜 및 시간 이전에 시작된 멀티파트 업로드를 모두 중단하고 업로드된 파트를 제거합니다. 사용자가 업로드를 시작하는 순간부터 완료 또는 중단할 때까지 업로드가 진행 중인 것으로 간주됩니다.

업로드된 파트와 관련된 모든 스토리지에 대해 비용이 청구되므로 멀티파트 업로드를 완료하여 객체를 생성하거나 멀티파트 업로드를 중단하여 업로드된 파트를 모두 제거하는 것이 중요합니다. Amazon S3 멀티파트

업로드에 대한 자세한 내용은 멀티파트 업로드 개요 (p. 178) 단원을 참조하십시오. 요금에 대한 자세한 내용은 멀티파트 업로드 및 요금 (p. 180) 단원을 참조하십시오.

다음 C# 예제는 1주일 이상 전에 특정 버킷에서 시작되어 진행 중인 멀티파트 업로드를 모두 중단합니다. 특정 버전의 .NET용 AWS SDK와 이 예제의 호환성 및 실제 예제를 작성하여 테스트하는 방법에 대한 지침은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

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

namespace Amazon.DocSamples.S3
{
    class AbortMPUUsingHighLevelAPITest
    {
        private const string bucketName = "*** provide 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()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            AbortMPUAsync().Wait();
        }

        private static async Task AbortMPUAsync()
        {
            try
            {
                var transferUtility = new TransferUtility(s3Client);

                // Abort all in-progress uploads initiated before the specified date.
                await transferUtility.AbortMultipartUploadsAsync(
                    bucketName, DateTime.Now.AddDays(-7));
            }
            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);
            }
        }
    }
}
```

Note

특정 멀티파트 업로드를 중단할 수도 있습니다. 자세한 내용은 .NET용 AWS SDK(하위 수준)를 사용하여 S3 버킷에 멀티파트 업로드 나열 (p. 199) 단원을 참조하십시오.

추가 정보

.NET용 AWS SDK

.NET용 AWS SDK(상위 수준 API)를 사용하여 S3 버킷에 대한 멀티파트 업로드 진행 상황 추적

다음 C# 예제는 `TransferUtility` 클래스를 사용하여 S3 버킷에 파일을 업로드하고 업로드 진행률을 추적합니다. 특정 버전의 .NET용 AWS SDK와 이 예제의 호환성 및 실제 예제를 작성하여 테스트하는 방법에 대한 지침은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

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

namespace Amazon.DocSamples.S3
{
    class TrackMPUUsingHighLevelAPITest
    {
        private const string bucketName = "*** provide the bucket name ***";
        private const string keyName = "*** provide the name for the uploaded object ***";
        private const string filePath = " *** provide the full path name of the file to
 upload **";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;


        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            TrackMPUAsync().Wait();
        }

        private static async Task TrackMPUAsync()
        {
            try
            {
                var fileTransferUtility = new TransferUtility(s3Client);

                // 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 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);
            }
        }
```

```
        static void uploadRequest_UploadPartProgressEvent(object sender, UploadProgressArgs
 e)
        {
            // Process event.
            Console.WriteLine("{0}/{1}", e.TransferredBytes, e.TotalBytes);
        }
    }
}
```

추가 정보

.NET용 AWS SDK

멀티파트 업로드에 .NET용 AWS SDK(하위 수준 API) 사용

.NET용 AWS SDK는 멀티파트 업로드용 Amazon S3 REST API와 매우 유사한 하위 수준 API를 표시합니다(멀티파트 업로드를 위해 REST API 사용 (p. 205) 참조). 멀티파트 업로드를 일시 중지했다 다시 시작해야 하거나, 업로드 중에 파트의 크기를 변경해야 하거나, 데이터 크기를 미리 확인하지 않은 경우 하위 수준 API를 사용합니다. 이러한 요구 사항이 없으면 상위 수준 API를 사용하십시오(멀티파트 업로드에 .NET용 AWS SDK(상위 수준 API) 사용 (p. 191) 참조).

주제

- .NET용 AWS SDK(하위 수준 API)를 사용하여 S3 버킷에 파일 업로드 (p. 197)
- .NET용 AWS SDK(하위 수준)를 사용하여 S3 버킷에 멀티파트 업로드 나열 (p. 199)
- .NET용 AWS SDK(하위 수준)를 사용하여 S3 버킷에 대한 멀티파트 업로드 진행률 추적 (p. 200)
- .NET용 AWS SDK(하위 수준)를 사용하여 S3 버킷에 멀티파트 업로드 중단 (p. 200)

.NET용 AWS SDK(하위 수준 API)를 사용하여 S3 버킷에 파일 업로드

다음은 하위 수준 .NET용 AWS SDK 멀티파트 업로드 API를 사용하여 S3 버킷에 파일을 업로드하는 방법을 보여주는 C# 예제입니다. Amazon S3 멀티파트 업로드에 대한 내용은 멀티파트 업로드 개요 (p. 178) 단원을 참조하십시오.

> **Note**
>
> .NET용 AWS SDK API를 사용하여 대용량 객체를 업로드하는 경우 요청 스트림에 데이터를 기록하는 중 시간 초과가 발생할 수 있습니다. `UploadPartRequest`를 사용하여 제한 시간을 명시적으로 설정할 수 있습니다.

다음은 하위 수준 멀티파트 업로드 API를 사용하여 S3 버킷에 파일을 업로드하는 C# 예제입니다. 특정 버전의 .NET용 AWS SDK와 이 예제의 호환성 및 실제 예제를 작성하여 테스트하는 방법에 대한 지침은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

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

namespace Amazon.DocSamples.S3
{
    class UploadFileMPULowLevelAPITest
    {
```

```
        private const string bucketName = "*** provide bucket name ***";
        private const string keyName = "*** provide a name for the uploaded object ***";
        private const string filePath = "*** provide the full path name of the file to
upload ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            Console.WriteLine("Uploading an object");
            UploadObjectAsync().Wait();
        }

        private static async Task UploadObjectAsync()
        {
            // Create list to store upload part responses.
            List<UploadPartResponse> uploadResponses = new List<UploadPartResponse>();

            // Setup information required to initiate the multipart upload.
            InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
            {
                BucketName = bucketName,
                Key = keyName
            };

            // Initiate the upload.
            InitiateMultipartUploadResponse initResponse =
                await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // Upload parts.
            long contentLength = new FileInfo(filePath).Length;
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

            try
            {
                Console.WriteLine("Uploading parts");

                long filePosition = 0;
                for (int i = 1; filePosition < contentLength; i++)
                {
                    UploadPartRequest uploadRequest = new UploadPartRequest
                        {
                            BucketName = bucketName,
                            Key = keyName,
                            UploadId = initResponse.UploadId,
                            PartNumber = i,
                            PartSize = partSize,
                            FilePosition = filePosition,
                            FilePath = filePath
                        };

                    // Track upload progress.
                    uploadRequest.StreamTransferProgress +=
                        new
EventHandler<StreamTransferProgressArgs>(UploadPartProgressEventCallback);

                    // Upload a part and add the response to our list.
                    uploadResponses.Add(await s3Client.UploadPartAsync(uploadRequest));

                    filePosition += partSize;
                }

                // Setup to complete the upload.
```

```
                CompleteMultipartUploadRequest completeRequest = new
 CompleteMultipartUploadRequest
                    {
                        BucketName = bucketName,
                        Key = keyName,
                        UploadId = initResponse.UploadId
                     };
                completeRequest.AddPartETags(uploadResponses);

                // Complete the upload.
                CompleteMultipartUploadResponse completeUploadResponse =
                    await s3Client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (Exception exception)
            {
                Console.WriteLine("An AmazonS3Exception was thrown: { 0}",
 exception.Message);

                // Abort the upload.
                AbortMultipartUploadRequest abortMPURequest = new
 AbortMultipartUploadRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    UploadId = initResponse.UploadId
                };
               await s3Client.AbortMultipartUploadAsync(abortMPURequest);
            }
        }
        public static void UploadPartProgressEventCallback(object sender,
 StreamTransferProgressArgs e)
        {
            // Process event.
            Console.WriteLine("{0}/{1}", e.TransferredBytes, e.TotalBytes);
        }
    }
}
```

추가 정보

.NET용 AWS SDK

.NET용 AWS SDK(하위 수준)를 사용하여 S3 버킷에 멀티파트 업로드 나열

특정 버킷에 대해 진행 중인 멀티파트 업로드를 모두 나열하려면 .NET용 AWS SDK 하위 수준 멀티파트 업로드 API의 `ListMultipartUploadsRequest` 클래스를 사용합니다. `AmazonS3Client.ListMultipartUploads` 메서드는 진행 중인 멀티파트 업로드에 대한 정보를 제공하는 `ListMultipartUploadsResponse` 클래스의 인스턴스를 반환합니다.

여기에서 진행 중인 멀티파트 업로드란 멀티파트 업로드 요청 시작을 통해 시작되었지만 아직 완료되거나 중단되지 않은 멀티파트 업로드를 말합니다. Amazon S3 멀티파트 업로드에 대한 자세한 내용은 멀티파트 업로드 개요 (p. 178) 단원을 참조하십시오.

다음 C# 예제는 .NET용 AWS SDK를 사용하여 버킷에 대해 진행 중인 모든 멀티파트 업로드를 나열하는 방법을 보여줍니다. 특정 버전의 .NET용 AWS SDK와 이 예제의 호환성 및 실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

```
ListMultipartUploadsRequest request = new ListMultipartUploadsRequest
{
  BucketName = bucketName // Bucket receiving the uploads.
};
```

```
ListMultipartUploadsResponse response = await
 AmazonS3Client.ListMultipartUploadsAsync(request);
```

추가 정보

.NET용 AWS SDK

.NET용 AWS SDK(하위 수준)를 사용하여 S3 버킷에 대한 멀티파트 업로드 진행률 추적

멀티파트 업로드 진행률을 추적하려면 .NET용 AWS SDK 하위 수준 멀티파트 업로드 API에서 제공하는 `UploadPartRequest.StreamTransferProgress` 이벤트를 사용합니다. 이 이벤트는 정기적으로 발생하며, 전송할 총 바이트 수와 전송된 바이트 수 등과 같은 정보를 반환합니다.

다음 C# 예제는 멀티파트 업로드 진행률을 추적하는 방법을 보여줍니다. 다음 코드가 포함된 전체 C# 예제는 .NET용 AWS SDK(하위 수준 API)를 사용하여 S3 버킷에 파일 업로드 (p. 197) 단원을 참조하십시오.

```
UploadPartRequest uploadRequest = new UploadPartRequest
{
// Provide the request data.
};

uploadRequest.StreamTransferProgress +=
     new EventHandler<StreamTransferProgressArgs>(UploadPartProgressEventCallback);

...
public static void UploadPartProgressEventCallback(object sender,
 StreamTransferProgressArgs e)
{
    // Process the event.
    Console.WriteLine("{0}/{1}", e.TransferredBytes, e.TotalBytes);
}
```

추가 정보

.NET용 AWS SDK

.NET용 AWS SDK(하위 수준)를 사용하여 S3 버킷에 멀티파트 업로드 중단

Amazon S3 메서드를 호출하여 진행 중인 멀티파트 업로드를 중단할 수 있습니다. 이 메서드는 업로드 중단 이외에 로 업로드된 모든 부분을 삭제합니다.

멀티파트 업로드를 중단하려면 업로드에 사용된 업로드 ID, 버킷 및 키 이름을 제공해야 합니다. 멀티파트 업로드를 중단하면 해당 업로드 ID를 사용해 추가 파트를 업로드할 수 없습니다. Amazon S3 멀티파트 업로드에 대한 자세한 내용은 멀티파트 업로드 개요 (p. 178) 단원을 참조하십시오.

다음 C# 예제는 멀티파트 업로드를 중단하는 방법을 보여줍니다. 다음 코드가 포함된 전체 C# 예제는 .NET용 AWS SDK(하위 수준 API)를 사용하여 S3 버킷에 파일 업로드 (p. 197) 단원을 참조하십시오.

```
AbortMultipartUploadRequest abortMPURequest = new AbortMultipartUploadRequest
{
    BucketName = existingBucketName,
    Key = keyName,
    UploadId = initResponse.UploadId
};
await AmazonS3Client.AbortMultipartUploadAsync(abortMPURequest);
```

또한 특정 시간 이전에 시작되어 진행 중인 멀티파트 업로드를 중단할 수도 있습니다. 이 정리 작업은 완료 또는 중단되지 않은 멀티파트 업로드를 중단할 때 유용합니다. 자세한 내용은 .NET용 AWS SDK(상위 수준 API)를 사용하여 S3 버킷으로 멀티파트 업로드 중단 (p. 194) 단원을 참조하십시오.

추가 정보

.NET용 AWS SDK

멀티파트 업로드를 위해 AWS PHP SDK 사용

대용량 파일을 분할하여 Amazon S3에 업로드할 수 있습니다. 멀티파트 업로드를 사용하려면 파일이 5GB보다 더 커야 합니다. PHP용 AWS SDK는 멀티파트 업로드를 간소화하는 MultipartUploader 클래스를 표시합니다.

`upload` 클래스의 `MultipartUploader` 메서드는 단순한 멀티파트 업로드에 가장 적합합니다. 멀티파트 업로드를 일시 중지했다 다시 시작해야 하거나 업로드 중에 파트의 크기를 변경해야 하거나 혹은 데이터 크기를 미리 확인하지 않은 경우 하위 수준 PHP API를 사용합니다. 자세한 내용은 멀티파트 업로드용 AWS PHP SDK(하위 수준 API) 사용 (p. 202) 단원을 참조하십시오.

멀티파트 업로드에 대한 자세한 내용은 멀티파트 업로드 API를 사용한 객체 업로드 (p. 177) 단원을 참조하십시오. 크기가 5GB 미만인 파일의 업로드에 대한 자세한 내용은 PHP용 AWS SDK를 사용하여 객체 업로드 (p. 176) 단원을 참조하십시오.

상위 수준 멀티파트 업로드를 사용한 파일 업로드

이 주제에서는 멀티파트 파일 업로드를 위해 PHP용 AWS SDK의 상위 수준 `Aws\S3\Model\MultipartUpload\UploadBuilder` 클래스를 사용하는 방법을 설명합니다. 이미 PHP용 AWS SDK 사용 및 PHP 예제 실행 (p. 736)의 지침에 따라 PHP용 AWS SDK가 올바르게 설치되어 있다고 가정합니다.

다음은 Amazon S3 버킷에 파일을 업로드하는 PHP 예제입니다. 이 예제는 `MultipartUploader` 객체에 대한 파라미터를 설정하는 방법을 보여 줍니다.

이 가이드의 PHP 예제 실행에 대한 자세한 내용은 PHP 예제 실행 (p. 737) 단원을 참조하십시오.

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

use Aws\Common\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

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

// Prepare the upload parameters.
$uploader = new MultipartUploader($s3, '/path/to/large/file.zip', [
    'bucket' => $bucket,
    'key'    => $keyname
]);

// Perform the upload.
try {
    $result = $uploader->upload();
    echo "Upload complete: {$result['ObjectURL']}" . PHP_EOL;
} catch (MultipartUploadException $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

관련 리소스

- Amazon S3용 PHP용 AWS SDK Aws\S3\S3Client 클래스

- Amazon S3 멀티파트 업로드
- PHP용 AWS SDK 설명서

멀티파트 업로드용 AWS PHP SDK(하위 수준 API) 사용

주제

- PHP SDK 하위 수준 API를 사용한 파일 분할 업로드 (p. 202)
- 하위 수준 PHP용 AWS SDK API를 사용한 멀티파트 업로드 나열 (p. 203)
- 멀티파트 업로드 중단 (p. 204)

PHP용 AWS SDK는 멀티파트 업로드용 Amazon S3 REST API와 매우 유사한 하위 수준 API를 표시합니다(멀티파트 업로드를 위해 REST API 사용 (p. 205) 참조). 멀티파트 업로드를 일시 중지했다 다시 시작해야 하거나 업로드 중에 파트 크기를 변경해야 하거나 혹은 데이터 크기를 미리 확인하지 않은 경우 하위 수준 API를 사용합니다. 이러한 요구 사항이 없는 경우에는 언제나 PHP용 AWS SDK 상위 수준 추상화를 사용하십시오(멀티파트 업로드를 위해 AWS PHP SDK 사용 (p. 201) 참조).

PHP SDK 하위 수준 API를 사용한 파일 분할 업로드

이 주제는 PHP용 AWS SDK 버전 3에서 하위 레벨 `uploadPart` 메서드를 사용하여 파일을 여러 부분으로 업로드하는 방법을 보여줍니다. 이미 PHP용 AWS SDK 사용 및 PHP 예제 실행 (p. 736)의 지침에 따라 PHP용 AWS SDK가 올바르게 설치되어 있다고 가정합니다.

다음은 하위 수준 PHP API 멀티파트 업로드를 사용하여 Amazon S3 버킷에 파일을 업로드하는 PHP 코드 예제입니다. 이 가이드의 PHP 예제 실행에 대한 자세한 내용은 PHP 예제 실행 (p. 737) 단원을 참조하십시오.

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

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';
$filename = '*** Path to and Name of the File to Upload ***';

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

$result = $s3->createMultipartUpload([
    'Bucket'       => $bucket,
    'Key'          => $keyname,
    'StorageClass' => 'REDUCED_REDUNDANCY',
    'ACL'          => 'public-read',
    'Metadata'     => [
        'param1' => 'value 1',
        'param2' => 'value 2',
        'param3' => 'value 3'
    ]
]);
$uploadId = $result['UploadId'];

// Upload the file in parts.
try {
    $file = fopen($filename, 'r');
    $partNumber = 1;
    while (!feof($file)) {
        $result = $s3->uploadPart([
            'Bucket'     => $bucket,
```

```
            'Key'        => $keyname,
            'UploadId'   => $uploadId,
            'PartNumber' => $partNumber,
            'Body'       => fread($file, 5 * 1024 * 1024),
        ]);
        $parts['Parts'][$partNumber] = [
            'PartNumber' => $partNumber,
            'ETag' => $result['ETag'],
        ];
        $partNumber++;

        echo "Uploading part {$partNumber} of {$filename}." . PHP_EOL;
    }
    fclose($file);
} catch (S3Exception $e) {
    $result = $s3->abortMultipartUpload([
        'Bucket'   => $bucket,
        'Key'      => $keyname,
        'UploadId' => $uploadId
    ]);

    echo "Upload of {$filename} failed." . PHP_EOL;
}

// Complete the multipart upload.
$result = $s3->completeMultipartUpload([
    'Bucket'   => $bucket,
    'Key'      => $keyname,
    'UploadId' => $uploadId,
    'MultipartUpload'    => $parts,
]);
$url = $result['Location'];

echo "Uploaded {$filename} to {$url}." . PHP_EOL;
```

관련 리소스

- Amazon S3용 PHP용 AWS SDK Aws\S3\S3Client 클래스
- Amazon S3 멀티파트 업로드
- PHP용 AWS SDK 설명서

하위 수준 PHP용 AWS SDK API를 사용한 멀티파트 업로드 나열

이 주제에서는 PHP용 AWS SDK 버전 3의 하위 수준 API 클래스를 사용하여 버킷에서 진행 중인 멀티파트 업로드를 모두 나열하는 방법을 보여줍니다. 이미 PHP용 AWS SDK 사용 및 PHP 예제 실행 (p. 736)의 지침에 따라 PHP용 AWS SDK가 올바르게 설치되어 있다고 가정합니다.

다음은 버킷에서 진행 중인 멀티파트 업로드를 모두 나열하는 방법을 보여 주는 PHP 예제입니다.

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

use Aws\S3\S3Client;

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

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

// Retrieve a list of the current multipart uploads.
```

```
$result = $s3->listMultipartUploads([
    'Bucket' => $bucket
]);

// Write the list of uploads to the page.
print_r($result->toArray());
```

관련 리소스

- Amazon S3용 PHP용 AWS SDK Aws\S3\S3Client 클래스
- Amazon S3 멀티파트 업로드
- PHP용 AWS SDK 설명서

멀티파트 업로드 중단

이 단원에서는 진행 중인 멀티파트 업로드를 중단하기 위해 PHP용 AWS SDK 버전 3의 클래스를 사용하는 방법을 설명합니다. 이미 PHP용 AWS SDK 사용 및 PHP 예제 실행 (p. 736)의 지침에 따라 PHP용 AWS SDK가 올바르게 설치되어 있다고 가정합니다.

다음은 `abortMultipartUpload()` 메서드를 사용하여 진행 중인 멀티파트 업로드를 중단하는 방법을 보여주는 PHP 예제입니다. 이 가이드의 PHP 예제 실행에 대한 자세한 내용은 PHP 예제 실행 (p. 737) 단원을 참조하십시오.

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

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';
$uploadId = '*** Upload ID of upload to Abort ***';

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

// Abort the multipart upload.
$s3->abortMultipartUpload([
    'Bucket'   => $bucket,
    'Key'      => $keyname,
    'UploadId' => $uploadId,
]);
```

관련 리소스

- Amazon S3용 PHP용 AWS SDK Aws\S3\S3Client 클래스
- Amazon S3 멀티파트 업로드
- PHP용 AWS SDK 설명서

멀티파트 업로드를 위한 Ruby용 AWS SDK 사용

Ruby용 AWS SDK 버전 3는 Amazon S3 멀티파트 업로드를 두 가지 방법으로 지원합니다. 첫 번째 옵션으로 관리형 파일 업로드를 사용할 수 있습니다. 자세한 내용은 AWS 개발자 블로그에서 Amazon S3에 파일 업로드를 참조하십시오. 관리형 파일 업로드는 파일을 버킷에 업로드하는 데 권장되는 방법입니다. 관리형 파일 업로드에는 다음과 같은 이점이 있습니다.

- 15MB를 초과하는 객체에 대해 멀티파트 업로드를 관리합니다.

- 인코딩 문제를 방지하기 위해 이진 모드에서 파일을 올바로 엽니다.
- 대형 객체의 부분 업로드에 대한 여러 스레드를 병렬로 사용합니다.

또는 다음의 멀티파트 업로드 클라이언트 작업을 직접 사용할 수 있습니다.

- create_multipart_upload – 멀티파트 업로드를 시작하고 업로드 ID를 반환합니다.
- upload_part – 멀티파트 업로드에서 파트를 업로드합니다.
- upload_part_copy – 기존 객체에서 데이터를 복사하여 파트를 데이터 소스로 업로드합니다.
- complete_multipart_upload – 이전에 업로드된 파트를 어셈블하여 멀티파트 업로드를 완료합니다.
- abort_multipart_upload – 멀티파트 업로드를 중단합니다.

자세한 내용은 Ruby용 AWS SDK 버전 3 사용 (p. 737) 단원을 참조하십시오.

멀티파트 업로드를 위해 REST API 사용

Amazon Simple Storage Service API Reference의 다음 단원에 멀티파트 업로드를 위한 REST API가 설명되어 있습니다.

- 멀티파트 업로드 시작
- 파트 업로드
- 멀티파트 업로드 완료
- 멀티파트 업로드 중단
- 파트 목록 조회
- 멀티파트 업로드 목록 조회

이러한 API를 사용하여 REST 요청을 만들거나, 제공된 SDK 중 하나를 사용할 수 있습니다. SDK에 대한 자세한 내용은 멀티파트 업로드를 위한 API 지원 (p. 181)을 참조하십시오.

미리 서명된 URL을 사용하여 객체 업로드

주제

- the section called “Java용 SDK 사용” (p. 206)
- the section called “.NET용 AWS SDK 사용” (p. 207)
- the section called “Ruby용 AWS SDK 사용” (p. 208)
- 미리 서명된 URL을 사용하여 객체 업로드(PHP용 AWS SDK)

미리 서명된 URL의 생성자가 해당 객체에 대한 액세스 권한을 보유할 경우, 미리 서명된 URL은 URL에서 식별된 객체에 대한 액세스를 부여합니다. 즉, 객체를 업로드하기 위해 미리 서명된 URL을 수신하는 경우, 미리 서명된 URL의 생성자가 해당 객체를 업로드하는 데 필요한 권한을 보유하는 경우에만 객체를 업로드할 수 있습니다.

모든 객체 및 버킷은 기본적으로 비공개입니다. 사용자/고객이 특정 객체를 버킷에 업로드할 수 있기를 원하는 경우 미리 서명된 URL이 유용하지만 AWS 보안 자격 증명 또는 권한을 요구하지는 않습니다. 미리 서명된 URL을 생성하는 경우 보안 자격 증명을 제공한 후 버킷 이름, 객체 키, HTTP 메서드(객체 업로드 PUT) 및 만료 날짜/시간을 지정해야 합니다. 사전 서명된 URL은 지정된 기간 동안만 유효합니다.

AWS SDK for Java, .NET, 루비, PHP, Node.js 및 파이썬을 사용하여 프로그래밍 방식으로 미리 서명된 URL을 생성할 수 있습니다. Microsoft Visual Studio를 사용하는 경우 AWS Explorer를 사용하여 코드를 작성하지

않고도 미리 서명된 객체 URL을 생성할 수도 있습니다. 누구라도 유효한 사전 서명된 URL을 수신하면 프로그래밍 방식으로 객체를 업로드할 수 있습니다.

자세한 내용은 AWS Explorer에서 Amazon S3 사용하기 단원을 참조하십시오.

AWS Explorer를 설치하는 방법에 대한 자세한 내용은 AWS SDK, CLI 및 Explorer 사용 (p. 727) 단원을 참조하십시오.

Note

유효한 보안 자격 증명을 가진 사용자는 누구나 미리 서명된 URL을 만들 수 있습니다. 단, 미리 서명된 URL에서 제공하려는 작업을 수행할 권한이 있는 사용자가 생성해야 이 URL을 통해 성공적으로 객체를 업로드할 수 있습니다.

미리 서명된 URL을 사용하는 객체 업로드(AWS SDK for Java)

AWS SDK for Java를 사용하여 미리 서명된 URL을 생성할 수 있습니다. 그러면 이 URL을 제공받은 누구나 이 URL을 사용하여 Amazon S3에 객체를 업로드할 수 있습니다. URL을 사용하여 객체를 업로드하는 경우가 지정된 버킷에 객체를 생성합니다. 미리 서명된 URL에 지정된 동일한 키를 사용하는 객체가 버킷에 이미 있으면 Amazon S3는 업로드된 객체로 기존 객체를 바꿉니다. 업로드를 성공적으로 완료하려면 다음을 수행해야 합니다.

- `GeneratePresignedUrlRequest` 및 `HttpURLConnection` 객체를 생성할 때 HTTP PUT 동사를 지원합니다.
- 업로드 완료 후 몇 가지 방식으로 `HttpURLConnection` 객체와 상호 작용합니다. 다음 예제에서는 `HttpURLConnection` 객체를 사용하여 상호 작용하여 HTTP 응답 코드를 확인합니다.

Example

이 예제에서는 미리 서명된 URL을 생성한 후 사용하여 샘플 데이터를 객체로 업로드합니다. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.HttpMethod;
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;
import com.amazonaws.services.s3.model.GeneratePresignedUrlRequest;
import com.amazonaws.services.s3.model.S3Object;

import java.io.IOException;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;

public class GeneratePresignedUrlAndUploadObject {

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

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();
```

```
            // Set the pre-signed URL to expire after one hour.
            java.util.Date expiration = new java.util.Date();
            long expTimeMillis = expiration.getTime();
            expTimeMillis += 1000 * 60 * 60;
            expiration.setTime(expTimeMillis);

            // Generate the pre-signed URL.
            System.out.println("Generating pre-signed URL.");
            GeneratePresignedUrlRequest generatePresignedUrlRequest = new
 GeneratePresignedUrlRequest(bucketName, objectKey)
                    .withMethod(HttpMethod.PUT)
                    .withExpiration(expiration);
            URL url = s3Client.generatePresignedUrl(generatePresignedUrlRequest);

            // Create the connection and use it to upload the new object using the pre-
signed URL.
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();
            connection.setDoOutput(true);
            connection.setRequestMethod("PUT");
            OutputStreamWriter out = new OutputStreamWriter(connection.getOutputStream());
            out.write("This text uploaded as an object via presigned URL.");
            out.close();

            // Check the HTTP response code. To complete the upload and make the object
 available,
            // you must interact with the connection object in some way.
            connection.getResponseCode();
            System.out.println("HTTP response code: " + connection.getResponseCode());

            // Check to make sure that the object was uploaded successfully.
            S3Object object = s3Client.getObject(bucketName, objectKey);
            System.out.println("Object " + object.getKey() + " created in bucket " +
 object.getBucketName());
        } 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();
        }
    }
}
```

미리 서명된 URL을 사용하여 객체 업로드(.NET용 AWS SDK)

다음은 .NET용 AWS SDK에서 미리 서명된 URL을 사용하여 S3 버킷에 객체를 업로드하는 방법을 보여주는 C# 예제입니다. 미리 서명된 URL에 대한 자세한 내용은 미리 서명된 URL을 사용하여 객체 업로드 (p. 205) 단원을 참조하십시오.

이 예제는 특정 객체를 위해 미리 서명된 URL을 생성한 후 이를 사용하여 파일을 업로드합니다. 특정 버전의 .NET용 AWS SDK와 이 예제의 호환성 및 실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

```
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.IO;
using System.Net;
```

```
namespace Amazon.DocSamples.S3
{
    class UploadObjectUsingPresignedURLTest
    {
        private const string bucketName = "*** provide bucket name ***";
        private const string objectKey  = "*** provide the name for the uploaded object
 ***";
        private const string filePath   = "*** provide the full path name of the file to
 upload ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            var url = GeneratePreSignedURL();
            UploadObject(url);
        }

        private static void UploadObject(string url)
        {
            HttpWebRequest httpRequest = WebRequest.Create(url) as HttpWebRequest;
            httpRequest.Method = "PUT";
            using (Stream dataStream = httpRequest.GetRequestStream())
            {
                var buffer = new byte[8000];
                using (FileStream fileStream = new FileStream(filePath, FileMode.Open,
 FileAccess.Read))
                {
                    int bytesRead = 0;
                    while ((bytesRead = fileStream.Read(buffer, 0, buffer.Length)) > 0)
                    {
                        dataStream.Write(buffer, 0, bytesRead);
                    }
                }
            }
            HttpWebResponse response = httpRequest.GetResponse() as HttpWebResponse;
        }

        private static string GeneratePreSignedURL()
        {
            var request = new GetPreSignedUrlRequest
            {
                BucketName = bucketName,
                Key        = objectKey,
                Verb       = HttpVerb.PUT,
                Expires    = DateTime.Now.AddMinutes(5)
            };

           string url = s3Client.GetPreSignedURL(request);
           return url;
        }
    }
}
```

추가 정보

.NET용 AWS SDK

미리 서명된 URL을 사용하는 객체 업로드(Ruby용 AWS SDK)

다음 작업은 Ruby용 SDK 버전 3에 대해 미리 서명된 URL을 사용하여 객체를 업로드하는 Ruby 스크립트를 사용하는 방법을 보여줍니다.

객체 업로드 - Ruby용 SDK - 버전 3

1	`Aws::S3::Resource` 클래스의 인스턴스를 만듭니다.
2	`#bucket[]` 클래스 인스턴스의 `#object[]` 및 `Aws::S3::Resource` 메서드를 호출하여 버킷 이름과 객체 키를 제공합니다. `URI` 클래스의 인스턴스를 만들어 미리 서명된 URL을 생성한 후 이를 사용하여 `Aws::S3::Resource` 클래스 인스턴스의 `.presigned_url` 메소드를 파싱합니다. `:put`을 `.presigned_url`에 대한 인수로 지정해야 하며, 객체를 업로드하려는 경우 `PUT`을 `Net::HTTP::Session#send_request`에 지정해야 합니다.
3	미리 서명된 URL을 가진 사용자는 누구나 객체를 업로드할 수 있습니다. 업로드는 객체를 생성하거나 기존 객체를 미리 서명된 URL에서 지정된 동일한 키로 대체합니다.

다음은 위에서 설명한 Ruby용 SDK 버전 3용 작업을 실행하는 Ruby 코드 예제입니다.

Example

```
# Upload an object using a presigned URL for SDK for Ruby - Version 3.

require 'aws-sdk-s3'
require 'net/http'

s3 = Aws::S3::Resource.new(region: 'us-west-2')

# Replace BucketName with the name of your bucket.
# Replace KeyName with the name of the object you are creating or replacing.
obj = s3.bucket('BucketName').object('KeyName')

url = URI.parse(obj.presigned_url(:put))

# The contents of your object, as a string
body = 'Hello World!'

Net::HTTP.start(url.host) do |http|
  http.send_request('PUT', url.request_uri, body,
                     # Or else Net::HTTP adds a default, unsigned content-type
                    'content-type' => '')
end

# Print the contents of your object to the terminal window
puts obj.get.body.read
```

객체 복사

복사 작업은 Amazon S3에 저장되어 있는 객체의 복사본을 만듭니다. 단일 원자성 작업으로 최대 5GB의 객체 복사본을 만들 수 있습니다. 그러나 5GB보다 큰 객체를 복사하려면 멀티파트 업로드 API를 사용해야 합니다. `copy`를 사용하여 다음과 같은 작업을 수행할 수 있습니다.

- 객체의 추가 복사본 생성
- 객체를 복사하여 이름을 변경한 후 원본 삭제
- 객체를 다른 Amazon S3 위치로 이동(예: us-west-1에서 유럽으로 이동)
- 객체 메타데이터 변경

  각 Amazon S3 객체에는 메타데이터가 있으며, 이름-값 페어의 집합으로 표시됩니다. 객체를 업로드할 때 객체 메타데이터를 설정할 수 있습니다. 객체를 업로드한 후에는 객체 메타데이터를 변경할 수 없습니다.

객체 메타데이터를 수정할 수 있는 유일한 방법은 객체 복사본을 만든 후 메타데이터를 설정하는 것입니다. 이때 복사 작업의 원본과 대상을 동일한 객체로 설정합니다.

각 객체에는 메타데이터가 있으며, 시스템 메타데이터와 사용자 정의 메타데이터의 두 종류 메타데이터가 있습니다. 객체에 대해 사용하도록 구성된 스토리지 클래스, 서버 측 암호화와 같은 일부 시스템 메타데이터는 사용자가 제어할 수 있습니다. 객체를 복사할 때 사용자 제어 시스템 메타데이터와 사용자 정의 메타데이터는 복사되지만, 시스템 제어 메타데이터는 Amazon S3에 의해 재설정됩니다. 예를 들어, 객체를 복사할 때 Amazon S3는 복사된 객체의 생성일을 재설정합니다. 복사 요청에서 이러한 값을 설정할 필요가 없습니다.

객체를 복사할 때 일부 메타데이터 값을 업데이트할 수 있습니다. 예를 들어, 원본 객체가 표준 스토리지를 사용하도록 구성된 경우 복사된 객체에 대해서는 RRS(Reduced redundancy storage)를 사용하도록 선택할 수 있습니다. 또한 원본 객체에 설정된 일부 사용자 정의 메타데이터 값을 변경할 수도 있습니다. 복사 과정에서 이처럼 사용자가 구성 가능한 객체 메타데이터(시스템 또는 사용자 정의 메타데이터)를 업데이트하려면 일부 메타데이터 값만 변경하려는 경우에도 요청의 원본 객체에 설정된 사용자가 구성 가능한 모든 메타데이터를 명시적으로 지정해야 합니다.

객체 메타데이터에 대한 자세한 내용은 객체 키와 메타데이터 (p. 102) 단원을 참조하십시오.

Note

- 서로 다른 위치로 객체를 복사할 경우 대역폭 비용이 청구됩니다.
- 원본 객체가 `S3 Glacier` 또는 `S3 Glacier Deep Archive`에 보관된 경우 먼저 임시 사본을 복원한 후 다른 버킷으로 객체를 복사할 수 있습니다. 객체 보관에 대한 자세한 내용은 S3 Glacier 및 S3 Glacier Deep Archive 스토리지 클래스로 전환(객체 보관) (p. 127) 단원을 참조하십시오.

객체를 복사할 때 Amazon S3이 AWS Key Management Service(AWS KMS) 고객 마스터 키(CMK), Amazon S3 관리형 암호화 키 또는 고객 제공 암호화 키를 사용하여 암호화한 대상 객체를 저장하도록 요청할 수 있습니다. 그러려면 요청에 암호화 정보를 지정해야 합니다. 복사 원본이 고객 제공 키를 사용한 서버 측 암호화로 Amazon S3에 저장된 객체인 경우 요청에 암호화 정보를 제공해야 Amazon S3이 객체를 해독하여 복사할 수 있습니다. 자세한 내용은 암호화를 사용하여 데이터 보호 (p. 262) 단원을 참조하십시오.

단일 요청으로 Amazon S3 객체를 하나 이상 복사하기 위해 Amazon S3 배치 작업을 사용할 수 있습니다. 작업할 객체 목록을 Amazon S3 Batch 작업에 제공합니다. Amazon S3 Batch 작업은 각각의 API를 호출하여 지정된 작업을 수행합니다. 단일 Amazon S3 Batch 작업으로 엑사바이트 규모의 데이터가 포함된 수십억 개의 객체에서 지정된 작업을 수행할 수 있습니다.

Amazon S3 Batch 작업은 진행 상황을 추적하고 알림을 보내며 모든 작업에 대한 자세한 완료 보고서를 저장하여 감사 가능한 종합 관리형 서버리스 환경을 제공합니다. AWS Management 콘솔, AWS CLI, AWS SDK 또는 REST API를 통해 Amazon S3 Batch 작업을 사용할 수 있습니다. 자세한 내용은 the section called “기본 사항: 작업” (p. 502) 단원을 참조하십시오.

관련 리소스

- AWS SDK, CLI 및 Explorer 사용 (p. 727)

단일 작업으로 객체 복사

이 단원의 예제에서는 단일 작업으로 최대 5GB의 객체를 복사하는 방법을 보여줍니다. 5GB보다 큰 객체를 복사하려면 멀티파트 업로드 API를 사용해야 합니다. 자세한 내용은 멀티파트 업로드 API를 사용한 객체 복사 (p. 215) 단원을 참조하십시오.

주제

- AWS SDK for Java를 사용하여 객체 복사 (p. 211)
- .NET용 AWS SDK를 사용하여 단일 작업으로 Amazon S3 객체 복사 (p. 211)
- PHP용 AWS SDK를 사용하여 객체 복사 (p. 212)
- Ruby용 AWS SDK를 사용하여 객체 복사 (p. 214)

- REST API를 사용하여 객체 복사 (p. 214)

AWS SDK for Java를 사용하여 객체 복사

Example

다음 예제에서는 AWS SDK for Java를 사용하여 Amazon S3의 객체를 복사하는 방법을 보여 줍니다. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
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;
import com.amazonaws.services.s3.model.CopyObjectRequest;

import java.io.IOException;

public class CopyObjectSingleOperation {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String sourceKey = "*** Source object key *** ";
        String destinationKey = "*** Destination object key ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Copy the object into a new object in the same bucket.
            CopyObjectRequest copyObjRequest = new CopyObjectRequest(bucketName, sourceKey,
 bucketName, destinationKey);
            s3Client.copyObject(copyObjRequest);
        } 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();
        }
    }
}
```

.NET용 AWS SDK를 사용하여 단일 작업으로 Amazon S3 객체 복사

다음 C# 예제에서는 상위 수준 .NET용 AWS SDK를 사용하여 단일 작업으로 최대 5GB의 객체를 복사하는 방법을 보여 줍니다. 5GB보다 큰 객체의 경우 .NET용 AWS SDK 멀티파트 업로드 API를 사용한 Amazon S3 객체 복사 (p. 217)에 설명된 멀티파트 업로드 복사 예제를 참조하십시오.

이 예제에서는 최대 5GB의 객체 복사본을 만듭니다. 특정 버전의 .NET용 AWS SDK와 이 예제의 호환성 및 실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

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

namespace Amazon.DocSamples.S3
{
    class CopyObjectTest
    {
        private const string sourceBucket = "*** provide the name of the bucket with source
 object ***";
        private const string destinationBucket = "*** provide the name of the bucket to
 copy the object to ***";
        private const string objectKey = "*** provide the name of object to copy ***";
        private const string destObjectKey = "*** provide the destination object key 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()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            Console.WriteLine("Copying an object");
            CopyingObjectAsync().Wait();
        }

        private static async Task CopyingObjectAsync()
        {
            try
            {
                CopyObjectRequest request = new CopyObjectRequest
                {
                    SourceBucket = sourceBucket,
                    SourceKey = objectKey,
                    DestinationBucket = destinationBucket,
                    DestinationKey = destObjectKey
                };
                CopyObjectResponse response = await s3Client.CopyObjectAsync(request);
            }
            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);
            }
        }
    }
}
```

추가 정보

.NET용 AWS SDK

PHP용 AWS SDK를 사용하여 객체 복사

이 단원에서는 PHP용 AWS SDK 버전 3의 클래스를 사용하여 한 버킷에서 다른 버킷으로 또는 동일한 버킷으로 Amazon S3 내 단일 객체 및 여러 객체를 복사하는 방법을 보여줍니다.

이 주제에서는 이미 PHP용 AWS SDK 사용 및 PHP 예제 실행 (p. 736)의 지침에 따라 PHP용 AWS SDK가 올바르게 설치되어 있다고 가정합니다.

다음 작업은 PHP SDK 클래스를 사용하여 Amazon S3에 저장되어 있는 객체를 복사하는 방법을 보여줍니다.

다음 작업은 PHP 클래스를 사용하여 Amazon S3 내에서 객체의 여러 복사본을 만드는 방법을 보여줍니다.

객체 복사

1	`Aws\S3\S3Client` 클래스 생성자를 사용하여 Amazon S3 클라이언트의 인스턴스를 만듭니다.
2	객체에 대해 여러 복사본을 만들려면 Aws\CommandInterface 클래스에서 상속되는 Amazon S3 클라이언트의 getCommand() 메서드에 대한 일괄 호출을 수행합니다. 첫 번째 인수로 `CopyObject` 명령을, 두 번째 인수로 원본 버킷, 원본 키 이름, 대상 버킷, 대상 키 이름이 포함된 배열을 제공합니다.

Example Amazon S3 내 객체 복사

다음 PHP 예제는 `copyObject()` 메서드를 사용하여 Amazon S3 내에서 단일 객체를 복사하고, `getcommand()` 메서드를 사용한 `CopyObject` 일괄 호출을 통해 객체의 여러 복사본을 만드는 과정을 보여줍니다.

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

use Aws\S3\S3Client;

$sourceBucket = '*** Your Source Bucket Name ***';
$sourceKeyname = '*** Your Source Object Key ***';
$targetBucket = '*** Your Target Bucket Name ***';

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

// Copy an object.
$s3->copyObject([
    'Bucket'     => $targetBucket,
    'Key'        => "{$sourceKeyname}-copy",
    'CopySource' => "{$sourceBucket}/{$sourceKeyname}",
]);

// Perform a batch of CopyObject operations.
$batch = array();
for ($i = 1; $i <= 3; $i++) {
    $batch[] = $s3->getCommand('CopyObject', [
        'Bucket'     => $targetBucket,
        'Key'        => "{targetKeyname}-{$i}",
        'CopySource' => "{$sourceBucket}/{$sourceKeyname}",
    ]);
}
try {
    $results = CommandPool::batch($s3, $batch);
    foreach($results as $result) {
        if ($result instanceof ResultInterface) {
            // Result handling here
        }
        if ($result instanceof AwsException) {
            // AwsException handling here
        }
```

```
    }
} catch (\Exception $e) {
    // General error handling here
}
```

관련 리소스

- Amazon S3용 PHP용 AWS SDK Aws\S3\S3Client 클래스
- PHP용 AWS SDK 설명서

Ruby용 AWS SDK를 사용하여 객체 복사

다음 작업은 Ruby 클래스를 사용하여 한 버킷에서 다른 버킷으로 또는 동일한 버킷으로 Amazon S3의 객체를 복사하는 방법을 보여줍니다.

객체 복사

1	Ruby용 AWS SDK 버전 3용 Amazon S3 모듈화 gem을 사용하고 'aws-sdk-s3'를 요구하고 AWS 자격 증명을 제공합니다. 자격 증명을 제공하는 방법에 관한 자세한 내용은 AWS 계정 또는 IAM 사용자 자격 증명을 사용하여 요청하기 (p. 18) 단원을 참조하십시오.
2	원본 버킷 이름, 원본 키 이름, 대상 버킷 이름, 대상 키 등의 요청 정보를 제공합니다.

다음 Ruby 코드 예제는 `#copy_object` 메서드를 사용하여 앞서 설명한, 한 버킷에서 다른 버킷으로 객체를 복사하는 작업을 수행하는 방법을 보여줍니다.

Example

```
require 'aws-sdk-s3'

source_bucket_name = '*** Provide bucket name ***'
target_bucket_name = '*** Provide bucket name ***'
source_key = '*** Provide source key ***'
target_key = '*** Provide target key ***'

begin
  s3 = Aws::S3::Client.new(region: 'us-west-2')
  s3.copy_object(bucket: target_bucket_name, copy_source: source_bucket_name + '/' +
 source_key, key: target_key)
rescue StandardError => ex
  puts 'Caught exception copying object ' + source_key + ' from bucket ' +
 source_bucket_name + ' to bucket ' + target_bucket_name + ' as ' + target_key + ':'
  puts ex.message
end

puts 'Copied ' +  source_key + ' from bucket ' + source_bucket_name + ' to bucket ' +
 target_bucket_name + ' as ' + target_key
```

REST API를 사용하여 객체 복사

이 예제에서는 REST를 사용하여 객체를 복사하는 방법을 보여줍니다. REST API에 대한 자세한 내용은 PUT Object (Copy) 단원을 참조하십시오.

이 예제는 메타데이터를 유지하면서 `flotsam` 버킷의 `pacific` 객체를 `jetsam` 버킷의 `atlantic` 객체로 복사합니다.

```
PUT /jetsam HTTP/1.1
```

```
Host: atlantic.s3.amazonaws.com
x-amz-copy-source: /pacific/flotsam
Authorization: AWS AKIAIOSFODNN7EXAMPLE:ENoSbxYByFA0UGLZUqJN5EUnLDg=
Date: Wed, 20 Feb 2008 22:12:21 +0000
```

다음 정보에 따라 서명이 생성됩니다.

```
PUT\r\n
\r\n
\r\n
Wed, 20 Feb 2008 22:12:21 +0000\r\n

x-amz-copy-source:/pacific/flotsam\r\n
/atlantic/jetsam
```

Amazon S3는 객체의 ETag, 최종 수정 날짜와 시간이 포함된 다음 응답을 반환합니다.

```
HTTP/1.1 200 OK
x-amz-id-2: Vyaxt7qEbzv34BnSu5hctyyNSlHTYZFMWK4FtzO+iX8JQNyaLdTshL0KxatbaOZt
x-amz-request-id: 6B13C3C5B34AF333
Date: Wed, 20 Feb 2008 22:13:01 +0000

Content-Type: application/xml
Transfer-Encoding: chunked
Connection: close
Server: AmazonS3
<?xml version="1.0" encoding="UTF-8"?>

<CopyObjectResult>
   <LastModified>2008-02-20T22:13:01</LastModified>
   <ETag>"7e9c608af58950deeb370c98608ed097"</ETag>
</CopyObjectResult>
```

멀티파트 업로드 API를 사용한 객체 복사

이 단원의 예제는 멀티파트 업로드 API를 사용하여 5GB보다 큰 객체를 복사하는 방법을 보여 줍니다. 단일 작업으로는 5GB보다 작은 객체를 복사할 수 있습니다. 자세한 내용은 단일 작업으로 객체 복사 (p. 210) 단원을 참조하십시오.

주제

- AWS SDK for Java 멀티파트 업로드 API를 사용한 객체 복사 (p. 215)
- .NET용 AWS SDK 멀티파트 업로드 API를 사용한 Amazon S3 객체 복사 (p. 217)
- REST 멀티파트 업로드 API를 사용한 객체 복사 (p. 219)

AWS SDK for Java 멀티파트 업로드 API를 사용한 객체 복사

AWS SDK for Java를 사용하여 5GB보다 큰 Amazon S3 객체를 복사하려면 하위 수준 Java API를 사용합니다. 5GB보다 작은 객체의 경우 AWS SDK for Java를 사용하여 객체 복사 (p. 211)에 설명된 단일 작업 복사를 사용하십시오.

하위 수준 Java API를 사용하여 객체를 복사하려면 다음을 수행합니다.

- `AmazonS3Client.initiateMultipartUpload()` 메서드를 실행하여 멀티파트 업로드를 시작합니다.
- `AmazonS3Client.initiateMultipartUpload()` 메서드가 반환하는 응답 객체에서 업로드 ID를 저장합니다. 이후의 각 파트 업로드 작업에서 이 업로드 ID를 제공합니다.

- 모든 파트를 복사합니다. 복사해야 하는 각 파트에 대해 `CopyPartRequest` 클래스의 새 인스턴스를 생성합니다. 소스 및 대상 버킷 이름, 소스 및 대상 객체 키, 업로드 ID, 파트의 첫 번째 및 마지막 바이트 위치와 파트 번호 등과 같은 파트 정보를 제공합니다.
- `AmazonS3Client.copyPart()` 메서드 호출에 대한 응답을 저장합니다. 각 응답에는 업로드된 파트의 파트 번호와 `ETag` 값이 포함되어 있습니다. 멀티파트 업로드를 완료하려면 이러한 정보가 필요합니다.
- `AmazonS3Client.completeMultipartUpload()` 메서드를 호출하여 복사 작업을 완료합니다.

Example

다음 예제에서는 Amazon S3 하위 수준 Java API를 사용하여 멀티파트 복사를 수행하는 방법을 보여줍니다. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
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;
import com.amazonaws.services.s3.model.*;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class LowLevelMultipartCopy {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String sourceBucketName = "*** Source bucket name ***";
        String sourceObjectKey = "*** Source object key ***";
        String destBucketName = "*** Target bucket name ***";
        String destObjectKey = "*** Target object key ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Initiate the multipart upload.
            InitiateMultipartUploadRequest initRequest = new
 InitiateMultipartUploadRequest(destBucketName, destObjectKey);
            InitiateMultipartUploadResult initResult =
 s3Client.initiateMultipartUpload(initRequest);

            // Get the object size to track the end of the copy operation.
            GetObjectMetadataRequest metadataRequest = new
 GetObjectMetadataRequest(sourceBucketName, sourceObjectKey);
            ObjectMetadata metadataResult = s3Client.getObjectMetadata(metadataRequest);
            long objectSize = metadataResult.getContentLength();

            // Copy the object using 5 MB parts.
            long partSize = 5 * 1024 * 1024;
            long bytePosition = 0;
            int partNum = 1;
            List<CopyPartResult> copyResponses = new ArrayList<CopyPartResult>();
            while (bytePosition < objectSize) {
                // The last part might be smaller than partSize, so check to make sure
                // that lastByte isn't beyond the end of the object.
                long lastByte = Math.min(bytePosition + partSize - 1, objectSize - 1);
```

```
                // Copy this part.
                CopyPartRequest copyRequest = new CopyPartRequest()
                        .withSourceBucketName(sourceBucketName)
                        .withSourceKey(sourceObjectKey)
                        .withDestinationBucketName(destBucketName)
                        .withDestinationKey(destObjectKey)
                        .withUploadId(initResult.getUploadId())
                        .withFirstByte(bytePosition)
                        .withLastByte(lastByte)
                        .withPartNumber(partNum++);
                copyResponses.add(s3Client.copyPart(copyRequest));
                bytePosition += partSize;
            }

            // Complete the upload request to concatenate all uploaded parts and make the
 copied object available.
            CompleteMultipartUploadRequest completeRequest = new
 CompleteMultipartUploadRequest(
                    destBucketName,
                    destObjectKey,
                    initResult.getUploadId(),
                    getETags(copyResponses));
            s3Client.completeMultipartUpload(completeRequest);
            System.out.println("Multipart copy complete.");
        } 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();
        }
    }

    // This is a helper function to construct a list of ETags.
    private static List<PartETag> getETags(List<CopyPartResult> responses) {
        List<PartETag> etags = new ArrayList<PartETag>();
        for (CopyPartResult response : responses) {
            etags.add(new PartETag(response.getPartNumber(), response.getETag()));
        }
        return etags;
    }
}
```

.NET용 AWS SDK 멀티파트 업로드 API를 사용한 Amazon S3 객체 복사

다음 C# 예제는 .NET용 AWS SDK를 사용하여 5GB보다 큰 Amazon S3 객체를 하나의 소스 위치에서 다른 위치(예: 한 버킷에서 다른 버킷으로)로 복사합니다. 5GB보다 작은 객체를 복사하려면 .NET용 AWS SDK를 사용하여 단일 작업으로 Amazon S3 객체 복사 (p. 211)에 설명된 단일 작업 복사 프로시저를 사용하십시오. Amazon S3 멀티파트 업로드에 대한 자세한 내용은 멀티파트 업로드 개요 (p. 178) 단원을 참조하십시오.

이 예제에서는 .NET용 AWS SDK 멀티파트 업로드 API를 사용하여 S3 버킷 간에 5GB보다 큰 Amazon S3 객체를 복사하는 방법을 보여줍니다. SDK 호환성 및 실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

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

```
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CopyObjectUsingMPUapiTest
    {
        private const string sourceBucket = "*** provide the name of the bucket with source
 object ***";
        private const string targetBucket = "*** provide the name of the bucket to copy the
 object to ***";
        private const string sourceObjectKey = "*** provide the name of object to copy
 ***";
        private const string targetObjectKey = "*** provide the name of the object copy
 ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            Console.WriteLine("Copying an object");
            MPUCopyObjectAsync().Wait();
        }
        private static async Task MPUCopyObjectAsync()
        {
            // Create a list to store the upload part responses.
            List<UploadPartResponse> uploadResponses = new List<UploadPartResponse>();
            List<CopyPartResponse> copyResponses = new List<CopyPartResponse>();

            // Setup information required to initiate the multipart upload.
            InitiateMultipartUploadRequest initiateRequest =
                new InitiateMultipartUploadRequest
                {
                    BucketName = targetBucket,
                    Key = targetObjectKey
                };

            // Initiate the upload.
            InitiateMultipartUploadResponse initResponse =
                await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // Save the upload ID.
            String uploadId = initResponse.UploadId;

            try
            {
                // Get the size of the object.
                GetObjectMetadataRequest metadataRequest = new GetObjectMetadataRequest
                {
                    BucketName = sourceBucket,
                    Key = sourceObjectKey
                };

                GetObjectMetadataResponse metadataResponse =
                    await s3Client.GetObjectMetadataAsync(metadataRequest);
                long objectSize = metadataResponse.ContentLength; // Length in bytes.

                // Copy the parts.
                long partSize = 5 * (long)Math.Pow(2, 20); // Part size is 5 MB.

                long bytePosition = 0;
                for (int i = 1; bytePosition < objectSize; i++)
                {
                    CopyPartRequest 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 s3Client.CopyPartAsync(copyRequest));

                    bytePosition += partSize;
                }

                // Set up to complete the copy.
                CompleteMultipartUploadRequest completeRequest =
                new CompleteMultipartUploadRequest
                {
                    BucketName = targetBucket,
                    Key = targetObjectKey,
                    UploadId = initResponse.UploadId
                };
                completeRequest.AddPartETags(copyResponses);

                // Complete the copy.
                CompleteMultipartUploadResponse completeUploadResponse =
                    await s3Client.CompleteMultipartUploadAsync(completeRequest);
            }
            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);
            }
        }
    }
}
```

추가 정보

.NET용 AWS SDK

REST 멀티파트 업로드 API를 사용한 객체 복사

Amazon Simple Storage Service API Reference의 단원에 멀티파트 업로드를 위한 REST API가 설명되어 있습니다. 기존 객체를 복사하려면, 파트 업로드(복사) API를 사용하고 요청에 `x-amz-copy-source` 요청 헤더를 추가하여 소스 객체를 지정합니다.

- 멀티파트 업로드 시작
- 파트 업로드
- 파트 업로드(복사)
- 멀티파트 업로드 완료
- 멀티파트 업로드 중단
- 파트 목록 조회

- 멀티파트 업로드 목록 조회

이러한 API를 사용하여 REST 요청을 만들거나, 제공된 SDK 중 하나를 사용할 수 있습니다. SDK에 대한 자세한 내용은 멀티파트 업로드를 위한 API 지원 (p. 181)을 참조하십시오.

객체 키 나열

각 접두사별로 키를 나열할 수 있습니다. 관련 키의 이름에 대해 공통 접두사를 선택하고 계층을 구분하는 특수 문자로 이러한 키를 표시함으로써 나열 작업을 사용하여 계층적 구조로 키를 선택하고 탐색할 수 있습니다. 이는 파일 시스템 내 각 디렉토리에 파일이 저장되는 방식과 유사합니다.

Amazon S3는 버킷에 포함된 키를 열거할 수 있는 나열 작업을 제공합니다. 버킷과 접두사를 기준으로 키를 선택하여 나열할 수 있습니다. 예를 들어, 모든 영어 단어에 대한 키가 포함된 "dictionary"라는 이름의 버킷을 가정해 보겠습니다. 버킷에서 "q"로 시작하는 모든 키를 나열하도록 요청할 수 있습니다. 나열 결과는 항상 UTF-8 이진 순서로 반환됩니다.

SOAP 및 REST 나열 작업은 모두 일치하는 키의 이름과 각 키에 의해 식별되는 객체에 대한 정보가 포함된 XML 문서를 반환합니다.

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

특수 구분 기호로 끝나는 접두사를 공유하는 키의 그룹은 나열의 목적을 위해 공통 접두사별로 분류할 수 있습니다. 이를 통해 애플리케이션은 파일 시스템에서 디렉토리별로 파일을 정렬하는 것과 유사한 방식으로 계층적 구조에 따라 키를 정렬하고 탐색할 수 있습니다. 예를 들어, 영어 외 다른 언어의 단어도 포함하도록 dictionary 버킷을 확장하려면 그 언어와 구분 기호(예: "프랑스어/logical")를 사용하여 각 단어에 접두사를 추가하여 키를 만들 것입니다. 이 명명 스키마와 계층적 나열 기능을 사용하여 프랑스어 단어의 목록만 검색할 수 있습니다. 또한 사전에 등재된 모든 단어에 대한 키를 살펴볼 필요 없이 최상위 수준에서 해당 단어의 목록만 탐색할 수 있습니다.

이 나열 기능에 대한 자세한 내용은 접두사와 구분 기호를 사용하여 계층적 구조로 키 나열 (p. 221) 단원을 참조하십시오.

효율적인 나열 기능 구현

나열 기능은 버킷의 총 키 개수 또는 접두사나 마커, 최대 키 또는 구분 기호 인수의 존재 여부에 의해 그 성능이 크게 좌우되지 않습니다.

여러 페이지의 결과 살펴보기

버킷에 포함할 수 있는 키는 거의 무제한이므로 나열 쿼리의 전체 결과는 매우 방대할 수 있습니다. 이처럼 대량의 결과 조합을 관리하기 위해 Amazon S3 API는 여러 개의 응답으로 나눌 수 있도록 페이지 매김을 지원합니다. 각 나열 키 응답은 최대 1,000개의 키로 구성된 페이지와 함께 응답이 잘린 경우 이를 나타내는 표시 기호를 반환합니다. 모든 키가 수신될 때까지 나열 키 요청을 반복해서 전송해야 합니다. AWS SDK 래퍼 라이브러리는 동일한 페이지 매김을 제공합니다.

다음 Java 및 .NET SDK 예제는 버킷에서 키를 나열할 때 페이지 매김을 사용하는 방법을 보여줍니다.

- AWS SDK for Java를 사용한 키 나열 (p. 222)
- .NET용 AWS SDK를 사용한 키 나열 (p. 223)
- PHP용 AWS SDK를 사용한 키 나열 (p. 224)

관련 리소스

- AWS SDK, CLI 및 Explorer 사용 (p. 727)

접두사와 구분 기호를 사용하여 계층적 구조로 키 나열

접두사와 구분 기호 파라미터는 나열 작업에서 반환되는 결과의 종류를 제한합니다. 접두사는 이처럼 지정된 접두사로 시작하는 키로 결과를 제한합니다. 구분 기호를 사용하면 목록 작업이 공통 접두사를 공유하는 모든 키를 단일 요약 목록 결과로 롤업합니다.

접두사와 구분 기호 파라미터의 목적은 키를 계층적 구조로 정렬하여 탐색할 수 있도록 하는 것입니다. 그러려면 먼저 슬래시(/)와 같이 버킷에서 키 이름에는 사용되지 않을 구분 기호를 선택합니다. 그런 다음 각 계층을 구분 기호로 분리하여 각 수준별로 모든 계층이 포함된 키 이름을 구성합니다.

예를 들어, 도시에 대한 정보를 저장할 경우 대륙, 국가 및 주를 기준으로 도시를 구성할 것입니다. 도시 이름에는 보통 문장 부호가 사용되지 않으므로 슬래시(/)를 구분 기호로 선택할 수 있습니다. 다음 예제는 슬래시(/) 구분 기호의 사용 방법을 보여줍니다.

- 유럽/프랑스/누벨아키텐/보르도
- 북미/캐나다/퀘벡/몬트리올
- 북미/미국/워싱턴/벨뷰
- 북미/미국/워싱턴/시애틀

이런 식으로 전세계의 모든 도시를 저장한 경우 계층 구분 없이 각 키 네임스페이스를 일괄 관리하기에는 많은 어려움이 있을 것입니다. 나열 작업에 `Prefix` 및 `Delimiter`를 사용하면 앞서 만든 계층 구조를 사용하여 데이터를 나열할 수 있습니다. 예를 들어, 미국의 모든 도시를 나열하기 위해 `Delimiter='/'`, `Prefix='북미/미국/'`으로 설정합니다. 데이터가 저장된 캐나다의 모든 지방을 나열하려면 `Delimiter='/'`, `Prefix='북미/캐나다/'`로 설정합니다.

구분 기호를 사용한 나열 요청은 각 수준별로 계층 구조를 탐색할 수 있어 하위 수준의 수백만 키를 건너뛸 수 있습니다. 예를 들어, 다음과 같은 키의 `ExampleBucket` 버킷이 있다고 가정해 보겠습니다.

```
sample.jpg

photos/2006/January/sample.jpg

photos/2006/February/sample2.jpg

photos/2006/February/sample3.jpg

photos/2006/February/sample4.jpg
```

이 예제 버킷에서 루트 수준에는 `sample.jpg` 객체만 있습니다. 버킷의 루트 수준 객체만 나열하려면 "/" 구분 기호 문자만 사용하여 버킷에 대해 GET 요청을 보냅니다. 그 응답으로 Amazon S3는 "/" 구분 기호 문자가 포함되지 않은 `sample.jpg` 객체 키를 반환합니다. 그 외 모든 키에는 구분 기호 문자가 포함되어 있습니다. Amazon S3는 이러한 키를 그룹화하여 `photos/` 접두사 값을 가진 단일 `CommonPrefixes` 요소를 반환합니다. 이는 해당 키의 시작 부분부터 지정된 구분 기호의 첫 번째 발생 지점까지의 범위에서 가져온 하위 문자열입니다.

Example

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>ExampleBucket</Name>
  <Prefix></Prefix>
  <Marker></Marker>
  <MaxKeys>1000</MaxKeys>
  <Delimiter>/</Delimiter>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>sample.jpg</Key>
    <LastModified>2011-07-24T19:39:30.000Z</LastModified>
    <ETag>&quot;d1a7fb5eab1c16cb4f7cf341cf188c3d&quot;</ETag>
    <Size>6</Size>
    <Owner>
```

```
      <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
      <DisplayName>displayname</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  <CommonPrefixes>
    <Prefix>photos/</Prefix>
  </CommonPrefixes>
</ListBucketResult>
```

AWS SDK for Java를 사용한 키 나열

Example

다음 예제에서는 버킷의 객체 키를 나열합니다. 이 예제는 페이지 매김을 사용하여 객체 키 세트를 검색합니다. 첫 페이지 이후 반환할 키가 더 있는 경우, Amazon S3는 응답에 연속 토큰을 포함시킵니다. 이 예제에서는 후속 요청에서 연속 토큰을 사용하여 다음 객체 키 세트를 가져옵니다.

실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
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;
import com.amazonaws.services.s3.model.ListObjectsV2Request;
import com.amazonaws.services.s3.model.ListObjectsV2Result;
import com.amazonaws.services.s3.model.S3ObjectSummary;

import java.io.IOException;

public class ListKeys {

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

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            System.out.println("Listing objects");

            // maxKeys is set to 2 to demonstrate the use of
            // ListObjectsV2Result.getNextContinuationToken()
            ListObjectsV2Request req = new
 ListObjectsV2Request().withBucketName(bucketName).withMaxKeys(2);
            ListObjectsV2Result result;

            do {
                result = s3Client.listObjectsV2(req);

                for (S3ObjectSummary objectSummary : result.getObjectSummaries()) {
                    System.out.printf(" - %s (size: %d)\n", objectSummary.getKey(),
 objectSummary.getSize());
                }
                // If there are more than maxKeys keys in the bucket, get a continuation
 token
```

```
                // and list the next objects.
                String token = result.getNextContinuationToken();
                System.out.println("Next Continuation Token: " + token);
                req.setContinuationToken(token);
            } while (result.isTruncated());
        } 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();
        }
    }
}
```

.NET용 AWS SDK를 사용한 키 나열

Example

다음 C# 예제에서는 버킷의 객체 키를 나열합니다. 이 예제에서는 페이지 매김을 사용하여 객체 키 세트를 검색합니다. 반환할 키가 더 있는 경우, Amazon S3는 응답에 연속 토큰을 포함시킵니다. 이 코드에서는 후속 요청에서 연속 토큰을 사용하여 다음 객체 키 세트를 가져옵니다.

실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

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

namespace Amazon.DocSamples.S3
{
    class ListObjectsTest
    {
        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()
        {
            client = new AmazonS3Client(bucketRegion);
            ListingObjectsAsync().Wait();
        }

        static async Task ListingObjectsAsync()
        {
            try
            {
                ListObjectsV2Request 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);
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine("S3 error occurred. Exception: " +
 amazonS3Exception.ToString());
                Console.ReadKey();
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: " + e.ToString());
                Console.ReadKey();
            }
        }
    }
}
```

PHP용 AWS SDK를 사용한 키 나열

이 단원에서는 PHP용 AWS SDK 버전 3의 클래스를 사용하여 Amazon S3 버킷에 포함된 객체 키를 나열하는 방법을 설명합니다.

이 주제에서는 이미 PHP용 AWS SDK 사용 및 PHP 예제 실행 (p. 736)의 지침에 따라 PHP용 AWS SDK가 올바르게 설치되어 있다고 가정합니다.

PHP용 AWS SDK를 사용하여 버킷에 포함된 객체 키를 나열하려면 먼저 버킷에 포함된 객체를 나열한 후 나열된 각 객체에서 키를 추출해야 합니다. 버킷의 객체를 나열할 때 하위 수준의 Aws\S3\S3Client::listObjects() 메서드 또는 상위 수준의 Aws\ResultPaginator 클래스를 사용할 수 있습니다.

하위 수준의 `listObjects()` 메서드는 기본 Amazon S3 REST API로 매핑됩니다. 각 `listObjects()` 메서드는 최대 1,000개의 객체가 포함된 페이지를 반환합니다. 버킷의 객체가 1,000개보다 많을 경우 응답은 잘려서 표시되며, 다음 1,000개의 객체 조합을 검색하려면 다시 `listObjects()` 요청을 전송해야 합니다.

상위 수준의 `ListObjects` 페이지네이터를 사용하면 좀 더 쉽게 버킷에 포함된 객체를 나열하는 작업을 수행할 수 있습니다. `ListObjects` 페이지네이터를 사용하여 객체를 나열하려면 `ListObjects` 명령을 첫 번째 인수로, 지정된 버킷에서 반환되는 객체가 포함될 배열을 두 번째 인수로 지정하여 Aws/AwsClientInterface 클래스에서 상속된 Amazon S3 클라이언트의 getPaginator() 메서드를 실행합니다. `ListObjects` 페이지네이터를 사용할 때 `getPaginator()` 메서드는 지정된 버킷에 포함된 모든 객체를 반환합니다. 이 경우에는 1,000개의 객체 한도가 없으므로 응답이 잘릴 것에 대해 염려하지 않아도 됩니다.

다음 작업은 PHP Amazon S3 클라이언트 메서드를 사용하여 버킷에 포함된 객체를 나열한 후 객체 키를 나열하는 방법을 보여줍니다.

Example 객체 키 나열

다음 PHP 예제는 지정된 버킷에서 키를 나열하는 방법을 보여줍니다. 버킷의 객체를 나열한 후 목록의 각 객체에서 키를 추출하는 상위 수준의 `getIterator()` 메서드 사용 방법과 함께 버킷의 객체를 나열한 후 반환된 목록의 각 객체에서 키를 추출하는 하위 수준의 `listObjects()` 메서드 사용 방법도 보여줍니다. 이 가이드의 PHP 예제 실행에 대한 자세한 내용은 PHP 예제 실행 (p. 737) 단원을 참조하십시오.

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

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

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

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

// Use the high-level iterators (returns ALL of your objects).
try {
    $results = $s3->getPaginator('ListObjects', [
        'Bucket' => $bucket
    ]);

    foreach ($results as $result) {
        foreach ($result['Contents'] as $object) {
            echo $object['Key'] . PHP_EOL;
        }
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}

// Use the plain API (returns ONLY up to 1000 of your objects).
try {
    $objects = $s3->listObjects([
        'Bucket' => $bucket
    ]);
    foreach ($objects['Contents']  as $object) {
        echo $object['Key'] . PHP_EOL;
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

관련 리소스

- Amazon S3용 PHP용 AWS SDK Aws\S3\S3Client 클래스
- 페이지네이터
- PHP용 AWS SDK 설명서

REST API를 사용한 키 나열

AWS SDK를 사용하여 버킷의 객체 키를 나열할 수 있습니다. 하지만 애플리케이션에서 요구할 경우 REST 요청을 직접 전송할 수 있습니다. GET 요청을 전송하여 버킷의 일부 또는 전체 객체를 반환하거나, 선택 조건을 사용하여 버킷의 일부 객체를 반환할 수 있습니다. 자세한 내용은 GET Bucket (List Objects) Version 2 단원을 참조하십시오.

객체 삭제

주제

- 버전 관리를 사용하는 버킷의 객체 삭제 (p. 226)
- MFA를 사용하는 버킷의 객체 삭제 (p. 226)

- 관련 리소스 (p. 226)
- 요청당 단일 객체 삭제 (p. 227)
- 요청당 여러 객체 삭제 (p. 232)

Amazon S3에서 바로 하나 이상의 객체를 삭제할 수 있으며, 다음과 같은 객체 삭제 옵션을 선택할 수 있습니다.

- 단일 객체 삭제–Amazon S3는 단일 HTTP 요청으로 하나의 객체를 삭제할 수 있는 DELETE API를 제공합니다.
- 여러 객체 삭제–Amazon S3는 또한 단일 HTTP 요청으로 최대 1,000개의 객체를 삭제할 수 있는 Multi-Object Delete API를 제공합니다.

버전 관리를 사용하지 않는 버킷에서 객체를 삭제할 때는 객체 키 이름만 제공하면 되지만, 버전 관리를 사용하는 버킷에서 객체를 삭제할 때는 객체의 버전 ID를 제공하여 객체의 특정 버전만 삭제할 수 있습니다.

버전 관리를 사용하는 버킷의 객체 삭제

버전 관리를 사용하는 버킷에서는 동일한 객체에 대해 여러 버전이 존재할 수 있습니다. 버전 관리를 사용하는 버킷에서 작업할 때 삭제 API는 다음과 같은 옵션을 제공합니다.

- 버전이 지정되지 않은 삭제 요청–즉, 객체의 키만 지정하며 버전 ID는 지정하지 않습니다. 이 경우 Amazon S3는 삭제 마커를 만들고 응답으로 그 버전 ID를 반환합니다. 그 결과 객체가 버킷에서 보이지 않습니다. 객체 버전 관리 및 삭제 마커 개념에 대한 자세한 내용은 객체 버전 관리 (p. 112) 단원을 참조하십시오.
- 버전이 지정된 삭제 요청–즉, 객체의 키와 버전 ID를 모두 지정합니다. 이 경우 다음 두 가지 결과가 가능합니다.
  - 버전 ID가 특정 객체 버전과 일치할 경우 Amazon S3는 해당 특정 버전의 객체를 삭제합니다.
  - 버전 ID가 해당 객체의 삭제 마커와 일치할 경우 Amazon S3는 삭제 마커를 삭제합니다. 그 결과 해당 객체가 버킷에서 다시 표시됩니다.

MFA를 사용하는 버킷의 객체 삭제

멀티 팩터 인증(MFA)을 사용하는 버킷에서 객체를 삭제할 경우 다음과 같은 사항에 주의합니다.

- 무효한 MFA 토큰을 제공할 경우 요청에 실패합니다.
- MFA가 활성화된 버킷에서 버전을 지정한 삭제를 요청(객체 키 및 버전 ID 제공)할 경우 MFA 토큰이 올바르지 않으면 요청에 실패합니다. 또한 MFA가 활성화된 버킷에서 Multi-Object Delete API를 사용할 때 버전이 지정된 삭제 요청(객체 키 및 버전 ID 제공) 시 MFA 토큰을 제공하지 않으면 요청이 모두 실패로 끝납니다.

다음은 성공적인 요청 사례입니다.

- MFA가 활성화된 버킷에서 버전이 지정되지 않은 삭제를 요청하고(버전이 지정된 객체를 삭제하지 않음) MFA 토큰을 제공하지 않으면 삭제가 성공합니다.
- Multi-Object Delete 요청이 MFA가 활성화된 버킷에서 버전이 지정되지 않은 객체들만 삭제하도록 지정하고 MFA 토큰을 제공하지 않으면 삭제가 성공합니다.

MFA 삭제에 대한 자세한 내용은 MFA Delete (p. 470) 단원을 참조하십시오.

관련 리소스

- AWS SDK, CLI 및 Explorer 사용 (p. 727)

요청당 단일 객체 삭제

주제

- AWS SDK for Java를 사용한 객체 삭제 (p. 227)
- .NET용 AWS SDK를 사용한 객체 삭제 (p. 229)
- PHP용 AWS SDK를 사용한 객체 삭제 (p. 231)
- REST API를 사용한 객체 삭제 (p. 232)

- Deleting an Object Using the AWS Command Line Interface를 사용한 객체 삭제

요청당 객체 하나를 삭제하려면 `DELETE` API를 사용합니다(DELETE 객체 참조). 객체 삭제에 대한 자세한 내용은 객체 삭제 (p. 225) 단원을 참조하십시오.

REST API를 직접 사용할 수도 있고, 애플리케이션 개발을 간소화해주는 AWS SDK가 제공하는 래퍼 라이브러리를 사용할 수 있습니다.

AWS SDK for Java를 사용한 객체 삭제

버킷에서 객체를 삭제할 수 있습니다. 버킷에서 S3 버전 관리를 사용하는 경우 다음 옵션을 사용할 수 있습니다.

- 버전 ID를 지정하여 특정 객체 버전을 삭제합니다.
- 버전 ID를 지정하지 않고 객체를 삭제합니다. 이 경우 S3에서는 객체에 삭제 마커를 추가합니다.

For more information about S3 버전 관리, see 객체 버전 관리 (p. 112).

Example 예제 1: 객체 삭제(버전이 지정되지 않은 버킷)

다음 예제는 버킷에서 객체를 삭제합니다. 이 예제에서는 버킷에서 버전 관리를 사용하지 않고 객체에 버전 ID가 없다고 가정합니다. 삭제 요청 시, 객체 키만 지정하며 버전 ID는 지정하지 않습니다. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
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;
import com.amazonaws.services.s3.model.DeleteObjectRequest;

import java.io.IOException;

public class DeleteObjectNonVersionedBucket {

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

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            s3Client.deleteObject(new DeleteObjectRequest(bucketName, keyName));
```

```
        } 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();
        }
    }
}
```

Example 예제 2: 객체 삭제(버전이 지정된 버킷)

다음 예제는 버전이 지정된 버킷에서 객체를 삭제합니다. 이 예제에서는 객체 키 이름 및 버전 ID를 지정하여 특정 객체 버전을 삭제합니다. 이 예제는 다음을 수행합니다.

1. 버킷에 샘플 객체를 추가합니다. 응답으로 Amazon S3가 새로 추가된 객체의 버전 ID를 반환합니다. 이 예제에서는 삭제 요청에 이 버전 ID를 사용합니다.
2. 객체 키 이름 및 버전 ID를 지정하여 객체 버전을 삭제합니다. 해당 객체의 다른 버전이 없는 경우 Amazon S3는 해당 객체를 완전히 삭제합니다. 그렇지 않은 경우 는 지정된 버전만 삭제합니다.

   Note

   `ListVersions` 요청을 보내 객체의 버전 ID를 가져올 수 있습니다.

```
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;
import com.amazonaws.services.s3.model.BucketVersioningConfiguration;
import com.amazonaws.services.s3.model.DeleteVersionRequest;
import com.amazonaws.services.s3.model.PutObjectResult;

import java.io.IOException;

public class DeleteObjectVersionEnabledBucket {

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

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Check to ensure that the bucket is versioning-enabled.
            String bucketVersionStatus =
 s3Client.getBucketVersioningConfiguration(bucketName).getStatus();
            if (!bucketVersionStatus.equals(BucketVersioningConfiguration.ENABLED)) {
                System.out.printf("Bucket %s is not versioning-enabled.", bucketName);
            } else {
                // Add an object.
                PutObjectResult putResult = s3Client.putObject(bucketName, keyName, "Sample
 content for deletion example.");
```

```
                System.out.printf("Object %s added to bucket %s\n", keyName, bucketName);

                // Delete the version of the object that we just created.
                System.out.println("Deleting versioned object " + keyName);
                s3Client.deleteVersion(new DeleteVersionRequest(bucketName, keyName,
 putResult.getVersionId()));
                System.out.printf("Object %s, version %s deleted\n", keyName,
 putResult.getVersionId());
            }
        } 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();
        }
    }
}
```

.NET용 AWS SDK를 사용한 객체 삭제

버전이 지정되지 않은 버킷에서 객체를 삭제하는 경우 해당 객체가 제거됩니다. 버킷에서 S3 버전 관리를 사용하는 경우 다음 옵션을 사용할 수 있습니다.

- 버전 ID를 지정하여 객체의 특정 버전을 삭제합니다.
- 버전 ID를 지정하지 않고 객체를 삭제합니다. Amazon S3가 삭제 마커를 추가합니다. 삭제 마커에 대한 자세한 내용은 객체 버전 관리 (p. 112) 단원을 참조하십시오.

다음 예제는 버전이 지정된 버킷과 버전이 지정되지 않는 버킷 둘 다에서 객체를 삭제하는 방법을 보여줍니다. For more information about S3 버전 관리, see 객체 버전 관리 (p. 112).

Example 버전이 지정되지 않은 버킷에서 단일 객체 삭제

다음 C# 예제는 버전이 지정되지 않은 버킷에서 객체를 삭제합니다. 이 예제에서는 해당 객체에 버전 ID가 없다고 가정합니다. 따라서 버전 ID를 지정하지 않습니다. 객체 키만 지정합니다. 실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

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

namespace Amazon.DocSamples.S3
{
    class DeleteObjectNonVersionedBucketTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** object key ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            DeleteObjectNonVersionedBucketAsync().Wait();
        }
        private static async Task DeleteObjectNonVersionedBucketAsync()
```

```
        {
            try
            {
                var deleteObjectRequest = new DeleteObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName
                };

                Console.WriteLine("Deleting an object");
                await client.DeleteObjectAsync(deleteObjectRequest);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when deleting
 an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 deleting an object", e.Message);
            }
        }
    }
}
```

Example 버전이 지정된 버킷에서 단일 객체 삭제

다음 C# 예제는 버전이 지정된 버킷에서 객체를 삭제합니다. 여기에서는 객체 키 이름 및 버전 ID를 지정하여 객체의 특정 버전을 삭제합니다.

이 코드에서는 다음 작업을 수행합니다.

1. 지정한 버킷에 대해 S3 버전 관리를 활성화합니다(버전 관리가 이미 활성화된 경우 적용되지 않음).
2. 버킷에 샘플 객체를 추가합니다. 응답으로 Amazon S3가 새로 추가된 객체의 버전 ID를 반환합니다. 이 예제에서는 삭제 요청에 이 버전 ID를 사용합니다.
3. 객체 키 이름 및 버전 ID를 지정하여 샘플 객체를 삭제합니다.

   Note

   `ListVersions` 요청을 보내 객체의 버전 ID를 가져올 수도 있습니다.

   ```
   var listResponse = client.ListVersions(new ListVersionsRequest { BucketName =
    bucketName, Prefix = keyName });
   ```

실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

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

namespace Amazon.DocSamples.S3
{
    class DeleteObjectVersion
    {
        private const string bucketName = "*** versioning-enabled bucket name ***";
        private const string keyName = "*** Object Key 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()
        {
            client = new AmazonS3Client(bucketRegion);
            CreateAndDeleteObjectVersionAsync().Wait();
        }

        private static async Task CreateAndDeleteObjectVersionAsync()
        {
            try
            {
                // Add a sample object.
                string versionID = await PutAnObject(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 e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when deleting
 an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 deleting an object", e.Message);
            }
        }

        static async Task<string> PutAnObject(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;
        }
    }
}
```

PHP용 AWS SDK를 사용한 객체 삭제

이 주제에서는 PHP용 AWS SDK 버전 3의 클래스를 사용하여 버전이 지정되지 않은 버킷에서 단일 객체를 삭제하는 방법을 보여줍니다. 버전이 지정된 버킷에서 객체를 삭제하는 방법에 대한 자세한 내용은 REST API를 사용한 객체 삭제 (p. 232) 단원을 참조하십시오.

이 주제에서는 이미 PHP용 AWS SDK 사용 및 PHP 예제 실행 (p. 736)의 지침에 따라 PHP용 AWS SDK가 올바르게 설치되어 있다고 가정합니다.

다음 PHP 예제는 버킷에서 객체를 삭제합니다. 이 예제에서는 버전이 지정되지 않은 버킷에서 객체를 삭제하는 방법을 보여주기 때문에 삭제 요청 시 버킷 이름과 객체 키(버전 ID 아님)만 제공합니다. 이 가이드의 PHP 예제 실행에 대한 자세한 내용은 PHP 예제 실행 (p. 737) 단원을 참조하십시오.

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

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

// Delete an object from the bucket.
$s3->deleteObject([
    'Bucket' => $bucket,
    'Key'    => $keyname
]);
```

관련 리소스

- Amazon S3용 PHP용 AWS SDK Aws\S3\S3Client 클래스
- PHP용 AWS SDK 설명서

REST API를 사용한 객체 삭제

AWS SDK를 사용하여 객체를 삭제할 수 있습니다. 하지만 애플리케이션에서 요구할 경우 REST 요청을 직접 전송할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API Reference의 DELETE Object 단원을 참조하십시오.

요청당 여러 객체 삭제

주제

- AWS SDK for Java를 사용한 여러 객체 삭제 (p. 232)
- .NET용 AWS SDK를 사용한 여러 객체 삭제 (p. 236)
- PHP용 AWS SDK를 사용한 여러 객체 삭제 (p. 241)
- REST API를 사용한 여러 객체 삭제 (p. 243)

Amazon S3는 단일 요청으로 여러 객체를 삭제할 수 있는 Multi-Object Delete API를 제공합니다(Delete - Multi-Object Delete 참조). API는 상세 정보 표시 모드 및 자동 모드의 두 가지 응답을 지원합니다. 상세 정보 표시 모드는 기본적으로 꺼져 있습니다. 상세 정보 표시 모드에서는 요청에 지정된 각 키의 삭제 결과가 응답에 포함됩니다. 자동 모드의 응답에는 삭제 작업에 오류가 발생한 키만 포함됩니다. 자동 모드를 사용하는 경우 모든 키가 성공적으로 삭제되면 Amazon S3가 빈 응답을 반환합니다.

객체 삭제에 대한 자세한 내용은 객체 삭제 (p. 225) 단원을 참조하십시오.

REST API를 직접 사용하거나 AWS SDK를 사용할 수 있습니다.

AWS SDK for Java를 사용한 여러 객체 삭제

AWS SDK for Java는 여러 객체를 삭제하기 위한 `AmazonS3Client.deleteObjects()` 메서드를 제공합니다. 삭제하려는 각 객체에 대해 키 이름을 지정합니다. 버전 관리를 사용하는 버킷의 경우, 다음 옵션을 사용할 수 있습니다.

- 객체의 키 이름만 지정합니다. Amazon S3가 객체에 삭제 마커를 추가합니다.

- 삭제하려는 객체의 키 이름 및 버전 ID를 모두 지정합니다. Amazon S3가 해당 객체의 지정된 버전을 삭제합니다.

Example

다음 코드 예제는 Multi-Object Delete API를 사용하여 버전 관리 미사용 버킷에서 객체를 삭제합니다. 이 예제에서는 샘플 객체를 버킷에 업로드한 다음 `AmazonS3Client.deleteObjects()` 메서드를 사용하여 단일 요청으로 객체를 삭제합니다. 이 예제에서는 객체에 버전 ID가 없으므로 `DeleteObjectsRequest`에서 객체 키 이름만 지정합니다.

객체 삭제에 대한 자세한 내용은 객체 삭제 (p. 225) 단원을 참조하십시오. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
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;
import com.amazonaws.services.s3.model.DeleteObjectRequest;

import java.io.IOException;

public class DeleteObjectNonVersionedBucket {

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

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            s3Client.deleteObject(new DeleteObjectRequest(bucketName, keyName));
        } 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();
        }
    }
}
```

Example

다음 코드 예제는 Multi-Object Delete API를 사용하여 버전 관리를 사용하는 버킷에서 객체를 삭제합니다. 이 예제에서는 작업을 수행합니다.

1. 샘플 객체를 만든 후 삭제할 각 객체의 키 이름과 버전 ID를 지정하여 삭제합니다. 이 작업에서는 지정된 객체 버전만 삭제합니다.
2. 샘플 객체를 만든 후 키 이름만 지정하여 삭제합니다. 이 예제에서는 버전 ID를 지정하지 않으므로 이 작업은 특정 객체 버전을 삭제하지 않고 각 객체에 삭제 마커를 추가합니다. 삭제 마커가 추가된 객체는 AWS Management 콘솔에 표시되지 않습니다.

3. 삭제 마커의 객체 키와 버전 ID를 지정하여 삭제 마커를 제거합니다. 이 작업은 삭제 마커를 삭제합니다. 그러면 해당 객체가 AWS Management 콘솔에 다시 표시됩니다.

```
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;
import com.amazonaws.services.s3.model.BucketVersioningConfiguration;
import com.amazonaws.services.s3.model.DeleteObjectsRequest;
import com.amazonaws.services.s3.model.DeleteObjectsRequest.KeyVersion;
import com.amazonaws.services.s3.model.DeleteObjectsResult;
import com.amazonaws.services.s3.model.DeleteObjectsResult.DeletedObject;
import com.amazonaws.services.s3.model.PutObjectResult;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class DeleteMultipleObjectsVersionEnabledBucket {
    private static AmazonS3 S3_CLIENT;
    private static String VERSIONED_BUCKET_NAME;

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

        try {
            S3_CLIENT = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Check to make sure that the bucket is versioning-enabled.
            String bucketVersionStatus =
 S3_CLIENT.getBucketVersioningConfiguration(VERSIONED_BUCKET_NAME).getStatus();
            if (!bucketVersionStatus.equals(BucketVersioningConfiguration.ENABLED)) {
                System.out.printf("Bucket %s is not versioning-enabled.",
 VERSIONED_BUCKET_NAME);
            } else {
                // Upload and delete sample objects, using specific object versions.
                uploadAndDeleteObjectsWithVersions();

                // Upload and delete sample objects without specifying version IDs.
                // Amazon S3 creates a delete marker for each object rather than deleting
                // specific versions.
                DeleteObjectsResult unversionedDeleteResult =
 uploadAndDeleteObjectsWithoutVersions();

                // Remove the delete markers placed on objects in the non-versioned create/
delete method.
                multiObjectVersionedDeleteRemoveDeleteMarkers(unversionedDeleteResult);
            }
        } 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();
        }
```

```
    }

    private static void uploadAndDeleteObjectsWithVersions() {
        System.out.println("Uploading and deleting objects with versions specified.");

        // Upload three sample objects.
        ArrayList<KeyVersion> keys = new ArrayList<KeyVersion>();
        for (int i = 0; i < 3; i++) {
            String keyName = "delete object without version ID example " + i;
            PutObjectResult putResult = S3_CLIENT.putObject(VERSIONED_BUCKET_NAME, keyName,
                    "Object number " + i + " to be deleted.");
            // Gather the new object keys with version IDs.
            keys.add(new KeyVersion(keyName, putResult.getVersionId()));
        }

        // Delete the specified versions of the sample objects.
        DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest(VERSIONED_BUCKET_NAME)
                .withKeys(keys)
                .withQuiet(false);

        // Verify that the object versions were successfully deleted.
        DeleteObjectsResult delObjRes = S3_CLIENT.deleteObjects(multiObjectDeleteRequest);
        int successfulDeletes = delObjRes.getDeletedObjects().size();
        System.out.println(successfulDeletes + " objects successfully deleted");
    }

    private static DeleteObjectsResult uploadAndDeleteObjectsWithoutVersions() {
        System.out.println("Uploading and deleting objects with no versions specified.");

        // Upload three sample objects.
        ArrayList<KeyVersion> keys = new ArrayList<KeyVersion>();
        for (int i = 0; i < 3; i++) {
            String keyName = "delete object with version ID example " + i;
            S3_CLIENT.putObject(VERSIONED_BUCKET_NAME, keyName, "Object number " + i + " to
be deleted.");
            // Gather the new object keys without version IDs.
            keys.add(new KeyVersion(keyName));
        }

        // Delete the sample objects without specifying versions.
        DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest(VERSIONED_BUCKET_NAME).withKeys(keys)
                .withQuiet(false);

        // Verify that delete markers were successfully added to the objects.
        DeleteObjectsResult delObjRes = S3_CLIENT.deleteObjects(multiObjectDeleteRequest);
        int successfulDeletes = delObjRes.getDeletedObjects().size();
        System.out.println(successfulDeletes + " objects successfully marked for deletion
without versions.");
        return delObjRes;
    }

    private static void multiObjectVersionedDeleteRemoveDeleteMarkers(DeleteObjectsResult
response) {
        List<KeyVersion> keyList = new ArrayList<KeyVersion>();
        for (DeletedObject deletedObject : response.getDeletedObjects()) {
            // Note that the specified version ID is the version ID for the delete marker.
            keyList.add(new KeyVersion(deletedObject.getKey(),
deletedObject.getDeleteMarkerVersionId()));
        }
        // Create a request to delete the delete markers.
        DeleteObjectsRequest deleteRequest = new
DeleteObjectsRequest(VERSIONED_BUCKET_NAME).withKeys(keyList);

        // Delete the delete markers, leaving the objects intact in the bucket.
```

```
        DeleteObjectsResult delObjRes = S3_CLIENT.deleteObjects(deleteRequest);
        int successfulDeletes = delObjRes.getDeletedObjects().size();
        System.out.println(successfulDeletes + " delete markers successfully deleted");
    }
}
```

.NET용 AWS SDK를 사용한 여러 객체 삭제

.NET용 AWS SDK는 여러 객체를 삭제하기 위한 편리한 메서드인 `DeleteObjects`를 제공합니다. 삭제하려는 각 객체에 대해 객체의 키 이름과 버전을 지정합니다. 버전 관리를 사용하지 않는 버킷의 경우, 버전 ID에 `null`을 지정합니다. 예외가 발생하면 `DeleteObjectsException` 응답을 검토하여 삭제되지 않은 객체와 이유를 확인합니다.

Example 버전 관리 미사용 버킷에서 여러 객체 삭제

다음 C# 예제는 multi-object delete API를 사용하여 버전 관리 미사용 버킷에서 객체를 삭제합니다. 이 예제에서는 샘플 객체를 버킷에 업로드한 다음 `DeleteObjects` 메서드를 사용하여 단일 요청으로 객체를 삭제합니다. 이 예제에서는 버전 ID가 null이므로 `DeleteObjectsRequest`에서 객체 키 이름만 지정합니다.

실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

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

namespace Amazon.DocSamples.S3
{
    class DeleteMultipleObjectsNonVersionedBucketTest
    {
        private const string bucketName = "*** versioning-enabled 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()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            MultiObjectDeleteAsync().Wait();
        }

        static async Task MultiObjectDeleteAsync()
        {
            // Create sample objects (for subsequent deletion).
            var keysAndVersions = await PutObjectsAsync(3);

            // a. multi-object delete by specifying the key names and version IDs.
            DeleteObjectsRequest multiObjectDeleteRequest = new DeleteObjectsRequest
            {
                BucketName = bucketName,
                Objects = keysAndVersions // This includes the object keys and null version
 IDs.
            };
            // You can add specific object key to the delete request using the .AddKey.
            // multiObjectDeleteRequest.AddKey("TickerReference.csv", null);
            try
            {
                DeleteObjectsResponse response = await
 s3Client.DeleteObjectsAsync(multiObjectDeleteRequest);
```

```
                Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
            }
            catch (DeleteObjectsException e)
            {
                PrintDeletionErrorStatus(e);
            }
        }

        private static void PrintDeletionErrorStatus(DeleteObjectsException e)
        {
            // var errorResponse = e.ErrorResponse;
            DeleteObjectsResponse errorResponse = e.Response;
            Console.WriteLine("x {0}", errorResponse.DeletedObjects.Count);

            Console.WriteLine("No. of objects successfully deleted = {0}",
errorResponse.DeletedObjects.Count);
            Console.WriteLine("No. of objects failed to delete = {0}",
errorResponse.DeleteErrors.Count);

            Console.WriteLine("Printing error data...");
            foreach (DeleteError deleteError in errorResponse.DeleteErrors)
            {
                Console.WriteLine("Object Key: {0}\t{1}\t{2}", deleteError.Key,
deleteError.Code, deleteError.Message);
            }
        }

        static async Task<List<KeyVersion>> PutObjectsAsync(int number)
        {
            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 s3Client.PutObjectAsync(request);
                KeyVersion keyVersion = new KeyVersion
                {
                    Key = key,
                    // For non-versioned bucket operations, we only need object key.
                    // VersionId = response.VersionId
                };
                keys.Add(keyVersion);
            }
            return keys;
        }
    }
}
```

Example 버전 관리 사용 버킷의 여러 객체 삭제

다음 C# 코드 예제는 multi-object delete API를 사용하여 버전 관리를 사용하는 버킷에서 객체를 삭제합니다. 이 예제에서는 다음 작업을 수행합니다.

1. 샘플 객체를 만들고 각 객체의 키 이름과 버전 ID를 지정하여 삭제합니다. 이 작업에서는 특정 객체 버전을 삭제합니다.

2. 샘플 객체를 만들고 키 이름만 지정하여 삭제합니다. 이 예제에서는 버전 ID를 지정하지 않으므로 이 작업은 삭제 마커만 추가합니다. 특정 객체 버전은 삭제하지 않습니다. 삭제 후 이러한 객체는 Amazon S3 콘솔에 표시되지 않습니다.
3. 삭제 마커의 객체 키와 버전 ID를 지정하여 삭제 마커를 삭제합니다. 이 작업에서 삭제 마커를 삭제하면 해당 객체가 콘솔에 다시 표시됩니다.

실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

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

namespace Amazon.DocSamples.S3
{
    class DeleteMultipleObjVersionedBucketTest
    {
        private const string bucketName = "*** versioning-enabled 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()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            DeleteMultipleObjectsFromVersionedBucketAsync().Wait();
        }

        private static async Task DeleteMultipleObjectsFromVersionedBucketAsync()
        {

            // Delete objects (specifying object version in the request).
            await DeleteObjectVersionsAsync();

            // Delete objects (without specifying object version in the request).
            var deletedObjects = await DeleteObjectsAsync();

            // Additional exercise - remove the delete markers S3 returned in 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(deletedObjects);
        }

        private static async Task<List<DeletedObject>> DeleteObjectsAsync()
        {
            // Upload the sample objects.
            var keysAndVersions2 = await PutObjectsAsync(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(keysAndVersions2);
            return deletedObjects;
        }

        private static async Task DeleteObjectVersionsAsync()
        {
            // Upload the sample objects.
            var keysAndVersions1 = await PutObjectsAsync(3);
```

```
            // Delete the specific object versions.
            await VersionedDeleteAsync(keysAndVersions1);
        }

        private static void PrintDeletionReport(DeleteObjectsException e)
        {
            var errorResponse = e.Response;
            Console.WriteLine("No. of objects successfully deleted = {0}",
errorResponse.DeletedObjects.Count);
            Console.WriteLine("No. of objects failed to delete = {0}",
errorResponse.DeleteErrors.Count);
            Console.WriteLine("Printing error data...");
            foreach (var deleteError in errorResponse.DeleteErrors)
            {
                Console.WriteLine("Object Key: {0}\t{1}\t{2}", deleteError.Key,
deleteError.Code, deleteError.Message);
            }
        }

        static async Task VersionedDeleteAsync(List<KeyVersion> keys)
        {
            // a. Perform a multi-object delete by specifying the key names and version
IDs.
            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
s3Client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
            }
            catch (DeleteObjectsException e)
            {
                PrintDeletionReport(e);
            }
        }

        static async Task<List<DeletedObject>> NonVersionedDeleteAsync(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 DeleteObjects - Amazon S3 add delete marker for each object
            // deletion. The objects disappear from your bucket.
            // You can verify that using the Amazon S3 console.
            DeleteObjectsResponse response;
            try
            {
                Console.WriteLine("Executing NonVersionedDelete...");
                response = await s3Client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
            }
            catch (DeleteObjectsException e)
```

```
            {
                PrintDeletionReport(e);
                throw; // Some deletes failed. Investigate before continuing.
            }
            // This response contains the DeletedObjects list which we use to delete the
delete markers.
            return response.DeletedObjects;
        }

        private static async Task RemoveDeleteMarkersAsync(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
s3Client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine("Successfully deleted all the {0} delete markers",
                                              deleteObjectResponse.DeletedObjects.Count);
            }
            catch (DeleteObjectsException e)
            {
                PrintDeletionReport(e);
            }
        }

        static async Task<List<KeyVersion>> PutObjectsAsync(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 s3Client.PutObjectAsync(request);
                KeyVersion keyVersion = new KeyVersion
                {
                    Key = key,
                    VersionId = response.VersionId
                };
```

```
                keys.Add(keyVersion);
            }
            return keys;
        }
    }
}
```

PHP용 AWS SDK를 사용한 여러 객체 삭제

이 주제에서는 PHP용 AWS SDK 버전 3의 클래스를 사용하여 버전이 지정되거나 지정되지 않은 Amazon S3 버킷에서 여러 객체를 삭제하는 방법을 보여 줍니다. 버전 관리에 대한 자세한 내용은 버전 관리 사용 (p. 468) 단원을 참조하십시오.

이 주제에서는 이미 PHP용 AWS SDK 사용 및 PHP 예제 실행 (p. 736)의 지침에 따라 PHP용 AWS SDK가 올바르게 설치되어 있다고 가정합니다.

Example 버전 관리 미사용 버킷에서 여러 객체 삭제

다음 PHP 예제는 `deleteObjects()` 메서드를 사용하여 버전 관리 미사용 버킷에서 여러 객체를 삭제합니다.

이 가이드의 PHP 예제 실행에 대한 자세한 내용은 PHP 예제 실행 (p. 737) 단원을 참조하십시오.

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;

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

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

// 1. Create a few objects.
for ($i = 1; $i <= 3; $i++) {
    $s3->putObject([
        'Bucket' => $bucket,
        'Key'    => "key{$i}",
        'Body'   => "content {$i}",
    ]);
}

// 2. List the objects and get the keys.
$keys = $s3->listObjects([
    'Bucket' => $bucket
]) ->getPath('Contents/*/Key');

// 3. Delete the objects.
$s3->deleteObjects([
    'Bucket'  => $bucket,
    'Delete' => [
        'Objects' => array_map(function ($key) {
            return ['Key' => $key];
        }, $keys)
    ],
]);
```

Example 버전 관리 사용 버킷에서 여러 객체 삭제

다음 PHP 예제는 `deleteObjects()` 메서드를 사용하여 버전이 지정된 버킷에서 여러 객체를 삭제합니다.

이 가이드의 PHP 예제 실행에 대한 자세한 내용은 PHP 예제 실행 (p. 737) 단원을 참조하십시오.

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

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

// 1. Enable object versioning for the bucket.
$s3->putBucketVersioning([
    'Bucket' => $bucket,
    'Status' => 'Enabled',
]);

// 2. Create a few versions of an object.
for ($i = 1; $i <= 3; $i++) {
    $s3->putObject([
        'Bucket' => $bucket,
        'Key'    => $keyname,
        'Body'   => "content {$i}",
    ]);
}

// 3. List the objects versions and get the keys and version IDs.
$versions = $s3->listObjectVersions(['Bucket' => $bucket])
    ->getPath('Versions');

// 4. Delete the object versions.
$s3->deleteObjects([
    'Bucket'  => $bucket,
    'Delete' => [
        'Objects' => array_map(function ($version) {
          return [
              'Key'       => $version['Key'],
              'VersionId' => $version['VersionId']
        }, $versions),
    ],
]);

echo "The following objects were deleted successfully:". PHP_EOL;
foreach ($result['Deleted'] as $object) {
    echo "Key: {$object['Key']}, VersionId: {$object['VersionId']}" . PHP_EOL;
}

echo PHP_EOL . "The following objects could not be deleted:" . PHP_EOL;
foreach ($result['Errors'] as $object) {
    echo "Key: {$object['Key']}, VersionId: {$object['VersionId']}" . PHP_EOL;
}

// 5. Suspend object versioning for the bucket.
$s3->putBucketVersioning([
    'Bucket' => $bucket,
    'Status' => 'Suspended',
```

```
]);
```

관련 리소스

- Amazon S3용 PHP용 AWS SDK Aws\S3\S3Client 클래스
- PHP용 AWS SDK 설명서

REST API를 사용한 여러 객체 삭제

AWS SDK를 사용하여 Multi-Object Delete API를 통해 여러 객체를 삭제할 수 있습니다. 하지만 애플리케이션에서 요구할 경우 REST 요청을 직접 전송할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API Reference의 여러 객체 삭제를 참조하십시오.

객체에서 콘텐츠 선택

Amazon S3 Select에서는 간단한 구조화 질의 언어(SQL) 문을 사용하여 Amazon S3 객체의 콘텐츠를 필터링하고 필요한 데이터 하위 집합만 검색할 수 있습니다. Amazon S3 Select를 통해 이 데이터를 필터링함으로써 Amazon S3가 전송하는 데이터의 양을 줄일 수 있고 따라서 이 데이터를 검색하는 데 드는 비용과 지연 시간이 감소됩니다.

Amazon S3 Select는 CSV, JSON 또는 Apache Parquet 형식으로 저장된 객체에 작동합니다. 이 기능은 GZIP 또는 BZIP2로 압축된 객체(CSV 및 JSON 객체에 한함)와 서버 측 암호화된 객체에도 사용할 수 있습니다. 결과 형식을 CSV 또는 JSON으로 지정할 수 있으며 결과의 레코드를 어떻게 구분할 것인지 결정할 수 있습니다.

요청에서 SQL 표현식을 Amazon S3로 전달합니다. Amazon S3 Select는 SQL의 하위 집합을 지원합니다. Amazon S3 Select가 지원하는 SQL 요소에 대한 자세한 내용은 Amazon S3 Select와 S3 Glacier Select의 SQL 참조 (p. 771) 단원을 참조하십시오.

AWS SDK, SELECT Object Content REST API, AWS Command Line Interface(AWS CLI) 또는 Amazon S3 콘솔을 사용하여 SQL 쿼리를 실행할 수 있습니다. Amazon S3 콘솔은 반환되는 데이터의 양을 40MB로 제한합니다. 더 많은 데이터를 검색하려면 AWS CLI 또는 API를 사용하십시오.

요구 사항 및 제한

다음은 Amazon S3 Select를 사용하기 위한 요구 사항입니다.

- 쿼리를 하는 객체에 대해 `s3:GetObject` 권한이 있어야 합니다.
- 사용자가 쿼리하는 객체가 고객 제공 암호화 키(SSE-C)로 암호화된 경우 `https`를 사용해야 하고 요청에서 암호화 키를 제공해야 합니다.

Amazon S3 Select를 사용할 때는 다음 제한이 적용됩니다.

- SQL 표현식의 최대 길이는 256KB입니다.
- 입력 또는 결과에 표시되는 레코드의 최대 길이는 1MB입니다.
- Amazon S3 Select는 JSON 출력 형식을 사용하는 중첩 데이터만 내보낼 수 있습니다.

Amazon S3 Select를 Parquet 객체와 함께 사용할 경우 추가 제한 사항:

- Amazon S3 Select는 GZIP 또는 Snappy를 사용한 열 기반 압축만 지원합니다. Amazon S3 Select는 Parquet 객체의 전체 객체 압축을 지원하지 않습니다.
- Amazon S3 Select는 Parquet 출력을 지원하지 않습니다. 출력 형식은 CSV 또는 JSON으로 지정해야 합니다.

- 최대 무압축 행 그룹 크기는 256MB입니다.
- 객체의 스키마에 지정된 데이터 형식을 사용해야 합니다.
- 반복 필드를 선택할 경우 마지막 값만 반환됩니다.

요청의 구성

요청을 구성할 때는 `InputSerialization` 객체를 사용하여 쿼리되는 객체의 세부 정보를 제공합니다. `OutputSerialization` 객체를 사용하여 결과가 어떻게 반환될 것인지에 대한 세부 정보를 제공합니다. Amazon S3가 요청을 필터링할 때 사용하는 SQL 표현식도 포함시킵니다.

Amazon S3 Select 요청의 구성에 대한 자세한 내용은 Amazon Simple Storage Service API Reference의 SELECT 객체 콘텐츠 단원을 참조하십시오. 또한, 다음 단원에서 SDK 코드의 예시 중 하나를 확인할 수 있습니다.

스캔 범위를 사용하여 요청

Amazon S3 Select를 이용하면 쿼리할 바이트 범위를 지정하여 객체의 하위 세트를 스캔할 수 있습니다. 이를 통해 작업을 일련의 겹치지 않는 스캔 범위에 대한 별도의 Amazon S3 Select 요청으로 분할하여 전체 객체 스캐닝을 병렬화할 수 있습니다. 스캔 범위는 레코드 경계에 맞게 조정할 필요는 없습니다. Amazon S3 Select 스캔 범위 요청은 지정된 바이트 범위에 걸쳐 실행됩니다. 지정된 스캔 범위 내에서 시작하지만 스캔 범위를 벗어나는 레코드는 쿼리에 의해 처리됩니다. 예를 들어 줄로 구분된 CSV 형식으로 된 일련의 레코드를 포함하는 Amazon S3 객체를 사용할 수 있습니다.

```
A,B
C,D
D,E
E,F
G,H
I,J
```

Amazon S3 Select `ScanRange` 파라미터를 사용하여 (바이트) 1에서 시작하고 (바이트) 4에서 종료합니다. 그러면 스캔 범위는 ","에서 시작하고, "C"에서 레코드 시작의 끝까지 스캔하고, 레코드의 끝이기 때문에 결과로 C, D를 반환합니다.

Amazon S3 Select 스캔 범위 요청은 Parquet, CSV(따옴표로 묶은 구분자 사용하지 않음) 및 JSON객체(LINES 모드에서만 해당)를 지원합니다. CSV 및 JSON 객체는 압축되어 있지 않아야 합니다. 선 기반 CSV 및 JSON 객체의 경우, 스캔 범위가 Amazon S3 Select 요청의 일부로 지정되면 스캔 범위 내에서 시작하는 모든 레코드가 처리됩니다. Parquet 객체의 경우, 요청된 스캔 범위 내에서 시작하는 모든 행 그룹이 처리됩니다.

Amazon S3 Select 스캔 범위 요청은 Amazon S3 CLI, API 및 SDK에서 실행할 수도 있습니다. 이 기능을 위한 Amazon S3 Select 요청의 `ScanRange` 파라미터를 사용할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API Reference의 Amazon S3 SELECT Object Content를 참조하십시오.

오류

Amazon S3 Select는 쿼리 실행을 시도하는 동안 문제가 발생할 때 오류 코드 및 관련 오류 메시지를 반환합니다. 오류 코드 및 설명 목록은 Amazon Simple Storage Service API Reference의 오류 응답 페이지에 있는 SELECT 객체 콘텐츠 오류 코드 단원을 참조하십시오.

주제

- 관련 리소스 (p. 245)
- Java용 SDK를 사용하여 객체에서 콘텐츠 선택 (p. 245)
- REST API를 사용하여 객체에서 콘텐츠 선택 (p. 247)
- 다른 SDK를 사용하여 객체에서 콘텐츠 선택 (p. 247)

관련 리소스

- AWS SDK, CLI 및 Explorer 사용 (p. 727)

Java용 SDK를 사용하여 객체에서 콘텐츠 선택

Amazon S3 Select를 사용하여 `selectObjectContent` 메서드로 Java가 적용된 객체의 콘텐츠를 선택합니다. 성공할 경우 SQL 표현식의 결과가 반환됩니다. 지정한 버킷 및 객체 키가 존재해야 합니다. 그렇지 않으면 오류가 발생합니다.

Example 예

아래의 Java 코드는 CSV 형식으로 저장된 데이터를 포함하는 객체에 저장되어 있는 모든 레코드에 대해 첫 번째 열의 값을 반환합니다. 또한 `Progress` 및 `Stats` 메시지를 반환하도록 요청합니다. 유효한 버킷 이름과 CSV 형식의 데이터를 포함하는 객체를 제공해야 합니다.

실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
package com.amazonaws;

import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CSVInput;
import com.amazonaws.services.s3.model.CSVOutput;
import com.amazonaws.services.s3.model.CompressionType;
import com.amazonaws.services.s3.model.ExpressionType;
import com.amazonaws.services.s3.model.InputSerialization;
import com.amazonaws.services.s3.model.OutputSerialization;
import com.amazonaws.services.s3.model.SelectObjectContentEvent;
import com.amazonaws.services.s3.model.SelectObjectContentEventVisitor;
import com.amazonaws.services.s3.model.SelectObjectContentRequest;
import com.amazonaws.services.s3.model.SelectObjectContentResult;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.concurrent.atomic.AtomicBoolean;

import static com.amazonaws.util.IOUtils.copy;

/**
 * This example shows how to query data from S3Select and consume the response in the form
 of an
 * InputStream of records and write it to a file.
 */

public class RecordInputStreamExample {

    private static final String BUCKET_NAME = "${my-s3-bucket}";
    private static final String CSV_OBJECT_KEY = "${my-csv-object-key}";
    private static final String S3_SELECT_RESULTS_PATH = "${my-s3-select-results-path}";
    private static final String QUERY = "select s._1 from S3Object s";

    public static void main(String[] args) throws Exception {
        final AmazonS3 s3Client = AmazonS3ClientBuilder.defaultClient();

        SelectObjectContentRequest request = generateBaseCSVRequest(BUCKET_NAME,
 CSV_OBJECT_KEY, QUERY);
        final AtomicBoolean isResultComplete = new AtomicBoolean(false);
```

```
        try (OutputStream fileOutputStream = new FileOutputStream(new File
 (S3_SELECT_RESULTS_PATH));
             SelectObjectContentResult result = s3Client.selectObjectContent(request)) {
            InputStream resultInputStream = result.getPayload().getRecordsInputStream(
                    new SelectObjectContentEventVisitor() {
                        @Override
                        public void visit(SelectObjectContentEvent.StatsEvent event)
                        {
                            System.out.println(
                                    "Received Stats, Bytes Scanned: " +
 event.getDetails().getBytesScanned()
                                            +  " Bytes Processed: " +
 event.getDetails().getBytesProcessed());
                        }

                        /*
                         * An End Event informs that the request has finished successfully.
                         */
                        @Override
                        public void visit(SelectObjectContentEvent.EndEvent event)
                        {
                            isResultComplete.set(true);
                            System.out.println("Received End Event. Result is complete.");
                        }
                    }
            );

            copy(resultInputStream, fileOutputStream);
        }

        /*
         * The End Event indicates all matching records have been transmitted.
         * If the End Event is not received, the results may be incomplete.
         */
        if (!isResultComplete.get()) {
            throw new Exception("S3 Select request was incomplete as End Event was not
 received.");
        }
    }

    private static SelectObjectContentRequest generateBaseCSVRequest(String bucket, String
 key, String query) {
        SelectObjectContentRequest request = new SelectObjectContentRequest();
        request.setBucketName(bucket);
        request.setKey(key);
        request.setExpression(query);
        request.setExpressionType(ExpressionType.SQL);

        InputSerialization inputSerialization = new InputSerialization();
        inputSerialization.setCsv(new CSVInput());
        inputSerialization.setCompressionType(CompressionType.NONE);
        request.setInputSerialization(inputSerialization);

        OutputSerialization outputSerialization = new OutputSerialization();
        outputSerialization.setCsv(new CSVOutput());
        request.setOutputSerialization(outputSerialization);

        return request;
    }
}
```

REST API를 사용하여 객체에서 콘텐츠 선택

AWS SDK를 사용하여 객체에서 콘텐츠를 선택할 수 있습니다. 하지만 애플리케이션에서 요구할 경우 REST 요청을 직접 전송할 수 있습니다. 요청 및 응답 형식에 대한 자세한 내용은 SELECT Object Content를 참조하십시오.

다른 SDK를 사용하여 객체에서 콘텐츠 선택

다른 SDK로 Amazon S3 Select를 사용하여 객체의 콘텐츠를 선택할 수 있습니다. 자세한 내용은 다음 자료를 참조하십시오.

- Python: AWS SDK for Python (Boto) 사용 (p. 738).

보관된 객체의 복원

S3 Glacier 또는 S3 Glacier Deep Archive 스토리지 클래스에 보관하는 객체는 실시간으로 액세스할 수 없습니다. 먼저 복원 요청을 시작한 후 요청에 지정된 기간(일수) 동안 객체의 임시 사본을 사용할 수 있게 될 때까지 기다려야 합니다. S3 Glacier, S3 Glacier Deep Archive 및 기타 Amazon S3 스토리지 클래스 비교 방법에 대한 자세한 내용은 Amazon S3 스토리지 클래스 (p. 107) 단원을 참조하십시오.

Amazon S3는 지정된 기간 동안만 객체의 임시 사본을 복원합니다. 그런 다음 복원된 객체 복사본을 삭제합니다. 복원을 다시 발행하여 복원된 복사본의 만료 기간을 수정할 수 있습니다. 이 경우 Amazon S3은 현재 시간을 기준으로 만료 기간을 업데이트합니다.

Amazon S3은 복원 요청에 지정된 일수를 현재 시간에 더하여 복원된 객체 복사본의 만료 시간을 계산합니다. 그런 다음 협정 세계시(UTC) 자정에 그 다음 날로 결과 시간을 반올림합니다. 예를 들어 객체가 2012년 10월 15일 오전 10시 30 분에 생성되었으며 복원 기간이 3일로 지정되었다고 가정하십시오. 이 경우 복원된 복사본은 2012년 10월 19일 00:00 UTC에 만료되며 이 시간에 Amazon S3이 객체 복사본을 삭제합니다.

복원된 객체의 임시 복사본을 받으면 객체의 스토리지 클래스는 S3 Glacier 또는 S3 Glacier Deep Archive에 유지됩니다. HEAD 객체 또는 GET 객체 API 작업 요청은 S3 Glacier 또는 S3 Glacier Deep Archive를 스토리지 클래스로 반환합니다.

복원 작업을 완료하는 데 걸리는 시간은 사용하는 아카이브 스토리지 클래스와 지정하는 검색 옵션에 따라 다릅니다(`Expedited`(S3 Glacier에만 사용 가능), `Standard` 또는 `Bulk`). 복원이 완료되면 Amazon S3 이벤트 알림을 사용하여 알림을 받을 수 있습니다. 자세한 내용은 Amazon S3 이벤트 알림 구성 (p. 587) 단원을 참조하십시오.

객체 사본의 복원 일수에는 제한이 없습니다. 하지만 객체 사본에 대해 스토리지 비용이 부과되기 때문에 필요한 기간 동안만 객체를 복원해야 합니다. 아카이브를 복원할 때 아카이브(S3 Glacier 또는 S3 Glacier Deep Archive 요금) 및 임시로 복원한 복사본(Reduced Redundancy Storage(RRS) 또는 표준 중 리전에서 가장 낮은 비용의 스토리지) 모두에 대해 비용이 부과됩니다. 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

필요한 경우 S3 Glacier 및 S3 Glacier Deep Archive 스토리지 클래스에 저장된 대용량 데이터 세그먼트를 복원할 수 있습니다. 예를 들어 보조 사본의 데이터를 복원하고자 할 수 있습니다. 그러나 많은 양의 데이터를 복원해야 하더라도, S3 Glacier 및 S3 Glacier Deep Archive 스토리지 클래스는 하루에 저장된 페비바이트(PiB)당 35개의 임의 복원 요청을 처리하도록 설계되었습니다.

수명 주기 전환을 사용하여 S3 Glacier 또는 S3 Glacier Deep Archive 스토리지 클래스로 객체를 이동하는 방법에 대한 자세한 내용은 S3 Glacier 및 S3 Glacier Deep Archive 스토리지 클래스로 전환(객체 보관) (p. 127) 단원을 참조하십시오.

단일 요청으로 Amazon S3 객체를 하나 이상 복원하려면 Amazon S3 배치 작업을 사용할 수 있습니다. 작업할 객체 목록을 Amazon S3 Batch 작업에 제공합니다. Amazon S3 Batch 작업은 각각의 API를 호출하여 지정된 작업을 수행합니다. 단일 Amazon S3 Batch 작업으로 엑사바이트 규모의 데이터가 포함된 수십억 개의 객체에서 지정된 작업을 수행할 수 있습니다.

Amazon S3 Batch 작업은 진행 상황을 추적하고 알림을 보내며 모든 작업에 대한 자세한 완료 보고서를 저장하여 감사 가능한 종합 관리형 서버리스 환경을 제공합니다. AWS Management 콘솔, AWS CLI, AWS SDK 또는 REST API를 통해 Amazon S3 Batch 작업을 사용할 수 있습니다. 자세한 내용은 the section called “기본 사항: 작업” (p. 502) 단원을 참조하십시오.

다음 단원에서는 객체 복원에 대한 자세한 내용을 제공합니다.

주제

- 아카이브 검색 옵션 (p. 248)
- 진행 중인 복원 속도 업그레이드 (p. 249)
- Amazon S3 콘솔을 사용하여 아카이브된 객체 복원 (p. 249)
- AWS SDK for Java를 사용하여 아카이브된 객체 복원 (p. 249)
- .NET용 AWS SDK를 사용하여 아카이브된 객체 복원 (p. 250)
- REST API를 사용하여 아카이브된 객체 복원 (p. 251)

아카이브 검색 옵션

다음은 아카이브된 객체를 복원할 때 사용 가능한 검색 옵션입니다.

- **`Expedited`** - 신속 검색을 사용하면 가끔 발생하는 아카이브의 하위 집합에 대한 긴급 요청이 필요할 때 S3 Glacier 스토리지 클래스에 저장된 데이터에 신속하게 액세스할 수 있습니다. 아카이브된 가장 큰 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. 자세한 내용은 프로비저닝된 용량 (p. 248) 단원을 참조하십시오. S3 Glacier Deep Archive 스토리지 클래스에 저장된 객체에는 신속 검색 및 프로비저닝된 용량을 사용할 수 없습니다.
- **`Standard`** - 표준 검색을 사용하면 몇 시간 내에 아카이브된 모든 객체에 액세스할 수 있습니다. S3 Glacier 및 S3 Glacier Deep Archive 검색 요청 시 검색 옵션을 지정하지 않을 경우 기본 옵션이 됩니다. 표준 검색은 일반적으로 S3 Glacier 스토리지 클래스에 저장된 객체의 경우 3~5시간 이내에 완료됩니다. 일반적으로 S3 Glacier Deep Archive 스토리지 클래스에 저장된 객체의 경우 12시간 이내에 완료됩니다.
- **`Bulk`** - 대량 검색은 Amazon S3 Glacier에서 가장 저렴한 검색 옵션으로 페타바이트 단위의 대용량 데이터도 저렴하게 검색할 수 있습니다. 대량 검색은 일반적으로 S3 Glacier 스토리지 클래스에 저장된 객체의 경우 5~12시간 이내에 완료됩니다. 일반적으로 S3 Glacier Deep Archive 스토리지 클래스에 저장된 객체의 경우 48시간 이내에 완료됩니다.

다음 테이블에는 아카이브 검색 옵션이 요약되어 있습니다.

`Expedited`, `Standard` 또는 `Bulk` 검색을 실행하려면, POST 객체 복원 REST API 요청의 `Tier` 요청 요소를 사용자가 원하는 옵션이나 AWS CLI 또는 AWS SDK의 동등한 옵션으로 설정합니다. 프로비저닝된 용량을 구매하였다면, 사용자의 프로비저닝된 용량을 통해 모든 신속 검색이 자동으로 수행됩니다.

프로그래밍 방식으로 또는 Amazon S3 콘솔을 사용하여 보관된 객체를 복원할 수 있습니다. Amazon S3는 각 객체에 대해 한 번에 하나의 복원 요청만 처리합니다. 콘솔 및 Amazon S3 API를 모두 사용하여 복원 상태, 그리고 Amazon S3가 복원된 사본을 삭제할 시기를 확인할 수 있습니다.

프로비저닝된 용량

프로비저닝된 용량을 통해 필요할 때 신속 검색에 대한 검색 용량이 보장됩니다. 각 용량 단위로 5분마다 신속 검색을 최소 3회 수행할 수 있고, 최대 150MB/s의 검색 처리량이 제공됩니다.

워크로드에 몇 분 내로 데이터의 하위 집합에 대한 매우 안전하고 예측 가능한 액세스가 필요한 경우 프로비저닝된 검색 용량을 구매해야 합니다. 프로비저닝된 용량이 없으면 신속 검색은 수요가 많은 기간에는 수락되지 않을 수도 있습니다. 모든 상황에서 신속 검색에 액세스해야 하는 경우 프로비저닝된 검색 용량을 구입하는 것이 좋습니다.

Amazon S3 콘솔, Amazon S3 Glacier 콘솔, 프로비저닝된 용량 구매 REST API, AWS SDK 또는 AWS CLI를 사용하여 프로비저닝된 용량을 구매할 수 있습니다. 프로비저닝된 용량의 요금에 대한 자세한 내용은 Amazon S3 요금 항목을 참조하십시오.

프로비저닝된 용량을 사용하는 신속 검색에는 여전히 요청 및 검색 요금이 부과되며 S3 Glacier Deep Archive 스토리지 클래스에는 사용할 수 없습니다.

진행 중인 복원 속도 업그레이드

Amazon S3 복원 속도 업그레이드를 사용하면 복원이 진행되는 동안 복원 속도를 더 빠른 속도로 변경할 수 있습니다. 복원 속도 업그레이드는 진행 중인 복원을 더 빠른 복원 티어로 재정의합니다. 진행 중인 복원 속도를 늦출 수 없습니다.

진행 중인 복원의 속도를 업그레이드하려면 POST 객체 복원 REST API에 새로운 `Tier` 요청 요소 또는 AWS CLI 또는 AWS SDK에 동등한 요소를 설정하여 동일한 객체에 다른 복원 요청을 발행합니다. 복원 티어를 업그레이드하라는 요청을 발행할 때는 진행 중인 복원이 사용하는 티어보다 빠른 티어를 선택해야 합니다. `Days` 요청 요소 등 다른 매개 변수는 변경하면 안 됩니다.

Amazon S3 이벤트 알림을 사용하여 복원 완료에 대한 알림을 받을 수 있습니다. 복원은 업그레이드된 티어 가격으로 청구됩니다. 복원 요금에 대한 자세한 내용은 Amazon S3 요금 항목을 참조하십시오.

Amazon S3 콘솔을 사용하여 아카이브된 객체 복원

Amazon S3 콘솔을 사용하여 Amazon S3 Glacier에 보관된 객체의 사본을 복원할 수 있습니다. AWS Management 콘솔을 사용하여 아카이브를 복원하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 Amazon S3 Glacier에 보관된 S3 객체를 복원하려면 어떻게 해야 합니까? 단원을 참조하십시오.

아카이브를 복원하면 아카이브 및 임시로 복원한 복사본 모두에 대해 요금이 청구됩니다. 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

AWS SDK for Java를 사용하여 아카이브된 객체 복원

Example

다음 예제에서는 AWS SDK for Java를 사용하여 Amazon S3 Glacier에 보관된 객체를 복원하는 방법을 보여줍니다. 다음 예제는 아카이브된 지정 객체에 대한 복원 요청을 시작하고 복원 상태를 확인합니다. 보관된 객체 복원에 대한 자세한 내용은 보관된 객체의 복원 (p. 247) 단원을 참조하십시오. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
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;
import com.amazonaws.services.s3.model.ObjectMetadata;
import com.amazonaws.services.s3.model.RestoreObjectRequest;

import java.io.IOException;

public class RestoreArchivedObject {

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

        try {
```

```
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Create and submit a request to restore an object from Glacier for two days.
            RestoreObjectRequest requestRestore = new RestoreObjectRequest(bucketName,
 keyName, 2);
            s3Client.restoreObjectV2(requestRestore);

            // Check the restoration status of the object.
            ObjectMetadata response = s3Client.getObjectMetadata(bucketName, keyName);
            Boolean restoreFlag = response.getOngoingRestore();
            System.out.format("Restoration status: %s.\n",
                    restoreFlag ? "in progress" : "not in progress (finished or failed)");
        } 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();
        }
    }
}
```

.NET용 AWS SDK를 사용하여 아카이브된 객체 복원

Example

다음 C# 예제는 2일간 보관된 객체에 대한 복원 요청을 시작합니다. Amazon S3는 객체 메타데이터에 복원 상태를 유지합니다. 요청을 시작한 후 이 예제에서는 객체 메타데이터를 가져와 `RestoreInProgress` 속성의 값을 확인합니다. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

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

namespace Amazon.DocSamples.S3
{
    class RestoreArchivedObjectTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string objectKey = "** archived object key 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()
        {
            client = new AmazonS3Client(bucketRegion);
            RestoreObjectAsync(client, bucketName, objectKey).Wait();
        }

        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("An AmazonS3Exception was thrown. Exception: " +
 amazonS3Exception.ToString());
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: " + e.ToString());
            }
        }

        static async Task CheckRestorationStatusAsync(IAmazonS3 client, string bucketName,
 string objectKey)
        {
            GetObjectMetadataRequest metadataRequest = new GetObjectMetadataRequest
            {
                BucketName = bucketName,
                Key = objectKey
            };
            GetObjectMetadataResponse response = await
 client.GetObjectMetadataAsync(metadataRequest);
            Console.WriteLine("restoration status: {0}", response.RestoreInProgress ? "in-
progress" : "finished or failed");
        }
    }
}
```

REST API를 사용하여 아카이브된 객체 복원

Amazon S3에서 제공한 API로 아카이브 복원을 시작합니다. 자세한 내용은 Amazon Simple Storage Service API Reference의 POST Object restore 단원을 참조하십시오.

보관된 객체의 쿼리

Select 유형의 POST Object restore로 간단한 구조화 질의 언어(SQL) 문을 Amazon S3이 S3 Glacier에 보관한 데이터에서 직접 사용하여 필터링 작업을 수행할 수 있습니다. 보관된 객체에 대해 SQL 쿼리를 제공하면 select가 쿼리를 실행하고 출력 결과를 S3 버킷에 작성합니다. 전체 객체를 Amazon S3로 복원할 필요 없이 S3 Glacier에 저장된 데이터에서 쿼리 및 사용자 지정 분석을 실행할 수 있습니다.

선택 쿼리를 수행할 때 S3 Glacier은 세 가지 데이터 액세스 계층—신속, 표준, 벌크를 제공합니다. 이 계층들은 데이터 액세스 시간과 비용이 서로 다르므로 데이터를 얼마나 빨리 사용하기 원하는지에 따라 선택하면 됩니다. 자세한 내용은 데이터 액세스 계층 (p. 253) 단원을 참조하십시오.

AWS SDK, S3 Glacier REST API 및 AWS Command Line Interface(AWS CLI)와 함께 Select 유형의 복원을 사용할 수 있습니다.

주제

- Select 요구 사항 및 제한 (p. 252)

- Select를 사용하여 데이터를 쿼리하려면 어떻게 해야 합니까? (p. 252)
- 오류 처리 (p. 253)
- 데이터 액세스 계층 (p. 253)
- 추가 정보 (p. 254)

Select 요구 사항 및 제한

다음은 Select를 사용하기 위한 요구 사항입니다.

- Select에 의해 쿼리되는 아카이브 객체는 압축되지 않은, 쉼표로 분리된 값(CSV) 형식이어야 합니다.
- 출력을 위한 S3 버킷. S3 Glacier Select 작업을 시작하는 데 사용하는 AWS 계정에는 S3 버킷에 대한 쓰기 권한이 있어야 합니다. 버킷은 쿼리되는 아카이브 객체를 포함하는 버킷과 동일한 AWS 리전에 있어야 합니다.
- 요청하는 AWS 계정에 `s3:RestoreObject` 및 `s3:GetObject` 작업을 수행할 수 있는 권한이 있어야 합니다. 이러한 권한에 대한 자세한 내용은 — 버킷 하위 리소스 작업 예제 (p. 344) 단원을 참조하십시오.
- 아카이브가 SSE-C 또는 클라이언트측 암호화를 사용하여 암호화되지 않아야 합니다.

Select를 사용할 때는 다음 제한이 적용됩니다.

- Select가 처리할 수 있는 레코드의 수에는 제한이 없습니다. 입력 또는 출력 레코드는 1MB를 초과해서는 안 되며, 초과할 경우 쿼리가 실패합니다. 레코드당 1,048,576열의 제한이 있습니다.
- 최종 결과의 크기에는 제한이 없습니다. 다만 결과는 여러 파트로 나뉩니다.
- SQL 표현식은 128KB로 제한됩니다.

Select를 사용하여 데이터를 쿼리하려면 어떻게 해야 합니까?

Select를 사용하면 SQL 명령으로 압축되지 않은 암호화 CSV 형식의 S3 Glacier 아카이브 객체를 쿼리할 수 있습니다. 이러한 제한이 적용되므로 에 있는 텍스트 기반 데이터에서 간단한 쿼리 작업을 수행할 수 있습니다. 예를 들어, 보관된 텍스트 파일 집합 중에서 특정 이름이나 ID를 찾을 수 있습니다.

S3 Glacier 데이터를 쿼리하려면 POST Object restore 작업을 사용하여 Select 요청을 생성합니다. select 요청을 수행할 때 SQL 표현식, 쿼리할 아카이브, 결과를 저장할 위치를 제공합니다.

다음 예제 표현식은 POST Object restore에 지정된 아카이브 객체로부터 모든 레코드를 반환합니다.

```
SELECT * FROM object
```

S3 Glacier Select는 ANSI SQL 언어 하위 집합을 지원합니다. `SELECT`, `FROM` 및 `WHERE` 같은 일반적인 필터링 SQL 절을 지원하지만 `SUM`, `COUNT`, `GROUP BY`, `JOINS`, `DISTINCT`, `UNION`, `ORDER BY` 및 `LIMIT`는 지원하지 않습니다. SQL 지원에 대한 자세한 내용은 Amazon Simple Storage Service 개발자 가이드의 Amazon S3 Select 및 S3 Glacier Select에 대한 SQL 참조 단원을 참조하십시오.

Select 출력

Select 작업을 시작할 때 선택 쿼리 결과의 출력 위치를 정의해야 합니다. 이 위치는 쿼리되는 보관된 객체가 포함된 버킷과 동일한 AWS 리전의 Amazon S3 버킷이어야 합니다. 작업을 시작하는 AWS 계정은 S3 버킷에 대한 쓰기 권한이 있어야 합니다.

Amazon S3에 저장되는 출력 객체의 Amazon S3 스토리지 클래스와 암호화를 지정할 수 있습니다. Select는 AWS Key Management Service(SSE-KMS) 및 Amazon S3(SSE-S3) 암호화를 지원합니다. Select는 SSE-C 및 클라이언트 측 암호화를 지원하지 않습니다. Amazon S3 스토리지 클래스 및 암호화에 대한 자세한 내용은 Amazon S3 스토리지 클래스 (p. 107) 및 서버 측 암호화를 사용하여 데이터 보호 (p. 263) 단원을 참조하십시오.

S3 Glacier Select 결과는 POST Object restore에서 지정된 출력 위치에 제공된 접두사를 사용하여 S3 버킷에 저장됩니다. 이 정보를 가지고 select는 작업 ID를 참조하는 고유 접두사를 생성합니다. (접두사는 공통 문자열로 객체 이름을 시작함으로써 객체를 그룹화하는 데 사용됩니다.) 이 고유한 접두사 아래에 새로운 접두사 2개가 생성됩니다(결과를 위한 접두사 `results`와 로그 및 오류를 위한 접두사 `errors`). 작업이 완료되면 모든 결과의 위치가 포함된 결과 매니페스트가 작성됩니다.

출력 위치에 작성되는 `job.txt`라는 자리 표시자 파일도 있습니다. 이 파일은 작성된 후 업데이트되지 않습니다. 자리 표시자 파일은 다음에 사용됩니다.

- 쓰기 권한과 SQL 구문 오류 대부분 동시 확인.
- 원할 때는 언제나 쉽게 참조할 수 있는 select 요청에 대한 정적 출력 제공.

예를 들어, 결과의 출력 위치가 `s3://example-bucket/my-prefix`로 지정된 Select 요청을 하고, 작업 응답이 `examplekne1209ualkdjh812elkassdu9012e`로 작업 ID를 반환한다고 가정해 보겠습니다. 선택 작업이 종료된 후 버킷에서 다음과 같은 객체를 볼 수 있습니다.

```
s3://example-bucket/my-prefix/examplekne1209ualkdjh812elkassdu9012e/job.txt
s3://example-bucket/my-prefix/examplekne1209ualkdjh812elkassdu9012e/results/abc
s3://example-bucket/my-prefix/examplekne1209ualkdjh812elkassdu9012e/results/def
s3://example-bucket/my-prefix/examplekne1209ualkdjh812elkassdu9012e/results/ghi
s3://example-bucket/my-prefix/examplekne1209ualkdjh812elkassdu9012e/result_manifest.txt
```

선택 쿼리 결과는 여러 파트로 나뉩니다. 이 예제에서 Select는 출력 위치를 설정할 때 지정한 접두사를 사용하고, 작업 ID와 `results` 접두사를 추가합니다. 그런 다음 결과를 세 파트로 작성합니다(객체 이름이 `abc`, `def`, `ghi`로 끝납니다). 결과 매니페스트에는 프로그래밍 방식의 가져오기를 허용하기 위해 3개 파일이 모두 포함됩니다. 작업이 오류와 함께 실패하면 오류 접두사 아래 파일이 표시되고 `error_manifest.txt`가 생성됩니다.

`result_manifest.txt` 파일이 존재하면서 `error_manifest.txt`가 없다면 작업이 성공적으로 완료된 것입니다. 결과가 어떻게 정렬되는지는 보장할 수 없습니다.

> **Note**
>
> 키라고도 하는 Amazon S3 객체 이름의 길이는 1,024바이트를 넘을 수 없습니다. S3 Glacier에서는 접두사를 위해 128바이트가 예약되어 있습니다. 위치 경로의 길이는 512바이트를 넘을 수 없습니다. 길이가 512바이트를 넘는 요청은 예외를 반환하며, 수락되지 않습니다.

오류 처리

Select는 두 가지 오류를 알립니다. 첫 번째 오류 집합은 사용자가 POST Object restore에서 쿼리를 제출하는 것과 동시에 사용자에게 전송됩니다. 이러한 오류는 HTTP 응답의 일부로 전송됩니다. 또 다른 오류 집합은 쿼리가 성공적으로 수락된 후에 발생할 수 있지만 쿼리 실행 중에 발생합니다. 이 경우, 오류는 `errors` 접두사 아래 지정된 출력 위치에 작성됩니다.

Select는 오류 발생 후 쿼리 실행을 중지합니다. 쿼리를 성공적으로 실행하려면 모든 오류를 해결해야 합니다. 로그를 확인하여 실패를 초래한 레코드를 식별할 수 있습니다.

쿼리가 여러 컴퓨터 노드에서 병렬로 실행되므로 오류는 순서대로 나열되지 않습니다. 예를 들어, 쿼리가 행 6,234에서의 오류와 함께 실패하는 경우, 행 6,234 앞의 모든 행이 성공적으로 처리되었음을 뜻하지 않습니다. 다음 쿼리 실행에서 다른 행의 오류가 표시될 수도 있습니다.

데이터 액세스 계층

보관된 객체를 쿼리할 때 다음 데이터 액세스 계층 중 하나를 지정할 수 있습니다.

- **`Expedited`** – 아카이브의 하위 집합에 대한 긴급 요청이 간헐적으로 필요한 경우, 신속 계층을 통해 빠르게 데이터에 액세스할 수 있습니다. 매우 큰 보관된 객체(250MB 이상)를 제외한 모든 경우, `Expedited`

가져오기를 사용하여 액세스된 데이터는 일반적으로 1~5분 안에 사용할 수 있게 됩니다. `Expedited` 데이터 액세스는 온디맨드와 프로비저닝의 두 가지 유형이 있습니다. 온디맨드 요청은 EC2 온디맨드 인스턴스와 유사하고, 대부분 경우에 이용할 수 있습니다. 프로비저닝 요청은 필요한 경우에 이용이 보장됩니다. 자세한 내용은 프로비저닝된 용량 (p. 254) 단원을 참조하십시오.

- **`Standard`** – 몇 시간 내에 보관된 객체 중 어느 것에든 액세스할 수 있습니다. 표준 검색은 보통 3-5시간 안에 완료됩니다. 이 값이 기본 계층입니다.
- **`Bulk`** – S3 Glacier에서 가장 낮은 비용의 데이터 액세스 옵션으로서 페타바이트 단위의 대용량 데이터도 저렴한 비용으로 하루 내에 가져올 수 있습니다. `Bulk`는 보통 5-12시간 안에 액세스할 수 있습니다.

`Expedited`, `Standard` 또는 `Bulk` 요청을 실행하려면, POST Object restore REST API 요청의 `Tier` 요청 요소를 사용자가 원하는 옵션 또는 AWS CLI 또는 AWS SDK의 동등한 옵션으로 설정합니다. `Expedited` 액세스의 경우 이를 온디맨드 또는 프로비저닝으로 지정할 필요가 없습니다. 프로비저닝된 용량을 구매하였다면, 사용자의 프로비저닝된 용량을 통해 모든 `Expedited` 검색이 자동으로 수행됩니다. 계층 요금에 대한 자세한 내용은 S3 Glacier 요금을 참조하십시오.

프로비저닝된 용량

프로비저닝된 용량을 통해 필요할 때 신속 검색에 대한 검색 용량이 보장됩니다. 각 용량 단위로 5분마다 신속 검색 3회를 수행할 수 있고, 최대 150MB/s의 검색 처리량이 제공됩니다.

워크로드에 몇 분 내로 데이터의 하위 집합에 대한 매우 안전하고 예측 가능한 액세스가 필요한 경우 프로비저닝된 검색 용량을 구매해야 합니다. 프로비저닝된 검색 용량이 없더라도 비정상적으로 수요가 높지 않은 경우를 제외하면 `Expedited` 검색은 허용됩니다. 하지만 모든 상황에서 `Expedited` 검색에 액세스해야 하는 경우 프로비저닝된 검색 용량을 구매해야 합니다. Amazon S3 콘솔, S3 Glacier 콘솔, 프로비저닝된 용량 구매 REST API, AWS SDK 또는 AWS CLI를 사용하여 프로비저닝된 용량을 구매할 수 있습니다. 프로비저닝된 용량의 요금에 대한 자세한 내용은 S3 Glacier 요금을 참조하십시오.

추가 정보

- Amazon Simple Storage Service API Reference의 POST Object restore
- Amazon Simple Storage Service 개발자 가이드의 Amazon S3 Select 및 S3 Glacier Select에 대한 SQL 참조

Amazon S3 분석 – 스토리지 클래스 분석

Amazon S3 분석 스토리지 클래스 분석을 이용하면 스토리지 액세스 패턴을 분석해 올바른 데이터를 올바른 스토리지 클래스로 옮길 시간을 결정할 수 있습니다. 이 새로운 Amazon S3 분석 기능은 데이터 액세스 패턴을 관찰해 자주 액세스하지 않는 STANDARD 스토리지를 STANDARD_IA (IA는 자주 액세스하지 않는다는 뜻입니다) 스토리지 클래스로 옮길 시점을 알려줍니다. 스토리지 클래스에 대한 자세한 정보는 Amazon S3 스토리지 클래스 (p. 107) 단원을 참조하십시오.

스토리지 클래스 분석 결과 필터링 데이터 세트가 일정 시간 동안 액세스 빈도가 떨어지는 패턴을 보인다면, 분석 결과를 이용해 수명 주기 정책을 개선할 수 있습니다. 버킷에 있는 모든 객체를 분석하도록 스토리지 클래스 분석을 구성할 수도 있습니다. 또는 분석을 위해 공통 접두사(이름이 공통 문자열로 시작하는 객체), 객체 태그 또는 접두사와 태그 모두를 기준으로 객체를 그룹화하도록 필터를 구성할 수도 있습니다. 스토리지 클래스 분석에서는 대체로 객체 그룹 기준 필터링이 가장 유익합니다.

> **Important**
>
> 스토리지 클래스 분석은 ONEZONE_IA 또는 S3 Glacier 스토리지 클래스로의 전환을 권장하지 않습니다.

버킷당 최대 1,000개에 달하는 스토리지 클래스 분석 필터를 설정하고, 필터 별로 개별 분석을 실시할 수 있습니다. 다중 필터 구성을 이용하면 특정 객체 그룹을 분석해 객체를 STANDARD_IA로 옮기는 수명 주기 정책을 개선할 수 있습니다.

스토리지 클래스 분석은 매일 업데이트되는 스토리지 사용 정보를 Amazon S3 콘솔에서 표시합니다. 또한 스토리지 사용 데이터를 매일 S3 버킷의 파일로 내보낼 수도 있습니다. 내보낸 사용 보고서 파일은 스프레드시트 애플리케이션에서 열거나, Amazon QuickSight 같은 비즈니스 정보 수집 도구에서 활용할 수 있습니다.

주제

- 스토리지 클래스 분석을 설정하려면 어떻게 해야 합니까? (p. 255)
- 스토리지 클래스 분석을 사용하려면 어떻게 해야 합니까? (p. 256)
- 스토리지 클래스 분석 데이터를 내보내려면 어떻게 해야 합니까? (p. 258)
- Amazon S3 Analytics REST APIs (p. 260)

스토리지 클래스 분석을 설정하려면 어떻게 해야 합니까?

스토리지 클래스 분석을 설정하려면 분석할 객체 데이터를 구성해야 합니다. 스토리지 클래스 분석을 구성하려면 다음을 수행해야 합니다.

- 버킷의 모든 콘텐츠 분석.

  버킷에 있는 모든 객체에 대한 분석을 받을 수 있습니다.
- 접두사와 태그를 기준으로 그룹화된 객체 분석.

접두사, 객체 태그 또는 접두사와 태그 조합을 기준으로 객체를 그룹화하는 필터를 구성할 수 있습니다. 구성한 필터 별로 별도의 분석을 받게 됩니다. 버킷에는 최대 1,000개의 필터를 구성할 수 있습니다.

- 분석 데이터 내보내기.

  버킷이나 필터의 스토리지 클래스 분석을 구성할 때, 분석 데이터를 매일 파일로 내보내도록 설정할 수 있습니다. 일일 분석은 파일에 추가되어, 구성한 필터의 기록 분석 로그를 구성하게 됩니다. 파일은 사용자가 선택한 대상에 매일 업데이트됩니다. 데이터 내보내기를 선택할 때, 파일을 쓸 대상 버킷과 옵션으로 대상 접두사를 지정할 수 있습니다.

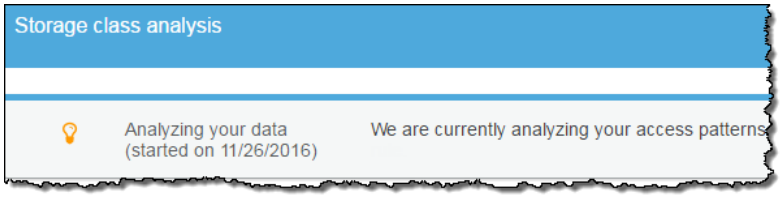
Amazon S3 콘솔, REST API나 AWS CLI 또는 AWS SDK를 이용해 스토리지 클래스 분석을 구성할 수 있습니다.

- Amazon S3 콘솔에서 스토리지 클래스 분석을 구성하는 방법에 대한 자세한 내용은 스토리지 클래스 분석을 구성하려면 어떻게 해야 합니까? 단원을 참조하십시오.
- Amazon S3 API를 사용하려면, AWS CLI나 AWS SDK에서 PutBucketAnalyticsConfiguration REST API 또는 이와 동등한 기능을 이용하십시오.

스토리지 클래스 분석을 사용하려면 어떻게 해야 합니까?

스토리지 클래스 분석을 이용하면 시간에 따른 데이터 액세스 패턴을 확인해 STANDARD_IA 스토리지의 수명 주기 관리를 개선할 수 있습니다 필터를 구성하고 나면, Amazon S3 콘솔에서 24~48시간 안에 필터를 적용한 데이터 분석을 확인할 수 있습니다. 하지만 스토리지 클래스 분석은 30일 이상으로 설정한, 필터링 데이터 세트의 액세스 패턴을 관찰해 분석 정보를 수집한 다음 결과를 도출합니다. 최초 결과가 도출되면 분석이 계속 진행되고 액세스 패턴이 변경되면 갱신됩니다.

필터를 구성하면, Amazon S3 콘솔에서 다음과 비슷한 형태의 메시지가 표시됩니다.



스토리지 클래스 분석은 30일 이상으로 설정한, 필터링 객체 데이터 세트의 액세스 패턴을 관찰해 분석에 필요한 충분한 정보를 수집합니다. 스토리지 클래스 분석이 충분한 정보를 수집하면, Amazon S3 콘솔에 다음과 유사한 메시지가 표시됩니다.

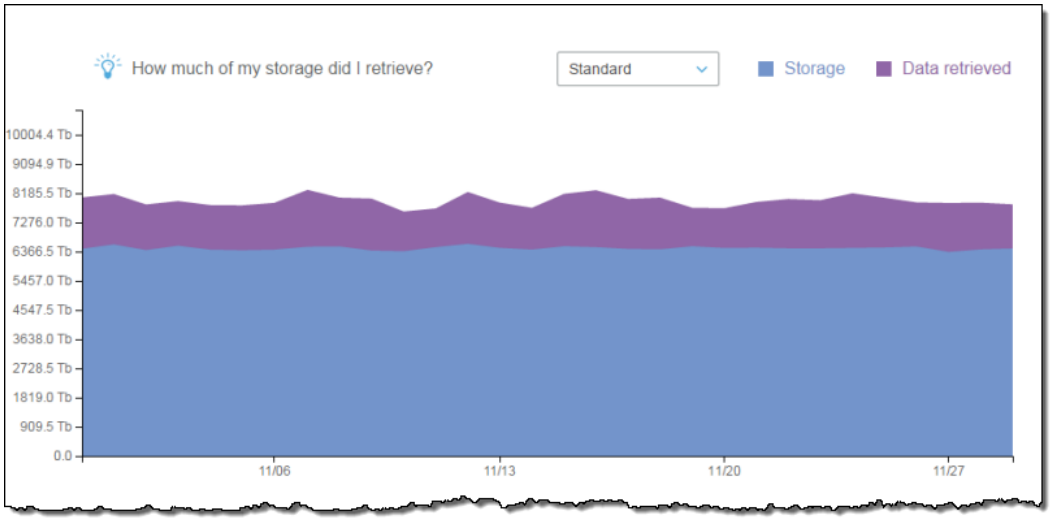
자주 액세스하지 않는 객체를 분석할 때, 스토리지 클래스는 Amazon S3에 업로드한 시점 이후의 연령을 바탕으로 그룹화한 필터링 객체 모음을 살펴봅니다. 스토리지 클래스 분석은 필터링 데이터 세트의 다음과 같은 요소를 확인해 연령 그룹이 자주 액세스하지 않는지를 판단합니다.

- STANDARD 스토리지 클래스에 있으며 128KB보다 큰 객체
- 연령 그룹당 평균 총 스토리지 개수.
- 연령 그룹당 전송된 평균 바이트(빈도 아님).

- 내보내기용 분석 데이터에는 스토리지 클래스 분석 관련 데이터가 포함된 요청만 포함됩니다. 이 때문에 요청 수 그리고 총 업로드 및 요청 바이트가 스토리지 측정치에 표시된 수치 또는 자체 내부 시스템에 의해 추적된 수치와 차이가 날 수 있습니다.
- 실패한 GET 및 PUT 요청은 분석에 포함되지 않습니다. 하지만 스토리지 측정치에는 실패한 요청이 표시됩니다.

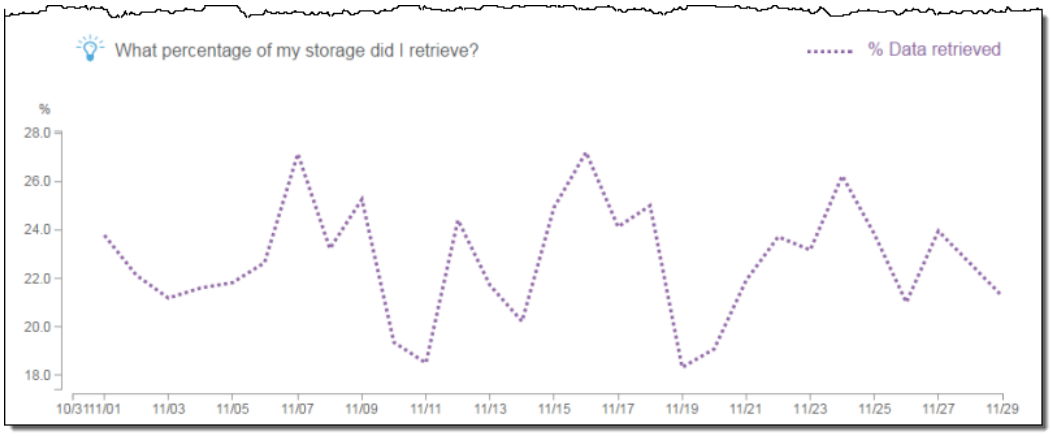
내 스토리지에서 검색한 내용이 얼마나 되는지 확인

Amazon S3 콘솔은 다음 예제에서처럼, 관찰 기간에 검색된 필터링 데이터 세트가 스토리지에 얼마나 있는지를 그래프로 표시합니다.



내 스토리지에서 검색한 내용의 비율을 어떻게 확인합니까?

Amazon S3 콘솔은 다음 예제에서처럼, 관찰 기간에 검색된 필터링 데이터 세트의 스토리지 내 비율도 그래프로 표시합니다.



이번 주제에서 앞서 언급했듯이, 자주 액세스하지 않는 객체를 분석할 때 스토리지 클래스는 Amazon S3에 업로드한 시점 이후의 연령을 바탕으로 그룹화한 필터링 객체 모음을 살펴봅니다. 스토리지 클래스 분석은 다음과 같은 사전 정의된 객체 연령 그룹을 이용합니다.

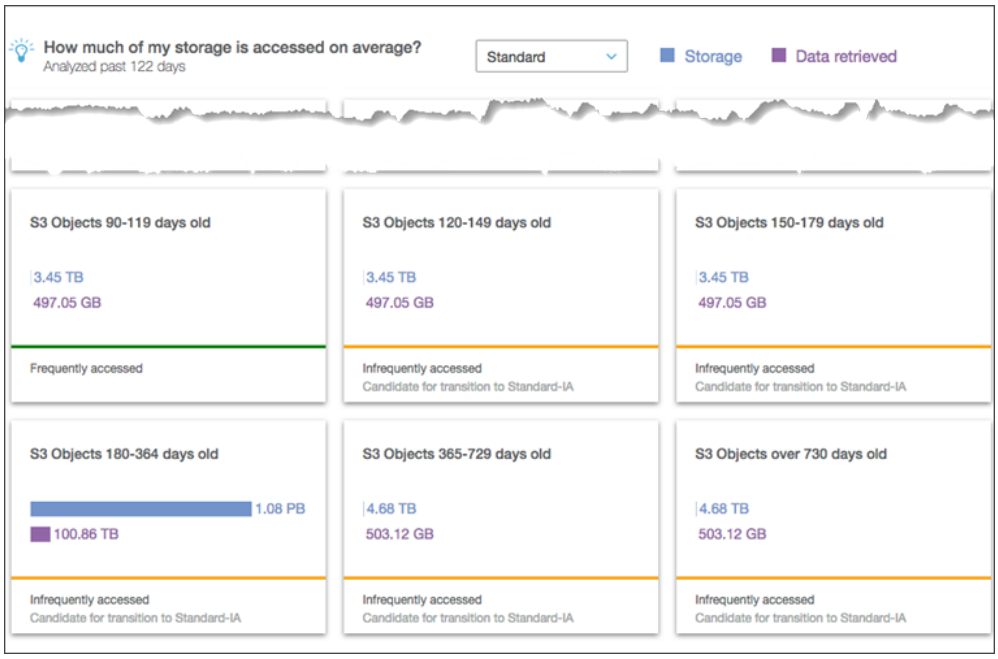
- Amazon S3 객체(15일 미만)
- Amazon S3 객체(15-29일)
- Amazon S3 객체(30-44일)

- Amazon S3 객체(45-59일)
- Amazon S3 객체(60-74일)
- Amazon S3 객체(75-89일)
- Amazon S3 객체(90-119일)
- Amazon S3 객체(120-149일)
- Amazon S3 객체(150-179일)
- Amazon S3 객체(180-364일)
- Amazon S3 객체(365-729일)
- Amazon S3 객체(730일 이상)

일반적으로 액세스 패턴을 관찰해 분석 결과를 도출하기에 충분한 정보를 모으는 데는 대략 30일이 걸립니다. 데이터의 고유 액세스 패턴에 따라 30일 넘게 걸릴 수도 있습니다. 하지만 필터를 구성하고 나면, Amazon S3 콘솔에서 24~48시간 안에 필터를 적용한 데이터 분석을 확인할 수 있습니다. Amazon S3 콘솔에서는 객체 연령 그룹을 바탕으로 구분된 객체 액세스에 대한 일일 분석을 확인할 수 있습니다.

내 스토리지의 액세스 빈도 확인

다음 예제에서처럼, Amazon S3 콘솔은 사전 정의된 객체 연령 그룹을 바탕으로 구분된 액세스 패턴을 보여줍니다.



수명 주기 생성 프로세스 중 자주 액세스함 또는 자주 액세스하지 않음 텍스트를 통해 시각적 도움을 받을 수 있습니다.

스토리지 클래스 분석 데이터를 내보내려면 어떻게 해야 합니까?

스토리지 클래스 분석이 분석 보고서를 CSV(쉼표로 구분된 값) 플랫 파일로 내보내도록 설정할 수 있습니다. 보고서는 매일 업데이트되며 사용자가 구성한 객체 연령 그룹 필터를 적용합니다. Amazon S3 콘솔을 사용

할 때는 필터 생성 시 보고서 내보내기 옵션을 선택할 수 있습니다. 데이터 내보내기를 선택할 때는, 파일을 쓸 대상 버킷과 옵션으로 대상 접두사를 지정할 수 있습니다. 데이터를 다른 계정에 있는 대상 버킷으로 내보낼 수도 있습니다. 대상 버킷은 분석을 구성하는 버킷과 같은 리전에 있어야 합니다.

AWS 계정이 소유한 버킷을 확인하고 객체를 정해진 위치에 있는 버킷에 쓸 수 있는 권한을 Amazon S3에 부여하는 버킷 정책을 대상 버킷 상에 생성해야 합니다. 정책에 대한 예는 Amazon S3 인벤토리 및 Amazon S3 분석에 권한 부여 (p. 414)을 참조하십시오.

스토리지 클래스 분석 보고서를 구성한 후 24시간이 지나면, 내보낸 보고서를 매일 받게 됩니다. 이후에는 Amazon S3이 모니터링을 계속 실시하고 매일 내보내기를 수행합니다.

CSV 파일은 스프레드시트 애플리케이션에서 열거나 Amazon QuickSight 같은 다른 애플리케이션에서 가져올 수 있습니다. Amazon QuickSight에서 Amazon S3 파일을 이용하는 방법은 Amazon QuickSight 사용 설명서의 Amazon S3 파일을 사용한 데이터 세트 생성 단원을 참조하십시오.

내보낸 파일에 있는 데이터는 다음 예제에서처럼 객체 연령 그룹의 날짜를 바탕으로 정렬됩니다. 스토리지 클래스가 STANDARD라면, 행에 `ObjectAgeForSIATransition`과 `RecommendedObjectAgeForSIATransition` 열의 데이터도 함께 표시됩니다.

Date	ConfigId	Filter	StorageClass	ObjectAge	ObjectCount	DataUploaded_MB	Storage_MB	DataRetrieved_MB	GetRequestCount	CumulativeAccessRatio	ObjectAgeForSIATransition	RecommendedObjectAgeForSIATransition
11/26/17	SalesMaterial	SalesMaterial	STANDARD	000-014	39376.428	3610.516	100409232	106131482	17724.24	1.056989281		
11/25/17	SalesMaterial	SalesMaterial	STANDARD	000-014	37904.412	3411.772	90017538.84	100602072	18068.4	1.11758301		
11/24/17	SalesMaterial	SalesMaterial	STANDARD	000-014	39744.432	3478.02	87888239.2	108298204.6	18584.64	1.232226355		
11/23/17	SalesMaterial	SalesMaterial	STANDARD	000-014	40480.44	3544.268	91903507.12	111984477.9	18928.8	1.218500593		
11/22/17	SalesMaterial	SalesMaterial	STANDARD	000-014	40112.436	3544.268	94405093.4	105851751.9	17724.24	1.121250433		
11/21/17	SalesMaterial	SalesMaterial	STANDARD	000-014	40480.44	3345.524	103598371.7	115342424	18412.56	1.113361359		
11/20/17	SalesMaterial	SalesMaterial	STANDARD	000-014	39008.424	3411.772	92668826.16	114416707.1	18756.72	1.234683893		
11/19/17	SalesMaterial	SalesMaterial	STANDARD	000-014	40480.44	3444.896	90254194.28	116396152.9	18068.4	1.28964813		
11/18/17	SalesMaterial	SalesMaterial	STANDARD	000-014	39744.432	3444.896	88724282.88	110218385.6	17724.24	1.242257271		
11/17/17	SalesMaterial	SalesMaterial	STANDARD	000-014	37904.412	3444.896	89554277.32	102845839	18584.64	1.148419061		
11/16/17	SalesMaterial	SalesMaterial	STANDARD	000-014	40112.436	3444.896	91362283.92	116521794.3	18928.8	1.275381802		
11/15/17	SalesMaterial	SalesMaterial	STANDARD	000-014	40112.436	3378.648	87730610.56	113237336.7	18756.72	1.290739184		
11/14/17	SalesMaterial	SalesMaterial	STANDARD	000-014	39744.432	3478.02	96131832.8	110526562.8	17896.32	1.149739473		

11/25/17	SalesMaterial	SalesMaterial	STANDARD	015-029	56856.618	5117.658	135026308.3	150903108.1	27102.6	1.11758301
11/24/17	SalesMaterial	SalesMaterial	STANDARD	015-029	59616.648	5217.03	131832358.8	162447307	27876.96	1.232226355
11/23/17	SalesMaterial	SalesMaterial	STANDARD	015-029	60720.66	5316.402	137855260.7	167976716.9	28393.2	1.218500593
11/22/17	SalesMaterial	SalesMaterial	STANDARD	015-029	60168.654	5316.402	141607640.1	158777627.8	26586.36	1.121250433
11/21/17	SalesMaterial	SalesMaterial	STANDARD	015-029	60720.66	5018.286	155397557.6	173013635.9	27618.84	1.113361359
11/20/17	SalesMaterial	SalesMaterial	STANDARD	015-029	58512.636	5117.658	139003239.2	171625060.6	28135.08	1.234683893
11/19/17	SalesMaterial	SalesMaterial	STANDARD	015-029	60720.66	5167.344	135381291.4	174594229.3	27102.6	1.28964813
11/18/17	SalesMaterial	SalesMaterial	STANDARD	015-029	59616.648	5167.344	133086424.3	165327578.3	26586.36	1.242257271
11/17/17	SalesMaterial	SalesMaterial	STANDARD	015-029	56856.618	5167.344	134331416	154268758.5	27876.96	1.148419061
11/16/17	SalesMaterial	SalesMaterial	STANDARD	015-029	60168.654	5167.344	137043425.9	174782691.5	28393.2	1.275381802
11/15/17	SalesMaterial	SalesMaterial	STANDARD	015-029	60168.654	5067.972	131595915.8	169856005	28135.08	1.290739184
11/14/17	SalesMaterial	SalesMaterial	STANDARD	015-029	59616.648	5217.03	144197749.2	165789844.1	26844.48	1.149739473

보고서 끝에 있는 객체 연령 그룹은 ALL로 제공됩니다. ALL 행에는 128KB보다 작은 객체를 포함하여 해당 날짜의 모든 연령 그룹에 대한 누적 합계가 포함됩니다.

11/25/17	SalesMaterial	SalesMaterial	STANDARD	ALL	777408.45	69229.16	3237567090	2066259214	363519	0.034	090-119	090-119
11/24/17	SalesMaterial	SalesMaterial	STANDARD	ALL	787528.56	69726.02	3222158279	2031486122	362228.4	0.034	090-119	090-119
11/23/17	SalesMaterial	SalesMaterial	STANDARD	ALL	775568.43	69643.21	3224228797	2143607314	365670	0.034	090-119	090-119
11/22/17	SalesMaterial	SalesMaterial	STANDARD	ALL	785688.54	70471.31	3268098456	2093598765	357926.4	0.034	090-119	090-119
11/21/17	SalesMaterial	SalesMaterial	STANDARD	ALL	772808.4	68069.82	3260462574	2152402776	366960.6	0.034	090-119	090-119
11/20/17	SalesMaterial	SalesMaterial	STANDARD	ALL	775568.43	69311.97	3235392872	2135278200	363088.8	0.034	090-119	090-119
11/19/17	SalesMaterial	SalesMaterial	STANDARD	ALL	793048.62	68566.68	3191517795	2156592855	361368	0.034	090-119	090-119
11/18/17	SalesMaterial	SalesMaterial	STANDARD	ALL	784768.53	69808.83	3241761620	2158880455	357926.4	0.034	090-119	090-119
11/17/17	SalesMaterial	SalesMaterial	STANDARD	ALL	782008.5	71050.98	3143115658	2130892653	372983.4	0.034	090-119	090-119
11/16/17	SalesMaterial	SalesMaterial	STANDARD	ALL	773728.41	70636.93	3237654725	2145622446	369111.6	0.034	090-119	090-119
11/15/17	SalesMaterial	SalesMaterial	STANDARD	ALL	773728.41	69477.59	3131717743	2091869264	369541.8	0.034	090-119	090-119
11/14/17	SalesMaterial	SalesMaterial	STANDARD	ALL	789368.58	70388.5	3169870122	2171336445	358356.6	0.034	090-119	090-119
11/13/17	SalesMaterial	SalesMaterial	STANDARD	ALL	767288.34	69643.21	3199668074	2128513319	367390.8	0.034	090-119	090-119
11/12/17	SalesMaterial	SalesMaterial	STANDARD	ALL	767288.34	68649.49	3135649104	2041228760	358356.6	0.034	090-119	090-119
11/11/17	SalesMaterial	SalesMaterial	STANDARD	ALL	774648.42	68897.92	3132710428	2154202987	356635.8	0.034	090-119	090-119
11/10/17	SalesMaterial	SalesMaterial	STANDARD	ALL	774648.42	69311.97	3181716041	2175053920	369541.8	0.034	090-119	090-119
11/9/17	SalesMaterial	SalesMaterial	STANDARD	ALL	771888.39	69643.21	3118609765	2197544222	364809.6	0.034	090-119	090-119
11/8/17	SalesMaterial	SalesMaterial	STANDARD	ALL	781088.49	70885.36	3136378996	2162511633	360507.6	0.034	090-119	090-119
11/7/17	SalesMaterial	SalesMaterial	STANDARD	ALL	783848.52	68649.49	3166312643	2019414755	355345.2	0.034	090-119	090-119
11/6/17	SalesMaterial	SalesMaterial	STANDARD	ALL	771888.39	69146.35	3087018554	2210161640	363519	0.034	090-119	090-119
11/5/17	SalesMaterial	SalesMaterial	STANDARD	ALL	770968.38	68235.44	3098188075	2177492429	360077.4	0.034	090-119	090-119
11/4/17	SalesMaterial	SalesMaterial	STANDARD	ALL	783848.52	69974.45	3115637244	2208645234	367390.8	0.034	090-119	090-119
11/3/17	SalesMaterial	SalesMaterial	STANDARD	ALL	770048.37	69311.97	3125766832	2195447440	354054.6	0.034	090-119	090-119

다음 단원에서는 보고서에서 사용한 행을 설명합니다.

내보낸 파일 레이아웃

다음 표는 내보낸 파일의 레이아웃을 설명합니다.

Amazon S3 Analytics REST APIs

다음은 스토리지 인벤토리에 사용되는 REST 작업입니다.

- DELETE Bucket 분석 구성
- GET Bucket 분석 구성
- List Bucket 분석 구성
- PUT Bucket 분석 구성

Amazon S3 보안

AWS에서 클라우드 보안을 가장 중요하게 생각합니다. AWS 고객은 보안에 가장 보안에 민감한 조직의 요구 사항에 부합하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 AWS와 귀하의 공동 책임입니다. 책임 분담 모델은 이 사항을 클라우드 내 보안 및 클라우드의 보안으로 설명합니다.

- 클라우드의 보안 – AWS는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다. 또한 AWS는 또한 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사원은 정기적으로 AWS 규정 준수 프로그램의 일환으로 보안 효과를 테스트하고 검증합니다. Amazon S3에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 제공 범위 내 AWS 제품을 참조하십시오.
- 클라우드 내 보안 – 귀하의 책임은 귀하가 사용하는 AWS 서비스에 의해 결정됩니다. 또한 데이터의 민감도, 조직의 요구 사항 및 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 Amazon S3 사용 시 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 항목에서는 보안 및 규정 준수 목표를 충족하도록 Amazon S3를 구성하는 방법을 보여줍니다. 또한 Amazon S3 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 제품의 사용 방법을 배우게 됩니다.

주제

- Amazon S3의 데이터 보호 (p. 261)
- Amazon S3의 Identity and Access Management (p. 299)
- Amazon S3의 로깅 및 모니터링 (p. 458)
- Amazon S3 규정 준수 확인 (p. 459)
- Amazon S3의 복원성 (p. 467)
- Amazon S3의 인프라 보안 (p. 496)
- Amazon S3의 구성 및 취약성 분석 (p. 497)
- Amazon S3 보안 모범 사례 (p. 497)

Amazon S3의 데이터 보호

Amazon S3에서는 미션 크리티컬 및 기본 데이터 스토리지에 적합하게 설계된, 내구성이 뛰어난 스토리지 인프라를 제공합니다. 객체는 Amazon S3 리전의 여러 시설에 걸쳐 여러 디바이스에 중복해서 저장됩니다. 향상된 데이터 내구성을 보장하기 위해 Amazon S3 `PUT` 및 `PUT Object copy` 작업으로 여러 시설에 데이터를 동기식으로 저장합니다. 객체가 저장되면 Amazon S3에서는 손실된 중복성을 빠르게 검색 및 복원하여 객체의 내구성을 유지합니다.

Amazon S3 표준 스토리지는 다음 기능을 제공합니다.

- Amazon S3 서비스 수준 계약으로 신뢰성 보장
- 지정된 기간 동안 객체에 대해 99.999999999%의 내구성과 99.99%의 가용성을 제공할 수 있도록 설계됨
- 두 시설의 데이터가 동시에 손실될 경우에도 작동을 유지할 수 있도록 설계됨

Amazon S3에서는 버전 관리를 사용하여 데이터에 대한 향상된 보호를 제공합니다. 버전 관리를 사용하여 Amazon S3 버킷에 저장된 모든 버전의 객체를 모두 보존, 검색 및 복원할 수 있습니다. 또한 의도치 않은 사용자 작업 및 애플리케이션 장애로부터 쉽게 복구할 수 있습니다. 기본적으로 요청을 통해 가장 최근에 작성된 버전을 검색합니다. 요청에서 객체의 버전을 지정하여 객체의 기존 버전을 가져올 수 있습니다.

다음 보안 모범 사례에서도 Amazon S3의 데이터 보호를 다룹니다.

- 서버 측 암호화 구현
- 전송 중 데이터의 암호화 적용
- Amazon S3와 함께 Amazon Macie 사용 고려
- 모든 Amazon S3 버킷 식별 및 감사
- AWS 보안 권고 사항 모니터링

인터네트워크 트래픽 개인 정보

이 주제에서는 Amazon S3이 해당 서비스에서 다른 위치로 향하는 연결을 보호하는 방법을 설명합니다.

서비스와 온프레미스 클라이언트 및 애플리케이션 간의 트래픽

프라이빗 네트워크와 AWS 사이에 두 연결 옵션이 있습니다.

- AWS Site-to-Site VPN 연결. 자세한 내용은 AWS Site-to-Site VPN이란 무엇입니까?를 참조하십시오.
- AWS Direct Connect 연결. 자세한 내용은 AWS Direct Connect란 무엇입니까?를 참조하십시오.

네트워크를 통한 Amazon S3 액세스는 AWS에서 게시한 API를 통해 이루어집니다. 클라이언트가 TLS(전송 계층 보안) 1.0을 지원해야 합니다. TLS 1.2 이상을 권장합니다. 클라이언트는 DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Diffie-Hellman Ephemeral)와 같은 PFS(전달 완전 보안)가 포함된 암호 제품군도 지원해야 합니다. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다. 또한 액세스 키 ID와 IAM 보안 주체와 연결된 보안 액세스 키를 사용하여 요청에 서명하거나 AWS Security Token Service(STS)를 사용해 임시 보안 자격 증명을 생성하여 요청에 서명할 수 있습니다.

같은 리전에 있는 AWS 리소스 사이의 트래픽

Amazon S3의 Amazon Virtual Private Cloud(Amazon VPC) 엔드포인트는 Amazon S3에만 연결을 허용하는 VPC 내 논리적 개체입니다. Amazon VPC는 Amazon S3으로 요청을 라우팅하고 응답을 다시 VPC로 라우팅합니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 참조하십시오. VPC 엔드포인트에서 S3 버킷 액세스를 제어하는 데 사용할 수 있는 예제 버킷 정책은 Amazon S3의 VPC 엔드포인트에 대한 예제 버킷 정책 (p. 414) 단원을 참조하십시오.

암호화를 사용하여 데이터 보호

데이터 보호란 전송 중(Amazon S3 안팎으로 데이터가 이동 중) 및 유휴(Amazon S3 데이터 센터의 디스크에 데이터가 저장된 동안) 데이터를 보호하는 것을 말합니다. Secure Sockets Layer(SSL) 또는 클라이언트 측 암호화를 사용하여 전송 중인 데이터를 보호할 수 있습니다. Amazon S3에서 유휴 데이터를 보호하는 다음과 같은 옵션이 있습니다.

- 서버 측 암호화 – 데이터 센터의 디스크에 저장하기 전에 객체를 암호화하고 객체를 다운로드할 때 이를 해독하도록 Amazon S3에 요청합니다.
- 클라이언트 측 암호화 – 클라이언트 측 데이터를 암호화하여 암호화된 데이터를 Amazon S3에 업로드합니다. 이 경우 사용자가 암호화 프로세스, 암호화 키 및 관련 도구를 관리합니다.

서버 측 암호화 및 클라이언트 측 암호화에 대한 자세한 내용은 아래 나열된 주제를 검토하십시오.

주제

- 서버 측 암호화를 사용하여 데이터 보호 (p. 263)
- 클라이언트측 암호화를 사용하여 데이터 보호 (p. 291)

서버 측 암호화를 사용하여 데이터 보호

서버 측 암호화는 데이터를 받는 애플리케이션 또는 서비스에 의해 해당 대상에서 데이터를 암호화하는 것입니다. Amazon S3에서는 데이터 센터의 디스크에 데이터를 쓸 때 객체 레벨에서 데이터를 암호화하고 해당 데이터에 액세스할 때 자동으로 암호 해독합니다. 요청을 인증하기만 하면 액세스 권한을 갖게 되며, 객체의 암호화 여부와 관계없이 액세스 방식에는 차이가 없습니다. 예를 들어, 미리 서명된 URL을 사용하여 객체를 공유하는 경우, 해당 URL은 암호화된 객체와 암호화되지 않은 객체에 동일하게 작동합니다. 또한 버킷에 객체를 나열하는 경우 목록 API에서는 암호화 여부와 관계없이 전체 객체의 목록을 반환합니다.

> **Note**
>
> 동일한 객체에 서로 다른 서버 측 암호화 유형을 동시에 적용할 수는 없습니다.

암호화 키 관리 방법으로 무엇을 선택하느냐에 따라 다음과 같은 세 가지 옵션을 독립적으로 사용할 수 있습니다.

Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)

Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)를 사용하면 각 객체는 고유한 키로 암호화됩니다. 또한 추가 보안 조치로 주기적으로 바뀌는 마스터 키를 사용하여 키 자체를 암호화합니다. Amazon S3 서버 측 암호화는 가장 강력한 블록 암호 중 하나인 256비트 고급 암호화 표준(AES-256)을 사용하여 데이터를 암호화합니다. 자세한 내용은 Amazon S3가 관리하는 암호화 키(SSE-S3)를 사용하는 서버 측 암호화로 데이터 보호 (p. 269) 단원을 참조하십시오.

AWS Key Management Service에 저장된 고객 마스터 키(CMK)를 사용한 서버 측 암호화(SSE-KMS)

AWS Key Management Service에 저장된 고객 마스터 키(CMK)를 사용한 서버 측 암호화(SSE-KMS)는 SSE-S3와 유사하지만 이 서비스 사용 시 몇 가지 추가적인 이점이 있고 비용이 발생합니다. Amazon S3의 객체에 대한 무단 액세스에 대응하여 추가적인 보호를 제공하는 CMK를 사용하려면 별도의 권한이 필요합니다. SSE-KMS도 CMK가 사용된 때와 사용 주체를 표시하는 감사 추적 기능을 제공합니다. 또한 고객 관리형 CMK를 생성하고 관리하거나 사용자, 서비스 및 리전에 고유한 AWS 관리형 CMK를 사용할 수 있습니다. 자세한 내용은 AWS Key Management Service에 저장된 CMK를 사용한 서버 측 암호화(SSE-KMS)로 데이터 보호 (p. 263) 단원을 참조하십시오.

고객 제공 키를 사용한 서버 측 암호화(SSE-C)

고객 제공 키를 사용한 서버 측 암호화(SSE-C)를 사용하면 사용자는 암호화 키를 관리하고 Amazon S3는 암호화(디스크에 쓸 때) 및 해독(객체에 액세스할 때)을 관리합니다. 자세한 내용은 고객 제공 암호화 키(SSE-C)로 서버 측 암호화를 사용하여 데이터 보호 (p. 278) 단원을 참조하십시오.

AWS Key Management Service에 저장된 CMK를 사용한 서버 측 암호화(SSE-KMS)로 데이터 보호

서버 측 암호화는 데이터를 받는 애플리케이션 또는 서비스에 의해 해당 대상에서 데이터를 암호화하는 것입니다. AWS Key Management Service(AWS KMS)는 클라우드에 맞게 확장된 키 관리 시스템을 제공하기 위해 안전하고 가용성이 높은 하드웨어 및 소프트웨어를 결합하는 서비스입니다. Amazon S3는 AWS KMS 고객 마스터 키(CMK)를 사용하여 Amazon S3 객체를 암호화합니다. AWS KMS는 객체 데이터만을 암호화합니다. 객체 메타데이터는 암호화되지 않습니다.

CMK를 사용하는 경우 AWS Management 콘솔 또는 AWS KMS API를 통해 AWS KMS를 사용하여 중앙 집중식으로 CMK를 생성하고, CMK가 사용되는 방식을 제어하는 정책을 정의하고, CMK 사용을 감사하여 CMK가 올바로 사용되는지 감사합니다. 이러한 CMK를 사용하여 Amazon S3 버킷에서 데이터를 보호할 수 있습니다. S3 버킷에 SSE-KMS 암호화를 사용할 때 AWS KMS CMK는 버킷과 동일한 리전에 있어야 합니다.

AWS KMS CMK를 사용하면 추가 요금이 부과됩니다. 자세한 내용은 AWS Key Management Service Developer Guide의 AWS Key Management Service 개념 - 고객 마스터 키(CMK) 및 AWS Key Management Service 요금을 참조하십시오.

> **Note**
>
> `kms:ReEncrypt`, `kms:GenerateDataKey` 및 `kms:DescribeKey` 권한 외에도 AWS KMS CMK로 암호화된 Amazon S3 객체를 업로드하거나 다운로드할 때 `kms:Decrypt` 권한이 필요합니다. 자세한 내용은 AWS KMS CMK를 사용하는 암호화로 Amazon S3에 대형 파일 업로드 실패를 참조하십시오.

AWS 관리형 CMK 및 고객 관리형 CMK

AWS KMS에서 서버 측 암호화(SSE-KMS)를 사용하는 경우 기본 AWS 관리형 CMK를 사용하거나 이미 생성한 고객 관리형 CMK를 지정할 수 있습니다.

고객 관리형 CMK를 지정하지 않으면 Amazon S3에서는 SSE-KMS로 암호화된 객체를 버킷에 처음 추가할 때 AWS 계정에 AWS 관리형 CMK를 자동으로 생성합니다. 기본적으로 Amazon S3에서는 SSE-KMS에 이 CMK를 사용합니다.

SSE-KMS에 고객 관리형 CMK를 사용하려는 경우 SSE-KMS를 구성하기 전에 CMK를 생성할 수 있습니다. 그런 다음 버킷에 대해 SSE-KMS를 구성할 때 기존 고객 관리형 CMK를 지정할 수 있습니다.

고객 관리형 CMK를 직접 생성하면 CMK를 보다 유연하게 제어할 수 있습니다. 예를 들어, 고객 관리형 CMK를 생성, 교체 및 비활성화할 수 있습니다. 또한 액세스 제어를 정의하고 데이터를 보호하는 데 사용하는 고객 관리형 CMK를 감사할 수 있습니다. 고객 관리형 및 AWS 관리형 CMK에 대한 자세한 내용은 AWS Key Management Service Developer Guide의 AWS KMS 개념을 참조하십시오.

> **Important**
>
> Amazon S3에서 서버 측 암호화에 AWS KMS CMK를 사용하는 경우 대칭 CMK를 선택해야 합니다. Amazon S3는 대칭 CMK만 지원하고 비대칭 CMK는 지원하지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 대칭 및 비대칭 키 사용을 참조하십시오.

AWS 서명 버전 4

SSE-KMS로 암호화된 객체를 업로드 또는 액세스하는 경우 강화된 보안을 위해 AWS 서명 버전 4를 사용해야 합니다. AWS SDK를 사용하여 이 작업을 수행하는 방법에 대한 자세한 내용은 요청 인증에서 서명 버전 지정을 참조하십시오.

> **Important**
>
> AWS KMS로 보호되는 객체에 대한 모든 GET 및 PUT 요청은 SSL을 통해, 또는 SigV4를 사용하여 요청되지 않을 경우 실패합니다.

SSE-KMS 주요 내용

SSE-KMS의 주요 내용은 다음과 같습니다.

- 사용자가 직접 생성하고 관리하는 고객 관리형 CMK를 선택하거나 Amazon S3가 사용자의 AWS 계정에서 생성하고 관리하는 AWS 관리형 CMK를 선택할 수 있습니다. 고객 관리형 CMK와 같이 사용자의 AWS 관리형 CMK는 사용자의 AWS 계정 및 리전에 고유합니다. Amazon S3만 사용자 대신 이 CMK를 사용할 수 있는 권한이 있습니다. Amazon S3는 대칭 CMK 만 지원합니다.
- AWS KMS 콘솔에서 감사 가능한 고객 관리형 CMK를 생성, 교체 및 비활성화할 수 있습니다.
- 응답의 ETag는 객체 데이터의 MD5가 아닙니다.
- 데이터를 암호화하는 데 사용되는 데이터 키도 암호화되어 보호되는 데이터와 함께 저장됩니다.
- AWS KMS 내 보안 관리가 암호화 관련 규정 준수 요구 사항을 충족하는 데 도움이 될 수 있습니다.

서버 측 암호화 필요

특정 Amazon S3 버킷의 모든 객체에 대한 서버 측 암호화를 요구하려면 정책을 사용하면 됩니다. 예를 들어, 다음 버킷 정책은 요청에 SSE-KMS를 사용한 서버 측 암호화를 요청하는 `s3:PutObject` 헤더가 포함되지 않을 경우 모든 사용자에게 객체 업로드(`x-amz-server-side-encryption`) 권한을 거부합니다.

```
{
   "Version":"2012-10-17",
   "Id":"PutObjPolicy"
   "Statement":[{
         "Sid":"DenyUnEncryptedObjectUploads",
         "Effect":"Deny",
         "Principal":"*",
         "Action":"s3:PutObject",
         "Resource":"arn:aws:s3:::YourBucket/*",
         "Condition":{
            "StringNotEquals":{
               "s3:x-amz-server-side-encryption":"aws:kms"
            }
         }
      }
   ]
}
```

버킷의 객체를 암호화하는 데 특정 AWS KMS CMK를 사용하도록 하려면 `s3:x-amz-server-side-encryption-aws-kms-key-id` 조건 키를 사용합니다. AWS KMS CMK를 지정하려면 "`arn:aws:kms:region:acct-id:key/key-id`" 형식의 키 Amazon 리소스 이름(ARN)을 사용해야 합니다.

> **Note**
>
> 객체를 업로드할 때 `x-amz-server-side-encryption-aws-kms-key-id` 헤더를 사용하여 AWS KMS CMK를 지정할 수 있습니다. 요청에 헤더가 없는 경우 Amazon S3은 AWS 관리형 CMK를 사용합니다. 이와 관계없이 Amazon S3이 객체 암호화에 사용하는 AWS KMS 키 ID는 정책의 AWS KMS 키 ID와 일치해야 합니다. 그렇지 않으면 Amazon S3이 요청을 거부합니다.

Amazon S3 전용 조건 키의 전체 목록과 조건 키 지정에 대한 자세한 내용은 Amazon S3 조건 키 (p. 345) 단원을 참조하십시오.

Amazon S3 콘솔에서 AWS Key Management Service 사용

AWS KMS에 저장된 CMK와 함께 Amazon S3 콘솔을 사용하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 객체에 암호화를 추가하려면 어떻게 합니까?를 참조하십시오.

Amazon S3에서 AWS Key Management Service를 위한 API 지원

객체 생성 REST API에서 SSE-KMS를 요청하려면 `x-amz-server-side-encryption` 요청 헤더를 사용합니다. 객체에 사용된 AWS KMS CMK의 ID를 지정하려면 `x-amz-server-side-encryption-aws-kms-key-id`를 사용합니다. Amazon S3 API는 `x-amz-server-side-encryption-context` 헤더가 있는 암호화 컨텍스트도 지원합니다. 자세한 내용은 REST API를 사용하여 Amazon S3에서 AWS Key Management Service 지정 (p. 268) 단원을 참조하십시오.

AWS SDK는 Amazon S3에서 SSE-KMS를 요청할 수 있도록 래퍼 API도 제공합니다. 자세한 내용은 AWS SDK를 사용하여 Amazon S3에서 AWS Key Management Service 지정 (p. 265) 단원을 참조하십시오.

AWS SDK를 사용하여 Amazon S3에서 AWS Key Management Service 지정

AWS SDK를 사용할 때 Amazon S3에서 AWS Key Management Service(AWS KMS) 고객 마스터 키(CMK)를 사용하도록 요청할 수 있습니다. 이 단원에서는 Java 및 .NET용 AWS SDK를 사용한 예제를 제공합니다. 다른 SDK에 대한 자세한 내용은 Sample Code and Libraries 단원을 참조하십시오.

> **Important**
>
> Amazon S3에서 서버 측 암호화에 AWS KMS CMK를 사용하는 경우 대칭 CMK를 선택해야 합니다. Amazon S3는 대칭 CMK만 지원하고 비대칭 CMK는 지원하지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 대칭 및 비대칭 키 사용을 참조하십시오.

Java용 AWS SDK

이 단원에서는 AWS SDK for Java를 사용하는 다양한 Amazon S3 작업 및 AWS KMS CMK를 사용하는 방법을 설명합니다.

PUT 작업

Java용 AWS SDK를 사용하여 객체를 업로드할 때 다음 요청에 표시된 대로 `SSEAwsKeyManagementParams` 속성을 추가하여 Amazon S3에서 AWS KMS CMK를 사용하도록 요청할 수 있습니다.

```
PutObjectRequest putRequest = new PutObjectRequest(bucketName,
   keyName, file).withSSEAwsKeyManagementParams(new SSEAwsKeyManagementParams());
```

이 경우 Amazon S3는 AWS 관리형 CMK를 사용합니다(AWS KMS에 저장된 CMK를 사용한 서버 측 암호화 사용 (p. 263) 참조). 필요한 경우 고객 관리형 CMK를 생성하고 요청에서 지정할 수 있습니다.

```
PutObjectRequest putRequest = new PutObjectRequest(bucketName,
   keyName, file).withSSEAwsKeyManagementParams(new SSEAwsKeyManagementParams(keyID));
```

고객 관리형 CMK를 생성하는 방법에 대한 자세한 내용은 AWS Key Management Service Developer Guide의 AWS KMS API 프로그래밍을 참조하십시오.

객체를 업로드하는 사용 가능한 코드 예제는 다음 주제를 참조하십시오. 코드 예제를 업데이트하고 이전 코드 조각에서 표시된 바와 같이 암호화 정보를 제공해야 합니다.

- 단일 작업으로 객체를 업로드하려면 AWS SDK for Java를 사용하여 객체 업로드 (p. 173) 단원을 참조하십시오.
- 멀티파트 업로드는 다음 주제를 참조하십시오.
  - 상위 수준 멀티파트 업로드 API 사용은 파일 업로드 (p. 184) 단원을 참조하십시오.
  - 하위 수준 멀티파트 업로드 API를 사용하는 경우 파일 업로드 (p. 187) 단원을 참조하십시오.

Copy 작업

객체를 복사할 때 동일한 요청 속성(`ServerSideEncryptionMethod` 및 `ServerSideEncryptionKeyManagementServiceKeyId`)을 추가하여 Amazon S3에서 AWS KMS CMK를 사용하도록 요청합니다. 객체 복사에 대한 자세한 내용은 객체 복사 (p. 209) 단원을 참조하십시오.

미리 서명된 URL

AWS KMS CMK를 사용하여 암호화된 객체에 대해 미리 서명된 URL을 생성할 때 서명 버전 4를 명시적으로 지정해야 합니다.

```
ClientConfiguration clientConfiguration = new ClientConfiguration();
clientConfiguration.setSignerOverride("AWSS3V4SignerType");
AmazonS3Client s3client = new AmazonS3Client(
        new ProfileCredentialsProvider(), clientConfiguration);
...
```

코드에 대한 예는 AWS SDK for Java를 사용하여 미리 서명된 객체 URL 생성 (p. 170) 단원을 참조하십시오.

.NET용 AWS SDK

이 단원에서는 .NET용 AWS SDK를 사용하는 다양한 Amazon S3 작업 및 AWS KMS CMK를 사용하는 방법을 설명합니다.

PUT 작업

.NET용 AWS SDK를 사용하여 객체를 업로드할 때 다음 요청에 표시된 대로 `ServerSideEncryptionMethod` 속성을 추가하여 Amazon S3에서 AWS KMS CMK를 사용하도록 요청할 수 있습니다.

```
PutObjectRequest putRequest = new PutObjectRequest
   {
       BucketName = bucketName,
       Key = keyName,
       // other properties.
       ServerSideEncryptionMethod = ServerSideEncryptionMethod.AWSKMS
   };
```

이 경우 Amazon S3은 AWS 관리형 CMK를 사용합니다. 자세한 내용은 AWS Key Management Service에 저장된 CMK를 사용한 서버 측 암호화(SSE-KMS)로 데이터 보호 (p. 263) 단원을 참조하십시오. 필요한 경우 자체 대칭 고객 관리형 CMK를 생성하고 요청에서 지정할 수 있습니다.

```
PutObjectRequest putRequest1 = new PutObjectRequest
{
    BucketName = bucketName,
    Key = keyName,
    // other properties.
    ServerSideEncryptionMethod = ServerSideEncryptionMethod.AWSKMS,
    ServerSideEncryptionKeyManagementServiceKeyId = keyId
};
```

고객 관리형 CMK를 생성하는 방법에 대한 자세한 내용은 AWS Key Management Service Developer Guide의 AWS KMS API 프로그래밍을 참조하십시오.

객체를 업로드하는 사용 가능한 코드 예제는 다음 주제를 참조하십시오. 코드 예제를 업데이트하고 이전 코드 조각에서 표시된 바와 같이 암호화 정보를 제공해야 합니다.

- 단일 작업으로 객체를 업로드하려면 .NET용 AWS SDK를 사용하여 객체 업로드 (p. 174) 단원을 참조하십시오.
- 멀티파트 업로드는 다음 주제를 참조하십시오.
  - 상위 수준 멀티파트 업로드 API 사용은 .NET용 AWS SDK(상위 수준 API)를 사용하여 S3 버킷에 파일 업로드 (p. 191) 단원을 참조하십시오.
  - 하위 수준 멀티파트 업로드 API 사용은 .NET용 AWS SDK(하위 수준 API)를 사용하여 S3 버킷에 파일 업로드 (p. 197) 단원을 참조하십시오.

Copy 작업

객체를 복사할 때 동일한 요청 속성(`ServerSideEncryptionMethod` 및 `ServerSideEncryptionKeyManagementServiceKeyId`)을 추가하여 Amazon S3에서 AWS KMS CMK를 사용하도록 요청합니다. 객체 복사에 대한 자세한 내용은 객체 복사 (p. 209) 단원을 참조하십시오.

미리 서명된 URL

AWS KMS CMK를 사용하여 암호화된 객체에 대해 미리 서명된 URL을 생성할 때 서명 버전 4를 명시적으로 지정해야 합니다.

```
AWSConfigs.S3Config.UseSignatureVersion4 = true;
```

코드에 대한 예는 .NET용 AWS SDK를 사용하여 미리 서명된 객체 URL 생성 (p. 171) 단원을 참조하십시오.

REST API를 사용하여 Amazon S3에서 AWS Key Management Service 지정

객체를 만들 때—즉 새 객체를 업로드하거나 기존 객체를 복사할 때—AWS Key Management Service(AWS KMS) 고객 마스터 키(CMK)를 사용하여 서버 측 암호화를 사용하여 데이터를 암호화할 수 있습니다. 이렇게 하려면 요청에 `x-amz-server-side-encryption` 헤더를 추가합니다. 헤더의 값을 암호화 알고리즘 `aws:kms`로 설정합니다. Amazon S3는 `x-amz-server-side-encryption` 응답 헤더를 반환하여 객체가 SSE-KMS를 사용하여 저장됨을 확인해 줍니다.

`aws:kms`의 값을 사용하여 `x-amz-server-side-encryption` 헤더를 지정하는 경우 다음 요청 헤더를 사용할 수도 있습니다.

- `x-amz-server-side-encryption-aws-kms-key-id`
- `x-amz-server-side-encryption-context`

SSE-KMS를 지원하는Amazon S3 REST API

다음 REST API는 `x-amz-server-side-encryption`, `x-amz-server-side-encryption-aws-kms-key-id` 및 `x-amz-server-side-encryption-context` 요청 헤더를 수락합니다.

- PUT Object — PUT API를 사용하여 데이터를 업로드할 때 이러한 요청 헤더를 지정할 수 있습니다.
- PUT Object - Copy— 객체를 복사할 때 원본 객체 및 대상 객체가 있어야 합니다. COPY 작업을 사용하여 SSE-KMS 헤더를 전달할 경우 헤더가 대상 객체에만 적용됩니다. 기존 객체를 복사할 때 원본 객체의 암호화 여부에 관계없이 명시적으로 서버 측 암호화를 요청하지 않는 한 대상 객체는 암호화되지 않습니다.
- POST Object— POST 작업을 사용하여 객체를 업로드할 경우에는 요청 헤더 대신 양식 필드에 동일한 정보를 제공합니다.
- 멀티파트 업로드 시작— 멀티파트 업로드 API를 사용하여 대형 객체를 업로드할 때 이 헤더를 지정할 수 있습니다. 시작 요청에서 이 헤더를 지정합니다.

객체가 서버 측 암호화를 사용하여 저장될 경우 다음 REST API의 응답 헤더는 `x-amz-server-side-encryption` 헤더를 반환합니다.

- PUT Object
- PUT Object - Copy
- POST Object
- 멀티파트 업로드 시작
- 파트 업로드
- Upload Part - Copy
- 멀티파트 업로드 완료
- Get Object
- Head Object

> **Important**
>
> - SSL(보안 소켓 언어) 또는 서명 버전 4를 사용하지 않으면 AWS KMS로 보호되는 객체에 대한 모든 GET 및 PUT 요청이 실패합니다.
> - 객체가 SSE-KMS를 사용하는 경우 `GET` 요청 및 `HEAD` 요청에 대해 암호화 요청 헤더를 전송하면 HTTP 400 BadRequest 오류가 발생합니다.

암호화 컨텍스트(`x-amz-server-side-encryption-context`)

`x-amz-server-side-encryption:aws:kms`를 지정하면 Amazon S3 API는 `x-amz-server-side-encryption-context` 헤더가 있는 암호화 컨텍스트를 지원합니다. 암호화 컨텍스트는 데이터에 대한 추

가 컨텍스트 정보를 포함할 수 있는 선택적 키-값 페어 집합입니다. 암호화 컨텍스트에 대한 자세한 내용은 AWS Key Management Service Developer Guide의 AWS Key Management Service 개념 - 암호화 컨텍스트를 참조하십시오.

헤더가 Base64로 인코딩된 JSON 형식을 준수한다면 암호화 컨텍스트는 원하는 어떤 값이든 될 수 있습니다. 하지만 암호화 컨텍스트는 암호화되지 않으며 AWS CloudTrail 로깅이 켜진 경우 로깅되기 때문에 암호화 컨텍스트에 중요한 정보가 포함되면 안 됩니다. 또한 AWS KMS에서 생성된 CloudTrail 이벤트를 더 잘 이해할 수 있도록 컨텍스트에서 암호화되거나 해독된 데이터를 설명하는 것이 좋습니다.

또한 Amazon S3에서 제공하는 암호화 컨텍스트에 대해 객체의 Amazon 리소스 이름(ARN)과 동일한 값을 사용하여 `aws:s3:arn`의 미리 정의된 키를 추가할 수 있습니다. 이는 `aws:s3:arn` 키가 사용자가 제공한 암호화 컨텍스트에 아직 없는 경우에만 발생합니다. 이 경우 Amazon S3가 PUT 요청을 처리할 때 미리 정의된 키가 추가됩니다. 사용자의 암호화 컨텍스트에 이 `aws:s3:arn` 키가 이미 있는 경우에는 암호화 컨텍스트에 키가 중복 추가되지 않습니다.

이 미리 정의된 키를 암호화 컨텍스트의 일부로 사용하면 CloudTrail에서 관련 요청을 추적할 수 있습니다. 따라서 어떤 Amazon S3 객체의 ARN이 어떤 암호화 키와 함께 사용되었는지 항상 확인할 수 있습니다. 또한 이 미리 정의된 키가 암호화 컨텍스트의 일부로 포함되면 다른 Amazon S3 객체에 동일한 암호화 컨텍스트가 사용되지 않도록 보장되므로 객체에 대한 추가 보안이 제공됩니다. 객체의 ARN과 같은 값이 있는지 전체 암호화 컨텍스트가 확인됩니다.

AWS KMS 키 ID(`x-amz-server-side-encryption-aws-kms-key-id`)

`x-amz-server-side-encryption-aws-kms-key-id` 헤더를 사용하여 데이터를 보호하는 데 사용되는 고객 관리 CMK의 ID를 지정할 수 있습니다. `x-amz-server-side-encryption:aws:kms`를 지정하지만 `x-amz-server-side-encryption-aws-kms-key-id`를 제공하지 않는 경우 Amazon S3는 AWS KMS에서 AWS 관리 CMK를 사용하여 데이터를 보호합니다. 고객 관리 AWS KMS CMK를 사용하려면 고객 관리 CMK의 `x-amz-server-side-encryption-aws-kms-key-id`를 제공해야 합니다.

> **Important**
>
> Amazon S3에서 서버 측 암호화에 AWS KMS CMK를 사용하는 경우 대칭 CMK를 선택해야 합니다. Amazon S3는 대칭 CMK만 지원하고 비대칭 CMK는 지원하지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 대칭 및 비대칭 키 사용을 참조하십시오.

Amazon S3가 관리하는 암호화 키(SSE-S3)를 사용하는 서버 측 암호화로 데이터 보호

서버측 암호화를 통해 유휴 데이터를 보호합니다. Amazon S3는 고유한 키로 각 객체를 암호화합니다. 또한 추가 보안 조치로 주기적으로 바뀌는 마스터 키를 사용하여 키 자체를 암호화합니다. Amazon S3 서버 측 암호화는 가장 강력한 블록 암호 중 하나인 256비트 고급 암호화 표준(AES-256)을 사용하여 데이터를 암호화합니다.

버킷에 저장된 모든 객체에 대해 서버 측 암호화가 필요할 경우 버킷 정책을 사용합니다. 예를 들어 다음 버킷 정책은 요청에 서버 측 암호화를 요청하는 `x-amz-server-side-encryption` 헤더가 포함되지 않을 경우 객체 업로드 권한을 거부합니다.

```
{
  "Version": "2012-10-17",
  "Id": "PutObjPolicy",
  "Statement": [
    {
      "Sid": "DenyIncorrectEncryptionHeader",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::YourBucket/*",
      "Condition": {
        "StringNotEquals": {
          "s3:x-amz-server-side-encryption": "AES256"
        }
```

```
      }
    },
    {
      "Sid": "DenyUnEncryptedObjectUploads",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::YourBucket/*",
      "Condition": {
        "Null": {
          "s3:x-amz-server-side-encryption": "true"
        }
      }
    }
  ]
}
```

Note

- 서버 측 암호화는 객체 메타데이터가 아니라 객체 데이터만 암호화합니다.

서버 측 암호화를 위한 API 지원

객체 생성 REST API를 사용하여 서버 측 암호화를 요청하려면 `x-amz-server-side-encryption` 요청 헤더를 제공합니다. REST API에 대한 정보는 REST API를 사용하여 서버 측 암호화 지정 (p. 277) 단원을 참조하십시오.

다음 Amazon S3 API는 이 헤더를 지원합니다.

- PUT 작업–PUT API를 사용하여 데이터를 업로드할 때 이러한 요청 헤더를 지정합니다. 자세한 내용은 PUT Object 단원을 참조하십시오.
- 멀티파트 업로드 시작–멀티파트 업로드 API를 사용하여 대용량 객체를 업로드할 때 시작 요청에 헤더를 지정할 수 있습니다. 자세한 내용은 멀티파트 업로드 시작을 참조하십시오.
- COPY 작업—객체를 복사하면 원본 객체 및 대상 객체를 모두 갖게 됩니다. 자세한 내용은 PUT Object - Copy를 참조하십시오.

Note

POST 작업을 사용하여 객체를 업로드할 경우에는 요청 헤더 대신 양식 필드에 동일한 정보를 제공합니다. 자세한 내용은 POST Object 단원을 참조하십시오.

AWS SDK 또한 서버 측 암호화를 요청하는 데 사용할 수 있는 래퍼 API를 제공하며, AWS Management 콘솔을 사용하여 객체를 업로드하고 서버 측 암호화를 요청할 수도 있습니다.

AWS SDK for Java를 사용하여 서버 측 암호화 지정

AWS SDK for Java를 사용하여 객체를 업로드할 때 서버 측 암호화를 사용하여 객체를 암호화할 수 있습니다. 서버 측 암호화를 요청하려면 `ObjectMetadata`의 `PutObjectRequest` 속성을 사용하여 `x-amz-server-side-encryption` 요청 헤더를 설정합니다. `AmazonS3Client`의 `putObject()` 메서드를 호출할 때 Amazon S3가 데이터를 암호화해 저장합니다.

멀티파트 업로드 API를 사용하여 객체를 업로드할 때도 서버 측 암호화를 요청할 수 있습니다.

- 상위 수준 멀티파트 업로드 API를 사용하는 경우 `TransferManager` 메서드를 사용하여 객체를 업로드할 때 객체에 서버 측 암호화를 적용합니다. `ObjectMetadata`를 파라미터로 사용하는 모든 업로드 메서드를 사용할 수 있습니다. 자세한 내용은 멀티파트 업로드용 AWS Java SDK(상위 수준 API) 사용 (p. 183) 단원을 참조하십시오.
- 하위 수준 멀티파트 업로드 API를 사용하는 경우 멀티파트 업로드를 시작할 때 서버 측 암호화를 지정합니다. `ObjectMetadata` 메서드를 호출하여

`InitiateMultipartUploadRequest.setObjectMetadata()` 속성을 추가합니다. 자세한 내용은 파일 업로드 (p. 187) 단원을 참조하십시오.

객체의 암호화 상태는 직접 변경할 수 없습니다(암호화되지 않은 객체 암호화 또는 암호화된 객체 암호 해독). 객체의 암호화 상태를 변경하려면 객체를 복사한 다음 사본에 대해 원하는 암호화 상태를 지정한 후 원본 객체를 삭제합니다. Amazon S3는 서버 측 암호화를 명시적으로 요청한 경우에만 객체의 사본을 암호화합니다. Java API를 통해 객체 사본 암호화를 요청하려면 `ObjectMetadata` 속성을 사용하여 `CopyObjectRequest`에서 서버 측 암호화를 지정합니다.

Example 예

다음 예제에서는 AWS SDK for Java를 사용하여 서버 측 암호화를 설정하는 방법을 보여줍니다. 이 예제에서는 다음 작업을 수행하는 방법을 설명합니다.

- 서버 측 암호화를 사용하여 새 객체를 업로드합니다.
- 객체의 사본을 만들어 객체의 암호화 상태를 변경합니다(이 예제에서는 이전에 암호화되지 않은 객체를 암호화함).
- 객체의 암호화 상태를 확인합니다.

서버 측 암호화에 대한 자세한 정보는 REST API를 사용하여 서버 측 암호화 지정 (p. 277) 단원을 참조하십시오. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
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;
import com.amazonaws.services.s3.internal.SSEResultBase;
import com.amazonaws.services.s3.model.*;

import java.io.ByteArrayInputStream;

public class SpecifyServerSideEncryption {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyNameToEncrypt = "*** Key name for an object to upload and encrypt ***";
        String keyNameToCopyAndEncrypt = "*** Key name for an unencrypted object to be
 encrypted by copying ***";
        String copiedObjectKeyName = "*** Key name for the encrypted copy of the
 unencrypted object ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();

            // Upload an object and encrypt it with SSE.
            uploadObjectWithSSEEncryption(s3Client, bucketName, keyNameToEncrypt);

            // Upload a new unencrypted object, then change its encryption state
            // to encrypted by making a copy.
            changeSSEEncryptionStatusByCopying(s3Client,
                    bucketName,
                    keyNameToCopyAndEncrypt,
```

```
                    copiedObjectKeyName);
        } 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();
        }
    }

    private static void uploadObjectWithSSEEncryption(AmazonS3 s3Client, String bucketName,
String keyName) {
        String objectContent = "Test object encrypted with SSE";
        byte[] objectBytes = objectContent.getBytes();

        // Specify server-side encryption.
        ObjectMetadata objectMetadata = new ObjectMetadata();
        objectMetadata.setContentLength(objectBytes.length);
        objectMetadata.setSSEAlgorithm(ObjectMetadata.AES_256_SERVER_SIDE_ENCRYPTION);
        PutObjectRequest putRequest = new PutObjectRequest(bucketName,
                keyName,
                new ByteArrayInputStream(objectBytes),
                objectMetadata);

        // Upload the object and check its encryption status.
        PutObjectResult putResult = s3Client.putObject(putRequest);
        System.out.println("Object \"" + keyName + "\" uploaded with SSE.");
        printEncryptionStatus(putResult);
    }

    private static void changeSSEEncryptionStatusByCopying(AmazonS3 s3Client,
                                                           String bucketName,
                                                           String sourceKey,
                                                           String destKey) {
        // Upload a new, unencrypted object.
        PutObjectResult putResult = s3Client.putObject(bucketName, sourceKey, "Object
example to encrypt by copying");
        System.out.println("Unencrypted object \"" + sourceKey + "\" uploaded.");
        printEncryptionStatus(putResult);

        // Make a copy of the object and use server-side encryption when storing the copy.
        CopyObjectRequest request = new CopyObjectRequest(bucketName,
                sourceKey,
                bucketName,
                destKey);
        ObjectMetadata objectMetadata = new ObjectMetadata();
        objectMetadata.setSSEAlgorithm(ObjectMetadata.AES_256_SERVER_SIDE_ENCRYPTION);
        request.setNewObjectMetadata(objectMetadata);

        // Perform the copy operation and display the copy's encryption status.
        CopyObjectResult response = s3Client.copyObject(request);
        System.out.println("Object \"" + destKey + "\" uploaded with SSE.");
        printEncryptionStatus(response);

        // Delete the original, unencrypted object, leaving only the encrypted copy in
Amazon S3.
        s3Client.deleteObject(bucketName, sourceKey);
        System.out.println("Unencrypted object \"" + sourceKey + "\" deleted.");
    }

    private static void printEncryptionStatus(SSEResultBase response) {
        String encryptionStatus = response.getSSEAlgorithm();
        if (encryptionStatus == null) {
            encryptionStatus = "Not encrypted with SSE";
```

```
        }
        System.out.println("Object encryption status is: " + encryptionStatus);
    }
}
```

.NET용 AWS SDK를 사용하여 서버 측 암호화 지정

객체를 업로드할 때 Amazon S3에 객체를 암호화하도록 지시할 수 있습니다. 기존 객체의 암호화 상태를 변경하기 위해서는 객체를 복사한 다음 원본 객체를 삭제합니다. 기본적으로 복사 작업은 대상 객체의 서버 측 암호화를 명시적으로 요청하는 경우에만 대상을 암호화합니다. `CopyObjectRequest`에 서버 측 암호화를 지정하려면 다음을 추가합니다.

```
ServerSideEncryptionMethod = ServerSideEncryptionMethod.AES256
```

객체를 복사하는 방법에 대한 실제 예제는 .NET용 AWS SDK를 사용하여 단일 작업으로 Amazon S3 객체 복사 (p. 211)를 참조하십시오.

다음은 객체를 업로드하는 예제입니다. 요청 시 이 예제는 Amazon S3에 객체를 암호화하도록 지시합니다. 그런 다음 객체 메타데이터를 검색해 사용된 암호화 방법을 확인합니다. 실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

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

namespace Amazon.DocSamples.S3
{
    class SpecifyServerSideEncryptionTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** key name for object created ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            WritingAnObjectAsync().Wait();
        }

        static async Task WritingAnObjectAsync()
        {
            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 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);
            }
        }
    }
}
```

PHP용 AWS SDK를 사용하여 서버 측 암호화 지정

이 주제에서는 Amazon Simple Storage Service(Amazon S3)로 업로드하는 객체에 서버 측 암호화를 추가하기 위해 PHP용 AWS SDK 버전 3의 클래스를 사용하는 방법을 보여줍니다. 이미 PHP용 AWS SDK 사용 및 PHP 예제 실행 (p. 736)의 지침에 따라 PHP용 AWS SDK가 올바르게 설치되어 있다고 가정합니다.

Amazon S3에 객체를 업로드하려면 Aws\S3\S3Client::putObject() 메서드를 사용합니다. 업로드 요청에 `x-amz-server-side-encryption` 요청 헤더를 추가하려면 다음 코드 예제에 나와 있듯이 `ServerSideEncryption` 파라미터를 `AES256` 값으로 지정합니다. 서버 측 암호화 요청에 대한 자세한 정보는 REST API를 사용하여 서버 측 암호화 지정 (p. 277) 단원을 참조하십시오.

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

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

// $filepath should be an absolute path to a file on disk.
$filepath = '*** Your File Path ***';

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

// Upload a file with server-side encryption.
$result = $s3->putObject([
    'Bucket'               => $bucket,
    'Key'                  => $keyname,
    'SourceFile'           => $filepath,
    'ServerSideEncryption' => 'AES256',
]);
```

이에 응답하여 Amazon S3가 객체 데이터 암호화에 사용된 암호화 알고리즘의 값과 함께 `x-amz-server-side-encryption` 헤더를 반환합니다.

멀티파트 업로드 API를 사용하여 대용량 객체를 업로드할 때 업로드할 객체에 대해 다음과 같이 서버 측 암호화를 지정할 수 있습니다.

- 하위 수준의 멀티파트 업로드 API를 사용하는 경우 Aws\S3\S3Client::createMultipartUpload() 메서드를 호출할 때 서버 측 암호화를 지정합니다. 요청에 `x-amz-server-side-encryption` 요청 헤더를 추가하려면 `array` 파라미터의 `ServerSideEncryption` 키를 값 `AES256`과 함께 지정합니다. 하위 수준 멀티파트 업로드 API에 대한 자세한 내용은 멀티파트 업로드용 AWS PHP SDK(하위 수준 API) 사용 (p. 202) 단원을 참조하십시오.
- 상위 수준 멀티파트 업로드 API를 사용하는 경우 CreateMultipartUpload 메서드의 `ServerSideEncryption` 파라미터를 사용하여 서버 측 암호화를 지정할 수 있습니다. 상위 수준 멀티파트 업로드 API와 함께 `setOption()` 메서드를 사용하는 예제는 멀티파트 업로드를 위해 AWS PHP SDK 사용 (p. 201) 단원을 참조하십시오.

사용된 암호화 알고리즘 확인

기존 객체의 암호화 상태를 확인하려면 다음 PHP 코드 예제에 나와 있듯이 Aws\S3\S3Client::headObject() 메서드를 호출하여 객체 메타데이터를 검색하십시오.

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

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

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

// Check which server-side encryption algorithm is used.
$result = $s3->headObject([
    'Bucket' => $bucket,
    'Key'    => $keyname,
]);
echo $result['ServerSideEncryption'];
```

기존 객체의 서버 측 암호화 변경(복사 작업)

기존 객체의 암호화 상태를 변경하려면 Aws\S3\S3Client::copyObject() 메서드를 사용하여 객체를 복사하고 원본 객체를 삭제하십시오. `ServerSideEncryption` 파라미터를 `AES256` 값으로 사용하여 대상 객체의 서버 측 암호화를 명시적으로 요청하지 않는 한 기본적으로 `copyObject()`는 대상을 암호화하지 않습니다. 다음 PHP 코드 예제에서는 객체를 복사하고 복사된 객체에 서버 측 암호화를 추가합니다.

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

use Aws\S3\S3Client;

$sourceBucket = '*** Your Source Bucket Name ***';
$sourceKeyname = '*** Your Source Object Key ***';

$targetBucket = '*** Your Target Bucket Name ***';
$targetKeyname = '*** Your Target Object Key ***';

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

// Copy an object and add server-side encryption.
$s3->copyObject([
    'Bucket'               => $targetBucket,
    'Key'                  => $targetKeyname,
    'CopySource'           => "{$sourceBucket}/{$sourceKeyname}",
```

```
    'ServerSideEncryption' => 'AES256',
]);
```

관련 리소스

- Amazon S3용 PHP용 AWS SDK Aws\S3\S3Client 클래스
- PHP용 AWS SDK 설명서

Ruby용 AWS SDK를 사용하여 서버 측 암호화 지정

Ruby용 AWS SDK를 사용하여 객체를 업로드하는 경우, 객체가 서버 측 암호화(SSE)를 통해 암호화된 상태로 저장되도록 지정할 수 있습니다. 다시 객체를 읽으면 객체가 자동으로 복호화됩니다.

다음 Ruby용 AWS SDK 버전 3 예제는 파일을 Amazon S3에 업로드한 후 암호화하여 저장하는 방법을 보여줍니다.

```
# The following example demonstrates how to specify that a file uploaded to Amazon S3 be
 encrypted at rest.
require 'aws-sdk-s3'

regionName = 'us-west-2'
bucketName = 'my-bucket'
key = 'key'
filePath = 'local/path/to/file'
encryptionType = 'AES256'

s3 = Aws::S3::Resource.new(region:regionName)
obj = s3.bucket(bucketName).object(key)
obj.upload_file(filePath, :server_side_encryption => encryptionType)
```

Ruby용 AWS SDK를 사용하여 객체 업로드 (p. 176)에서 SSE를 사용하지 않고 객체를 업로드하는 방법에 대한 예제를 참조할 수 있습니다.

사용된 암호화 알고리즘 확인

다음 코드 예제는 기존 객체의 암호화 상태를 확인하는 방법을 보여줍니다.

```
# Determine server-side encryption of an object.
require 'aws-sdk-s3'

regionName = 'us-west-2'
bucketName='bucket-name'
key = 'key'

s3 = Aws::S3::Resource.new(region:regionName)
enc = s3.bucket(bucketName).object(key).server_side_encryption
enc_state = (enc != nil) ? enc : "not set"
puts "Encryption state is #{enc_state}."
```

Amazon S3에 저장된 객체에 서버 측 암호화가 사용되지 않은 경우 메서드는 null을 반환합니다.

기존 객체의 서버 측 암호화 변경(복사 작업)

기존 객체의 암호화 상태를 변경하려면 객체를 복사한 다음 원본 객체를 삭제합니다. 기본적으로 복사 메서드는 대상을 암호화하지 않으며, 암호화하려면 서버 측 암호화를 명시적으로 요청해야 합니다. 다음 Ruby 코드 예제와 같이 옵션 해시 인수에 `server_side_encryption` 값을 지정하여 대상 객체의 암호화를 요청할 수 있습니다. 이 코드 예제는 객체를 복사하고 복사본을 암호화하는 방법을 보여줍니다.

```
require 'aws-sdk-s3'
```

```
regionName = 'us-west-2'
encryptionType = 'AES256'

s3 = Aws::S3::Resource.new(region:regionName)
bucket1 = s3.bucket('source-bucket-name')
bucket2 = s3.bucket('target-bucket-name')
obj1 = bucket1.object('Bucket1Key')
obj2 = bucket2.object('Bucket2Key')

obj1.copy_to(obj2, :server_side_encryption => encryptionType)
```

암호화하지 않고 객체를 복사하는 방법에 대한 예제는 Ruby용 AWS SDK를 사용하여 객체 복사 (p. 214)를 참조하십시오.

REST API를 사용하여 서버 측 암호화 지정

새 객체를 업로드하거나 기존 객체를 복사하여 객체가 생성될 경우 요청에 `x-amz-server-side-encryption` 헤더를 추가하여 Amazon S3가 데이터를 암호화하도록 지정할 수 있습니다. Amazon S3가 지원하는 암호화 알고리즘인 `AES256`으로 헤더의 값을 설정합니다. Amazon S3는 응답 헤더 `x-amz-server-side-encryption`을 반환하여 객체가 서버 측 암호화를 사용하여 저장됨을 확인해 줍니다.

다음 REST 업로드 API는 `x-amz-server-side-encryption` 요청 헤더를 수락합니다.

- PUT Object
- PUT Object - Copy
- POST Object
- 멀티파트 업로드 시작

멀티파트 업로드 API를 사용하여 대형 객체를 업로드할 경우 Initiate Multipart Upload 요청에 `x-amz-server-side-encryption` 헤더를 추가하여 서버 측 암호화를 지정할 수 있습니다. 기존 객체를 복사할 때 원본 객체의 암호화 여부에 관계없이 명시적으로 서버 측 암호화를 요청하지 않는 한 대상 객체는 암호화되지 않습니다.

객체가 서버 측 암호화를 사용하여 저장될 경우 다음 REST API의 응답 헤더는 `x-amz-server-side-encryption` 헤더를 반환합니다.

- PUT Object
- PUT Object - Copy
- POST Object
- 멀티파트 업로드 시작
- 파트 업로드
- Upload Part - Copy
- 멀티파트 업로드 완료
- Get Object
- Head Object

> **Note**
>
> 객체가 SSE-S3를 사용할 경우 `GET` 요청 및 `HEAD` 요청에 대해 암호화 요청 헤더를 전송하면 HTTP 400 BadRequest 오류가 발생합니다.

AWS Management 콘솔을 사용하여 서버 측 암호화 지정

AWS Management 콘솔을 사용하여 객체를 업로드할 때 서버 측 암호화를 지정할 수 있습니다. 객체를 업로드하는 방법에 대한 예는 S3 객체 업로드를 참조하십시오.

AWS Management 콘솔을 사용하여 객체를 복사할 경우 콘솔은 원본 그대로 객체를 복사합니다. 따라서 복사 원본이 암호화되어 있으면 대상 객체도 암호화됩니다.또한 콘솔에서 객체에 암호화를 추가할 수도 있습니다. 자세한 내용은 S3 객체에 암호화를 추가하려면 어떻게 해야 합니까?를 참조하십시오.

추가 정보

- S3 버킷에 대한 Amazon S3 기본 암호화 (p. 65)

고객 제공 암호화 키(SSE-C)로 서버 측 암호화를 사용하여 데이터 보호

서버 측 암호화는 저장된 데이터를 보호하기 위한 것입니다. 서버 측 암호화는 객체 메타데이터가 아니라 객체 데이터만 암호화합니다. 고객 제공 암호화 키(SSE-C)로 서버 측 암호화를 사용하여 자체 암호화 키를 설정할 수 있습니다. 요청의 일부로 제공한 암호화 키를 사용하여 Amazon S3는 디스크에 쓸 때 암호화를 관리하고 객체에 액세스할 때 암호 해독을 관리합니다. 따라서 데이터 암호화 및 암호 해독을 수행하기 위해 어떠한 코드도 사용할 필요가 없으며, 제공한 암호화 키를 관리하기만 하면 됩니다.

객체를 업로드하면 Amazon S3는 제공된 암호화 키를 사용하여 AES-256 암호화를 데이터에 적용하고 메모리에서 암호화 키를 삭제합니다. 객체를 검색할 경우 요청에 포함된 것과 동일한 암호화 키를 제공해야 합니다. Amazon S3는 제공된 암호화 키가 일치하는지 확인한 후, 객체의 암호를 해독하고 객체 데이터를 반환합니다.

> **Important**
>
> Amazon S3는 제공된 암호화 키를 저장하지 않습니다. 대신에 임의로 암호화 키의 솔트 HMAC 값을 저장하여 향후 요청을 검증합니다. 솔트 HMAC 값은 암호화 키의 값을 파생하거나 암호화된 객체의 콘텐츠를 해독하기 위해 사용할 수 없습니다. 즉, 암호화 키가 없어지면 객체도 없어집니다.

SSE-C 개요

이 단원에서는 SSE-C 개요를 다룹니다.

- HTTPS를 사용해야 합니다.

  > **Important**
  >
  > Amazon S3은 SSE-C를 사용할 때 HTTP를 통해 전송된 모든 요청을 거부합니다. 보안상의 이유로 인해, 보안에 취약할 수 있는 HTTP를 사용하여 키를 보내면 오류가 발생할 수 있음을 유의하십시오. 키를 취소하고 적절하게 교체해야 합니다.
- 응답의 ETag는 객체 데이터의 MD5가 아닙니다.
- 사용자는 객체를 암호화하는 데 사용된 암호화 키의 매핑을 관리해야 합니다. Amazon S3는 암호화 키를 저장하지 않습니다. 객체에 대해 제공한 암호화 키는 직접 추적해야 합니다.
  - 버전 관리가 활성화된 버킷의 경우, 이 기능을 사용하여 업로드 한 각 객체의 버전에는 자체 암호화 키가 있습니다. 어떤 객체 버전에 어떤 암호화 키가 사용되었는지는 직접 추적해야 합니다.
  - 암호화 키는 클라이언트 측에서 관리하기 때문에 클라이언트 측에서 키 교체와 같은 추가적인 보안 조치를 관리합니다.

    > **Warning**
    >
    > 암호화 키가 없어지면 암호화 키를 사용하지 않은 객체에 대한 모든 GET 요청이 실패하고 객체도 잃게 됩니다.

주제

자세한 내용은 다음 주제를 참조하십시오.

- AWS SDK for Java를 사용하여 고객 제공 암호화 키로 서버 측 암호화 지정 (p. 279)
- .NET용 AWS SDK를 사용하여 고객 제공 암호화 키로 서버 측 암호화 지정 (p. 283)
- REST API를 사용하여 고객 제공 암호화 키로 서버 측 암호화 지정 (p. 290)

AWS SDK for Java를 사용하여 고객 제공 암호화 키로 서버 측 암호화 지정

다음 예제에서는 객체에 대해 고객 제공 키를 사용한 서버 측 암호화(SSE-C)를 요청하는 방법을 보여줍니다. 이 예제에서는 다음 작업을 수행합니다. 각 작업에서는 요청에 SSE-C 관련 헤더를 지정하는 방법을 보여 줍니다.

- 객체 넣기 — 객체를 업로드하고 고객 제공 암호화 키를 사용하여 서버 측 암호화를 요청합니다.
- 객체 가져오기 — 이전 단계에서 업로드한 객체를 다운로드합니다. 요청 시 객체가 업로드되었을 때 제공한 것과 동일한 암호화 정보를 제공합니다. Amazon S3는 객체를 암호화하여 사용자에게 반환하기 위해 이 정보가 필요합니다.
- 객체 메타데이터 가져오기 — 객체의 메타데이터를 가져옵니다. 객체 생성 시 사용한 것과 동일한 암호화 정보를 제공합니다.
- 객체 복사 — 이전에 업로드한 객체의 복사본을 만듭니다. 원본 객체가 SSE-C를 사용하여 저장되기 때문에 복사 요청에 암호화 정보를 제공해야 합니다. 기본적으로 Amazon S3은 사용자가 명시적으로 요청하는 경우에만 객체의 사본을 암호화합니다. 이 예제에서는 Amazon S3에 새 SSE-C 키를 사용하여 객체의 암호화된 사본을 저장하도록 지시합니다.

> **Note**
>
> 이 예제에서는 단일 작업으로 객체를 업로드하는 방법을 보여 줍니다. 멀티파트 업로드 API를 사용하여 대용량 객체를 업로드할 때는 다음 예제에서 보여주는 방식과 동일한 방식으로 암호화 정보를 제공합니다. AWS SDK for Java를 사용하는 멀티파트 업로드에 대한 예제는 멀티파트 업로드용 AWS Java SDK(상위 수준 API) 사용 (p. 183) 및 멀티파트 업로드용 AWS Java SDK(하위 수준 API) 사용 (p. 187) 단원을 참조하십시오.

필요한 암호화 정보를 추가하려면 요청에 `SSECustomerKey`를 포함합니다. `SSECustomerKey` 클래스에 대한 자세한 내용은 REST API를 사용하여 고객 제공 암호화 키로 서버 측 암호화 지정 (p. 290)를 참조하십시오.

SSE-C에 대한 자세한 내용은 고객 제공 암호화 키(SSE-C)로 서버 측 암호화를 사용하여 데이터 보호 (p. 278) 단원을 참조하십시오. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

Example

```
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;
import com.amazonaws.services.s3.model.*;

import javax.crypto.KeyGenerator;
import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;

public class ServerSideEncryptionUsingClientSideEncryptionKey {
    private static SSECustomerKey SSE_KEY;
    private static AmazonS3 S3_CLIENT;
    private static KeyGenerator KEY_GENERATOR;

    public static void main(String[] args) throws IOException, NoSuchAlgorithmException {
        Regions clientRegion = Regions.DEFAULT_REGION;
```

```
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";
        String uploadFileName = "*** File path ***";
        String targetKeyName = "*** Target key name ***";

        // Create an encryption key.
        KEY_GENERATOR = KeyGenerator.getInstance("AES");
        KEY_GENERATOR.init(256, new SecureRandom());
        SSE_KEY = new SSECustomerKey(KEY_GENERATOR.generateKey());

        try {
            S3_CLIENT = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Upload an object.
            uploadObject(bucketName, keyName, new File(uploadFileName));

            // Download the object.
            downloadObject(bucketName, keyName);

            // Verify that the object is properly encrypted by attempting to retrieve it
            // using the encryption key.
            retrieveObjectMetadata(bucketName, keyName);

            // Copy the object into a new object that also uses SSE-C.
            copyObject(bucketName, keyName, targetKeyName);
        } 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();
        }
    }

    private static void uploadObject(String bucketName, String keyName, File file) {
        PutObjectRequest putRequest = new PutObjectRequest(bucketName, keyName,
file).withSSECustomerKey(SSE_KEY);
        S3_CLIENT.putObject(putRequest);
        System.out.println("Object uploaded");
    }

    private static void downloadObject(String bucketName, String keyName) throws
IOException {
        GetObjectRequest getObjectRequest = new GetObjectRequest(bucketName,
keyName).withSSECustomerKey(SSE_KEY);
        S3Object object = S3_CLIENT.getObject(getObjectRequest);

        System.out.println("Object content: ");
        displayTextInputStream(object.getObjectContent());
    }

    private static void retrieveObjectMetadata(String bucketName, String keyName) {
        GetObjectMetadataRequest getMetadataRequest = new
GetObjectMetadataRequest(bucketName, keyName)
                .withSSECustomerKey(SSE_KEY);
        ObjectMetadata objectMetadata = S3_CLIENT.getObjectMetadata(getMetadataRequest);
        System.out.println("Metadata retrieved. Object size: " +
objectMetadata.getContentLength());
    }

    private static void copyObject(String bucketName, String keyName, String targetKeyName)
```

```
            throws NoSuchAlgorithmException {
        // Create a new encryption key for target so that the target is saved using SSE-C.
        SSECustomerKey newSSEKey = new SSECustomerKey(KEY_GENERATOR.generateKey());

        CopyObjectRequest copyRequest = new CopyObjectRequest(bucketName, keyName,
 bucketName, targetKeyName)
                .withSourceSSECustomerKey(SSE_KEY)
                .withDestinationSSECustomerKey(newSSEKey);

        S3_CLIENT.copyObject(copyRequest);
        System.out.println("Object copied");
    }

    private static void displayTextInputStream(S3ObjectInputStream input) throws
 IOException {
        // Read one line at a time from the input stream and display each line.
        BufferedReader reader = new BufferedReader(new InputStreamReader(input));
        String line;
        while ((line = reader.readLine()) != null) {
            System.out.println(line);
        }
        System.out.println();
    }
}
```

AWS SDK for Java를 사용하는 SSE-C가 적용된 기타 Amazon S3 작업

이전 단원의 예제에서는 PUT, GET, Head 및 Copy 작업에서 고객 제공 키로 서버 측 암호화(SSE-C)를 요청하는 방법을 보여 줍니다. 이 단원에서는 SSE-C를 지원하는 기타 API에 대해 소개합니다.

멀티파트 업로드 API를 사용하여 대형 객체를 업로드할 수 있습니다(멀티파트 업로드 API를 사용한 객체 업로드 (p. 177) 참조). 상위 수준 또는 하위 수준의 API를 사용하여 대형 객체를 업로드할 수 있습니다. 이들 API는 요청의 암호화 관련 헤더를 지원합니다.

- 고급 `TransferManager` API를 사용할 때는 `PutObjectRequest`에 암호화 관련 헤더를 제공합니다(멀티파트 업로드용 AWS Java SDK(상위 수준 API) 사용 (p. 183) 단원 참조).
- 하위 수준 API를 사용할 경우 `InitiateMultipartUploadRequest`에 암호화 관련 정보를 제공하고 각 `UploadPartRequest`에 동일한 암호화 정보를 제공합니다. `CompleteMultipartUploadRequest`에서는 암호화 관련 헤더를 제공할 필요가 없습니다. 예제는 멀티파트 업로드용 AWS Java SDK(하위 수준 API) 사용 (p. 187)을 참조하십시오.

다음 예제에서는 `TransferManager`를 사용하여 객체를 만들고 SSE-C 관련 정보를 제공하는 방법을 보여 줍니다. 이 예제는 다음을 수행합니다.

- `TransferManager.upload()` 메서드를 사용하여 객체를 생성합니다. `PutObjectRequest` 인스턴스에서는 사용자가 요청에 암호화 키 정보를 제공합니다. Amazon S3는 고객이 제공하는 암호화 키를 사용하여 객체를 암호화합니다.
- `TransferManager.copy()` 메서드를 호출하여 객체의 복사본을 생성합니다. 이 예제에서는 Amazon S3에게 새 `SSECustomerKey`를 사용하여 객체 복사본을 암호화하도록 지시합니다. 원본 객체가 SSE-C를 사용하여 암호화되기 때문에 Amazon S3가 복사 전에 객체의 암호를 해독할 수 있도록 `CopyObjectRequest` 실행 시 원본 객체의 암호화 키도 제공됩니다.

Example

```
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;
import com.amazonaws.services.s3.model.CopyObjectRequest;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.amazonaws.services.s3.model.SSECustomerKey;
import com.amazonaws.services.s3.transfer.Copy;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
import com.amazonaws.services.s3.transfer.Upload;

import javax.crypto.KeyGenerator;
import java.io.File;
import java.security.SecureRandom;

public class ServerSideEncryptionCopyObjectUsingHLwithSSEC {

    public static void main(String[] args) throws Exception {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String fileToUpload = "*** File path ***";
        String keyName = "*** New object key name ***";
        String targetKeyName = "*** Key name for object copy ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();
            TransferManager tm = TransferManagerBuilder.standard()
                    .withS3Client(s3Client)
                    .build();

            // Create an object from a file.
            PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, keyName,
 new File(fileToUpload));

            // Create an encryption key.
            KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
            keyGenerator.init(256, new SecureRandom());
            SSECustomerKey sseCustomerEncryptionKey = new
 SSECustomerKey(keyGenerator.generateKey());

            // Upload the object. TransferManager uploads asynchronously, so this call
 returns immediately.
            putObjectRequest.setSSECustomerKey(sseCustomerEncryptionKey);
            Upload upload = tm.upload(putObjectRequest);

            // Optionally, wait for the upload to finish before continuing.
            upload.waitForCompletion();
            System.out.println("Object created.");

            // Copy the object and store the copy using SSE-C with a new key.
            CopyObjectRequest copyObjectRequest = new CopyObjectRequest(bucketName,
 keyName, bucketName, targetKeyName);
            SSECustomerKey sseTargetObjectEncryptionKey = new
 SSECustomerKey(keyGenerator.generateKey());
            copyObjectRequest.setSourceSSECustomerKey(sseCustomerEncryptionKey);
            copyObjectRequest.setDestinationSSECustomerKey(sseTargetObjectEncryptionKey);

            // Copy the object. TransferManager copies asynchronously, so this call returns
 immediately.
            Copy copy = tm.copy(copyObjectRequest);

            // Optionally, wait for the upload to finish before continuing.
            copy.waitForCompletion();
```

```
            System.out.println("Copy complete.");
        } 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();
        }
    }
}
```

.NET용 AWS SDK를 사용하여 고객 제공 암호화 키로 서버 측 암호화 지정

다음 C# 예제에서는 고객 제공 키(SSE-C)가 작동하는 방법을 보여 줍니다. 이 예제에서는 다음 작업을 수행합니다. 각 작업에서는 요청에 SSE-C 관련 헤더를 지정하는 방법을 보여 줍니다.

- 객체 넣기–객체를 업로드하고 고객 제공 암호화 키를 사용하여 서버 측 암호화를 요청합니다.
- 객체 가져오기–이전 단계에서 업로드한 객체를 다운로드합니다. 요청은 객체가 업로드되었을 때 제공된 것과 동일한 암호화 정보를 제공합니다. Amazon S3는 객체를 암호화하여 사용자에게 반환하기 위해 이 정보가 필요합니다.
- 객체 메타데이터 가져오기–객체의 메타데이터를 검색하기 위해 객체가 생성되었을 때 사용된 것과 동일한 암호화 정보를 제공합니다.
- 객체 복사–업로드한 객체의 복사본을 만듭니다. 원본 객체가 SSE-C를 사용하여 저장되었으므로 복사 요청에서도 암호화 정보를 제공해야 합니다. 기본적으로 Amazon S3은 객체의 복사본을 암호화하지 않습니다. 이 코드는 대상에 대한 암호화 관련 정보를 제공하여 Amazon S3에 SSE-C를 사용하여 복사된 객체를 암호화하라고 지시합니다. 또한 대상을 저장합니다.

> **Note**
>
> 멀티파트 업로드 API를 사용하여 대형 객체를 업로드하는 예제는 멀티파트 업로드에 .NET용 AWS SDK(상위 수준 API) 사용 (p. 191) 및 멀티파트 업로드에 .NET용 AWS SDK(하위 수준 API) 사용 (p. 197) 단원을 참조하십시오.

SSE-C에 대한 자세한 내용은 고객 제공 암호화 키(SSE-C)로 서버 측 암호화를 사용하여 데이터 보호 (p. 278) 단원을 참조하십시오. 실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

Example

```
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class SSEClientEncryptionKeyObjectOperationsTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** key name for new object created ***";
        private const string copyTargetKeyName = "*** key name for object copy ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 client;
```

```
        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            ObjectOpsUsingClientEncryptionKeyAsync().Wait();
        }
        private static async Task ObjectOpsUsingClientEncryptionKeyAsync()
        {
            try
            {
                // Create an encryption key.
                Aes aesEncryption = Aes.Create();
                aesEncryption.KeySize = 256;
                aesEncryption.GenerateKey();
                string base64Key = Convert.ToBase64String(aesEncryption.Key);

                // 1. Upload the object.
                PutObjectRequest putObjectRequest = await UploadObjectAsync(base64Key);
                // 2. Download the object and verify that its contents matches what you
uploaded.
                await DownloadObjectAsync(base64Key, putObjectRequest);
                // 3. Get object metadata and verify that the object uses AES-256
encryption.
                await GetObjectMetadataAsync(base64Key);
                // 4. Copy both the source and target objects using server-side encryption
with
                //    a customer-provided encryption key.
                await CopyObjectAsync(aesEncryption, base64Key);
            }
            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<PutObjectRequest> UploadObjectAsync(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;
        }
        private static async Task DownloadObjectAsync(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");

                // Assert.AreEqual(putObjectRequest.ContentBody, content);
                // Assert.AreEqual(ServerSideEncryptionCustomerMethod.AES256,
getResponse.ServerSideEncryptionCustomerMethod);
            }
        }
        private static async Task GetObjectMetadataAsync(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);
            // Assert.AreEqual(ServerSideEncryptionCustomerMethod.AES256,
getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
        }
        private static async Task CopyObjectAsync(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);
        }
    }
}
```

기타 Amazon S3 작업 및 SSE-C

앞 단원의 예제에서는 PUT, GET, Head 및 Copy 작업에서 고객 제공 키(SSE-C)를 사용하는 서버 측 암호화를 요청하는 방법을 알아보았으며, 이 단원에서는 SSE-C를 지원하는 Amazon S3 API에 대해 소개합니다.

멀티파트 업로드 API를 사용하여 대형 객체를 업로드할 수 있습니다(멀티파트 업로드 API를 사용한 객체 업로드 (p. 177) 참조). .NET용 AWS SDK는 대형 객체를 업로드하는 상위 수준 또는 하위 수준의 API를 제공합니다. 이들 API는 요청의 암호화 관련 헤더를 지원합니다.

- 상위 수준의 `Transfer-Utility` API를 사용할 때는 아래와 같이 `TransferUtilityUploadRequest`에 암호화 관련 헤더를 제공합니다. 코드 예제는 멀티파트 업로드에 .NET용 AWS SDK(상위 수준 API) 사용 (p. 191) 단원을 참조하십시오.

```
TransferUtilityUploadRequest request = new TransferUtilityUploadRequest()
{
    FilePath = filePath,
    BucketName = existingBucketName,
    Key = keyName,
    // Provide encryption information.
    ServerSideEncryptionCustomerMethod = ServerSideEncryptionCustomerMethod.AES256,
    ServerSideEncryptionCustomerProvidedKey = base64Key,
};
```

- 하위 수준의 API를 사용할 경우 멀티파트 업로드 요청을 시작할 때 암호화 관련 정보를 제공하고, 이후 부분 업로드 요청 시에도 동일한 암호화 정보를 제공합니다. 멀티파트 업로드 완료 요청에서는 암호화 관련 헤더를 제공할 필요가 없습니다. 예제는 멀티파트 업로드에 .NET용 AWS SDK(하위 수준 API) 사용 (p. 197)을 참조하십시오.

  다음은 기존 대형 객체의 복사본을 만드는 하위 수준의 멀티파트 업로드 예제입니다. 이 예제에서 원본 객체는 SSE-C를 사용하여 Amazon S3에 저장되고, 대상 객체 또한 SSE-C를 사용하여 저장됩니다. 이 예제에서는 다음 작업을 수행합니다.
  - 암호화 키와 관련 정보를 제공하여 멀티파트 업로드 요청을 시작합니다.
  - `CopyPartRequest`에 원본 및 대상 객체 암호화 키와 관련 정보를 제공합니다.
  - 객체 메타데이터를 검색하여 복사할 원본 객체의 크기를 확인합니다.
  - 5MB 단위로 객체를 업로드합니다.

  Example

```
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class SSECLowLevelMPUcopyObjectTest
    {
        private const string existingBucketName = "*** bucket name ***";
        private const string sourceKeyName      = "*** source object key name ***";
```

```
        private const string targetKeyName      = "*** key name for the target object
***";
        private const string filePath           = @"*** file path ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            CopyObjClientEncryptionKeyAsync().Wait();
        }

        private static async Task CopyObjClientEncryptionKeyAsync()
        {
            Aes aesEncryption = Aes.Create();
            aesEncryption.KeySize = 256;
            aesEncryption.GenerateKey();
            string base64Key = Convert.ToBase64String(aesEncryption.Key);

            await CreateSampleObjUsingClientEncryptionKeyAsync(base64Key, s3Client);

            await CopyObjectAsync(s3Client, base64Key);
        }
        private static async Task CopyObjectAsync(IAmazonS3 s3Client, string base64Key)
        {
            List<CopyPartResponse> uploadResponses = new List<CopyPartResponse>();

            // 1. Initialize.
            InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = targetKeyName,
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            InitiateMultipartUploadResponse initResponse =
                await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // 2. Upload Parts.
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB
            long firstByte = 0;
            long lastByte = partSize;

            try
            {
                // First find source object size. Because object is stored encrypted with
                // customer provided key you need to provide encryption information in
your request.
                GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest()
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                    ServerSideEncryptionCustomerProvidedKey = base64Key // " * **source
object encryption key ***"
                };

                GetObjectMetadataResponse getObjectMetadataResponse = await
s3Client.GetObjectMetadataAsync(getObjectMetadataRequest);

                long filePosition = 0;
```

```
                for (int i = 1; filePosition < getObjectMetadataResponse.ContentLength; i
++)
                {
                    CopyPartRequest copyPartRequest = new CopyPartRequest
                    {
                        UploadId = initResponse.UploadId,
                        // Source.
                        SourceBucket = existingBucketName,
                        SourceKey = sourceKeyName,
                        // Source object is stored using SSE-C. Provide encryption
 information.
                        CopySourceServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                        CopySourceServerSideEncryptionCustomerProvidedKey =
 base64Key, //"***source object encryption key ***",
                        FirstByte = firstByte,
                        // If the last part is smaller then our normal part size then use
 the remaining size.
                        LastByte = lastByte > getObjectMetadataResponse.ContentLength ?
                            getObjectMetadataResponse.ContentLength - 1 : lastByte,

                        // Target.
                        DestinationBucket = existingBucketName,
                        DestinationKey = targetKeyName,
                        PartNumber = i,
                        // Encryption information for the target object.
                        ServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                        ServerSideEncryptionCustomerProvidedKey = base64Key
                    };
                    uploadResponses.Add(await s3Client.CopyPartAsync(copyPartRequest));
                    filePosition += partSize;
                    firstByte += partSize;
                    lastByte += partSize;
                }

                // Step 3: complete.
                CompleteMultipartUploadRequest completeRequest = new
 CompleteMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = targetKeyName,
                    UploadId = initResponse.UploadId,
                };
                completeRequest.AddPartETags(uploadResponses);

                CompleteMultipartUploadResponse completeUploadResponse =
                    await s3Client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (Exception exception)
            {
                Console.WriteLine("Exception occurred: {0}", exception.Message);
                AbortMultipartUploadRequest abortMPURequest = new
 AbortMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = targetKeyName,
                    UploadId = initResponse.UploadId
                };
                s3Client.AbortMultipartUpload(abortMPURequest);
            }
        }
        private static async Task CreateSampleObjUsingClientEncryptionKeyAsync(string
 base64Key, IAmazonS3 s3Client)
        {
            // List to store upload part responses.
```

```
            List<UploadPartResponse> uploadResponses = new List<UploadPartResponse>();

            // 1. Initialize.
            InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = sourceKeyName,
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key
            };

            InitiateMultipartUploadResponse initResponse =
               await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // 2. Upload Parts.
            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 s3Client.UploadPartAsync(uploadRequest));

                    filePosition += partSize;
                }

                // Step 3: complete.
                CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId,
                    //PartETags = new List<PartETag>(uploadResponses)

                };
                completeRequest.AddPartETags(uploadResponses);

                CompleteMultipartUploadResponse completeUploadResponse =
                    await s3Client.CompleteMultipartUploadAsync(completeRequest);

            }
            catch (Exception exception)
            {
                Console.WriteLine("Exception occurred: {0}", exception.Message);
                AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
```

```
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId
                };
                await s3Client.AbortMultipartUploadAsync(abortMPURequest);
            }
        }
    }
}
```

REST API를 사용하여 고객 제공 암호화 키로 서버 측 암호화 지정

REST API를 사용하여 객체를 생성할 때 고객 제공 암호화 키(SSE-C)를 사용하여 서버 측 암호화를 지정할 수 있습니다. SSE-C를 사용하는 경우 다음 요청 헤더를 사용하여 암호화 키 정보를 제공해야 합니다.

이름	설명
`x-amz-server-side-encryption-customer-algorithm`	헤더를 사용하여 암호화 알고리즘을 지정합니다. 헤더 값은 "AES256"이어야 합니다.
`x-amz-server-side-encryption-customer-key`	이 헤더를 사용하여 256비트의 base64 인코딩 암호화 키를 Amazon S3에 제공하여 데이터를 암호화하거나 암호 해독합니다.
`x-amz-server-side-encryption-customer-key-MD5`	이 헤더를 사용하여 RFC 1321에 따라 암호화 키의 128비트 base64 인코딩 MD5 다이제스트를 제공합니다. Amazon S3은 메시지 무결성 검사에 이 헤더를 사용하여 암호화 키가 오류 없이 전송되었음을 확인합니다.

AWS SDK 래퍼 라이브러리를 사용하여 이러한 헤더를 요청에 추가할 수 있습니다. 필요한 경우 애플리케이션에서 직접 Amazon S3 REST API를 호출할 수 있습니다.

> **Note**
>
> Amazon S3 콘솔을 사용하여 객체를 업로드하거나 SSE-C를 요청할 수 없습니다. 또한 SSE-C를 사용하여 저장된 기존 객체를 업데이트할 수 도 없습니다(예: 스토리지 클래스 변경 또는 메타데이터 추가).

SSE-C를 지원하는 Amazon S3 REST API

다음 Amazon S3 API는 고객 제공 암호화 키(SSE-C)로 서버 측 암호화를 지원합니다.

- GET 작업 — GET API(GET Object 참조)를 사용하여 객체를 검색할 때 요청 헤더를 지정할 수 있습니다. Torrents는 SSE-C를 사용한 객체 암호화를 지원하지 않습니다.
- HEAD 작업 — HEAD API(HEAD Object 참조)를 사용하여 객체 메타데이터를 검색하려면 이러한 요청 헤더를 지정하면 됩니다.
- PUT 작업 — PUT Object API(PUT Object 참조)를 사용하여 데이터를 업로드할 때 이러한 요청 헤더를 지정할 수 있습니다.
- 멀티파트 업로드 — 멀티파트 업로드 API를 사용하여 대형 객체를 업로드할 때 이 헤더를 지정할 수 있습니다. 시작 요청에서 이 헤더를 지정하고(멀티파트 업로드 시작 참조) 각각의 후속 부분은 요청을 업로드합니다(파트 업로드 또는

  파트 업로드 - 복사)

). 각 파트 업로드 요청에서 암호화 정보는 시작 멀티파트 업로드 요청의 시작에서 제공한 내용과 동일해야 합니다.

- POST 작업— POST 작업(POST Object 참조)을 사용하여 객체를 업로드할 경우에는 요청 헤더 대신 양식 필드에 동일한 정보를 제공합니다.
- Copy 작업— 객체를 복사(PUT Object - Copy 참조)하려면 원본 객체 및 대상 객체가 있어야 합니다.
  - AWS 관리형 키로 서버 측 암호를 사용하여 대상 객체를 암호화하려는 경우, `x-amz-server-side-encryption` 요청 헤더를 제공해야 합니다.
  - SSE-C를 사용하여 대상 객체를 암호화하려는 경우, 앞의 표에서 설명한 3개의 헤더를 사용하여 암호화 정보를 제공해야 합니다.
  - SSE-C를 사용하여 원본 객체가 암호화된 경우, 다음 헤더를 사용하여 암호화 키 정보를 제공해야만 Amazon S3가 객체 복사를 위해 객체의 암호화를 해독할 수 있습니다.

이름	설명
`x-amz-copy-source-server-side-encryption-customer-algorithm`	이 헤더를 포함하여 Amazon S3가 원본 객체 해독에 사용해야 하는 알고리즘을 지정합니다. 이 값은 `AES256`이어야 합니다.
`x-amz-copy-source-server-side-encryption-customer-key`	이 헤더를 포함하여 Amazon S3가 원본 객체의 암호화를 해독하는 데 사용할 base64 인코딩 암호화 키를 제공합니다. 이 암호화 키는 원본 객체를 만들 때 Amazon S3에 제공한 것이어야 하며, 그렇지 않으면 Amazon S3이 객체 암호를 해독할 수 없습니다.
`x-amz-copy-source-server-side-encryption-customer-key-MD5`	이 헤더를 포함하여 RFC 1321에 따라 암호화 키의 128비트 base64 인코딩 MD5 다이제스트를 제공합니다.

미리 서명된 URL 및 SSE-C

새로운 객체 업로드, 기존 객체 또는 객체 메타데이터 검색과 같은 작업에 사용되는 미리 서명된 URL을 만들 수 있습니다. 미리 서명된 URL은 다음과 같이 SSE-C를 지원합니다.

- 미리 서명된 URL을 만들 때 서명 계산에 `x-amz-server-side-encryption-customer-algorithm`을 사용하여 알고리즘을 지정해야 합니다.
- 새로운 객체 업로드, 기존 객체 또는 객체 메타데이터 전용 검색에 미리 서명된 URL을 사용하는 경우, 클라이언트 애플리케이션에서 모든 암호화 헤더를 제공해야 합니다.

  > **Note**
  >
  > 비 SSE-C 객체의 경우, 예를 들어 데이터에 액세스하기 위해 미리 서명된 URL을 생성하여 브라우저에 직접 붙여 넣을 수 있습니다.
  > 하지만 SSE-C 객체에는 이렇게 할 수 없습니다. 미리 서명된 URL 이외에 SSE-C 객체에 고유한 HTTP 헤더도 포함시켜야 하기 때문입니다. 따라서 프로그래밍 방식으로만 SSE-C 객체에 미리 서명된 URL을 생성할 수 있습니다.

추가 정보

- .NET용 AWS SDK를 사용하여 고객 제공 암호화 키로 서버 측 암호화 지정 (p. 283)
- AWS SDK for Java를 사용하여 고객 제공 암호화 키로 서버 측 암호화 지정 (p. 279)

클라이언트측 암호화를 사용하여 데이터 보호

클라이언트 측 암호화는 Amazon S3로 보내기 전에 데이터를 암호화하는 것을 가리킵니다. 클라이언트 측 암호화를 활성화하기 위한 다음 옵션이 있습니다.

- AWS Key Management Service(AWS KMS)에 저장된 고객 마스터 키(CMK)를 사용합니다.
- 애플리케이션 내에 저장한 마스터 키를 사용합니다.

다음 AWS SDK에서 클라이언트 측 암호화를 지원합니다.

- .NET용 AWS SDK
- Go용 AWS SDK
- AWS SDK for Java
- PHP용 AWS SDK
- Ruby용 AWS SDK
- C++용 AWS SDK

옵션 1: AWS KMS에 저장된 CMK 사용

- 객체를 업로드할 때 — 고객 마스터 키(CMK) ID를 사용하여 클라이언트가 먼저 AWS KMS로 객체 데이터를 암호화하는 데 사용할 수 있는 CMK를 요청합니다. AWS KMS는 임의로 생성된 데이터 키의 다음 두 가지 버전을 반환합니다.
  - 클라이언트가 객체 데이터를 암호화하는 데 사용하는 일반 텍스트 버전의 데이터 키
  - 클라이언트가 Amazon S3에 객체 메타데이터로 업로드하는 동일한 데이터 키의 암호 BLOB

    Note

    클라이언트는 업로드하는 각 객체에 대해 고유한 데이터 키를 가져옵니다.
- 객체를 다운로드할 때 — 클라이언트가 Amazon S3에서 암호화된 객체를 객체 메타데이터로 저장된 데이터 키의 암호 BLOB 버전과 함께 다운로드합니다. 그런 다음 클라이언트가 암호 BLOB를 AWS KMS로 보내 객체 데이터의 암호 해독을 위해 일반 텍스트 버전의 데이터 키를 가져옵니다.

AWS KMS에 대한 자세한 내용은 AWS Key Management Service Developer Guide의 AWS Key Management Service란 무엇입니까?를 참조하십시오.

Example

다음 예제에서는 AWS SDK for Java에서 AWS KMS를 사용하여 Amazon S3로 객체를 업로드합니다. 이 예제에서는 데이터를 Amazon S3으로 업로드하기 전에 AWS 관리형 CMK를 사용하여 클라이언트 측 데이터를 암호화합니다. CMK가 이미 있는 경우에는 샘플 코드에서 `kms_cmk_id` 변수의 값을 지정하여 이 키를 사용할 수 있습니다. CMK가 없거나 다른 키가 필요한 경우 Java API를 통해 하나 생성할 수 있습니다. 다음 예제에서는 CMK를 생성하는 방법을 보여줍니다.

실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.RegionUtils;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.kms.AWSKMS;
import com.amazonaws.services.kms.AWSKMSClientBuilder;
import com.amazonaws.services.kms.model.CreateKeyResult;
import com.amazonaws.services.s3.AmazonS3Encryption;
import com.amazonaws.services.s3.AmazonS3EncryptionClientBuilder;
import com.amazonaws.services.s3.model.CryptoConfiguration;
import com.amazonaws.services.s3.model.KMSEncryptionMaterialsProvider;
```

```
import com.amazonaws.services.s3.model.S3Object;
import com.amazonaws.services.s3.model.S3ObjectInputStream;

import java.io.ByteArrayOutputStream;
import java.io.IOException;

public class UploadObjectKMSKey {

    public static void main(String[] args) throws IOException {
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Object key name ***";
        Regions clientRegion = Regions.DEFAULT_REGION;
        String kms_cmk_id = "*** AWS KMS customer master key ID ***";
        int readChunkSize = 4096;

        try {
            // Optional: If you don't have a KMS key (or need another one),
            // create one. This example creates a key with AWS-created
            // key material.
            AWSKMS kmsClient = AWSKMSClientBuilder.standard()
                    .withRegion(clientRegion)
                    .build();
            CreateKeyResult keyResult = kmsClient.createKey();
            kms_cmk_id = keyResult.getKeyMetadata().getKeyId();

            // Create the encryption client.
            KMSEncryptionMaterialsProvider materialProvider = new
 KMSEncryptionMaterialsProvider(kms_cmk_id);
            CryptoConfiguration cryptoConfig = new CryptoConfiguration()
                    .withAwsKmsRegion(RegionUtils.getRegion(clientRegion.toString()));
            AmazonS3Encryption encryptionClient =
 AmazonS3EncryptionClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withEncryptionMaterials(materialProvider)
                    .withCryptoConfiguration(cryptoConfig)
                    .withRegion(clientRegion).build();

            // Upload an object using the encryption client.
            String origContent = "S3 Encrypted Object Using KMS-Managed Customer Master
 Key.";
            int origContentLength = origContent.length();
            encryptionClient.putObject(bucketName, keyName, origContent);

            // Download the object. The downloaded object is still encrypted.
            S3Object downloadedObject = encryptionClient.getObject(bucketName, keyName);
            S3ObjectInputStream input = downloadedObject.getObjectContent();

            // Decrypt and read the object and close the input stream.
            byte[] readBuffer = new byte[readChunkSize];
            ByteArrayOutputStream baos = new ByteArrayOutputStream(readChunkSize);
            int bytesRead = 0;
            int decryptedContentLength = 0;

            while ((bytesRead = input.read(readBuffer)) != -1) {
                baos.write(readBuffer, 0, bytesRead);
                decryptedContentLength += bytesRead;
            }
            input.close();

            // Verify that the original and decrypted contents are the same size.
            System.out.println("Original content length: " + origContentLength);
            System.out.println("Decrypted content length: " + decryptedContentLength);
        } 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();
        }
    }
}
```

옵션 2: 애플리케이션 내에 저장된 마스터 키 사용

이 단원에서는 클라이언트 측 데이터 암호화에 애플리케이션 내에 저장된 마스터 키를 사용하는 방법을 보여줍니다.

> **Important**
>
> 클라이언트 측 마스터 키와 암호화되지 않은 데이터는 AWS로 전송되지 않습니다. 따라서 암호화 키를 안전하게 관리하는 것이 중요합니다. 해당 키를 잃어버리면 데이터의 암호를 해독할 수 없습니다.

작동 방법은 다음과 같습니다.

- 객체를 업로드할 때Amazon S3 암호화 클라이언트에 클라이언트 측 마스터 키를 제공합니다. 클라이언트에서는 임의로 생성하는 데이터 암호화 키를 암호화하기 위해서만 이 마스터 키를 사용합니다. 프로세스는 다음과 같습니다.
  1. Amazon S3 암호화 클라이언트에서 일회용 대칭 키(즉 데이터 암호화 키 또는 데이터 키)를 로컬로 생성합니다. 또한 데이터 키를 사용하여 단일 객체의 데이터를 암호화합니다. 클라이언트는 각 객체에 대한 개별 데이터 키를 생성합니다.
  2. 클라이언트에서 사용자가 제공하는 마스터 키를 사용하여 데이터 암호화 키를 암호화합니다. 클라이언트에서 암호화된 데이터 키 및 해당 구성 요소 설명을 객체 메타데이터의 일부로 업로드합니다. 클라이언트는 구성 요소 설명을 사용하여 어떤 클라이언트 측 마스터 키를 암호화에 사용할지 결정합니다.
  3. 그런 다음 클라이언트에서 기본적으로 암호화된 데이터를 Amazon S3로 업로드하고 암호화된 데이터 키를 객체 메타데이터(`x-amz-meta-x-amz-key`)로 Amazon S3에 저장합니다.
- 객체를 다운로드할 때클라이언트가 먼저 Amazon S3에서 암호화된 객체를 다운로드합니다. 클라이언트에서 먼저 객체 메타데이터의 구성 요소 설명을 사용하여 데이터 키의 암호 해독에 어떤 마스터 키를 사용할지를 결정합니다. 클라이언트는 해당 마스터 키를 사용하여 데이터 키를 암호 해독하고 나서 데이터 키를 사용하여 객체를 암호 해독합니다.

사용자가 제공하는 클라이언트 측 마스터 키는 대칭 키 또는 퍼블릭/프라이빗 키 페어일 수 있습니다. 다음 예제에서는 두 가지 유형의 키를 사용하는 방법을 보여줍니다.

자세한 내용은 AWS SDK for Java 및 Amazon S3를 사용하는 클라이언트 측 암호화를 참조하십시오.

> **Note**
>
> 처음으로 암호화 API를 사용할 때 암호 암호화 오류 메시지가 표시되는 경우 사용 중인 버전의 JDK에 암호화 및 암호 해독 변환에 대한 최대 키 길이를 128비트로 제한하는 Java Cryptography Extension(JCE) Jurisdiction Policy File이 있을 수 있습니다. AWS SDK에서는 최대 키 길이가 256비트여야 합니다. 최대 키 길이를 확인하려면 `getMaxAllowedKeyLength()` 클래스의 `javax.crypto.Cipher` 메서드를 사용하십시오. 키 길이 제한을 제거하려면 Java Cryptography Extension(JCE) Unlimited Strength Jurisdiction Policy File을 Java SE 다운로드 페이지에서 설치하십시오.

Example

다음 예에서는 이러한 작업을 수행하는 방법을 보여줍니다.

- 256비트 AES 키 생성
- AES 키를 저장한 다음 파일 시스템에/에서 로드
- AES 키를 사용하여 Amazon S3에 데이터를 보내기 전에 클라이언트 측에서 데이터 암호화
- AES 키를 사용하여 Amazon S3에서 가져온 데이터 암호 해독
- 암호 해독된 데이터가 원래 데이터와 동일한지 확인

실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
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.AmazonS3EncryptionClientBuilder;
import com.amazonaws.services.s3.model.*;

import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.io.*;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.X509EncodedKeySpec;

public class S3ClientSideEncryptionSymMasterKey {

    public static void main(String[] args) throws Exception {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String objectKeyName = "*** Object key name ***";
        String masterKeyDir = System.getProperty("java.io.tmpdir");
        String masterKeyName = "secret.key";

        // Generate a symmetric 256-bit AES key.
        KeyGenerator symKeyGenerator = KeyGenerator.getInstance("AES");
        symKeyGenerator.init(256);
        SecretKey symKey = symKeyGenerator.generateKey();

        // To see how it works, save and load the key to and from the file system.
        saveSymmetricKey(masterKeyDir, masterKeyName, symKey);
        symKey = loadSymmetricAESKey(masterKeyDir, masterKeyName, "AES");

        try {
            // Create the Amazon S3 encryption client.
            EncryptionMaterials encryptionMaterials = new EncryptionMaterials(symKey);
            AmazonS3 s3EncryptionClient = AmazonS3EncryptionClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withEncryptionMaterials(new
 StaticEncryptionMaterialsProvider(encryptionMaterials))
                    .withRegion(clientRegion)
                    .build();

            // Upload a new object. The encryption client automatically encrypts it.
            byte[] plaintext = "S3 Object Encrypted Using Client-Side Symmetric Master
 Key.".getBytes();
            s3EncryptionClient.putObject(new PutObjectRequest(bucketName,
                    objectKeyName,
                    new ByteArrayInputStream(plaintext),
                    new ObjectMetadata()));
```

```
            // Download and decrypt the object.
            S3Object downloadedObject = s3EncryptionClient.getObject(bucketName,
 objectKeyName);
            byte[] decrypted =
 com.amazonaws.util.IOUtils.toByteArray(downloadedObject.getObjectContent());

            // Verify that the data that you downloaded is the same as the original data.
            System.out.println("Plaintext: " + new String(plaintext));
            System.out.println("Decrypted text: " + new String(decrypted));
        } 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();
        }
    }

    private static void saveSymmetricKey(String masterKeyDir, String masterKeyName,
 SecretKey secretKey) throws IOException {
        X509EncodedKeySpec x509EncodedKeySpec = new
 X509EncodedKeySpec(secretKey.getEncoded());
        FileOutputStream keyOutputStream = new FileOutputStream(masterKeyDir +
 File.separator + masterKeyName);
        keyOutputStream.write(x509EncodedKeySpec.getEncoded());
        keyOutputStream.close();
    }

    private static SecretKey loadSymmetricAESKey(String masterKeyDir, String masterKeyName,
 String algorithm)
            throws IOException, NoSuchAlgorithmException, InvalidKeySpecException,
 InvalidKeyException {
        // Read the key from the specified file.
        File keyFile = new File(masterKeyDir + File.separator + masterKeyName);
        FileInputStream keyInputStream = new FileInputStream(keyFile);
        byte[] encodedPrivateKey = new byte[(int) keyFile.length()];
        keyInputStream.read(encodedPrivateKey);
        keyInputStream.close();

        // Reconstruct and return the master key.
        return new SecretKeySpec(encodedPrivateKey, "AES");
    }
}
```

다음 예에서는 이러한 작업을 수행하는 방법을 보여줍니다.

- 1024비트 RSA 키 페어를 생성합니다.
- RSA 키를 저장한 다음 파일 시스템에/에서 로드합니다.
- RSA 키를 사용하여 Amazon S3에 데이터를 보내기 전에 클라이언트 측에서 데이터를 암호화합니다.
- RSA 키를 사용하여 Amazon S3에서 가져온 데이터 암호를 해독합니다.
- 암호 해독된 데이터가 원래 데이터와 동일한지 확인합니다.

실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
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.AmazonS3EncryptionClientBuilder;
import com.amazonaws.services.s3.model.*;
import com.amazonaws.util.IOUtils;

import java.io.*;
import java.security.*;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;

public class S3ClientSideEncryptionAsymmetricMasterKey {

    public static void main(String[] args) throws Exception {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String objectKeyName = "*** Key name ***";
        String rsaKeyDir = System.getProperty("java.io.tmpdir");
        String publicKeyName = "public.key";
        String privateKeyName = "private.key";

        // Generate a 1024-bit RSA key pair.
        KeyPairGenerator keyGenerator = KeyPairGenerator.getInstance("RSA");
        keyGenerator.initialize(1024, new SecureRandom());
        KeyPair origKeyPair = keyGenerator.generateKeyPair();

        // To see how it works, save and load the key pair to and from the file system.
        saveKeyPair(rsaKeyDir, publicKeyName, privateKeyName, origKeyPair);
        KeyPair keyPair = loadKeyPair(rsaKeyDir, publicKeyName, privateKeyName, "RSA");

        try {
            // Create the encryption client.
            EncryptionMaterials encryptionMaterials = new EncryptionMaterials(keyPair);
            AmazonS3 s3EncryptionClient = AmazonS3EncryptionClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withEncryptionMaterials(new
 StaticEncryptionMaterialsProvider(encryptionMaterials))
                    .withRegion(clientRegion)
                    .build();

            // Create a new object.
            byte[] plaintext = "S3 Object Encrypted Using Client-Side Asymmetric Master
 Key.".getBytes();
            S3Object object = new S3Object();
            object.setKey(objectKeyName);
            object.setObjectContent(new ByteArrayInputStream(plaintext));
            ObjectMetadata metadata = new ObjectMetadata();
            metadata.setContentLength(plaintext.length);

            // Upload the object. The encryption client automatically encrypts it.
            PutObjectRequest putRequest = new PutObjectRequest(bucketName,
                    object.getKey(),
                    object.getObjectContent(),
                    metadata);
            s3EncryptionClient.putObject(putRequest);

            // Download and decrypt the object.
            S3Object downloadedObject = s3EncryptionClient.getObject(bucketName,
 object.getKey());
            byte[] decrypted = IOUtils.toByteArray(downloadedObject.getObjectContent());

            // Verify that the data that you downloaded is the same as the original data.
            System.out.println("Plaintext: " + new String(plaintext));
```

```
            System.out.println("Decrypted text: " + new String(decrypted));
        } 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();
        }
    }

    private static void saveKeyPair(String dir,
                                    String publicKeyName,
                                    String privateKeyName,
                                    KeyPair keyPair) throws IOException {
        PrivateKey privateKey = keyPair.getPrivate();
        PublicKey publicKey = keyPair.getPublic();

        // Write the public key to the specified file.
        X509EncodedKeySpec x509EncodedKeySpec = new
 X509EncodedKeySpec(publicKey.getEncoded());
        FileOutputStream publicKeyOutputStream = new FileOutputStream(dir + File.separator
 + publicKeyName);
        publicKeyOutputStream.write(x509EncodedKeySpec.getEncoded());
        publicKeyOutputStream.close();

        // Write the private key to the specified file.
        PKCS8EncodedKeySpec pkcs8EncodedKeySpec = new
 PKCS8EncodedKeySpec(privateKey.getEncoded());
        FileOutputStream privateKeyOutputStream = new FileOutputStream(dir + File.separator
 + privateKeyName);
        privateKeyOutputStream.write(pkcs8EncodedKeySpec.getEncoded());
        privateKeyOutputStream.close();
    }

    private static KeyPair loadKeyPair(String dir,
                                       String publicKeyName,
                                       String privateKeyName,
                                       String algorithm)
            throws IOException, NoSuchAlgorithmException, InvalidKeySpecException {
        // Read the public key from the specified file.
        File publicKeyFile = new File(dir + File.separator + publicKeyName);
        FileInputStream publicKeyInputStream = new FileInputStream(publicKeyFile);
        byte[] encodedPublicKey = new byte[(int) publicKeyFile.length()];
        publicKeyInputStream.read(encodedPublicKey);
        publicKeyInputStream.close();

        // Read the private key from the specified file.
        File privateKeyFile = new File(dir + File.separator + privateKeyName);
        FileInputStream privateKeyInputStream = new FileInputStream(privateKeyFile);
        byte[] encodedPrivateKey = new byte[(int) privateKeyFile.length()];
        privateKeyInputStream.read(encodedPrivateKey);
        privateKeyInputStream.close();

        // Convert the keys into a key pair.
        KeyFactory keyFactory = KeyFactory.getInstance(algorithm);
        X509EncodedKeySpec publicKeySpec = new X509EncodedKeySpec(encodedPublicKey);
        PublicKey publicKey = keyFactory.generatePublic(publicKeySpec);

        PKCS8EncodedKeySpec privateKeySpec = new PKCS8EncodedKeySpec(encodedPrivateKey);
        PrivateKey privateKey = keyFactory.generatePrivate(privateKeySpec);

        return new KeyPair(publicKey, privateKey);
    }
}
```

Amazon S3의 Identity and Access Management

기본적으로 버킷, 객체 및 관련 하위 리소스(예: `lifecycle` 구성 및 `website` 구성)를 비롯한 모든 Amazon S3 리소스는 비공개입니다. 즉, 리소스를 만든 AWS 계정인 리소스 소유자만 해당 리소스에 액세스할 수 있습니다. 리소스 소유자는 액세스 정책을 작성하여 다른 사람에게 액세스 권한을 부여할 수도 있습니다.

Amazon S3에서는 리소스 기반 정책과 사용자 정책으로 대략 분류된 액세스 정책 옵션을 제공합니다. 리소스(버킷 및 객체)에 연결하는 액세스 정책을 리소스 기반 정책이라고 합니다. 예를 들어, 버킷 정책 및 ACL(액세스 통제 목록)은 리소스 기반 정책입니다. 본인의 계정에 속한 사용자에게 액세스 정책을 연결할 수도 있습니다. 이를 사용자 정책이라고 합니다. 리소스 기반 정책, 사용자 정책 또는 이러한 정책의 조합을 사용하도록 선택하여 Amazon S3 리소스에 대한 권한을 관리할 수 있습니다. 소개 주제에서 권한 관리에 대한 일반적인 지침을 제공합니다.

Amazon S3 리소스에 대한 액세스 관리 소개

Amazon S3 리소스에 대한 액세스를 관리하는 옵션을 설명하는 소개 주제를 먼저 검토하는 것이 좋습니다.

- 액세스 관리 개요 (p. 299)
- Amazon S3에서 요청에 권한을 부여하는 방법 (p. 305)
- 제공되는 액세스 정책 옵션 사용 지침 (p. 311)
- 예제 안내: Amazon S3 리소스에 대한 액세스 관리 (p. 313)

다음과 같이 여러 보안 모범 사례에서 액세스 제어를 다루기도 합니다.

- Amazon S3 버킷에 대한 퍼블릭 액세스 방지
- 최소 권한 액세스 구현
- IAM 역할 사용
- Multi-Factor Authentication(MFA) 삭제 활성화
- 모든 Amazon S3 버킷 식별 및 감사
- AWS 보안 권고 사항 모니터링

Amazon S3 리소스 액세스 옵션

Amazon S3 리소스에 대한 액세스 관리에 대한 소개 주제를 검토한 후 다음 주제를 사용하여 특정 액세스 정책 옵션에 대한 자세한 내용을 확인할 수 있습니다.

- 버킷 정책 및 사용자 정책 사용 (p. 338)
- ACL을 사용한 액세스 관리 (p. 437)
- Amazon S3 퍼블릭 액세스 차단 사용 (p. 450)

액세스 관리 개요

권한을 부여하려면 권한을 부여 받을 사용자, 권한 대상이 되는 Amazon S3 리소스, 해당 리소스에 허용되는 특정 작업을 결정합니다.

주제

- Amazon S3 리소스: 버킷 및 객체 (p. 300)
- Amazon S3 버킷 및 객체 소유권 (p. 300)
- 리소스 작업 (p. 301)
- 리소스 액세스 관리 (p. 301)
- 어떤 액세스 제어 방법을 사용해야 합니까? (p. 304)
- 추가 정보 (p. 305)

Amazon S3 리소스: 버킷 및 객체

Amazon Web Services(AWS)에서 리소스는 작업할 수 있는 객체입니다. Amazon S3에서 버킷 및 객체가 리소스이며 둘 다 연결된 하위 리소스가 있습니다. 다음은 버킷 하위 리소스의 몇 가지 예입니다.

- `lifecycle` – – 수명 주기 구성 정보를 저장합니다(객체 수명 주기 관리 (p. 123) 참조).
- `website` – 버킷을 웹 사이트 호스팅용으로 구성하는 경우 웹 사이트 구성 정보를 저장합니다(Amazon S3 정적 웹 사이트 호스팅 (p. 548) 참조).
- `versioning` – 버전 관리 구성을 저장합니다(PUT 버킷 버전 관리 참조).
- `policy` 및 `acl`(액세스 제어 목록) – 버킷에 대한 액세스 권한 정보를 저장합니다.
- `cors`(cross-origin 리소스 공유) – 교차 원본 요청을 허용하는 버킷 구성을 지원합니다(CORS(Cross-Origin Resource Sharing) (p. 154) 참조).
- `logging` – Amazon S3에 요청해 버킷 액세스 로그를 저장할 수 있습니다.

다음은 객체 하위 리소스의 몇 가지 예입니다.

- `acl` – 객체에 대한 액세스 권한 목록을 저장합니다. 본 주제에서는 이 하위 리소스로 객체 권한을 관리하는 방법을 다룹니다(ACL을 사용한 액세스 관리 (p. 437) 참조).
- `restore` – 보관된 객체의 임시 복원을 지원합니다(POST 객체 복원 참조). 스토리지 클래스의 객체는 아카이브된 객체입니다. 객체에 액세스하려면, 우선 복원 요청을 해야 하며 이에 따라 아카이빙된 객체의 복사본이 복원됩니다. 요청 시 복원하려는 객체 복사본의 복원 일수를 지정합니다. 객체 아카이빙에 대한 자세한 정보는 객체 수명 주기 관리 (p. 123)을 참조하십시오.

Amazon S3 버킷 및 객체 소유권

버킷 및 객체는 Amazon S3 리소스입니다. 기본적으로 리소스 소유자만 이러한 리소스에 액세스할 수 있습니다. 리소스 소유자란 리소스를 생성한 AWS 계정을 말합니다. 예:

- 버킷 생성 및 객체 업로드에 사용하는 AWS 계정은 해당 리소스를 소유합니다.

- AWS Identity and Access Management(IAM) 사용자 또는 역할 자격 증명을 사용하여 객체를 업로드하는 경우 사용자 또는 역할이 속한 AWS 계정이 객체를 소유합니다.

- 버킷 소유자는 다른 AWS 계정(또는 다른 계정의 사용자)에 객체를 업로드할 수 있는 교차 계정 권한을 부여할 수 있습니다. 이 경우, 객체를 업로드하는 AWS 계정이 해당 객체의 소유자입니다. 버킷 소유자는 다른 계정 소유의 객체에 대한 권한이 없습니다. 단, 다음 경우는 예외입니다.
  - 버킷 소유자가 요금을 지불하는 경우. 버킷 소유자는 객체의 소유 여부와 관계 없이 모든 객체에 대한 액세스를 거부하거나 버킷에 있는 객체를 삭제할 수 있습니다.
  - 버킷 소유자는 객체 소유 여부에 관계 없이 모든 객체를 아카이빙하거나 아카이빙된 객체를 복원할 수 있습니다. 아카이빙이란 객체를 저장하는 데 사용하는 스토리지 클래스를 말합니다. 자세한 내용은 객체 수명 주기 관리 (p. 123) 단원을 참조하십시오.

소유권 및 인증 요청

버킷에 대한 모든 요청은 인증되었거나 인증되지 않았습니다. 인증된 요청에는 요청 발신자를 인증하는 서명값이 포함되어야 하며 인증되지 않은 요청에는 포함되지 않아야 합니다. 인증 요청에 대한 자세한 내용은 요청 만들기 (p. 10) 항목을 참조하십시오.

버킷 소유자는 인증되지 않은 요청을 허용할 수 있습니다. 예를 들어 인증되지 않은 PUT 객체 요청은 버킷에 퍼블릭 버킷 정책이 있거나 버킷 ACL가 모든 사용자 그룹 또는 익명 사용자에 대해 `WRITE` 또는 `FULL_CONTROL` 액세스 권한을 부여할 때 허용됩니다. 퍼블릭 버킷 정책 및 퍼블릭 ACL에 대한 자세한 내용은 "퍼블릭"의 의미 (p. 453) 항목을 참조하십시오.

모든 인증되지 않은 요청은 익명 사용자가 수행합니다. 이 사용자는 액세스 제어 목록(ACL)에서 특정 정식 사용자 ID `65a011a29cdf8ec533ec3d1ccaae921c`로 표시됩니다. 인증되지 않은 요청을 통해 객체가 버킷에 업로드되면 익명 사용자가 객체를 소유합니다. 기본 객체 ACL은 익명 사용자에게 객체 소유자로 `FULL_CONTROL`을 부여합니다. 따라서 Amazon S3은 인증되지 않은 요청으로 객체를 검색하거나 ACL을 수정할 수 있습니다.

익명 사용자가 객체를 수정하지 못하도록 하려면 익명 사용자가 버킷에 퍼블릭 쓰기를 허용하는 버킷 정책을 구현하지 않거나 익명 사용자가 버킷에 대한 쓰기 액세스를 허용하는 ACL을 사용하는 것이 좋습니다. Amazon S3 퍼블릭 액세스 차단을 사용하여 이 권장 동작을 적용할 수 있습니다.

퍼블릭 액세스 차단에 대한 자세한 내용은 Amazon S3 퍼블릭 액세스 차단 사용 (p. 450) 항목을 참조하십시오. ACL에 대한 자세한 내용은 ACL(액세스 통제 목록) 개요 (p. 438) 단원을 참조하십시오.

> **Important**
>
> AWS 계정의 루트 사용자 자격 증명을 사용하여 인증된 요청을 하지 않는 것이 좋습니다. 대신, IAM 사용자를 생성하고 해당 사용자에게 모든 액세스 권한을 부여합니다. 이러한 사용자를 관리자 사용자라고 합니다. AWS 계정 루트 사용자 자격 증명 대신 관리자 사용자 자격 증명을 사용하여 AWS와 상호 작용하고 버킷 생성, 사용자 생성, 권한 부여 등의 작업을 수행할 수 있습니다. 자세한 내용은 AWS 계정 루트 사용자 자격 증명과 IAM 사용자 자격 증명(AWS General Reference) 및 IAM 모범 사례(IAM 사용 설명서) 단원을 참조하십시오.

리소스 작업

Amazon S3는 Amazon S3 리소스를 처리하기 위한 작업을 제공합니다. 가능한 작업에 대한 자세한 정보는 Amazon Simple Storage Service API Reference의 Operations on Buckets 및 Operations on Objects 단원을 참조하십시오.

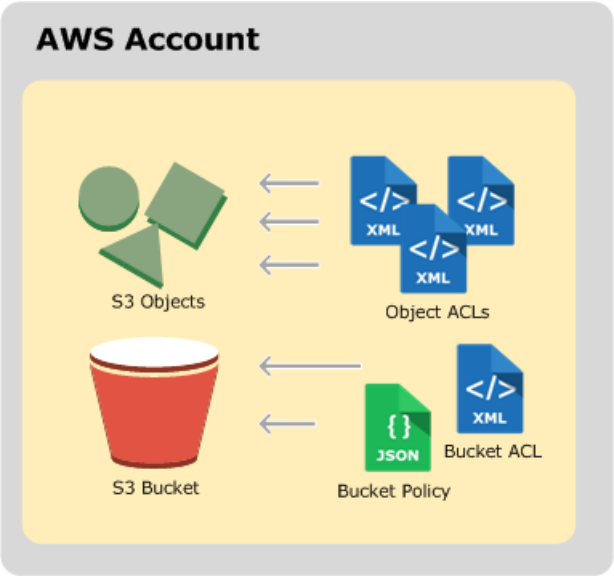
리소스 액세스 관리

액세스 관리란 액세스 정책을 만들어 제3자(AWS 계정 및 사용자)에게 리소스 작업을 수행할 권한을 부여하는 것을 말합니다. 예를 들어, AWS 계정의 사용자에게 `PUT Object` 권한을 부여하면 해당 사용자가 귀하의 버킷에 객체를 업로드할 수 있습니다. 개별 사용자와 계정에 대한 권한 부여 뿐만 아니라 전체(익명 액세스) 사용자와 인증된 모든 사용자(AWS 자격 증명이 있는 사용자)에게 권한을 부여할 수 있습니다. 예를 들어, 버킷을 웹사이트로 구성하는 경우 누구에게나 `GET Object` 권한을 부여해 객체를 공개할 수 있습니다.

액세스 정책 옵션

액세스 정책은 누가 무엇에 액세스 할 수 있는지를 나타냅니다. 액세스 정책은 리소스(버킷 및 객체) 또는 사용자와 연결할 수 있습니다. 이에 따른 사용 가능한 Amazon S3 액세스 정책을 다음과 같이 분류할 수 있습니다.

- 리소스 기반 정책 – 버킷 정책과 ACL(액세스 제어 목록)은 Amazon S3 리소스에 연결되는 리소스 기반입니다.



- ACL – 각 버킷과 객체마다 연결된 ACL이 있습니다. ACL은 피부여자와 그에 부여된 권한을 식별하는 권한 부여 목록입니다. ACL로 다른 AWS 계정에 기본적인 읽기/쓰기 권한을 부여합니다. ACL은 Amazon S3별 XML 스키마를 사용합니다.

  다음은 버킷 ACL의 예입니다. ACL 권한 부여는 완전한 제어 권한을 지닌 버킷 소유자를 나타냅니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
    <ID>*** Owner-Canonical-User-ID ***</ID>
    <DisplayName>owner-display-name</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:type="Canonical User">
        <ID>*** Owner-Canonical-User-ID ***</ID>
        <DisplayName>display-name</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

  버킷과 객체 ACL 모두 동일한 XML 스키마를 사용합니다.

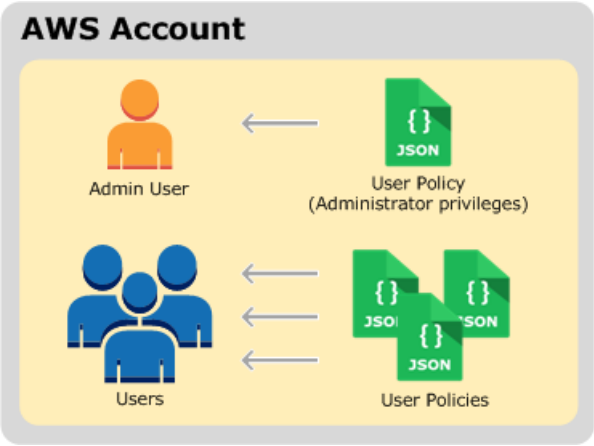
- 버킷 정책 – 본인의 버킷에 한해 다른 AWS 계정이나 IAM 사용자에게 버킷과 버킷에 포함된 객체에 대한 권한을 부여하는 버킷 정책을 추가할 수 있습니다. 모든 객체 권한은 해당 버킷 소유자가 생성하는 객체에만 적용됩니다. 버킷 정책은 ACL 기반 액세스 정책을 보완하며 대부분의 경우 액세스 정책을 대신합니다.

  다음은 버킷 정책의 예입니다. 버킷 정책(및 사용자 정책)은 JSON 파일로 나타냅니다. 이 정책은 버킷의 모든 객체에 대한 익명 읽기 권한을 허가합니다. 버킷 정책에 이름이 `s3:GetObject`인 버킷의 객체에 대해 `examplebucket` 작업(읽기 권한)을 허용한다는 설명이 있습니다. 해당 정책은 와일드카드(*)를 사용해 `principal`을 지정하여 익명 액세스를 허용하므로 신중하게 사용해야 합니다. 예를 들어 다음 버킷 정책은 누구나 객체에 액세스할 수 있도록 설정합니다.

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

```
    "Statement": [
        {
            "Effect":"Allow",
            "Principal": "*",
            "Action":["s3:GetObject"],
            "Resource":["arn:aws:s3:::examplebucket/*"]
        }
    ]
}
```

- 사용자 정책 – IAM로 Amazon S3 리소스에 대한 액세스를 관리할 수 있습니다. 내 계정으로 IAM 사용자, 그룹, 역할을 만들고 액세스 정책을 연결하면 등 AWS 리소스에 액세스할 수 있습니다.



IAM에 대한 자세한 정보는 AWS Identity and Access Management(IAM) 제품 세부 정보 페이지를 참조하십시오.

다음은 사용자 정책의 예입니다. IAM 사용자 정책은 사용자와 연결되므로 이 정책에서는 익명 권한을 부여할 수 없습니다. 예시로 든 정책은 연결된 사용자가 버킷과 버킷에 포함된 객체에 대해 6개의 다양한 Amazon S3 작업을 수행할 수 있도록 허용해 줍니다. 이 정책을 특정 IAM 사용자, 그룹 및 역할과 연결할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ExampleStatement1",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:DeleteObject",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                 "arn:aws:s3:::examplebucket/*",
                 "arn:aws:s3:::examplebucket"
            ]
        },
        {
            "Sid": "ExampleStatement2",
            "Effect": "Allow",
            "Action": "s3:ListAllMyBuckets",
            "Resource": "*"
        }
    ]
```

```
}
```

요청을 받으면 Amazon S3이 모든 액세스 정책을 평가해 권한 부여나 요청 거부를 결정합니다. Amazon S3의 해당 정책 평가에 대한 자세한 정보는 Amazon S3에서 요청에 권한을 부여하는 방법 (p. 305) 단원을 참조하십시오.

Access Analyzer for S3

Amazon S3 콘솔에서 Access Analyzer for S3를 사용하여 퍼블릭 액세스 권한 또는 공유 액세스 권한을 부여하는 버킷 액세스 제어 목록(ACL) 또는 버킷 정책이 있는 모든 버킷을 검토할 수 있습니다. Access Analyzer for S3는 인터넷상의 모든 사용자 또는 조직 외부의 AWS 계정을 포함한 다른 AWS 계정에 액세스를 허용하도록 구성된 버킷에 대한 알림을 제공합니다. 각 퍼블릭 버킷 또는 공유 버킷에 대해 퍼블릭 액세스 또는 공유 액세스의 수준과 원본을 보고하는 결과가 수신됩니다.

Access Analyzer for S3에서는 한 번의 클릭으로 버킷에 대한 모든 퍼블릭 액세스를 차단할 수 있습니다. 특정 사용 사례를 지원하기 위해 퍼블릭 액세스가 필요하지 않은 경우 버킷에 대한 모든 액세스를 차단하는 것이 좋습니다. 모든 퍼블릭 액세스를 차단하기 전에 애플리케이션이 퍼블릭 액세스 없이 계속 올바르게 작동하는지 확인하십시오. 자세한 내용은 Amazon S3 퍼블릭 액세스 차단 사용 (p. 450) 단원을 참조하십시오.

또한 버킷 수준 권한 설정으로 드릴다운하여 세부적인 액세스 수준을 구성할 수 있습니다. 퍼블릭 액세스 또는 공유 액세스가 필요한 것으로 확인된 특정 사용 사례의 경우, 버킷에 대한 결과를 보관하여 버킷을 퍼블릭 상태로 유지할 것인지 공유 상태로 유지할 것인지 확인하고 기록할 수 있습니다. 언제든지 재방문하여 해당 버킷 구성을 수정할 수 있습니다. 감사 목적으로 결과를 CSV 보고서로 다운로드할 수도 있습니다.

Access Analyzer for S3는 Amazon S3 콘솔에서 추가 비용 없이 사용할 수 있습니다. Access Analyzer for S3는 AWS Identity and Access Management(IAM)Access Analyzer에서 제공합니다. Amazon S3 콘솔에서 Access Analyzer for S3를 사용하려면 IAM 콘솔을 방문하여 리전별로 IAM Access Analyzer을 활성화해야 합니다.

Access Analyzer for S3에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 Access Analyzer for S3 사용을 참조하십시오.

어떤 액세스 제어 방법을 사용해야 합니까?

액세스 정책 생성 옵션을 사용하는 경우, 다음과 같은 질문을 합니다.

- 각 액세스 제어 방법에 따른 사용시기는 언제입니까? 예를 들어, 버킷 권한을 부여하려면 버킷 정책과 버킷 ACL 중 어느 것을 사용해야 합니까? 나는 1개의 버킷과 그 버킷에 든 객체를 소유하고 있습니다. 리소스 기반 액세스 정책과 IAM 사용자 정책 중 어느 것을 사용해야 합니까? 리소스 기반 액세스 정책을 사용하는 경우, 객체 권한 관리에 버킷 정책과 객체 ACL 중 어느 것을 사용해야 합니까?
- 1개의 버킷을 소유하고 있더라도 그 버킷에 있는 모든 객체를 소유한 것은 아닙니다. 다른 사람이 소유하고 있는 객체에 대한 액세스 권한은 어떻게 관리합니까?
- 이러한 액세스 정책 옵션을 결합하여 액세스를 허용하는 경우, Amazon S3은(는) 사용자에게 요청 작업 수행 권한이 있는지 여부를 어떻게 판단합니까?

다음 단원에서는 이러한 액세스 제어 옵션과 Amazon S3의 액세스 제어 방식 평가 방법, 액세스 제어 방법에 따른 사용 시기를 살펴봅니다. 또한 실습 예제도 제시합니다.

- Amazon S3에서 요청에 권한을 부여하는 방법 (p. 305)
- 제공되는 액세스 정책 옵션 사용 지침 (p. 311)
- 예제 안내: Amazon S3 리소스에 대한 액세스 관리 (p. 313)

추가 정보

Amazon S3 리소스 액세스 관리에 쓰이는 옵션에 대한 주제별 설명을 우선 읽어 보는 것이 좋습니다. 자세한 내용은 Amazon S3의 Identity and Access Management (p. 299) 단원을 참조하십시오. 그 후, 다음과 같이 주제별로 특정 액세스 정책 옵션에 대한 자세한 정보를 얻을 수 있습니다.

- 버킷 정책 및 사용자 정책 사용 (p. 338)
- ACL을 사용한 액세스 관리 (p. 437)

Amazon S3에서 요청에 권한을 부여하는 방법

주제

- 관련 주제 (p. 306)
- Amazon S3에서 버킷 작업 요청에 권한을 부여하는 방법 (p. 306)
- Amazon S3에서 객체 작업에 대한 요청에 권한을 부여하는 방법 (p. 309)

Amazon S3가 요청을 수신하면(예: 버킷 또는 객체 작업) 먼저 요청자가 필요한 권한을 보유하고 있는지 확인합니다. Amazon S3는 요청에 권한을 부여할지 여부를 결정할 때 모든 관련 액세스 정책, 사용자 정책 및 리소스 기반 정책(버킷 정책, 버킷 ACL, 객체 ACL)을 평가합니다. 다음은 예제 시나리오의 일부입니다.

- 요청자가 IAM 보안 주체일 경우, Amazon S3에서는 보안 주체가 속한 상위 AWS 계정에서 작업 수행에 필요한 권한을 보안 주체에게 부여했는지를 확인해야 합니다. 또한 버킷 콘텐츠 나열 요청과 같은 버킷 작업에 대한 요청일 경우, Amazon S3에서는 버킷 소유자가 요청자가 작업을 수행할 수 있도록 권한을 부여했는지를 확인해야 합니다.

  Note

  리소스에서 특정 작업을 수행하려면 IAM 보안 주체는 소속된 상위 AWS 계정과 리소스를 소유한 AWS 계정에서 모두 권한이 필요합니다.
- 버킷 소유자가 소유하지 않은 객체 작업에 대한 요청일 경우, 요청자가 객체 소유자의 권한을 보유하고 있는지 확인해야 할 뿐만 아니라 Amazon S3은 버킷 소유자가 객체에 대한 명백한 거부를 설정하지 않도록 버킷 정책을 확인해야 합니다.

  Note

  버킷 소유자(요금 지불자)는 객체의 소유 여부와 관계 없이 버킷의 객체에 대한 액세스를 명백하게 거부할 수 있습니다. 또한 버킷 소유자는 버킷의 모든 객체를 삭제할 수도 있습니다.

요청자가 특정 작업을 수행할 권한을 보유하고 있는지 확인하기 위해 Amazon S3은 요청을 수신할 경우 다음을 순서대로 수행합니다.

1. 런타임 시 모든 관련 액세스 정책(사용자 정책, 버킷 정책, ACL)을 평가용 정책 세트로 변환합니다.
2. 다음과 같은 단계로 발생한 정책 세트를 평가합니다. 각 단계에서 Amazon S3은 컨텍스트 권한에 기반해 특정 컨텍스트에서 하위 정책 세트를 평가합니다.
   a. 사용자 컨텍스트 – 사용자 컨텍스트에서 사용자가 속한 상위 계정은 컨텍스트 권한입니다.

      Amazon S3은 상위 계정이 소유한 하위 정책 세트를 평가합니다. 이러한 하위 세트에는 상위 계정에서 사용자에게 연결된 사용자 정책이 포함됩니다. 상위 계정에서 요청 시 리소스(버킷, 객체)를 소유할 경우에도, Amazon S3은 동시에 해당 리소스 정책(버킷 정책, 버킷 ACL, 객체 ACL)도 평가합니다.

      사용자는 작업 수행을 위해 상위 계정의 권한을 보유해야 합니다.

      이 단계는 AWS 계정의 사용자가 요청한 경우에만 적용됩니다. AWS 계정의 루트 자격 증명을 사용하여 요청이 이루어진 경우, Amazon S3은 이 단계를 건너뜁니다.

b. 버킷 컨텍스트 – 버킷 컨텍스트에서 Amazon S3은 버킷을 소유한 AWS 계정이 소유한 정책을 평가합니다.

   버킷 작업에 대한 요청의 경우, 요청자는 버킷 소유자의 권한을 보유해야 합니다. 객체에 대한 요청의 경우, Amazon S3은 버킷 소유자가 객체에 대한 액세스를 명백하게 거부했는지 확인하기 위해 버킷 소유자가 소유한 모든 정책을 평가합니다. 명시적 거부가 있을 경우 Amazon S3은 요청에 권한을 부여하지 않습니다.

c. 객체 컨텍스트 – 객체에 대한 요청의 경우, Amazon S3은 객체 소유자가 소유한 하위 정책 세트를 평가합니다.

다음 단원에서는 상세한 설명과 예를 제공합니다.

- Amazon S3에서 버킷 작업 요청에 권한을 부여하는 방법 (p. 306)
- Amazon S3에서 객체 작업에 대한 요청에 권한을 부여하는 방법 (p. 309)

관련 주제

Amazon S3 리소스에 대한 액세스를 관리하는 옵션을 설명하는 소개 주제를 먼저 검토하는 것이 좋습니다. 자세한 내용은 Amazon S3 리소스에 대한 액세스 관리 소개 (p. 299) 단원을 참조하십시오. 그 후, 다음과 같이 주제별로 특정 액세스 정책 옵션에 대한 자세한 정보를 얻을 수 있습니다.

- 버킷 정책 및 사용자 정책 사용 (p. 338)
- ACL을 사용한 액세스 관리 (p. 437)

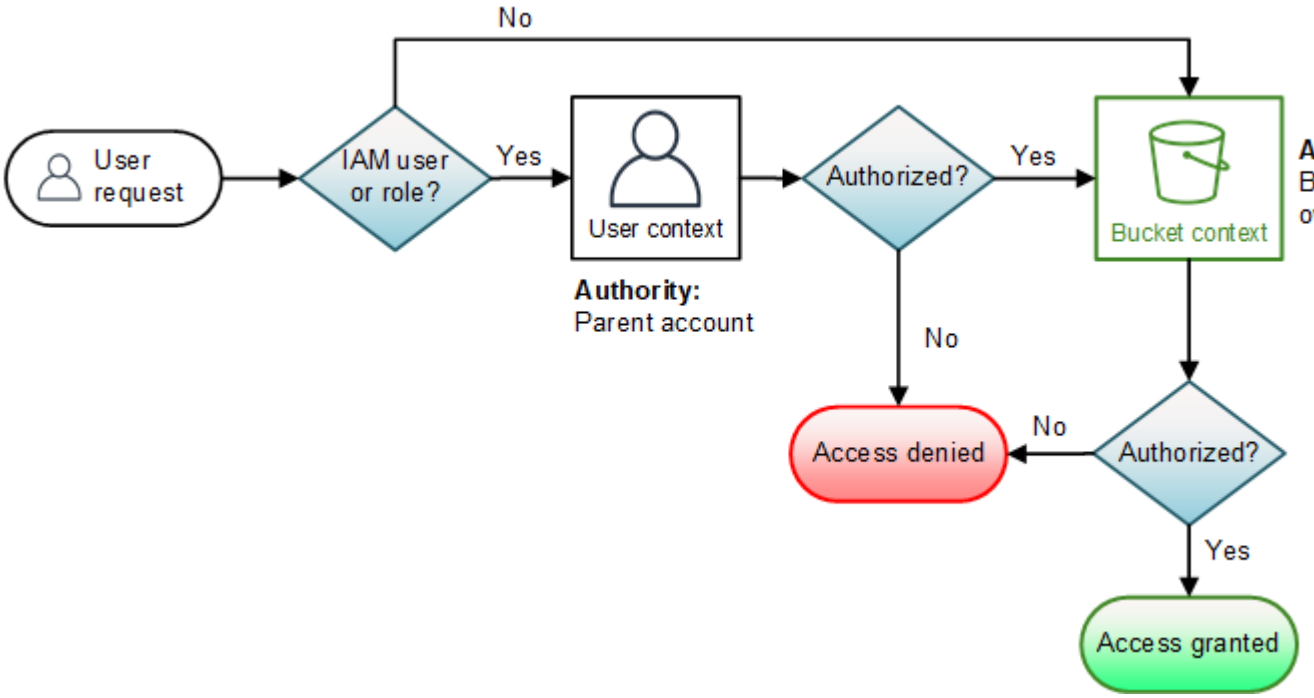
Amazon S3에서 버킷 작업 요청에 권한을 부여하는 방법

Amazon S3는 버킷 작업 요청을 수신하면 모든 관련 권한(리소스 기반 권한(버킷 정책, 버킷 액세스 제어 목록(ACL)) 및 해당 요청이 IAM 보안 주체로부터 수신된 경우 IAM 사용자 정책)을 런타임 시 평가할 정책 세트로 변환합니다. 그런 다음 특정 컨텍스트(사용자 컨텍스트 또는 버킷 컨텍스트)에 따라 일련의 단계에서 발생한 정책 세트를 평가합니다.

1. 사용자 컨텍스트 – 요청자가 IAM 보안 주체일 경우, 이 보안 주체는 소속된 상위 AWS 계정의 권한을 보유해야 합니다. 이 단계에서 Amazon S3은 상위 계정(또한 컨텍스트 권한으로 불림)이 소유한 하위 정책 세트를 평가합니다. 이러한 하위 정책 세트에는 상위 계정이 보안 주체에게 연결한 사용자 정책이 포함됩니다. 또한 상위 계정에서 요청 시 리소스를 소유한 경우(이 경우에는 버킷), Amazon S3은 동시에 해당 리소스 정책(버킷 정책, 버킷 ACL)을 평가합니다. 버킷 작업에 대한 요청이 만들어질 때마다 서버 액세스 로그에 요청자의 정식 ID가 기록됩니다. 자세한 내용은 Amazon S3 서버 액세스 로깅 (p. 705) 단원을 참조하십시오.
2. 버킷 컨텍스트 – 요청자는 특정 버킷 작업을 수행할 수 있는 버킷 소유자의 권한을 보유해야 합니다. 이 단계에서 Amazon S3은 버킷을 소유한 AWS 계정이 소유한 하위 정책 세트를 평가합니다.

   버킷 소유자는 버킷 정책 또는 버킷 ACL을 사용하여 권한을 부여할 수 있습니다. 버킷을 소유한 AWS 계정이 IAM 보안 주체의 상위 계정이기도 한 경우 사용자 정책에서 버킷 권한을 구성할 수 있습니다.

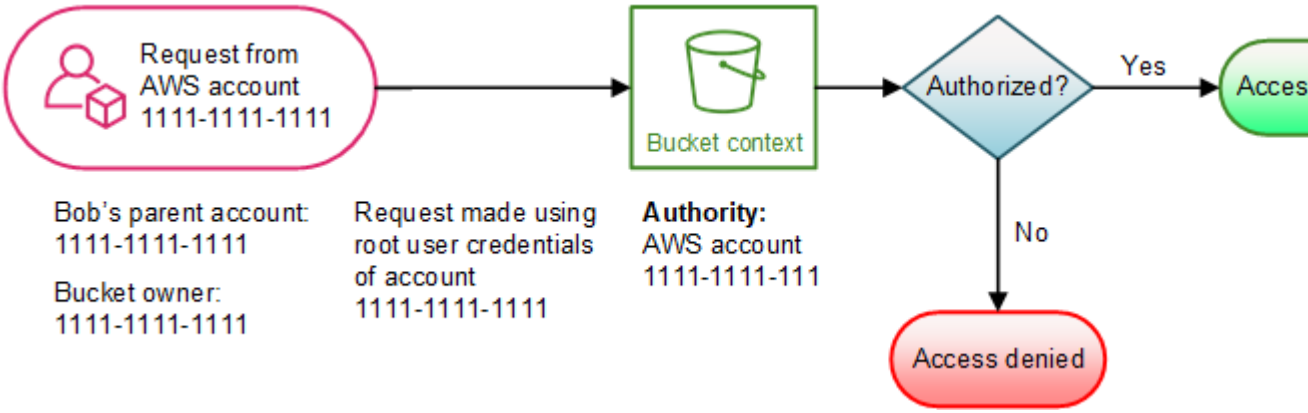
다음은 버킷 작업의 컨텍스트 기반 평가에 대한 그래픽을 이용한 그림입니다.



다음 예는 평가 논리를 설명합니다.

예 1: 버킷 소유자가 요청한 버킷 작업

이 예에서 버킷 소유자는 AWS 계정의 루트 자격 증명을 사용하여 버킷 작업을 위한 요청을 보냅니다.

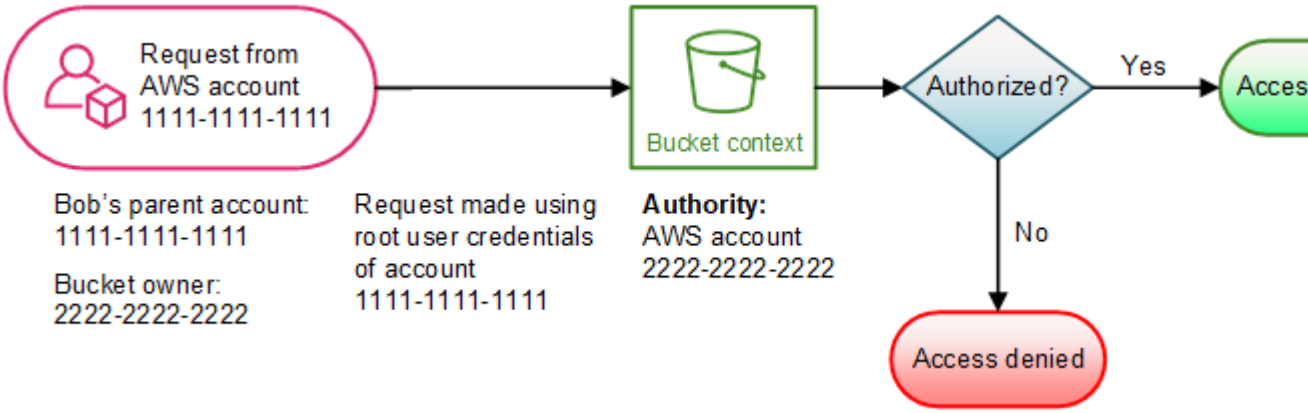


Amazon S3은 다음과 같이 컨텍스트 평가를 수행합니다.

1. AWS 계정의 루트 자격 증명을 사용하여 요청이 되므로 사용자 컨텍스트는 평가되지 않습니다.
2. 버킷 컨텍스트에서 Amazon S3은 요청자가 작업을 수행할 수 있는 권한을 보유하고 있는지 확인하기 위해 버킷 정책을 검토합니다. Amazon S3에서 요청에 권한을 부여합니다.

예 2: 버킷 소유자가 아닌 AWS 계정에서 요청한 버킷 작업

이 예에서는 AWS 계정 2222-2222-2222가 소유한 버킷 작업을 위해 AWS 계정 1111-1111-1111의 루트 자격 증명을 사용하여 요청이 이루어졌습니다. 이 요청과 관련된 어떤 IAM 사용자도 없습니다.

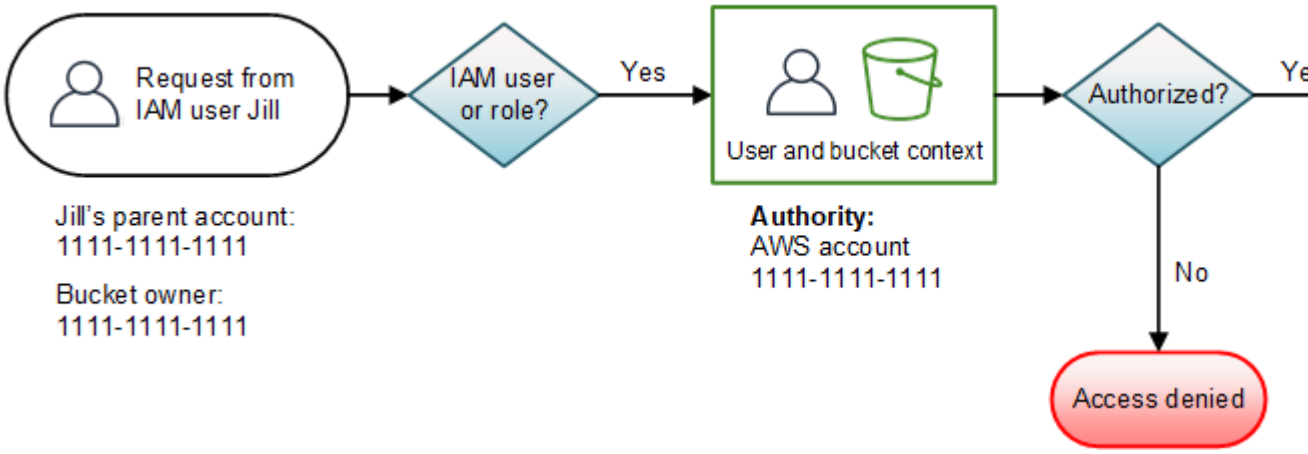


이 경우, Amazon S3은 다음과 같이 컨텍스트를 평가합니다.

1. AWS 계정의 루트 자격 증명을 사용하여 요청이 되므로 사용자 컨텍스트는 평가되지 않습니다.
2. 버킷 컨텍스트에서 Amazon S3은 버킷 정책을 검토합니다. 버킷 소유자(AWS 계정 2222-2222-2222)가 요청된 작업을 수행할 수 있도록 AWS 계정 1111-1111-1111에 권한을 부여하지 않은 경우, Amazon S3이 요청을 거부합니다. 그렇지 않으면 Amazon S3은 요청에 권한을 부여하고 해당 작업을 수행합니다.

예 3: 상위 AWS 계정이 버킷 소유자인 IAM 보안 주체가 요청한 버킷 작업

이 예에서는 또한 버킷을 소유하고 있는 AWS 계정 1111-1111-1111의 IAM 사용자인 Jill이 요청을 보냅니다.



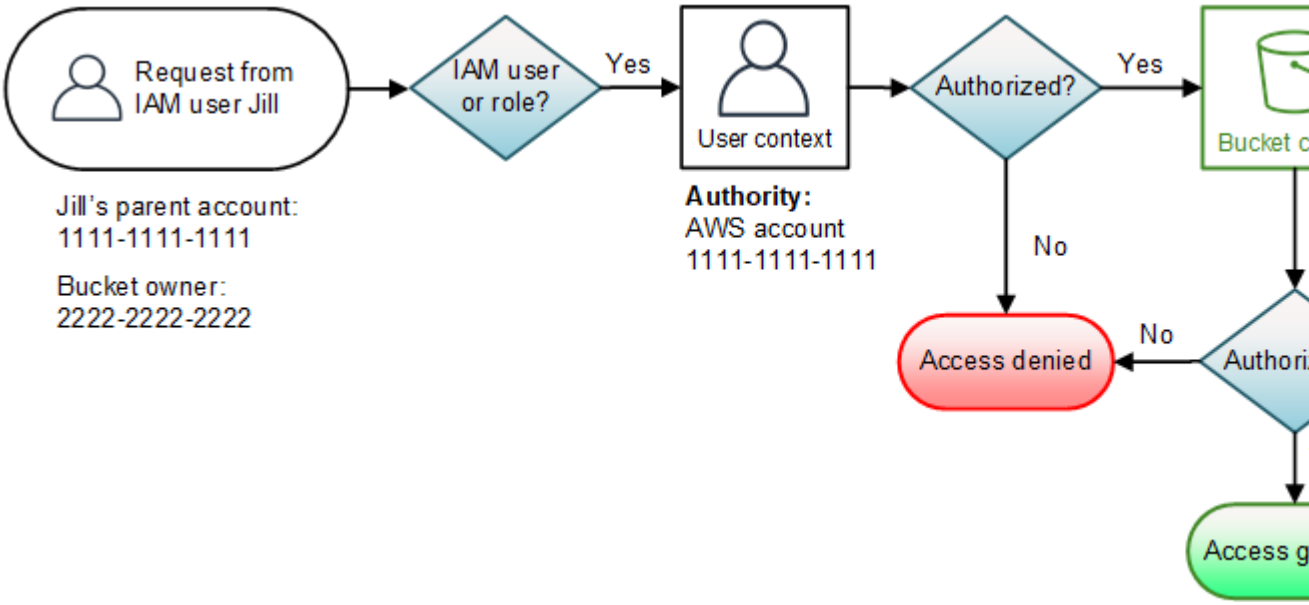
Amazon S3은 다음의 컨텍스트 평가를 수행합니다.

1. 사용자 컨텍스트에서 IAM 보안 주체가 요청을 하기 때문에 Amazon S3은 상위 AWS 계정에 속한 모든 정책을 평가하여 Jill이 작업을 수행할 권한을 보유하고 있는지 확인합니다.

   이 예에서는 보안 주체가 속한 상위 AWS 계정 1111-1111-1111도 버킷 소유자입니다. 따라서 사용자 정책 이외에도 Amazon S3에서는 이들이 동일한 계정에 속하기 때문에 동일한 컨텍스트에서 버킷 정책과 버킷 ACL을 평가합니다.
2. Amazon S3은 사용자 컨텍스트의 일부로 버킷 정책과 버킷 ACL을 평가했기 때문에 버킷 컨텍스트는 평가하지 않습니다.

예 4: 상위 AWS 계정이 버킷 소유자가 아닌 IAM 보안 주체가 요청한 버킷 작업

이 예에서는 상위 AWS 계정이 1111-1111-1111인 IAM 사용자인 Jill이 요청을 전송하지만 버킷은 다른 AWS 계정, 2222-2222-2222가 소유합니다.



Jill은 상위 AWS 계정과 버킷 소유자 모두의 권한이 필요합니다. Amazon S3은 다음과 같이 컨텍스트를 평가합니다.

1. IAM 보안 주체가 요청을 하기 때문에 Amazon S3은 Jill이 필요한 권한을 보유하고 있는지 확인하기 위해 계정에서 작성한 정책을 검토하여 사용자 컨텍스트를 평가합니다. Jill이 권한을 보유한 경우 Amazon S3은 계속하여 버킷 컨텍스트를 평가합니다. 그렇지 않은 경우 요청을 거부합니다.
2. 버킷 컨텍스트에서 Amazon S3은 요청된 작업을 수행하기 위해 버킷 소유자 2222-2222-2222가 Jill(또는 Jill의 상위 AWS 계정)에게 권한을 부여했는지를 확인합니다. Jill이 해당 권한을 보유한 경우 Amazon S3은 작업을 요청하고 수행합니다. 그렇지 않은 경우, Amazon S3은 요청을 거부합니다.

Amazon S3에서 객체 작업에 대한 요청에 권한을 부여하는 방법

Amazon S3는 객체 작업 요청을 수신하면 모든 관련 권한(리소스 기반 권한(객체 액세스 제어 목록(ACL), 버킷 정책, 버킷 ACL) 및 IAM 사용자 정책)을 런타임 시 평가할 정책 세트로 변환합니다. 그런 다음 일련의 단계에서 발생한 정책 세트를 평가합니다. 각 단계에서 Amazon S3은 세 개의 특정 컨텍스트(사용자 컨텍스트, 버킷 컨텍스트 및 객체 컨텍스트)의 하위 정책 세트를 평가합니다.

1. 사용자 컨텍스트 – 요청자가 IAM 보안 주체일 경우, 이 보안 주체는 소속된 상위 AWS 계정의 권한을 보유해야 합니다. 이 단계에서 Amazon S3은 상위 계정(또한 컨텍스트 권한을 불림)이 소유한 하위 정책 세트를 평가합니다. 이러한 하위 정책 세트에는 상위 계정이 보안 주체에게 연결한 사용자 정책이 포함됩니다. 상위 계정에서 요청 시 리소스(버킷, 객체)를 소유한 경우, Amazon S3은 동시에 해당 리소스 정책(버킷 정책, 버킷 ACL, 객체 ACL)을 평가합니다.

   Note

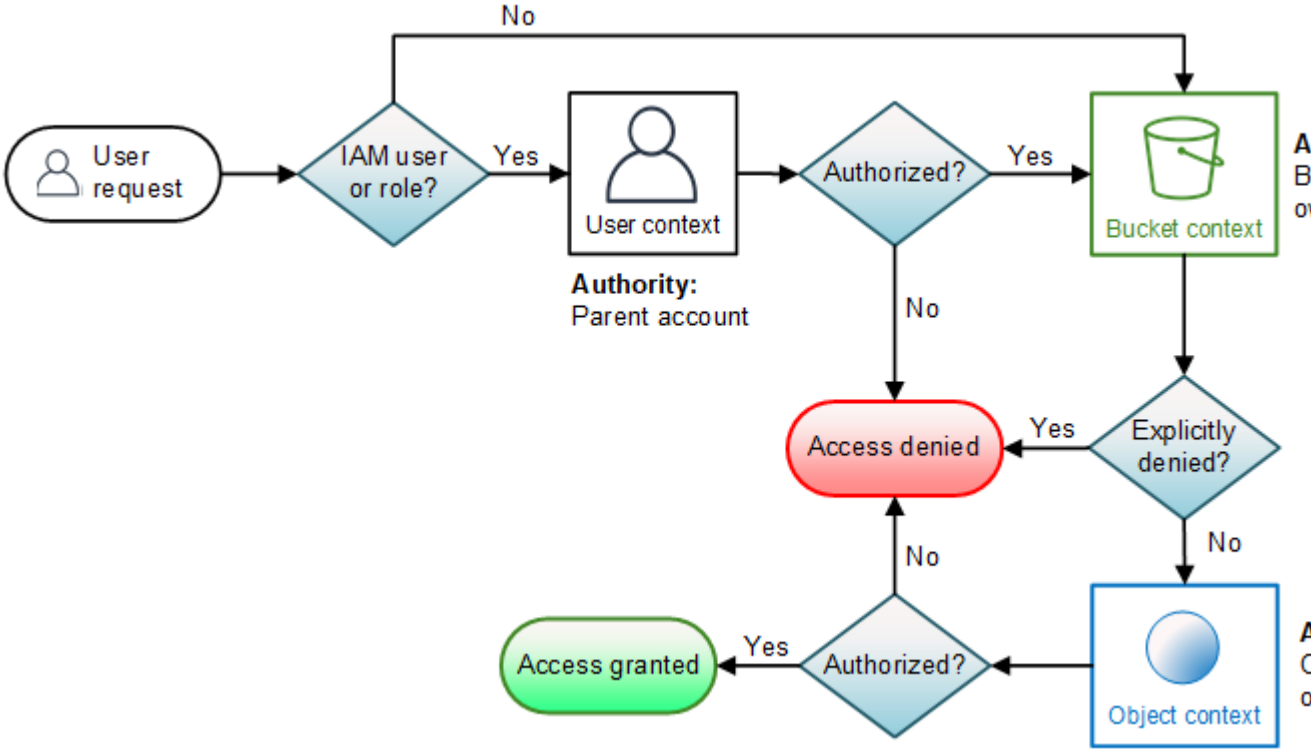
   상위 AWS 계정이 리소스(버킷 또는 객체)를 소유한 경우, 사용자 정책이나 리소스 정책 중 하나를 사용하여 IAM 보안 주체에게 리소스 권한을 부여할 수 있습니다.
2. 버킷 컨텍스트 – 이 컨텍스트에서 Amazon S3은 버킷을 소유한 AWS 계정이 소유한 정책을 평가합니다.

   요청 시 객체를 소유한 AWS 계정이 버킷 소유자와 동일하지 않으면, Amazon S3은 버킷 컨텍스트에서 버킷 소유자가 객체에 대한 액세스를 명백하게 거부했을 경우 정책을 평가합니다. 객체에 명시적으로 거부된 세트가 있는 경우 Amazon S3은 해당 요청에 권한을 부여하지 않습니다.
3. 객체 컨텍스트 – 요청자는 특정 객체 작업을 수행하기 위한 객체 소유자의 권한을 보유해야 합니다. 이 단계에서 Amazon S3은 ACL을 평가합니다.

Note

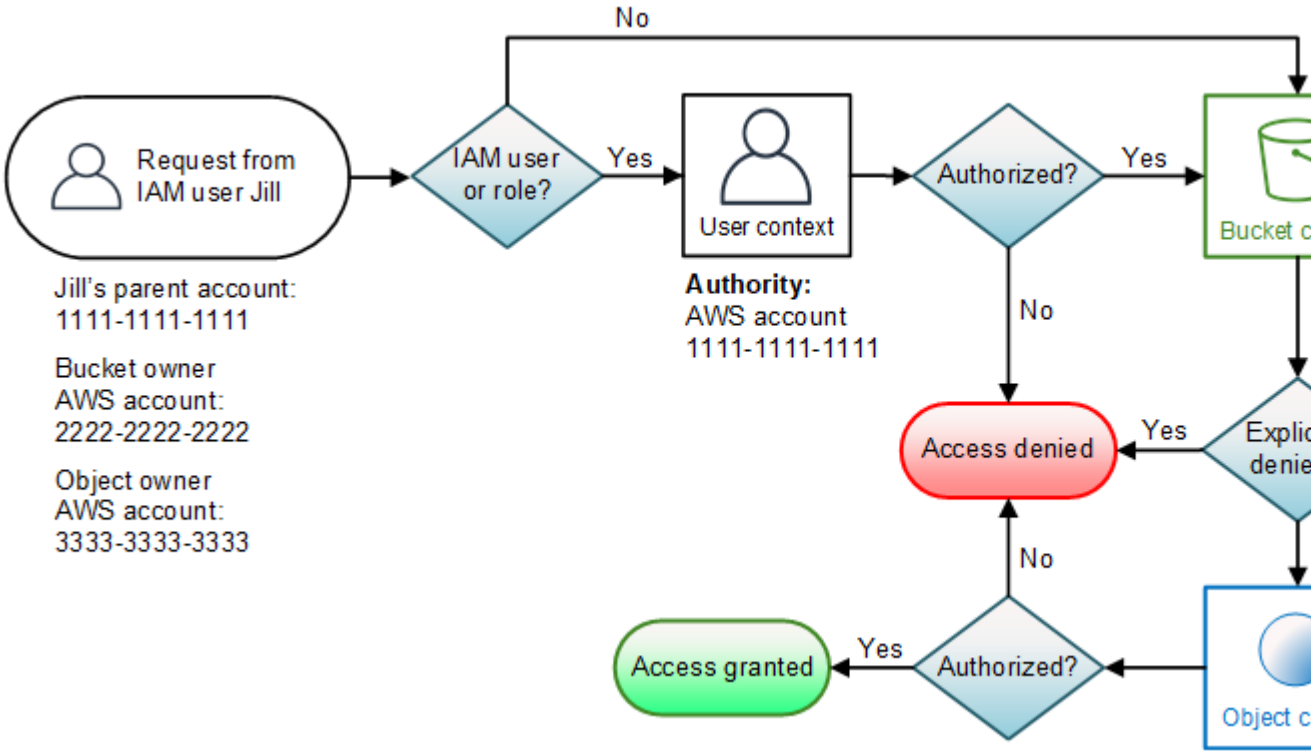
버킷과 객체 소유자가 같지 않은 경우 버킷 컨텍스트에서 평가되는 버킷 정책에서 객체에 대한 액세스를 부여할 수 있습니다. 소유자가 다르면 객체 소유자는 객체 ACL을 사용하여 권한을 부여해야 합니다. 객체를 소유한 AWS 계정이 IAM 보안 주체가 속한 상위 계정이기도 한 경우, 사용자 컨텍스트에서 평가되는 사용자 정책에서 객체 권한을 구성할 수 있습니다. 이러한 액세스 정책 대안을 사용하는 것에 대한 자세한 내용은 제공되는 액세스 정책 옵션 사용 지침 (p. 311)을 참조하십시오.

다음은 객체 작업의 컨텍스트 기반 평가에 대한 그림입니다.



예 1: 객체 작업 요청

이 예에서는 상위 AWS 계정이 1111-1111-1111인 IAM 사용자 Jill이 AWS 계정 2222-2222-2222가 소유한 버킷에서 AWS 계정 3333-3333-3333이 소유한 객체에 대해 객체 작업 요청(예: 객체 가져오기)을 전송합니다.



Jill은 상위 AWS 계정, 버킷 소유자 및 객체 소유자의 권한이 필요합니다. Amazon S3은 다음과 같이 컨텍스트를 평가합니다.

1. IAM 보안 주체가 요청을 하기 때문에 Amazon S3은 상위 AWS 계정 1111-1111-1111에서 요청된 작업 수행을 위해 Jill에게 권한을 부여했는지 확인하기 위해 사용자 컨텍스트를 평가합니다. Jill이 해당 권한을 보유한 경우 Amazon S3은 버킷 컨텍스트를 평가합니다. 그렇지 않으면 Amazon S3은 요청을 거부합니다.
2. 버킷 컨텍스트에서, 버킷 소유자 AWS 계정 2222-2222-2222는 컨텍스트 권한입니다. Amazon S3은 버킷 소유자가 객체에 대한 Jill의 액세스를 명백하게 거부했는지 확인하기 위해 버킷 정책을 검토합니다.
3. 객체 컨텍스트에서 컨텍스트 권한은 AWS 계정 3333-3333-3333인 객체 소유자입니다. Amazon S3은 Jill이 객체에 액세스하는 권한을 보유 중인지 확인하기 위해 객체 ACL을 평가합니다. 권한을 보유한 경우 Amazon S3은 요청에 권한을 부여합니다.

제공되는 액세스 정책 옵션 사용 지침

Amazon S3에서는 Amazon S3 리소스에 대한 액세스를 관리하기 위해 리소스 기반 정책 및 사용자 정책이 지원됩니다(리소스 액세스 관리 (p. 301) 참조). 리소스 기반 정책에는 버킷 정책, 버킷 ACL 및 객체 ACL이 포함됩니다. 이 단원에서는 Amazon S3 리소스에 대한 액세스를 관리하기 위해 리소스 기반 액세스 정책을 사용하기 위한 특정 시나리오에 대해 설명합니다.

ACL 기반 액세스 정책(버킷 및 객체 ACL)을 사용하는 경우

버킷과 객체 모두에 권한을 부여하는 데 사용할 수 있는 ACL이 연결되어 있습니다. 다음 단원에서는 객체 ACL 및 버킷 ACL을 사용하기 위한 시나리오에 대해 설명합니다.

객체 ACL을 사용하는 경우

객체 ACL 외에 다른 방법으로도 객체 소유자가 객체 권한을 관리할 수 있습니다. 다음 예를 참조하십시오.

- 객체를 소유하는 AWS 계정이 버킷도 소유하는 경우 객체 권한을 관리하기 위해 버킷 정책을 작성할 수 있습니다.
- 객체를 소유하는 AWS 계정이 자신의 계정에 속한 사용자에게 권한을 부여하려는 경우에는 사용자 정책을 사용할 수 있습니다.

그러면 객체 ACL을 사용하여 객체 권한을 관리하는 것은 어떤 경우일까요? 다음은 객체 ACL을 사용하여 객체 권한을 관리하는 시나리오입니다.

- 객체 ACL이 버킷 소유자가 소유하지 않는 객체에 대한 액세스를 관리하는 유일한 방법임 – 버킷을 소유하는 AWS 계정은 객체를 업로드하기 위해 다른 AWS 계정 권한을 부여할 수 있습니다. 버킷 소유자는 이러한 객체를 소유하지 않습니다. 객체를 만든 AWS 계정은 객체 ACL을 사용하여 권한을 부여해야 합니다.

  Note

  버킷 소유자는 자신이 소유하지 않는 객체에 대한 권한을 부여할 수 없습니다. 예를 들어, 객체 권한을 부여하는 버킷 정책은 버킷 소유자가 소유하는 객체에만 적용됩니다. 그러나 요금을 지불하는 버킷 소유자는 소유자와 관계없이 버킷의 모든 객체에 대한 액세스를 거부하는 버킷 정책을 작성할 수 있습니다. 또한 버킷 소유자는 버킷의 모든 객체를 삭제할 수 있습니다.
- 권한이 객체별로 다르며 객체 수준에서 권한을 관리해야 함 – 특정 키 이름 접두사가 지정된 수백만 개의 객체에 대한 읽기 권한을 AWS 계정에 부여하는 단일 정책 설명을 작성할 수 있습니다. 예를 들어, 키 이름 접두사 "logs"로 시작하는 객체에 대해 읽기 권한을 부여합니다. 그러나 액세스 권한이 객체별로 다른 경우 버킷 정책을 사용하여 개별 객체에 권한을 부여하는 것은 실용적이지 않을 수 있습니다. 또한 버킷 정책은 크기가 20KB로 제한됩니다.

  이 경우 객체 ACL을 사용하는 것이 적절한 대안이 될 수 있습니다. 그러나 객체 ACL도 최대 100건의 허가로 제한됩니다(ACL(액세스 통제 목록) 개요 (p. 438) 참조).
- 객체 ACL이 객체 수준 권한만 제어함 – 전체 버킷에 대한 단일 버킷 정책이 있지만 객체 ACL은 객체별로 지정됩니다.

  버킷을 소유하는 AWS 계정은 액세스 정책을 관리하기 위한 권한을 다른 AWS 계정에 부여할 수 있습니다. 해당 계정은 정책의 모든 사항을 변경할 수 있게 됩니다. 권한을 보다 효과적으로 관리하기 위해 광범위한 권한을 부여하는 대신 객체의 하위 세트에 대해 READ-ACP 및 WRITE-ACP 권한만 부여하도록 선택할 수 있습니다. 이렇게 하면 계정이 개별 객체 ACL을 업데이트하여 특정 객체에 대해서만 권한을 관리하도록 제한됩니다.

버킷 ACL을 사용하는 경우

버킷 ACL에 대해 유일하게 권장되는 사용 사례는 Amazon S3 Log Delivery 그룹에 액세스 로그 객체를 버킷에 작성하기 위한 쓰기 권한을 부여하는 것입니다(Amazon S3 서버 액세스 로깅 (p. 705) 참조). Amazon S3에서 액세스 로그를 버킷으로 전송하게 하려는 경우 Log Delivery 그룹에 버킷에 대한 쓰기 권한을 부여해야 합니다. Log Delivery 그룹에 필요한 권한을 부여할 수 있는 유일한 방법은 다음 버킷 ACL 조각에 나와 있듯이 버킷 ACL을 통하는 것 뿐입니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
   ...
  </Owner>
  <AccessControlList>
    <Grant>
        ...
    </Grant>
    <Grant>
       <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
       </Grantee>
       <Permission>WRITE</Permission>
    </Grant>
```

```
  </AccessControlList>
</AccessControlPolicy>
```

버킷 정책을 사용하는 경우

버킷을 소유하는 AWS 계정이 자신의 계정에 속한 사용자에게 권한을 부여하려는 경우 버킷 정책 또는 사용자 정책을 사용할 수 있습니다. 그러나 다음 시나리오에서는 버킷 정책을 사용해야 합니다.

- 모든 Amazon S3 권한에 대한 교차 계정 권한을 관리하려고 함 – ACL을 사용하여 다른 계정에 교차 계정 권한을 부여할 수 있지만 ACL은 한정된 권한 세트(부여할 수 있는 권한 (p. 440))만 지원하며 여기에 모든 Amazon S3 권한이 포함되는 것은 아닙니다. 예를 들어, ACL을 사용하여 버킷 하위 리소스에 대한 권한을 부여할 수 없습니다(Amazon S3의 Identity and Access Management (p. 299) 참조).

  버킷 정책과 사용자 정책 모두 모든 Amazon S3 작업에 대한 권한 부여 작업을 지원하지만(Amazon S3 작업 (p. 342) 참조) 사용자 정책은 계정에 속한 사용자에 대한 권한을 관리하기 위한 것입니다. 다른 AWS 계정 또는 다른 계정에 속한 사용자에 대한 교차 계정 권한의 경우 버킷 정책을 사용해야 합니다.

사용자 정책을 사용하는 경우

일반적으로, 사용자 정책 또는 버킷 정책을 사용하여 권한을 관리할 수 있습니다. 사용자를 만들고 사용자에게 정책을 연결하여 권한을 개별적으로 관리하는 방법으로 권한을 관리하도록 선택하거나 버킷 정책과 같은 리소스 기반 정책이 현재 시나리오에 더 적합하다고 판단할 수도 있습니다.

AWS Identity and Access Management(IAM)를 사용하면 AWS 계정 내에서 다중 사용자를 만들고 사용자 정책을 통해 해당 권한을 관리할 수 있습니다. IAM 사용자에게는 자신이 속한 상위 계정의 권한과 사용자가 액세스하려는 리소스를 소유하는 AWS 계정의 권한이 있어야 합니다. 권한은 다음과 같이 부여할 수 있습니다.

- 상위 계정의 권한 – 상위 계정은 사용자 정책을 연결하여 자신의 사용자에게 권한을 부여할 수 있습니다.
- 리소스 소유자의 권한 – 리소스 소유자는 IAM 사용자(버킷 정책 사용) 또는 상위 계정(버킷 정책, 버킷 ACL 또는 객체 ACL 사용)에 권한을 부여할 수 있습니다.

이는 다른 아이 소유의 장난감을 가지고 놀고 싶어하는 아이와 같습니다. 이 경우 아이는 부모와 장난감 주인으로부터 허락을 받아야 그 장난감을 가지고 놀 수 있습니다.

권한 위임

AWS 계정이 리소스를 소유하는 경우 다른 AWS 계정에 해당 권한을 부여할 수 있습니다. 그런 다음 해당 계정은 권한 또는 권한의 하위 세트를 계정에 속한 사용자에게 위임할 수 있습니다. 이를 권한 위임이라고 합니다. 그러나 다른 계정으로부터 권한을 받는 계정은 다른 AWS 계정에 교차 계정 권한을 위임할 수 없습니다.

관련 주제

먼저 Amazon S3 리소스에 대한 액세스를 관리하는 방법을 설명하는 모든 소개 주제와 관련 지침을 검토하는 것이 좋습니다. 자세한 내용은 Amazon S3 리소스에 대한 액세스 관리 소개 (p. 299) 단원을 참조하십시오. 그 후, 다음과 같이 주제별로 특정 액세스 정책 옵션에 대한 자세한 정보를 얻을 수 있습니다.

- 버킷 정책 및 사용자 정책 사용 (p. 338)
- ACL(액세스 통제 목록) 개요 (p. 438)
- ACL을 사용한 액세스 관리 (p. 437)

예제 안내: Amazon S3 리소스에 대한 액세스 관리

이 항목에서는 Amazon S3 리소스에 대한 액세스 권한을 부여하는 예제 안내를 소개합니다. 이들 예제는 AWS Management 콘솔을 사용하여 리소스(버킷, 객체, 사용자)를 만들고 그러한 리소스에 권한을 부여합니

다. 그런 다음 명령줄 도구를 사용하여 권한을 확인하는 방법을 보여 주므로 코드를 작성할 필요가 없습니다. Amazon에서는 AWS 명령줄 인터페이스(CLI)와 Windows PowerShell용 AWS 도구를 모두 제공합니다.

- 예제 1: 버킷 소유자가 자신의 사용자에게 버킷 권한 부여 (p. 316)

  계정에 생성한 IAM 사용자는 기본적으로 권한이 없습니다. 이 연습에서는 사용자에게 버킷 및 객체 작업을 수행할 수 있는 권한을 부여합니다.
- 예제 2: 버킷 소유자가 교차 계정 버킷 권한 부여 (p. 320)

  이 연습에서 버킷 소유자인 계정 A는 다른 AWS 계정인 계정 B에게 교차 계정 권한을 부여합니다. 그런 다음 계정 B는 그러한 계정을 해당 계정의 사용자에게 위임합니다.
- 객체와 버킷 소유자가 다를 경우의 객체 권한 관리

  이 예제 시나리오는 버킷 소유자가 다른 사용자에게 객체 권한을 부여하지만, 버킷의 모든 객체를 버킷 소유자가 소유하는 것은 아닌 경우입니다. 버킷 소유자가 필요한 권한은 무엇이며, 이러한 권한을 어떻게 위임할 수 있습니까?

  버킷을 생성한 AWS 계정을 버킷 소유자라고 합니다. 소유자는 다른 AWS 계정에 객체를 업로드할 수 있는 권한을 부여할 수 있고. 객체를 만든 AWS 계정은 그 객체를 소유합니다. 버킷 소유자는 다른 AWS 계정이 만든 객체에 대한 권한이 없습니다. 버킷 소유자가 객체에 대한 액세스를 부여하는 버킷 정책을 작성할 경우, 이 정책은 다른 계정이 소유한 객체에는 적용되지 않습니다.

  이 경우, 객체 소유자는 우선 객체 ACL을 사용하여 버킷 소유자에게 권한을 부여해야 합니다. 그런 후에 다음 예에 나와 있는 것처럼 버킷 소유자가 다른 사용자나, 자신이 소유한 계정의 사용자 또는 다른 AWS 계정에 이러한 객체 권한을 위임할 수 있습니다.
  - 예제 3: 버킷 소유자가 자신의 사용자에게 자신이 소유하지 않는 객체에 대한 권한 부여 (p. 326)

    이 연습에서 먼저 버킷 소유자는 객체 소유자로부터 권한을 받습니다. 그런 다음 버킷 소유자는 이러한 권한을 자신이 소유한 계정의 사용자에게 위임합니다.
  - 예 4: 버킷 소유자가 자신의 소유가 아닌 객체에 교차-계정 권한 부여 (p. 331)

    교차 계정 위임은 지원되지 않기 때문에 버킷 소유자는 객체 소유자로부터 권한을 받은 후 다른 AWS 계정에 권한을 위임할 수 없습니다(권한 위임 (p. 313) 참조). 대신에 버킷 소유자는 특정 작업(예: 객체 받기)을 수행할 수 있는 권한을 가진 IAM 역할을 만들고, 다른 AWS 계정이 이 역할을 맡도록 허용할 수 있습니다. 역할을 맡으면 누구든 객체에 액세스할 수 있습니다. 이 예제는 버킷 소유자가 IAM 역할을 사용하여 이 교차 계정 위임을 활성화하는 방법을 보여 줍니다.

예제 안내를 실습하기 전에 알아 두어야 할 사항

이들 예제는 AWS Management 콘솔을 사용하여 리소스를 만들고 권한을 부여합니다. 또한 권한을 테스트하기 위해 명령줄 도구인 AWS Command Line Interface(CLI)와 Windows PowerShell용 AWS 도구를 사용하므로 직접 코드를 작성할 필요가 없습니다. 권한을 테스트하려면 이러한 도구 중 하나를 설정해야 합니다. 자세한 내용은 예제 안내를 위한 도구 설정 (p. 315) 단원을 참조하십시오.

또한 리소스를 만들 경우, 이 예제는 AWS 계정의 루트 자격 증명을 사용하지 않습니다. 대신 해당 계정에 관리자 사용자를 만들어 이러한 작업을 수행합니다.

관리자 사용자를 사용하여 리소스를 만들고 권한을 부여하는 것에 대한 소개

AWS Identity and Access Management(IAM)는 AWS 계정의 루트 자격 증명을 사용하여 요청하지 않을 것을 권장합니다. 대신 IAM 사용자를 만들고 이 사용자에게 모든 권한을 부여한 후 이 사용자의 자격 증명을 사용하여 AWS 작업을 수행하도록 합니다. 이 사용자를 관리자 사용자라고 합니다. 자세한 내용은 루트 계정 자격 증명 대 IAM 사용자 자격 증명(AWS General Reference) 및 IAM 모범 사례(IAM 사용 설명서) 단원을 참조하십시오.

이 단원의 모든 예제 안내는 관리자 사용자 자격 증명을 사용합니다. AWS 계정에 대한 관리자 사용자가 없을 경우 이 항목에서는 그 방법을 보여 줍니다.

사용자 자격 증명을 사용하여 AWS Management 콘솔에 로그인하려면 IAM User 로그인 URL을 사용해야 합니다. IAM 콘솔은 AWS 계정에 이 URL을 제공합니다. 이 항목에서는 URL을 받는 방법을 보여 줍니다.

예제 안내를 위한 도구 설정

이 소개 예제는 (예제 안내: Amazon S3 리소스에 대한 액세스 관리 (p. 313) 참조) AWS Management 콘솔을 사용하여 리소스를 만들고 권한을 부여합니다. 또한 권한을 테스트하기 위해 명령줄 도구인 AWS Command Line Interface(CLI)와 Windows PowerShell용 AWS 도구를 사용하므로 직접 코드를 작성할 필요가 없습니다. 권한을 테스트하려면 이러한 도구 중 하나를 설정해야 합니다.

AWS CLI를 설정하려면

1. AWS CLI를 다운로드하고 구성합니다. 지침은 AWS Command Line Interface 사용 설명서에서 다음 항목을 참조하십시오.

   AWS 명령줄 인터페이스로 설정

   AWS 명령줄 인터페이스 설치

   AWS 명령줄 인터페이스 구성
2. 기본 프로필을 설정합니다.

   AWS CLI 구성 파일에 사용자 자격 증명을 저장합니다. AWS 계정 자격 증명을 사용하여 구성 파일에 기본 프로필을 만듭니다. AWS CLI 구성 파일을 찾고 편집하는 방법에 대한 지침은 구성 및 자격 증명 파일 단원을 참조하십시오.

   ```
   [default]
   aws_access_key_id = access key ID
   aws_secret_access_key = secret access key
   region = us-west-2
   ```
3. 명령 프롬프트에 다음 명령을 입력하여 설정을 확인합니다. 이들 명령은 명시적으로 자격 증명을 제공하지 않으므로 기본 프로필의 자격 증명이 사용됩니다.

   - help 명령 사용해 보기

     ```
     aws help
     ```
   - `aws s3 ls`를 사용하여 구성된 계정의 버킷 목록을 가져옵니다.

     ```
     aws s3 ls
     ```

예제 안내에 따라 다음과 같이 사용자를 만들고, 프로필을 작성하여 구성 파일에 사용자 자격 증명을 저장합니다. 이러한 프로필의 이름은 AccountAadmin과 AccountBadmin입니다.

```
[profile AccountAadmin]
aws_access_key_id = User AccountAadmin access key ID
aws_secret_access_key = User AccountAadmin secret access key
region = us-west-2

[profile AccountBadmin]
aws_access_key_id = Account B access key ID
aws_secret_access_key = Account B secret access key
region = us-east-1
```

이러한 사용자 자격 증명을 사용하여 명령을 실행하기 위해 프로필 이름을 지정하는 `--profile` 파라미터를 추가합니다. 다음 AWS CLI 명령은 `examplebucket`의 객체 목록을 검색하고 `AccountBadmin` 프로필을 지정합니다.

```
aws s3 ls s3://examplebucket --profile AccountBadmin
```

또는 명령 프롬프트에서 `AWS_DEFAULT_PROFILE` 환경 변수를 변경하여 사용자 자격 증명 중 한 세트를 기본 프로필로 구성할 수 있습니다. 이렇게 하고 나면 `--profile` 파라미터 없이 AWS CLI 명령을 실행할 때마다 AWS CLI에서, 환경 변수에 기본 프로필로 설정된 프로필을 사용합니다.

```
$ export AWS_DEFAULT_PROFILE=AccountAadmin
```

Windows PowerShell용 AWS 도구를 설정하려면

1. Windows PowerShell용 AWS 도구를 다운로드하여 구성합니다. 지침은 Windows PowerShell용 AWS 도구 사용 설명서의 Windows PowerShell용 AWS 도구 다운로드 및 설치를 참조하십시오.

   Note

   Windows PowerShell 모듈용 AWS 도구를 로드하려면 PowerShell 스크립트를 실행해야 합니다. 자세한 내용은 Windows PowerShell용 AWS 도구 사용 설명서의 스크립트 실행 활성화를 참조하십시오.

2. 이 실습에서는 `Set-AWSCredentials` 명령을 사용하여 세션별로 AWS 자격 증명을 지정합니다. 이 명령은 영구 저장소에 자격 증명을 저장합니다(`-StoreAs` 파라미터).

   ```
   Set-AWSCredentials -AccessKey AccessKeyID -SecretKey SecretAccessKey -storeas string
   ```

3. 설정을 확인합니다.

   - `Get-Command`를 실행하여 Amazon S3 작업에서 사용할 수 있는 명령 목록을 검색합니다.

     ```
     Get-Command -module awspowershell -noun s3* -StoredCredentials string
     ```

   - `Get-S3Object` 명령을 실행하여 버킷의 객체 목록을 검색합니다.

     ```
     Get-S3Object -BucketName bucketname -StoredCredentials string
     ```

명령 목록은 Amazon Simple Storage Service Cmdlets를 참조하십시오.

이제 실습 준비가 되었습니다. 단원 시작 부분에 있는 링크를 따라 가십시오.

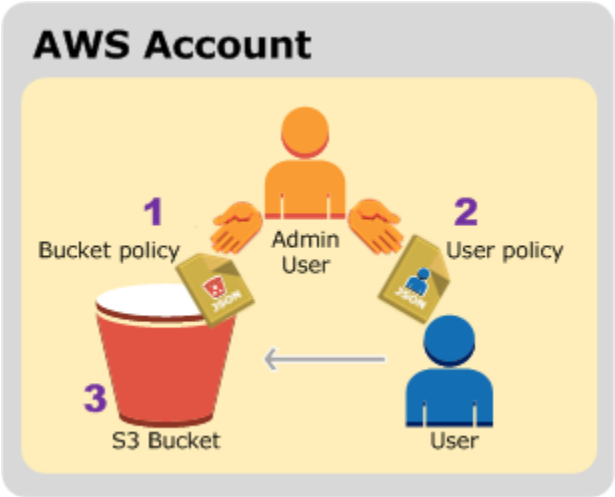
예제 1: 버킷 소유자가 자신의 사용자에게 버킷 권한 부여

주제

- 0단계: 연습 준비 (p. 317)
- 1단계: 계정 A에서 리소스(버킷 및 IAM 사용자)를 만들고 권한 부여 (p. 318)
- 2단계: 권한 테스트 (p. 319)

이 연습에서는 AWS 계정이 버킷을 소유하며, 해당 계정의 IAM 사용자가 있습니다. 기본적으로 사용자에게 권한이 없습니다. 사용자가 태스크를 수행하려면 상위 계정이 이러한 사용자에게 권한을 부여해야 합니다. 버킷 소유자와 상위 계정은 동일합니다. 따라서 AWS 계정이 버킷 정책, 사용자 정책 또는 둘 다를 사용하여 사용자에게 버킷에 대한 권한을 부여할 수 있습니다. 계정 소유자는 버킷 정책을 사용하여 권한 중 일부를 부여하고 사용자 정책을 사용하여 나머지 권한을 부여합니다.

다음은 안내 단계의 요약입니다.



1. 계정 관리자가 사용자에게 권한 세트를 부여하는 버킷 정책을 만듭니다.
2. 계정 관리자가 추가 권한을 부여하는 사용자 정책을 사용자에게 연결합니다.
3. 그런 다음 사용자가 버킷 정책과 사용자 정책 모두를 통해 부여 받은 권한을 시험해 봅니다.

이 예제에서는 AWS 계정이 필요합니다. 계정의 루트 자격 증명을 사용하는 대신 관리자 사용자를 만듭니다 (관리자 사용자를 사용하여 리소스를 만들고 권한을 부여하는 것에 대한 소개 (p. 314) 참조). AWS 계정 및 관리자 사용자는 다음과 같이 참조합니다.

계정 ID	계정 이름	계정의 관리자 사용자
1111-1111-1111	계정 A	AccountAadmin

사용자를 만들고 권한을 부여하는 모든 작업은 AWS Management 콘솔에서 수행됩니다. 권한을 확인하기 위해 본 연습에서는 명령줄 도구, AWS 명령줄 인터페이스 및 Windows PowerShell용 AWS 도구를 사용합니다. 따라서 사용자가 코드를 작성할 필요가 없습니다.

0단계: 연습 준비

1. AWS 계정이 있으며 이 계정에 관리자 권한이 있는 사용자가 있는지 확인합니다.
   a. 필요한 경우 계정에 가입합니다. 이 계정을 계정 A라고 합니다.
      i. https://aws.amazon.com/s3로 이동하여 가입을 클릭합니다.
      ii. 화면에 표시되는 지시 사항을 따릅니다.

         계정이 활성화되고 사용 가능한 상태가 되면 AWS에서 사용자에게 이메일로 알립니다.
   b. 계정 A에서 관리자 사용자인 AccountAadmin을 만듭니다. 계정 A의 자격 증명을 사용하여 IAM 콘솔에 로그인하고 다음을 수행합니다.
      i. 사용자 AccountAadmin을 만들고 사용자 보안 자격 증명을 적어 둡니다.

         지침은 IAM 사용 설명서의 AWS 계정의 IAM 사용자 생성 단원을 참조하십시오.
      ii. 모든 권한을 제공하는 사용자 정책을 연결하여 AccountAadmin 관리자에게 권한을 부여합니다.

         지침은 IAM 사용 설명서의 정책 작업 단원을 참조하십시오.

iii. AccountAadmin의 IAM User Sign-In URL(IAM 사용자 로그인 URL)을 적어 둡니다. AWS Management 콘솔에 로그인할 때 이 URL을 사용해야 합니다. 이 URL을 찾을 수 있는 위치에 대한 자세한 내용은 IAM 사용 설명서의 사용자의 계정 로그인 방법 단원을 참조하십시오. 각 계정의 URL을 적어 둡니다.

2. AWS 명령행 인터페이스(CLI)나 Windows PowerShell용 AWS 도구를 설정합니다. 관리자 사용자 자격 증명은 다음과 같이 저장합니다.

   - AWS CLI를 사용하는 경우 구성 파일에서 AccountAadmin 프로필을 만듭니다.
   - Windows PowerShell용 AWS 툴을 사용하는 경우 세션의 자격 증명을 AccountAadmin으로 저장해야 합니다.

   지침은 예제 안내를 위한 도구 설정 (p. 315) 단원을 참조하십시오.

1단계: 계정 A에서 리소스(버킷 및 IAM 사용자)를 만들고 권한 부여

계정 A의 사용자 AccountAadmin에 대한 자격 증명과 특정 IAM 사용자 로그인 URL을 사용하여 AWS Management 콘솔에 로그인하고 다음을 수행합니다.

1. 리소스(버킷 및 IAM 사용자) 만들기

   a. Amazon S3 콘솔에서 버킷을 만듭니다. 버킷을 만든 AWS 리전을 적어 둡니다. 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷 정책을 생성하려면 어떻게 해야 합니까? 단원을 참조하십시오.

   b. IAM 콘솔에서 다음을 수행합니다.

      i. 사용자 Dave를 만듭니다.

         이에 관한 지침은 IAM 사용 설명서의 IAM 사용자 만들기(AWS Management Console) 단원을 참조하십시오.

      ii. UserDave 자격 증명을 적어 둡니다.

      iii. 사용자 Dave의 Amazon 리소스 이름(ARN)을 적어 둡니다. IAM 콘솔에서 해당 사용자를 선택하면 요약 탭에 사용자 ARN이 제공됩니다.

2. 권한 부여

   버킷 소유자와 사용자가 속한 상위 계정이 같기 때문에 AWS 계정이 버킷 정책, 사용자 정책 또는 둘 다를 사용하여 사용자에게 권한을 부여할 수 있습니다. 이 예제에서는 둘 다를 사용합니다. 같은 계정이 객체를 소유하는 경우 버킷 소유자가 버킷 정책 또는 IAM 정책에서 객체 권한을 부여할 수 있습니다.

   a. Amazon S3 콘솔에서 *`examplebucket`*에 다음 버킷 정책을 연결합니다.

      이 정책에는 두 설명문이 있습니다.

      - 첫 번째 설명문은 Dave에게 버킷 작업 권한인 `s3:GetBucketLocation` 및 `s3:ListBucket`을 부여합니다.
      - 두 번째 설명문은 `s3:GetObject` 권한을 부여합니다. 계정 A가 객체도 소유하기 때문에 계정 관리자가 `s3:GetObject` 권한을 부여할 수 있습니다.

      `Principal` 설명문에서 Dave는 해당 사용자 ARN으로 식별됩니다. 정책 요소에 대한 자세한 내용은 Amazon S3의 정책 및 권한 (p. 339)를 참조하십시오.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "statement1",
```

```
            "Effect": "Allow",
            "Principal": {
               "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
            },
            "Action": [
               "s3:GetBucketLocation",
               "s3:ListBucket"
            ],
            "Resource": [
               "arn:aws:s3:::examplebucket"
            ]
         },
         {
            "Sid": "statement2",
            "Effect": "Allow",
            "Principal": {
               "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
            },
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
               "arn:aws:s3:::examplebucket/*"
            ]
         }
      ]
}
```

b. 다음 정책을 사용하여 사용자 Dave에 대한 인라인 정책을 만듭니다. 이 정책은 Dave에게 `s3:PutObject` 권한을 부여합니다. 버킷 이름을 제공하여 정책을 업데이트해야 합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "PermissionForObjectOperations",
         "Effect": "Allow",
         "Action": [
            "s3:PutObject"
         ],
         "Resource": [
            "arn:aws:s3:::examplebucket/*"
         ]
      }
   ]
}
```

지침은 IAM 사용 설명서의 인라인 정책 작업 단원을 참조하십시오. 계정 A의 자격 증명을 사용하여 콘솔에 로그인해야 합니다.

2단계: 권한 테스트

Dave의 자격 증명을 사용하여 권한이 작동하는지 확인하십시오. 다음 두 가지 절차 중 하나를 사용할 수 있습니다.

AWS CLI를 사용하여 테스트

1. 다음 UserDaveAccountA 프로파일을 추가하여 AWS CLI 구성 파일을 업데이트합니다. 자세한 내용은 예제 안내를 위한 도구 설정 (p. 315) 단원을 참조하십시오.

```
[profile UserDaveAccountA]
```

```
aws_access_key_id = access-key
aws_secret_access_key = secret-access-key
region = us-east-1
```

2. Dave가 사용자 정책에서 권한을 부여 받은 작업을 수행할 수 있는지 확인합니다. 다음 AWS CLI `put-object` 명령을 사용하여 샘플 객체를 업로드합니다.

   명령의 `--body` 파라미터를 통해 업로드할 원본 파일이 식별됩니다. 예를 들어, 파일이 Windows 시스템의 C: 드라이브 루트에 있는 경우 `c:\HappyFace.jpg`를 지정합니다. `--key` 파라미터는 객체의 키 이름을 제공합니다.

```
aws s3api put-object --bucket examplebucket --key HappyFace.jpg --body HappyFace.jpg --
profile UserDaveAccountA
```

   다음 AWS CLI 명령을 실행하여 객체를 가져옵니다.

```
aws s3api get-object --bucket examplebucket --key HappyFace.jpg OutputFile.jpg --
profile UserDaveAccountA
```

Windows PowerShell용 AWS 도구를 사용하여 테스트

1. Dave의 자격 증명을 AccountADave로 저장합니다. 그런 다음 이러한 자격 증명을 사용하여 객체에 대해 PUT 및 GET을 실행합니다.

```
set-awscredentials -AccessKey AccessKeyID -SecretKey SecretAccessKey -storeas
 AccountADave
```

2. 사용자 Dave의 저장된 자격 증명을 사용하여 Windows PowerShell용 AWS 도구의 `Write-S3Object` 명령을 통해 샘플 객체를 업로드합니다.

```
Write-S3Object -bucketname examplebucket -key HappyFace.jpg -file HappyFace.jpg -
StoredCredentials AccountADave
```

   이전에 업로드한 객체를 다운로드합니다.

```
Read-S3Object -bucketname examplebucket -key HappyFace.jpg -file Output.jpg -
StoredCredentials AccountADave
```

예제 2: 버킷 소유자가 교차 계정 버킷 권한 부여

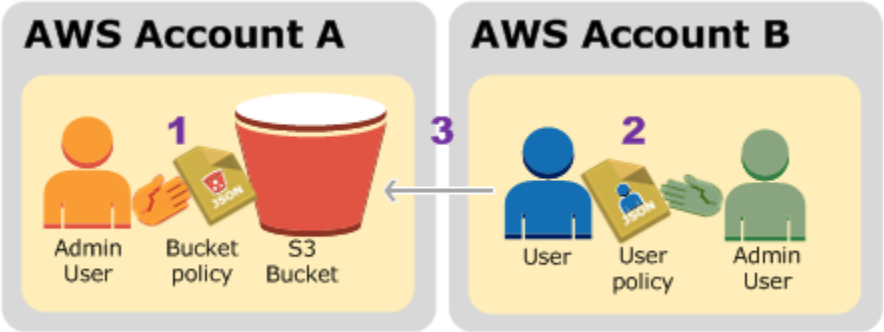
주제

- 0단계: 연습 준비 (p. 321)
- 1단계: 계정 A 작업 수행 (p. 322)
- 2단계: 계정 B 작업 수행 (p. 323)
- 3단계: 추가 작업: 명시적 거부 시도 (p. 325)
- 4단계: 정리 (p. 326)

AWS 계정(예: 계정 A)은 다른 AWS 계정(계정 B)에 버킷 및 객체와 같은 해당 리소스에 대한 액세스 권한을 부여할 수 있습니다. 그런 다음 계정 B는 해당 권한을 자신의 계정에 속한 사용자에게 위임할 수 있습니다. 이 시나리오 예제에서는 버킷 소유자가 특정 버킷 작업을 수행할 수 있도록 다른 계정에 교차 계정 권한을 부여합니다.

> **Note**
>
> 계정 A는 버킷 정책을 사용하여 계정 B에 속한 사용자에게 직접 권한을 부여할 수도 있습니다. 그러나 해당 사용자는 여전히 자신이 속하는 상위 계정인 계정 B로부터 허가를 받아야 합니다. 이는 계정 B가 계정 A로부터 허가를 받지 못했더라도 마찬가지입니다. 사용자가 두 리소스 소유자 모두와 상위 계정으로부터 허가를 받는 한 해당 사용자는 리소스에 액세스할 수 있습니다.

각 연습 단계는 다음과 같이 간략히 설명할 수 있습니다.



1. 계정 A의 관리자 사용자가 특정 버킷 작업을 수행할 수 있도록 계정 B에 교차 계정 권한을 부여하는 버킷 정책을 연결합니다.

   계정 B에 속하는 관리자 사용자는 자동으로 권한을 상속합니다.
2. 계정 B의 관리자 사용자가 계정 A로부터 받은 권한을 위임하는 사용자 정책을 사용자에게 연결합니다.
3. 그런 다음 계정 B에 속한 사용자가 계정 A 소유의 버킷에 있는 객체에 액세스하여 권한을 확인합니다.

이 예제에서는 두 계정이 필요합니다. 다음 표에서는 이러한 계정과 해당 계정에 속하는 관리자 사용자를 나타내는 방법을 보여 줍니다. IAM 지침(관리자 사용자를 사용하여 리소스를 만들고 권한을 부여하는 것에 대한 소개 (p. 314) 참조)에 따라 본 연습에서는 계정 루트 자격 증명을 사용하지 않습니다. 대신 사용자가 각 계정에서 관리자 사용자를 만든 후 해당 자격 증명을 사용하여 리소스를 만들고 해당 리소스에 권한을 부여합니다.

AWS 계정 ID	계정 이름	계정의 관리자 사용자
1111-1111-1111	계정 A	AccountAadmin
2222-2222-2222	계정 B	AccountBadmin

사용자를 만들고 권한을 부여하는 모든 작업은 AWS Management 콘솔에서 수행됩니다. 권한을 확인하기 위해 본 연습에서는 명령줄 도구, AWS 명령줄 인터페이스 및 Windows PowerShell용 AWS 도구를 사용합니다. 따라서 사용자가 코드를 작성할 필요가 없습니다.

0단계: 연습 준비

1. AWS 계정이 두 개 있으며 각 계정에 이전 단원의 표에 나와 있는 것과 같이 관리자 사용자가 한 명 있는지 확인합니다.

   a. 필요한 경우 AWS 계정에 가입합니다.

      i. https://aws.amazon.com/s3/로 이동하여 Create an AWS Account(AWS 계정 생성)를 클릭합니다.

      ii. 화면에 표시되는 지시 사항을 따릅니다.

      계정이 활성화되고 사용 가능한 상태가 되면 AWS에서 사용자에게 이메일로 알립니다.

b. 계정 A의 자격 증명을 사용하여 IAM 콘솔에 로그인하여 관리자 사용자를 만듭니다.

   i. 사용자 AccountAadmin을 만들고 보안 자격 증명을 적어 둡니다. 지침은 IAM 사용 설명서의 AWS 계정의 IAM 사용자 생성 단원을 참조하십시오.

   ii. 모든 권한을 제공하는 사용자 정책을 연결하여 AccountAadmin 관리자에게 권한을 부여합니다. 지침은 IAM 사용 설명서의 정책 작업 단원을 참조하십시오.

c. IAM 콘솔의 대시보드에서 IAM User Sign-In URL(IAM 사용자 로그인 URL)을 적어 둡니다. 계정에 속한 모든 사용자가 AWS Management 콘솔에 로그인할 때 이 URL을 사용해야 합니다.

   자세한 내용은 IAM 사용 설명서의 사용자의 계정 로그인 방법 단원을 참조하십시오.

d. 계정 B의 자격 증명을 사용하여 이전 단계를 반복하고 관리자 사용자인 AccountBadmin을 만듭니다.

2. AWS 명령행 인터페이스(CLI)나 Windows PowerShell용 AWS 도구를 설정합니다. 관리자 사용자 자격 증명은 다음과 같이 저장합니다.

   - AWS CLI를 사용하는 경우 구성 파일에서 AccountAadmin과 AccountBadmin이라는 두 프로파일을 만듭니다.
   - Windows PowerShell용 AWS 툴을 사용하는 경우, 세션의 자격 증명을 AccountAadmin과 AccountBadmin으로 저장해야 합니다.

   지침은 예제 안내를 위한 도구 설정 (p. 315) 단원을 참조하십시오.

3. 프로파일이라고도 하는 관리자 사용자 자격 증명을 저장합니다. 입력하는 각 명령에 대해 자격 증명을 지정하는 대신 프로파일 이름을 사용할 수 있습니다. 자세한 내용은 예제 안내를 위한 도구 설정 (p. 315) 단원을 참조하십시오.

   a. 두 계정에 속한 각 관리자 사용자에 대해 AWS CLI 자격 증명 파일에서 프로파일을 추가합니다.

```
[AccountAadmin]
aws_access_key_id = access-key-ID
aws_secret_access_key = secret-access-key
region = us-east-1

[AccountBadmin]
aws_access_key_id = access-key-ID
aws_secret_access_key = secret-access-key
region = us-east-1
```

   b. Windows PowerShell용 AWS 도구를 사용하는 경우

```
set-awscredentials -AccessKey AcctA-access-key-ID -SecretKey AcctA-secret-access-
key -storeas AccountAadmin
set-awscredentials -AccessKey AcctB-access-key-ID -SecretKey AcctB-secret-access-
key -storeas AccountBadmin
```

1단계: 계정 A 작업 수행

1.1단계: AWS Management Console에 로그인

계정 A의 IAM 사용자 로그인 URL을 사용하여 먼저 AccountAadmin 사용자로 AWS Management 콘솔에 로그인합니다. 이 사용자가 버킷을 만들고 버킷에 정책을 연결합니다.

1.2단계: 버킷 만들기

1. Amazon S3 콘솔에서 버킷을 만듭니다. 이 연습에서는 버킷이 미국 동부(버지니아 북부) 리전에서 생성되며 이름이 `examplebucket`이라고 가정합니다.

지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷 정책을 생성하려면 어떻게 해야 합니까? 단원을 참조하십시오.

2. 버킷에 샘플 객체를 업로드합니다.

   지침은 Amazon Simple Storage Service 시작 안내서의 버킷에 객체 추가 단원을 참조하십시오.

1.3단계: 계정 B에 교차 계정 권한을 부여하기 위해 버킷 정책 연결

버킷 정책을 사용하면 계정 B에 `s3:GetBucketLocation` 및 `s3:ListBucket` 권한이 부여됩니다. AccountAadmin 사용자 자격 증명을 사용하여 여전히 콘솔에 로그인된 상태라고 가정됩니다.

1. `examplebucket`에 다음 버킷 정책을 연결합니다. 이 정책은 계정 B에게 `s3:GetBucketLocation` 및 `s3:ListBucket` 작업에 대한 권한을 부여합니다.

   지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷 정책을 추가하려면 어떻게 해야 합니까? 단원을 참조하십시오.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Example permissions",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountB-ID:root"
         },
         "Action": [
            "s3:GetBucketLocation",
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::examplebucket"
         ]
      }
   ]
}
```

2. 계정 B와 해당 관리자 사용자가 작업을 수행할 수 있는지 확인합니다.

   - AWS CLI 사용

```
aws s3 ls s3://examplebucket --profile AccountBadmin
aws s3api get-bucket-location --bucket examplebucket --profile AccountBadmin
```

   - Windows PowerShell용 AWS 도구 사용

```
get-s3object -BucketName example2bucket -StoredCredentials AccountBadmin
get-s3bucketlocation -BucketName example2bucket -StoredCredentials AccountBadmin
```

2단계: 계정 B 작업 수행

이제 계정 B의 관리자가 사용자 Dave를 만들고 계정 A로부터 받은 권한을 위임합니다.

2.1단계: AWS Management Console에 로그인

계정 B의 IAM 사용자 로그인 URL을 사용하여 먼저 AccountBadmin 사용자로 AWS Management 콘솔에 로그인합니다.

2.2단계: 계정 B에서 사용자 Dave 만들기

IAM 콘솔에서 사용자 Dave를 만듭니다.

이에 관한 지침은 IAM 사용 설명서의 IAM 사용자 만들기(AWS Management Console) 단원을 참조하십시오.

2.3단계: 사용자 Dave에게 권한 위임

다음 정책을 사용하여 사용자 Dave에 대한 인라인 정책을 만듭니다. 버킷 이름을 제공하여 정책을 업데이트해야 합니다.

AccountBadmin 사용자 자격 증명을 사용하여 콘솔에 로그인했다고 가정됩니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Example",
         "Effect": "Allow",
         "Action": [
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::examplebucket"
         ]
      }
   ]
}
```

지침은 IAM 사용 설명서의 인라인 정책 작업 단원을 참조하십시오.

2.4단계: 권한 테스트

이제 계정 B에 속한 Dave는 계정 A가 소유하는 `examplebucket`의 콘텐츠를 나열할 수 있습니다. 다음 절차 중 하나를 사용하여 권한을 확인할 수 있습니다.

AWS CLI를 사용하여 테스트

1. AWS CLI 구성 파일에 UserDave 프로파일을 추가합니다. 구성 파일에 대한 자세한 내용은 예제 안내를 위한 도구 설정 (p. 315)을 참조하십시오.

   ```
   [profile UserDave]
   aws_access_key_id = access-key
   aws_secret_access_key = secret-access-key
   region = us-east-1
   ```

2. 명령 프롬프트에서 다음 AWS CLI 명령을 입력하여 이제 Dave가 계정 A 소유의 `examplebucket`에서 객체 목록을 가져올 수 있는지 확인합니다. 명령에 UserDave 프로파일이 지정된 것을 확인합니다.

   ```
   aws s3 ls s3://examplebucket --profile UserDave
   ```

   Dave에게는 다른 권한이 없습니다. 따라서 Dave가 다음 get-bucket-location과 같은 다른 작업을 시도할 경우 Amazon S3에서 권한 거부를 반환합니다.

   ```
   aws s3api get-bucket-location --bucket examplebucket --profile UserDave
   ```

Windows PowerShell용 AWS 도구를 사용하여 테스트

1. Dave의 자격 증명을 AccountBDave로 저장합니다.

```
set-awscredentials -AccessKey AccessKeyID -SecretKey SecretAccessKey -storeas
 AccountBDave
```

2. List Bucket 명령을 시도합니다.

```
get-s3object -BucketName example2bucket -StoredCredentials AccountBDave
```

Dave에게는 다른 권한이 없습니다. 따라서 Dave가 다음 get-bucket-location과 같은 다른 작업을 시도할 경우 Amazon S3에서 권한 거부를 반환합니다.

```
get-s3bucketlocation -BucketName example2bucket -StoredCredentials AccountBDave
```

3단계: 추가 작업: 명시적 거부 시도

ACL, 버킷 정책 및 사용자 정책을 통해 권한을 부여 받을 수 있습니다. 그러나 버킷 정책 또는 사용자 정책을 통해 설정된 명시적 거부가 있을 경우 명시적 거부가 다른 모든 권한보다 우선 적용됩니다. 테스트를 위해 버킷 정책을 업데이트하고 계정 B의 `s3:ListBucket` 권한을 명시적으로 거부해 보겠습니다. 이 정책은 `s3:ListBucket` 권한도 부여하지만 명시적 거부가 우선 적용되어 계정 B 또는 계정 B에 속한 사용자가 `examplebucket`의 객체를 나열할 수 없습니다.

1. 계정 A에 속한 사용자 AccountAadmin의 자격 증명을 사용하여 버킷 정책을 다음으로 대체합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Example permissions",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountB-ID:root"
         },
         "Action": [
            "s3:GetBucketLocation",
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::examplebucket"
         ]
      },
      {
         "Sid": "Deny permission",
         "Effect": "Deny",
         "Principal": {
            "AWS": "arn:aws:iam::AccountB-ID:root"
         },
         "Action": [
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::examplebucket"
         ]
      }
   ]
}
```

2. 이제 AccountBadmin 자격 증명을 사용하여 버킷 목록을 가져오려고 하면 액세스 거부가 표시됩니다.
   - AWS CLI 사용:

```
aws s3 ls s3://examplebucket --profile AccountBadmin
```

- Windows PowerShell용 AWS 도구 사용:

```
get-s3object -BucketName example2bucket -StoredCredentials AccountBDave
```

4단계: 정리

1. 시험을 완료한 후, 다음과 같이 수행해 정리합니다.
   - 계정 A의 자격 증명을 사용하여 AWS Management 콘솔(AWS Management 콘솔)에 로그인하고 다음을 수행합니다.
     - Amazon S3 콘솔에서 *examplebucket*에 연결된 버킷 정책을 제거합니다. 버킷 속성의 권한 섹션에서 정책을 삭제합니다.
     - 이 연습을 위해 버킷을 만들었다면 Amazon S3 콘솔에서 객체를 삭제한 뒤 버킷을 삭제합니다.
     - IAM 콘솔에서 AccountAadmin 사용자를 삭제합니다.
2. 계정 B의 자격 증명을 사용하여 AWS Management 콘솔(AWS Management 콘솔)에 로그인합니다. IAM 콘솔에서 사용자 AccountBadmin을 삭제합니다.

예제 3: 버킷 소유자가 자신의 사용자에게 자신이 소유하지 않는 객체에 대한 권한 부여

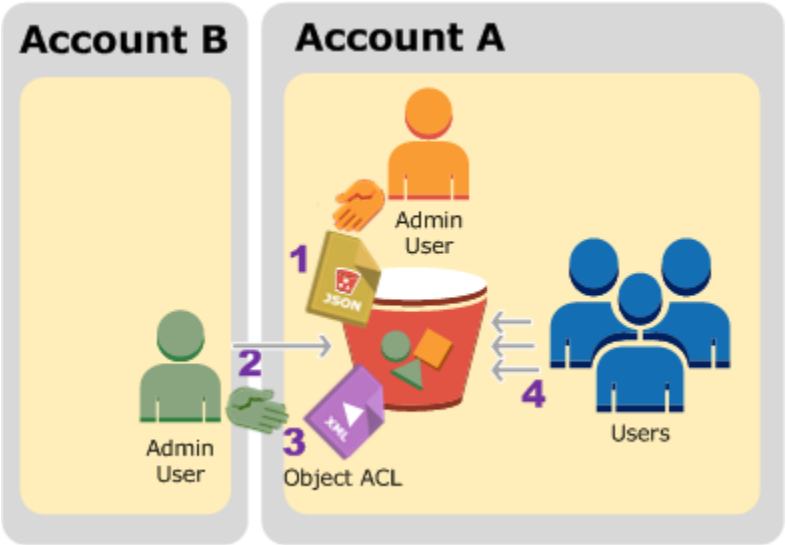
주제

- 0단계: 연습 준비 (p. 327)
- 1단계: 계정 A 작업 수행 (p. 328)
- 2단계: 계정 B 작업 수행 (p. 329)
- 3단계: 권한 테스트 (p. 330)
- 4단계: 정리 (p. 330)

이 예제의 시나리오는 버킷 소유자가 객체 액세스를 위해 권한을 부여하고자 하지만 버킷 소유자가 버킷의 일부 객체를 소유하지 않는다는 것입니다. 이 경우 버킷 소유자가 자신이 소유하지 않는 객체에 대한 권한을 부여하려면 어떻게 해야 할까요? 이 예제에서 버킷 소유자는 자신의 계정에 속한 사용자에게 권한을 부여하려고 합니다.

버킷 소유자는 객체를 업로드하기 위해 다른 AWS 계정을 활성화할 수 있습니다. 이러한 객체는 해당 객체를 만든 계정이 소유합니다. 버킷 소유자는 자신이 만들지 않은 객체를 소유하지 않습니다. 따라서 버킷 소유자가 이러한 객체에 대한 액세스 권한을 부여하려면 먼저 객체 소유자가 객체 ACL을 사용하여 버킷 소유자에게 권한을 부여해야 합니다. 자세한 내용은 Amazon S3 버킷 및 객체 소유권 (p. 300) 단원을 참조하십시오.

이 예제에서는 버킷 소유자가 자신의 계정에 속한 사용자에게 권한을 위임합니다. 각 연습 단계는 다음과 같이 간략히 설명할 수 있습니다.



1. 계정 A의 관리자 사용자가 두 설명문이 포함된 버킷 정책을 연결합니다.
   - 객체 업로드를 위해 계정 B에게 교차 계정 권한을 허용합니다.
   - 자신의 계정에 속한 사용자가 버킷의 객체에 액세스할 수 있게 허용합니다.
2. 계정 B의 관리자 사용자가 계정 A가 소유하는 버킷에 객체를 업로드합니다.
3. 계정 B의 관리자가 버킷 소유자에게 객체에 대한 모든 권한을 부여하는 허가를 추가하며 객체 ACL을 업데이트합니다.
4. 계정 A에 속한 사용자가 소유자에 관계없이 버킷의 객체에 액세스하여 이를 확인합니다.

이 예제에서는 두 계정이 필요합니다. 다음 표는 세 계정과 해당 계정의 관리자 사용자를 나타내는 방법을 보여 줍니다. IAM 지침(관리자 사용자를 사용하여 리소스를 만들고 권한을 부여하는 것에 대한 소개 (p. 314) 참조)에 따라 본 연습에서는 계정 루트 자격 증명을 사용하지 않습니다. 대신 사용자가 각 계정에서 관리자 사용자를 만든 후 해당 자격 증명을 사용하여 리소스를 만들고 해당 리소스에 권한을 부여합니다.

AWS 계정 ID	계정 이름	계정의 관리자 사용자
1111-1111-1111	계정 A	AccountAadmin
2222-2222-2222	계정 B	AccountBadmin

사용자를 만들고 권한을 부여하는 모든 작업은 AWS Management 콘솔에서 수행됩니다. 권한을 확인하기 위해 본 연습에서는 명령줄 도구, AWS 명령줄 인터페이스 및 Windows PowerShell용 AWS 도구를 사용합니다. 따라서 사용자가 코드를 작성할 필요가 없습니다.

0단계: 연습 준비

1. AWS 계정이 두 개 있으며 각 계정에 이전 단원의 표에 나와 있는 것과 같이 관리자 사용자가 한 명 있는지 확인합니다.

   a. 필요한 경우 AWS 계정에 가입합니다.

      i. https://aws.amazon.com/s3/로 이동하여 Create an AWS Account(AWS 계정 생성)를 클릭합니다.

      ii. 화면에 표시되는 지시 사항을 따릅니다. 계정이 활성화되고 사용 가능한 상태가 되면 AWS에서 사용자에게 이메일로 알립니다.

b. 계정 A의 자격 증명을 사용하여 IAM 콘솔에 로그인하고 다음을 수행하여 관리자 사용자를 만듭니다.

- 사용자 AccountAadmin을 만들고 보안 자격 증명을 적어 둡니다. 사용자 추가에 대한 자세한 내용은 IAM 사용 설명서의 AWS 계정의 IAM 사용자 생성 단원을 참조하십시오.
- 모든 권한을 제공하는 사용자 정책을 연결하여 AccountAadmin 관리자에게 권한을 부여합니다. 지침은 IAM 사용 설명서의 정책 작업 단원을 참조하십시오.
- IAM 콘솔의 대시보드에서 IAM User Sign-In URL(IAM User 로그인 URL)을 적어 둡니다. 해당 계정 사용자는 AWS Management 콘솔에 로그인할 때 이 URL을 사용해야 합니다. 자세한 내용은 IAM 사용 설명서의 사용자의 계정 로그인 방법 단원을 참조하십시오.

c. 계정 B의 자격 증명을 사용하여 이전 단계를 반복하고 관리자 사용자인 AccountBadmin을 만듭니다.

2. AWS 명령행 인터페이스(CLI)나 Windows PowerShell용 AWS 도구를 설정합니다. 관리자 사용자 자격 증명은 다음과 같이 저장합니다.

   - AWS CLI를 사용하는 경우 구성 파일에서 AccountAadmin과 AccountBadmin이라는 두 프로파일을 만듭니다.
   - Windows PowerShell용 AWS 툴을 사용하는 경우, 세션의 자격 증명을 AccountAadmin과 AccountBadmin으로 저장해야 합니다.

   지침은 예제 안내를 위한 도구 설정 (p. 315) 단원을 참조하십시오.

1단계: 계정 A 작업 수행

1.1단계: AWS Management Console에 로그인

계정 A의 IAM 사용자 로그인 URL을 사용하여 먼저 AccountAadmin 사용자로 AWS Management 콘솔에 로그인합니다. 이 사용자가 버킷을 만들고 버킷에 정책을 연결합니다.

1.2단계: 버킷 및 사용자를 만들고 사용자 권한을 부여하는 버킷 정책 추가

1. Amazon S3 콘솔에서 버킷을 만듭니다. 이 연습에서는 버킷이 미국 동부(버지니아 북부) 리전에서 생성되며 이름이 `examplebucket`이라고 가정합니다.

   지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷 정책을 생성하려면 어떻게 해야 합니까? 단원을 참조하십시오.

2. IAM 콘솔에서 사용자 Dave를 만듭니다.

   이에 관한 지침은 IAM 사용 설명서의 IAM 사용자 만들기(AWS Management Console) 단원을 참조하십시오.

3. Dave의 자격 증명을 적어 둡니다.
4. Amazon S3 콘솔에서 `examplebucket` 버킷에 다음 버킷 정책을 연결합니다. 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷 정책을 추가하려면 어떻게 해야 합니까? 단원을 참조하십시오. 단계를 따라 버킷 정책을 추가합니다. 계정 ID를 찾는 방법에 대한 자세한 내용은 AWS 계정 ID 찾기를 참조하십시오.

   이 정책은 계정 B에게 `s3:PutObject` 및 `s3:ListBucket` 권한을 부여합니다. 또한 이 정책은 사용자 Dave에게 `s3:GetObject` 권한을 부여합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Statement1",
         "Effect": "Allow",
```

```
            "Principal": {
               "AWS": "arn:aws:iam::AccountB-ID:root"
            },
            "Action": [
               "s3:PutObject",
               "s3:ListBucket"
            ],
            "Resource": [
            "arn:aws:s3:::examplebucket/*"
      "arn:aws:s3:::examplebucket"
            ]
         },
         {
            "Sid": "Statement3",
            "Effect": "Allow",
            "Principal": {
               "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
            },
            "Action": [
               "s3:GetObject"
            ],
            "Resource": [
               "arn:aws:s3:::examplebucket/*"
            ]
         }
      ]
}
```

2단계: 계정 B 작업 수행

이제 계정 B에게 계정 A의 버킷에 대한 작업을 수행할 수 있는 권한이 있으므로 계정 B의 관리자가 다음을 수행합니다.

- 계정 A의 버킷에 객체 업로드
- 객체 ACL에서 버킷 소유자인 계정 A에게 전체 제어 권한을 부여하는 허용을 추가합니다.

AWS CLI 사용

1. `put-object` AWS CLI 명령을 사용하여 객체를 업로드합니다. 명령의 `--body` 파라미터를 통해 업로드할 원본 파일이 식별됩니다. 예를 들어, 해당 파일이 Windows 시스템의 `c:` 드라이브에 있는 경우 `c:\HappyFace.jpg`를 지정합니다. `--key` 파라미터는 객체의 키 이름을 제공합니다.

   ```
   aws s3api put-object --bucket examplebucket --key HappyFace.jpg --body HappyFace.jpg --
   profile AccountBadmin
   ```

2. 객체 ACL에서 버킷 소유자에게 객체에 대한 모든 권한을 부여하는 허가를 추가합니다. 정식 사용자 ID를 찾는 방법에 대한 자세한 내용은 계정 정식 사용자 ID 찾기를 참조하십시오.

   ```
   aws s3api put-object-acl --bucket examplebucket --key HappyFace.jpg --grant-full-
   control id="AccountA-CanonicalUserID" --profile AccountBadmin
   ```

Windows PowerShell용 AWS 도구 사용

1. Windows PowerShell용 AWS 도구의 `Write-S3Object` 명령을 사용하여 객체를 업로드합니다.

   ```
   Write-S3Object -BucketName examplebucket -key HappyFace.jpg -file HappyFace.jpg -
   StoredCredentials AccountBadmin
   ```

2. 객체 ACL에서 버킷 소유자에게 객체에 대한 모든 권한을 부여하는 허가를 추가합니다.

```
Set-S3ACL -BucketName examplebucket -Key HappyFace.jpg -CannedACLName "bucket-owner-
full-control" -StoredCreden
```

3단계: 권한 테스트

이제 계정 A에 속한 사용자 Dave가 계정 B가 소유하는 객체에 액세스할 수 있는지 확인하십시오.

AWS CLI 사용

1. AWS CLI 구성 파일에 사용자 Dave의 자격 증명을 추가하고 새로운 프로파일인 `UserDaveAccountA`를 만듭니다. 자세한 내용은 예제 안내를 위한 도구 설정 (p. 315) 단원을 참조하십시오.

```
[profile UserDaveAccountA]
aws_access_key_id = access-key
aws_secret_access_key = secret-access-key
region = us-east-1
```

2. `get-object` AWS CLI 명령을 실행하여 `HappyFace.jpg`를 다운로드하고 로컬에 저장합니다. 사용자 Dave의 자격 증명은 `--profile` 파라미터를 추가하여 제공합니다.

```
aws s3api get-object --bucket examplebucket --key HappyFace.jpg Outputfile.jpg --
profile UserDaveAccountA
```

Windows PowerShell용 AWS 도구 사용

1. 사용자 Dave의 AWS 자격 증명을 UserDaveAccountA로 영구 스토어에 저장합니다.

```
Set-AWSCredentials -AccessKey UserDave-AccessKey -SecretKey UserDave-SecretAccessKey -
storeas UserDaveAccountA
```

2. Read-S3Object 명령을 실행하여 `HappyFace.jpg` 객체를 다운로드하고 로컬에 저장합니다. 사용자 Dave의 자격 증명은 `-StoredCredentials` 파라미터를 추가하여 제공합니다.

```
Read-S3Object -BucketName examplebucket -Key HappyFace.jpg -file HappyFace.jpg  -
StoredCredentials UserDaveAccountA
```

4단계: 정리

1. 시험을 완료한 후, 다음과 같이 수행해 정리합니다.
   - 계정 A의 자격 증명을 사용하여 AWS Management 콘솔(AWS Management 콘솔)에 로그인하고 다음을 수행합니다.
     - Amazon S3 콘솔에서 *`examplebucket`*에 연결된 버킷 정책을 제거합니다. 버킷 속성의 권한 섹션에서 정책을 삭제합니다.
     - 이 연습을 위해 버킷을 만들었다면 Amazon S3 콘솔에서 객체를 삭제한 뒤 버킷을 삭제합니다.
     - IAM 콘솔에서 AccountAadmin 사용자를 삭제합니다.
2. 계정 B의 자격 증명을 사용하여 AWS Management 콘솔(AWS Management 콘솔)에 로그인합니다. IAM 콘솔에서 사용자 AccountBadmin을 삭제합니다.

예 4: 버킷 소유자가 자신의 소유가 아닌 객체에 교차-계정 권한 부여

주제

- 배경: 교차 계정 권한과 IAM 역할 사용 (p. 331)
- 0단계: 연습 준비 (p. 332)
- 1단계: 계정 A 작업 수행 (p. 334)
- 2단계: 계정 B 작업 수행 (p. 336)
- 3단계: 계정 C 작업 수행 (p. 336)
- 4단계: 정리 (p. 338)
- 관련 리소스 (p. 338)

이러한 예로 사용자가 한 버킷을 소유하고 있고 다른 AWS 계정에 객체 업로드를 허용하는 시나리오를 살펴봅니다. 이는 소유자의 버킷에 다른 AWS 계정이 소유한 객체를 둘 수 있다는 것을 의미합니다.

이번에는 버킷 소유자가 객체 소유자에 관계 없이 객체에 대한 교차 계정 권한을 다른 계정 사용자에게 부여하는 시나리오를 살펴봅니다. 예를 들어, 해당 사용자는 객체 메타데이터에 액세스해야 하는 요금 청구 애플리케이션이 될 수 있습니다. 이 경우, 두 가지 핵심 사안이 있습니다.

- 버킷 소유자는 다른 AWS 계정이 만든 객체에 대한 권한이 없습니다. 그러므로 버킷 소유자가 자신의 소유가 아닌 객체에 대한 권한을 부여하려면, 우선 해당 객체를 만든 AWS 계정에서 버킷 소유자에게 권한을 부여해야 합니다. 그러면 버킷 소유자가 해당 권한을 위임할 수 있습니다.
- 버킷 소유자 계정은 본인 계정 사용자에게는 권한을 위임할 수 있으나 (예제 3: 버킷 소유자가 자신의 사용자에게 자신이 소유하지 않는 객체에 대한 권한 부여 (p. 326) 참조), 교차 계정 위임은 이를 지원하지 않아 다른 AWS 계정에는 권한을 위임할 수 없습니다.

이 시나리오에서 버킷 소유자는 객체 액세스 권한을 가진 AWS Identity and Access Management(IAM) 역할을 생성하고 다른 AWS 계정에 이 역할을 임시로 맡겨 버킷 객체 액세스 허용 권한을 부여할 수 있습니다.

배경: 교차 계정 권한과 IAM 역할 사용

IAM 역할은 사용자 리소스 액세스를 위임하는 여러 시나리오를 활성화하는 데 있으며, 교차 계정 액세스는 키 시나리오 중 한 가지에 해당합니다. 이번에는 버킷 소유자인 계정 A는 IAM 역할에 임시로 다른 AWS 계정 사용자인 계정 C에게 객체 액세스 교차 계정을 위임하는 시나리오를 살펴봅니다. 사용자가 만든 각 IAM 역할에 다음과 같이 두 개의 정책이 연결됩니다.

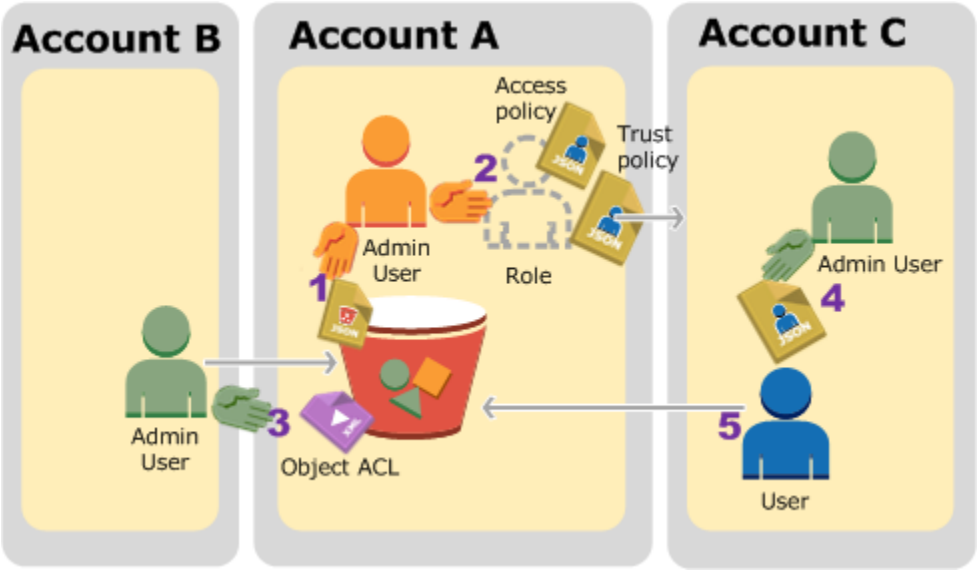
- 역할을 맡을 다른 AWS 계정을 식별하는 신뢰 정책.
- 사용자가 역할을 수임할 때 `s3:GetObject` 등 권한의 내용을 정의하는 액세스 정책. 정책에서 지정할 수 있는 권한 목록에 대해서는 Amazon S3 작업 (p. 342)을 참조하십시오.

신뢰 정책에서 식별된 AWS 계정은 사용자에게 해당 역할을 수임할 권한을 부여합니다. 사용자는 객체에 액세스하기 위해 다음과 같이 수행합니다.

- 역할을 맡으면 임시 보안 자격 증명이 주어집니다.
- 임시 보안 자격 증명으로 버킷 내 객체에 액세스합니다.

IAM 역할에 대한 자세한 내용은 IAM 사용 설명서의 IAM 역할 단원을 참조하십시오.

각 연습 단계는 다음과 같이 간략히 설명할 수 있습니다.



1. 계정 A 관리자 사용자는 계정 B에게 객체 업로드 조건부 권한을 부여하는 버킷 정책을 연결합니다.
2. 계정 A의 관리자가 IAM 역할을 만들어 계정 C와의 신뢰를 설정하면 계정 C 사용자는 계정 A에 액세스할 수 있습니다. 해당 역할에 연결된 액세스 정책은 사용자가 계정 A에 액세스할 때 계정 C의 사용자가 수행할 수 있는 작업을 제한합니다.
3. 계정 B의 관리자는 계정 A가 소유한 버킷에 객체를 업로드해 버킷 소유자에게 전적인 제어 권한을 부여합니다.
4. 계정 C 관리자는 사용자를 생성하고 사용자 정책을 연결해 해당 사용자에게 역할을 수임하게 합니다.
5. 계정 C의 사용자는 우선 역할을 맡고, 사용자 임시 보안 자격 증명을 반환합니다. 이러한 임시 자격 증명으로 사용자는 버킷 내 객체에 액세스합니다.

이번에는 세 계정이 필요한 경우를 예로 들어봅니다. 다음 표는 세 계정과 해당 계정의 관리자 사용자를 나타내는 방법을 보여 줍니다. IAM 지침(관리자 사용자를 사용하여 리소스를 만들고 권한을 부여하는 것에 대한 소개 (p. 314) 참조)에 따라 본 연습에서는 계정 루트 자격 증명을 사용하지 않습니다. 대신 사용자는 각 계정에서 관리자 사용자를 만든 후 해당 자격 증명을 사용해 리소스를 만들어 해당 리소스에 권한을 부여합니다.

AWS 계정 ID	계정 이름	계정의 관리자 사용자
1111-1111-1111	계정 A	AccountAadmin
2222-2222-2222	계정 B	AccountBadmin
3333-3333-3333	계정 C	AccountCadmin

0단계: 연습 준비

Note

텍스트 편집기를 열고 각 단계를 진행하면서 필요한 정보를 적는 것이 좋습니다. 계정 ID, 정식 사용자 ID, 콘솔에 연결할 각 계정의 IAM User Sign-in URL, IAM 사용자의 ARN(Amazon 리소스 이름)과 그 역할이 필요하기 때문입니다.

1. AWS 계정이 세 개인지, 이전 단원의 표와 같이 각 계정에 관리자 사용자가 한 명인지 확인합니다.

a. 필요한 경우 AWS 계정에 가입합니다. 이 계정을 계정 A, 계정 B, 계정 C라고 칭합니다.

   i. https://aws.amazon.com/s3/로 이동하여 Create an AWS Account(AWS 계정 생성)를 클릭합니다.

   ii. 화면에 표시되는 지시 사항을 따릅니다.

   계정이 활성화되고 사용 가능한 상태가 되면 AWS에서 사용자에게 이메일로 알립니다.

b. 계정 A의 자격 증명을 사용하여 IAM 콘솔에 로그인하고 다음을 수행하여 관리자 사용자를 만듭니다.

   - 사용자 AccountAadmin을 만들고 보안 자격 증명을 적어 둡니다. 사용자 추가에 대한 자세한 내용은 IAM 사용 설명서의 AWS 계정의 IAM 사용자 생성 단원을 참조하십시오.
   - 모든 권한을 제공하는 사용자 정책을 연결하여 AccountAadmin 관리자에게 권한을 부여합니다. 지침은 IAM 사용 설명서의 정책 작업 단원을 참조하십시오.
   - IAM Console 대시보드에 IAM User Sign-In URL(IAM 사용자 로그인 URL)을 적어 둡니다. 해당 계정 사용자는 AWS Management 콘솔에 로그인할 때 이 URL을 사용해야 합니다. 자세한 내용은 IAM 사용 설명서의 How Users Sign In to Your Account를 참조하십시오.

c. 이전 단계를 반복하며 계정 B와 계정 C에 속한 관리자 사용자를 만듭니다.

2. 계정 C의 경우 정식 사용자 ID를 적어 둡니다.

   계정 A에 IAM 역할을 생성하면 신뢰 정책에 따라 계정 ID를 지정해 계정 C에게 역할을 수임할 권한을 부여합니다. 계정 정보는 다음과 같이 찾을 수 있습니다.

   a. AWS 계정 ID나 계정 별칭, IAM 사용자 이름 및 암호를 사용하여 Amazon S3 콘솔에 로그인합니다.

   b. 버킷에 대한 세부 정보를 보려는 Amazon S3 버킷의 이름을 선택합니다.

   c. 권한 탭을 선택한 다음 액세스 제어 목록을 선택합니다.

   d. Access for your AWS account(AWS 계정 액세스) 섹션의 계정 열에 있는 것이 긴 식별자입니다(예: `c1daexampleaaf850ea79cf0430f33d72579fd1611c97f7ded193374c0b163b6`). 이것이 정식 사용자 ID입니다.

3. 버킷 정책을 만들려면 다음과 같은 정보가 필요합니다. 이 값을 적어 둡니다.

   - 계정 A의 정식 사용자 ID – 계정 A 관리자가 계정 B 관리자에게 조건부 업로드 객체 권한을 위임할 때, 해당 조건에 객체의 전적인 권한을 갖게 될 계정 A 사용자의 정식 사용자 ID를 지정합니다.

     Note

     정식 사용자 ID는 Amazon S3로만 사용합니다. 계정 ID의 64자 난독화 버전입니다.

   - 계정 B 관리자의 사용자 ARN – IAM 콘솔 사용자 ARN을 찾을 수 있습니다. 사용자를 선택하고 요약 탭에서 해당 사용자 ARN을 찾아야 합니다.

     버킷 정책에서 AccountBadmin에 객체 업로드 권한을 위임하고 ARN을 이용해 사용자를 지정합니다. 다음은 ARN 값의 예입니다.

     ```
     arn:aws:iam::AccountB-ID:user/AccountBadmin
     ```

4. AWS 명령행 인터페이스(CLI)나 Windows PowerShell용 AWS 도구를 설정합니다. 관리자 사용자 자격 증명은 다음과 같이 저장합니다.

   - AWS CLI를 사용하는 경우, 구성 파일에서 AccountAadmin과 AccountBadmin 프로필을 만듭니다.
   - Windows PowerShell용 AWS 툴을 사용하는 경우, 세션의 자격 증명을 AccountAadmin과 AccountBadmin으로 저장해야 합니다.

   지침은 예제 안내를 위한 도구 설정 (p. 315) 단원을 참조하십시오.

1단계: 계정 A 작업 수행

계정 A를 버킷 소유자로 예를 들어봅니다. 계정 A가 버킷 소유자이므로 계정 A 사용자 AccountAadmin은 버킷을 만들어 계정 B 관리자에게 객체 업로드 권한을 부여하는 버킷 정책을 연결하고, 계정 C에 역할을 수임할 권한을 부여하는 IAM 역할을 만들어 버킷 내 객체에 액세스할 수 있게 해줍니다.

1.1단계: AWS Management Console에 로그인

우선 계정 A의 IAM User Sign-in URL을 사용해 AccountAadmin 사용자로 AWS Management 콘솔에 로그인합니다. 이 사용자가 버킷을 만들고 버킷에 정책을 연결합니다.

1.2단계: 버킷 생성과 버킷 정책 연결

Amazon S3 콘솔에서 다음과 같이 수행합니다.

1. 버킷을 만듭니다. 이 연습에서는 버킷 이름을 `examplebucket`(으)로 가정합니다.

   지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷 정책을 생성하려면 어떻게 해야 합니까? 단원을 참조하십시오.
2. 다음의 버킷 정책에 연결해 계정 B 관리자에게 객체 업로드 조건부 권한을 위임합니다.

   examplebucket, *AccountB-ID*, *CanonicalUserId-of-AWSaccountA-BucketOwner*에 대한 자신의 사용자 값을 입력해 정책을 업데이트합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "111",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::examplebucket/*"
        },
        {
            "Sid": "112",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::examplebucket/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-grant-full-control": "id=CanonicalUserId-of-AWSaccountA-
BucketOwner"
                }
            }
        }
    ]
}
```

1.3단계: 계정 A에서 계정 C의 교차 계정 액세스를 허용하는 IAM 역할 생성

IAM 콘솔에서 계정 C에게 역할을 수임할 권한을 부여하는 IAM 역할("examplerole")을 생성합니다. 반드시 계정 A에서 역할을 만들어야 하므로 사용자는 계정 A 관리자로 로그인을 유지합니다.

1. 역할을 만들기 전에 먼저 해당 역할에 필요한 권한을 정의하는 관리형 정책을 준비합니다. 차후 단계에서 이 정책을 해당 역할에 연결합니다.

a. 왼쪽 탐색 창에서 정책을 클릭한 후 정책 생성을 클릭합니다.
b. Create Your Own Policy(자체 정책 생성) 옆의 선택을 클릭합니다.
c. 정책 이름 필드에 `access-accountA-bucket`을 입력합니다.
d. 다음 액세스 정책을 복사해 정책 문서 필드에 붙여 넣습니다. 액세스 정책은 이 역할에 `s3:GetObject` 권한을 부여해 계정 C 사용자가 해당 역할을 맡으면 `s3:GetObject` 작업만 수행할 수 있게 해줍니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::examplebucket/*"
    }
  ]
}
```

e. 정책 생성을 클릭합니다.

   새로운 정책이 관리형 정책 목록에 나타납니다.

2. 왼쪽 탐색 창에서 역할을 클릭한 뒤, 새 역할 생성을 클릭합니다.
3. 역할 유형 선택에서 Role for Cross-Account Access(교차 계정 액세스를 위한 역할)를 선택한 뒤, Provide access between AWS accounts you own(소유한 AWS 계정 간 액세스 제공) 옆의 선택 버튼을 클릭합니다.
4. 계정 C 계정 ID를 입력합니다.

   이 연습에서는 사용자가 역할을 수임할 MFA(멀티 팩터 인증)를 필요로 하지 않으므로 해당 옵션을 선택하지 않고 그대로 둡니다.
5. 다음 단계를 클릭해 역할과 연결되는 권한을 설정합니다.
6. 사용자가 만든 `access-accountA-bucket` 정책 옆의 상자를 선택한 뒤, 다음 단계를 클릭합니다.

   Review 페이지를 보면서 역할이 완전히 생성되기 전에 해당 설정을 확인할 수 있습니다. 이 페이지에서 확인할 한 가지 주요 항목은 해당 역할을 사용해야 하는 사용자에게 보낼 수 있는 링크입니다. 사용자가 링크를 클릭하면 계정 ID와 기재된 역할 이름 필드로 된 Switch Role 페이지로 바로 연결됩니다. 이 링크는 모든 교차 계정 역할에 대한 Role Summary 페이지에서도 확인할 수 있습니다.
7. 역할 이름에 `examplerole`을 입력한 뒤, 다음 단계를 클릭합니다.
8. 역할을 검토하고 역할 생성을 클릭합니다.

   역할 목록에 `examplerole` 역할이 표시됩니다.
9. 역할 이름 [`examplerole`]을 클릭합니다.
10. 신뢰 관계 탭을 선택합니다.
11. Show policy document(정책 문서 표시)를 클릭해 나타나는 신뢰 정책이 다음 정책과 일치하는지 확인합니다.

    다음 신뢰 정책으로 계정 C에 `sts:AssumeRole` 작업을 허용해 계정 C와의 신뢰를 설정합니다. 자세한 내용은 AWS Security Token Service API Reference의 AssumeRole을 참조하십시오.

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

```
        "AWS": "arn:aws:iam::AccountC-ID:root"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

12. 사용자가 생성한 `examplerole` 역할의 Amazon 리소스 이름(ARN)을 적어 둡니다.

    다음 단계 후반에서, IAM 사용자에 이 역할의 수임을 허용할 사용자 정책을 연결하고 ARN 값으로 역할을 식별합니다.

2단계: 계정 B 작업 수행

계정 A가 소유한 examplebucket은 다른 계정 소유의 객체가 필요합니다. 이 단계에서 계정 B 관리자는 명령행 도구를 이용해 객체를 업로드합니다.

- put-object AWS CLI 명령으로 `examplebucket`에 객체를 업로드합니다.

```
aws s3api put-object --bucket examplebucket --key HappyFace.jpg --body HappyFace.jpg --
grant-full-control id="canonicalUserId-ofTheBucketOwner" --profile AccountBadmin
```

  다음을 참조하십시오.

  - `--Profile` 파라미터가 AccountBadmin 프로필을 지정하면 계정 B가 객체를 소유하게 됩니다.
  - 파라미터 `grant-full-control`은 버킷 정책에 따라 버킷 소유자에게 객체에 대한 전적인 권한을 부여합니다.
  - `--body` 파라미터가 업로드할 원본 파일을 식별합니다. 예를 들어, 파일이 Windows 컴퓨터의 C: 드라이브에 있는 경우 `c:\HappyFace.jpg`로 지정합니다.

3단계: 계정 C 작업 수행

이전 단계에서 계정 A가 역할 `examplerole`을 만들어 계정 C와의 신뢰를 설정했기 때문에 계정 C 사용자는 계정 A에 액세스할 수 있습니다. 이번 단계에서는 계정 C 관리자가 사용자(Dave)를 생성해 계정 A에서 받은 `sts:AssumeRole` 권한을 Dave에게 위임합니다. Dave는 이 권한으로 `examplerole`을 수임해 계정 A에 임시로 액세스할 수 있습니다. 계정 A가 이 역할에 연결한 액세스 정책은 Dave가 계정 A에 액세스해 할 수 있는 활동을 제한하는데, 그 중에서도 특히 `examplebucket`의 객체에 대한 활동을 제한합니다.

3.1단계: 계정 C 사용자 생성과 `examplerole` 담당 권한 위임

1. 먼저 계정 C의 IAM 사용자 로그인 URL을 사용해 AccountCadmin 사용자로 AWS Management 콘솔에 로그인합니다.
2. IAM 콘솔에서 사용자 Dave를 만듭니다.

   이에 관한 지침은 IAM 사용 설명서의 IAM 사용자 만들기(AWS Management Console) 단원을 참조하십시오.
3. Dave의 자격 증명을 적어 둡니다. Dave가 `examplerole` 역할을 수임하려면 이 자격 증명이 필요합니다.
4. Dave IAM 사용자에 대한 인라인 정책을 만들고 계정 A의 `sts:AssumeRole` 역할에 대한 `examplerole` 권한을 Dave에게 위임합니다.

   a. 왼쪽 탐색 창에서 사용자를 클릭합니다.
   b. 사용자 이름 Dave를 클릭합니다.
   c. 사용자 세부 정보 페이지에서 권한 탭을 선택한 다음 Inline Policies(인라인 정책) 단원을 확장합니다.

d. 여기를 클릭하십시오 또는 Create User Policy(사용자 정책 생성)를 선택합니다.
e. 사용자 지정 정책을 클릭한 후 선택을 클릭합니다.
f. 정책 이름 필드에 정책 이름을 입력합니다.
g. 다음 정책을 정책 문서 필드에 복사해 넣습니다.

   계정 A ID를 입력해 정책을 업데이트합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["sts:AssumeRole"],
      "Resource": "arn:aws:iam::AccountA-ID:role/examplerole"
    }
  ]
}
```

h. Apply Policy(정책 적용)를 클릭합니다.

5. 또 다른 프로필 AccountCDave를 추가해 AWS CLI의 구성 파일에 Dave의 자격 증명을 저장합니다.

```
[profile AccountCDave]
aws_access_key_id = UserDaveAccessKeyID
aws_secret_access_key = UserDaveSecretAccessKey
region = us-west-2
```

3.2단계: 역할(examplerole) 수임과 액세스 객체

이제 Dave는 다음과 같이 계정 A가 소유한 버킷 내 객체에 액세스할 수 있습니다.

- Dave는 먼저 자신의 자격 증명으로 `examplerole`을 수임합니다. 이 때 임시 자격 증명이 반환됩니다.
- 임시 자격 증명으로 Dave는 계정 A의 버킷 내 객체에 액세스합니다.

1. 명령 프롬프트에서 AccountCDave 프로필로 다음 AWS CLI `assume-role` 명령을 실행합니다.

   `examplerole`가 정의된 계정 A ID를 입력해 명령의 ARN 값을 업데이트합니다.

```
aws sts assume-role --role-arn arn:aws:iam::accountA-ID:role/examplerole --profile
 AccountCDave --role-session-name test
```

   이에 응답하여 AWS Security Token Service(STS)가 임시 보안 자격 증명(액세스 키 ID, 보안 액세스 키 및 세션 토큰)을 반환합니다.

2. 임시 보안 자격 증명을 `TempCred` 프로필의 AWS CLI 구성 파일에 저장합니다.

```
[profile TempCred]
aws_access_key_id = temp-access-key-ID
aws_secret_access_key = temp-secret-access-key
aws_session_token = session-token
region = us-west-2
```

3. 명령 프롬프트에서 임시 자격 증명으로 다음 AWS CLI 명령을 실행해 객체에 액세스합니다. 예를 들어, 명령으로 헤드 객체 API를 지정하고 `HappyFace.jpg` 객체에 대한 객체 메타데이터를 검색합니다.

```
aws s3api get-object --bucket examplebucket --key HappyFace.jpg SaveFileAs.jpg --
profile TempCred
```

`examplerole`에 연결된 액세스 정책에서 해당 작업을 허용해주어 Amazon S3가 액세스 요청을 처리합니다. 버킷 내 객체에 모든 작업 수행이 가능합니다.

예를 들어 `get-object-acl` 등 다른 작업을 시도할 경우, 해당 작업이 허용되지 않는 역할이기 때문에 거부당합니다.

```
aws s3api get-object-acl --bucket examplebucket --key HappyFace.jpg --profile TempCred
```

사용자 Dave로 그 역할을 수임해 임시 자격 증명으로 객체에 액세스합니다. `examplebucket`의 객체에 액세스하는 계정 C의 애플리케이션으로도 엑세스가 가능합니다. 애플리케이션으로 임시 보안 자격 증명을 얻을 수 있습니다. 또한, 계정 C는 애플리케이션에 `examplerole`을 맡을 권한을 위임할 수 있습니다.

4단계: 정리

1. 시험을 완료한 후, 다음과 같이 수행해 정리합니다.
   - 계정 A의 자격 증명으로 AWS Management 콘솔(AWS Management 콘솔)에 로그인하고 다음과 같이 수행합니다.
     - Amazon S3 콘솔에서 *`examplebucket`*에 연결된 버킷 정책을 제거합니다. 버킷 속성의 권한 섹션에서 정책을 삭제합니다.
     - 이 연습을 위해 버킷을 만들었다면 Amazon S3 콘솔에서 객체를 삭제한 뒤 버킷을 삭제합니다.
     - IAM 콘솔에서 계정 A로 만든 `examplerole`을 삭제합니다.
     - IAM 콘솔에서 AccountAadmin 사용자를 삭제합니다.
2. 계정 B의 자격 증명을 사용하여 AWS Management 콘솔(AWS Management 콘솔)에 로그인합니다. IAM 콘솔에서 사용자 AccountBadmin을 삭제합니다.
3. 계정 C의 자격 증명으로 AWS Management 콘솔(AWS Management 콘솔)에 로그인합니다. IAM 콘솔에서 사용자 AccountCadmin와 사용자 Dave를 삭제합니다.

관련 리소스

- IAM 사용 설명서의 역할을 만들어 IAM 사용자에게 권한 위임 단원.
- IAM 사용 설명서의 Tutorial: Delegate Access Across AWS Accounts Using IAM Roles 단원.
- IAM 사용 설명서의 정책 작업 단원.

버킷 정책 및 사용자 정책 사용

버킷 정책과 사용자 정책은 Amazon S3 리소스에 권한을 부여하는 데 사용할 수 있는 두 가지 액세스 정책 옵션입니다. 두 가지 옵션 모두 JSON 기반 액세스 정책 언어를 사용합니다.

이 단원의 주제에서는 주요 정책 언어 요소에 대해 설명하는 한편, Amazon S3에 관한 세부 정보를 중점적으로 살펴보면서 버킷 및 사용자 정책 예제를 제공합니다.

> **Important**
>
> 버킷 정책은 크기가 20KB로 제한됩니다.
> Amazon S3 리소스에 대한 액세스 관리를 위해 제공되는 기본 개념과 옵션 설명이 나온 소개 주제 부분을 먼저 읽는 것이 좋습니다. 자세한 내용은 Amazon S3 리소스에 대한 액세스 관리 소개 (p. 299) 단원을 참조하십시오.

주제

- Amazon S3의 정책 및 권한 (p. 339)
- 버킷 정책 예제 (p. 408)
- 사용자 정책 예제 (p. 416)

Amazon S3의 정책 및 권한

이 페이지에서는 Amazon S3의 버킷 및 사용자 정책에 대한 개요를 제공하고 정책의 기본 요소에 대해 설명합니다. 나열된 각 요소는 해당 요소에 대한 자세한 내용과 사용 방법의 예제로 연결됩니다.

Amazon S3 작업, 리소스 및 조건의 전체 목록은 Amazon S3에 사용되는 작업, 리소스 및 조건 키 (p. 356) 단원을 참조하십시오.

정책 언어 개요

가장 기본적인 경우 정책에는 다음 요소가 포함되어 있습니다.

- 리소스 (p. 340) – 버킷, 객체, 액세스 포인트 및 작업은 권한을 허용하거나 거부할 수 있는 Amazon S3 리소스입니다. 정책에서는 Amazon 리소스 이름(ARN)을 사용하여 리소스를 식별해야 합니다. 자세한 내용은 Amazon S3 리소스 (p. 340) 단원을 참조하십시오.
- 작업 (p. 342) – 각 리소스에 대해 Amazon S3에서는 작업의 집합을 지원합니다. 작업 키워드를 사용하여 허용(또는 거부)할 리소스 작업을 식별합니다.

  예를 들어 `s3:ListBucket` 권한을 통해 사용자는 Amazon S3 GET Bucket(List Objects) 작업을 허용할 수 있습니다. 자세한 내용은 Amazon S3 작업 (p. 342) 단원을 참조하십시오.
- 효과 – 사용자가 특정 작업을 요청하는 경우의 결과입니다. 이는 허용 또는 거부 중에 하나가 될 수 있습니다.

  명시적으로 리소스에 대한 액세스 권한을 부여(허용)하지 않는 경우, 액세스는 암시적으로 거부됩니다. 한 리소스에 대한 액세스를 명시적으로 거부할 수도 있으며, 다른 정책에 따라 액세스 권한이 부여되더라도 사용자가 이 리소스에 액세스하지 못하도록 이 작업을 수행할 수 있습니다. 자세한 내용은 IAM JSON 정책 요소: 효과를 참조하십시오.
- 보안 주체 (p. 341) – 문에서의 작업 및 리소스에 액세스할 수 있는 계정 또는 사용자입니다. 버킷 정책에서 보안 주체는 사용자, 계정, 서비스 또는 이 권한의 수신자인 기타 주체입니다. 자세한 내용은 Principal (p. 341) 단원을 참조하십시오.
- 조건 (p. 345) – 정책이 적용되기 위한 조건입니다. AWS-와이드 키와 Amazon S3-특정 키를 사용하여 Amazon S3 액세스 정책에서 조건을 지정할 수 있습니다. 자세한 내용은 Amazon S3 조건 키 (p. 345) 단원을 참조하십시오.

다음 버킷 정책 예제는 전술한 공통 정책 요소를 나타냅니다. 이 정책에서는 *`Account-ID`* 계정의 사용자인 Dave에게 `examplebucket` 버킷의 `s3:GetObject`, `s3:GetBucketLocation` 및 `s3:ListBucket` Amazon S3 권한을 허용합니다.

```
{
    "Version": "2012-10-17",
    "Id": "ExamplePolicy01",
    "Statement": [
        {
            "Sid": "ExampleStatement01",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::Account-ID:user/Dave"
            },
            "Action": [
                "s3:GetObject",
                "s3:GetBucketLocation",
                "s3:ListBucket"
```

```
            ],
            "Resource": [
                "arn:aws:s3:::examplebucket/*",
                "arn:aws:s3:::examplebucket"
            ]
        }
    ]
}
```

전체 정책 언어 정보는 IAM 사용 설명서의 정책 및 권한과 IAM JSON 정책 참조를 참조하십시오.

Amazon S3 리소스

다음과 같은 일반적인 Amazon 리소스 이름(ARN) 형식은 AWS에서 리소스를 식별합니다.

```
arn:partition:service:region:namespace:relative-id
```

ARN에 대한 자세한 내용은 AWS General Reference의 Amazon 리소스 이름(ARN)을 참조하십시오.

리소스에 대한 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소: 리소스를 참조하십시오.

Amazon S3 ARN은 AWS 리전 및 네임스페이스를 제외하지만 다음을 포함합니다.

- 파티션 - `aws` 공통 파티션 이름입니다. 리소스가 중국(베이징) 리전에 있는 경우, `aws-cn`이 파티션 이름입니다.
- 서비스 - `s3`.
- 상대 ID - `bucket-name` 또는 `bucket-name/object-key`. 와일드카드를 사용할 수 있습니다.

Amazon S3 리소스에 대한 ARN 형식을 다음과 같이 줄입니다.

```
arn:aws:s3:::bucket_name/key_name
```

전체 Amazon S3 리소스 목록은 Amazon S3에 사용되는 작업, 리소스 및 조건 키 (p. 356) 단원을 참조하십시오.

S3 버킷의 ARN을 찾아보려면 Amazon S3 콘솔 버킷 정책 또는 CORS 구성 권한 페이지를 확인해 보십시오. 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서에서 다음 주제를 참조하십시오.

- S3 버킷 정책을 추가하려면 어떻게 해야 합니까?
- CORS와의 교차 도메인 리소스 공유를 허용하려면 어떻게 해야 합니까?

Amazon S3 ARN 예제

다음은 Amazon S3 리소스 ARN의 예입니다.

버킷 이름 및 객체 키 지정

다음 ARN은 `examplebucket` 버킷의 `/developers/design_info.doc` 객체를 식별합니다.

```
arn:aws:s3:::examplebucket/developers/design_info.doc
```

와일드카드

와일드카드는 리소스 ARN에도 사용할 수 있습니다. ARN 세그먼트(콜론으로 구분된 부분) 내에서 와일드카드 문자(`*`과 `?`)를 사용할 수 있습니다. 별표(`*`)는 0개 이상의 문자 조합을 나타내고 물음표(`?`)는 단일 문자를

나타냅니다. * 또는 ? 문자를 각 세그먼트에서 여러 번 사용할 수 있지만, 와일드카드 한 개를 여러 세그먼트에 걸쳐서 적용할 수는 없습니다.

- 다음 ARN에서는 ARN의 상대 ID 부분에 와일드카드 문자 *를 사용하여 examplebucket 버킷의 모든 객체를 나타냅니다.

```
arn:aws:s3:::examplebucket/*
```

- 다음 ARN에서는 *를 사용하여 모든 Amazon S3 리소스(계정의 모든 S3 버킷 및 객체)를 나타냅니다.

```
arn:aws:s3:::*
```

- 다음 ARN에서는 relative-ID 부분에 와일드카드 문자 * 및 ?를 모두 사용합니다. 이를 통해 example1bucket, example2bucket, example3bucket 등 버킷의 모든 객체를 나타냅니다.

```
arn:aws:s3:::example?bucket/*
```

정책 변수

Amazon S3 ARN에서 정책 변수를 사용할 수 있습니다. 정책 평가 시에는 이러한 사전 정의된 변수가 해당 값으로 대체됩니다. 폴더 하나당 각 사용자용으로 할당된 폴더의 모음으로 버킷을 구성한다고 가정합니다. 폴더 이름은 사용자 이름과 같습니다. 해당 폴더에 대한 사용자 권한을 부여하려면 다음과 같이 리소스 ARN에 정책 변수를 지정하면 됩니다.

```
arn:aws:s3:::bucket_name/developers/${aws:username}/
```

런타임 시 정책이 평가되면 리소스 ARN의 ${aws:username} 변수가 요청을 제기하는 사용자 이름으로 대체됩니다.

Principal

Principal 요소는 사용자, 계정, 서비스 또는 리소스에 대한 액세스가 허용 또는 거부된 기타 주체를 지정합니다. 다음은 Principal 지정의 예입니다. 자세한 내용은 IAM 사용 설명서의 보안 주체를 참조하십시오.

AWS 계정에 권한 부여

AWS 계정에 권한을 부여하려면 다음 형식을 사용하여 계정을 식별합니다.

```
"AWS":"account-ARN"
```

예를 들면 다음과 같습니다.

```
"Principal":{"AWS":"arn:aws:iam::AccountNumber-WithoutHyphens:root"}
```

```
"Principal":{"AWS":["arn:aws:iam::AccountNumber1-
WithoutHyphens:root","arn:aws:iam::AccountNumber2-WithoutHyphens:root"]}
```

또한 Amazon S3에서는 난독화된 형식의 AWS 계정 ID인 정식 사용자 ID를 지원합니다. 다음 형식을 사용하여 이 ID를 지정할 수 있습니다.

```
"CanonicalUser":"64-digit-alphanumeric-value"
```

다음은 예제입니다.

```
"Principal":{"CanonicalUser":"64-digit-alphanumeric-value"}
```

계정의 정식 사용자 ID를 찾는 방법에 대한 정보는 계정 정식 사용자 ID 찾기를 참조하십시오.

> **Important**
>
> 정책에 정식 사용자 ID를 사용하는 경우 Amazon S3가 정식 ID를 해당 AWS 계정 ID로 변경할 수 있습니다. 이러한 ID는 둘 다 동일한 계정을 식별하기 때문에 정책에 영향을 주지는 않습니다.

IAM 사용자에게 권한 부여

계정 내의 IAM 사용자에 대한 권한을 부여하려면 `"AWS":"user-ARN"` 이름-값 페어를 제공해야 합니다.

```
"Principal":{"AWS":"arn:aws:iam::account-number-without-hyphens:user/username"}
```

익명 권한 부여

모두에게 권한을 부여하고 익명 액세스로도 참조되려면 와일드카드 문자(`"*"`)를 `Principal` 값으로 설정합니다. 예를 들어 버킷을 웹 사이트로 구성하는 경우 버킷의 모든 객체를 공개적으로 액세스 가능하게 하려 할 수 있습니다. 다음은 동일합니다.

```
"Principal":"*"
```

```
"Principal":{"AWS":"*"}
```

> **Warning**
>
> S3 버킷에 익명 액세스 권한을 부여할 때는 주의해야 합니다. 익명 액세스 권한을 부여하면 전 세계 누구나 버킷에 액세스할 수 있습니다. S3 버킷에 대한 익명 쓰기 액세스 권한을 부여하지 않는 것이 좋습니다.

CloudFront URL을 통한 액세스 필요

사용자가 Amazon S3 URL 대신 Amazon CloudFront URL을 사용하여 Amazon S3 콘텐츠에 액세스하도록 요구할 수 있습니다. 이렇게 하려면 CloudFront 원본 액세스 ID(OAI)를 만듭니다. 그런 다음 버킷 또는 버킷의 객체에 대한 권한을 변경합니다. `Principal` 문에서 OAI를 지정하기 위한 형식은 다음과 같습니다.

```
"Principal":{"CanonicalUser":"Amazon S3 Canonical User ID assigned to origin access
 identity"}
```

자세한 내용은 Amazon CloudFront 개발자 안내서의 원본 액세스 ID를 사용하여 Amazon S3 콘텐츠에 대한 액세스 제한 단원을 참조하십시오.

Amazon S3 작업

Amazon S3에서는 정책에서 지정할 수 있는 권한 집합을 정의합니다. 이것은 특정 Amazon S3 작업에 각각 매핑되는 키워드입니다. Amazon S3 작업에 대한 자세한 내용은 Amazon Simple Storage Service API Reference의 작업을 참조하십시오.

Amazon S3 정책에서 권한을 지정하는 방법을 보려면 다음 정책 예제를 검토하십시오. 전체 Amazon S3 권한 목록은 Amazon S3에 사용되는 작업, 리소스 및 조건 키 (p. 356) 단원을 참조하십시오.

주제

- — 객체 작업 예제 (p. 343)
- — 버킷 작업 예제 (p. 343)
- — 버킷 하위 리소스 작업 예제 (p. 344)
- — 계정 작업 예제 (p. 344)

— 객체 작업 예제

다음 버킷 정책 예제에서는 사용자(Dave)에게 `s3:PutObject` 및 `s3:PutObjectAcl` 권한을 부여합니다. `Principal` 요소를 제거하면 이 정책을 사용자에게 연결할 수 있습니다. 이러한 작업은 객체 작업입니다. 따라서 `Resource` ARN `relative-id` 부분은 객체(`examplebucket/*`)를 식별합니다. 자세한 내용은 Amazon S3 리소스 (p. 340) 단원을 참조하십시오.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/Dave"
            },
            "Action":   ["s3:PutObject","s3:PutObjectAcl"],
            "Resource": "arn:aws:s3:::examplebucket/*"
        }
    ]
}
```

모든 Amazon S3 작업에 대한 권한

와일드카드를 사용하여 전체 Amazon S3 작업에 대한 권한을 부여할 수 있습니다.

```
"Action":   "*"
```

— 버킷 작업 예제

다음 사용자 정책 예제에서는 사용자에게 `s3:CreateBucket`, `s3:ListAllMyBuckets` 및 `s3:GetBucketLocation` 권한을 부여합니다. 이러한 모든 권한에 대해 `Resource` ARN의 `relative-id` 일부를 “*”로 설정합니다. 다른 모든 버킷 작업에 대해 버킷 이름을 지정해야 합니다. 자세한 내용은 Amazon S3 리소스 (p. 340) 단원을 참조하십시오.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action":[
            "s3:CreateBucket",
            "s3:ListAllMyBuckets",
            "s3:GetBucketLocation"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ]
       }
    ]
}
```

콘솔 액세스에 대한 정책

사용자가 AWS Management 콘솔을 사용하여 버킷과 해당 버킷의 내용을 보려면 `s3:ListAllMyBuckets` 및 `s3:GetBucketLocation` 권한이 있어야 합니다. 이에 대한 예제는 블로그 게시물 IAM 정책 작성: Amazon S3 버킷에 대한 액세스를 허용하는 방법에서 콘솔 액세스 정책을 참조하십시오.

— 버킷 하위 리소스 작업 예제

다음 사용자 정책은 사용자 Dave에게 `s3:GetBucketAcl` 버킷에 대한 `examplebucket` 권한을 부여합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "statement1",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::Account-ID:user/Dave"
      },
      "Action": [
        "s3:GetObjectVersion",
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::examplebucket"
    }
  ]
}
```

객체 권한 삭제

Amazon S3에서 객체의 수명이 만료되었을 때 이를 제거할 수 있도록 명시적으로 DELETE Object API를 호출하거나 해당 수명 주기(객체 수명 주기 관리 (p. 123) 참조)를 구성하여 객체를 삭제할 수 있습니다. 객체를 삭제하지 못하도록 명시적으로 사용자 또는 계정을 차단하려면 이들에 대해 `s3:DeleteObject`, `s3:DeleteObjectVersion` 및 `s3:PutLifecycleConfiguration` 권한을 명시적으로 거부합니다.

명시적 거부

기본적으로 사용자에게는 아무 권한이 없습니다. 그러나 사용자를 만들고 그룹에 사용자를 추가하고 권한을 부여할 때 사용자에게 부여하지 않으려는 특정 권한이 부여될 수 있습니다. 이는 명시적 거부를 사용할 수 있는 경우로, 사용자가 보유할 수 있는 다른 모든 권한에 우선하며 특정 작업에 대한 사용자 권한을 거부합니다.

— 계정 작업 예제

다음 사용자 정책 예제에서는 사용자에게 `s3:GetAccountPublicAccessBlock` 권한을 부여합니다. 이러한 권한에 대해 `Resource` 값을 `"*"`로 설정합니다. 자세한 내용은 Amazon S3 리소스 (p. 340) 단원을 참조하십시오.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action":[
            "s3:GetAccountPublicAccessBlock"
         ],
         "Resource":[
            "*"
         ]
```

```
        }
    ]
}
```

Amazon S3 조건 키

권한 부여 시 액세스 정책 언어를 통해 조건을 지정할 수 있습니다. 정책이 적용되기 위한 조건을 지정하려면 선택적 `Condition` 요소 또는 `Condition` 블록을 사용하여 정책이 적용되기 위한 조건을 지정할 수 있습니다. 미리 정의된 AWS-와이드 키와 Amazon S3-특정 키를 사용하여 Amazon S3 액세스 정책에 조건을 지정할 수 있습니다.

`Condition` 요소에서 요청의 값과 조건을 매칭하기 위해 부울 연산자(equal, less than 등)를 사용하는 식을 작성할 수 있습니다. 예를 들어 객체를 업로드하는 사용자 권한을 부여하는 경우, 버킷 소유자는 다음과 같이 `StringEquals` 조건을 추가하여 객체가 공개적으로 읽기 가능하도록 요구할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "statement1",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::examplebucket/*"
      ],
      "Condition": {
        "StringEquals": {
          "s3:x-amz-acl": [
            "public-read"
          ]
        }
      }
    }
  ]
}
```

이 예제에서 `Condition` 블록은 지정된 키-값 페어 `"s3:x-amz-acl":["public-read"]`에 적용되는 `StringEquals` 조건을 지정합니다. 조건을 나타내는 데에 사용할 수 있는 사전 정의된 키 집합이 있습니다. 이 예제에서는 `s3:x-amz-acl` 조건 키를 사용합니다. 이 조건에서는 사용자에게 모든 PUT Object 요청에 `public-read` 값을 가진 `x-amz-acl` 헤더를 포함하도록 요구합니다.

주제

- AWS- 차원 조건 키 (p. 345)
- Amazon S3- 전용 조건 키 (p. 346)
- 객체 작업에 대한 예제 — Amazon S3 조건 키 (p. 347)
- 버킷 작업에 대한 예제 — Amazon S3 조건 키 (p. 352)

AWS- 차원 조건 키

AWS에서는 정책을 지원하는 전체 AWS 서비스에서 지원되는 공통 키 집합을 제공합니다. 이러한 키는AWS- 차원 키라고 하며 접두사 `aws:`를 사용합니다. AWS- 차원 조건 키의 전체 목록은 IAM 사용 설명서의 사용 가능한 AWS조건 키를 참조하십시오.

Amazon S3에서 AWS- 차원 조건 키를 사용할 수 있습니다. 다음 버킷 정책 예제에서는 요청이 특정 범위의 IP 주소(192.0.2.0.*)에서 비롯된 경우 `s3:GetObject` 작업을 사용하기 위한 인증된 사용자 권한을 허용합니다. 단, IP 주소가 192.0.2.188인 경우는 예외입니다. 조건 블록에서 `IpAddress` 및 `NotIpAddress`가

조건에 해당하며 각 조건에는 평가를 위한 키-값 페어가 제공됩니다. 이 예제에서의 키-값 페어는 모두 `aws:SourceIp` AWS-라는 차원의 키를 사용합니다.

> **Note**
>
> 조건에 지정된 `IPAddress` 및 `NotIpAddress` 키 값은 RFC 4632에 설명된 것과 같은 CIDR 표기법을 사용합니다. 자세한 내용은 http://www.rfc-editor.org/rfc/rfc4632.txt를 참조하십시오.

```
{
    "Version": "2012-10-17",
    "Id": "S3PolicyId1",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": "*",
            "Action":["s3:GetObject"]  ,
            "Resource": "arn:aws:s3:::examplebucket/*",
            "Condition" : {
                "IpAddress" : {
                    "aws:SourceIp": "192.0.2.0/24"
                },
                "NotIpAddress" : {
                    "aws:SourceIp": "192.0.2.188/32"
                }
            }
        }
    ]
}
```

Amazon S3 정책에서 다른 AWS- 차원 조건 키를 사용할 수도 있습니다. 예를 들어 VPC 엔드포인트에 대한 버킷 정책에서 `aws:sourceVpce` 및 `aws:sourceVpc` 조건 키를 지정할 수 있습니다. 예제는 Amazon S3의 VPC 엔드포인트에 대한 예제 버킷 정책 (p. 414)을 참조하십시오.

Amazon S3- 전용 조건 키

특정 Amazon S3 작업에 Amazon S3 조건 키를 사용할 수 있습니다. 각 조건 키는 조건을 설정할 수 있는 API에서 허용하는 동일한 이름 요청 헤더에 매핑됩니다. Amazon S3- 전용 조건 키는 동일한 이름 요청 헤더의 동작을 지시합니다. Amazon S3- 전용 조건 키의 전체 목록은 Amazon S3에 사용되는 작업, 리소스 및 조건 키 (p. 356) 단원을 참조하십시오.

예를 들어
`s3:PutObject`
권한에 대한 조건 권한을 부여하는 데 사용할 수 있는 조건 키 `s3:x-amz-acl`은 PUT Object API에서 지원되는 `x-amz-acl` 요청 헤더의 동작을 정의합니다. `s3:VersionId` 권한에 대한 조건부 권한을 부여하는 데 사용할 수 있는 조건 키
`s3:GetObjectVersion`
는 GET Object 요청에서 설정한 `versionId` 쿼리 파라미터의 동작을 정의합니다.

다음 버킷 정책에서는 객체를 공개적으로 읽기 가능하도록 만드는 `x-amz-acl` 헤더가 요청에 포함된 경우 두 개의 AWS 계정에 대해 `s3:PutObject` 권한을 허용합니다. `Condition` 블록에서는 `StringEquals` 조건을 사용하며 여기에는 평가를 위한 키-값 페어 `"s3:x-amz-acl":["public-read"]`가 제공됩니다. 키-값 페어에서 `s3:x-amz-acl`은 Amazon S3 전용 키로서, 접두사 `s3:`로 표시됩니다.

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

```
                "AWS": ["arn:aws:iam::account1-ID:root","arn:aws:iam::account2-ID:root"]
            },
        "Action":["s3:PutObject"],
            "Resource": ["arn:aws:s3:::examplebucket/*"],
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl":["public-read"]
                }
            }
        }
    ]
}
```

> **Important**
>
> 모든 조건이 전체 작업에 적용되는 것은 아닙니다. 예를 들어 `s3:CreateBucket` Amazon S3 권한을 부여하는 정책에는 `s3:LocationConstraint` 조건을 포함하는 것이 가능합니다. 그러나 `s3:GetObject` 권한을 부여하는 정책에 이 조건을 포함시키는 것은 의미가 없습니다. Amazon S3는 Amazon S3– 전용 조건을 포함하는 이 유형의 의미론적 오류를 테스트할 수 있습니다. 그러나, IAM 사용자에 대해 정책을 만드는 경우 의미 체계상으로 잘못된 Amazon S3 조건을 포함한다 해도, IAM에서 Amazon S3 조건을 검증할 수 없으므로 오류는 보고되지 않습니다.

객체 작업에 대한 예제 — Amazon S3 조건 키

이 단원에서는 객체 작업에 Amazon S3- 전용 조건 키를 사용할 수 있는 방법을 보여 주는 예제를 제공합니다. 정책에서 지정할 수 있는 Amazon S3 작업, 조건 키 및 리소스의 전체 목록은 Amazon S3에 사용되는 작업, 리소스 및 조건 키 (p. 356) 단원을 참조하십시오.

몇 가지 예제 정책은 PUT Object 작업에서 조건 키를 사용할 수 있는 방법을 보여 줍니다. PUT Object 작업은 ACL(액세스 제어 목록) 기반 권한을 부여하는 데 사용할 수 있는 ACL 전용 헤더를 허용합니다. 이러한 키를 사용하면 버킷 소유자는 사용자가 객체를 업로드할 때 특정 액세스 권한을 요구하도록 조건을 설정할 수 있습니다. 또한 PutObjectAcl 작업을 사용하여 ACL 기반 권한을 부여할 수 있습니다. 자세한 내용은 Amazon S3 Amazon Simple Storage Service API Reference의 PutObjectAcl을 참조하십시오. ACL에 대한 자세한 내용은 ACL(액세스 통제 목록) 개요 (p. 438) 단원을 참조하십시오.

예제 1: 버킷 소유자가 전체 권한을 가져오도록 요구하는 조건으로 s3:PutObject 권한 부여

PUT Object 작업은 ACL 기반 권한을 부여하는 데 사용할 수 있는 ACL(액세스 제어 목록) 전용 헤더를 허용합니다. 이러한 키를 사용하면 버킷 소유자는 사용자가 객체를 업로드할 때 특정 액세스 권한을 요구하도록 조건을 설정할 수 있습니다.

Account A가 버킷을 소유하고 있고 계정 관리자는 Dave라는 Account B의 사용자에게 객체 업로드 권한을 부여하려 한다고 가정합니다. 기본적으로 Dave가 업로드하는 객체는 Account B의 소유이며 Account A에는 이러한 객체에 대한 권한이 없습니다. 버킷 소유자는 비용을 지불하고 있으므로 Dave가 업로드한 객체에 대한 전체 권한을 갖길 원합니다. Account A 관리자는 요청에 명시적으로 전체 권한을 부여하거나 미리 준비된 ACL을 사용하는 ACL 전용 헤더를 포함하는 조건으로 `s3:PutObject` 권한을 Dave에게 부여함으로써 이 작업을 수행할 수 있습니다. 자세한 내용은 PUT Object 단원을 참조하십시오.

`x-amz-full-control` 헤더 필요

버킷 소유자에 대한 전체 제어 권한으로 요청에 `x-amz-full-control` 헤더를 포함하도록 요구할 수 있습니다. 다음 버킷 정책은 `s3:PutObject` 조건 키를 사용하는 조건으로 사용자 Dave에게 `s3:x-amz-grant-full-control` 권한을 부여합니다. 이 조건 키는 요청에 `x-amz-full-control` 헤더를 포함하도록 요구합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "statement1",
```

```
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::AccountB-ID:user/Dave"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::examplebucket/*",
      "Condition": {
        "StringEquals": {
          "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"
        }
      }
    }
  ]
}
```

Note

이 예제는 계정 간 권한에 관한 것입니다. 그러나 Dave(권한을 가져오는 사람)가 버킷을 소유한 AWS 계정에 속한 경우 이러한 조건부 권한은 필요하지 않습니다. Dave가 속한 상위 계정이 사용자가 업로드한 객체를 소유하고 있기 때문입니다.

명시적 거부 추가

앞서 다뤄본 버킷 정책에서는 조건부 권한을 Account B의 사용자 Dave에게 부여했습니다. 반면 이 정책이 적용되면 Dave는 다른 정책을 통한 어떠한 조건 없이도 같은 권한을 가질 수 있습니다. 예를 들어, Dave는 특정 그룹에 속할 수 있으며 이 그룹에 아무런 조건 없이 `s3:PutObject` 권한을 부여합니다. 그러한 권한 순환 고리를 끊기 위해 명시적 거부를 추가하여 더 강력한 액세스 정책을 작성할 수 있습니다. 이 예제에서는 버킷 소유자에게 전체 권한을 부여하는 요청에 필수 헤더를 포함하지 않은 경우 사용자 Dave의 업로드 권한을 명시적으로 거부합니다. 명시적 거부는 이미 부여된 다른 모든 권한에 항상 우선합니다. 다음은 명시적 거부가 추가된 개정된 액세스 정책 예제입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::examplebucket/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"
                }
            }
        },
        {
            "Sid": "statement2",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::examplebucket/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"
                }
            }
        }
    ]
```

```
}
```

AWS CLI를 사용한 정책 테스트

두 개의 AWS 계정이 있는 경우 AWS Command Line Interface(AWS CLI)를 사용하여 정책을 테스트할 수 있습니다. 정책을 연결하고 Dave의 자격 증명을 사용하여 다음 AWS CLI `put-object` 명령을 통해 권한을 테스트합니다. `--profile` 파라미터를 추가하여 Dave의 자격 증명을 제공합니다. `--grant-full-control` 파라미터를 추가하여 버킷 소유자에게 전체 제어 권한을 부여합니다. AWS CLI 설정 및 사용에 대한 자세한 내용은 예제 안내를 위한 도구 설정 (p. 315) 단원을 참조하십시오.

```
aws s3api put-object --bucket examplebucket --key HappyFace.jpg --body c:\HappyFace.jpg --
grant-full-control id="AccountA-CanonicalUserID" --profile AccountBUserProfile
```

`x-amz-acl` 헤더 필요

버킷 소유자에 대한 전체 제어 권한을 부여하는 미리 준비된 ACL 사용 `x-amz-acl` 헤더를 포함하도록 요구할 수 있습니다. 요청에 `x-amz-acl` 헤더를 포함하도록 요구하려면 다음 예제와 같이 `Condition` 블록의 키-값 페어를 대체하고 `s3:x-amz-acl` 조건 키를 지정할 수 있습니다.

```
"Condition": {
 "StringNotEquals": {
  "s3:x-amz-acl": "bucket-owner-full-control"
 }
```

AWS CLI를 사용하는 권한을 테스트하려면 `--acl` 파라미터를 지정합니다. 그런 다음 AWS CLI에서는 요청을 보낼 때 `x-amz-acl` 헤더를 추가합니다.

```
aws s3api put-object --bucket examplebucket --key HappyFace.jpg --body c:\HappyFace.jpg --
acl "bucket-owner-full-control" --profile AccountBadmin
```

예제 2: 서버 측 암호화를 사용하여 저장된 객체를 요구하는 s3:PutObject 권한 부여

Account A에서 버킷을 소유하고 있다고 가정합니다. 계정 관리자는 Account A의 사용자인 Jane이 Amazon S3에서 암호화된 객체를 저장할 수 있도록 항상 서버 측 암호화를 요청한다는 조건으로 그녀에게 객체 업로드 권한을 부여하려 한다고 가정합니다. Account A 관리자는 다음과 같이 `s3:x-amz-server-side-encryption` 조건 키를 사용하여 이 작업을 수행할 수 있습니다. `Condition` 블록의 키-값 페어는 `s3:x-amz-server-side-encryption` 키를 지정합니다.

```
"Condition": {
     "StringNotEquals": {
         "s3:x-amz-server-side-encryption": "AES256"
     }
```

AWS CLI를 사용하여 권한을 테스트하는 경우 `--server-side-encryption` 파라미터를 사용하여 필요한 파라미터를 추가해야 합니다.

```
aws s3api put-object --bucket example1bucket --key HappyFace.jpg --body c:\HappyFace.jpg --
server-side-encryption "AES256" --profile AccountBadmin
```

예제 3: 복사 원본에 대한 제한이 있는 s3:PutObject 객체 복사 권한을 부여

PUT Object 요청에서 원본 객체를 지정하는 경우, 이는 복사 작업에 해당합니다(PUT Object - Copy 참조). 따라서 버킷 소유자는 원본에 대한 제한이 있는 객체 복사 권한을 사용자에게 부여할 수 있습니다. 예를 들면 다음과 같습니다.

- `sourcebucket` 버킷에서만 객체를 복사하도록 허용합니다.

- sourcebucket 버킷에서 객체를 복사하도록 허용하며 키 이름의 접두사가 public/ f로 시작하는 객체만 복사하도록 허용합니다(예: sourcebucket/public/*).
- sourcebucket에서 특정 객체만 복사하도록 허용합니다(예: sourcebucket/example.jpg).

다음 버킷 정책은 사용자(Dave)에게 s3:PutObject 권한을 부여합니다. 이는 요청에 s3:x-amz-copy-source 헤더를 포함하고 헤더 값으로 /examplebucket/public/* 키 이름 접두사를 지정하는 조건을 충족하는 객체만 복사하도록 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
       {
            "Sid": "cross-account permission to user in your own account",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
            },
            "Action": ["s3:PutObject"],
            "Resource": "arn:aws:s3:::examplebucket/*"
        },
        {
            "Sid": "Deny your user permission to upload object if copy source is not /
bucket/folder",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::examplebucket/*",
            "Condition": {
                "StringNotLike": {
                    "s3:x-amz-copy-source": "examplebucket/public/*"
                }
            }
        }
    ]
}
```

AWS CLI를 사용한 정책 테스트

AWS CLI copy-object 명령을 사용하는 권한을 테스트할 수 있습니다. --copy-source 파라미터를 추가하여 원본을 지정할 수 있습니다. 이때 키 이름은 정책에 허용된 접두사와 일치해야 합니다. --profile 파라미터를 사용하는 사용자 Dave의 자격 증명을 제공해야 합니다. AWS CLI 설정에 대한 자세한 내용은 예제 안내를 위한 도구 설정 (p. 315) 단원을 참조하십시오.

```
aws s3api copy-object --bucket examplebucket --key HappyFace.jpg
--copy-source examplebucket/public/PublicHappyFace1.jpg --profile AccountADave
```

특정 객체만 복사할 수 있는 권한 부여

앞서 다룬 정책에서는 StringNotLike 조건을 사용합니다. 특정 객체만 복사할 수 있는 권한을 부여하려면 StringNotLike에서 StringNotEquals로 조건을 변경한 뒤 다음과 같이 정확한 객체 키를 지정해야 합니다.

```
"Condition": {
       "StringNotEquals": {
           "s3:x-amz-copy-source": "examplebucket/public/PublicHappyFace1.jpg"
       }
}
```

예제 4: 특정 버전의 객체에 대한 액세스 권한을 부여

Account A에서 버전 관리를 사용하는 버킷을 소유하고 있다고 가정합니다. 버킷에는 몇 가지 버전의 `HappyFace.jpg` 객체가 있습니다. 계정 관리자는 현재 해당 계정의 사용자 Dave에게 특정 버전의 객체만 가져올 수 있는 권한을 부여하려 합니다. 계정 관리자는 다음과 같이 조건부로 Dave에게 `s3:GetObjectVersion` 권한을 부여함으로써 이 작업을 수행할 수 있습니다. `Condition` 블록의 키-값 페어는 `s3:VersionId` 조건 키를 지정합니다. 이 경우 Dave는 가져오려는 객체의 정확한 객체 버전 ID를 알고 있어야 합니다.

자세한 내용은 Amazon Simple Storage Service API Reference의 GetObject를 참조하십시오.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
            },
            "Action": ["s3:GetObjectVersion"],
            "Resource": "arn:aws:s3:::examplebucketversionenabled/HappyFace.jpg"
        },
        {
            "Sid": "statement2",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
            },
            "Action": ["s3:GetObjectVersion"],
            "Resource": "arn:aws:s3:::examplebucketversionenabled/HappyFace.jpg",
            "Condition": {
                "StringNotEquals": {
                    "s3:VersionId": "AaaHbAQitwiL_h47_44lRO2DDfLlBO5e"
                }
            }
        }
    ]
}
```

AWS CLI를 사용한 정책 테스트

특정 객체 버전을 식별하는 `--version-id` 파라미터를 적용한 AWS CLI `get-object` 명령을 사용하여 권한을 테스트할 수 있습니다. 이 명령은 객체를 가져와서 이를 `OutputFile.jpg` 파일로 저장합니다.

```
aws s3api get-object --bucket examplebucketversionenabled --key HappyFace.jpg
 OutputFile.jpg --version-id AaaHbAQitwiL_h47_44lRO2DDfLlBO5e --profile AccountADave
```

예 5: 특정 스토리지 클래스를 이용한 객체 업로드 제한

Account A에서 버킷을 소유하고 있다고 가정합니다. Account A가 버킷을 소유하고 있고, 계정 관리자는 Account A에 속한 사용자 Dave만 `STANDARD_IA` 스토리지 클래스를 이용해 저장할 버킷에 객체를 업로드할 수 있도록 허용하려고 합니다. 객체 업로드를 특정 스토리지 클래스로 제한하기 위해 Account A 관리자는 다음 버킷 정책 예와 같이 `s3:x-amz-storage-class` 조건 키를 사용할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "statement1",
      "Effect": "Allow",
      "Principal": {
```

```
        "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
      },
      "Action": "s3:PutObject",
      "Resource": [
        "arn:aws:s3:::examplebucket/*"
      ],
      "Condition": {
        "StringEquals": {
          "s3:x-amz-storage-class": [
            "STANDARD_IA"
          ]
        }
      }
    }
  ]
}
```

예제 6: 객체 태그 기반 권한 부여

Amazon S3 작업에 객체 태그 지정 조건 키를 사용하는 방법에 대한 예는 객체 태그 지정 및 액세스 제어 정책 (p. 117) 단원을 참조하십시오.

버킷 작업에 대한 예제 — Amazon S3 조건 키

이 단원에서는 버킷 작업에 Amazon S3- 전용 조건 키를 사용하는 방법을 보여 주는 정책 예제를 제공합니다.

예제 1: 사용자에게 특정 리전에서만 버킷을 만들 수 있는 권한 부여

AWS 계정 관리자가 해당 계정 사용자(Dave)에게 남아메리카(상파울루) 리전에서만 버킷을 생성하는 권한을 부여하려고 한다고 가정합니다. 계정 관리자는 다음과 같은 조건으로 `s3:CreateBucket` 권한을 부여하는 다음 사용자 정책을 연결할 수 있습니다. `Condition` 블록의 키-값 페어는 `s3:LocationConstraint` 조건 키와 `sa-east-1` 리전을 해당 값으로 지정합니다.

> **Note**
>
> 이 예제에서 버킷 소유자는 해당 사용자 중 하나에게 권한을 부여하려 하므로 버킷 정책 또는 사용자 정책 중 하나를 사용할 수 있습니다. 이 예제는 사용자 정책을 나타냅니다.

Amazon S3 리전의 목록은 AWS General Reference의 리전 및 엔드포인트를 참조하십시오.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action":[
            "s3:CreateBucket"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition": {
             "StringLike": {
                 "s3:LocationConstraint": "sa-east-1"
             }
         }
       }
    ]
}
```

명시적 거부 추가

앞의 정책은 sa-east-1을 제외한 다른 리전에서 버킷을 생성하지 못하도록 사용자를 제한합니다. 그러나 일부 다른 정책에서는 이 사용자에게 다른 리전에서 버킷을 생성할 수 있는 권한을 부여할 수 있습니다. 예를 들어 사용자가 그룹에 속한 경우 그룹 내 전체 사용자에게 다른 리전에서 버킷을 생성할 수 있는 권한을 허용하는 정책이 이 그룹에 연결되어 있을 수 있습니다. 사용자가 다른 리전에서 버킷을 생성할 수 있는 권한을 갖지 못하게 하려면 위 정책에 명시적 거부 문을 추가할 수 있습니다.

Deny 문은 StringNotLike 조건을 사용합니다. 즉, 버킷 생성 요청은 위치 제한이 sa-east-1이 아닌 경우 거부됩니다. 명시적 거부는 사용자가 다른 어떤 권한을 갖고 있더라도 사용자가 다른 리전에서 버킷을 생성하는 것을 허용하지 않습니다. 아래 정책에는 명시적 거부 문이 포함되어 있습니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action":[
            "s3:CreateBucket"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition": {
             "StringLike": {
                 "s3:LocationConstraint": "sa-east-1"
             }
         }
       },
      {
         "Sid":"statement2",
         "Effect":"Deny",
         "Action":[
            "s3:CreateBucket"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition": {
             "StringNotLike": {
                 "s3:LocationConstraint": "sa-east-1"
             }
         }
       }
    ]
}
```

AWS CLI를 사용한 정책 테스트

다음 create-bucket AWS CLI 명령을 사용하는 정책을 테스트할 수 있습니다. 이 예제에서는 bucketconfig.txt 파일을 사용하여 위치 제한을 지정합니다. Windows 파일 경로를 확인하여 버킷 이름과 경로를 적절히 업데이트해야 합니다. --profile 파라미터를 사용하여 사용자 자격 증명을 제공해야 합니다. AWS CLI 설정 및 사용에 대한 자세한 내용은 예제 안내를 위한 도구 설정 (p. 315) 단원을 참조하십시오.

```
aws s3api create-bucket --bucket examplebucket --profile AccountADave --create-bucket-
configuration file://c:/Users/someUser/bucketconfig.txt
```

bucketconfig.txt 파일은 다음과 같은 구성을 지정합니다.

```
{"LocationConstraint": "sa-east-1"}
```

예제 2: 특정 접두사가 있는 버킷의 객체 목록 가져오기

`s3:prefix` 조건 키를 사용하여 GET 버킷(ListObjects) API의 응답을 특정 접두사가 있는 키 이름으로 제한할 수 있습니다. 버킷 소유자는 버킷에 있는 특정 접두사의 콘텐츠를 나열하도록 사용자를 제한할 수 있습니다. 이 조건 키는 버킷의 객체가 키 이름 접두사로 정리되어 있는 경우 유용합니다. Amazon S3 콘솔은 키 이름 접두사를 사용하여 폴더 개념을 표시합니다. 콘솔만 폴더 개념을 지원하고 Amazon S3 API는 버킷과 객체만 지원합니다. 접두사 및 구분 기호를 사용하여 액세스 권한을 필터링하는 방법에 대한 자세한 내용은 연습: 사용자 정책을 사용하여 버킷에 대한 액세스 제어 (p. 421) 단원을 참조하십시오.

예를 들어 키 이름이 `public/object1.jpg` 및 `public/object2.jpg`인 두 개의 객체가 있는 경우, 콘솔에서는 `public` 폴더 아래에 객체가 표시됩니다. Amazon S3 API에서 이러한 객체는 폴더의 객체가 아니라 접두사가 있는 객체입니다. 그러나 Amazon S3 API에서 그러한 접두사를 사용하여 객체 키를 구성하는 경우, 사용자가 특정 접두사가 있는 키 이름 목록을 가져오도록 허용하는 `s3:prefix` 조건으로 `s3:ListBucket` 권한을 부여할 수 있습니다.

이 예제에서, 버킷 소유자와 사용자가 속한 상위 계정은 동일합니다. 따라서 버킷 소유자는 버킷 정책 또는 사용자 정책 중 하나를 사용할 수 있습니다. GET 버킷(ListObjects) API와 함께 사용할 수 있는 다른 조건 키에 대한 자세한 내용은 ListObjects를 참조하십시오.

사용자 정책

다음 사용자 정책은 사용자가 요청에서 `prefix`의 값을 `projects`로 지정하도록 요구하는 조건으로 `s3:ListBucket` 권한을 부여합니다(GET Bucket (List Objects) 참조).

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::examplebucket"
         ],
         "Condition" : {
             "StringEquals" : {
                 "s3:prefix": "projects"
             }
          }
       },
      {
         "Sid":"statement2",
         "Effect":"Deny",
         "Action":[
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::examplebucket"
         ],
         "Condition" : {
             "StringNotEquals" : {
                 "s3:prefix": "projects"
             }
          }
       }
    ]
}
```

이 조건은 사용자가 `projects` 접두사가 있는 객체 키만 수신하도록 제한합니다. 추가된 명시적 거부는 사용자가 보유할 수 있는 다른 모든 권한에도 불구하고 다른 접두사가 있는 키의 목록을 가져오려는 사용자 요

청을 거부합니다. 사용자가 앞서 다룬 사용자 정책으로 업데이트하거나 버킷 정책을 통해 아무런 제한 없이 객체 키 목록을 가져올 권한을 갖게 될 수 있습니다. 명시적 거부가 항상 우선하므로 `project` 접두사가 아닌 키의 목록을 가져오려는 사용자 요청은 거부됩니다.

버킷 정책

사용자를 식별하는 `Principal` 요소를 위의 사용자 정책에 추가하는 경우, 다음과 같이 버킷 정책을 갖게 됩니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Principal": {
            "AWS": "arn:aws:iam::BucketOwner-accountID:user/user-name"
         },
         "Action":[
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::examplebucket"
         ],
         "Condition" : {
             "StringEquals" : {
                 "s3:prefix": "examplefolder"
             }
          }
       },
      {
         "Sid":"statement2",
         "Effect":"Deny",
         "Principal": {
            "AWS": "arn:aws:iam::BucketOwner-AccountID:user/user-name"
         },
         "Action":[
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::examplebucket"
         ],
         "Condition" : {
             "StringNotEquals" : {
                 "s3:prefix": "examplefolder"
             }
          }
       }
    ]
}
```

AWS CLI를 사용한 정책 테스트

다음 `list-object` AWS CLI 명령을 사용하는 정책을 테스트할 수 있습니다. 이 명령에서 `--profile` 파라미터를 사용하여 사용자 자격 증명을 제공합니다. AWS CLI 설정 및 사용에 대한 자세한 내용은 예제 안내를 위한 도구 설정 (p. 315) 단원을 참조하십시오.

```
aws s3api list-objects --bucket examplebucket --prefix examplefolder --profile AccountADave
```

버킷에서 버전 관리를 사용하는 경우 버킷의 객체 목록을 가져오려면 `s3:ListBucket` 권한 대신 앞서 다룬 정책에 `s3:ListBucketVersions` 권한을 부여해야 합니다. 또한 이 권한은 `s3:prefix` 조건 키를 지원합니다.

예제 3: 최대 키 수 설정

`s3:max-keys` 조건 키를 사용하여 요청자가 GET 버킷(ListObjects) 또는 ListObjectVersions 요청에서 반환할 수 있는 최대 키 수를 설정할 수 있습니다. 기본적으로 API에서는 최대 1,000개의 키를 반환합니다. `s3:max-keys`와 함께 사용할 수 있는 숫자 조건 연산자 리스트는 IAM 사용 설명서의 숫자 조건 연산자를 참조하십시오.

Amazon S3에 사용되는 작업, 리소스 및 조건 키

Amazon S3(서비스 접두사: `s3`)은 IAM 권한 정책에 사용할 수 있는 다음과 같은 서비스별 리소스, 작업 및 조건 컨텍스트 키를 제공합니다.

참조:

- 이 서비스를 구성하는 방법을 알아봅니다.
- 이 서비스에 사용 가능한 API 작업의 목록을 봅니다.
- IAM 권한 정책을 사용하여 이 서비스와 리소스를 보호하는 방법을 알아봅니다.

> **Note**
>
> 이 섹션을 업데이트하는 중입니다. 전체 테이블을 보려면 이번 달 후반에 다시 확인하십시오.

주제

- Amazon S3에서 정의한 작업 (p. 356)
- Amazon S3에서 정의한 리소스 유형 (p. 405)
- Amazon S3에 사용되는 조건 키 (p. 405)

Amazon S3에서 정의한 작업

IAM 정책 설명의 `Action` 요소에서 다음 작업을 지정할 수 있습니다. 정책을 사용하여 AWS에서 작업할 수 있는 권한을 부여합니다. 정책에서 작업을 사용하면 일반적으로 이름이 같은 API 작업 또는 CLI 명령에 대한 액세스를 허용하거나 거부합니다. 그러나 경우에 따라 하나의 작업으로 둘 이상의 작업에 대한 액세스가 제어됩니다. 또는 일부 작업을 수행하려면 다양한 작업이 필요합니다.

리소스 유형 열에는 각 작업이 리소스 수준 권한을 지원하는지 여부가 표시됩니다. 리소스 열에 값이 없으면 정책 문의 `Resource` 요소에서 모든 리소스("*")를 지정해야 합니다. 리소스 열에 리소스 유형이 포함되어 있으면 해당 작업 시 문에서 해당 유형의 ARN을 지정할 수 있습니다. 필수 리소스는 테이블에서 별표(*)로 표시됩니다. 이 작업을 사용해 문에서 리소스 레벨 권한 ARN을 지정할 경우 이 유형이 되어야 합니다. 일부 작업은 다수의 리소스 유형을 지원합니다. 리소스 유형이 옵션(필수 리소스로 표시되지 않은 경우)인 경우에는 한 가지를 선택할 수 있지만 나머지는 선택하지 못합니다.

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
AbortMultipartUpload	멀티파트 업로드를 중단합니다.	쓰기	object* (p. 405)		
				s3:DataAccessPointArn (p. 406) s3:DataAccessPointAccount (p. 406) s3:AccessPointNetworkOrigin (p. 406)	

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
BypassGovernanceRetention	거버넌스 모드 객체 보존 설정의 우회를 허용합니다.	권한 관리	object* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:RequestObjectTag/<key> (p. 406) s3:RequestObjectTagKeys (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-acl (p. 407) s3:x-amz-content-sha256 (p. 407) s3:x-amz-copy-source (p. 407) s3:x-amz-grant-full-control (p. 407) s3:x-amz-grant-read (p. 407) s3:x-amz-grant-read-acp (p. 407) s3:x-amz-grant-write (p. 407)	

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:x-amz-grant-write-acp (p. 407) s3:x-amz-metadata-directive (p. 407) s3:x-amz-server-side-encryption (p. 407) s3:x-amz-server-side-encryption-aws-kms-key-id (p. 407) s3:x-amz-storage-class (p. 407) s3:x-amz-website-redirect-location (p. 407) s3:object-lock-mode (p. 406) s3:object-lock-retain-until-date (p. 407) s3:object-lock-remaining-retention-days (p. 407) s3:object-lock-legal-hold (p. 406)	

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
CreateAccessPoint	새 액세스 포인트를 생성합니다.	쓰기	accesspoint* (p. 405)		
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:authtype (p. 406) s3:locationconstraint (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-acl (p. 407) s3:x-amz-content-sha256 (p. 407)	
CreateBucket	새 버킷을 만듭니다.	쓰기	bucket* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:authtype (p. 406) s3:locationconstraint (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-acl (p. 407) s3:x-amz-content-sha256 (p. 407) s3:x-amz-grant-full-control (p. 407) s3:x-amz-grant-read (p. 407) s3:x-amz-grant-read-acp (p. 407) s3:x-amz-grant-write (p. 407) s3:x-amz-grant-write-acp (p. 407)	

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
CreateJob	새 Amazon S3 Batch Operations 작업을 생성합니다.	쓰기		s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407) s3:RequestJobPriority (p. 406) s3:RequestJobOperation (p. 406)	
DeleteAccessPoint	URI에 명명된 액세스 포인트를 삭제합니다.	쓰기	accesspoint* (p. 405)		
				s3:DataAccessPointArn (p. 406) s3:DataAccessPointAccount (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
DeleteAccessPointPolicy	지정된 액세스 포인트에서 정책을 삭제합니다.	권한 관리	accesspoint* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:DataAccessPointArn (p. 406) s3:DataAccessPointAccount (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
DeleteBucket	URI에 이름을 올린 버킷을 삭제합니다.	쓰기	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
DeleteBucketPolicy	지정된 버킷에서 정책을 삭제합니다.	권한 관리	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
DeleteBucketWebsite	버킷에 대한 웹 사이트 구성을 제거합니다.	쓰기	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
DeleteObject	객체의 널(null) 버전(있는 경우)을 제거하고 삭제 마커를 삽입합니다(그러면 삭제 마커가 객체의 현재 버전이 됩니다).	쓰기	object* (p. 405)		
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
DeleteObjectTagging	DELETE 작업의 이 구현은 태깅 하위 자원을 사용하여 지정된 객체에서 전체 태그 집합을 제거합니다.	태그 지정	object* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:ExistingObjectTag/<key> (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
DeleteObjectVersion	객체의 지정된 버전을 제거하려면 버킷 소유자여야 하고 versionId 하위 자원을 사용해야 합니다.	쓰기	object* (p. 405)		
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:versionid (p. 407) s3:x-amz-content-sha256 (p. 407)	

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
DeleteObjectVersionTagging	DELETE Object 태그 지정(특정 버전의 객체용)	태그 지정	object* (p. 405)		
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:ExistingObjectTag/<key> (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:versionid (p. 407) s3:x-amz-content-sha256 (p. 407)	
DescribeJob	Amazon S3 Batch Operations 작업에 대한 구성 파라미터 및 상태를 검색합니다.	Read	job* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetAccelerateConfiguration	GET 작업의 이 구현은 가속화 하위 자원을 사용하여 버킷의 Transfer Acceleration 상태(활성 또는 일시 중지됨)를 반환합니다.	Read	bucket* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetAccessPoint	액세스 포인트 메타데이터를 검색합니다.	Read		s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetAccessPointPolicy	지정된 액세스 포인트의 정책을 반환합니다.	Read	accesspoint* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetAccessPointPolicyStatus	특정 액세스 포인트의 정책에 대한 정책 상태를 검색합니다.	Read	accesspoint* (p. 405)		
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
GetAccountPublicAccessBlock	AWS 계정에 대한 PublicAccessBlock 구성을 검색합니다.	Read		s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetAnalyticsConfiguration	GET 작업의 이 구현은 버킷에서 분석 구성(분석 구성 ID로 식별됨)을 반환합니다.	Read	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetBucketAcl	버킷의 ACL(액세스 제어 목록)을 반환합니다.	Read	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetBucketCORS	버킷에 대한 CORS 정보 집합을 반환합니다.	Read	bucket* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetBucketLocation	버킷의 리전을 반환합니다.	Read	bucket* (p. 405)		
GetBucketLogging	사용자가 해당 상태를 보고 수정해야 하는 버킷 및 권한의 로깅 상태를 반환합니다.	Read	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetBucketNotification	버킷의 알림 구성을 반환합니다.	Read	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetBucketObjectLockConfiguration	특정 버킷에 대한 객체 잠금 구성을 가져옵니다.	Read	bucket* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407)	
GetBucketPolicy	지정된 버킷의 정책을 반환합니다.	Read	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetBucketPolicyStatus	특정 S3 버킷에 대해 해당 버킷이 퍼블릭임을 나타내는 정책 상태를 검색합니다.	Read	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetBucketPublicAccessBlock	특정 S3 버킷에 대한 PublicAccessBlock 구성을 검색합니다.	Read	bucket* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetBucketRequestPayment	버킷의 결제 요청 구성을 반환합니다.	Read	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetBucketTagging	버킷과 연결된 태그 집합을 반환합니다.	Read	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetBucketVersioning	버킷의 버전 관리 상태를 반환합니다.	Read	bucket* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetBucketWebsite	버킷과 연결된 웹 사이트 구성을 반환합니다.	Read	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetEncryptionConfiguration	버킷에 설정된 암호화 구성 정보를 반환합니다.	Read	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetInventoryConfiguration	GET 작업의 이 구현은 버킷에서 인벤토리 구성(인벤토리 구성 ID로 식별됨)을 반환합니다.	Read	bucket* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetLifecycleConfiguration	버킷에 설정된 수명 주기 구성 정보를 반환합니다.	Read	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetMetricsConfiguration	버킷의 CloudWatch 요청 지표(지표 구성 ID로 지정)에 대한 지표 구성을 가져옵니다. 단, 여기에 일간 스토리지 지표는 포함되지 않습니다.	Read	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetObject	Amazon S3에서 객체를 검색합니다.	Read	object* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:ExistingObjectTag/<key> (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetObjectAcl	객체의 ACL(액세스 제어 목록)을 반환합니다.	Read	object* (p. 405)		
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:ExistingObjectTag/<key> (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
GetObjectLegalHold	특정 객체에 대한 객체 법적 보존을 가져옵니다.	Read	object* (p. 405)		
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetObjectRetention	특정 객체에 대한 객체 법적 보존을 가져옵니다.	Read	object* (p. 405)		
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
GetObjectTagging	GET 작업의 이 구현은 객체와 연결된 태그를 반환합니다. 그러면 객체와 연결된 태깅 하위 리소스에 대한 GET 요청을 보낼 수 있습니다.	Read	object* (p. 405)		
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:ExistingObjectTag/<key> (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetObjectTorrent	버킷에서 토렌트 파일을 반환합니다.	Read	object* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetObjectVersion	여러 다른 버전을 반환하려면 versionId 하위 리소스를 사용합니다.	Read	object* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:ExistingObjectTag/<key> (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:versionid (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetObjectVersionAcl	여러 다른 버전에 대한 ACL 정보를 반환하려면 versionId 하위 리소스를 사용합니다.	Read	object* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:ExistingObjectTag/<key> (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:versionid (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetObjectVersionForReplication	S3 복제에서 수행된 권한	Read	object* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetObjectVersionTagging	GET Object 태그 지정(특정 버전의 객체에 적용)	Read	object* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:ExistingObjectTag/<key> (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:versionid (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetObjectVersionTorrent	여러 다른 버전에 대한 Torrent 파일을 반환하려면 versionId 하위 리소스를 사용합니다.	Read	object* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:versionid (p. 407) s3:x-amz-content-sha256 (p. 407)	
GetReplicationConfiguration	버킷에 설정된 복제 구성 정보를 반환합니다.	Read	bucket* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
ListAccessPoints	액세스 포인트를 나열합니다.	Read		s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
ListAllMyBuckets	요청의 인증된 발신자가 소유한 모든 버킷의 목록을 반환합니다.	List		s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
ListBucket	버킷의 객체를 일부 또는 전부(최대 1,000개) 반환합니다.	List	bucket* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:authtype (p. 406) s3:delimiter (p. 406) s3:max-keys (p. 406) s3:prefix (p. 407) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
ListBucketMultipartUploads	진행 중인 멀티파트 업로드를 나열합니다.	Read	bucket* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
ListBucketVersions	버전 하위 리소스를 사용하여 버킷에 있는 객체의 모든 버전에 대한 메타데이터를 나열합니다.	Read	bucket* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:authtype (p. 406) s3:delimiter (p. 406) s3:max-keys (p. 406) s3:prefix (p. 407) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
ListJobs	현재 작업 및 최근 종료된 작업을 나열합니다.	Read		s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
ListMultipartUploadParts	특정 멀티파트 업로드에 대해 업로드된 파트를 나열합니다.	Read	object* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
ObjectOwnerOverrideToBucketOwner	S3 복제에서 수행된 권한	권한 관리	object* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
PutAccelerateConfiguration	PUT 작업의 이 구현은 가속화 하위 자원을 사용하여 기존 버킷의 Transfer Acceleration 상태를 설정합니다.	쓰기	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
PutAccessPointPolicy	액세스 포인트에 대한 데이터 정책을 추가하거나 바꿉니다.	권한 관리	accesspoint* (p. 405)		
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
PutAccountPublicAccessBlock	AWS 계정에 대한 PublicAccessBlock 구성을 생성 또는 수정합니다.	권한 관리		s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
PutAnalyticsConfiguration	PUT 작업의 이 구현은 버킷에 분석 구성(분석 ID로 식별됨)을 추가합니다. 버킷당 최대 1,000개의 분석 구성을 설정할 수 있습니다.	쓰기	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
PutBucketAcl	ACL(액세스 제어 목록)을 사용하여 기존 버킷에 대한 권한을 설정합니다.	권한 관리	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-acl (p. 407) s3:x-amz-content-sha256 (p. 407) s3:x-amz-grant-full-control (p. 407) s3:x-amz-grant-read (p. 407) s3:x-amz-grant-read-acp (p. 407) s3:x-amz-grant-write (p. 407) s3:x-amz-grant-write-acp (p. 407)	
PutBucketCORS	버킷에 대한 CORS 구성을 설정합니다.	쓰기	bucket* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
PutBucketLogging	버킷에 대한 로깅 파라미터를 설정합니다.	쓰기	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
PutBucketNotification	버킷에서 특정 이벤트가 발생할 때 알림을 받을 수 있습니다.	쓰기	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
PutBucketObjectLockConfiguration	특정 버킷에 대한 객체 잠금 구성을 배포합니다.	쓰기	bucket* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407)	
PutBucketPolicy	버킷에 대한 정책을 추가하거나 대체합니다.	권한 관리	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
PutBucketPublicAccessBlock	특정 S3 버킷에 대한 PublicAccessBlock 구성을 생성 또는 수정합니다.	권한 관리	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
PutBucketRequestPayment	버킷의 결제 요청 구성을 설정합니다.	쓰기	bucket* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
PutBucketTagging	기존 버킷에 태그 집합을 추가합니다.	태그 지정	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
PutBucketVersioning	기존 버킷의 버전 관리 상태를 설정합니다.	쓰기	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
PutBucketWebsite	웹 사이트 하위 리소스에 지정된 웹사이트의 구성을 설정합니다.	쓰기	bucket* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
PutEncryptionConfiguration	버킷에 대한 암호화 구성을 설정합니다.	쓰기	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
PutInventoryConfiguration	PUT 작업의 이 구현은 버킷에 인벤토리 구성(인벤토리 ID로 식별됨)을 추가합니다. 버킷당 최대 1,000개의 인벤토리 구성을 설정할 수 있습니다.	쓰기	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
PutLifecycleConfiguration	버킷에 새로운 수명 주기 구성을 생성하거나 기존 수명 주기 구성을 대체합니다.	쓰기	bucket* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
PutMetricsConfiguration	버킷의 CloudWatch 요청 지표(지표 구성 ID로 지정)에 대한 지표 구성을 설정하거나 업데이트합니다.	쓰기	bucket* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
PutObject	버킷에 객체를 추가합니다.	쓰기	object* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:RequestObjectTag/<key> (p. 406) s3:RequestObjectTagKeys (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-acl (p. 407) s3:x-amz-content-sha256 (p. 407) s3:x-amz-copy-source (p. 407) s3:x-amz-grant-full-control (p. 407) s3:x-amz-grant-read (p. 407) s3:x-amz-grant-read-acp (p. 407) s3:x-amz-grant-write (p. 407)	

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:x-amz-grant-write-acp (p. 407) s3:x-amz-metadata-directive (p. 407) s3:x-amz-server-side-encryption (p. 407) s3:x-amz-server-side-encryption-aws-kms-key-id (p. 407) s3:x-amz-storage-class (p. 407) s3:x-amz-website-redirect-location (p. 407) s3:object-lock-mode (p. 406) s3:object-lock-retain-until-date (p. 407) s3:object-lock-remaining-retention-days (p. 407) s3:object-lock-legal-hold (p. 406)	

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
PutObjectAcl	버킷에 이미 존재하는 객체에 대한 ACL(액세스 제어 목록) 권한을 설정합니다.	권한 관리	object* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:ExistingObjectTag/<key> (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-acl (p. 407) s3:x-amz-content-sha256 (p. 407) s3:x-amz-grant-full-control (p. 407) s3:x-amz-grant-read (p. 407) s3:x-amz-grant-read-acp (p. 407) s3:x-amz-grant-write (p. 407) s3:x-amz-grant-write-acp (p. 407) s3:x-amz-storage-	

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				class (p. 407)	
PutObjectLegalHold	특정 객체에 대한 객체 법적 보존을 내보냅니다.	쓰기	object* (p. 405)		
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407) s3:object-lock-legal-hold (p. 406)	
PutObjectRetention	특정 객체에 대한 객체 보존을 내보냅니다.	쓰기	object* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407) s3:object-lock-mode (p. 406) s3:object-lock-retain-until-date (p. 407) s3:object-lock-remaining-retention-days (p. 407)	
PutObjectTagging	PUT 작업의 이 구현은 태깅 하위 자원을 사용하여 기존 객체에 태그 집합을 추가합니다.	태그 지정	object* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:ExistingObjectTag/<key> (p. 406) s3:RequestObjectTag/<key> (p. 406) s3:RequestObjectTagKeys (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
PutObjectVersionAcl	객체의 ACL은 객체 버전 수준에서 설정됩니다.	권한 관리	object* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:ExistingObjectTag/<key> (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:versionid (p. 407) s3:x-amz-acl (p. 407) s3:x-amz-content-sha256 (p. 407) s3:x-amz-grant-full-control (p. 407) s3:x-amz-grant-read (p. 407) s3:x-amz-grant-read-acp (p. 407) s3:x-amz-grant-write (p. 407) s3:x-amz-grant-write-acp (p. 407)	

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:x-amz-storage-class (p. 407)	
PutObjectVersionTagging	PUT Object 태그 지정(특정 버전의 객체에 적용)	태그 지정	object* (p. 405)		
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:ExistingObjectTag/<key> (p. 406) s3:RequestObjectTag/<key> (p. 406) s3:RequestObjectTagKeys (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:versionid (p. 407) s3:x-amz-content-sha256 (p. 407)	
PutReplicationConfiguration	버전 관리를 사용하는 버킷에서 이 작업을 새로운 복제 구성을 생성합니다(또는 기존 구성(있는 경우)을 대체합니다).	쓰기	bucket* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
ReplicateDelete	S3 복제에서 수행된 권한	쓰기	object* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
ReplicateObject	S3 복제에서 수행된 권한	쓰기	object* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407) s3:x-amz-server-side-encryption (p. 407) s3:x-amz-server-side-encryption-aws-kms-key-id (p. 407)	
ReplicateTags	S3 복제에서 수행된 권한	태그 지정	object* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
RestoreObject	아카이브된 객체의 임시 사본을 복원합니다.	쓰기	object* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:DataAccessPointAccount (p. 406) s3:DataAccessPointArn (p. 406) s3:AccessPointNetworkOrigin (p. 406) s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407)	
UpdateJobPriority	기존 작업의 우선 순위를 업데이트합니다.	쓰기	job* (p. 405)		
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407) s3:RequestJobPriority (p. 406) s3:ExistingJobPriority (p. 406) s3:ExistingJobOperation (p. 406)	
UpdateJobStatus	지정된 작업의 상태를 업데이트합니다.	쓰기	job* (p. 405)		

Actions	설명	액세스 레벨	리소스 유형(*필수)	조건 키	종속 작업
				s3:authtype (p. 406) s3:signatureage (p. 407) s3:signatureversion (p. 407) s3:x-amz-content-sha256 (p. 407) s3:ExistingJobPriority (p. 406) s3:ExistingJobOperation (p. 406) s3:JobSuspendedCause (p. 406)	

Amazon S3에서 정의한 리소스 유형

다음 리소스 유형은 이 서비스에서 정의되며 IAM 권한 정책 설명의 `Resource` 요소에서 사용할 수 있습니다. 작업 테이블 (p. 356)의 각 작업은 해당 작업으로 지정할 수 있는 리소스 유형을 식별합니다. 리소스 유형은 정책에 포함할 수 있는 조건 키를 정의할 수도 있습니다. 이러한 키는 테이블의 마지막 열에 표시됩니다.

리소스 유형	ARN	조건 키
accesspoint	`arn:${Partition}:s3:${Region}:${Account}:accesspoint/${AccessPointName}`	
bucket	`arn:${Partition}:s3:::${BucketName}`	
object	`arn:${Partition}:s3:::${BucketName}/${ObjectName}`	
job	`arn:${Partition}:s3:${Region}:${Account}:job/${JobId}`	

Amazon S3에 사용되는 조건 키

Amazon S3는 IAM 정책의 `Condition` 요소에 사용할 수 있는 다음과 같은 조건 키를 정의합니다. 이러한 키를 사용하여 정책 설명이 적용되는 조건을 추가로 재정의할 수 있습니다.

모든 서비스에 사용할 수 있는 글로벌 조건 키를 보려면 IAM 정책 참조의 사용 가능한 글로벌 조건 키를 참조하십시오.

조건 키	설명	유형
s3:AccessPointNetworkOrigin	요청과 관련된 액세스 포인트에서 수신할 수 있는 트래픽의 네트워크 유형	문자열
s3:DataAccessPointAccount	요청과 관련된 데이터 작업 액세스 포인트를 소유하는 계정의 AWS 계정 ID	문자열
s3:DataAccessPointArn	요청과 관련된 데이터 작업 액세스 포인트의 ARN	문자열
s3:ExistingJobOperation	작업별로 작업 우선 순위 업데이트에 대한 액세스를 필터링합니다.	문자열
s3:ExistingJobPriority	우선 순위 범위별로 기존 작업 취소에 대한 액세스를 필터링합니다.	숫자
s3:ExistingObjectTag/<key>	기존 객체 태그에 특정 태그 키 및 값이 있다는 것을 확인할 수 있습니다.	문자열
s3:JobSuspendedCause	일시 중단된 특정 작업 원인(예: AWAITING_CONFIRMATION)별로 일시 중단된 작업 취소에 대한 액세스를 필터링합니다.	문자열
s3:LocationConstraint	사용자가 특정 리전에서만 버킷을 만들도록 제한할 수 있습니다.	문자열
s3:RequestJobOperation	작업별로 작업 우선 순위 업데이트에 대한 액세스를 필터링합니다.	문자열
s3:RequestJobPriority	우선 순위 범위별로 새 작업 생성에 대한 액세스를 필터링합니다.	숫자
s3:RequestObjectTag/<key>	객체에 대해 허용하고자 하는 태그 키 및 값을 제한합니다.	문자열
s3:RequestObjectTagKeys	객체에 대해 허용하고자 하는 태그 키를 제한합니다.	문자열
s3:VersionId	s3:PutObjectVersionTagging 작업에 대한 권한을 특정 객체 버전으로 제한할 수 있습니다.	문자열
s3:authtype	들어오는 요청을 특정 인증 방법으로 제한합니다.	문자열
s3:delimiter	사용자에게 GET Bucket Object versions 요청에 구분 기호 파라미터를 지정하도록 요구할 수 있습니다.	문자열
s3:locationconstraint	사용자가 특정 리전에서만 버킷을 만들도록 제한할 수 있습니다.	문자열
s3:max-keys	사용자에게 max-keys(최대 키 개수) 파라미터를 지정하도록 요구함으로써 ListBucket 요청에 대한 응답으로 Amazon S3가 반환하는 키의 개수를 제한할 수 있습니다.	숫자
s3:object-lock-legal-hold	지정된 객체 법적 보존 상태를 강제 적용합니다.	문자열
s3:object-lock-mode	지정된 객체 보존 모드를 강제 적용합니다.	문자열

조건 키	설명	유형
s3:object-lock-remaining-retention-days	남은 보존 일수를 기준으로 객체를 강제 적용합니다.	문자열
s3:object-lock-retain-until-date	특정 retain-until-date를 강제 적용합니다.	문자열
s3:prefix	ListBucket API의 응답을 키 이름에 특정 접두사가 있는 경우로 제한할 수 있습니다.	문자열
s3:signatureage	인증된 요청에서 서명이 유효한 시간(밀리초)을 식별합니다.	숫자
s3:signatureversion	인증된 요청에 대해 지원되는 AWS 서명 버전을 식별합니다.	문자열
s3:versionid	특정 객체 버전별로 액세스를 필터링합니다.	문자열
s3:x-amz-acl	객체를 업로딩할 때 특정 액세스 권한을 요구할 수 있습니다.	문자열
s3:x-amz-content-sha256	버킷의 서명되지 않은 콘텐츠를 허용하지 않습니다.	문자열
s3:x-amz-copy-source	복사 원본을 특정 버킷, 버킷의 특정 폴더 또는 버킷의 특정 객체로 제한할 수 있습니다.	문자열
s3:x-amz-grant-full-control	요청에 x-amz-grant-full-control(전체 제어) 헤더가 필요합니다.	문자열
s3:x-amz-grant-read	요청에 x-amz-grant-read(읽기 액세스) 헤더가 필요합니다.	문자열
s3:x-amz-grant-read-acp	요청에 x-amz-grant-read-acp(ACL에 대한 읽기 권한) 헤더가 필요합니다.	문자열
s3:x-amz-grant-write	요청에 x-amz-grant-write(쓰기 액세스) 헤더가 필요합니다.	문자열
s3:x-amz-grant-write-acp	요청에 x-amz-grant-write-acp(ACL에 대한 쓰기 권한) 헤더가 필요합니다.	문자열
s3:x-amz-metadata-directive	객체가 업로드될 때 특정 동작(COPY 또는 REPLACE)을 강제로 실행할 수 있습니다.	문자열
s3:x-amz-server-side-encryption	사용자가 업로드한 객체를 저장할 때 암호화하도록 요청에 이 헤더를 지정하도록 사용자에게 요구할 수 있습니다.	문자열
s3:x-amz-server-side-encryption-aws-kms-key-id	서버 측 암호화를 위해 특정 AWS KMS 고객 관리형 CMK가 필요합니다.	문자열
s3:x-amz-storage-class	스토리지 클래스별로 액세스를 필터링합니다.	문자열
s3:x-amz-website-redirect-location	특정 웹 사이트 리디렉션 위치로 액세스를 필터링합니다.	문자열

버킷 정책 예제

이 단원에서는 몇 가지 버킷 정책의 일반적인 사용 사례에 대한 예제를 제시합니다. 정책에서는 리소스 값에 *bucket*과 *examplebucket* 문자열을 사용합니다. 이러한 정책을 테스트하려면 이 문자열을 버킷 이름으로 대체합니다. 액세스 정책 언어에 대한 자세한 내용은 Amazon S3의 정책 및 권한 (p. 339) 단원을 참조하십시오.

> **Note**
>
> 버킷 정책은 크기가 20KB로 제한됩니다.

AWS 정책 생성기를 사용하여 Amazon S3 버킷에 대한 버킷 정책을 생성할 수 있습니다. 그런 다음 생성된 문서를 사용하여 여러 타사 도구 또는 애플리케이션을 통해 Amazon S3 콘솔을 사용하여 버킷 정책을 설정할 수 있습니다.

> **Important**
>
> Amazon S3 콘솔을 사용하여 권한을 테스트할 경우 콘솔이 요구하는 추가 권한 즉, `s3:ListAllMyBuckets`, `s3:GetBucketLocation` 및 `s3:ListBucket` 권한을 부여해야 합니다. 콘솔을 사용하여 사용자에게 권한을 부여하고 테스트하는 방법에 대한 예는 연습: 사용자 정책을 사용하여 버킷에 대한 액세스 제어 (p. 421)을 참조하십시오.

주제

- 추가 조건을 포함하여 여러 계정에 권한 부여 (p. 408)
- 익명 사용자에게 읽기 전용 권한 부여 (p. 409)
- 특정 IP 주소 액세스 제한 (p. 409)
- 특정 HTTP Referer에 대한 액세스 제한 (p. 410)
- Amazon CloudFront OAI에 권한 부여 (p. 411)
- MFA가 필요한 버킷 정책 추가 (p. 411)
- 버킷 소유자가 완벽하게 제어할 수 있도록 보증하면서 객체에 업로드하는 교차 계정 권한 부여 (p. 413)
- Amazon S3 인벤토리 및 Amazon S3 분석에 권한 부여 (p. 414)
- Amazon S3의 VPC 엔드포인트에 대한 예제 버킷 정책 (p. 414)

추가 조건을 포함하여 여러 계정에 권한 부여

다음 정책 예제는 `s3:PutObject` 및 `s3:PutObjectAcl` 권한을 여러 AWS 계정에 부여하고, 이러한 작업에 대한 요청에 `public-read` 표준 ACL(액세스 제어 목록)을 포함하도록 요구합니다. 자세한 내용은 Amazon S3 작업 (p. 342) 및 Amazon S3 조건 키 (p. 345) 단원을 참조하십시오.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"AddCannedAcl",
      "Effect":"Allow",
      "Principal": {"AWS":
 ["arn:aws:iam::111122223333:root","arn:aws:iam::444455556666:root"]},
      "Action":["s3:PutObject","s3:PutObjectAcl"],
      "Resource":["arn:aws:s3:::examplebucket/*"],
      "Condition":{"StringEquals":{"s3:x-amz-acl":["public-read"]}}
    }
  ]
}
```

익명 사용자에게 읽기 전용 권한 부여

다음 예제 정책은 모든 퍼블릭 익명 사용자에게 `s3:GetObject` 권한을 부여합니다. (이들에게 허용되는 권한 및 작업 목록은 Amazon S3 작업 (p. 342)을 참조하십시오.) 이 권한을 사용하면 모든 사용자가 객체 데이터를 읽을 수 있어, 버킷을 웹 사이트로 구성할 때 그리고 모든 사람이 버킷의 객체를 읽을 수 있기를 원할 때 유용합니다. 버킷 정책을 사용하여 익명 사용자에게 읽기 전용 권한을 부여하기 전에 버킷에 대한 퍼블릭 액세스 차단 설정을 비활성화해야 합니다. 자세한 내용은 웹 사이트 액세스에 대한 권한 설정 (p. 553) 단원을 참조하십시오.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"PublicRead",
      "Effect":"Allow",
      "Principal": "*",
      "Action":["s3:GetObject"],
      "Resource":["arn:aws:s3:::examplebucket/*"]
    }
  ]
}
```

> **Warning**
>
> Amazon S3 버킷에 대한 익명 액세스 권한을 부여하거나 퍼블릭 액세스 차단 설정을 비활성화할 때 주의하십시오. 익명 액세스 권한을 부여하면 전 세계 누구나 버킷에 액세스할 수 있습니다. 정적 웹 사이트 호스팅 (p. 548)용과 같이 특별히 필요한 경우가 아니면 Amazon S3 버킷에 익명 액세스 권한을 부여하지 않는 것이 좋습니다.

특정 IP 주소 액세스 제한

다음 예제에서는 조건에 지정된 IP 주소 범위에서 요청이 시작되지 않는 한, 모든 사용자가 지정된 S3 버킷의 객체에 대한 Amazon S3 작업을 수행할 수 있는 권한을 거부합니다.

이 문은 54.240.143.0/24를 허용되는 IPv4(인터넷 프로토콜 버전 4) IP 주소의 범위로 식별합니다.

`Condition` 블록은 Amazon S3 조건과 AWS 전체 범위 조건 키인 `aws:SourceIp` 조건 키를 사용합니다. 이러한 조건 키에 대한 자세한 내용은 Amazon S3 조건 키 (p. 345)을 참조하십시오. `aws:SourceIp` IPv4 값은 표준 CIDR 표기법을 사용합니다. 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하십시오.

> **Important**
>
> 이 정책을 사용하기 전에 이 예제의 IP 주소 범위를 사용 사례에 적합한 값으로 바꾸십시오. 그렇지 않으면 버킷에 액세스할 수 없습니다.

```
{
  "Version": "2012-10-17",
  "Id": "S3PolicyId1",

  "Statement": [

    {
      "Sid": "IPAllow",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::examplebucket/*",
      "Condition": {
         "NotIpAddress": {"aws:SourceIp": "54.240.143.0/24"}
```

```
      }
    }
  ]
}
```

IPv4 및 IPv6 주소 허용

IPv6 주소를 사용하고자 하는 경우, IPv6으로 전환하는 동안 조직의 정책이 계속 적용되도록 하려면 기존 IPv4 범위 외에도 IPv6 주소 범위를 포함하도록 조직의 모든 정책을 업데이트하는 것이 좋습니다.

다음 버킷 정책 예제에서는 IPv4 및 IPv6 주소 범위를 혼합하여 조직의 유효 IP 주소를 모두 표현하는 방법을 보여 줍니다. 이 정책 예제에서는 IP 주소 예 `54.240.143.1` 및 `2001:DB8:1234:5678::1`에 대한 액세스를 허용하며, 주소 `54.240.143.129` 및 `2001:DB8:1234:5678:ABCD::1`에 대한 액세스는 거부합니다.

`aws:SourceIp`에 대한 IPv6 값은 표준 CIDR 형식이어야 합니다. IPv6의 경우 0의 범위를 나타내기 위해 `::`을 사용할 수 있습니다(예: `2032001:DB8:1234:5678::/64`). 자세한 내용은 IAM 사용 설명서의 IP 주소 조건 연산자를 참조하십시오.

> **Important**
>
> 이 정책을 사용하기 전에 이 예제의 IP 주소 범위를 사용 사례에 적합한 값으로 바꾸십시오. 그렇지 않으면 버킷에 액세스할 수 없습니다.

```
{
  "Id":"PolicyId2",
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"AllowIPmix",
      "Effect":"Allow",
      "Principal":"*",
      "Action":"s3:*",
      "Resource":"arn:aws:s3:::examplebucket/*",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": [
            "54.240.143.0/24",
            "2001:DB8:1234:5678::/64"
          ]
        },
        "NotIpAddress": {
          "aws:SourceIp": [
             "54.240.143.128/30",
             "2001:DB8:1234:5678:ABCD::/80"
          ]
        }
      }
    }
  ]
}
```

특정 HTTP Referer에 대한 액세스 제한

Amazon S3 버킷인 `examplebucket`에 저장된 사진과 동영상 링크가 포함된 도메인 이름(`www.example.com` 또는 `example.com`)의 웹 사이트가 있다고 가정합니다. 기본적으로 모든 Amazon S3 리소스는 프라이빗이므로 리소스를 생성한 AWS 계정만 이 리소스에 액세스할 수 있습니다. 웹 사이트에서 이들 객체에 대한 읽기 권한을 허용하려면, `s3:GetObject` 키를 사용하여 특정 웹 페이지에서만 GET 요청을 허용하 조건으로 `aws:Referer` 권한을 허용하는 버킷 정책을 추가할 수 있습니다. 다음 정책은 `StringLike` 조건 키로 `aws:Referer` 조건을 지정합니다.

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

```
  "Id":"http referer policy example",
  "Statement":[
    {
      "Sid":"Allow get requests originating from www.example.com and example.com.",
      "Effect":"Allow",
      "Principal":"*",
      "Action":"s3:GetObject",
      "Resource":"arn:aws:s3:::examplebucket/*",
      "Condition":{
        "StringLike":{"aws:Referer":["http://www.example.com/*","http://example.com/*"]}
      }
    }
  ]
}
```

사용하는 브라우저에는 요청의 HTTP `referer` 헤더가 포함되어야 합니다.

Amazon CloudFront OAI에 권한 부여

다음 예제 버킷 정책은 CloudFront 오리진 액세스 ID(OAI)에 권한을 부여하여 Amazon S3 버킷에서 모든 객체를 가져옵니다(읽습니다). CloudFront OAI를 사용하면 사용자가 Amazon S3가 아니라 CloudFront를 통해 버킷의 객체에 직접 액세스하도록 허용할 수 있습니다. 자세한 내용은 Amazon CloudFront 개발자 안내서의 오리진 액세스 ID를 사용하여 Amazon S3 콘텐츠에 대한 액세스 제한을 참조하십시오.

다음 정책은 OAI의 ID를 정책의 `Principal`로 사용합니다. S3 버킷 정책을 사용하여 CloudFront OAI에 액세스 권한을 부여하는 방법에 대한 자세한 내용은 Amazon CloudFront 개발자 안내서의 Amazon S3 버킷 정책 사용을 참조하십시오.

이 예제를 사용하려면

- *`EH1HDMB1FH2TC`*를 OAI의 ID로 대체합니다. OAI의 ID를 찾으려면 CloudFront 콘솔에서 오리진 액세스 ID 페이지를 참조하거나 CloudFront API에서 ListCloudFrontOriginAccessIdentities를 사용합니다.
- *`aws-example-bucket`*을 Amazon S3 버킷의 이름으로 대체합니다.

```
{
    "Version": "2012-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
 Identity EH1HDMB1FH2TC"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::aws-example-bucket/*"
        }
    ]
}
```

MFA가 필요한 버킷 정책 추가

Amazon S3은 Amazon S3 리소스에 액세스하기 위해 멀티 팩터 인증(MFA)을 적용할 수 있는 기능인 MFA 보호 API 액세스를 지원합니다. Multi-Factor Authentication은 AWS 환경에 적용할 수 있는 추가 보안 레벨을 제공합니다. 이는 사용자가 유효한 MFA 코드를 제공하여 MFA 디바이스를 물리적으로 가지고 있음을 증명하게 하는 보안 기능입니다. 자세한 내용은 AWS Multi-Factor Authentication을 참조하십시오. Amazon S3 리소스에 액세스하는 모든 요청에 대해 MFA가 필요할 수 있습니다.

버킷 정책에서 `aws:MultiFactorAuthAge` 키를 사용하여 MFA 요구 사항을 적용할 수 있습니다. AWS Identity and Access Management(IAM) 사용자는 AWS Security Token Service(AWS STS)에서 발행한 임시

자격 증명을 사용하여 Amazon S3 리소스에 액세스할 수 있습니다. AWS STS 요청 시 MFA 코드를 제공합니다.

Amazon S3에서 Multi-Factor Authentication으로 요청을 수신한 경우 `aws:MultiFactorAuthAge` 키는 임시 자격 증명이 생성된 시간(초 단위)을 나타내는 숫자 값을 제공합니다. 요청 시 제공된 임시 자격 증명이 MFA 디바이스를 사용하여 만들어진 경우 이 키 값은 null(없음)입니다. 버킷 정책에서는 다음 예제 버킷 정책에서 나온 것처럼 이 값을 확인할 수 있는 조건을 추가할 수 있습니다. MFA를 사용하여 요청이 인증되지 않으면 정책은 `examplebucket` 버킷의 `/taxdocuments` 폴더에서 모든 Amazon S3 작업을 거부합니다. MFA에 대해 자세히 알아보려면 IAM 사용 설명서의 AWS에서 Multi-Factor Authentication(MFA) 사용하기를 참조하십시오.

```
{
    "Version": "2012-10-17",
    "Id": "123",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::examplebucket/taxdocuments/*",
        "Condition": { "Null": { "aws:MultiFactorAuthAge": true }}
      }
    ]
 }
```

`Null` 블록의 `Condition` 조건이 true로 평가되려면 `aws:MultiFactorAuthAge` 키 값이 null이 되어 요청 시 임시 보안 자격 증명이 MFA 키 없이 만들어 졌음을 나타내야 합니다.

다음 버킷 정책은 이전 버킷 정책을 확장한 것입니다. 이 버킷 정책에는 두 가지 정책 명령문이 포함됩니다. 하나의 문은 모든 사용자에게 버킷(`examplebucket`)에 대한 `s3:GetObject` 권한을 허용합니다. 다른 문은 버킷의 `examplebucket/taxdocuments` 폴더에 액세스할 때 MFA를 요구함으로써 액세스를 제한합니다.

```
{
    "Version": "2012-10-17",
    "Id": "123",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::examplebucket/taxdocuments/*",
        "Condition": { "Null": { "aws:MultiFactorAuthAge": true } }
      },
      {
        "Sid": "",
        "Effect": "Allow",
        "Principal": "*",
        "Action": ["s3:GetObject"],
        "Resource": "arn:aws:s3:::examplebucket/*"
      }
    ]
 }
```

요청 인증 시 사용된 임시 보안 자격 증명의 수명 주기에 상관없이 `aws:MultiFactorAuthAge` 키가 유효한 기간을 제한하는 숫자 조건을 선택적으로 사용할 수 있습니다. 예를 들어, 다음 버킷 정책은 MFA 인증 요구 이외에도 임시 세션이 얼마 전에 생성되었는지도 확인합니다. `aws:MultiFactorAuthAge` 키 값이 임시 세션이 1시간(3,600초) 전에 생성되었음을 나타낼 경우 해당 정책은 모든 작업을 거부합니다.

```
{
    "Version": "2012-10-17",
    "Id": "123",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::examplebucket/taxdocuments/*",
        "Condition": {"Null": {"aws:MultiFactorAuthAge": true }}
      },
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::examplebucket/taxdocuments/*",
        "Condition": {"NumericGreaterThan": {"aws:MultiFactorAuthAge": 3600 }}
       },
       {
         "Sid": "",
         "Effect": "Allow",
         "Principal": "*",
         "Action": ["s3:GetObject"],
         "Resource": "arn:aws:s3:::examplebucket/*"
       }
    ]
 }
```

버킷 소유자가 완벽하게 제어할 수 있도록 보증하면서 객체에 업로드하는 교차 계정 권한 부여

다른 AWS 계정이 버킷에 객체를 업로드할 수 있도록 허용할 수 있습니다. 하지만 버킷 소유자로서 버킷에 업로드된 객체를 완벽하게 제어해야 한다고 결정할 수 있습니다. 해당 계정이 이메일 주소 (`xyz@amazon.com`)에서 식별된 버킷 소유자에 대한 액세스를 완벽히 제어하는 권한을 부여하지 않으면 다음 정책은 특정 AWS 계정(`111111111111`)이 객체를 업로드할 수 없도록 합니다. 정책의 `StringNotEquals` 조건은 요구 사항을 표현하도록 `s3:x-amz-grant-full-control` 조건 키를 지정합니다(Amazon S3 조건 키 (p. 345) 참조).

```
{
   "Version":"2012-10-17",
   "Statement":[
     {
       "Sid":"111",
       "Effect":"Allow",
       "Principal":{"AWS":"1111111111"},
       "Action":"s3:PutObject",
       "Resource":"arn:aws:s3:::examplebucket/*"
     },
     {
       "Sid":"112",
       "Effect":"Deny",
       "Principal":{"AWS":"1111111111" },
       "Action":"s3:PutObject",
       "Resource":"arn:aws:s3:::examplebucket/*",
       "Condition": {
         "StringNotEquals": {"s3:x-amz-grant-full-control":["emailAddress=xyz@amazon.com"]}
       }
     }
   ]
}
```

Amazon S3 인벤토리 및 Amazon S3 분석에 권한 부여

Amazon S3 인벤토리는 Amazon S3 버킷의 객체 목록을 생성하고, Amazon S3 분석 내보내기는 분석에서 사용되는 데이터의 출력 파일을 생성합니다. 인벤토리가 객체를 열거하는 버킷을 원본 버킷이라고 합니다. 인벤토리 파일이 작성되는 버킷과 분석 내보내기 파일이 작성되는 버킷을 대상 버킷이라고 합니다. Amazon S3 버킷의 인벤토리를 설정할 때와 분석 내보내기를 설정할 때는 대상 버킷의 버킷 정책을 생성해야 합니다. 자세한 내용은 Amazon S3 인벤토리 (p. 459) 및 Amazon S3 분석 – 스토리지 클래스 분석 (p. 255) 단원을 참조하십시오.

다음 버킷 정책 예제는 원본 버킷의 계정에서 대상 버킷에 객체를 쓸 수 있는 Amazon S3 권한(PUT)을 부여합니다. Amazon S3 인벤토리와 Amazon S3 분석 내보내기를 설정할 때는 대상 버킷에서 이와 같은 버킷 정책을 사용합니다.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"InventoryAndAnalyticsExamplePolicy",
      "Effect":"Allow",
      "Principal": {"Service": "s3.amazonaws.com"},
      "Action":["s3:PutObject"],
      "Resource":["arn:aws:s3:::destination-bucket/*"],
      "Condition": {
          "ArnLike": {
              "aws:SourceArn": "arn:aws:s3:::source-bucket"
           },
         "StringEquals": {
             "aws:SourceAccount": "1234567890",
             "s3:x-amz-acl": "bucket-owner-full-control"
          }
       }
    }
  ]
}
```

Amazon S3의 VPC 엔드포인트에 대한 예제 버킷 정책

Amazon S3 버킷 정책을 사용하여 특정 Amazon Virtual Private Cloud (Amazon VPC) 엔드포인트나 특정 VPC에서 버킷에 대한 액세스를 제어할 수 있습니다. 이 단원에는 VPC 종단점에서 Amazon S3 버킷 액세스를 제어하는 데 사용할 수 있는 버킷 정책 예제가 포함되어 있습니다. VPC 엔드포인트 설정 방법을 알아보려면 Amazon VPC 사용 설명서에서 VPC 엔드포인트 항목을 참조하십시오.

Amazon VPC에서는 AWS 리소스를 사용자가 정의한 가상 네트워크로 시작할 수 있습니다. VPC 종단점에서 사용자는 인터넷, VPN 연결, NAT 인스턴스 또는 AWS Direct Connect를 통한 액세스를 요구하지 않고 사용자의 VPC와 다른 AWS 서비스 간 프라이빗 연결을 생성할 수 있습니다.

Amazon S3의 VPC 엔드포인트는 Amazon S3에만 연결을 허용하는 VPC 내 논리적 대상입니다. VPC 엔드포인트는 Amazon S3에 대한 요청을 라우팅하고 VPC에 대한 응답을 라우팅합니다. VPC 엔드포인트에서만 요청을 라우팅하는 방법을 변경합니다. Amazon S3 공인 엔드포인트와 DNS 이름은 계속해서 VPC 엔드포인트로 작업합니다. Amazon S3이 포함된 Amazon VPC 엔드포인트 사용에 대한 중요 정보는 Amazon VPC 사용 설명서의 게이트웨이 VPC 엔드포인트 및 Amazon S3에 대한 엔드포인트 항목을 참조하십시오.

Amazon S3의 VPC 엔드포인트는 Amazon S3 데이터에 대한 액세스를 제어하는 두 가지 방법을 제공합니다.

- 사용자는 특정 VPC 종단점을 통해 허용되는 요청, 사용자 또는 그룹을 제어할 수 있습니다. 이러한 유형의 액세스 제어에 대한 자세한 내용은 Amazon VPC 사용 설명서의 VPC 종단점으로 서비스에 대한 액세스 제어를 참조하십시오.
- Amazon S3 버킷 정책을 사용하여 버킷에 액세스할 수 있는 VPC 또는 VPC 종단점을 제어할 수 있습니다. 이러한 유형의 버킷 정책 제어에 대한 예는 액세스 제한에 대한 다음 주제를 참조하십시오.

주제

- 특정 VPC 종단점에 대한 액세스 제한 (p. 415)
- 특정 VPC에 대한 액세스 제한 (p. 415)
- 관련 리소스 (p. 416)

> **Important**
>
> 이 단원에서 설명하는 VPC 종단점에 Amazon S3 버킷 정책을 적용할 경우 의도치 않게 버킷에 대한 액세스를 차단할 수 있습니다. VPC 종단점에서 시작하는 연결에 대한 버킷 액세스를 특별히 제한하기 위한 버킷 권한은 버킷에 대한 모든 연결을 차단할 수 있습니다. 이 문제를 해결하는 방법에 대한 자세한 내용은 내 버킷 정책의 VPC 또는 VPC 종단점 ID가 잘못되었습니다. 버킷에 액세스할 수 있도록 정책을 수정하는 방법은 무엇입니까?(AWS 지원 지식 센터)를 참조하십시오.

특정 VPC 종단점에 대한 액세스 제한

다음은 ID가 `vpce-1a2b3c4d`인 VPC 종단점에서만 특정 버킷 `examplebucket`에 대한 액세스를 제한하는 Amazon S3 버킷 정책의 예제입니다. 이 정책은 지정된 엔드포인트가 사용되지 않으면 버킷에 대한 모든 액세스를 거부합니다. `aws:sourceVpce` 조건이 엔드포인트를 지정하는 데 사용됩니다. `aws:sourceVpce` 조건은 VPC 종단점 리소스의 Amazon 리소스 이름(ARN)을 요구하지 않고 VPC 종단점 ID만 요구합니다. 정책에서 조건을 사용하는 것에 대한 자세한 내용은 Amazon S3 조건 키 (p. 345)을 참조하십시오.

> **Important**
>
> 이 정책을 사용하기 전에 이 예제의 VPC 엔드포인트 ID를 사용 사례에 적합한 값으로 바꾸십시오. 그렇지 않으면 버킷에 액세스할 수 없습니다.

```
{
   "Version": "2012-10-17",
   "Id": "Policy1415115909152",
   "Statement": [
     {
       "Sid": "Access-to-specific-VPCE-only",
       "Principal": "*",
       "Action": "s3:*",
       "Effect": "Deny",
       "Resource": ["arn:aws:s3:::examplebucket",
                    "arn:aws:s3:::examplebucket/*"],
       "Condition": {
         "StringNotEquals": {
           "aws:sourceVpce": "vpce-1a2b3c4d"
         }
       }
     }
   ]
}
```

특정 VPC에 대한 액세스 제한

`aws:sourceVpc` 조건을 사용하여 특정 VPC에 대한 액세스를 제한하는 버킷 정책을 만들 수 있습니다. 이는 동일한 VPC에 여러 VPC 종단점이 구성되어 있으며, 모든 엔드포인트에 대해 Amazon S3 버킷에 대한 액세스를 관리하려고 할 때 유용합니다. 다음은 VPC `vpc-111bbb22`가 `examplebucket`과 해당 객체에 액세스할 수 있도록 허용하는 정책의 예입니다. 이 정책은 지정된 VPC가 사용되지 않으면 버킷에 대한 모든 액세스를 거부합니다. `vpc-111bbb22` 조건 키는 VPC 리소스의 ARN을 요구하지 않고 VPC ID만 요구합니다.

> **Important**
>
> 이 정책을 사용하기 전에 이 예제의 VPC ID를 사용 사례에 적합한 값으로 바꾸십시오. 그렇지 않으면 버킷에 액세스할 수 없습니다.

```
{
   "Version": "2012-10-17",
   "Id": "Policy1415115909153",
   "Statement": [
     {
       "Sid": "Access-to-specific-VPC-only",
       "Principal": "*",
       "Action": "s3:*",
       "Effect": "Deny",
       "Resource": ["arn:aws:s3:::examplebucket",
                    "arn:aws:s3:::examplebucket/*"],
       "Condition": {
         "StringNotEquals": {
           "aws:sourceVpc": "vpc-111bbb22"
         }
       }
     }
   ]
}
```

관련 리소스

- 버킷 정책 예제 (p. 408)
- Amazon VPC 사용 설명서의 VPC 엔드포인트

사용자 정책 예제

이 단원에서는 Amazon S3에 대한 사용자 액세스를 제어하는 몇 가지 IAM 사용자 정책을 보여줍니다. 액세스 정책 언어에 대한 자세한 내용은 Amazon S3의 정책 및 권한 (p. 339) 단원을 참조하십시오.

다음 정책 예제는 프로그래밍 방식으로 테스트하는 경우 작동합니다. 그러나 Amazon S3 콘솔을 통해 정책을 사용하기 위해서는 콘솔에 필요한 추가 권한을 부여해야 합니다. Amazon S3 콘솔에서 이와 같은 정책을 사용하는 방법에 대한 자세한 내용은 연습: 사용자 정책을 사용하여 버킷에 대한 액세스 제어 (p. 421) 단원을 참조하십시오.

주제

- 버킷 중 하나에 대한 IAM 사용자 액세스 허용 (p. 416)
- 버킷의 폴더에 대한 각 IAM 사용자의 액세스 허용 (p. 417)
- Amazon S3에 공유 폴더를 보유할 수 있도록 그룹을 허용 (p. 420)
- 모든 사용자가 기업 버킷의 일부로 객체를 읽을 수 있도록 허용 (p. 420)
- 파트너가 기업 버킷의 특정 부분에 파일을 둘 수 있도록 허용 (p. 420)
- 연습: 사용자 정책을 사용하여 버킷에 대한 액세스 제어 (p. 421)

버킷 중 하나에 대한 IAM 사용자 액세스 허용

이 예제에서는 AWS 계정의 IAM 사용자에게 버킷 중 하나인 `examplebucket`에 대한 액세스 권한을 부여하고 이 사용자에게 객체를 추가, 업데이트, 삭제하도록 허용하려 합니다.

`s3:PutObject`, `s3:GetObject` 및 `s3:DeleteObject` 권한을 사용자에게 부여하는 것에 추가로, 이 정책에서는 `s3:ListAllMyBuckets`, `s3:GetBucketLocation` 및 `s3:ListBucket` 권한 역시 부여합니다. 이러한 권한은 콘솔에 필요한 추가 권한입니다. 또한 콘솔에서 객체를 복사, 자르기 및 붙여넣기를 할 수 있으려면 `s3:PutObjectAcl` 및 `s3:GetObjectAcl` 작업이 필요합니다. 콘솔을 사용하여 사용자에게 권한을 부여하고 테스트하는 방법에 대한 예는 연습: 사용자 정책을 사용하여 버킷에 대한 액세스 제어 (p. 421)을 참조하십시오.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListAllMyBuckets"
         ],
         "Resource":"arn:aws:s3:::*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket",
            "s3:GetBucketLocation"
         ],
         "Resource":"arn:aws:s3:::examplebucket"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:GetObject",
            "s3:GetObjectAcl",
            "s3:DeleteObject"
         ],
         "Resource":"arn:aws:s3:::examplebucket/*"
      }
   ]
}
```

버킷의 폴더에 대한 각 IAM 사용자의 액세스 허용

이 예제에서는 두 명의 IAM 사용자인 Alice와 Bob에게 `examplebucket` 버킷에 대한 액세스 권한을 부여함으로써 이들이 객체를 추가, 업데이트, 삭제할 수 있게 하려 합니다. 그러나 각 사용자의 액세스는 버킷의 단일 폴더로 제한하려 합니다. 이 경우, 사용자 이름과 일치하는 이름으로 폴더를 만들 수 있습니다.

```
examplebucket
   Alice/
   Bob/
```

각 사용자에게 자신의 폴더에 대해서만 액세스 권한을 부여하기 위해 각 사용자에 대한 정책을 작성하고 이를 개별적으로 연결할 수 있습니다. 예를 들어 다음 정책을 사용자 중 Alice에게 연결하여 `examplebucket/Alice` 폴더에 대한 특정 Amazon S3 권한을 허용할 수 있습니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::examplebucket/Alice/*"
      }
   ]
```

```
}
```

그런 뒤 Bob 값의 Bob 폴더를 식별하는 유사한 정책을 사용자인 Resource에게 연결합니다.

정책을 개별 사용자에게 연결하는 대신 정책 변수를 사용하는 단일 정책을 작성하여 이 정책을 그룹에 연결할 수도 있습니다. 우선 그룹을 하나 만들어 Alice와 Bob을 모두 이 그룹에 추가합니다. 다음 정책 예제에서는 examplebucket/${aws:username} 폴더의 Amazon S3 권한 집합을 허용합니다. 정책이 평가될 때 정책 변수 ${aws:username}은 요청자의 사용자 이름으로 대체됩니다. 예를 들어, Alice가 요청을 보내 객체를 배치하는 경우, 이 작업은 Alice가 객체를 examplebucket/Alice 폴더에 업로드하는 경우에만 허용됩니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::examplebucket/${aws:username}/*"
      }
   ]
}
```

Note

정책 변수를 사용하는 경우 정책에 명시적으로 버전 2012-10-17을 지정해야 합니다. 액세스 정책 언어의 기본 버전인 2008-10-17은 정책 변수를 지원하지 않습니다.

앞서 다룬 정책을 Amazon S3 콘솔에서 테스트하려는 경우, 콘솔에서는 다음 정책에서와 같이 추가 Amazon S3 권한에 대한 권한을 요구합니다. 콘솔에서 이와 같은 정책을 사용하는 방법에 대한 자세한 내용은 연습: 사용자 정책을 사용하여 버킷에 대한 액세스 제어 (p. 421) 단원을 참조하십시오.

```
{
 "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGroupToSeeBucketListInTheConsole",
      "Action": [ "s3:ListAllMyBuckets", "s3:GetBucketLocation" ],
      "Effect": "Allow",
      "Resource": [ "arn:aws:s3:::*"  ]
    },
    {
      "Sid": "AllowRootLevelListingOfTheBucket",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::examplebucket"],
      "Condition":{
            "StringEquals":{
                    "s3:prefix":[""], "s3:delimiter":["/"]
                            }
                  }
    },
    {
      "Sid": "AllowListBucketOfASpecificUserPrefix",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
```

```
      "Resource": ["arn:aws:s3:::examplebucket"],
      "Condition":{  "StringLike":{"s3:prefix":["${aws:username}/*"] }
       }
    },
      {
     "Sid": "AllowUserSpecificActionsOnlyInTheSpecificUserPrefix",
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::examplebucket/${aws:username}/*"
      }
  ]
}
```

Note

2012-10-17 버전의 정책에서, 정책 변수는 `$`로 시작합니다. 이러한 구문상의 변경은 객체 키에 `$`가 포함된 경우 잠재적으로 충돌을 야기합니다. 예를 들어, 객체 키 `my$file`을 정책에 포함하려면 `$` 문자를 `${$}`로 지정하여 `my${$}file`과 같이 나타내야 합니다.

IAM 사용자 이름은 읽기 쉬운 표시 이름 식별자이지만 고유한 전역 이름일 필요는 없습니다. 예를 들어, Bob이라는 사용자가 조직을 떠난 후 또 다른 Bob이 조직에 합류하는 경우, 새로 온 Bob이 이전 Bob의 정보에 액세스할 수도 있습니다. 사용자 이름을 사용하는 대신 사용자 ID를 기반으로 한 폴더를 만들 수 있습니다. 각 사용자 ID는 고유합니다. 이 경우, `${aws:userid}` 정책 변수를 사용하여 앞서 다룬 정책을 수정해야 합니다. 사용자 ID에 대한 자세한 내용은 IAM 사용 설명서의 IAM 식별자 단원을 참조하십시오.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::my_corporate_bucket/home/${aws:userid}/*"
      }
   ]
}
```

버킷의 폴더에 대한 비 IAM 사용자(모바일 앱 사용자)의 액세스 허용

S3 버킷에 사용자 데이터를 저장하는 모바일 게임 앱을 개발하려 한다고 가정합니다. 각 앱 사용자에 대해 버킷에 폴더를 만들려고 합니다. 또한 각 사용자의 액세스를 각자의 폴더로 제한하려 합니다. 그러나 누군가 앱을 다운로드하여 게임 플레이를 시작하기 전에는 사용자 ID가 없으므로 폴더를 만들 수 없습니다.

이 경우 사용자에게 Login with Amazon, Facebook 또는 Google과 같은 퍼블릭 ID 공급자를 사용하여 앱에 로그인하도록 요구할 수 있습니다. 사용자가 이들 공급자 중 하나를 통해 앱에 로그인한 후에는 런타임 시 사용자별 폴더 생성에 사용할 수 있는 사용자 ID를 갖게 됩니다.

그런 뒤 AWS Security Token Service의 웹 자격 증명 연동을 사용하여 ID 공급자에게 받은 정보를 앱과 통합하고 각 사용자에 대한 임시 보안 자격 증명을 얻을 수 있습니다. 그런 뒤에는 앱에서 버킷에 액세스할 수 있

도록 허용하고 사용자별 폴더를 만들고 데이터를 업로드하는 등의 작업을 수행하는 IAM 정책을 만들 수 있습니다. 웹 자격 증명 연동에 대한 자세한 내용은 IAM 사용 설명서의 웹 자격 증명 연동에 대하여 단원을 참조하십시오.

Amazon S3에 공유 폴더를 보유할 수 있도록 그룹을 허용

다음 정책을 그룹에 연결하면 그룹 내 모두에게 Amazon S3의 `my_corporate_bucket/share/marketing` 폴더에 대한 액세스 권한이 부여됩니다. 그룹 멤버는 정책에 나와 있는 특정 Amazon S3 권한과 지정된 폴더의 객체에 대해서만 액세스할 수 있도록 허용됩니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::my_corporate_bucket/share/marketing/*"
      }
   ]
}
```

모든 사용자가 기업 버킷의 일부로 객체를 읽을 수 있도록 허용

이 예제에서는 AWS 계정에서 소유한 모든 IAM 사용자를 포함하는 `AllUsers`라는 그룹을 생성합니다. 그런 뒤 그룹에 `GetObject` 및 `GetObjectVersion`에 대해서만 `my_corporate_bucket/readonly` 폴더의 객체에 대한 액세스 권한을 부여하는 정책을 연결합니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Resource":"arn:aws:s3:::my_corporate_bucket/readonly/*"
      }
   ]
}
```

파트너가 기업 버킷의 특정 부분에 파일을 둘 수 있도록 허용

이 예제에서는 파트너사를 나타내는 `WidgetCo`라는 그룹을 만듭니다. 파트너사에서 액세스 권한을 필요로 하는 사람이나 애플리케이션을 위해 IAM 사용자를 만든 다음 그 사용자를 그룹에 넣습니다.

그런 뒤 그룹에 기업 버킷의 `PutObject` 폴더에 대한 `my_corporate_bucket/uploads/widgetco` 액세스 권한을 부여하는 정책을 연결합니다.

`WidgetCo` 그룹이 버킷에 대한 다른 작업을 못하게 하려면 AWS 계정의 Amazon S3 리소스에 대한 `PutObject`를 제외한 모든 Amazon S3 권한에 권한을 명시적으로 거부하는 문을 추가합니다. 이 단계는 AWS 계정의 다른 위치에 사용자에게 Amazon S3 리소스에 대한 폭넓은 액세스 권한을 부여하는 사용 중인 광범위한 정책이 있는 경우에만 필요합니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"s3:PutObject",
         "Resource":"arn:aws:s3:::my_corporate_bucket/uploads/widgetco/*"
      },
      {
         "Effect":"Deny",
         "NotAction":"s3:PutObject",
         "Resource":"arn:aws:s3:::my_corporate_bucket/uploads/widgetco/*"
      },
      {
         "Effect":"Deny",
         "Action":"s3:*",
         "NotResource":"arn:aws:s3:::my_corporate_bucket/uploads/widgetco/*"
      }
   ]
}
```

연습: 사용자 정책을 사용하여 버킷에 대한 액세스 제어

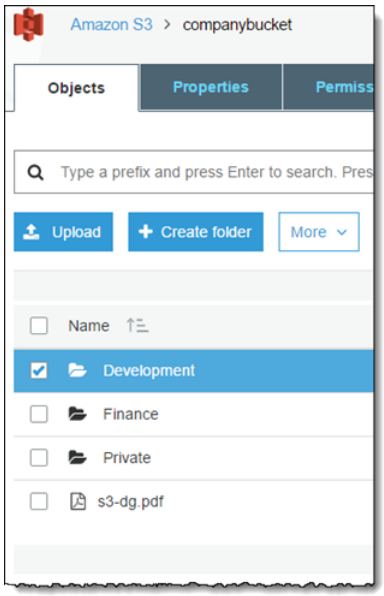
본 연습에서는 Amazon S3에서 사용자 권한이 어떻게 사용되는지를 설명합니다. 이 예제에서는 폴더가 있는 버킷을 만듭니다. 그런 다음 AWS 계정에서 AWS Identity and Access Management(IAM) 사용자를 만들고 이 사용자에게 Amazon S3 버킷과 이 버킷 안에 있는 폴더에 대한 증분 권한을 부여합니다.

주제

- 버킷 및 폴더 기본 사항 (p. 421)
- 연습 요약 (p. 423)
- 연습 준비 (p. 424)
- 1단계: 버킷 만들기 (p. 424)
- 2단계: IAM 사용자 및 그룹 만들기 (p. 425)
- 3단계: IAM 사용자가 권한을 가지고 있지 않는지 확인 (p. 425)
- 4단계: 그룹 수준 권한 부여 (p. 425)
- 5단계: IAM 사용자인 Alice에게 특정 권한 부여 (p. 432)
- 6단계: IAM 사용자인 Bob에게 특정 권한 부여 (p. 435)
- 7단계: Private 폴더에 보안 지정 (p. 436)
- 8단계: 정리 (p. 437)
- 관련 리소스 (p. 437)

버킷 및 폴더 기본 사항

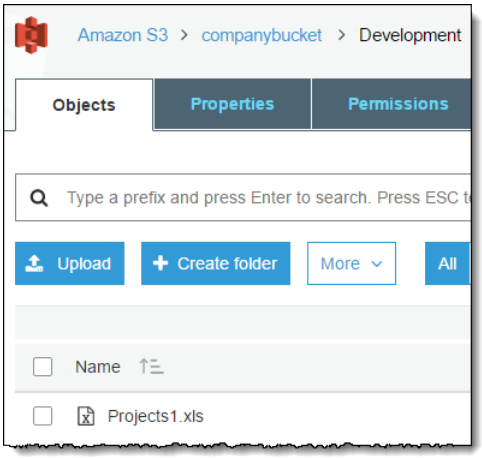
Amazon S3 데이터 모델은 플랫 구조입니다. 버킷을 만들면 이 버킷에 객체가 저장됩니다. 하위 버킷 또는 하위 폴더의 계층 구조는 없지만, 폴더 계층 구조를 에뮬레이션할 수 있습니다. Amazon S3 콘솔과 같은 도구는 다음 이미지와 같이 버킷의 이러한 논리적 폴더 및 하위 폴더의 보기를 제공할 수 있습니다.



콘솔에는 companybucket이라는 버킷에 세 개의 폴더 Private, Development 및 Finance와 하나의 객체 s3-dg.pdf가 있다는 것이 표시됩니다. 콘솔에서는 객체 이름(키)을 사용하여 폴더 및 하위 폴더로 이루어진 논리적 계층 구조를 만듭니다. 다음 예제를 고려하십시오.

- Development 폴더를 만들면 콘솔에서 Development/라는 키를 사용하여 객체가 생성됩니다. 후행 슬래시(/ 구분 기호에 주의합니다.
- Projects1.xls라는 객체를 Development 폴더에 업로드하면 콘솔에서 객체가 업로드되고 객체에 키 Development/Projects1.xls가 제공됩니다.

  키에서 Development는 접두사이고 /는 구분 기호입니다. Amazon S3 API 작업에서는 접두사와 구분 기호가 지원됩니다. 예를 들어, 특정 접두사와 구분 기호를 사용하여 버킷에서 모든 객체의 목록을 가져올 수 있습니다. 콘솔에서 Development 폴더를 열면 콘솔에 해당 폴더 안에 있는 객체가 나열됩니다. 다음 예제에서는 Development 폴더에 객체 하나가 포함되어 있습니다.



콘솔에 companybucket 버킷의 Development 폴더가 나열되면 Amazon S3로 요청이 전송되고 여기에서 요청에 Development라는 접두사와 / 구분 기호가 지정됩니다. 콘솔의 응답은 컴퓨터 파일 시스템의

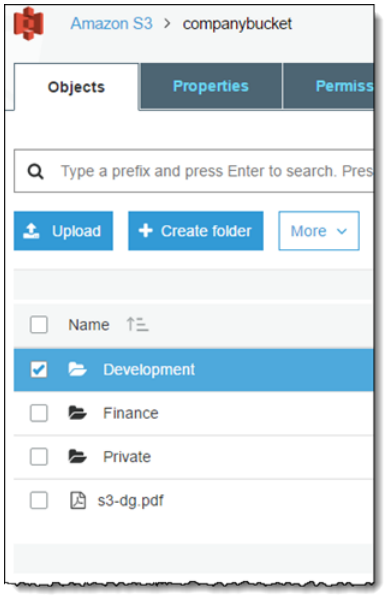
폴더 목록과 같습니다. 이전 예제에서는 `companybucket` 버킷에 `Development/Projects1.xls` 키가 지정된 객체가 하나 있음을 보여 줍니다.

콘솔에서는 논리적 계층 구조를 유추하기 위해 객체 키가 사용됩니다. Amazon S3에는 물리적 계층 구조가 없으며 플랫 파일 구조로 객체가 포함된 버킷만 있습니다. Amazon S3 API를 사용하여 객체를 만들면 논리적 계층 구조를 암시하는 객체 키를 사용할 수 있습니다. 객체의 논리적 계층 구조를 만들면 이 안내에서 시연되는 것과 같이 개별 폴더에 대한 액세스를 관리할 수 있습니다.

시작하기 전에 루트 수준 버킷 콘텐츠의 개념에 익숙해야 합니다. `companybucket` 버킷에 다음과 같은 객체가 있다고 가정합니다.

- `Private/privDoc1.txt`
- `Private/privDoc2.zip`
- `Development/project1.xls`
- `Development/project2.xls`
- `Finance/Tax2011/document1.pdf`
- `Finance/Tax2011/document2.pdf`
- `s3-dg.pdf`

이러한 객체 키를 통해 `Private`, `Development` 및 루트 수준 폴더인 `Finance`와 루트 수준 객체인 `s3-dg.pdf`가 포함된 논리적 계층 구조가 생성됩니다. Amazon S3 콘솔에서 버킷 이름을 선택하면 다음 이미지와 같이 루트 수준 항목이 나타납니다. 콘솔에는 최상위 수준 접두사(`Private/`, `Development/` 및 `Finance/`)가 루트 수준 폴더로 표시됩니다. 객체 키 `s3-dg.pdf`에는 접두사가 없으므로 이 키는 루트 수준 항목으로 나타납니다.



연습 요약

이 연습에서는 안에 세 개의 폴더(`Private`, `Development` 및 `Finance`)가 있는 버킷을 생성합니다.

Alice와 Bob이라는 사용자 두 명이 있습니다. Alice는 `Development` 폴더에만 액세스하도록 하고, Bob은 `Finance` 폴더에만 액세스하도록 하려고 합니다. `Private` 폴더 콘텐츠는 프라이빗으로 유지하려고 합니다. 이 연습에서는 IAM 사용자(이 예제에서는 Alice와 Bob이라는 사용자 이름을 사용)를 만들고 이 사용자에게 필요한 권한을 부여하여 액세스를 관리합니다.

IAM에서는 사용자 그룹을 만들어 이 그룹의 모든 사용자에게 적용되는 그룹 수준 권한을 부여할 수도 있습니다. 이렇게 하면 권한을 보다 효과적으로 관리할 수 있습니다. 이 연습에서는 Alice와 Bob 모두에게 몇 가지 공통적인 권한이 필요합니다. 따라서 `Consultants`라는 그룹도 만든 다음 Alice와 Bob을 모두 이 그룹에 추가합니다. 먼저 그룹 정책을 그룹에 연결하여 권한을 부여합니다. 그런 다음 정책을 특정 사용자에게 연결하여 사용자별 권한을 추가합니다.

> **Note**
>
> 이 연습에서는 `companybucket`을 버킷 이름으로 사용하고, Alice와 Bob을 IAM 사용자로 사용하며, `Consultants`를 그룹 이름으로 사용합니다. Amazon S3에서는 버킷 이름이 글로벌로 고유해야 하기 때문에 버킷 이름을 새로 만드는 이름으로 바꿔야 합니다.

연습 준비

이 예제에서는 AWS 계정 자격 증명을 사용하여 IAM 사용자를 만듭니다. 처음에는 이러한 사용자에게 권한이 없습니다. 이러한 사용자에게 특정 Amazon S3 작업을 수행할 수 있는 권한을 증분 방식으로 부여합니다. 또한 이러한 권한을 테스트하기 위해 각 사용자의 자격 증명을 사용하여 콘솔에 로그인합니다. AWS 계정 소유자로 권한을 증분 방식으로 부여하고 IAM 사용자로 권한을 테스트하면서 로그인과 로그아웃을 반복해야 하는데 매번 다른 자격 증명을 사용해야 합니다. 하나의 브라우저에서 이 테스트를 수행할 수 있지만, 두 개의 브라우저를 사용할 수 있는 경우 프로세스가 더 빠르게 진행됩니다. 한 브라우저를 사용하여 AWS 계정 자격 증명으로 AWS Management 콘솔에 연결하고 다른 브라우저를 사용하여 IAM 사용자 자격 증명과 연결합니다.

AWS 계정 자격 증명으로 AWS Management 콘솔에 로그인하려면 https://console.aws.amazon.com/로 이동합니다. IAM 사용자는 동일한 링크를 사용하여 로그인할 수 없습니다. IAM 사용자는 IAM 지원 로그인 페이지를 사용해야 합니다. 계정 소유자는 사용자에게 이 링크를 제공할 수 있습니다.

IAM에 대한 자세한 내용은 IAM 사용 설명서의 AWS Management 콘솔 로그인 페이지를 참조하십시오.

IAM 사용자를 위한 로그인 링크를 제공하려면

1. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 IAM 대시보드를 선택합니다.
3. IAM 사용자 로그인 링크: 아래의 URL을 기록합니다. 자신의 IAM 사용자 이름 및 암호로 콘솔에 로그인할 수 있도록 IAM 사용자에게 이 링크를 제공하게 됩니다.

1단계: 버킷 만들기

이 단계에서는 AWS 계정 자격 증명을 사용하여 Amazon S3 콘솔에 로그인하고, 버킷을 만든 다음, 폴더(`Development`, `Finance` 및 `Private`)를 버킷에 추가하고, 한 개 또는 두 개의 샘플 문서를 각 폴더에 업로드합니다.

1. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷을 만듭니다.

   단계별 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷을 생성하려면 어떻게 해야 합니까? 단원을 참조하십시오.
3. 버킷에 문서를 하나 업로드합니다.

   이 연습에서는 이 버킷의 루트 수준에 `s3-dg.pdf` 문서가 있다고 가정합니다. 다른 문서를 업로드하는 경우 `s3-dg.pdf`를 해당 파일 이름으로 대체합니다.
4. `Private`, `Finance` 및 `Development`라는 세 개의 폴더를 버킷에 추가합니다.

   폴더를 만들기 위한 단계별 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 폴더 생성을 참조하십시오.
5. 각 폴더에 문서를 한두 개 업로드합니다.

이 연습에서는 문서 몇 개를 각 폴더에 업로드했으며 결과적으로 버킷에 다음 키를 사용하는 객체가 있다고 가정합니다.

- `Private/privDoc1.txt`
- `Private/privDoc2.zip`
- `Development/project1.xls`
- `Development/project2.xls`
- `Finance/Tax2011/document1.pdf`
- `Finance/Tax2011/document2.pdf`
- `s3-dg.pdf`

단계별 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷에 파일 및 폴더를 업로드하려면 어떻게 해야 합니까? 단원을 참조하십시오.

2단계: IAM 사용자 및 그룹 만들기

이제 IAM 콘솔을 사용하여 Alice와 Bob이라는 두 IAM 사용자를 AWS 계정에 추가합니다. 또한 `Consultants`라는 관리자 그룹을 만든 다음, 두 사용자를 모두 이 그룹에 추가합니다.

> **Warning**
>
> 사용자와 그룹을 추가할 때 이러한 사용자에게 권한을 부여하는 정책을 연결하지 마십시오. 처음에는 이러한 사용자가 어떠한 권한도 가지고 있지 않습니다. 다음 단원에서는 증분 방식으로 권한을 부여합니다. 먼저 이러한 IAM 사용자에게 암호를 할당했는지 확인해야 합니다. 이러한 사용자 자격 증명을 사용하여 Amazon S3 작업을 테스트하고 권한이 예상대로 작동하는지 확인합니다.

새 IAM 사용자 만들기에 대한 단계별 지침은 IAM 사용 설명서의 AWS 계정에서 IAM 사용자 생성을 참조하십시오. 이 연습에서 사용자를 만들 때는 AWS Management Console 액세스를 선택하고 프로그래밍 방식 액세스의 선택을 취소합니다.

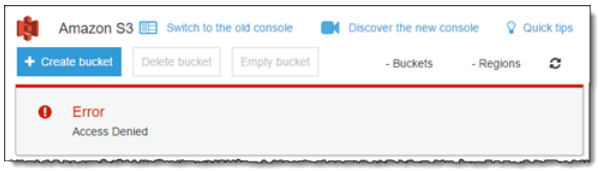
관리자 그룹 만들기에 대한 단계별 지침은 IAM 사용 설명서의 IAM 사용자와 관리자 그룹 처음 만들기를 참조하십시오.

3단계: IAM 사용자가 권한을 가지고 있지 않는지 확인

두 개의 브라우저를 사용하는 경우 이제 두 번째 브라우저를 사용하여 IAM 사용자 자격 증명 중 하나로 콘솔에 로그인할 수 있습니다.

1. IAM 사용자 로그인 링크(IAM 사용자를 위한 로그인 링크를 제공하려면 (p. 424) 참조)를 사용하여 IAM 사용자 자격 증명 중 하나로 AWS Management 콘솔에 로그인합니다.
2. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.

   액세스가 거부되었다는 다음 콘솔 메시지를 확인합니다.



이제 사용자에게 증분 권한을 부여하는 작업을 시작할 수 있습니다. 먼저, 두 사용자가 모두 가지고 있어야 하는 권한을 부여하는 그룹 정책을 연결합니다.

4단계: 그룹 수준 권한 부여

사용자가 다음을 수행할 수 있게 만들려고 합니다.

- 상위 계정이 소유한 모든 버킷 나열 이렇게 하려면 Bob과 Alice에게 `s3:ListAllMyBuckets` 작업을 위한 권한이 있어야 합니다.
- `companybucket` 버킷의 루트 수준 항목, 폴더 및 객체를 나열합니다. 이렇게 하려면 Bob과 Alice에게 `s3:ListBucket` 버킷에 `companybucket` 작업을 수행하기 위한 권한이 있어야 합니다.

먼저, 이러한 권한을 부여하는 정책을 만든 다음 이 정책을 `Consultants` 그룹에 연결합니다.

4.1단계: 모든 버킷을 나열하는 권한 부여

이 단계에서는 사용자가 상위 계정이 소유한 모든 버킷을 나열할 수 있도록 사용자에게 최소한의 권한을 부여하는 관리형 정책을 만듭니다. 그런 다음 정책을 `Consultants` 그룹에 연결합니다. 관리형 정책을 사용자 또는 그룹에 연결하면 상위 AWS 계정이 소유한 버킷의 목록을 가져올 수 있는 사용자 또는 그룹 권한을 부여하는 것입니다.

1. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.

   Note

   사용자 권한을 부여하고 있으므로 IAM 사용자로 로그인하지 않고 AWS 계정 자격 증명을 사용하여 로그인합니다.

2. 관리형 정책을 만듭니다.

   a. 왼쪽 탐색 창에서 정책을 선택한 다음 정책 생성을 선택합니다.

   b. JSON 탭을 선택합니다.

   c. 다음 액세스 정책을 복사하여 정책 텍스트 필드에 붙여 넣습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGroupToSeeBucketListInTheConsole",
      "Action": ["s3:ListAllMyBuckets"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::*"]
    }
  ]
}
```

   정책은 JSON 문서입니다. 문서에서 `Statement`는 객체 배열로, 각 객체는 이름-값 페어 모음을 사용하여 권한을 설명합니다. 이전 정책은 하나의 특정 권한을 설명합니다. `Action`은 액세스 유형을 지정합니다. 정책에서 `s3:ListAllMyBuckets`는 사전 정의된 Amazon S3 작업입니다. 이 작업은 인증된 발신자가 소유한 모든 버킷의 목록을 반환하는 Amazon S3 GET Service 작업을 포함합니다. `Effect` 요소 값은 특정 권한이 허용되는지 또는 거부되는지를 결정합니다.

   d. Review policy(정책 검토)를 선택합니다. 다음 페이지에서 Name(이름) 필드에 `AllowGroupToSeeBucketListInTheConsole`을 입력하고 Create policy(정책 생성)를 선택합니다.

   Note

   요약 항목에 이 정책이 어떠한 권한도 부여하지 않는다고 알리는 메시지가 표시됩니다. 이 연습의 경우 이 메시지를 무시해도 됩니다.

3. 만든 `AllowGroupToSeeBucketListInTheConsole` 관리형 정책을 `Consultants` 그룹에 연결합니다.

   관리형 정책 연결에 대한 단계별 지침은 IAM 사용 설명서의 IAM 정책 추가 및 제거를 참조하십시오.

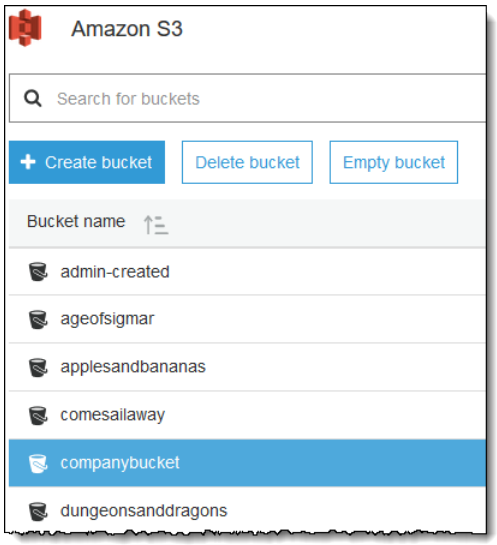
IAM 콘솔의 IAM 사용자 및 그룹에 정책 문서를 연결합니다. 두 사용자가 모두 버킷을 나열할 수 있게 만들려고 하기 때문에 정책을 그룹에 연결합니다.

4. 권한을 테스트합니다.

   a. IAM 사용자 로그인 링크(IAM 사용자를 위한 로그인 링크를 제공하려면 (p. 424) 참조)를 사용하여 IAM 사용자 자격 증명 중 하나로 콘솔에 로그인합니다.

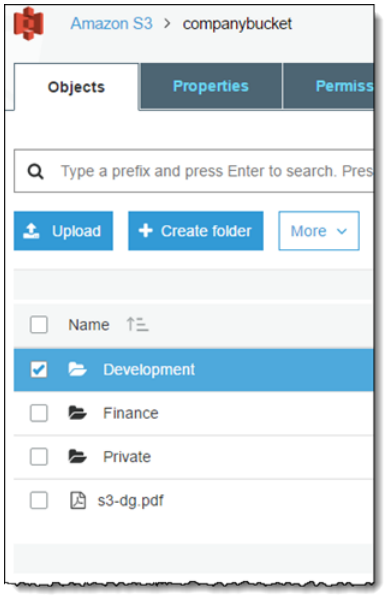
   b. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.

      이제 콘솔에 모든 버킷이 나열되지만 버킷의 객체는 아닙니다.



4.2단계: 사용자가 버킷의 루트 수준 콘텐츠를 나열할 수 있도록 허용

다음에는 `Consultants` 그룹의 모든 사용자가 루트 수준 `companybucket` 버킷 항목을 나열하도록 허용합니다. 사용자가 Amazon S3 콘솔에서 회사 버킷을 선택하면 이 사용자는 버킷에 있는 루트 수준 항목을 볼 수 있습니다.



> **Note**
>
> 이 예제에서는 예시를 위해 `companybucket`을 사용합니다. 각자 자신이 만든 버킷의 이름을 사용해야 합니다.

버킷 이름을 선택할 때 콘솔에서 Amazon S3로 보내는 요청, Amazon S3에서 반환하는 응답 및 콘솔에서 응답을 해석하는 방식을 이해하려면 약간 더 자세하게 살펴보아야 합니다.

버킷 이름을 선택하면 콘솔에서 GET Bucket (List Objects) 요청을 Amazon S3로 전송합니다. 이 요청에는 다음 파라미터가 포함되어 있습니다.

- 빈 문자열이 값으로 포함된 `prefix` 파라미터입니다.
- `/`가 값으로 포함된 `delimiter` 파라미터입니다.

다음은 예제 요청입니다.

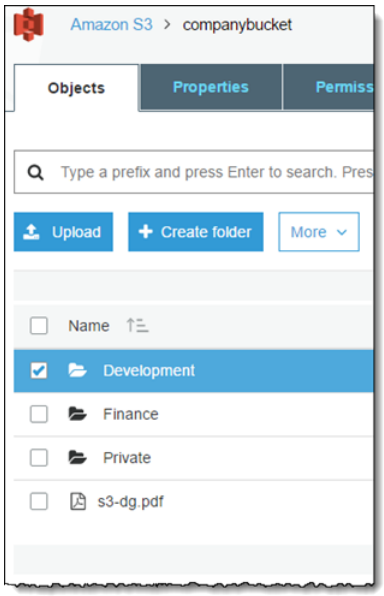
```
GET ?prefix=&delimiter=/ HTTP/1.1
Host: companybucket.s3.amazonaws.com
Date: Wed, 01 Aug  2012 12:00:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

Amazon S3에서 다음 `<ListBucketResult/>` 요소가 포함된 응답을 반환합니다.

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>companybucket</Name>
  <Prefix></Prefix>
  <Delimiter>/</Delimiter>
   ...
  <Contents>
    <Key>s3-dg.pdf</Key>
    ...
  </Contents>
  <CommonPrefixes>
    <Prefix>Development/</Prefix>
  </CommonPrefixes>
  <CommonPrefixes>
    <Prefix>Finance/</Prefix>
  </CommonPrefixes>
  <CommonPrefixes>
    <Prefix>Private/</Prefix>
  </CommonPrefixes>
</ListBucketResult>
```

키 `s3-dg.pdf` 객체에는 슬래시(`/`) 구분 기호가 포함되지 않으며, Amazon S3는 이 키를 `<Contents>` 요소에 반환합니다. 하지만 예제 버킷에 있는 다른 모든 키에는 `/` 구분 기호가 포함되어 있습니다. Amazon S3는 이러한 키를 그룹화하여 각 개별 접두사 값 `Development/`, `Finance/` 및 `Private/`에 대해 `<CommonPrefixes>` 요소를 반환합니다. 이러한 각 접두사 값은 이러한 키의 시작 부분부터 지정된 `/` 구분 기호의 첫 번째 발생 지점까지의 하위 문자열입니다.

콘솔에서 이 결과를 해석하여 루트 수준 항목을 폴더 3개와 객체 키 하나로 표시합니다.



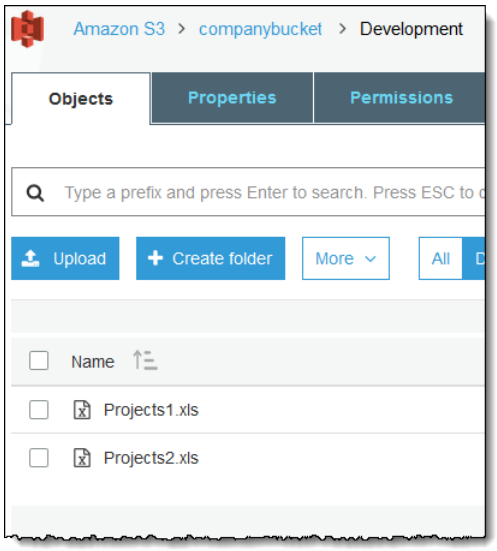
Bob 또는 Alice가 Development(개발) 폴더를 열면 콘솔에서 `prefix` 및 `delimiter` 파라미터가 다음 값으로 설정된 GET Bucket(List Objects) 요청이 Amazon S3로 전송됩니다.

- 값 `Development/`가 있는 `prefix` 파라미터입니다.
- "`/`" 값이 있는 `delimiter` 파라미터입니다.

이에 응답하여 Amazon S3에서 지정된 접두사로 시작하는 객체 키를 반환합니다.

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>companybucket</Name>
  <Prefix>Development</Prefix>
  <Delimiter>/</Delimiter>
   ...
  <Contents>
    <Key>Project1.xls</Key>
    ...
  </Contents>
  <Contents>
    <Key>Project2.xls</Key>
    ...
  </Contents>
</ListBucketResult>
```

콘솔에 객체 키가 표시됩니다.



이제 루트 수준 버킷 항목을 나열할 수 있는 사용자 권한을 부여하는 작업으로 돌아갑니다. 버킷 콘텐츠를 나열하려면 다음 정책 설명에 나와 있듯이 사용자에게 `s3:ListBucket` 작업을 호출할 수 있는 권한이 필요합니다. 사용자가 루트 수준 콘텐츠만 보도록 하기 위해 사용자가 요청에서 빈 `prefix`를 지정해야 한다는 조건을 추가합니다. 이렇게 하면 사용자는 루트 수준 폴더를 두 번 클릭할 수 없습니다. 마지막으로, 사용자 요청에 "/" 값이 있는 `delimiter` 파라미터가 포함되도록 하여 폴더 스타일 액세스를 요구하는 조건을 추가합니다.

```
{
  "Sid": "AllowRootLevelListingOfCompanyBucket",
  "Action": ["s3:ListBucket"],
  "Effect": "Allow",
  "Resource": ["arn:aws:s3:::companybucket"],
  "Condition":{
         "StringEquals":{
             "s3:prefix":[""], "s3:delimiter":["/"]
                        }
              }
}
```

Amazon S3 콘솔에서 버킷을 선택하면 콘솔은 먼저 GET Bucket location 요청을 전송하여 버킷이 배포되는 AWS 리전을 찾습니다. 그런 다음 콘솔은 버킷에 대한 리전별 엔드포인트를 사용하여 GET Bucket (List Objects) 요청을 전송합니다. 결과적으로 사용자가 콘솔을 사용하려는 경우 다음 정책 설명과 같이 `s3:GetBucketLocation` 작업에 대한 권한을 부여해야 합니다.

```
{
   "Sid": "RequiredByS3Console",
   "Action": ["s3:GetBucketLocation"],
   "Effect": "Allow",
   "Resource": ["arn:aws:s3:::*"]
}
```

사용자가 루트 수준 버킷 콘텐츠를 나열할 수 있도록 허용하려면

1. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.

   IAM 사용자의 자격 증명이 아니라 AWS 계정 자격 증명을 사용하여 콘솔에 로그인합니다.

2. `Consultants` 그룹에 연결된 기존 `AllowGroupToSeeBucketListInTheConsole` 관리형 정책을 다음 정책으로 바꿉니다. 다음 정책은 `s3:ListBucket` 작업도 허용합니다. 정책에 있는 *`companybucket`*을 버킷 이름이 있는 `Resource`로 바꿔야 합니다.

   단계별 지침은 IAM 사용 설명서의 IAM 정책 편집을 참조하십시오. 단계별 지침을 따를 때는 정책이 연결된 모든 보안 주체에 변경 사항을 적용하기 위한 단계를 따라야 합니다.

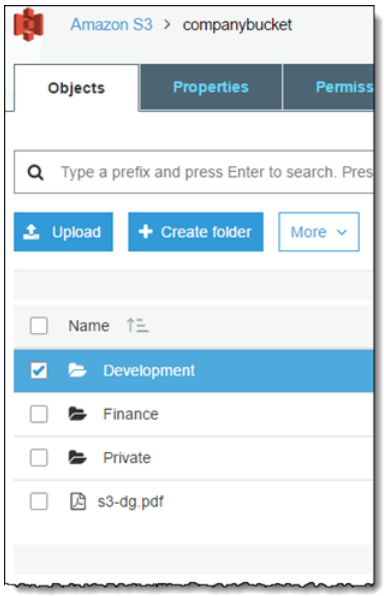
```
{
  "Version": "2012-10-17",
  "Statement": [
     {
       "Sid":
 "AllowGroupToSeeBucketListAndAlsoAllowGetBucketLocationRequiredForListBucket",
       "Action": [ "s3:ListAllMyBuckets", "s3:GetBucketLocation" ],
       "Effect": "Allow",
       "Resource": [ "arn:aws:s3:::*"  ]
     },
     {
       "Sid": "AllowRootLevelListingOfCompanyBucket",
       "Action": ["s3:ListBucket"],
       "Effect": "Allow",
       "Resource": ["arn:aws:s3:::companybucket"],
       "Condition":{
             "StringEquals":{
                    "s3:prefix":[""], "s3:delimiter":["/"]
                           }
                    }
     }
  ]
}
```

3. 업데이트된 권한을 테스트합니다.

   a. IAM 사용자 로그인 링크(IAM 사용자를 위한 로그인 링크를 제공하려면 (p. 424) 참조)를 사용하여 AWS Management 콘솔에 로그인합니다.

      https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.

   b. 만든 버킷을 선택합니다. 그러면 콘솔에 루트 수준 버킷 항목이 표시됩니다. 버킷에서 폴더 하나를 선택하면 폴더 콘텐츠를 볼 수 없습니다. 아직 해당 권한을 부여하지 않았기 때문입니다.



사용자가 Amazon S3 콘솔을 사용할 때는 이 테스트가 성공합니다. 콘솔에서 버킷을 선택하면 콘솔 구현에서 빈 문자열이 값으로 있는 `prefix` 파라미터와 "/"가 값으로 있는 `delimiter` 파라미터가 포함된 요청이 전송됩니다.

4.3단계: 그룹 정책 요약

추가한 그룹 정책의 순 효과는 IAM 사용자인 Alice와 Bob에게 다음과 같은 최소한의 권한을 부여하는 것입니다.

- 상위 계정이 소유한 모든 버킷 나열
- `companybucket` 버킷의 루트 수준 항목 확인

하지만 사용자는 여전히 많은 작업을 수행할 수 없습니다. 다음에는 아래와 같이 사용자별 권한을 부여합니다.

- Alice가 객체를 가져와서 `Development` 폴더에 넣을 수 있도록 허용합니다.
- Bob이 객체를 가져와서 `Finance` 폴더에 넣을 수 있도록 허용합니다.

사용자별 권한의 경우 그룹이 아니라 특정 사용자에게 정책을 연결합니다. 다음 단원에서는 Alice에게 `Development` 폴더에서 작업할 수 있는 권한을 부여합니다. 이 단계를 반복하여 Bob에게 `Finance` 폴더에서 작업을 수행할 수 있는 비슷한 권한을 부여할 수 있습니다.

5단계: IAM 사용자인 Alice에게 특정 권한 부여

이제 Alice가 `Development` 폴더의 콘텐츠를 보고 객체를 가져와서 해당 폴더에 넣을 수 있도록 Alice에게 추가 권한을 부여합니다.

5.1단계: IAM 사용자인 Alice에게 Development 폴더 콘텐츠 나열 권한 부여

Alice가 `Development` 폴더 콘텐츠를 나열하려면 요청에 접두사 `Development/`가 포함되는 경우 `companybucket` 버킷의 `s3:ListBucket` 작업에 대한 권한을 부여하는 정책을 Alice 사용자에게 적용해야 합니다. 이 정책을 사용자 Alice에게만 적용하려고 하므로 인라인 정책을 사용합니다. 인라인 정책에 대한 자세한 내용은 IAM 사용 설명서에서 관리형 정책과 인라인 정책 단원을 참조하십시오.

1. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.

   IAM 사용자의 자격 증명이 아니라 AWS 계정 자격 증명을 사용하여 콘솔에 로그인합니다.
2. 사용자 Alice에게 `Development` 폴더 콘텐츠를 나열할 수 있는 권한을 부여하기 위해 인라인 정책을 만듭니다.

   a. 왼쪽에 있는 탐색 창에서 Users(사용자)를 선택합니다.
   b. 사용자 이름 Alice를 선택합니다.
   c. 사용자 세부 정보 페이지에서 Permissions(권한) 탭을 선택한 다음 Add inline policy(인라인 정책 추가)를 선택합니다.
   d. JSON 탭을 선택합니다.
   e. 다음 액세스 정책을 복사하여 정책 텍스트 필드에 붙여 넣습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
      "Sid": "AllowListBucketIfSpecificPrefixIsIncludedInRequest",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
```

```
        "Resource": ["arn:aws:s3:::companybucket"],
        "Condition":{  "StringLike":{"s3:prefix":["Development/*"] }
         }
      }
   ]
}
```

f. Review policy(정책 검토)를 선택합니다. 다음 페이지에서 Name(이름) 필드에 이름을 입력하고 Create policy(정책 생성)를 선택합니다.

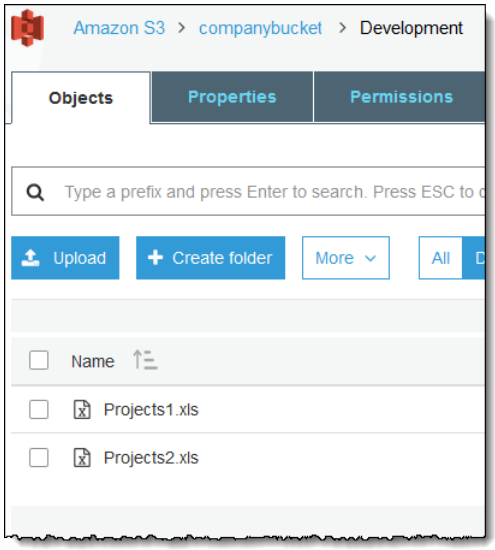
3. Alice의 권한에 대한 변경 사항을 테스트합니다.

   a. IAM 사용자 로그인 링크(IAM 사용자를 위한 로그인 링크를 제공하려면 (p. 424) 참조)를 사용하여 AWS Management 콘솔에 로그인합니다.

   b. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.

   c. Amazon S3 콘솔에서 Alice가 버킷의 `Development/` 폴더에 있는 객체의 목록을 볼 수 있는지 확인합니다.

      사용자가 `/Development` 폴더를 선택하여 그 안에 있는 객체의 목록을 보면 Amazon S3 콘솔에서 접두사 `/Development`가 있는 `ListObjects` 요청이 Amazon S3로 전송됩니다. 접두사 `Development` 및 구분 기호 `/`이 있는 객체 목록을 볼 수 있는 권한이 사용자에게 부여되기 때문에 Amazon S3에서 키 접두사 `Development/`가 있는 객체의 목록이 반환되고 콘솔에 이 목록이 표시됩니다.



5.2단계: IAM 사용자인 Alice에게 객체를 가져와 Development 폴더에 넣을 수 있는 권한 부여

Alice가 객체를 가져와서 `s3:PutObject` 폴더에 넣으려면 `Development` 및 `s3:GetObject` 작업을 호출할 수 있는 권한이 필요합니다. 다음 정책 설명은 요청에 값 `prefix`가 있는 `Development/` 파라미터가 포함되는 경우 이러한 권한을 부여합니다.

```
{
    "Sid":"AllowUserToReadWriteObjectData",
    "Action":["s3:GetObject", "s3:PutObject"],
    "Effect":"Allow",
    "Resource":["arn:aws:s3:::companybucket/Development/*"]
 }
```

1. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.

IAM 사용자의 자격 증명이 아니라 AWS 계정 자격 증명을 사용하여 콘솔에 로그인합니다.

2. 이전 단계에서 만든 인라인 정책을 편집합니다.

   a. 왼쪽에 있는 탐색 창에서 Users(사용자)를 선택합니다.
   b. 사용자 이름 Alice를 선택합니다.
   c. 사용자 세부 정보 페이지에서 Permissions(권한) 탭을 선택하고 Inline Policies(인라인 정책) 섹션을 확장합니다.
   d. 이전 단계에서 만든 정책의 이름 옆에 있는 Edit Policy(정책 편집)를 선택합니다.
   e. 다음 정책을 복사하여 정책 텍스트 필드에 붙여 넣어 기존 정책을 바꿉니다.

```
{
     "Version": "2012-10-17",
     "Statement":[
      {
         "Sid":"AllowListBucketIfSpecificPrefixIsIncludedInRequest",
         "Action":["s3:ListBucket"],
         "Effect":"Allow",
         "Resource":["arn:aws:s3:::companybucket"],
         "Condition":{
            "StringLike":{"s3:prefix":["Development/*"]
            }
         }
      },
      {
        "Sid":"AllowUserToReadWriteObjectDataInDevelopmentFolder",
        "Action":["s3:GetObject", "s3:PutObject"],
        "Effect":"Allow",
        "Resource":["arn:aws:s3:::companybucket/Development/*"]
      }
   ]
}
```

3. 업데이트된 정책을 테스트합니다.

   a. IAM 사용자 로그인 링크(IAM 사용자를 위한 로그인 링크를 제공하려면 (p. 424) 참조)를 사용하여 AWS Management 콘솔에 로그인합니다.
   b. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
   c. Amazon S3 콘솔에서 Alice가 이제 `Development` 폴더에서 객체를 추가하고 객체를 다운로드할 수 있는지 확인합니다.

5.3단계: 버킷의 다른 모든 폴더에 대한 IAM 사용자 Alice의 권한 명시적으로 거부

이제 사용자 Alice가 `companybucket` 버킷의 루트 수준 콘텐츠를 나열할 수 있습니다. Alice는 객체를 가져와 `Development` 폴더에 넣을 수도 있습니다. 액세스 권한을 강화하고 싶은 경우 버킷의 다른 모든 폴더에 대한 Alice의 액세스 권한을 명시적으로 거부할 수 있습니다. Alice에게 버킷의 다른 모든 폴더에 대한 액세스 권한을 부여하는 다른 정책(버킷 정책 또는 ACL)이 있는 경우 이 명시적 거부로 인해 해당 권한이 재정의됩니다.

Alice가 Amazon S3로 보내는 모든 요청에 `prefix` 파라미터(값: `Development/*` 또는 빈 문자열)를 포함할 것을 요구하는 다음 설명을 사용자 Alice의 정책에 추가할 수 있습니다.

```
{
   "Sid": "ExplicitlyDenyAnyRequestsForAllOtherFoldersExceptDevelopment",
   "Action": ["s3:ListBucket"],
   "Effect": "Deny",
   "Resource": ["arn:aws:s3:::companybucket"],
   "Condition":{  "StringNotLike": {"s3:prefix":["Development/*",""] },
                  "Null"         : {"s3:prefix":false }
```

```
    }
}
```

Condition 블록에는 두 개의 조건식이 있습니다. 이러한 조건식의 결과는 논리적 AND를 사용하여 결합됩니다. 두 조건이 모두 true인 경우 결합된 조건의 결과가 true입니다. 이 정책에서 Effect는 Deny이기 때문에 Condition이 ture로 평가되면 사용자는 지정된 Action을 수행할 수 없습니다.

- Null 조건식은 Alice가 보내는 요청에 prefix 파라미터가 포함되도록 합니다.

  prefix 파라미터는 폴더와 같은 액세스를 요구합니다. prefix 파라미터 없이 요청을 보낼 경우 Amazon S3에서 모든 객체 키를 반환합니다.

  요청에 널 값이 있는 prefix 파라미터가 포함되는 경우 식은 true로 평가되며 따라서 전체 Condition이 true로 평가됩니다. prefix 파라미터의 값으로 빈 문자열을 허용해야 합니다. 이전 논의에서 null 문자열을 허용하면 Alice가 앞의 논의에서 콘솔이 그랬던 것처럼 루트 수준 버킷 항목을 검색할 수 있습니다. 자세한 내용은 4.2단계: 사용자가 버킷의 루트 수준 콘텐츠를 나열할 수 있도록 허용 (p. 427) 단원을 참조하십시오.
- StringNotLike 조건식은 prefix 파라미터의 값이 지정되고 Development/*가 아닌 경우 요청이 실패하도록 합니다.

이전 단원의 단계를 따라 사용자 Alice에 대해 만든 인라인 정책을 다시 업데이트합니다.

다음 정책을 복사하여 정책 텍스트 필드에 붙여 넣어 기존 정책을 바꿉니다.

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Sid":"AllowListBucketIfSpecificPrefixIsIncludedInRequest",
         "Action":["s3:ListBucket"],
         "Effect":"Allow",
         "Resource":["arn:aws:s3:::companybucket"],
         "Condition":{
            "StringLike":{"s3:prefix":["Development/*"]
            }
         }
      },
      {
        "Sid":"AllowUserToReadWriteObjectDataInDevelopmentFolder",
        "Action":["s3:GetObject", "s3:PutObject"],
        "Effect":"Allow",
        "Resource":["arn:aws:s3:::companybucket/Development/*"]
      },
      {
         "Sid": "ExplicitlyDenyAnyRequestsForAllOtherFoldersExceptDevelopment",
         "Action": ["s3:ListBucket"],
         "Effect": "Deny",
         "Resource": ["arn:aws:s3:::companybucket"],
         "Condition":{  "StringNotLike": {"s3:prefix":["Development/*",""] },
                        "Null"         : {"s3:prefix":false }
          }
      }
   ]
}
```

6단계: IAM 사용자인 Bob에게 특정 권한 부여

이제 Bob에게 Finance 폴더에 대한 권한을 부여하려고 합니다. 앞에서 Alice에게 권한을 부여하는 데 사용한 단계를 따르지만, Development 폴더를 Finance 폴더로 바꿉니다. 단계별 지침은 5단계: IAM 사용자인 Alice에게 특정 권한 부여 (p. 432)를 참조하십시오.

7단계: Private 폴더에 보안 지정

이 예제에서는 사용자가 두 명뿐입니다. 그룹 수준에서 필요한 모든 최소한의 권한을 부여하고 필요할 때만 사용자 수준 권한을 개별 사용자 수준에 있는 권한에 부여했습니다. 이 접근 방식을 취하면 권한 관리 작업을 최소화하는 데 도움이 됩니다. 사용자 수가 증가하면서 권한 관리 작업이 부담이 될 수 있습니다. 예를 들어, 이 예제의 어떤 사용자도 `Private` 폴더의 콘텐츠에 액세스하지 못하게 하려고 합니다. 이 폴더에 실수로 사용자 권한을 부여하지 않도록 하려면 어떻게 해야 할까요? 해당 폴더에 대한 액세스 권한을 명시적으로 거부하는 정책을 추가하면 됩니다. 명시적 거부로 인해 다른 모든 권한이 재정의됩니다.

`Private` 폴더를 프라이빗으로 유지하려면 다음 두 개의 거부 문을 그룹 정책에 추가할 수 있습니다.

- `Private` 폴더의 리소스(`companybucket/Private/*`)에 대한 모든 작업을 명시적으로 거부하려면 다음 문을 추가합니다.

```
{
  "Sid": "ExplictDenyAccessToPrivateFolderToEveryoneInTheGroup",
  "Action": ["s3:*"],
  "Effect": "Deny",
  "Resource":["arn:aws:s3:::companybucket/Private/*"]
}
```

- 요청에서 `Private/` 접두사를 지정하면 객체 나열 작업에 대한 권한도 거부하게 됩니다. 콘솔에서 Bob 또는 Alice가 `Private` 폴더를 열면 이 정책으로 인해 Amazon S3에서 오류 응답을 반환합니다.

```
{
  "Sid": "DenyListBucketOnPrivateFolder",
  "Action": ["s3:ListBucket"],
  "Effect": "Deny",
  "Resource": ["arn:aws:s3:::*"],
  "Condition":{
      "StringLike":{"s3:prefix":["Private/"]}
   }
}
```

`Consultants` 그룹 정책을 이전의 거부 문을 포함하는 업데이트된 정책으로 대체합니다. 업데이트된 정책이 적용된 후에는 그룹에 있는 어떤 사용자도 버킷의 `Private` 폴더에 액세스할 수 없습니다.

1. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.

   IAM 사용자의 자격 증명이 아니라 AWS 계정 자격 증명을 사용하여 콘솔에 로그인합니다.

2. `Consultants` 그룹에 연결된 기존 `AllowGroupToSeeBucketListInTheConsole` 관리형 정책을 다음 정책으로 바꿉니다. 정책에서 `companybucket`을 버킷의 이름으로 대체해야 합니다.

   지침은 IAM 사용 설명서의 고객 관리형 정책 편집을 참조하십시오. 지침을 따를 때 정책이 연결된 모든 보안 주체에 변경 사항을 적용하는 작업에 대한 지시 사항을 반드시 따르십시오.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid":
 "AllowGroupToSeeBucketListAndAlsoAllowGetBucketLocationRequiredForListBucket",
      "Action": ["s3:ListAllMyBuckets", "s3:GetBucketLocation"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::*"]
    },
    {
      "Sid": "AllowRootLevelListingOfCompanyBucket",
```

```
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::companybucket"],
      "Condition":{
          "StringEquals":{"s3:prefix":[""]}
       }
    },
    {
      "Sid": "RequireFolderStyleList",
      "Action": ["s3:ListBucket"],
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::*"],
      "Condition":{
          "StringNotEquals":{"s3:delimiter":"/"}
       }
     },
    {
      "Sid": "ExplictDenyAccessToPrivateFolderToEveryoneInTheGroup",
      "Action": ["s3:*"],
      "Effect": "Deny",
      "Resource":["arn:aws:s3:::companybucket/Private/*"]
    },
    {
      "Sid": "DenyListBucketOnPrivateFolder",
      "Action": ["s3:ListBucket"],
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::*"],
      "Condition":{
          "StringLike":{"s3:prefix":["Private/"]}
       }
    }
  ]
}
```

8단계: 정리

정리하려면 IAM 콘솔을 열고 사용자 Alice와 Bob을 제거합니다. 단계별 지침은 IAM 사용 설명서의 IAM 사용자 삭제를 참조하십시오.

스토리지에 대한 추가 요금이 부과되지 않도록 하려면 이 연습을 위해 만든 객체와 버킷도 삭제해야 합니다.

관련 리소스

- IAM 사용 설명서의 IAM 정책 관리

ACL을 사용한 액세스 관리

주제

- ACL(액세스 통제 목록) 개요 (p. 438)
- ACL 관리 (p. 444)

ACL(액세스 통제 목록)은 리소스 기반 액세스 정책 옵션 중 하나로(액세스 관리 개요 (p. 299) 참조), 해당 옵션을 사용해 버킷과 객체에 대한 액세스를 관리할 수 있습니다. ACL로 다른 AWS 계정에 기본적인 읽기/쓰기 권한을 부여할 수 있습니다. ACL로 하는 권한 관리에는 한계가 있습니다. 예를 들어, 다른 AWS 계정에만 권한을 부여할 수 있고 본인 계정의 사용자에게는 권한을 부여할 수 없습니다. 조건부 권한을 부여하거나 명시적으로 권한을 거부할 수 없습니다. ACL은 특정한 시나리오에 적합합니다. 예를 들어, 버킷 소유자가 다른 AWS 계정에 객체 업로드를 허용한다면 이러한 객체에 대한 권한은 해당 객체를 소유한 AWS 계정이 객체 ACL을 사용할 경우에만 관리할 수 있습니다.

다음 주제에서는 Amazon S3 리소스 액세스 관리의 기본 개념과 사용 옵션에 대해 설명하고, 어떤 액세스 정책 옵션을 언제 사용할지에 대한 가이드라인을 제공합니다.

- Amazon S3 리소스에 대한 액세스 관리 소개 (p. 299)
- 제공되는 액세스 정책 옵션 사용 지침 (p. 311)

ACL(액세스 통제 목록) 개요

주제

- 피부여자란? (p. 438)
- 부여할 수 있는 권한 (p. 440)
- 샘플 ACL (p. 442)
- 미리 제공된 ACL (p. 443)
- ACL 지정 방법 (p. 443)

Amazon S3 ACL(액세스 제어 목록)로 버킷과 객체에 대한 액세스를 관리합니다. 각 버킷과 객체마다 하위 리소스로서 연결되어 있는 ACL이 있습니다. ACL은 액세스를 허용할 AWS 계정이나 그룹과 액세스 유형을 정의합니다. 리소스에 대한 요청을 수신하면, Amazon S3는 해당 ACL을 확인해 요청자가 필요한 액세스 권한을 보유하고 있는지 판단합니다.

버킷이나 객체를 생성하면 Amazon S3는 리소스에 대한 모든 권한을 리소스 소유자에게 부여하는 기본 ACL을 생성합니다. 이 정보는 다음 샘플 버킷 ACL(기본 객체 ACL도 동일한 구조)에 표시됩니다.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
    <ID>*** Owner-Canonical-User-ID ***</ID>
    <DisplayName>owner-display-name</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:type="Canonical User">
        <ID>*** Owner-Canonical-User-ID ***</ID>
        <DisplayName>display-name</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

샘플 ACL에는 AWS 계정의 정식 사용자 ID로 소유자를 식별하는 `Owner` 요소가 포함되어 있습니다. 정식 사용자 ID를 찾는 방법에 대한 자세한 내용은 AWS 계정 정식 사용자 ID 찾기 (p. 439)을 참조하십시오. `Grant` 요소는 피부여자(AWS 계정 또는 미리 정의된 그룹)와 부여된 권한을 식별합니다. 이 기본 ACL에는 소유자에 대해 한 개의 `Grant` 요소가 있습니다. 각각 피부여자와 권한을 식별하는 `Grant` 요소를 추가해 권한을 부여합니다.

> **Note**
>
> ACL은 최고 100개의 권한을 부여할 수 있습니다.

피부여자란?

피부여자는 AWS 계정 또는 미리 정의된 Amazon S3 그룹 중 하나가 될 수 있습니다. AWS 계정의 이메일 주소나 정식 사용자 ID로 권한을 부여합니다. 단, 권한 요청에 이메일 주소를 기재한 경우, Amazon S3에서 해

당 계정의 정식 사용자 ID를 찾아서 ACL에 추가합니다. 그 결과 ACL에는 AWS 계정의 이메일 주소가 아니라 AWS 계정의 정식 사용자 ID가 항상 포함됩니다.

액세스 권한을 부여할 때 유형=값 쌍으로 각 피부여자를 지정합니다. 여기서 유형은 다음 중 하나입니다.

- `id` — 지정된 값이 AWS 계정의 정식 사용자 ID인 경우
- `uri` — 미리 정의된 그룹에 권한을 부여하는 경우
- `emailAddress` — 지정된 값이 AWS 계정의 이메일 주소인 경우

> **Important**
>
> 피부여자를 지정하기 위해 이메일 주소를 사용하는 것은 다음 AWS 리전에서만 지원됩니다.
>
> - 미국 동부(버지니아 북부)
> - 미국 서부(캘리포니아 북부 지역)
> - 미국 서부(오레곤)
> - 아시아 태평양(싱가포르)
> - 아시아 태평양(시드니)
> - 아시아 태평양(도쿄)
> - 유럽(아일랜드)
> - 남아메리카(상파울루)
>
> 모든 Amazon S3 지원 리전 및 엔드포인트 목록은 AWS 일반 참조의 리전 및 엔드포인트 단원을 참조하십시오.

Example 예: 이메일 주소

예를 들어 다음 `x-amz-grant-read` 헤더는 이메일 주소로 식별된 AWS 계정에 객체 데이터와 메타데이터를 읽을 수 있는 권한을 부여합니다.

```
x-amz-grant-read: emailAddress="xyz@amazon.com", emailAddress="abc@amazon.com"
```

> **Warning**
>
> 다른 AWS 계정에 본인의 리소스 액세스를 허용하는 경우, AWS 계정이 자신의 계정에 속한 사용자에게 그 권한을 위임할 수도 있다는 사실을 염두에 두어야 합니다. 이것을 교차 계정 액세스라고 합니다. 교차 계정 액세스를 사용하는 자세한 방법은 IAM 사용 설명서의 역할을 만들어 IAM 사용자에게 권한 위임 단원을 참조하십시오.

AWS 계정 정식 사용자 ID 찾기

정식 사용자 ID는 AWS 계정과 연결되어 있습니다. 이 ID는 `79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be`와 같은 긴 문자열입니다. 계정의 정식 사용자 ID를 찾는 방법에 대한 정보는 계정 정식 사용자 ID 찾기를 참조하십시오.

AWS 계정에서 액세스 권한을 보유한 버킷이나 객체의 ACL을 확인해 AWS 계정의 정식 사용자 ID를 찾을 수도 있습니다. 권한 요청에 따라 개별 AWS 계정에 권한이 부여될 때, 권한 항목이 AWS 계정 정식 사용자 ID와 함께 ACL에 추가됩니다.

> **Note**
>
> 버킷을 공개하면(권장하지 않음) 인증되지 않은 모든 사용자들이 그 버킷에 객체를 업로드할 수 있습니다. 이러한 익명의 사용자에게는 AWS 계정이 없습니다. 익명의 사용자가 귀하의 버킷에 객체를 업로드하면 Amazon S3는 ACL의 객체 소유자로서 특별한 정식 사용자

ID(`65a011a29cdf8ec533ec3d1ccaae921c`)를 추가합니다. 자세한 내용은 Amazon S3 버킷 및 객체 소유권 (p. 300) 단원을 참조하십시오.

Amazon S3의 미리 정의된 그룹

Amazon S3에는 여러 개의 미리 정의된 그룹이 있습니다. 그룹에 계정 액세스를 허용하려면 정식 사용자 ID 대신 URI 하나를 지정합니다. 미리 정의된 그룹은 다음과 같이 제공됩니다.

- 인증된 사용자 그룹(Authenticated Users group) – `http://acs.amazonaws.com/groups/global/AuthenticatedUsers`로 표시.

  이 그룹은 모든 AWS 계정을 나타냅니다. 이 그룹의 액세스 권한은 모든 AWS 계정에 리소스 액세스를 허용합니다. 단, 모든 요청에는 서명을(인증) 해야 합니다.

  > **Warning**
  >
  > 인증된 사용자 그룹(Authenticated Users group)에 액세스 권한을 부여하면 AWS 인증을 받은 전 세계 사용자가 리소스에 액세스할 수 있습니다.

- 전체 사용자 그룹(All Users group) – `http://acs.amazonaws.com/groups/global/AllUsers`로 표시.

  이 그룹의 액세스 권한으로 전 세계 누구나 리소스에 액세스할 수 있습니다. 요청에 서명을 할 수도(인증) 있고 하지 않을(익명) 수도 있습니다. 서명되지 않은 요청은 요청에 인증 헤더를 생략합니다.

  > **Warning**
  >
  > 전체 사용자 그룹(All Users group) `WRITE`, `WRITE_ACP` 또는 `FULL_CONTROL` 권한을 부여하지 않는 것이 좋습니다. 예를 들어, `WRITE` 권한이 있는 누구나 비용이 청구되는 버킷에 객체를 저장할 수 있습니다. 또한 계속 유지하고 싶은 객체를 다른 사용자가 삭제하도록 허용할 수도 있습니다. 이러한 권한에 대한 자세한 내용은 다음 부여할 수 있는 권한 (p. 440) 단원을 참조하십시오.

- 로그 전달 그룹(Log Delivery group) – `http://acs.amazonaws.com/groups/s3/LogDelivery`로 표시.

  버킷 내 WRITE(쓰기) 권한으로 이 그룹은 버킷에 서버 액세스 로그를 쓸 수 있습니다(Amazon S3 서버 액세스 로깅 (p. 705) 참조).

> **Note**
>
> ACL을 사용하는 경우, 피부여자는 AWS 계정이나 미리 정의된 Amazon S3 그룹 중 하나가 됩니다. 하지만, IAM 사용자는 피부여자가 될 수 없습니다. AWS 사용자나 IAM 내 권한에 대한 자세한 내용은 Using AWS Identity and Access Management을 참조하십시오.

부여할 수 있는 권한

다음 표에는 ACL에서 Amazon S3가 지원하는 권한 목록이 나와 있습니다. ACL 권한 집합은 객체 ACL 및 버킷 ACL과 동일합니다. 단, 상황에 따라 이 ACL 권한(버킷 ACL이나 객체 ACL) 은 특정 버킷이나 객체 작업에 대한 권한을 부여하기도 합니다. 표에는 권한 목록이 나열되어 있으며 객체 및 버킷에 따른 권한에 대한 설명이 수록되어 있습니다.

권한	버킷에 대한 권한을 부여하는 경우	객체에 대한 권한을 부여하는 경우
`READ`	피부여자에게 버킷의 객체 목록 생성을 허용합니다	피부여자에게 객체 데이터와 그 메타데이터를 읽도록 허용합니다
`WRITE`	피부여자에게 버킷에 속한 객체의 생성, 덮어쓰기, 삭제를 허용합니다	해당 사항 없음
`READ_ACP`	피부여자에게 버킷 ACL 읽기를 허용합니다	피부여자에게 객체 ACL 읽기를 허용합니다

권한	버킷에 대한 권한을 부여하는 경우	객체에 대한 권한을 부여하는 경우
`WRITE_ACP`	피부여자에게 해당 버킷에 대한 ACL 쓰기를 허용합니다	피부여자에게 해당 객체에 대한 ACL 쓰기를 허용합니다
`FULL_CONTROL`	피부여자에게 버킷에 대한 WRITE, READ, READ_ACP, WRITE_ACP 권한을 허용합니다.	피부여자에게 객체에 대한 읽기, 쓰기, READ_ACP, WRITE_ACP 권한을 허용합니다

> **Warning**
>
> S3 버킷과 객체에 액세스 권한을 부여할 때 주의하십시오. 예를 들어, 버킷에 `WRITE` 권한을 부여하면, 피부여자에게 버킷에 속한 객체의 생성, 덮어쓰기, 삭제를 허용합니다. 권한을 부여하기 전에 이 ACL(액세스 통제 목록) 개요 (p. 438) 단원 전체를 잘 읽으시기 바랍니다.

ACL 권한과 액세스 정책 권한의 매핑

앞의 표에서와 같이, ACL은 액세스 정책에서 설정할 수 있는 권한 수와 비교해 유한한 권한 집합만 허용합니다(Amazon S3 작업 (p. 342) 참조). 이러한 권한은 각각 한 개 이상의 Amazon S3 작업을 허용합니다.

다음 표는 각 ACL 권한이 어떻게 해당 액세스 정책 권한에 매핑되는지 보여줍니다. 보시다시피 액세스 정책은 ACL보다 많은 권한을 허용합니다. 주로 ACL을 사용하여 파일 시스템 권한과 유사한 기본 읽기/쓰기 권한을 부여합니다. ACL을 사용하는 경우에 대한 자세한 내용은 제공되는 액세스 정책 옵션 사용 지침 (p. 311)을 참조하십시오.

ACL 권한	버킷에 대한 ACL 권한 부여 시 해당 액세스 정책 권한	객체에 대한 ACL 권한 부여 시 해당 액세스 정책 권한
`READ`	`s3:ListBucket`, `s3:ListBucketVersions`, 및 `s3:ListBucketMultipartUploads`	`s3:GetObject`, `s3:GetObjectVersion`, 및 `s3:GetObjectTorrent`
`WRITE`	`s3:PutObject` 및 `s3:DeleteObject`. 또한, 피부여자가 버킷 소유자인 경우 버킷 ACL의 `WRITE` 권한을 부여해 해당 버킷의 어떤 버전에서든 `s3:DeleteObjectVersion` 작업의 수행할 수 있습니다.	해당 사항 없음
`READ_ACP`	`s3:GetBucketAcl`	`s3:GetObjectAcl` 및 `s3:GetObjectVersionAcl`
`WRITE_ACP`	`s3:PutBucketAcl`	`s3:PutObjectAcl` 및 `s3:PutObjectVersionAcl`
`FULL_CONTROL`	`READ`, `WRITE`, `READ_ACP`, `WRITE_ACP` ACL 권한을 부여하는 것과 동일합니다. 따라서, 이 ACL 권한은 해당 액세스 정책 권한의 조합과 매핑됩니다.	`READ`, `READ_ACP`, `WRITE_ACP` ACL 권한을 부여하는 것과 동일합니다. 따라서, 이 ACL 권한은 해당 액세스 정책 권한의 조합과 매핑됩니다.

조건 키

액세스 정책 권한을 부여할 때 조건 키를 사용하여 버킷 정책을 통해 객체의 ACL 값을 제한할 수 있습니다. 아래의 컨텍스트 키는 ACL에 해당합니다. 이러한 컨텍스트 키를 사용하여 요청에서 특정 ACL을 사용하도록 지정할 수 있습니다.

- `s3:x-amz-grant-read` - 읽기 액세스가 필요합니다.
- `s3:x-amz-grant-write` - 쓰기 액세스가 필요합니다.
- `s3:x-amz-grant-read-acp` - 버킷 ACL에 대한 읽기 권한이 필요합니다.
- `s3:x-amz-grant-write-acp` - 버킷 ACL에 대한 쓰기 권한이 필요합니다.
- `s3:x-amz-grant-full-control` - 완전한 제어가 필요합니다.
- `s3:x-amz-acl` - 미리 제공된 ACL (p. 443)이 필요합니다.

ACL 관련 헤더를 포함하는 정책 예는 예제 1: 버킷 소유자가 전체 권한을 가져오도록 요구하는 조건으로 s3:PutObject 권한 부여 (p. 347)을 참조하십시오. Amazon S3- 전용 조건 키의 전체 목록은 Amazon S3에 사용되는 작업, 리소스 및 조건 키 (p. 356) 단원을 참조하십시오.

샘플 ACL

다음 샘플 버킷 ACL은 리소스 소유자와 권한 부여 집합을 식별합니다. 해당 형식은 Amazon S3 REST API에서 ACL의 XML 표시입니다. 버킷 소유자는 리소스의 `FULL_CONTROL`을 보유합니다. 또한, ACL은 이전 단원에서 다룬 2가지 사전 정의된 Amazon S3 그룹과 정식 사용자 ID로 식별되는 2가지 AWS 계정에 리소스에 대한 권한을 부여하는 방법을 보여줍니다.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
    <ID>Owner-canonical-user-ID</ID>
    <DisplayName>display-name</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="CanonicalUser">
        <ID>Owner-canonical-user-ID</ID>
        <DisplayName>display-name</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>

    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="CanonicalUser">
        <ID>user1-canonical-user-ID</ID>
        <DisplayName>display-name</DisplayName>
      </Grantee>
      <Permission>WRITE</Permission>
    </Grant>

    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="CanonicalUser">
        <ID>user2-canonical-user-ID</ID>
        <DisplayName>display-name</DisplayName>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>

    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
```

```
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
      </Grantee>
      <Permission>WRITE</Permission>
    </Grant>

  </AccessControlList>
</AccessControlPolicy>
```

미리 제공된 ACL

Amazon S3는 미리 제공된 ACL이라고 하는 사전 정의된 권한 부여 집합을 지원합니다. 미리 제공된 각 ACL에는 사전정의된 피부여자와 권한 집합이 있습니다. 다음 표에는 미리 제공된 ACL 집합과 그에 연결된 사전 정의된 권한 부여 목록이 있습니다.

미리 제공된 ACL	적용 대상	ACL에 추가된 권한
`private`	버킷과 객체	소유자는 `FULL_CONTROL`을 가집니다. 다른 누구도 액세스 권한이 없습니다(기본).
`public-read`	버킷과 객체	소유자는 `FULL_CONTROL`을 가집니다. `AllUsers` 그룹은(피부여자란? (p. 438) 참조) `READ` 액세스 권한을 가집니다.
`public-read-write`	버킷과 객체	소유자는 `FULL_CONTROL`을 가집니다. `AllUsers` 그룹은 `READ`와 `WRITE`에 대한 액세스 권한을 가집니다. 버킷에 이를 허용하는 것은 일반적으로 권장하지 않습니다.
`aws-exec-read`	버킷과 객체	소유자는 `FULL_CONTROL`을 얻습니다. Amazon EC2는 Amazon 머신 이미지(AMI) 번들을 `GET`하기 위해 Amazon S3에서 `READ` 액세스 권한을 얻습니다.
`authenticated-read`	버킷과 객체	소유자는 `FULL_CONTROL`을 가집니다. `AuthenticatedUsers` 그룹은 `READ` 액세스 권한을 가집니다.
`bucket-owner-read`	객체	객체 소유자는 `FULL_CONTROL`을 가집니다. 버킷 소유자는 `READ` 액세스 권한을 가집니다. 버킷 생성 시 미리 제공된 이 ACL을 지정하면 Amazon S3는 이를 무시합니다.
`bucket-owner-full-control`	객체	객체 소유자와 버킷 소유자 모두 객체에 대해 `FULL_CONTROL`을 가집니다. 버킷 생성 시 미리 제공된 이 ACL을 지정하면 Amazon S3는 이를 무시합니다.
`log-delivery-write`	버킷	`LogDelivery` 그룹은 버킷에 대해 `WRITE`과 `READ_ACP` 권한을 가집니다. 로그에 대한 자세한 내용은 Amazon S3 서버 액세스 로깅 (p. 705) 단원을 참조하십시오.

> **Note**
>
> 이러한 미리 제공된 ACL은 요청에 하나만 지정할 수 있습니다.

`x-amz-acl` 요청 헤더로 요청에 미리 제공된 ACL을 지정합니다. Amazon S3에서 미리 제공된 ACL이 포함된 요청을 수신하면, 사전정의된 권한을 리소스 ACL에 추가합니다.

ACL 지정 방법

ACL 지정에 대한 자세한 내용은 다음 항목을 참조하십시오.

- AWS Management 콘솔을 사용한 ACL 관리 (p. 444)
- AWS SDK for Java을 사용한 ACL 관리 (p. 444)
- .NET용 AWS SDK을 사용한 ACL 관리 (p. 446)
- REST API를 사용해 ACL 관리 (p. 449)

ACL 관리

주제

- AWS Management 콘솔을 사용한 ACL 관리 (p. 444)
- AWS SDK for Java을 사용한 ACL 관리 (p. 444)
- .NET용 AWS SDK을 사용한 ACL 관리 (p. 446)
- REST API를 사용해 ACL 관리 (p. 449)

리소스 ACL에 권한을 추가할 수 있는 방법에는 여러 가지가 있습니다. 코드를 작성하지 않고도 권한을 관리하는 UI를 제공하는 AWS Management 콘솔을 사용할 수 있습니다. REST API나 AWS SDK 중 하나를 사용할 수 있습니다. 이 라이브러리는 프로그래밍 작업을 훨씬 간소화해 줍니다.

AWS Management 콘솔을 사용한 ACL 관리

AWS Management 콘솔은 버킷과 객체에 ACL 기반 액세스 권한을 부여할 수 있는 UI를 제공합니다. 콘솔에서 ACL 기반 액세스 권한을 설정하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 ACL 버킷 권한을 설정하려면 어떻게 해야 합니까? 및 객체에 대한 권한은 어떻게 설정합니까?를 참조하십시오.

AWS SDK for Java을 사용한 ACL 관리

이 섹션에서는 버킷 및 객체에 대한 ACL(액세스 제어 목록) 권한 부여를 구성하는 방법을 보여주는 예제를 제공합니다. 첫 번째 예제에서는 준비된 ACL을 사용하여 버킷을 생성하고(미리 제공된 ACL (p. 443) 단원 참조), 사용자 지정 권한 부여 목록을 생성한 다음 사용자 지정 권한 부여가 포함된 ACL로 준비된 ACL을 대체합니다. 두 번째 예제에서는 `AccessControlList.grantPermission()` 메서드를 사용하여 ACL을 수정하는 방법을 보여줍니다.

ACL 권한 부여 설정

Example

이 예제에서는 버킷을 생성합니다. 요청 시 이 예제는 로그 전달 그룹 권한을 부여하는 준비된 ACL을 지정하여 버킷에 로그를 기록합니다.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.io.IOException;
import java.util.ArrayList;

public class CreateBucketWithACL {

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

```
        String userEmailForReadPermission = "*** user@example.com ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .build();

            // Create a bucket with a canned ACL. This ACL will be replaced by the
setBucketAcl()
            // calls below. It is included here for demonstration purposes.
            CreateBucketRequest createBucketRequest = new CreateBucketRequest(bucketName,
clientRegion.getName())
                    .withCannedAcl(CannedAccessControlList.LogDeliveryWrite);
            s3Client.createBucket(createBucketRequest);

            // Create a collection of grants to add to the bucket.
            ArrayList<Grant> grantCollection = new ArrayList<Grant>();

            // Grant the account owner full control.
            Grant grant1 = new Grant(new
CanonicalGrantee(s3Client.getS3AccountOwner().getId()), Permission.FullControl);
            grantCollection.add(grant1);

            // Grant the LogDelivery group permission to write to the bucket.
            Grant grant2 = new Grant(GroupGrantee.LogDelivery, Permission.Write);
            grantCollection.add(grant2);

            // Save grants by replacing all current ACL grants with the two we just
created.
            AccessControlList bucketAcl = new AccessControlList();
            bucketAcl.grantAllPermissions(grantCollection.toArray(new Grant[0]));
            s3Client.setBucketAcl(bucketName, bucketAcl);

            // Retrieve the bucket's ACL, add another grant, and then save the new ACL.
            AccessControlList newBucketAcl = s3Client.getBucketAcl(bucketName);
            Grant grant3 = new Grant(new EmailAddressGrantee(userEmailForReadPermission),
Permission.Read);
            newBucketAcl.grantAllPermissions(grant3);
            s3Client.setBucketAcl(bucketName, newBucketAcl);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and 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();
        }
    }
}
```

기존 객체에 대한 ACL 권한 부여 구성

Example

이 예제에서는 객체에 대한 ACL을 업데이트합니다. 이 예에서는 다음과 같은 작업을 수행합니다.

- 객체의 ACL 가져오기
- 기존 권한을 모두 제거하여 ACL 지우기
- 두 가지 권한 추가: 소유자의 경우 모든 액세스 권한 및 이메일 주소로 식별되는 사용자의 경우 WRITE_ACP(부여할 수 있는 권한 (p. 440) 단원 참조)
- 객체에 대한 ACL 저장

```
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;
import com.amazonaws.services.s3.model.AccessControlList;
import com.amazonaws.services.s3.model.CanonicalGrantee;
import com.amazonaws.services.s3.model.EmailAddressGrantee;
import com.amazonaws.services.s3.model.Permission;

import java.io.IOException;

public class ModifyACLExistingObject {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";
        String emailGrantee = "*** user@example.com ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Get the existing object ACL that we want to modify.
            AccessControlList acl = s3Client.getObjectAcl(bucketName, keyName);

            // Clear the existing list of grants.
            acl.getGrantsAsList().clear();

            // Grant a sample set of permissions, using the existing ACL owner for Full
 Control permissions.
            acl.grantPermission(new CanonicalGrantee(acl.getOwner().getId()),
 Permission.FullControl);
            acl.grantPermission(new EmailAddressGrantee(emailGrantee),
 Permission.WriteAcp);

            // Save the modified ACL back to the object.
            s3Client.setObjectAcl(bucketName, keyName, acl);
        } 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();
        }
    }
}
```

.NET용 AWS SDK을 사용한 ACL 관리

이 단원에서는 Amazon S3 버킷 및 객체에 대한 ACL 권한 부여를 구성하는 예제를 제공합니다.

예제 1: 버킷 생성 및 준비된 ACL을 사용하여 권한 설정

이 C# 예제에서는 버킷을 생성합니다. 요청 시 이 코드는 로그 전달 그룹 권한을 부여하는 준비된 ACL을 지정하여 버킷에 로그를 기록합니다.

실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

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

namespace Amazon.DocSamples.S3
{
    class ManagingBucketACLTest
    {
        private const string newBucketName = "*** 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()
        {
            client = new AmazonS3Client(bucketRegion);
            CreateBucketUseCannedACLAsync().Wait();
        }

        private static async Task CreateBucketUseCannedACLAsync()
        {
            try
            {
                // Add bucket (specify canned ACL).
                PutBucketRequest putBucketRequest = new PutBucketRequest()
                {
                    BucketName = newBucketName,
                    BucketRegion = S3Region.EUW1, // S3Region.US,
                                                  // Add canned ACL.
                    CannedACL = S3CannedACL.LogDeliveryWrite
                };
                PutBucketResponse putBucketResponse = await
 client.PutBucketAsync(putBucketRequest);

                // Retrieve bucket ACL.
                GetACLResponse getACLResponse = await client.GetACLAsync(new GetACLRequest
                {
                    BucketName = newBucketName
                });
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine("S3 error occurred. Exception: " +
 amazonS3Exception.ToString());
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: " + e.ToString());
            }
        }
    }
}
```

예제 2: 기존 객체에 대한 ACL 권한 부여 구성

이 C# 예제에서는 기존 객체에 대한 ACL을 업데이트합니다. 이 예에서는 다음과 같은 작업을 수행합니다.

- 객체의 ACL을 가져옵니다.
- 기존 권한을 모두 제거하여 ACL을 지웁니다.

- 소유자의 경우 모든 액세스 권한 및 이메일 주소로 식별되는 사용자의 경우 WRITE_ACP 권한, 이렇게 두 가지 권한을 추가합니다.
- `PutAcl` 요청을 보내 ACL을 저장합니다.

실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

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

namespace Amazon.DocSamples.S3
{
    class ManagingObjectACLTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** object key name ***";
        private const string emailAddress = "*** email address ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 client;
        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            TestObjectACLTestAsync().Wait();
        }
        private static async Task TestObjectACLTestAsync()
        {
            try
            {
                    // Retrieve the ACL for the object.
                    GetACLResponse aclResponse = await client.GetACLAsync(new GetACLRequest
                    {
                        BucketName = bucketName,
                        Key = keyName
                    });

                    S3AccessControlList acl = aclResponse.AccessControlList;

                    // Retrieve the owner (we use this to re-add permissions after we clear
 the ACL).
                    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).
                    S3Grant fullControlGrant = new S3Grant
                    {
                        Grantee = new S3Grantee { CanonicalUser = owner.Id },
                        Permission = S3Permission.FULL_CONTROL

                    };

                    // Describe the grant for the permission using an email address.
                    S3Grant grantUsingEmail = new S3Grant
                    {
                        Grantee = new S3Grantee { EmailAddress = emailAddress },
                        Permission = S3Permission.WRITE_ACP
```

```
                    };
                    acl.Grants.AddRange(new List<S3Grant> { fullControlGrant,
 grantUsingEmail });

                    // Set a new ACL.
                    PutACLResponse response = await client.PutACLAsync(new PutACLRequest
                    {
                        BucketName = bucketName,
                        Key = keyName,
                        AccessControlList = acl
                    });
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine("An AmazonS3Exception was thrown. Exception: " +
 amazonS3Exception.ToString());
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: " + e.ToString());
            }
        }
    }
}
```

REST API를 사용해 ACL 관리

Amazon S3 API로 버킷이나 객체 생성 시 ACL을 설정할 수 있습니다. Amazon S3는 기존 버킷이나 객체에 ACL을 설정하는 API도 제공합니다. 이러한 API는 다음과 같은 ACL 설정 메서드를 제공합니다.

- 요청 헤더를 사용한 ACL 설정– 리소스(버킷이나 객체) 생성 요청 전송 시, 요청 헤더로 ACL을 설정합니다. 이러한 헤더를 사용하여 미리 제공된 ACL을 지정하거나 권한을 명시적으로 지정(명시적으로 피부여자와 권한을 식별)할 수 있습니다.
- 요청 본문을 사용한 ACL 설정– 기존 리소스에 대한 ACL 설정 요청을 전송 시, 요청 헤더나 본문에 ACL을 설정할 수 있습니다.

ACL 관리에 필요한 REST API 지원에 대한 자세한 내용은 Amazon Simple Storage Service API Reference에서 다음 단원을 참조하십시오.

- GET Bucket acl
- PUT Bucket acl
- GET Object acl
- PUT Object acl
- PUT Object
- PUT Bucket
- PUT Object - Copy
- 멀티파트 업로드 시작

ACL(액세스 제어 목록) 특정 요청 헤더

헤더를 사용하여 ACL(액세스 제어 목록) 기반 권한을 부여할 수 있습니다. 기본적으로 모든 객체는 비공개이며, 소유자만 모든 액세스 제어 권한을 가집니다. 새 객체를 추가할 때 개별 AWS 계정에 또는 Amazon S3에서 정의한 사전 정의된 그룹에 권한을 부여할 수 있습니다. 그런 다음 이러한 사용 권한이 개체의 ACL(액세스 제어 목록)에 추가됩니다. 자세한 내용은 ACL 사용을 참조하십시오.

이 작업을 통해 다음 두 가지 방법 중 하나를 사용하여 액세스 권한을 부여할 수 있습니다.

- 미리 제공된 ACL(**`x-amz-acl`**) — Amazon S3는 미리 제공된 ACL이라고 하는 사전정의된 ACL 집합을 지원합니다. 미리 제공된 각 ACL에는 사전정의된 피부여자와 권한 집합이 있습니다. 자세한 내용은 미리 제공된 ACL (p. 443) 단원을 참조하십시오.
- 액세스 권한 — 특정 AWS 계정 또는 그룹에 액세스 권한을 명시적으로 부여하려면 다음 헤더를 사용합니다. 각 헤더는 Amazon S3가 ACL에서 지원하는 특정 권한에 매핑됩니다. 자세한 내용은 ACL(액세스 통제 목록) 개요 (p. 438) 단원을 참조하십시오. 헤더에서 특정 권한을 받는 피부여자 목록을 지정합니다.
  - x-amz-grant-read
  - x-amz-grant-write
  - x-amz-grant-read-acp
  - x-amz-grant-write-acp
  - x-amz-grant-full-control

Amazon S3 퍼블릭 액세스 차단 사용

Amazon S3 퍼블릭 액세스 차단 기능은 액세스 포인트, 버킷 및 계정에 대한 설정을 제공하여 Amazon S3 리소스에 대한 퍼블릭 액세스를 관리하는 데 도움을 줍니다. 기본적으로 새 버킷, 액세스 포인트 및 객체는 퍼블릭 액세스를 허용하지 않습니다. 그러나 사용자는 퍼블릭 액세스를 허용하도록 버킷 정책, 액세스 포인트 정책 또는 객체 권한을 수정할 수 있습니다. S3 퍼블릭 액세스 차단 설정은 이러한 정책 및 권한을 무시하므로 이러한 리소스에 대한 퍼블릭 액세스를 제한할 수 있습니다.

S3 퍼블릭 액세스 차단을 사용하면 계정 관리자 및 버킷 소유자가 리소스 생성 방식에 관계없이 적용되는 Amazon S3 리소스에 대한 퍼블릭 액세스를 제한하는 중앙 집중식 제어를 쉽게 설정할 수 있습니다.

Amazon S3가 버킷이나 객체에 대한 액세스 요청을 받으면 버킷이나 버킷 소유자 계정에 적용된 퍼블릭 액세스 차단 설정이 있는지 판단합니다. 액세스 포인트를 통해 요청이 이루어진 경우 Amazon S3는 액세스 포인트에 대한 퍼블릭 액세스 차단 설정도 확인합니다. 요청된 액세스를 금지하는 퍼블릭 액세스 차단 설정이 있는 경우 Amazon S3가 요청을 거부합니다.

Amazon S3 퍼블릭 액세스 차단은 네 가지 설정을 제공합니다. 각각의 설정은 독립적이며 어떤 조합으로든 사용할 수 있습니다. 각 설정은 액세스 포인트, 버킷 또는 전체 AWS 계정에 적용할 수 있습니다. 액세스 포인트, 버킷 또는 계정에 대한 퍼블릭 액세스 차단 설정이 다른 경우 Amazon S3에서는 액세스 포인트, 버킷 및 계정 설정의 가장 제한적인 조합을 적용합니다.

따라서 Amazon S3가 퍼블릭 액세스 차단 설정에 의해 작업이 금지되는지 평가하는 경우, 액세스 포인트, 버킷 또는 계정 설정을 위반하는 요청을 거부합니다.

Note

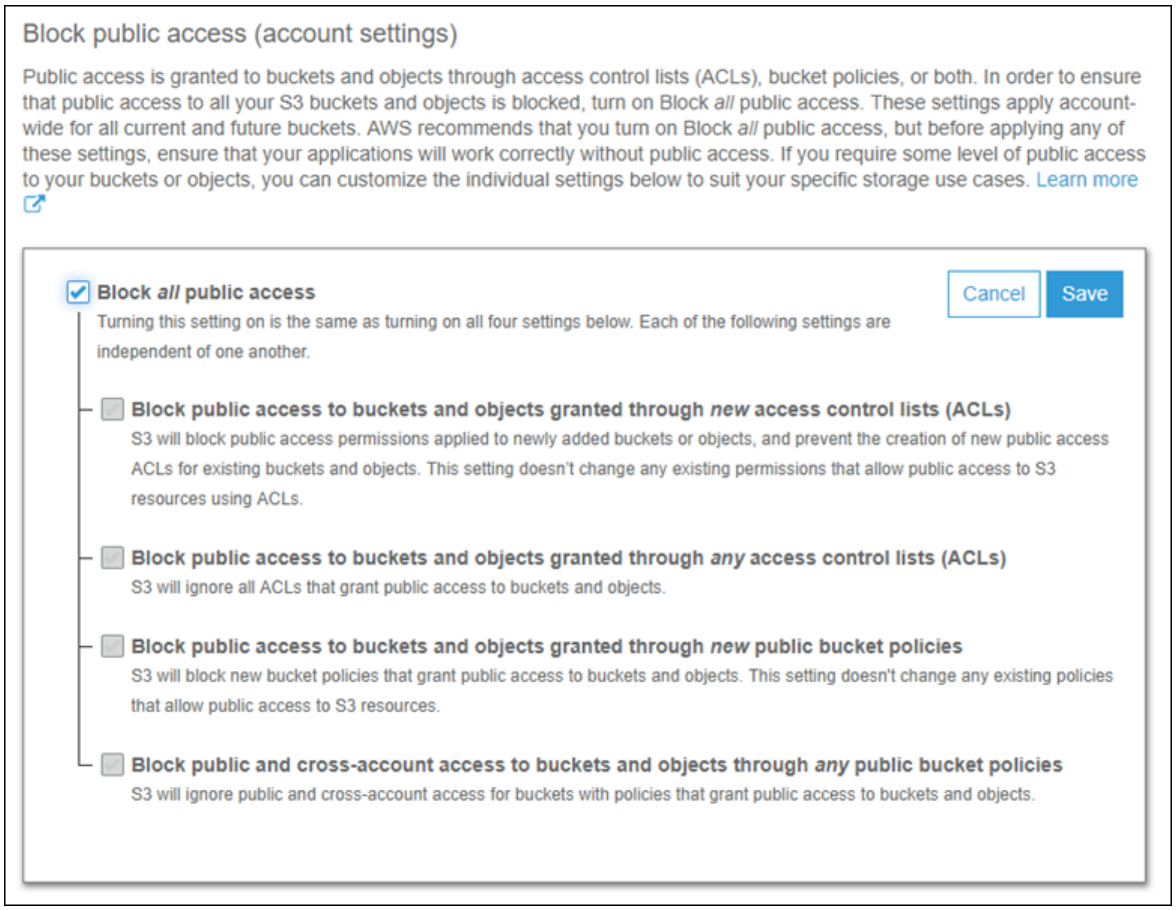
- 액세스 포인트, 버킷 및 AWS 계정에 대해서만 퍼블릭 액세스 차단 설정을 활성화할 수 있습니다. Amazon S3는 객체 단위로 퍼블릭 액세스 차단 설정을 지원하지 않습니다.
- 계정에 퍼블릭 액세스 차단 설정을 적용하면 해당 설정은 전 세계 모든 AWS 리전에 적용됩니다. 설정이 모든 리전에서 즉시 또는 동시에 적용되지는 않지만 결국 모든 리전으로 전파됩니다.

주제

- Amazon S3 콘솔에서 퍼블릭 액세스 차단 활성화 (p. 451)
- 퍼블릭 액세스 차단 설정 (p. 451)
- "퍼블릭"의 의미 (p. 453)
- Access Analyzer for S3를 사용하여 퍼블릭 버킷 검토 (p. 455)
- 권한 (p. 455)
- 예제 (p. 456)

Amazon S3 콘솔에서 퍼블릭 액세스 차단 활성화

Amazon S3 퍼블릭 액세스 차단은 네 가지 설정을 제공합니다. 이러한 설정은 개별 액세스 포인트, 버킷 또는 전체 AWS 계정에 어떤 조합으로든 적용할 수 있습니다. 다음 이미지는 계정의 Amazon S3 콘솔에서 퍼블릭 액세스 차단을 활성화하는 방법을 보여줍니다. 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 권한 설정: 퍼블릭 액세스 차단을 참조하십시오.



퍼블릭 액세스 차단 설정

S3 퍼블릭 액세스 차단은 네 가지 설정을 제공합니다. 이러한 설정은 개별 액세스 포인트, 버킷 또는 전체 AWS 계정에 어떤 조합으로든 적용할 수 있습니다. 설정을 계정에 적용하면 해당 계정이 소유한 모든 버킷 및 액세스 포인트에 적용됩니다. 마찬가지로 버킷에 설정을 적용하면 해당 버킷과 연결된 모든 액세스 포인트에 적용됩니다.

다음 테이블에는 사용 가능한 설정이 포함되어 있습니다.

이름	설명
`BlockPublicAcls`	이 옵션을 `TRUE`로 설정하면 다음과 같은 동작이 발생합니다. • 지정된 ACL(액세스 제어 목록)이 퍼블릭이면 PUT Bucket acl 및 PUT Object acl 호출이 실패합니다. • 요청에 퍼블릭 ACL이 포함되어 있으면 PUT Object 호출이 실패합니다. • 이 설정이 계정에 적용되면, 요청에 퍼블릭 ACL이 포함된 경우 PUT Bucket 호출이 실패합니다.

이름	설명
	이 설정을 `TRUE`로 설정하면 지정된 작업이 실패합니다(REST API, AWS CLI 또는 AWS SDK 등 설정 경로와 상관없이). 그러나 버킷 및 객체에 대한 기존 정책 및 ACL은 수정되지 않습니다. 이 설정을 사용하면 퍼블릭 액세스를 차단하는 동시에 버킷과 객체에 대한 기존 정책 및 ACL을 감사, 미세 조정하거나 변경할 수 있습니다. Note 액세스 포인트에 연결된 ACL이 없습니다. 이 설정을 액세스 포인트에 적용하면 기본 버킷에 대한 패스스루 역할을 합니다. 액세스 포인트에 이 설정이 활성화된 경우, 버킷에 실제로 이 설정이 활성화되어 있는지 여부와 상관없이 액세스 포인트를 통해 이루어진 요청은 기본 버킷에 이 설정이 활성화된 것처럼 동작합니다.
`IgnorePublicAcls`	이 옵션을 `TRUE`로 설정하면 Amazon S3가 버킷의 모든 퍼블릭 ACL 및 거기에 포함된 모든 객체를 무시합니다. 이 설정을 사용하면 ACL에 의해 부여된 퍼블릭 액세스를 안전하게 차단하면서 퍼블릭 ACL을 포함하는 PUT Object 호출을 허용할 수 있습니다(퍼블릭 ACL이 포함된 PUT Object 호출을 거부하는 `BlockPublicAcls`과 반대). 이 설정을 활성화하면 기존 ACL의 지속성에 영향을 주지 않으며 새 퍼블릭 ACL이 설정되는 것을 방지하지 않습니다. Note 액세스 포인트에 연결된 ACL이 없습니다. 이 설정을 액세스 포인트에 적용하면 기본 버킷에 대한 패스스루 역할을 합니다. 액세스 포인트에 이 설정이 활성화된 경우, 버킷에 실제로 이 설정이 활성화되어 있는지 여부와 상관없이 액세스 포인트를 통해 이루어진 요청은 기본 버킷에 이 설정이 활성화된 것처럼 동작합니다.
`BlockPublicPolicy`	버킷에 대해 이 옵션을 `TRUE`로 설정하면 Amazon S3는 지정된 버킷 정책이 퍼블릭 액세스를 허용하는 경우 PUT 버킷 정책에 대한 호출을 거부하고, 지정된 정책에서 퍼블릭 액세스를 허용하는 경우 버킷의 모든 액세스 포인트에 대해 PUT 액세스 포인트 정책에 대한 호출을 거부합니다. 액세스 포인트에 대해 이 옵션을 `TRUE`로 설정하면 Amazon S3는 지정된 정책(액세스 포인트 또는 기본 버킷)이 퍼블릭인 경우 액세스 포인트를 통해 이루어지는 PUT 액세스 포인트 정책 및 PUT 버킷 정책에 대한 호출을 거부합니다. 이 설정을 사용하면 버킷 또는 버킷에 포함된 객체를 공개적으로 공유하지 않고 사용자가 액세스 포인트 및 버킷 정책을 관리할 수 있습니다. 이 설정을 활성화해도 기존 액세스 포인트 또는 버킷 정책에는 영향을 주지 않습니다. Important 이 설정을 효과적으로 사용하려면 계정 레벨에서 적용해야 합니다. 버킷 정책을 통해 사용자가 버킷의 퍼블릭 액세스 차단 설정을 변경할 수 있습니다. 따라서 버킷 정책을 변경할 권한이 있는 사용자는 버킷에 대한 퍼블릭 액세스 차단 설정을 비활성화하도록 허용하는 정책을 삽입할 수 있습니다. 특정 버킷이 아닌 전체 계정에 이 설정이 활성화되면 사용자가 이 설정을 비활성화하기 위해 버킷 정책을 변경한 경우에도 Amazon S3가 퍼블릭 정책을 차단합니다.

이름	설명
`RestrictPublicBuckets`	이 옵션을 `TRUE`로 설정하면 퍼블릭 정책이 있는 액세스 포인트 또는 버킷에 대한 액세스가 버킷 소유자 계정 내에서 권한 있는 사용자와 AWS 서비스로만 제한됩니다. 이 설정은 AWS 서비스에서 제외한 액세스 포인트 또는 버킷에 대한 모든 교차 계정 액세스를 차단하지만 계정 내 사용자는 계속 액세스 포인트 또는 버킷을 관리할 수 있습니다. 이 설정을 활성화하면 기존 액세스 포인트 또는 버킷 정책에 영향을 주지 않습니다. 단, Amazon S3는 특정 계정에 대한 비공개 위임을 포함하여 모든 퍼블릭 액세스 포인트 또는 버킷 정책에서 파생된 퍼블릭 액세스 및 교차 계정 액세스를 차단합니다.

Important

- GET Bucket acl 및 GET Object acl에 대한 호출은 항상 지정된 버킷 또는 객체에 대한 효율적인 권한을 반환합니다. 예를 들어, 버킷에 퍼블릭 액세스를 부여하는 ACL이 있고 버킷에 `IgnorePublicAcls` 설정도 활성화되어 있다고 가정하겠습니다. 이 경우 GET Bucket acl은 버킷과 연결된 실제 ACL이 아닌 Amazon S3가 적용되는 액세스 권한을 반영하는 ACL을 반환합니다.
- 퍼블릭 액세스 차단 설정은 기존 정책이나 ACL을 변경하지 않습니다. 따라서 퍼블릭 액세스 차단 설정을 제거하면 퍼블릭 정책이나 ACL이 있는 버킷이나 객체가 다시 공개적으로 액세스할 수 있습니다.

"퍼블릭"의 의미

버킷

ACL

Amazon S3는 미리 정의된 `AllUsers` 또는 `AuthenticatedUsers` 그룹의 구성원에게 권한을 부여하는 경우 버킷 또는 객체 ACL을 퍼블릭으로 간주합니다. 미리 정의된 그룹에 대한 자세한 내용은 Amazon S3의 미리 정의된 그룹 (p. 440) 단원을 참조하십시오.

정책

버킷 정책을 평가할 때 Amazon S3는 정책을 퍼블릭으로 가정하여 시작합니다. 그런 다음 정책을 평가하여 비공개로 판단할 수 있는지 결정합니다. 퍼블릭이 아닌 것으로 평가되려면 버킷 정책은 다음 중 하나 이상의 고정 값(와일드카드가 없는 값)에만 액세스 권한을 부여해야 합니다.

- `aws:SourceIp`를 사용하는 클래스 없는 도메인 간 라우팅(CIDR) 집합. CIDR에 대한 자세한 내용은 RFC Editor 웹 사이트에서 RFC 4632를 참조하십시오.
- AWS 보안 주체, 사용자, 역할 또는 서비스 보안 주체(예: `aws:PrincipalOrgID`)
- `aws:SourceArn`
- `aws:SourceVpc`
- `aws:SourceVpce`
- `aws:SourceOwner`
- `aws:SourceAccount`
- `s3:x-amz-server-side-encryption-aws-kms-key-id`
- `aws:userid`, 패턴 외부 `"AROLEID:*"`
- `s3:DataAccessPointArn`

  Note

  버킷 정책에 사용되는 경우 계정 ID가 고정되어 있는 한 정책을 퍼블릭으로 렌더링하지 않고도 액세스 포인트 이름에 대한 와일드카드가 이 값에 포함될 수 있습니다. 예를 들어

`arn:aws:s3:us-west-2:123456789012:accesspoint/*`에 대한 액세스를 허용하면 버킷 정책을 퍼블릭으로 렌더링하지 않고도 리전 `us-west-2`의 계정 `123456789012`와 연결된 모든 액세스 포인트에 대한 액세스가 허용됩니다. 이 동작은 액세스 포인트 정책에 따라 다릅니다. 자세한 내용은 액세스 포인트 (p. 455) 단원을 참조하십시오.

- `s3:DataAccessPointAccount`

이 규칙에서 다음 예제 정책은 퍼블릭으로 간주됩니다.

```
{
  "Principal": { "Federated": "graph.facebook.com" },
  "Resource": "*",
  "Action": "s3:PutObject",
  "Effect": "Allow"
 }
```

```
{
  "Principal": "*",
  "Resource": "*",
  "Action": "s3:PutObject",
  "Effect": "Allow"
 }
```

```
{
  "Principal": "*",
  "Resource": "*",
  "Action": "s3:PutObject",
  "Effect": "Allow",
  "Condition": { "StringLike": {"aws:SourceVpc": "vpc-*"}}
 }
```

고정 값을 사용하여 이전에 나열된 조건 키를 포함하면 이러한 정책을 비공개로 만들 수 있습니다. 예를 들어 앞의 마지막 정책은 다음과 같이 `aws:SourceVpc`를 고정 값으로 설정하여 비공개로 만들 수 있습니다.

```
{
  "Principal": "*",
  "Resource": "*",
  "Action": "s3:PutObject",
  "Effect": "Allow",
  "Condition": {"StringEquals": {"aws:SourceVpc": "vpc-91237329"}}
 }
```

버킷 정책에 대한 자세한 내용은 버킷 정책 및 사용자 정책 사용 (p. 338)를 참조하십시오.

예

이 예제는 Amazon S3가 퍼블릭 액세스 권한과 퍼블릭이 아닌 액세스 권한을 모두 포함하는 버킷 정책을 어떻게 평가하는지 보여줍니다.

버킷에 고정된 보안 주체 집합에 대한 액세스를 부여하는 정책이 있다고 가정하겠습니다. 이전에 설명한 규칙에 따르면 이 정책은 퍼블릭이 아닙니다. 따라서 `RestrictPublicBuckets` 설정을 활성화하면 `RestrictPublicBuckets`이 퍼블릭 정책이 있는 버킷에만 적용되므로 정책은 작성된 대로 유효하게 유지됩니다. 그러나 정책에 퍼블릭 설명을 추가하면 `RestrictPublicBuckets`이 버킷에 영향을 줍니다. 그러면 버킷 소유자 계정의 권한 있는 사용자와 AWS 서비스 주체만 버킷에 액세스할 수 있습니다.

예를 들어, "Account-1"이 소유한 버킷에 다음을 포함하는 정책이 있다고 가정하겠습니다.

1. AWS CloudTrail(AWS 서비스 보안 주체)에 액세스 권한을 부여하는 설명문

2. "Account-2" 계정에 액세스 권한을 부여하는 설명문
3. 예를 들어 `Condition` 제한 없이 `"Principal": "*"`를 지정하여 퍼블릭에 액세스 권한을 부여하는 설명문

이 정책은 세 번째 설명문으로 인해 퍼블릭으로 평가됩니다. 이 정책이 있고 `RestrictPublicBuckets`이 활성화되면, Amazon S3는 CloudTrail의 액세스만 허용합니다. 설명문 2가 퍼블릭이 아니더라도 Amazon S3는 "Account-2"에 의한 액세스를 비활성화합니다. 설명문 3이 전체 정책을 퍼블릭으로 만들기 때문으로, `RestrictPublicBuckets`이 적용됩니다. 결과적으로 정책이 특정 계정 "Account-2"에 대한 액세스를 위임하더라도 Amazon S3는 교차 계정 액세스를 비활성화합니다. 그러나 정책에서 설명문 3을 제거하면 정책은 퍼블릭으로 평가되지 않으며 `RestrictPublicBuckets`은 더 이상 적용되지 않습니다. 따라서 `RestrictPublicBuckets`을 활성화해 두어도 "Account-2"는 버킷에 대한 액세스를 다시 얻습니다.

액세스 포인트

Amazon S3는 버킷과 비교하여 액세스 포인트에 대해 퍼블릭 액세스 차단 설정을 약간 다르게 평가합니다. 액세스 포인트 정책이 퍼블릭인 시점을 결정하기 위해 Amazon S3가 적용하는 규칙은 다음과 같은 경우를 제외하고 일반적으로 버킷에 대한 규칙과 액세스 포인트에 대한 규칙이 동일합니다.

- 네트워크 오리진이 VPC인 액세스 포인트는 액세스 포인트 정책의 내용에 관계없이 항상 퍼블릭이 아닌 것으로 간주됩니다.
- `s3:DataAccessPointArn`를 사용하여 액세스 포인트 집합에 대한 액세스 권한을 부여하는 액세스 포인트 정책은 퍼블릭으로 간주됩니다. 이 동작은 버킷 정책과 다릅니다. 예를 들어, `arn:aws:s3:us-west-2:123456789012:accesspoint/*`와 일치하는 `s3:DataAccessPointArn` 값에 액세스 권한을 부여하는 버킷 정책은 공개로 간주되지 않습니다. 그러나 액세스 포인트 정책의 동일한 명령문은 액세스 포인트를 퍼블릭으로 렌더링합니다.

Access Analyzer for S3를 사용하여 퍼블릭 버킷 검토

Access Analyzer for S3를 사용하여 퍼블릭 액세스 권한을 부여하는 버킷 ACL 또는 버킷 정책이 있는 버킷을 검토할 수 있습니다. Access Analyzer for S3는 인터넷상의 모든 사용자 또는 조직 외부의 AWS 계정을 포함한 다른 AWS 계정에 대한 액세스를 허용하도록 구성된 버킷에 대한 알림을 제공합니다. 각 퍼블릭 버킷 또는 공유 버킷에 대해 퍼블릭 액세스 또는 공유 액세스의 수준과 원본을 보고하는 결과가 수신됩니다.

결과에 제시된 지식을 활용하면 즉각적이고 정확한 시정 조치를 취할 수 있습니다. Access Analyzer for S3에서는 한 번의 클릭으로 버킷에 대한 모든 퍼블릭 액세스를 차단할 수 있습니다. 또한 버킷 수준 권한 설정으로 드릴다운하여 세부적인 액세스 수준을 구성할 수 있습니다. 퍼블릭 액세스 또는 공유 액세스가 필요한 것으로 확인된 특정 사용 사례의 경우, 버킷에 대한 결과를 보관하여 버킷을 퍼블릭 상태로 유지할 것인지 공유 상태로 유지할 것인지 확인하고 기록할 수 있습니다.

드문 경우이지만 Access Analyzer for S3에서는 Amazon S3 퍼블릭 액세스 차단 평가에서 공개적으로 보고하는 버킷에 대한 결과를 보고되지 않을 수 있습니다. 이는 Amazon S3 퍼블릭 액세스 차단이 현재 작업 및 향후 추가될 수 있는 모든 잠재적 작업에 대한 정책을 검토하기 때문에 발생하므로 버킷이 퍼블릭이 됩니다. 반면에 Access Analyzer for S3는 액세스 상태를 평가할 때 Amazon S3 서비스에 대해 지정된 현재 작업만 분석합니다.

Access Analyzer for S3에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 Access Analyzer for S3 사용을 참조하십시오.

권한

Amazon S3 퍼블릭 액세스 차단 기능을 사용하려면 다음 권한이 있어야 합니다.

작업	필요한 권한
GET 버킷 정책 상태	`s3:GetBucketPolicyStatus`

작업	필요한 권한
버킷 퍼블릭 액세스 차단 설정 GET	`s3:GetBucketPublicAccessBlock`
버킷 퍼블릭 액세스 차단 설정 PUT	`s3:PutBucketPublicAccessBlock`
버킷 퍼블릭 액세스 차단 설정 DELETE	`s3:PutBucketPublicAccessBlock`
계정 퍼블릭 액세스 차단 설정 GET	`s3:GetAccountPublicAccessBlock`
계정 퍼블릭 액세스 차단 설정 PUT	`s3:PutAccountPublicAccessBlock`
계정 퍼블릭 액세스 차단 설정 DELETE	`s3:PutAccountPublicAccessBlock`
PUT 액세스 포인트 퍼블릭 액세스 차단 설정	`s3:PutAccessPointPublicAccessBlock`

Note

DELETE 작업에는 PUT 작업과 동일한 권한이 필요합니다. DELETE 작업에 대한 별도의 권한은 없습니다.

예제

AWS CLI의 퍼블릭 액세스 차단 사용

AWS CLI를 통해 Amazon S3 퍼블릭 액세스 차단을 사용할 수 있습니다. 사용하는 명령은 액세스 포인트, 버킷 또는 계정에서 퍼블릭 액세스 차단 호출을 수행할지 여부에 따라 다릅니다. AWS CLI의 설정 및 사용에 대한 자세한 내용은 AWS Command Line Interface란? 단원을 참조하십시오.

- 액세스 포인트

  액세스 포인트에서 퍼블릭 액세스 차단 작업을 수행하려면 AWS CLI 서비스 `s3control`을 사용하십시오. 현재는 액세스 포인트를 생성한 후에 액세스 포인트의 퍼블릭 액세스 차단 설정을 변경할 수 없습니다. 따라서 액세스 포인트에 대한 퍼블릭 액세스 차단 설정을 지정하는 유일한 방법은 액세스 포인트를 생성할 때 포함시키는 것입니다.

  버킷

  버킷에서 퍼블릭 액세스 차단 작업을 수행하려면 AWS CLI 서비스 `s3api`를 사용하십시오. 이 서비스를 사용하는 버킷 수준 작업은 다음과 같습니다.

  - PUT PublicAccessBlock(버킷)
  - GET PublicAccessBlock(버킷)
  - DELETE PublicAccessBlock(버킷)
  - GET BucketPolicyStatus

  계정

  계정에서 퍼블릭 액세스 차단 작업을 수행하려면 AWS CLI 서비스 `s3control`을 사용하십시오. 이 서비스를 사용하는 계정 수준 작업은 다음과 같습니다.

  - PUT PublicAccessBlock(계정)
  - GET PublicAccessBlock(계정)
  - DELETE PublicAccessBlock(계정)

AWS SDK for Java의 퍼블릭 액세스 차단 사용

다음 예제는 AWS SDK for Java에서 Amazon S3 퍼블릭 액세스 차단을 사용하는 방법을 보여줍니다. 실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 AWS SDK for Java 사용 (p. 733) 단원을 참조하십시오.

예제 1

이 예제는 AWS SDK for Java를 사용하여 S3 버킷에 퍼블릭 액세스 차단 구성을 설정하는 방법을 보여줍니다.

```
AmazonS3 client = AmazonS3ClientBuilder.standard()
   .withCredentials(<credentials>)
   .build();

client.setPublicAccessBlock(new SetPublicAccessBlockRequest()
  .withBucketName(<bucket-name>)
  .withPublicAccessBlockConfiguration(new PublicAccessBlockConfiguration()
    .withBlockPublicAcls(<value>)
    .withIgnorePublicAcls(<value>)
    .withBlockPublicPolicy(<value>)
    .withRestrictPublicBuckets(<value>)));
```

> **Important**
>
> 이 예제는 `AmazonS3` 클라이언트 클래스를 사용하는 버킷 수준 작업에만 적용됩니다. 계정 수준 작업의 경우 다음 예제를 참조하십시오.

예제 2

이 예제는 AWS SDK for Java를 사용하여 Amazon S3 계정에 퍼블릭 액세스 차단 구성을 설정하는 방법을 보여줍니다.

```
AWSS3ControlClientBuilder controlClientBuilder = AWSS3ControlClientBuilder.standard();
controlClientBuilder.setRegion(<region>);
controlClientBuilder.setCredentials(<credentials>);

AWSS3Control client = controlClientBuilder.build();
client.putPublicAccessBlock(new PutPublicAccessBlockRequest()
  .withAccountId(<account-id>)
  .withPublicAccessBlockConfiguration(new PublicAccessBlockConfiguration()
    .withIgnorePublicAcls(<value>)
    .withBlockPublicAcls(<value>)
    .withBlockPublicPolicy(<value>)
    .withRestrictPublicBuckets(<value>)));
```

> **Important**
>
> 이 예제는 `AWSS3Control` 클라이언트 클래스를 사용하는 계정 수준 작업에만 적용됩니다. 버킷 수준 작업의 경우 앞의 예를 참조하십시오.

다른 AWS SDK의 퍼블릭 액세스 차단 사용

다른 AWS SDK 사용 방법에 대한 자세한 내용은 AWS SDK, CLI 및 Explorer 사용 (p. 727) 단원을 참조하십시오.

REST API의 퍼블릭 액세스 차단 사용

REST API를 통한 Amazon S3 퍼블릭 액세스 차단 사용에 대한 자세한 내용은 Amazon Simple Storage Service API Reference에서 다음 주제를 참조하십시오.

- 계정 수준 작업

  - PUT PublicAccessBlock
  - GET PublicAccessBlock
  - DELETE PublicAccessBlock

- 버킷 수준 작업
  - PUT PublicAccessBlock
  - GET PublicAccessBlock
  - DELETE PublicAccessBlock
  - GET BucketPolicyStatus

Amazon S3의 로깅 및 모니터링

모니터링은 Amazon S3과 사용자 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 중요한 역할을 합니다. 다중 지점 실패가 발생할 경우 보다 쉽게 디버깅할 수 있도록 AWS 솔루션의 모든 부분으로부터 모니터링 데이터를 수집해야 합니다. AWS는 Amazon S3 리소스를 모니터링하고 잠재적 인시던트에 대응하기 위한 여러 도구를 제공합니다.

Amazon CloudWatch 경보

Amazon CloudWatch 경보를 사용하면 지정한 기간 동안 단일 지표를 감시합니다. 지표가 지정된 임계값을 초과하면 Amazon SNS 주제 또는 AWS Auto Scaling 정책에 알림이 전송됩니다. CloudWatch 경보는 특정 상태에 있기 때문에 작업을 호출하지 않습니다. 대신, 상태가 변경되어 지정된 기간 동안 유지되어야 합니다. 자세한 내용은 Amazon CloudWatch로 지표 모니터링 (p. 669) 단원을 참조하십시오.

AWS CloudTrail 로그

CloudTrail은 Amazon S3에서 사용자, 역할 또는 AWS 제품이 수행한 작업의 레코드를 제공합니다. CloudTrail에서 수집하는 정보를 사용하여 Amazon S3에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다. 자세한 내용은 AWS CloudTrail을 사용하여 Amazon S3 API 호출 로깅 (p. 679) 단원을 참조하십시오.

Amazon S3 액세스 로그

서버 액세스 로그는 버킷에 대한 요청의 상세한 레코드를 제공합니다. 서버 액세스 로그는 많은 애플리케이션에 있어 유용합니다. 예를 들어 액세스 로그 정보는 보안 및 액세스 감사에 유용할 수 있습니다. 자세한 내용은 Amazon S3 서버 액세스 로깅 (p. 705) 단원을 참조하십시오.

AWS Trusted Advisor

Trusted Advisor는 수십만 명의 AWS 고객에게 서비스를 제공하면서 익힌 모범 사례를 활용합니다. Trusted Advisor는 AWS 환경을 검사한 후 비용 절감, 시스템 가용성 및 성능 향상 또는 보안 격차를 해결할 기회가 있을 때 권장 사항을 제시합니다. 모든 AWS 고객은 5개의 Trusted Advisor 점검 항목에 액세스할 수 있습니다. Business 또는 Enterprise Support 플랜을 보유한 고객은 모든 Trusted Advisor 점검 항목을 볼 수 있습니다.

Trusted Advisor에는 다음과 같이 Amazon S3 관련 검사가 있습니다.

- Amazon S3 버킷의 로깅 구성 검사
- 공개 액세스 권한을 가진 Amazon S3 버킷에 대한 보안 검사
- 버전 관리가 활성화되지 않았거나 버전 관리가 중지된 Amazon S3 버킷에 대한 내결함성 검사

자세한 정보는 AWS Support 사용 설명서의 AWS Trusted Advisor 단원을 참조하십시오.

다음 보안 모범 사례에서도 로깅 및 모니터링을 다룹니다.

- 모든 Amazon S3 버킷 식별 및 감사

- AWS 모니터링 도구를 사용하여 모니터링 구현
- AWS Config 활성화
- Amazon S3 서버 액세스 로깅 활성화
- AWS CloudTrail 사용
- AWS 보안 권고 사항 모니터링

Amazon S3 규정 준수 확인

타사 감사자는 다음을 포함하는 여러 AWS 규정 준수 프로그램의 일환으로 Amazon S3의 보안 및 규정 준수를 평가합니다.

- SOC(시스템 및 조직 제어)
- PCI DSS(지불 카드 산업 데이터 보안 표준)
- 연방정부의 위험 및 인증 관리 프로그램(FedRAMP)
- HIPAA(미국 건강 보험 양도 및 책임에 관한 법)

AWS는 규정 준수 프로그램 제공 범위 내 AWS 제품 페이지에서 특정 규정 준수 프로그램 범위 내 자주 업데이트되는 AWS 제품의 목록을 제공합니다.

AWS Artifact를 사용하여 다운로드할 수 있는 타사 감사 보고서가 있습니다. 자세한 내용은 AWS 아티팩트의 보고서 다운로드를 참조하십시오.

AWS 규정 준수 프로그램에 대한 자세한 내용은 AWS 규정 준수 프로그램을 참조하십시오.

Amazon S3 사용 시 규정 준수 책임은 데이터의 민감도, 조직의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다. Amazon S3 사용 시 HIPAA, PCI, FedRAMP와 같은 표준을 준수해야 하는 경우 다음과 같은 AWS의 도움말 리소스를 활용하십시오.

- 보안 및 규정 준수 빠른 시작 안내서에서는 아키텍처 고려 사항에 대해 설명하고 보안 및 규정 준수에 중점을 둔 기본 AWS 환경을 배포하기 위한 단계를 제공합니다.
- HIPAA 보안 및 규정 준수를 위한 설계 백서에서는 기업에서 AWS를 사용하여 HIPAA 요구 사항을 충족하는 방법을 설명합니다.
- AWS 규정 준수 리소스는 귀사의 산업 및 위치에 적용될 수 있는 몇몇 종류별 워크북 및 안내서를 제공합니다.
- AWS Config를 사용하여 리소스 구성이 내부 관행, 업계 지침 및 규정을 준수하는 정도를 평가할 수 있습니다.
- AWS Security Hub는 보안 업계 표준 및 모범 사례 준수 여부를 확인하는 데 도움이 되는 AWS 내 보안 상태에 대한 포괄적인 관점을 제공합니다.
- Amazon S3 객체 잠금으로 특정한 유형의 도서 및 기록 정보에 WORM(write once, read many) 데이터 스토리지를 요구하는 금융 서비스 규제 기관(예: SEC, FINRA, CFTC)의 기술 요구 사항을 충족할 수 있습니다.
- Amazon S3 인벤토리 (p. 459)를 사용하여 비즈니스, 규정 준수 및 규제 요건에 대한 객체의 복제 및 암호화 상태를 감사하고 보고할 수 있습니다.

Amazon S3 인벤토리

Amazon S3 인벤토리는 스토리지 관리에 도움이 되는 Amazon S3의 도구 중 하나입니다. 이 인벤토리를 사용하여 비즈니스, 규정 준수 및 규제 요건에 대한 객체의 복제 및 암호화 상태를 감사하고 보고할 수 있습니다. 또한 Amazon S3 동기식 `List` API 작업의 대안으로 Amazon S3 인벤토리를 사용하면 비즈니스 워크플로우 및 빅 데이터 업무를 단순화하고 속도를 높일 수 있습니다.

Amazon S3 인벤토리는 S3 버킷 또는 공유 접두사(즉, 공통 문자열로 시작하는 이름을 가진 객체)에 대해 매일 혹은 매주 객체 및 해당 메타데이터의 CSV(쉼표로 구분되는 값), Apache ORC(Optimized Row Columnar) 또는 Apache Parquet(Parquet) 파일 출력을 가능하게 합니다. 매주로 설정한 경우 보고서는 초기 보고서 이후 7일마다 생성됩니다. Amazon S3 인벤토리 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

하나의 버킷에 대해 복수의 인벤토리 목록을 구성할 수 있습니다. 인벤토리에 포함시킬 객체 메타데이터, 모든 객체 버전을 목록에 포함시킬 것인지 혹은 최신 버전만을 목록에 포함시킬 것인지, 인벤토리 목록 파일 출력을 저장할 위치, 인벤토리를 매일 생성할 것인지 혹은 매주 생성할 것인지를 설정할 수 있습니다. 인벤토리 목록 파일을 암호화하도록 지정할 수도 있습니다.

Amazon Athena, Amazon Redshift Spectrum 및 기타 도구(예: Presto, Apache Hive 및 Apache Spark)를 사용하여 표준 SQL을 통해 Amazon S3 인벤토리를 쿼리할 수 있습니다. Athena를 사용하면 인벤토리 파일에 대한 쿼리를 쉽게 실행할 수 있습니다. Athena가 사용 가능한 모든 리전에서 Athena를 사용하여 Amazon S3 인벤토리를 쿼리할 수 있습니다.

주제

- Amazon S3 인벤토리 설정은 어떻게 합니까? (p. 460)
- Amazon S3 인벤토리에 포함된 항목은 무엇입니까? (p. 462)
- 인벤토리 목록의 위치는 어디입니까? (p. 463)
- 인벤토리가 완료되는 시점을 어떻게 알 수 있습니까? (p. 465)
- Amazon Athena로 인벤토리 쿼리 (p. 466)
- Amazon S3 인벤토리 REST API (p. 467)

Amazon S3 인벤토리 설정은 어떻게 합니까?

이 단원에서는 인벤토리 원본 및 대상 버킷에 대한 세부 정보를 비롯하여 인벤토리를 설정하는 방법에 대해 설명합니다.

Amazon S3 인벤토리 원본 및 대상 버킷

인벤토리가 객체를 열거하는 버킷을 원본 버킷이라고 합니다. 인벤토리 목록 파일이 저장되는 버킷을 대상 버킷이라고 합니다.

원본 버킷

인벤토리는 원본 버킷에 저장되어 있는 객체를 열거합니다. 전체 버킷에 대한 인벤토리 목록 또는 (객체 키 이름) 접두사로 필터링한 인벤토리 목록을 생성할 수 있습니다.

원본 버킷:

- 인벤토리에 열거되어 있는 객체를 포함합니다.
- 인벤토리 구성을 포함합니다.

대상 버킷

Amazon S3 인벤토리 목록 파일이 대상 버킷에 작성됩니다. 대상 버킷 내 일반적인 위치에 있는 모든 인벤토리 목록 파일을 그룹화하려면 인벤토리 구성에서 대상(객체 키 이름) 접두사를 지정할 수 있습니다.

대상 버킷:

- 인벤토리 파일 목록을 포함합니다.
- 대상 버킷에 저장되어 있는 모든 파일 인벤토리 목록이 열거된 매니페스트 파일을 포함합니다. 자세한 내용은 인벤토리 매니페스트는 무엇입니까? (p. 464) 단원을 참조하십시오.

- Amazon S3가 버킷의 소유권을 검증하고 파일을 버킷에 쓸 수 있도록 허용하는 버킷 정책이 반드시 있어야 합니다.
- 원본 버킷과 동일한 AWS 리전에 있어야 합니다.
- 원본 버킷과 같을 수 있습니다.
- 원본 버킷을 소유한 계정과 다른 AWS 계정에서 소유할 수 있습니다.

Amazon S3 인벤토리 설정

Amazon S3 인벤토리를 사용하면 정해진 일정에 객체 목록이 S3 버킷에 생성되므로 스토리지를 용이하게 관리할 수 있습니다. 하나의 버킷에 대해 복수의 인벤토리 목록을 구성할 수 있습니다. 인벤토리 목록은 대상 버킷에 있는 CSV, ORC 또는 Parquet 파일에 게시됩니다.

인벤토리를 설정하는 가장 쉬운 방법은 AWS Management 콘솔을 사용하는 것이지만, REST API, AWS CLI 또는 AWS SDK를 사용할 수도 있습니다. 콘솔에서는 대상 버킷에 버킷 정책을 추가하는 절차의 첫 번째 단계를 자동으로 수행합니다.

S3 버킷의 Amazon S3 인벤토리를 설정하려면

1. 대상 버킷의 버킷 정책을 추가합니다.

   객체를 정해진 위치에 있는 버킷에 쓸 수 있는 권한을 Amazon S3에 부여하는 버킷 정책을 대상 버킷 상에 생성해야 합니다. 정책에 대한 예는 Amazon S3 인벤토리 및 Amazon S3 분석에 권한 부여 (p. 414)을 참조하십시오.
2. 원본 버킷에 있는 객체를 열거하여 대상 버킷에 목록을 게시하도록 인벤토리를 구성합니다.

   원본 버킷에 대한 인벤토리 목록을 구성할 때 목록을 저장하고자 하는 대상 버킷을 지정하고 목록을 매일 혹은 매주 생성할지 정합니다. 또한 포함시킬 객체 메타데이터와 모든 객체 버전을 목록에 포함시킬지 아니면 최신 버전만을 목록에 포함시킬지 구성할 수 있습니다.

   Amazon S3 관리형 키(SSE-S3) 또는 AWS Key Management Service(AWS KMS) 고객 관리형 고객 마스터 키(CMK)를 사용하여 인벤토리 목록 파일을 암호화하도록 지정할 수 있습니다. SSE-S3 및 SSE-KMS에 대한 자세한 내용은 서버 측 암호화를 사용하여 데이터 보호 (p. 263) 단원을 참조하십시오. SSE-KMS 암호화를 사용하려면 3단계를 참조하십시오.
   - 콘솔을 사용하여 인벤토리 목록을 구성하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 Amazon S3 인벤토리를 구성하려면? 단원을 참조하십시오.
   - Amazon S3 API를 사용하여 인벤토리 목록을 구성하려면, PUT Bucket 인벤토리 구성 REST API 또는 AWS CLI나 AWS SDK의 동일한 기능을 사용하면 됩니다.
3. SSE-KMS를 사용하여 인벤토리 목록 파일을 암호화하려면 AWS KMS에 저장된 CMK 사용 권한을 Amazon S3에 부여합니다.

   AWS Management 콘솔, REST API, AWS CLI 또는 AWS SDK를 사용하여 인벤토리 목록 파일에 대한 암호화를 구성할 수 있습니다. 어느 방법을 선택하든 인벤토리 파일을 암호화하려면 Amazon S3에 AWS KMS 고객 관리형 CMK 사용 권한을 부여해야 합니다. 인벤토리 파일을 암호화하는 데 사용할 고객 관리형 CMK의 키 정책을 수정하여 Amazon S3에 권한을 부여합니다. 자세한 정보는 다음 섹션을 참조하십시오, Amazon S3에 암호화를 위해 AWS KMS CMK를 사용할 수 있는 권한 부여 (p. 461).

Amazon S3에 암호화를 위해 AWS KMS CMK를 사용할 수 있는 권한 부여

고객 관리형 AWS Key Management Service(AWS KMS) 고객 마스터 키(CMK)를 사용하여 암호화할 수 있는 권한을 Amazon S3에 부여하려면 키 정책을 사용해야 합니다. AWS KMS 고객 관리형 CMK를 사용하여 인벤토리 파일을 암호화할 수 있도록 키 정책을 업데이트하려면 다음 단계를 따르십시오.

AWS KMS CMK를 사용하여 암호화할 수 있는 권한을 부여하려면

1. 고객 관리형 CMK를 소유한 AWS 계정을 사용하여 AWS Management 콘솔에 로그인합니다.

2. https://console.aws.amazon.com/kms에서 AWS KMS 콘솔을 엽니다.
3. AWS 리전을 변경하려면 페이지의 오른쪽 위 모서리에 있는 리전 선택기를 사용합니다.
4. 탐색 창에서 고객 관리형 키를 선택합니다.
5. 고객 관리형 키에서 인벤토리 파일을 암호화하는 데 사용할 고객 관리 CMK를 선택합니다.
6. 키 정책에서 정책 보기로 전환을 선택합니다.
7. 키 정책을 업데이트하려면 편집을 선택합니다.
8. 키 정책 편집에서 기존 키 정책에 다음 키 정책을 추가합니다.

```
{
    "Sid": "Allow Amazon S3 use of the CMK",
    "Effect": "Allow",
    "Principal": {
        "Service": "s3.amazonaws.com"
    },
    "Action": [
        "kms:GenerateDataKey"
    ],
    "Resource": "*"
}
```

9. Save changes(변경 사항 저장)를 선택합니다.

   AWS KMS 고객 관리형 CMK 생성 및 키 정책 사용에 대한 자세한 내용은 AWS Key Management Service Developer Guide의 다음 주제를 참조하십시오.

   - 시작하기
   - AWS KMS에서 키 정책 사용

   AWS KMS PUT 키 정책 API PutKeyPolicy를 사용하여 인벤토리 파일을 암호화하는 데 사용할 고객 관리형 CMK에 키 정책을 복사할 수도 있습니다.

Amazon S3 인벤토리에 포함된 항목은 무엇입니까?

인벤토리 목록 파일에는 원본 버킷에 있는 객체의 목록과 각 객체의 메타데이터가 포함됩니다. 인벤토리 목록은 대상 버킷에 GZIP으로 압축된 CSV 파일, ZLIB로 압축된 Apache ORC(Optimized Row Columnar) 파일 또는 Snappy로 압축된 Apache Parquet(Parquet) 파일로 저장됩니다.

인벤토리 목록에는 S3 버킷에 있는 객체 목록과 목록 상의 각 객체에 대한 다음의 메타데이터가 포함됩니다.

- 버킷 이름 – 인벤토리 대상 버킷의 이름.
- Key name – 버킷에 있는 객체를 고유하게 식별하는 객체 키 이름(또는 키). CSV 파일 형식을 사용하는 경우 키 이름이 URL로 암호화되므로 사용하려면 디코딩해야 합니다.
- Version ID – 객체 버전 ID. 버전 관리를 사용하는 버킷의 경우 Amazon S3는 버킷에 추가된 객체에 버전 번호를 지정합니다. 자세한 내용은 객체 버전 관리 (p. 112) 단원을 참조하십시오. (목록에 최신 버전의 객체만 있을 경우 이 필드는 포함되지 않습니다.)
- IsLatest – 객체가 최신 버전인 경우 `True`로 설정됩니다. (목록에 최신 버전의 객체만 있을 경우 이 필드는 포함되지 않습니다.)
- Size – 객체 크기(바이트).
- Last modified date – 객체 생성일 또는 최종 수정일 중 최근 날짜.
- ETag – 개체 태그는 객체의 해시입니다. ETag는 객체의 콘텐츠에 대한 변경 사항만 반영하고 메타데이터에 대한 변경을 반영하지 않습니다. ETag는 객체 데이터의 MD5 다이제스트일 수도 아닐 수도 있습니다. 다이제스트인지 여부는 객체 생성 방식 및 암호화 방식에 좌우됩니다.
- Storage class – 객체 저장에 사용되는 스토리지 클래스. 자세한 내용은 Amazon S3 스토리지 클래스 (p. 107) 단원을 참조하십시오.

- Intelligent-Tiering access tier – 지능형 계층화로 저장된 경우 객체의 액세스 티어(Frequent Access 또는 Infrequent Access). 자세한 내용은 Amazon S3 지능형 계층화를 참조하십시오.
- Multipart upload flag – 객체가 멀티파트 업로드로 업로드된 경우 `True`로 설정됩니다. 자세한 내용은 멀티파트 업로드 개요 (p. 178) 단원을 참조하십시오.
- Delete marker – 객체가 삭제 마커인 경우 `True`로 설정됩니다. 자세한 내용은 객체 버전 관리 (p. 112) 단원을 참조하십시오. (모든 버전의 객체를 포함하도록 보고서를 구성한 경우 이 필드는 보고서에 자동으로 추가됩니다.)
- Replication status – `PENDING`, `COMPLETED`, `FAILED` 또는 `REPLICA.`로 설정됩니다. 자세한 내용은 복제 상태 정보 (p. 654) 단원을 참조하십시오.
- Encryption status – `SSE-S3`, `SSE-C`, `SSE-KMS` 또는 `NOT-SSE`로 설정됩니다. SSE-S3, SSE-KMS 및 고객 제공 키를 사용하는 SSE(SSE-C)에 대한 서버 측 암호화 상태입니다. `NOT-SSE` 상태는 객체가 서버 측 암호화를 사용하여 암호화되지 않은 것을 의미합니다. 자세한 내용은 암호화를 사용하여 데이터 보호 (p. 262) 단원을 참조하십시오.
- 객체 잠금 보관 종료일 – 잠긴 객체를 삭제할 수 없는 기한. 자세한 내용은 Amazon S3 객체 잠금을 사용하여 객체 잠금 (p. 490) 단원을 참조하십시오.
- 객체 잠금 모드 – 잠긴 객체에 대해 `Governance` 또는 `Compliance`로 설정합니다. 자세한 내용은 Amazon S3 객체 잠금을 사용하여 객체 잠금 (p. 490) 단원을 참조하십시오.
- 객체 잠금 법적 보존 상태 – 법적 보존이 객체에 적용된 경우 `On`, 그렇지 않으면 `Off`로 설정합니다. 자세한 내용은 Amazon S3 객체 잠금을 사용하여 객체 잠금 (p. 490) 단원을 참조하십시오.

오래된 인벤토리 목록을 삭제하는 수명 주기 정책을 생성할 것을 권장합니다. 자세한 내용은 객체 수명 주기 관리 (p. 123) 단원을 참조하십시오.

인벤토리 일관성

모든 객체가 각 인벤토리 목록에 나타나지 않을 수도 있습니다. 인벤토리 목록은 새 객체 및 덮어쓰기의 PUT에 대한 최종 일관성과 DELETE에 대한 최종 일관성을 제공합니다. 인벤토리 목록은 최종 일관성을 지닌 버킷 항목의 롤링 스냅샷입니다(즉, 최근에 추가되거나 삭제된 객체가 목록에 포함되지 않을 수 있습니다).

객체에 대한 작업을 수행하기 전에 객체 상태의 유효성을 검증하기 위해 객체의 메타데이터를 가져오도록 `HEAD Object` REST API 요청을 수행하거나 Amazon S3 콘솔에서 객체의 속성을 확인합니다. AWS CLI 또는 AWS SDKS를 사용하여 객체 메타데이터를 확인할 수도 있습니다. 자세한 내용은 Amazon Simple Storage Service API Reference의 HEAD Object를 참조하십시오.

인벤토리 목록의 위치는 어디입니까?

매니페스트 파일은 인벤토리 목록이 게시될 때 대상 버킷의 다음 위치에 게시됩니다.

```
destination-prefix/source-bucket/config-ID/YYYY-MM-DDTHH-MMZ/manifest.json
destination-prefix/source-bucket/config-ID/YYYY-MM-DDTHH-MMZ/manifest.checksum
destination-prefix/source-bucket/config-ID/hive/dt=YYYY-MM-DD-HH-MM/symlink.txt
```

- *`destination-prefix`*는 인벤토리 구성에서 설정하여 대상 버킷 내의 일반적인 위치에 있는 모든 인벤토리 목록 파일을 그룹화하는 데 사용할 수 있는 (객체 키 이름) 접두사입니다.
- *`source-bucket`*은 인벤토리 목록에 객체가 열거되는 버킷입니다. 여러 원본 버킷에서 복수의 인벤토리 보고서가 동일한 대상 버킷으로 전송되는 경우 충돌을 방지하기 위해 추가됩니다.
- *`config-ID`*는 동일한 원본 버킷에서 동일한 대상 버킷으로 복수의 인벤토리 보고서가 전송되는 경우 이들의 충돌을 방지하기 위해 추가됩니다. *`config-ID`*는 인벤토리 보고서 구성에서 비롯되며 설정에 정의된 보고서의 이름입니다.
- *`YYYY-MM-DDTHH-MMZ`*는 인벤토리 보고서 생성이 버킷 스캔을 시작한 시작 시간 및 날짜로 구성된 타임스탬프입니다(예: `2016-11-06T21-32Z`).
- `manifest.json`은 매니페스트 파일입니다.

- `manifest.checksum`은 `manifest.json` 파일 콘텐츠의 MD5입니다.
- `symlink.txt`는 Apache Hive 호환 매니페스트 파일입니다.

인벤토리 목록은 매일 또는 매주 대상 버킷의 다음 위치에 게시됩니다.

```
destination-prefix/source-bucket/config-ID/example-file-name.csv.gz
...
destination-prefix/source-bucket/config-ID/example-file-name-1.csv.gz
```

- *destination-prefix*는 인벤토리 구성에서 설정되는 (객체 키 이름) 접두사이며, 대상 버킷 내의 일반적인 위치에 있는 모든 인벤토리 목록 파일을 그룹화하는 데 사용할 수 있습니다.
- *source-bucket*은 인벤토리 목록에 객체가 열거되는 버킷입니다. 여러 원본 버킷에서 복수의 인벤토리 보고서가 동일한 대상 버킷으로 전송되는 경우 충돌을 방지하기 위해 추가됩니다.
- *example-file-name*`.csv.gz`는 CSV 인벤토리 파일의 하나입니다. ORC 인벤토리 이름은 파일 이름 확장자 `.orc`로 끝나고, Parquet 인벤토리 이름은 파일 이름 확장자 `.parquet`으로 끝납니다.

인벤토리 매니페스트는 무엇입니까?

매니페스트 파일 `manifest.json` 및 `symlink.txt`는 인벤토리 파일의 위치에 대해 설명합니다. 새 인벤토리 목록이 제공될 때마다 새 매니페스트 파일 세트가 함께 제공됩니다.

`manifest.json` 파일에 포함된 각 매니페스트에는 인벤토리에 대한 기타 기본 정보와 메타데이터가 들어 있습니다. 이 정보에는 다음 사항이 포함됩니다.

- 원본 버킷 이름
- 대상 버킷 이름
- 인벤토리의 버전
- 인벤토리 보고서 생성이 버킷 스캔을 시작한 시작 시간 및 날짜로 구성된 epoch 날짜 형식의 생성 타임스탬프
- 인벤토리 파일의 형식 및 스키마
- 대상 버킷에 있는 인벤토리 파일의 실제 목록

`manifest.json` 파일이 쓰여질 때마다 `manifest.checksum` 파일 콘텐츠의 MD5인 `manifest.json` 파일이 수반됩니다.

다음은 CSV 형식 인벤토리에 대해 `manifest.json` 파일에 있는 매니페스트의 예입니다.

```
{
    "sourceBucket": "example-source-bucket",
    "destinationBucket": "arn:aws:s3:::example-inventory-destination-bucket",
    "version": "2016-11-30",
    "creationTimestamp" : "1514944800000",
    "fileFormat": "CSV",
    "fileSchema": "Bucket, Key, VersionId, IsLatest, IsDeleteMarker, Size,
 LastModifiedDate, ETag, StorageClass, IsMultipartUploaded, ReplicationStatus,
 EncryptionStatus, ObjectLockRetainUntilDate, ObjectLockMode, ObjectLockLegalHoldStatus",
    "files": [
        {
            "key": "Inventory/example-source-bucket/2016-11-06T21-32Z/
files/939c6d46-85a9-4ba8-87bd-9db705a579ce.csv.gz",
            "size": 2147483647,
            "MD5checksum": "f11166069f1990abeb9c97ace9cdfabc"
        }
    ]
```

```
}
```

다음은 ORC 형식 인벤토리에 대해 `manifest.json` 파일에 있는 매니페스트의 예입니다.

```
{
    "sourceBucket": "example-source-bucket",
    "destinationBucket": "arn:aws:s3:::example-destination-bucket",
    "version": "2016-11-30",
    "creationTimestamp" : "1514944800000",
    "fileFormat": "ORC",
    "fileSchema":
 "struct<bucket:string,key:string,version_id:string,is_latest:boolean,is_delete_marker:boolean,size:big
    "files": [
        {
            "key": "inventory/example-source-bucket/data/
d794c570-95bb-4271-9128-26023c8b4900.orc",
            "size": 56291,
            "MD5checksum": "5925f4e78e1695c2d020b9f6eexample"
        }
    ]
}
```

다음은 Parquet 형식 인벤토리에 대해 `manifest.json` 파일에 있는 매니페스트의 예입니다.

```
{
    "sourceBucket": "example-source-bucket",
    "destinationBucket": "arn:aws:s3:::example-destination-bucket",
    "version": "2016-11-30",
    "creationTimestamp" : "1514944800000",
    "fileFormat": "Parquet",
    "fileSchema": "message s3.inventory { required binary bucket (UTF8); required binary
 key (UTF8); optional binary version_id (UTF8); optional boolean is_latest; optional
 boolean is_delete_marker;  optional int64 size;  optional int64 last_modified_date
 (TIMESTAMP_MILLIS);  optional binary e_tag (UTF8);  optional binary storage_class (UTF8);
  optional boolean is_multipart_uploaded;  optional binary replication_status (UTF8);
 optional binary encryption_status (UTF8);}"
  "files": [
        {
           "key": "inventory/example-source-bucket/data/
d754c470-85bb-4255-9218-47023c8b4910.parquet",
            "size": 56291,
            "MD5checksum": "5825f2e18e1695c2d030b9f6eexample"
        }
    ]
}
```

`symlink.txt` 파일은 Apache Hive 호환 매니페스트 파일이며, Hive에서 인벤토리 파일과 연결된 데이터 파일을 자동으로 검색할 수 있도록 해줍니다. Hive 호환 매니페스트는 Hive 호환 서비스인 Athena 및 Amazon Redshift Spectrum과 함께 작동합니다. 또한 Presto, Apache Hive, Apache Spark 등과 같은 Hive 호환 애플리케이션과 함께 작동합니다.

> **Important**
>
> `symlink.txt` Apache Hive 호환 매니페스트 파일은 현재 AWS Glue에서 작동하지 않습니다. Apache Hive 및 Apache Spark로 `symlink.txt` 읽기는 ORC 및 Parquet 형식 인벤토리 파일에 지원되지 않습니다.

인벤토리가 완료되는 시점을 어떻게 알 수 있습니까?

매니페스트 체크섬 파일이 생성될 때 인벤토리 목록이 대상 버킷에 추가되었음을 알려 주는 통보를 수신하도록 Amazon S3 이벤트 알림을 설정할 수 있습니다. 매니페스트는 대상 위치에 있는 모든 인벤토리 목록의 최신 목록입니다.

Amazon S3는 이벤트를 Amazon Simple Notification Service(Amazon SNS) 주제, Amazon Simple Queue Service(Amazon SQS) 대기열 또는 AWS Lambda 함수에 게시할 수 있습니다. 자세한 내용은 Amazon S3 이벤트 알림 구성 (p. 587) 단원을 참조하십시오.

다음의 알림 구성은 대상 버킷에 새롭게 추가된 모든 `manifest.checksum` 파일이 AWS Lambda `cloud-function-list-write`에 의해 처리되도록 구성합니다.

```
<NotificationConfiguration>
  <QueueConfiguration>
      <Id>1</Id>
      <Filter>
          <S3Key>
              <FilterRule>
                  <Name>prefix</Name>
                  <Value>destination-prefix/source-bucket</Value>
              </FilterRule>
              <FilterRule>
                  <Name>suffix</Name>
                  <Value>checksum</Value>
              </FilterRule>
          </S3Key>
     </Filter>
     <Cloudcode>arn:aws:lambda:us-west-2:222233334444:cloud-function-list-write</Cloudcode>
     <Event>s3:ObjectCreated:*</Event>
  </QueueConfiguration>
  </NotificationConfiguration>
```

자세한 내용은 AWS Lambda Developer Guide의 Amazon S3와 함께 AWS Lambda사용 단원을 참조하십시오.

Amazon Athena로 인벤토리 쿼리

Athena가 사용 가능한 모든 리전에서 Amazon Athena를 사용하여 표준 SQL로 Amazon S3 인벤토리를 쿼리할 수 있습니다. AWS 리전 가용성을 확인하려면 AWS 리전 표를 참조하십시오.

Athena는 ORC, Parquet 또는 CSV 형식 Amazon S3 인벤토리 파일을 쿼리할 수 있습니다. Athena를 사용하여 인벤토리를 쿼리할 경우 ORC 형식 또는 Parquet 형식 인벤토리 파일을 사용하는 것이 좋습니다. ORC 및 Parquet 형식은 더 빠른 쿼리 성능을 제공하지만 쿼리 비용이 더 저렴합니다. ORC 및 Parquet은 Apache 하둡용으로 설계된 자기 기술성 유형 인식 컬럼 방식의 파일 형식입니다. 이 컬럼 방식의 형식을 통해 리더가 현재 쿼리에 필요한 컬럼만 읽고 압축 해제하며 처리합니다. Amazon S3 인벤토리에 대한 ORC 및 Parquet 형식은 모든 AWS 리전에서 사용할 수 있습니다.

Athena를 사용하여 Amazon S3 인벤토리 쿼리를 시작하려면

1. Athena 테이블을 생성합니다. 테이블을 생성하는 방법에 대한 자세한 내용은 Amazon Athena 사용 설명서의 Amazon Athena에서 테이블 생성을 참조하십시오.

   다음 쿼리 예에는 ORC 형식 인벤토리 보고서의 모든 선택적 필드가 포함되어 있습니다. 쿼리가 인벤토리에 대해 선택한 필드와 일치하도록 인벤토리에 대해 선택하지 않은 선택적 필드를 삭제합니다. 또한 버킷 이름과 위치를 사용해야 합니다. 위치는 인벤토리 대상 경로(예: `s3://destination-prefix/source-bucket/config-ID/hive/`)를 가리킵니다.

```
CREATE EXTERNAL TABLE your_table_name(
  `bucket` string,
  key string,
  version_id string,
  is_latest boolean,
  is_delete_marker boolean,
  size bigint,
  last_modified_date timestamp,
```

```
  e_tag string,
  storage_class string,
  is_multipart_uploaded boolean,
  replication_status string,
  encryption_status string,
  object_lock_retain_until_date timestamp,
  object_lock_mode string,
  object_lock_legal_hold_status string
  )
  PARTITIONED BY (dt string)
  ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.orc.OrcSerde'
  STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
  OUTPUTFORMAT  'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'
  LOCATION 's3://destination-prefix/source-bucket/config-ID/hive/';
```

Athena를 사용하여 Parquet 형식 인벤토리 보고서를 쿼리하려면 `ROW FORMAT SERDE` 문에서 ORC SerDe 자리에 다음 Parquet SerDe를 사용하십시오.

```
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
```

2. 테이블에 새 인벤토리 목록을 추가하려면 다음 `MSCK REPAIR TABLE` 명령을 사용합니다.

```
MSCK REPAIR TABLE your-table-name;
```

3. 처음 두 단계를 수행한 후 다음 예제와 같이 인벤토리에 대해 임시 쿼리를 실행할 수 있습니다.

```
# Get list of latest inventory report dates available
SELECT DISTINCT dt FROM your-table-name ORDER BY 1 DESC limit 10;

# Get encryption status for a provided report date.
SELECT encryption_status, count(*) FROM your-table-name WHERE dt = 'YYYY-MM-DD-HH-MM'
 GROUP BY encryption_status;

# Get encryption status for report dates in the provided range.
SELECT dt, encryption_status, count(*) FROM your-table-name
WHERE dt > 'YYYY-MM-DD-HH-MM' AND dt < 'YYYY-MM-DD-HH-MM' GROUP BY dt,
 encryption_status;
```

Athena 사용에 관한 자세한 내용은 Amazon Athena 사용 설명서 단원을 참조하십시오.

Amazon S3 인벤토리 REST API

다음은 Amazon S3 인벤토리에 사용되는 REST 작업입니다.

- DELETE Bucket 인벤토리
- GET Bucket 인벤토리
- LIST Bucket 인벤토리
- PUT Bucket 인벤토리

Amazon S3의 복원성

AWS 글로벌 인프라는 리전 및 가용 영역을 중심으로 구축됩니다. AWS 리전에서는 물리적으로 분리되고 격리된 다수의 가용 영역을 제공하며 이러한 가용 영역은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크에 연결되어 있습니다. 이러한 가용 영역은 응용 프로그램과 데이터베이스를 설계하고 운영하는 효과적인 방법을 제공합니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성,

확장성이 뛰어납니다. 특히 데이터를 더 멀리 복제해야 하는 경우, 여러 AWS 리전의 버킷 간에 객체를 비동기식으로 자동 복사할 수 있게 해주는 복제 (p. 607)을 사용할 수 있습니다.

AWS 리전마다 여러 가용 영역이 있습니다. 내결함성과 짧은 지연 시간을 위해 같은 리전에 있는 여러 가용 영역에 애플리케이션을 배포할 수 있습니다. 가용 영역은 빠른 프라이빗 광섬유 네트워킹으로 서로 연결되어 있어 중단 없이 가용 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션을 손쉽게 설계할 수 있습니다.

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

AWS 글로벌 인프라 외에 Amazon S3도 데이터 복원성과 백업 요구 사항을 지원하는 다양한 기능을 제공합니다.

수명 주기 구성

수명 주기 구성은 Amazon S3이 객체 그룹에 적용되는 작업을 정의하는 규칙 세트입니다. 수명 주기 구성 규칙을 사용하면 Amazon S3이 객체를 더 저렴한 스토리지 클래스로 전환하거나 보관하거나 삭제하도록 유도할 수 있습니다. 자세한 내용은 객체 수명 주기 관리 (p. 123) 단원을 참조하십시오.

버전 관리

버전 관리는 동일 버킷 내에 여러 개의 객체 변형을 보유하는 것을 의미합니다. 버전 관리를 사용하여 Amazon S3 버킷에 저장된 모든 버전의 객체를 모두 보존, 검색 및 복원할 수 있습니다. 또한 의도치 않은 사용자 작업 및 애플리케이션 장애로부터 쉽게 복구할 수 있습니다. 자세한 내용은 버전 관리 사용 (p. 468) 단원을 참조하십시오.

Amazon S3 객체 잠금

Amazon S3 객체 잠금을 사용하면 WORM(write once, read many) 모델을 사용하여 객체를 저장할 수 있습니다. Amazon S3 객체 잠금을 사용하면 일정 시간 동안 또는 무기한으로 객체가 삭제되거나 덮어쓰이지 않도록 방지할 수 있습니다. Amazon S3 객체 잠금은 WORM 스토리지가 필요한 규제 요구 사항을 충족하거나 객체 변경 및 삭제에 대한 보호 계층을 추가하는 데 도움이 됩니다. 자세한 내용은 Amazon S3 객체 잠금을 사용하여 객체 잠금 (p. 490) 단원을 참조하십시오.

스토리지 클래스

Amazon S3는 저장된 객체에 대해 다양한 스토리지 클래스를 제공합니다. 두 스토리지 클래스 (STANDARD_IA 및 ONEZONE_IA)는 백업처럼 수명이 길고 자주 액세스하지 않는 데이터를 위해 설계되었습니다. S3 Glacier 스토리지 클래스를 사용하여 실시간으로 액세스할 필요가 없는 객체를 보관할 수도 있습니다. 자세한 내용은 Amazon S3 스토리지 클래스 (p. 107) 단원을 참조하십시오.

다음 보안 모범 사례에서도 복원성을 다룹니다.

- 버전 관리 사용
- Amazon S3 교차 리전 복제 고려
- 모든 Amazon S3 버킷 식별 및 감사

Amazon S3 백업 암호화

Amazon S3을 사용하여 백업을 저장하는 경우 백업 암호화는 해당 버킷의 구성에 따라 결정됩니다. Amazon S3에서는 S3 버킷의 기본 암호화 동작을 설정하는 방법을 제공합니다. 버킷에 저장되는 모든 객체를 암호화하도록 버킷에 대한 기본 암호화를 설정할 수 있습니다. 기본 암호화는 AWS KMS(SSE-KMS)에 저장된 키를 지원합니다. 자세한 내용은 S3 버킷에 대한 Amazon S3 기본 암호화 (p. 65) 단원을 참조하십시오.

버전 관리 사용

버전 관리는 동일 버킷 내에 여러 개의 객체 변형을 보유하는 것을 의미합니다. 버전 관리를 사용하면 Amazon S3 버킷에 저장된 모든 버전의 모든 객체를 보존, 검색 및 복원할 수 있습니다. 또한 의도치 않은

사용자 작업 및 애플리케이션 장애로부터 쉽게 복구할 수 있습니다. 버킷에 대해 버전 관리를 활성화하면 Amazon S3가 동일한 객체에 대해 여러 쓰기 요청을 동시에 수신하는 경우 모든 객체가 저장됩니다.

버킷에 대한 버전 관리를 활성화하면 Amazon S3는 저장되는 객체에 대해 고유한 버전 ID를 자동으로 생성합니다. 예를 들어, `photo.gif`(버전 111111) 및 `photo.gif` (버전 121212)와 같이 하나의 버킷에서 키는 동일하지만 버전 ID가 다른 두 개의 객체를 보유할 수 있습니다.

버전 관리를 사용하는 버킷을 통해 실수로 삭제되거나 덮어쓴 객체를 복구할 수 있습니다. 다음 예를 참조하십시오.

- 객체를 영구적으로 제거하는 대신 삭제하는 경우, Amazon S3에서는 삭제 마커를 삽입하고 이는 현재 객체 버전이 됩니다. 따라서 언제든지 이전 버전을 복원할 수 있습니다. 자세한 내용은 객체 버전 삭제 (p. 481) 단원을 참조하십시오.
- 객체를 덮어쓴 경우 버킷에 새 객체 버전이 생깁니다. 따라서 언제든지 이전 버전을 복원할 수 있습니다.

> **Important**
>
> 버전이 지정되지 않은 버킷에 객체 만료 수명 주기 정책이 있고 버전 관리를 사용할 때 같은 영구 삭제 행동을 유지하고자 하는 경우에는 비 최신 버전의 만료 정책을 추가해야 합니다. 비 최신 버전의 만료 수명 주기 정책은 버전 관리를 사용하는 버킷에서 비 최신 객체 버전의 삭제를 관리합니다. (버전 관리가 활성화된 버킷은 하나의 현재 객체 버전과 0개 이상의 최신이 아닌 객체 버전을 유지합니다.) 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷에 대한 수명 주기 정책을 생성하려면 어떻게 해야 합니까?를 참조하십시오.

버킷은 버전 관리 미사용(기본값), 버전 관리 활성화 또는 버전 관리 일시 중지의 세 가지 상태 중 하나가 될 수 있습니다.

> **Important**
>
> 버킷의 버전 관리를 활성화한 다음에는 버전 관리 미사용 상태로 돌아갈 수 없습니다. 그러나 해당 버킷에서 버전 관리를 일시 중지할 수는 있습니다.

버전 관리 상태는 해당 버킷의 전체 객체에 적용되며 일부에만 적용할 수는 없습니다. 처음에 버킷의 버전 관리를 활성화하면 그 이후로는 이 버킷 내의 객체는 항상 버전 관리를 사용하게 되며 고유한 버전 ID가 부여됩니다. 다음을 참조하십시오.

- 버전 관리 상태로 설정하기 전에 버킷에 저장된 객체의 버전 ID는 `null`이 됩니다. 버전 관리를 활성화할 때 버킷의 기존 객체는 변경되지 않습니다. 이후 요청에서 Amazon S3가 객체를 처리하는 방법만 변경됩니다. 자세한 내용은 버전 관리를 사용하는 버킷의 객체 관리 (p. 474) 단원을 참조하십시오.
- 버킷 소유자(또는 적절한 권한을 가진 사용자)는 버전 관리를 일시 중지하여 객체 버전 발생을 중단할 수 있습니다. 버전 관리를 일시 중지할 때 버킷의 기존 객체는 변경되지 않습니다. 이후 요청에서 Amazon S3가 객체를 처리하는 방법만 변경됩니다. 자세한 내용은 버전 관리가 일시 중지된 버킷의 객체 관리 (p. 487) 단원을 참조하십시오.

버킷의 버전 관리를 구성하는 방법

다음 방법 중 하나를 사용하여 버킷 버전 관리를 구성할 수 있습니다.

- Amazon S3 콘솔을 사용하여 버전 관리를 구성합니다.
- AWS SDK를 사용하여 프로그래밍 방식으로 버전 관리를 구성합니다.

콘솔 및 SDK 모두 Amazon S3가 버전 관리 기능을 관리하기 위해 제공하는 REST API를 호출합니다.

> **Note**
>
> 필요한 경우 코드에서 직접 Amazon S3 REST API를 호출하게 할 수도 있습니다. 하지만 이 방법은 요청 인증을 위한 코드를 작성해야 하므로 번거로울 수 있습니다.

생성한 각 버킷에는 버전 관리 하위 리소스가 연결되어 있습니다(버킷 구성 옵션 (p. 55) 참조). 기본적으로 버킷은 버전 관리 미사용 상태이며, 따라서 버전 관리 하위 리소스는 빈 버전 관리 구성을 저장합니다.

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
</VersioningConfiguration>
```

버전 관리를 활성화하려면 Amazon S3에 상태가 포함된 버전 관리 구성을 사용하여 요청을 보냅니다.

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>Enabled</Status>
</VersioningConfiguration>
```

버전 관리를 일시 중지하려면 상태 값을 `Suspended`로 설정합니다.

버킷 소유자, 버킷을 생성한 AWS 계정(루트 계정), 인증된 사용자는 버킷의 버전 관리 상태를 구성할 수 있습니다. 권한에 대한 자세한 내용은 Amazon S3의 Identity and Access Management (p. 299) 단원을 참조하십시오.

버전 관리 구성의 예는 버킷 버전 관리 활성화의 예제 (p. 471) 단원을 참조하십시오.

MFA Delete

선택적으로 MFA(멀티 팩터 인증) Delete를 활성화하도록 버킷을 구성하여 다른 보안 계층을 추가할 수 있습니다. 이 경우, 다음 작업 중 하나에 대한 추가 인증이 필요합니다:

- 버킷의 버전 관리 상태 변경
- 객체 버전 영구 삭제

MFA Delete는 다음 두 가지 인증 유형이 모두 필요합니다.

- 보안 자격 증명
- 승인된 인증 디바이스에 표시된 유효한 일련 번호, 공백, 6자리 코드 연결

MFA Delete는 이러한 추가된 보안을 보안 자격 증명이 손상된 경우 등에 제공합니다.

MFA Delete를 활성화 또는 비활성화하려면 버킷에 대한 버전 관리를 구성하는 데 사용한 것과 동일한 API를 사용합니다. Amazon S3는 버킷의 버전 관리 상태를 저장하는 동일한 versioning 하위 리소스에 MFA delete 구성을 저장합니다.

MFA 삭제를 사용하면 다음을 수행하여 잘못해서 버킷을 삭제하는 일을 방지할 수 있습니다.

- 삭제 작업을 시작하는 사용자에게 MFA 코드를 사용하여 MFA 디바이스의 물리적 소유를 입증하도록 요청합니다.
- 삭제 작업에 별도의 마찰 및 보안 계층을 추가합니다.

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
```

```
  <Status>VersioningState</Status>
  <MfaDelete>MfaDeleteState</MfaDelete>
</VersioningConfiguration>
```

> **Note**
>
> 버킷 소유자, 버킷을 생성한 AWS 계정(루트 계정), 모든 인증된 IAM 사용자는 버전 관리를 활성화할 수 있지만, 버킷 소유자(루트 계정)만 MFA Delete를 활성화할 수 있습니다. 자세한 내용은 MFA 삭제 및 버전 관리에 대한 AWS 블로그 게시물을 참조하십시오.

MFA Delete를 사용하려면 하드웨어 또는 가상 MFA 디바이스 중에 하나를 사용하여 인증 코드를 생성할 수 있습니다. 다음은 생성된 인증 코드가 하드웨어 디바이스에 표시된 것을 보여 주는 예제입니다.

> **Note**
>
> MFA Delete 및 MFA 보호 API 액세스는 서로 다른 시나리오에 대한 보호를 제공하기 위해 마련된 기능입니다. 버킷의 데이터가 실수로 삭제될 수 없도록 하려면 버킷에 MFA Delete를 구성합니다. MFA 보호 API 액세스는 민감한 Amazon S3 리소스에 액세스할 때 다른 인증 요소(MFA 코드)를 적용하기 위해 사용됩니다. 이러한 Amazon S3 리소스에 대한 모든 작업은 MFA를 사용하여 생성한 임시 자격 증명으로 수행되도록 요구할 수 있습니다. 문제 해결 예는 MFA가 필요한 버킷 정책 추가 (p. 411) 단원을 참조하십시오.
> 인증 디바이스를 구입하고 활성화하는 방법에 대한 자세한 내용은 https://aws.amazon.com/iam/details/mfa/를 참조하십시오.

관련 주제

자세한 내용은 다음 항목을 참조하십시오.

- 버킷 버전 관리 활성화의 예제 (p. 471)
- 버전 관리를 사용하는 버킷의 객체 관리 (p. 474)
- 버전 관리가 일시 중지된 버킷의 객체 관리 (p. 487)
- 버전 관리를 사용하는 버킷에 대한 Amazon S3 요청에 대해 HTTP 503 응답이 크게 증가 (p. 701)

버킷 버전 관리 활성화의 예제

주제

- Amazon S3 콘솔 사용 (p. 471)
- Java용 AWS SDK 사용 (p. 472)
- .NET용 AWS SDK 사용 (p. 472)
- 다른 AWS SDK 사용 (p. 474)

이 단원에서는 버킷의 버전 관리를 활성화하는 예제를 제공합니다. 예제에서는 먼저 버킷에 대한 버전 관리를 활성화한 다음 버전 관리 상태를 가져옵니다. 버전 관리에 관한 소개 부분은 버전 관리 사용 (p. 468) 단원을 참조하십시오.

Amazon S3 콘솔 사용

Amazon S3 콘솔을 사용하여 버킷의 버전 관리를 활성화하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷의 버전 관리 기능을 활성화하거나 일시 중지하려면 어떻게 해야 합니까? 단원을 참조하십시오.

Java용 AWS SDK 사용

Example

실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
import java.io.IOException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.AmazonS3Exception;
import com.amazonaws.services.s3.model.BucketVersioningConfiguration;
import com.amazonaws.services.s3.model.SetBucketVersioningConfigurationRequest;

public class BucketVersioningConfigurationExample {
    public static String bucketName = "*** bucket name ***";
    public static AmazonS3Client s3Client;

    public static void main(String[] args) throws IOException {
        s3Client = new AmazonS3Client(new ProfileCredentialsProvider());
        s3Client.setRegion(Region.getRegion(Regions.US_EAST_1));
        try {

            // 1. Enable versioning on the bucket.
         BucketVersioningConfiguration configuration =
           new BucketVersioningConfiguration().withStatus("Enabled");

   SetBucketVersioningConfigurationRequest setBucketVersioningConfigurationRequest =
     new SetBucketVersioningConfigurationRequest(bucketName,configuration);

   s3Client.setBucketVersioningConfiguration(setBucketVersioningConfigurationRequest);

   // 2. Get bucket versioning configuration information.
   BucketVersioningConfiguration conf =
 s3Client.getBucketVersioningConfiguration(bucketName);
    System.out.println("bucket versioning configuration status:    " + conf.getStatus());

        } catch (AmazonS3Exception amazonS3Exception) {
            System.out.format("An Amazon S3 error occurred. Exception: %s",
 amazonS3Exception.toString());
        } catch (Exception ex) {
            System.out.format("Exception: %s", ex.toString());
        }
    }
}
```

.NET용 AWS SDK 사용

실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

Example

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

namespace s3.amazon.com.docsamples
```

```
{
    class BucketVersioningConfiguration
    {
        static string bucketName = "*** bucket name ***";

        public static void Main(string[] args)
        {
            using (var client = new AmazonS3Client(Amazon.RegionEndpoint.USEast1))
            {
                try
                {
                    EnableVersioningOnBucket(client);
                    string bucketVersioningStatus =
 RetrieveBucketVersioningConfiguration(client);
                }
                catch (AmazonS3Exception amazonS3Exception)
                {
                    if (amazonS3Exception.ErrorCode != null &&
                        (amazonS3Exception.ErrorCode.Equals("InvalidAccessKeyId")
                        ||
                        amazonS3Exception.ErrorCode.Equals("InvalidSecurity")))
                    {
                        Console.WriteLine("Check the provided AWS Credentials.");
                        Console.WriteLine(
                        "To sign up for service, go to http://aws.amazon.com/s3");
                    }
                    else
                    {
                        Console.WriteLine(
                         "Error occurred. Message:'{0}' when listing objects",
                         amazonS3Exception.Message);
                    }
                }
            }

            Console.WriteLine("Press any key to continue...");
            Console.ReadKey();
        }

        static void EnableVersioningOnBucket(IAmazonS3 client)
        {

                PutBucketVersioningRequest request = new PutBucketVersioningRequest
                {
                    BucketName = bucketName,
                    VersioningConfig = new S3BucketVersioningConfig
                    {
                        Status = VersionStatus.Enabled
                    }
                };

                PutBucketVersioningResponse response = client.PutBucketVersioning(request);
        }


        static string RetrieveBucketVersioningConfiguration(IAmazonS3 client)
        {
                GetBucketVersioningRequest request = new GetBucketVersioningRequest
                {
                    BucketName = bucketName
                };

                GetBucketVersioningResponse response = client.GetBucketVersioning(request);
                return response.VersioningConfig.Status;
            }
    }
```

```
}
```

다른 AWS SDK 사용

다른 AWS SDK 사용에 대한 자세한 내용은 Sample Code and Libraries 단원을 참조하십시오.

버전 관리를 사용하는 버킷의 객체 관리

주제

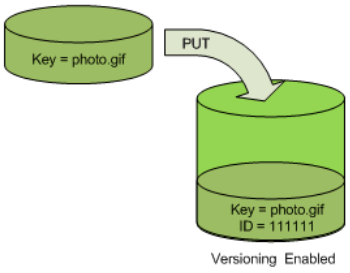
- 버전 관리를 사용하는 버킷에 객체 추가 (p. 474)
- 버전 관리를 사용하는 버킷의 객체 목록 (p. 475)
- 객체 버전 가져오기 (p. 479)
- 객체 버전 삭제 (p. 481)
- 객체 버전 이전 (p. 486)
- 이전 버전 복원 (p. 486)
- 버전이 지정된 객체의 권한 (p. 486)

버전 관리 상태로 설정하기 전에 버킷에 저장된 객체의 버전 ID는 `null`이 됩니다. 버전 관리를 활성화할 때 버킷의 기존 객체는 변경되지 않습니다. 이후 요청에서 Amazon S3가 객체를 처리하는 방법만 변경됩니다. 이 단원의 주제에서는 버전 관리를 사용하는 버킷의 다양한 객체 작업에 관해 설명합니다.

버전 관리를 사용하는 버킷에 객체 추가

버킷에 대한 버전 관리를 활성화하면 Amazon S3에서는 자동으로 고유한 버전 ID를 버킷에 저장된 각 객체에 추가합니다(`PUT`, `POST` 또는 `COPY` 사용).

다음 그림은 버전 관리를 사용하는 버킷에 객체를 추가할 때 Amazon S3에서 고유한 버전 ID를 객체에 추가하는 것을 보여 줍니다.



주제

- 콘솔 사용 (p. 474)
- AWS SDK 사용 (p. 474)
- REST API 사용 (p. 475)

콘솔 사용

지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷에 객체를 업로드하려면 어떻게 해야 합니까? 단원을 참조하십시오.

AWS SDK 사용

Java, .NET, PHP용 AWS SDK를 사용한 객체 업로드 예제는 객체 업로드 (p. 172) 단원을 참조하십시오. 버전 관리 미사용 버킷과 버전 관리를 사용하는 버킷에 객체를 업로드하는 것의 예제는 동일하며, 버전 관리를

사용하는 버킷인 경우라도 Amazon S3에서는 버전 번호를 할당합니다. 그렇지 않은 경우 버전 번호는 null이 됩니다.

다른 AWS SDK 사용에 대한 자세한 내용은 Sample Code and Libraries 단원을 참조하십시오.

REST API 사용

버전 관리를 사용하는 버킷에 객체 추가

1	`PUT Bucket versioning` 요청을 사용하여 버킷의 버전 관리를 활성화합니다. 자세한 내용은 PUT Bucket versioning 단원을 참조하십시오.
2	`PUT`, `POST` 또는 `COPY` 요청을 보내 버킷에 객체를 저장합니다.

버전 관리를 사용하는 버킷에 객체를 추가할 경우, Amazon S3에서는 객체의 버전 ID를 `x-amz-version-id` 응답 헤더로 반환합니다. 예:

```
x-amz-version-id: 3/L4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY
```

> **Note**
>
> 저장되거나 전송된 객체의 모든 버전에는 일반 Amazon S3 요금이 적용됩니다. 각 객체 버전은 이전 버전과의 차이점만이 아닌 완전한 객체입니다. 따라서 세 가지 버전의 객체가 저장된 경우 객체 세 개에 대한 요금이 부과됩니다.

> **Note**
>
> Amazon S3에서 할당하는 버전 ID 값은 URL로 사용 가능합니다(URI의 일부로 포함할 수 있음).

버전 관리를 사용하는 버킷의 객체 목록

주제

- 콘솔 사용 (p. 475)
- AWS SDK 사용 (p. 475)
- REST API 사용 (p. 478)

이 단원에서는 버전 관리를 사용하는 버킷의 객체 버전을 나열하는 예제를 제공합니다. Amazon S3는 버킷과 연결된 versions 하위 리소스(버킷 구성 옵션 (p. 55) 참조)에 객체 버전 정보를 저장합니다.

콘솔 사용

Amazon S3 콘솔을 사용하여 객체 버전을 나열하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의S3 객체의 버전을 보려면 어떻게 해야 합니까? 단원을 참조하십시오.

AWS SDK 사용

이 단원의 예제에서는 버전 관리를 사용하는 버킷에서 객체 목록을 가져오는 방법을 보여줍니다. 더 낮은 개수를 지정했더라도 각 요청에서는 최대 1,000개의 버전을 반환합니다. 버킷에 이 제한보다 더 많은 버전이 포함되어 있는 경우 일련의 요청을 보내 모든 버전 목록을 가져옵니다. "페이지"에 표시되는 이러한 결과 반환 프로세스를 페이지 매김이라고 합니다. 페이지 매김 작동 방식을 설명하기 위해 예제에서는 각 응답을 두 가지 객체 버전으로 제한합니다. 결과의 첫 페이지를 가져온 후 각 예제에서는 버전 목록이 잘렸는지 여부를 확인합니다. 잘린 경우 계속해서 모든 버전을 가져올 때까지 페이지를 가져옵니다.

> **Note**
>
> 다음 예제 역시 버전 관리를 사용하지 않는 버킷이나 개별 버전이 없는 객체를 사용합니다. 이러한 경우 Amazon S3는 버전 ID `null`을 사용하여 버전 목록을 반환합니다.

다른 AWS SDK 사용에 대한 자세한 내용은 Sample Code and Libraries 단원을 참조하십시오.

AWS SDK for Java 사용

실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

Example

```
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;
import com.amazonaws.services.s3.model.ListVersionsRequest;
import com.amazonaws.services.s3.model.S3VersionSummary;
import com.amazonaws.services.s3.model.VersionListing;

public class ListKeysVersioningEnabledBucket {

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

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Retrieve the list of versions. If the bucket contains more versions
            // than the specified maximum number of results, Amazon S3 returns
            // one page of results per request.
            ListVersionsRequest request = new ListVersionsRequest()
                    .withBucketName(bucketName)
                    .withMaxResults(2);
            VersionListing versionListing = s3Client.listVersions(request);
            int numVersions = 0, numPages = 0;
            while (true) {
                numPages++;
                for (S3VersionSummary objectSummary :
                        versionListing.getVersionSummaries()) {
                    System.out.printf("Retrieved object %s, version %s\n",
                            objectSummary.getKey(),
                            objectSummary.getVersionId());
                    numVersions++;
                }
                // Check whether there are more pages of versions to retrieve. If
                // there are, retrieve them. Otherwise, exit the loop.
                if (versionListing.isTruncated()) {
                    versionListing = s3Client.listNextBatchOfVersions(versionListing);
                } else {
                    break;
                }
            }
            System.out.println(numVersions + " object versions retrieved in " + numPages +
 " pages");
        } 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();
```

```
        }
    }
}
```

.NET용 AWS SDK 사용

실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

Example

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

namespace Amazon.DocSamples.S3
{
    class ListObjectsVersioningEnabledBucketTest
    {
        static 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(string[] args)
        {
            s3Client = new AmazonS3Client(bucketRegion);
            GetObjectListWithAllVersionsAsync().Wait();
        }

        static async Task GetObjectListWithAllVersionsAsync()
        {
            try
            {
                ListVersionsRequest request = new ListVersionsRequest()
                {
                    BucketName = bucketName,
                    // You can optionally specify key name prefix in the request
                    // if you want list of object versions of a specific object.

                    // For this example we limit response to return list of 2 versions.
                    MaxKeys = 2
                };
                do
                {
                    ListVersionsResponse response = await
 s3Client.ListVersionsAsync(request);
                    // Process response.
                    foreach (S3ObjectVersion entry in response.Versions)
                    {
                        Console.WriteLine("key = {0} size = {1}",
                            entry.Key, 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 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);
            }
        }
    }
}
```

REST API 사용

버킷에 있는 모든 객체의 모든 버전을 나열하려면 `GET Bucket` 요청에 `versions` 하위 리소스를 사용합니다. Amazon S3는 최대 1,000개의 객체만 가져올 수 있으며, 각 객체 버전은 각각 하나의 객체로 온전히 취급됩니다. 따라서 버킷에 두 개의 키(예: `photo.gif` 및 `picture.jpg`)가 포함되어 있고 첫 번째 키에 990개의 버전이, 두 번째 키에 400개의 버전이 있는 경우, 단일 요청으로는 `photo.gif`의 990개 버전 모드를 가져오고 `picture.jpg`의 최신 버전 10개만 가져오게 됩니다.

Amazon S3에서는 객체 버전을 객체가 저장된 순서대로 가장 최근에 저장된 것을 먼저 반환합니다.

버킷의 전체 객체 버전 목록을 보려면

- `GET Bucket` 요청에 `versions` 하위 리소스를 포함합니다.

```
GET /?versions HTTP/1.1
Host: bucketName.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 +0000
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

버킷에서 객체의 하위 집합 가져오기

이 단원에서는 다음 두 가지 예제 시나리오를 다룹니다.

- 예를 들어, 버킷에서 전체 객체 버전의 하위 집합을 가져오려는 경우 특정 객체의 전체 버전을 가져옵니다.
- 응답의 객체 버전 수가 `max-key`의 값(기본값은 1,000)을 초과하므로 나머지 객체 버전을 가져오기 위한 두 번째 요청을 제출해야 합니다.

객체 버전의 하위 집합을 가져오려면 GET Bucket에 요청 파라미터를 사용합니다. 자세한 내용은 GET Bucket 단원을 참조하십시오.

예제 1: 특정 객체 하나의 전체 버전 가져오기

다음 프로세스에 따라 `versions` 하위 리소스와 `prefix` 요청 파라미터를 사용하여 객체의 전체 버전을 가져올 수 있습니다. `prefix`에 대한 자세한 내용은 GET Bucket 단원을 참조하십시오.

키의 전체 버전 가져오기

1	`prefix` 파라미터를 가져오려는 객체의 키에 설정합니다.

2	GET Bucket 하위 리소스 및 versions를 사용하여 prefix 요청을 보냅니다. GET /?versions&prefix=objectName HTTP/1.1

Example 접두사를 사용하여 객체 가져오기

다음 예제에서는 키가 myObject이거나 이렇게 시작하는 객체를 가져옵니다.

```
GET /?versions&prefix=myObject HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

기타 요청 파라미터를 사용하여 전체 객체 버전의 하위 집합을 가져올 수 있습니다. 자세한 내용은 GET Bucket 단원을 참조하십시오.

예제 2: 응답이 잘린 경우 추가 객체의 목록 가져오기

GET 요청에서 반환될 수 있는 객체 수가 max-keys 값을 초과한 경우, 응답에는 <isTruncated>true</isTruncated>가 포함되며 요청을 충족하지만 반환되지 않은 첫 번째 키와 첫 번째 버전 ID를 각각 NextKeyMarker와 NextVersionIdMarker에 포함시킵니다. 그러한 반환된 값을 후속 요청의 시작 지점으로 사용하여 GET 요청을 충족하는 추가 객체를 가져옵니다.

다음 프로세스를 사용하여 버킷에서 GET Bucket versions 요청을 충족하는 추가 객체를 가져옵니다. key-marker, version-id-marker, NextKeyMarker 및 NextVersionIdMarker에 대한 자세한 내용은 GET Bucket 단원을 참조하십시오.

원본 GET 요청을 충족하는 추가 응답 가져오기

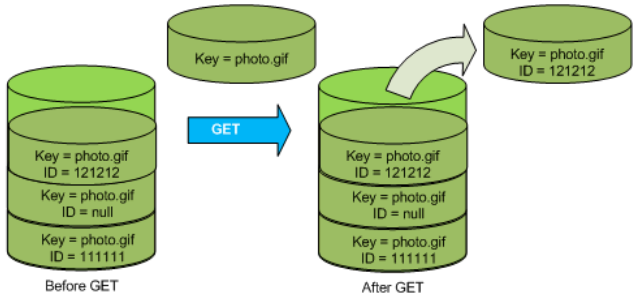
1	key-marker의 값을 이전 응답의 NextKeyMarker에서 반환된 키로 설정합니다.
2	version-id-marker의 값을 이전 응답의 NextVersionIdMarker에서 반환된 버전 ID로 설정합니다.
3	GET Bucket versions 및 key-marker를 사용하여 version-id-marker 요청을 보냅니다.

Example 지정된 키 및 버전 ID로 시작하는 객체 가져오기

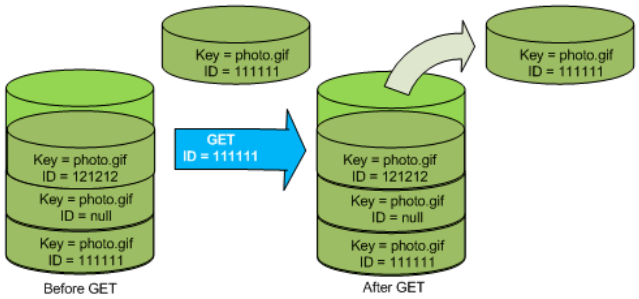
```
GET /?versions&key-marker=myObject&version-id-marker=298459348571 HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

객체 버전 가져오기

단순한 GET 요청은 현재 버전의 객체를 가져옵니다. 다음 그림은 GET 요청에서 현재 버전의 객체인 photo.gif를 반환하는 과정을 보여 줍니다.



특정 버전을 가져오려면 해당 버전 ID를 지정해야 합니다. 다음 그림은 `GET versionId` 요청에서 특정 버전의 객체(현재 버전이 아니어도 됨)를 가져오는 과정을 보여 줍니다.



콘솔 사용

지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 객체의 버전을 보려면 어떻게 해야 합니까? 단원을 참조하십시오.

AWS SDK 사용

Java, .NET, PHP용 AWS SDK를 사용한 객체 업로드 예제는 객체 가져오기 (p. 164) 단원을 참조하십시오. 버전 관리 미사용 버킷과 버전 관리를 사용하는 버킷에 객체를 업로드하는 것의 예제는 동일하며, 버전 관리를 사용하는 버킷인 경우라도 Amazon S3에서는 버전 번호를 할당합니다. 그렇지 않은 경우 버전 번호는 null이 됩니다.

다른 AWS SDK 사용에 대한 자세한 내용은 Sample Code and Libraries 단원을 참조하십시오.

REST 사용

특정 객체 버전을 가져오려면

1. `versionId`를 가져오려는 객체의 버전 ID에 설정합니다.
2. `GET Object versionId` 요청을 보냅니다.

Example 버전이 지정된 객체 가져오기

다음 요청은 `my-image.jpg`의 L4kqtJlcpXroDTDmpUMLUo 버전을 가져옵니다.

```
GET /my-image.jpg?versionId=L4kqtJlcpXroDTDmpUMLUo HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

관련 주제

객체 버전의 메타데이터 가져오기 (p. 481)

객체 버전의 메타데이터 가져오기

객체의 메타데이터만 가져오고 해당 콘텐츠는 가져오지 않으려는 경우 `HEAD` 작업을 사용합니다. 기본적으로 가장 최근 버전의 메타데이터를 가져옵니다. 특정 객체 버전의 메타데이터를 가져오려면 해당 버전 ID를 지정해야 합니다.

객체 버전의 메타데이터를 가져오려면

1. 메타데이터를 가져오려는 객체의 버전 ID에 `versionId`를 설정합니다.
2. `HEAD Object versionId` 요청을 보냅니다.

Example 버전이 지정된 객체의 메타데이터를 가져오려면

다음 요청은 3HL4kqCxf3vjVBH40Nrjfkd 버전 `my-image.jpg`의 메타데이터를 가져옵니다.

```
HEAD /my-image.jpg?versionId=3HL4kqCxf3vjVBH40Nrjfkd HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

다음은 샘플 응답을 보여 줍니다.

```
HTTP/1.1 200 OK
x-amz-id-2: ef8yU9AS1ed4OpIszj7UDNEHGran
x-amz-request-id: 318BC8BC143432E5
x-amz-version-id: 3HL4kqtJlcpXroDTDmjVBH40Nrjfkd
Date: Wed, 28 Oct 2009 22:32:00 GMT
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
ETag: "fba9dede5f27731c9771645a39863328"
Content-Length: 434234
Content-Type: text/plain
Connection: close
Server: AmazonS3
```

객체 버전 삭제

언제든지 원할 때 객체 버전을 삭제할 수 있습니다. 또한 수명 주기가 명확하게 정의된 객체에 대해 수명 주기 구성 규칙을 정의하여 Amazon S3에 현재 객체 버전을 만료시키거나 최신이 아닌 객체 버전을 영구적으로 제거하도록 요청할 수도 있습니다. 버킷에서 버전 관리가 활성화 또는 일시 중지된 경우 수명 주기 구성 작업은 다음과 같이 작동합니다.

- `Expiration` 작업이 현재 객체 버전에 적용되고 현재 객체 버전을 삭제하는 대신 Amazon S3에서는 나중에 현재 버전이 되는 삭제 마커를 추가하여 현재 버전을 최신이 아닌 버전으로 유지합니다.
- `NoncurrentVersionExpiration` 작업이 최신이 아닌 객체 버전에 적용되고 Amazon S3에서는 이러한 객체 버전을 영구적으로 제거합니다. 영구적으로 제거된 객체는 복구할 수 없습니다.

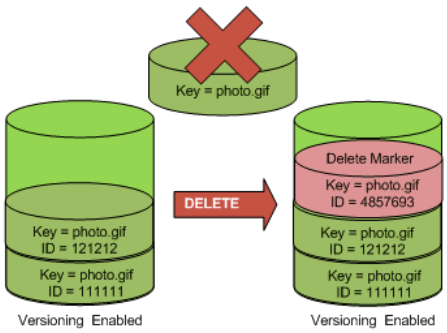
자세한 내용은 객체 수명 주기 관리 (p. 123) 단원을 참조하십시오.

`DELETE` 요청의 사용 사례는 다음과 같습니다.

- 버전 관리가 활성화된 경우, 단순 `DELETE` 요청은 객체를 영구적으로 삭제하지 않습니다.

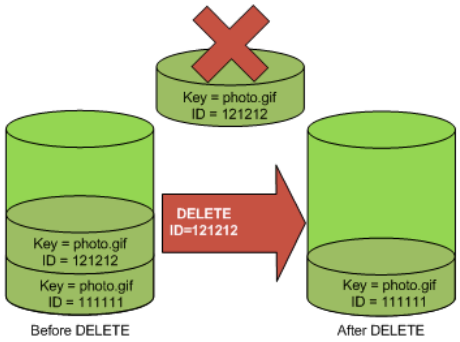
  대신, Amazon S3는 삭제 마커를 버킷에 삽입하고 해당 마커는 새로운 ID를 가진 객체의 현재 버전이 됩니다. 현재 버전이 삭제 마커인 객체에 대해 `GET`을 시도하는 경우, Amazon S3는 객체가 삭제된 것처럼 동작하여 404 오류를 반환합니다.

  다음 그림은 단순한 `DELETE` 요청이 지정된 객체를 실제로 제거하지 않음을 보여 줍니다. 대신 Amazon S3에서 삭제 마커를 삽입합니다.



- 버전이 지정된 객체를 영구적으로 삭제하려면 `DELETE Object versionId`를 사용해야 합니다.

  다음 그림은 지정된 객체 버전을 삭제하면 해당 객체가 영구적으로 제거됨을 보여 줍니다.



콘솔 사용

지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 객체의 버전을 보려면 어떻게 해야 합니까? 단원을 참조하십시오.

AWS SDK 사용

Java, .NET, PHP용 AWS SDK를 사용하여 객체를 삭제하는 예는 객체 삭제 (p. 225) 단원을 참조하십시오. 버전 관리 미사용 버킷과 버전 관리를 사용하는 버킷에서 객체를 삭제하는 예는 동일하며, 버전 관리를 사용하는 버킷인 경우라도 Amazon S3에서는 버전 번호를 할당합니다. 그렇지 않은 경우 버전 번호는 null이 됩니다.

다른 AWS SDK 사용에 대한 자세한 내용은 Sample Code and Libraries 단원을 참조하십시오.

REST 사용

특정 버전의 객체를 삭제하려면

- `DELETE`에서 버전 ID를 지정합니다.

Example 특정 버전 삭제

다음 예제에서는 UIORUnfnd89493jJFJ 버전의 `photo.gif`를 삭제하는 방법을 보여 줍니다.

```
DELETE /photo.gif?versionId=UIORUnfnd89493jJFJ HTTP/1.1

Host: bucket.s3.amazonaws.com
Date: Wed, 12 Oct 2009 17:50:00 GMT
```

```
Authorization: AWS AKIAIOSFODNN7EXAMPLE:xQE0diMbLRepdf3YB+FIEXAMPLE=
Content-Type: text/plain
Content-Length: 0
```

관련 주제

MFA Delete 사용 (p. 483)

삭제 마커를 통한 작업 (p. 483)

삭제 마커 제거 (p. 484)

버전 관리 사용 (p. 468)

MFA Delete 사용

버킷의 버전 관리 구성에 MFA Delete가 활성화된 경우, 버킷 소유자는 요청에 `x-amz-mfa` 요청 헤더를 포함하여 객체 버전을 영구적으로 삭제하거나 버킷의 버전 관리 상태를 변경해야 합니다. `x-amz-mfa`를 포함하는 요청은 HTTPS를 사용해야 합니다. 헤더의 값은 인증 디바이스의 일련 번호, 공백, 여기에 표시된 인증 코드가 연속된 값입니다. 이 요청 헤더를 포함하지 않은 경우 요청은 실패합니다.

인증 디바이스에 대한 자세한 내용은 https://aws.amazon.com/iam/details/mfa/ 단원을 참조하십시오.

Example MFA Delete 사용 버킷에서 객체 삭제

다음 예제는 MFA Delete를 사용하도록 구성된 버킷에 있는 `my-image.jpg`를 지정된 버전으로 삭제하는 방법을 보여 줍니다. *`SerialNumber`* 및 *`AuthenticationCode`* 사이의 공백에 주의하십시오. 자세한 내용은 DELETE Object 단원을 참조하십시오.

```
DELETE /my-image.jpg?versionId=3HL4kqCxf3vjVBH40Nrjfkd HTTPS/1.1
Host: bucketName.s3.amazonaws.com
x-amz-mfa: 20899872 301749
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

MFA Delete 활성화에 대한 자세한 내용은 MFA Delete (p. 470) 단원을 참조하십시오.

삭제 마커를 통한 작업

삭제 마커는 단순 `DELETE` 요청에 이름이 올랐던 버전이 지정된 객체의 자리 표시자(마커)입니다. 객체가 버전 관리를 사용하는 버킷에 있었던 이유로 이 객체는 삭제되지 않았습니다. 그러나 삭제 마커는 Amazon S3가 객체가 삭제된 것처럼 동작하게 만듭니다.

삭제 마커에는 다른 객체들과 마찬가지로 키 이름(또는 키)과 버전 ID가 있습니다. 그러나 삭제 마커는 다음과 같은 부분에서 다른 객체와는 다릅니다.

- 데이터와 연결되지 않습니다.
- ACL(액세스 통제 목록) 값과 연결되어 있지 않습니다.
- 데이터가 없으므로 `GET` 요청에서 아무것도 가져오지 않으며 404 오류가 발생합니다.
- 삭제 마커에서 사용할 수 있는 유일한 작업은 Amazon S3 API `DELETE` 호출입니다. 이렇게 하려면 적절한 권한이 있는 역할 또는 AWS Identity and Access Management(IAM) 사용자를 활용하여 `DELETE` 요청을 수행해야 합니다.

삭제 마커는 Amazon S3의 스토리지에 대해 일반 요금을 발생시킵니다. 삭제 마커의 스토리지 크기는 삭제 마커의 키 이름 크기와 같습니다. 키 이름은 유니코드 문자열입니다. UTF-8 인코딩은 1~4바이트의 스토리지

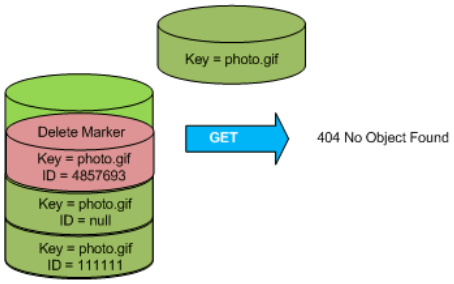
에서 이름의 각 문자별 버킷으로 추가됩니다. 키 이름에 대한 자세한 내용은 객체 키 (p. 103) 단원을 참조하십시오. 삭제 마커에 대한 자세한 내용은 삭제 마커 제거 (p. 484) 단원을 참조하십시오.

Amazon S3에서만 삭제 마커를 생성할 수 있으며, 버전 관리가 활성화되었거나 일시 중지된 버킷의 객체에 대해 사용자가 `DELETE Object` 요청을 전송할 경우에만 삭제 마커를 생성합니다. `DELETE` 요청에 이름을 올린 객체는 실제로 삭제되는 것이 아닙니다. 그 대신, 삭제 마커가 객체의 현재 버전이 됩니다. (객체의 키 이름(또는 키)은 삭제 마커의 키가 됩니다.) 객체를 가져오려는 경우 객체의 현재 버전이 삭제 마커라면, Amazon S3에서는 다음 응답을 내보냅니다.

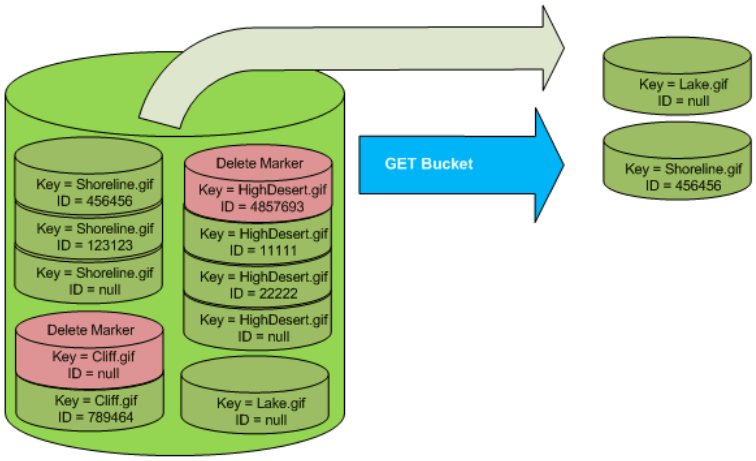
- 404(객체를 찾을 수 없음) 오류
- 응답 헤더 x-amz-delete-marker: true

응답 헤더는 액세스한 객체가 삭제 마커였음을 보고합니다. 이 응답 헤더는 `false`로 돌아가지 않습니다. 이 값이 `false`인 경우 Amazon S3에서는 이 응답 헤더를 응답에 포함하지 않습니다.

다음 그림은 현재 버전이 삭제 마커인 객체에 대해 단순 `GET` 요청이 어떻게 404 No Object Found 오류를 반환하는지를 보여 줍니다.



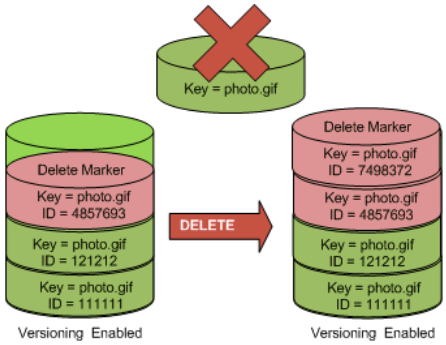
삭제 마커 목록(객체의 다른 버전 포함)을 표시하는 유일한 방법은 `versions` 요청에 `GET Bucket versions` 하위 리소스를 사용하는 것입니다. 단순 `GET` 요청은 삭제 마커 객체를 가져오지 않습니다. 다음 그림은 `GET Bucket` 요청은 현재 버전이 삭제 마커인 객체를 반환하지 않음을 보여 줍니다.



삭제 마커 제거

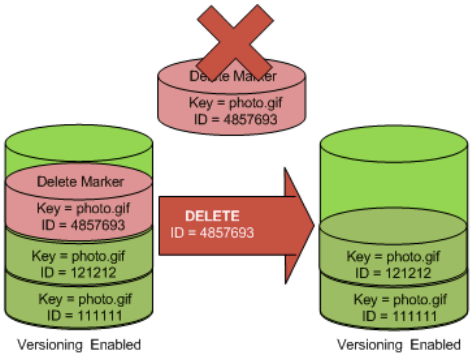
삭제 마커를 삭제하려면 `DELETE Object versionId` 요청에 삭제 마커의 버전 ID를 지정해야 합니다. `DELETE` 요청을 사용하여 삭제 마커를 삭제하는 경우(삭제 마커의 버전 ID 지정 없이), Amazon S3에서는 삭제 마커를 삭제하지 않고 대신 다른 삭제 마커를 삽입합니다.

다음 그림은 삭제 마커에 대한 단순 `DELETE` 요청이 아무것도 제거하지 않고 버킷에 새로운 삭제 마커를 추가하는 동작을 보여 줍니다.



버전 관리를 사용하는 버킷에서, 이 새로운 삭제 마커는 고유한 버전 ID를 갖습니다. 따라서 하나의 버킷 내에 동일 객체의 삭제 마커를 여러 개 보유할 수 있습니다.

삭제 마커를 영구적으로 삭제하려면 `DELETE Object versionId` 요청에 삭제 마커의 버전 ID를 포함해야 합니다. 다음 그림은 `DELETE Object versionId` 요청이 어떻게 삭제 마커를 영구적으로 제거하는지를 보여 줍니다. 버킷 소유자만이 삭제 마커를 영구적으로 제거할 수 있습니다.



삭제 마커 제거의 영향으로 단순 `GET` 요청은 이제 현재 버전(121212)의 객체를 가져옵니다.

삭제 마커를 영구적으로 제거하려면

1. `versionId`를 제거하려는 삭제 마커의 버전 ID에 설정합니다.
2. `DELETE Object versionId` 요청을 보냅니다.

Example 삭제 마커 제거

다음 예제에서는 `photo.gif` 버전 4857693에 대한 삭제 마커를 제거합니다.

```
DELETE /photo.gif?versionId=4857693 HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

삭제 마커를 삭제하는 경우 Amazon S3에서는 응답에 다음을 포함합니다.

```
204 NoContent
x-amz-version-id: versionID
x-amz-delete-marker: true
```

객체 버전 이전

또한 수명 주기가 명확하게 정의된 객체에 대해 수명 주기 구성 규칙을 정의하여 객체 수명의 특정 시점에 `S3 Glacier` 스토리지 클래스로 객체 버전을 이전할 수 있습니다. 자세한 내용은 객체 수명 주기 관리 (p. 123) 단원을 참조하십시오.

이전 버전 복원

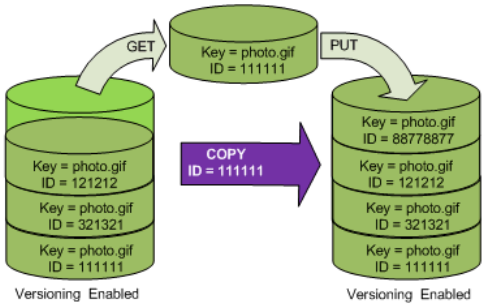
버전 관리의 가치 제안 중 하나는 객체의 이전 버전을 가져올 수 있는 기능입니다. 이를 수행하기 위한 접근 방식으로 다음 두 가지가 있습니다.

- 이전 버전의 객체를 동일한 버킷으로 복사합니다.

  복사된 객체는 해당 객체의 현재 버전이 되고 모든 객체 버전은 유지됩니다.
- 객체의 현재 버전 영구 삭제

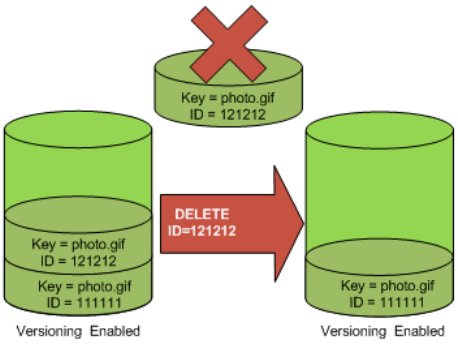
  현재 객체 버전을 삭제하는 것은 실제로 해당 객체의 이전 버전을 현재 버전으로 만드는 것과 같습니다.

전체 객체 버전이 유지되므로 특정 버전의 객체를 동일 버킷에 복사하여 현재 버전보다 이전의 버전을 만들 수 있습니다. 다음 그림에서 원본 객체(ID = 111111)는 동일한 버킷에 복사됩니다. Amazon S3가 새 ID(88778877)를 제공하며 이것이 객체의 현재 버전이 됩니다. 따라서 버킷에는 원래 객체 버전(111111)과 그 사본(88778877)이 모두 들어 있습니다.



후속 `GET` 요청은 버전 88778877을 가져옵니다.

다음 그림은 현재 객체로 이전 버전(111111)을 남겨 둔 채로 객체의 현재 버전(121212)을 삭제하는 과정을 보여 줍니다.



후속 `GET` 요청은 버전 111111을 가져옵니다.

버전이 지정된 객체의 권한

권한은 버전 수준에서 설정됩니다. 각 버전별로 객체 소유자가 있으며, 객체 버전을 생성한 AWS 계정이 소유자가 됩니다. 따라서 동일 객체의 서로 다른 버전에 대해 서로 다른 권한을 설정할 수 있습니다. 이렇게 하

려면 `PUT Object versionId acl` 요청에 권한을 설정하려는 객체의 버전 ID를 지정해야 합니다. ACL 사용과 관련한 자세한 설명과 지침은 Amazon S3의 Identity and Access Management (p. 299) 단원을 참조하십시오.

Example 객체 버전에 대한 권한 설정

다음 요청은 피부여자 `BucketOwner@amazon.com`의 권한을 키 `FULL_CONTROL`, 버전 ID 3HL4kqtJvjVBH40Nrjfkd에 대한 `my-image.jpg`로 설정합니다.

```
PUT /my-image.jpg?acl&versionId=3HL4kqtJvjVBH40Nrjfkd HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
Content-Length: 124

<AccessControlPolicy>
  <Owner>
    <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
    <DisplayName>mtd@amazon.com</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="CanonicalUser">
        <ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
        <DisplayName>BucketOwner@amazon.com</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
  </AccessControlPolicy>
```

마찬가지로 특정 객체 버전의 권한을 얻으려면 `GET Object versionId acl` 요청에 객체의 버전 ID를 지정해야 합니다. 기본적으로 `GET Object acl`에서는 현재 버전 객체의 권한을 반환하므로 버전 ID를 포함해야 합니다.

Example 지정한 객체 버전에 대한 권한 가져오기

다음 예제에서 Amazon S3는 키 `my-image.jpg`, 버전 ID DVBH40Nr8X8gUMLUo에 대한 권한을 반환합니다.

```
GET /my-image.jpg?versionId=DVBH40Nr8X8gUMLUo&acl HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU
```

자세한 내용은 GET Object ACL 단원을 참조하십시오.

버전 관리가 일시 중지된 버킷의 객체 관리

주제

- 버전 관리가 일시 중지된 버킷에 객체 추가 (p. 488)
- 버전 관리가 일시 중지된 버킷에서 객체 가져오기 (p. 488)
- 버전 관리가 일시 중지된 버킷에서 객체 삭제 (p. 489)

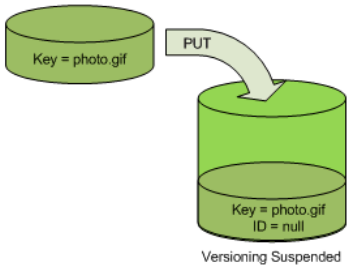
버킷 내 동일 객체의 새 버전이 발생하지 않도록 버전 관리를 일시 중지합니다. 버킷 내 단일 버전의 객체만 보유하길 원하는 이유에서 일시 중지할 수 있으며, 또는 여러 버전에 대한 요금이 발생하는 것을 원치 않을 수 있습니다.

버전 관리를 일시 중지할 때 버킷의 기존 객체는 변경되지 않습니다. 이후 요청에서 Amazon S3가 객체를 처리하는 방법만 변경됩니다. 이 단원의 주제에서는 버전 관리가 일시 중지된 버킷의 다양한 객체 작업에 관해 설명합니다.

버전 관리가 일시 중지된 버킷에 객체 추가

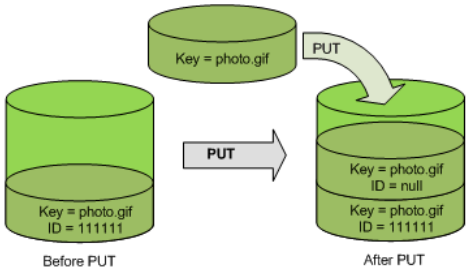
버킷에 대한 버전 관리를 일시 중지하면 Amazon S3에서는 자동으로 버전 ID `null`을 해당 버킷에 이후로 저장되는 후속 객체 각각에 추가합니다(`PUT`, `POST` 또는 `COPY` 사용).

다음 그림은 버전 관리가 일시 중지된 버킷에 객체를 추가할 때 Amazon S3에서 버전 ID `null`을 객체에 추가하는 과정을 보여 줍니다.

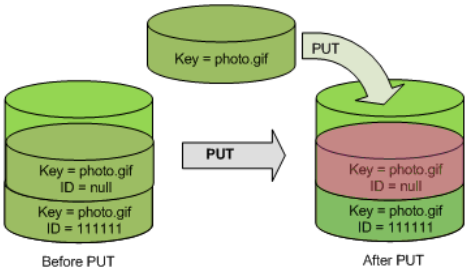


null 버전이 버킷에 이미 있는 경우 다른 객체를 같은 키로 추가하면 추가된 객체는 원래 null 버전을 덮어씁니다.

버전이 지정된 객체가 버킷에 있는 경우, `PUT`을 수행한 버전은 객체의 현재 버전이 됩니다. 다음 그림은 버전이 지정된 객체가 포함된 버킷에 객체를 추가해도 버킷에 이미 있던 객체는 덮어쓰지 않음을 보여 줍니다. 이 경우, 버전 111111은 버킷에 원래 존재했습니다. Amazon S3는 null 버전 ID를 추가되는 객체에 연결하고 이를 버킷에 저장합니다. 따라서 이는 버전 111111을 덮어쓰지 않습니다.



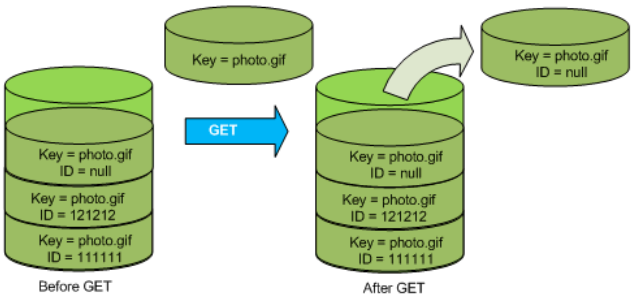
null 버전이 버킷에 이미 존재하는 경우, 다음 그림과 같이 이 null 버전을 덮어쓰게 됩니다.



null 버전의 키 및 버전 ID(`null`)가 `PUT` 수행 이전과 이후에 서로 같더라도, 버킷에 원래 저장되어 있던 null 버전의 콘텐츠는 버킷에 `PUT`을 수행한 객체의 콘텐츠로 대체됩니다.

버전 관리가 일시 중지된 버킷에서 객체 가져오기

`GET Object` 요청은 버킷에 대해 버전 관리를 활성화했는지 여부와 관계없이 현재 버전의 객체를 반환합니다. 다음 그림은 단순 `GET` 요청에서 현재 버전의 객체를 반환하는 과정을 보여 줍니다.



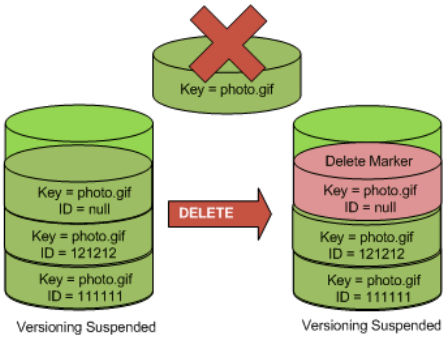
버전 관리가 일시 중지된 버킷에서 객체 삭제

버전 관리가 일시 중지된 경우 `DELETE` 요청을 통해 다음을 수행할 수 있습니다.

- 버전 ID가 `null`인 객체만 제거할 수 있습니다.

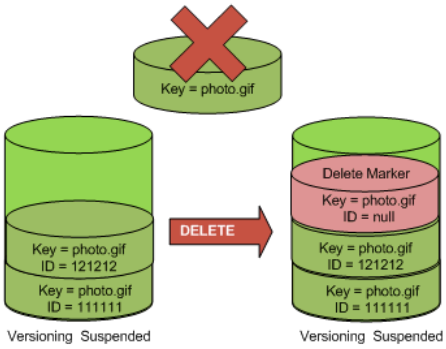
  버킷에 null 버전의 객체가 없는 경우 어떤 것도 제거하지 않습니다.
- 버킷에 삭제 마커를 삽입합니다.

다음 그림은 단순 `DELETE` 요청이 null 버전을 제거하고 Amazon S3에서 그 자리에 삭제 마커를 `null` 버전 ID로 삽입하는 과정을 보여 줍니다.

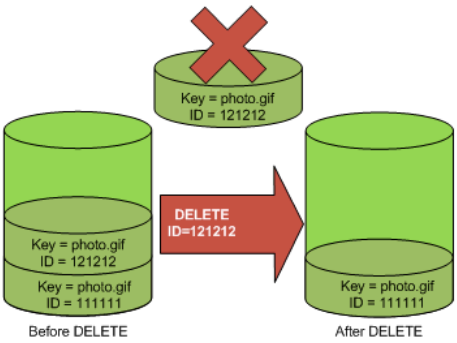


삭제 마커에는 콘텐츠가 없으며, 따라서 삭제 마커로 대체될 때 null 버전의 콘텐츠가 손실된다는 점에 유의하십시오.

다음 그림은 null 버전이 없는 버킷을 보여 줍니다. 이 경우 `DELETE` 요청은 아무것도 제거하지 않으며 Amazon S3에서는 삭제 마커만 삽입합니다.



버전 관리가 일시 정지된 버킷에서도 버킷 소유자는 특정 버전을 영구적으로 삭제할 수 있습니다. 다음 그림은 지정된 객체 버전을 삭제하면 해당 객체가 영구적으로 제거됨을 보여 줍니다. 버킷 소유자만이 지정된 객체 버전을 삭제할 수 있습니다.



Amazon S3 객체 잠금을 사용하여 객체 잠금

Amazon S3 object lock을 사용하면 write-once-read-many(WORM) 모델을 사용하여 객체를 저장할 수 있습니다. 일정 시간 동안 또는 무기한으로 객체가 삭제되거나 덮어쓰이지 않도록 방지할 수 있습니다. Amazon S3 object lock은 WORM 스토리지가 필요한 규제 요구 사항을 충족하거나 객체 변경 및 삭제에 대한 보호 계층을 추가하는 데 도움이 됩니다.

Amazon S3 object lock은 SEC 17a-4, CTCC 및 FINRA 규정의 적용을 받는 환경에서 Cohasset Associates에 의해 평가되었습니다. Amazon S3 object lock과 이러한 규정의 관계에 대한 자세한 내용은 Cohasset Associates 규정 준수 평가를 참조하십시오.

Amazon S3 object lock은 보관 기간 및 법적 보존 등, 객체 보관을 관리하는 두 가지 방법을 제공합니다.

- 보관 기간은 객체가 잠겨 있는 동안 고정된 기간을 지정합니다. 이 기간 동안 객체는 WORM으로 보호되며 덮어쓰거나 삭제할 수 없습니다.
- 법적 보존은 보관 기간과 동일한 보호를 제공하지만 만료 날짜는 없습니다. 대신, 명시적으로 제거할 때까지 법적 보존이 유지됩니다. 법적 보존은 보관 기간과 독립적입니다.

객체 버전은 보관 기간과 법적 보존 둘 다 또는 둘 중 하나 또는 둘 모두 아닐 수 있습니다. 자세한 내용은 Amazon S3 객체 잠금 개요 (p. 491) 단원을 참조하십시오.

Amazon S3 object lock은 버전이 지정된 버킷에서만 작동하며, 보관 기간과 법적 보유가 개별 객체 버전에 적용됩니다. 객체 버전을 잠그면 Amazon S3는 해당 객체 버전의 메타데이터에 잠금 정보를 저장합니다. 객체에 보관 기간 또는 법적 보존을 설정하면 요청에 지정된 버전만 보호됩니다. 객체의 새로운 버전이 생성되는 것을 차단하지 않습니다. 객체를 기존의 보호된 객체와 동일한 키 이름을 가진 버킷에 넣으면 Amazon S3가 해당 객체의 새 버전을 생성하고 요청된 대로 버킷에 저장한 다음 요청이 성공적으로 완료된 것으로 보고합니다. 기존의 보호된 버전의 객체는 보관 구성에 따라 잠긴 상태로 유지됩니다.

Amazon S3 object lock을 사용하려면 이러한 기본 단계를 따릅니다.

1. Amazon S3 object lock이 활성화된 새 버킷을 생성합니다.
2. (선택 사항) 버킷에 있는 객체의 기본 보관 기간을 구성합니다.
3. 버킷에 잠그고자 하는 객체를 넣습니다.
4. 보호하려는 객체에 보관 기간, 법적 보존 또는 둘 다 적용합니다.

AWS Management 콘솔에서 Amazon S3 object lock 사용에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 Amazon S3 객체를 어떻게 잠급니까?를 참조하십시오.

주제

- Amazon S3 객체 잠금 개요 (p. 491)
- Amazon S3 객체 잠금 관리 (p. 494)

Amazon S3 객체 잠금 개요

Amazon S3 object lock을 사용하면 write-once-read-many(WORM) 모델을 사용하여 객체를 저장할 수 있습니다. 고정된 시간 동안 또는 무기한으로 객체를 삭제하거나 덮어쓰지 않도록 할 수 있습니다. Amazon S3 object lock을 사용하여 WORM 스토리지가 필요한 규제 요구 사항을 충족하거나 객체 변경 및 삭제에 대한 보호 계층을 추가할 수 있습니다.

Amazon S3 객체의 잠금 상태 관리에 대한 자세한 내용은 the section called “객체 잠금 관리” (p. 494) 단원을 참조하십시오.

> **Note**
>
> Amazon S3 객체 잠금이 있는 S3 버킷은 서버 액세스 로깅 (p. 705)의 대상 버킷으로 사용할 수 없습니다.

다음 단원에서는 Amazon S3 object lock의 주요 기능에 대해 설명합니다.

주제

- 보관 모드 (p. 491)
- 보관 기간 (p. 492)
- 법적 보존 (p. 492)
- 버킷 구성 (p. 492)
- 필요한 권한 (p. 493)

보관 모드

Amazon S3 object lock은 두 가지 보관 모드를 제공합니다.

- 거버넌스 모드
- 규정 준수 모드

이러한 보관 모드는 객체에 대해 다양한 수준의 보호를 적용합니다. 둘 중 한 가지 보관 모드를 Amazon S3 object lock에 의해 보호되는 객체 버전에 적용할 수 있습니다.

거버넌스 모드에서 특별한 권한이 없는 한 사용자는 객체 버전을 덮어쓰거나 삭제하거나 잠금 설정을 변경할 수 없습니다. 거버넌스 모드를 사용하면 대부분의 사용자가 객체를 삭제하지 못하도록 보호하지만, 필요에 따라 일부 사용자에게 보관 설정을 변경하거나 객체를 삭제할 수 있는 권한을 부여할 수 있습니다. 규정 준수 모드 보관 기간을 생성하기 전에 거버넌스 모드를 사용하여 보관 기간 설정을 테스트할 수도 있습니다. 거버넌스 모드 보관 설정을 재정의하거나 제거하려면 사용자가 `s3:BypassGovernanceRetention` 권한을 갖고 있어야 하며 재정의 모드를 재정의해야 하는 요청과 함께 요청 헤더로 `x-amz-bypass-governance-retention:true`를 명시적으로 포함해야 합니다.

> **Note**
>
> Amazon S3 콘솔에는 기본적으로 `x-amz-bypass-governance-retention:true` 헤더가 포함됩니다. 거버넌스 모드로 보호되는 객체를 삭제하려는 경우 `s3:BypassGovernanceRetention` 또는 `s3:GetBucketObjectLockConfiguration` 권한이 있으면 작업이 성공합니다.

규정 준수 모드에서 보호된 객체 버전은 AWS 계정의 루트 사용자를 포함한 어떤 사용자도 덮어쓰거나 삭제할 수 없습니다. 규정 준수 모드에서 객체를 잠그면 보관 모드를 변경할 수 없으며 보관 기간을 줄일 수 없습니다. 규정 준수 모드는 보관 기간 동안 객체 버전을 덮어쓰거나 삭제할 수 없도록 합니다.

> **Note**
>
> 객체 잠금을 설정하거나 변경할 때 발생하는 객체 버전의 메타데이터 업데이트는 객체 버전을 덮어쓰거나 해당 `Last-Modified` 타임스탬프를 재설정하지 않습니다.

보관 기간

보관 기간은 정해진 시간 동안 객체 버전을 보호합니다. 객체 버전에 보관 기간을 설정하면 Amazon S3는 객체 버전의 메타데이터에 타임스탬프를 저장하여 보관 기간이 만료되는 시점을 표시합니다. 보관 기간이 만료된 후 객체 버전에 법적 보존을 설정하지 않는 한 객체 버전을 덮어쓰거나 삭제할 수 있습니다.

보관 기간은 명시적으로 또는 버킷 기본 설정을 통해 객체 버전에 적용할 수 있습니다. 객체 버전에 명시적으로 보관 기간을 적용할 경우 객체 버전에 대해 보관 종료일을 지정합니다. Amazon S3는 객체 버전의 메타데이터에 보관 종료일 설정을 저장하고 보관 기간이 만료될 때까지 객체 버전을 보호합니다.

버킷 기본 설정을 사용할 때는 보관 종료일을 지정하지 않습니다. 대신, 버킷에 있는 모든 객체 버전을 보호해야 하는 기간을 일 또는 연 단위로 지정합니다. 객체를 버킷에 배치하면 Amazon S3는 객체 버전의 생성 타임스탬프에 지정된 기간을 추가하여 객체 버전에 대한 보관 종료일을 계산합니다. 객체 버전의 메타데이터에 보관 종료일을 저장합니다. 그러면 객체 버전에 해당 보관 기간의 잠금을 명시적으로 배치한 것처럼 객체 버전이 보호됩니다.

> **Note**
>
> 버킷에 객체 버전을 배치하라는 요청이 명시적 보관 모드와 기간을 포함하면 해당 설정은 해당 객체 버전에 대한 버킷 기본 설정을 재정의합니다.

다른 모든 Amazon S3 object lock 설정과 마찬가지로 보관 기간은 개별 객체 버전에 적용됩니다. 단일 객체의 서로 다른 버전은 보관 모드 및 기간이 각기 다를 수 있습니다.

예를 들어 보관 기간 15-30일인 객체가 있고 같은 이름으로 60일 보관 기간으로 객체를 Amazon S3에 `PUT`한다고 가정하십시오. 이 경우 `PUT`이 성공하면 Amazon S3는 보관 기간이 60일인 새 버전의 객체가 생성됩니다. 이전 버전은 원래 보관 기간을 유지하고 15일 후에 삭제할 수 있게 됩니다.

보관 설정을 객체 버전에 적용한 후에는 보관 기간을 연장할 수 있습니다. 이렇게 하려면 `Retain Until Date`이 현재 객체 버전에 구성된 날짜보다 뒤에 있는 객체 버전에 대해 새 잠금 요청을 제출합니다. Amazon S3는 기존 보관 기간을 새롭고 더 긴 기간으로 대체합니다. 객체 보관 기간을 설정할 권한이 있는 사용자는 두 모드 중 하나에서 잠긴 객체 버전에 대해 보관 기간을 연장할 수 있습니다.

법적 보존

Amazon S3 object lock을 사용하면 객체 버전에 법적 보존을 설정할 수도 있습니다. 보관 기간과 마찬가지로 법적 보존을 사용하면 객체 버전을 덮어쓰거나 삭제할 수 없습니다. 그러나 법적 보존에는 연결된 보관 기간이 없고, 제거될 때까지 유효합니다. 법적 보존은 `s3:PutObjectLegalHold` 권한을 가진 사용자가 자유롭게 배치하고 제거할 수 있습니다. 전체 Amazon S3 권한 목록은 Amazon S3에 사용되는 작업, 리소스 및 조건 키 (p. 356) 단원을 참조하십시오.

법적 보존은 보관 기간과 독립적입니다. 객체가 포함된 버킷에 Amazon S3 object lock이 활성화되어 있으면 지정된 객체 버전에 보관 기간이 설정되었는지 여부에 관계없이 법적 보존을 설정하고 제거할 수 있습니다. 객체 버전에 법적 보존을 설정해도 해당 객체 버전의 보관 모드 또는 보관 기간에는 영향을 주지 않습니다. 예를 들어, 객체 버전을 보관 기간으로도 보호하는 동시에, 객체 버전에 법적 보존을 적용한다고 가정합니다. 보관 기간이 만료되면 객체의 WORM 보호가 손실되지 않습니다. 오히려, 법적 보유는 권한 있는 사용자가 명시적으로 제거할 때까지 객체를 계속 보호합니다. 마찬가지로 객체 버전의 보관 기간이 유효한 상태에서 법적 보존을 제거하면 보관 기간이 만료될 때까지 객체 버전이 보호됩니다.

버킷 구성

Amazon S3 object lock을 사용하려면 버킷에 대해 활성화합니다. 버킷에 있는 새 객체에 적용할 기본 보관 모드와 기간을 선택적으로 구성할 수도 있습니다.

객체 잠금 활성화

객체를 잠그기 전에 버킷이 Amazon S3 object lock을 사용하도록 구성해야 합니다. 이를 위해 버킷을 구성할 때 Amazon S3 object lock을 활성화하기로 지정해야 합니다. Amazon S3 object lock에 버킷을 구성하면 보관 기간, 법적 보존 또는 두 가지를 모두 사용하여 해당 버킷의 객체를 잠글 수 있습니다.

Note

- 새 버킷의 Amazon S3 object lock만 활성화할 수 있습니다. 기존 버킷에 대해 Amazon S3 object lock을 적용하려면 AWS Support에 문의하십시오.
- Amazon S3 object lock이 활성화된 버킷을 생성하면 Amazon S3가 버킷의 버전 관리를 자동으로 활성화합니다.
- Amazon S3 object lock이 활성화된 버킷을 생성하면 객체 잠금을 비활성화하거나 버킷의 버전 관리를 일시 중지할 수 없습니다.

콘솔에서 Amazon S3 object lock 활성화에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 Amazon S3 객체를 어떻게 잠급니까?를 참조하십시오.

기본 보관 설정

버킷에 Amazon S3 object lock을 켜면 버킷은 보호된 객체를 저장할 수 있습니다. 그러나 이 설정은 버킷에 넣은 객체를 자동으로 보호하지 않습니다. 버킷에 있는 객체 버전을 자동으로 보호하려면 기본 보관 기간을 구성할 수 있습니다. 객체를 만들 때 객체에 다른 보관 모드와 기간을 명시적으로 지정하지 않으면 기본 설정이 버킷에 있는 모든 새 객체에 적용됩니다.

Tip

버킷에 배치된 모든 새 객체 버전에 대해 버킷 기본 보관 모드 및 기간을 적용하려는 경우 버킷 기본값을 설정하고 사용자가 객체 보관 설정을 적용하는 권한을 거부하면 됩니다. 그러면 Amazon S3가 버킷에 있는 새 객체 버전에 기본 보관 모드와 기간을 적용하고 보관 모드와 설정이 포함된 객체를 구성하라는 요청은 거절합니다.

버킷 기본 설정에는 모드와 기간이 모두 필요합니다. 버킷 기본 모드는 거버넌스 또는 규정 준수입니다. 자세한 내용은 보관 모드 (p. 491) 단원을 참조하십시오.

기본 보관 기간은 타임스탬프가 아닌 일 또는 연 단위 기간으로 설명됩니다. 기본 보관 기간으로 버킷에 객체 버전을 배치하면 Amazon S3 object lock이 보관 종료일을 계산합니다. 객체 버전의 생성 타임스탬프에 기본 보관 기간을 추가하여 보관 종료일을 계산합니다. 마치 수동으로 타임스탬프를 계산하여 객체 버전에 직접 배치한 것처럼 Amazon S3가 결과 타임스탬프를 객체 버전의 보관 종료일로 저장합니다.

기본 설정은 버킷에 배치된 새 객체에만 적용됩니다. 버킷에 기본 보관 설정을 지정해도 이미 버킷에 있는 객체에는 보관 설정이 적용되지 않습니다.

Important

객체 잠금은 개별 객체 버전에만 적용됩니다. 기본 보관 기간이 있는 버킷에 객체를 배치하고 해당 객체의 보관 기간을 명시적으로 지정하지 않으면 Amazon S3는 버킷 기본값과 일치하는 보관 기간을 사용하여 객체를 생성합니다. 객체가 생성된 후 보관 기간은 버킷의 기본 보관 기간과 독립적입니다. 버킷의 기본 보관 기간을 변경해도 해당 버킷의 모든 객체에 대한 기존 보관 기간은 변경되지 않습니다.

Note

버킷에 기본 보존 기간을 구성하면 해당 버킷에 대한 객체 업로드 요청에 `Content-MD5` 헤더가 포함되어야 합니다. 자세한 내용은 Amazon Simple Storage Service API Reference의 PUT Object를 참조하십시오.

필요한 권한

Amazon S3 object lock 작업에는 특정 권한이 필요합니다. 필요한 권한에 대한 자세한 내용은 — 객체 작업 예제 (p. 343) 단원을 참조하십시오. 권한과 함께 조건을 사용하는 방법에 대한 자세한 내용은 Amazon S3 조건 키 (p. 345) 단원을 참조하십시오.

Amazon S3 객체 잠금 관리

Amazon S3 object lock을 사용하면 Amazon S3에서 write-once-read-many(WORM) 모델을 사용하여 객체를 저장할 수 있습니다. 이를 사용하여 Amazon S3 객체의 객체 잠금 상태를 보고 구성하고 관리할 수 있습니다. Amazon S3 object lock 기능에 대한 자세한 내용은 Amazon S3 객체 잠금 개요 (p. 491) 단원을 참조하십시오.

주제

- 객체의 잠금 정보 보기 (p. 494)
- 거버넌스 모드 무시 (p. 494)
- 이벤트 및 알림 구성 (p. 494)
- 보관 제한 설정 (p. 495)
- 삭제 마커 및 객체 수명 주기 관리 (p. 495)
- 복제에 객체 잠금 사용 (p. 496)

객체의 잠금 정보 보기

`GET Object` 또는 `HEAD Object` 명령을 사용하여 Amazon S3 객체 버전의 객체 잠금 상태를 볼 수 있습니다. 두 명령 모두 지정된 객체 버전에 대한 보관 모드, `Retain Until Date` 및 법적 보존 상태를 반환합니다.

객체 버전의 보관 모드 및 보관 기간을 보려면 `s3:GetObjectRetention` 권한이 있어야 합니다. 객체 버전의 법적 보존 상태를 보려면 `s3:GetObjectLegalHold` 권한이 있어야 합니다. `GET` 또는 `HEAD` 객체 버전이지만 잠금 상태를 보는 데 필요한 권한이 없는 경우 요청이 성공합니다. 그러나 볼 수 있는 권한이 없는 정보는 반환하지 않습니다.

버킷의 기본 보관 구성(있는 경우)을 보려면 버킷의 객체 잠금 구성을 요청하십시오. 이렇게 하려면 `s3:GetBucketObjectLockConfiguration` 권한이 있어야 합니다. Amazon S3 object lock이 활성화되지 않은 버킷에 대해 객체 잠금 구성을 요청하면 Amazon S3가 오류를 반환합니다. 권한에 대한 자세한 내용은 — 객체 작업 예제 (p. 343) 단원을 참조하십시오.

버킷의 Amazon S3 인벤토리 보고서를 구성하여 버킷에 있는 모든 객체의 `Retain Until Date`, `object lock Mode` 및 `Legal Hold Status`를 포함할 수 있습니다. 자세한 내용은 Amazon S3 인벤토리 (p. 459) 단원을 참조하십시오.

거버넌스 모드 무시

`s3:BypassGovernanceRetention` 권한이 있는 경우, 거버넌스 모드에서 잠긴 객체 버전에 대한 작업을 보호되지 않은 것처럼 수행할 수 있습니다. 이러한 작업에는 객체 버전 삭제, 보관 기간 단축 또는 빈 파라미터가 있는 새 잠금을 배치하여 객체 잠금을 제거하는 작업이 포함됩니다. 거버넌스 모드를 무시하려면 요청에 이 모드를 무시할 것인지 명시해야 합니다. 이를 위해 요청 시 `x-amz-bypass-governance-retention:true` 헤더를 포함시키거나 AWS CLI 또는 AWS SDK를 통해 만든 요청과 동등한 파라미터를 사용합니다. 거버넌스 모드를 우회하는 데 필요한 권한이 있을 경우, 콘솔을 통해 만든 요청에 대한 이 헤더를 AWS Management 콘솔이 자동으로 적용합니다.

> **Note**
>
> 거버넌스 모드 무시는 객체 버전의 법적 보존 상태에 영향을 주지 않습니다. 객체 버전에 법적 보존이 활성화된 경우, 법적 보존은 유효하게 유지되고, 요구가 객체 버전을 덮어쓰거나 삭제하지 못합니다.

이벤트 및 알림 구성

Amazon S3 이벤트는 객체 잠금 버킷의 객체 수준 작업에 맞게 구성할 수 있습니다. `PUT Object`, `HEAD Object` 및 `GET Object` 호출에 객체 잠금 메타데이터가 포함되어 있으면 이러한 호출의 이벤트에 해당 메

타데이터 값이 포함됩니다. 객체 잠금 메타데이터가 객체에 추가되거나 업데이트되면 이러한 작업이 이벤트도 트리거합니다. 이러한 이벤트는 `PUT` 또는 `GET` 객체 보관 또는 법적 보존 정보가 있을 때마다 발생합니다.

Amazon S3 이벤트에 대한 자세한 내용은 Amazon S3 이벤트 알림 구성 (p. 587) 단원을 참조하십시오.

Amazon S3 이벤트 알림을 사용하여 객체 잠금 구성 및 AWS CloudTrail 사용 데이터에 대한 액세스 및 변경을 추적할 수 있습니다. CloudTrail에 대한 자세한 내용은 AWS CloudTrail 설명서를 참조하십시오.

Amazon CloudWatch를 사용하여 이 데이터 기반 알림을 생성할 수도 있습니다. CloudWatch에 대한 자세한 내용은 Amazon CloudWatch 설명서를 참조하십시오.

보관 제한 설정

버킷 정책을 사용하여 버킷의 최소 및 최대 허용 보관 기간을 설정할 수 있습니다. `s3:object-lock-remaining-retention-days` 조건 키를 사용하면 됩니다. 다음 예에서는 `s3:object-lock-remaining-retention-days` 조건 키를 사용하여 최대 보관 기간을 10일로 설정하는 버킷 정책을 보여줍니다.

```
{
    "Version": "2012-10-17",
    "Id": "<Policy1436912751980>",
    "Statement": [
        {
            "Sid": "<Stmt1436912698057>",
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "s3:PutObjectRetention"
            ],
            "Resource": "arn:aws:s3:::<example-bucket>/*",
            "Condition": {
                "NumericGreaterThan": {
                    "s3:object-lock-remaining-retention-days": "10"
                }
            }
        }
    ]
}
```

> **Note**
>
> 버킷이 복제 정책의 대상 버킷이고 복제를 사용하여 생성된 객체 복제본에 대해 최소 및 최대 허용 보존 기간을 설정하려는 경우, 버킷 정책에 `s3:ReplicateObject` 작업을 포함시켜야 합니다.

자세한 내용은 다음 주제를 참조하십시오.

- Amazon S3에 사용되는 작업, 리소스 및 조건 키 (p. 356)
- — 객체 작업 예제 (p. 343)
- Amazon S3 조건 키 (p. 345)

삭제 마커 및 객체 수명 주기 관리

보호된 객체 버전은 삭제할 수 없지만 해당 객체에 대한 삭제 마커를 생성할 수 있습니다. 객체에 삭제 마커를 배치해도 객체 버전이 삭제되지 않습니다. 그러나 대부분의 경우 객체가 삭제된 것처럼 Amazon S3가 동작을 수행하게 됩니다. 자세한 내용은 삭제 마커를 통한 작업 (p. 483) 단원을 참조하십시오.

> **Note**
>
> 기본 객체에 설정된 보관 기간이나 법적 보존과 상관없이 삭제 마커는 WORM으로 보호되지 않습니다.

객체 수명 주기 관리 구성은 삭제 마커 배치를 포함하여 보호된 객체에서 계속 정상적으로 작동합니다. 그러나 보호된 객체 버전은 수명 주기 구성에 의해 삭제되거나 덮어쓰기 되지 않도록 안전하게 유지됩니다. 객체 수명 주기 관리에 대한 자세한 내용은 객체 수명 주기 관리 (p. 123)을 참조하십시오.

복제에 객체 잠금 사용

복제에 Amazon S3 객체 잠금을 사용하여 잠긴 객체와 보존 메타데이터를 서로 다르거나 동일한 AWS 리전에 있는 S3 버킷에 비동기식으로 자동 복사할 수 있습니다. 복제를 사용하면 원본 버킷의 객체가 대상 버킷으로 복제됩니다. 자세한 내용은 복제 (p. 607) 단원을 참조하십시오.

복제에 객체 잠금을 설정하려면 다음 옵션 중 하나를 선택할 수 있습니다.

옵션 1: 먼저 객체 잠금을 활성화합니다.

1. 대상 버킷에서 또는 원본과 대상 버킷 모두에서 객체 잠금을 활성화합니다.
2. 원본 버킷과 대상 버킷 간에 복제를 설정합니다.

옵션 2: 복제를 먼저 설정합니다.

1. 원본 버킷과 대상 버킷 간에 복제를 설정합니다.
2. 대상 버킷에만 또는 원본과 대상 버킷 모두에서 객체 잠금을 활성화합니다.

앞에 나온 옵션의 2단계를 완료하려면 AWS Support에 문의해야 합니다. 복제를 올바르게 구성하려면 꼭 필요한 일입니다.

AWS Support에 문의하기 전에 복제에 객체 잠금을 설정하기 위한 다음의 요구 사항을 검토하십시오.

- Amazon S3 대상 버킷에 객체 잠금이 활성화되어 있어야 합니다.
- 복제 설정에 사용할 AWS Identity and Access Management(IAM) 역할에서 원본 S3 버킷에 두 가지 새로운 권한을 부여해야 합니다. 두 가지 새로운 권한은 `s3:GetObjectRetention`과 `s3:GetObjectLegalHold`입니다. 역할에 `s3:Get*` 권한이 있으면 요구 사항이 충족됩니다. 자세한 내용은 복제에 대한 권한 설정 (p. 620) 단원을 참조하십시오.

Amazon S3 객체 잠금에 대한 자세한 내용은 Amazon S3 객체 잠금을 사용하여 객체 잠금 (p. 490) 단원을 참조하십시오.

Amazon S3의 인프라 보안

관리형 서비스인 Amazon S3은 Amazon Web Services: 보안 프로세스 개요 백서에 설명된 AWS 글로벌 네트워크 보안 절차로 보호됩니다.

네트워크를 통한 Amazon S3 액세스는 AWS에서 게시한 API를 통해 이루어집니다. 클라이언트가 TLS(전송 계층 보안) 1.0을 지원해야 합니다. TLS 1.2를 권장합니다. 클라이언트는 DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Diffie-Hellman Ephemeral)와 같은 PFS(전달 완전 보안)가 포함된 암호 제품군도 지원해야 합니다. 또한 유효한 자격 증명을 제공해야 하는 AWS Signature V4 또는 AWS Signature V2를 사용하여 요청에 서명해야 합니다.

네트워크 위치에서 이 API를 호출할 수 있습니다. 하지만 Amazon S3은 원본 IP 주소 기반의 제한을 포함할 수 있는 리소스 기반 액세스 정책을 지원합니다. Amazon S3 버킷 정책을 사용하여 특정 Amazon Virtual Private Cloud(Amazon VPC) 엔드포인트나 특정 VPC에서 버킷에 대한 액세스를 제어할 수도 있습니다. 그러면 AWS 네트워크의 특정 VPC에서만 주어진 Amazon S3 버킷에 대한 네트워크 액세스가 효과적으로 격리됩니다. 자세한 내용은 Amazon S3의 VPC 엔드포인트에 대한 예제 버킷 정책 (p. 414) 단원을 참조하십시오.

다음 보안 모범 사례에서도 Amazon S3의 인프라 보안을 다룹니다.

- Amazon S3 액세스의 VPC 엔드포인트 고려
- 모든 Amazon S3 버킷 식별 및 감사

Amazon S3의 구성 및 취약성 분석

AWS가 게스트 운영 체제(OS), 데이터베이스 패치, 방화벽 구성, 재해 복구 등의 기본 보안 작업을 처리합니다. 적합한 제3자가 이 절차를 검토하고 인증하였습니다. 자세한 내용은 다음 리소스를 참조하십시오.

- Amazon S3 규정 준수 확인 (p. 459)
- 공동 책임 모델
- Amazon Web Services: 보안 프로세스의 개요(백서)

다음 보안 모범 사례에서도 Amazon S3의 구성 및 취약성 분석을 다룹니다.

- 모든 Amazon S3 버킷 식별 및 감사
- AWS Config 활성화

Amazon S3 보안 모범 사례

Amazon S3은 자체 보안 정책을 개발하고 구현할 때 고려해야 할 여러 가지 보안 기능을 제공합니다. 다음 모범 사례는 일반적인 지침이며 완벽한 보안 솔루션을 나타내지는 않습니다. 이러한 모범 사례는 환경에 적절하지 않거나 충분하지 않을 수 있으므로 참고용으로만 사용해 주십시오.

주제

- Amazon S3 예방 보안 모범 사례 (p. 497)
- Amazon S3 모니터링 및 감사 모범 사례 (p. 500)

Amazon S3 예방 보안 모범 사례

다음과 같은 Amazon S3 모범 사례를 통해 보안 사고를 예방할 수 있습니다.

Amazon S3 버킷이 올바른 정책을 사용하게 하고 이 버킷에 대한 퍼블릭 액세스 방지

인터넷의 누군가에게 S3 버킷을 읽거나 이 버킷에 쓰도록 명시적으로 요구하지 않을 경우 S3 버킷이 공개되지 않도록 해야 합니다. 다음과 같이 몇 가지 단계를 수행할 수 있습니다.

- Amazon S3 퍼블릭 액세스 차단을 사용합니다. Amazon S3 퍼블릭 액세스 차단을 사용하면 계정 관리자와 버킷 소유자가 리소스 생성 방식에 관계없이 적용되는 Amazon S3 리소스에 대한 퍼블릭 액세스를 제한하는 중앙 집중식 제어를 쉽게 설정할 수 있습니다. 자세한 내용은 Amazon S3 퍼블릭 액세스 차단 사용 (p. 450) 단원을 참조하십시오.
- 보안 주체 “*”(“아무나”를 효과적으로 표현)와 같은 와일드카드 자격 증명을 허용하거나 와일드카드 작업 “*”(Amazon S3 버킷에서 아무 작업이나 수행하도록 사용자에게 효과적으로 허용)를 허용하는 Amazon S3 버킷 정책을 식별합니다.
- 마찬가지로 “모든 사람”이나 “인증된 모든 AWS 사용자”에게 읽기, 쓰기 또는 모든 액세스 권한을 제공하는 Amazon S3 버킷 ACL(액세스 제어 목록)을 메모합니다.
- `ListBuckets` API를 사용하여 Amazon S3 버킷을 모두 스캔합니다. 그런 다음 `GetBucketAcl`, `GetBucketWebsite` 및 `GetBucketPolicy`를 사용하여 버킷에 규정에 맞는 액세스 제어와 구성이 있는지 확인합니다.

- AWS Trusted Advisor를 사용하여 Amazon S3 구현을 조사합니다.
- s3-bucket-public-read-prohibited 및 s3-bucket-public-write-prohibited 관리형 AWS Config 규칙를 사용하여 지속적인 탐지 제어를 구현해 보십시오.

자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 버킷 및 객체 액세스 권한 설정을 참조하십시오.

최소 권한 액세스 구현

권한을 부여할 때 누가 어떤 Amazon S3 리소스에 대해 어떤 권한을 갖는지 결정합니다. 해당 리소스에서 허용할 작업을 활성화합니다. 따라서 작업을 수행하는 데 필요한 권한만 부여해야 합니다. 최소 권한 액세스를 구현하는 것이 오류 또는 악의적인 의도로 인해 발생할 수 있는 보안 위험과 영향을 최소화할 수 있는 근본적인 방법입니다.

다음과 같은 도구를 사용하여 최소 권한 액세스를 구현할 수 있습니다.

- IAM 사용자 정책 및 IAM 개체에 대한 권한 경계
- Amazon S3 버킷 정책
- Amazon S3 ACL(액세스 제어 목록)
- 서비스 제어 정책

앞에 나온 메커니즘 중 하나 이상을 선택할 때 고려할 내용은 Amazon S3 리소스에 대한 액세스 관리 소개 (p. 299) 단원을 참조하십시오.

Amazon S3 액세스를 요구하는 애플리케이션 및 AWS 제품에 IAM 역할 사용

Amazon EC2나 다른 AWS 제품의 애플리케이션이 Amazon S3 리소스에 액세스하기 위해서는 AWS API 요청에 유효한 AWS 자격 증명이 있어야 합니다. AWS 자격 증명을 애플리케이션이나 Amazon EC2 인스턴스에 직접 저장하면 안 됩니다. 이러한 자격 증명은 자동으로 교체되지 않으며 손상된 경우 비즈니스에 큰 영향을 줄 수 있는 장기 자격 증명입니다.

대신 IAM 역할을 사용하여 Amazon S3에 액세스해야 하는 애플리케이션이나 서비스의 임시 자격 증명을 관리해야 합니다. 역할을 사용할 때 Amazon EC2 인스턴스나 AWS 제품(예: AWS Lambda)에 장기 자격 증명(예: 사용자 이름과 암호 또는 액세스 키)을 배포할 필요가 없습니다. 애플리케이션에서 다른 AWS 리소스를 호출할 때 사용할 수 있는 임시 권한을 역할이 제공합니다.

자세한 정보는 IAM 사용 설명서에서 다음 주제를 참조하십시오.

- IAM 역할
- 역할에 대한 일반적인 시나리오: 사용자, 애플리케이션 및 서비스

Multi-Factor Authentication(MFA) 삭제 활성화

MFA 삭제를 사용하면 잘못해서 버킷을 삭제하는 일을 방지할 수 있습니다. MFA 삭제를 활성화하지 않으면 권한이 충분한 루트 또는 IAM 사용자의 암호를 가진 사용자는 누구든지 Amazon S3 객체를 영구히 삭제할 수 있습니다.

MFA 삭제를 사용하려면 다음 작업 중 하나를 위한 추가 인증이 필요합니다.

- 버킷의 버전 관리 상태 변경
- 객체 버전 영구 삭제

자세한 내용은 MFA Delete (p. 470) 단원을 참조하십시오.

유휴 데이터의 암호화 고려

Amazon S3에서 유휴 데이터를 보호하는 다음과 같은 옵션이 있습니다.

- 서버 측 암호화 – 데이터 센터의 디스크에 저장하기 전에 객체를 암호화하고 객체를 다운로드할 때 이를 해독하도록 Amazon S3에 요청합니다. 서버 측 암호화를 사용하면 데이터 자체를 저장하는 것과는 다른 메커니즘으로 저장되는 키로 데이터를 암호화하여 데이터에 가해지는 위험을 줄일 수 있습니다.

Amazon S3에서는 여러 가지 서버 측 암호화 옵션을 제공합니다. 자세한 내용은 서버 측 암호화를 사용하여 데이터 보호 (p. 263) 단원을 참조하십시오.

- 클라이언트 측 암호화 – 클라이언트 측 데이터를 암호화하여 암호화된 데이터를 Amazon S3에 업로드합니다. 이 경우 사용자가 암호화 프로세스, 암호화 키 및 관련 도구를 관리합니다. 서버 측 암호화를 사용할 때처럼, 클라이언트 측 암호화를 사용하면 데이터 자체를 저장하는 것과는 다른 메커니즘으로 저장되는 키로 데이터를 암호화하여 위험을 줄일 수 있습니다.

  Amazon S3에서는 여러 가지 클라이언트 측 암호화 옵션을 제공합니다. 자세한 내용은 클라이언트측 암호화를 사용하여 데이터 보호 (p. 291) 단원을 참조하십시오.

전송 중 데이터의 암호화 적용

HTTPS(TLS)를 사용하여 잠재적 공격자가 네트워크 트래픽을 염탐하거나 중간자 또는 그와 유사한 공격을 사용하여 네트워크 트래픽을 조작하지 못하게 할 수 있습니다. Amazon S3 버킷 정책에 aws:SecureTransport 조건을 사용하여 HTTPS(TLS)를 통해 암호화된 연결만 허용해야 합니다.

s3-bucket-ssl-requests-only 관리형 AWS Config 규칙을 사용하여 지속적 탐지 제어를 구현하는 것도 고려해 보십시오.

Amazon S3 객체 잠금 고려

Amazon S3 객체 잠금을 사용하면 WORM("Write Once Read Many) 모델을 사용하여 객체를 저장할 수 있습니다. Amazon S3 객체 잠금은 부적절하거나 실수로 인한 데이터 삭제를 예방하는 데 도움이 됩니다. 예를 들어 Amazon S3 객체 잠금을 사용하여 AWS CloudTrail 로그를 보호할 수 있습니다.

버전 관리 사용

버전 관리는 동일 버킷 내에 여러 개의 객체 변형을 보유하는 것을 의미합니다. 버전 관리를 사용하면 Amazon S3 버킷에 저장된 모든 버전의 모든 객체를 보존, 검색 및 복원할 수 있습니다. 또한 의도치 않은 사용자 작업 및 애플리케이션 장애로부터 쉽게 복구할 수 있습니다.

s3-bucket-versioning-enabled 관리형 AWS Config 규칙을 사용하여 지속적 탐지 제어를 구현하는 것도 고려해 보십시오.

자세한 내용은 버전 관리 사용 (p. 468) 단원을 참조하십시오.

Amazon S3 교차 리전 복제 고려

Amazon S3이 기본적으로 여러 지역에 있는 다양한 가용 영역에 데이터를 저장하지만, 규정 준수 요구 사항에 따라 훨씬 먼 거리에 데이터를 저장해야 할 수도 있습니다. CRR(교차 리전 복제)을 사용하면 멀리 있는 AWS 리전 간에 데이터를 복제하여 이 요구 사항을 충족할 수 있습니다. CRR은 여러 AWS 리전에 있는 버킷 간에 객체를 비동기식으로 자동 복제할 수 있게 합니다. 자세한 내용은 복제 (p. 607) 단원을 참조하십시오.

> **Note**
>
> CRR을 사용할 경우 원본 및 대상 S3 버킷 둘 다 버전 관리가 활성화되어야 합니다.

s3-bucket-replication-enabled 관리형 AWS Config 규칙을 사용하여 지속적 탐지 제어를 구현하는 것도 고려해 보십시오.

Amazon S3 액세스의 VPC 엔드포인트 고려

Amazon S3의 VPC 엔드포인트는 Amazon S3에만 연결을 허용하는 Amazon Virtual Private Cloud(Amazon VPC) 내 논리적 개체입니다. Amazon S3 버킷 정책을 사용하여 특정 Amazon VPC 엔드포인트나 특정 VPC에서 버킷에 대한 액세스를 제어할 수 있습니다. VPC 엔드포인트는 트래픽이 공개된 인터넷을 잠재적으로 통과하고 공개된 인터넷 환경에 있게 되는 것을 방지해 줍니다.

Amazon S3의 VPC 엔드포인트는 Amazon S3 데이터에 대한 액세스를 제어하는 두 가지 방법을 제공합니다.

- 사용자는 특정 VPC 종단점을 통해 허용되는 요청, 사용자 또는 그룹을 제어할 수 있습니다.

- S3 버킷 정책을 사용하여 S3 버킷에 대한 액세스를 갖게되는 VPC 또는 VPC 종단점을 제어할 수 있습니다.
- 인터넷 게이트웨이가 없는 VPC를 사용하여 데이터 유출을 방지할 수 있습니다.

자세한 내용은 Amazon S3의 VPC 엔드포인트에 대한 예제 버킷 정책 (p. 414) 단원을 참조하십시오.

Amazon S3 모니터링 및 감사 모범 사례

다음과 같은 Amazon S3 모범 사례가 잠재적 보안 약점과 사고를 탐지하는 데 도움이 됩니다.

모든 Amazon S3 버킷 식별 및 감사

IT 자산 식별은 거버넌스와 보안의 중요한 측면입니다. 모든 Amazon S3 리소스를 파악하여 보안 태세를 평가하고 약점이 될 수 있는 부분에 조치를 취해야 합니다.

Tag Editor를 사용하여 보안이나 감사에 민감한 리소스를 식별한 후, 이 리소스를 검색해야 할 때 태그를 이용하십시오. 자세한 내용은 태그를 지정할 리소스 검색을 참조하십시오.

Amazon S3 인벤토리를 사용하여 비즈니스, 규정 준수 및 규제 요건에 대한 객체의 복제 및 암호화 상태를 감사하고 보고합니다. 자세한 내용은 Amazon S3 인벤토리 (p. 459) 단원을 참조하십시오.

Amazon S3 리소스의 리소스 그룹을 만드십시오. 자세한 내용은 AWS 리소스 그룹이란 무엇입니까?를 참조하십시오.

AWS 모니터링 도구를 사용하여 모니터링 구현

모니터링은 Amazon S3과 AWS 솔루션의 안정성, 보안, 가용성 및 성능을 유지하는 데 중요한 부분입니다. AWS에서는 Amazon S3과 다른 AWS 제품을 모니터링할 수 있는 여러 가지 도구와 서비스를 제공합니다. 예를 들어 Amazon S3을 위한 CloudWatch 지표, 특히 `PutRequests`, `GetRequests`, `4xxErrors` 및 `DeleteRequests`를 모니터링할 수 있습니다. 자세한 내용은 Amazon CloudWatch로 지표 모니터링 (p. 669) 및 Amazon S3 모니터링 (p. 668) 단원을 참조하십시오.

두 번째 예는 예제: Amazon S3 버킷 작업을 참조하십시오. 이 예에서는 버킷 정책, 버킷 수명 주기 또는 버킷 복제를 PUT 또는 DELETE하거나 버킷 ACL을 PUT하기 위해 Amazon S3 API가 호출될 때 트리거되는 Amazon CloudWatch 경보를 생성하는 방법을 설명합니다.

Amazon S3 서버 액세스 로깅 활성화

서버 액세스 로깅은 버킷에 대해 이루어진 요청의 상세 레코드를 제공합니다. 서버 액세스 로그는 보안 및 액세스 감사에 도움이 되고, 고객 기반을 파악하고 Amazon S3 청구서를 이해하는 데 유용합니다. 서버 액세스 로깅 활성화에 대한 자세한 내용은 Amazon S3 서버 액세스 로깅 (p. 705) 단원을 참조하십시오.

s3-bucket-logging-enabled AWS Config 관리형 규칙을 사용하여 지속적 탐지 제어를 구현하는 것도 고려해 보십시오.

AWS CloudTrail 사용

AWS CloudTrail는 Amazon S3에서 사용자, 역할 또는 AWS 서비스가 수행한 작업에 대한 레코드를 제공합니다. CloudTrail에서 수집하는 정보를 사용하여 Amazon S3에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다. 예를 들어, 데이터 액세스에 영향을 주는 Put 작업(특히 `PutBucketAcl`, `PutObjectAcl`, `PutBucketPolicy`, `PutBucketWebsite`)의 CloudTrail 항목을 식별할 수 있습니다. AWS 계정을 설정하면 CloudTrail가 기본적으로 활성화됩니다. CloudTrail 콘솔에서 최근 이벤트를 볼 수 있습니다. Amazon S3 버킷에 대해 진행 중인 활동 및 이벤트 레코드를 만들려면 CloudTrail 콘솔에서 추적을 생성하면 됩니다. 자세한 내용은 AWS CloudTrail User Guide의 추적을 위해 데이터 이벤트 로깅을 참조하십시오.

추적을 생성할 때 데이터 이벤트를 로그로 기록하도록 CloudTrail를 구성할 수 있습니다. 데이터 이벤트는 리소스에 대해 또는 리소스 내에서 수행되는 리소스 작업의 레코드입니다. Amazon S3에서 데이터 이벤트는 개별 버킷에 대한 객체 수준 API 활동을 기록합니다. CloudTrail에서는 `GetObject`,

`DeleteObject` 및 `PutObject`와 같은 Amazon S3 객체 수준 API 작업의 하위 집합을 지원합니다. CloudTrail가 Amazon S3를 사용하는 방법에 대한 자세한 내용은 AWS CloudTrail을 사용하여 Amazon S3 API 호출 로깅 (p. 679) 단원을 참조하십시오. Amazon S3 콘솔에서 CloudTrail에 대한 객체 수준 로깅을 활성화하도록 S3 버킷을 구성할 수도 있습니다.

AWS Config는 하나 이상의 CloudTrail 추적이 S3 버킷에 대한 데이터 이벤트를 로그로 기록하고 있는지 확인하는 데 사용할 수 있는 관리형 규칙(`cloudtrail-s3-dataevents-enabled`)을 제공합니다. 자세한 내용은 AWS Config Developer Guide의 `cloudtrail-s3-dataevents-enabled` 단원을 참조하십시오.

AWS Config 활성화

이 주제에 나오는 여러 가지 모범 사례는 AWS Config 규칙 생성을 제안합니다. AWS Config를 사용하면 AWS 리소스의 구성을 평가, 감사 및 측정할 수 있습니다. AWS Config는 리소스 구성을 모니터링하여 원하는 보안 구성을 기준으로 기록된 구성을 평가할 수 있게 해줍니다. AWS Config를 사용하면 AWS 리소스 간 구성 및 관계 변화를 검토하고, 자세한 리소스 구성 기록을 조사하고, 내부 지침에 지정되어 있는 구성을 기준으로 전반적인 규정 준수 여부를 확인할 수 있습니다. 이를 사용하면 규정 준수 감사, 보안 분석, 변경 관리 및 운영 문제 해결 작업을 간소화할 수 있습니다. 자세한 내용은 AWS Config 개발자 안내서의 콘솔을 통해 AWS Config 설정을 참조하십시오. 기록할 리소스 유형을 지정할 때 Amazon S3 리소스를 포함해야 합니다.

AWS Config를 사용하여, 퍼블릭 액세스를 허용하는 Amazon S3 버킷을 모니터링하고 이에 응답하는 방법의 예를 보려면 AWS 보안 블로그의 How to Use AWS Config to Monitor for and Respond to Amazon S3 Buckets Allowing Public Access를 참조하십시오.

Amazon S3와 함께 Amazon Macie 사용 고려

Macie는 기계 학습을 사용하여 AWS에서 민감한 데이터를 자동으로 검색, 분류 및 보호합니다. Macie는 PII(개인 식별 정보) 또는 지적 재산과 같은 민감한 데이터를 인식합니다. 이 데이터에 대한 액세스와 이동이 어떻게 이루어지는지 알 수 있는 대시보드와 알림을 제공합니다. 자세한 내용은 Amazon Macie란 무엇입니까?를 참조하십시오.

AWS 보안 권고 사항 모니터링

AWS 계정의 Trusted Advisor에 게시되는 보안 권고 사항을 정기적으로 확인해야 합니다. 특히 “공개 액세스 권한”이 있는 Amazon S3 버킷에 대한 경고에 유의하십시오. 프로그래밍 방식으로 describe-trusted-advisor-checks를 사용하면 됩니다.

뿐만 아니라, 각 AWS 계정에 등록된 기본 이메일 주소를 적극 모니터링하십시오. 사용자에게 영향을 줄 수 있는 보안 문제가 생기면 AWS에서 이 이메일 주소를 사용하여 연락해 드립니다.

널리 영향을 미치는 AWS 운영 문제는 AWS 서비스 상태 대시보드에 게시됩니다. Personal Health Dashboard를 통해 개별 계정에도 운영 문제가 게시됩니다. 자세한 내용은 AWS Health 설명서를 참조하십시오.

배치 작업 수행

Amazon S3 Batch Operations을 사용하여 Amazon S3 객체에 대해 대규모 배치 작업을 수행할 수 있습니다. Amazon S3 배치 작업은 지정된 Amazon S3 객체의 목록에 대해 단일 작업을 실행할 수 있습니다. 단일 작업은 엑사바이트 규모 데이터를 포함하는 수십억 개 객체에서 지정된 작업을 수행할 수 있습니다. Amazon S3는 진행률을 추적하고, 알림을 전송하고, 모든 작업의 세부 완료 보고서를 저장하여 감사 가능한 완전 관리형 서버리스 경험을 제공합니다. AWS Management 콘솔, AWS CLI, AWS SDK 또는 REST API를 통해 Amazon S3 Batch Operations를 사용할 수 있습니다.

Amazon S3 Batch Operations을 시용하여 객체를 복사하고 객체 태그 또는 액세스 제어 목록(ACL)을 설정합니다. 또한 Amazon S3 Glacier에서 객체 복원을 시작하거나 AWS Lambda 함수를 호출하여 객체를 사용하는 사용자 지정 작업을 수행할 수도 있습니다. 사용자 지정 객체 목록에서 이러한 작업을 수행하거나 Amazon S3 인벤토리 보고서를 사용하여 아주 큰 객체 목록도 쉽게 생성할 수 있습니다. Amazon S3 배치 작업는 Amazon S3에서 이미 사용하는 것과 동일한 Amazon S3 API를 사용하므로 그 인터페이스가 익숙할 것입니다.

주제

- 용어 (p. 502)
- 기본 사항: Amazon S3 배치 작업 건 (p. 502)
- Amazon S3 배치 작업 건 생성 (p. 504)
- 작업 (p. 509)
- 배치 작업 건 관리 (p. 519)
- Amazon S3 배치 작업 예제 (p. 524)

용어

이 단원에서는 작업, 작업(operation) 및 작업(task)이라는 용어를 사용하며, 각각 다음과 같이 정의됩니다.

작업

작업(job)은 Amazon S3 Batch Operations의 기본 작업 단위입니다. 작업은 매니페스트에 나열된 객체에 대해 지정된 작업을 실행하는 데 필요한 모든 정보를 포함합니다. 이 정보를 제공하고 작업 시작을 요청하면 작업은 매니페스트의 각 객체에 대해 작업(operation)을 실행합니다.

작업

작업(operation)은 배치 작업 작업(job)을 실행할 객체 복사와 같은 API 작업의 유형입니다. 각 작업(job)은 매니페스트에 지정된 모든 개체에서 단일 유형의 작업(operation)을 수행합니다.

작업

작업(task)은 작업(job)의 실행 단위입니다. 작업(task)은 하나의 객체에 대해 작업의 작업(operation)을 수행하기 위한 Amazon S3 또는 AWS Lambda API 작업에 대한 단일 호출을 나타냅니다. 작업 수명 주기 동안 Amazon S3 Batch Operations는 매니페스트에 지정된 각 객체에 대해 하나의 작업(task)을 생성합니다.

기본 사항: Amazon S3 배치 작업 건

Amazon S3 Batch Operations를 사용하여 Amazon S3 객체에 대해 대규모 배치 작업을 수행할 수 있습니다. Amazon S3 Batch Operations은 지정된 Amazon S3 객체의 목록에 대해 단일 작업을 실행할 수 있습니다.

주제

- Amazon S3 배치 작업 건의 작동 방식 (p. 503)
- 매니페스트 지정 (p. 503)

Amazon S3 배치 작업 건의 작동 방식

작업(job)은 Amazon S3 Batch Operations의 기본 작업 단위입니다. 작업은 객체 목록에 대해 지정된 작업을 실행하는 데 필요한 모든 정보를 포함합니다.

작업을 생성하려면 Amazon S3 Batch Operations에 객체 목록을 제공하고 해당 객체에서 수행할 작업을 지정합니다. Amazon S3 Batch Operations는 다음 작업(operation)을 지원합니다.

- PUT 객체 복사
- PUT 객체 태그 지정
- PUT 객체 ACL
- 복원 S3 Glacier 시작
- AWS Lambda 함수 호출

작업을 수행할 객체가 매니페스트 객체에 나열됩니다. 작업은 매니페스트에 포함된 각 객체에 대해 지정된 작업을 수행합니다. CSV 형식의 Amazon S3 인벤토리 (p. 459) 보고서를 매니페스트로 사용하면 버킷에 있는 긴 객체 목록을 쉽게 만들 수 있습니다. 또한 단일 버킷에 포함된 객체의 사용자 지정된 목록에 대해 배치 작업을 수행할 수 있는 단순 CSV 형식의 매니페스트를 지정할 수도 있습니다.

작업을 생성하면 Amazon S3가 매니페스트의 객체 목록을 처리하고 각 객체에 지정된 작업(operation)을 실행합니다. 작업이 실행되는 동안 프로그래밍 방식으로 또는 Amazon S3 콘솔을 통해 진행 상황을 모니터링할 수 있습니다. 작업이 완료되면 완료 보고서를 생성하도록 작업을 구성할 수도 있습니다. 완료 보고서는 작업에 의해 실행된 각 작업(task)의 결과를 설명합니다. 작업 모니터링에 대한 자세한 내용은 배치 작업 건 관리 (p. 519) 단원을 참조하십시오.

매니페스트 지정

매니페스트는 Amazon S3가 작업을 수행할 객체 키를 나열하는 Amazon S3 객체입니다. 작업에 대해 매니페스트를 생성하려면 매니페스트 객체, ETag 및 선택적으로 버전 ID를 지정합니다. 매니페스트의 콘텐츠는 URL로 인코딩되어야 합니다. 고객 제공 키를 사용한 서버 측 암호화(SSE-C) 및 AWS Key Management Service 고객 마스터 키(CMK)를 사용한 서버 측 암호화(SSE-KMS)를 사용하는 매니페스트는 지원되지 않습니다. 매니페스트에는 버킷 이름, 객체 키, 각 객체의 객체 버전(선택 사항)이 포함되어야 합니다. 매니페스트의 다른 모든 필드는 Amazon S3 배치 작업에서 사용되지 않습니다.

다음 두 형식 중 하나를 사용하여 작업 생성 요청에 매니페스트를 지정할 수 있습니다.

- Amazon S3 인벤토리 보고서 — CSV 형식 Amazon S3 인벤토리 보고서여야 합니다. 인벤토리 보고서와 연결된 `manifest.json` 파일을 지정해야 합니다. 인벤토리 보고서에 대한 자세한 내용은 Amazon S3 인벤토리 (p. 459) 단원을 참조하십시오. 인벤토리 보고서가 버전 ID를 포함할 경우, Amazon S3 Batch Operations가 특정 객체 버전에만 작용합니다.
- CSV 파일 — 파일 내 각 행이 버킷 이름, 객체 키 및 선택적으로 객체 버전을 포함해야 합니다. 객체 키는 다음 예제와 같이 URL로 인코딩되어야 합니다. 매니페스트는 모든 객체에 대한 버전 ID를 포함하거나 모든 객체에 대한 버전 ID를 생략해야 합니다. CSV 매니페스트 형식에 대한 자세한 내용은 Amazon Simple Storage Service API Reference에서 JobManifestSpec을 참조하십시오.

  다음은 버전 ID가 없는 CSV 형식의 매니페스트 예입니다.

```
Examplebucket,objectkey1
Examplebucket,objectkey2
Examplebucket,objectkey3
Examplebucket,photos/jpgs/objectkey4
Examplebucket,photos/jpgs/newjersey/objectkey5
```

```
Examplebucket,object%20key%20with%20spaces
```

다음은 버전 ID를 포함하는 CSV 형식의 매니페스트 예입니다.

```
Examplebucket,objectkey1,PZ9ibn9D5lP6p298B7S9_ceqx1n5EJ0p
Examplebucket,objectkey2,YY_ouuAJByNW1LRBfFMfxMge7XQWxMBF
Examplebucket,objectkey3,jbo9_jhdPEyB4RrmOxWS0kU0EoNrU_oI
Examplebucket,photos/jpgs/objectkey4,6EqlikJJxLTsHsnbZbSRffn24_eh5Ny4
Examplebucket,photos/jpgs/newjersey/objectkey5,imHf3FAiRsvBW_EHB8GOu.NHunHO1gVs
Examplebucket,object%20key%20with%20spaces,9HkPvDaZY5MVbMhn6TMn1YTb5ArQAo3w
```

> **Important**
>
> 매니페스트의 객체가 버전 지정된 버킷에 있는 경우 객체의 버전 ID를 지정해야 합니다. 작업을 생성하면 Amazon S3 Batch Operations가 작업을 실행하기 전에 전체 매니페스트를 구문 분석합니다. 하지만 버킷 상태 "스냅샷"을 생성하지는 않습니다.
> 매니페스트에는 수십억 개의 객체가 포함될 수 있으므로 작업을 실행하는 데 오래 걸릴 수 있습니다. 해당 객체에 대한 버전 ID를 지정하지 않은 상태에서 작업이 실행되는 동안 새 버전으로 객체를 덮어쓰면 Amazon S3는 작업 생성 시 존재했던 버전이 아니라 객체의 최신 버전에서 작업(operation)을 수행합니다. 이 문제를 피하는 유일한 방법은 매니페스트에 나열된 객체에 버전 ID를 지정하는 것입니다.

Amazon S3 배치 작업 건 생성

Amazon S3 Batch Operations를 사용하여 특정 Amazon S3 객체 목록에 대해 대규모 배치 작업을 수행할 수 있습니다. AWS Management 콘솔, AWS Command Line Interface(AWS CLI), AWS SDK 또는 REST API를 사용하여 S3 배치 작업 작업을 생성할 수 있습니다.

이 단원에서는 Amazon S3 Batch Operations 작업을 생성하는 데 필요한 정보를 설명합니다. `Create Job` 요청의 결과에 대해서도 설명합니다.

> **Note**
>
> Amazon S3 콘솔을 사용하여 작업을 생성하는 방법에 대한 단계별 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 Amazon S3 배치 작업의 작업 생성을 참조하십시오.

작업 생성 요청

배치 작업 작업을 생성하려면 다음 정보를 제공해야 합니다.

작업

매니페스트 내 객체에 대해 Amazon S3 Batch Operations가 실행할 작업을 지정합니다. 각 작업 유형은 해당 작업과 관련된 파라미터를 허용합니다. 이를 통해 각 객체에서 작업을 하나씩 수행하는 것과 동일한 작업을 수행할 수 있습니다.

매니페스트

매니페스트는 Amazon S3 Batch Operations에서 지정된 작업을 실행해야 하는 모든 객체의 목록입니다. CSV 형식 Amazon S3 인벤토리 (p. 459) 보고서를 매니페스트로 사용하거나 고유한 사용자 지정 CSV 객체 목록을 사용할 수 있습니다.

자세한 내용은 매니페스트 지정 (p. 503) 단원을 참조하십시오.

우선 순위

작업 우선 순위를 사용하여 이 작업의 상대적 우선 순위를 계정에서 실행 중인 다른 작업에 표시합니다. 숫자가 높을수록 우선 순위가 높아집니다.

작업 우선 순위는 동일한 계정 및 리전의 다른 작업에 대해 설정된 우선 순위와 관련되어야만 의미가 있습니다. 따라서 자신에게 적합한 번호 지정 시스템을 선택할 수 있습니다. 예를 들어, 모든 `Initiate Restore Object` 작업에 우선 순위 1, 모든 `PUT Object Copy` 작업에 우선 순위 2, 모든 `Put Object ACL` 작업에 우선 순위 3을 할당할 수 있습니다.

배치 작업은 우선 순위 번호에 따라 작업 우선 순위를 부여하지만 정확한 순서가 보장되지 않습니다. 따라서 어떤 작업이 다른 작업보다 먼저 시작되거나 끝나기 위해서는 작업 우선순위를 사용해서는 안 됩니다. 정확한 순서가 필요한 경우 다음 작업을 시작하기 전에 하나의 작업이 완료될 때까지 기다려야 합니다.

RoleArn

작업을 실행할 AWS Identity and Access Management(IAM) 역할을 지정합니다. 사용하는 IAM 역할에는 작업에 지정된 작업(operation)을 수행하기에 충분한 권한이 있어야 합니다. 예를 들어, `PUT Object Copy` 작업을 실행하려면 IAM 역할이 원본 버킷에 대한 `s3:GetObject` 권한과 대상 버킷에 대한 `s3:PutObject` 권한을 가져야 합니다. 또한 역할에는 매니페스트를 읽고 작업 완료 보고서를 작성할 수 있는 권한도 필요합니다.

IAM 역할에 대한 자세한 내용은 IAM 사용 설명서의 IAM 역할을 참조하십시오.

Amazon S3 권한에 대한 자세한 정보는 Amazon S3 작업 (p. 342)를 참조하십시오.

보고서

Amazon S3 Batch Operations이 작업 완료 보고서를 생성할 것인지 지정합니다. 작업 완료 보고서를 요청하면 이 요소에서 보고서에 대한 파라미터도 제공해야 합니다. 필요한 정보에는 보고서를 저장할 버킷, 보고서 형식, 보고서에 모든 작업의 세부 정보를 포함할지 실패한 작업만 포함할지 여부, 선택적 접두사 문자열이 포함됩니다.

태그(선택 사항)

태그를 추가하여 Amazon S3 Batch Operations 작업에 대한 레이블을 지정하고 액세스를 제어할 수 있습니다. 태그는 배치 작업 작업 담당자를 식별하는 데 사용할 수 있습니다. 태그가 연결된 작업을 생성할 수 있으며 작업을 생성한 후 작업에 태그를 추가할 수 있습니다. 예를 들어, `"Department=Finance"` 태그로 작업이 생성된 경우 `CreateJob`을 호출할 수 있는 권한을 IAM 사용자에게 부여할 수 있습니다.

자세한 내용은 the section called “태그를 사용하여 액세스 제어 및 작업 레이블 지정” (p. 520) 단원을 참조하십시오.

설명(선택 사항)

작업을 추적하고 모니터링하기 위해 최대 256자의 설명을 제공할 수도 있습니다. Amazon S3는 작업에 대한 정보를 반환하거나 Amazon S3 콘솔에 작업 세부 정보를 표시할 때마다 이 설명을 포함합니다. 그러므로 지정한 설명에 따라 손쉽게 작업을 정렬하고 필터링할 수 있습니다. 설명이 고유할 필요는 없으므로 설명을 범주(예: "주간 로그 복사 작업")로 사용하면 유사한 작업 그룹을 추적하는 데 도움이 됩니다.

작업 생성 요청

`Create Job` 요청이 성공하면 Amazon S3가 작업 ID를 반환합니다. 작업 ID는 Amazon S3가 자동으로 생성하는 고유 식별자이므로 배치 작업 작업을 식별하고 상태를 모니터링할 수 있습니다.

AWS CLI, AWS SDK 또는 REST API를 통해 작업을 생성할 때 자동으로 작업 처리를 시작하도록 Amazon S3 Batch Operations를 설정할 수 있습니다. 작업은 준비가 완료되는 즉시 실행되고 우선 순위가 더 높은 작업이 완료되도록 대기하지 않습니다.

AWS Management 콘솔을 통해 작업을 생성할 때 작업 세부 정보를 검토하고 배치 작업가 처리를 시작하기 전에 실행할 것인지 확인해야 합니다. 작업을 실행하기로 확인하면, 마치 다른 방법 중 하나를 통해 작업을 생성한 것처럼 작업이 진행됩니다. 작업이 30일 이상 일시 중지된 상태로 있으면 작업이 실패합니다.

Amazon S3 배치 작업에 대한 권한 부여

이 단원에서는 Amazon S3 Batch Operations의 작업을 생성하고 수행하는 데 필요한 권한을 부여하는 방법을 설명합니다.

주제

- Amazon S3 배치 작업(Operation)의 작업(Job) 생성에 필요한 권한 (p. 506)
- Amazon S3 배치 작업 IAM 역할 생성 (p. 506)

Amazon S3 배치 작업(Operation)의 작업(Job) 생성에 필요한 권한

Amazon S3 배치 작업의 작업(job)을 생성하려면 `s3:CreateJob` 권한이 필요합니다. 또한 작업을 생성하는 동일한 개체는 `iam:PassRole` 권한이 있어야 작업에 지정된 AWS Identity and Access Management(IAM) 역할을 Amazon S3 배치 작업에 전달할 수 있습니다. 이 IAM 역할 생성에 대한 자세한 내용은 다음 Amazon S3 배치 작업 IAM 역할 생성 (p. 506) 주제를 참조하십시오.

Amazon S3 배치 작업 IAM 역할 생성

Amazon S3에게는 사용자 대신 배치 작업을 수행할 권한이 있어야 합니다. AWS Identity and Access Management(IAM) 역할을 통해 그와 같은 권한을 부여합니다. 이 단원에서는 IAM 역할을 만들 때 사용하는 신뢰 및 권한 정책의 예제를 제공합니다. 자세한 내용은 IAM 역할을 참조하십시오. 예제는 예제: 작업 태그를 사용하여 Amazon S3 Batch Operations에 대한 권한 제어 (p. 524)을 참조하십시오.

IAM 정책에서 조건 키를 사용하여 Amazon S3 배치 작업 건에 대한 액세스 권한을 필터링할 수도 있습니다. 자세한 내용과 Amazon S3-특정 조건 키의 전체 목록은 Amazon S3에 사용되는 작업, 리소스 및 조건 키 (p. 356) 단원을 참조하십시오.

다음 동영상은 AWS Management 콘솔을 사용하여 배치 작업 작업에 대해 IAM 권한을 설정하는 방법을 보여줍니다.

신뢰 정책

Amazon S3 Batch Operations 서비스 보안 주체가 IAM 역할을 수임하도록 허용하려면 다음 신뢰 정책을 역할에 연결하십시오.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

권한 정책

작업(operation) 유형에 따라 다음 정책 중 하나를 연결할 수 있습니다.

> **Note**
>
> - 작업(operation)과 상관없이, Amazon S3는 S3 버킷에서 매니페스트 객체를 읽고 선택적으로 버킷에 보고서를 쓸 수 있는 권한이 필요합니다. 따라서 다음의 모든 정책은 이러한 권한을 포함합니다.

- Amazon S3 인벤토리 보고서 매니페스트의 경우 Amazon S3 Batch Operations에는 manifest.json 객체와 모든 관련 CSV 데이터 파일을 읽을 수 있는 권한이 필요합니다.
- 객체의 버전 ID를 지정할 때는 `s3:GetObjectVersion`과 같은 버전 관련 권한만 필요합니다.
- 암호화된 객체에서 Amazon S3 Batch Operations를 실행 중인 경우 IAM 역할은 해당 객체를 암호화하는 데 사용된 AWS KMS 키에 액세스해야 합니다.

- PUT 객체 복사

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectTagging"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::{{DestinationBucket}}/*"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectTagging"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::{{SourceBucket}}/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::{{ManifestBucket}}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::{{ReportBucket}}/*"
            ]
        }
    ]
}
```

- PUT 객체 태그 지정

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObjectTagging",
```

```
        "s3:PutObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::{{TargetResource}}/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::{{ManifestBucket}}/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject",
        "s3:GetBucketLocation"
      ],
      "Resource":[
        "arn:aws:s3:::{{ReportBucket}}/*"
      ]
    }
  ]
}
```

- PUT 객체 ACL

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObjectAcl",
        "s3:PutObjectVersionAcl"
      ],
      "Resource": "arn:aws:s3:::{{TargetResource}}/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::{{ManifestBucket}}/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject",
        "s3:GetBucketLocation"
      ],
      "Resource":[
        "arn:aws:s3:::{{ReportBucket}}/*"
      ]
    }
  ]
}
```

- S3 Glacier 복원 시작

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
          "s3:RestoreObject"
      ],
      "Resource": "arn:aws:s3:::{{TargetResource}}/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::{{ManifestBucket}}/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject",
        "s3:GetBucketLocation"
      ],
      "Resource":[
        "arn:aws:s3:::{{ReportBucket}}/*"
      ]
    }
  ]
}
```

관련 리소스

- 기본 사항: Amazon S3 배치 작업 건 (p. 502)
- 작업 (p. 509)
- 배치 작업 건 관리 (p. 519)

작업

Amazon S3 Batch Operations는 5가지 작업(operation)을 지원합니다. 이 주제의 단원에서는 각 작업(operation)에 대해 설명합니다.

주제

- PUT 객체 복사 (p. 510)
- 객체 복원 시작 (p. 510)
- Amazon S3 배치 작업에서 Lambda 함수 호출 (p. 511)
- PUT 객체 ACL (p. 518)
- PUT 객체 태그 지정 (p. 519)

PUT 객체 복사

PUT 객체 복사 작업은 매니페스트에 지정된 각 객체를 복사합니다. 객체를 동일한 AWS 리전의 다른 버킷 또는 다른 리전의 버킷으로 복사할 수 있습니다. Amazon S3 Batch Operations은 객체를 복사할 때 Amazon S3를 통해 사용 가능한 대부분의 옵션을 지원합니다. 이러한 옵션에는 객체 메타데이터 설정, 권한 설정, 객체 스토리지 클래스 변경이 포함됩니다. 객체 복사를 위해 Amazon S3에서 사용 가능한 기능에 대한 자세한 내용은 객체 복사 (p. 209) 단원을 참조하십시오.

규제 및 제한

- 모든 원본 객체가 한 버킷에 있어야 합니다.
- 모든 대상 객체가 한 버킷에 있어야 합니다.
- 원본 버킷에 대한 읽기 권한과 대상 버킷에 대한 쓰기 권한이 있어야 합니다.
- 복사할 객체는 크기가 최대 5GB까지 가능합니다.
- PUT 객체 복사 작업은 객체를 복사하려는 대상 리전에서 생성해야 합니다.
- ETag 및 고객 제공 암호화 키를 사용한 서버 측 암호화에 대한 조건부 검사를 제외하고 모든 PUT 객체 복사 옵션이 지원됩니다.
- 버킷이 버전 관리되지 않는 경우 키 이름이 동일한 객체를 덮어씁니다.
- 객체는 반드시 매니페스트에 나열된 것과 동일한 순서로 복사되는 것은 아닙니다. 따라서 버전이 지정된 버킷의 경우, 최신/최신이 아닌 버전 순서를 유지하는 것이 중요하다면 먼저 최신이 아닌 모든 버전을 복사하고 나중에 첫 번째 작업이 완료된 후 후속 작업에서 최신 버전을 복사해야 합니다.
- RRS(Reduced Redundancy Storage) 스토리지 클래스로 객체를 복사하는 것은 지원되지 않습니다.

객체 복원 시작

Amazon S3 Batch Operations를 사용하여 Amazon S3 객체에 대해 대규모 배치 작업를 수행할 수 있습니다. Amazon S3 Batch Operations는 Amazon S3 Glacier에서 객체 복원 시작을 포함하여 지정된 Amazon S3 객체의 목록에 대해 단일 작업을 실행할 수 있습니다. 자세한 내용은 배치 작업 (p. 502) 단원을 참조하십시오.

S3 Glacier 또는 S3 Glacier Deep Archive 스토리지 클래스에 보관하는 객체는 실시간으로 액세스할 수 없습니다. Amazon S3 Batch Operations의 `InitiateRestore` 작업은 매니페스트에 지정된 각 객체에 대한 복원 요청을 S3 Glacier로 보냅니다. 객체 복원 시작 작업을 생성하려면 요청에 두 가지 요소를 포함해야 합니다.

- ExpirationInDays

  S3 Glacier에서 객체를 복원할 경우 복원된 객체는 단지 임시 복사본으로, 일정 기간이 경과하면 Amazon S3가 이들을 삭제합니다. 이 요소는 Amazon S3가 임시 복사본을 유지하는 기간을 지정합니다. 임시 복사본이 만료된 후에는 S3 Glacier에서 객체를 다시 복원해야만 검색이 가능합니다. 객체 복구에 대한 자세한 내용은 보관된 객체의 복원 (p. 247) 단원을 참조하십시오.

- GlacierJobTier

  Amazon S3는 신속, 표준, 벌크 등 3가지 검색 티어에 따라 S3 Glacier에서 객체를 복원할 수 있습니다. Amazon S3 Batch Operations는 표준 및 벌크 티어만 지원합니다. S3 Glacier 검색 티어에 대한 자세한 내용은 아카이브 검색 옵션 (p. 248) 단원을 참조하십시오. 각 티어 요금에 대한 자세한 내용은 Amazon S3 Glacier 요금에서 "검색 요금"을 참조하십시오.

  Important

  객체 복원 시작 작업은 객체 복원 요청만 시작합니다. 각 객체에 대해 요청이 시작되면 Amazon S3 Batch Operations가 해당 객체에 대해 작업으로 완료로 보고합니다. Amazon S3는 객체 복원 시 작

업을 업데이트하거나 알림을 제공하지 않습니다. 하지만 이벤트 알림을 사용하여 Amazon S3에서 객체를 사용할 수 있을 때 알림을 받을 수 있습니다. 자세한 내용은 Amazon S3 이벤트 알림 구성 (p. 587) 단원을 참조하십시오.

중복 복원

객체 복원 시작 작업이 이미 복원 과정 중에 있는 객체를 복원하려고 시도할 경우 Amazon S3 Batch Operations가 다음과 같이 동작합니다.

다음 조건 중 하나가 true일 경우 해당 객체의 복원 작업은 성공합니다.

- 이미 진행 중인 복원 요청과 비교해, 이 작업의 `ExpirationInDays`가 동일하고 `GlacierJobTier`가 더 빠른 경우.
- 이전 복원 요청이 이미 완료되었으며 현재 객체를 사용할 수 있습니다. 이 경우, Amazon S3 Batch Operations가 복원된 객체의 만료 날짜를 이 작업에 지정된 `ExpirationInDays`와 일치하도록 업데이트합니다.

다음 조건 중 하나가 true일 경우 해당 객체의 복원 작업은 실패합니다.

- 이미 진행 중인 복원 요청이 아직 완료되지 않고 이 작업의 복원 기간(`ExpirationInDays`로 지정됨)이 진행 중인 복원 요청에 지정된 복원 기간과 다른 경우.
- 이 작업의 복원 티어(`GlacierJobTier`로 지정됨)이 이미 진행 중인 복원 요청에 지정된 복원 티어보다 느리거나 같은 경우.

제한 사항

객체 복원 시작 작업에는 다음 제한이 적용됩니다.

- 보관된 객체와 동일한 리전에서 객체 복원 시작 작업을 생성해야 합니다.
- Amazon S3 Batch Operations는 S3 Glacier SELECT를 지원하지 않습니다.
- Amazon S3 Batch Operations는 신속 검색 티어를 지원하지 않습니다.

Amazon S3 배치 작업에서 Lambda 함수 호출

Amazon S3 Batch Operations은 AWS Lambda 함수를 호출하여 매니페스트에 나열된 객체에 대한 사용자 지정 작업을 수행할 수 있습니다. 이 섹션에서는 Amazon S3 Batch Operations와 함께 사용할 Lambda 함수 생성 방법과 함수 호출 작업 생성 방법을 설명합니다. Amazon S3 Batch Operations 작업은 `LambdaInvoke` 작업을 사용하여 매니페스트에 나열된 각 객체에 대해 Lambda 함수를 실행합니다.

AWS Management 콘솔, AWS Command Line Interface(AWS CLI), AWS SDK 또는 REST API를 사용하여 Lambda에 대해 Amazon S3 Batch Operations로 작업할 수 있습니다. Lambda 사용에 대한 자세한 내용은 AWS Lambda Developer Guide에서 AWS Lambda 시작하기 항목을 참조하십시오.

다음 섹션은 Amazon S3 Batch Operations을 Lambda와 함께 사용하여 시작할 수 있는 방법에 대해 설명합니다.

주제

- Amazon S3 배치 작업으로 Lambda 사용 (p. 512)
- Amazon S3 배치 작업과 함께 사용할 Lambda 함수 생성 (p. 512)
- Lambda 함수를 호출하는 Amazon S3 배치 작업 건 생성 (p. 515)
- Lambda 매니페스트에서 작업 수준 정보 제공 (p. 516)

Amazon S3 배치 작업으로 Lambda 사용

AWS Lambda에 Amazon S3 Batch Operations을 사용할 때 Amazon S3 Batch Operations과 함께 사용할 새로운 Lambda 함수를 생성해야 합니다. 기존의 Amazon S3 이벤트 기반 함수를 Amazon S3 Batch Operations와 함께 재사용할 수는 없습니다. 이벤트 함수는 메시지만 받을 수 있습니다. 메시지를 반환하지 않습니다. Amazon S3 Batch Operations와 함께 사용되는 Lambda 함수는 메시지를 수락하여 반환해야 합니다. Amazon S3와 함께 Lambda 사용에 대한 자세한 내용은 AWS Lambda Developer Guide에서 AWS Lambda을 Amazon S3과 함께 사용 항목을 참조하십시오.

Lambda 함수를 호출하는 Amazon S3 Batch Operations 작업을 생성합니다. 이 작업은 매니페스트에 나열된 모든 객체에 대해 동일한 Lambda 함수를 실행합니다. 매니페스트의 객체를 처리하는 동안 사용할 Lambda 함수의 버전을 제어할 수 있습니다. Amazon S3 Batch Operations은 비한정적 Amazon 리소스 이름(ARN), 별칭 및 특정 버전을 지원합니다. 자세한 내용은 AWS Lambda Developer Guide의 AWS Lambda 버전 관리 소개 항목을 참조하십시오.

별칭 또는 `$LATEST` 한정어를 사용하는 함수 ARN에 Amazon S3 Batch Operations 작업을 제공하고 해당 버전 중 하나를 업데이트하면 Amazon S3 Batch Operations에서 Lambda 함수의 새 버전 호출을 시작합니다. 이는 라지 작업을 통해 기능 부분을 업데이트하고자 할 때 유용할 수 있습니다. Amazon S3 Batch Operations에서 사용되는 버전을 변경하지 않으려면 작업을 생성할 때 `FunctionARN` 파라미터에 특정 버전을 제공합니다.

응답 및 결과 코드

Amazon S3 Batch Operations이 Lambda 함수에서 기대하는 코드에는 두 가지 레벨이 있습니다. 첫 번째는 전체 요청에 대한 응답 코드이고 두 번째는 작업별 결과 코드입니다. 다음 표는 응답 코드를 포함합니다.

Response Code	설명
`Succeeded`	작업이 정상적으로 완료되었습니다. 작업 완료 보고서를 요청한 경우 작업의 결과 문자열이 보고서에 포함됩니다.
`TemporaryFailure`	작업이 일시적으로 실패하여 작업이 완료되기 전에 리드라이브됩니다. 결과 문자열은 무시됩니다. 이것이 최종 리드라이브인 경우 오류 메시지가 최종 보고서에 포함됩니다.
`PermanentFailure`	작업에 영구 실패가 발생했습니다. 작업 완료 보고서를 요청한 경우 작업은 `Failed`로 표시되고 오류 메시지 문자열을 포함합니다. 실패한 작업의 결과 문자열은 무시됩니다.

Amazon S3 배치 작업과 함께 사용할 Lambda 함수 생성

이 섹션에는 Lambda 함수와 함께 사용해야 하는 예제 AWS Identity and Access Management(IAM) 권한이 나와 있습니다. 또한 Amazon S3 Batch Operations와 함께 사용할 예제 Lambda 함수가 포함되어 있습니다. 이전에 Lambda 함수를 생성한 적이 없다면 AWS Lambda Developer Guide에서 자습서: Amazon S3과 함께 AWS Lambda 사용을 참조하십시오.

Amazon S3 Batch Operations와 함께 사용하기 위해 특별히 Lambda 함수를 생성해야 합니다. 기존의 Amazon S3 이벤트 기반 Lambda 함수를 재사용할 수는 없습니다. Amazon S3 Batch Operations에 사용되는 Lambda 함수는 특수 데이터 필드를 수락 및 반환해야 하기 때문입니다.

예제 IAM 권한

다음은 Lambda 함수를 Amazon S3 Batch Operations과 함께 사용하는 데 필요한 IAM 권한의 예제입니다.

Example — Amazon S3 배치 작업 신뢰 정책

다음은 배치 작업 IAM 역할에서 사용할 수 있는 신뢰 정책의 예제입니다. 이 IAM 역할은 작업을 생성할 때 지정되며 IAM 역할을 수임할 배치 작업 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "batchoperations.s3.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Example — Lambda IAM 정책

다음은 Lambda 함수를 호출하고 입력 매니페스트를 읽는 Amazon S3 Batch Operations 권한을 제공하는 IAM 정책의 예제입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "BatchOperationsLambdaPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:PutObject",
                "lambda:InvokeFunction"
            ],
            "Resource": "*"
        }
    ]
}
```

예제 요청 및 응답

이 섹션에는 Lambda 함수에 대한 요청 및 응답 예제가 나와 있습니다.

Example 요청

다음은 Lambda 함수 요청에 대한 JSON 예제입니다.

```
{
    "invocationSchemaVersion": "1.0",
    "invocationId": "YXNkbGZqYWRmaiBhc2RmdW9hZHNmZGpmaGFzbGtkaGZza2RmaAo",
    "job": {
        "id": "f3cc4f60-61f6-4a2b-8a21-d07600c373ce"
    },
    "tasks": [
        {
            "taskId": "dGFza2lkZ29lc2hlcmUK",
            "s3Key": "customerImage1.jpg",
            "s3VersionId": "1",
            "s3BucketArn": "arn:aws:s3:us-east-1:0123456788:awsexamplebucket"
        }
```

```
    ]
}
```

Example 응답

다음은 Lambda 함수에 대한 JSON 응답의 예제입니다.

```
{
  "invocationSchemaVersion": "1.0",
  "treatMissingKeysAs" : "PermanentFailure",
  "invocationId" : "YXNkbGZqYWRmaiBhc2RmdW9hZHNmZGpmaGFzbGtkaGZza2RmaAo",
  "results": [
    {
      "taskId": "dGFza2lkZ29lc2hlcmUK",
      "resultCode": "Succeeded",
      "resultString": "[\"Mary Major", \"John Stiles\"]"
    }
  ]
}
```

Amazon S3 배치 작업에 대한 Lambda 함수 예제

다음의 Python Lambda 함수 예제에서는 반복하여 매니페스트를 복사하고 각 객체의 이름을 바꿉니다.

예제에서 알 수 있듯이 Amazon S3 Batch Operations의 키는 URL로 인코딩됩니다. Amazon S3를 다른 AWS 서비스와 함께 사용하려면 Amazon S3 Batch Operations에서 전달된 키를 URL 디코딩해야 합니다.

```
import boto3
import urllib
from botocore.exceptions import ClientError

def lambda_handler(event, context):
    # Instantiate boto client
    s3Client = boto3.client('s3')

    # Parse job parameters from Amazon S3 batch operations
    jobId = event['job']['id']
    invocationId = event['invocationId']
    invocationSchemaVersion = event['invocationSchemaVersion']

    # Prepare results
    results = []

    # Parse Amazon S3 Key, Key Version, and Bucket ARN
    taskId = event['tasks'][0]['taskId']
    s3Key = urllib.unquote(event['tasks'][0]['s3Key']).decode('utf8')
    s3VersionId = event['tasks'][0]['s3VersionId']
    s3BucketArn = event['tasks'][0]['s3BucketArn']
    s3Bucket = s3BucketArn.split(':::')[-1]

    # Construct CopySource with VersionId
    copySrc = {'Bucket': s3Bucket, 'Key': s3Key}
    if s3VersionId is not None:
        copySrc['VersionId'] = s3VersionId

    # Copy object to new bucket with new key name
    try:
        # Prepare result code and string
        resultCode = None
        resultString = None

        # Construct New Key
        newKey = rename_key(s3Key)
```

```
        newBucket = 'destination-bucket-name'

        # Copy Object to New Bucket
        response = s3Client.copy_object(
            CopySource = copySrc,
            Bucket = newBucket,
            Key = newKey
        )

        # Mark as succeeded
        resultCode = 'Succeeded'
        resultString = str(response)
    except ClientError as e:
        # If request timed out, mark as a temp failure
        # and Amason S3 batch operations will make the task for retry. If
        # any other exceptions are received, mark as permanent failure.
        errorCode = e.response['Error']['Code']
        errorMessage = e.response['Error']['Message']
        if errorCode == 'RequestTimeout':
            resultCode = 'TemporaryFailure'
            resultString = 'Retry request to Amazon S3 due to timeout.'
        else:
            resultCode = 'PermanentFailure'
            resultString = '{}: {}'.format(errorCode, errorMessage)
    except Exception as e:
        # Catch all exceptions to permanently fail the task
        resultCode = 'PermanentFailure'
        resultString = 'Exception: {}'.format(e.message)
    finally:
        results.append({
            'taskId': taskId,
            'resultCode': resultCode,
            'resultString': resultString
        })

    return {
        'invocationSchemaVersion': invocationSchemaVersion,
        'treatMissingKeysAs': 'PermanentFailure',
        'invocationId': invocationId,
        'results': results
    }

def rename_key(s3Key):
    # Rename the key by adding additional suffix
    return s3Key + '_new_suffix'
```

Lambda 함수를 호출하는 Amazon S3 배치 작업 건 생성

Lambda 함수를 호출하기 위해 Amazon S3 Batch Operations 작업을 생성할 때 다음을 제공해야 합니다.

- Lambda 함수의 ARN(함수 별칭 또는 특정 버전 번호를 포함할 수도 있음)
- 함수 호출 권한이 있는 IAM 역할
- 작업 파라미터 `LambdaInvokeFunction`

Amazon S3 Batch Operations 작업 생성에 대한 자세한 내용은 Amazon S3 배치 작업 건 생성 (p. 504) 및 작업 (p. 509) 항목을 참조하십시오.

다음 예제에서는 AWS CLI를 사용하여 Lambda 함수를 호출하는 Amazon S3 Batch Operations 작업을 생성합니다.

```
aws s3control create-job
```

```
    --account-id <AccountID>
    --operation  '{"LambdaInvoke": { "FunctionArn":
 "arn:aws:lambda:Region:AccountID:function:LambdaFunctionName" } }'
    --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
["Bucket","Key"]},"Location":
{"ObjectArn":"arn:aws:s3:::ManifestLocation","ETag":"ManifestETag"}}'
    --report
 '{"Bucket":"arn:aws:s3:::awsexamplebucket","Format":"Report_CSV_20180820","Enabled":true,"Prefix":"Rep
    --priority 2
    --role-arn arn:aws:iam::AccountID:role/BatchOperationsRole
    --region Region
    --description “Lambda Function"
```

Lambda 매니페스트에서 작업 수준 정보 제공

Amazon S3 배치 작업에서 AWS Lambda 함수를 사용할 때 운영되는 각 작업/키를 포함하는 추가 데이터를 원할 수 있습니다. 예를 들어 원본 객체 키와 새로운 객체 키가 모두 제공되도록 하고 싶을 수 있습니다. 이렇게 하면 Lambda 함수는 새 S3 버킷에 원본을 새로운 이름으로 복사할 수 있습니다. 기본적으로 Amazon S3 배치 작업은 작업에 대해 입력 매니페스트의 대상 버킷과 원본 키 목록만 지정할 수 있도록 합니다. 아래에는 보다 복잡한 Lambda 함수를 실행할 수 있도록 매니페스트에 추가 데이터를 포함시킬 수 있는 방법이 나와 있습니다.

Amazon S3 배치 작업 매니페스트에 Lambda 함수 코드에서 사용할 키별 파라미터를 지정하려면 다음과 같이 URL 인코딩된 JSON 형식을 사용하십시오. 마치 Amazon S3 객체 키가 있는 것처럼 `key` 필드가 Lambda 함수로 전달됩니다. 그러나 아래에서와 같이 다른 값이나 여러 키를 포함시키기 위해 Lambda 함수에 의해 해석될 수 있습니다.

> **Note**
>
> 매니페스트에서 `key` 필드의 최대 문자 수는 1,024자입니다.

Example — Amazon S3 키"를 JSON 문자열로 대체하는 매니페스트

URL 인코딩 버전을 Amazon S3 배치 작업에 제공해야 합니다.

```
my-bucket,{"origKey": "object1key", "newKey": "newObject1Key"}
my-bucket,{"origKey": "object2key", "newKey": "newObject2Key"}
my-bucket,{"origKey": "object3key", "newKey": "newObject3Key"}
```

Example — URL 인코딩된 매니페스트

이러한 URL 인코딩 버전을 Amazon S3 배치 작업에 제공해야 합니다. URL 인코딩 버전이 아니면 작동하지 않습니다.

```
my-bucket,%7B%22origKey%22%3A%20%22object1key%22%2C%20%22newKey%22%3A%20%22newObject1Key
%22%7D
my-bucket,%7B%22origKey%22%3A%20%22object2key%22%2C%20%22newKey%22%3A%20%22newObject2Key
%22%7D
my-bucket,%7B%22origKey%22%3A%20%22object3key%22%2C%20%22newKey%22%3A%20%22newObject3Key
%22%7D
```

Example — 작업 보고서에 결과를 기록하는 매니페스트 형식을 지원하는 Lambda 함수

이 Lambda 함수는 Amazon S3 배치 작업 매니페스트로 인코딩된 JSON을 구문 분석하는 방법을 보여줍니다.

```
import json
```

```
from urllib.parse import unquote_plus


# This example Lambda function shows how to parse JSON that is encoded into the Amazon S3
 batch
# operations manifest containing lines like this:
#
# bucket,encoded-json
# bucket,encoded-json
# bucket,encoded-json
#
# For example, if we wanted to send the following JSON to this Lambda function:
#
# bucket,{"origKey": "object1key", "newKey": "newObject1Key"}
# bucket,{"origKey": "object2key", "newKey": "newObject2Key"}
# bucket,{"origKey": "object3key", "newKey": "newObject3Key"}
#
# We would simply URL-encode the JSON like this to create the real manifest to create a
 batch
# operations job with:
#
# my-bucket,%7B%22origKey%22%3A%20%22object1key%22%2C%20%22newKey%22%3A%20%22newObject1Key
%22%7D
# my-bucket,%7B%22origKey%22%3A%20%22object2key%22%2C%20%22newKey%22%3A%20%22newObject2Key
%22%7D
# my-bucket,%7B%22origKey%22%3A%20%22object3key%22%2C%20%22newKey%22%3A%20%22newObject3Key
%22%7D
#
def lambda_handler(event, context):
    # Parse job parameters from Amazon S3 batch operations
    jobId = event['job']['id']
    invocationId = event['invocationId']
    invocationSchemaVersion = event['invocationSchemaVersion']

    # Prepare results
    results = []

    # S3 batch operations currently only passes a single task at a time in the array of
 tasks.
    task = event['tasks'][0]

    # Extract the task values we might want to use
    taskId = task['taskId']
    s3Key = task['s3Key']
    s3VersionId = task['s3VersionId']
    s3BucketArn = task['s3BucketArn']
    s3BucketName = s3BucketArn.split(':::')[-1]

    try:
        # Assume it will succeed for now
        resultCode = 'Succeeded'
        resultString = ''

        # Decode the JSON string that was encoded into the S3 Key value and convert the
        # resulting string into a JSON structure.
        s3Key_decoded = unquote_plus(s3Key)
        keyJson = json.loads(s3Key_decoded)

        # Extract some values from the JSON that we might want to operate on.  In this
 example
        # we won't do anything except return the concatenated string as a fake result.
        newKey = keyJson['newKey']
        origKey = keyJson['origKey']
        resultString = origKey + " --> " + newKey

    except Exception as e:
```

```
        # If we run into any exceptions, fail this task so batch operations does not retry
it and
        # return the exception string so we can see the failure message in the final report
        # created by batch operations.
        resultCode = 'PermanentFailure'
        resultString = 'Exception: {}'.format(e)
    finally:
        # Send back the results for this task.
        results.append({
            'taskId': taskId,
            'resultCode': resultCode,
            'resultString': resultString
        })

    return {
        'invocationSchemaVersion': invocationSchemaVersion,
        'treatMissingKeysAs': 'PermanentFailure',
        'invocationId': invocationId,
        'results': results
    }
```

PUT 객체 ACL

PUT 객체 ACL 작업은 매니페스트에 나열된 각 객체의 Amazon S3 액세스 제어 목록(ACL)을 바꿉니다. ACL을 사용하여 사용자가 액세스할 수 잇는 객체와 수행할 수 있는 작업을 정의할 수 있습니다.

Amazon S3 Batch Operations는 사용자가 정의하는 사용자 지정 ACL과 Amazon S3가 사전 정의한 액세스 권한 집합을 포함하는 미리 제공된 ACL을 지원합니다.

매니페스트의 객체가 버전 지정된 버킷에 있는 경우 각 객체의 특정 버전에 ACL을 적용할 수 있습니다. 이렇게 하려면 매니페스트의 각 객체에 버전 ID를 지정합니다. 객체에 대한 버전 ID를 포함하지 않을 경우 Amazon S3 Batch Operations가 해당 객체의 최신 버전에 ACL을 적용합니다.

> **Note**
>
> 버킷의 모든 객체에 대한 퍼블릭 액세스를 제한하려면 Amazon S3 Batch Operations 대신 Amazon S3 퍼블릭 액세스 차단을 사용해야 합니다. 퍼블릭 액세스 차단은 빠르게 효과를 발휘하는 간단한 단일 작업으로 버킷별로 또는 계정 전체에서 퍼블릭 액세스를 제한할 수 있습니다. 이 기능은 버킷 또는 계정의 모든 객체에 대한 퍼블릭 액세스를 제어하려는 경우 더 나은 선택입니다. 매니페스트의 각 객체에 대해 사용자 지정된 ACL을 적용해야 할 경우에는 Amazon S3 Batch Operations를 사용합니다. Amazon S3 퍼블릭 액세스 차단에 대한 자세한 내용은 Amazon S3 퍼블릭 액세스 차단 사용 (p. 450) 단원을 참조하십시오.

규제 및 제한

- PUT 객체 ACL 작업을 실행하도록 지정하는 역할에는 기본 Amazon S3 PUT 객체 ACL 작업을 수행할 권한이 있어야 합니다. 필요한 권한에 대한 자세한 내용은 Amazon Simple Storage Service API Reference에서 PUT 객체 ACL을 참조하십시오.
- Amazon S3 Batch Operations는 Amazon S3 PUT 객체 ACL 작업을 사용하여 매니페스트의 각 객체에 지정된 ACL을 적용합니다. 그러므로 기본 PUT 객체 ACL 작업에 적용되는 모든 규제 및 제한은 Amazon S3 Batch Operations PUT 객체 ACL 작업에도 적용됩니다. 자세한 내용은 이 페이지의 관련 리소스 (p. 518) 단원을 참조하십시오.

관련 리소스

- ACL을 사용한 액세스 관리 (p. 437)

- Amazon Simple Storage Service API Reference의 GET 객체 ACL

PUT 객체 태그 지정

PUT 객체 태그 지정 작업은 매니페스트에 나열된 각 객체의 Amazon S3 객체 태그를 바꿉니다. Amazon S3 객체 태그는 객체에 대한 메타데이터를 저장하는 데 사용할 수 있는 문자열의 키-값 쌍입니다.

PUT 객체 태그 지정 작업을 생성하려면 적용할 태그 세트를 제공합니다. Amazon S3 Batch Operations는 각 객체에 동일한 태그 세트를 적용합니다. 사용자가 제공한 태그 세트는 매니페스트에서 이미 객체에 연결된 모든 태그 세트를 대체합니다. Amazon S3 Batch Operations는 기존 태그는 그대로 두고 객체에 태그를 추가하는 것은 지원하지 않습니다.

매니페스트의 객체가 버전 지정된 버킷에 있는 경우 각 객체의 특정 버전에 태그 세트를 적용할 수 있습니다. 이렇게 하려면 매니페스트의 각 객체에 버전 ID를 지정합니다. 각 객체의 버전 ID를 포함하지 않을 경우 Amazon S3 Batch Operations가 각 객체의 최신 버전에 태그 세트를 적용합니다.

규제 및 제한

- PUT 객체 태그 지정 작업을 실행하도록 지정하는 역할에는 기본 Amazon S3 PUT 객체 태그 지정 작업을 수행할 권한이 있어야 합니다. 필요한 권한에 대한 자세한 내용은 Amazon Simple Storage Service API Reference의 PUT 객체 태그 지정을 참조하십시오.
- Amazon S3 Batch Operations는 Amazon S3 PUT 객체 태그 지정 작업을 사용하여 매니페스트 내 각 객체에 태그를 적용합니다. 그러므로 기본 PUT 객체 태그 지정 작업에 적용되는 모든 규제 및 제한은 Amazon S3 Batch Operations PUT 객체 태그 지정 작업에도 적용됩니다. 자세한 내용은 이 페이지의 관련 리소스 (p. 519) 단원을 참조하십시오.

관련 리소스

- 객체 태그 지정 (p. 114)
- Amazon Simple Storage Service API Reference의 GET 객체 태그 지정
- Amazon Simple Storage Service API Reference의 PUT 객체 태그 지정

배치 작업 건 관리

Amazon S3은 배치 작업 작업을 생성한 후 관리하는 데 도움이 되는 강력한 도구 세트를 제공합니다. 이 단원에서는 작업(job)을 관리하는 데 사용할 수 있는 작업(operation)에 대해 설명합니다. 이 단원에 나열된 모든 작업은 AWS Management 콘솔, AWS CLI, AWS SDK 또는 REST API를 사용하여 수행할 수 있습니다.

주제

- 작업 나열 (p. 520)
- 작업 세부 정보 보기 (p. 520)
- 태그를 사용하여 액세스 제어 및 작업 레이블 지정 (p. 520)
- 작업 우선 순위 지정 (p. 521)
- 작업 상태 (p. 521)
- 작업 실패 추적 (p. 523)
- 알림 및 로깅 (p. 523)
- 완료 보고서 (p. 524)

다음 동영상에서는 Amazon S3 콘솔을 사용하여 Amazon S3 Batch Operations 작업을 관리하는 방법에 대해 간략하게 설명합니다.

작업 나열

배치 작업 건의 목록을 검색할 수 있습니다. 목록에는 진행 중이거나 지난 90일 이내에 완료된 작업이 포함됩니다. 작업 목록에는 ID, 설명, 우선 순위, 현재 상태 및 성공 및 실패한 작업(task) 수 등 각 작업에 대한 정보가 포함됩니다. 상태를 기준으로 작업 목록을 필터링할 수 있습니다. 콘솔을 통해 작업 목록을 검색하면 설명 또는 ID 기준으로 작업을 검색하고 AWS 리전별로 필터링할 수도 있습니다.

작업 세부 정보 보기

작업을 나열하여 검색할 수 있는 것보다 더 많은 정보를 원할 경우 단일 작업에 대한 모든 세부 정보를 볼 수 있습니다. 작업 목록에서 반환되는 정보 이외에 단일 작업의 세부 정보는 다른 항목도 포함합니다. 이 정보에는 작업 파라미터, 매니페스트에 대한 세부 정보, 완료 보고서에 대한 정보(작업을 생성할 때 구성한 경우), 작업을 실행하기 위해 할당한 사용자 역할의 Amazon 리소스 이름(ARN)이 포함됩니다. 개별 작업의 세부 정보를 보면서 작업의 전체 구성에 액세스할 수 있습니다.

태그를 사용하여 액세스 제어 및 작업 레이블 지정

태그를 추가하여 Amazon S3 Batch Operations 작업에 대한 레이블을 지정하고 액세스를 제어할 수 있습니다. 태그는 배치 작업 작업 담당자를 식별하는 데 사용할 수 있습니다. 작업 태그가 있을 경우 사용자가 작업을 취소하고, 확인 상태의 작업을 활성화하거나 작업의 우선 순위 레벨을 바꿀 수 있는 능력을 부여하거나 제한할 수 있습니다. 태그가 연결된 작업을 생성할 수 있으며 작업을 생성한 후 작업에 태그를 추가할 수 있습니다. 각 태그는 작업을 생성하거나 나중에 업데이트할 때 포함할 수 있는 키-값 페어입니다.

> **Warning**
>
> 작업 태그는 기밀 정보 또는 개인 데이터를 포함해서는 안 됩니다.

다음 태그 지정 예를 참조하십시오. 재무 부서에서 배치 작업 작업을 생성하려고 한다고 가정합니다. `Finance` 값이 할당된 `Department` 태그로 작업을 생성하는 경우 사용자가 `CreateJob`을 호출할 수 있도록 허용하는 AWS Identity and Access Management(IAM) 정책을 작성할 수 있습니다. 또한 재무 부서의 구성원인 모든 사용자에게 해당 정책을 연결할 수 있습니다.

이 예에서는 사용자가 원하는 태그가 있는 작업의 우선 순위를 업데이트하거나 해당 태그가 있는 작업을 취소할 수 있도록 허용하는 정책을 생성할 수 있습니다. 자세한 내용은 the section called “작업 태그 예제” (p. 524) 단원을 참조하십시오.

새 Amazon S3 Batch Operations 작업을 생성할 때 태그를 추가하거나 기존 작업에 추가할 수 있습니다.

다음과 같은 태그 제한 사항이 있습니다.

- 고유한 태그 키가 있는 경우 작업에 최대 50개의 태그를 연결할 수 있습니다.
- 태그 키의 최대 길이는 128개 유니코드 문자이며, 태그 값의 최대 길이는 256개 유니코드 문자입니다.
- 키와 값은 대/소문자를 구분합니다.

태그 제한에 대한 자세한 내용은 AWS Billing and Cost Management 사용 설명서의 사용자 정의 태그 제한을 참조하십시오.

Amazon S3 Batch Operations 작업 태그 지정과 관련된 API 작업

Amazon S3은 Amazon S3 Batch Operations 작업 태그 지정을 위해 다음 API 작업을 지원합니다.

- GetJobTagging — 배치 작업 작업과 연관된 태그 세트를 반환합니다.
- PutJobTagging — 작업과 연관된 태그 세트를 대체합니다. 이 API 작업을 사용하는 Amazon S3 Batch Operations 작업 태그 관리에 대한 시나리오 두 가지가 있습니다.

- 작업에 태그가 없는 경우 — 작업에 태그 세트를 추가할 수 있습니다(작업에 이전 태그가 없음).
- 작업에 기존 태그 세트가 있는 경우 — 기존 태그 세트를 수정하려면 기존 태그 세트를 완전히 바꾸거나, 아니면 GetJobTagging을 사용하여 기존 태그 세트를 검색한 다음 기존 태그 세트 내에서 변경하고, 해당 태그 세트를 수정하고, 이 API 작업을 사용하여 수정한 태그 세트로 바꿉니다.

  Note

  빈 태그 세트로 이 요청을 전송하는 경우 S3 배치 작업는 객체에 있는 기존 태그 세트를 삭제합니다. 이 방법을 사용하면 계층 1 요청(PUT)에 대한 요금이 부과됩니다. 자세한 내용은 Amazon S3 요금을 참조하십시오.
  배치 작업 작업에 대한 기존 태그를 삭제하려면 비용이 발생하지 않고 동일한 결과를 얻을 수 있는 `DeleteJobTagging` 작업이 선호됩니다.
- DeleteJobTagging — 배치 작업 작업과 연관된 태그 세트를 삭제합니다.

작업 우선 순위 지정

각 작업에 숫자 우선 순위(임의의 양수)를 할당할 수 있습니다. Amazon S3 Batch Operations는 할당된 우선 순위에 따라 작업의 우선 순위를 결정합니다. 우선순위가 높은(즉 우선 순위 파라미터의 숫자 값이 높은) 작업이 먼저 평가됩니다. 우선 순위는 내림차순으로 결정됩니다. 예를 들어, 우선 순위 값이 1인 작업 대기열은 우선 순위 값이 10인 작업 대기열보다 먼저 일정이 예약됩니다.

작업이 실행되는 동안 작업의 우선 순위를 변경할 수 있습니다. 작업이 실행되는 동안 더 높은 우선 순위로 새 작업을 제출하면 우선 순위가 낮은 작업이 일시 중지되어 우선 순위가 높은 작업이 실행될 수 있습니다.

Note

Amazon S3 Batch Operations는 최대 효과에 기초하여 작업 우선 순위를 준수합니다. 우선 순위가 높은 작업이 일반적으로 우선 순위가 낮은 작업보다 우선하지만, Amazon S3는 엄격한 작업 순서를 보장하지 않습니다.

작업 상태

작업을 생성하면 작업은 일련의 상태를 거칩니다. 다음 표에서는 작업이 취할 수 있는 각 상태 및 상태 사이에서 가능한 전환에 대해 설명합니다.

상태	설명	Transitions
`New`	생성된 작업은 `New` 상태에서 시작합니다.	Amazon S3가 매니페스트 객체를 처리하기 시작하면 작업이 자동으로 `Preparing` 상태로 이동합니다.
`Preparing`	Amazon S3는 작업을 설정 및 실행하기 위한 매니페스트 객체 및 기타 작업 파라미터를 처리합니다.	Amazon S3가 매니페스트 및 기타 파라미터 처리를 완료하면 작업이 자동으로 `Ready` 상태로 이동합니다. 그러면 작업은 매니페스트에 나열된 객체에 대해 지정된 작업을 실행할 준비가 된 것입니다. 예를 들어 Amazon S3 콘솔에서 작업을 생성했을 때와 같이 작업을 실행하기 전에 확인이 필요한 경우 작업이 `Preparing` 상태에서 `Suspended` 상태로 전환합니다. 작업은 사용자가 실행을 확인

상태	설명	Transitions
		할 때까지 `Suspended` 상태로 유지됩니다.
`Suspended`	작업이 확인을 필요로 하지만 사용자가 아직 실행을 확인하지 않았습니다. Amazon S3 콘솔을 사용하여 생성하는 작업만 확인이 필요합니다. 콘솔을 사용해 생성한 작업은 `Preparing` 상태에서 바로 `Suspended` 상태로 들어갑니다. 사용자가 작업 실행을 확인하여 `Ready` 상태가 된 이후에는 작업이 절대로 `Suspended` 상태로 되돌아가지 않습니다.	작업 실행을 확인하면 작업 상태가 `Ready`로 변경됩니다.
`Ready`	Amazon S3가 요청된 객체 작업을 시작할 준비가 되었습니다.	Amazon S3가 작업을 실행하기 시작하면 작업이 작동으로 `Active` 상태로 이동합니다. 작업이 `Ready` 상태를 유지하는 시간은 이미 우선 순위가 높은 작업이 실행 중인지 여부와 이들 작업을 완료하는 데 얼마나 걸리는지에 따라 달라집니다.
`Active`	Amazon S3가 매니페스트에 나열된 객체에 대해 요청된 작업을 실행합니다. 작업이 `Active` 상태일 때 Amazon S3 콘솔을 사용하거나 REST API, AWS CLI 또는 AWS SDK를 통해 `DescribeJob` 작업을 사용하여 진행률을 모니터링할 수 있습니다.	작업이 더 이상 객체에 대한 작업을 실행하지 않으면 `Active` 상태에서 다른 상태로 이동합니다. 이는 작업이 성공적으로 완료되거나 실패한 경우와 같이 자동으로 발생할 수 있습니다. 또는 작업 취소와 같이 사용자 작업의 결과로 발생할 수 있습니다. 이후 작업이 이동하는 상태는 전환 이유에 따라 다릅니다.
`Pausing`	작업이 다른 상태에서 `Paused` 상태로 전환합니다.	`Pausing` 단계가 완료되면 작업이 자동으로 `Paused` 상태로 이동합니다.
`Paused`	작업이 실행되는 동안 더 높은 우선 순위로 새 작업을 제출하면 현재 작업이 `Paused` 상태가 될 수 있습니다.	작업 실행을 차단하는 더 높은 우선 순위의 작업이 완료, 실패 또는 일시 중지되면 `Paused` 작업이 자동으로 `Active` 상태로 되돌아갑니다.
`Complete`	작업이 매니페스트의 모든 객체에 대해 요청된 작업을 완료했습니다. 각 객체에 대해 작업(operation)이 성공했거나 실패했을 수 있습니다. 작업이 완료 보고서를 생성하도록 구성한 경우 작업이 `Complete` 상태가 되는 즉시 보고서가 생성됩니다.	`Complete`는 최종 상태입니다. 작업이 `Complete` 상태에 도달하면 다른 어떤 상태로도 전환하지 않습니다.

상태	설명	Transitions
`Cancelling`	작업이 `Cancelled` 상태로 전환 중입니다.	`Cancelling` 단계가 완료되면 작업이 자동으로 `Cancelled` 상태로 이동합니다.
`Cancelled`	사용자가 작업 취소를 요청했고 Amazon S3 Batch Operations가 성공적으로 작업을 취소했습니다. 작업이 Amazon S3에 새로운 요청을 제출하지 않습니다.	`Cancelled`는 최종 상태입니다. 작업이 `Cancelled` 상태에 도달하면 다른 어떤 상태로도 전환하지 않습니다.
`Failing`	작업이 `Failed` 상태로 전환 중입니다.	`Failing` 단계가 완료되면 작업이 자동으로 `Failed` 상태로 이동합니다.
`Failed`	작업이 실패했고 더 이상 실행되지 않습니다. 작업 실패에 대한 자세한 내용은 작업 실패 추적 (p. 523) 단원을 참조하십시오.	`Failed`는 최종 상태입니다. 작업이 `Failed` 상태에 도달하면 다른 어떤 상태로도 전환하지 않습니다.

작업 실패 추적

Amazon S3 Batch Operations 작업이 지정된 매니페스트를 읽을 수 없는 경우처럼 성공적으로 실행할 수 없는 문제가 발생하면 해당 작업이 실패합니다. 작업이 실패하면 하나 이상의 실패 코드 또는 실패 이유를 생성합니다. Amazon S3 Batch Operations는 작업의 세부 정보를 요청하여 실패 코드 및 이유를 볼 수 있도록 작업과 함께 실패 코드 및 이유를 저장합니다. 해당 작업에 대한 완료 보고서를 요청한 경우 실패 코드와 이유도 여기에 나타납니다.

작업이 실패한 작업(operation)을 대량으로 실행하는 것을 방지하기 위해 Amazon S3가 모든 배치 작업 작업에 작업(task) 실패 임계값을 부여합니다. 하나의 작업이 최소 1,000개의 작업(task)을 실행하면 Amazon S3가 작업(task) 실패율을 모니터링합니다. 언제든 실패율(실행된 총 작업 수 대비 실패한 작업 수의 비율)이 50%를 초과하면 작업이 실패합니다. 작업(task) 실패율 임계값이 초과되어 작업이 실패할 경우 실패의 원인을 식별할 수 있습니다. 예를 들어 지정된 버킷에 없는 일부 객체를 실수로 매니페스트에 포함했을 수 있습니다. 오류를 수정한 후에 작업을 다시 제출할 수 있습니다.

> **Note**
>
> Amazon S3 Batch Operations는 비동기식으로 작동하며, 반드시 매니페스트에 객체가 나열된 순서대로 작업(task)을 실행하지는 않습니다. 그러므로 매니페스트 순서를 사용하여 어떤 객체의 작업(task)이 성공했거나 실패했는지 판단할 수 없습니다. 대신, 작업 완료 보고서(요청한 경우)를 검토하거나 AWS CloudTrail 이벤트 로그를 검토하여 실패의 원인을 확인할 수 있습니다.

알림 및 로깅

완료 보고서를 요청할 뿐만 아니라, AWS CloudTrail를 사용하여 배치 작업 활동을 캡처, 검토 및 감사할 수도 있습니다. 배치 작업은 기존 Amazon S3 API를 사용하여 작업(task)을 수행하기 때문에 이러한 작업 역시 직접 호출한 경우와 동일한 이벤트를 방출합니다. 따라서 이미 Amazon S3에서 사용하는 것과 동일한 알림, 로깅 및 감사 도구와 프로세스를 사용하여 작업(job) 진행률 및 모든 작업(task)을 추적하고 기록할 수 있습니다.

Amazon S3 이벤트에 대한 자세한 내용은 Amazon S3 이벤트 알림 구성 (p. 587) 단원을 참조하십시오.

완료 보고서

작업을 생성할 때 완료 보고서를 요청할 수 있습니다. Amazon S3 Batch Operations가 하나 이상의 작업을 성공적으로 호출하는 한 Amazon S3이 작업 실행이 끝나거나 실패하거나 취소된 후 완료 보고서를 생성합니다. 모든 작업 또는 실패한 작업만 포함하도록 완료 보고서를 구성할 수 있습니다.

완료 보고서에는 작업(job) 구성 및 상태, 그리고 객체 키와 버전, 상태, 오류 코드 및 오류 설명 등 각 작업(task)의 정보가 포함됩니다. 완료 보고서를 구성하지 않은 경우에도 여전히 CloudTrail 및 Amazon CloudWatch를 사용하여 작업(job) 및 그 작업(task)을 모니터링하고 감사할 수 있습니다. 하지만 완료 보고서를 사용하면 추가 설정을 하지 않아도 통합된 형식으로 작업 결과를 손쉽게 볼 수 있습니다. 완료 보고서의 예는 예제: Amazon S3 Batch Operations 완료 보고서 요청 (p. 529) 단원을 참조하십시오.

Amazon S3 배치 작업 예제

Amazon S3 Batch Operations를 사용하여 엑사바이트 급의 데이터가 포함된 수십억 개의 S3 객체에 대해 대규모 배치 작업을 수행할 수 있습니다. AWS Management 콘솔, AWS Command Line Interface(AWS CLI), AWS SDK 또는 REST API를 사용하여 배치 작업 작업을 관리할 수 있습니다.

이 단원에서는 Amazon S3에서 배치 작업 작업을 생성하고 관리하는 다음 예제를 설명합니다.

주제

- 예제: 작업 태그를 사용하여 Amazon S3 Batch Operations에 대한 권한 제어 (p. 524)
- 예제: Amazon S3 Batch Operations 완료 보고서 요청 (p. 529)
- 예제: Amazon S3 배치 작업을 사용하여 AWS 계정 간에 객체 복사 (p. 531)
- AWS CLI를 사용하는 Amazon S3 Batch Operations 예제 (p. 536)
- 예: AWS SDK for Java를 사용하여 Amazon S3 Batch Operations 작업 생성 및 관리 (p. 541)

예제: 작업 태그를 사용하여 Amazon S3 Batch Operations에 대한 권한 제어

Amazon S3 Batch Operations 작업 관리를 돕기 위해 작업 태그를 추가할 수 있습니다. 작업 태그를 사용하여 배치 작업 작업에 대한 액세스를 제어하고 작업이 생성될 때 태그를 적용할 수 있습니다.

각 배치 작업 작업에 최대 50개의 작업 태그를 적용할 수 있습니다. 이렇게 하면 작업을 편집할 수 있는 사용자 집합을 제한하는 매우 세부적인 정책을 설정할 수 있습니다. 작업 태그가 있을 경우 사용자가 작업을 취소하고, 확인 상태의 작업을 활성화하거나 작업의 우선 순위 레벨을 변경할 수 있는 능력을 부여하거나 제한할 수 있습니다. 또한 모든 새 작업에 태그를 적용하고 태그에 대해 허용되는 키-값 페어를 지정할 수 있습니다. 동일한 IAM 정책 언어를 사용하여 이러한 모든 조건을 표현할 수 있습니다. 자세한 내용은 Amazon S3에 사용되는 작업, 리소스 및 조건 키 (p. 356) 단원을 참조하십시오.

다음 예제에서는 Amazon S3 Batch Operations 작업 태그를 사용하여 특정 department(예: Finance 또는 Compliance 부서)에서 실행되는 작업만 생성 및 편집할 수 있는 권한을 사용자에게 부여하는 방법을 보여줍니다. QA 또는 Production 등 관련된 development의 단계에 따라 작업을 할당할 수도 있습니다.

이 예제에서는 AWS Identity and Access Management(IAM) 정책의 Amazon S3 Batch Operations 작업 태그를 사용하여 사용자에게 부서 내에서 실행 중인 작업만 생성하고 편집할 수 있는 권한을 부여합니다. QA 또는 Production 등 관련된 개발의 단계에 따라 작업을 할당합니다.

이 예제에서는 다음과 같은 부서를 사용하며 각 부서는 서로 다른 방식으로 배치 작업를 사용합니다.

- 재무
- 규정 준수

- 비즈니스 인텔리전스
- 엔지니어링

사용자 및 리소스에 태그를 할당하여 액세스 제어

이 시나리오에서는 관리자가 ABAC(속성 기반 액세스 제어)를 사용하고 있습니다. ABAC는 IAM 사용자와 AWS 리소스 모두에 태그를 연결하여 권한을 정의하는 IAM 권한 부여 전략입니다.

사용자 및 작업에는 다음 부서 태그 중 하나가 할당됩니다.

키 : 값

- `department : Finance`
- `department : Compliance`
- `department : BusinessIntelligence`
- `department : Engineering`

> **Note**
>
> 작업 태그 키와 값은 대/소문자를 구분합니다.

ABAC 액세스 제어 전략을 사용하면 재무 부서의 사용자에게 `department=Finance` 태그를 IAM 사용자와 연결하여 부서 내에서 Amazon S3 Batch Operations 작업을 생성하고 관리할 수 있는 권한을 부여합니다.

또한 회사의 모든 사용자가 해당 부서 내에서 Amazon S3 Batch Operations 작업을 생성하거나 수정할 수 있도록 허용하는 관리형 정책을 IAM 사용자에게 연결할 수 있습니다.

이 예제의 정책에는 다음과 같은 세 가지 정책 문이 포함되어 있습니다.

- 정책의 첫 번째 문은 작업 생성 요청에 해당 부서와 일치하는 작업 태그가 포함되어 있는 경우 사용자가 배치 작업 작업을 생성할 수 있도록 허용합니다. 이는 정책 평가 시 IAM 사용자의 부서 태그로 대체되는 `"${aws:PrincipalTag/department}"` 구문을 사용하여 표현됩니다. (`"aws:RequestTag/department"`) 요청에서 부서 태그에 대해 제공된 값이 사용자의 부서와 일치하면 조건이 충족됩니다.
- 정책의 두 번째 문은 사용자가 업데이트 중인 작업이 사용자의 부서와 일치하는 경우 사용자가 작업의 우선 순위를 변경하거나 작업 상태를 업데이트할 수 있도록 허용합니다.
- 세 번째 문은 (1) 부서 태그가 보존되고 (2) 업데이트하려는 작업이 해당 부서 내에 있는 한 사용자가 `PutJobTagging` 요청을 통해 언제든지 배치 작업 작업 태그를 업데이트할 수 있도록 허용합니다.

```
{
      "Version": "2012-10-17",
      "Statement": [
            {
                  "Effect": "Allow",
                  "Action": "s3:CreateJob",
                  "Resource": "*",
                  "Condition": {
                        "StringEquals": {
                              "aws:RequestTag/department": "${aws:PrincipalTag/department}"

                }
            }
        },
            {
                  "Effect": "Allow",
                  "Action": [
                        "s3:UpdateJobPriority",
```

```
                        "s3:UpdateJobStatus"
            ],
                  "Resource": "*",
                  "Condition": {
                        "StringEquals": {
                              "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                  }
            }
        },
            {
                  "Effect": "Allow",
                  "Action": "s3:PutJobTagging",
                  "Resource": "*",
                  "Condition": {
                        "StringEquals": {
                              "aws:RequestTag/department": "${aws:PrincipalTag/
department}",
                              "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                  }
            }
        }
    ]
}
```

단계별 배치 작업 작업 태그 지정 및 작업 우선 순위 제한 적용

모든 Amazon S3 Batch Operations 작업에는 숫자 우선 순위가 있으며, Amazon S3는 작업을 실행할 순서를 결정하는 데 사용합니다. 이 예제에서는 다음과 같이 대부분의 사용자가 작업에 할당할 수 있는 최대 우선 순위를 제한하고, 제한된 권한 사용자 집합에 대해 높은 우선 순위 범위를 예약합니다.

- QA 스테이지 우선 순위 범위(낮음): 1-100
- 프로덕션 단계 우선 순위 범위(높음): 1-300

이를 위해 작업의 단계를 나타내는 새 태그 세트를 도입합니다.

키 : 값

- `stage : QA`
- `stage : Production`

부서 내에서 우선 순위가 낮은 작업 생성 및 업데이트

이 정책은 부서 기반 제한 외에 Amazon S3 Batch Operations 작업 생성 및 업데이트에 대한 두 가지 새로운 제한을 도입합니다.

- 사용자는 해당 부서에서 작업에 `stage=QA` 태그를 포함해야 하는 새 조건으로 작업을 생성하거나 업데이트할 수 있습니다.
- 사용자는 작업의 우선 순위를 새로운 최대값인 100개까지 생성하거나 업데이트할 수 있습니다.

```
{
        "Version": "2012-10-17",
        "Statement": [
        {
        "Effect": "Allow",
        "Action": "s3:CreateJob",
```

```
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/department": "${aws:PrincipalTag/department}",
                "aws:RequestTag/stage": "QA"
            },
            "NumericLessThanEquals": {
                "s3:RequestJobPriority": 100
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:UpdateJobStatus"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/department": "${aws:PrincipalTag/department}"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": "s3:UpdateJobPriority",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/department": "${aws:PrincipalTag/department}",
                "aws:ResourceTag/stage": "QA"
            },
            "NumericLessThanEquals": {
                "s3:RequestJobPriority": 100
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": "s3:PutJobTagging",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/department" : "${aws:PrincipalTag/department}",
                "aws:ResourceTag/department": "${aws:PrincipalTag/department}",
                "aws:RequestTag/stage": "QA",
                "aws:ResourceTag/stage": "QA"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": "s3:GetJobTagging",
        "Resource": "*"
    }
    ]
}
```

부서 내에서 우선 순위가 높은 작업 생성 및 업데이트

일부 사용자는 QA 또는 Production 중 하나에서 우선 순위가 높은 작업을 생성할 수 있어야 합니다. 이 요구를 지원하기 위해 이전 단원의 우선 순위가 낮은 정책에 맞게 조정된 관리형 정책을 생성합니다.

이 정책은 다음을 수행합니다.

- 사용자가 `stage=QA` 태그 또는 `stage=Production` 태그를 사용하여 부서에서 작업을 생성하거나 업데이트할 수 있습니다.
- 사용자는 작업의 우선 순위를 최대 300개까지 생성하거나 업데이트할 수 있습니다.

```
{
      "Version": "2012-10-17",
      "Statement": [
          {
                "Effect": "Allow",
                "Action": "s3:CreateJob",
                "Resource": "*",
                "Condition": {
                      "ForAnyValue:StringEquals": {
                            "aws:RequestTag/stage": [
                                  "QA",
                                  "Production"
                    ]
                },
                      "StringEquals": {
                            "aws:RequestTag/department": "${aws:PrincipalTag/department}"

                },
                      "NumericLessThanEquals": {
                            "s3:RequestJobPriority": 300
                }
            }
        },
          {
                "Effect": "Allow",
                "Action": [
                      "s3:UpdateJobStatus"
            ],
                "Resource": "*",
                "Condition": {
                      "StringEquals": {
                            "aws:ResourceTag/department": "${aws:PrincipalTag/department}"

                }
            }
        },
          {
                "Effect": "Allow",
                "Action": "s3:UpdateJobPriority",
                "Resource": "*",
                "Condition": {
                      "ForAnyValue:StringEquals": {
                            "aws:ResourceTag/stage": [
                                  "QA",
                                  "Production"
                    ]
                },
                      "StringEquals": {
                            "aws:ResourceTag/department": "${aws:PrincipalTag/department}"

                },
                      "NumericLessThanEquals": {
                            "s3:RequestJobPriority": 300
                }
            }
        },
          {
                "Effect": "Allow",
                "Action": "s3:PutJobTagging",
```

```
                "Resource": "*",
                "Condition": {
                      "StringEquals": {
                            "aws:RequestTag/department": "${aws:PrincipalTag/department}",
                            "aws:ResourceTag/department": "${aws:PrincipalTag/department}"

                },
                      "ForAnyValue:StringEquals": {
                            "aws:RequestTag/stage": [
                                  "QA",
                                  "Production"
                    ],
                            "aws:ResourceTag/stage": [
                                  "QA",
                                  "Production"
                    ]
                }
            }
        }
    ]
}
```

예제: Amazon S3 Batch Operations 완료 보고서 요청

Amazon S3 Batch Operations 작업을 생성할 때 모든 작업 또는 실패한 작업에 대한 완료 보고서를 요청할 수 있습니다. 하나 이상의 작업이 성공적으로 호출된 경우 Amazon S3 Batch Operations는 완료, 실패 또는 취소된 작업에 대한 보고서를 생성합니다.

완료 보고서에는 객체 키 이름 및 버전, 상태, 오류 코드 및 오류 설명을 포함한 각 작업에 대한 추가 정보가 수록됩니다. 실패한 각 작업의 오류 설명은 작업 생성 중 발생한 문제(예: 권한)를 진단하는 데 사용할 수 있습니다.

Example — 상위 매니페스트 결과 파일

상위 `manifest.json` 파일에는 다음 예제와 같이 성공한 각 보고서의 위치와 (작업이 실패한 경우) 실패한 보고서의 위치가 포함됩니다.

```
{
    "Format": "Report_CSV_20180820",
    "ReportCreationDate": "2019-04-05T17:48:39.725Z",
    "Results": [
        {
            "TaskExecutionStatus": "succeeded",
            "Bucket": "my-job-reports",
            "MD5Checksum": "83b1c4cbe93fc893f54053697e10fd6e",
            "Key": "job-f8fb9d89-a3aa-461d-bddc-ea6a1b131955/
results/6217b0fab0de85c408b4be96aeaca9b195a7daa5.csv"
        },
        {
            "TaskExecutionStatus": "failed",
            "Bucket": "my-job-reports",
            "MD5Checksum": "22ee037f3515975f7719699e5c416eaa",
            "Key": "job-f8fb9d89-a3aa-461d-bddc-ea6a1b131955/results/
b2ddad417e94331e9f37b44f1faf8c7ed5873f2e.csv"
        }
    ],
    "ReportSchema": "Bucket, Key, VersionId, TaskStatus, ErrorCode, HTTPStatusCode,
 ResultMessage"
}
```

Example — 실패한 작업 보고서

실패한 작업 보고서에는 실패한 모든 작업에 대한 다음 정보가 포함됩니다.

- `Bucket`
- `Key`
- `VersionId`
- `TaskStatus`
- `ErrorCode`
- `HTTPStatusCode`
- `ResultMessage`

다음 예제 보고서는 AWS Lambda 함수가 시간을 초과하여 실패 임계값을 초과한 경우를 보여줍니다. 그런 다음 `PermanentFailure`로 표기되었습니다.

```
awsexamplebucket,image_14975,,failed,200,PermanentFailure,"Lambda returned function error:
 {""errorMessage"":""2019-04-05T17:35:21.155Z 2845ca0d-38d9-4c4b-abcf-379dc749c452 Task
 timed out after 3.00 seconds""}"
awsexamplebucket,image_15897,,failed,200,PermanentFailure,"Lambda returned function error:
 {""errorMessage"":""2019-04-05T17:35:29.610Z 2d0a330b-de9b-425f-b511-29232fde5fe4 Task
 timed out after 3.00 seconds""}"
awsexamplebucket,image_14819,,failed,200,PermanentFailure,"Lambda returned function error:
 {""errorMessage"":""2019-04-05T17:35:22.362Z fcf5efde-74d4-4e6d-b37a-c7f18827f551 Task
 timed out after 3.00 seconds""}"
awsexamplebucket,image_15930,,failed,200,PermanentFailure,"Lambda returned function error:
 {""errorMessage"":""2019-04-05T17:35:29.809Z 3dd5b57c-4a4a-48aa-8a35-cbf027b7957e Task
 timed out after 3.00 seconds""}"
awsexamplebucket,image_17644,,failed,200,PermanentFailure,"Lambda returned function error:
 {""errorMessage"":""2019-04-05T17:35:46.025Z 10a764e4-2b26-4d8c-9056-1e1072b4723f Task
 timed out after 3.00 seconds""}"
awsexamplebucket,image_17398,,failed,200,PermanentFailure,"Lambda returned function error:
 {""errorMessage"":""2019-04-05T17:35:44.661Z 1e306352-4c54-4eba-aee8-4d02f8c0235c Task
 timed out after 3.00 seconds""}"
```

Example — 성공한 작업 보고서

성공한 작업 보고서에는 완료된 작업에 대한 다음 정보가 포함됩니다.

- `Bucket`
- `Key`
- `VersionId`
- `TaskStatus`
- `ErrorCode`
- `HTTPStatusCode`
- `ResultMessage`

다음 예제에서 Lambda 함수는 Amazon S3 객체를 다른 버킷에 성공적으로 복사했습니다. 반환된 Amazon S3 응답은 Amazon S3 Batch Operations로 전달된 다음, 최종 완료 보고서에 기록됩니다.

```
awsexamplebucket,image_17775,,succeeded,200,,"{u'CopySourceVersionId':
 'xVR78haVKlRnurYofbTfYr3ufYbktF8h', u'CopyObjectResult': {u'LastModified':
 datetime.datetime(2019, 4, 5, 17, 35, 39, tzinfo=tzlocal()), u'ETag':
 '""fe66f4390c50f29798f040d7aae72784""'}, 'ResponseMetadata': {'HTTPStatusCode':
 200, 'RetryAttempts': 0, 'HostId': 'nXNaClIMxEJzWNmeMNQV2KpjbaCJLn0OGoXWZpuVOFS/
```

```
iQYWxb3QtTvzX9SVfx2lA3oTKLwImKw=', 'RequestId': '3ED5852152014362', 'HTTPHeaders':
 {'content-length': '234', 'x-amz-id-2': 'nXNaClIMxEJzWNmeMNQV2KpjbaCJLn0OGoXWZpuVOFS/
iQYWxb3QtTvzX9SVfx2lA3oTKLwImKw=', 'x-amz-copy-source-version-id':
 'xVR78haVKlRnurYofbTfYr3ufYbktF8h', 'server': 'AmazonS3', 'x-amz-request-id':
 '3ED5852152014362', 'date': 'Fri, 05 Apr 2019 17:35:39 GMT', 'content-type': 'application/
xml'}}}"
awsexamplebucket,image_17763,,succeeded,200,,"{u'CopySourceVersionId':
 '6HjOUSim4Wj6BTcbxToXW44pSZ.40pwq', u'CopyObjectResult': {u'LastModified':
 datetime.datetime(2019, 4, 5, 17, 35, 39, tzinfo=tzlocal()),
 u'ETag': '""fe66f4390c50f29798f040d7aae72784""'}, 'ResponseMetadata':
 {'HTTPStatusCode': 200, 'RetryAttempts': 0, 'HostId': 'GiCZNYr8LHd/
Thyk6beTRP96IGZk2sYxujLe13TuuLpq6U2RD3we0YoluuIdm1PRvkMwnEW1aFc=', 'RequestId':
 '1BC9F5B1B95D7000', 'HTTPHeaders': {'content-length': '234', 'x-amz-id-2': 'GiCZNYr8LHd/
Thyk6beTRP96IGZk2sYxujLe13TuuLpq6U2RD3we0YoluuIdm1PRvkMwnEW1aFc=', 'x-amz-copy-source-
version-id': '6HjOUSim4Wj6BTcbxToXW44pSZ.40pwq', 'server': 'AmazonS3', 'x-amz-request-id':
 '1BC9F5B1B95D7000', 'date': 'Fri, 05 Apr 2019 17:35:39 GMT', 'content-type': 'application/
xml'}}}"
awsexamplebucket,image_17860,,succeeded,200,,"{u'CopySourceVersionId':
 'm.MDD0g_QsUnYZ8TBzVFrp.TmjN8PJyX', u'CopyObjectResult': {u'LastModified':
 datetime.datetime(2019, 4, 5, 17, 35, 40, tzinfo=tzlocal()), u'ETag':
 '""fe66f4390c50f29798f040d7aae72784""'}, 'ResponseMetadata': {'HTTPStatusCode':
 200, 'RetryAttempts': 0, 'HostId': 'F9ooZOgpE5g9sNgBZxjdiPHqB4+0DNWgj3qbsir
+sKai4fv7rQEcF2fBN1VeeFc2WH45a9ygb2g=', 'RequestId': '8D9CA56A56813DF3', 'HTTPHeaders':
 {'content-length': '234', 'x-amz-id-2': 'F9ooZOgpE5g9sNgBZxjdiPHqB4+0DNWgj3qbsir
+sKai4fv7rQEcF2fBN1VeeFc2WH45a9ygb2g=', 'x-amz-copy-source-version-id':
 'm.MDD0g_QsUnYZ8TBzVFrp.TmjN8PJyX', 'server': 'AmazonS3', 'x-amz-request-id':
 '8D9CA56A56813DF3', 'date': 'Fri, 05 Apr 2019 17:35:40 GMT', 'content-type': 'application/
xml'}}}"
```

예제: Amazon S3 배치 작업을 사용하여 AWS 계정 간에 객체 복사

Amazon S3 Batch Operations를 사용하여 객체를 다른 AWS 계정(대상 계정)으로 복사하는 PUT 복사 작업을 만들 수 있습니다. 이 작업 수행 시 Amazon S3 인벤토리를 사용하여 작업 생성 중에 사용할 수 있도록 대상 계정에 인벤토리 보고서를 전달할 수 있습니다. 또는 원본이나 대상 계정에서 CSV(쉼표로 구분된 값) 매니페스트를 사용할 수 있습니다.

다음 단원에서는 다른 AWS 계정에 있는 매니페스트를 저장하고 사용하는 방법에 대해 설명합니다.

주제

- 대상 계정에 전달된 인벤토리 보고서 사용 (p. 531)
- 원본 계정에 저장된 CSV 매니페스트 사용 (p. 534)

대상 계정에 전달된 인벤토리 보고서 사용

Amazon S3 인벤토리는 버킷에 객체의 인벤토리를 생성합니다. 결과 목록은 출력 파일에 게시됩니다. 인벤토리에 추가된 버킷을 원본 버킷이라고 하고 인벤토리 보고서 파일이 저장되는 버킷을 대상 버킷이라고 합니다.

Amazon S3 인벤토리 보고서를 다른 AWS 계정으로 전달하도록 인벤토리 보고서를 구성할 수 있습니다. 그러면 대상 AWS 계정에서 작업이 생성될 때 Amazon S3 Batch Operations에서 인벤토리 보고서를 읽을 수 있습니다.

Amazon S3 인벤토리 원본 및 대상 버킷에 대한 자세한 내용은 Amazon S3 인벤토리 설정은 어떻게 합니까? (p. 460) 단원을 참조하십시오.

인벤토리를 설정하는 가장 쉬운 방법은 AWS Management 콘솔을 사용하는 것이지만 REST API, AWS Command Line Interface(AWS CLI) 또는 AWS SDK를 사용할 수도 있습니다.

다음 콘솔 절차에는 Amazon S3 Batch Operations 작업에 대한 권한을 설정하기 위한 상위 단계가 포함되어 있습니다. 이 절차에서는 원본 계정에서 대상 계정으로 객체를 복사하고, 대상 AWS 계정에 인벤토리 보고서를 저장합니다.

다른 계정이 소유한 원본 버킷과 대상 버킷에 Amazon S3 인벤토리를 설정하려면

1. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 인벤토리 보고서를 저장할 대상 버킷을 선택합니다.

   인벤토리 보고서를 저장할 대상 매니페스트 버킷을 결정합니다. 이 절차에서 대상 계정은 대상 매니페스트 버킷과 객체가 복사되는 버킷을 모두 소유한 계정입니다.
3. 원본 버킷에 있는 객체를 열거하여 대상 매니페스트 버킷에 목록을 게시하도록 인벤토리를 구성합니다.

   원본 버킷에 대한 인벤토리 목록을 구성합니다. 이 작업을 수행할 때 목록을 저장할 대상 버킷을 지정합니다. 원본 버킷에 대한 인벤토리 보고서가 대상 버킷에 게시됩니다. 이 절차에서 원본 계정은 원본 버킷을 소유한 계정입니다.

   콘솔을 사용하여 인벤토리를 구성하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 Amazon S3인벤토리를 구성하려면? 단원을 참조하십시오.

   출력 형식으로 CSV를 선택하십시오.

   대상 버킷에 대한 정보를 입력할 때 다른 계정의 버킷을 선택하십시오. 그런 다음 대상 매니페스트 버킷의 이름을 입력합니다. 선택적으로 대상 계정의 계정 ID를 입력할 수 있습니다.

   인벤토리 구성을 저장하면 콘솔에 다음과 유사한 메시지가 표시됩니다.

   Amazon S3는 대상 버킷에 버킷 정책을 만들 수 없습니다. Amazon S3가 대상 버킷에 데이터를 저장할 수 있도록 다음 버킷 정책을 추가하려면 대상 버킷 소유자에게 요청하십시오.

   그러면 콘솔에 대상 버킷에 사용할 수 있는 버킷 정책이 표시됩니다.
4. 콘솔에 표시되는 대상 버킷 정책을 복사하십시오.
5. 대상 계정에서 인벤토리 보고서가 저장된 대상 매니페스트 버킷에 복사한 버킷 정책을 추가합니다.
6. Amazon S3 Batch Operations 신뢰 정책을 기반으로 하는 대상 계정에서 역할을 생성합니다. 신뢰 정책에 대한 자세한 내용은 신뢰 정책 (p. 506) 단원을 참조하십시오.

   역할 생성에 대한 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하십시오.

   역할의 이름을 입력하십시오(예제 역할은 이름 `BatchOperationsDestinationRoleCOPY` 사용). S3 서비스를 선택한 다음, 신뢰 정책을 역할에 적용하는 S3 bucket Batch Operations(S3 버킷 배치 작업) 사용 사례를 선택하십시오.

   그리고 정책 생성을 선택하여 다음 정책을 역할에 연결하십시오.

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "AllowBatchOperationsDestinationObjectCOPY",
                "Effect": "Allow",
                "Action": [
                    "s3:PutObject",
                    "s3:PutObjectVersionAcl",
                    "s3:PutObjectAcl",
                    "s3:PutObjectVersionTagging",
                    "s3:PutObjectTagging",
                    "s3:GetObject",
```

```
                    "s3:GetObjectVersion",
                    "s3:GetObjectAcl",
                    "s3:GetObjectTagging",
                    "s3:GetObjectVersionAcl",
                    "s3:GetObjectVersionTagging"
                ],
                "Resource": [
                    "arn:aws:s3:::ObjectDestinationBucket/*",
                    "arn:aws:s3:::ObjectSourceBucket/*",
                    "arn:aws:s3:::ObjectDestinationManifestBucket/*"
                ]
            }
        ]
}
```

이 역할은 정책을 사용하여 대상 버킷의 매니페스트를 읽을 수 있는 `batchoperations.s3.amazonaws.com` 권한을 부여합니다. 또한 원본 객체 버킷의 GET 객체, 액세스 제어 목록(ACL), 태그 및 버전에 대한 권한을 부여합니다. 또한 대상 객체 버킷의 PUT 객체 , ACL, 태그 및 버전에 대한 권한을 부여합니다.

7. 원본 계정에서, 원본 버킷의 GET 객체, ACL, 태그 및 버전에 이전 단계에서 생성한 역할을 제공하는 원본 버킷에 대한 버킷 정책을 생성합니다. 이 단계에서는 신뢰할 수 있는 역할을 통해 Amazon S3 Batch Operations가 원본 버킷에서 객체를 가져올 수 있습니다.

   다음은 원본 계정에 대한 버킷 정책의 예입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowBatchOperationsSourceObjectCOPY",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::DestinationAccountNumber:role/
BatchOperationsDestinationRoleCOPY"
            },
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:GetObjectAcl",
                "s3:GetObjectTagging",
                "s3:GetObjectVersionAcl",
                "s3:GetObjectVersionTagging"
            ],
            "Resource": "arn:aws:s3:::ObjectSourceBucket/*"
        }
    ]
}
```

8. 인벤토리 보고서를 사용할 수 있게 되면 대상 계정에 Amazon S3 Batch Operations PUT 객체 복사 작업을 생성하고 대상 매니페스트 버킷에서 인벤토리 보고서를 선택합니다. 대상 계정에서 생성한 역할에 대한 ARN이 필요합니다.

   작업 생성에 대한 일반적인 정보는 Amazon S3 배치 작업 건 생성 (p. 504) 단원을 참조하십시오.

   콘솔을 사용하여 작업을 생성하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 Amazon S3의 배치 작업 건 생성을 참조하십시오.

원본 계정에 저장된 CSV 매니페스트 사용

Amazon S3 Batch Operations 작업의 매니페스트와 다른 AWS 계정에 저장된 CSV 파일을 사용할 수 있습니다.

다음 절차에서는 Amazon S3 Batch Operations 작업을 사용하여 원본 계정의 객체를 원본 계정에 저장된 CSV 매니페스트 파일이 있는 대상 계정으로 복사할 때 권한을 설정하는 방법을 보여줍니다.

다른 AWS 계정에 저장된 CSV 매니페스트를 설정하려면

1. Amazon S3 Batch Operations 신뢰 정책을 기반으로 하는 대상 계정에 역할을 생성합니다. 이 절차에서 대상 계정은 객체가 복사되는 계정입니다.

   신뢰 정책에 대한 자세한 내용은 신뢰 정책 (p. 506) 단원을 참조하십시오.

   역할 생성에 대한 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하십시오.

   콘솔을 사용하여 역할을 생성하는 경우, 역할의 이름을 입력하십시오(예제 역할은 이름 `BatchOperationsDestinationRoleCOPY` 사용). S3 서비스를 선택한 다음, 신뢰 정책을 역할에 적용하는 S3 bucket Batch Operations(S3 버킷 배치 작업) 사용 사례를 선택하십시오.

   그리고 정책 생성을 선택하여 다음 정책을 역할에 연결하십시오.

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "AllowBatchOperationsDestinationObjectCOPY",
                "Effect": "Allow",
                "Action": [
                    "s3:PutObject",
                    "s3:PutObjectVersionAcl",
                    "s3:PutObjectAcl",
                    "s3:PutObjectVersionTagging",
                    "s3:PutObjectTagging",
                    "s3:GetObject",
                    "s3:GetObjectVersion",
                    "s3:GetObjectAcl",
                    "s3:GetObjectTagging",
                    "s3:GetObjectVersionAcl",
                    "s3:GetObjectVersionTagging"
                ],
                "Resource": [
                    "arn:aws:s3:::ObjectDestinationBucket/*",
                    "arn:aws:s3:::ObjectSourceBucket/*",
                    "arn:aws:s3:::ObjectSourceManifestBucket/*"
                ]
            }
        ]
}
```

   이 역할은 정책을 사용하여 원본 매니페스트 버킷의 매니페스트를 읽을 수 있는 `batchoperations.s3.amazonaws.com` 권한을 부여합니다. 원본 객체 버킷의 GET 객체, ACL, 태그 및 버전에 대한 권한을 부여합니다. 또한 대상 객체 버킷의 PUT 객체 , ACL, 태그 및 버전에 대한 권한을 부여합니다.

2. 원본 계정에서, 이전 단계에 생성한 역할을 원본 매니페스트 버킷의 GET 객체와 버전에 제공하려면 매니페스트가 포함된 버킷에 대한 버킷 정책을 생성합니다.

   이 단계에서는 Amazon S3 Batch Operations가 신뢰할 수 있는 역할을 사용하여 매니페스트를 읽을 수 있습니다. 매니페스트가 포함된 버킷에 버킷 정책을 적용합니다.

다음은 원본 매니페스트 버킷에 적용할 버킷 정책의 예입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowBatchOperationsSourceManfiestRead",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::DestinationAccountNumber:user/
ConsoleUserCreatingJob",
                    "arn:aws:iam::DestinationAccountNumber:role/
BatchOperationsDestinationRoleCOPY"
                ]
            },
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::ObjectSourceManifestBucket/*"
        }
    ]
}
```

또한 이 정책은 대상 계정에서 작업을 생성하는 콘솔 사용자가 동일한 버킷 정책을 통해 원본 매니페스트 버킷의 동일한 권한을 사용할 수 있는 권한을 부여합니다.

3. 원본 계정에서, 생성한 역할을 원본 객체 버킷의 GET 객체, ACL, 태그 및 버전에 제공하는 원본 버킷에 대한 버킷 정책을 생성합니다. 그러면 Amazon S3 Batch Operations가 신뢰할 수 있는 역할을 통해 원본 버킷에서 객체를 가져올 수 있습니다.

   다음은 원본 객체가 포함된 버킷에 대한 버킷 정책의 예입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowBatchOperationsSourceObjectCOPY",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::DestinationAccountNumber:role/
BatchOperationsDestinationRoleCOPY"
            },
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:GetObjectAcl",
                "s3:GetObjectTagging",
                "s3:GetObjectVersionAcl",
                "s3:GetObjectVersionTagging"
            ],
            "Resource": "arn:aws:s3:::ObjectSourceBucket/*"
        }
    ]
}
```

4. 대상 계정에 Amazon S3 Batch Operations 작업을 생성합니다. 대상 계정에서 생성한 역할에 대한 Amazon 리소스 이름(ARN)이 필요합니다.

   작업 생성에 대한 일반적인 정보는 Amazon S3 배치 작업 건 생성 (p. 504) 단원을 참조하십시오.

콘솔을 사용하여 작업을 생성하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 Amazon S3의 배치 작업 건 생성을 참조하십시오.

AWS CLI를 사용하는 Amazon S3 Batch Operations 예제

Amazon S3 Batch 작업은 진행 상황을 추적하고 알림을 보내며 모든 작업에 대한 자세한 완료 보고서를 저장하여 감사 가능한 종합 관리형 서버리스 환경을 제공합니다. AWS Management 콘솔, AWS CLI, AWS SDK 또는 REST API를 통해 Amazon S3 Batch 작업을 사용할 수 있습니다. 자세한 내용은 the section called “기본 사항: 작업” (p. 502) 단원을 참조하십시오.

다음은 Amazon S3 Batch Operations와 함께 AWS Command Line Interface(AWS CLI)를 사용할 수 있는 방법을 보여주는 예제입니다.

주제

- AWS CLI를 사용하여 Amazon S3 Batch Operations 작업 생성 (p. 536)
- AWS CLI를 사용하여 Amazon S3 Batch Operations 작업 태그 지정 (p. 538)

AWS CLI를 사용하여 Amazon S3 Batch Operations 작업 생성

다음 예제에서는 AWS CLI를 사용하여 Amazon S3 Batch Operations `S3PutObjectTagging` 작업을 생성합니다.

배치 작업 **`S3PutObjectTagging`** 작업을 생성하려면

1. AWS Identity and Access Management(IAM) 역할을 생성하고 권한을 할당합니다. 이 역할은 다음 단계에서 작업을 생성하려는 객체 태그를 추가할 Amazon S3 권한을 부여합니다.

   a. 다음과 같이 IAM 역할을 생성합니다.

```
aws iam create-role \
 --role-name S3BatchJobRole \
 --assume-role-policy-document '{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}'
```

   역할의 Amazon 리소스 이름(ARN)을 기록합니다. 작업을 생성할 때 ARN이 필요합니다.

   b. 권한을 사용하여 IAM 정책을 생성하고 이전 단계에서 생성한 IAM 역할에 연결합니다. 권한에 대한 자세한 내용은 Amazon S3 배치 작업에 대한 권한 부여 (p. 506) 단원을 참조하십시오.

```
aws iam put-role-policy \
  --role-name S3BatchJobRole \
  --policy-name PutObjectTaggingBatchJobPolicy \
  --policy-document '{
  "Version":"2012-10-17",
  "Statement":[
```

```
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObjectTagging",
        "s3:PutObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::{{TargetResource}}/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::{{ManifestBucket}}/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject",
        "s3:GetBucketLocation"
      ],
      "Resource":[
        "arn:aws:s3:::{{ReportBucket}}/*"
      ]
    }
  ]
}'
```

2. `S3PutObjectTagging` 작업을 생성합니다.

   `manifest.csv` 파일은 일련의 버킷 및 객체 키 값을 제공합니다. 작업은 매니페스트에서 식별된 객체에 지정된 태그를 적용합니다. `ETag`는 `manifest.csv` 객체의 ETag이며 Amazon S3 콘솔에서 가져올 수 있습니다. 요청은 `no-confirmation-required` 파라미터를 지정합니다. 그러므로 사용자가 `udpate-job-status` 명령을 사용하여 확인하지 않아도 Amazon S3가 작업을 실행 가능하도록 만듭니다.

```
aws s3control create-job \
    --region us-west-2 \
    --account-id acct-id \
    --operation '{"S3PutObjectTagging": { "TagSet": [{"Key":"keyOne",
 "Value":"ValueOne"}] }}' \
    --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
["Bucket","Key"]},"Location":{"ObjectArn":"arn:aws:s3:::my_manifests/
manifest.csv","ETag":"60e460c9d1046e73f7dde5043ac3ae85"}}' \
    --report '{"Bucket":"arn:aws:s3:::bucket-where-
completion-report-goes","Prefix":"final-reports",
 "Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}' \
    --priority 42 \
    --role-arn IAM-role \
    --client-request-token $(uuidgen) \
    --description "job Description" \
    --no-confirmation-required
```

   Amazon S3는 응답으로 작업 ID(예: `00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c`)를 반환합니다. 다음 명령에 이 ID가 필요합니다.

3. 작업 설명을 가져옵니다

```
aws s3control describe-job \
```

```
    --region us-west-2 \
    --account-id acct-id \
    --job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c
```

4. `Active` 및 `Complete` 작업의 목록을 가져옵니다.

```
aws s3control list-jobs \
    --region us-west-2 \
    --account-id acct-id \
    --job-statuses '["Active","Complete"]' \
    --max-results 20
```

5. 작업 우선 순위를 업데이트합니다(숫자가 높을수록 실행 우선 순위가 높음).

```
aws s3control update-job-priority \
    --region us-west-2 \
    --account-id acct-id \
    --priority 98 \
    --job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c
```

6. `create-job`에서 `--no-confirmation-required` 파라미터를 지정하지 않은 경우 작업은 사용자가 그 상태를 `Ready`로 설정하여 확인할 때 까지 보류 상태로 유지됩니다. 그러면 Amazon S3가 해당 작업을 실행 가능하도록 만듭니다.

```
aws s3control update-job-status \
    --region us-west-2 \
    --account-id 181572960644 \
    --job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c \
    --requested-job-status 'Ready'
```

7. 작업 상태를 `Cancelled`로 설정하여 작업을 취소합니다.

```
aws s3control update-job-status \
     --region us-west-2 \
     --account-id 181572960644 \
     --job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c \
     --status-update-reason "No longer needed" \
     --requested-job-status Cancelled
```

AWS CLI를 사용하여 Amazon S3 Batch Operations 작업 태그 지정

태그를 추가하여 Amazon S3 Batch Operations 작업에 대한 레이블을 지정하고 액세스를 제어할 수 있습니다. 태그는 배치 작업 작업 담당자를 식별하는 데 사용할 수 있습니다. 태그가 연결된 작업을 생성할 수 있으며 작업을 생성한 후 작업에 태그를 추가할 수 있습니다. 자세한 내용은 the section called “태그를 사용하여 액세스 제어 및 작업 레이블 지정” (p. 520) 단원을 참조하십시오.

주제

- 태그를 사용하여 Amazon S3 Batch Operations 작업 생성 (p. 539)
- Amazon S3 Batch Operations 작업의 태그 삭제 (p. 540)
- Amazon S3 Batch Operations 작업의 작업 태그 가져오기 (p. 540)
- 기존 Amazon S3 Batch Operations 작업에 작업 태그 지정 (p. 540)

태그를 사용하여 Amazon S3 Batch Operations 작업 생성

다음 예제에서는 AWS CLI를 사용하는 작업의 레이블로 작업 태그를 사용하여 Amazon S3 Batch Operations `S3PutObjectCopy` 작업을 생성합니다.

1. 배치 작업 작업에서 수행하려는 작업 또는 `OPERATION`을 선택하고 `TargetResource`를 선택합니다.

```
read -d '' OPERATION <<EOF
{
  "S3PutObjectCopy": {
    "TargetResource": "arn:aws:s3:::destination-bucket"
  }
}
EOF
```

2. 작업에 대해 원하는 작업 `TAGS`를 식별합니다. 이 경우 두 개의 태그 `department` 및 `FiscalYear`를 각각 `2020` 및 `Marketing` 값과 함께 적용합니다.

```
read -d '' TAGS <<EOF
[
  {
    "Key": "department",
    "Value": "Marketing"
  },
  {
    "Key": "FiscalYear",
    "Value": "2020"
  }
]
EOF
```

3. 배치 작업 작업에 대해 `MANIFEST`를 지정합니다.

```
read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "EXAMPLE_S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::example-bucket/example_manifest.csv",
    "ETag": "example-5dc7a8bfb90808fc5d546218"
  }
}
EOF
```

4. 배치 작업 작업에 대해 `REPORT`를 구성합니다.

```
read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::example-report-bucket",
  "Format": "Example_Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/copy-with-replace-metadata",
  "ReportScope": "AllTasks"
}
EOF
```

5. `create-job` 작업을 실행하여 이전 단계에서 설정된 입력으로 배치 작업 작업을 생성합니다.

```
aws \
    s3control create-job \
    --account-id 123456789012 \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn arn:aws:iam::123456789012:role/batch-operations-role \
    --tags "${TAGS//$'\n'/}" \
    --client-request-token "$(uuidgen)" \
    --region us-west-2 \
    --description "Copy with Replace Metadata";
```

Amazon S3 Batch Operations 작업의 태그 삭제

다음 예제에서는 AWS CLI를 사용하여 배치 작업 작업에서 태그를 삭제합니다.

```
aws \
    s3control delete-job-tagging \
    --account-id 123456789012 \
    --job-id Example-e25a-4ed2-8bee-7f8ed7fc2f1c \
    --region us-east-1;
```

Amazon S3 Batch Operations 작업의 작업 태그 가져오기

다음 예제에서는 AWS CLI를 사용하여 배치 작업 작업의 태그를 가져옵니다.

```
aws \
    s3control get-job-tagging \
    --account-id 123456789012 \
    --job-id Example-e25a-4ed2-8bee-7f8ed7fc2f1c \
    --region us-east-1;
```

기존 Amazon S3 Batch Operations 작업에 작업 태그 지정

다음은 AWS CLI를 사용하는 `s3control put-job-tagging`을 통해 Amazon S3 Batch Operations 작업에 작업 태그를 추가하는 예제입니다.

> **Note**
>
> 빈 태그 세트로 이 요청을 전송하는 경우 S3 배치 작업는 객체에 있는 기존 태그 세트를 삭제합니다. 또한 이 방법을 사용하면 계층 1 요청(PUT)에 대한 요금이 부과됩니다. 자세한 내용은 Amazon S3 요금을 참조하십시오.
> 배치 작업 작업에 대한 기존 태그를 삭제하려면 비용이 발생하지 않고 동일한 결과를 얻을 수 있는 `DeleteJobTagging` 작업이 선호됩니다.

1. 작업에 대해 원하는 작업 `TAGS`를 식별합니다. 이 경우 두 개의 태그 `department` 및 `FiscalYear`를 각각 `2020` 및 `Marketing` 값과 함께 적용합니다.

```
read -d '' TAGS <<EOF
[
  {
    "Key": "department",
    "Value": "Marketing"
  },
  {
    "Key": "FiscalYear",
    "Value": "2020"
```

```
  }
]
EOF
```

2. 필요한 파라미터를 사용하여 `put-job-tagging` 작업을 실행합니다.

```
aws \
    s3control put-job-tagging \
    --account-id 123456789012 \
    --tags "${TAGS//$'\n'/}" \
    --job-id Example-e25a-4ed2-8bee-7f8ed7fc2f1c \
    --region us-east-1;
```

예: AWS SDK for Java를 사용하여 Amazon S3 Batch Operations 작업 생성 및 관리

이 단원에서는 AWS SDK for Java를 사용하여 Amazon S3 Batch Operations 작업을 생성 및 관리하는 방법에 대한 예제를 제공합니다. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

주제

- 배치 작업 작업 생성 (p. 541)
- 배치 작업 작업 취소 (p. 542)
- 배치 작업 작업의 상태 업데이트 (p. 543)
- 배치 작업 작업의 우선 순위 업데이트 (p. 544)
- 태그가 있는 배치 작업 사용 (p. 545)

배치 작업 작업 생성

다음 예제에서는 AWS SDK for Java를 사용하여 Amazon S3 Batch Operations 작업을 생성합니다.

작업 생성에 대한 자세한 내용은 Amazon S3 배치 작업 건 생성 (p. 504) 단원을 참조하십시오.

작업을 생성하는 데 필요한 권한 설정에 대한 자세한 내용은 Amazon S3 배치 작업에 대한 권한 부여 (p. 506) 단원을 참조하십시오.

Example

```
package aws.example.s3control;



import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.*;

import java.util.UUID;
import java.util.ArrayList;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateJob {
```

```
    public static void main(String[] args) {
        String accountId = "Account ID";
        String iamRoleArn = "IAM Role ARN";
        String reportBucketName = "arn:aws:s3:::bucket-where-completion-report-goes";
        String uuid = UUID.randomUUID().toString();

        ArrayList tagSet = new ArrayList<S3Tag>();
        tagSet.add(new S3Tag().withKey("keyOne").withValue("ValueOne"));


        try {
            JobOperation jobOperation = new JobOperation()
                    .withS3PutObjectTagging(new S3SetObjectTaggingOperation()
                            .withTagSet(tagSet)
                    );

            JobManifest manifest = new JobManifest()
                    .withSpec(new JobManifestSpec()
                            .withFormat("S3BatchOperations_CSV_20180820")
                            .withFields(new String[]{
                                    "Bucket", "Key"
                            }))
                    .withLocation(new JobManifestLocation()
                            .withObjectArn("arn:aws:s3:::my_manifests/manifest.csv")
                            .withETag("60e460c9d1046e73f7dde5043ac3ae85"));
            JobReport jobReport = new JobReport()
                    .withBucket(reportBucketName)
                    .withPrefix("reports")
                    .withFormat("Report_CSV_20180820")
                    .withEnabled(true)
                    .withReportScope("AllTasks");

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.createJob(new CreateJobRequest()
                    .withAccountId(accountId)
                    .withOperation(jobOperation)
                    .withManifest(manifest)
                    .withReport(jobReport)
                    .withPriority(42)
                    .withRoleArn(iamRoleArn)
                    .withClientRequestToken(uuid)
                    .withDescription("job description")
                    .withConfirmationRequired(false)
            );

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and 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();
        }
    }
}
```

배치 작업 작업 취소

다음 예제에서는 AWS SDK for Java를 사용하여 Amazon S3 Batch Operations 작업을 취소합니다.

Example

```
package aws.example.s3control;


import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.UpdateJobStatusRequest;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CancelJob {
    public static void main(String[] args) {
        String accountId = "Account ID";
        String jobId = "00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c";

        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.updateJobStatus(new UpdateJobStatusRequest()
                    .withAccountId(accountId)
                    .withJobId(jobId)
                    .withStatusUpdateReason("No longer needed")
                    .withRequestedJobStatus("Cancelled"));

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and 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();
        }
    }
}
```

배치 작업 작업의 상태 업데이트

다음 예제에서는 AWS SDK for Java를 사용하여 Amazon S3 Batch Operations 작업의 상태를 업데이트합니다.

작업 상태에 대한 자세한 내용은 작업 상태 (p. 521) 단원을 참조하십시오.

Example

```
package aws.example.s3control;


import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.UpdateJobStatusRequest;
```

```
import static com.amazonaws.regions.Regions.US_WEST_2;

public class UpdateJobStatus {
    public static void main(String[] args) {
        String accountId = "Account ID";
        String jobId = "00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c";

        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.updateJobStatus(new UpdateJobStatusRequest()
                    .withAccountId(accountId)
                    .withJobId(jobId)
                    .withRequestedJobStatus("Ready"));

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and 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();
        }
    }
}
```

배치 작업 작업의 우선 순위 업데이트

다음 예제에서는 AWS SDK for Java를 사용하여 Amazon S3 Batch Operations 작업의 우선 순위를 업데이트합니다.

작업 우선 순위에 대한 자세한 내용은 작업 우선 순위 지정 (p. 521) 단원을 참조하십시오.

Example

```
package aws.example.s3control;



import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.UpdateJobPriorityRequest;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class UpdateJobPriority {
    public static void main(String[] args) {
        String accountId = "Account ID";
        String jobId = "00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c";

        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
```

```
                    .build();

            s3ControlClient.updateJobPriority(new UpdateJobPriorityRequest()
                    .withAccountId(accountId)
                    .withJobId(jobId)
                    .withPriority(98));

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and 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();
        }
    }
}
```

태그가 있는 배치 작업 사용

태그를 추가하여 Amazon S3 Batch Operations 작업에 대한 레이블을 지정하고 액세스를 제어할 수 있습니다. 태그는 배치 작업 작업 담당자를 식별하는 데 사용할 수 있습니다. 태그가 연결된 작업을 생성할 수 있으며 작업을 생성한 후 작업에 태그를 추가할 수 있습니다. 자세한 내용은 the section called “태그를 사용하여 액세스 제어 및 작업 레이블 지정” (p. 520) 단원을 참조하십시오.

주제

- 레이블 지정에 작업 태그를 사용하는 배치 작업 작업 생성 (p. 545)
- 배치 작업 작업의 작업 태그 삭제 (p. 546)
- 배치 작업 작업의 작업 태그 가져오기 (p. 546)
- 배치 작업 작업의 작업 태그 넣기 (p. 547)

레이블 지정에 작업 태그를 사용하는 배치 작업 작업 생성

Example

다음 예제에서는 AWS SDK for Java를 사용하여 태그가 있는 Amazon S3 Batch Operations 작업을 생성합니다.

```
public String createJob(final AWSS3ControlClient awss3ControlClient) {
    final String manifestObjectArn = "arn:aws:s3:::example-manifest-bucket/
manifests/10_manifest.csv";
    final String manifestObjectVersionId = "example-5dc7a8bfb90808fc5d546218";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new
 JobManifestSpec().withFormat(JobManifestFormat.S3InventoryReport_CSV_20161130);

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::example-report-bucket";
    final String jobReportPrefix = "example-job-reports";
```

```
    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final String lambdaFunctionArn = "arn:aws:lambda:us-
west-2:123456789012:function:example-function";

    final JobOperation jobOperation = new JobOperation()
            .withLambdaInvoke(new
 LambdaInvokeOperation().withFunctionArn(lambdaFunctionArn));

    final S3Tag departmentTag = new S3Tag().withKey("department").withValue("Marketing");
    final S3Tag fiscalYearTag = new S3Tag().withKey("FiscalYear").withValue("2020");

    final String roleArn = "arn:aws:iam::123456789012:role/example-batch-operations-role";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Test lambda job")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withTags(departmentTag, fiscalYearTag)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

배치 작업 작업의 작업 태그 삭제

Example

다음 예제에서는 AWS SDK for Java를 사용하여 Amazon S3 Batch Operations 작업의 태그를 삭제합니다.

```
public void deleteJobTagging(final AWSS3ControlClient awss3ControlClient,
                             final String jobId) {
    final DeleteJobTaggingRequest deleteJobTaggingRequest = new DeleteJobTaggingRequest()
            .withJobId(jobId);

    final DeleteJobTaggingResult deleteJobTaggingResult =
                awss3ControlClient.deleteJobTagging(deleteJobTaggingRequest);
}
```

배치 작업 작업의 작업 태그 가져오기

Example

다음 예제에서는 AWS SDK for Java를 사용하여 Amazon S3 Batch Operations 작업의 태그를 가져옵니다.

```
public List<S3Tag> getJobTagging(final AWSS3ControlClient awss3ControlClient,
                                 final String jobId) {
    final GetJobTaggingRequest getJobTaggingRequest = new GetJobTaggingRequest()
            .withJobId(jobId);
```

```
    final GetJobTaggingResult getJobTaggingResult =
            awss3ControlClient.getJobTagging(getJobTaggingRequest);

    final List<S3Tag> tags = getJobTaggingResult.getTags();

    return tags;
}
```

배치 작업 작업의 작업 태그 넣기

Example

다음 예제에서는 AWS SDK for Java를 사용하여 Amazon S3 Batch Operations 작업의 태그를 넣습니다.

```
public void putJobTagging(final AWSS3ControlClient awss3ControlClient,
                          final String jobId) {
    final S3Tag departmentTag = new S3Tag().withKey("department").withValue("Marketing");
    final S3Tag fiscalYearTag = new S3Tag().withKey("FiscalYear").withValue("2020");

    final PutJobTaggingRequest putJobTaggingRequest = new PutJobTaggingRequest()
            .withJobId(jobId)
            .withTags(departmentTag, fiscalYearTag);

    final PutJobTaggingResult putJobTaggingResult =
 awss3ControlClient.putJobTagging(putJobTaggingRequest);
}
```

Amazon S3 정적 웹 사이트 호스팅

Amazon S3를 사용하여 정적 웹 사이트를 호스팅할 수 있습니다. 정적 웹 사이트에서 개별 웹 페이지는 정적 콘텐츠를 포함합니다. 클라이언트 측 스크립트를 포함할 수도 있습니다.

이와는 대조적으로 동적 웹 사이트는 PHP, JSP 또는 ASP.NET과 같은 서버 측 스크립트를 포함한 서버 측 처리에 의존합니다. Amazon S3에서는 서버 측 스크립팅을 지원하지 않지만 AWS에는 동적 웹 사이트를 호스팅하는 다른 리소스가 있습니다. AWS에서의 웹 사이트 호스팅에 대한 자세한 내용은 웹 호스팅을 참조하십시오.

Amazon S3에서 정적 웹 사이트를 호스팅하려면 Amazon S3 버킷을 웹 사이트 호스팅용으로 구성한 후 웹 사이트 콘텐츠를 버킷에 업로드합니다. 버킷을 정적 웹 사이트로 구성할 때 정적 웹 사이트 호스팅을 활성화하고, 권한을 설정하고, 인덱스 문서를 추가합니다. 웹 사이트 요구 사항에 따라 리디렉션, 웹 트래픽 로깅 및 사용자 지정 오류 문서 등의 다른 옵션을 구성할 수도 있습니다.

버킷을 정적 웹 사이트로 구성한 후 버킷의 AWS 리전별 Amazon S3 웹 사이트 엔드포인트를 통해 버킷에 액세스할 수 있습니다. 웹 사이트 엔드포인트는 사용자가 REST API 요청을 보내는 엔드포인트와 다릅니다. 자세한 내용은 웹 사이트 엔드포인트 (p. 548) 단원을 참조하십시오.

정적 웹 사이트 호스팅용 버킷을 구성하려면 코드를 작성하지 않고 AWS Management 콘솔을 사용할 수 있습니다. 또한 AWS SDK를 사용한 프로그래밍 방식으로 웹 사이트 구성을 생성, 업데이트, 삭제할 수 있습니다. SDK는 Amazon S3 REST API를 둘러싼 래퍼 클래스를 제공합니다. 애플리케이션에서 필요한 경우, 해당 애플리케이션에서 직접 REST API 요청할 수 있습니다.

지침 및 단계별 연습을 비롯한 자세한 내용은 다음 주제를 참조하십시오.

주제

- 웹 사이트 엔드포인트 (p. 548)
- AWS Management 콘솔을 사용하여 버킷을 정적 웹 사이트로 구성 (p. 550)
- 프로그래밍 방식으로 버킷을 정적 웹 사이트로 구성 (p. 564)
- 연습 예제 - Amazon S3 웹 사이트 호스팅 (p. 568)

웹 사이트 엔드포인트

버킷을 정적 웹 사이트로 구성하면 버킷의 AWS 리전별 웹 사이트 엔드포인트를 통해 웹 사이트를 사용할 수 있습니다. 웹 사이트 엔드포인트는 사용자가 REST API 요청을 보내는 엔드포인트와 다릅니다. 엔드포인트 간의 차이에 대한 자세한 내용은 웹 사이트 엔드포인트와 REST API 엔드포인트 간의 주요 차이점 (p. 550)을 참조하십시오.

리전에 따라 Amazon S3 웹 사이트 엔드포인트는 다음 두 형식 중 하나를 따릅니다. URL은 사용자가 웹 사이트에 대해 구성한 기본 인덱스 문서를 반환합니다.

```
http://bucket-name.s3-website.Region.amazonaws.com
```

```
http://bucket-name.s3-website-Region.amazonaws.com
```

Amazon S3 웹 사이트 엔드포인트의 전체 목록은 AWS General Reference의 Amazon S3 웹 사이트 엔드포인트를 참조하십시오.

고객이 웹 사이트 엔드포인트의 콘텐츠에 액세스하려면 콘텐츠가 공개적으로 읽기 가능하도록 설정되어 있어야 합니다. 이를 위해 버킷에 대한 S3 퍼블릭 액세스 차단 설정을 편집할 수 있습니다. 자세한 내용은 Amazon S3 퍼블릭 액세스 차단 사용 (p. 450) 단원을 참조하십시오. 그런 다음 객체에 버킷 정책 또는 ACL(액세스 제어 목록)을 사용하여 필요한 권한을 부여합니다. 자세한 내용은 웹 사이트 액세스에 대한 권한 설정 (p. 553) 단원을 참조하십시오.

Important

Amazon S3 웹 사이트 엔드포인트는 HTTPS를 지원하지 않습니다. Amazon S3 버킷과 함께 HTTPS를 사용하는 방법에 대한 자세한 내용은 다음을 참조하십시오.

- CloudFront를 사용하여 내 Amazon S3 버킷에 대한 HTTPS 요청을 처리하는 방법은 무엇입니까?
- CloudFront와 Amazon S3 오리진 간 통신에 HTTPS 필요

요청자 지불 버킷은 웹 사이트 엔드포인트를 통한 액세스를 허용하지 않습니다. 해당 버킷으로의 모든 요청은 403 Access Denied 응답을 수신합니다. 자세한 내용은 요청자 지불 버킷 (p. 74) 단원을 참조하십시오.

주제

- 웹 사이트 엔드포인트 예제 (p. 549)
- DNS CNAME 추가 (p. 549)
- Route 53와 함께 사용자 지정 도메인 사용 (p. 550)
- 웹 사이트 엔드포인트와 REST API 엔드포인트 간의 주요 차이점 (p. 550)

웹 사이트 엔드포인트 예제

다음 예제에서는 정적 웹 사이트로 구성된 Amazon S3 버킷에 액세스하는 방법을 보여줍니다.

Example — 루트 수준의 객체 요청

버킷의 루트 수준에 저장된 특정 객체를 요청하려면 다음 URL 구조를 사용합니다.

```
http://bucket-name.s3-website.Region.amazonaws.com/object-name
```

예를 들어, 다음 URL은 버킷의 루트 수준에 저장된 `photo.jpg` 객체를 요청합니다.

```
http://example-bucket.s3-website.us-west-2.amazonaws.com/photo.jpg
```

Example — 접두사로 객체 요청

버킷의 폴더에 저장된 객체를 요청하려면 다음 URL 구조를 사용합니다.

```
http://bucket-name.s3-website.Region.amazonaws.com/folder-name/object-name
```

다음 URL은 버킷의 `docs/doc1.html` 객체를 요청합니다.

```
http://example-bucket.s3-website.us-west-2.amazonaws.com/docs/doc1.html
```

DNS CNAME 추가

등록된 도메인이 있는 경우 DNS CNAME 항목을 추가하면 Amazon S3 웹 사이트 엔드포인트를 가리킬 수 있습니다. 예를 들어, 등록된 `www.example-bucket.com` 도메인이 있다면 `www.example-bucket.com` 버킷을 생성하고 `www.example-bucket.com.s3-website.Region.amazonaws.com`을 가리키는 DNS CNAME 레코드를 추가할 수 있습니다. `http://www.example-bucket.com`으로의 모든 요청은 `www.example-bucket.com.s3-website.Region.amazonaws.com`으로 라우팅됩니다.

자세한 내용은 Amazon S3 URL을 CNAME으로 사용자 지정 (p. 46) 단원을 참조하십시오.

Route 53와 함께 사용자 지정 도메인 사용

Amazon S3 웹 사이트 엔드포인트를 사용하여 웹 사이트에 액세스하는 대신, Amazon Route 53에 등록된 자체 도메인을 사용하여 콘텐츠를 처리할 수 있습니다(예: `example.com`). Amazon S3 및 Route 53를 사용하여 루트 도메인에서 웹 사이트를 호스팅할 수 있습니다. 예를 들어, 루트 도메인 `example.com`을 보유하고 Amazon S3에서 웹 사이트를 호스팅하는 경우 웹 사이트 방문자는 `http://www.example.com` 또는 `http://example.com`을 입력하여 브라우저에서 사이트에 액세스할 수 있습니다.

예제 연습은 Route 53에 등록된 사용자 지정 도메인을 사용하여 정적 웹 사이트 구성 (p. 574)을 참조하십시오.

웹 사이트 엔드포인트와 REST API 엔드포인트 간의 주요 차이점

Amazon S3 웹 사이트 엔드포인트는 웹 브라우저를 통한 액세스에 최적화되었습니다. 다음 표에는 REST API 엔드포인트와 웹 사이트 엔드포인트 간의 주요 차이점이 요약되어 있습니다.

주요 차이점	REST API 엔드포인트	웹 사이트 엔드포인트
액세스 제어	퍼블릭 콘텐츠 및 프라이빗 콘텐츠 모두 지원	공개적으로 읽기 가능한 콘텐츠만 지원
오류 메시지 처리	XML 형식의 오류 응답 반환	HTML 문서 반환
리디렉션 지원	해당 사항 없음	객체 수준 및 버킷 수준의 리디렉션 모두 지원
요청 지원	모든 버킷 및 객체 작업 지원	객체에 대한 GET 및 HEAD 요청 지원
버킷의 루트에서 GET 및 Head 요청에 대한 응답	버킷의 객체 키 목록 반환	웹 사이트 구성에 지정된 인덱스 문서 반환
Secure Sockets Layer(SSL) 지원	SSL 연결 지원	SSL 연결을 지원하지 않음

Amazon S3 엔드포인트의 전체 목록은 AWS General Reference의 Amazon S3 서비스 엔드포인트 및 할당량을 참조하십시오.

AWS Management 콘솔을 사용하여 버킷을 정적 웹 사이트로 구성

AWS Management 콘솔을 사용하면 코드를 작성하지 않고 Amazon S3 버킷을 정적 웹 사이트로 구성할 수 있습니다. 버킷을 정적 웹 사이트로 구성할 때 정적 웹 사이트 호스팅을 활성화하고, 권한을 설정하고, 인덱스 문서를 추가합니다. 웹 사이트 요구 사항에 따라 리디렉션, 웹 트래픽 로깅 및 사용자 지정 오류 문서를 비롯한 일부 선택적 구성을 사용할 수도 있습니다.

필요한 구성:

- 웹 사이트 호스팅 사용 (p. 551)
- 인덱스 문서 구성 (p. 551)
- 웹 사이트 액세스에 대한 권한 설정 (p. 553)

구성 옵션:

- (선택 사항) 웹 트래픽 로깅 구성 (p. 556)
- (선택 사항)사용자 지정 오류 문서 구성 (p. 557)
- (선택 사항) 웹 페이지 리디렉션 구성 (p. 559)

웹 사이트 호스팅 사용

다음 단계에 따라 Amazon S3 콘솔을 사용하여 Amazon S3 버킷에서 웹 사이트 호스팅을 활성화합니다. 사용자 지정 도메인과 같은 선택적 구성을 사용하여 웹 사이트를 설정하는 방법을 보여 주는 예제 연습은 연습 예제 - Amazon S3 웹 사이트 호스팅 (p. 568) 단원을 참조하십시오.

정적 웹 사이트 호스팅을 활성화하려면

1. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷 이름 목록에서 정적 웹 사이트에 사용할 버킷을 선택합니다.
3. [Properties]를 선택합니다.
4. [Static website hosting]을 선택합니다.
5. 이 버킷을 사용하여 웹 사이트를 호스팅합니다.를 선택합니다.
6. 인덱스 문서의 이름을 입력합니다.

   인덱스 문서 이름은 일반적으로 `index.html`입니다. 인덱스 문서 이름은 대소문자를 구분하며 S3 버킷에 업로드하려는 HTML 인덱스 문서의 파일 이름과 정확히 일치해야 합니다. 자세한 내용은 인덱스 문서 구성 (p. 551) 단원을 참조하십시오.
7. (선택 사항) 사용자 지정 오류 문서를 추가할 경우 오류 문서 상자에 오류 문서 이름(예: **`error.html`**)을 입력합니다.

   오류 문서 이름은 대소문자를 구분하며 S3 버킷에 업로드하려는 HTML 오류 문서의 파일 이름과 정확히 일치해야 합니다. 자세한 내용은 (선택 사항)사용자 지정 오류 문서 구성 (p. 557) 단원을 참조하십시오.
8. (선택 사항) 고급 리디렉션 규칙을 지정하려면 Edit redirection rules(리디렉션 규칙 편집)에서 XML을 사용하여 규칙을 설명합니다.

   자세한 내용은 고급 조건부 리디렉션 구성 (p. 561) 단원을 참조하십시오.
9. 정적 웹 사이트 호스팅에서 엔드포인트를 기록합니다.

   엔드포인트는 버킷의 Amazon S3 웹 사이트 엔드포인트입니다. 버킷을 정적 웹 사이트로 구성한 후 이 엔드포인트를 사용하여 웹 사이트를 테스트할 수 있습니다.
10. 저장을 선택합니다.

인덱스 문서 구성

인덱스 문서는 웹 사이트의 루트나 임의의 하위 폴더로 요청이 전달되는 경우에 Amazon S3가 반환하는 웹 페이지입니다. 예를 들어, 사용자가 브라우저에 `http://www.example.com`을 입력하는 경우 사용자는 특정 페이지를 요청하는 것입니다. 이 경우 Amazon S3가 기본 페이지라고도 하는 인덱스 문서를 표시합니다.

버킷용 정적 웹 사이트 호스팅을 활성화할 때 인덱스 문서의 이름(예: `index.html`)을 입력합니다. 버킷용 정적 웹 사이트 호스팅을 활성화한 후 인덱스 문서 이름이 있는 HTML 파일을 버킷에 업로드합니다.

루트 수준 URL의 후행 슬래쉬는 선택 사항입니다. 예를 들어, 인덱스 문서가 `index.html`인 웹 사이트를 구성하는 경우 다음과 같은 URL 중 하나에서 `index.html`이 반환됩니다.

```
http://example-bucket.s3-website.Region.amazonaws.com/
http://example-bucket.s3-website.Region.amazonaws.com
```

Amazon S3 웹 사이트 엔드포인트에 대한 자세한 정보는 웹 사이트 엔드포인트 (p. 548)를 참조하십시오.

인덱스 문서 및 폴더

Amazon S3에서 버킷은 객체의 플랫 컨테이너입니다. 버킷은 컴퓨터의 파일 시스템과 같이 계층 구조는 제공하지 않습니다. 폴더 구조를 의미하는 객체 키 이름을 사용하여 논리적 계층 구조를 만들 수 있습니다.

예를 들어, 키 이름이 다음과 같은 세 개의 객체가 있는 버킷이 있다고 생각해 보십시오. 객체가 물리적 계층 구조에 따라 저장되지는 않지만 키 이름에 비추어 논리적 폴더 구조가 다음과 같음을 유추할 수 있습니다.

- `sample1.jpg` — 객체는 버킷의 루트에 있습니다.
- `photos/2006/Jan/sample2.jpg` — 객체는 `photos/2006/Jan` 하위 폴더에 있습니다.
- `photos/2006/Feb/sample3.jpg` — 객체는 `photos/2006/Feb` 하위 폴더에 있습니다.

Amazon S3 콘솔에서 버킷에 폴더를 만들 수도 있습니다. 예를 들어 `photos`라는 폴더를 만들 수 있습니다. 이 버킷에 혹은 버킷 내부의 `photos` 폴더에 객체를 업로드할 수 있습니다. 버킷에 객체 `sample.jpg`를 추가하는 경우 키 이름은 `sample.jpg`입니다. `photos` 폴더에 객체를 업로드하는 경우 객체 키 이름은 `photos/sample.jpg`입니다.

버킷에 폴더 구조를 만드는 경우 수준별로 인덱스 문서가 있어야 합니다. 각 폴더에서 인덱스 문서의 이름은 동일해야 합니다(예: `index.html`). 사용자가 폴더 조회 URL을 지정할 때 후행 슬래시의 유무로 웹 사이트의 동작을 판단하게 됩니다. 예를 들어, 뒤에 슬래시가 있는 다음과 같은 URL은 `photos/index.html` 인덱스 문서를 반환합니다.

```
http://bucket-name.s3-website.Region.amazonaws.com/photos/
```

하지만 선행 URL에서 후행 슬래시를 제외하는 경우 Amazon S3은 가장 먼저 버킷의 객체 `photos`를 찾습니다. `photos` 객체를 찾을 수 없는 경우 인덱스 문서인 `photos/index.html`이 검색 대상입니다. 해당 문서를 찾은 경우 Amazon S3가 `302 Found` 메시지를 반환하고 `photos/` 키를 가리킵니다. `photos/`에 대한 후속 요청의 경우 Amazon S3가 `photos/index.html`을 반환합니다. 인덱스 문서를 찾을 수 없는 경우 Amazon S3에서 오류를 반환합니다.

인덱스 문서 구성

버킷용 정적 웹 사이트 호스팅을 활성화할 때 인덱스 문서의 이름(예: **`index.html`**)을 입력합니다. 버킷용 정적 웹 사이트 호스팅을 활성화한 후 인덱스 문서 이름이 있는 HTML 파일을 버킷에 업로드합니다.

인덱스 문서를 구성하려면

1. `index.html` 파일을 생성합니다.

   `index.html` 파일이 없으면 다음 HTML을 사용하여 파일을 생성할 수 있습니다.

   ```
   <html xmlns="http://www.w3.org/1999/xhtml" >
   <head>
       <title>My Website Home Page</title>
   </head>
   <body>
     <h1>Welcome to my website</h1>
     <p>Now hosted on Amazon S3!</p>
   </body>
   </html>
   ```

2. 인덱스 파일을 로컬에 저장하고 파일 이름(예: `index.html`)을 기록합니다.

정적 웹 사이트 호스팅을 활성화하고 인덱스 문서 이름을 입력하는 경우 정확한 파일 이름(예: `index.html`)을 입력합니다.

3. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
4. 버킷 목록에서 정적 웹 사이트를 호스팅하는 데 사용할 버킷의 이름을 선택합니다.
5. 버킷에 정적 웹 사이트 호스팅을 활성화하고 인덱스 문서의 정확한 이름(예: `index.html`)을 입력합니다. 자세한 내용은 웹 사이트 호스팅 사용 (p. 551) 단원을 참조하십시오.

   정적 웹 사이트 호스팅을 활성화한 후 6단계로 이동합니다.
6. 버킷에 인덱스 문서를 업로드하려면 다음 중 하나를 수행합니다.
   - 인덱스 파일을 콘솔 버킷 목록으로 끌어다 놓습니다.
   - 업로드를 선택하고 프롬프트의 메시지에 따라 인덱스 파일을 선택하고 업로드합니다.

   단계별 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 Amazon S3 버킷에 파일 및 폴더를 업로드하려면 어떻게 해야 합니까?를 참조하십시오.
7. (선택 사항) 버킷에 다른 웹 사이트 콘텐츠를 업로드합니다.

웹 사이트 액세스에 대한 권한 설정

버킷을 웹 사이트로 구성할 때 모든 사용자가 웹 사이트에 액세스할 수 있도록 버킷에 퍼블릭 읽기 액세스 권한을 부여해야 합니다. 버킷을 공개적으로 읽기 가능하게 만들려면 버킷에 대해 퍼블릭 액세스 차단 설정을 비활성화하고 버킷 정책을 작성해야 합니다. 버킷에 버킷 소유자가 소유하지 않은 객체가 포함되어 있는 경우 모든 사용자에게 읽기 권한을 부여하는 객체 ACL(액세스 제어 목록)을 추가해야 할 수도 있습니다.

> **Note**
>
> 웹 사이트 엔드포인트에서 사용자가 존재하지 않는 객체를 요청하는 경우 Amazon S3은 HTTP 응답 코드 `404 (Not Found)`를 반환합니다. 객체가 존재하지만 해당 객체에 대한 읽기 권한을 부여하지 않았다면 웹 사이트 엔드포인트는 HTTP 응답 코드 `403 (Access Denied)`을 반환합니다. 이 응답 코드를 사용하여 특정 객체가 존재하는지 여부를 유추할 수 있습니다. 이 동작을 원하지 않을 경우, 버킷에 웹 사이트 지원을 설정해서는 안 됩니다.

1단계: 퍼블릭 액세스 차단 설정 편집

기존 버킷을 퍼블릭 액세스를 포함하는 정적 웹 사이트로 구성하려면 해당 버킷에 대한 퍼블릭 액세스 차단 설정을 편집해야 합니다. 또한 계정 레벨의 퍼블릭 액세스 차단 설정을 편집해야 할 수도 있습니다. Amazon S3은 버킷 레벨 및 계정 레벨의 퍼블릭 액세스 차단 설정의 가장 제한적인 조합을 적용합니다.

예를 들어 버킷에 대한 퍼블릭 액세스는 허용하지만 계정 레벨의 모든 퍼블릭 액세스는 차단하는 경우 Amazon S3에서 해당 버킷에 대한 퍼블릭 액세스를 계속 차단합니다. 이 시나리오에서는 버킷 레벨 및 계정 레벨의 퍼블릭 액세스 차단 설정을 편집해야 할 수도 있습니다. 자세한 내용은 Amazon S3 퍼블릭 액세스 차단 사용 (p. 450) 단원을 참조하십시오.

기본적으로 Amazon S3은 계정 및 버킷에 대한 퍼블릭 액세스를 차단합니다. 버킷을 사용하여 정적 웹 사이트를 호스팅하려는 경우 이러한 단계를 사용하여 퍼블릭 액세스 차단 설정을 편집할 수 있습니다.

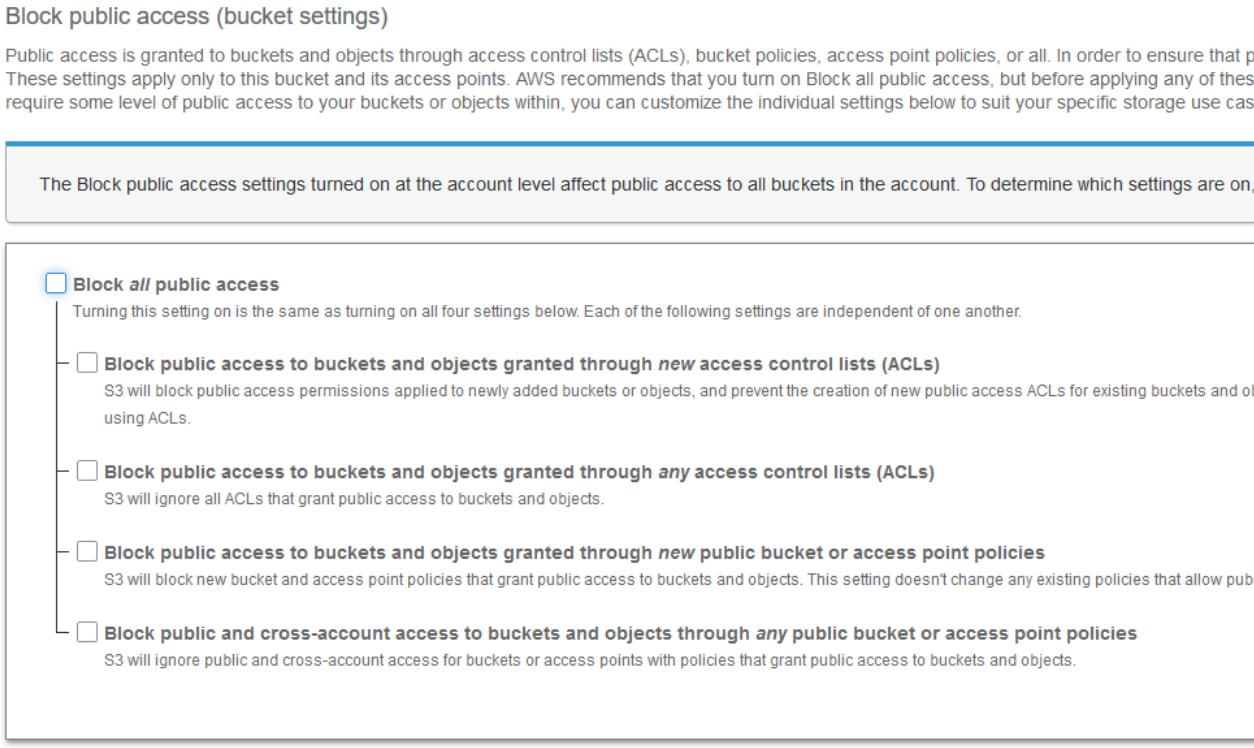
> **Warning**
>
> 이 단계를 완료하기 전에 Amazon S3 퍼블릭 액세스 차단 사용을 검토하여 퍼블릭 액세스 허용과 관련된 위험을 이해하고 이에 동의하는지 확인하십시오. 퍼블릭 액세스 차단 설정을 해제하여 버킷을 퍼블릭으로 만들면 인터넷상의 모든 사용자가 버킷에 액세스할 수 있습니다. 버킷에 대한 모든 퍼블릭 액세스를 차단하는 것이 좋습니다.

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 정적 웹 사이트로 구성한 버킷의 이름을 선택합니다.

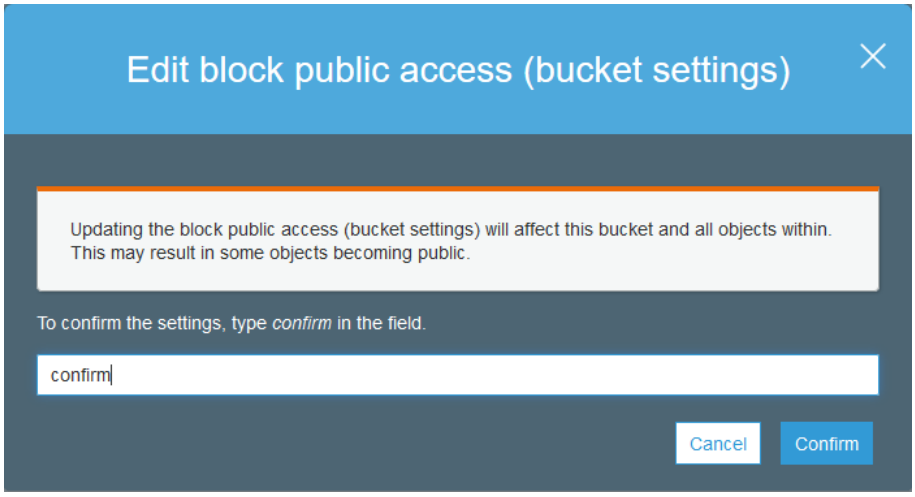
3. Permissions를 선택합니다.
4. [Edit]를 선택합니다.
5. Block all public access(모든 퍼블릭 액세스 차단)를 선택 취소하고 저장을 선택합니다.

   Warning

   이 단계를 완료하기 전에 Amazon S3 퍼블릭 액세스 차단 사용을 검토하여 퍼블릭 액세스 허용과 관련된 위험을 이해하고 이에 동의하는지 확인하십시오. 퍼블릭 액세스 차단 설정을 해제하여 버킷을 퍼블릭으로 만들면 인터넷상의 모든 사용자가 버킷에 액세스할 수 있습니다. 버킷에 대한 모든 퍼블릭 액세스를 차단하는 것이 좋습니다.



6. 확인 상자에 **`confirm`**을 입력한 다음 확인을 선택합니다.



S3 버킷에서 버킷에 대한 액세스가 Objects can be public(객체가 퍼블릭이 될 수 있음)으로 업데이트됩니다. 이제 버킷의 객체를 공개적으로 읽기 가능하게 만들 버킷 정책을 추가할 수 있습니다. 액세스가 계

속 Bucket and objects not public(버킷 및 객체가 퍼블릭이 아님)으로 표시되는 경우, 버킷 정책을 추가하기 전에 계정에 대해 퍼블릭 액세스 차단 설정을 편집해야 할 수도 있습니다.

2단계: 버킷 정책 추가

버킷의 객체를 공개적으로 읽기 가능하게 만들려면 모든 사용자에게 `s3:GetObject` 권한을 부여하는 버킷 정책을 작성해야 합니다. 다음의 예제 버킷 정책은 모든 사용자에게 특정 폴더의 객체에 대한 액세스를 허용합니다.

버킷에 퍼블릭 읽기 액세스 권한을 부여하는 버킷 정책을 추가하기 전에 해당 버킷에 대한 퍼블릭 액세스 차단을 비활성화했는지 확인합니다. 다음 버킷 정책을 사용하려면 버킷 이름과 일치하도록 예제 버킷 이름(*example-bucket*)을 업데이트합니다. 버킷 정책에 대한 자세한 내용은 버킷 정책 및 사용자 정책 사용 (p. 338)를 참조하십시오.

```
{
  "Version":"2012-10-17",
  "Statement":[{
 "Sid":"PublicReadGetObject",
        "Effect":"Allow",
   "Principal": "*",
      "Action":["s3:GetObject"],
      "Resource":["arn:aws:s3:::example-bucket/*"
      ]
    }
  ]
}
```

S3 퍼블릭 액세스 차단 설정을 편집한 후에는 버킷 정책을 추가하여 버킷에 퍼블릭 읽기 액세스 권한을 부여할 수 있습니다. 퍼블릭 읽기 액세스 권한을 부여하면 인터넷의 모든 사용자가 버킷에 액세스할 수 있습니다.

> **Important**
>
> 다음 정책은 하나의 예일 뿐이며 버킷의 콘텐츠에 대한 전체 액세스를 허용합니다. 이 단계를 진행하기 전에 Amazon S3 버킷의 파일을 보호하려면 어떻게 해야 합니까?를 검토하여 S3 버킷의 파일 보안을 위한 모범 사례 및 퍼블릭 액세스 권한 부여와 관련된 위험을 파악할 수 있습니다.

1. 버킷에서 버킷의 이름을 선택합니다.
2. Permissions를 선택합니다.
3. [Bucket Policy]를 선택합니다.
4. 웹 사이트에 대한 퍼블릭 읽기 액세스 권한을 부여하려면 다음 버킷 정책을 복사한 후 버킷 정책 편집기에 붙여 넣습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::example.com/*"
            ]
        }
    ]
}
```

5. `Resource`를 업데이트하여 버킷 이름을 포함합니다.

   앞의 예제 버킷 정책에서 *`example.com`*은 버킷 이름입니다. 자체 버킷에 이 버킷 정책을 사용하려면 자체 버킷 이름과 일치하도록 이 이름을 업데이트해야 합니다.
6. 저장을 선택합니다.

   버킷에 퍼블릭 액세스 권한이 있음을 나타내는 경고가 나타납니다. 버킷 정책에 퍼블릭 레이블이 나타납니다.

   `Policy has invalid resource`라는 오류가 표시되면 버킷 정책의 버킷 이름이 사용자의 버킷 이름과 일치하는지 확인합니다. 버킷 정책 추가에 대한 자세한 내용은 S3 버킷 정책을 추가하려면 어떻게 해야 합니까?를 참조하십시오.

   오류 - 액세스 거부됨 경고가 표시되고 버킷 정책 편집기에서 버킷 정책을 저장할 수 없는 경우 계정 수준 및 버킷 수준 퍼블릭 액세스 차단 설정을 확인하여 버킷에 대한 퍼블릭 액세스를 허용하는지 확인하십시오. 웹 사이트 권한에 대한 자세한 내용은 웹 사이트 액세스에 필요한 권한을 참조하십시오.

객체 ACL(액세스 제어 목록)

버킷 정책을 사용하여 객체에 퍼블릭 읽기 권한을 부여할 수 있습니다. 그러나 버킷 정책은 버킷 소유자가 소유한 객체에만 적용됩니다. 버킷에 버킷 소유자가 소유하지 않은 객체가 포함된 경우 버킷 소유자는 객체 ACL(액세스 제어 목록)을 사용하여 해당 객체에 대한 퍼블릭 읽기 권한을 부여해야 합니다.

ACL을 사용하여 객체를 공개적으로 읽기 가능하게 만들려면 다음 부여 요소에 나와 있는 대로 `AllUsers` 그룹에 읽기 권한을 부여합니다. 객체 ACL에 이 부여 요소를 추가합니다. ACL 관리에 관한 자세한 내용은 ACL을 사용한 액세스 관리 (p. 437) 단원을 참조하십시오.

```
<Grant>
  <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:type="Group">
    <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
  </Grantee>
  <Permission>READ</Permission>
</Grant>
```

(선택 사항) 웹 트래픽 로깅 구성

웹 사이트에 액세스하는 방문자의 수를 추적하려는 경우 필요에 따라 로깅을 활성화할 수 있습니다. 정적 웹 사이트로 구성한 버킷에 대한 서버 액세스 로깅을 활성화합니다. 자세한 내용은 Amazon S3 서버 액세스 로깅을 참조하십시오. 그러나 웹 사이트 속도를 높이기 위해 Amazon CloudFront를 사용하려는 경우 CloudFront 로깅을 사용할 수도 있습니다.

다음 단계에서는 서버 액세스 로깅 및 CloudFront 로깅을 위한 로그 전송 버킷을 설정하고 서버 액세스 로깅을 활성화하는 방법을 보여줍니다. CloudFront 로깅 구성에 대한 자세한 내용은 예제: Amazon CloudFront로 웹 사이트 속도 향상을 참조하십시오.

정적 웹 사이트 버킷에 대한 로깅을 활성화하려면

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 정적 웹 사이트로 구성된 버킷을 생성한 리전에서 로깅용 버킷을 생성합니다(예: `logs.example.com`).
3. 서버 액세스 로깅 로그 파일에 대한 폴더를 생성합니다(예: `logs`).
4. (선택 사항) CloudFront를 사용하여 웹 사이트 성능을 개선하려면 CloudFront 로그 파일에 대한 폴더(예: `cdn`)를 생성합니다.
5. 버킷 목록에서 버킷을 선택합니다.
6. [Properties]를 선택합니다.

7. Server access logging(서버 액세스 로깅)을 선택합니다.
8. 로깅 활성화를 선택합니다.
9. 대상 버킷에서 로그 파일에 대해 생성한 버킷(예: `logs-example-bucket`)을 선택합니다.
10. 대상 접두사에서 로그 파일에 대해 생성한 폴더의 이름을 구분 기호(/)와 함께 입력합니다(예: **`logs/`**).

    대상 접두사를 설정할 때 쉽게 찾을 수 있도록 로그 데이터 파일을 폴더로 그룹화합니다.
11. 저장을 선택합니다.

    이제 로그 버킷에서 로그에 액세스할 수 있습니다. Amazon S3는 2시간마다 웹 사이트 액세스 로그를 로그 버킷에 기록합니다.
12. 로그를 보려면 개요를 선택하고 폴더를 선택합니다.

(선택 사항)사용자 지정 오류 문서 구성

버킷을 정적 웹 사이트로 구성한 후 오류가 발생하면 Amazon S3에서 HTML 오류 문서를 반환합니다. 선택적으로 사용자 지정 오류 문서를 사용하여 버킷을 구성하고 오류가 발생하면 Amazon S3에서 해당 문서를 반환하도록 할 수 있습니다.

> **Note**
>
> 일부 브라우저는 오류가 발생하는 경우 자체 오류 메시지를 표시하므로 Amazon S3가 반환하는 오류 문서가 무시됩니다. 예를 들어 HTTP 404 Not Found(HTTP 404 찾을 수 없음) 오류가 발생할 때 Google Chrome은 Amazon S3이 반환하는 오류 문서를 무시하고 자체 오류를 표시할 수도 있습니다.

주제

- Amazon S3 HTTP 응답 코드 (p. 557)
- 사용자 지정 오류 문서 구성 (p. 558)

Amazon S3 HTTP 응답 코드

다음 표는 오류 발생 시 Amazon S3이 반환하는 HTTP 응답 코드의 하위 집합 목록입니다.

HTTP 오류 코드	설명
301 Moved Permanently(301 영구 이동됨)	사용자가 Amazon S3 웹 사이트 엔드포인트(`http://s3-website.`*`Region`*`.amazonaws.com/`)에 곧바로 요청을 보내는 경우 Amazon S3는 301 Moved Permanently(301 영구 이동됨) 응답을 반환하고 해당 요청을 `https://aws.amazon.com/s3/`로 리디렉션합니다.
302 Found(302 찾음)	`http://`*`bucket-name`*`.s3-website.`*`Region`*`.amazonaws.com/x`의 `x` 키에 대한 요청이 후행 슬래시 없이 Amazon S3에 수신되는 경우, 키 이름이 `x`인 객체가 첫 검색 대상이 됩니다. 객체를 찾을 수 없는 경우 Amazon S3은 해당 요청이 하위 폴더 `x`에 대한 것으로 판단하므로 맨 뒤에 슬래시를 추가하여 요청을 리디렉션하고 302 Found(302 찾음)를 반환합니다.
304 Not Modified(304 수정되지 않음)	Amazon S3 사용자는 헤더 `If-Modified-Since`, `If-Unmodified-Since`, `If-Match` 및/또는 `If-None-Match`를 요청하여 클라이언트가 보유하는 캐시된 사본과 요청된 객체가 동일한지 확인합니다. 객체가 동일한 경우 웹 사이트 엔드포인트가 304 Not Modified(304 수정되지 않음) 응답을 반환합니다.
400 Malformed Request(400 형식이 잘못된 요청)	잘못된 리전 엔드포인트를 통해 사용자가 버킷에 액세스하려는 경우 웹 사이트 엔드포인트가 400 Malformed Request(400 형식이 잘못된 요청)로 응답합니다.

HTTP 오류 코드	설명
[403 Forbidden]	사용자 요청이 공개적으로 읽기 가능한 객체로 변환되는 경우 웹 사이트 엔드포인트가 403 Forbidden(403 금지됨)으로 응답합니다. 객체 소유자는 버킷 정책이나 ACL을 사용하여 객체를 공개적으로 읽기 가능하도록 설정해야 합니다.
404 Not Found(404 찾을 수 없음)	웹 사이트 엔드포인트가 404 Not Found(404 찾을 수 없음)로 응답하는 이유는 다음과 같습니다. • Amazon S3가 웹 사이트 URL이 존재하지 않는 객체 키를 참조한다고 판단함. • Amazon S3이 존재하지 않는 인덱스 문서에 대한 요청이라고 유추합니다. • URL에 지정된 버킷이 존재하지 않음. • URL에 지정된 버킷이 존재하지만 웹 사이트로 구성되지 않음. 404 Not Found(404 찾을 수 없음)에 대해 반환되는 사용자 지정 문서를 만들 수 있습니다. 반드시 웹 사이트처럼 구성된 버킷에 문서를 업로드하고 해당 문서를 사용하는 것으로 웹 사이트 호스팅을 구성해야 합니다. Amazon S3이 URL을 객체나 인덱스 문서에 대한 요청으로 해석하는 작동 방법에 대한 정보는 인덱스 문서 구성 (p. 551) 단원을 참조하십시오.
500 Service Error(500 서비스 오류)	내부 서버 오류가 발생하는 경우 웹 사이트 엔드포인트가 500 Service Error(500 서비스 오류)로 응답합니다.
[503 Service Unavailable]	사용자가 요청 속도를 줄여야 한다고 Amazon S3이 판단하는 경우 웹 사이트 엔드포인트가 503 Service Unavailable(503 서비스 사용 불가)로 응답합니다.

이러한 각 오류에 대해 Amazon S3는 사전 정의된 HTML 메시지를 반환합니다. 다음은 403 Forbidden(403 금지됨) 요청에 대해 반환된 HTML 메시지 예입니다.

사용자 지정 오류 문서 구성

버킷을 정적 웹 사이트로 구성할 때 사용자에게 친숙한 오류 메시지와 추가 도움말이 포함된 사용자 지정 오류 문서를 선택적으로 제공할 수 있습니다. Amazon S3는 HTTP 4XX 클래스 오류 코드에 대해서만 사용자 지정 오류 문서를 반환합니다.

사용자 지정 오류 문서를 구성하려면

1. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷 목록에서 버킷 이름을 선택합니다.

3. [Properties]를 선택합니다.
4. [Static website hosting]을 선택합니다.

   버킷이 이미 정적 웹 사이트로 구성되어 있는 경우 다음 단계에 따라 오류 문서 정보를 업데이트하거나 추가할 수 있습니다. 버킷을 정적 웹 사이트로 구성하지 않은 경우 먼저 필요한 구성을 설정해야 합니다. 자세한 내용은 웹 사이트 호스팅 사용 (p. 551) 단원을 참조하십시오.
5. 오류 문서 상자에 오류 문서 이름을 입력합니다.
6. 저장을 선택합니다.

REST API를 사용하여 사용자 지정 오류 문서가 있는 정적 웹 사이트로 버킷을 구성하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service API Reference의 PutBucketWebsite를 참조하십시오.

(선택 사항) 웹 페이지 리디렉션 구성

Amazon S3 버킷이 웹 사이트 호스팅용으로 구성된 경우 웹 페이지 리디렉션을 구성할 수 있습니다. 리디렉션 구성에 대해 다음과 같은 옵션이 있습니다.

주제

- Amazon S3 콘솔에서 페이지 리디렉션 설정 (p. 559)
- REST API를 통한 페이지 리디렉션 설정 (p. 560)
- 버킷의 웹 사이트 엔드포인트에 대한 요청을 다른 호스트로 리디렉션 (p. 560)
- 고급 조건부 리디렉션 구성 (p. 561)

Amazon S3 콘솔에서 페이지 리디렉션 설정

객체의 메타데이터에 웹 사이트 리디렉션 위치를 설정하여 객체에 대한 요청을 다른 객체 또는 URL로 리디렉션할 수 있습니다. 객체 메타데이터에 `x-amz-website-redirect-location` 속성을 추가하여 리디렉션을 설정할 수 있습니다. Amazon S3 콘솔에서 객체의 메타데이터에 웹 사이트 리디렉션 위치를 설정합니다. Amazon S3 API (p. 560)를 사용하는 경우 `x-amz-website-redirect-location`을 설정합니다. 이 경우 웹 사이트가 해당 객체를 301 리디렉션으로 해석합니다.

다른 객체로 요청을 리디렉션하려면 대상 객체의 키에 이르도록 리디렉션 위치를 설정합니다. 외부 URL로 요청을 리디렉션하려면 리디렉션 위치를 원하는 URL로 설정합니다. 객체 메타데이터에 대한 자세한 정보는 시스템 정의 객체 메타데이터 (p. 106) 단원을 참조하십시오.

페이지 리디렉션을 설정하는 경우 원본 객체 콘텐츠를 유지하거나 삭제할 수 있습니다. 예를 들어 버킷에 `page1.html` 객체가 있는 경우 이 페이지에 대한 모든 요청을 다른 객체 `page2.html`로 리디렉션할 수 있습니다. 여기에는 두 가지 옵션이 있습니다.

- `page1.html` 객체의 내용을 유지하고 페이지 요청을 리디렉션합니다.
- `page1.html`의 내용을 삭제하고 `page1.html`라는 0바이트 객체를 업로드하여 기존 객체를 대체하고 페이지 요청을 리디렉션합니다.

객체에 대한 요청을 리디렉션하려면

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 정적 웹 사이트로 구성한 버킷의 이름을 선택합니다(예: `example.com`).
3. 버킷의 개요 탭에서 리디렉션을 생성할 객체를 선택합니다.
4. [Properties]를 선택합니다.
5. 메타데이터를 선택합니다.
6. + 메타데이터 추가를 선택합니다.
7. 키에서 Website-Redirect-Location을 선택합니다.

8. 값에 리디렉션할 객체의 키 이름(예: `/page2.html`)을 입력합니다.

   같은 버킷에 있는 다른 객체의 경우 값의 `/` 접두사가 필요합니다. 또한 이 값을 외부 URL에 설정할 수 있습니다(예: `http://www.example.com`).
9. 저장을 선택합니다.

REST API를 통한 페이지 리디렉션 설정

다음 Amazon S3 API 작업은 요청의 `x-amz-website-redirect-location` 헤더를 지원합니다. Amazon S3는 객체 메타데이터의 헤더 값을 `x-amz-website-redirect-location`으로 저장합니다.

- PUT Object
- 멀티파트 업로드 시작
- POST Object
- PUT Object - Copy

웹 사이트 호스팅용으로 구성된 버킷은 웹 사이트 엔드포인트와 REST 엔드포인트를 모두 포함합니다. 301 리디렉션으로 구성된 페이지에 대한 요청은 요청의 엔드포인트에 따라 요청 결과가 다음과 같을 수 있습니다.

- 리전별 웹 사이트 엔드포인트 – Amazon S3가 `x-amz-website-redirect-location` 속성의 값에 따라 페이지 요청을 리디렉션합니다.
- REST 엔드포인트 – Amazon S3가 페이지 요청을 리디렉션하지 않습니다. 이 경우 요청된 객체가 반환됩니다.

엔드포인트에 대한 자세한 내용은 웹 사이트 엔드포인트와 REST API 엔드포인트 간의 주요 차이점 (p. 550) 단원을 참조하십시오.

페이지 리디렉션을 설정하는 경우 객체 콘텐츠를 유지하거나 삭제할 수 있습니다. 예를 들면, 버킷에 `page1.html` 객체가 있다고 가정해 보십시오.

- `page1.html`의 콘텐츠를 유지하고 페이지 요청만 리디렉션하려는 경우 PUT Object - Copy 요청을 제출하면 `page1.html` 객체를 원본으로 사용하는 새 `page1.html` 객체를 만들 수 있습니다. 해당 요청에는 `x-amz-website-redirect-location` 헤더를 설정합니다. 요청이 완료되면 콘텐츠가 변경되지 않은 원래 페이지가 생기지만 사용자가 지정한 리디렉션 위치로 Amazon S3이 해당 페이지에 대한 모든 요청을 리디렉션합니다.
- `page1.html` 객체의 콘텐츠를 삭제하고 해당 페이지에 대한 요청을 리디렉션하려는 경우 PUT Object 요청을 보내면 객체 키가 같은 0바이트 객체인 `page1.html`을 업로드할 수 있습니다. PUT 요청에서 새 객체에 `x-amz-website-redirect-location`의 `page1.html`을 설정합니다. 요청이 완료되면 `page1.html`은 콘텐츠가 없어지고 `x-amz-website-redirect-location`에 따라 지정된 위치로 요청이 리디렉션됩니다.

GET Object 작업을 사용하여 다른 객체 메타데이터와 함께 객체를 검색하는 경우 Amazon S3이 해당 응답의 `x-amz-website-redirect-location` 헤더를 반환합니다.

버킷의 웹 사이트 엔드포인트에 대한 요청을 다른 호스트로 리디렉션

버킷의 웹 사이트 엔드포인트에 대한 모든 요청을 다른 호스트로 리디렉션할 수 있습니다. 모든 요청을 리디렉션하면 웹 사이트 엔드포인트로 전송된 요청이 지정된 호스트 이름으로 리디렉션됩니다.

예를 들어, 루트 도메인이 `example.com`이고 `http://example.com` 및 `http://www.example.com`에 대한 요청을 모두 처리할 경우, `example.com` 및 `www.example.com`라는 이름의 버킷 2개를 만들 수 있습

니다. 그런 다음 `example.com` 버킷의 콘텐츠는 유지하고 다른 `www.example.com` 버킷에서 모든 요청을 `example.com` 버킷으로 리디렉션하도록 구성합니다. 자세한 내용은 사용자 지정 도메인 이름을 사용하여 정적 웹 사이트 구성을 참조하십시오.

버킷 웹 사이트 엔드포인트에 대한 요청을 리디렉션하려면

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 정적 웹 사이트로 구성한 버킷의 이름을 선택합니다(예: `example.com`).
3. [Properties]를 선택합니다.
4. [Static website hosting]을 선택합니다.
5. [Redirect requests]를 선택합니다.
6. 대상 버킷 또는 도메인 상자에 리디렉션할 버킷이나 도메인을 입력합니다.

   예를 들어, 루트 도메인 주소로 리디렉션하려면 **`example.com`**을 입력해야 합니다.
7. 프로토콜 상자에 리디렉션된 요청에 대한 프로토콜(**`http`** 또는 **`https`**)을 입력합니다.

   프로토콜을 지정하지 않으면 원래 요청의 프로토콜이 사용됩니다.
8. Save를 선택합니다.

고급 조건부 리디렉션 구성

어드밴스 리디렉션 규칙을 사용하면 특정 객체 키 이름, 요청 접두사 또는 응답 코드에 따라 조건부로 요청을 라우팅할 수 있습니다. 예를 들어, 버킷 내 객체를 삭제하거나 이름을 변경하는 경우를 가정해봅시다. 요청을 다른 객체에 리디렉션하는 라우팅 규칙을 추가할 수 있습니다. 폴더를 사용할 수 없게 하려면 다른 웹 페이지에 요청을 리디렉션하도록 라우팅 규칙을 추가하면 됩니다. 또한 오류를 반환하는 요청을 오류 처리 시 다른 도메인으로 라우팅하여 오류 조건을 처리하도록 라우팅 규칙을 추가할 수도 있습니다.

웹 사이트 호스팅용 버킷을 구성할 때 고급 리디렉션 규칙을 지정할 수 있습니다. Amazon S3에는 웹 사이트 구성당 50개의 라우팅 규칙 제한이 있습니다. 50개 이상의 라우팅 규칙이 필요한 경우 객체 리디렉션을 사용할 수 있습니다. 자세한 내용은 (선택 사항) 웹 페이지 리디렉션 구성 (p. 559) 단원을 참조하십시오.

정적 웹 사이트에 대한 리디렉션 규칙을 구성하려면

정적 웹 사이트 호스팅이 이미 활성화되어 있는 버킷에 대한 리디렉션 규칙을 추가하려면 다음 단계를 따르십시오.

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷 목록에서 정적 웹 사이트로 구성한 버킷의 이름을 선택합니다.
3. [Properties]를 선택합니다.
4. [Static website hosting]을 선택합니다.
5. 리디렉션 규칙에 리디렉션 규칙을 입력합니다.

   규칙 설명 시 XML을 사용합니다. 리디렉션 규칙 지정에 대한 일반 구문 및 예는 라우팅 규칙 지정 구문 (p. 561) 단원을 참조하십시오. Amazon S3에는 웹 사이트 구성당 50개의 라우팅 규칙의 제한이 있습니다. 50개 이상의 라우팅 규칙이 필요한 경우 객체 리디렉션을 사용할 수 있습니다. 자세한 내용은 Amazon S3 콘솔에서 페이지 리디렉션 설정 (p. 559) 단원을 참조하십시오.
6. Save를 선택합니다.

라우팅 규칙 지정 구문

웹 사이트 구성의 라우팅 규칙을 정의하기 위한 일반 구문은 다음과 같습니다.

```
<RoutingRules> =
```

```
    <RoutingRules>
         <RoutingRule>...</RoutingRule>
         [<RoutingRule>...</RoutingRule>
         ...]
    </RoutingRules>

<RoutingRule> =
   <RoutingRule>
      [ <Condition>...</Condition> ]
      <Redirect>...</Redirect>
   </RoutingRule>

<Condition> =
   <Condition>
      [ <KeyPrefixEquals>...</KeyPrefixEquals> ]
      [ <HttpErrorCodeReturnedEquals>...</HttpErrorCodeReturnedEquals> ]
   </Condition>
    Note: <Condition> must have at least one child element.

<Redirect> =
   <Redirect>
      [ <HostName>...</HostName> ]
      [ <Protocol>...</Protocol> ]
      [ <ReplaceKeyPrefixWith>...</ReplaceKeyPrefixWith>  ]
      [ <ReplaceKeyWith>...</ReplaceKeyWith> ]
      [ <HttpRedirectCode>...</HttpRedirectCode> ]
   </Redirect>
    Note: <Redirect> must have at least one child element.
           Also, you can have either ReplaceKeyPrefix with or ReplaceKeyWith,
           but not both.
```

다음 표에서는 라우팅 규칙 요소를 설명합니다.

이름	설명
`RoutingRules`	`RoutingRule` 요소 모음용 컨테이너입니다.
`RoutingRule`	조건 및 조건이 충족되었을 때 적용되는 리디렉션을 식별하는 규칙입니다. 조건: • `RoutingRules` 컨테이너에는 적어도 1개 이상의 라우팅 규칙이 포함되어 있어야 합니다.
`Condition`	지정된 리디렉션이 적용되려면 충족되어야 할 조건을 설명하기 위한 컨테이너입니다. 라우팅 규칙이 조건을 포함하지 않는 경우 해당 규칙은 모든 요청에 적용됩니다.
`KeyPrefixEquals`	리디렉션된 요청을 보내는 객체 키 이름 접두사입니다. `KeyPrefixEquals` `HttpErrorCodeReturnedEquals`를 지정하지 않을 경우 가 필요합니다. `KeyPrefixEquals` 및 `HttpErrorCodeReturnedEquals`가 모두 지정되는 경우 두 항목 모두 true로 설정되어야 조건이 충족됩니다.
`HttpErrorCodeReturnedEquals`	리디렉션 적용 조건과 일치하는 HTTP 오류 코드입니다. 오류가 발생하고 오류 코드가 이 값에 해당하는 경우, 지정된 리디렉션이 적용됩니다. `HttpErrorCodeReturnedEquals` `KeyPrefixEquals`를 지정하지 않을 경우 가 필요합니다. `KeyPrefixEquals` 및

이름	설명
	`HttpErrorCodeReturnedEquals`가 모두 지정되는 경우 두 항목 모두 true로 설정되어야 조건이 충족됩니다.
`Redirect`	요청에 대한 리디렉션 지침을 제공하는 컨테이너 요소입니다. 다른 호스트나 다른 페이지로 요청을 리디렉션할 수 있으며, 사용할 다른 프로토콜을 지정할 수 있습니다. `RoutingRule`에는 `Redirect` 요소가 있어야 합니다. `Redirect` 요소는 `Protocol`, `HostName`, `ReplaceKeyPrefixWith`, `ReplaceKeyWith` 또는 `HttpRedirectCode` 중 한 개 이상의 형제 요소를 포함해야 합니다.
`Protocol`	응답에서 반환된 `Location` 헤더에 사용할 `http` 또는 `https` 프로토콜입니다. 해당 형제 요소 중 하나가 제공되는 경우 `Protocol`는 필요하지 않습니다.
`HostName`	응답에서 반환된 `Location` 헤더에 사용할 호스트 이름입니다. 해당 형제 요소 중 하나가 제공되는 경우 `HostName`는 필요하지 않습니다.
`ReplaceKeyPrefixWith`	리디렉션 요청의 `KeyPrefixEquals` 값을 대체하는 객체 키 이름의 접두사입니다. 해당 형제 요소 중 하나가 제공되는 경우 `ReplaceKeyPrefixWith`는 필요하지 않습니다. `ReplaceKeyWith`가 제공되지 않는 경우에만 제공 가능한 파라미터입니다.
`ReplaceKeyWith`	응답에서 반환된 `Location` 헤더에 사용할 객체 키입니다. 해당 형제 요소 중 하나가 제공되는 경우 `ReplaceKeyWith`는 필요하지 않습니다. `ReplaceKeyPrefixWith`가 제공되지 않는 경우에만 제공 가능한 파라미터입니다.
`HttpRedirectCode`	응답에서 반환된 `Location` 헤더에 사용할 HTTP 리디렉션 코드입니다. 해당 형제 요소 중 하나가 제공되는 경우 `HttpRedirectCode`는 필요하지 않습니다.

예

다음 예제는 다음과 같은 일반적인 리디렉션 작업을 설명합니다.

Example 1: 키 접두사 이름을 바꾼 후 리디렉션

버킷에 다음과 같은 객체가 포함되어 있다고 가정해 보십시오.

- index.html
- docs/article1.html
- docs/article2.html

해당 폴더의 이름을 `docs/`에서 `documents/`로 바꾸기로 합니다. 이렇게 이름을 변경한 후에는 접두사에 대한 요청을 `docs/`에서 `documents/`로 리디렉션해야 합니다. 예를 들어, `docs/article1.html`에 대한 요청은 `documents/article1.html`로 리디렉션됩니다.

이 경우 웹 사이트 구성에 다음 라우팅 규칙을 추가합니다.

```
<RoutingRules>
  <RoutingRule>
  <Condition>
    <KeyPrefixEquals>docs/</KeyPrefixEquals>
  </Condition>
  <Redirect>
    <ReplaceKeyPrefixWith>documents/</ReplaceKeyPrefixWith>
  </Redirect>
  </RoutingRule>
</RoutingRules>
```

Example 2: 삭제된 폴더에 대한 요청을 페이지로 리디렉션

`images/` 폴더를 삭제한다고 가정해 보십시오(즉, 키 접두사가 `images/`인 모든 객체를 삭제). 키 접두사가 `images/`인 객체에 대한 요청을 `folderdeleted.html`라는 이름의 페이지로 리디렉션하는 라우팅 규칙을 추가할 수 있습니다.

```
<RoutingRules>
  <RoutingRule>
  <Condition>
     <KeyPrefixEquals>images/</KeyPrefixEquals>
  </Condition>
  <Redirect>
    <ReplaceKeyWith>folderdeleted.html</ReplaceKeyWith>
  </Redirect>
  </RoutingRule>
</RoutingRules>
```

Example 3: HTTP 오류에 대한 리디렉션

요청된 객체를 찾을 수 없는 경우 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스로 요청을 리디렉션한다고 가정해 보십시오. HTTP 상태 코드 404 (Not Found)(404(찾을 수 없음))가 반환되는 경우 해당 요청을 처리하는 Amazon EC2 인스턴스로 사이트 방문자가 리디렉션되도록 리디렉션 규칙을 추가합니다.

또한, 다음 예제에서는 리디렉션에서 객체 키 접두사 `report-404/`의 삽입을 설명합니다. 예를 들어, `ExamplePage.html` 페이지를 요청하여 HTTP 404 오류가 반환되는 경우, 해당 요청은 지정된 Amazon EC2 인스턴스의 `report-404/ExamplePage.html` 페이지로 리디렉션됩니다. 라우팅 규칙이 하나도 없고 HTTP 오류 404가 발생하는 경우, 구성에 지정된 오류 문서가 반환됩니다.

```
<RoutingRules>
  <RoutingRule>
  <Condition>
    <HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals >
  </Condition>
  <Redirect>
    <HostName>ec2-11-22-333-44.compute-1.amazonaws.com</HostName>
    <ReplaceKeyPrefixWith>report-404/</ReplaceKeyPrefixWith>
  </Redirect>
  </RoutingRule>
</RoutingRules>
```

프로그래밍 방식으로 버킷을 정적 웹 사이트로 구성

Amazon S3에서 정적 웹 사이트를 호스팅하려면 Amazon S3 버킷을 웹 사이트 호스팅용으로 구성한 후 웹 사이트 콘텐츠를 버킷에 업로드합니다. 또한 AWS SDK를 사용하여 프로그래밍 방식으로 웹 사이트 구성을 생성, 업데이트 및 삭제할 수 있습니다. SDK는 Amazon S3 REST API를 둘러싼 래퍼 클래스를 제공합니다. 애플리케이션에서 필요한 경우, 해당 애플리케이션에서 직접 REST API 요청할 수 있습니다.

AWS Management 콘솔을 사용하여 정적 웹 사이트 호스팅용으로 버킷을 구성하는 방법에 대한 자세한 내용은 AWS Management 콘솔을 사용하여 버킷을 정적 웹 사이트로 구성 (p. 550) 단원을 참조하십시오.

AWS CLI를 사용하여 S3 버킷을 정적 웹 사이트로 구성하는 방법에 대한 자세한 내용은 AWS CLI Command Reference의 웹 사이트를 참조하십시오. 프로그래밍 방식으로 S3 버킷을 정적 웹 사이트로 구성하는 방법에 대한 자세한 내용은 다음 주제를 참조하십시오.

주제

- AWS SDK for Java로 웹 사이트 관리 (p. 565)
- .NET용 AWS SDK로 웹 사이트 관리 (p. 566)
- PHP용 AWS SDK로 웹 사이트 관리 (p. 567)
- REST API로 웹 사이트 관리 (p. 568)

AWS SDK for Java로 웹 사이트 관리

다음 예제는 AWS SDK for Java를 사용하여 버킷에 대한 웹 사이트 구성을 관리하는 방법을 보여줍니다. 버킷에 웹 사이트 구성을 추가하려면 버킷 이름 및 웹 사이트 구성을 제공해야 합니다. 웹 사이트 구성에는 인덱스 문서가 포함되어 있어야 하고 선택적 오류 문서를 포함할 수 있습니다. 이러한 문서는 버킷에 이미 있어야 합니다. 자세한 내용은 PUT 버킷 웹 사이트를 참조하십시오. Amazon S3 웹 사이트 기능에 대한 자세한 내용은 Amazon S3 정적 웹 사이트 호스팅 (p. 548)을 참조하십시오.

Example

다음 예제에서는 AWS SDK for Java를 사용하여 버킷에 웹 사이트 구성을 추가하고, 구성을 검색 및 인쇄하고, 구성을 삭제하고, 삭제를 확인합니다. 실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
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;
import com.amazonaws.services.s3.model.BucketWebsiteConfiguration;

import java.io.IOException;

public class WebsiteConfiguration {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String indexDocName = "*** Index document name ***";
        String errorDocName = "*** Error document name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();

            // Print the existing website configuration, if it exists.
            printWebsiteConfig(s3Client, bucketName);

            // Set the new website configuration.
            s3Client.setBucketWebsiteConfiguration(bucketName, new
 BucketWebsiteConfiguration(indexDocName, errorDocName));

            // Verify that the configuration was set properly by printing it.
```

```
            printWebsiteConfig(s3Client, bucketName);

            // Delete the website configuration.
            s3Client.deleteBucketWebsiteConfiguration(bucketName);

            // Verify that the website configuration was deleted by printing it.
            printWebsiteConfig(s3Client, 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();
        }
    }

    private static void printWebsiteConfig(AmazonS3 s3Client, String bucketName) {
        System.out.println("Website configuration: ");
        BucketWebsiteConfiguration bucketWebsiteConfig =
 s3Client.getBucketWebsiteConfiguration(bucketName);
        if (bucketWebsiteConfig == null) {
            System.out.println("No website config.");
        } else {
            System.out.println("Index doc: " +
 bucketWebsiteConfig.getIndexDocumentSuffix());
            System.out.println("Error doc: " + bucketWebsiteConfig.getErrorDocument());
        }
    }
}
```

.NET용 AWS SDK로 웹 사이트 관리

다음 예제는 .NET용 AWS SDK를 사용하여 버킷에 대한 웹 사이트 구성을 관리하는 방법을 보여줍니다. 버킷에 웹 사이트 구성을 추가하려면 버킷 이름 및 웹 사이트 구성을 제공해야 합니다. 웹 사이트 구성에는 인덱스 문서가 포함되어 있어야 하고 선택적 오류 문서를 포함할 수 있습니다. 이러한 문서는 버킷에 저장되어 있어야 합니다. 자세한 내용은 PUT 버킷 웹 사이트를 참조하십시오. Amazon S3 웹 사이트 기능에 대한 자세한 내용은 Amazon S3 정적 웹 사이트 호스팅 (p. 548)을 참조하십시오.

Example

다음 C# 코드로 지정된 버킷에 웹 사이트 구성을 추가하는 예를 살펴봅니다. 이 구성은 인덱스 문서와 오류 문서 이름을 모두 지정합니다. 실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

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

namespace Amazon.DocSamples.S3
{
    class WebsiteConfigTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string indexDocumentSuffix = "*** index object key ***"; // For
 example, index.html.
        private const string errorDocument = "*** error object key ***"; // For example,
 error.html.
        // Specify your bucket region (an example region is shown).
```

```
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 client;
        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            AddWebsiteConfigurationAsync(bucketName, indexDocumentSuffix,
errorDocument).Wait();
        }

        static async Task AddWebsiteConfigurationAsync(string bucketName,
                                        string indexDocumentSuffix,
                                        string errorDocument)
        {
            try
            {
                // 1. Put the website configuration.
                PutBucketWebsiteRequest putRequest = new PutBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                    WebsiteConfiguration = new WebsiteConfiguration()
                    {
                        IndexDocumentSuffix = indexDocumentSuffix,
                        ErrorDocument = errorDocument
                    }
                };
                PutBucketWebsiteResponse response = await
client.PutBucketWebsiteAsync(putRequest);

                // 2. Get the website configuration.
                GetBucketWebsiteRequest getRequest = new GetBucketWebsiteRequest()
                {
                    BucketName = bucketName
                };
                GetBucketWebsiteResponse getResponse = await
client.GetBucketWebsiteAsync(getRequest);
                Console.WriteLine("Index document: {0}",
getResponse.WebsiteConfiguration.IndexDocumentSuffix);
                Console.WriteLine("Error document: {0}",
getResponse.WebsiteConfiguration.ErrorDocument);
            }
            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);
            }
        }
    }
}
```

PHP용 AWS SDK로 웹 사이트 관리

이 주제에서는 PHP용 AWS SDK의 클래스를 사용하여 웹 사이트 호스팅용 Amazon S3 버킷을 구성하고 관리하는 방법을 살펴봅니다. 이미 PHP용 AWS SDK 사용 및 PHP 예제 실행 (p. 736)의 지침에 따라 PHP용 AWS SDK가 올바르게 설치되어 있다고 가정합니다. Amazon S3 웹 사이트 기능에 대한 자세한 내용은 Amazon S3 정적 웹 사이트 호스팅 (p. 548)을 참조하십시오.

다음 PHP로 지정된 버킷에 웹 사이트 구성을 추가하는 예를 살펴봅니다. `create_website_config` 메서드는 인덱스 문서와 오류 문서 이름을 명시적으로 제공합니다. 또한, 샘플이 웹 사이트 구성을 검색해

응답을 인쇄합니다. Amazon S3 웹 사이트 기능에 대한 자세한 내용은 Amazon S3 정적 웹 사이트 호스팅 (p. 548)을 참조하십시오.

실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 PHP용 AWS SDK 사용 및 PHP 예제 실행 (p. 736) 단원을 참조하십시오.

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

use Aws\S3\S3Client;

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

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


// Add the website configuration.
$s3->putBucketWebsite([
    'Bucket'                 => $bucket,
    'WebsiteConfiguration'   => [
        'IndexDocument' => ['Suffix' => 'index.html'],
        'ErrorDocument' => ['Key' => 'error.html']
    ]
]);

// Retrieve the website configuration.
$result = $s3->getBucketWebsite([
    'Bucket' => $bucket
]);
echo $result->getPath('IndexDocument/Suffix');

// Delete the website configuration.
$s3->deleteBucketWebsite([
    'Bucket' => $bucket
]);
```

관련 리소스

- Amazon S3용 PHP용 AWS SDK Aws\S3\S3Client 클래스
- PHP용 AWS SDK 설명서

REST API로 웹 사이트 관리

AWS Management 콘솔(이)나 AWS SDK를 사용해 버킷을 웹 사이트로 구성할 수 있습니다. 하지만 애플리케이션에서 요구할 경우 REST 요청을 직접 전송할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API Reference의 다음 단원을 참조하십시오.

- PUT 버킷 웹 사이트
- GET 버킷 웹 사이트
- DELETE 버킷 웹 사이트

연습 예제 - Amazon S3 웹 사이트 호스팅

이 단원에서 두 개의 예를 들겠습니다. 첫 번째 예에서는 웹 사이트 호스팅용 버킷을 구성하고 샘플 인덱스 문서를 업로드하고 해당 버킷의 Amazon S3 웹 사이트 엔드포인트를 사용하여 웹 사이트를 테스트합니다.

두 번째 예에서는 Amazon S3 버킷 웹 사이트 엔드포인트 대신에 `example.com`과 같은 자체 도메인을 사용하고 웹 사이트로 구성된 Amazon S3 버킷의 콘텐츠를 제공하는 방법을 보여줍니다. 또한, 이 예에서는 Amazon S3을 통한 루트 도메인 지원 방법을 알아봅니다.

주제

- 정적 웹 사이트 구성 (p. 569)
- Route 53에 등록된 사용자 지정 도메인을 사용하여 정적 웹 사이트 구성 (p. 574)

정적 웹 사이트 구성

웹 사이트처럼 작동하도록 Amazon S3 버킷을 구성할 수 있습니다. 이 예제에서는 Amazon S3에서 웹 사이트를 호스팅하는 절차를 단계별로 살펴봅니다.

주제

- 1단계: 버킷 만들기 (p. 569)
- 2단계: 정적 웹 사이트 호스팅 활성화 (p. 569)
- 3단계: 퍼블릭 액세스 차단 설정 편집 (p. 570)
- 4단계: 버킷 콘텐츠를 공개적으로 사용 가능하도록 설정하는 버킷 정책 추가 (p. 571)
- 5단계: 인덱스 문서 구성 (p. 572)
- 6단계: 웹 사이트 엔드포인트 테스트 (p. 573)
- 7단계: 정리 (p. 574)

1단계: 버킷 만들기

아래 지침에서는 웹 사이트 호스팅용 버킷을 생성하는 방법에 대한 개요를 제공합니다. 버킷을 만드는 자세한 단계별 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷을 생성하려면 어떻게 해야 합니까? 단원을 참조하십시오.

버킷을 만들려면

1. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷 만들기를 선택합니다.
3. 버킷 이름(예: `example.com`)을 입력합니다.
4. 버킷을 생성하려는 리전을 선택합니다.

   가까운 리전을 선택하면 지연 시간과 요금을 최소화하고, 규제 요건을 다룰 수 있습니다. 선택한 리전에 따라 Amazon S3 웹 사이트 엔드포인트가 결정됩니다. 자세한 내용은 웹 사이트 엔드포인트 (p. 548) 단원을 참조하십시오.
5. 기본 설정을 적용하고 버킷을 생성하려면 [Create]를 선택합니다.

2단계: 정적 웹 사이트 호스팅 활성화

버킷을 생성한 후 버킷에 정적 웹 사이트 호스팅을 활성화할 수 있습니다. 새 버킷을 만들거나 기존 버킷을 사용할 수 있습니다.

정적 웹 사이트 호스팅을 활성화하려면

1. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷 이름 목록에서 정적 웹 사이트에 사용할 버킷을 선택합니다.

3. [Properties]를 선택합니다.
4. [Static website hosting]을 선택합니다.
5. 이 버킷을 사용하여 웹 사이트를 호스팅합니다.를 선택합니다.
6. 인덱스 문서의 이름을 입력합니다.

   인덱스 문서 이름은 일반적으로 `index.html`입니다. 인덱스 문서 이름은 대소문자를 구분하며 S3 버킷에 업로드하려는 HTML 인덱스 문서의 파일 이름과 정확히 일치해야 합니다. 자세한 내용은 인덱스 문서 구성 (p. 551) 단원을 참조하십시오.
7. (선택 사항) 사용자 지정 오류 문서를 추가할 경우 오류 문서 상자에 오류 문서 이름(예: **`error.html`**)을 입력합니다.

   오류 문서 이름은 대소문자를 구분하며 S3 버킷에 업로드하려는 HTML 오류 문서의 파일 이름과 정확히 일치해야 합니다. 자세한 내용은 (선택 사항)사용자 지정 오류 문서 구성 (p. 557) 단원을 참조하십시오.
8. (선택 사항) 고급 리디렉션 규칙을 지정하려면 Edit redirection rules(리디렉션 규칙 편집)에서 XML을 사용하여 규칙을 설명합니다.

   자세한 내용은 고급 조건부 리디렉션 구성 (p. 561) 단원을 참조하십시오.
9. 정적 웹 사이트 호스팅에서 엔드포인트를 기록합니다.

   엔드포인트는 버킷의 Amazon S3 웹 사이트 엔드포인트입니다. 버킷을 정적 웹 사이트로 구성한 후 이 엔드포인트를 사용하여 웹 사이트를 테스트할 수 있습니다.
10. 저장을 선택합니다.

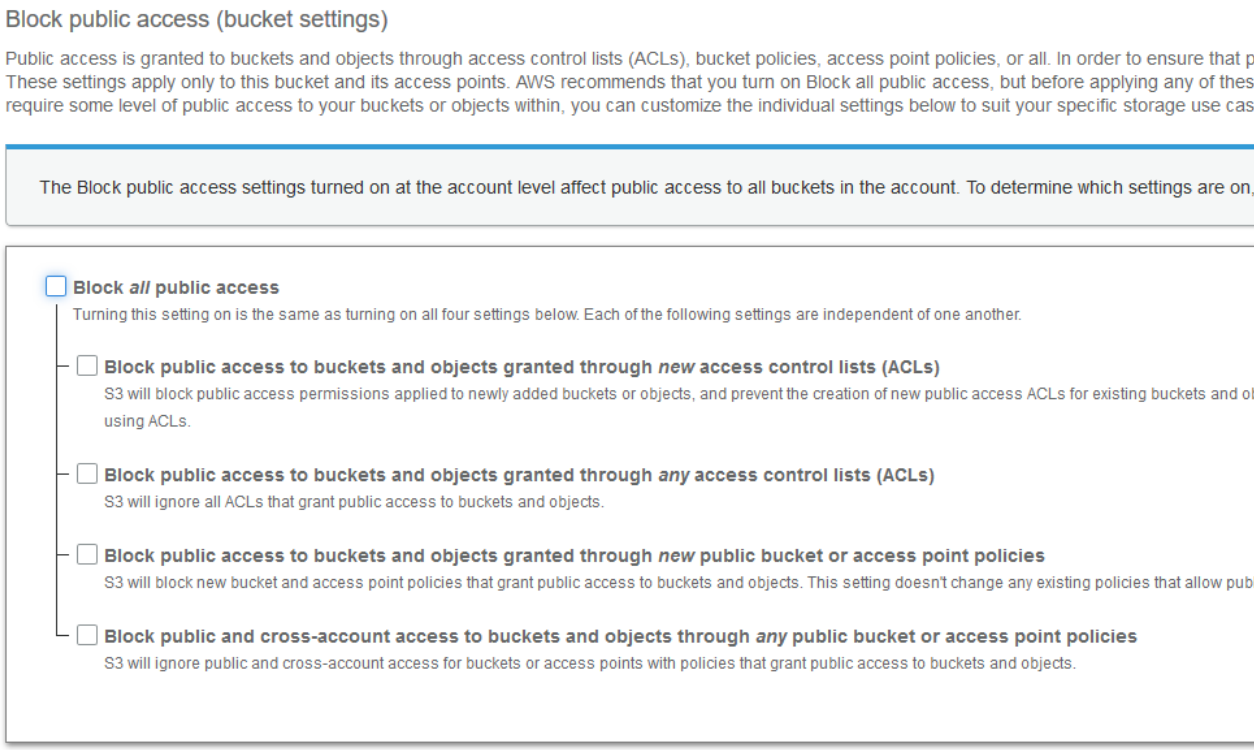
3단계: 퍼블릭 액세스 차단 설정 편집

기본적으로 Amazon S3은 계정 및 버킷에 대한 퍼블릭 액세스를 차단합니다. 버킷을 사용하여 정적 웹 사이트를 호스팅하려는 경우 이러한 단계를 사용하여 퍼블릭 액세스 차단 설정을 편집할 수 있습니다.

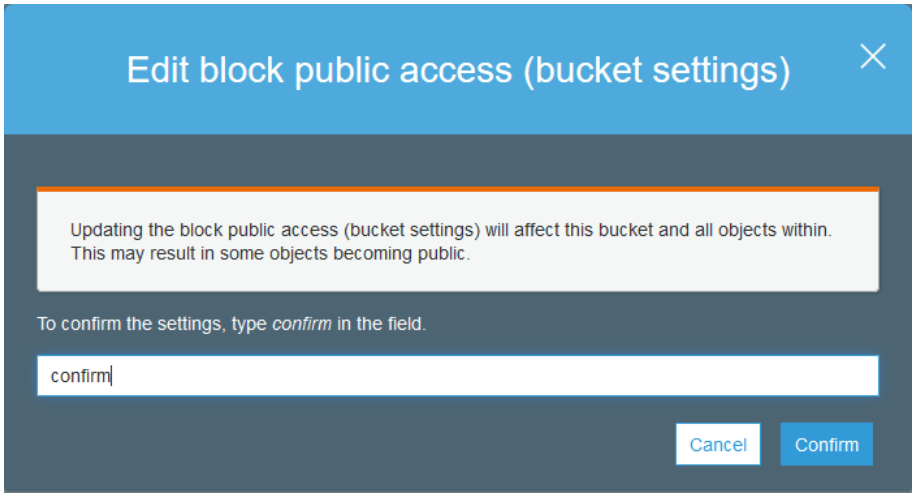
> Warning
>
> 이 단계를 완료하기 전에 Amazon S3 퍼블릭 액세스 차단 사용을 검토하여 퍼블릭 액세스 허용과 관련된 위험을 이해하고 이에 동의하는지 확인하십시오. 퍼블릭 액세스 차단 설정을 해제하여 버킷을 퍼블릭으로 만들면 인터넷상의 모든 사용자가 버킷에 액세스할 수 있습니다. 버킷에 대한 모든 퍼블릭 액세스를 차단하는 것이 좋습니다.

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 정적 웹 사이트로 구성한 버킷의 이름을 선택합니다.
3. Permissions를 선택합니다.
4. [Edit]를 선택합니다.
5. Block all public access(모든 퍼블릭 액세스 차단)를 선택 취소하고 저장을 선택합니다.

   > Warning
   >
   > 이 단계를 완료하기 전에 Amazon S3 퍼블릭 액세스 차단 사용을 검토하여 퍼블릭 액세스 허용과 관련된 위험을 이해하고 이에 동의하는지 확인하십시오. 퍼블릭 액세스 차단 설정을 해제하여 버킷을 퍼블릭으로 만들면 인터넷상의 모든 사용자가 버킷에 액세스할 수 있습니다. 버킷에 대한 모든 퍼블릭 액세스를 차단하는 것이 좋습니다.



6. 확인 상자에 **`confirm`**을 입력한 다음 확인을 선택합니다.



S3 버킷에서 버킷에 대한 액세스가 Objects can be public(객체가 퍼블릭이 될 수 있음)으로 업데이트됩니다. 이제 버킷의 객체를 공개적으로 읽기 가능하게 만들 버킷 정책을 추가할 수 있습니다. 액세스가 계속 Bucket and objects not public(버킷 및 객체가 퍼블릭이 아님)으로 표시되는 경우, 버킷 정책을 추가하기 전에 계정에 대해 퍼블릭 액세스 차단 설정을 편집해야 할 수도 있습니다.

4단계: 버킷 콘텐츠를 공개적으로 사용 가능하도록 설정하는 버킷 정책 추가

S3 퍼블릭 액세스 차단 설정을 편집한 후에는 버킷 정책을 추가하여 버킷에 퍼블릭 읽기 액세스 권한을 부여할 수 있습니다. 퍼블릭 읽기 액세스 권한을 부여하면 인터넷의 모든 사용자가 버킷에 액세스할 수 있습니다.

Important

다음 정책은 하나의 예일 뿐이며 버킷의 콘텐츠에 대한 전체 액세스를 허용합니다. 이 단계를 진행하기 전에 Amazon S3 버킷의 파일을 보호하려면 어떻게 해야 합니까?를 검토하여 S3 버킷의 파일 보안을 위한 모범 사례 및 퍼블릭 액세스 권한 부여와 관련된 위험을 파악할 수 있습니다.

1. 버킷에서 버킷의 이름을 선택합니다.
2. Permissions를 선택합니다.
3. [Bucket Policy]를 선택합니다.
4. 웹 사이트에 대한 퍼블릭 읽기 액세스 권한을 부여하려면 다음 버킷 정책을 복사한 후 버킷 정책 편집기에 붙여 넣습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::example.com/*"
            ]
        }
    ]
}
```

5. `Resource`를 업데이트하여 버킷 이름을 포함합니다.

   앞의 예제 버킷 정책에서 *`example.com`*은 버킷 이름입니다. 자체 버킷에 이 버킷 정책을 사용하려면 자체 버킷 이름과 일치하도록 이 이름을 업데이트해야 합니다.
6. 저장을 선택합니다.

   버킷에 퍼블릭 액세스 권한이 있음을 나타내는 경고가 나타납니다. 버킷 정책에 퍼블릭 레이블이 나타납니다.

   `Policy has invalid resource`라는 오류가 표시되면 버킷 정책의 버킷 이름이 사용자의 버킷 이름과 일치하는지 확인합니다. 버킷 정책 추가에 대한 자세한 내용은 S3 버킷 정책을 추가하려면 어떻게 해야 합니까?를 참조하십시오.

   오류 - 액세스 거부됨 경고가 표시되고 버킷 정책 편집기에서 버킷 정책을 저장할 수 없는 경우 계정 수준 및 버킷 수준 퍼블릭 액세스 차단 설정을 확인하여 버킷에 대한 퍼블릭 액세스를 허용하는지 확인하십시오. 웹 사이트 권한에 대한 자세한 내용은 웹 사이트 액세스에 필요한 권한을 참조하십시오.

5단계: 인덱스 문서 구성

버킷용 정적 웹 사이트 호스팅을 활성화할 때 인덱스 문서의 이름(예: **`index.html`**)을 입력합니다. 버킷용 정적 웹 사이트 호스팅을 활성화한 후 인덱스 문서 이름이 있는 HTML 파일을 버킷에 업로드합니다.

인덱스 문서를 구성하려면

1. `index.html` 파일을 생성합니다.

   `index.html` 파일이 없으면 다음 HTML을 사용하여 파일을 생성할 수 있습니다.

```
<html xmlns="http://www.w3.org/1999/xhtml" >
```

```
<head>
    <title>My Website Home Page</title>
</head>
<body>
  <h1>Welcome to my website</h1>
  <p>Now hosted on Amazon S3!</p>
</body>
</html>
```

2. 인덱스 파일을 로컬에 저장하고 파일 이름(예: `index.html`)을 기록합니다.

   정적 웹 사이트 호스팅을 활성화하고 인덱스 문서 이름을 입력하는 경우 정확한 파일 이름(예: `index.html`)을 입력합니다.
3. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
4. 버킷 목록에서 정적 웹 사이트를 호스팅하는 데 사용할 버킷의 이름을 선택합니다.
5. 버킷에 정적 웹 사이트 호스팅을 활성화하고 인덱스 문서의 정확한 이름(예: `index.html`)을 입력합니다. 자세한 내용은 웹 사이트 호스팅 사용 (p. 551) 단원을 참조하십시오.

   정적 웹 사이트 호스팅을 활성화한 후 6단계로 이동합니다.
6. 버킷에 인덱스 문서를 업로드하려면 다음 중 하나를 수행합니다.
   - 인덱스 파일을 콘솔 버킷 목록으로 끌어다 놓습니다.
   - 업로드를 선택하고 프롬프트의 메시지에 따라 인덱스 파일을 선택하고 업로드합니다.

   단계별 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 Amazon S3 버킷에 파일 및 폴더를 업로드하려면 어떻게 해야 합니까?를 참조하십시오.
7. (선택 사항) 버킷에 다른 웹 사이트 콘텐츠를 업로드합니다.

6단계: 웹 사이트 엔드포인트 테스트

버킷에 대한 정적 웹 사이트 호스팅을 구성한 후 웹 사이트 엔드포인트를 테스트할 수 있습니다.

> **Note**
>
> Amazon S3에서는 웹 사이트에 대한 HTTPS 액세스를 지원하지 않습니다. HTTPS를 사용하려는 경우 Amazon CloudFront를 사용하여 Amazon S3에서 호스팅되는 정적 웹 사이트를 제공할 수 있습니다.
> 자세한 내용은 에서 CloudFront를 사용하여 Amazon S3에서 호스팅되는 정적 웹 사이트 제공 방법 및 CloudFront 및 Amazon S3 오리진 간 통신에 HTTPS 필요를 참조하십시오.

웹 사이트 엔드포인트를 테스트하려면

정적 웹 사이트 호스팅을 활성화할 때 웹 사이트 엔드포인트를 기록한 경우 웹 사이트를 테스트하려면 브라우저에 웹 사이트 엔드포인트를 입력합니다. 브라우저에 `index.html` 페이지가 나타나는 경우 웹 사이트가 정상적으로 표시되는 것입니다. 자세한 내용은 Amazon S3 웹 사이트 엔드포인트를 참조하십시오.

테스트하기 전에 웹 사이트 엔드포인트를 가져와야 하는 경우 다음 단계를 따르십시오.

1. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷 목록에서 정적 웹 사이트를 호스팅하는 데 사용할 버킷의 이름을 선택합니다.
3. [Properties]를 선택합니다.
4. [Static website hosting]을 선택합니다.

5. 웹 사이트 엔드포인트를 테스트하려면 엔드포인트 옆에 있는 웹 사이트 엔드포인트를 선택합니다.

   브라우저에 `index.html` 페이지가 나타나는 경우 웹 사이트가 정상적으로 표시되는 것입니다.

이제 Amazon S3에 웹 사이트가 호스팅되었습니다. 이 웹 사이트는 Amazon S3 웹 사이트 엔드포인트를 통해 제공됩니다. 하지만, 사용자가 만든 웹 사이트의 콘텐츠를 표시하기 위해 `example.com`과 같은 도메인을 사용하고자 할 수도 있습니다. 또한 Amazon S3의 루트 도메인 지원을 통해 `http://www.example.com` 및 `http://example.com` 모두에 대한 요청을 처리하고자 할 수도 있습니다. 이 경우 다음 단계를 따라야 합니다. 문제 해결 예는 Route 53에 등록된 사용자 지정 도메인을 사용하여 정적 웹 사이트 구성 (p. 574) 단원을 참조하십시오.

7단계: 정리

실습용으로만 정적 웹 사이트를 생성한 경우에는 요금이 발생하지 않도록 할당한 AWS 리소스를 삭제합니다. AWS 리소스를 삭제한 후에는 웹 사이트를 사용할 수 없습니다. 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷을 삭제하려면 어떻게 해야 합니까? 단원을 참조하십시오.

Route 53에 등록된 사용자 지정 도메인을 사용하여 정적 웹 사이트 구성

Amazon S3에서 정적 웹 사이트를 호스팅하는 경우를 가정합니다. 예를 들어 `example.com`과 같은 도메인을 Amazon Route 53에 등록하고 Amazon S3 콘텐츠에서 `http://www.example.com`과 `http://example.com`에 대한 요청을 수행하게 합니다. 이 연습을 사용하여 정적 웹 사이트를 호스팅하고 사용자 지정 도메인 이름이 Route 53에 등록된 웹 사이트에 대한 리디렉션을 Amazon S3에 만드는 방법을 학습할 수 있습니다. Amazon S3에서 호스트할 기존 웹 사이트로 작업하거나 이 연습을 사용하여 처음부터 시작할 수 있습니다.

이 연습을 완료한 후에는 필요에 따라 Amazon CloudFront를 사용하여 웹 사이트의 성능을 향상시킬 수 있습니다. 자세한 내용은 Amazon CloudFront를 이용해 웹 사이트 속도 높이기 (p. 583) 단원을 참조하십시오.

주제

- 시작하기 전 (p. 574)
- 1단계: Route 53에 사용자 지정 도메인 등록 (p. 575)
- 2단계: 두 개의 버킷 생성 (p. 575)
- 3단계: 웹 사이트 호스팅용 루트 도메인 버킷 구성 (p. 576)
- 4단계: 웹 사이트 리디렉션용 하위 도메인 버킷 구성 (p. 576)
- 5단계: 웹 사이트 트래픽용 로깅 구성 (p. 577)
- 6단계: 인덱스 및 웹 사이트 콘텐츠 업로드 (p. 577)
- 7단계: 퍼블릭 액세스 차단 설정 편집 (p. 578)
- 8단계: 버킷 정책 연결 (p. 579)
- 9단계: 도메인 엔드포인트 테스트 (p. 580)
- 10단계: 도메인 및 하위 도메인에 대한 별칭 레코드 추가 (p. 581)
- 11단계: 웹 사이트 테스트 (p. 583)
- Amazon CloudFront를 이용해 웹 사이트 속도 높이기 (p. 583)
- 예제 리소스 정리 (p. 585)

시작하기 전

이 예제의 단계에 따르려면 다음과 같이 작업을 합니다.

Amazon Route 53 – Route 53을 사용하여 도메인을 등록하고 도메인의 인터넷 트래픽을 라우팅할 위치를 정의합니다. 이 예제는 도메인(`example.com`) 및 하위 도메인(`www.example.com`)의 트래픽을 HTML 파일이 포함된 Amazon S3 버킷으로 라우팅하는 Route 53 별칭 레코드의 생성 방법을 보여줍니다.

Amazon S3 – Amazon S3를 사용하여 버킷을 만들고 샘플 웹 사이트 페이지를 업로드해 모든 사용자가 콘텐츠를 볼 수 있도록 권한을 구성한 뒤, 웹 사이트 호스팅용 버킷을 구성합니다.

1단계: Route 53에 사용자 지정 도메인 등록

등록된 도메인 이름(예: `example.com`)이 아직 없으면 Route 53에 등록합니다. 자세한 내용은 Amazon Route 53 개발자 안내서의 새 도메인 등록 단원을 참조하십시오. 도메인 이름을 등록한 후 웹 사이트 호스팅용 Amazon S3 버킷을 생성하고 구성할 수 있습니다.

2단계: 두 개의 버킷 생성

루트 도메인 및 하위 도메인의 요청을 모두 지원하려면 두 개의 버킷을 만듭니다.

- 도메인 버킷 - `example.com`
- 하위 도메인 버킷 - `www.example.com`

루트 도메인 버킷(`example.com`)에서 콘텐츠를 호스팅합니다. 하위 도메인 버킷(`www.example.com`)에 대한 리디렉션 요청을 만듭니다. 누군가가 브라우저에 `www.example.com`을 입력하면 `example.com`으로 리디렉션되고 해당 이름을 가진 Amazon S3 버킷에서 호스팅되는 콘텐츠가 표시됩니다.

웹 사이트 호스팅용 버킷을 생성하려면

아래 지침에서는 웹 사이트 호스팅용 버킷을 생성하는 방법에 대한 개요를 제공합니다. 버킷을 만드는 자세한 단계별 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷을 생성하려면 어떻게 해야 합니까? 단원을 참조하십시오.

1. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 루트 도메인 버킷을 생성합니다.
   a. 버킷 만들기를 선택합니다.
   b. 버킷 이름(예: `example.com`)을 입력합니다.
   c. 버킷을 생성하려는 리전을 선택합니다.

      가까운 리전을 선택하면 지연 시간과 요금을 최소화하고, 규제 요건을 다룰 수 있습니다. 선택한 리전에 따라 Amazon S3 웹 사이트 엔드포인트가 결정됩니다. 자세한 내용은 웹 사이트 엔드포인트 (p. 548) 단원을 참조하십시오.
   d. 기본 설정을 적용하고 버킷을 생성하려면 [Create]를 선택합니다.
3. 하위 도메인 버킷을 생성합니다.
   a. 버킷 만들기를 선택합니다.
   b. 버킷 이름(예: `www.example.com`)을 입력합니다.
   c. 버킷을 생성하려는 리전을 선택합니다.

      가까운 리전을 선택하면 지연 시간과 요금을 최소화하고, 규제 요건을 다룰 수 있습니다. 선택한 리전에 따라 Amazon S3 웹 사이트 엔드포인트가 결정됩니다. 자세한 내용은 웹 사이트 엔드포인트 (p. 548) 단원을 참조하십시오.
   d. 기본 설정을 적용하고 버킷을 생성하려면 [Create]를 선택합니다.

다음 단계에서는 웹 사이트 호스팅을 위한 `example.com`을 구성합니다.

3단계: 웹 사이트 호스팅용 루트 도메인 버킷 구성

이 단계에서는 루트 도메인 버킷(`example.com`)을 웹 사이트로 구성합니다. 이 버킷은 사용자의 웹 사이트 콘텐츠를 포함합니다. 웹 사이트 호스팅용 버킷을 구성할 때 웹 사이트 엔드포인트 (p. 548)를 사용하여 웹 사이트에 액세스할 수 있습니다.

정적 웹 사이트 호스팅을 활성화하려면

1. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷 이름 목록에서 정적 웹 사이트에 사용할 버킷을 선택합니다.
3. [Properties]를 선택합니다.
4. [Static website hosting]을 선택합니다.
5. 이 버킷을 사용하여 웹 사이트를 호스팅합니다.를 선택합니다.
6. 인덱스 문서의 이름을 입력합니다.

   인덱스 문서 이름은 일반적으로 `index.html`입니다. 인덱스 문서 이름은 대소문자를 구분하며 S3 버킷에 업로드하려는 HTML 인덱스 문서의 파일 이름과 정확히 일치해야 합니다. 자세한 내용은 인덱스 문서 구성 (p. 551) 단원을 참조하십시오.
7. (선택 사항) 사용자 지정 오류 문서를 추가할 경우 오류 문서 상자에 오류 문서 이름(예: **`error.html`**)을 입력합니다.

   오류 문서 이름은 대소문자를 구분하며 S3 버킷에 업로드하려는 HTML 오류 문서의 파일 이름과 정확히 일치해야 합니다. 자세한 내용은 (선택 사항)사용자 지정 오류 문서 구성 (p. 557) 단원을 참조하십시오.
8. (선택 사항) 고급 리디렉션 규칙을 지정하려면 Edit redirection rules(리디렉션 규칙 편집)에서 XML을 사용하여 규칙을 설명합니다.

   자세한 내용은 고급 조건부 리디렉션 구성 (p. 561) 단원을 참조하십시오.
9. 정적 웹 사이트 호스팅에서 엔드포인트를 기록합니다.

   엔드포인트는 버킷의 Amazon S3 웹 사이트 엔드포인트입니다. 버킷을 정적 웹 사이트로 구성한 후 이 엔드포인트를 사용하여 웹 사이트를 테스트할 수 있습니다.
10. 저장을 선택합니다.

다음 단계에서는 하위 도메인(`www.example.com`)을 구성하여 요청을 도메인(`example.com`)으로 리디렉션합니다.

4단계: 웹 사이트 리디렉션용 하위 도메인 버킷 구성

웹 사이트 호스팅용 루트 도메인 버킷을 구성하면 도메인에 대한 모든 요청을 리디렉션하도록 하위 도메인 버킷을 구성할 수 있습니다. 이 예제에서 `www.example.com`에 대한 모든 요청은 `example.com`으로 리디렉션됩니다.

리디렉션 요청을 구성하려면

1. Amazon S3 콘솔의 버킷 목록에서 하위 도메인 버킷(이 예제에서는 `www.example.com`)을 선택합니다.
2. [Properties]를 선택합니다.
3. [Static website hosting]을 선택합니다.
4. [Redirect requests]를 선택합니다.
5. 대상 버킷 또는 도메인 상자에 도메인을 입력합니다(예: **`example.com`**).

6. 프로토콜 상자에 **`http`**를 입력합니다.
7. Save를 선택합니다.

5단계: 웹 사이트 트래픽용 로깅 구성

웹 사이트에 접속하는 방문자들의 수를 추적하려는 경우 필요에 따라 루트 도메인 버킷에 대한 로깅을 활성화할 수 있습니다. 정적 웹 사이트로 구성한 버킷에 대한 서버 액세스 로깅을 활성화합니다. 자세한 내용은 Amazon S3 서버 액세스 로깅을 참조하십시오. 그러나 웹 사이트 속도를 높이기 위해 Amazon CloudFront를 사용하려는 경우 CloudFront 로깅을 사용할 수도 있습니다.

다음 단계에서는 서버 액세스 로깅 및 CloudFront 로깅을 위한 로그 전송 버킷을 설정하고 서버 액세스 로깅을 활성화하는 방법을 보여줍니다. CloudFront 로깅 구성에 대한 자세한 내용은 예제: Amazon CloudFront로 웹 사이트 속도 향상을 참조하십시오.

정적 웹 사이트 버킷에 대한 로깅을 활성화하려면

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 정적 웹 사이트로 구성된 버킷을 생성한 리전에서 로깅용 버킷을 생성합니다(예: `logs.example.com`).
3. 서버 액세스 로깅 로그 파일에 대한 폴더를 생성합니다(예: `logs`).
4. (선택 사항) CloudFront를 사용하여 웹 사이트 성능을 개선하려면 CloudFront 로그 파일에 대한 폴더(예: `cdn`)를 생성합니다.
5. 버킷 목록에서 버킷을 선택합니다.
6. [Properties]를 선택합니다.
7. Server access logging(서버 액세스 로깅)을 선택합니다.
8. 로깅 활성화를 선택합니다.
9. 대상 버킷에서 로그 파일에 대해 생성한 버킷(예: `logs-example-bucket`)을 선택합니다.
10. 대상 접두사에서 로그 파일에 대해 생성한 폴더의 이름을 구분 기호(/)와 함께 입력합니다(예: **`logs/`**).

    대상 접두사를 설정할 때 쉽게 찾을 수 있도록 로그 데이터 파일을 폴더로 그룹화합니다.
11. 저장을 선택합니다.

    이제 로그 버킷에서 로그에 액세스할 수 있습니다. Amazon S3는 2시간마다 웹 사이트 액세스 로그를 로그 버킷에 기록합니다.
12. 로그를 보려면 개요를 선택하고 폴더를 선택합니다.

6단계: 인덱스 및 웹 사이트 콘텐츠 업로드

이 단계에서 인덱스 문서와 선택적 웹 사이트 콘텐츠를 루트 도메인 버킷에 업로드합니다.

버킷용 정적 웹 사이트 호스팅을 활성화할 때 인덱스 문서의 이름(예: **`index.html`**)을 입력합니다. 버킷용 정적 웹 사이트 호스팅을 활성화한 후 인덱스 문서 이름이 있는 HTML 파일을 버킷에 업로드합니다.

인덱스 문서를 구성하려면

1. `index.html` 파일을 생성합니다.

   `index.html` 파일이 없으면 다음 HTML을 사용하여 파일을 생성할 수 있습니다.

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>My Website Home Page</title>
```

```
</head>
<body>
  <h1>Welcome to my website</h1>
  <p>Now hosted on Amazon S3!</p>
</body>
</html>
```

2. 인덱스 파일을 로컬에 저장하고 파일 이름(예: `index.html`)을 기록합니다.

   정적 웹 사이트 호스팅을 활성화하고 인덱스 문서 이름을 입력하는 경우 정확한 파일 이름(예: `index.html`)을 입력합니다.
3. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
4. 버킷 목록에서 정적 웹 사이트를 호스팅하는 데 사용할 버킷의 이름을 선택합니다.
5. 버킷에 정적 웹 사이트 호스팅을 활성화하고 인덱스 문서의 정확한 이름(예: `index.html`)을 입력합니다. 자세한 내용은 웹 사이트 호스팅 사용 (p. 551) 단원을 참조하십시오.

   정적 웹 사이트 호스팅을 활성화한 후 6단계로 이동합니다.
6. 버킷에 인덱스 문서를 업로드하려면 다음 중 하나를 수행합니다.
   - 인덱스 파일을 콘솔 버킷 목록으로 끌어다 놓습니다.
   - 업로드를 선택하고 프롬프트의 메시지에 따라 인덱스 파일을 선택하고 업로드합니다.

   단계별 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 Amazon S3 버킷에 파일 및 폴더를 업로드하려면 어떻게 해야 합니까?를 참조하십시오.
7. (선택 사항) 버킷에 다른 웹 사이트 콘텐츠를 업로드합니다.

7단계: 퍼블릭 액세스 차단 설정 편집

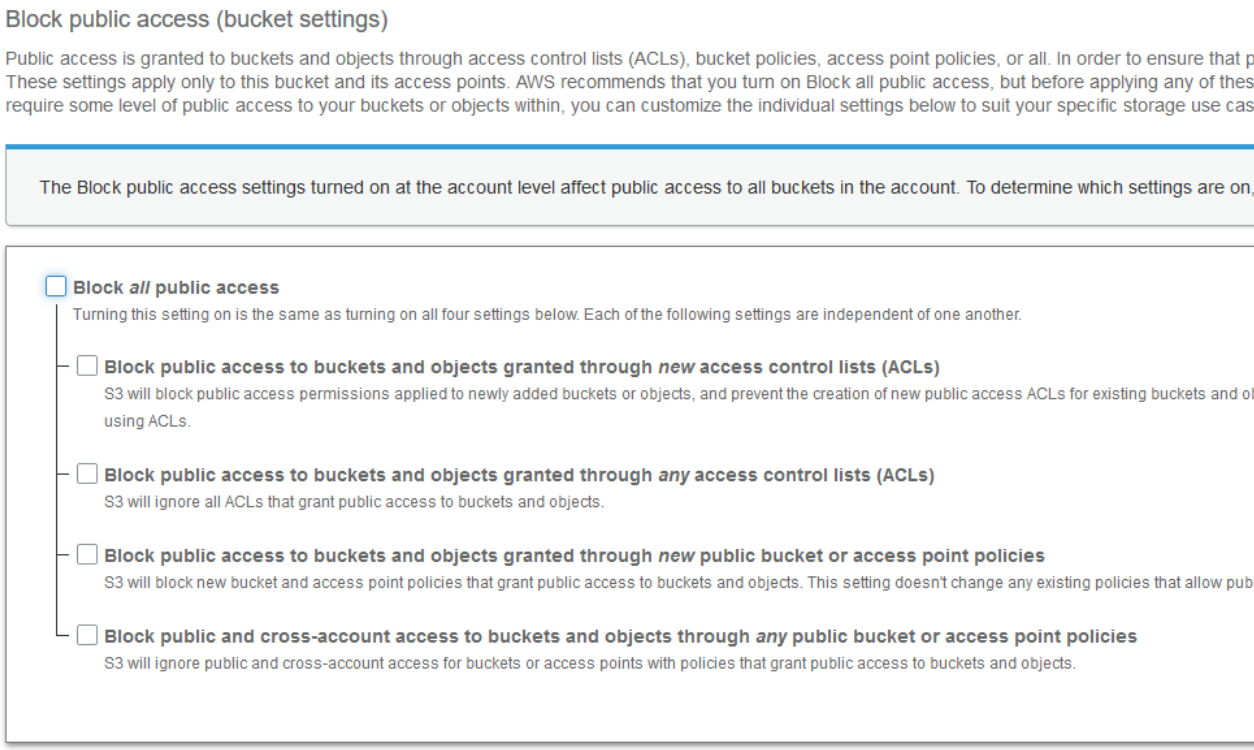
이 예제에서는 퍼블릭 액세스를 허용하도록 도메인 버킷(`example.com`)에 대한 퍼블릭 액세스 차단 설정을 편집합니다.

기본적으로 Amazon S3은 계정 및 버킷에 대한 퍼블릭 액세스를 차단합니다. 버킷을 사용하여 정적 웹 사이트를 호스팅하려는 경우 이러한 단계를 사용하여 퍼블릭 액세스 차단 설정을 편집할 수 있습니다.

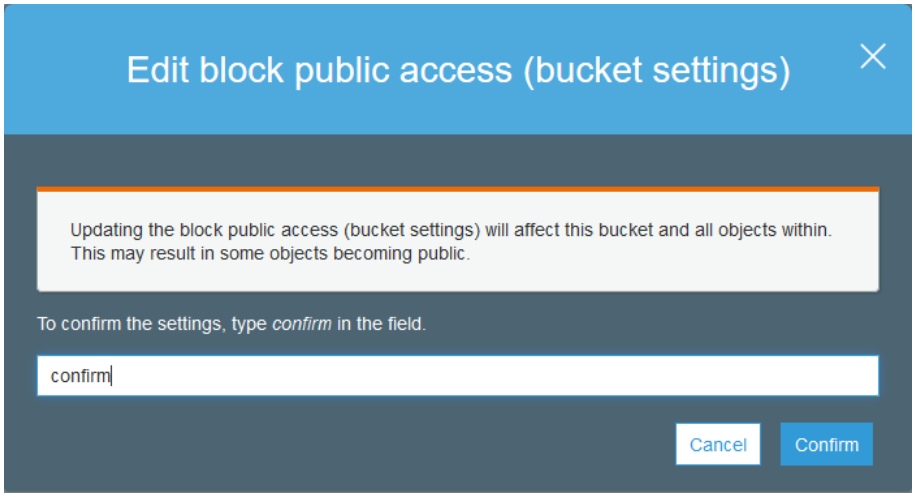
> **Warning**
>
> 이 단계를 완료하기 전에 Amazon S3 퍼블릭 액세스 차단 사용을 검토하여 퍼블릭 액세스 허용과 관련된 위험을 이해하고 이에 동의하는지 확인하십시오. 퍼블릭 액세스 차단 설정을 해제하여 버킷을 퍼블릭으로 만들면 인터넷상의 모든 사용자가 버킷에 액세스할 수 있습니다. 버킷에 대한 모든 퍼블릭 액세스를 차단하는 것이 좋습니다.

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 정적 웹 사이트로 구성한 버킷의 이름을 선택합니다.
3. Permissions를 선택합니다.
4. [Edit]를 선택합니다.
5. Block all public access(모든 퍼블릭 액세스 차단)를 선택 취소하고 저장을 선택합니다.

   > **Warning**
   >
   > 이 단계를 완료하기 전에 Amazon S3 퍼블릭 액세스 차단 사용을 검토하여 퍼블릭 액세스 허용과 관련된 위험을 이해하고 이에 동의하는지 확인하십시오. 퍼블릭 액세스 차단 설정을 해제하여 버킷을 퍼블릭으로 만들면 인터넷상의 모든 사용자가 버킷에 액세스할 수 있습니다. 버킷에 대한 모든 퍼블릭 액세스를 차단하는 것이 좋습니다.



6. 확인 상자에 **`confirm`**을 입력한 다음 확인을 선택합니다.



S3 버킷에서 버킷에 대한 액세스가 Objects can be public(객체가 퍼블릭이 될 수 있음)으로 업데이트됩니다. 이제 버킷의 객체를 공개적으로 읽기 가능하게 만들 버킷 정책을 추가할 수 있습니다. 액세스가 계속 Bucket and objects not public(버킷 및 객체가 퍼블릭이 아님)으로 표시되는 경우, 버킷 정책을 추가하기 전에 계정에 대해 퍼블릭 액세스 차단 설정을 편집해야 할 수도 있습니다.

8단계: 버킷 정책 연결

S3 퍼블릭 액세스 차단 설정을 편집한 후에는 버킷 정책을 추가하여 버킷에 퍼블릭 읽기 액세스 권한을 부여할 수 있습니다. 퍼블릭 읽기 액세스 권한을 부여하면 인터넷의 모든 사용자가 버킷에 액세스할 수 있습니다.

Important

다음 정책은 하나의 예일 뿐이며 버킷의 콘텐츠에 대한 전체 액세스를 허용합니다. 이 단계를 진행하기 전에 Amazon S3 버킷의 파일을 보호하려면 어떻게 해야 합니까?를 검토하여 S3 버킷의 파일 보안을 위한 모범 사례 및 퍼블릭 액세스 권한 부여와 관련된 위험을 파악할 수 있습니다.

1. 버킷에서 버킷의 이름을 선택합니다.
2. Permissions를 선택합니다.
3. [Bucket Policy]를 선택합니다.
4. 웹 사이트에 대한 퍼블릭 읽기 액세스 권한을 부여하려면 다음 버킷 정책을 복사한 후 버킷 정책 편집기에 붙여 넣습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::example.com/*"
            ]
        }
    ]
}
```

5. `Resource`를 업데이트하여 버킷 이름을 포함합니다.

   앞의 예제 버킷 정책에서 *`example.com`*은 버킷 이름입니다. 자체 버킷에 이 버킷 정책을 사용하려면 자체 버킷 이름과 일치하도록 이 이름을 업데이트해야 합니다.
6. 저장을 선택합니다.

   버킷에 퍼블릭 액세스 권한이 있음을 나타내는 경고가 나타납니다. 버킷 정책에 퍼블릭 레이블이 나타납니다.

   `Policy has invalid resource`라는 오류가 표시되면 버킷 정책의 버킷 이름이 사용자의 버킷 이름과 일치하는지 확인합니다. 버킷 정책 추가에 대한 자세한 내용은 S3 버킷 정책을 추가하려면 어떻게 해야 합니까?를 참조하십시오.

   오류 - 액세스 거부됨 경고가 표시되고 버킷 정책 편집기에서 버킷 정책을 저장할 수 없는 경우 계정 수준 및 버킷 수준 퍼블릭 액세스 차단 설정을 확인하여 버킷에 대한 퍼블릭 액세스를 허용하는지 확인하십시오. 웹 사이트 권한에 대한 자세한 내용은 웹 사이트 액세스에 필요한 권한을 참조하십시오.

다음 단계에서는 웹 사이트 엔드포인트를 파악하고 도메인 엔드포인트를 테스트할 수 있습니다.

9단계: 도메인 엔드포인트 테스트

퍼블릭 웹 사이트를 호스팅하도록 도메인 버킷을 구성한 후 엔드포인트를 테스트할 수 있습니다. 자세한 내용은 웹 사이트 엔드포인트 (p. 548) 단원을 참조하십시오. 하위 도메인 버킷은 정적 웹 사이트 호스팅이 아닌 웹 사이트 리디렉션에 대해 설정되어 있으므로 도메인 버킷의 엔드포인트만 테스트할 수 있습니다.

Note

Amazon S3에서는 웹 사이트에 대한 HTTPS 액세스를 지원하지 않습니다. HTTPS를 사용하려는 경우 Amazon CloudFront를 사용하여 Amazon S3에서 호스팅되는 정적 웹 사이트를 제공할 수 있습니다.

자세한 내용은 에서 CloudFront를 사용하여 Amazon S3에서 호스팅되는 정적 웹 사이트 제공 방법 및 CloudFront 및 Amazon S3 오리진 간 통신에 HTTPS 필요를 참조하십시오.

웹 사이트 엔드포인트를 테스트하려면

정적 웹 사이트 호스팅을 활성화할 때 웹 사이트 엔드포인트를 기록한 경우 웹 사이트를 테스트하려면 브라우저에 웹 사이트 엔드포인트를 입력합니다. 브라우저에 `index.html` 페이지가 나타나는 경우 웹 사이트가 정상적으로 표시되는 것입니다. 자세한 내용은 Amazon S3 웹 사이트 엔드포인트를 참조하십시오.

테스트하기 전에 웹 사이트 엔드포인트를 가져와야 하는 경우 다음 단계를 따르십시오.

1. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷 목록에서 정적 웹 사이트를 호스팅하는 데 사용할 버킷의 이름을 선택합니다.
3. [Properties]를 선택합니다.
4. [Static website hosting]을 선택합니다.
5. 웹 사이트 엔드포인트를 테스트하려면 엔드포인트 옆에 있는 웹 사이트 엔드포인트를 선택합니다.

   브라우저에 `index.html` 페이지가 나타나는 경우 웹 사이트가 정상적으로 표시되는 것입니다.

다음 단계에서는 Amazon Route 53을 이용해 고객이 사용자 지정 URL을 모두 사용하여 해당 사이트를 탐색할 수 있도록 합니다.

10단계: 도메인 및 하위 도메인에 대한 별칭 레코드 추가

이 단계에서 도메인 맵 `example.com` 및 `www.example.com`에 따른 호스팅 영역에 추가하는 별칭 레코드를 생성합니다. 별칭 레코드는 IP 주소 대신 Amazon S3 웹 사이트 엔드포인트를 사용합니다. Amazon Route 53은 별칭 레코드와 Amazon S3 버킷이 존재하는 IP 주소 간 매핑을 유지합니다. 별칭 레코드를 두 개(루트 도메인용 및 하위 도메인용) 만듭니다.

루트 도메인(**`example.com`**)에 대한 별칭 레코드를 추가하려면

1. https://console.aws.amazon.com/route53/에서 Route 53 콘솔을 엽니다.

   Note

   Route 53을 사용하고 있지 않으면 Amazon Route 53 개발자 안내서의 1단계: 도메인 등록을 참조하십시오. 설정을 완료했으면 지시 사항을 다시 시작할 수 있습니다.
2. Hosted Zones(호스팅 영역)를 선택합니다.
3. 호스팅 영역의 목록에서 사용자의 도메인 이름과 일치하는 호스팅 영역의 이름을 선택합니다.
4. [Create Record Set]를 선택합니다.
5. 다음 값을 지정합니다.

   이름

   기본값(호스팅 영역과 도메인의 이름)을 그대로 사용합니다.

   루트 도메인의 경우 이름 필드에 추가 정보를 입력할 필요는 없습니다.

   유형

   A – IPv4 주소를 선택합니다.

   별칭

   예를 선택합니다.

별칭 대상

목록의 S3 website endpoints(S3 웹 사이트 엔드포인트) 섹션에서 버킷 이름을 선택합니다.

버킷 이름은 이름 상자에 나타나는 이름과 일치해야 합니다. Alias Target(별칭 대상) 목록에서 버킷 이름 뒤에는 버킷이 생성된 리전의 Amazon S3 웹 사이트 엔드포인트가 옵니다(예: `example.com (s3-website-us-west-2)`). 다음과 같은 경우 Alias Target(별칭 대상)에 하나의 버킷이 나타납니다.

- 버킷을 정적 웹 사이트로 구성한 경우
- 버킷 이름이 생성 중인 레코드의 이름과 동일한 경우
- 현재 AWS 계정에서 버킷을 생성한 경우

버킷이 Alias Target(별칭 대상) 목록에 나타나지 않으면 버킷이 생성된 리전의 Amazon S3 웹 사이트 엔드포인트를 입력합니다(예: `s3-website-us-west-2`). Amazon S3 웹 사이트 엔드포인트의 전체 목록은 Amazon S3 웹 사이트 엔드포인트를 참조하십시오. 별칭 대상에 대한 자세한 내용은 Amazon Route 53 개발자 안내서의 별칭 대상을 참조하십시오.

라우팅 정책

기본값인 [Simple]을 수락합니다.

대상 상태 평가

기본값인 [No]를 수락합니다.

6. Create를 선택합니다.

하위 도메인(`www.example.com`)에 별칭 레코드를 추가하려면

1. 루트 도메인(`example.com`)에 대한 호스팅 영역에서 레코드 세트 생성을 선택합니다.
2. 다음 값을 지정합니다.

   이름

   하위 도메인에 대해 `www`를 상자에 입력합니다.

   Type

   A – IPv4 주소를 선택합니다.

   별칭

   예를 선택합니다.

   별칭 대상

   목록의 S3 website endpoints(S3 웹 사이트 엔드포인트) 섹션에서 이름 필드에 표시되는 동일한 버킷 이름을 선택합니다(예: `www.example.com (s3-website-us-west-2)`).

   라우팅 정책

   기본값인 [Simple]을 수락합니다.

   대상 상태 평가

   기본값인 [No]를 수락합니다.

3. Create를 선택합니다.

   Note

   변경 사항은 일반적으로 60초 이내에 모든 Route 53 서버로 전파됩니다. 전파가 완료되면 이 절차에서 생성한 별칭 레코드의 이름을 사용하여 트래픽을 Amazon S3 버킷으로 라우팅할 수 있습니다.

11단계: 웹 사이트 테스트

웹 사이트 및 리디렉션 작업이 올바르게 작동하는지 확인합니다. 브라우저에 해당 URL을 입력합니다. 이 예제에서는 다음 URL을 이용할 수 있습니다.

- 도메인(`http://example.com`) – `example.com` 버킷의 인덱스 문서를 표시합니다.
- 하위 도메인(`http://www.example.com`) – 요청을 `http://example.com`으로 리디렉션합니다. `example.com` 버킷의 인덱스 문서가 표시됩니다.

예상 동작을 확인하기 위해 웹 브라우저의 캐시를 지워야 하는 경우도 있습니다.

Amazon CloudFront 배포를 설정하여 웹 사이트의 성능을 개선하고 웹 사이트 트래픽을 검토하는 데 사용할 수 있는 로그를 제공할 수 있습니다. 자세한 내용은 Amazon CloudFront를 이용해 웹 사이트 속도 높이기 (p. 583) 단원을 참조하십시오.

Amazon CloudFront를 이용해 웹 사이트 속도 높이기

Amazon CloudFront를 사용해 웹 사이트 성능을 개선할 수 있습니다. CloudFront를 통해 웹 사이트 파일(HTML, 이미지, 동영상 등)을 엣지 로케이션이라고 하는 전 세계 각지의 데이터 센터에서 사용할 수 있습니다. 방문자가 웹 사이트에서 파일을 요청할 때 CloudFront는 가장 가까운 엣지 로케이션에 있는 파일 사본으로 요청을 자동 리디렉션합니다. 이렇게 하면 방문자가 더 멀리 있는 데이터 센터에서 콘텐츠를 요청한 경우보다 다운로드 시간이 빨라집니다.

CloudFront는 지정된 기간 동안 엣지 로케이션에 있는 콘텐츠를 캐시합니다. 방문자가 만료일보다 더 오래 캐시된 콘텐츠를 요청하는 경우에 CloudFront는 그 콘텐츠의 최신 버전이 사용 가능한지 알아보기 위해 오리진 서버를 확인합니다. 최신 버전이 사용 가능하다면, CloudFront는 새로운 버전을 엣지 로케이션에 복사합니다. 원본 콘텐츠에 대한 변경 사항은 방문객이 그 콘텐츠를 요청할 때 엣지 로케이션에 복제됩니다.

주제

- 1단계: CloudFront 배포 만들기 (p. 583)
- 2단계: 도메인 및 하위 도메인용 레코드 세트 업데이트 (p. 584)
- (선택 사항) 3단계: 로그 파일 확인 (p. 585)

1단계: CloudFront 배포 만들기

먼저 CloudFront 배포를 만듭니다. 이를 통해 전 세계 각지의 데이터 센터에서 웹 사이트를 사용할 수 있게 됩니다.

Amazon S3 오리진으로 배포를 생성하려면

1. https://console.aws.amazon.com/cloudfront/에서 CloudFront 콘솔을 엽니다.
2. [Create Distribution]을 선택합니다.
3. Select a delivery method for your content 페이지의 Web(웹)에서 시작하기를 선택합니다.
4. Create Distribution(배포 생성) 페이지의 Origin Settings(오리진 설정) 섹션에서 Origin Domain Name(오리진 도메인 이름) 상자에 해당 버킷에 대한 Amazon S3 웹 사이트 엔드포인트를 입력합니다(예: `example.com.s3-website.us-west-1.amazonaws.com`).

   CloudFront가 사용자 대신 Origin ID(오리진 ID)를 채웁니다.
5. Default Cache Behavior Settings(기본 캐시 동작 설정)에서는 기본값으로 설정되어 있는 값을 유지합니다.

   이 구성 옵션에 대한 자세한 내용은 Amazon CloudFront 개발자 안내서의 웹 배포의 생성 또는 업데이트 시 지정하는 값을 참조하십시오.
6. Distribution Settings(배포 설정)에서 다음 작업을 수행합니다.

a. Use All Edge Locations (Best Performance)(전체 엣지 로케이션 사용(최상의 성능))로 설정된 Price Class(가격 등급)를 그대로 둡니다.
b. Alternate Domain Names (CNAMEs)(대체 도메인 이름(CNAME))를 비워 둡니다.
c. 기본 루트 객체에 인덱스 문서의 이름을 입력합니다(예: `index.html`).

   배포에 액세스하는 데 사용된 URL에 파일 이름이 없으면 CloudFront 배포가 인덱스 문서를 반환합니다. 기본 루트 객체는 정적 웹 사이트에 대한 인덱스 문서의 이름과 정확히 일치해야 합니다. 자세한 내용은 인덱스 문서 구성 (p. 551) 단원을 참조하십시오.
d. 로깅을 설정으로 설정합니다.
e. Bucket for Logs(로그용 버킷)에서 앞서 생성한 로깅 버킷을 선택합니다.

   로깅 버킷 구성에 대한 자세한 내용은 (선택 사항) 웹 트래픽 로깅 구성 (p. 556) 단원을 참조하십시오.
f. CloudFront 배포에 대한 트래픽에 의해 생성된 로그를 폴더에 저장하려면 Log Prefix에 폴더 이름을 입력합니다.
g. 기타 모든 설정은 기본값을 유지합니다.

7. [Create Distribution]을 선택합니다.
8. 현재의 배포 상태를 보려면, 콘솔에서 배포를 찾아 상태 열을 확인합니다.

   `InProgress` 상태는 배포가 아직 완전히 이루어지지 않았음을 뜻합니다.

   배포가 완료되면 새 CloudFront 도메인 이름으로 콘텐츠를 참조할 수 있습니다.
9. CloudFront 콘솔에 표시되는 도메인 이름의 값을 기록하십시오(예: `dj4p1rv6mvubz.cloudfront.net`).
10. CloudFront 배포가 이루어지고 있는지 확인하려면, 웹 브라우저에서 배포의 도메인 이름을 입력하십시오.

    웹 사이트가 표시되면 CloudFront 배포가 작동합니다. 웹 사이트의 Amazon Route 53에 사용자 정의 도메인이 등록되어 있는 경우 다음 단계에서 레코드 세트를 업데이트하려면 CloudFront 도메인 이름이 필요합니다.

2단계: 도메인 및 하위 도메인용 레코드 세트 업데이트

CloudFront 배포를 성공적으로 생성했으므로, Route 53의 별칭 레코드를 업데이트하여 새로운 CloudFront 배포를 가리키도록 합니다.

CloudFront 배포를 가리키도록 별칭 레코드를 업데이트하려면

1. https://console.aws.amazon.com/route53/에서 Route 53 콘솔을 엽니다.
2. Hosted Zones(호스팅 영역) 페이지에서 하위 도메인을 위해 생성한 호스팅 영역을 선택합니다.
3. Go to Record Set를 선택합니다.
4. 하위 도메인에 대해 생성한 A 레코드를 선택합니다(예: `www.example.com`).
5. Alias Target(별칭 대상)에서 CloudFront 배포를 선택합니다.
6. Save Record Set를 선택합니다.
7. 루트 도메인에 대한 A 레코드를 CloudFront 배포로 리디렉션하려면 이 절차를 반복합니다.

   레코드 세트를 업데이트한 내용은 2 –48시간 내에 적용됩니다.
8. 새 A 레코드가 적용되었는지 확인하려면 웹 브라우저에서 하위 도메인 URL을 입력합니다(예: `http://www.example.com`).

   브라우저가 루트 도메인(예: `http://example.com`)으로 리디렉션되지 않는 경우 새 A 레코드가 적용된 것입니다. 새로운 A 레코드가 적용되면 새로운 A 레코드에 의해 CloudFront 배포로 라우팅

되는 트래픽은 루트 도메인으로 리디렉션되지 않습니다. `http://example.com` 또는 `http://www.example.com`를 이용해 사이트를 참조하는 방문자가 가장 가까운 CloudFront 엣지 로케이션으로 리디렉션됨으로써, 더 빨라진 다운로드 시간의 이점을 누릴 수 있습니다.

Tip

브라우저는 리디렉션 설정을 캐시할 수 있습니다. 새로운 A 레코드 설정이 이미 적용되었어야 하는데 여전히 브라우저에서 `http://www.example.com`이 `http://example.com`으로 리디렉션된다면, 브라우저 애플리케이션을 닫았다가 다시 열어 브라우저의 검색 기록 및 캐시를 삭제하거나 다른 웹 브라우저를 사용하십시오.

(선택 사항) 3단계: 로그 파일 확인

액세스 로그는 웹 사이트를 방문하는 사람의 수를 알려 줍니다. 또한 Amazon EMR과 같은 다른 서비스로 분석할 수 있는 중요 비즈니스 데이터를 포함하고 있습니다.

CloudFront 로그는 CloudFront 배포를 생성하고 로깅을 활성화할 때 선택한 버킷 및 폴더에 저장됩니다. CloudFront는 해당 요청이 이루어진 후 24시간 이내에 로그 버킷에 로그를 기록합니다.

웹 사이트의 로그 파일을 보는 방법

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 웹 사이트에 대한 로깅 버킷을 선택합니다.
3. CloudFront 로그 폴더를 선택합니다.
4. 열기 전에 CloudFront가 작성한 `..gzip` 파일을 다운로드합니다.

   실습용으로만 웹 사이트를 생성한 경우에는 요금이 발생하지 않도록 할당한 리소스를 삭제할 수 있습니다. 그렇게 하려면 예제 리소스 정리 (p. 585) 단원을 참조하십시오. AWS 리소스를 삭제한 후에는 웹 사이트를 사용할 수 없습니다.

예제 리소스 정리

실습용으로 정적 웹 사이트를 생성한 경우에는 요금이 발생하지 않도록 할당한 AWS 리소스를 삭제해야 합니다. AWS 리소스를 삭제한 후에는 웹 사이트를 사용할 수 없습니다.

작업

- 1단계: Amazon CloudFront 배포 삭제 (p. 585)
- 2단계: Route 53 호스팅 영역 삭제 (p. 586)
- 3단계: 로깅 비활성화 및 S3 버킷 삭제 (p. 586)

1단계: Amazon CloudFront 배포 삭제

Amazon CloudFront 배포를 삭제하기 전에 반드시 배포를 비활성화해야 합니다. 비활성화된 배포가 작동하지 않아 요금이 발생하지 않습니다. 언제든지 비활성화된 배포를 활성화할 수 있습니다. 비활성화된 배포는 삭제 후 사용할 수 없습니다.

CloudFront 배포를 비활성화 및 삭제하는 방법

1. https://console.aws.amazon.com/cloudfront/에서 CloudFront 콘솔을 엽니다.
2. 비활성화하려는 배포를 선택한 후 비활성화를 선택합니다.
3. 확인 메시지가 표시되면 예, 비활성화를 선택합니다.
4. 비활성화된 배포를 선택한 후 삭제를 선택합니다.
5. 확인 메시지가 나타나면 예, 삭제합니다를 선택합니다.

2단계: Route 53 호스팅 영역 삭제

호스팅 영역을 삭제하기 전에 반드시 앞서 만든 레코드 세트를 삭제해야 합니다. NS 및 SOA 레코드는 삭제할 필요가 없습니다. 이 둘은 호스팅 영역을 삭제할 때 자동으로 삭제됩니다.

레코드 세트를 삭제하려면

1. https://console.aws.amazon.com/route53/에서 Route 53 콘솔을 엽니다.
2. 도메인 이름 목록에서 도메인 이름을 선택한 후 Go to Record Sets(레코드 세트로 이동)를 선택합니다.
3. 레코드 세트 목록에서 생성한 A 레코드를 선택합니다.

   각 레코드 세트의 유형은 유형 열에 나열됩니다.
4. Delete Record Set(레코드 세트 삭제)를 선택합니다.
5. 확인 메시지가 표시되면 확인을 선택합니다.

Route 53 호스팅 영역을 삭제하려면

1. 이전 절차에 이어서 Back to Hosted Zones(호스팅 영역으로 돌아가기)를 선택합니다.
2. 도메인 이름을 선택한 후 Delete Hosted Zone(호스팅 영역 삭제)을 선택합니다.
3. 확인 메시지가 표시되면 확인을 선택합니다.

3단계: 로깅 비활성화 및 S3 버킷 삭제

S3 버킷을 삭제하기 전에 버킷에 대한 로깅이 비활성화되어 있는지 확인하십시오. 그렇지 않으면 삭제해도 AWS가 해당 버킷에 계속 로그를 기록합니다.

버킷에 대한 로깅을 비활성화하려면

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷에서 버킷 이름을 선택한 다음 속성을 선택합니다.
3. 속성에서 로깅을 선택합니다.
4. 활성화 확인란의 선택을 취소합니다.
5. Save를 선택합니다.

이제 버킷을 삭제할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷을 삭제하려면 어떻게 해야 합니까? 단원을 참조하십시오.

Amazon S3 이벤트 알림 구성

Amazon S3 알림 기능은 버킷에서 특정 이벤트가 발생할 때 알림을 보냅니다. 알림을 활성화하려면 먼저 Amazon S3가 게시할 이벤트와 Amazon S3가 알림을 보낼 대상을 식별하는 알림 구성을 추가해야 합니다. 버킷에 연결된 알림 하위 리소스에 이 구성을 저장합니다. 자세한 내용은 버킷 구성 옵션 (p. 55)를 참조하십시오. Amazon S3는 이 하위 리소스를 관리하기 위한 API를 제공합니다.

> **Important**
>
> Amazon S3 이벤트 알림은 일반적으로 몇 초 안에 이벤트를 전송하지만 1분 이상 소요되는 경우도 있습니다.
> 버전이 지정되지 않은 단일 객체에 대해 동시에 두 번의 쓰기가 수행되는 경우 이벤트 알림이 한 번만 전송될 수 있습니다. 쓰기에 성공할 때마다 이벤트 알림이 전송되도록 하려면 버킷에서 버전 관리를 활성화합니다. 버전 관리를 사용하면 쓰기가 성공적으로 수행될 때마다 객체의 새 버전이 생성되고 이벤트 알림도 전송됩니다.

주제

- 알림 개요 (p. 587)
- 이벤트 알림을 활성화하는 방법 (p. 588)
- 이벤트 알림 유형 및 대상 (p. 590)
- 객체 키 이름 필터링으로 알림 구성 (p. 591)
- 대상에 이벤트 알림 메시지를 게시할 권한 부여 (p. 596)
- 예제 안내: 버킷의 알림 구성(메시지 대상: SNS 주제 및 SQS 대기열) (p. 598)
- 이벤트 메시지 구조 (p. 603)

알림 개요

현재 Amazon S3는 다음 이벤트에 대한 알림을 게시할 수 있습니다.

- 새 객체 생성 이벤트 — Amazon S3는 여러 객체 생성 API를 지원합니다. 따라서 특정 API(예: `s3:ObjectCreated:Put`)를 사용할 경우에만 알림을 보내도록 요청하거나, 와일드카드(예: `s3:ObjectCreated:*`)를 사용하여 API에 관계없이 객체가 생성되면 알림을 보내도록 요청할 수 있습니다.
- 객체 제거 이벤트 — Amazon S3는 버전이 지정된 객체 및 버전이 지정되지 않은 객체의 삭제를 지원합니다. 객체 버전 관리에 대한 자세한 내용은 객체 버전 관리 (p. 112) 및 버전 관리 사용 (p. 468)를 참조하십시오.

  `s3:ObjectRemoved:Delete` 이벤트 유형을 사용하여 객체가 삭제되거나 버전이 지정된 객체가 영구적으로 삭제될 경우 알림을 요청할 수 있습니다. 또한 `s3:ObjectRemoved:DeleteMarkerCreated`를 사용하여 버전이 지정된 객체에 대해 삭제 마커가 생성될 경우 알림을 요청할 수 있습니다. `s3:ObjectRemoved:*` 와일드카드를 사용하여 객체가 삭제되는 모든 경우에 알림을 요청할 수도 있습니다. 버전 지정된 객체 삭제에 대한 자세한 내용은 객체 버전 삭제 (p. 481)를 참조하십시오.
- 객체 복원 이벤트 — Amazon S3는 S3 Glacier 스토리지 클래스에 보관된 객체의 복원을 지원합니다. `s3:ObjectRestore:Completed`를 사용하여 객체 복원 완료 알림을 요청합니다. `s3:ObjectRestore:Post`를 사용하여 복원 시작 알림을 요청합니다.
- RRS(Reduced Redundancy Storage) 객체 손실 이벤트 — Amazon S3가 RRS 스토리지 클래스에서 객체 손실이 확인되면 알림 메시지를 보냅니다.

- 복제 이벤트 — Amazon S3는 S3 Replication Time Control (S3 RTC)가 활성화된 복제 구성에 대한 이벤트 알림을 보냅니다. 객체 복제가 실패할 경우, 객체가 15분 임계값을 초과할 경우, 객체가 15분 임계값 이후에 복제될 경우, 객체가 더 이상 복제 지표에 의해 추적되지 않을 경우 이러한 알림을 보냅니다. 객체가 대상 리전으로 복제될 때 두 번째 이벤트를 게시합니다.

지원되는 이벤트 유형의 전체 목록은 지원되는 이벤트 유형 (p. 590) 단원을 참조하십시오.

Amazon S3는 다음과 같은 이벤트 게시 대상을 지원합니다.

- Amazon Simple Notification Service(Amazon SNS) 주제

  Amazon SNS는 유연하며 완벽하게 관리되는 푸시 메시징 서비스입니다. 이 서비스를 사용하여 모바일 디바이스 또는 분산 서비스로 메시지를 푸시할 수 있습니다. SNS를 사용해 메시지를 한 번 게시하고 한 번 이상 전송할 수 있습니다. SNS에 대한 자세한 내용은 Amazon SNS 제품 상세 페이지를 참조하십시오.
- Amazon Simple Queue Service(Amazon SQS) 대기열

  Amazon SQS는 확장 가능하며 완벽하게 관리되는 메시지 대기열 서비스입니다. SQS를 사용하면 다른 서비스를 항상 가용 상태로 유지하지 않고도 모든 데이터 볼륨을 전송할 수 있습니다. 알림 구성에서 Amazon S3가 SQS 대기열에 이벤트를 게시하도록 요청할 수 있습니다.

  현재 스탠다드 SQS 대기열은 Amazon S3 이벤트 알림 대상으로만 허용되지만 FIFO SQS 대기열은 허용되지 않습니다. Amazon SQS에 대한 자세한 내용은 Amazon SQS 제품 세부 정보 페이지를 참조하십시오.
- AWS Lambda

  AWS Lambda는 간편하게 새 정보에 신속하게 응답하는 애플리케이션을 구축할 수 있는 컴퓨팅 서비스입니다. AWS Lambda는 이미지 업로드, 앱에서의 작업, 웹 사이트 클릭 또는 연결된 디바이스의 출력과 같은 이벤트에 대한 응답으로 사용자가 작성한 코드를 실행합니다.

  AWS Lambda을 사용하여 사용자 지정 로직으로 다른 AWS 서비스를 확장하거나 AWS 규모와 성능, 보안에 따라 작동하는 자체 백엔드를 만들 수 있습니다. AWS Lambda를 사용하면 손쉽게 필요할 경우에만 실행되며, 하루 몇 번의 요청에서 초당 수천 개의 요청으로 자동 확장되는 분산 이벤트 중심 애플리케이션을 만들 수 있습니다.

  AWS Lambda는 Amazon S3 버킷 이벤트에 대한 응답으로 맞춤형 로직을 실행할 수 있습니다. 맞춤 코드를 AWS Lambda로 업로드하여 Lambda 함수를 만듭니다. Amazon S3가 특정 유형의 이벤트(예: 객체 생성 이벤트)를 감지한 후 AWS Lambda에 이벤트를 게시하고 Lambda 함수를 호출할 수 있습니다. 그 응답으로 AWS Lambda는 함수를 실행합니다.

  Warning

  알림이 알림을 트리거하는 버킷에 기록되면 실행 루프가 발생할 수 있습니다. 예를 들어, 객체가 업로드될 때마다 버킷이 Lambda 함수를 트리거하고 그 함수가 객체를 버킷에 업로드하는 경우 함수는 간접적으로 자신을 트리거합니다 이렇게 되지 않도록 하려면 두 개의 버킷을 사용하거나, 수신 객체에 사용되는 접두사에만 적용되도록 트리거를 구성합니다.
  AWS Lambda를 이용한 Amazon S3 알림에 대한 자세한 내용과 사용 예는 AWS Lambda Developer Guide의 Amazon S3와 함께 AWS Lambda 사용을 참조하십시오.

이벤트 알림을 활성화하는 방법

알림 활성화는 버킷 수준의 작업이며, 버킷에 연결된 알림 하위 리소스에 알림 구성 정보를 저장합니다. 다음과 같은 메서드를 사용하여 알림 구성을 관리할 수 있습니다.

- Amazon S3 콘솔 사용

콘솔 UI를 사용하면 코드를 작성하지 않고도 버킷에 알림 구성을 설정할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷에 대한 이벤트 알림을 활성화하고 구성하려면 어떻게 해야 합니까? 단원을 참조하십시오.

- AWS SDK를 사용한 프로그래밍 방식

  Note

  필요한 경우 코드에서 직접 Amazon S3 REST API를 호출하게 할 수도 있습니다. 하지만 이 방법은 요청 인증을 위한 코드를 작성해야 하므로 번거로울 수 있습니다.

내부적으로는 콘솔이나 SDK 모두 Amazon S3 REST API를 호출하여 버킷과 연결된 알림 하위 리소스를 관리합니다. AWS SDK를 사용하여 알림을 구성하는 예제는 위 섹션에 제공된 연습 링크를 참조하십시오.

사용하는 메서드에 관계없이 Amazon S3는 버킷과 연결된 알림 하위 리소스에 알림 구성을 XML로 저장합니다. 버킷 하위 리소스에 대한 자세한 내용은 버킷 구성 옵션 (p. 55) 단원을 참조하십시오.

기본적으로 모든 유형의 이벤트에 대해 알림이 비활성화되어 있습니다. 따라서 처음에 알림 하위 리소스에는 빈 구성이 저장되어 있습니다.

```
<NotificationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
</NotificationConfiguration>
```

특정 유형의 이벤트에 대해 알림을 설정하려면 먼저 Amazon S3가 게시할 이벤트 유형 및 게시된 이벤트에 대한 알림을 보낼 대상을 지정한 구성으로 이 XML을 교체하십시오. 각 대상에 대해 해당 XML 구성을 추가해야 합니다. 예:

- 이벤트 메시지를 SQS 대기열에 게시 — 하나 이상의 이벤트 유형의 알림 대상으로 SQS 대기열을 설정하려면 `QueueConfiguration`을 추가합니다.

```
<NotificationConfiguration>
  <QueueConfiguration>
    <Id>optional-id-string</Id>
    <Queue>sqs-queue-arn</Queue>
    <Event>event-type</Event>
    <Event>event-type</Event>
     ...
  </QueueConfiguration>
   ...
</NotificationConfiguration>
```

- 이벤트 메시지를 SNS 주제에 게시 — 특정 이벤트 유형의 알림 대상으로 SNS 주제를 설정하려면 `TopicConfiguration`을 추가합니다.

```
<NotificationConfiguration>
  <TopicConfiguration>
     <Id>optional-id-string</Id>
     <Topic>sns-topic-arn</Topic>
     <Event>event-type</Event>
     <Event>event-type</Event>
      ...
  </TopicConfiguration>
   ...
</NotificationConfiguration>
```

- AWS Lambda 함수 호출 및 이벤트 메시지를 인수로 제공 — 특정 이벤트 유형의 알림 대상으로 Lambda 함수를 설정하려면 `CloudFunctionConfiguration`을 추가합니다.

```
<NotificationConfiguration>
  <CloudFunctionConfiguration>
     <Id>optional-id-string</Id>
```

```
      <Cloudcode>cloud-function-arn</Cloudcode>
      <Event>event-type</Event>
      <Event>event-type</Event>
       ...
  </CloudFunctionConfiguration>
   ...
</NotificationConfiguration>
```

버킷에 구성된 모든 알림을 삭제하려면 알림 하위 리소스에 빈 `<NotificationConfiguration/>` 요소를 저장합니다.

Amazon S3는 특정 유형의 이벤트를 감지하면 이벤트 정보와 함께 메시지를 게시합니다. 자세한 내용은 이벤트 메시지 구조 (p. 603) 단원을 참조하십시오.

이벤트 알림 유형 및 대상

이 단원에서는 Amazon S3에서 지원하는 이벤트 알림 유형과, 알림을 게시할 수 있는 대상 유형에 대해 설명합니다.

지원되는 이벤트 유형

Amazon S3는 다음과 같은 유형의 이벤트를 게시할 수 있습니다. 알림 구성에 이러한 이벤트 유형을 지정합니다.

이벤트 유형	설명
s3:ObjectCreated:* s3:ObjectCreated:Put s3:ObjectCreated:Post s3:ObjectCreated:Copy s3:ObjectCreated:CompleteMultipartUpload	PUT, POST 및 COPY와 같은 Amazon S3 API는 객체를 생성할 수 있습니다. 이러한 이벤트 유형을 사용하면 특정 API를 사용하여 객체가 생성될 때 알림을 활성화할 수 있습니다. 또는 s3:ObjectCreated:* 이벤트 유형을 사용하여 사용된 API에 관계없이 객체가 생성될 때마다 알림을 제공하도록 요청할 수 있습니다. 실패한 작업에서는 이벤트 알림이 수신되지 않습니다.
s3:ObjectRemoved:* s3:ObjectRemoved:Delete s3:ObjectRemoved:DeleteMarkerCreated	ObjectRemoved 이벤트 유형을 사용하여 객체나 객체 그룹이 버킷에서 삭제될 경우 알림을 활성화할 수 있습니다. s3:ObjectRemoved:Delete 이벤트 유형을 사용하여 객체가 삭제되거나 버전이 지정된 객체가 영구적으로 삭제될 경우 알림을 요청할 수 있습니다. 또한 s3:ObjectRemoved:DeleteMarkerCreated를 사용하여 버전이 지정된 객체에 대해 삭제 마커가 생성될 경우 알림을 요청할 수 있습니다. 버전 지정된 객체 삭제에 대한 자세한 내용은 객체 버전 삭제 (p. 481)를 참조하십시오. `s3:ObjectRemoved:*` 와일드카드를 사용하여 객체가 삭제되는 모든 경우에 알림을 요청할 수도 있습니다. 수명 주기 정책에 따른 자동 삭제 또는 작업이 실패한 경우에는 이벤트 알림이 수신되지 않습니다.
s3:ObjectRestore:Post s3:ObjectRestore:Completed	객체 복원 이벤트 유형을 사용하면 S3 Glacier 스토리지 클래스에서 객체를 복원할 때 초기화 및 완료 알림을 받을 수 있습니다.

이벤트 유형	설명
	`s3:ObjectRestore:Post`를 사용하여 객체 복원 시작 알림을 요청합니다. `s3:ObjectRestore:Completed`를 사용하여 복원 완료 알림을 요청합니다.
s3:ReducedRedundancyLostObject	이 이벤트 유형을 사용하여 Amazon S3가 RRS 스토리지 클래스에서 객체가 손실된 것을 확인할 때 Amazon S3가 알림 메시지를 보내도록 요청할 수 있습니다.
s3:Replication:OperationFailedReplication	Amazon S3 Replication Time Control을 사용하여 복제할 수 있는 객체를 복제하지 못한 경우 이 알림 이벤트가 수신됩니다.
s3:Replication:OperationMissedThreshold	Amazon S3 Replication Time Control을 사용하여 복제할 수 있는 객체가 복제에 대한 15분 임계값을 초과하는 경우 이 알림 이벤트가 수신됩니다.
s3:Replication:OperationReplicatedAfterThreshold	Amazon S3 Replication Time Control 기능을 사용하여 복제할 수 있는 객체가 15분 임계값 후 복제된 경우 이 알림 이벤트가 수신됩니다.
s3:Replication:OperationNotTracked	Amazon S3 Replication Time Control을 사용하여 복제할 수 있지만 더 이상 복제 지표로 추적되지 않는 객체에 대해 이 알림 이벤트가 수신됩니다.

지원되는 대상

Amazon S3는 다음과 같은 대상으로 이벤트 알림 메시지를 보낼 수 있습니다. 알림 구성에 이러한 대상의 ARN 값을 지정합니다.

- Amazon Simple Notification Service(Amazon SNS) 주제에 이벤트 메시지 게시
- Amazon Simple Queue Service(Amazon SQS) 대기열에 이벤트 메시지 게시

  Note

  대상 대기열 또는 주제가 SSE를 사용하는 경우 Amazon S3은 메시지 암호화를 활성화하기 위해 연관된 AWS Key Management Service(AWS KMS) 고객 마스터 키(CMK)에 액세스해야 합니다.
- Lambda 함수를 호출하고 이벤트 메시지를 인수로 제공하여 AWS Lambda에 이벤트 메시지 게시

Amazon SNS 주제 또는 Amazon SQS 대기열에 메시지를 게시할 수 있는 권한을 Amazon S3에 부여해야 합니다. 또한 Amazon S3에 사용자를 대신하여 AWS Lambda 함수를 호출할 수 있는 권한도 부여해야 합니다. 이러한 권한 부여에 대한 자세한 내용은 대상에 이벤트 알림 메시지를 게시할 권한 부여 (p. 596)를 참조하십시오.

객체 키 이름 필터링으로 알림 구성

객체 키 이름의 접두사 및 접미사로 알림을 필터링하도록 구성할 수 있습니다. 예를 들어, 파일 이름 확장명이 `".jpg"`인 이미지 파일이 버킷에 추가될 경우에만 알림을 받도록 구성을 설정할 수 있습니다. 또는 접두사가 `"images/"`인 객체가 버킷에 추가될 때에는 Amazon SNS 주제에 알림을 전송하고, 동일한 버킷에서 접두사가 `"logs/"`인 객체에 대한 알림은 AWS Lambda 함수로 전송하도록 구성할 수도 있습니다.

Amazon S3 콘솔에서 그리고 AWS SDK 또는 REST API를 통해 직접 Amazon S3 API를 사용하여 객체 키 이름 필터링을 사용하는 알림 구성을 설정할 수 있습니다. 콘솔 UI를 사용하여 버킷에 대한 알림 구성을 설정하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷에 대한 이벤트 알림을 활성화하고 구성하려면 어떻게 해야 합니까? 단원을 참조하십시오.

이벤트 알림을 활성화하는 방법 (p. 588)에서 설명한 대로, Amazon S3는 버킷과 연결된 알림 하위 리소스에 XML로 알림 구성을 저장합니다. 알림의 규칙을 정의하여 객체 키 이름의 접두사 및/또는 접미사로 알림을 필터링하려면 `Filter` XML 구조를 사용합니다. `Filter` XML 구조에 대한 자세한 내용은 Amazon Simple Storage Service API Reference의 PUT 버킷 알림을 참조하십시오.

`Filter`를 사용하는 알림 구성은 중첩 접두사, 중첩 접미사 또는 접두사 및 접미사 중첩을 포함하는 필터링 규칙을 정의할 수 있습니다. 다음 단원에는 객체 키 이름 필터링을 사용하는 유효한 알림 구성의 예가 나와 있습니다. 또한 접두사/접미사 중첩으로 인해 유효하지 않은 알림 구성 예제가 포함되어 있습니다.

객체 키 이름 필터링을 포함하는 유효한 알림 구성 예제

다음 알림 구성에는 `s3:ObjectCreated:Put` 유형의 이벤트를 게시하기 위해 Amazon S3에 대한 Amazon SQS 대기열을 식별하는 대기열 구성이 포함되어 있습니다. 이 이벤트는 접두사가 `images/`이고 접미사가 `jpg`인 객체가 버킷에 추가(PUT)될 때마다 게시됩니다.

```
<NotificationConfiguration>
  <QueueConfiguration>
      <Id>1</Id>
      <Filter>
          <S3Key>
              <FilterRule>
                  <Name>prefix</Name>
                  <Value>images/</Value>
              </FilterRule>
              <FilterRule>
                  <Name>suffix</Name>
                  <Value>jpg</Value>
              </FilterRule>
          </S3Key>
     </Filter>
     <Queue>arn:aws:sqs:us-west-2:444455556666:s3notificationqueue</Queue>
     <Event>s3:ObjectCreated:Put</Event>
  </QueueConfiguration>
</NotificationConfiguration>
```

다음 알림 구성에는 여러 개의 비중첩 접두사가 있습니다. 이 구성은 `images/` 폴더에서 PUT 요청에 대한 알림을 대기열 A로 보내고, `logs/` 폴더에서 PUT 요청에 대한 알림은 대기열 B로 보내도록 정의합니다.

```
<NotificationConfiguration>
  <QueueConfiguration>
     <Id>1</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>prefix</Name>
                    <Value>images/</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <Queue>arn:aws:sqs:us-west-2:444455556666:sqs-queue-A</Queue>
     <Event>s3:ObjectCreated:Put</Event>
  </QueueConfiguration>
  <QueueConfiguration>
     <Id>2</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>prefix</Name>
                    <Value>logs/</Value>
                </FilterRule>
            </S3Key>
```

```
    </Filter>
    <Queue>arn:aws:sqs:us-west-2:444455556666:sqs-queue-B</Queue>
    <Event>s3:ObjectCreated:Put</Event>
  </QueueConfiguration>
</NotificationConfiguration>
```

다음 알림 구성에는 여러 개의 비중첩 접미사가 있습니다. 이 구성은 버킷에 새로 추가된 모든 `.jpg` 이미지를 Lambda 클라우드 함수 A에서 처리하고, 새로 추가된 모든 `.png` 이미지는 클라우드 함수 B에서 처리하도록 구성합니다. 접미사 `.png`와 `.jpg`는 마지막 문자가 같기는 하지만 중첩되지는 않습니다. 한 문자열이 두 접미사로 끝날 수 있으면 두 개의 접미사가 중첩되는 것으로 간주됩니다. 한 문자열이 `.png`와 `.jpg`로 끝날 수 없으므로 예제 구성의 접미사는 중첩 접미사가 아닙니다.

```
<NotificationConfiguration>
  <CloudFunctionConfiguration>
     <Id>1</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.jpg</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <Cloudcode>arn:aws:lambda:us-west-2:444455556666:cloud-function-A</Cloudcode>
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
  <CloudFunctionConfiguration>
     <Id>2</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.png</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <Cloudcode>arn:aws:lambda:us-west-2:444455556666:cloud-function-B</Cloudcode>
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
</NotificationConfiguration>
```

`Filter`를 사용하는 알림 구성은 동일한 이벤트 유형에 대해 중첩 접두사를 포함하는 필터링 규칙을 정의할 수 없습니다. 단 중첩 접두사가 중첩되지 않는 접미사와 함께 사용될 경우에는 가능합니다. 다음 예제 구성은 일반 접두사를 갖지만 접미사가 중첩되지 않게 생성된 객체가 어떻게 다른 대상으로 전송될 수 있는지를 보여 줍니다.

```
<NotificationConfiguration>
  <CloudFunctionConfiguration>
     <Id>1</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>prefix</Name>
                    <Value>images</Value>
                </FilterRule>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.jpg</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <Cloudcode>arn:aws:lambda:us-west-2:444455556666:cloud-function-A</Cloudcode>
```

```
    <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
  <CloudFunctionConfiguration>
    <Id>2</Id>
    <Filter>
           <S3Key>
               <FilterRule>
                   <Name>prefix</Name>
                   <Value>images</Value>
               </FilterRule>
               <FilterRule>
                   <Name>suffix</Name>
                   <Value>.png</Value>
               </FilterRule>
           </S3Key>
    </Filter>
    <Cloudcode>arn:aws:lambda:us-west-2:444455556666:cloud-function-B</Cloudcode>
    <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
</NotificationConfiguration>
```

유효하지 않은 접두사/접미사 중첩을 포함하는 알림 구성 예제

`Filter`를 사용하는 알림 구성은 대부분의 경우 동일한 이벤트 유형에 대해 중첩 접두사, 중첩 접미사 또는 접두사와 접미사의 중첩 조합을 포함하는 필터링 규칙을 정의할 수 없습니다. (접미사가 중첩되지 않을 경우에 한해 중첩 접두사를 사용할 수 있습니다. 관련 예제는 객체 키 이름 필터링으로 알림 구성 (p. 591).)

이벤트 유형이 서로 다른 중첩 객체 키 이름 필터를 사용할 수 있습니다. 예를 들어, `image/` 이벤트 유형에 대해 `ObjectCreated:Put` 접두사를 사용하고, `image/` 이벤트 유형에 대해 `ObjectDeleted:*` 접두사를 사용하는 알림 구성을 만들 수 있습니다.

Amazon S3 콘솔이나 API를 사용할 경우 동일한 이벤트 유형에 대해 유효하지 않은 중첩 이름 필터를 가진 알림 구성을 저장하려고 하면 오류가 발생합니다. 이 단원에서는 중첩된 이름 필터로 인해 유효하지 않은 알림 구성의 예를 보여줍니다.

기존의 알림 구성 규칙이 다른 접두사 및 접미사와 각각 일치하는 기본 접두사와 접미사를 갖는다고 가정해 보겠습니다. 다음 알림 구성은 중첩 접두사가 있기 때문에 유효하지 않습니다. 루트 접두사가 다른 접두사와 중첩됩니다. (이 예제에서 접두사 대신 접미사를 사용할 경우에도 마찬가지입니다. 루트 접미사가 다른 접미사와 중첩됩니다.)

```
<NotificationConfiguration>
     <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-notification-one</Topic>
         <Event>s3:ObjectCreated:*</Event>
    </TopicConfiguration>
    <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-notification-two</Topic>
         <Event>s3:ObjectCreated:*</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>prefix</Name>
                     <Value>images</Value>
                 </FilterRule>
             </S3Key>
        </Filter>
    </TopicConfiguration>
</NotificationConfiguration>
```

다음 알림 구성은 중첩 접미사가 있기 때문에 유효하지 않습니다. 한 문자열이 두 접미사로 끝날 수 있으면 두 개의 접미사가 중첩되는 것으로 간주됩니다. 문자열이 `jpg`와 `pg`로 끝날 수 있으므로 접미사가 중첩됩니다. (접두사의 경우에도 마찬가지로, 한 문자열이 두 접두사로 시작할 수 있을 경우 두 개의 접두사가 중첩되는 것으로 간주됩니다.)

```
 <NotificationConfiguration>
     <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-one</Topic>
         <Event>s3:ObjectCreated:*</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>suffix</Name>
                     <Value>jpg</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
    <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-two</Topic>
         <Event>s3:ObjectCreated:Put</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>suffix</Name>
                     <Value>pg</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
</NotificationConfiguration
```

다음 알림 구성은 접두사와 접미사가 중첩되기 때문에 유효하지 않습니다.

```
<NotificationConfiguration>
     <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-one</Topic>
         <Event>s3:ObjectCreated:*</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>prefix</Name>
                     <Value>images</Value>
                 </FilterRule>
                 <FilterRule>
                     <Name>suffix</Name>
                     <Value>jpg</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
    <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-two</Topic>
         <Event>s3:ObjectCreated:Put</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>suffix</Name>
                     <Value>jpg</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
```

```
</NotificationConfiguration>
```

대상에 이벤트 알림 메시지를 게시할 권한 부여

Amazon S3에서 대상으로 메시지를 게시할 수 있으려면 먼저 Amazon S3 보안 주체에 관련 API를 호출하여 SNS 주제, SQS 대기열 또는 Lambda 함수로 메시지를 게시할 수 있는 권한을 부여해야 합니다.

AWS Lambda 함수 호출 권한 부여

Amazon S3는 Lambda 함수를 호출하고 그 인수로 이벤트 메시지를 제공하여 AWS Lambda에 이벤트 메시지를 게시합니다.

Amazon S3 콘솔을 사용하여 Amazon S3 버킷에 Lambda 함수에 대한 이벤트 알림을 구성할 때 콘솔은 Amazon S3에 버킷에서 함수를 호출하기 위한 권한을 부여하도록 Lambda 함수에 필수 권한을 설정합니다. 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷에 대한 이벤트 알림을 활성화하고 구성하려면 어떻게 해야 합니까? 단원을 참조하십시오.

또한 Amazon S3에 Lambda 함수를 호출할 AWS Lambda의 권한도 부여할 수 있습니다. 자세한 내용은 AWS Lambda Developer Guide에서 Tutorial: Using the AWS Lambda with Amazon S3 단원을 참조하십시오.

SNS 주제 또는 SQS 대기열로 메시지를 게시할 권한 부여

Amazon S3에 SNS 주제 또는 SQS 대기열에 메시지를 게시할 수 있는 권한을 부여하려면 대상 SNS 주제 또는 SQS 대기열에 AWS Identity and Access Management(IAM) 정책을 연결합니다.

SNS 주제 또는 SQS 대기열에 정책을 연결하는 방법에 대한 예제는 예제 안내: 버킷의 알림 구성(메시지 대상: SNS 주제 및 SQS 대기열) (p. 598) 단원을 참조하십시오. 권한에 대한 자세한 내용은 다음 단원을 참조하십시오.

- Amazon Simple Notification Service 개발자 안내서의 Example Cases for Amazon SNS Access Control
- Amazon Simple Queue Service 개발자 안내서의 Access Control Using AWS Identity and Access Management (IAM)

대상 SNS 주제에 대한 IAM 정책

다음은 대상 SNS 주제에 연결하는 IAM 정책의 예제입니다.

```
{
 "Version": "2008-10-17",
 "Id": "example-ID",
 "Statement": [
  {
   "Sid": "example-statement-ID",
   "Effect": "Allow",
   "Principal": {
     "Service": "s3.amazonaws.com"
   },
   "Action": [
    "SNS:Publish"
   ],
   "Resource": "arn:aws:sns:REGION:ACCOUNT-ID:TOPICNAME",
```

```
    "Condition": {
       "ArnLike": { "aws:SourceArn": "arn:aws:s3:::bucket-name" },
       "StringEquals": { "aws:SourceAccount": "bucket-owner-account-id" }
    }
   }
  ]
}
```

대상 SQS 대기열에 대한 IAM 정책

다음은 대상 SQS 대기열에 연결하는 IAM 정책의 예제입니다.

```
{
 "Version": "2008-10-17",
 "Id": "example-ID",
 "Statement": [
  {
   "Sid": "example-statement-ID",
   "Effect": "Allow",
   "Principal": {
     "Service": "s3.amazonaws.com"
   },
   "Action": [
    "SQS:SendMessage"
   ],
   "Resource": "arn:aws:sqs:REGION:ACCOUNT-ID:QUEUENAMEHERE",
   "Condition": {
      "ArnLike": { "aws:SourceArn": "arn:aws:s3:*:*:bucket-name" }
   }
  }
 ]
}
```

Amazon SNS 및 Amazon SQS IAM 정책 모두 `ArnLike` 조건 대신, `StringLike` 조건을 정책에 지정할 수 있습니다.

```
"Condition": {
  "StringLike": { "aws:SourceArn": "arn:aws:s3:*:*:bucket-name" }
  }
```

AWS KMS 키 정책

SQS 대기열 또는 SNS 주제가 AWS Key Management Service(AWS KMS) 고객 관리형 고객 마스터 키(CMK)로 암호화되는 경우 Amazon S3 서비스 보안 주체에게 암호화된 주제 및/또는 대기열을 사용할 수 있는 권한을 부여해야 합니다. Amazon S3 서비스 보안 주체에게 권한을 부여하려면 고객 관리형 CMK의 주요 정책에 다음 설명을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
```

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

AWS KMS 키 정책에 대한 자세한 내용은 AWS Key Management Service Developer Guide의 AWS KMS에서 키 정책 사용을 참조하십시오. Amazon SQS 및 Amazon SNS에서 AWS KMS와 함께 서버 측 암호화를 사용하는 방법에 대한 자세한 내용은 다음을 참조하십시오.

- Amazon Simple Notification Service 개발자 안내서의 Amazon SNS에 대한 AWS KMS 권한 구성.
- Amazon Simple Queue Service 개발자 안내서의 Amazon SQS에 대한 AWS KMS 권한 구성.
- AWS 컴퓨팅 블로그의 AWS KMS를 활용하여 Amazon SNS에 게시된 메시지 암호화

예제 안내: 버킷의 알림 구성(메시지 대상: SNS 주제 및 SQS 대기열)

주제

- 안내 요약 (p. 598)
- 1단계: Amazon SNS 주제 만들기 (p. 599)
- 2단계: Amazon SQS 대기열 만들기 (p. 599)
- 3단계: 버킷에 알림 구성 추가 (p. 600)
- 4단계: 설정 테스트 (p. 603)

안내 요약

이 예제에서는 Amazon S3가 다음을 수행하도록 요청하는 알림 구성을 버킷에 추가합니다.

- Amazon SQS 대기열에 `s3:ObjectCreated:*` 유형의 이벤트를 게시합니다.
- `s3:ReducedRedundancyLostObject` 주제에 Amazon SNS 유형의 이벤트를 게시합니다.

알림 구성에 대한 자세한 내용은 Amazon S3 이벤트 알림 구성 (p. 587) 단원을 참조하십시오.

콘솔을 사용하면 코드를 생성하지 않고도 이러한 모든 단계를 수행할 수 있습니다. 또한 Java 및 .NET용 AWS SDK를 사용하는 코드 예제가 제공되므로 프로그래밍 방식으로 알림 구성을 추가할 수 있습니다.

이 안내에서는 다음을 수행합니다.

1. Amazon SNS 주제를 만듭니다.

   Amazon SNS 콘솔을 사용하여 SNS 주제를 만들고 이 주제를 구독하면 해당 주제에 게시된 모든 이벤트가 전송됩니다. 통신 프로토콜로 이메일을 지정합니다. 주제를 만든 후 Amazon SNS에서 이메일을 보냅니다. 전자 메일에 포함된 링크를 클릭하여 주제 구독을 확인해야 합니다.

   주제에 액세스 정책을 연결하여 Amazon S3에 메시지를 게시할 수 있는 권한을 부여합니다.
2. Amazon SQS 대기열을 생성합니다.

   Amazon SQS 콘솔을 사용하여 SQS 대기열을 만듭니다. 프로그래밍 방식으로 Amazon S3가 대기열로 전송하는 모든 메시지에 액세스할 수 있습니다. 그러나 이 안내에서는 콘솔에서 알림 메시지를 확인합니다.

   주제에 액세스 정책을 연결하여 Amazon S3에 메시지를 게시할 수 있는 권한을 부여합니다.

3. 버킷에 알림 구성을 추가합니다.

1단계: Amazon SNS 주제 만들기

다음 단계에 따라 Amazon Simple Notification Service(Amazon SNS) 주제를 만들고 구독합니다.

1. Amazon SNS 콘솔을 사용하여 주제를 만듭니다. 지침은 Amazon Simple Notification Service 개발자 안내서의 주제 생성을 참조하십시오.
2. 주제를 구독합니다. 이 실습에서는 통신 프로토콜로 전자 메일을 사용합니다. 지침은 Amazon Simple Notification Service 개발자 안내서의 주제 구독을 참조하십시오.

   주제 구독을 확인하도록 요청하는 전자 메일이 전송되면 구독을 확인합니다.
3. 주제에 연결된 액세스 정책을 다음 정책으로 교체합니다. SNS 주제 Amazon 리소스 이름(ARN) 및 버킷 이름을 제공하여 정책을 업데이트해야 합니다.

```
{
 "Version": "2008-10-17",
 "Id": "example-ID",
 "Statement": [
  {
   "Sid": "example-statement-ID",
   "Effect": "Allow",
   "Principal": {
    "AWS":"*"
   },
   "Action": [
    "SNS:Publish"
   ],
   "Resource": "SNS-topic-ARN",
   "Condition": {
      "ArnLike": { "aws:SourceArn": "arn:aws:s3:*:*:bucket-name" }
   }
  }
 ]
}
```

4. 주제 ARN을 기록합니다.

   생성된 SNS 주제는 AWS 계정의 리소스이며 따라서 고유한 Amazon 리소스 이름(ARN)을 가집니다. 다음 단계에서 이 ARN을 사용합니다. ARN의 형식은 다음과 같습니다.

```
arn:aws:sns:aws-region:account-id:topic-name
```

2단계: Amazon SQS 대기열 만들기

다음 단계에 따라 Amazon Simple Queue Service(Amazon SQS) 대기열을 만들고 구독합니다.

1. Amazon SQS 콘솔을 사용하여 대기열을 만듭니다. 자세한 방법은 Amazon Simple Queue Service 개발자 안내서의 Amazon SQS 시작하기를 참조하십시오.
2. 대기열에 연결된 액세스 정책을 다음 정책으로 교체합니다. Amazon SQS 콘솔에서 대기열을 선택하고 권한 탭에서 정책 문서 편집(고급)을 선택합니다.

```
{
 "Version": "2012-10-17",
 "Id": "example-ID",
 "Statement": [
```

```
  {
   "Sid": "example-statement-ID",
   "Effect": "Allow",
   "Principal": {
    "AWS":"*"
   },
   "Action": [
    "SQS:SendMessage"
   ],
   "Resource": "SQS-queue-ARN",
   "Condition": {
      "ArnLike": { "aws:SourceArn": "arn:aws:s3:*:*:bucket-name" }
   }
  }
 ]
}
```

3. (선택 사항) AWS Key Management Service(AWS KMS)를 사용하여 Amazon SQS 대기열 또는 Amazon SNS 주제에 서버 측 암호화(SSE)가 활성화되어 있는 경우 연결된 대칭 고객 관리형 AWS KMS CMK에 다음 정책을 추가합니다.

   Amazon SQS 또는 Amazon SNS용 AWS 관리형 CMK를 수정할 수 없으므로 고객 관리형 CMK에 정책을 추가해야 합니다.

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

   AWS KMS에서 Amazon SQS 및 Amazon SNS에 대한 자세한 내용은 다음을 참조하십시오.

   - Amazon Simple Notification Service 개발자 안내서의 Amazon SNS에 대한 AWS KMS 권한 구성.
   - Amazon Simple Queue Service 개발자 안내서의 Amazon SQS에 대한 AWS KMS 권한 구성.

4. 대기열 ARN을 기록합니다.

   생성된 SQS 대기열은 AWS 계정의 리소스이며 따라서 고유한 Amazon 리소스 이름(ARN)을 가집니다. 다음 단계에서 이 ARN을 사용합니다. ARN의 형식은 다음과 같습니다.

```
arn:aws:sqs:aws-region:account-id:queue-name
```

3단계: 버킷에 알림 구성 추가

Amazon S3 콘솔 또는 AWS SDK를 사용한 프로그래밍 방식으로 버킷 알림을 활성화할 수 있습니다. 다음 옵션 중 하나를 선택하여 버킷에 알림을 구성합니다. 이 단원에서는 Java 및 .NET용 AWS SDK를 사용한 코드 예제를 제공합니다.

3단계(옵션 a): 콘솔을 사용하여 버킷에 대한 알림 활성화

Amazon S3 콘솔을 사용하여 Amazon S3에서 다음을 수행하도록 요청하는 알림 구성을 추가합니다.

- Amazon SQS 대기열에 `s3:ObjectCreated:*` 유형의 이벤트를 게시합니다.
- Amazon SNS 주제에 `s3:ReducedRedundancyLostObject` 유형의 이벤트를 게시합니다.

알림 구성을 저장하면 Amazon S3가 테스트 메시지를 게시하고 이메일을 통해 이 테스트 메시지가 전송됩니다.

지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷에 대한 이벤트 알림을 활성화하고 구성하려면 어떻게 해야 합니까? 단원을 참조하십시오.

3단계(옵션 b): .NET용 AWS SDK를 사용하여 버킷에 대한 알림 활성화

다음 C# 코드 예제는 버킷에 알림 구성을 추가하는 전체 코드를 제공합니다. 이 코드를 업데이트하여 버킷 이름과 SNS 주제 ARN을 제공해야 합니다. 실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

Example

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

namespace Amazon.DocSamples.S3
{
    class EnableNotificationsTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string snsTopic = "*** SNS topic ARN ***";
        private const string sqsQueue = "*** SQS topic ARN ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            EnableNotificationAsync().Wait();
        }

        static async Task EnableNotificationAsync()
        {
            try
            {
               PutBucketNotificationRequest request = new PutBucketNotificationRequest
                {
                    BucketName = bucketName
                };

                TopicConfiguration c = new TopicConfiguration
                {
                    Events = new List<EventType> { EventType.ObjectCreatedCopy },
                    Topic = snsTopic
                };
                request.TopicConfigurations = new List<TopicConfiguration>();
```

```
                request.TopicConfigurations.Add(c);
                request.QueueConfigurations = new List<QueueConfiguration>();
                request.QueueConfigurations.Add(new QueueConfiguration()
                {
                    Events = new List<EventType> { EventType.ObjectCreatedPut },
                    Queue = sqsQueue
                });

                PutBucketNotificationResponse response = await
 client.PutBucketNotificationAsync(request);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' ",
 e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown error encountered on server. Message:'{0}' ",
 e.Message);
            }
        }
    }
}
```

3단계(옵션 c): AWS SDK for Java를 사용하여 버킷에 대한 알림 활성화

다음 예제에서는 버킷에 알림 구성을 추가하는 방법을 보여줍니다. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

Example

```
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;
import com.amazonaws.services.s3.model.*;

import java.io.IOException;
import java.util.EnumSet;

public class EnableNotificationOnABucket {

    public static void main(String[] args) throws IOException {
        String bucketName = "*** Bucket name ***";
        Regions clientRegion = Regions.DEFAULT_REGION;
        String snsTopicARN = "*** SNS Topic ARN ***";
        String sqsQueueARN = "*** SQS Queue ARN ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();
            BucketNotificationConfiguration notificationConfiguration = new
 BucketNotificationConfiguration();

            // Add an SNS topic notification.
            notificationConfiguration.addConfiguration("snsTopicConfig",
```

```
                    new TopicConfiguration(snsTopicARN,
 EnumSet.of(S3Event.ObjectCreated)));

            // Add an SQS queue notification.
            notificationConfiguration.addConfiguration("sqsQueueConfig",
                    new QueueConfiguration(sqsQueueARN,
 EnumSet.of(S3Event.ObjectCreated)));

            // Create the notification configuration request and set the bucket
 notification configuration.
            SetBucketNotificationConfigurationRequest request = new
 SetBucketNotificationConfigurationRequest(
                    bucketName, notificationConfiguration);
            s3Client.setBucketNotificationConfiguration(request);
        } 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();
        }
    }
}
```

4단계: 설정 테스트

이제 버킷에 객체를 업로드한 후 Amazon SQS 콘솔에서 이벤트 알림을 확인하여 설정을 테스트할 수 있습니다. 자세한 방법은 Amazon Simple Queue Service 개발자 안내서 "시작하기" 단원의 메시지 수신 단원을 참조하십시오.

이벤트 메시지 구조

이벤트를 게시하기 위해 Amazon S3가 전송하는 알림 메시지는 JSON 형식입니다. 다음 예제는 JSON 메시지의 구조를 보여 줍니다.

이 예에 대해 다음을 참조하십시오.

- `eventVersion` 키 값에는 `<major>.<minor>` 형식의 메이저 및 마이너 버전이 포함됩니다.

  Amazon S3가 이전 버전과 호환되지 않는 이벤트 구조를 변경하는 경우 메이저 버전이 증가합니다. 이에는 이미 존재하는 JSON 필드를 제거하거나 필드의 콘텐츠가 표시되는 방식(예: 날짜 형식)을 변경하는 것이 포함됩니다.

  Amazon S3가 이벤트 구조에 새 필드를 추가하는 경우 마이너 버전이 증가합니다. 일부 또는 모든 기존 이벤트에 새로운 정보가 공급되거나 완전히 새롭게 도입된 이벤트 유형에 새로운 정보가 공급되는 경우 이러한 경우가 발생할 수 있습니다. 애플리케이션은 새로운 필드를 무시하여 이벤트 구조의 새로운 마이너 버전의 이후 버전과 호환성을 유지해야 합니다.

  새로운 이벤트 유형이 도입되었지만 그와 달리 이벤트 구조는 수정되지 않는 경우, 이벤트 버전이 변경되지 않습니다.

  애플리케이션이 이벤트 구조의 구문을 올바로 분석할 수 있도록 메이저 버전 번호에 대해 같은 값인지 비교하는 것이 좋습니다. 애플리케이션이 예상하는 필드가 반드시 존재할 수 있도록 마이너 버전에 대해 크거나 같은 값인지 비교하는 것이 좋습니다.
- AWS Support의 도움을 받아 요청을 추적하려는 경우 `responseElements` 키 값을 유용하게 사용할 수 있습니다. `x-amz-request-id` 및 `x-amz-id-2` 모두 Amazon S3가 개별 요청을 추적하는 데 도움이 됩

니다. 이러한 값은 이벤트를 시작한 요청에 대한 응답에서 Amazon S3가 반환하는 값과 동일하므로 이벤트를 요청과 일치시키는 데 사용할 수 있습니다.

- `s3` 키는 이벤트와 연관된 버킷 및 객체에 대한 정보를 제공합니다. 객체의 키 이름 값은 URL로 인코딩되어 있습니다. 예를 들어 "red flower.jpg"는 "red+flower.jpg"가 됩니다(Amazon S3는 응답의 콘텐츠 유형으로 "`application/x-www-form-urlencoded`"를 반환함).
- `sequencer` 키는 이벤트 시퀀스를 정의하는 방법을 제공합니다. 이벤트 알림은 반드시 이벤트가 발생한 순서로 도착하지는 않습니다. 하지만 객체를 생성하고(`PUT`) 객체를 삭제하는 이벤트에는 `sequencer`가 포함되며, 이는 특정 객체 키에 대한 이벤트의 순서를 정의하는 데 사용할 수 있습니다.

  `sequencer` 문자열을 동일한 객체 키의 두 가지 이벤트 알림과 비교할 경우 `sequencer` 16진수 값이 더 큰 이벤트 알림이 나중에 발생한 이벤트입니다. 이벤트 알림을 사용하여 별도의 데이터베이스 또는 Amazon S3 객체의 인덱스를 관리할 경우 각 이벤트 알림을 처리할 때 `sequencer` 값을 비교하고 저장할 수 있습니다.

  다음을 참조하십시오.

  - `sequencer`는 다른 객체 키의 이벤트에 대한 순서를 정의하는 데 사용할 수 없습니다.
  - 시퀀서는 길이가 서로 다를 수 있습니다. 따라서 이러한 값을 비교하려면 먼저 좌측 패드를 통해 짧은 값을 0으로 채운 후 사전식 순서 비교를 수행합니다.
- `glacierEventData` 키는 `s3:ObjectRestore:Completed` 이벤트에 대해서만 표시됩니다.
- `restoreEventData` 키에는 복원 요청과 관련된 속성이 포함되어 있습니다.
- `replicationEventData` 키는 복제 이벤트에 대해서만 표시됩니다.

다음 예제는 2.2 버전의 이벤트 메시지 JSON 구조(현재 Amazon S3에서 사용 중인 버전)를 보여줍니다.

```
{
   "Records":[
      {
         "eventVersion":"2.2",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":The time, in ISO-8601 format, for example, 1970-01-01T00:00:00.000Z,
 when Amazon S3 finished processing the request,
         "eventName":"event-type",
         "userIdentity":{
            "principalId":"Amazon-customer-ID-of-the-user-who-caused-the-event"
         },
         "requestParameters":{
            "sourceIPAddress":"ip-address-where-request-came-from"
         },
         "responseElements":{
            "x-amz-request-id":"Amazon S3 generated request ID",
            "x-amz-id-2":"Amazon S3 host that processed the request"
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"ID found in the bucket notification configuration",
            "bucket":{
               "name":"bucket-name",
               "ownerIdentity":{
                  "principalId":"Amazon-customer-ID-of-the-bucket-owner"
               },
               "arn":"bucket-ARN"
            },
            "object":{
               "key":"object-key",
               "size":object-size,
               "eTag":"object eTag",
               "versionId":"object version if bucket is versioning-enabled, otherwise
 null",
```

```
                "sequencer": "a string representation of a hexadecimal value used to
 determine event sequence,
                    only used with PUTs and DELETEs"
            }
        },
        "glacierEventData": {
            "restoreEventData": {
                "lifecycleRestorationExpiryTime": "The time, in ISO-8601 format, for
 example, 1970-01-01T00:00:00.000Z, of Restore Expiry",
                "lifecycleRestoreStorageClass": "Source storage class for restore"
            }
        }
    }
  ]
}
```

다음 예제는 2.0 버전의 이벤트 메시지 구조(더 이상 Amazon S3에서 사용하지 않는 버전)를 보여 줍니다.

```
{
   "Records":[
      {
         "eventVersion":"2.0",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":The time, in ISO-8601 format, for example, 1970-01-01T00:00:00.000Z,
 when S3 finished processing the request,
         "eventName":"event-type",
         "userIdentity":{
            "principalId":"Amazon-customer-ID-of-the-user-who-caused-the-event"
         },
         "requestParameters":{
            "sourceIPAddress":"ip-address-where-request-came-from"
         },
         "responseElements":{
            "x-amz-request-id":"Amazon S3 generated request ID",
            "x-amz-id-2":"Amazon S3 host that processed the request"
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"ID found in the bucket notification configuration",
            "bucket":{
               "name":"bucket-name",
               "ownerIdentity":{
                  "principalId":"Amazon-customer-ID-of-the-bucket-owner"
               },
               "arn":"bucket-ARN"
            },
            "object":{
               "key":"object-key",
               "size":object-size,
               "eTag":"object eTag",
               "versionId":"object version if bucket is versioning-enabled, otherwise
 null",
               "sequencer": "a string representation of a hexadecimal value used to
 determine event sequence,
                    only used with PUTs and DELETEs"
            }
         }
      }
   ]
}
```

다음은 예제 메시지입니다.

- 테스트 메시지 — 버킷에 이벤트 알림을 구성하면 Amazon S3가 다음과 같은 테스트 메시지를 전송합니다.

```
{
   "Service":"Amazon S3",
   "Event":"s3:TestEvent",
   "Time":"2014-10-13T15:57:02.089Z",
   "Bucket":"bucketname",
   "RequestId":"5582815E1AEA5ADF",
   "HostId":"8cLeGAmw098X5cv4Zkwcmo8vvZa3eH3eKxsPzbB9wrR+YstdA6Knx4Ip8EXAMPLE"
}
```

- PUT 요청을 사용하여 생성된 객체에 대한 예제 메시지 — 다음 메시지는 Amazon S3가 `s3:ObjectCreated:Put` 이벤트를 게시하기 위해 전송하는 메시지 예제입니다.

```
{
   "Records":[
      {
         "eventVersion":"2.1",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"1970-01-01T00:00:00.000Z",
         "eventName":"ObjectCreated:Put",
         "userIdentity":{
            "principalId":"AIDAJDPLRKLG7UEXAMPLE"
         },
         "requestParameters":{
            "sourceIPAddress":"127.0.0.1"
         },
         "responseElements":{
            "x-amz-request-id":"C3D13FE58DE4C810",
            "x-amz-id-2":"FMyUVURIY8/IgAtTv8xRjskZQpcIZ9KG4V5Wp6S7S/
JRWeUWerMUE5JgHvANOjpD"
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"testConfigRule",
            "bucket":{
               "name":"mybucket",
               "ownerIdentity":{
                  "principalId":"A3NL1KOZZKExample"
               },
               "arn":"arn:aws:s3:::mybucket"
            },
            "object":{
               "key":"HappyFace.jpg",
               "size":1024,
               "eTag":"d41d8cd98f00b204e9800998ecf8427e",
               "versionId":"096fKKXTRTtl3on89fVO.nfljtsv6qko",
               "sequencer":"0055AED6DCD90281E5"
            }
         }
      }
   ]
}
```

각 IAM ID 접두사(AIDA, AROA, AGPA 등)의 정의는 고유 ID 접두사에 대한 이해를 참조하십시오.

복제

복제는 Amazon S3 버킷 간에 객체를 비동기식으로 자동 복제할 수 있게 합니다. 객체 복제를 위해 구성된 버킷은 동일한 AWS 계정 또는 다른 계정이 소유할 수 있습니다. 다른 AWS 리전 간에, 또는 동일한 리전 내에서 객체를 복사할 수 있습니다.

객체 복제를 활성화하려면 원본 버킷에 복제 구성을 추가합니다. 최소 구성에서는 다음을 제공합니다.

- Amazon S3가 객체를 복제할 대상 버킷
- Amazon S3가 사용자 대신 객체를 복제하기 위해 수임할 수 있는 AWS Identity and Access Management(IAM) 역할

추가 구성 옵션을 사용할 수 있습니다. 자세한 내용은 추가 복제 구성 (p. 623) 단원을 참조하십시오.

객체 복제 유형

다른 AWS 리전 간에, 또는 동일한 AWS 리전 내에서 객체를 복사할 수 있습니다.

- 교차 리전 복제(CRR)는 서로 다른 AWS 리전의 Amazon S3 버킷에서 객체를 복사하는 데 사용됩니다.
- 동일 리전 복제(SRR)는 같은 AWS 리전의 Amazon S3 버킷에서 객체를 복사하는 데 사용됩니다.

복제를 사용하는 이유

복제는 다음을 지원합니다.

- 메타데이터를 유지하면서 객체 복제 — 복제를 사용하여 원본 객체 생성 시간 및 버전 ID와 같은 모든 메타데이터를 유지하는 객체의 복사본을 생성할 수 있습니다. 이 기능은 복제본이 원본 객체와 동일한지 확인해야 하는 경우에 중요합니다.

- 서로 다른 스토리지 클래스로 객체 복제 — 복제를 사용하여 객체를 S3 Glacier, S3 Glacier Deep Archive 또는 대상 버킷의 다른 스토리지 클래스에 직접 배치할 수 있습니다. 또한 데이터를 동일한 스토리지 클래스에 복제하고 대상 버킷에 대한 수명 주기 정책을 사용하여 객체를 오래된 스토리지 클래스로 이동시킬 수 있습니다.

- 다른 소유권으로 객체 복사본 유지 — 원본 객체의 소유자에 상관없이 대상 버킷을 소유한 AWS 계정으로 복제본 소유권을 변경하도록 Amazon S3에 지시할 수 있습니다. 이를 사용자 재정의 옵션이라고 합니다. 이 옵션을 사용하여 객체 복제본에 대한 액세스를 제한할 수 있습니다.

- 15분 이내에 객체 복제 — S3 Replication Time Control (S3 RTC)를 사용하여 예측 가능한 기간 내에 동일한 AWS 리전에 또는 여러 리전 간에 데이터를 복제할 수 있습니다. S3 RTC는 15분 내에 Amazon S3에 저장된 새 객체의 99.99%를 복제합니다(서비스 수준 계약에 따라 지원). 자세한 내용은 the section called “S3 Replication Time Control (S3 RTC)” (p. 624) 단원을 참조하십시오.

CRR을 사용해야 하는 경우

교차 리전 복제는 다음을 지원합니다.

- 규정 준수 요구 사항 준수 — 기본적으로 Amazon S3는 지리적으로 동떨어진 여러 가용 영역 간에 데이터를 저장하지만, 규정 준수 요구 사항에 따라 훨씬 더 먼 거리에 데이터를 저장하게 지정할 수 있습니다. 교차 리전 복제를 사용하면 먼 거리의 AWS 리전 간에 데이터를 복제하여 이러한 요구 사항을 충족할 수 있습니다.

- 지연 시간 최소화 — 고객이 두 군데의 지리적 위치를 갖는 경우 사용자와 지리적으로 더 가까운 AWS 리전에 객체 복사본을 유지하여 객체 액세스 지연 시간을 최소화할 수 있습니다.

- 운영 효율성 증가 — 두 AWS 리전에 동일한 객체 집합을 분석하는 컴퓨팅 클러스터가 있는 경우, 해당 리전에 객체 복사본을 유지할 수 있습니다.

SRR을 사용해야 하는 경우

동일 리전 복제는 다음을 지원합니다.

- 단일 버킷에 로그 집계 — 여러 버킷에 또는 여러 계정 간에 로그를 저장하는 경우, 리전 내 단일 버킷으로 로그를 손쉽게 복제할 수 있습니다. 이렇게 하면 단일 위치에서 로그를 보다 간편하게 처리할 수 있습니다.

- 프로덕션 계정과 테스트 계정 간에 라이브 복제 구성 — 사용자 또는 사용자의 고객이 동일한 데이터를 사용하는 프로덕션 계정과 테스트 계정을 가지고 있는 경우, SRR 규칙을 실행하여 객체 메타데이터를 유지한 상태로 여러 계정 간에 객체를 복제할 수 있습니다.

- 데이터 주권 법률 준수 — 특정 리전 내의 개별 AWS 계정에 데이터의 여러 복사본을 저장해야 할 수 있습니다. 동일 리전 복제를 사용하면 규정 준수 규정으로 인해 데이터가 해당 국가를 벗어날 수 없을 때 중요 데이터를 자동으로 복제할 수 있습니다.

복제 요구 사항

복제 요구 사항은 다음과 같습니다.

- 원본 버킷 소유자는 자신의 계정에 대해 원본 및 대상 AWS 리전을 활성화해야 합니다. 대상 버킷 소유자는 자신의 계정에 대해 대상 리전을 활성화해야 합니다. AWS 리전 활성화 또는 비활성화에 대한 자세한 내용은 AWS General Reference의 AWS 서비스 엔드포인트를 참조하십시오.
- 원본 버킷과 대상 버킷 모두에서 버전 관리를 활성화해야 합니다.
- Amazon S3가 사용자를 대신해서 대상 버킷에 해당 원본 버킷의 객체를 복제할 권한을 가지고 있어야 합니다.
- 원본 버킷 소유자가 버킷 내 객체를 소유하지 않은 경우 객체 소유자가 객체 액세스 제어 목록(ACL)을 사용하여 버킷 소유자에게 `READ` 및 `READ_ACP` 권한을 부여해야 합니다. 자세한 내용은 ACL을 사용한 액세스 관리 (p. 437) 단원을 참조하십시오.
- 원본 버킷에 Amazon S3 객체 잠금이 활성화되어 있는 경우, 대상 버킷에도 객체 잠금이 활성화되어 있어야 합니다. 자세한 내용은 Amazon S3 객체 잠금을 사용하여 객체 잠금 (p. 490) 단원을 참조하십시오.

  객체 잠금이 활성화되어 있는 버킷에서 복제를 활성화하려면 AWS Support에 문의하십시오.

자세한 내용은 복제 설정 개요 (p. 611) 단원을 참조하십시오.

원본 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유하는 교차 계정 시나리오에서 복제 구성을 설정하는 경우, 다음과 같은 추가 요구 사항이 적용됩니다.

- 대상 버킷 소유자가 버킷 정책을 사용하여 원본 버킷 소유자에게 객체를 복제할 수 있는 권한을 부여해야 합니다. 자세한 내용은 원본 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유할 경우 권한 부여 (p. 622) 단원을 참조하십시오.

Amazon S3가 복제하는 객체

Amazon S3는 버킷에서 복제가 구성된 특정 항목만 복제합니다.

복제 가능한 객체

기본적으로 Amazon S3는 다음 객체를 복제합니다.

- 복제 구성을 추가한 후에 생성된 객체입니다.

- 암호화되지 않은 객체입니다.

- Amazon S3 관리형 키(SSE-S3) 또는 AWS Key Management Service에 저장된 고객 마스터 키(CMK)(SSE-KMS)에서 유휴 시 암호화된 객체입니다. AWS KMS에 저장된 CMK로 암호화된 객체를 복제하려면 이 옵션을 명시적으로 활성화해야 합니다. 객체의 복제본은 소스 객체에 사용된 동일한 유형의 서버 측 암호화를 사용하여 암호화됩니다. 서버 측 암호화에 대한 자세한 정보는 서버 측 암호화를 사용하여 데이터 보호 (p. 263) 단원을 참조하십시오.

- 객체 메타데이터.

- 원본 버킷에서 버킷 소유자가 객체 및 ACL(액세스 제어 목록)을 읽을 권한이 있는 객체만. 리소스 소유권에 대한 자세한 내용은 Amazon S3 버킷 및 객체 소유권 (p. 300)를 참조하십시오.

- 객체 ACL 업데이트. 단, 사용자가 원본 버킷과 대상 버킷을 동일한 계정이 소유하지 않은 경우 Amazon S3에게 복제본 소유권을 변경하도록 지시하지 않은 경우에 한합니다. 자세한 내용은 복제본 소유자 변경 (p. 626) 단원을 참조하십시오.

  Amazon S3에서 두 ACL을 동기화하려면 시간이 약간 걸릴 수 있습니다. 이는 버킷에 복제 구성을 추가한 후 생성된 객체에만 적용됩니다.

- 객체 태그(있는 경우).

- Amazon S3 객체 잠금 보존 정보(있는 경우). Amazon S3는 보존 정보가 적용된 객체를 복제할 때 복제본에 동일한 보존 제어를 적용하여 대상 버킷에 구성된 기본 보존 기간을 재정의합니다. 원본 버킷의 객체에 보존 제어가 적용되지 않았고 기본 보존 기간이 설정된 대상 버킷에 복제하는 경우, 대상 버킷의 기본 보존

기간이 객체 복제본에 적용됩니다. 자세한 내용은 Amazon S3 객체 잠금을 사용하여 객체 잠금 (p. 490) 단원을 참조하십시오.

삭제 작업이 복제에 미치는 영향

원본 버킷에서 객체를 삭제할 경우 다음과 같은 동작이 발생합니다.

- 객체 버전 ID를 지정하지 않고 DELETE 요청을 하면 Amazon S3는 삭제 마커를 추가합니다. Amazon S3가 삭제 마커를 처리하는 방식은 다음과 같습니다.
  - 최신 버전의 복제 구성을 사용하는 경우(사용자가 복제 구성 규칙의 `Filter` 요소를 지정) Amazon S3는 삭제 마커를 복제하지 않습니다.
  - 사용자가 `Filter` 요소를 지정하지 않을 경우 Amazon S3는 복제 구성이 이전 버전 V1이라고 가정합니다. 이전 버전에서는 Amazon S3가 삭제 마커 복제를 다르게 처리했습니다. 자세한 내용은 이전 버전과의 호환성 (p. 620) 단원을 참조하십시오.
- 사용자가 DELETE 요청에서 삭제할 객체 버전 ID를 지정할 경우, Amazon S3가 원본 버킷에서 해당 객체 버전을 삭제합니다. 하지만 대상 버킷에서는 삭제를 복제하지 않습니다. 즉, 대상 버킷에서 동일한 객체 버전을 삭제하지 않습니다. 이는 악의적 삭제로부터 데이터를 보호합니다.

복제 불가능한 객체

기본적으로 Amazon S3는 다음 객체를 복제하지 않습니다.

- 버킷에 복제 구성을 추가하기 전에 존재한 객체. 즉, Amazon S3는 소급하여 객체를 복제하지 않습니다.

- 다음과 같은 암호화된 객체:
  - 고객 제공(SSE-C) 암호화 키를 사용하는 서버 측 암호화(SSE-C)로 생성된 객체
  - AWS KMS에 저장된 CMK를 사용한 서버 측 암호화로 생성된 객체입니다. 기본적으로 Amazon S3은 KMS CMK를 사용하여 암호화된 객체를 복제하지 않습니다. 하지만 복제 구성에서 명시적으로 이러한 객체의 복제를 활성화하고 Amazon S3이 객체를 복제할 수 있도록 관련 정보를 제공할 수 있습니다.

  서버 측 암호화에 대한 자세한 정보는 서버 측 암호화를 사용하여 데이터 보호 (p. 263) 단원을 참조하십시오.

- S3 Glacier 또는 S3 Glacier Deep Archive 스토리지 클래스에 저장된 객체입니다. Amazon S3 Glacier 서비스에 대한 자세한 내용은 Amazon S3 Glacier 개발자 안내서 항목을 참조하십시오.

- 원본 버킷에서 버킷 소유자에게 권한이 없는 객체(버킷 소유자가 객체 소유자가 아닌 경우). 객체 소유자가 버킷 소유자에게 권한을 부여할 수 있는 방법에 대한 자세한 내용은 버킷 소유자가 완벽하게 제어할 수 있도록 보증하면서 객체에 업로드하는 교차 계정 권한 부여 (p. 413) 단원을 참조하십시오.

- 버킷 레벨 하위 리소스에 대한 업데이트. 예를 들어, 수명 주기 구성을 변경하거나 원본 버킷에 알림 구성을 추가할 경우 이러한 변경 사항은 대상 버킷에 적용되지 않습니다. 그러므로 원본 버킷과 대상 버킷에 서로 다른 버킷 구성을 할 수 있습니다.

- 수명 주기 구성에 의해 수행되는 작업.

  예를 들어, 원본 버킷에서만 수명 주기 구성이 활성화되면 Amazon S3는 만료된 객체에 대한 삭제 마커를 만들지만 그 마커를 복제하지는 않습니다. 원본 버킷과 대상 버킷에 동일한 수명 주기 구성을 적용하려는 경우 두 버킷 모두에서 동일한 수명 주기를 활성화합니다.

수명 주기 구성에 대한 자세한 내용은 객체 수명 주기 관리 (p. 123) 단원을 참조하십시오.

Note

최신 버전의 복제 구성을 사용하는 경우(XML이 `Filter`를 `Rule`의 하위로 지정), 사용자 작업에 의해 생성되거나 Amazon S3이 수명 주기 작업의 일환으로 생성한 삭제 마커는 복제되지 않습니다. 하지만 이전 버전의 복제 구성을 사용하는 경우(XML이 `Prefix`를 `Rule`의 하위로 지정), 사용자 작업의 결과로 생성된 삭제 마커가 복제됩니다. 자세한 내용은 이전 버전과의 호환성 (p. 620) 단원을 참조하십시오.

- 원본 버킷의 객체는 다른 복제 규칙에 따라 생성된 복제본입니다.

  원본 버킷의 객체는 오직 한 개의 대상 버킷에만 복제가 가능합니다. Amazon S3가 객체를 복제한 이후에는 해당 객체를 다시 복제할 수 없습니다. 예를 들어, 기존 복제 구성에서 대상 버킷을 변경하더라도 Amazon S3가 해당 객체를 다시 복제하지 않습니다.

  또 하나의 예: 버킷 A가 원본이고 버킷 B는 대상인 복제를 구성한다고 가정하겠습니다. 이제 버킷 B가 원본이고 버킷 C가 대상인 다른 복제 구성을 추가한다고 가정합니다. 이 경우, 버킷 A 객체의 복제본인 버킷 B의 객체는 버킷 C로 복제되지 않습니다.

기존 객체 복제

계정에 대해 기존 객체 복제를 활성화하려면 AWS Support에 문의해야 합니다. 요청이 지연되지 않도록 방지하려면 AWS Support 사례에 “기존 객체에 대한 복제”라고 제목을 지정하고 다음 정보를 포함해야 합니다.

- 원본 버킷
- 대상 버킷
- 복제할 예상 스토리지 볼륨(테라바이트 단위)
- 복제할 예상 스토리지 객체 수

관련 주제

- 복제 (p. 607)
- 복제 설정 개요 (p. 611)
- 복제 상태 정보 (p. 654)

복제 설정 개요

복제를 활성화하려면 원본 버킷에 복제 구성을 추가하기만 하면 됩니다. 이 구성은 Amazon S3에게 지정된 대로 객체를 복제하도록 지시합니다. 복제 구성에서는 다음을 제공해야 합니다.

- 대상 버킷 — Amazon S3가 객체를 복제할 버킷입니다.

- 복제할 객체 — 원본 버킷 또는 하위 집합에 있는 모든 객체를 복제할 수 있습니다. 구성에 키 이름 접두사, 하나 이상의 객체 태그, 또는 둘 모두를 제공하여 하위 집합을 식별합니다.

  예를 들어, 키 이름 접두사 `Tax/`가 포함된 객체만 복제할 복제 규칙을 구성할 경우 Amazon S3은 `Tax/doc1` 또는 `Tax/doc2`와 같은 키가 있는 객체를 복제합니다. 그러나 `Legal/doc3` 키가 있는 객체는 복제하지 않습니다. 접두사와 하나 이상의 태그를 함께 지정할 경우 Amazon S3는 특정 키 접두사와 태그가 있는 객체만 복제합니다.

복제본은 원본 객체와 동일한 키 이름 및 메타데이터(예: 생성 시간, 사용자 정의 메타데이터, 버전 ID)를 가집니다. Amazon S3는 Secure Sockets Layer(SSL)를 사용하여 전송 중인 모든 데이터를 암호화합니다.

이러한 최소 요건에 더해, 다음 옵션을 선택할 수 있습니다.

- 기본적으로 Amazon S3는 원본 객체와 동일한 스토리지 클래스를 사용하여 객체 복제본을 저장합니다. 복제본에 대해 다른 스토리지 클래스를 지정할 수 있습니다.

- Amazon S3는 원본 객체 소유자가 객체 복제본을 계속 소유하게 된다고 가정합니다. 따라서 객체를 복제할 때 해당되는 객체 ACL(액세스 제어 목록)도 복제됩니다. 원본 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유하는 경우 복제본 소유자를 대상 버킷을 소유한 AWS 계정으로 변경하도록 복제를 구성할 수 있습니다.

추가 구성 옵션을 사용할 수 있습니다. 자세한 내용은 추가 복제 구성 (p. 623) 단원을 참조하십시오.

또한 Amazon S3은 복제 규칙 설정을 지원하기 위해 API를 제공합니다. 자세한 내용은 Amazon Simple Storage Service API Reference에서 다음 주제를 참조하십시오.

- PUT Bucket replication
- GET Bucket replication
- DELETE Bucket replication

코드에서 이러한 API를 직접 호출하는 대신 AWS SDK, AWS CLI 또는 Amazon S3 콘솔을 사용하여 복제 구성을 버킷에 추가할 수 있습니다. 콘솔을 사용하는 편이 가장 간단합니다. 단계별 실행 지침이 포함된 예제는 복제 실습 과정 (p. 633) 단원을 참조하십시오.

복제 구성을 처음 사용하는 경우 예제 및 옵션 구성을 살펴보기 전에 다음 개요를 읽는 것이 좋습니다. 기본 복제 구성을 설정하는 방법에 대한 단계적 지침을 제공하는 예제는 복제 구성 개요 (p. 612) 단원을 참조하십시오.

주제

- 복제 구성 개요 (p. 612)
- 복제에 대한 권한 설정 (p. 620)

복제 구성 개요

Amazon S3는 복제 구성을 XML로 저장합니다. 복제 구성 XML 파일에서 AWS Identity and Access Management(IAM) 규칙과 하나 이상의 규칙을 지정합니다.

```
<ReplicationConfiguration>
    <Role>IAM-role-ARN</Role>
    <Rule>
        ...
    </Rule>
    <Rule>
         ...
    </Rule>
     ...
</ReplicationConfiguration>
```

Amazon S3는 사용자가 부여한 권한 없이 객체를 복제할 수 없습니다. 사용자는 복제 구성에서 지정하는 IAM 역할을 사용하여 권한을 부여합니다. Amazon S3는 이 IAM 역할을 수임하여 사용자 대신 객체를 복제합니다. 먼저 IAM 역할에 필요한 권한을 부여해야 합니다. 권한 관리에 대한 자세한 내용은 복제에 대한 권한 설정 (p. 620) 단원을 참조하십시오.

다음 시나리오에서는 복제 구성에 한 가지 규칙을 추가합니다.

- 모든 객체를 복제하려는 경우
- 객체의 하위 집합을 복제하려는 경우 규칙에 필터를 추가하여 객체 하위 집합을 식별하는 경우 필터에서 객체 키 접두사, 태그 또는 이 두 가지를 모두 지정하여 객체에서 규칙이 적용될 하위 집합을 식별합니다.

객체에서 다른 하위 집합을 선택하려면 복제 구성에 여러 규칙을 추가할 수 있습니다. 각 규칙에서 다른 객체 하위 집합을 선택하는 필터를 지정합니다. 예를 들어 `tax/` 또는 `document/` 키 접두사를 갖는 객체를 복제하도록 선택할 수 있습니다. 2개의 규칙을 추가하여 한 규칙에는 `tax/` 키 접두사 필터를 지정하고 다른 규칙에는 `document/` 키 접두사를 지정합니다.

다음 단원에서 정보를 추가로 제공합니다.

주제

- 기본 규칙 구성 (p. 613)
- 선택 사항: 필터 지정 (p. 614)
- 추가 대상 구성 (p. 615)
- 예제 복제 구성 (p. 616)
- 이전 버전과의 호환성 (p. 620)

기본 규칙 구성

각 규칙은 규칙의 상태 및 우선 순위를 포함하고 삭제 마커를 복제할지 여부를 표시해야 합니다.

- `Status`는 규칙이 활성화 또는 비활성화되었는지 표시합니다. 규칙이 비활성화되면 Amazon S3는 규칙에 지정된 작업을 수행하지 않습니다.
- `Priority`는 여러 규칙이 한 객체에 적용될 경우 우선 순위가 있는 규칙을 표시합니다.
- 현재, 삭제 마커는 복제되지 않으며, 사용자가 `DeleteMarkerReplication`을 `Disabled`로 설정해야 합니다.

대상 구성에서 Amazon S3가 객체를 복제할 버킷의 이름을 제공해야 합니다.

다음 코드는 규칙에 대한 최소 요구 사항을 보여줍니다.

```
...
    <Rule>
        <ID>Rule-1</ID>
        <Status>rule-Enabled-or-Disabled</Status>
        <Priority>integer</Priority>
        <DeleteMarkerReplication>
           <Status>Disabled</Status>
        </DeleteMarkerReplication>
        <Destination>
           <Bucket>arn:aws:s3:::bucket-name</Bucket>
        </Destination>
    </Rule>
    <Rule>
         ...
    </Rule>
     ...
...
```

또한 다른 구성 옵션도 지정할 수도 있습니다. 예를 들어 원본 객체의 스토리지 클래스와 다른 객체 복제본 스토리지 클래스를 사용할 수 있습니다.

선택 사항: 필터 지정

객체에서 규칙이 적용되는 하위 집합을 선택하려면 옵션 필터를 추가합니다. 객체 키 접두사, 객체 태그 또는 이 두 가지를 모두 사용하여 필터를 지정할 수 있습니다. 키 접두사와 객체 태그를 모두 사용하여 필터링하는 경우, Amazon S3는 논리 AND 연산자를 사용하여 필터를 조합합니다. 즉, 규칙은 특정 키 접두사와 특정 태그를 가진 객체의 하위 집합에 적용됩니다.

객체 키 접두사를 기반으로 한 필터를 사용하여 규칙을 지정하려면 다음 코드를 사용합니다. 접두사는 하나만 지정할 수 있습니다.

```
<Rule>
    ...
    <Filter>
        <Prefix>key-prefix</Prefix>
    </Filter>
    ...
</Rule>
...
```

객체 태그를 기반으로 한 필터를 사용하여 규칙을 지정하려면 다음 코드를 사용합니다. 객체 태그는 하나 이상 지정할 수 있습니다.

```
<Rule>
    ...
    <Filter>
        <And>
            <Tag>
                <Key>key1</Key>
                <Value>value1</Value>
            </Tag>
            <Tag>
                <Key>key2</Key>
                <Value>value2</Value>
            </Tag>
             ...
        </And>
    </Filter>
    ...
</Rule>
...
```

키 접두사와 객체 태그가 조합된 규칙 필터를 지정하려면 다음 코드를 사용합니다. 이들 필터를 AND 상위 요소에 래핑합니다. Amazon S3가 논리 AND 연산을 수행해 이들 필터를 조합합니다. 즉, 규칙은 특정 키 접두사와 특정 태그를 가진 객체의 하위 집합에 적용됩니다.

```
<Rule>
    ...
    <Filter>
        <And>
            <Prefix>key-prefix</Prefix>
            <Tag>
                <Key>key1</Key>
                <Value>value1</Value>
            </Tag>
            <Tag>
                <Key>key2</Key>
                <Value>value2</Value>
            </Tag>
             ...
    </Filter>
```

```
    ...
</Rule>
...
```

추가 대상 구성

대상 구성에서 Amazon S3가 객체를 복제할 버킷을 지정합니다. 한 원본 버킷에서 한 대상 버킷으로 객체를 복제하도록 복제를 구성할 수 있습니다. 복제 구성에 여러 규칙을 추가하는 경우 모든 규칙에서 동일한 대상 버킷을 지정해야 합니다.

```
...
<Destination>
    <Bucket>arn:aws:s3:::destination-bucket</Bucket>
</Destination>
...
```

`<Destination>` 요소에 다음 옵션을 추가할 수 있습니다.

- 객체 복제본에 대해 스토리지 클래스를 지정할 수 있습니다. 기본적으로 Amazon S3는 다음 예제와 같이 원본 객체의 스토리지 클래스를 사용하여 객체 복제본을 생성합니다.

```
...
<Destination>
       <Bucket>arn:aws:s3:::destinationbucket</Bucket>
       <StorageClass>storage-class</StorageClass>
</Destination>
...
```

- 원본 버킷과 대상 버킷을 동일한 계정에서 소유하지 않는 경우 `AccessControlTranslation` 요소를 추가하여 복제본 소유권을 대상 버킷을 소유한 AWS 계정으로 변경할 수 있습니다.

```
...
<Destination>
   <Bucket>arn:aws:s3:::destinationbucket</Bucket>
   <Account>destination-bucket-owner-account-id</Account>
   <AccessControlTranslation>
       <Owner>Destination</Owner>
   </AccessControlTranslation>
</Destination>
...
```

  복제 구성에 이 요소를 추가하지 않으면 원본 객체를 소유한 계정과 동일한 AWS 계정이 복제본을 소유합니다. 자세한 내용은 복제본 소유자 변경 (p. 626) 단원을 참조하십시오.

- 복제 구성에서 S3 Replication Time Control (S3 RTC)를 활성화할 수 있습니다. S3 RTC는 대부분의 객체를 몇 초 만에 복제하고 이러한 객체의 99.99%를 15분 내에 복제합니다(서비스 수준 계약에 따라).

  Note

  `EventThreshold` 및 `Time`에는 유효한 값의 `<Minutes>15</Minutes>`만 허용됩니다.

```
...
• <Destination>
• ...
•        <Bucket>arn:aws:s3:::destinationbucket</Bucket>
•             <Metrics>
•                 <Status>Enabled</Status>
•                 <EventThreshold>
•                     <Minutes>15</Minutes>
•                 </EventThreshold>
```

```
•             </Metrics>
•             <ReplicationTime>
•                 <Status>Enabled</Status>
•                 <Time>
•                     <Minutes>15</Minutes>
•                 </Time>
•             <ReplicationTime>...
• </Destination>

...
```

자세한 내용은 S3 Replication Time Control (S3 RTC)를 사용하여 객체 복제 (p. 624) 단원을 참조하십시오. API 예제는 Amazon Simple Storage Service API Reference의 PutBucketReplication을 참조하십시오.

- AWS KMS에 저장된 키를 사용하는 서버 측 암호화를 사용하여 생성되는 객체가 원본 버킷에 포함될 수 있습니다. 기본적으로 Amazon S3는 이러한 객체를 복제하지 않습니다. 필요한 경우 Amazon S3에 이러한 객체를 복제하도록 지시할 수 있습니다. 먼저 `SourceSelectionCriteria` 요소를 추가하여 이 기능을 명시적으로 선택한 다음 객체 복제본 암호화에 사용할 AWS KMS CMK(대상 버킷의 AWS 리전용)를 제공합니다.

```
...
<SourceSelectionCriteria>
  <SseKmsEncryptedObjects>
    <Status>Enabled</Status>
  </SseKmsEncryptedObjects>
</SourceSelectionCriteria>
<Destination>
  <Bucket>arn:aws:s3:::dest-bucket-name</Bucket>
  <EncryptionConfiguration>
    <ReplicaKmsKeyID>AWS KMS key IDs to use for encrypting object replicas</
ReplicaKmsKeyID>
  </EncryptionConfiguration>
</Destination>
...
```

자세한 내용은 AWS KMS에 저장된 암호화 키를 사용한 서버 측 암호화(SSE)로 생성된 객체 복제 (p. 629) 단원을 참조하십시오.

예제 복제 구성

시작하려면, 다음 예제 복제 구성을 적절히 버킷에 추가할 수 있습니다.

> **Important**
>
> 버킷에 복제 구성을 추가하려면 `iam:PassRole` 권한이 있어야 합니다. 이 권한은 Amazon S3에 복제 권한을 부여하는 IAM 역할을 전달하도록 허용합니다. 복제 구성 XML의 `Role` 요소에서 사용되는 Amazon 리소스 이름(ARN)을 제공하여 IAM 역할을 지정합니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 AWS 서비스에 역할을 전달할 수 있는 권한 부여를 참조하십시오.

Example 1: 한 가지 규칙으로 이루어진 복제 구성

다음 기본 복제 구성은 한 규칙을 지정합니다. 이 규칙은 Amazon S3가 수임할 수 있는 IAM 규칙과 객체 복제본의 대상 버킷을 지정합니다. 규칙 `Status`는 이 규칙이 유효한 상태임을 표시합니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::AcctID:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
```

```
    <Destination><Bucket>arn:aws:s3:::destinationbucket</Bucket></Destination>

  </Rule>
</ReplicationConfiguration>
```

복제할 객체 하위 집합을 선택하려면 필터를 추가할 수 있습니다. 다음 구성에서 필터는 객체 키 접두사를 지정합니다. 이 규칙은 키 이름에 접두사 `Tax/`가 포함된 객체에 적용됩니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::AcctID:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>

    <Filter>
       <Prefix>Tax/</Prefix>
    </Filter>

    <Destination><Bucket>arn:aws:s3:::destinationbucket</Bucket></Destination>

  </Rule>
</ReplicationConfiguration>
```

`Filter` 요소를 지정하는 경우 `Priority` 및 `DeleteMarkerReplication` 요소도 포함시켜야 합니다. 이 예제에서는 규칙이 하나뿐이므로 우선 순위는 상관이 없습니다.

다음 구성에서 필터는 하나의 접두사 두 개의 태그를 지정합니다. 이 규칙은 객체에서 지정된 접두사와 태그가 있는 하위 집합에 적용됩니다. 구체적으로, 이 규칙은 키 이름에 `Tax/` 접두사가 포함되고 두 개의 지정된 객체 태그가 있는 객체에 적용됩니다. 규칙이 하나뿐이므로 우선 순위는 적용되지 않습니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::AcctID:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>

    <Filter>
        <And>
          <Prefix>Tax/</Prefix>
          <Tag>
             <Tag>
                <Key>tagA</Key>
                <Value>valueA</Value>
             </Tag>
          </Tag>
          <Tag>
             <Tag>
                <Key>tagB</Key>
                <Value>valueB</Value>
             </Tag>
          </Tag>
       </And>

    </Filter>
```

```
    <Destination><Bucket>arn:aws:s3:::destinationbucket</Bucket></Destination>

  </Rule>
</ReplicationConfiguration>
```

다음과 같이 객체 복제본에 대해 스토리지 클래스를 지정할 수 있습니다.

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

<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Destination>
       <Bucket>arn:aws:s3:::destinationbucket</Bucket>
       <StorageClass>storage-class</StorageClass>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

Amazon S3가 지원하는 모든 스토리지 클래스를 지정할 수 있습니다.

Example 2: 두 가지 규칙으로 이루어진 복제 구성

Example

다음 복제 구성에서,

- 각 규칙은 다른 키 접두사를 기반으로 필터링합니다. 따라서 각 규칙이 객체의 특정 하위 집합에 적용됩니다. Amazon S3는 키 이름이 `Tax/doc1.pdf` 및 `Project/project1.txt`인 객체를 복제하지만 키 이름이 `PersonalDoc/documentA`인 객체는 복제하지 않습니다.
- 규칙이 서로 다른 객체 집합에 적용되므로 규칙 우선 순위는 상관이 없습니다. 다음 예제는 규칙 우선 순위가 적용되면 발생하는 상황을 보여 줍니다.
- 두 번째 규칙은 객체 복제본의 스토리지 클래스를 지정합니다. Amazon S3는 객체 복제본에 지정된 스토리지 클래스를 사용합니다.
- 두 규칙 모두 동일한 대상 버킷을 지정합니다. 몇 개의 규칙을 지정하든 하나의 대상 버킷만 지정할 수 있습니다.

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

<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>Tax</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Destination>
      <Bucket>arn:aws:s3:::destinationbucket</Bucket>
    </Destination>
     ...
  </Rule>
 <Rule>
    <Status>Enabled</Status>
```

```
    <Priority>2</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>Project</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Destination>
      <Bucket>arn:aws:s3:::destinationbucket</Bucket>
     <StorageClass>STANDARD_IA</StorageClass>
    </Destination>
     ...
  </Rule>


</ReplicationConfiguration>
```

Example 3: 접두사가 중첩되는 2개의 규칙을 포함하는 복제 구성

이 구성에는 `star/`와 `starship` 같이 중첩되는 키 접두사를 포함하는 필터를 지정하는 규칙이 두 개 있습니다. 두 규칙은 키 이름이 `starship-x`인 객체에 모두 적용됩니다. 이 경우, Amazon S3는 규칙 우선 순위를 사용하여 어느 규칙을 적용할지 결정합니다.

```
<ReplicationConfiguration>

  <Role>arn:aws:iam::AcctID:role/role-name</Role>

  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>star</Prefix>
    </Filter>
    <Destination>
      <Bucket>arn:aws:s3:::destinationbucket</Bucket>
    </Destination>
  </Rule>
  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>starship</Prefix>
    </Filter>
    <Destination>
      <Bucket>arn:aws:s3:::destinationbucket</Bucket>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

Example 4: 연습 예제

예제 안내에 대해서는 복제 실습 과정 (p. 633)을 참조하십시오.

복제 구성의 XML 구조에 대한 자세한 내용은 Amazon Simple Storage Service API Reference의 PutBucketReplication 단원을 참조하십시오.

이전 버전과의 호환성

최신 버전의 복제 구성 XML은 V2입니다. 이전 버전과의 호환성을 위해 Amazon S3는 V1 구성을 계속 지원합니다. 복제 구성 XML V1을 사용한 경우, 이전 버전과의 호환성에 영향을 미치는 다음 문제를 고려하십시오.

- 복제 구성 XML V2는 규칙에 대한 `Filter` 요소를 포함합니다. `Filter` 요소를 사용하면 객체 키 접두사, 태그 또는 이 두 가지를 기반으로 객체 필터를 지정하여 규칙이 적용되는 객체의 범위를 설정할 수 있습니다. 복제 구성 XML V1은 키 접두사를 기반으로 한 필터링만 지원합니다. 이 경우 다음 예제와 같이 `Rule` 요소의 하위 요소로 `Prefix`를 직접 추가합니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::AcctID:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Prefix>key-prefix</Prefix>
    <Destination><Bucket>arn:aws:s3:::destinationbucket</Bucket></Destination>

  </Rule>
</ReplicationConfiguration>
```

  이전 버전과의 호환성을 위해 `Amazon S3` 는 V1 구성을 계속 지원합니다.

- 객체 버전 ID를 지정하지 않고 원본 버킷에서 객체를 삭제할 경우 Amazon S3는 삭제 마커를 추가합니다. 복제 구성 XML V1을 사용하는 경우, Amazon S3는 사용자 작업의 결과로 발생한 삭제 마커를 복제합니다. 즉, 사용자가 객체를 삭제한 경우에는 삭제 마커를 복제하고, Amazon S3가 수명 주기 작업에 의해 만료된 객체를 삭제한 경우에는 삭제 마커를 복제하지 않습니다. V2에서 Amazon S3는 삭제 마커를 복제하지 않습니다. 따라서 `DeleteMarkerReplication` 요소를 `Disabled`으로 설정해야 합니다.

```
...
    <Rule>
        <ID>Rule-1</ID>
        <Status>rule-Enabled-or-Disabled</Status>
        <Priority>integer</Priority>
        <DeleteMarkerReplication>
           <Status>Disabled</Status>
        </DeleteMarkerReplication>
        <Destination>
           <Bucket>arn:aws:s3:::bucket-name</Bucket>
        </Destination>
    </Rule>
...
```

복제에 대한 권한 설정

복제를 설정할 때 다음과 같이 필요한 권한을 획득해야 합니다.

- IAM 역할 생성 – Amazon S3는 사용자 대신 객체를 복제하려면 권한이 필요합니다. IAM 역할을 생성하고 복제 구성에서 이 역할을 지정하여 이러한 권한을 부여합니다.
- 원본 버킷과 대상 버킷을 동일한 계정에서 소유하지 않는 경우 대상 버킷 소유자는 원본 버킷 소유자에게 복제본을 저장할 권한을 부여해야 합니다.

주제

- IAM 역할 생성 (p. 621)
- 원본 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유할 경우 권한 부여 (p. 622)

IAM 역할 생성

기본적으로 Amazon S3 리소스인 버킷, 객체 및 관련 하위 리소스는 모두 비공개이며 리소스 소유자만 리소스에 액세스할 수 있습니다. 원본 버킷에서 객체를 읽고 대상 버킷에 복제하려면 Amazon S3는 이러한 작업을 수행할 권한이 필요합니다. IAM 역할을 생성하고 복제 구성에서 이 역할을 지정하여 이러한 권한을 부여합니다.

이 단원에서는 신뢰 정책과 필요한 최소한의 권한 정책을 설명합니다. 연습 예제가 IAM 역할을 생성하는 단계별 지침을 제공합니다. 자세한 내용은 복제 실습 과정 (p. 633) 단원을 참조하십시오.

- 다음은 Amazon S3를 역할을 수임할 수 있는 서비스 보안 주체로 식별하는 신뢰 정책을 보여줍니다.

```
{

   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

  IAM 역할에 대한 자세한 내용은 IAM 사용 설명서의 IAM 역할 단원을 참조하십시오.
- 다음은 역할에 사용자 대신 복제 작업을 수행할 권한을 부여하는 액세스 정책을 보여줍니다. Amazon S3가 이 역할을 맡으면 이 정책에 지정된 권한을 보유하게 됩니다.

```
{

   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::source-bucket"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[

            "s3:GetObjectVersion",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"

         ],
         "Resource":[
            "arn:aws:s3:::source-bucket/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
```

```
            "s3:ReplicateTags"
         ],
         "Resource":"arn:aws:s3:::destination-bucket/*"
      }
   ]
}
```

액세스 정책은 다음 작업을 수행할 권한을 부여합니다.

- `s3:GetReplicationConfiguration` 및 `s3:ListBucket`Amazon S3가 원본 버킷에 대한 이러한 작업 권한을 사용하여 복제 구성을 검색하고 버킷 콘텐츠를 나열할 수 있습니다(현재의 권한 모델은 삭제 마커에 액세스하기 위해 `s3:ListBucket` 권한이 필요함).
- `s3:GetObjectVersion` 및 `s3:GetObjectVersionAcl`Amazon S3가 모든 객체에 대해 부여되는 이러한 작업 권한을 사용하여 특정객체 버전과 객체와 연결된 ACL(액세스 통제 목록)을 가져올 수 있습니다.
- `s3:ReplicateObject` 및 `s3:ReplicateDelete`Amazon S3가 대상 버킷의 객체에 대한 이러한 작업 권한을 사용하여 대상 버킷에 객체 또는 삭제 마커를 복제할 수 있습니다. 삭제 마커에 대한 자세한 내용은 삭제 작업이 복제에 미치는 영향 (p. 610)을 참조하십시오.

  > **Note**
  >
  > 대상 버킷에 대한 `s3:ReplicateObject` 작업 권한은 객체 태그 복제도 허용합니다. 따라서 `s3:ReplicateTags` 작업에 대한 권한을 명시적으로 부여할 필요가 없습니다.
- `s3:GetObjectVersionTagging`Amazon S3가 원본 버킷의 객체에 대한 이 작업 권한을 사용하여 복제 대상 객체 태그를 읽을 수 있습니다(객체 태그 지정 (p. 114) 단원 참조). Amazon S3는 이 권한이 없을 경우 객체는 복제하지만 객체 태그(있는 경우)는 복제하지 않습니다.

Amazon S3 작업 목록에 대해서는 Amazon S3 작업 (p. 342) 단원을 참조하십시오.

> **Important**
>
> IAM 역할을 소유한 AWS 계정은 해당 IAM 역할에 부여된 작업을 수행할 권한이 있어야 합니다. 예를 들어, 원본 버킷에 다른 AWS 계정이 소유한 객체가 포함되어 있다고 가정하겠습니다. 객체 소유자는 객체 ACL을 통해 IAM 역할을 소유한 AWS 계정에 필요한 권한을 명시적으로 부여해야 합니다. 그렇지 않으면 Amazon S3가 객체에 액세스할 수 없으므로 객체의 복제가 실패합니다. ACL 권한에 대한 자세한 내용은 ACL(액세스 통제 목록) 개요 (p. 438) 단원을 참조하십시오. 여기에서 설명하는 권한은 최소 복제 구성과 관련됩니다. 선택적 복제 구성을 추가하려면 Amazon S3에 추가 권한을 부여해야 합니다. 자세한 내용은 추가 복제 구성 (p. 623) 단원을 참조하십시오.

원본 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유할 경우 권한 부여

원본 버킷과 대상 버킷을 동일한 계정에서 소유하지 않는 경우, 다음과 같이 대상 버킷 소유자가 원본 버킷 소유자에게 복제 작업 권한을 부여하는 버킷 정책도 추가해야 합니다.

```
{
   "Version":"2008-10-17",
   "Id":"PolicyForDestinationBucket",
   "Statement":[
      {
         "Sid":"1",
         "Effect":"Allow",
         "Principal":{
            "AWS":"SourceBucket-AcctID"
         },
         "Action":[
            "s3:ReplicateDelete",
            "s3:ReplicateObject"
```

```
            ],
            "Resource":"arn:aws:s3:::destinationbucket/*"
        },
        {
            "Sid":"2",
            "Effect":"Allow",
            "Principal":{
                "AWS":"SourceBucket-AcctID"
            },
            "Action": [
                "s3:List*",
                "s3:GetBucketVersioning",
                "s3:PutBucketVersioning"
            ],
            "Resource":"arn:aws:s3:::destinationbucket"
        }
    ]
}
```

문제 해결 예는 예제 2: 원본 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유한 경우 복제를 구성 (p. 642) 단원을 참조하십시오.

원본 버킷에 있는 객체에 태그가 지정된 경우, 다음에 주의하십시오.

- 원본 버킷 소유자가 객체 태그를 복사할 수 있도록 `s3:GetObjectVersionTagging` 및 `s3:ReplicateTags` 작업 권한을 Amazon S3에 부여하는 경우(IAM 역할을 통해), Amazon S3가 객체와 함께 객체 태그를 복제하게 됩니다. IAM 역할에 대한 자세한 내용은 IAM 역할 생성 (p. 621) 단원을 참조하십시오.
- 대상 버킷 소유자가 태그 복제를 원하지 않는 경우, 소유자는 다음의 명령문을 대상 버킷 정책에 추가해 `s3:ReplicateTags` 작업에 대한 권한을 명시적으로 거부할 수 있습니다.

```
...
    "Statement":[
        {
            "Effect":"Deny",
            "Principal":{
                "AWS":"arn:aws:iam::SourceBucket-AcctID:root"
            },
            "Action":["s3:ReplicateTags"],
            "Resource":"arn:aws:s3:::destinationbucket/*"
        }
    ]
...
```

복제본 소유권 변경

서로 다른 AWS 계정이 원본 버킷과 대상 버킷을 소유한 경우, Amazon S3에 복제본 소유권을 대상 버킷을 소유한 AWS 계정으로 변경하도록 지시할 수 있습니다. 이를 사용자 재정의 옵션이라고 합니다. 자세한 내용은 복제본 소유자 변경 (p. 626) 단원을 참조하십시오.

추가 복제 구성

이 단원에서는 Amazon S3에서 사용할 수 있는 추가 복제 구성 옵션을 설명합니다. 핵심 복제 구성에 대한 자세한 내용은 복제 설정 개요 (p. 611) 단원을 참조하십시오.

주제

- S3 Replication Time Control (S3 RTC)를 사용하여 객체 복제 (p. 624)

- 복제본 소유자 변경 (p. 626)
- AWS KMS에 저장된 암호화 키를 사용한 서버 측 암호화(SSE)로 생성된 객체 복제 (p. 629)

S3 Replication Time Control (S3 RTC)를 사용하여 객체 복제

S3 Replication Time Control (S3 RTC)는 데이터 복제에 대한 규정 준수 또는 비즈니스 요구 사항을 충족하고 Amazon S3 복제 시간을 파악할 수 있도록 지원합니다. S3 RTC는 Amazon S3에 업로드하는 대부분의 객체를 몇 초 만에 복제하고 이러한 객체의 99.99%를 15분 내에 복제합니다.

복제 지표를 사용하여 복제 보류 중인 객체의 총 수와 크기 및 대상 리전까지의 최대 복제 시간을 모니터링할 수 있습니다. 복제 지표는 Amazon S3 콘솔 및 Amazon CloudWatch를 통해 사용할 수 있습니다. 자세한 내용은 the section called “Amazon S3 CloudWatch 복제 지표” (p. 673) 단원을 참조하십시오.

S3 RTC의 경우, Amazon S3 이벤트는 드물지만 객체가 15분 이내에 복제되지 않는 경우와 해당 객체가 대상 리전에 성공적으로 복제되는 경우를 알려줍니다. Amazon S3 이벤트는 Amazon SQS, Amazon SNS 또는 AWS Lambda을 통해 사용할 수 있습니다. 자세한 내용은 알림 (p. 587) 단원을 참조하십시오.

주제

- S3 Replication Time Control 활성화 (p. 624)
- 복제 지표를 사용하여 Amazon S3 복제 구성 모니터링 (p. 624)
- Amazon S3 이벤트를 사용하여 S3 Replication Time Control 객체 추적 (p. 625)
- S3 Replication Time Control 사용에 대한 모범 사례 및 지침 (p. 625)

S3 Replication Time Control 활성화

새 복제 규칙 또는 기존 복제 규칙과 함께 S3 Replication Time Control (S3 RTC)를 사용하여 시작할 수 있습니다. 복제 규칙을 전체 S3 버킷에 적용하거나 특정 접두사 또는 태그가 있는 Amazon S3 객체에 적용하도록 선택할 수 있습니다. S3 RTC를 활성화하면 복제 규칙에서도 복제 지표가 활성화됩니다.

> **Note**
>
> 복제 지표는 Amazon CloudWatch 사용자 지정 지표와 동일한 요금으로 청구됩니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

Amazon S3 콘솔, Amazon S3 API, AWS SDK 및 AWS 명령줄 인터페이스(AWS CLI)를 사용하여 복제 시간 제어를 구성할 수 있습니다.

자세한 내용은 복제 구성 개요 (p. 612)를 참조하십시오.

복제 지표를 사용하여 Amazon S3 복제 구성 모니터링

S3 Replication Time Control (S3 RTC)가 활성화된 각 복제 규칙은 복제 지표를 게시합니다. 복제 지표를 사용하여 복제 보류 중인 객체의 총 수와 크기 및 대상 리전까지의 최대 복제 시간을 모니터링할 수 있습니다. 그런 다음 별도로 복제하는 각 데이터 세트를 모니터링할 수 있습니다.

복제 지표는 S3 RTC를 활성화한 후 15분 이내에 사용할 수 있습니다. 복제 지표는 Amazon S3 콘솔, Amazon S3 API, AWS SDK, AWS 명령줄 인터페이스(AWS CLI) 및 Amazon CloudWatch를 통해 사용할 수 있습니다. 자세한 내용은 the section called “CloudWatch로 지표 모니터링” (p. 669) 단원을 참조하십시오.

Amazon S3 이벤트를 사용하여 S3 Replication Time Control 객체 추적

S3 Replication Time Control (S3 RTC)가 게시하는 특정 이벤트 알림을 모니터링하여 15분 이내에 복제되지 않은 객체의 복제 시간을 추적할 수 있습니다. 이러한 이벤트는 S3 RTC를 사용하여 복제할 수 있는 객체가 15분 이내에 복제되지 않은 경우 및 해당 객체가 대상 리전으로 복제되는 경우 게시됩니다.

복제 이벤트는 S3 RTC를 활성화한 후 15분 이내에 사용할 수 있습니다. Amazon S3 이벤트는 Amazon SQS, Amazon SNS 또는 AWS Lambda을 통해 사용할 수 있습니다. 자세한 내용은 the section called “이벤트 알림 유형 및 대상” (p. 590) 단원을 참조하십시오.

S3 Replication Time Control 사용에 대한 모범 사례 및 지침

S3 Replication Time Control (S3 RTC)를 사용하여 Amazon S3의 데이터를 복제할 때는 다음 모범 사례 및 지침에 따라 워크로드에 맞게 복제 성능을 최적화하십시오.

주제

- Amazon S3 복제 및 요청 속도 성능 지침 (p. 625)
- 복제 요청 속도 추정 (p. 625)
- S3 RTC 데이터 전송 속도 제한을 초과할 경우 조치 수행 (p. 626)
- AWS KMS 암호화된 객체 복제 요청 속도 (p. 626)
- AWS KMS 요청 속도 제한을 초과할 경우 조치 수행 (p. 626)

Amazon S3 복제 및 요청 속도 성능 지침

애플리케이션은 Amazon S3에서 스토리지를 업로드하고 검색할 때 요청 성능에서 초당 수천 회의 트랜잭션을 성취할 수 있습니다. 예를 들어, 애플리케이션은 S3 복제가 사용자를 대신하여 수행하는 요청을 포함하여 S3 버킷의 접두사에 대해 초당 3,500개 이상의 PUT/COPY/POST/DELETE 및 초당 5,500개 이상의 GET/HEAD 요청을 달성할 수 있습니다. 버킷의 접두사 수에는 제한이 없습니다. 읽기를 병렬화하여 읽기 또는 쓰기 성능을 향상시킬 수 있습니다. 예를 들어, S3 버킷에서 접두사 10개를 만들어 읽기를 병렬화하는 경우 읽기 성능을 초당 읽기 요청 55,000개로 조정할 수 있습니다.

Amazon S3는 이러한 지침을 초과하는 연속 요청 속도 또는 LIST 요청과 동시에 연속 요청 속도에 따라 자동으로 확장됩니다. Amazon S3가 새로운 요청 속도에 대해 내부적으로 최적화하는 동안 최적화가 완료될 때까지 일시적으로 HTTP 503 요청 응답을 받게 됩니다. 이는 초당 요청 속도가 증가하거나 S3 RTC를 처음 활성화할 때 발생할 수 있습니다. 이 기간 동안 복제 지연 시간이 증가할 수 있습니다. 초당 요청에 대한 Amazon S3 성능 지침이 초과되는 기간에는 S3 RTC SLA(서비스 수준 계약)가 적용되지 않습니다.

복제 데이터 전송 속도가 기본 1Gbps 한도를 초과하는 기간에는 S3 RTC SLA도 적용되지 않습니다. 복제 전송 속도가 1Gbps를 초과할 것으로 예상되는 경우 AWS Support Center에 문의하거나 서비스 할당량을 사용하여 한도 증가를 요청할 수 있습니다.

복제 요청 속도 추정

사용자를 대신하여 Amazon S3 복제가 수행하는 요청을 포함한 총 요청 속도는 복제 원본 및 대상 버킷 모두에 대한 Amazon S3 요청 속도 지침 내에 있어야 합니다. 복제된 각 객체에 대해 Amazon S3 복제는 최대 5개의 GET/HEAD 요청과 원본 버킷에 대한 PUT 요청 1개 및 대상 버킷에 대한 PUT 요청 1개로 구성됩니다.

예를 들어 초당 100개의 객체를 복제하려는 경우 Amazon S3 복제는 원본 S3 버킷에 대해 초당 총 200개의 PUT 요청에 추가로 100개의 PUT 요청을 사용자 대신 수행할 수 있습니다. Amazon S3 복제도 최대 500개의 GET/HEAD(복제된 각 객체에 대해 5개의 GET/HEAD 요청)를 수행할 수 있습니다.

> **Note**
>
> 복제된 객체당 하나의 PUT 요청에만 비용이 발생합니다. 자세한 내용은 복제 관련Amazon S3 FAQ의 요금 정보를 참조하십시오.

S3 RTC 데이터 전송 속도 제한을 초과할 경우 조치 수행

S3 Replication Time Control 데이터 전송 속도가 기본 1Gbps 한도를 초과할 것으로 예상되는 경우 AWS Support Center에 문의하거나 서비스 할당량을 사용하여 한도 증가를 요청하십시오.

AWS KMS 암호화된 객체 복제 요청 속도

Amazon S3 복제를 사용한 서버 측 암호화(SSE-KMS)로 암호화된 객체를 복제하는 경우 초당 AWS Key Management Service(AWS KMS) 요청 제한이 적용됩니다. AWS KMS는 요청 속도가 초당 요청 수 제한을 초과하므로 유효한 요청을 거부할 수 있습니다. 요청이 조절되면 AWS KMS가 `ThrottlingException` 오류를 반환합니다. AWS KMS 요청 속도 제한은 사용자가 직접 수행하는 요청과 사용자를 대신하여 Amazon S3 복제가 수행한 요청에 적용됩니다.

예를 들어 초당 1,000개의 객체를 복제하려는 경우 AWS KMS 요청 속도 제한에서 2,000개의 요청을 뺄 수 있습니다. 초당 요청 속도는 복제를 제외한 AWS KMS 워크로드에 대해 사용할 수 있습니다. Amazon CloudWatch의 AWS KMS 요청 지표를 사용하여 AWS 계정의 총 AWS KMS 요청 속도를 모니터링할 수 있습니다.

AWS KMS 요청 속도 제한을 초과할 경우 조치 수행

사용자를 대신하여 Amazon S3 복제가 수행한 복제 요청을 포함한 AWS KMS 요청 속도가 계정에 대한 AWS KMS 요청 속도 제한을 초과한다고 판단되는 경우 서비스 할당량을 사용하여 제한 증가를 요청할 수 있습니다. AWS KMS의 Service Quotas를 AWS 리전에서 사용할 수 없는 경우 AWS Support Center에 문의하십시오.

복제본 소유자 변경

복제에서는 기본적으로 원본 객체 소유자가 복제본도 소유합니다. 원본 버킷과 대상 버킷을 서로 다른 AWS 계정이 소유한 경우 복제본 소유권을 대상 버킷을 소유한 AWS 계정으로 변경하는 선택적 구성 설정을 추가할 수 있습니다. 이 설정을 통해 예를 들어 객체 복제본에 대한 액세스를 제한할 수도 있습니다. 이를 복제 구성의 소유자 재정의 옵션이라고도 합니다. 이 단원에서는 관련된 추가 구성 설정에 대해서만 설명합니다. 복제 구성 설정에 대한 자세한 내용은 복제 (p. 607) 단원을 참조하십시오.

소유자 재정의를 구성하려면 다음을 수행합니다.

- Amazon S3에 복제본 소유권을 변경하도록 지시하는 소유자 재정의 옵션을 복제 구성에 추가합니다.
- Amazon S3에 복제본 소유권을 변경할 권한을 부여합니다.
- 대상 버킷 정책에 복제본 소유권 변경을 허용할 권한을 추가합니다. 그러면 대상 버킷 소유자가 객체 복제본의 소유권을 수락할 수 있습니다.

다음 단원에서는 이러한 작업을 수행하는 방법을 설명합니다. 단계별 지침이 포함된 작업 예제는 예제 3: 원본 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유한 경우 복제본 소유자 변경 (p. 643) 단원을 참조하십시오.

복제 구성에 소유자 재정의 옵션 추가

> **Warning**
>
> 서로 다른 AWS 계정이 원본 버킷과 대상 버킷을 소유한 경우에만 소유자 재정의 옵션을 추가하십시오. Amazon S3는 버킷을 동일한 계정 또는 서로 다른 계정에서 소유하는지 확인하지 않습니다. 두 버킷을 동일한 AWS 계정에서 소유한 경우에도 소유자 재정의를 추가하면 Amazon S3가 소유자 재정의를 적용합니다. 즉, 대상 버킷 소유자에게 모든 권한을 부여하고, 후속 업데이트를 원본 객체 ACL(액세스 통제 목록)에 복제하지 않습니다. 복제본 소유자는 `PUT ACL` 요청을 사용하여 복제본과 연결된 ACL을 직접 변경할 수 있지만, 복제를 통해서는 변경할 수 없습니다.

소유자 재정의 옵션을 지정하려면 다음을 `Destination` 요소에 추가합니다.

- Amazon S3에게 복제본 소유자 변경을 지시하는 `AccessControlTranslation` 요소
- 대상 버킷 소유자의 AWS 계정을 지정하는 `Account` 요소

```
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    ...
    <Destination>
      ...
      <AccessControlTranslation>
           <Owner>Destination</Owner>
       </AccessControlTranslation>
      <Account>destination-bucket-owner-account-id</Account>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

다음 복제 구성 예제는 Amazon S3에게 키 접두사가 `Tax`인 객체를 대상 버킷으로 복제하고 복제본 소유권을 변경하도록 지시합니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Role>arn:aws:iam::account-id:role/role-name</Role>
   <Rule>
      <ID>Rule-1</ID>
      <Priority>1</Priority>
      <Status>Enabled</Status>
      <Status>Enabled</Status>
      <DeleteMarkerReplication>
         <Status>Disabled</Status>
      </DeleteMarkerReplication>
      <Filter>
         <Prefix>Tax</Prefix>
      </Filter>
      <Destination>
         <Bucket>arn:aws:s3:::destination-bucket</Bucket>
         <Account>destination-bucket-owner-account-id</Account>
         <AccessControlTranslation>
            <Owner>Destination</Owner>
         </AccessControlTranslation>
      </Destination>
   </Rule>
</ReplicationConfiguration>
```

Amazon S3에 복제본 소유권을 변경할 권한 부여

IAM 역할과 연결된 권한 정책에 `s3:ObjectOwnerOverrideToBucketOwner` 작업에 대한 권한을 추가하여 Amazon S3에 복제본 소유권을 변경할 권한을 부여합니다. 이 IAM 역할은 복제 구성에서 지정된 것이며, Amazon S3가 이 역할을 맡아 사용자 대신 객체를 복제할 수 있습니다.

```
...
{
    "Effect":"Allow",
         "Action":[
       "s3:ObjectOwnerOverrideToBucketOwner"
    ],
    "Resource":"arn:aws:s3:::destination-bucket/*"
}
...
```

대상 버킷 정책에 복제본 소유권 변경을 허용할 권한을 추가

대상 버킷 소유자가 원본 버킷 소유자에게 복제본 소유권을 변경할 수 있는 권한을 부여해야 합니다. 대상 버킷 소유자는 원본 버킷 소유자에게 `s3:ObjectOwnerOverrideToBucketOwner` 작업에 대한 권한을 부여합니다. 그러면 대상 버킷 소유자가 객체 복제본의 소유권을 수락할 수 있습니다. 다음 버킷 정책 문 예제에서는 이를 수행하는 방법을 보여줍니다.

```
...
{
    "Sid":"1",
    "Effect":"Allow",
    "Principal":{"AWS":"source-bucket-account-id"},
    "Action":["s3:ObjectOwnerOverrideToBucketOwner"],
    "Resource":"arn:aws:s3:::destination-bucket/*"
}
...
```

추가 고려 사항

소유권 재정의 옵션을 구성할 때는 다음 사항을 고려해야 합니다.

- 기본적으로 원본 객체 소유자가 복제본도 소유합니다. Amazon S3는 객체 버전 및 이와 연결된 ALC을 복제합니다.

  소유자 재정의를 추가할 경우, Amazon S3는 객체 버전만 복제하고 ACL은 복제하지 않습니다. 또한 Amazon S3는 원본 객체 ACL에 대한 후속 변경을 복제하지 않습니다. Amazon S3는 복제본에서 대상 버킷 소유자에게 모든 권한을 부여하는 ACL을 설정합니다.

- 복제 구성을 업데이트하여 소유자 재정의를 활성화 또는 비활성화하면 다음과 같은 효과가 발생합니다.

  - 복제 구성에 소유자 재정의 옵션을 추가할 경우:

    Amazon S3가 객체 버전을 복제할 때 원본 객체에 연결된 ACL을 무시합니다. 대신 복제본에서 대상 버킷 소유자에게 모든 권한을 부여하는 ACL을 설정합니다. 원본 객체 ACL에 대한 후속 변경을 복제하지 않습니다. 하지만 이 ACL 변경은 소유자 재정의 옵션을 설정하기 이전에 복제된 객체 버전에 적용되지 않습니다. 소유자 재정의를 설정하기 이전에 복제된 원본 객체에 대한 ACL 업데이트의 경우 객체와 복제본의 소유자가 동일하므로 계속해서 복제됩니다.

  - 복제 구성에서 소유자 재정의 옵션을 제거할 경우

    Amazon S3는 원본 버킷에 나타나는 새 객체 및 연결된 ACL을 대상 버킷으로 복제합니다. 소유자 재정의를 제거하기 전에 복제된 객체의 경우 Amazon S3가 ACL을 복제하지 않습니다. Amazon S3이 수행한 객체 소유권 변경이 유효하기 때문입니다. 즉, 소유자 재정의가 설정되었을 때 복제된 객체 버전에 적용된 ACL은 계속해서 복제되지 않습니다.

AWS KMS에 저장된 암호화 키를 사용한 서버 측 암호화(SSE)로 생성된 객체 복제

기본적으로 Amazon S3은 AWS KMS에 저장된 고객 마스터 키(CMK)를 사용한 서버 측 암호화로 유휴 시 저장된 객체를 복제하지 않습니다. 이 단원에서는 Amazon S3에 이러한 객체를 복제하도록 지시하기 위해 추가하는 추가 구성을 설명합니다.

> **Important**
>
> 암호화된 데이터의 복제는 완전히 Amazon S3 내에서 수행되는 서버 측 프로세스입니다. 복제는 클라이언트 측 암호화를 지원하지 않습니다.

단계별 지침이 포함된 예제는 예제 4: 암호화된 객체 복제 (p. 647) 단원을 참조하십시오. 복제 구성 생성에 대한 자세한 내용은 복제 (p. 607) 단원을 참조하십시오.

주제

- 복제 구성에서 추가 정보 지정 (p. 629)
- IAM 역할에 추가 권한 부여 (p. 630)
- 교차 계정 시나리오를 위한 추가 권한 부여 (p. 632)
- AWS KMS 트랜잭션 제한 고려 사항 (p. 633)

복제 구성에서 추가 정보 지정

복제 구성에서 다음을 수행합니다.

- `Destination` 구성에서 Amazon S3이 객체 복제본을 암호화하는 데 사용할 대칭 고객 관리형 AWS KMS CMK를 추가합니다.
- `SourceSelectionCriteria` 요소를 추가하여 AWS KMS CMK로 암호화된 객체 복제를 활성화함으로써 명시적으로 옵트인합니다.

```
<ReplicationConfiguration>
   <Rule>
      ...
      <SourceSelectionCriteria>
         <SseKmsEncryptedObjects>
           <Status>Enabled</Status>
         </SseKmsEncryptedObjects>
      </SourceSelectionCriteria>

      <Destination>
          ...
          <EncryptionConfiguration>
             <ReplicaKmsKeyID>AWS KMS key ID for the AWS region of the destination
 bucket.</ReplicaKmsKeyID>
          </EncryptionConfiguration>
       </Destination>
      ...
   </Rule>
</ReplicationConfiguration>
```

> **Important**
>
> AWS KMS CMK는 대상 버킷과 동일한 AWS 리전에서 생성되어야 합니다.
> AWS KMS CMK가 유효해야 합니다. `PUT` 버킷 복제 API는 AWS KMS CMK가 유효한지 확인하지 않습니다. 유효하지 않은 CMK를 사용할 경우 응답에서 200 OK 상태 코드를 받지만 복제는 실패합니다.

다음은 선택적 구성 요소가 포함된 복제 구성의 예입니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration>
   <Role>arn:aws:iam::account-id:role/role-name</Role>
   <Rule>
      <ID>Rule-1</ID>
      <Priority>1</Priority>
      <Status>Enabled</Status>
      <DeleteMarkerReplication>
         <Status>Disabled</Status>
      </DeleteMarkerReplication>
      <Filter>
         <Prefix>Tax</Prefix>
      </Filter>
      <Destination>
         <Bucket>arn:aws:s3:::destination-bucket</Bucket>
         <EncryptionConfiguration>
            <ReplicaKmsKeyID>The AWS KMS key ID for the AWS region of the destination
 bucket (S3 uses it to encrypt object replicas).</ReplicaKmsKeyID>
         </EncryptionConfiguration>
      </Destination>
      <SourceSelectionCriteria>
         <SseKmsEncryptedObjects>
            <Status>Enabled</Status>
         </SseKmsEncryptedObjects>
      </SourceSelectionCriteria>
   </Rule>
</ReplicationConfiguration>
```

이 복제 구성에는 한 가지 규칙이 있습니다. 이 규칙은 키 접두사가 `Tax`인 객체에 적용됩니다. Amazon S3는 AWS KMS 키 ID를 사용해 이러한 객체 복제본을 암호화합니다.

IAM 역할에 추가 권한 부여

AWS Key Management Service(AWS KMS)에서 유휴 시 암호화된 객체를 복제하려면 복제 구성에 지정하는 IAM 역할에 다음 추가 권한을 부여합니다. IAM 역할과 연결된 권한 정책을 업데이트하여 이러한 권한을 부여합니다. 고객 제공 암호화 키를 사용하는 서버 측 암호화(SSE-C)로 생성된 객체는 복제되지 않습니다.

- 원본 객체에 대한 `s3:GetObjectVersionForReplication` 작업 권한. 이 작업 권한은 Amazon S3이 암호화되지 않은 객체와 Amazon S3 관리형 암호화 키를 사용한 서버 측 암호화(SSE-S3) 또는 AWS KMS에 저장된 CMK를 사용한 서버 측 암호화(SSE-KMS)로 생성된 객체를 모두 복제할 수 있도록 허용합니다.

  Note

  Amazon S3에 복제에 필요한 최소한의 권한만 제공하므로 `s3:GetObjectVersion` 작업 대신 `s3:GetObjectVersionForReplication` 작업을 사용하는 것이 좋습니다. 또한 `s3:GetObjectVersion` 작업 권한은 암호화되지 않은 객체와 SSE-S3 암호화 객체의 복제를 허용하지만 AWS KMS에 저장된 CMK를 사용하여 생성된 객체의 복제는 허용하지 않습니다.
- 다음 AWS KMS 작업에 대한 권한:
  - 원본 객체를 암호화하는 데 사용된 AWS KMS CMK에 대한 `kms:Decrypt` 권한
  - 객체 복제본을 암호화하는 데 사용된 AWS KMS CMK에 대한 `kms:Encrypt` 권한

  다음 예제 정책 설명과 같이 이러한 권한을 AWS KMS 조건 키를 사용하는 특정 버킷 및 객체로 제한하는 것이 좋습니다.

```
{
    "Action": ["kms:Decrypt"],
    "Effect": "Allow",
    "Condition": {
        "StringLike": {
```

```
            "kms:ViaService": "s3.source-bucket-region.amazonaws.com",
            "kms:EncryptionContext:aws:s3:arn": [
                "arn:aws:s3:::source-bucket-name/key-prefix1*",
            ]
        }
    },
    "Resource": [
        "List of AWS KMS key IDs used to encrypt source objects.",
    ]
},
{
    "Action": ["kms:Encrypt"],
    "Effect": "Allow",
    "Condition": {
        "StringLike": {
            "kms:ViaService": "s3.destination-bucket-region.amazonaws.com",
            "kms:EncryptionContext:aws:s3:arn": [
                "arn:aws:s3:::destination-bucket-name/key-prefix1*",
            ]
        }
    },
    "Resource": [
         "AWS KMS key IDs (for the AWS region of the destination bucket). S3 uses it to
 encrypt object replicas",
    ]
}
```

IAM 역할을 소유한 AWS 계정은 정책에 나열된 AWS KMS CMK의 이러한 AWS KMS 작업 (`kms:Encrypt` 및 `kms:Decrypt`)에 대한 권한이 있어야 합니다. 다른 AWS 계정이 AWS KMS CMK를 소유한 경우 CMK 소유자는 IAM 역할을 소유한 AWS 계정에 이러한 권한을 부여해야 합니다. 이러한 CMK에 대한 액세스를 관리하는 방법에 대한 자세한 내용은 AWS Key Management Service Developer Guide의 AWS KMS에서 IAM 정책 사용을 참조하십시오.

다음은 암호화되지 않은 객체, Amazon S3 관리형 암호화 키를 사용한 서버 측 암호화로 생성된 객체 및 AWS KMS에 저장된 CMK를 사용한 서버 측 암호화로 생성된 객체를 복제하는 데 필요한 권한을 부여하는 완전한 IAM 정책입니다.

> **Note**
>
> 고객 제공 암호화 키를 사용하는 서버 측 암호화(SSE-C)로 생성된 객체는 복제되지 않습니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::source-bucket"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl"
         ],
         "Resource":[
            "arn:aws:s3:::source-bucket/key-prefix1*"
```

```
            ]
        },
        {
            "Effect":"Allow",
            "Action":[
                "s3:ReplicateObject",
                "s3:ReplicateDelete"
            ],
            "Resource":"arn:aws:s3:::destination-bucket/key-prefix1*"
        },
        {
            "Action":[
                "kms:Decrypt"
            ],
            "Effect":"Allow",
            "Condition":{
                "StringLike":{
                    "kms:ViaService":"s3.source-bucket-region.amazonaws.com",
                    "kms:EncryptionContext:aws:s3:arn":[
                        "arn:aws:s3:::source-bucket-name/key-prefix1*"
                    ]
                }
            },
            "Resource":[
              "List of AWS KMS key IDs used to encrypt source objects."
            ]
        },
        {
            "Action":[
                "kms:Encrypt"
            ],
            "Effect":"Allow",
            "Condition":{
                "StringLike":{
                    "kms:ViaService":"s3.destination-bucket-region.amazonaws.com",
                    "kms:EncryptionContext:aws:s3:arn":[
                        "arn:aws:s3:::destination-bucket-name/prefix1*"
                    ]
                }
            },
            "Resource":[
                "AWS KMS key IDs (for the AWS region of the destination bucket) to use for
 encrypting object replicas"
            ]
        }
    ]
}
```

교차 계정 시나리오를 위한 추가 권한 부여

버킷과 *##* 버킷을 서로 다른 AWS 계정에서 소유하는 교차 계정 시나리오에서는 고객 관리형 CMK를 사용하여 객체 복제본을 암호화할 수 있습니다. 그러나 CMK 소유자는 원본 버킷 소유자에게 CMK를 사용할 수 있는 권한을 부여해야 합니다.

AWS KMS CMK를 사용할 수 있는 권한을 원본 버킷 소유자에게 부여하려면(IAM 콘솔)

1. AWS Management 콘솔에 로그인한 후 https://console.aws.amazon.com/kms에서 AWS Key Management Service(AWS KMS) 콘솔을 엽니다.
2. AWS 리전을 변경하려면 페이지의 오른쪽 위 모서리에 있는 리전 선택기를 사용합니다.
3. 해당 계정에서 직접 생성하고 관리하는 키를 보려면 탐색 창에서 고객 관리형 키를 선택합니다.
4. CMK를 선택합니다.
5. General configuration(일반 구성)에서 Key policy(키 정책) 탭을 선택합니다.

6. Other AWS Accounts(다른 AWS 계정)를 선택합니다.
7. Add another AWS Account(다른 AWS 계정 추가)를 선택합니다.
8. arn:aws:iam::에 원본 버킷 계정 ID를 입력합니다.
9. 변경 사항 저장을 선택합니다.

AWS KMS CMK를 사용할 수 있는 권한을 원본 버킷 소유자에게 부여하려면(AWS CLI)

- 자세한 내용은 AWS CLI 명령 참조의 put-key-policy를 참조하십시오. 기본 API에 대한 자세한 내용은 AWS Key Management Service API Reference의 PutKeyPolicy를 참조하십시오.

AWS KMS 트랜잭션 제한 고려 사항

교차 리전 복제(CRR)을 활성화한 후, AWS KMS 암호화로 많은 수의 새 객체를 추가할 때 조절(HTTP 503 속도 하락 오류)가 발생할 수 있습니다. 조절은 AWS KMS 초당 KMS 트랜잭션 수가 현재 제한을 초과할 경우 발생합니다. 자세한 내용은 AWS Key Management Service Developer Guide의 제한 단원을 참조하십시오.

제한 증가를 요청하려면 Service Quotas 단원을 참조하십시오. 자세한 내용은 Amazon Web Services 제한을 참조하십시오. Service Quotas가 해당 리전에서 지원되지 않는 경우 AWS Support 사례를 엽니다.

복제 실습 과정

다음 예제에서는 일반 사용 사례를 위한 복제 구성 방법을 보여줍니다. 이 예제에서는 Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 및 AWS SDK를 사용하여 복제 구성을 보여줍니다(Java 및 .NET SDK 예제 표시). AWS CLI 설치 및 구성에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서에서 다음 단원을 참조하십시오.

- AWS 명령줄 인터페이스 설치
- AWS CLI 구성 - 하나 이상의 프로필을 설정해야 합니다. 교차 계정 시나리오에 대해 살펴보려면 두 개의 프로필을 설정해야 합니다.

AWS SDK에 대한 자세한 내용은 Java용 AWS SDK 및 .NET용 AWS SDK 단원을 참조하십시오.

주제

- 예제 1 | 원본 버킷과 대상 버킷을 동일한 계정에서 소유한 경우 복제 구성 (p. 633)
- 예제 2: 원본 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유한 경우 복제를 구성 (p. 642)
- 예제 3: 원본 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유한 경우 복제본 소유자 변경 (p. 643)
- 예제 4: 암호화된 객체 복제 (p. 647)
- 예 5: S3 Replication Time Control (S3 RTC) 구성 (p. 651)

예제 1 | 원본 버킷과 대상 버킷을 동일한 계정에서 소유한 경우 복제 구성

이 예제에서는 원본 버킷과 대상 버킷을 동일한 AWS 계정에서 소유한 복제를 설정합니다. Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 및 AWS SDK for Java/.NET용 AWS SDK를 사용한 예제가 제공됩니다.

동일한 계정(콘솔)에서 버킷을 소유할 때 복제 구성

단계별 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷에 복제 규칙을 추가하는 방법 단원을 참조하십시오. 이 주제에서는 버킷을 동일한 AWS 계정이 소유한 경우와 서로 다른 AWS 계정이 소유한 경우에 복제 구성을 설정하는 지침을 제공합니다.

동일한 계정(AWS CLI)에서 버킷을 소유할 때 복제 구성

원본 버킷과 대상 버킷을 동일한 AWS 계정이 소유한 경우 AWS CLI를 사용하여 복제를 설정하려면 원본 및 대상 버킷을 생성하고, 이들 버킷에서 버전 관리를 활성화하고, Amazon S3에 객체를 복제할 권한을 부여하는 IAM 역할을 생성하고, 복제 구성을 원본 버킷에 추가합니다. 설정을 확인하려면 테스트합니다.

원본 버킷과 대상 버킷을 동일한 AWS 계정에서 소유한 경우 복제를 설정하려면

1. AWS CLI의 자격 증명 프로필을 설정합니다. 이 예제에서는 프로필 이름 `acctA`를 사용합니다. 자격 증명 프로필 설정에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 명명된 프로필 단원을 참조하십시오.

   > **Important**
   >
   > 이 연습에 사용하는 프로필에는 필요한 권한이 있어야 합니다. 예를 들어 복제 구성에서 Amazon S3가 맡을 수 있는 IAM 역할을 지정합니다. 사용하는 프로필에 `iam:PassRole` 권한이 있을 경우 이 권한만 사용할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 AWS 서비스에 역할을 전달할 수 있는 권한 부여 단원을 참조하십시오. 관리자 사용자 자격 증명을 사용하여 명명된 프로필을 생성할 경우 모든 작업을 수행할 수 있습니다.

2. *##* 버킷을 생성하고 버킷에서 버전 관리를 활성화합니다. 다음 코드는 미국 동부(버지니아 북부)(us-east-1) 리전에 *##* 버킷을 생성합니다.

```
aws s3api create-bucket \
--bucket source \
--region us-east-1 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket source \
--versioning-configuration Status=Enabled \
--profile acctA
```

3. *##* 버킷을 생성하고 버킷에서 버전 관리를 활성화합니다. 다음 코드는 미국 서부(오레곤)(us-west-2) 리전에 *##* 버킷을 생성합니다.

   > **Note**
   >
   > 원본 버킷과 대상 버킷이 모두 동일한 AWS 계정에 있을 때 복제 구성을 설정하려면 동일한 프로필을 사용합니다. 이 예제에서는 `acctA`를 사용합니다. 두 버킷을 서로 다른 AWS 계정에서 소유한 경우의 복제 구성을 테스트하려면 각각 다른 프로필을 지정합니다. 이 예제에서는 대상 버킷에 대한 `acctB` 프로필을 사용합니다.

```
aws s3api create-bucket \
--bucket destination \
--region us-west-2 \
--create-bucket-configuration LocationConstraint=us-west-2 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket destination \
--versioning-configuration Status=Enabled \
--profile acctA
```

4. IAM 역할을 생성합니다. 나중에 ## 버킷에 추가하는 복제 구성에서 이 역할을 지정합니다. Amazon S3 가 사용자 대신 객체를 복제하기 위해 이 역할을 맡습니다. IAM 역할을 두 단계로 생성합니다.

   - 역할을 생성합니다.
   - 역할에 권한 정책을 연결합니다.

   a. IAM 역할을 생성합니다.

      i. 다음 신뢰 정책을 복사하여 로컬 컴퓨터의 현재 디렉터리에 `s3-role-trust-policy.json`이라는 이름의 파일로 저장합니다. 이 정책은 Amazon S3 서비스 보안 주체에 역할을 맡을 권한을 부여합니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

      ii. 다음 명령을 실행해 역할을 생성합니다.

```
$ aws iam create-role \
--role-name replicationRole \
--assume-role-policy-document file://s3-role-trust-policy.json  \
--profile acctA
```

   b. 역할에 권한 정책을 연결합니다.

      i. 다음 권한 정책을 복사하여 로컬 컴퓨터의 현재 디렉터리에 `s3-role-permissions-policy.json` 파일로 저장합니다. 이 정책은 다양한 Amazon S3 버킷 및 객체 작업에 대한 권한을 부여합니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl"
         ],
         "Resource":[
            "arn:aws:s3:::source-bucket/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket",
            "s3:GetReplicationConfiguration"
         ],
         "Resource":[
            "arn:aws:s3:::source-bucket"
         ]
      },
```

```
        {
           "Effect":"Allow",
           "Action":[
              "s3:ReplicateObject",
              "s3:ReplicateDelete",
              "s3:ReplicateTags",
              "s3:GetObjectVersionTagging"

           ],
           "Resource":"arn:aws:s3:::destination-bucket/*"
        }
    ]
}
```

ii. 다음 명령을 실행하여 정책을 생성하고 이를 역할에 연결합니다.

```
$ aws iam put-role-policy \
--role-name replicationRole \
--policy-document file://s3-role-permissions-policy.json \
--policy-name replicationRolePolicy \
--profile acctA
```

5. *##* 버킷에 복제 구성을 추가합니다.

   a. Amazon S3 API는 XML 형식의 복제 구성을 요구하지만, AWS CLI는 JSON으로 지정된 복제 구성을 요구합니다. 다음 JSON을 로컬 컴퓨터의 현재 디렉터리에 `replication.json` 파일로 저장합니다.

```
{
  "Role": "IAM-role-ARN",
  "Rules": [
    {
      "Status": "Enabled",
      "Priority": 1,
      "DeleteMarkerReplication": { "Status": "Disabled" },
      "Filter" : { "Prefix": "Tax"},
      "Destination": {
        "Bucket": "arn:aws:s3:::destination-bucket"
      }
    }
  ]
}
```

   b. *destination-bucket* 및 *IAM-role-ARN*에 값을 제공하여 JSON을 업데이트합니다. 변경 사항을 저장합니다.

   c. 다음 명령을 실행하여 원본 버킷에 복제 구성을 추가합니다. 반드시 *##* 버킷 이름을 제공해야 합니다.

```
$ aws s3api put-bucket-replication \
--replication-configuration file://replication.json \
--bucket source \
--profile acctA
```

   복제 구성을 검색하려면 `get-bucket-replication` 명령을 사용합니다.

```
$ aws s3api get-bucket-replication \
--bucket source \
--profile acctA
```

6. Amazon S3 콘솔에서 설정을 테스트합니다.

a. AWS Management Console에 로그인하고 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.

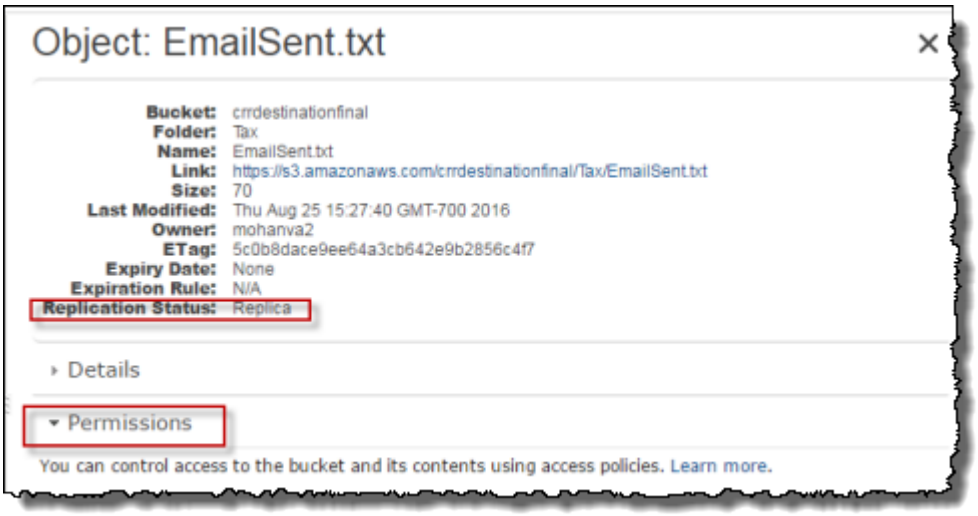
b. ## 버킷에서 `Tax` 폴더를 생성합니다.

c. ## 버킷의 `Tax` 폴더에 샘플 객체를 추가합니다.

> **Note**
>
> Amazon S3가 객체를 복제하는 데 걸리는 시간은 객체 크기에 따라 다릅니다. 복제 상태를 확인하는 방법에 대한 자세한 내용은 복제 상태 정보 (p. 654) 단원을 참조하십시오.

버킷에서 다음을 확인합니다.

- Amazon S3가 객체를 복제함.
- 객체 속성에서, 복제 상태가 `Replica`로 설정됨(이 객체를 복제본 객체로 식별).
- 객체 속성에서, 권한 섹션에 아무 권한이 없음. 이는 복제본이 여전히 ## 버킷 소유자의 소유이고 ## 버킷 소유자는 객체 복제본에 대한 권한이 없음을 의미합니다. Amazon S3에 복제본 소유권을 변경하도록 지시하는 선택적 구성을 추가할 수 있습니다. 문제 해결 예는 예제 3: 원본 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유한 경우 복제본 소유자 변경 (p. 643) 단원을 참조하십시오.



d. ## 버킷의 객체 ACL을 업데이트하고, ## 버킷에 변경 내용이 나타나는지 확인합니다.

지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 객체에 대한 권한은 어떻게 설정하나요? 단원을 참조하십시오.

동일한 계정(AWS SDK)에서 버킷을 소유할 때 복제 구성

다음 코드 예제를 사용하여 AWS SDK for Java 및 .NET용 AWS SDK를 각각 사용하여 버킷에 복제 구성을 추가합니다.

Java

다음 예에서는 버킷에 복제 구성을 추가한 후 해당 구성을 검색 및 확인합니다. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 Amazon S3 Java 코드 예제 테스트 (p. 735) 단원을 참조하십시오.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
```

```
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.CreateRoleRequest;
import com.amazonaws.services.identitymanagement.model.PutRolePolicyRequest;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.BucketReplicationConfiguration;
import com.amazonaws.services.s3.model.BucketVersioningConfiguration;
import com.amazonaws.services.s3.model.CreateBucketRequest;
import com.amazonaws.services.s3.model.DeleteMarkerReplication;
import com.amazonaws.services.s3.model.DeleteMarkerReplicationStatus;
import com.amazonaws.services.s3.model.ReplicationDestinationConfig;
import com.amazonaws.services.s3.model.ReplicationRule;
import com.amazonaws.services.s3.model.ReplicationRuleStatus;
import com.amazonaws.services.s3.model.SetBucketVersioningConfigurationRequest;
import com.amazonaws.services.s3.model.StorageClass;
import com.amazonaws.services.s3.model.replication.ReplicationFilter;
import com.amazonaws.services.s3.model.replication.ReplicationFilterPredicate;
import com.amazonaws.services.s3.model.replication.ReplicationPrefixPredicate;

import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class CrossRegionReplication {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String accountId = "*** Account ID ***";
        String roleName = "*** Role name ***";
        String sourceBucketName = "*** Source bucket name ***";
        String destBucketName = "*** Destination bucket name ***";
        String prefix = "Tax/";

        String roleARN = String.format("arn:aws:iam::%s:role/%s", accountId, roleName);
        String destinationBucketARN = "arn:aws:s3:::" + destBucketName;

        AmazonS3 s3Client = AmazonS3Client.builder()
            .withCredentials(new ProfileCredentialsProvider())
            .withRegion(clientRegion)
            .build();

        createBucket(s3Client, clientRegion, sourceBucketName);
        createBucket(s3Client, clientRegion, destBucketName);
        assignRole(roleName, clientRegion, sourceBucketName, destBucketName);


        try {


            // Create the replication rule.
            List<ReplicationFilterPredicate> andOperands = new
 ArrayList<ReplicationFilterPredicate>();
            andOperands.add(new ReplicationPrefixPredicate(prefix));


            Map<String, ReplicationRule> replicationRules = new HashMap<String,
 ReplicationRule>();
            replicationRules.put("ReplicationRule1",
                new ReplicationRule()
                    .withPriority(0)
                    .withStatus(ReplicationRuleStatus.Enabled)
```

```
                    .withDeleteMarkerReplication(new
 DeleteMarkerReplication().withStatus(DeleteMarkerReplicationStatus.DISABLED))
                    .withFilter(new ReplicationFilter().withPredicate(new
 ReplicationPrefixPredicate(prefix)))
                    .withDestinationConfig(new ReplicationDestinationConfig()
                        .withBucketARN(destinationBucketARN)
                        .withStorageClass(StorageClass.Standard)));

            // Save the replication rule to the source bucket.
            s3Client.setBucketReplicationConfiguration(sourceBucketName,
                new BucketReplicationConfiguration()
                    .withRoleARN(roleARN)
                    .withRules(replicationRules));

            // Retrieve the replication configuration and verify that the configuration
            // matches the rule we just set.
            BucketReplicationConfiguration replicationConfig =
 s3Client.getBucketReplicationConfiguration(sourceBucketName);
            ReplicationRule rule = replicationConfig.getRule("ReplicationRule1");
            System.out.println("Retrieved destination bucket ARN: " +
 rule.getDestinationConfig().getBucketARN());
            System.out.println("Retrieved priority: " + rule.getPriority());
            System.out.println("Retrieved source-bucket replication rule status: " +
 rule.getStatus());
        } 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();
        }
    }

    private static void createBucket(AmazonS3 s3Client, Regions region, String
 bucketName) {
        CreateBucketRequest request = new CreateBucketRequest(bucketName,
 region.getName());
        s3Client.createBucket(request);
        BucketVersioningConfiguration configuration = new
 BucketVersioningConfiguration().withStatus(BucketVersioningConfiguration.ENABLED);

        SetBucketVersioningConfigurationRequest enableVersioningRequest = new
 SetBucketVersioningConfigurationRequest(bucketName, configuration);
        s3Client.setBucketVersioningConfiguration(enableVersioningRequest);


    }

    private static void assignRole(String roleName, Regions region, String
 sourceBucket, String destinationBucket) {
        AmazonIdentityManagement iamClient =
 AmazonIdentityManagementClientBuilder.standard()
            .withRegion(region)
            .withCredentials(new ProfileCredentialsProvider())
            .build();
        StringBuilder trustPolicy = new StringBuilder();
        trustPolicy.append("{\\r\\n   ");
        trustPolicy.append("\\\"Version\\\":\\\"2012-10-17\\\",\\r\\n   ");
        trustPolicy.append("\\\"Statement\\\":[\\r\\n      {\\r\\n         ");
        trustPolicy.append("\\\"Effect\\\":\\\"Allow\\\",\\r\\n         \\\"Principal\\
\":{\\r\\n            ");
        trustPolicy.append("\\\"Service\\\":\\\"s3.amazonaws.com\\\"\\r\\n         },\
\r\\n         ");
```

```
        trustPolicy.append("\\\"Action\\\":\\\"sts:AssumeRole\\\"\\r\\n      }\\r\\n
  ]\\r\\n}");

        CreateRoleRequest createRoleRequest = new CreateRoleRequest()
            .withRoleName(roleName)
            .withAssumeRolePolicyDocument(trustPolicy.toString());

        iamClient.createRole(createRoleRequest);

        StringBuilder permissionPolicy = new StringBuilder();
        permissionPolicy.append("{\\r\\n   \\\"Version\\\":\\\"2012-10-17\\\",\\r\\n
 \\\"Statement\\\":[\\r\\n      {\\r\\n          ");
        permissionPolicy.append("\\\"Effect\\\":\\\"Allow\\\",\\r\\n         \\\"Action
\\\":[\\r\\n             ");
        permissionPolicy.append("\\\"s3:GetObjectVersionForReplication\\\",\\r\\n
      ");
        permissionPolicy.append("\\\"s3:GetObjectVersionAcl\\\"\\r\\n         ],\\r\\n
        \\\"Resource\\\":[\\r\\n            ");
        permissionPolicy.append("\\\"arn:aws:s3:::");
        permissionPolicy.append(sourceBucket);
        permissionPolicy.append("/*\\\"\\r\\n         ]\\r\\n      },\\r\\n      {\\r\
\n         ");
        permissionPolicy.append("\\\"Effect\\\":\\\"Allow\\\",\\r\\n         \\\"Action
\\\":[\\r\\n            ");
        permissionPolicy.append("\\\"s3:ListBucket\\\",\\r\\n            \\
\"s3:GetReplicationConfiguration\\\"\\r\\n         ");
        permissionPolicy.append("],\\r\\n         \\\"Resource\\\":[\\r\\n            \
\\"arn:aws:s3:::");
        permissionPolicy.append(sourceBucket);
        permissionPolicy.append("\\r\\n         ");
        permissionPolicy.append("]\\r\\n      },\\r\\n      {\\r\\n         \\\"Effect\
\\":\\\"Allow\\\",\\r\\n         ");
        permissionPolicy.append("\\\"Action\\\":[\\r\\n            \\
\"s3:ReplicateObject\\\",\\r\\n            ");
        permissionPolicy.append("\\\"s3:ReplicateDelete\\\",\\r\\n            \\
\"s3:ReplicateTags\\\",\\r\\n            ");
        permissionPolicy.append("\\\"s3:GetObjectVersionTagging\\\"\\r\\n\\r\\n
  ],\\r\\n         ");
        permissionPolicy.append("\\\"Resource\\\":\\\"arn:aws:s3:::");
        permissionPolicy.append(destinationBucket);
        permissionPolicy.append("/*\\\"\\r\\n      }\\r\\n   ]\\r\\n}");

        PutRolePolicyRequest putRolePolicyRequest = new PutRolePolicyRequest()
            .withRoleName(roleName)
            .withPolicyDocument(permissionPolicy.toString())
            .withPolicyName("crrRolePolicy");

        iamClient.putRolePolicy(putRolePolicyRequest);


    }
}
```

C#

다음은 버킷에 복제 구성을 추가한 다음 해당 구성을 검색하는 .NET용 AWS SDK 코드의 예입니다. 이 코드를 사용하려면 버킷 이름과 IAM 역할의 Amazon 리소스 이름(ARN)을 제공합니다. 실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

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

```
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CrossRegionReplicationTest
    {
        private const string sourceBucket = "*** source bucket ***";
        // Bucket ARN example - arn:aws:s3:::destinationbucket
        private const string destinationBucketArn = "*** destination bucket ARN ***";
        private const string roleArn = "*** IAM Role ARN ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint sourceBucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {
            s3Client = new AmazonS3Client(sourceBucketRegion);
            EnableReplicationAsync().Wait();
        }
        static async Task EnableReplicationAsync()
        {
            try
            {
                ReplicationConfiguration replConfig = new ReplicationConfiguration
                {
                    Role = roleArn,
                    Rules =
                        {
                            new ReplicationRule
                            {
                                Prefix = "Tax",
                                Status = ReplicationRuleStatus.Enabled,
                                Destination = new ReplicationDestination
                                {
                                    BucketArn = destinationBucketArn
                                }
                            }
                        }
                };

                PutBucketReplicationRequest putRequest = new
 PutBucketReplicationRequest
                {
                    BucketName = sourceBucket,
                    Configuration = replConfig
                };

                PutBucketReplicationResponse putResponse = await
 s3Client.PutBucketReplicationAsync(putRequest);

                // Verify configuration by retrieving it.
                await RetrieveReplicationConfigurationAsync(s3Client);
            }
            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);
            }
        }
```

```
        private static async Task RetrieveReplicationConfigurationAsync(IAmazonS3
 client)
        {
            // Retrieve the configuration.
            GetBucketReplicationRequest getRequest = new GetBucketReplicationRequest
            {
                BucketName = sourceBucket
            };
            GetBucketReplicationResponse getResponse = await
 client.GetBucketReplicationAsync(getRequest);
            // Print.
            Console.WriteLine("Printing replication configuration information...");
            Console.WriteLine("Role ARN: {0}", getResponse.Configuration.Role);
            foreach (var rule in getResponse.Configuration.Rules)
            {
                Console.WriteLine("ID: {0}", rule.Id);
                Console.WriteLine("Prefix: {0}", rule.Prefix);
                Console.WriteLine("Status: {0}", rule.Status);
            }
        }
    }
}
```

예제 2: 원본 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유한 경우 복제를 구성

서로 다른 AWS 계정이 *##* 버킷과 *##* 버킷을 소유한 경우의 복제 설정은 동일한 계정이 두 버킷을 모두 소유한 경우와 비슷합니다. 유일한 차이는 *##* 버킷 소유자가 버킷 정책을 추가하여 *##* 버킷 소유자에게 객체를 복제할 권한을 부여해야 한다는 것입니다.

원본 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유한 경우 복제를 구서하려면

1. 이 예제에서는 *##* 버킷과 *##* 버킷을 두 AWS 계정에서 생성합니다. AWS CLI에 대해 두 개의 자격 증명 프로필을 설정해야 합니다(이 예제에서는 프로필 이름으로 `acctA` 및 `acctB`를 사용). 자격 증명 프로필 설정에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 명명된 프로필 단원을 참조하십시오.
2. 예제 1: 동일한 계정에서 버킷 구성 (p. 633)의 단계별 지침을 따르되 다음 사항을 변경합니다.
   - *##* 버킷 활동(*##* 버킷 생성, 버전 관리 활성화, IAM 역할 생성)과 관련된 모든 AWS CLI 명령은 `acctA` 프로필을 사용합니다. `acctB` 프로필을 사용해 *##* 버킷을 생성합니다.
   - 권한 정책이 이 예제에서 생성한 *##* 및 *##* 버킷을 지정해야 합니다.
3. 콘솔에서 *##* 버킷에 다음 버킷 정책을 추가해 *##* 버킷 소유자가 객체를 복제하도록 허용합니다. 반드시 *##* 버킷 소유자의 AWS 계정 ID 및 *##* 버킷 이름을 제공하여 정책을 편집해야 합니다.

```
{
   "Version":"2008-10-17",
   "Id":"",
   "Statement":[
      {
         "Sid":"1",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::source-bucket-owner-AWS-acct-ID:root"
         },
         "Action":["s3:ReplicateObject", "s3:ReplicateDelete"],
         "Resource":"arn:aws:s3:::destination/*"
      },
      {
```

```
            "Sid":"2",
            "Effect":"Allow",
            "Principal":{
               "AWS":"arn:aws:iam::source-bucket-owner-AWS-acct-ID:root"
            },
            "Action":["s3:GetBucketVersioning", "s3:PutBucketVersioning"],
            "Resource":"arn:aws:s3:::destination"
         }
      ]
}
```

버킷을 선택하고 버킷 정책을 추가합니다. 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷 정책을 추가하려면 어떻게 해야 합니까? 단원을 참조하십시오.

예제 3: 원본 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유한 경우 복제본 소유자 변경

복제의 ## 버킷과 ## 버킷을 서로 다른 AWS 계정이 소유한 경우 Amazon S3에게 복제본 소유권을 ## 버킷을 소유한 AWS 계정으로 변경하도록 지시할 수 있습니다. 이 예제에서는 Amazon S3 콘솔 및 AWS CLI를 사용하여 복제본 소유권을 변경하는 방법을 설명합니다. 자세한 내용은 복제본 소유자 변경 (p. 626) 단원을 참조하십시오.

버킷을 서로 다른 AWS 계정에서 소유한 경우 복제본 소유자 변경(콘솔)

단계별 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 대상 버킷이 다른 AWS 계정에 있는 경우 복제 규칙 구성 단원을 참조하십시오.

버킷을 서로 다른 AWS 계정에서 소유한 경우 복제본 소유자 변경(AWS CLI)

AWS CLI를 사용하여 복제본 소유권을 변경하려면 버킷을 생성하고, 버킷에서 버전 관리를 활성화하고, Amazon S3에 객체를 복제할 권한을 부여하는 IAM 역할을 생성하고, 복제 구성을 원본 버킷에 추가합니다. 복제 구성에서 Amazon S3에 복제본 소유자를 변경하도록 지시합니다. 또한 다음 설정을 테스트합니다.

원본 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유한 경우 복제본 소유권을 변경하려면(AWS CLI)

1. 이 예제에서는 ## 버킷과 ## 버킷을 두 AWS 계정에서 생성합니다. 2개의 명명된 프로필을 포함하는 AWS CLI를 구성합니다. 이 예제에서는 이름이 `acctA` 및 `acctB`인 프로필을 사용합니다. 자격 증명 프로필 설정에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 명명된 프로필 단원을 참조하십시오.

   > **Important**
   >
   > 이 연습에 사용하는 프로필에 필요한 권한이 있어야 합니다. 예를 들어 복제 구성에서 Amazon S3가 맡을 수 있는 IAM 역할을 지정합니다. 사용하는 프로필에 `iam:PassRole` 권한이 있을 경우 이 권한만 사용할 수 있습니다. 관리자 사용자 자격 증명을 사용하여 명명된 프로필을 생성할 경우 모든 작업을 수행할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 AWS 서비스에 역할을 전달할 수 있는 권한 부여 단원을 참조하십시오.

   이들 프로필에 필요한 권한이 있는지 확인해야 합니다. 예를 들어 복제 구성이 Amazon S3가 맡을 수 있는 IAM 역할을 포함합니다. 그러한 구성을 버킷에 연결하는 데 사용하는 명명된 프로필은 `iam:PassRole` 권한이 있어야만 연결 작업을 수행할 수 있습니다. 이러한 명명된 프로필을 생성할 때 관리자 사용자 자격 증명을 지정할 경우 모든 권한이 부여됩니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 AWS 서비스에 역할을 전달할 수 있는 권한 부여 단원을 참조하십시오.
2. ## 버킷을 생성하고 버전 관리를 활성화합니다. 이 예제에서는 ## 버킷을 미국 동부(버지니아 북부)(us-east-1) 리전에서 생성합니다.

```
aws s3api create-bucket \
--bucket source \
--region us-east-1 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket source \
--versioning-configuration Status=Enabled \
--profile acctA
```

3. *##* 버킷을 생성하고 버전 관리를 활성화합니다. 이 예제에서는 *##* 버킷을 미국 서부(오레곤)(us-west-2) 리전에서 생성합니다. *##* 버킷을 생성할 때 사용한 것과 다른 AWS 계정 프로필을 사용합니다.

```
aws s3api create-bucket \
--bucket destination \
--region us-west-2 \
--create-bucket-configuration LocationConstraint=us-west-2 \
--profile acctB
```

```
aws s3api put-bucket-versioning \
--bucket destination \
--versioning-configuration Status=Enabled \
--profile acctB
```

4. *##* 버킷 정책에 복제본 소유권 변경을 허용할 권한을 추가해야 합니다.

   a. 다음 정책을 *destination-bucket-policy*.json로 저장합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "destination_bucket_policy_sid",
      "Principal": {
        "AWS": "source-bucket-owner-account-id"
      },
      "Action": [
        "s3:ReplicateObject",
        "s3:ReplicateDelete"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::destination/*"
      ]
    }
  ]
}
```

   b. *##* 버킷에 위의정책을 적용합니다.

```
aws s3api put-bucket-policy --region $ {destination_region} --
bucket $ {destination} --policy file://destination_bucket_policy.json
```

5. IAM 역할을 생성합니다. 나중에 *##* 버킷에 추가하는 복제 구성에서 이 역할을 지정합니다. Amazon S3가 사용자 대신 객체를 복제하기 위해 이 역할을 맡습니다. IAM 역할을 두 단계로 생성합니다.

   - 역할을 생성합니다.
   - 역할에 권한 정책을 연결합니다.

a. IAM 역할 생성.

   i. 다음 신뢰 정책을 복사하여 로컬 컴퓨터의 현재 디렉터리에 `S3-role-trust-policy.json`이라는 이름의 파일로 저장합니다. 이 정책은 Amazon S3에 역할을 맡을 권한을 부여합니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

   ii. 다음 AWS CLI 명령을 실행해 역할을 생성합니다.

```
$ aws iam create-role \
--role-name replicationRole \
--assume-role-policy-document file://s3-role-trust-policy.json  \
--profile acctA
```

b. 역할에 권한 정책을 연결합니다.

   i. 다음 권한 정책을 복사하여 로컬 컴퓨터의 현재 디렉터리에 `s3-role-perm-pol-changeowner.json` 파일로 저장합니다. 이 정책은 다양한 Amazon S3 버킷 및 객체 작업에 대한 권한을 부여합니다. 다음 단계에서 IAM 역할을 생성하여 이 정책을 역할에 연결합니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl"
         ],
         "Resource":[
            "arn:aws:s3:::source/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket",
            "s3:GetReplicationConfiguration"
         ],
         "Resource":[
            "arn:aws:s3:::source"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ObjectOwnerOverrideToBucketOwner",
```

```
            "s3:ReplicateTags",
            "s3:GetObjectVersionTagging"
         ],
         "Resource":"arn:aws:s3:::destination/*"
      }
   ]
}
```

ii. 정책을 생성하고 역할에 연결하려면 다음 명령을 실행합니다.

```
$ aws iam put-role-policy \
--role-name replicationRole \
--policy-document file://s3-role-perm-pol-changeowner.json \
--policy-name replicationRolechangeownerPolicy \
--profile acctA
```

6. 원본 버킷에 복제 구성을 추가합니다.

   a. AWS CLI를 사용하려면 복제 구성을 JSON으로 지정해야 합니다. 다음 JSON을 로컬 컴퓨터의 현재 디렉터리에 `replication.json` 파일로 저장합니다. 구성에서 `AccessControlTranslation`을 추가하여 복제본 소유권 변경을 표시합니다.

```
{
   "Role":"IAM-role-ARN",
   "Rules":[
      {
         "Status":"Enabled",
         "Priority":"1",
         "DeleteMarkerReplication":{
            "Status":"Disabled"
         },
         "Filter":{
            "Prefix":"Tax"
         },
         "Status":"Enabled",
         "Destination":{
            "Bucket":"arn:aws:s3:::destination",
            "Account":"destination-bucket-owner-account-id",
            "AccessControlTranslation":{
               "Owner":"Destination"
            }
         }
      }
   ]
}
```

   b. *##* 버킷 소유자 계정 ID 및 *IAM-role-ARN*에 값을 입력하여 JSON을 편집합니다. 변경 사항을 저장합니다.

   c. 원본 버킷에 복제 구성을 추가하려면 다음 명령을 실행합니다. *##* 버킷 이름을 제공합니다.

```
$ aws s3api put-bucket-replication \
--replication-configuration file://replication-changeowner.json \
--bucket source \
--profile acctA
```

7. Amazon S3 콘솔에서 복제본 소유권을 확인합니다.

   a. AWS Management Console에 로그인하고 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.

   b. *##* 버킷에서 `Tax` 폴더를 생성합니다.

c. *##* 버킷에서 이 폴더에 객체를 추가합니다. *##* 버킷에 객체 복제본이 저장되었는지, 복제본 소유권이 *##* 버킷을 소유한 AWS 계정으로 변경되었는지 확인합니다.

버킷을 서로 다른 AWS 계정에서 소유한 경우 복제본 소유자 변경(AWS SDK)

복제 구성을 추가하는 코드 예제는 동일한 계정(AWS SDK)에서 버킷을 소유할 때 복제 구성 (p. 637) 단원을 참조하십시오. 복제 구성을 적절히 수정해야 합니다. 개념적 정보는 복제본 소유자 변경 (p. 626) 단원을 참조하십시오.

예제 4: 암호화된 객체 복제

기본적으로 Amazon S3은 AWS Key Management Service(AWS KMS) 고객 마스터 키(CMK)를 사용한 서버 측 암호화로 유휴 시 저장된 객체를 복제하지 않습니다. 암호화된 객체를 복제하려면 버킷 복제 구성을 수정하여 Amazon S3에 이러한 객체를 복제하도록 지시해야 합니다. 이 예제에서는 Amazon S3 콘솔 및 AWS Command Line Interface(AWS CLI)를 사용해 암호화된 객체 복제를 활성화하도록 버킷 복제 구성을 변경하는 방법을 설명합니다. 자세한 내용은 AWS KMS에 저장된 암호화 키를 사용한 서버 측 암호화(SSE)로 생성된 객체 복제 (p. 629) 단원을 참조하십시오.

암호화된 객체 복제(콘솔)

단계별 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷에 복제 규칙을 추가하는 방법 단원을 참조하십시오. 이 주제에서는 버킷을 동일한 AWS 계정이 소유한 경우와 서로 다른 AWS 계정이 소유한 경우에 복제 구성을 설정하는 지침을 제공합니다.

암호화된 객체 복제(AWS CLI)

AWS CLI를 사용하여 암호화된 객체를 복제하려면 버킷을 생성하고, 버킷에서 버전 관리를 활성화하고, Amazon S3에 객체를 복제할 권한을 부여하는 IAM 역할을 생성하고, 복제 구성을 원본 버킷에 추가합니다. 복제 구성이 KMS 키를 사용하여 암호화된 객체를 복제하는 데 관련된 정보를 제공합니다. IAM 역할 권한은 암호화된 객체를 복제하는 데 필요한 권한을 포함합니다. 또한 다음 설정을 테스트합니다.

서버 측 암호화된 객체를 복제하려면(AWS CLI)

1. 이 예제에서는 동일한 AWS 계정에서 *##* 버킷과 *##* 버킷을 모두 생성합니다. AWS CLI 자격 증명 프로필을 설정합니다. 이 예제에서는 프로필 이름 `acctA`를 사용합니다. 자격 증명 프로필 설정에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 명명된 프로필 단원을 참조하십시오.
2. *##* 버킷을 생성하고 버킷에서 버전 관리를 활성화합니다. 이 예제에서는 *##* 버킷을 미국 동부(버지니아 북부)(us-east-1) 리전에서 생성합니다.

```
aws s3api create-bucket \
--bucket source \
--region us-east-1 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket source \
--versioning-configuration Status=Enabled \
--profile acctA
```

3. *##* 버킷을 생성하고 버킷에서 버전 관리를 활성화합니다. 이 예제에서는 *##* 버킷을 미국 서부(오레곤)(us-west-2) 리전에서 생성합니다.

   Note

   ## 버킷과 *##* 버킷이 모두 동일한 AWS 계정에 있을 때 복제 구성을 설정하려면 동일한 프로필을 사용합니다. 이 예제에서는 [`acctA`]을 사용합니다. 두 버킷을 서로 다른 AWS 계정에서 소유한 경우의 복제 구성을 테스트하려면 각각 다른 프로필을 지정합니다.

```
aws s3api create-bucket \
--bucket destination \
--region us-west-2 \
--create-bucket-configuration LocationConstraint=us-west-2 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket destination \
--versioning-configuration Status=Enabled \
--profile acctA
```

4. IAM 역할을 생성합니다. 나중에 ## 버킷에 추가하는 복제 구성에서 이 역할을 지정합니다. Amazon S3가 사용자 대신 객체를 복제하기 위해 이 역할을 맡습니다. IAM 역할을 두 단계로 생성합니다.

   - 역할 생성
   - 역할에 권한 정책 연결

   a. IAM 역할 생성.

      i. 다음 신뢰 정책을 복사하여 로컬 컴퓨터의 현재 디렉터리에 `s3-role-trust-policy-kmsobj.json` 파일로 저장합니다. 이 정책은 Amazon S3가 사용자 대신 작업을 수행할 수 있도록 Amazon S3 서비스 보안 주체에 역할을 맡을 권한을 부여합니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

      ii. 역할을 생성합니다.

```
$ aws iam create-role \
--role-name replicationRolekmsobj \
--assume-role-policy-document file://s3-role-trust-policy-kmsobj.json  \
--profile acctA
```

   b. 역할에 권한 정책을 연결합니다. 이 정책은 다양한 Amazon S3 버킷 및 객체 작업에 대한 권한을 부여합니다.

      i. 다음 권한 정책을 복사하여 로컬 컴퓨터의 현재 디렉터리에 `s3-role-permissions-policykmsobj.json` 파일로 저장합니다. IAM 역할을 생성하여 나중에 정책을 여기에 연결합니다.

         Important

         권한 정책에서 원본 및 `destination` 버킷 암호화에 사용할 AWS KMS 키 ID를 지정합니다. `source` 및 `destination` 버킷에 별도의 AWS KMS CMK 2개를 생성해야 합니다. AWS KMS CMK는 해당 CMK가 생성된 AWS 리전 외부에서 공유되지 않습니다.

```
{
```

```
  "Version":"2012-10-17",
  "Statement":[
     {
        "Action":[
           "s3:ListBucket",
           "s3:GetReplicationConfiguration",
           "s3:GetObjectVersionForReplication",
           "s3:GetObjectVersionAcl"
        ],
        "Effect":"Allow",
        "Resource":[
           "arn:aws:s3:::source",
           "arn:aws:s3:::source/*"
        ]
     },
     {
        "Action":[
           "s3:ReplicateObject",
           "s3:ReplicateDelete",
           "s3:ReplicateTags",
           "s3:GetObjectVersionTagging"
        ],
        "Effect":"Allow",
        "Condition":{
           "StringLikeIfExists":{
              "s3:x-amz-server-side-encryption":[
                 "aws:kms",
                 "AES256"
              ],
              "s3:x-amz-server-side-encryption-aws-kms-key-id":[
                 "AWS KMS key IDs(in ARN format) to use for encrypting object
replicas"
              ]
           }
        },
        "Resource":"arn:aws:s3:::destination/*"
     },
     {
        "Action":[
           "kms:Decrypt"
        ],
        "Effect":"Allow",
        "Condition":{
           "StringLike":{
              "kms:ViaService":"s3.us-east-1.amazonaws.com",
              "kms:EncryptionContext:aws:s3:arn":[
                 "arn:aws:s3:::source/*"
              ]
           }
        },
        "Resource":[
           "AWS KMS key IDs(in ARN format) used to encrypt source objects."
        ]
     },
     {
        "Action":[
           "kms:Encrypt"
        ],
        "Effect":"Allow",
        "Condition":{
           "StringLike":{
              "kms:ViaService":"s3.us-west-2.amazonaws.com",
              "kms:EncryptionContext:aws:s3:arn":[
                 "arn:aws:s3:::destination/*"
              ]
           }
```

```
            },
            "Resource":[
               "AWS KMS key IDs(in ARN format) to use for encrypting object
 replicas"
            ]
         }
      ]
}
```

ii. 정책을 생성하여 역할에 연결합니다.

```
$ aws iam put-role-policy \
--role-name replicationRolekmsobj \
--policy-document file://s3-role-permissions-policykmsobj.json \
--policy-name replicationRolechangeownerPolicy \
--profile acctA
```

5. 다음 복제 구성을 *##* 버킷에 추가합니다. 이 구성은 Amazon S3에 `Tax/` 접두사가 있는 객체를 *##* 버킷으로 복제하도록 지시합니다.

   Important

   복제 구성에서 Amazon S3가 맡을 수 있는 IAM 역할을 지정할 수 있습니다. `iam:PassRole` 권한이 있을 경우 이 권한만 사용할 수 있습니다. CLI 명령에서 지정한 프로필에 권한이 있어야 합니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 AWS 서비스에 역할을 전달할 수 있는 권한 부여 단원을 참조하십시오.

```
<ReplicationConfiguration>
 <Role>IAM-Role-ARN</Role>
 <Rule>
   <Status>Enabled</Status>
   <Priority>1</Priority>
   <DeleteMarkerReplication>
      <Status>Disabled</Status>
   </DeleteMarkerReplication>
   <Filter>
      <Prefix>Tax</Prefix>
   </Filter>
   <Status>Enabled</Status>
   <SourceSelectionCriteria>
     <SseKmsEncryptedObjects>
       <Status>Enabled</Status>
     </SseKmsEncryptedObjects>
   </SourceSelectionCriteria>
   <Destination>
     <Bucket>arn:aws:s3:::dest-bucket-name</Bucket>
     <EncryptionConfiguration>
       <ReplicaKmsKeyID>AWS KMS key IDs to use for encrypting object replicas</
ReplicaKmsKeyID>
     </EncryptionConfiguration>
   </Destination>
 </Rule>
</ReplicationConfiguration>
```

버킷에 복제 구성을 추가하려면 다음을 수행합니다.

a. AWS CLI를 사용하려면 복제 구성을 JSON으로 지정해야 합니다. 다음 JSON을 로컬 컴퓨터의 현재 디렉터리에 파일(`replication.json`)로 저장합니다.

```
{
   "Role":"IAM-Role-ARN",
   "Rules":[
```

```
      {
         "Status":"Enabled",
         "Priority":1,
         "DeleteMarkerReplication":{
            "Status":"Disabled"
         },
         "Filter":{
            "Prefix":"Tax"
         },
         "Destination":{
            "Bucket":"arn:aws:s3:::destination",
            "EncryptionConfiguration":{
               "ReplicaKmsKeyID":"AWS KMS key IDs(in ARN format)  to use for
 encrypting object replicas"
            }
         },
         "SourceSelectionCriteria":{
            "SseKmsEncryptedObjects":{
               "Status":"Enabled"
            }
         }
      }
   ]
}
```

b. JSON을 편집하여 *##* 버킷, *KMS ID ARN* 및 *IAM-role-ARN*의 값을 제공합니다. 변경 사항을 저장합니다.

c. *##* 버킷에 복제 구성을 추가합니다. 반드시 *##* 버킷 이름을 제공해야 합니다.

```
$ aws s3api put-bucket-replication \
--replication-configuration file://replication.json \
--bucket source \
--profile acctA
```

6. 설정을 테스트하여 암호화된 객체가 복제되는지 확인합니다. Amazon S3 콘솔에서

   a. AWS Management Console에 로그인하고 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.

   b. *##* 버킷에서 `Tax` 폴더를 생성합니다.

   c. 폴더에 샘플 객체를 추가합니다. 반드시 암호화 옵션을 선택하고 객체를 암호화할 AWS KMS CMK를 지정해야 합니다.

   d. *##* 버킷에 객체 복제본이 있으며 해당 복제본이 구성에 지정된 AWS KMS CMK를 사용하여 암호화되었는지 확인합니다.

암호화된 객체 복제(AWS SDK)

복제 구성을 추가하는 코드 예제는 동일한 계정(AWS SDK)에서 버킷을 소유할 때 복제 구성 (p. 637) 단원을 참조하십시오. 복제 구성을 적절히 수정해야 합니다.

개념적 정보는 AWS KMS에 저장된 암호화 키를 사용한 서버 측 암호화(SSE)로 생성된 객체 복제 (p. 629) 단원을 참조하십시오.

예 5: S3 Replication Time Control (S3 RTC) 구성

S3 Replication Time Control (S3 RTC)는 데이터 복제에 대한 규정 준수 또는 비즈니스 요구 사항을 충족하고 Amazon S3 복제 시간을 파악할 수 있도록 지원합니다. S3 RTC는 Amazon S3에 업로드하는 대부분의 객체를 몇 초 만에 복제하고 이러한 객체의 99.99%를 15분 내에 복제합니다.

S3 RTC를 사용하여 복제 보류 중인 객체의 총 수와 크기 및 대상 리전까지의 최대 복제 시간을 모니터링할 수 있습니다. 복제 지표는 AWS Management 콘솔 및 Amazon CloudWatch 사용 설명서를 통해 사용할 수 있습니다. 자세한 내용은 the section called “Amazon S3 CloudWatch 복제 지표” (p. 673) 단원을 참조하십시오.

Amazon S3 콘솔에서 S3 RTC 활성화

단계별 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷에 복제 규칙을 추가하는 방법 단원을 참조하십시오. 이 주제에서는 버킷을 동일한 AWS 계정이 소유한 경우와 서로 다른 AWS 계정이 소유한 경우에 복제 구성에서 S3 RTC를 활성화하는 지침을 제공합니다.

Amazon S3 Replication Time Control을 사용하여 객체 복제(AWS CLI)

AWS CLI를 사용하여 S3 RTC가 활성화된 객체를 복제하려면 버킷을 생성하고, 버킷에서 버전 관리를 활성화하고, Amazon S3에 객체를 복제할 권한을 부여하는 IAM 역할을 생성하고, 복제 구성을 원본 버킷에 추가합니다. 복제 구성에는 S3 Replication Time Control (S3 RTC)가 활성화되어 있어야 합니다.

S3 RTC가 활성화된 상태로 복제하려면(AWS CLI)

- 이 예제에서는 `ReplicationTime` 및 `Metric`을 설정하고 복제 구성을 원본 버킷에 추가합니다.

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "DeleteMarkerReplication": {
                "Status": "Disabled"
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::destination",
                "Metrics": {
                    "Status": "Enabled",
                    "EventThreshold": {
                        "Minutes": 15
                    }
                },
                "ReplicationTime": {
                    "Status": "Enabled",
                    "Time": {
                        "Minutes": 15
                    }
                }
            },
            "Priority": 1
        }
    ],
    "Role": "IAM-Role-ARN"
}
```

> **Important**
>
> `Metrics:EventThreshold:Minutes` 및 `ReplicationTime:Time:Minutes`는 유효한 값인 15만 가질 수 있습니다.

Amazon S3 Replication Time Control을 사용하여 객체 복제(AWS SDK)

다음은 S3 Replication Time Control (S3 RTC)를 통해 복제 구성을 추가하는 Java 예제입니다.

```
import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.DeleteMarkerReplication;
import software.amazon.awssdk.services.s3.model.Destination;
import software.amazon.awssdk.services.s3.model.Metrics;
import software.amazon.awssdk.services.s3.model.MetricsStatus;
import software.amazon.awssdk.services.s3.model.PutBucketReplicationRequest;
import software.amazon.awssdk.services.s3.model.ReplicationConfiguration;
import software.amazon.awssdk.services.s3.model.ReplicationRule;
import software.amazon.awssdk.services.s3.model.ReplicationRuleFilter;
import software.amazon.awssdk.services.s3.model.ReplicationTime;
import software.amazon.awssdk.services.s3.model.ReplicationTimeStatus;
import software.amazon.awssdk.services.s3.model.ReplicationTimeValue;

public class Main {

  public static void main(String[] args) {
    S3Client s3 = S3Client.builder()
      .region(Region.US_EAST_1)
      .credentialsProvider(() -> AwsBasicCredentials.create(
          "AWS_ACCESS_KEY_ID",
          "AWS_SECRET_ACCESS_KEY")
      )
      .build();

    ReplicationConfiguration replicationConfig = ReplicationConfiguration
      .builder()
      .rules(
          ReplicationRule
            .builder()
            .status("Enabled")
            .priority(1)
            .deleteMarkerReplication(
                DeleteMarkerReplication
                    .builder()
                    .status("Disabled")
                    .build()
            )
            .destination(
                Destination
                    .builder()
                    .bucket("destination_bucket_arn")
                    .replicationTime(
                        ReplicationTime.builder().time(
                            ReplicationTimeValue.builder().minutes(15).build()
                        ).status(
                            ReplicationTimeStatus.ENABLED
                        ).build()
                    )
                    .metrics(
                        Metrics.builder().eventThreshold(
                            ReplicationTimeValue.builder().minutes(15).build()
                        ).status(
                            MetricsStatus.ENABLED
                        ).build()
                    )
                    .build()
            )
            .filter(
                ReplicationRuleFilter
                    .builder()
                    .prefix("testtest")
                    .build()
            )
        .build())
```

```
        .role("role_arn")
        .build();

    // Put replication configuration
    PutBucketReplicationRequest putBucketReplicationRequest = PutBucketReplicationRequest
      .builder()
      .bucket("source_bucket")
      .replicationConfiguration(replicationConfig)
      .build();

    s3.putBucketReplication(putBucketReplicationRequest);
  }
}
```

자세한 내용은 S3 Replication Time Control (S3 RTC)를 사용하여 객체 복제 (p. 624) 단원을 참조하십시오.

복제 상태 정보

버킷 내 객체의 복제) 상태를 가져오려면 Amazon S3 인벤토리 도구를 사용합니다. Amazon S3는 CSV 파일을 사용자가 인벤토리 구성에서 지정하는 대상 버킷으로 전송합니다. 또한 Amazon Athena를 사용하여 인벤토리 보고서의 복제 상태를 쿼리할 수도 있습니다. Amazon S3 인벤토리에 대한 자세한 내용은 Amazon S3 인벤토리 (p. 459) 단원을 참조하십시오.

복제에는 복제를 구성하는 원본 버킷과 Amazon S3가 객체를 복제하는 대상 버킷이 있습니다. 이러한 버킷에서 객체(`GET` 객체 사용)나 객체 메타데이터(`HEAD` 객체를 사용)를 요청하면 Amazon S3가 그에 응답해 `x-amz-replication-status` 헤더를 반환합니다.

- 원본 버킷에서 객체를 요청하면 요청한 객체가 복제에 적합한 경우 Amazon S3가 `x-amz-replication-status` 헤더를 반환합니다.

  예를 들어, 복제 구성에 객체 접두사 `TaxDocs`를 지정해 Amazon S3에게 키 이름 접두사가 `TaxDocs`인 객체만 복제하도록 지시한다고 가정해 봅시다. 이 키 이름 접두사(예: `TaxDocs/document1.pdf`)를 갖는 객체를 업로드하면 모두 복제됩니다. 이 키 이름 접두사를 사용한 객체 요청에 대해 Amazon S3는 객체 복제 상태를 표시하는 값 `PENDING`, `COMPLETED` 또는 `FAILED`가 포함된 `x-amz-replication-status` 헤더를 반환합니다.

  > **Note**
  >
  > 객체를 업로드한 후 객체 복제가 실패할 경우 복제를 재시도할 수 없습니다. 객체를 다시 업로드해야 합니다.

- 대상 버킷으로부터 객체를 요청하면 해당 객체가 Amazon S3에서 생성한 복제본인 경우 Amazon S3는 값이 `REPLICA`인 `x-amz-replication-status` 헤더를 반환합니다.

콘솔, AWS Command Line Interface(AWS CLI) 또는 AWS SDK를 사용해 객체 복제 상태를 확인할 수 있습니다.

- 콘솔: 객체를 선택하고 속성을 선택하여 복제 상태를 비롯한 객체 속성을 확인합니다.
- AWS CLI: `head-object`AWS CLI 명령을 사용해 객체 메타데이터를 검색합니다.

```
aws s3api head-object --bucket source-bucket --key object-key --version-id object-version-id
```

이 명령은 다음의 응답 예와 같이 `ReplicationStatus`가 포함된 객체 메타데이터를 반환합니다.

```
{
```

```
    "AcceptRanges":"bytes",
    "ContentType":"image/jpeg",
    "LastModified":"Mon, 23 Mar 2015 21:02:29 GMT",
    "ContentLength":3191,
    "ReplicationStatus":"COMPLETED",
    "VersionId":"jfnW.HIMOfYiD_9rGbSkmroXsFj3fqZ.",
    "ETag":"\"6805f2cfc46c0f04559748bb039d69ae\"",
    "Metadata":{

    }
}
```

- AWS SDK: 다음 코드 조각은 각각 AWS SDK for Java 및 .NET용 AWS SDK를 사용해 복제 상태를 가져옵니다.
  - AWS SDK for Java

```
GetObjectMetadataRequest metadataRequest = new GetObjectMetadataRequest(bucketName,
 key);
ObjectMetadata metadata = s3Client.getObjectMetadata(metadataRequest);

System.out.println("Replication Status : " +
 metadata.getRawMetadataValue(Headers.OBJECT_REPLICATION_STATUS));
```

  - .NET용 AWS SDK

```
GetObjectMetadataRequest getmetadataRequest = new GetObjectMetadataRequest
    {
         BucketName = sourceBucket,
         Key        = objectKey
    };

GetObjectMetadataResponse getmetadataResponse =
 client.GetObjectMetadata(getmetadataRequest);
Console.WriteLine("Object replication status: {0}",
 getmetadataResponse.ReplicationStatus);
```

Note

복제가 활성화된 원본 버킷에서 객체를 삭제하려면, 먼저 해당 객체의 복제 상태를 통해 복제가 완료되었는지 확인해야 합니다.
원본 버킷에서 수명 주기 구성이 활성화된 경우, 객체 상태가 `COMPLETED` 또는 `FAILED`로 표시될 때까지 Amazon S3는 수명 주기 작업을 보류합니다.

관련 주제

복제 (p. 607)

애플리케이션 문제 해결

복제를 구성한 후 대상 버킷에 객체 복제본이 표시되지 않는다면 다음 문제 해결 도움말을 사용하여 문제를 해결합니다.

- 대다수의 객체는 15분 이내에 복제되지만 때로는 두 시간 정도 걸릴 수도 있습니다. 드물지만 복제 시간이 더 오래 걸릴 수도 있습니다. Amazon S3가 객체를 복제하는 데 걸리는 시간은 원본 및 대상 리전 쌍, 객체 크기를 비롯한 다양한 요소에 따라 다릅니다. 큰 객체의 경우 복제에 몇 시간이 걸릴 수도 있습니다. 복제하는 객체가 크다면 얼마간 기다렸다 대상 버킷에 해당 객체가 나타나는지 확인하십시오. 또한 원본 객체 복제 상태를 확인할 수도 있습니다. 객체 복제 상태가 `pending`이라면 Amazon S3가 복제를 완료하지 않

은 것임을 알 수 있습니다. 객체 복제 상태가 `failed`인 경우 원본 버킷에서 설정된 복제 구성을 확인합니다.

- 원본 버킷의 복제 구성에서 다음을 확인합니다.
  - 대상 버킷의 Amazon 리소스 이름(ARN)이 정확합니다.
  - 키 이름 접두사가 정확합니다. 예를 들어, 접두사 `Tax`로 객체 복제를 구성한 경우라면 `Tax/document1` 또는 `Tax/document2`와 같이 키 이름이 포함된 객체만 복제됩니다. 키 이름이 `document3`인 객체는 복제되지 않습니다.
  - 상태가 `enabled`입니다.
- 다른 AWS 계정이 대상 버킷을 소유한 경우, 해당 대상 버킷에 소스 버킷 소유자의 객체 복제를 허용하는 버킷 정책이 있는지 확인합니다. 예제는 예제 2: 원본 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유한 경우 복제를 구성 (p. 642) 단원을 참조하십시오.
- 객체 복제본이 대상 버킷에 나타나지 않는 경우, 다음 이유로 복제가 금지된 것일 수 있습니다.
  - Amazon S3는 원본 버킷에서 다른 복제 구성이 생성한 복제본을 복제하지 않습니다. 예를 들어, 버킷 A에서 버킷 B로, 버킷 B에서 버킷 C로 복제 구성을 설정하면 Amazon S3가 버킷 B에 있는 객체 복제본은 버킷 C로 복제하지 않습니다.
  - 원본 버킷 소유자는 다른 AWS 계정에 객체 업로드 권한을 부여할 수 있습니다. 기본적으로 원본 버킷 소유자는 다른 계정에서 생성한 객체에 대해 권한이 없습니다. 복제 구성은 원본 버킷 소유자가 액세스 권한을 가진 객체만 복제합니다. 원본 버킷 소유자는 다른 AWS 계정에 해당 객체에 대한 명시적 액세스 권한을 조건부로 요구하는 객체 생성 권한을 부여할 수 있습니다. 정책에 대한 예는 버킷 소유자가 완벽하게 제어할 수 있도록 보증하면서 객체에 업로드하는 교차 계정 권한 부여 (p. 413)을 참조하십시오.
- 복제 구성에서 객체 중 특정 태그를 갖는 하위 집합을 복제하는 규칙을 추가하는 것으로 가정해 봅시다. 이 경우 , Amazon S3가 객체를 복제할 객체를 생성할 때 특정 태그 키 및 값을 할당해야 합니다. 먼저 객체를 생성한 후 기존 객체에 태그를 추가할 경우 Amazon S3는 해당 객체를 복제하지 않습니다.

관련 주제

복제 (p. 607)

복제 추가 고려 사항

또한 Amazon S3는 다음과 같은 버킷 구성을 지원합니다.

- 버전 관리 — 자세한 내용은 버전 관리 사용 (p. 468) 단원을 참조하십시오.
- 웹사이트 호스팅 — 자세한 내용은 Amazon S3 정적 웹 사이트 호스팅 (p. 548) 단원을 참조하십시오.
- 버킷 정책 또는 ACL(액세스 제어 목록)을 통한 버킷 액세스 — 자세한 내용은 버킷 정책 및 사용자 정책 사용 (p. 338) 단원과 ACL을 사용한 액세스 관리 (p. 437) 단원을 참조하십시오.
- 로그 스토리지 — 자세한 내용은 Amazon S3 서버 액세스 로깅 (p. 705) 단원을 참조하십시오.
- 버킷 내 객체에 대한 수명 주기 관리 — 자세한 내용은 객체 수명 주기 관리 (p. 123) 단원을 참조하십시오.

이 주제에서는 버킷 복제 구성이 다음 버킷 구성의 기능에 어떤 영향을 주는지 설명합니다.

주제

- 수명 주기 구성 및 객체 복제본 (p. 657)
- 버전 관리 구성 및 복제 구성 (p. 657)
- 로그 구성 및 복제 구성 (p. 657)
- CRR 및 대상 리전 (p. 657)
- 복제 일시 중지 (p. 657)
- 관련 주제 (p. 657)

수명 주기 구성 및 객체 복제본

Amazon S3가 객체를 복제하는 데 걸리는 시간은 객체 크기에 따라 다릅니다. 큰 객체의 경우 여러 시간이 걸릴 수 있습니다. 대상 버킷에서 복제본을 사용할 수 있기까지는 시간이 다소 걸리지만, 복제본 생성 시간은 원본 버킷에서 객체를 생성하는 데 걸린 시간과 동일합니다. 대상 버킷에서 수명 주기 정책이 활성화된 경우, 수명 주기 규칙은 대상 버킷에서 복제본이 가용해진 시간이 아니라 객체의 원본 생성 시간을 적용합니다.

복제 구성을 사용하려면 버킷에서 버전 관리를 사용해야 합니다. 버킷에서 버전 관리를 사용하는 경우 다음에 유의하십시오.

- 객체 만료 수명 주기 정책이 있는 경우 버전 관리를 활성화한 후 `NonCurrentVersionExpiration` 정책을 추가하여 버전 관리를 활성화하기 전과 동일한 영구 삭제 행동을 유지합니다.
- 이전 수명 주기 정책이 있는 경우 버전 관리를 활성화한 후 `NonCurrentVersionTransition` 정책을 추가할 것을 고려합니다.

버전 관리 구성 및 복제 구성

버킷에 대한 복제를 구성하려면 원본 및 대상 버킷 모두 버전 관리가 활성화되어 있어야 합니다. 원본 및 대상 버킷에 버전 관리를 활성화하고 원본 버킷의 복제를 구성한 후, 다음과 같은 문제가 발생할 수 있습니다.

- 원본 버킷의 버전 관리를 비활성화하려는 경우, Amazon S3는 오류를 반환합니다. 원본 버킷의 버전 관리를 비활성화하려면 우선 복제 구성을 삭제해야 합니다.
- 대상 버킷의 버전 관리를 비활성화하는 경우, 복제가 실패합니다. 원본 객체의 복제 상태가 `Failed`입니다.

로그 구성 및 복제 구성

Amazon S3가 복제가 활성화된 버킷에 대한 로그를 제공하는 경우 해당 로그 객체가 복제됩니다.

원본 또는 대상 버킷에서 서버 액세스 로그(Amazon S3 서버 액세스 로깅 (p. 705)) 또는 AWS CloudTrail 로그(AWS CloudTrail을 사용하여 Amazon S3 API 호출 로깅 (p. 679))가 활성화된 경우 Amazon S3의 로그에 복제 관련 요청이 포함됩니다. 예를 들어, Amazon S3는 복제되는 각 객체를 기록합니다.

CRR 및 대상 리전

교차 리전 복제(CRR) 구성에서는 원본 버킷과 대상 버킷이 서로 다른 AWS 리전에 위치해야 합니다. 비즈니스 요구 사항 또는 비용 고려 사항에 따라 대상 버킷의 리전을 선택할 수 있습니다. 예를 들어, 리전 간 데이터 전송 요금은 선택한 리전에 따라 다릅니다. 원본 버킷의 리전으로 미국 동부(버지니아 북부)(us-east-1)을 선택했다고 가정하겠습니다. 미국 서부(오레곤)(us-west-2)를 대상 버킷 리전으로 선택한 경우 미국 동부(오하이오)(us-east-2) 리전을 선택한 경우보다 더 많은 요금을 지불해야 합니다. 요금 정보는 Amazon S3 요금 단원의 "데이터 전송 요금"을 참조하십시오. 동일 리전 복제(SRR)와 관련된 데이터 전송 요금은 없습니다.

복제 일시 중지

복제를 일시 중지하려면 복제 구성에서 관련 규칙을 비활성화합니다.

복제가 활성화되었지만 Amazon S3에 필요한 권한을 부여하는 IAM 역할을 제거하면 복제가 실패합니다. Amazon S3가 해당 객체의 복제 상태를 `Failed`로 보고합니다.

관련 주제

복제 (p. 607)

라우팅 요청

주제

- 요청 리디렉션 및 REST API (p. 658)
- DNS 고려 사항 (p. 661)

<CreateBucketConfiguration> API를 사용하여 만든 버킷에 대한 요청을 하는 프로그램은 리디렉션을 지원해야 합니다. 또한 DNS TTL을 지키지 않는 일부 클라이언트에서 문제가 발생할 수 있습니다.

이 단원에서는 Amazon S3와 함께 사용할 서비스 또는 애플리케이션을 설계할 때 고려할 라우팅 및 DNS 문제에 대해 설명합니다.

요청 리디렉션 및 REST API

Amazon S3에서는 DNS(도메인 이름 시스템)를 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.

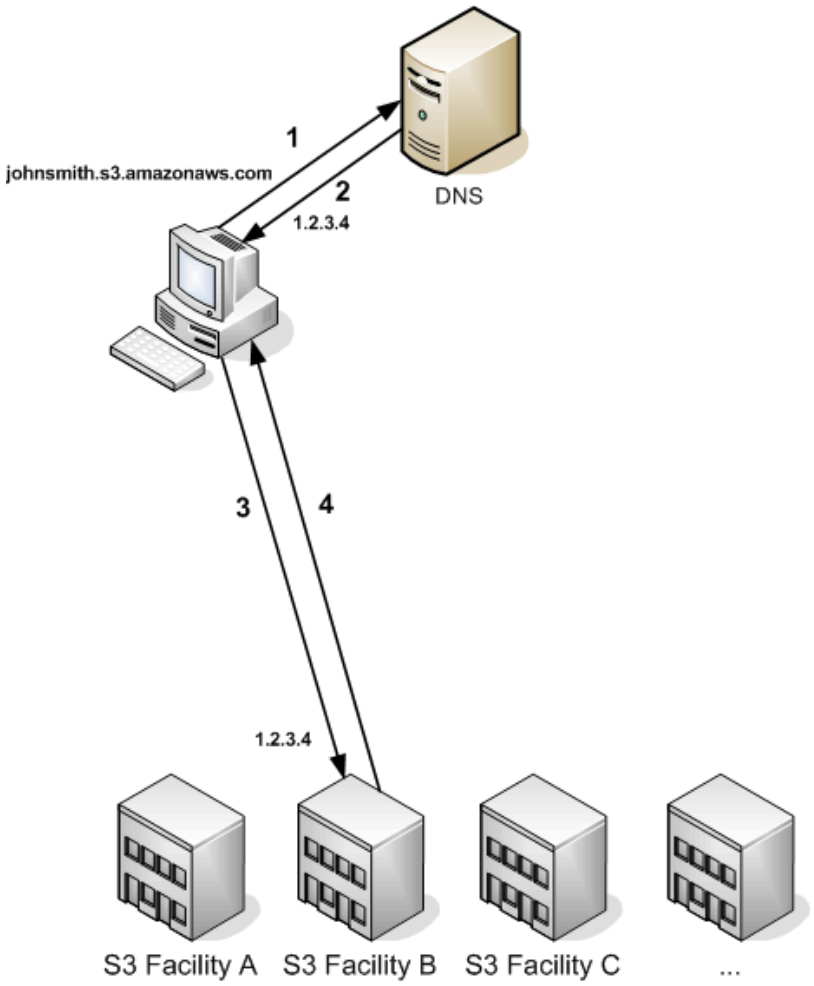
> **Important**
>
> 이 기능을 사용하려면 Amazon S3 리디렉션 응답을 처리할 수 있는 애플리케이션이 있어야 합니다. `<CreateBucketConfiguration>` 없이 생성된 버킷으로만 작동하는 애플리케이션에 대해서는 유일하게 예외가 적용됩니다. 위치 제약에 대한 자세한 내용은 버킷 액세스 (p. 53) 단원을 참조하십시오.
> 2019년 3월 20일 이후에 시작된 모든 리전에 대해 요청이 잘못된 Amazon S3 위치에 도달하면 Amazon S3이 HTTP 400 잘못된 요청 오류를 반환합니다.
> AWS 리전 활성화 또는 비활성화에 대한 자세한 내용은 AWS General Reference의 AWS 리전 및 엔드포인트를 참조하십시오.

주제

- DNS 라우팅 (p. 658)
- 임시 요청 리디렉션 (p. 659)
- 영구 요청 리디렉션 (p. 661)
- 요청 리디렉션 예제 (p. 661)

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 만듭니다.
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.

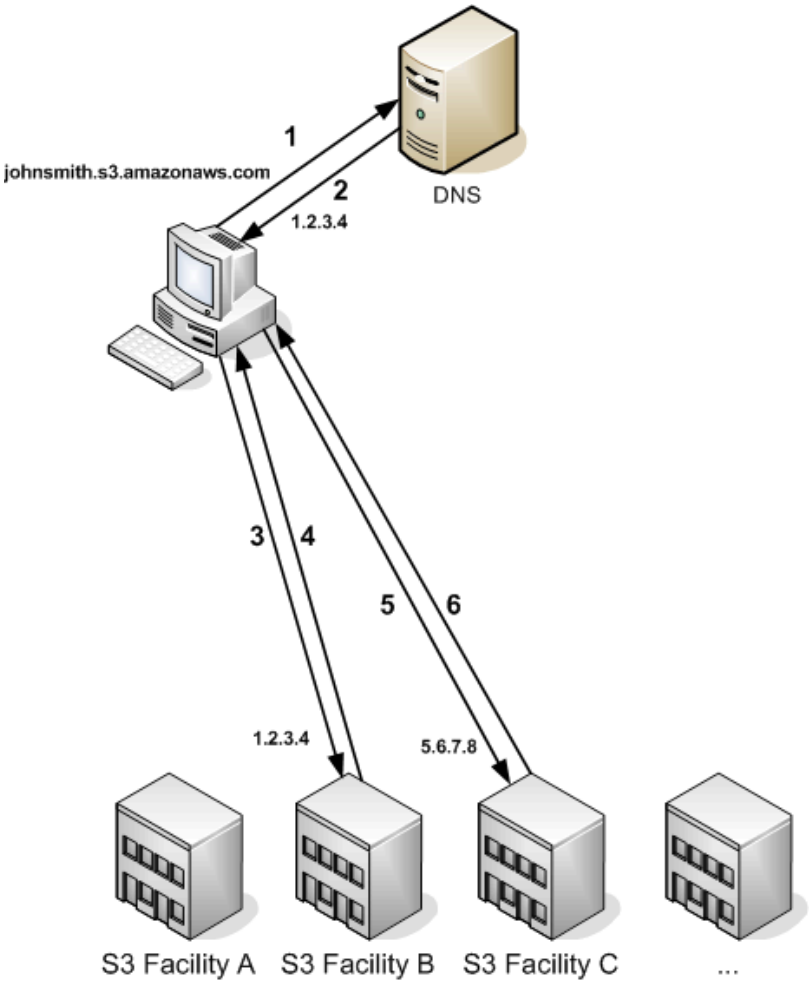
예를 들어, 새 버킷을 만들고 이 버킷에 즉시 요청을 보내는 경우, 버킷의 위치 제한에 따라 임시 리디렉션을 수신할 수 있습니다. 미국 동부(버지니아 북부) AWS 리전에서 버킷을 생성한 경우 이곳 역시 기본 Amazon S3 엔드포인트이므로 리디렉션 현상이 나타나지 않습니다.

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 만듭니다.
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 버킷 액세스 (p. 53) 단원을 참조하십시오.

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://awsexamplebucket.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>awsexamplebucket.s3-gztb4pa9sq.amazonaws.com</Endpoint>
</Error>
```

SOAP API 임시 리디렉션 응답

> **Note**
>
> HTTP를 통한 SOAP 지원은 중단되었지만 HTTPS를 통해 계속해서 사용할 수 있습니다. 새로운 Amazon S3 기능은 SOAP에 대해 지원되지 않습니다. 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-gztb4pa9sq.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 설명서의 "InetAddress Caching" 단원을 참조하십시오.
- PHP의 경우, 가장 인기 있는 배포 구성에서 실행되는 PHP VM이 VM이 재시작될 때까지 DNS 조회를 캐싱합니다. getHostByName PHP 문서를 참조하십시오.

모범 사례 설계 패턴: Amazon S3 성능 최적화

애플리케이션으로 Amazon S3의 스토리지를 업로드하고 검색할 때 요청 성능에서 초당 수천 회의 트랜잭션을 쉽게 달성할 수 있습니다. Amazon S3은 높은 요청 빈도로 자동으로 조정됩니다. 예를 들어, 애플리케이션으로 버킷에서 접두사별로 초당 최소 3,500개의 PUT/COPY/POST/DELETE 또는 5,500개의 GET/HEAD 요청을 달성할 수 있습니다. 버킷의 접두사 수에는 제한이 없습니다. 읽기를 병렬화하여 읽기 또는 쓰기 성능을 향상시킬 수 있습니다. 예를 들어, Amazon S3 버킷에서 접두사 10개를 만들어 읽기를 병렬화하는 경우 읽기 성능을 초당 읽기 요청 55,000개로 조정할 수 있습니다.

Amazon S3의 일부 데이터 레이크 애플리케이션은 페타바이트 이상의 데이터를 실행하는 수백만 또는 수십억 개의 객체를 검색합니다. 이러한 데이터 레이크 애플리케이션으로 단일 인스턴스에서 최대 100Gb/s가 될 수 있는 Amazon EC2 인스턴스의 네트워크 인터페이스 사용을 극대화하는 단일 인스턴스 전송 속도를 달성합니다. 그런 다음 이러한 애플리케이션은 여러 인스턴스에서 처리량을 집계하여 초당 여러 테라비트를 얻습니다.

또 다른 예는 소셜 미디어 메시징 애플리케이션처럼 지연 시간에 민감한 애플리케이션입니다. 이러한 애플리케이션은 약 100–200밀리초의 일관된 작은 객체 지연 시간(큰 객체의 경우 첫 번째 바이트 출력 지연 시간)을 달성할 수 있습니다.

다른 AWS 서비스도 다른 애플리케이션 아키텍처의 성능을 가속화하는 데 도움이 됩니다. 예를 들어, 단일 HTTP 연결 또는 한자리 수 밀리초 지연 시간에 비해 더 높은 전송 속도를 원한다면 Amazon S3로 캐싱하기 위해 Amazon CloudFront 또는 Amazon ElastiCache을 사용하십시오.

또한 클라이언트와 S3 버킷 간 장거리에 걸쳐 고속 데이터 전송을 원할 경우 Amazon S3 Transfer Acceleration (p. 67)를 사용하십시오. Transfer Acceleration은 CloudFront에서 전 세계에 분산된 엣지 로케이션을 사용하여 지리적으로 먼 거리 간 데이터 전송을 가속화합니다. Amazon S3의 워크로드에서 서버 측 암호화와 함께 AWS Key Management Service(SSE-KMS)가 사용되는 경우 사용 사례에 지원되는 요청 빈도에 대해 자세히 알아보려면 AWS Key Management Service 개발자 안내서의 AWS KMS 제한을 참조하십시오.

다음 주제에서는 Amazon S3를 사용하는 애플리케이션의 성능을 최적화하기 위한 모범 사례 지침과 설계 패턴에 대해 설명합니다. 이 지침은 Amazon S3의 성능 최적화에 대한 이전 지침을 대체합니다. 예를 들어, 이전의 Amazon S3 성능 지침에서는 잦은 데이터 검색 성능을 최적화하기 위해 해시된 문자를 사용하여 접두사 이름을 임의로 지정할 것을 권장했습니다. 그러나 더 이상 성능을 위해 접두사 이름을 무작위로 지정할 필요가 없으며 접두사에 대해 순차적 날짜 기반의 이름을 지정할 수 있습니다. Amazon S3의 성능 최적화에 대한 최신 정보는 Amazon S3의 성능 지침 (p. 663) 및 Amazon S3의 성능 디자인 패턴 (p. 665) 단원을 참조하십시오.

주제

- Amazon S3의 성능 지침 (p. 663)
- Amazon S3의 성능 디자인 패턴 (p. 665)

Amazon S3의 성능 지침

Amazon S3에서 객체를 업로드 및 검색하는 애플리케이션을 빌드할 때 모범 사례 지침에 따라 성능을 최적화합니다. 자세한 내용은 성능 디자인 패턴 (p. 665) 단원을 참조하십시오.

Amazon S3에서 애플리케이션의 성능을 최상으로 유지하려면 다음 지침을 따르는 것이 좋습니다.

주제

- 성능 측정 (p. 664)
- 스토리지 연결 수평 확장 (p. 664)
- 바이트 범위 가져오기 사용 (p. 664)
- 지연 시간에 민감한 애플리케이션 요청 재시도 (p. 664)
- 동일한 AWS 리전에서 Amazon S3(스토리지) 및 Amazon EC2(컴퓨팅) 결합 (p. 664)
- 거리에 의해 발생하는 지연 시간을 최소화하기 위해 Amazon S3 Transfer Acceleration 사용 (p. 665)
- 최신 버전의 AWS SDK 사용 (p. 665)

성능 측정

성능을 최적화할 때 네트워크 처리량, CPU 및 DRAM 요구 사항을 살펴보십시오. 이처럼 다양한 리소스에 대한 요구 조건의 조합에 따라 다른 Amazon EC2 인스턴스 유형을 평가할 가치가 있습니다. 인스턴스 유형에 대한 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 인스턴스 유형을 참조하십시오.

성능을 측정할 때 HTTP 분석 도구를 사용하여 DNS 조회 시간, 지연 시간 및 데이터 전송 속도를 살펴 보는 것도 도움이 됩니다.

스토리지 연결 수평 확장

여러 연결에 걸쳐 요청을 분산시키는 것은 성능을 수평으로 확장하는 일반적인 설계 패턴입니다. 고성능 애플리케이션을 빌드할 때는 Amazon S3를 기존의 스토리지 서버 같은 단일 네트워크 엔드포인트가 아닌 아주 큰 분산 시스템처럼 생각하십시오. Amazon S3로 여러 건의 동시 요청을 보내 최상의 성능을 달성할 수 있습니다. 이러한 요청을 별도의 여러 연결로 분산하여 Amazon S3에서 액세스 가능한 대역폭을 최대화하십시오. Amazon S3는 버킷에 대한 연결 수를 제한하지 않습니다.

바이트 범위 가져오기 사용

GET Object 요청에서 `Range` HTTP 헤더를 사용하면 객체에서 바이트 범위를 가져와 지정된 부분만 전송할 수 있습니다. Amazon S3에 대한 동시 연결을 사용하여 동일한 객체 내에서 서로 다른 바이트 범위를 가져올 수 있습니다. 그러면 전체 객체 요청 한 건에 비해 집계 처리량을 높일 수 있습니다. 더 작은 범위의 큰 객체를 가져와 요청이 중단되었을 때 애플리케이션의 재시도 횟수를 개선할 수도 있습니다. 자세한 내용은 객체 가져오기 (p. 164)을 참조하십시오.

바이트 범위 요청의 일반적인 크기는 8MB 또는 16MB입니다. 객체가 멀티파트 업로드를 사용해 PUT되는 경우 최상의 성능을 위해 동일한 파트 크기(또는 최소한 파트 경계에 맞춤)로 GET하는 것이 좋습니다. GET 요청은 개별 파트(예: `GET ?partNumber=N.`)를 직접 처리할 수 있습니다.

지연 시간에 민감한 애플리케이션 요청 재시도

공격적인 제한 시간과 재시도는 일관된 지연 시간을 유지하는 데 도움이 됩니다. Amazon S3의 큰 규모로 볼 때, 첫 번째 요청이 느린 경우 재시도된 요청은 다른 경로를 취하여 신속하게 성공할 가능성이 높습니다. AWS SDK에는 특정 애플리케이션의 허용 오차에 따라 튜닝할 수 있는 구성 가능한 제한 시간 및 재시도 값이 있습니다.

동일한 AWS 리전에서 Amazon S3(스토리지) 및 Amazon EC2(컴퓨팅) 결합

S3 버킷 이름은 전 세계적으로 고유하지만 각 버킷은 버킷을 만들 때 선택한 리전에 저장됩니다. 성능을 최적화하려면, 가급적 같은 AWS 리전의 Amazon EC2 인스턴스의 버킷에 액세스하는 것이 좋습니다. 그러면 네트워크 지연 시간 및 데이터 전송 비용을 줄일 수 있습니다.

데이터 전송 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

거리에 의해 발생하는 지연 시간을 최소화하기 위해 Amazon S3 Transfer Acceleration 사용

Amazon S3 Transfer Acceleration (p. 67)는 지리적으로 먼 거리에 걸쳐 클라이언트와 S3 버킷 사이에 파일을 빠르고 쉽고 안전하게 전송합니다. Transfer Acceleration은 Amazon CloudFront에서 전 세계에 분산된 엣지 로케이션을 활용합니다. 엣지 로케이션에 도착한 데이터는 최적화된 네트워크 경로를 통해 Amazon S3로 라우팅됩니다. Transfer Acceleration은 여러 대륙에 걸쳐 기가바이트 또는 테라바이트 용량의 데이터를 정기적으로 전송하는 데 이상적입니다. 전 세계의 중앙 집중식 버킷에 업로드하는 클라이언트에도 유용합니다.

Amazon S3 Transfer Acceleration 속도 비교 도구를 사용하면 Amazon S3 리전에서 속도를 높인 경우와 그렇지 않은 경우의 업로드 속도를 비교할 수 있습니다. 이 속도 비교 도구는 멀티파트 업로드를 통해 Amazon S3 Transfer Acceleration을 사용하거나 사용하지 않으면서 브라우저에서 여러 Amazon S3 리전으로 파일을 전송합니다.

최신 버전의 AWS SDK 사용

AWS SDK는 Amazon S3 성능을 최적화하기 위한 다양한 권장 지침을 기본적으로 지원합니다. SDK는 애플리케이션 내에서 Amazon S3를 활용할 수 있는 더 간단한 API를 제공하며 최신 모범 사례를 따르기 위해 정기적으로 업데이트됩니다. 예를 들어 SDK에는 HTTP 503 오류에 대한 요청을 자동으로 다시 시도하는 로직이 포함되어 있으며 느린 연결에 응답하고 적응하는 코드에 투자하고 있습니다.

SDK는 또한 경우에 따라 바이트 범위 요청을 사용하여 초당 수천 건의 요청을 달성하기 위해 수평 확장 연결을 자동화하는 Transfer Manager를 제공합니다. 최신 성능 최적화 기능을 사용하려면 최신 버전의 AWS SDK를 사용하는 것이 중요합니다.

HTTP REST API 요청을 사용하여 성능을 최적화 할 수도 있습니다. REST API를 사용할 때는 SDK에 포함된 것과 동일한 모범 사례를 따라야 합니다. 느린 요청에 대해 제한 시간 및 재시도를 허용하고 객체 데이터를 병렬로 가져올 수 있도록 다중 연결을 허용합니다. REST API 사용에 대한 자세한 내용은 Amazon Simple Storage Service API Reference 단원을 참조하십시오.

Amazon S3의 성능 디자인 패턴

Amazon S3에서 객체를 업로드하고 검색할 애플리케이션을 설계할 때, 최상의 애플리케이션 성능을 얻기 위해 모범 사례 설계 패턴을 사용합니다. 또한 애플리케이션 아키텍처를 계획할 때 고려해야 할 성능 지침 (p. 663)도 제공합니다.

성능을 최적화하려면 다음과 같은 설계 패턴을 사용할 수 있습니다.

주제

- 자주 액세스하는 콘텐츠에 캐싱 사용 (p. 665)
- 지연 시간에 민감한 애플리케이션의 제한 시간 및 재시도 (p. 666)
- 높은 처리량을 위한 수평 확장 및 요청 병렬화 (p. 666)
- Amazon S3 Transfer Acceleration을 사용하여 지리적으로 다른 데이터 전송 가속화 (p. 667)

자주 액세스하는 콘텐츠에 캐싱 사용

Amazon S3에서 데이터를 저장하는 많은 애플리케이션은 사용자가 반복적으로 요청하는 데이터의 "작업 집합"을 제공합니다. 워크로드가 공통 객체 집합에 반복적인 GET 요청을 보내는 경우 Amazon CloudFront, Amazon ElastiCache 또는 AWS Elemental MediaStore 등을 사용하여 성능을 최적화할 수 있습니다. 캐시를

성공적으로 채택하면 지연 시간이 짧아지고 데이터 전송률이 높아질 수 있습니다. 그리고 캐싱을 사용하는 애플리케이션은 Amazon S3에 직접 요청을 거의 보내지 않아 요청 비용을 줄일 수 있습니다.

Amazon CloudFront는 지리적으로 분산된 대규모 PoP(Point of Presence) 집합에서 Amazon S3의 데이터를 투명하게 캐시하는 고속 콘텐츠 전송 네트워크(CDN)입니다. 여러 리전 또는 인터넷을 통해 객체에 액세스할 수 있는 경우 CloudFront를 사용하면 객체에 액세스하는 사용자 가까이에서 데이터를 캐싱할 수 있습니다. 이로 인해 인기 있는 Amazon S3 콘텐츠를 고성능으로 전달할 수 있습니다. CloudFront에 대한 자세한 내용은 Amazon CloudFront 개발자 안내서를 참조하십시오.

Amazon ElastiCache는 관리형 인메모리 캐시입니다. ElastiCache를 사용하면 메모리에 객체를 캐시하는 Amazon EC2 인스턴스를 프로비저닝할 수 있습니다. 이 캐싱은 GET 지연 시간이 크게 감소하고 다운로드 처리량이 상당히 증가합니다. ElastiCache를 사용하려면 두 가지 모두 캐시를 핫 객체로 채우도록 로직을 수정하고 Amazon S3에서 핫 객체를 요청하기 전에 그에 대한 캐시를 확인합니다. Amazon S3 GET 성능을 개선하기 위해 ElastiCache를 사용하는 예는 블로그 게시물 Turbocharge Amazon S3 with Amazon ElastiCache for Redis를 참조하십시오.

AWS Elemental MediaStore는 Amazon S3의 비디오 워크플로 및 미디어 전달 전용으로 제작된 캐싱 및 콘텐츠 배포 시스템입니다. MediaStore는 비디오 전용 통합 스토리지 API를 제공하며, 성능에 민감한 비디오 워크로드에 권장됩니다. MediaStore에 대한 자세한 내용은 AWS Elemental MediaStore 사용 설명서.를 참조하십시오.

지연 시간에 민감한 애플리케이션의 제한 시간 및 재시도

애플리케이션이 Amazon S3에서 재시도가 필요하다는 응답을 수신하는 경우가 있습니다. Amazon S3는 버킷과 객체 이름을 연결된 객체 데이터에 매핑합니다. 애플리케이션의 요청 속도가 높으면(일반적으로 적은 수의 객체에 대해 초당 5,000건 이상의 연속 요청) HTTP 503 slowdown 응답이 수신될 수 있습니다. 이러한 오류가 발생하면 각 AWS SDK는 지수 백오프를 사용하여 자동 재시도 로직을 구현합니다. AWS SDK를 사용하지 않는 경우 HTTP 503 오류 수신 시 재시도 로직을 구현해야 합니다. 백오프 기법에 대한 자세한 내용은 Amazon Web Services 일반 참조에서 AWS의 오류 재시도 횟수 및 지수 백오프를 참조하십시오.

Amazon S3는 새로운 연속 요청 속도에 따라 자동으로 확장되어 동적으로 성능을 최적화합니다. Amazon S3가 새로운 요청 속도에 대해 내부적으로 최적화하는 동안 최적화가 완료될 때까지 일시적으로 HTTP 503 요청 응답을 받게 됩니다. Amazon S3가 새 요청 비율에 대해 성능을 내부적으로 최적화한 후에는 일반적으로 모든 요청이 재시도 없이 처리됩니다.

지연 시간에 민감한 애플리케이션의 경우 Amazon S3는 느린 작업을 추적하고 적극적으로 재시도할 것을 권고합니다. 요청을 재시도 할 때 Amazon S3에 대한 새 연결을 사용하고 새로운 DNS 조회를 수행하는 것이 좋습니다.

다양한 크기의 요청(예: 128MB 이상)을 생성할 때, 달성 중인 처리량을 추적하고 요청 중 가장 느린 5%를 재시도하는 것이 좋습니다. 일반적으로 지연 시간 중간값이 수십 밀리초 범위인 작은 요청(예: 512KB 미만)을 생성할 때는 2초 후 GET 또는 PUT 작업을 재시도하는 것이 좋습니다. 추가 재시도가 필요할 경우 가장 좋은 방법은 백오프입니다. 예를 들어 2초 후 재시도하고, 그로부터 4초 후 두 번째로 재시도하는 것이 좋습니다.

애플리케이션이 Amazon S3에 고정 크기 요청을 하는 경우, 각 요청에 대해보다 일관된 응답 시간을 기대할 것입니다. 이 경우 간단한 전략은 요청 중 가장 느린 1%를 확인하고 재시도하는 것입니다. 흔히 한 번의 재시도만으로도 지연 시간을 줄이는 데 효과적입니다.

서버 측 암호화에 AWS Key Management Service(AWS KMS)를 사용하는 경우 사용 사례에 지원되는 요청 속도에 대한 정보는 AWS Key Management Service Developer Guide의 제한을 참조하십시오.

높은 처리량을 위한 수평 확장 및 요청 병렬화

Amazon S3는 매우 큰 분산 시스템입니다. 규모를 활용하려면 병렬 요청을 Amazon S3 서비스 엔드포인트까지 수평으로 확장하는 것이 좋습니다. 이 유형의 확장 방법은 Amazon S3 내에서 요청을 분산할 뿐만 아니라, 네트워크를 통해 여러 경로에 걸쳐 부하를 분산하기에도 좋습니다.

높은 처리량 전송의 경우, Amazon S3는 다중 연결을 사용하여 병렬로 데이터를 GET 또는 PUT하는 애플리케이션 사용을 권고합니다. 예를 들어 AWS Java SDK의 Amazon S3 Transfer Manager에서 지원하며, 다른 AWS SDK의 대부분은 비슷한 구조를 제공합니다. 일부 애플리케이션의 경우 서로 다른 애플리케이션 스레드나 다른 애플리케이션 인스턴스에서 동시에 여러 요청을 실행하여 병렬 연결을 구현할 수 있습니다. 취할 수 있는 가장 좋은 방법은 애플리케이션과 액세스 중인 객체의 구조에 따라 다릅니다.

AWS SDK에서 전송 관리를 사용하는 대신 AWS SDK를 사용하여 GET 및 PUT 요청을 직접 실행할 수 있습니다. 이 방법을 사용하면 워크로드를 보다 직접적으로 조정할 수 있으며 SDK의 재시도 지원과 발생할 수 있는 HTTP 503 응답 처리 기능을 계속 활용할 수 있습니다. 일반적으로 리전 내 큰 객체를 Amazon S3에서 Amazon EC2으로 다운로드할 때 객체의 바이트 범위에 대한 동시 요청을 8–16MB의 세부 수준으로 수행하는 것이 좋습니다. 원하는 각각의 85–90MB/s의 네트워크 처리량에 대해 한 건의 동시 요청을 하십시오. 10Gb/s 네트워크 인터페이스 카드(NIC)를 포화시키려면 별도의 연결을 통해 약 15건의 동시 요청을 사용할 수 있습니다. 더 많은 연결을 통해 동시 요청을 수직 확장하여 25Gb/s 또는 100Gb/s NIC와 같은 더 빠른 NIC를 포화시킬 수 있습니다.

동시에 생성할 요청 수를 조정할 때 성능 측정이 중요합니다. 한 번에 하나의 요청으로 시작하는 것이 좋습니다. 달성되는 네트워크 대역폭과 애플리케이션이 데이터 처리에 사용하는 다른 리소스의 사용을 측정하십시오. 그런 다음 병목 리소스(즉, 사용률이 가장 높은 리소스), 따라서 유용할 가능성이 있는 요청 수를 식별할 수 있습니다. 예를 들어 한 번에 하나의 요청을 처리하면 CPU 사용량이 25%가되며 최대 4개의 동시 요청을 수용할 수 있습니다. 측정은 필수적이며, 요청 속도가 증가함에 따라 리소스 사용을 확인할 필요가 있습니다.

애플리케이션에서 REST API를 사용하여 Amazon S3에 직접 요청을 제출하는 경우 HTTP 연결 풀을 사용하고 일련의 요청에 대해 각각의 연결을 다시 사용하는 것이 좋습니다. 요청별 연결 설정을 피하면 요청마다 TCP slow-start 및 SSL(Secure Sockets Layer) 핸드셰이크를 수행할 필요가 없습니다. REST API 사용에 대한 자세한 내용은 Amazon Simple Storage Service API Reference 단원을 참조하십시오.

마지막으로, DNS 및 Amazon S3 IP 주소의 광범위한 풀로 요청이 확산되고 있는지 다시 확인하는 것이 중요합니다. DNS는 수많은 IP 엔드포인트 목록을 통해 Amazon S3 주기에 대해 쿼리합니다. 그러나 단일 IP 주소를 재사용하는 캐싱 해석기 또는 애플리케이션 코드는 주소 다양성과 그에 따른 로드 밸런싱의 이점을 얻지 못합니다. `netstat` 명령줄 도구와 같은 네트워크 유틸리티 도구는 Amazon S3와의 통신에 사용되는 IP 주소를 표시할 수 있으며, 사용할 DNS 구성에 대한 지침을 제공합니다. 이 지침에 대한 자세한 내용은 DNS 고려 사항 (p. 661) 단원을 참조하십시오.

Amazon S3 Transfer Acceleration을 사용하여 지리적으로 다른 데이터 전송 가속화

Amazon S3 Transfer Acceleration (p. 67)은 Amazon S3를 사용하여 전 세계에 분산된 클라이언트와 리전 애플리케이션 간의 물리적 거리로 인해 발생하는 지연 시간을 최소화하거나 없애는 데 효과적입니다. Transfer Acceleration은 데이터 전송을 위해 CloudFront에서 전 세계에 분산된 엣지 로케이션을 사용합니다. AWS 엣지 네트워크는 50개 이상의 로케이션에 존재합니다. 현재는 CloudFront를 통해 콘텐츠를 배포하고 Amazon Route 53에 대한 DNS 쿼리에 신속하게 응답하는 데 사용됩니다.

또한 엣지 네트워크는 Amazon S3와 주고 받는 데이터 전송을 가속화합니다. 대륙 내부 및 대륙 간에 데이터를 전송하고, 인터넷 연결이 빠르며, 대용량 객체를 사용하거나 업로드할 콘텐츠가 많은 애플리케이션에 이상적입니다. 엣지 로케이션에 도착한 데이터는 최적화된 네트워크 경로를 통해 Amazon S3로 라우팅됩니다. 일반적으로 Amazon S3 리전에서 멀어질수록 Transfer Acceleration 사용에서 기대할 수 있는 속도가 향상됩니다.

새 버킷 또는 기존 버킷에 Transfer Acceleration을 설정할 수 있습니다. 별도의 Amazon S3 Transfer Acceleration 엔드포인트를 사용하여 AWS 엣지 로케이션을 사용할 수 있습니다. Transfer Acceleration이 클라이언트 요청 성능에 유익한지 테스트하는 가장 좋은 방법은 Amazon S3 Transfer Acceleration 속도 비교 도구를 사용하는 것입니다. 네트워크 구성 및 조건은 때때로 달라지며 위치에 따라 다릅니다. 따라서 Amazon S3 Transfer Acceleration으로 업로드 성능을 향상시킬 가능성이 있는 전송에 대해서만 요금이 부과됩니다. 다른 AWS SDK와 함께 Transfer Acceleration을 사용하는 방법에 대한 정보는 Amazon S3 Transfer Acceleration 예 (p. 69) 단원을 참조하십시오.

Amazon S3 모니터링

모니터링은 Amazon S3와 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다. 발생하는 다중 지점 실패를 보다 쉽게 디버깅할 수 있도록 AWS 솔루션의 모든 부분으로부터 모니터링 데이터를 수집해야 합니다. Amazon S3에 대한 모니터링을 시작하기 전에 다음 질문에 대한 답변을 포함하는 모니터링 계획을 작성해야 합니다

- 모니터링의 목표
- 모니터링할 리소스
- 이러한 리소스를 모니터링하는 빈도
- 사용할 모니터링 도구
- 모니터링 작업을 수행할 사람
- 문제 발생 시 알려야 할 대상

주제

- 모니터링 도구 (p. 668)
- Amazon CloudWatch로 지표 모니터링 (p. 669)
- 버킷에 대한 지표 구성 (p. 676)
- Amazon S3를 사용한 로깅 (p. 678)
- AWS CloudTrail을 사용하여 Amazon S3 API 호출 로깅 (p. 679)
- AWS CloudTrail을 사용하여 Amazon S3 요청 식별 (p. 686)

모니터링 도구

AWS는 Amazon S3를 모니터링하는 데 사용할 수 있는 다양한 도구를 제공합니다. 이들 도구 중에는 모니터링을 자동으로 수행하도록 구성할 수 있는 도구도 있지만, 수동 작업이 필요한 도구도 있습니다. 모니터링 작업을 최대한 자동화하는 것이 좋습니다.

자동 모니터링 도구

다음과 같은 자동 모니터링 도구를 사용하여 Amazon S3를 관찰하고 문제 발생 시 보고할 수 있습니다.

- Amazon CloudWatch 알람 – 지정한 기간 동안 단일 지표를 감시하고 여러 기간에 대해 지정된 임계 값을 기준으로 지표 값을 기준으로 하나 이상의 작업을 수행합니다. 작업은 Amazon Simple Notification Service(Amazon SNS) 주제 또는 Amazon EC2 Auto Scaling 정책으로 송신된 통지입니다. CloudWatch 경보는 단순히 특정 상태에 있기 때문에 작업을 호출하지 않습니다. 상태가 변경되어 지정된 기간 수 동안 유지되어야 합니다. 자세한 내용은 Amazon CloudWatch로 지표 모니터링 (p. 669) 단원을 참조하십시오.
- AWS CloudTrail 로그 모니터링 – 계정 간에 로그 파일을 공유하고, CloudTrail 로그 파일을 CloudWatch Logs에 전송하여 실시간으로 모니터링하며, 로그 처리 애플리케이션을 Java로 작성하고, 로그 파일이 CloudTrail 전송 후 변경되지 않았는지 확인합니다. 자세한 내용은 AWS CloudTrail을 사용하여 Amazon S3 API 호출 로깅 (p. 679) 단원을 참조하십시오.

수동 모니터링 도구

Amazon S3 모니터링의 또 한 가지 중요한 부분은 CloudWatch 경보에 포함되지 않는 항목을 수동으로 모니터링해야 한다는 점입니다. Amazon S3, CloudWatch, Trusted Advisor 및 기타 AWS Management 콘솔 대

시보드에서는 AWS 환경의 상태를 한눈에 볼 수 있습니다. 서버 액세스 로깅을 사용하여 버킷에 대한 액세스 요청을 추적할 수 있습니다. 각 액세스 로그 레코드는 한 액세스 요청에 대한 세부 정보(요청자, 버킷 이름, 요청 시간, 요청 작업, 응답 상태, 오류 코드 등)를 제공합니다. 자세한 내용은 Amazon Simple Storage Service 개발자 가이드의 Amazon S3 서버 액세스 로깅 (p. 705) 단원을 참조하십시오.

- Amazon S3 대시보드는 다음 정보를 표시합니다.
  - 버킷과 객체 및 해당 속성
- CloudWatch 홈 페이지에 표시되는 항목은 다음과 같습니다.
  - 현재 경보 및 상태.
  - 경보 및 리소스 그래프.
  - 서비스 상태.

  또한 CloudWatch를 사용하여 다음 작업을 수행할 수도 있습니다.
  - 사용자 정의 대시보드를 만들어 원하는 서비스 모니터링.
  - 지표 데이터를 그래프로 작성하여 문제를 해결하고 추세 파악.
  - 모든 AWS 리소스 지표 검색 및 찾아보기.
  - 문제에 대해 알려주는 경보 생성 및 편집.
- AWS Trusted Advisor을(를) 통해 AWS 리소스를 모니터링함으로써 성능, 안정성, 보안 및 경제성을 향상할 수 있습니다. 모든 사용자에 대해 4가지 Trusted Advisor 검사를 수행할 수 있고, 비즈니스 또는 엔터프라이즈 지원 플랜에 따라 사용자에 대해 50개 이상의 검사를 수행할 수 있습니다. 자세한 정보는 AWS Trusted Advisor 단원을 참조하십시오.

  Trusted Advisor는 Amazon S3에 대해 다음과 같은 검사를 수행합니다.
  - Amazon S3 버킷의 로그 구성 검사
  - 공개 액세스 권한을 가진 Amazon S3 버킷에 대한 보안 검사
  - 버전 지정이 활성화되지 않았거나 버전 지정이 중지된 Amazon S3 버킷에 대한 내결함성 검사

Amazon CloudWatch로 지표 모니터링

Amazon S3용 Amazon CloudWatch 지표는 Amazon S3를 사용하는 애플리케이션의 성능을 이해하고 개선하는 데 도움을 줄 수 있습니다. Amazon S3와 함께 CloudWatch를 사용할 수 있는 몇 가지 방법이 있습니다.

- 버킷에 대한 일간 스토리지 지표 - Amazon S3에서 스토리지 데이터를 수집하여 읽을 수 있는 일간 지표로 처리하는 CloudWatch를 사용하여 bucket 스토리지를 모니터링합니다. Amazon S3에 대한 이러한 스토리지 지표는 하루에 한 번 보고되고 추가 비용 없이 모든 고객에게 제공됩니다.
- 지표 요청 - 운영 문제를 신속하게 확인하여 조치하기 위해 Amazon S3 요청을 모니터링합니다. 지표들은 약간의 처리 지연시간 후에 1분 간격으로 제공됩니다. 이들 CloudWatch 지표는 Amazon CloudWatch 사용자 지정 지표와 동일한 요금으로 청구됩니다. CloudWatch 요금에 대한 자세한 정보는 Amazon CloudWatch 요금을 참조하십시오. 옵트인하여 이들 지표를 얻는 방법에 대해 알아보려면 버킷에 대한 지표 구성 (p. 676) 항목을 참조하십시오.

  활성화되면, 모든 객체 운영에 대한 요청 지표들이 보고됩니다. 기본 설정상, 이 1분 지표들은 Amazon S3 버킷 수준에서 제공됩니다. 공유 접두사 또는 객체 태그를 사용하여 수집한 지표에 대한 필터를 정의할 수도 있습니다. 이로써 특정한 비즈니스 애플리케이션, 워크플로우, 또는 내부 조직에 맞춰 지표 필터를 적용할 수 있습니다.
- 복제 지표 - 복제 보류 중인 객체의 총 수와 크기 및 대상 리전까지의 최대 복제 시간을 모니터링합니다. S3 Replication Time Control (S3 RTC)가 활성화된 복제 규칙만 복제 지표를 게시합니다.

  CloudWatch 스토리지 및 요청 지표와 달리 접두사 및 태그별로 복제 지표를 필터링할 수 없습니다. 그러나 접두사 및 태그를 기반으로 복제 규칙을 설정할 수 있습니다. 그러면 복제 지표가 사용자가 지정한 접두사 및 태그에 대한 복제를 모니터링합니다. 자세한 내용은 the section called “S3 Replication Time Control (S3 RTC)” (p. 624) 단원을 참조하십시오.

모든 CloudWatch 통계는 15개월간 유지되므로 기록 정보를 보고 웹 애플리케이션이나 서비스가 어떻게 실행되고 있는지 전체적으로 더 잘 파악할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch란 무엇입니까?를 참조하십시오.

지표와 차원

Amazon S3에서 CloudWatch로 전송하는 스토리지 지표와 차원은 아래에 나와 있습니다.

Amazon S3 CloudWatch 버킷에 대한 일별 스토리지 지표

`AWS/S3` 네임스페이스에는 버킷에 대한 다음 일간 스토리지 지표가 포함되어 있습니다.

지표	설명
`BucketSizeBytes`	STANDARD 스토리지 클래스, INTELLIGENT_TIERING 스토리지 클래스, Standard - Infrequent Access(STANDARD_IA) 스토리지 클래스, OneZone - Infrequent Access(ONEZONE_IA), Reduced Redundancy Storage(RRS) 클래스, Deep Archive Storage(S3 Glacier Deep Archive) 클래스 또는 Glacier(GLACIER) 스토리지 클래스의 버킷 내 저장된 데이터의 양(바이트)입니다. 이 값은 버킷에 대한 모든 불완전 멀티파트 업로드의 모든 파트 크기를 포함하여 버킷(현재 객체 및 현재가 아닌 객체 모두)의 모든 객체 크기를 합산하여 계산됩니다. 올바른 스토리지 유형 필터: `StandardStorage`, `IntelligentTieringStorage`, `StandardIAStorage`, `StandardIASizeOverhead`, `StandardIAObjectOverhead`, `OneZoneIAStorage`, `OneZoneIASizeOverhead`, `ReducedRedundancyStorage`, `GlacierStorage`, `GlacierStagingStorage`, `GlacierObjectOverhead`, `GlacierS3ObjectOverhead`, `DeepArchiveStorage`, `DeepArchiveObjectOverhead`, `DeepArchiveS3ObjectOverhead`, `DeepArchiveStagingStorage`(`StorageType` 차원 참조) 단위: 바이트 유효 통계: Average
`NumberOfObjects`	GLACIER 스토리지 클래스를 제외한 모든 스토리지 클래스의 버킷에 저장된 총 객체의 수. 이 값은 버킷(현재 객체 및 현재가 아닌 객체 모두)의 모든 객체 수와 버킷에 대한 모든 불완전 멀티파트 업로드의 총 파트 수를 합산하여 계산됩니다. 올바른 스토리지 유형 필터 : `AllStorageTypes`(`StorageType` 차원 참조) 단위: 개수 유효 통계: Average

Amazon S3 CloudWatch 요청 지표

`AWS/S3` 네임스페이스에는 다음 요청 지표가 포함되어 있습니다.

지표	설명
`AllRequests`	유형에 관계 없이 Amazon S3 버킷에 실행된 HTTP 요청의 총 횟수. 필터와 함께 지표 구성을 사용하는 경우, 이 지표는 그 필터의 요건을 충족하는 버킷 내 객체들에 대해 실행된 HTTP 요청만을 반환합니다. 단위: 개수 유효 통계: Sum
`GetRequests`	Amazon S3 버킷 내 객체들에 대해 실행된 HTTP GET 요청의 횟수. 여기에는 나열 작업이 포함되지 않습니다. 단위: 개수 유효 통계: Sum Note 멀티파트 업로드 목록 조회, 파트 목록 조회, 버킷 객체 버전 가져오기 등과 같이 페이지 번호가 매겨지는 나열형 요청은 이 지표에 포함되지 않습니다.
`PutRequests`	Amazon S3 버킷 내 객체들에 대해 실행된 HTTP PUT 요청의 횟수. 단위: 개수 유효 통계: Sum
`DeleteRequests`	Amazon S3 버킷 내 객체들에 대해 실행된 HTTP DELETE 요청의 횟수. 여기에는 여러 개체 삭제 요청도 포함됩니다. 이 지표는 삭제된 객체 수가 아니라 요청 수를 나타냅니다. 단위: 개수 유효 통계: Sum
`HeadRequests`	Amazon S3 버킷에 대해 실행된 HTTP HEAD 요청의 횟수. 단위: 개수 유효 통계: Sum
`PostRequests`	Amazon S3 버킷에 대해 실행된 HTTP POST 요청의 횟수. 단위: 개수 유효 통계: Sum Note 여러 객체 삭제 및 SELECT Object Content(객체 콘텐츠 선택) 요청은 이 지표에 포함되지 않았습니다.
`SelectRequests`	Amazon S3 버킷 내 객체에 대해 이루어진 Amazon S3 객체 콘텐츠 선택 요청 건수. 단위: 개수 유효 통계: Sum

지표	설명
`SelectScannedBytes`	Amazon S3 버킷 내 Amazon S3 객체 콘텐츠 선택 요청 건으로 검사된 데이터 바이트 수. 단위: 바이트 유효한 통계: Average(요청당 바이트), Sum(기간당 바이트), Sample Count, Min, Max(p100과 동일), p0.0과 p99.9 사이의 모든 백분위수
`SelectReturnedBytes`	Amazon S3 버킷 내 Amazon S3 객체 콘텐츠 선택 요청 건으로 반환된 데이터 바이트 수. 단위: 바이트 유효한 통계: Average(요청당 바이트), Sum(기간당 바이트), Sample Count, Min, Max(p100과 동일), p0.0과 p99.9 사이의 모든 백분위수
`ListRequests`	버킷의 콘텐츠를 나열하는 HTTP 요청의 수입니다. 단위: 개수 유효 통계: Sum
`BytesDownloaded`	Amazon S3 버킷에 실행된 요청에 대해 다운로드된 바이트 수로, 여기서 응답에는 본문이 포함됩니다. 단위: 바이트 유효한 통계: Average(요청당 바이트), Sum(기간당 바이트), Sample Count, Min, Max(p100과 동일), p0.0과 p99.9 사이의 모든 백분위수
`BytesUploaded`	Amazon S3 버킷에 대해 실행된 요청 본문을 포함하여 업로드된 바이트 수입니다. 단위: 바이트 유효한 통계: Average(요청당 바이트), Sum(기간당 바이트), Sample Count, Min, Max(p100과 동일), p0.0과 p99.9 사이의 모든 백분위수
`4xxErrors`	HTTP 상태 코드가 4xx 값(클라이언트 오류)인 뷰어 요청 수입니다Amazon S3. `average` 통계는 오류율을 보여주고 `sum` 통계는 각 기간 동안 그 유형의 오류 개수를 보여줍니다. 단위: 개수 유효 통계: Average(요청당 보고), Sum(기간당 보고), Min, Max, Sample Count
`5xxErrors`	0 또는 1의 값을 가진 Amazon S3 버킷에 실행된 HTTP 5xx 서버 오류 상태 코드 요청 수입니다. `average` 통계는 오류율을 보여주고 `sum` 통계는 각 기간 동안 그 유형의 오류 개수를 보여줍니다. 단위: 개수 유효 통계: Average(요청당 보고), Sum(기간당 보고), Min, Max, Sample Count

지표	설명
`FirstByteLatency`	Amazon S3 버킷이 전체 요청을 수신할 때부터 응답이 반환되기 시작하는 때까지의 요청당 시간입니다. 단위: 밀리초 유효한 통계: Average, Sum, Min, Max(p100과 동일), Sample Count, p0.0과 p100 사이의 모든 백분위수
`TotalRequestLatency`	최초 수신된 바이트부터 Amazon S3 버킷으로 전송된 마지막 바이트까지의 요청당 경과 시간입니다. 여기에는 요청 본문을 수신하고 요청 본문을 전송하는 데 소요된 시간이 포함되며, 이것은 `FirstByteLatency`에 포함되지 않습니다. 단위: 밀리초 유효한 통계: Average, Sum, Min, Max(p100과 동일), Sample Count, p0.0과 p100 사이의 모든 백분위수

Amazon S3 CloudWatch 복제 지표

복제 지표를 사용하여 복제 보류 중인 객체의 총 수와 크기 및 대상 리전까지의 최대 복제 시간을 모니터링할 수 있습니다. S3 Replication Time Control (S3 RTC)가 활성화된 복제 규칙만 복제 지표를 게시할 수 있습니다.

CloudWatch 스토리지 및 요청 지표와 달리 접두사 또는 태그별로 복제 지표를 필터링할 수 없습니다. 그러나 접두사 및 태그를 기반으로 복제 규칙을 설정할 수 있습니다. 그런 다음 복제 지표는 지정한 접두사 및 태그에 대한 복제를 모니터링합니다. 자세한 내용은 the section called “S3 Replication Time Control (S3 RTC)” (p. 624) 단원을 참조하십시오.

> **Note**
>
> Amazon CloudWatch에서 복제 지표에 대한 경보를 활성화할 수 있습니다. 복제 지표에 대한 경보를 설정할 때 누락된 데이터 처리 필드를 누락된 데이터를 무시로 처리(경보 상태 유지)로 설정합니다.

지표	설명
`ReplicationLatency`	지정된 복제 규칙에 대해 복제 대상 리전이 소스 리전 뒤에 있는 최대 시간(초)입니다. 단위: 초 유효한 통계: Max
`BytesPendingReplication`	지정된 복제 규칙에 대해 복제 보류 중인 객체의 총 바이트 수입니다. 단위: 바이트 유효한 통계: Max
`OperationsPendingReplication`	지정된 복제 규칙에 대해 복제 보류 중인 작업 수입니다. 단위: 개수 유효한 통계: Max

Amazon S3 CloudWatch 차원

다음 차원은 Amazon S3 지표 필터링에 사용됩니다.

차원	설명
`BucketName`	이 차원은 식별된 버킷에 한해 요청한 데이터를 필터링합니다.
`StorageType`	이 차원은 다음 스토리지 유형별 버킷에 저장된 데이터를 필터링합니다. • `StandardStorage` - STANDARD 스토리지 클래스의 객체에 사용된 바이트 수입니다. • `IntelligentTieringFAStorage` - INTELLIGENT_TIERING 스토리지 클래스의 Frequent Access 티어에 있는 객체에 사용된 바이트 수입니다. • `IntelligentTieringIAStorage` - INTELLIGENT_TIERING 스토리지 클래스의 Infrequent Access 티어에 있는 객체에 사용된 바이트 수입니다. • `StandardIAStorage` - STANDARD_IA(Standard - Infrequent Access) 스토리지 클래스의 객체에 사용된 바이트 수입니다. • `StandardIASizeOverhead` - STANDARD_IA 스토리지 클래스에서 128KB보다 작은 객체에 사용된 바이트 수입니다. • `OneZoneIAStorage` - ONEZONE_IA(OneZone - Infrequent Access) 스토리지 클래스의 객체에 사용된 바이트 수입니다. • `OneZoneIASizeOverhead` - ONEZONE_IA 스토리지 클래스에서 128KB보다 작은 객체에 사용된 바이트 수입니다. • `ReducedRedundancyStorage` - Reduced Redundancy Storage(RRS) 클래스의 객체에 사용된 바이트 수입니다. • `GlacierStorage` - GLACIER 스토리지 클래스의 객체에 사용된 바이트 수입니다. • `GlacierStagingStorage` - GLACIER 스토리지 클래스에서 객체에 대한 CompleteMultipartUpload 요청이 완료되기 전 멀티파트 객체의 일부에 사용된 바이트 수입니다. • `GlacierObjectOverhead` - 보관된 각 객체에 대해 GLACIER는 인덱스 및 관련 메타데이터용으로 32KB의 스토리지를 추가합니다. 이 추가 데이터는 객체를 식별하고 복원하는 데 필요합니다. 이 추가 스토리지에 대해 GLACIER 요금이 청구됩니다. • `GlacierS3ObjectOverhead` - GLACIER에 보관되는 각 객체에 대해 Amazon S3는 객체 이름 및 기타 메타데이터용으로 8KB의 스토리지를 사용합니다. 이 추가 스토리지에 대해 STANDARD 요금이 청구됩니다. • `DeepArchiveStorage` - S3 Glacier Deep Archive 스토리지 클래스의 객체에 사용된 바이트 수입니다. • `DeepArchiveObjectOverhead` - 보관된 각 객체에 대해 S3 Glacier Deep Archive는 인덱스 및 관련 메타데이터용으로 32KB의 스토리지를 추가합니다. 이 추가 데이터는 객체를 식별하고 복원하는 데 필요합니다. 이 추가 스토리지에 대해 S3 Glacier Deep Archive 요금이 청구됩니다. • `DeepArchiveS3ObjectOverhead` - S3 Glacier Deep Archive에 보관되는 각 객체에 대해 Amazon S3은 객체 이름 및 기타 메타데이터용으로 8KB의 스토리지를 사용합니다. 이 추가 스토리지에 대해 STANDARD 요금이 청구됩니다.

차원	설명
	• `DeepArchiveStagingStorage` - S3 Glacier Deep Archive 스토리지 클래스에서 객체에 대한 CompleteMultipartUpload 요청이 완료되기 전 멀티파트 객체의 일부에 사용된 바이트 수입니다.
`FilterId`	이 차원은 특정 버킷에 대해 사용자가 요청 지표를 지정하는 지표 구성들을 필터링합니다(예: 접두사 또는 태그). 지표 구성을 만들 때 필터 ID를 지정하십시오. 자세한 정보는 버킷에 대한 지표 구성. 단원을 참조하십시오.

CloudWatch 측정치 보기

다음 절차에 따라 Amazon S3에 대한 스토리지 측정치를 볼 수 있습니다. 관련된 Amazon S3 지표를 받기 위해서는 시작 및 종료 타임스탬프를 설정해야 합니다. 지정한 24시간 동안의 지표를 받으려면 시간 기간을 86400초(하루에 해당하는 초)로 설정합니다. 또한 `BucketName` 및 `StorageType` 차원을 설정하십시오.

예를 들어, AWS CLI를 사용하여 특정 버킷의 평균 크기를 바이트 단위로 받을 경우 다음 명령을 사용할 수 있습니다.

```
aws cloudwatch get-metric-statistics --metric-name BucketSizeBytes --namespace AWS/S3
 --start-time 2016-10-19T00:00:00Z --end-time 2016-10-20T00:00:00Z --statistics Average
 --unit Bytes --region us-west-2 --dimensions Name=BucketName,Value=ExampleBucket
 Name=StorageType,Value=StandardStorage --period 86400 --output json
```

이 예에서는 다음과 같은 출력을 생성합니다.

```
{
    "Datapoints": [
        {
            "Timestamp": "2016-10-19T00:00:00Z",
            "Average": 1025328.0,
            "Unit": "Bytes"
        }
    ],
    "Label": "BucketSizeBytes"
}
```

CloudWatch 콘솔을 사용한 메트릭 확인

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 [Metrics]를 선택합니다.
3. S3 네임스페이스를 선택합니다.
4. (선택 사항) 지표를 보려면 검색 상자에 지표 이름을 입력합니다.
5. (선택 사항) StorageType 차원으로 필터링하려면 검색 상자에 스토리지 클래스 이름을 입력합니다.

AWS CLI를 사용하여 AWS 계정에 대해 저장된 유효한 측정치 목록을 보려면

- 명령 프롬프트에서 다음 명령을 사용합니다.

```
aws cloudwatch list-metrics --namespace "AWS/S3"
```

관련 리소스

- Amazon CloudWatch Logs API Reference
- Amazon CloudWatch 사용 설명서
- AWS CLI Command Reference의 list-metrics 작업
- AWS CLI Command Reference의 get-metric-statistics 작업
- 버킷에 대한 지표 구성 (p. 676)를 선택하십시오.

버킷에 대한 지표 구성

Amazon S3에 대한 Amazon CloudWatch 요청 지표의 경우, 1분 CloudWatch 지표를 수신하고 CloudWatch 경보를 설정하며 CloudWatch 대시보드에 액세스하여 사용자의 Amazon S3 스토리지 운영과 성능을 거의 실시간으로 확인할 수 있습니다. 클라우드 스토리지에 의존하는 애플리케이션의 경우, 이 지표를 사용하면 운영 문제를 신속하게 확인하여 조치할 수 있습니다. 이것을 활성화하면 이 1분 지표들이 기본적으로 Amazon S3 버킷 수준에서 제공됩니다.

버킷의 객체에 대한 CloudWatch 요청 지표를 얻으려면 그 버킷에 대해 측정 구성을 생성해야 합니다. 공유 접두사 또는 객체 태그를 사용하여 수집한 지표에 대한 필터를 정의할 수도 있습니다. 이로써 특정한 비즈니스 애플리케이션, 워크플로우, 또는 내부 조직에 맞춰 지표 필터를 적용할 수 있습니다.

사용 가능한 CloudWatch 지표 및 스토리지와 요청 지표의 차이점에 대한 자세한 내용은 Amazon CloudWatch로 지표 모니터링 (p. 669) 단원을 참조하십시오.

지표 구성을 사용할 때는 다음 사항에 주의하십시오.

- 버킷당 최대 1,000개의 지표 구성을 설정할 수 있습니다.
- 여러 가지 필터를 사용하여 지표 구성에 포함시킬 버킷 객체를 선택할 수 있습니다. 공유 접두사 또는 객체 태그로 필터링함으로써 특정한 비즈니스 애플리케이션, 워크플로우, 또는 내부 조직에 맞춰 지표 필터를 적용할 수 있습니다. 전체 버킷에 대한 지표를 요청하려면 필터 없이 지표 구성을 생성합니다.
- 지표 구성은 요청 지표를 활성화할 때만 필요합니다. 버킷 수준의 일간 스토리지 지표는 항상 켜져 있으며, 추가 비용 없이 제공됩니다. 현재는 필터링된 객체 하위 세트에 대한 일간 스토리지 지표를 확인할 수 없습니다.
- 각각의 지표 구성은 사용 가능한 요청 지표 (p. 670) 전체를 활성화합니다. 작업별 지표(예: `PostRequests`)는 해당하는 작업 유형에 대한 요청이 버킷 또는 필터에 들어올 때만 보고됩니다.
- 객체 수준 운영에 대한 요청 지표들이 보고됩니다. 또한 버킷 가져오기(개체 목록 조회), 버킷 객체 버전 가져오기, 멀티파트 업로드 목록 조회와 같이 버킷 콘텐츠를 나열하는 작업에 대해서도 보고되지만 버킷의 다른 작업에 대해서는 보고되지 않습니다.
- 요청 지표는 접두사별 필터링을 지원하지만 스토리지 지표는 지원하지 않습니다.

CloudWatch 지표 전송은 보장되지 않음

CloudWatch 지표를 전송하고자 최선을 다할 것입니다. 요청 지표가 있는 Amazon S3 객체에 대한 요청은 대개 CloudWatch로 데이터 요소를 전송하게 됩니다.

모든 지표가 제때 전송될 것이라고 보장할 수는 없습니다. 특정 요청에 대한 데이터 요소는 그 요청이 실제로 처리되었을 때보다 더 늦은 타임스탬프와 함께 반환될 수도 있습니다. 또는 어떤 시점의 데이터 요소는 CloudWatch를 통해 제공되기 전에 지연될 수도 있으며 어떤 경우에는 아예 전송되지 않을 수도 있습니다. CloudWatch 요청 지표는 버킷에 대한 트래픽의 특성을 거의 실시간으로 파악할 수 있도록 합니다. 모든 요청의 완벽한 기록을 제공할 목적으로 개발된 것이 아닙니다.

완벽한 전송을 보장할 수 없는 그 특성에 따라 대금 및 비용 관리 대시보드에 제공되는 보고서에는 버킷 지표에 나타나지 않는 액세스 요청이 하나 이상 포함될 수 있습니다.

지표 구성 필터링

CloudWatch 지표 구성을 설정할 때, 하나의 버킷 내에서 관련된 객체들의 그룹들로 구성을 필터링할 수 있습니다. 다음 요소들 중 하나 이상에 기초하여 버킷 내의 객체들이 한 지표 구성에 포함되도록 필터링할 수 있습니다.

- 객체 키 이름 접두사 – Amazon S3 데이터 모델이 단일 구조이긴 하지만, 접두사를 사용하여 계층을 유추할 수 있습니다. Amazon S3 콘솔에서는 폴더 개념에 이 같은 접두사를 지원합니다. 접두사로 필터링할 경우, 동일한 접두사를 갖는 객체들이 지표 구성에 포함됩니다.
- 태그 – 객체에 태그, 키 값 이름 페어를 추가할 수 있습니다. 태그를 사용하면 객체들을 쉽게 찾아서 조직할 수 있습니다. 또, 태그를 지표 구성을 위한 필터로 사용할 수도 있습니다.

필터를 지정하는 경우, 단일 객체에서 작동하는 요청만 해당 필터와 일치하므로 보고된 지표에 그 요청만 포함됩니다. 여러 객체 삭제 요청이나 `List` 요청은 필터를 사용해도 구성 지표를 반환하지 않습니다.

더 복잡한 필터링을 요청하려면 요소를 둘 이상 선택하십시오. 그러면 해당 요소가 전부 있는 객체만 지표 구성에 포함됩니다. 필터를 설정하지 않으면 버킷 내의 모든 객체가 지표 구성에 포함됩니다.

지표 구성 추가 방법

Amazon S3 콘솔을 통해, AWS CLI를 사용하여 또는 Amazon S3 REST API를 사용하여 버킷에 지표 구성들을 추가할 수 있습니다. AWS Management 콘솔에서 이 작업을 수행하는 방법에 관한 설명은 Amazon Simple Storage Service 콘솔 사용 설명서의 요청 지표를 S3 버킷용으로 구성하려면 어떻게 해야 합니까?를 참조하십시오.

AWS CLI를 사용하여 지표 구성을 추가하려면

1. AWS CLI를 설치 및 설정합니다. 지침은 AWS Command Line Interface 사용 설명서의 AWS 명령줄 인터페이스 시작하기 단원을 참조하십시오.
2. 터미널을 엽니다.
3. 다음 명령을 실행하여 지표 구성을 추가합니다.

```
aws s3api put-bucket-metrics-configuration --endpoint https://s3.us-
west-2.amazonaws.com --bucket bucket-name --id metrics-config-id --metrics-
configuration '{"Id":"metrics-config-id","Filter":{"Prefix":"prefix1"}}'
```

4. 구성이 추가되었는지 확인하려면 다음 명령을 실행하십시오.

```
aws s3api get-bucket-metrics-configuration --endpoint https://s3.us-
west-2.amazonaws.com --bucket bucket-name --id metrics-config-id
```

그러면 다음과 같은 응답이 반환됩니다.

```
{
    "MetricsConfiguration": {
        "Filter": {
            "Prefix": "prefix1"
        },
        "Id": "metrics-config-id"
    }
}
```

Amazon S3 REST API를 이용한 프로그래밍 방법으로 지표 구성을 추가할 수도 있습니다. 자세한 내용은 Amazon Simple Storage Service API Reference에서 다음 주제를 참조하십시오.

- 버킷 지표 구성 적용
- 버킷 지표 구성 가져오기
- 버킷 지표 구성 목록 조회
- 버킷 지표 구성 삭제

Amazon S3를 사용한 로깅

Amazon S3 리소스에 대해 사용자, 역할 또는 AWS 서비스가 수행하는 작업을 기록하고 감사 및 규정 준수 목적으로 로그 레코드를 유지 관리할 수 있습니다. 이를 수행하려면, 서버 액세스 로깅 (p. 705), AWS CloudTrail 로그 또는 이 두 가지의 조합을 사용할 수 있습니다. Amazon S3 리소스의 버킷 및 객체 수준 작업 로깅에 AWS CloudTrail을 사용하는 것이 좋습니다.

다음 표에 AWS CloudTrail 로그와 Amazon S3 서버 액세스 로그의 주요 속성이 나와 있습니다.

로그 속성	AWS CloudTrail	Amazon S3 서버 로그
다른 시스템(CloudWatch Logs, CloudWatch 이벤트)으로 전달할 수 있음	예	
둘 이상의 대상으로 로그 전송(예: 서로 다른 두 버킷으로 동일한 로그 전송)	예	
객체의 하위 세트(접두사)에 대해 로그 켜기	예	
교차 계정 로그 전송(서로 다른 계정이 소유한 대상 및 소스 버킷)	예	
디지털 서명/해싱을 사용하여 로그 파일의 무결성 검증	예	
로그 파일에 대해 기본/선택 암호화	예	
객체 작업(Amazon S3 API 사용)	예	예
버킷 작업(Amazon S3 API 사용)	예	예
로그에 대해 검색 가능한 UI	예	
객체 잠금 파라미터의 필드, Amazon S3는 로그 레코드의 속성 선택	예	
로그 레코드의 `Object Size`, `Total Time`, `Turn-Around Time`, `HTTP Referer`의 필드		예
수명 주기 전환, 만료, 복원		예
일괄 삭제 작업에서 키 로깅		예
인증 실패[1]		예
로그가 전달되는 계정	버킷 소유자[2] 및 요청자	버킷 소유자만
Performance and Cost	AWS CloudTrail	Amazon S3 Server Logs
가격	관리 이벤트(첫 전달)는 무료,	로그 저장 외에 추가 비용 없음

로그 속성	AWS CloudTrail	Amazon S3 서버 로그
	로그 저장 외에 데이터 이벤트에도 요금 발생	
로그 전달 속도	5분마다 데이터 이벤트, 15분마다 관리 이벤트	몇 시간 내에
로그 형식	JSON	공백으로 구분되고 줄 바꿈으로 구분된 레코드가 있는 로그 파일

참고:

1. CloudTrail은 인증에 실패한 요청(제공된 자격 증명이 유효하지 않은 요청)에 대해 로그를 전달하지 않습니다. 그러나 권한 부여에 실패한 요청(`AccessDenied`)과 익명 사용자의 요청에 대한 로그는 포함합니다.
2. S3 버킷 소유자는 계정이 요청의 객체를 소유했거나 이에 대한 모든 액세스를 보유한 경우에만 CloudTrail 로그를 수신합니다. 자세한 내용은 교차 계정 시나리오에서의 객체 수준 작업 (p. 682) 단원을 참조하십시오.

AWS CloudTrail을 사용하여 Amazon S3 API 호출 로깅

Amazon S3은 Amazon S3에서 사용자, 역할 또는 AWS 서비스가 수행한 작업의 레코드를 제공하는 서비스인 AWS CloudTrail과 통합됩니다. CloudTrail은 Amazon S3 콘솔의 호출 및 Amazon S3 API에 대한 코드 호출의 호출을 포함하여 Amazon S3에 대한 API 호출의 하위 세트를 이벤트로 캡처합니다. 추적을 생성하면 CloudTrail 이벤트를 비롯하여 Amazon S3 이벤트를 Amazon S3 버킷으로 지속적으로 배포할 수 있습니다. 추적을 구성하지 않은 경우 Event history(이벤트 기록)에서 CloudTrail 콘솔의 최신 이벤트를 볼 수도 있습니다. CloudTrail에서 수집하는 정보를 사용하여 Amazon S3에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

그 구성 및 활성화 방법을 포함하여 CloudTrail에 대한 자세한 내용은 AWS CloudTrail User Guide를 참조하십시오.

CloudTrail의 Amazon S3 정보

CloudTrail은 계정 생성 시 AWS 계정에서 활성화됩니다. 지원되는 이벤트 활동이 Amazon S3에서 이루어지면 해당 활동이 이벤트 이력의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 내용은 CloudTrail 이벤트 기록에서 이벤트 보기를 참조하십시오.

Amazon S3 이벤트를 비롯하여 AWS 계정에 이벤트를 지속적으로 기록하려면 추적을 생성하십시오. 추적은 CloudTrail이 Amazon S3 버킷으로 로그 파일을 전송할 수 있도록 합니다. 콘솔에서 추적을 생성하면 기본적으로 모든 리전에 추적이 적용됩니다. 추적은 AWS 파티션에 있는 모든 리전의 이벤트를 로깅하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또는 CloudTrail 로그에서 수집된 이벤트 데이터를 추가 분석 및 처리하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하십시오.

- 추적 생성 개요
- CloudTrail 지원 서비스 및 통합

- CloudTrail에 대한 Amazon SNS 알림 구성
- 여러 리전에서 CloudTrail 로그 파일 받기 및 여러 계정에서 CloudTrail 로그 파일 받기

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 자격 증명 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 IAM 사용자 자격 증명으로 했는지 여부
- 역할 또는 연합된 사용자에 대한 임시 보안 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지 여부.

자세한 내용은 CloudTrail userIdentity 요소를 참조하십시오.

원하는 기간만큼 버킷에 로그 파일을 저장할 수 있습니다. 그러나 Amazon S3 수명 주기 규칙을 정의하여 자동으로 로그 파일을 보관하거나 삭제할 수도 있습니다. 기본적으로 로그 파일은 Amazon S3 서버 쪽 암호화(SSE)를 사용하여 암호화합니다.

CloudTrail 로깅을 통해 추적되는 Amazon S3 버킷 수준 작업

기본적으로 CloudTrail은 버킷 수준 작업을 기록합니다. Amazon S3 레코드는 다른 AWS 서비스 레코드와 함께 로그 파일에 기록됩니다. CloudTrail은 기간 및 파일 크기에 따라 새 파일을 언제 생성하고 쓸지를 결정합니다.

이 단원의 표에는 CloudTrail 로깅이 지원되는 Amazon S3 버킷 수준 작업이 나와 있습니다.

CloudTrail 로깅을 통해 추적되는 Amazon S3 버킷 수준 작업

REST API 이름	CloudTrail 로그에 사용되는 API 이벤트 이름
DELETE Bucket	DeleteBucket
DELETE Bucket cors	DeleteBucketCors
DELETE 버킷 암호화	DeleteBucketEncryption
DELETE Bucket lifecycle	DeleteBucketLifecycle
DELETE Bucket policy	DeleteBucketPolicy
DELETE Bucket replication	DeleteBucketReplication
버킷 태깅 삭제(DELETE)	DeleteBucketTagging
DELETE 버킷 웹 사이트	DeleteBucketWebsite
GET Bucket acl	GetBucketAcl
GET Bucket cors	GetBucketCors
GET 버킷 암호화	GetBucketEncryption
GET Bucket lifecycle	GetBucketLifecycle
GET Bucket location	GetBucketLocation
GET Bucket logging	GetBucketLogging
GET Bucket notification	GetBucketNotification
GET Bucket policy	GetBucketPolicy

REST API 이름	CloudTrail 로그에 사용되는 API 이벤트 이름
GET Bucket replication	GetBucketReplication
GET Bucket request payment	GetBucketRequestPayment
GET Bucket tagging	GetBucketTagging
GET Bucket versioning	GetBucketVersioning
GET 버킷 웹 사이트	GetBucketWebsite
서비스 확인(GET)(모든 버킷 조회)	ListBuckets
PUT Bucket	CreateBucket
PUT Bucket acl	PutBucketAcl
PUT Bucket cors	PutBucketCors
PUT 버킷 암호화	PutBucketEncryption
PUT Bucket lifecycle	PutBucketLifecycle
PUT Bucket logging	PutBucketLogging
PUT Bucket notification	PutBucketNotification
PUT Bucket policy	PutBucketPolicy
PUT Bucket replication	PutBucketReplication
PUT Bucket request payment	PutBucketRequestPayment
PUT Bucket tagging	PutBucketTagging
PUT Bucket versioning	PutBucketVersioning
PUT 버킷 웹 사이트	PutBucketWebsite

이러한 API 작업들 이외에 OPTIONS 객체 객체 수준 작업을 사용할 수도 있습니다. 이 작업은 버킷의 CORS 구성을 확인하므로 CloudTrail 로깅에서 버킷 수준 작업처럼 간주됩니다.

CloudTrail 로깅을 통해 추적되는 Amazon S3 객체 수준 작업

객체 수준 Amazon S3 작업을 위해 CloudTrail 로그를 가져올 수도 있습니다. 이를 위해 추적에 대한 Amazon S3 객체를 지정하십시오. 사용자의 계정에 객체 수준 작업이 나타나면 CloudTrail가 추적 설정을 평가합니다. 이벤트가 추적에서 지정한 객체와 일치할 경우, 그 이벤트가 기록됩니다. 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 AWS CloudTrail 데이터 이벤트로 S3 버킷에 대해 객체 수준 로깅을 활성화하려면 어떻게 합니까?와 AWS CloudTrail User Guide의 추적을 위해 데이터 이벤트 로깅을 참조하십시오.

다음 표는 CloudTrail가 기록할 수 있는 객체 수준 작업의 목록입니다.

REST API 이름	CloudTrail 로그에 사용되는 API 이벤트 이름
멀티파트 업로드 중단	AbortMultipartUpload
멀티파트 업로드 완료	CompleteMultipartUpload

REST API 이름	CloudTrail 로그에 사용되는 API 이벤트 이름
다중 객체 삭제	DeleteObjects
DELETE Object	DeleteObject
GET Object	GetObject
GET Object ACL	GetObjectAcl
GET Object 태그 지정	GetObjectTagging
GET Object torrent	GetObjectTorrent
HEAD Object	HeadObject
멀티파트 업로드 시작	CreateMultipartUpload
파트 목록 조회	ListParts
POST Object	PostObject
POST Object restore	RestoreObject
PUT Object	PutObject
PUT Object acl	PutObjectAcl
PUT Object 태그 지정	PutObjectTagging
PUT Object - Copy	CopyObject
SELECT Object Content	SelectObjectContent
파트 업로드	UploadPart
Upload Part - Copy	UploadPartCopy

이러한 작업 이외에, 일부 조건에서는 다음과 같은 버킷 수준 작업을 사용하여 CloudTrail 로그를 객체 수준 Amazon S3 작업으로 가져올 수 있습니다.

- GET Bucket(List Objects) 버전 2 – 추적에서 지정한 접두사를 선택합니다.
- GET Bucket Object 버전 – 추적에서 지정한 접두사를 선택합니다.
- HEAD Bucket – 버킷과 빈 접두사를 지정합니다.
- Delete Multiple Objects – 버킷과 빈 접두사를 지정합니다.

> **Note**
>
> CloudTrail은 다중 객체 삭제 작업을 사용하여 삭제된 키의 키 이름을 로깅하지 않습니다.

교차 계정 시나리오에서의 객체 수준 작업

다음은 교차 계정 시나리오에서의 객체 수준 API 호출과 관련한 특별한 용례로, CloudTrail 로그가 어떻게 보고되는지 보여줍니다. CloudTrail은 항상 로그를 요청자(API 호출을 한 사람)에게 전송합니다. 교차 계정 액세스를 설정할 때, 이 단원의 예시들을 고려하십시오.

> **Note**
>
> 이 예에서는 CloudTrail 로그가 올바르게 구성되어 있다고 가정합니다.

예 1: CloudTrail가 버킷 소유자에게 액세스 로그를 전송함

CloudTrail는 버킷 소유자가 동일한 객체 API에 대한 권한을 가진 경우에만 버킷 소유자에게 액세스 로그를 전송합니다. 다음 교차 계정 시나리오를 고려해 보십시오.

- 계정-A가 버킷을 소유하고 있습니다.
- 계정-B(요청자)가 그 버킷의 객체에 액세스하려고 시도합니다.

CloudTrail는 항상 객체 수준 API 액세스 로그를 요청자에게 전송합니다. 또한 CloudTrail는 버킷 소유자가 그 객체에 대해 동일한 API 작동 권한을 가진 경우에만 버킷 소유자에게 동일한 로그를 전송합니다.

> **Note**
>
> 버킷 소유자가 객체 소유자이기도 한 경우, 버킷 소유자는 객체 액세스 로그를 얻습니다. 그렇지 않은 경우, 동일한 객체-액세스 API 로그를 얻으려면 버킷 소유자가 객체 ACL을 통해 동일한 객체 API에 대한 권한을 얻어야 합니다.

예 2: CloudTrail가 객체 ACL 설정에 사용된 이메일 주소들을 확산시키지 않음

다음 교차 계정 시나리오를 고려해 보십시오.

- 계정-A가 버킷을 소유하고 있습니다.
- 계정-B(요청자)가 이메일 주소를 사용하여 객체 ACL 권한 부여를 설정하는 요청을 전송합니다. ACL에 대한 자세한 내용은 ACL(액세스 통제 목록) 개요 (p. 438) 단원을 참조하십시오.

이 요청은 이메일 정보와 함께 로그를 가져옵니다. 그러나 버킷 소유자는— 예 1에서와 같이 로그를 수신할 자격이 있는 경우— 이벤트를 보고하는 CloudTrail 로그를 가져옵니다. 그러나 버킷 소유자는 ACL 구성 정보, 특히 피부여자 이메일과 권한 부여를 가져오지 않습니다. 로그가 버킷 소유자에게 제공하는 유일한 정보는 ACL API 호출이 계정-B에 의해 실행되었다는 것입니다.

Amazon S3 SOAP API 호출을 통한 CloudTrail 추적

CloudTrail은 Amazon S3 SOAP API 호출을 추적합니다. Amazon S3 HTTP를 통한 SOAP 지원은 중단되었지만 HTTPS를 통해 계속해서 사용할 수 있습니다. Amazon S3 SOAP 지원에 대한 자세한 내용은 부록 A: SOAP API 사용 (p. 740)을 참조하십시오.

> **Important**
>
> 최신 Amazon S3 기능은 SOAP를 지원하지 않습니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

CloudTrail 로깅을 통해 추적되는 Amazon S3 SOAP 작업

SOAP API 이름	CloudTrail 로그에 사용되는 API 이벤트 이름
ListAllMyBuckets	ListBuckets
CreateBucket	CreateBucket
DeleteBucket	DeleteBucket
GetBucketAccessControlPolicy	GetBucketAcl
SetBucketAccessControlPolicy	PutBucketAcl
GetBucketLoggingStatus	GetBucketLogging

SOAP API 이름	CloudTrail 로그에 사용되는 API 이벤트 이름
SetBucketLoggingStatus	PutBucketLogging

Amazon S3 서버 액세스 로그 및 CloudWatch Logs와 함께 CloudTrail 로그 사용

AWS CloudTrail 로그는 Amazon S3의 사용자, 역할 또는 AWS 서비스가 수행한 작업의 레코드를 제공하는 반면, Amazon S3 서버 액세스 로그는 S3 버킷에 대해 이루어진 요청에 대한 세부 레코드를 제공합니다. 여러 로그의 작동 방식과 속성, 성능 및 비용에 대한 자세한 내용은 the section called “Amazon S3를 사용한 로깅” (p. 678) 단원을 참조하십시오.

AWS CloudTrail 로그를 Amazon S3의 서버 액세스 로그와 함께 사용할 수 있습니다. CloudTrail 로그는 Amazon S3 버킷 수준 및 객체 수준 작업에 대한 자세한 API 추적을 제공합니다. Amazon S3의 서버 액세스 로그는 Amazon S3의 데이터에 대한 객체 수준 작업을 파악할 수 있게 해줍니다. 서버 액세스 로그에 대한 자세한 내용은 Amazon S3 서버 액세스 로깅 (p. 705)을 참조하십시오.

CloudTrail 로그와 Amazon S3에 대한 CloudWatch를 함께 사용할 수도 있습니다. CloudTrail과 CloudWatch Logs를 통합하면 CloudTrail에서 캡처한 S3 버킷 수준의 API 동작이 사용자가 지정한 CloudWatch 로그 그룹의 CloudWatch 로그 스트림으로 전송됩니다. 특정 API를 모니터링하는 CloudWatch 경보를 생성하여 해당 API 동작이 발생했을 때 이메일 알림을 받을 수 있습니다. 특정 API 동작을 모니터링하기 위한 CloudWatch 경보에 대한 자세한 내용은 AWS CloudTrail User Guide를 참조하십시오. CloudWatch와 Amazon S3를 함께 사용하는 방법에 대한 자세한 내용은 Amazon CloudWatch로 지표 모니터링 (p. 669) 단원을 참조하십시오.

예: Amazon S3 로그 파일 항목

추적은 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 제공할 수 있도록 해 주는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함됩니다. 이벤트는 원본의 단일 요청을 나타내며 요청된 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보가 들어 있습니다. CloudTrail 로그 파일은 퍼블릭 API 호출의 주문 스택 추적이 아니기 때문에 특정 순서로 표시되지 않습니다.

다음 예제는 GET Service, PUT Bucket acl, GET Bucket versioning 작업을 나타내는 CloudTrail 로그 항목을 보여줍니다.

```
{
    "Records": [
    {
        "eventVersion": "1.03",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "111122223333",
            "arn": "arn:aws:iam::111122223333:user/myUserName",
            "accountId": "111122223333",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "myUserName"
        },
        "eventTime": "2019-02-01T03:18:19Z",
        "eventSource": "s3.amazonaws.com",
        "eventName": "ListBuckets",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "[]",
        "requestParameters": {
            "host": [
                "s3.us-west-2.amazonaws.com"
```

```
            ]
        },
        "responseElements": null,
        "additionalEventData": {
            "SignatureVersion": "SigV2",
            "AuthenticationMethod": "QueryString"
    },
        "requestID": "47B8E8D397DCE7A6",
        "eventID": "cdc4b7ed-e171-4cef-975a-ad829d4123e8",
        "eventType": "AwsApiCall",
        "recipientAccountId": "111122223333"
    },
    {
       "eventVersion": "1.03",
       "userIdentity": {
            "type": "IAMUser",
            "principalId": "111122223333",
            "arn": "arn:aws:iam::111122223333:user/myUserName",
            "accountId": "111122223333",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "myUserName"
        },
      "eventTime": "2019-02-01T03:22:33Z",
      "eventSource": "s3.amazonaws.com",
      "eventName": "PutBucketAcl",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "",
      "userAgent": "[]",
      "requestParameters": {
          "bucketName": "",
          "AccessControlPolicy": {
              "AccessControlList": {
                  "Grant": {
                      "Grantee": {
                          "xsi:type": "CanonicalUser",
                          "xmlns:xsi": "http://www.w3.org/2001/XMLSchema-instance",
                          "ID":
"d25639fbe9c19cd30a4c0f43fbf00e2d3f96400a9aa8dabfbbebe1906Example"
                       },
                      "Permission": "FULL_CONTROL"
                   }
              },
              "xmlns": "http://s3.amazonaws.com/doc/2006-03-01/",
              "Owner": {
                  "ID": "d25639fbe9c19cd30a4c0f43fbf00e2d3f96400a9aa8dabfbbebe1906Example"
              }
          }
          "host": [
              "s3.us-west-2.amazonaws.com"
          ],
          "acl": [
              ""
          ]
      },
      "responseElements": null,
      "additionalEventData": {
          "SignatureVersion": "SigV4",
          "CipherSuite": "ECDHE-RSA-AES128-SHA",
          "AuthenticationMethod": "AuthHeader"
      },
      "requestID": "BD8798EACDD16751",
      "eventID": "607b9532-1423-41c7-b048-ec2641693c47",
      "eventType": "AwsApiCall",
      "recipientAccountId": "111122223333"
    },
    {
```

```
      "eventVersion": "1.03",
      "userIdentity": {
          "type": "IAMUser",
          "principalId": "111122223333",
          "arn": "arn:aws:iam::111122223333:user/myUserName",
          "accountId": "111122223333",
          "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
          "userName": "myUserName"
        },
      "eventTime": "2019-02-01T03:26:37Z",
      "eventSource": "s3.amazonaws.com",
      "eventName": "GetBucketVersioning",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "",
      "userAgent": "[]",
      "requestParameters": {
          "host": [
              "s3.us-west-2.amazonaws.com"
          ],
          "bucketName": "aws-s3-bucket1",
          "versioning": [
              ""
          ]
      },
      "responseElements": null,
      "additionalEventData": {
          "SignatureVersion": "SigV4",
          "CipherSuite": "ECDHE-RSA-AES128-SHA",
          "AuthenticationMethod": "AuthHeader",
    },
      "requestID": "07D681279BD94AED",
      "eventID": "f2b287f3-0df1-4961-a2f4-c4bdfed47657",
      "eventType": "AwsApiCall",
      "recipientAccountId": "111122223333"
    }
  ]
}
```


관련 리소스

- AWS CloudTrail User Guide
- CloudTrail 이벤트 참조
- AWS CloudTrail을 사용하여 Amazon S3 요청 식별 (p. 686)

AWS CloudTrail을 사용하여 Amazon S3 요청 식별

Amazon S3를 통해 AWS CloudTrail 이벤트 로그를 사용하여 요청을 식별할 수 있습니다. AWS CloudTrail은 Amazon S3 요청을 식별하는 기본 방법이지만, Amazon S3 서버 액세스 로그를 사용하는 경우 the section called “Amazon S3 액세스 로그를 사용하여 Amazon S3 요청 식별” (p. 720) 단원을 참조하십시오.

주제

- CloudTrail가 Amazon S3에 대한 요청을 캡처하는 방식 (p. 687)
- S3 버킷 및 객체에 대한 CloudTrail 이벤트 로깅 활성화 (p. 687)
- CloudTrail 로그에서 Amazon S3에 대한 요청 식별 (p. 687)
- AWS CloudTrail를 사용하여 Amazon S3 서명 버전 2 요청 식별 (p. 689)
- AWS CloudTrail을 사용하여 Amazon S3 객체에 대한 액세스 식별 (p. 691)

- 관련 리소스 (p. 686)

CloudTrail가 Amazon S3에 대한 요청을 캡처하는 방식

기본적으로 CloudTrail는 최근 90일 동안의 S3 버킷 수준 API 호출을 기록하지만 객체에 대한 로그 요청은 기록하지 않습니다. 버킷 수준 호출에는 `CreateBucket`, `DeleteBucket`, `PutBucketLifeCycle`, `PutBucketPolicy` 등의 이벤트가 포함됩니다. CloudTrail 콘솔에서 버킷 수준 이벤트를 볼 수 있습니다. 그러나 거기에서 데이터 이벤트(Amazon S3 객체 수준 호출)를 볼 수 없으므로 데이터 이벤트에 대한 CloudTrail 로그를 구분 분석하거나 쿼리해야 합니다.

CloudTrail에서 캡처한 Amazon S3 API 호출에 대한 정보는 CloudTrail의 Amazon S3 정보 (p. 679) 단원을 참조하십시오.

S3 버킷 및 객체에 대한 CloudTrail 이벤트 로깅 활성화

CloudTrail 데이터 이벤트를 통해 버킷 및 객체 수준 요청에 대한 정보를 가져올 수 있습니다. 특정 버킷에 대해 CloudTrail 데이터 이벤트를 활성화하려면 Amazon Simple Storage Service 콘솔 사용 설명서에서 AWS CloudTrail 데이터 이벤트가 있는 S3 버킷에 대한 객체 수준 로깅을 사용하는 방법을 참조하십시오.

모든 버킷 또는 특정 버킷 목록에 대해 CloudTrail 데이터 이벤트를 사용하려면 CloudTrail에서 수동으로 추적을 생성해야 합니다.

Note

- CloudTrail의 기본 설정은 관리 이벤트만 찾는 것입니다. 계정에 데이터 이벤트가 활성화되어 있는지 확인하십시오.
- 많은 워크로드를 생성하는 S3 버킷의 경우, 단시간 내에 수천 개의 로그를 신속하게 생성할 수 있습니다. 사용량이 많은 버킷에 대해 얼마나 길게 CloudTrail 데이터 이벤트를 활성화할 것인지, 주의해서 선택해야 합니다.

CloudTrail는 선택한 S3 버킷에 Amazon S3 데이터 이벤트 로그를 저장합니다. 별도의 AWS 계정에서 버킷을 사용하면 소유할 수 있는 여러 버킷의 이벤트를 중앙 위치에서 더 잘 정리할 수 있으므로 쿼리 및 분석이 더 쉽습니다. 별도의 AWS 계정에서 버킷을 사용 AWS Organizations를 사용하면 모니터링하는 버킷을 소유한 계정과 연결된 AWS 계정을 쉽게 만들 수 있습니다. 자세한 내용은 AWS Organizations 사용 설명서의 AWS Organizations란 무엇입니까?를 참조하십시오.

CloudTrail에서 추적을 생성할 때 데이터 이벤트 섹션에서 Select all S3 buckets in your account(계정의 모든 S3 버킷 선택) 확인란을 선택하여 모든 객체 수준 이벤트를 기록할 수 있습니다.

Note

- AWS CloudTrail 데이터 이벤트 버킷에 대해 Amazon S3 수명 주기 정책을 생성하는 것이 가장 좋습니다. 감사해야 한다고 생각되는 기간이 지나면 로그 파일을 주기적으로 제거하도록 수명 주기 정책을 구성합니다. 그러면 Athena에서 쿼리마다 분석하는 데이터의 양이 줄어듭니다.
- 로깅 형식에 대한 자세한 내용은 AWS CloudTrail을 사용하여 Amazon S3 API 호출 기록을 참조하십시오.
- CloudTrail 로그를 쿼리하는 방법의 예는 Analyze Security, Compliance, and Operational Activity Using AWS CloudTrail and Amazon Athena를 참조하십시오.

CloudTrail 로그에서 Amazon S3에 대한 요청 식별

CloudTrail에서 기록한 이벤트는 S3 버킷에 GZip으로 압축된 JSON 객체로 저장됩니다. 요청을 효율적으로 찾으려면 Amazon Athena 같은 서비스를 사용하여 CloudTrail 로그를 인덱싱하고 쿼리해야 합니다. CloudTrail 및 Athena에 대한 자세한 정보는 AWS CloudTrail 로그 쿼리를 참조하십시오.

CloudTrail 로그에 Athena 사용

버킷으로 이벤트를 전달하도록 CloudTrail를 설정한 후 Amazon S3 콘솔에서 객체가 대상 버킷으로 이동하는지 확인해야 합니다. 이들은 다음과 같은 서식이 지정됩니다. *s3://<myawsexamplebucket>/AWSLogs/<111122223333>/CloudTrail/<Region>/<yyyy>/<mm>/<dd>*

Example — Athena를 사용하여 특정 요청에 대한 CloudTrail 이벤트 로그 쿼리

CloudTrail 이벤트 로그를 찾습니다.

```
s3://myawsexamplebucket/AWSLogs/111122223333/CloudTrail/us-east-2/2019/04/14
```

CloudTrail 이벤트 로그를 사용하면 다음과 같이 Athena 데이터베이스와 테이블을 생성하여 쿼리할 수 있습니다.

1. https://console.aws.amazon.com/athena/에서 Athena 콘솔을 엽니다.
2. AWS 리전을 CloudTrail 대상 S3 버킷과 동일하게 변경하십시오.
3. 쿼리 창에서 CloudTrail 이벤트에 대한 Athena 데이터베이스를 생성합니다.

```
CREATE DATABASE s3_cloudtrail_events_db
```

4. 다음 쿼리를 사용하여 버킷의 모든 CloudTrail 이벤트에 대한 테이블을 생성합니다. 버킷 이름을 *<CloudTrail_myawsexamplebucket>*에서 본인의 버킷 이름으로 변경하십시오. 또한 버킷에 사용되는 *AWS_account_ID* CloudTrail를 입력하십시오.

```
CREATE EXTERNAL TABLE s3_cloudtrail_events_db.cloudtrail_myawsexamplebucket_table(
        eventversion STRING,
        useridentity STRUCT<
            type:STRING,
            principalid:STRING,
            arn:STRING,
            accountid:STRING,
            invokedby:STRING,
            accesskeyid:STRING,
            userName:STRING,
            sessioncontext:STRUCT<
                attributes:STRUCT<
                        mfaauthenticated:STRING,
                        creationdate:STRING>,
                        sessionissuer:STRUCT<
                        type:STRING,
                        principalId:STRING,
                        arn:STRING,
                        accountId:STRING,
                        userName:STRING>
                >
             >,
        eventtime STRING,
        eventsource STRING,
        eventname STRING,
        awsregion STRING,
        sourceipaddress STRING,
        useragent STRING,
        errorcode STRING,
        errormessage STRING,
        requestparameters STRING,
        responseelements STRING,
        additionaleventdata STRING,
        requestid STRING,
        eventid STRING,
        resources ARRAY<STRUCT<
```

```
            ARN:STRING,
            accountId:STRING,
            type:STRING>>,
        eventtype STRING,
        apiversion STRING,
        readonly STRING,
        recipientaccountid STRING,
        serviceeventdetails STRING,
        sharedeventid STRING,
        vpcendpointid STRING
    )
    ROW FORMAT SERDE 'com.amazon.emr.hive.serde.CloudTrailSerde'
    STORED AS INPUTFORMAT 'com.amazon.emr.cloudtrail.CloudTrailInputFormat'
    OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
    LOCATION 's3://<myawsexamplebucket>/AWSLogs/<111122223333>/';
```

5. 쿼리가 작동하는지 확인하려면 Athena를 테스트하십시오.

```
SELECT * FROM s3_cloudtrail_events_db.cloudtrail_myawsexamplebucket_table
WHERE eventsource='s3.amazonaws.com'
LIMIT 2;
```

AWS CloudTrail를 사용하여 Amazon S3 서명 버전 2 요청 식별

Amazon S3를 통해 AWS CloudTrail 이벤트 로그를 사용하여 요청에 서명하는 데 사용된 API 서명 버전을 식별할 수 있습니다. 서명 버전 2에 대한 지원이 중단(사용되지 않음)될 예정이므로 이 기능은 중요합니다. 그 이후 Amazon S3는 더 이상 서명 버전 2를 사용하는 요청을 수락하지 않으며 모든 요청은 서명 버전 4 서명을 사용해야 합니다.

CloudTrail를 사용하여 워크플로에서 서명 버전 2 서명을 사용하고 있는지 확인하는 것이 좋습니다. 비즈니스에 영향을 미치지 않도록, 라이브러리와 코드가 서명 버전 4를 사용하도록 업그레이드하여 문제를 해결하십시오.

자세한 내용은 AWS 토론 포럼의 공지: Amazon S3용 AWS CloudTrail, 향상된 보안 감사를 위해 새로운 필드 추가를 참조하십시오.

> **Note**
>
> Amazon S3에 대한 CloudTrail 이벤트에는 ‘`additionalEventData`’ 키 이름 아래에 요청 세부 정보의 서명 버전이 포함되어 있습니다. GET, PUT 및 DELETE 등 Amazon S3의 객체에 대한 요청의 서명 버전은 기본적으로 비활성화되어 있으므로, 검색하려면 CloudTrail 데이터 이벤트를 활성화해야 합니다.

AWS CloudTrail은 서명 버전 2 요청을 식별하는 기본 방법이며, Amazon S3 서버 액세스 로그를 사용하는 경우 Amazon S3 액세스 로그를 사용하여 서명 버전 2 요청 식별 (p. 724) 단원을 참조하십시오.

Amazon S3 서명 버전 2 요청을 식별하기 위한 Athena 쿼리 예제

Example — 서명 버전 2인 모든 이벤트를 선택하고 EventTime, S3 Action, Request_Parameters, Region, SourceIP, UserAgent만 인쇄

다음 Athena 쿼리에서 *<s3_cloudtrail_events_db.cloudtrail_myawsexamplebucket_table>*을 Athena 세부 정보로 대체하고 필요에 따라 제한을 늘리거나 제거하십시오.

```
```

```
SELECT EventTime, EventName as S3_Action, requestParameters as Request_Parameters,
 awsregion as AWS_Region, sourceipaddress as Source_IP, useragent as User_Agent
FROM s3_cloudtrail_events_db.cloudtrail_myawsexamplebucket_table
WHERE eventsource='s3.amazonaws.com'
AND json_extract_scalar(additionalEventData, '$.SignatureVersion')='SigV2'
LIMIT 10;
```

Example — 서명 버전 2 트래픽을 보내는 모든 요청자 선택

```
SELECT useridentity.arn, Count(requestid) as RequestCount
FROM s3_cloudtrail_events_db.cloudtrail_myawsexamplebucket_table
WHERE eventsource='s3.amazonaws.com'
    and json_extract_scalar(additionalEventData, '$.SignatureVersion')='SigV2'
Group by useridentity.arn
```

파티셔닝 서명 버전 2 데이터

쿼리해야 하는 데이터가 많은 경우, 분할된 테이블을 작성하여 Athena의 비용과 런타임을 줄일 수 있습니다.

이를 위해 다음과 같이 파티션이 있는 새 테이블을 만드십시오.

```
   CREATE EXTERNAL TABLE
 s3_cloudtrail_events_db.cloudtrail_myawsexamplebucket_table_partitioned(
        eventversion STRING,
        userIdentity STRUCT<
            type:STRING,
            principalid:STRING,
            arn:STRING,
            accountid:STRING,
            invokedby:STRING,
            accesskeyid:STRING,
            userName:STRING,
         sessioncontext:STRUCT<
                    attributes:STRUCT<
                    mfaauthenticated:STRING,
                    creationdate:STRING>,
                    sessionIssuer:STRUCT<
                    type:STRING,
                    principalId:STRING,
                    arn:STRING,
                    accountId:STRING,
                    userName:STRING>
                >
             >,
        eventTime STRING,
        eventSource STRING,
        eventName STRING,
        awsRegion STRING,
        sourceIpAddress STRING,
        userAgent STRING,
        errorCode STRING,
        errorMessage STRING,
        requestParameters STRING,
        responseElements STRING,
        additionalEventData STRING,
        requestId STRING,
        eventId STRING,
        resources ARRAY<STRUCT<ARN:STRING,accountId: STRING,type:STRING>>,
```

```
        eventType STRING,
        apiVersion STRING,
        readOnly STRING,
        recipientAccountId STRING,
        serviceEventDetails STRING,
        sharedEventID STRING,
        vpcEndpointId STRING
    )
    PARTITIONED BY (region string, year string, month string, day string)
    ROW FORMAT SERDE 'com.amazon.emr.hive.serde.CloudTrailSerde'
    STORED AS INPUTFORMAT 'com.amazon.emr.cloudtrail.CloudTrailInputFormat'
    OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
    LOCATION 's3://myawsexamplebucket/AWSLogs/111122223333/';
```

그런 다음 개별적으로 파티션을 만듭니다. 작성하지 않은 날짜의 결과는 가져올 수 없습니다.

```
ALTER TABLE s3_cloudtrail_events_db.cloudtrail_myawsexamplebucket_table_partitioned ADD
    PARTITION (region= 'us-east-1', year= '2019', month= '02', day= '19') LOCATION 's3://
myawsexamplebucket/AWSLogs/111122223333/CloudTrail/us-east-1/2019/02/19/'
    PARTITION (region= 'us-west-1', year= '2019', month= '02', day= '19') LOCATION 's3://
myawsexamplebucket/AWSLogs/111122223333/CloudTrail/us-west-1/2019/02/19/'
    PARTITION (region= 'us-west-2', year= '2019', month= '02', day= '19') LOCATION 's3://
myawsexamplebucket/AWSLogs/111122223333/CloudTrail/us-west-2/2019/02/19/'
    PARTITION (region= 'ap-southeast-1', year= '2019', month= '02', day= '19') LOCATION
 's3://myawsexamplebucket/AWSLogs/111122223333/CloudTrail/ap-southeast-1/2019/02/19/'
    PARTITION (region= 'ap-southeast-2', year= '2019', month= '02', day= '19') LOCATION
 's3://myawsexamplebucket/AWSLogs/111122223333/CloudTrail/ap-southeast-2/2019/02/19/'
    PARTITION (region= 'ap-northeast-1', year= '2019', month= '02', day= '19') LOCATION
 's3://myawsexamplebucket/AWSLogs/111122223333/CloudTrail/ap-northeast-1/2019/02/19/'
    PARTITION (region= 'eu-west-1', year= '2019', month= '02', day= '19') LOCATION 's3://
myawsexamplebucket/AWSLogs/111122223333/CloudTrail/eu-west-1/2019/02/19/'
    PARTITION (region= 'sa-east-1', year= '2019', month= '02', day= '19') LOCATION 's3://
myawsexamplebucket/AWSLogs/111122223333/CloudTrail/sa-east-1/2019/02/19/';
```

그런 다음 이 파티션을 기반으로 요청할 수 있으며 전체 버킷을 로드할 필요가 없습니다.

```
SELECT useridentity.arn,
Count(requestid) AS RequestCount
FROM s3_cloudtrail_events_db.cloudtrail_myawsexamplebucket_table_partitioned
WHERE eventsource='s3.amazonaws.com'
AND json_extract_scalar(additionalEventData, '$.SignatureVersion')='SigV2'
AND region='us-east-1'
AND year='2019'
AND month='02'
AND day='19'
Group by useridentity.arn
```

AWS CloudTrail을 사용하여 Amazon S3 객체에 대한 액세스 식별

AWS CloudTrail 이벤트 로그를 사용하여 GetObject, DeleteObject, PutObject 등의 데이터 이벤트에 대한 Amazon S3 객체 액세스 요청을 식별하고, 그러한 요청에 대한 추가 정보를 검색할 수 있습니다.

다음 예제는 AWS CloudTrail 이벤트 로그에서 Amazon S3에 대한 모든 PUT 객체 요청을 가져오는 방법을 보여 줍니다.

Amazon S3 객체 액세스 요청을 식별하기 위한 Athena 쿼리 예제

다음 Athena 쿼리 예제에서 *<s3_cloudtrail_events_db.cloudtrail_myawsexamplebucket_table>*을 Athena 세부 정보로 대체하고 필요에 따라 날짜 범위를 수정하십시오.

Example — PUT 객체 액세스 요청이 있는 모든 이벤트를 선택하고, EventTime, EventSource, SourceIP, UserAgent, BucketName, Object, UserARN만 인쇄

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  requestParameters.bucketName as bucketName,
  requestParameters.key as object,
  userIdentity.arn as userArn
FROM
  s3_cloudtrail_events_db.cloudtrail_myawsexamplebucket_table
WHERE
  eventName = ‘PutObject'
  AND eventTime BETWEEN "2019-07-05T00:00:00Z" and "2019-07-06T00:00:00Z"
```

Example — GET 객체 액세스 요청이 있는 모든 이벤트를 선택하고, EventTime, EventSource, SourceIP, UserAgent, BucketName, Object, UserARN만 인쇄

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  requestParameters.bucketName as bucketName,
  requestParameters.key as object,
  userIdentity.arn as userArn
FROM
  s3_cloudtrail_events_db.cloudtrail_myawsexamplebucket_table
WHERE
  eventName = ‘GetObject'
  AND eventTime BETWEEN "2019-07-05T00:00:00Z" and "2019-07-06T00:00:00Z"
```

Example — 특정 기간에 버킷에 대한 모든 익명 요청자 이벤트를 선택하고, EventTime, EventSource, SourceIP, UserAgent, BucketName, UserIdentity, UserARN만 인쇄

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  requestParameters.bucketName as bucketName,
  userIdentity.arn as userArn,
  userIdentity.principalId
FROM
  s3_cloudtrail_events_db.cloudtrail_myawsexamplebucket_table
WHERE
  userIdentity.principalId='ANONYMOUS_PRINCIPAL'
```

```
AND eventTime BETWEEN "2019-07-05T00:00:00Z" and "2019-07-06T00:00:00Z"
```

Note

- 이러한 쿼리 예제는 보안 모니터링에도 유용할 수 있습니다. 예상치 못하거나 승인되지 않은 IP 주소/요청자의 `PutObject` 또는 `GetObject` 호출 결과를 검토하고 버킷에 대한 익명 요청을 식별할 수 있습니다.
- 이 쿼리는 로깅이 활성화된 시간부터의 정보만 검색합니다.

Amazon S3 서버 액세스 로그를 사용하는 경우 Amazon S3 액세스 로그를 사용하여 객체 액세스 요청 식별 (p. 725) 단원을 참조하십시오.

관련 리소스

- AWS CloudTrail User Guide
- CloudTrail 이벤트 참조

Amazon S3에서 BitTorrent 사용

주제

- BitTorrent 전송에 대한 요금이 부과되는 방법 (p. 694)
- BitTorrent를 사용하여 Amazon S3에 저장된 객체 검색 (p. 695)
- Amazon S3 및 BitTorrent를 사용하여 콘텐츠 게시 (p. 695)

BitTorrent는 파일을 배포하기 위한 공개 피어 투 피어 프로토콜입니다. BitTorrent 프로토콜을 사용하여 Amazon S3에서 공개적으로 액세스 가능한 객체를 모두 검색할 수 있습니다. 이 단원에서는 Amazon S3로부터 데이터를 배포하기 위해 BitTorrent를 사용하는 이유와 그 방법에 대해 설명합니다.

Amazon S3는 개발자가 대량으로 콘텐츠를 배포할 때 비용을 절약할 수 있도록 BitTorrent 프로토콜을 지원합니다. Amazon S3는 어떤 데이터도 간편하고 안정적으로 저장할 수 있는 유용한 스토리지입니다. Amazon S3 데이터의 기본 배포 메커니즘은 클라이언트/서버 다운로드입니다. 클라이언트/서버 배포에서는 전체 객체가 Amazon S3에서 해당 객체를 요청한 모든 권한이 있는 사용자까지 지점 간 전송됩니다. 클라이언트/서버 전송이 광범위한 사용 사례에서 적절하지만 모든 사용자에게 최적은 아닙니다. 특히, 객체를 다운로드하는 사용자 수가 증가할수록 클라이언트/서버 배포 비용도 비례하여 증가합니다. 그러므로 인기 있는 객체를 배포할 경우 많은 비용이 발생할 수 있습니다.

BitTorrent는 객체를 다운로드하는 클라이언트 자체를 배포자로 활용함으로써 이 문제를 해결합니다. 각 클라이언트는 Amazon S3로부터 객체의 일부 조각을 다운로드하고 다른 클라이언트로부터 일부 조각을 다운로드하며, 이와 동시에 다른 관심 있는 "피어"에게 동일한 객체의 조각을 업로드합니다. 게시자 입장에서는, 대용량의 인기 있는 파일의 경우 Amazon S3에서 실제로 제공하는 데이터의 양이 클라이언트/서버 다운로드를 통해 동일한 클라이언트에게 데이터를 제공할 때보다 현저히 감소한다는 것이 이점입니다. 데이터 전송량의 감소는 객체 게시자의 비용 절감을 의미합니다.

Note

- Amazon S3는 2016년 5월 30일 이후 시작된 AWS 리전에서 BitTorrent 프로토콜을 지원하지 않습니다.
- 크기가 5GB 미만인 객체에 대한 Torrent 파일만 가져올 수 있습니다.

BitTorrent 전송에 대한 요금이 부과되는 방법

Amazon S3와 함께 BitTorrent를 사용할 때는 별도의 요금이 부과되지 않습니다. BitTorrent 프로토콜을 통한 데이터 전송은 클라이언트/서버 전송과 같은 요금으로 측정됩니다. 정확하게 말하자면, 다운로드하는 BitTorrent 클라이언트가 Amazon S3 "시더"에서 객체 "조각"을 요청할 때마다 REST 또는 SOAP 프로토콜을 사용하여 해당 조각에 대한 익명 요청이 실행된 것처럼 요금이 발생합니다. 이러한 요금은 Amazon S3 청구서 및 사용 보고서에 동일한 방식으로 나타납니다. 많은 클라이언트에서 BitTorrent를 통해 동시에 같은 객체를 요청하는 경우 해당 클라이언트를 만족시키기 위해 Amazon S3에서 처리해야 하는 데이터의 양이 클라이언트/서버 전송 때보다 적다는 점이 다릅니다. 이는 BitTorrent 클라이언트 자체에서 동시에 업로드 및 다운로드 작업이 이루어지기 때문입니다.

Note

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

BitTorrent를 사용하여 얻을 수 있는 데이터 전송량 감소 이득은 객체가 얼마나 많이 사용되는지에 따라 크게 달라집니다. 덜 사용되는 객체의 경우 클라이언트를 처리하기 위해 "시더"를 더 많이 사용해야 하므로 이러한 객체에서 BitTorrent 배포 비용과 클라이언트/서버 배포 비용의 차이는 그리 크지 않을 수 있습니다. 특히, 한

클라이언트에서만 한 번에 특정 객체를 다운로드하는 경우에는 BitTorrent 전송 비용이 직접 다운로드와 같습니다.

BitTorrent를 사용하여 Amazon S3에 저장된 객체 검색

익명으로 읽을 수 있는 Amazon S3의 모든 객체는 BitTorrent를 통해 다운로드할 수 있습니다. 이렇게 하려면 BitTorrent 클라이언트 애플리케이션이 필요합니다. Amazon에서는 BitTorrent 클라이언트 애플리케이션을 배포하지 않지만 무료로 사용할 수 있는 클라이언트가 많이 있습니다. Amazon S3BitTorrent 구현은 공식 BitTorrent 클라이언트를 사용한 작업에 대해 테스트되었습니다(http://www.bittorrent.com/ 참조).

BitTorrent 다운로드의 시작점은 .torrent 파일입니다. 이 작은 파일이 모든 BitTorrent 클라이언트에게 다운로드할 데이터와 그러한 데이터를 검색할 위치를 알려 줍니다. .torrent 파일은 다운로드할 실제 객체를 구성하는 작은 조각입니다. BitTorrent 클라이언트 애플리케이션에 Amazon S3에서 생성한 .torrent 파일을 제공하면 즉시 Amazon S3 및 "피어" BitTorrent 클라이언트로부터 다운로드를 시작합니다.

공개적으로 사용할 수 있는 객체에 대한 .torrent 파일을 검색하는 방법은 간단합니다. 객체에 대한 REST GET 요청의 끝에 "?torrent" 쿼리 문자열 파라미터를 추가하면 됩니다. 인증은 필요하지 않습니다. BitTorrent 클라이언트를 설치한 후 BitTorrent 다운로드를 사용하여 객체를 다운로드하는 것은 웹 브라우저에서 URL을 여는 것 만큼 간단합니다.

SOAP API를 사용하여 Amazon S3 객체에 대한 .torrent를 가져오는 방법은 없습니다.

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

Example

이 예제는 "quotes" 버킷의 "Nelson" 객체에 대한 Torrent 파일을 검색합니다.

Sample Request

```
GET /quotes/Nelson?torrent HTTP/1.0
Date: Wed, 25 Nov 2009 12:00:00 GMT
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-request-id: 7CD745EBB7AB5ED9
Date: Wed, 25 Nov 2009 12:00:00 GMT
Content-Disposition: attachment; filename=Nelson.torrent;
Content-Type: application/x-bittorrent
Content-Length: 537
Server: AmazonS3

<body: a Bencoded dictionary as defined by the BitTorrent specification>
```

Amazon S3 및 BitTorrent를 사용하여 콘텐츠 게시

Amazon S3에 저장된 익명으로 읽을 수 있는 모든 객체를 BitTorrent를 사용하여 자동으로 다운로드할 수 있습니다. 익명 `READ` 작업을 허용하기 위해 객체에 대한 ACL을 변경하는 프로세스는 Amazon S3의 Identity and Access Management (p. 299)에 설명되어 있습니다.

.torrent 파일을 바로 주거나 객체의 ?torrent URL에 대한 링크를 게시하여 클라이언트를 BitTorrent 액세스 가능 객체로 보낼 수 있습니다. Amazon S3 객체를 설명하는 .torrent 파일은 REST ?torrent 리소스를 통해 처음 요청될 때 온디맨드 방식으로 생성된다는 점에 유의해야 합니다. 객체에 대한 .torrent를 생성하는 데에는 시간이 걸리며, 이는 객체 크기에 비례합니다. 대형 객체의 경우 이 시간은 상당할 수 있습니다. 따라서 ?torrent 링크를 게시하기 전에 스스로 첫 번째 요청을 하는 것이 좋습니다. Amazon S3가 .torrent 파일을 생성하여 이 첫 번째 요청에 응답하는 데 몇 분 정도 걸릴 수 있습니다. 해당 객체를 업데이트하지 않는 한 .torrent에 대한 후속 요청은 빨라집니다. ?torrent 링크를 배포하기 전에 이 절차를 따르면 고객의 BitTorrent 다운로드 경험이 원활해집니다.

BitTorrent를 사용한 파일 배포를 중지하려면 해당 파일에 대한 익명 액세스를 제거하기만 하면 됩니다. 이는 Amazon S3에서 파일을 삭제하거나 익명 읽기가 금지되도록 액세스 제어 정책을 수정하여 수행할 수 있습니다. 이렇게 한 후에는 Amazon S3가 BitTorrent 네트워크에서 더 이상 해당 파일에 대한 "시더" 역할을 하지 않으며, 더 이상 ?torrent REST API를 통해 .torrent 파일을 처리하지 않습니다. 그러나 파일에 대한 .torrent가 게시된 후에는 BitTorrent 피어 투 피어 네트워크만 사용하여 발생하는 해당 객체에 대한 공개 다운로드가 이 작업으로 중지되지 않을 수 있습니다.

REST 및 SOAP 오류 처리

주제

- REST 오류 응답 (p. 697)
- SOAP 오류 응답 (p. 698)
- Amazon S3 오류 모범 사례 (p. 699)

이 단원에서는 REST 및 SOAP 오류와, 그러한 오류를 처리하는 방법을 설명합니다.

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

REST 오류 응답

주제

- 응답 헤더 (p. 697)
- 오류 응답 (p. 698)

REST 요청에 오류가 발생할 경우 HTTP 응답에는 다음이 포함됩니다.

- XML 오류 문서 - 응답의 본문
- 콘텐츠 유형: application/xml
- 적합한 3xx, 4xx 또는 5xx HTTP 상태 코드

다음은 REST 오류 응답의 예입니다.

```
<?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>
```

Amazon S3 오류에 대한 자세한 내용은 ErrorCodeList를 참조하십시오.

응답 헤더

다음은 모든 작업에서 반환되는 응답 헤더입니다.

- `x-amz-request-id:` 시스템에서 각 요청에 할당한 고유 ID. Amazon S3에 문제가 있을 경우 Amazon은 이를 활용하여 문제를 해결합니다.
- `x-amz-id-2:` 문제 해결에 도움이 되는 특별 토큰.

오류 응답

주제

- 오류 코드 (p. 698)
- 오류 메시지 (p. 698)
- 추가 세부 정보 (p. 698)

Amazon S3 요청에 오류가 발생할 경우 클라이언트는 오류 응답을 받습니다. 오류 응답의 정확한 형식은 API마다 다릅니다. 예를 들어, REST 오류 응답은 SOAP 오류 응답과 다릅니다. 하지만 모든 오류 응답에는 공통 요소가 있습니다.

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

오류 코드

오류 코드는 오류 상태를 식별하는 고유한 문자열입니다. 이 코드는 오류를 감지하여 유형별로 처리하는 프로그램에서 판독하고 이해하기 위한 것입니다. 대부분의 오류 코드는 SOAP와 REST API에서 공통적이지만 일부는 API마다 다릅니다. 예를 들어, NoSuchKey는 공통적이지만, UnexpectedContent는 잘못된 REST 요청에 대한 응답에서만 발생할 수 있습니다. 모든 경우에 SOAP 고장 코드는 오류 코드 표에 표시된 접두사를 포함하므로, 실제로 NoSuchKey 오류는 SOAP에서 Client.NoSuchKey로 반환됩니다.

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

오류 메시지

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 직접 메시지를 표시합니다. 좀 더 포괄적인 오류 처리 기능을 갖고 있고 적절히 국제화된 고급 프로그램은 오류 메시지를 무시할 가능성이 큽니다.

추가 세부 정보

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.

SOAP 오류 응답

> **Note**
>
> HTTP를 통한 SOAP 지원은 중단되었지만 HTTPS를 통해 계속해서 사용할 수 있습니다. 새로운 Amazon S3 기능은 SOAP에 대해 지원되지 않습니다. 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.

Example

```
<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>
```

Amazon S3 오류에 대한 자세한 내용은 ErrorCodeList를 참조하십시오.

Amazon S3 오류 모범 사례

Amazon S3와 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 S3가 InternalError 응답을 반환하는 경우, 요청을 재시도하십시오.

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 정보를 알고 싶다거나, SlowDown 오류가 자주 발생하거나 예기치 않은 경우에 발생할 경우 Amazon S3 개발자 포럼 https://forums.aws.amazon.com/에 글을 게시하거나 AWS Premium Supporthttps://aws.amazon.com/premiumsupport/에 가입하십시오.

오류 격리

Note

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

Amazon S3는 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 사용하십시오. 또한 애플리케이션을 디버깅할 경우, 사람이 읽을 수 있는 XML 오류 응답의 <Details> 요소를 참조해야 합니다.

Amazon S3 문제 해결

이 단원에서는 Amazon S3 문제를 해결하는 방법과 AWS Support에 연락할 때 사용해야 하는 요청 ID를 받는 방법에 대해 설명합니다.

주제

- 증상별 Amazon S3 문제 해결 (p. 701)
- AWS Support용 Amazon S3 요청 ID 가져오기 (p. 702)
- 관련 주제 (p. 704)

증상별 Amazon S3 문제 해결

다음 주제는 Amazon S3 작업 시 발생할 수 있는 문제 중 일부를 해결하는 데 도움이 되는 증상을 나열한 것입니다.

증상

- 버전 관리를 사용하는 버킷에 대한 Amazon S3 요청에 대해 HTTP 503 응답이 크게 증가 (p. 701)
- CORS로 버킷 세트에 액세스할 때 예상하지 못한 동작 (p. 701)

버전 관리를 사용하는 버킷에 대한 Amazon S3 요청에 대해 HTTP 503 응답이 크게 증가

버전 관리를 사용하는 버킷에 대한 Amazon S3 PUT 또는 DELETE 객체 요청에서 받은 HTTP 503 속도 저하 응답의 횟수가 크게 증가한다면 버전이 무수히 많은 객체가 하나 이상 버킷에 있을 수 있습니다. 객체에 버전이 무수히 많을 경우 Amazon S3가 자동으로 버킷에 대한 요청을 제한하여 과도한 요청 트래픽으로부터 고객을 보호합니다. 과도한 트래픽은 동일한 버킷에 다른 요청이 만들어지는 것을 지연시킬 수 있습니다.

무수히 많은 버전이 있는 S3 객체를 파악하려면 Amazon S3 인벤토리 도구를 사용합니다. 인벤토리 도구는 버킷에 있는 객체의 플랫 파일 목록을 제공하는 보고서를 생성합니다. 자세한 내용은 Amazon S3 인벤토리 (p. 459) 단원을 참조하십시오.

Amazon S3 팀은 같은 S3 객체를 반복적으로 덮어써서 해당 객체에 수백만 개 버전이 생성될 가능성이 있는 응용 프로그램을 조사할 것을 권장합니다. 조사를 통해 응용 프로그램이 의도한 대로 작동하는지 판단할 수 있습니다. 하나 이상의 S3 객체에 수백만 개 버전이 필요한 사용 사례가 있는 경우, AWS 지원 팀(AWS Support)에 연락하여 사용 사례를 논의하고 귀사의 사용 사례 시나리오에 맞는 최적의 솔루션을 찾는 데 필요한 도움을 받으십시오.

이 문제를 방지하려면 다음 모범 사례를 고려하십시오.

- 수명 주기 관리 "NonCurrentVersion" 만료 정책 및 "ExpiredObjectDeleteMarker" 정책을 활성화하여 버킷에 연결된 데이터 객체 없이 이전 버전의 객체 및 삭제 마커를 만료합니다.
- 디렉터리 구조를 가능한 단조롭게 유지하고 각 디렉터리 이름을 고유하게 만듭니다.

CORS로 버킷 세트에 액세스할 때 예상하지 못한 동작

CORS(cross-origin 리소스 공유) 구성으로 버킷 세트에 액세스할 때 예상하지 못한 동작이 발생할 경우 CORS 문제 해결 (p. 163)을 참조하십시오.

AWS Support용 Amazon S3 요청 ID 가져오기

Amazon S3에 오류나 예기치 않은 동작이 발생하여 AWS Support에 문의할 때는 실패한 작업과 연결된 요청 ID를 알아야 합니다. 이 요청 ID를 알면 AWS Support에서 해당 문제의 해결을 지원할 수 있습니다. 요청 ID는 쌍으로 되어 있고 Amazon S3에서 처리하는 모든 응답(잘못된 응답이라도)에 반환되며 상세 정보 로그를 통해 액세스할 수 있습니다. S3 액세스 로그 및 CloudTrail 이벤트/데이터 이벤트를 포함하여 요청 ID을 가져오는 일반적인 방법은 여러 가지가 있습니다.

이 로그를 복구한 후에는 AWS Support에 문의할 때 필요하므로 두 값을 복사하여 보관하십시오. AWS Support 문의에 대한 내용은 문의처에서 확인하십시오.

주제

- HTTP를 이용한 요청 ID 가져오기 (p. 702)
- 웹 브라우저를 이용한 요청 ID 가져오기 (p. 702)
- AWS SDK를 이용한 요청 ID 가져오기 (p. 702)
- AWS CLI를 이용한 요청 ID 가져오기 (p. 704)

HTTP를 이용한 요청 ID 가져오기

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

HTTP 요청의 경우 요청 ID 쌍은 다음 예와 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 볼 수 있게 해주는 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 개발자 도구를 사용하여 HTTP 응답 헤더를 봐야 합니다. 특정 브라우저의 개발자 도구에 대한 자세한 정보는 AWS 개발자 포럼에서 Amazon S3 문제 해결 - S3 요청 ID 복구 방법을 참조하십시오.

AWS SDK를 이용한 요청 ID 가져오기

다음 단원에는 AWS SDK를 사용한 로깅 구성에 대한 정보가 있습니다. 모든 요청과 응답에서 로깅을 상세 정보 로깅을 활성화할 수 있지만 대량 요청/응답은 애플리케이션 속도를 크게 떨어뜨릴 수 있으므로 프로덕션 시스템에서는 로깅을 활성화하지 마십시오.

AWS SDK 요청의 경우 요청 ID 쌍은 다음 예와 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
```

PHP용 SDK를 이용한 요청 ID 가져오기

PHP를 사용하여 로깅을 구성할 수 있습니다. 자세한 내용은 PHP용 AWS SDK FAQ에서 네트워크를 통해 전송되는 데이터를 확인하는 방법을 참조하십시오.

Java용 SDK를 이용한 요청 ID 가져오기

특정 요청이나 응답에 대한 로깅을 활성화하여 관련 헤더만 포착하고 반환할 수 있습니다. 그러려면 `com.amazonaws.services.s3.s3ResponseMetadata` 클래스를 가져오십시오. 그런 다음 실제 요청을 수행하기 전에 변수에 요청을 저장할 수 있습니다. `getCachedResponseMetadata(AmazonWebServiceRequest request).getRequestID()`를 호출하여 로깅된 요청이나 응답을 가져옵니다.

Example

```
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 정보는 AWS SDK for Java Developer Guide의 Logging AWS SDK for Java Calls 단원에서 Verbose Wire Logging 부분을 참조하십시오.

.NET용 AWS SDK를 이용한 요청 ID 가져오기

기본 제공된 `System.Diagnostics` 로깅 도구를 사용하여 .NET용 AWS SDK에서 로깅을 구성할 수 있습니다. 자세한 내용은 Logging with the AWS SDK for .NET AWS 개발자 블로그 게시물을 참조하십시오.

> **Note**
>
> 반환되는 정보에는 기본적으로 오류 정보만 포함됩니다. 구성 파일에는 `AWSLogMetrics`(및 선택적으로 `AWSResponseLogging`)가 포함되어 있어야 요청 ID를 가져올 수 있습니다.

SDK for Python를 이용한 요청 ID 가져오기

디버그 정보를 파일에 출력하기 위해 코드에 다음 줄을 추가하여 Python으로 로깅을 구성할 수 있습니다.

```
import logging
logging.basicConfig(filename="mylog.log", level=logging.DEBUG)
```

AWS용 Boto Python 인터페이스를 사용하는 경우 Boto 문서(여기)에 따라 디버그 수준을 2로 설정할 수 있습니다.

Ruby용 SDK를 이용한 요청 ID 가져오기

Ruby용 SDK 버전 1, 버전 2 또는 버전 3을 사용하여 요청 ID를 가져올 수 있습니다.

- Ruby용 SDK 버전 1 사용 – 다음 코드 줄을 사용하여 HTTP 유선 로깅을 전역적으로 활성화할 수 있습니다.

```
s3 = AWS::S3.new(:logger => Logger.new($stdout), :http_wire_trace => true)
```

- Ruby용 SDK 버전 2 또는 버전 3 사용 – 다음 코드 줄을 사용하여 HTTP 유선 로깅을 전역적으로 활성화할 수 있습니다.

```
s3 = Aws::S3::Client.new(:logger => Logger.new($stdout), :http_wire_trace => true)
```

AWS CLI를 이용한 요청 ID 가져오기

명령에 `--debug`를 추가하여 AWS CLI에서 요청 ID를 가져올 수 있습니다.

관련 주제

그 밖의 문제 해결 및 지원 항목은 다음을 참조하십시오.

- CORS 문제 해결 (p. 163)
- REST 및 SOAP 오류 처리 (p. 697)
- AWS Support 설명서

타사 도구와 관련된 문제 해결 정보는 AWS 개발자 포럼의 Amazon S3 요청 ID 가져오기를 참조하십시오.

Amazon S3 서버 액세스 로깅

서버 액세스 로깅은 버킷에 대해 이루어진 요청에 따른 상세 레코드를 제공합니다. 서버 액세스 로그는 많은 애플리케이션에 있어 유용합니다. 예를 들어 액세스 로그 정보는 보안 및 액세스 감사에 유용할 수 있습니다. 또한 고객층을 파악하고 Amazon S3 결제 요금을 확인할 수 있습니다.

> **Note**
>
> 서버 액세스 로그는 2019년 3월 20일 이후에 시작된 리전의 잘못된 리전 리디렉션 오류에 대한 정보를 기록하지 않습니다. 버킷이 존재하는 리전의 엔드포인트가 아닌 엔드포인트에 대해 객체/ 버킷 요청이 생성되면 잘못된 리전 리디렉션 오류가 발생합니다.

주제

- 서버 액세스 로깅 활성화 방법 (p. 705)
- 로그 객체 키 형식 (p. 707)
- 로그 전송 방법 (p. 707)
- 서버 로그 전송은 보장되지 않음 (p. 707)
- 버킷 로깅 상태 변경 시 일정 기간에 걸쳐 단계적으로 반영됨 (p. 708)
- 콘솔을 이용하여 로깅 활성화 (p. 708)
- 프로그래밍 방식으로 로깅 활성화 (p. 708)
- Amazon S3 서버 액세스 로그 형식 (p. 711)
- Amazon S3 로그 파일 삭제 (p. 719)
- Amazon S3 액세스 로그를 사용하여 요청 식별 (p. 720)

서버 액세스 로깅 활성화 방법

버킷 액세스 요청을 추적하기 위해 서버 액세스 로깅을 활성화할 수 있습니다. 각 액세스 로그 레코드는 한 액세스 요청에 대한 세부 정보(요청자, 버킷 이름, 요청 시간, 요청 작업, 응답 상태, 오류 코드(해당되는 경우) 등)를 제공합니다.

> **Note**
>
> Amazon S3 버킷에서 서버 액세스 로깅을 활성화하는 데 대한 추가 비용은 없으며, 로그가 버킷에 PUT이 실행되는 경우에도 요금이 부과되지 않습니다. 단, 시스템이 버킷에 전달하는 로그 파일에 대해서는 일반적인 스토리지 요금이 발생합니다. 이러한 로그 파일은 언제든지 삭제할 수 있습니다. 이러한 로그 파일에 대한 후속 읽기 및 기타 요청은 데이터 전송 요금을 포함하여 다른 모든 객체와 마찬가지로 정상적으로 청구됩니다.

기본적으로 로깅은 비활성화되어 있으며, 로깅이 활성화된 경우 소스 버킷과 동일한 AWS 리전에 있는 버킷에 로그가 저장됩니다.

액세스 로깅을 활성화하려면 다음과 같은 작업을 수행해야 합니다.

- Amazon S3에서 액세스 로그를 전달하려는 버킷에 로깅 구성을 추가하여 로그 전달을 활성화합니다. 이 버킷을 원본 버킷이라고 합니다.

- Amazon S3 Log Delivery 그룹에 액세스 로그를 저장할 버킷에 대한 쓰기 권한을 부여합니다. 이 버킷을 대상 버킷이라고 합니다.

  Note

  - Amazon S3는 버킷 정책이 아닌 버킷 ACL을 통해 액세스 로그를 전송할 권한 부여만 지원합니다.
  - 버킷 정책에 deny 조건을 추가하면 Amazon S3가 액세스 로그를 전송하지 못할 수 있습니다.
  - 대상 버킷의 기본 버킷 암호화는 AES256(SSE-S3)이 선택된 경우에만 사용할 수 있습니다. SSE-KMS 암호화는 지원되지 않습니다.
  - Amazon S3 객체 잠금은 로그 대상 버킷에서 활성화할 수 없습니다.

로그가 전송되도록 하기 위해 다음 로깅 구성 정보를 제공합니다.

- Amazon S3가 액세스 로그를 객체로 저장할 대상 버킷의 이름입니다. 소스 버킷 자체를 포함하여 소스 버킷과 동일한 리전에 있는 자신의 고유 버킷에 로그를 전달할 수 있습니다.

  손쉽게 로그를 관리하기 위해서는 소스와 다른 버킷에 액세스 로그를 저장하는 것이 좋습니다. 소스 버킷에 액세스 로그를 저장해야 한다면 객체 이름이 공통 문자열로 시작하고 로그 객체를 더 쉽게 식별할 수 있도록 모든 로그 객체 키에 접두사를 지정하는 것이 좋습니다.

  소스 버킷과 대상 버킷이 동일한 버킷이면 버킷에 작성되는 로그에 대해 추가 로그가 생성됩니다. 이러한 방식은 스토리지 결제 요금이 약간 증가할 수 있으므로 사용 사례 면에서 이상적이지 않을 수 있습니다. 또한 로그에 대한 추가 로그로 인해 원하는 로그를 찾기가 힘들어질 수 있습니다.

  키 접두사는 여러 버킷이 동일한 대상 버킷에 로깅할 때 소스 버킷을 서로 구별하는 데 유용합니다.

  Note

  소스 및 대상 버킷 모두 동일한 AWS 계정에서 소유하고 있어야 하며 해당 버킷 모두 동일한 리전에 있어야 합니다.
- (선택 사항) Amazon S3에서 모든 로그 객체 키에 지정한 접두사. 접두사를 사용하면 더 쉽게 로그 객체를 구분할 수 있습니다.

  예를 들어, 접두사 값을 `logs/`로 지정할 경우 Amazon S3가 만드는 각 로그 객체의 키는 다음 예와 같이 `logs/` 접두사로 시작합니다.

  ```
  logs/2013-11-01-21-32-16-E568B2907131C0C0
  ```

  로그를 삭제할 경우에도 키 접두사가 유용하게 사용됩니다. 예를 들어, Amazon S3가 특정 키 접두사를 가진 객체를 삭제하도록 수명 주기 구성 규칙을 설정할 수 있습니다. 자세한 내용은 Amazon S3 로그 파일 삭제 (p. 719) 단원을 참조하십시오.
- (선택 사항) 생성된 로그의 액세스 권한. 기본적으로 버킷 소유자에게는 항상 로그 객체에 대한 모든 액세스 권한이 부여됩니다. 필요할 경우 다른 사용자에게 액세스 권한을 부여할 수 있습니다.

서버 액세스 로깅 활성화에 대한 자세한 내용은 콘솔을 이용하여 로깅 활성화 (p. 708) 및 프로그래밍 방식으로 로깅 활성화 (p. 708) 단원을 참조하십시오.

추가 로깅 고려 사항

Note

- 소스 및 대상 S3 버킷 모두 동일한 AWS 계정에서 소유하고 있어야 하며 S3 버킷 모두 동일한 리전에 있어야 합니다.

- Amazon S3는 버킷 정책이 아닌 버킷 ACL을 통해 액세스 로그를 전송할 권한 부여만 지원합니다.
- 버킷 정책에 deny 조건을 추가하면 Amazon S3가 액세스 로그를 전송하지 못할 수 있습니다.
- 대상 버킷의 기본 버킷 암호화는 AES256(SSE-S3)이 선택된 경우에만 사용할 수 있습니다. SSE-KMS 암호화는 지원되지 않습니다.

로그 객체 키 형식

Amazon S3는 대상 버킷에 업로드한 로그 객체에 대해 다음과 같은 객체 키 형식을 사용합니다.

```
TargetPrefixYYYY-mm-DD-HH-MM-SS-UniqueString/
```

키에서 `YYYY`, `mm`, `DD`, `HH`, `MM` 및 `SS`는 각각 로그 파일이 전송된 연도와 월, 일, 시, 분, 초에 대한 숫자를 나타냅니다. 이러한 날짜와 시간은 협정 세계시(UTC)로 표시됩니다.

로그 파일에는 해당 파일이 전송된 시간 이전에 기록된 레코드가 포함될 수 있습니다. 특정 기간의 모든 로그 레코드가 전송되었는지 여부는 확인할 수 없습니다.

키의 `UniqueString` 구성 요소는 파일의 덮어쓰기를 방지할 목적으로 사용되는 것으로, 특별한 의미가 없으므로 로그 처리 소프트웨어가 무시해도 됩니다.

후행 슬래시 /는 접두사의 끝을 나타내는 데 필요합니다.

로그 전송 방법

Amazon S3는 주기적으로 액세스 로그 레코드를 수집하여 로그 파일에 레코드를 통합한 다음 대상 버킷의 로그 객체로 로그 파일을 업로드합니다. 여러 원본 버킷에서 로깅을 활성화하고 동일한 대상 버킷을 지정한 경우 대상 버킷에 이러한 모든 원본 버킷의 액세스 로그가 저장됩니다. 그러나 각 로그 객체는 특정 원본 버킷의 액세스 로그 레코드만 보고합니다.

Amazon S3는 Log Delivery 그룹이라고 하는 특별 로그 전달 계정을 사용하여 액세스 로그를 기록합니다. 이 쓰기는 일반적인 액세스 제어 제약 조건을 따릅니다. 버킷의 ACL(액세스 통제 목록)에 권한 부여 항목을 추가하여 Log Delivery 그룹에 대상 버킷에 대한 쓰기 권한을 부여해야 합니다. Amazon S3 콘솔을 사용하여 버킷의 로깅을 활성화할 경우 콘솔은 원본 버킷에 로깅을 활성화하고, Log Delivery 그룹에 쓰기 권한을 부여하도록 대상 버킷의 ACL을 업데이트합니다.

서버 로그 전송은 보장되지 않음

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

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

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

버킷 로깅 상태 변경 시 일정 기간에 걸쳐 단계적으로 반영됨

버킷의 로깅 상태를 변경한 후 실제 로그 파일의 전송에 반영되려면 어느 정도 시간이 지나야 합니다. 예를 들어, 버킷에 대해 로깅을 활성화할 경우 이후 1시간 동안 이루어진 요청 중 일부는 기록되지만 일부는 기록되지 않을 수도 있습니다. 버킷 A에서 버킷 B로 로깅의 대상 버킷을 변경할 경우 이후 1시간 동안 일부 로그는 버킷 A로 계속 전송될 수 있지만, 다른 로그는 새로운 대상 버킷 B로 전송될 수 있습니다. 그러나 추가 작업을 수행하지 않아도 어느 정도 기간이 지나면 새 설정에 따라 모든 로그가 전송됩니다.

콘솔을 이용하여 로깅 활성화

AWS Management 콘솔에서 Amazon S3 서버 액세스 로깅 (p. 705)를 활성화하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷에 대해 서버 액세스 로깅을 활성화하려면 어떻게 해야 합니까? 단원을 참조하십시오.

버킷의 로깅을 활성화할 때 콘솔은 원본 버킷에 로깅을 활성화하고, 대상 버킷의 ACL(액세스 통제 목록)에 권한을 추가하여 Log Delivery 그룹에 쓰기 권한을 부여합니다.

프로그래밍 방식으로 로깅을 활성화하는 자세한 방법은 프로그래밍 방식으로 로깅 활성화 (p. 708) 섹션을 참조하십시오.

필드 목록과 그 설명을 포함하여 로그의 레코드 형식에 대한 자세한 내용은 Amazon S3 서버 액세스 로그 형식 (p. 711) 단원을 참조하십시오.

프로그래밍 방식으로 로깅 활성화

Amazon S3 API 또는 AWS SDK를 사용하여 프로그래밍 방식으로 로깅을 활성화 또는 비활성화할 수 있습니다. 그러려면 버킷에 로깅을 활성화하고, 대상 버킷의 Log Delivery 그룹에 로그를 쓸 수 있는 권한을 부여해야 합니다.

주제

- 로깅 활성화 (p. 708)
- Log Delivery 그룹에 WRITE 및 READ_ACP 권한 부여 (p. 709)
- 예: .NET용 AWS SDK (p. 709)
- 관련 리소스 (p. 711)

로깅 활성화

로깅을 활성화하기 위해 원본 버킷에 로깅 구성을 추가하는 PUT Bucket logging 요청을 제출합니다. 요청에 대상 버킷, 그리고 선택 사항으로 모든 로그 객체 키에 사용할 접두사를 지정합니다. 다음 예제에서는 대상 버킷으로 `logbucket`을, 접두사로 `logs/`를 지정합니다.

```
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <LoggingEnabled>
    <TargetBucket>logbucket</TargetBucket>
    <TargetPrefix>logs/</TargetPrefix>
  </LoggingEnabled>
</BucketLoggingStatus>
```

로그 객체는 Log Delivery 계정에서 쓰고 소유하며, 버킷 소유자에게는 로그 객체에 대한 모든 권한이 부여됩니다. 또한 필요할 경우 다른 사용자에게 로그에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 내용은 PUT Bucket logging 단원을 참조하십시오.

또한 Amazon S3는 버킷의 로깅 구성을 검색할 수 있는 GET Bucket logging API를 제공합니다. 로깅 구성을 삭제하려면 빈 `BucketLoggingStatus`로 PUT Bucket logging 요청을 보냅니다.

```
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
</BucketLoggingStatus>
```

Amazon S3 API 또는 AWS SDK 래퍼 라이브러리를 사용하여 버킷 로깅을 활성화할 수 있습니다.

Log Delivery 그룹에 WRITE 및 READ_ACP 권한 부여

Amazon S3는 대상 버킷에 로그 파일을 쓰기 위해 Log Delivery라는 사전 정의된 Amazon S3 그룹을 제공합니다. 이 쓰기는 일반적인 액세스 제어 제약 조건을 따릅니다. 대상 버킷의 ACL(액세스 통제 목록)에 권한을 추가하여 이 그룹에 `s3:GetObjectAcl` 및 `s3:PutObject` 권한을 부여해야 합니다. Log Delivery 그룹을 나타내는 URL은 입니다.

```
http://acs.amazonaws.com/groups/s3/LogDelivery
```

`WRITE` 및 `READ_ACP` 권한을 부여하기 위해 다음 권한 부여를 추가합니다. ACL에 대한 자세한 내용은 ACL을 사용한 액세스 관리 (p. 437) 단원을 참조하십시오.

```
<Grant>
    <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
    </Grantee>
    <Permission>WRITE</Permission>
</Grant>
<Grant>
    <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
    </Grantee>
    <Permission>READ_ACP</Permission>
</Grant>
```

AWS SDK를 사용하여 프로그래밍 방식으로 ACL에 권한 부여를 추가하는 예제는 AWS SDK for Java을 사용한 ACL 관리 (p. 444) 및 .NET용 AWS SDK을 사용한 ACL 관리 (p. 446) 섹션을 참조하십시오.

예: .NET용 AWS SDK

다음은 버킷의 로깅을 활성화하는 C# 예제입니다. 원본 버킷과 대상 버킷으로 2개의 버킷을 만들어야 합니다. 이 예제에서는 먼저 대상 버킷에 로그를 쓰는 데 필요한 권한을 Log Delivery 그룹에 부여한 다음 원본 버킷에서 로깅을 활성화합니다. 자세한 내용은 프로그래밍 방식으로 로깅 활성화 (p. 708) 단원을 참조하십시오. 실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 Amazon S3 .NET 코드 예시 실행 (p. 736) 단원을 참조하십시오.

Example

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

namespace Amazon.DocSamples.S3
```

```
{
    class ServerAccesLoggingTest
    {
        private const string bucketName = "*** bucket name for which to enable logging
 ***";
        private const string targetBucketName = "*** bucket name where you want access logs
 stored ***";
        private const string logObjectKeyPrefix = "Logs";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            EnableLoggingAsync().Wait();
        }

        private static async Task EnableLoggingAsync()
        {
            try
            {
                // Step 1 - Grant Log Delivery group permission to write log to the target
 bucket.
                await GrantPermissionsToWriteLogsAsync();
                // Step 2 - Enable logging on the source bucket.
                await EnableDisableLoggingAsync();
            }
            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);
            }
        }

        private static async Task GrantPermissionsToWriteLogsAsync()
        {
            var bucketACL = new S3AccessControlList();
            var aclResponse = client.GetACL(new GetACLRequest { BucketName =
 targetBucketName });
            bucketACL = aclResponse.AccessControlList;
            bucketACL.AddGrant(new S3Grantee { URI = "http://acs.amazonaws.com/groups/s3/
LogDelivery" }, S3Permission.WRITE);
            bucketACL.AddGrant(new S3Grantee { URI = "http://acs.amazonaws.com/groups/s3/
LogDelivery" }, S3Permission.READ_ACP);
            var setACLRequest = new PutACLRequest
            {
                AccessControlList = bucketACL,
                BucketName = targetBucketName
            };
            await client.PutACLAsync(setACLRequest);
        }

        private static async Task EnableDisableLoggingAsync()
        {
            var loggingConfig = new S3BucketLoggingConfig
            {
                TargetBucketName = targetBucketName,
                TargetPrefix = logObjectKeyPrefix
            };
```

```
            // Send request.
            var putBucketLoggingRequest = new PutBucketLoggingRequest
            {
                BucketName = bucketName,
                LoggingConfig = loggingConfig
            };
            await client.PutBucketLoggingAsync(putBucketLoggingRequest);
        }
    }
}
```

관련 리소스

- Amazon S3 서버 액세스 로깅 (p. 705)
- AWS CloudFormation 사용 설명서의 AWS::S3::Bucket

Amazon S3 서버 액세스 로그 형식

이 섹션에서는 Amazon S3 서버 액세스 로그 파일에 대해 설명합니다.

주제

- 복사 작업을 위한 추가 로깅 (p. 715)
- 사용자 지정 액세 로그 정보 (p. 719)
- 확장 가능 서버 액세스 로그 형식 프로그래밍 고려 사항 (p. 719)

서버 액세스 로그 파일은 새 줄로 구분된 로그 레코드의 시퀀스로 구성됩니다. 각 로그 레코드는 하나의 요청을 표시하며 공백으로 구분된 필드로 구성됩니다. 다음은 5개 로그 레코드로 구성된 로그 예제입니다.

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 awsexamplebucket [06/Feb/2019:00:00:38 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 3E57427F3EXAMPLE
 REST.GET.VERSIONING - "GET /awsexamplebucket?versioning HTTP/1.1" 200 - 113 - 7 - "-"
 "S3Console/0.4" - s9lzHYrFp76ZVxRcpX9+5cjAnEH2ROuNkd2BHfIa6UkFVdtjf5mKR3/eTPFvsiP/
XV/VLi31234= SigV2 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader awsexamplebucket.s3.us-
west-1.amazonaws.com TLSV1.1
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 awsexamplebucket [06/Feb/2019:00:00:38 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 891CE47D2EXAMPLE
 REST.GET.LOGGING_STATUS - "GET /awsexamplebucket?logging HTTP/1.1" 200 - 242
 - 11 - "-" "S3Console/0.4" - 9vKBE6vMhrNiWHZmb2L0mXOcqPGzQOI5XLnCtZNPxev+Hf
+7tpT6sxDwDty4LHBUOZJG96N1234= SigV2 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader
 awsexamplebucket.s3.us-west-1.amazonaws.com TLSV1.1
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 awsexamplebucket [06/Feb/2019:00:00:38 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be A1206F460EXAMPLE
 REST.GET.BUCKETPOLICY - "GET /awsexamplebucket?policy HTTP/1.1" 404
 NoSuchBucketPolicy 297 - 38 - "-" "S3Console/0.4" - BNaBsXZQQDbssi6xMBdBU2sLt
+Yf5kZDmeBUP35sFoKa3sLLeMC78iwEIWxs99CRUrbS4n11234= SigV2 ECDHE-RSA-AES128-GCM-SHA256
 AuthHeader awsexamplebucket.s3.us-west-1.amazonaws.com TLSV1.1
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 awsexamplebucket [06/Feb/2019:00:01:00 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 7B4A0FABBEXAMPLE
 REST.GET.VERSIONING - "GET /awsexamplebucket?versioning HTTP/1.1" 200 - 113 - 33 - "-"
 "S3Console/0.4" - Ke1bUcazaN1jWuUlPJaxF64cQVpUEhoZKEG/hmy/gijN/I1DeWqDfFvnpybfEseEME/
u7ME1234= SigV2 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader awsexamplebucket.s3.us-
west-1.amazonaws.com TLSV1.1
```

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 awsexamplebucket [06/Feb/2019:00:01:57 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 DD6CC733AEXAMPLE REST.PUT.OBJECT s3-dg.pdf "PUT /awsexamplebucket/
s3-dg.pdf HTTP/1.1" 200 - - 4406583 41754 28 "-" "S3Console/0.4" -
 10S62Zv81kBW7BB6SX4XJ48o6kpcl6LPwEoizZQQxJd5qDSCTLX0TgS37kYUBKQW3+bPdrg1234= SigV4 ECDHE-
RSA-AES128-SHA AuthHeader awsexamplebucket.s3.us-west-1.amazonaws.com TLSV1.1
```

> **Note**
>
> 아무 필드나 -로 설정하여 데이터를 알 수 없거나 사용할 수 없음 또는 해당 필드에 이 요청이 적용되지 않음을 표시할 수 있습니다.

다음 목록에서는 로그 레코드 필드에 대해 설명합니다.

버킷 소유자

원본 버킷의 정식 사용자 ID입니다. 정식 사용자 ID는 또 다른 형식의 AWS 계정 ID입니다. 정식 사용자 ID에 대한 자세한 내용은 AWS 계정 식별자를 참조하십시오. 계정의 정식 사용자 ID를 찾는 방법에 대한 정보는 계정 정식 사용자 ID 찾기를 참조하십시오.

입력 예

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

버킷

요청이 처리된 버킷의 이름. 시스템이 잘못된 양식의 요청을 수신하여 버킷을 결정할 수 없을 경우 해당 요청이 어떤 서버 액세스 로그에도 표시되지 않습니다.

입력 예

```
awsexamplebucket
```

시간

요청이 수신된 시간입니다. 이러한 날짜 및 시간은 협정 세계시(UTC)로 표시됩니다. `strftime()` 용어를 사용하는 형식은 `[%d/%b/%Y:%H:%M:%S %z]`입니다.

입력 예

```
[06/Feb/2019:00:00:38 +0000]
```

원격 IP

요청자의 명백한 인터넷 주소입니다. 중간 프록시 및 방화벽이 요청 시스템의 실제 주소를 가릴 수 있습니다.

입력 예

```
192.0.2.3
```

요청자

요청자의 정식 사용자 ID 또는 인증되지 않은 요청의 -입니다. 요청자가 IAM 사용자인 경우 이 필드는 IAM 사용자가 속한 AWS 루트 계정과 함께 요청자의 IAM 사용자 이름을 반환합니다. 이 식별자는 액세스 제어 목적으로 사용되는 것과 동일합니다.

입력 예

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

요청 ID

각 요청을 고유하게 식별하기 위해 Amazon S3에서 생성한 문자열입니다.

입력 예

```
3E57427F33A59F07
```

작업

여기에 나열된 작업은수명 주기 작업의 경우 `SOAP.operation`, `REST.HTTP_method.resource_type`, `WEBSITE.HTTP_method.resource_type` 또는 `BATCH.DELETE.OBJECT`나 `S3.action.resource_type`으로 선언됩니다.

입력 예

```
REST.PUT.OBJECT
```

키

요청의 "키" 부분. URL로 인코딩되거나 작업에 키 파라미터가 없을 경우 "-".

입력 예

```
/photos/2019/08/puppy.jpg
```

요청-URI

HTTP 요청 메시지의 Request-URI 부분.

입력 예

```
"GET /awsexamplebucket/photos/2019/08/puppy.jpg?x-foo=bar HTTP/1.1"
```

HTTP 상태

응답의 숫자 HTTP 상태 코드.

입력 예

```
200
```

오류 코드

Amazon S3 오류 코드 (p. 698), 또는 오류가 없을 경우 "-".

입력 예

```
NoSuchBucket
```

보낸 바이트

HTTP 프로토콜 오버헤드를 제외한 보낸 응답 바이트 수, 영일 경우 "-".

입력 예

```
2662992
```

객체 크기

해당 객체의 총 크기.

입력 예

```
3462992
```

총 시간

서버 관점에서 요청이 플라이트 상태를 유지한 시간(밀리초). 이 값은 요청이 수신된 시간부터 응답의 마지막 바이트가 전송된 시간까지 측정됩니다. 클라이언트 관점의 측정값은 네트워크 지연 시간으로 인해 더 길 수 있습니다.

입력 예

```
70
```

반환 시간

Amazon S3가 요청을 처리하는 데 소비한 시간(밀리초). 이 값은 요청의 마지막 바이트가 수신된 시간부터 응답의 첫 바이트가 전송된 시간까지 측정됩니다.

입력 예

```
10
```

Referer

HTTP Referrer 헤더의 값(있는 경우). HTTP 사용자 에이전트(예: 브라우저)는 일반적으로 이 헤더를 요청 시 연결 또는 포함 페이지의 URL로 설정합니다.

입력 예

```
"http://www.amazon.com/webservices"
```

사용자 에이전트

HTTP User-Agent 헤더의 값.

입력 예

```
"curl/7.15.1"
```

버전 ID

요청의 버전 ID, 또는 작업에 `versionId` 파라미터가 없을 경우 "-".

입력 예

```
3HL4kqtJvjVBH40Nrjfkd
```

호스트 Id

x-amz-id-2 또는 Amazon S3 확장 요청 ID.

입력 예

```
s9lzHYrFp76ZVxRcpX9+5cjAnEH2ROuNkd2BHfIa6UkFVdtjf5mKR3/eTPFvsiP/XV/VLi31234=
```

서명 버전

요청을 인증하는 데 사용된 서명 버전, `SigV2` 또는 `SigV4` 또는 인증되지 않은 요청에 대한 `-`.

입력 예

```
SigV2
```

암호 그룹

HTTPS 요청에 대해 협상된 Secure Sockets Layer (SSL) 암호 또는 HTTP에 대해 협상된 - 암호.

입력 예

```
ECDHE-RSA-AES128-GCM-SHA256
```

인증 유형

사용된 요청 인증 유형, 인증 헤더의 경우 `AuthHeader`, 쿼리 문자열(사전 서명된 URL)에 대해서는 `QueryString`, 인증되지 않은 요청에 대해서는 `-`.

입력 예

```
AuthHeader
```

호스트 헤더

Amazon S3에 연결하는 데 사용된 엔드포인트

입력 예

```
s3.us-west-2.amazonaws.com
```

일부 이전 리전에서 레거시 엔드포인트를 지원합니다. 서버 액세스 로그 또는 CloudTrail 로그에 이러한 엔드포인트가 표시될 수 있습니다. 자세한 내용은 레거시 엔드포인트 (p. 47) 단원을 참조하십시오. Amazon S3 리전 및 엔드포인트의 전체 목록은 AWS 일반 참조의 Amazon S3 리전 및 엔드포인트를 참조하십시오.

TLS 버전

클라이언트가 협상한 TLS(전송 계층 보안) 버전. 값은 `TLSv1`, `TLSv1.1`, `TLSv1.2` 또는 `-`(TLS가 사용되지 않은 경우) 중 하나입니다.

입력 예

```
TLSv1.2
```

복사 작업을 위한 추가 로깅

복사 작업에는 `GET` 및 `PUT`이 관련됩니다. 그러므로 복사 작업을 수행할 때 2개의 레코드가 로그됩니다. 위 표에서는 작업의 `PUT` 부분과 관련된 필드를 설명합니다. 아래 목록에서는 복사 작업의 `GET` 부분과 관련된 레코드의 필드에 대해 설명합니다.

버킷 소유자

복사 중인 객체가 저장되어 있는 버킷의 정식 사용자 ID입니다. 정식 사용자 ID는 또 다른 형식의 AWS 계정 ID입니다. 정식 사용자 ID에 대한 자세한 내용은 AWS 계정 식별자를 참조하십시오. 계정의 정식 사용자 ID를 찾는 방법에 대한 정보는 계정 정식 사용자 ID 찾기를 참조하십시오.

입력 예

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

버킷

복사 중인 객체가 저장되어 있는 버킷의 이름.

입력 예

```
awsexamplebucket
```

시간

요청이 수신된 시간입니다. 이러한 날짜 및 시간은 협정 세계시(UTC)로 표시됩니다. `strftime()` 용어를 사용하는 형식은 `[%d/%B/%Y:%H:%M:%S %z]`입니다.

입력 예

```
[06/Feb/2019:00:00:38 +0000]
```

원격 IP

요청자의 명백한 인터넷 주소입니다. 중간 프록시 및 방화벽이 요청 시스템의 실제 주소를 가릴 수 있습니다.

입력 예

```
192.0.2.3
```

요청자

요청자의 정식 사용자 ID 또는 인증되지 않은 요청의 -입니다. 요청자가 IAM 사용자일 경우 IAM 사용자가 속한 AWS 루트 계정과 함께 요청자의 IAM 사용자 이름이 이 필드에 반환됩니다. 이 식별자는 액세스 제어 목적으로 사용되는 것과 동일합니다.

입력 예

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

요청 ID

각 요청을 고유하게 식별하기 위해 Amazon S3에서 생성한 문자열입니다.

입력 예

```
3E57427F33A59F07
```

작업

여기에 나열된 작업은 `SOAP.`*`operation`*, `REST.`*`HTTP_method`*`.`*`resource_type`*, `WEBSITE.`*`HTTP_method`*`.`*`resource_type` 또는 `BATCH.DELETE.OBJECT`로 선언됩니다.

입력 예

```
REST.COPY.OBJECT_GET
```

키

복사 중인 객체의 "키", 작업에 키 파라미터가 없을 경우 "-".

입력 예

```
/photos/2019/08/puppy.jpg
```

요청-URI

HTTP 요청 메시지의 Request-URI 부분.

입력 예

```
"GET /awsexamplebucket/photos/2019/08/puppy.jpg?x-foo=bar"
```

HTTP 상태

복사 작업 `GET` 부분의 숫자 HTTP 상태 코드.

입력 예

```
200
```

오류 코드

복사 작업 `GET` 부분의 Amazon S3 오류 코드 (p. 698), 또는 오류가 없을 경우 "-"

입력 예

```
NoSuchBucket
```

보낸 바이트

HTTP 프로토콜 오버헤드를 제외한 보낸 응답 바이트 수, 영일 경우 "-".

입력 예

```
2662992
```

객체 크기

해당 객체의 총 크기.

입력 예

```
3462992
```

총 시간

서버 관점에서 요청이 플라이트 상태를 유지한 시간(밀리초). 이 값은 요청이 수신된 시간부터 응답의 마지막 바이트가 전송된 시간까지 측정됩니다. 클라이언트 관점의 측정값은 네트워크 지연 시간으로 인해 더 길 수 있습니다.

입력 예

```
70
```

반환 시간

Amazon S3가 요청을 처리하는 데 소비한 시간(밀리초). 이 값은 요청의 마지막 바이트가 수신된 시간부터 응답의 첫 바이트가 전송된 시간까지 측정됩니다.

입력 예

```
10
```

Referer

HTTP Referrer 헤더의 값(있는 경우). HTTP 사용자 에이전트(예: 브라우저)는 일반적으로 이 헤더를 요청 시 연결 또는 포함 페이지의 URL로 설정합니다.

입력 예

```
"http://www.amazon.com/webservices"
```

사용자 에이전트

HTTP User-Agent 헤더의 값.

입력 예

```
"curl/7.15.1"
```

버전 ID

복사 중인 객체의 버전 ID 또는 `x-amz-copy-source` 헤더가 복사 소스의 일부로 `versionId` 파라미터를 지정하지 않은 경우 "-"입니다.

입력 예

```
3HL4kqtJvjVBH40Nrjfkd
```

호스트 Id

x-amz-id-2 또는 Amazon S3 확장 요청 ID.

입력 예

```
s9lzHYrFp76ZVxRcpX9+5cjAnEH2ROuNkd2BHfIa6UkFVdtjf5mKR3/eTPFvsiP/XV/VLi31234=
```

서명 버전

요청을 인증하는 데 사용된 서명 버전, `SigV2` 또는 `SigV4` 또는 인증되지 않은 요청에 대한 `-`.

입력 예

```
SigV2
```

암호 그룹

HTTPS 요청에 대해 협상된 Secure Sockets Layer (SSL) 암호 또는 HTTP에 대해 협상된 - 암호.

입력 예

```
ECDHE-RSA-AES128-GCM-SHA256
```

인증 유형

사용된 요청 인증 유형, 인증 헤더의 경우 `AuthHeader`, 쿼리 문자열(사전 서명된 URL)에 대해서는 `QueryString`, 인증되지 않은 요청에 대해서는 `-`.

입력 예

```
AuthHeader
```

호스트 헤더

Amazon S3에 연결하는 데 사용된 엔드포인트

입력 예

```
s3.us-west-2.amazonaws.com
```

일부 이전 리전에서 레거시 엔드포인트를 지원합니다. 서버 액세스 로그 또는 CloudTrail 로그에 이러한 엔드포인트가 표시될 수 있습니다. 자세한 내용은 레거시 엔드포인트 (p. 47) 단원을 참조하십시오. Amazon S3 리전 및 엔드포인트의 전체 목록은 AWS 일반 참조의 Amazon S3 리전 및 엔드포인트를 참조하십시오.

TLS 버전

클라이언트가 협상한 TLS(전송 계층 보안) 버전. 값은 `TLSv1`, `TLSv1.1`, `TLSv1.2` 또는 `-`(TLS가 사용되지 않은 경우) 중 하나입니다.

입력 예

```
TLSv1.2
```

사용자 지정 액세 로그 정보

요청 URL에 사용자 지정 쿼리 문자열 파라미터를 추가하여 요청에 대한 액세스 로그 레코드에 저장할 사용자 지정 정보를 포함시킬 수 있습니다. Amazon S3는 "x-"로 시작하는 쿼리 문자열 파라미터를 무시하지만 해당 파라미터를 요청에 대한 액세스 로그 레코드에 로그 레코드의 `Request-URI` 필드의 일부로 포함시킵니다. 예를 들어 "s3.amazonaws.com/awsexamplebucket/photos/2019/08/puppy.jpg?x-user=johndoe"에 대한 `GET` 요청은 "s3.amazonaws.com/awsexamplebucket/photos/2019/08/puppy.jpg"에 대한 동일한 요청과 동일하게 작용하지만, 연결된 로그 레코드의 `Request-URI` 필드에 "x-user=johndoe" 문자열이 포함되는 점이 다릅니다. 이 기능은 REST 인터페이스에서만 사용할 수 있습니다.

확장 가능 서버 액세스 로그 형식 프로그래밍 고려 사항

수시로 각 줄의 끝에 새로운 필드를 추가하여 액세스 로그 레코드 형식을 확장할 수 있습니다. 서버 액세스 로그를 파싱하는 코드가 인식하지 못하는 후행 필드를 처리하도록 코드를 작성해야 합니다.

Amazon S3 로그 파일 삭제

시간이 지날수록 서버 액세스 로깅이 활성화된 S3 버킷에 수많은 서버 로그 객체가 누적될 수 있습니다. 생성 후 일정 기간 동안 애플리케이션에 이러한 액세스 로그가 필요할 수 있지만, 이후 더 이상 필요가 없어지

면 삭제하는 것이 좋습니다. Amazon S3 수명 주기 구성을 사용하여 Amazon S3가 자동으로 수명이 끝난 이러한 객체를 삭제하도록 규칙을 설정할 수 있습니다.

공유 접두사를 사용하여 S3 버킷의 일부 객체, 즉 공통 문자열로 시작하는 이름을 가진 객체에 대해 수명 주기 구성을 정의할 수 있습니다. 서버 액세스 로깅 구성에 접두사를 지정한 경우 해당 접두사를 가진 로그 객체를 삭제하도록 수명 주기 구성 규칙을 설정할 수 있습니다. 예를 들어 로그 객체 접두사가 `logs/`인 경우 지정된 기간이 지난 후 접두사가 `/logs`인 모든 객체를 버킷에서 삭제하도록 수명 주기 구성 규칙을 설정할 수 있습니다. 수명 주기 구성에 대한 자세한 내용은 객체 수명 주기 관리 (p. 123) 단원을 참조하십시오.

관련 리소스

Amazon S3 서버 액세스 로깅 (p. 705)

Amazon S3 액세스 로그를 사용하여 요청 식별

Amazon S3 액세스 로그를 사용하여 Amazon S3 요청을 식별할 수 있습니다.

Note

- Amazon S3 액세스 로그 대신 AWS CloudTrail 데이터 이벤트를 사용하는 것이 좋습니다. CloudTrail 데이터 이벤트는 더 쉽게 설정할 수 있고 더 많은 정보를 포함합니다. 자세한 내용은 AWS CloudTrail을 사용하여 Amazon S3 요청 식별 (p. 686) 단원을 참조하십시오.
- 받는 액세스 요청의 수에 따라 로그를 분석하는 데 더 많은 리소스 및/또는 더 많은 시간이 필요할 수 있습니다.

주제

- 요청에 대해 Amazon S3 액세스 로그 활성화 (p. 720)
- 요청에 대해 Amazon S3 액세스 로그 쿼리 (p. 722)
- Amazon S3 액세스 로그를 사용하여 서명 버전 2 요청 식별 (p. 724)
- Amazon S3 액세스 로그를 사용하여 객체 액세스 요청 식별 (p. 725)
- 관련 리소스 (p. 726)

요청에 대해 Amazon S3 액세스 로그 활성화

S3 버킷이 있는 각 AWS 리전에 전용 로깅 버킷을 생성하는 것이 좋습니다. 그런 다음 해당 S3 버킷에 Amazon S3 액세스 로그를 전달하십시오.

Example — 두 개 리전에 걸쳐 다섯 개 버킷으로 액세스 로그 활성화

이 예제에서는 다음과 같은 다섯 개 버킷이 있습니다.

- `1-awsexamplebucket-us-east-1`
- `2-awsexamplebucket-us-east-1`
- `3-awsexamplebucket-us-east-1`
- `1-awsexamplebucket-us-west-2`
- `2-awsexamplebucket-us-west-2`

1. 다음 리전에 두 개의 로깅 버킷을 생성합니다.

   - `awsexamplebucket-logs-us-east-1`
   - `awsexamplebucket-logs-us-west-2`

2. 그런 다음 다음과 같이 Amazon S3 액세스 로그를 활성화합니다.
   - `1-awsexamplebucket-us-east-1`은 접두사 `1-awsexamplebucket-us-east-1`이 있는 S3 버킷 `awsexamplebucket-logs-us-east-1`에 로깅합니다.
   - `2-awsexamplebucket-us-east-1`은 접두사 `2-awsexamplebucket-us-east-1`이 있는 S3 버킷 `awsexamplebucket-logs-us-east-1`에 로깅합니다.
   - `1-awsexamplebucket-us-east-1`은 접두사 `3-awsexamplebucket-us-east-1`이 있는 S3 버킷 `awsexamplebucket-logs-us-east-1`에 로깅합니다.
   - `1-awsexamplebucket-us-west-2`은 접두사 `1-awsexamplebucket-us-west-2`이 있는 S3 버킷 `awsexamplebucket-logs-us-west-2`에 로깅합니다.
   - `2-awsexamplebucket-us-west-2`은 접두사 `2-awsexamplebucket-us-west-2`이 있는 S3 버킷 `awsexamplebucket-logs-us-west-2`에 로깅합니다.
3. 그러면 다음 방법을 사용하여 Amazon S3 액세스 로그를 활성화할 수 있습니다.
   - AWS Management Console 사용 또는
   - 프로그래밍 방식으로 로깅 활성화 (p. 708) 또는
   - AWS CLI put-bucket-logging 명령을 사용하여 다음 명령을 통한 버킷의 액세스 로그를 프로그래밍 방식으로 활성화합니다.

     1. 먼저 `put-bucket-acl`을 사용하여 Amazon S3 권한을 부여합니다.

     ```
      aws s3api put-bucket-acl --bucket awsexamplebucket-logs  --grant-write
      URI=http://acs.amazonaws.com/groups/s3/LogDelivery --grant-read-acp URI=http://
     acs.amazonaws.com/groups/s3/LogDelivery
     ```

     2. 그런 다음 로깅 정책을 적용하십시오.

     ```
     aws s3api put-bucket-logging --bucket awsexamplebucket --bucket-logging-status
      file://logging.json
     ```

     `Logging.json`은 로깅 정책이 포함된 현재 폴더의 JSON 문서입니다.

     ```
      {
          "LoggingEnabled": {
              "TargetBucket": "awsexamplebucket-logs",
              "TargetPrefix": "awsexamplebucket/",
              "TargetGrants": [
                   {
                      "Grantee": {
                          "Type": "AmazonCustomerByEmail",
                          "EmailAddress": "user@example.com"
                       },
                      "Permission": "FULL_CONTROL"
                   }
               ]
           }
       }
     ```

     Note

     Amazon S3의 로그 전달 시스템에 필수 권한(write 및 read-acp 권한)을 부여하려면 `put-bucket-acl` 명령이 필요합니다.

3. bash 스크립트를 사용하여 계정의 모든 버킷에 대한 액세스 로깅을 추가하십시오.

```
  loggingBucket='awsexamplebucket-logs'
  region='us-west-2'


  # Create Logging bucket
  aws s3 mb s3://$loggingBucket --region $region

  aws s3api put-bucket-acl --bucket $loggingBucket --grant-write URI=http://
acs.amazonaws.com/groups/s3/LogDelivery --grant-read-acp URI=http://
acs.amazonaws.com/groups/s3/LogDelivery

  # List buckets in this account
  buckets="$(aws s3 ls | awk '{print $3}')"

  # Put bucket logging on each bucket
  for bucket in $buckets
      do printf '{
     "LoggingEnabled": {
         "TargetBucket": "%s",
         "TargetPrefix": "%s/"
          }
      }' "$loggingBucket" "$bucket"  > logging.json
      aws s3api put-bucket-logging --bucket $bucket --bucket-logging-status
 file://logging.json
      echo "$bucket done"
  done

  rm logging.json

  echo "Complete"
```

Note

모든 버킷이 같은 리전에 있는 경우에만 가능합니다. 여러 리전에 버킷이 있는 경우 스크립트를 조정해야 합니다.

요청에 대해 Amazon S3 액세스 로그 쿼리

Amazon S3에서는 서버 액세스 로그를 S3 버킷에 객체로 저장합니다. Amazon S3에서 로그를 분석할 수 있는 도구를 사용하는 것이 더 쉽습니다. Athena는 S3 객체의 분석을 지원하고 Amazon S3 액세스 로그를 쿼리하는 데 사용할 수 있습니다.

Example

다음 예에서는 Amazon Athena에서 Amazon S3 서버 액세스 로그를 쿼리하는 방법을 보여줍니다.

Note

Athena 쿼리에서 Amazon S3 위치를 지정하려면 다음과 같이 대상 버킷 이름과 대상 접두사가 필요합니다. `s3://awsexamplebucket-logs/prefix/`

1. https://console.aws.amazon.com/athena/에서 Athena 콘솔을 엽니다.
2. 쿼리 편집기에서 다음과 유사한 명령을 실행하십시오.

```
create database s3_access_logs_db
```

Note

S3 버킷과 동일한 AWS 리전에 데이터베이스를 생성하는 것이 가장 좋습니다.

3. 쿼리 편집기에서 다음과 비슷한 명령을 실행하여 2단계에서 만든 데이터베이스에 테이블 스키마를 생성합니다. `STRING` 및 `BIGINT` 데이터 형식 값이 액세스 로그 속성입니다. Athena에서 이 속성을 쿼리할 수 있습니다. `LOCATION`의 경우 앞서 설명한 대로 S3 버킷 및 접두사 경로를 입력하십시오.

```
  CREATE EXTERNAL TABLE IF NOT EXISTS s3_access_logs_db.mybucket_logs(
     BucketOwner STRING,
     Bucket STRING,
     RequestDateTime STRING,
     RemoteIP STRING,
     Requester STRING,
     RequestID STRING,
     Operation STRING,
     Key STRING,
     RequestURI_operation STRING,
     RequestURI_key STRING,
     RequestURI_httpProtoversion STRING,
     HTTPstatus STRING,
     ErrorCode STRING,
     BytesSent BIGINT,
     ObjectSize BIGINT,
     TotalTime STRING,
     TurnAroundTime STRING,
     Referrer STRING,
     UserAgent STRING,
     VersionId STRING,
     HostId STRING,
     SigV STRING,
     CipherSuite STRING,
     AuthType STRING,
     EndPoint STRING,
     TLSVersion STRING
  )
  ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
  WITH SERDEPROPERTIES (
              'serialization.format' = '1', 'input.regex' = '([^ ]*) ([^ ]*)
              \\[(.*?)\\] ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) \\\"([^ ]*) ([^ ]*)
 (- |[^ ]*)
              \\\" (-|[0-9]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) (\"[^
\"]*\") ([^ ]*)
              (?: ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*))?.*$' )
     LOCATION 's3://awsexamplebucket-logs/prefix'
```

4. 탐색 창의 데이터베이스 아래에서 데이터베이스를 선택하십시오.
5. 테이블에서 테이블 이름 옆의 Preview table(테이블 미리보기)을 선택하십시오.

   결과 창에 `bucketowner`, `bucket`, `requestdatetime` 등 서버 액세스 로그의 데이터가 표시됩니다. 즉, Athena 테이블이 만들어졌다는 뜻입니다. 이제 Amazon S3 서버 액세스 로그를 쿼리할 수 있습니다.

Example — 객체를 삭제한 사람과 시기 표시(타임스탬프, IP 주소 및 IAM 사용자)

```
SELECT RequestDateTime, RemoteIP, Requester, Key
FROM s3_access_logs_db.mybucket_logs
WHERE key = 'images/picture.jpg' AND operation like '%DELETE%';
```

Example — IAM 사용자가 실행한 모든 작업 표시

```
SELECT *
FROM s3_access_logs_db.mybucket_logs
WHERE requester='arn:aws:iam::123456789123:user/user_name';
```

Example — 특정 기간 동안 객체에 수행한 모든 작업 표시

```
SELECT *
FROM s3_access_logs_db.mybucket_logs
WHERE Key='prefix/images/picture.jpg'
    AND parse_datetime(RequestDateTime,'dd/MMM/yyyy:HH:mm:ss Z')
    BETWEEN parse_datetime('2017-02-18:07:00:00','yyyy-MM-dd:HH:mm:ss')
    AND parse_datetime('2017-02-18:08:00:00','yyyy-MM-dd:HH:mm:ss');
```

Example — 특정 기간 동안 특정 IP 주소에서 전송한 데이터의 양 표시

```
SELECT SUM(bytessent) AS uploadTotal,
      SUM(objectsize) AS downloadTotal,
      SUM(bytessent + objectsize) AS Total
FROM s3_access_logs_db.mybucket_logs
WHERE RemoteIP='1.2.3.4'
AND parse_datetime(RequestDateTime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2017-06-01','yyyy-MM-dd')
AND parse_datetime('2017-07-01','yyyy-MM-dd');
```

Note

로그를 보관하는 시간을 줄이려면 서버 액세스 로그 버킷에 대해 Amazon S3 수명주기 정책을 만들 수 있습니다. 주기적으로 로그 파일을 제거하도록 수명 주기 정책을 구성하십시오. 그러면 Athena 에서 쿼리마다 분석하는 데이터의 양이 줄어듭니다.

Amazon S3 액세스 로그를 사용하여 서명 버전 2 요청 식별

, 서명 버전 2에 대한 Amazon S3 지원이 비활성화됩니다(사용되지 않음). 이후 Amazon S3는 더 이상 서명 버전 2를 사용하는 요청을 수락하지 않으며 모든 요청은 서명 버전 4 서명을 사용해야 합니다. Amazon S3 액세스 로그를 사용하여 서명 버전 2 액세스 요청을 식별할 수 있습니다.

Note

- Amazon S3 액세스 로그 대신 AWS CloudTrail 데이터 이벤트를 사용하는 것이 좋습니다. CloudTrail 데이터 이벤트는 더 쉽게 설정할 수 있고 더 많은 정보를 포함합니다. 자세한 내용은 AWS CloudTrail를 사용하여 Amazon S3 서명 버전 2 요청 식별 (p. 689) 단원을 참조하십시오.

Example — 서명 버전 2 트래픽을 보내는 모든 요청자 표시

```
SELECT requester, Sigv, Count(Sigv) as SigCount
FROM s3_access_logs_db.mybucket_logs
```

```
                    GROUP BY requester, Sigv;
```

Amazon S3 액세스 로그를 사용하여 객체 액세스 요청 식별

Amazon S3 서버 액세스 로그에 대한 쿼리를 사용하여 GET, PUT, DELETE 등의 작업에 대해 Amazon S3 객체 액세스 요청을 식별하고 그러한 요청에 대한 추가 정보를 검색할 수 있습니다.

다음 Amazon Athena 쿼리 예제는 서버 액세스 로그에서 Amazon S3에 대한 모든 PUT 객체 요청을 가져오는 방법을 보여 줍니다.

Example — 특정 기간에 PUT 객체 요청을 보내는 모든 요청자 표시

```
SELECT Bucket, Requester, RemoteIP, Key, HTTPStatus, ErrorCode, RequestDateTime
FROM s3_access_logs_db
WHERE Operation='REST.PUT.OBJECT' AND
parse_datetime(RequestDateTime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2019-07-01:00:42:42','yyyy-MM-dd:HH:mm:ss')
AND
parse_datetime('2019-07-02:00:42:42','yyyy-MM-dd:HH:mm:ss')
```

다음 Amazon Athena 쿼리 예제는 서버 액세스 로그에서 Amazon S3에 대한 모든 GET 객체 요청을 가져오는 방법을 보여 줍니다.

Example — 특정 기간에 GET 객체 요청을 보내는 모든 요청자 표시

```
SELECT Bucket, Requester, RemoteIP, Key, HTTPStatus, ErrorCode, RequestDateTime
FROM s3_access_logs_db
WHERE Operation='REST.GET.OBJECT' AND
parse_datetime(RequestDateTime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2019-07-01:00:42:42','yyyy-MM-dd:HH:mm:ss')
AND
parse_datetime('2019-07-02:00:42:42','yyyy-MM-dd:HH:mm:ss')
```

다음 Amazon Athena 쿼리 예제는 서버 액세스 로그에서 S3 버킷에 대한 모든 익명 요청을 가져오는 방법을 보여 줍니다.

Example — 특정 기간에 버킷에 요청하는 모든 익명 요청자 표시

```
SELECT Bucket, Requester, RemoteIP, Key, HTTPStatus, ErrorCode, RequestDateTime
FROM s3_access_logs_db.mybucket_logs
WHERE Requester IS NULL AND
parse_datetime(RequestDateTime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2019-07-01:00:42:42','yyyy-MM-dd:HH:mm:ss')
AND
parse_datetime('2019-07-02:00:42:42','yyyy-MM-dd:HH:mm:ss')
```

Note

- 필요에 따라 요구에 맞게 날짜 범위를 수정할 수 있습니다.
- 이러한 쿼리 예제는 보안 모니터링에도 유용할 수 있습니다. 예상치 못하거나 승인되지 않은 IP 주소/요청자의 `PutObject` 또는 `GetObject` 호출 결과를 검토하고 버킷에 대한 익명 요청을 식별할 수 있습니다.

- 이 쿼리는 로깅이 활성화된 시간부터의 정보만 검색합니다.
- Amazon S3 AWS CloudTrail 로그를 사용하는 경우, AWS CloudTrail을 사용하여 Amazon S3 객체에 대한 액세스 식별 (p. 691) 단원을 참조하십시오.

관련 리소스

- Amazon S3 서버 액세스 로그 형식 (p. 711)
- AWS 서비스 로그 쿼리

AWS SDK, CLI 및 Explorer 사용

Amazon S3로 애플리케이션을 개발할 때 AWS SDK를 사용할 수 있습니다. AWS SDK에서는 기본 REST API를 래핑하여 프로그래밍 작업을 단순화합니다. 또한 AWS를 사용하여 커넥티드 모바일 애플리케이션과 웹 애플리케이션을 빌드하는 데에 AWS Mobile SDK와 AWS Amplify JavaScript 라이브러리를 사용할 수 있습니다.

이 단원에서는 Amazon S3 애플리케이션을 개발하기 위한 AWS SDK 사용 개요를 제공합니다. 또한 이 단원에서는 이 가이드에 제공된 AWS SDK 코드 예제를 테스트할 수 있는 방법을 설명합니다.

주제

- 요청 인증에서 서명 버전 지정 (p. 728)
- AWS CLI 설정 (p. 733)
- AWS SDK for Java 사용 (p. 733)
- .NET용 AWS SDK 사용 (p. 735)
- PHP용 AWS SDK 사용 및 PHP 예제 실행 (p. 736)
- Ruby용 AWS SDK 버전 3 사용 (p. 737)
- AWS SDK for Python (Boto) 사용 (p. 738)
- iOS용 및 Android용 AWS Mobile SDK 사용 (p. 738)
- AWS Amplify JavaScript 라이브러리 사용 (p. 739)

AWS SDK에 추가로, Visual Studio 및 Eclipse for Java IDE에서 AWS Explorer를 사용할 수 있습니다. 이 경우 SDK 및 Explorer가 AWS Toolkit으로 함께 번들 제공됩니다.

AWS 명령줄 인터페이스(AWS CLI)를 사용하여 Amazon S3 버킷과 객체를 관리할 수도 있습니다.

AWS Toolkit for Eclipse

AWS Toolkit for Eclipse에는 AWS SDK for Java 및 Eclipse용 AWS Explorer가 모두 포함되어 있습니다. AWS Explorer for Eclipse는 AWS를 사용하여 개발자가 쉽게 Java 애플리케이션을 개발, 디버깅 및 배포할 수 있도록 하는 Eclipse for Java IDE용 오픈 소스 플러그인입니다. 사용이 간편한 GUI를 통해 Amazon S3를 비롯한 AWS 인프라에 대한 액세스 및 관리가 가능합니다. 애플리케이션을 개발하면서 동시에 버킷 및 객체 관리, IAM 정책 설정과 같은 공통적인 작업을 Eclipse for Java IDE의 컨텍스트 내에서 모두 수행할 수 있습니다. 설치 지침은 Set up the Toolkit을 참조하십시오. Explorer 사용 예제는 How to Access AWS Explorer 단원을 참조하십시오.

AWS Toolkit for Visual Studio

AWS Explorer for Visual Studio는 Microsoft Visual Studio용 확장 프로그램입니다. 이를 통해 Amazon Web Services를 사용하여 .NET 애플리케이션을 쉽게 개발하고 디버깅하여 배포할 수 있습니다. 사용이 간편한 GUI를 통해 Amazon S3를 비롯한 AWS 인프라에 대한 액세스 및 관리가 가능합니다. 애플리케이션을 개발하면서 동시에 버킷 및 객체 관리, IAM 정책 설정과 같은 공통적인 작업을 Visual Studio의 컨텍스트 내에서 모두 수행할 수 있습니다. 설정 지침은 AWS Toolkit for Visual Studio 설정을 참조하십시오. Explorer를 사용한 Amazon S3 사용 예제는 AWS Explorer에서 Amazon S3 사용하기를 참조하십시오.

AWS SDK

SDK만 다운로드할 수 있습니다. SDK 라이브러리 다운로드에 대한 정보는 Sample Code & Libraries 단원을 참조하십시오.

AWS CLI

AWS CLI는 Amazon S3를 비롯한 AWS 서비스를 관리하는 통합 도구입니다. AWS CLI 다운로드에 대한 자세한 내용은 AWS Command Line Interface 단원을 참조하십시오.

요청 인증에서 서명 버전 지정

Amazon S3는 대부분의 AWS 리전에서 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. 서명 버전 2 지원 중단에 대한 자세한 내용은 Amazon S3에서 AWS 서명 버전 2 사용 중지 (p. 729) 단원을 참조하십시오.

모든 Amazon S3 리전과 지원되는 서명 버전의 목록은 AWS 일반 참조의 리전 및 엔드포인트를 참조하십시오.

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 프로필의 경우 다음 명령을 실행합니다:
	`$ aws configure set default.s3.signature_version s3v4`
	사용자 지정 프로필의 경우 다음 명령을 실행합니다:
	`$ aws configure set profile.your_profile_name.s3.signature_version s3v4`

Java SDK	코드에 다음을 추가합니다:
	`System.setProperty(SDKGlobalConfiguration.ENABLE_S3_SIGV4_SYSTEM_PROPER "true");`
	또는, 명령줄에 다음을 지정합니다:
	`-Dcom.amazonaws.services.s3.enableV4`

JavaScript SDK	클라이언트 생성 시에 `signatureVersion` 파라미터를 `v4`로 설정합니다: `var s3 = new AWS.S3({signatureVersion: 'v4'});`

PHP SDK	PHP SDK v2에 대한 Amazon S3 서비스 클라이언트 생성 시 `signature` 파라미터를 `v4`로 설정합니다.
	`<?php`
	`$client = S3Client::factory([`
	`'region' => 'YOUR-REGION',`
	`'version' => 'latest',`
	`'signature' => 'v4'`
	`]);`

SDK	요청 인증을 위한 서명 버전 4 요청

	PHP SDK v3를 사용하는 경우 Amazon S3 서비스 클라이언트를 생성하는 동안 `signature_version` 파라미터를 `v4`로 설정하십시오.
	
	`<?php`
	`$s3 = new Aws\S3\S3Client([`
	`  'version' => '2006-03-01',`
	`  'region' => 'YOUR-REGION',`
	`  'signature_version' => 'v4'`
	`]);`

Python-Boto SDK	기본 config 파일인 boto 파일에 다음을 지정합니다: `[s3] use-sigv4 = True`

Ruby SDK	Ruby SDK - 버전 1: 클라이언트 생성 시에 `:s3_signature_version` 파라미터를 `:v4`로 설정합니다:
	
	`s3 = AWS::S3::Client.new(:s3_signature_version => :v4)`
	
	Ruby SDK - 버전 3: 클라이언트 생성 시에 `signature_version` 파라미터를 `v4`로 설정합니다:
	
	`s3 = Aws::S3::Client.new(signature_version: 'v4')`

.NET SDK	Amazon S3 클라이언트를 만들기 전에 다음을 코드에 추가합니다.
	
	`AWSConfigsS3.UseSignatureVersion4 = true;`
	
	또는, config 파일에 다음을 추가합니다:
	
	`<appSettings>`
	`  <add key="AWS.S3.UseSignatureVersion4" value="true" />`
	`</appSettings>`

Amazon S3에서 AWS 서명 버전 2 사용 중지

Amazon S3에서 서명 버전 2의 사용이 중지됩니다. 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 요청을 악의적 사용자로부터 보호할 수 있게 해줍니다.

AWS 요청 서명에 대한 자세한 내용은 AWS General Reference에서 AWS API 요청에 서명을 참조하십시오.

무엇을 업데이트합니까?

AWS는 현재 서명 버전 2 및 서명 버전 4 프로세스를 사용하여 서명된 Amazon S3 API 요청만 지원합니다. 이후로는 Amazon S3가 서명 버전 4를 사용하여 서명된 요청만 수락합니다.

AWS 요청 서명에 대한 자세한 내용은 AWS General Reference에서 서명 버전 4 변경 사항을 참조하십시오.

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 출범한 리전에서만 지원됩니다. 이 업데이트를 통해 AWS에서는 모든 리전에서 보다 일관된 경험을 제공할 수 있습니다.

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 SKD는 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 사용하여 애플리케이션을 수정해야 합니다.

서명 버전 4로 전환 시 사용할 AWS SDK 버전에 대한 정보는 서명 버전 2에서 서명 버전 4로 전환 (p. 730) 단원을 참조하십시오.

Amazon S3 REST API를 통한 서명 버전 4 사용에 대한 자세한 내용은 Amazon Simple Storage Service API Reference에서 요청 인증(AWS 서명 버전 4)을 참조하십시오.

업데이트하지 않으면 어떻게 됩니까?

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 서명을 참조하십시오.
- 서명 버전 2와 서명 버전 4 간 변경 사항의 목록은 서명 버전 4 변경 사항을 참조하십시오.
- AWS 포럼에서 게시물 Amazon S3 API 요청 서명을 위해 AWS 서명 버전 2를 대체하는 AWS 서명 버전 4를 읽어 보십시오.
- 추가 문의 사항은 AWS Support로 문의하십시오.

서명 버전 2에서 서명 버전 4로 전환

현재 Amazon S3 API 요청 인증에 서명 버전 2를 사용 중인 경우 서명 버전 4로 전환해야 합니다. Amazon S3에서 AWS 서명 버전 2 사용 중지 (p. 729)에서 설명한 대로 서명 버전 2에 대한 지원이 종료할 예정입니다.

Amazon S3 REST API를 통한 서명 버전 4 사용에 대한 자세한 내용은 Amazon Simple Storage Service API Reference에서 요청 인증(AWS 서명 버전 4)을 참조하십시오.

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 (p. 728) 단원을 참조하십시오.

현재 사용 중인 SDK/제품	업그레이드할 SDK 버전	Sigv4를 사용하려면 클라이언트에서 코드를 변경해야 합니까?	SDK 설명서 링크
AWS SDK for Java v1	2018년 4분기에 Java 1.11.201+ 또는 v2로 업그레이드.	예	요청 인증에서 서명 버전 지정 (p. 728)
AWS SDK for Java v2(프리뷰)	SDK 업그레이드가 필요하지 않음.	아니요	AWS SDK for Java
.NET용 AWS SDK v1	3.1.10 이상으로 업그레이드.	예	.NET용 AWS SDK
.NET용 AWS SDK v2	3.1.10 이상으로 업그레이드.	아니요	.NET용 AWS SDK v2
.NET용 AWS SDK v3	3.3.0.0 이상으로 업그레이드.	예	.NET용 AWS SDK 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
PHP용 AWS SDK v1	S3 클라이언트 구성에서 서명 파라미터를 v4로 설정하고 PHP의 최신 버전(v2.7.4 이상)으로 업그레이드할 것을 권장합니다.	예	PHP용 AWS SDK
PHP용 AWS SDK v2	S3 클라이언트 구성에서 서명 파라미터를 v4로 설정하고 PHP의 최신 버전(v2.7.4 이상)으로 업그레이	아니요	PHP용 AWS SDK

현재 사용 중인 SDK/제품	업그레이드할 SDK 버전	Sigv4를 사용하려면 클라이언트에서 코드를 변경해야 합니까?	SDK 설명서 링크
	드할 것을 권장합니다.		
PHP용 AWS SDK v3	SDK 업그레이드가 필요하지 않음.	아니요	PHP용 AWS SDK
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 - Python용 AWS SDK
AWS CLI	1.11.108로 업그레이드.	예	AWS 명령줄 인터페이스
AWS CLI v2(프리뷰)	SDK 업그레이드가 필요하지 않음.	아니요	AWS 명령줄 인터페이스 버전 2
Ruby용 AWS SDK v1	Ruby V3로 업그레이드.	예	AWS용 Ruby V3
Ruby용 AWS SDK v2	Ruby V3로 업그레이드.	예	AWS용 Ruby V3
Ruby용 AWS SDK v3	SDK 업그레이드가 필요하지 않음.	아니요	AWS용 Ruby V3
Go	SDK 업그레이드가 필요하지 않음.	아니요	Go용 AWS SDK
C++	SDK 업그레이드가 필요하지 않음.	아니요	C++용 AWS SDK

Windows PowerShell용 AWS 도구 또는 PowerShell Core용 AWS 도구

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 CLI 설정

절차에 따라 AWS 명령줄 인터페이스(AWS CLI)를 다운로드하고 구성합니다.

> **Note**
>
> Amazon S3 같은 AWS 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 특정 URL이나 그 IAM 사용자의 자격 증명을 사용하여 AWS에 액세스할 수 있습니다. 지침은 IAM 사용 설명서의 IAM 사용자와 관리자 그룹 처음 만들기 단원을 참조하십시오.

AWS CLI를 설정하려면

1. AWS CLI를 다운로드하고 구성합니다. 지침은 AWS 명령줄 인터페이스 사용 설명서의 다음 주제를 참조하십시오.
   - AWS 명령줄 인터페이스로 설정
   - AWS 명령줄 인터페이스 구성
2. AWS CLI 구성 파일에서 관리자 사용자의 명명된 프로필을 추가합니다. 이 프로필은 AWS CLI 명령을 실행할 때 사용합니다.

   ```
   [adminuser]
   aws_access_key_id = adminuser access key ID
   aws_secret_access_key = adminuser secret access key
   region = aws-region
   ```

   사용 가능한 AWS 리전 목록은 AWS General Reference의 리전 및 엔드포인트를 참조하십시오.
3. 명령 프롬프트에 다음 명령을 입력하여 설정을 확인합니다.
   - `help` 명령을 실행하여 AWS CLI가 컴퓨터에 설치되어 있는지 확인합니다.

     ```
     aws help
     ```
   - `s3` 명령을 실행하여 사용자가 Amazon S3에 접속할 수 있는지 확인합니다. 이 명령은 계정의 버킷을 나열합니다. AWS CLI는 `adminuser` 자격 증명을 사용하여 요청을 인증합니다.

     ```
     aws s3 ls --profile adminuser
     ```

AWS SDK for Java 사용

AWS SDK for Java에서는 Amazon S3 버킷 및 객체 작업용 API를 제공합니다. 객체 작업의 경우 단일 작업에서 객체를 업로드하기 위해 API를 제공하는 것에 추가로, SDK에서는 대용량 객체를 여러 부분으로 나누어

업로드하기 위해 API를 제공합니다. 자세한 내용은 멀티파트 업로드 API를 사용한 객체 업로드 (p. 177) 단원을 참조하십시오.

주제

- Java API 구성 (p. 734)
- Amazon S3 Java 코드 예제 테스트 (p. 735)

AWS SDK for Java에서는 고위 수준 또는 하위 수준의 API를 사용하는 옵션을 제공합니다.

하위 수준 API

하위 수준 API는 버킷과 객체에 적용되는 작업의 생성, 업데이트, 삭제 등의 기본 Amazon S3 REST 작업에 적합합니다. 하위 수준 멀티파트 업로드 API를 사용하여 큰 객체를 업로드하면 더 많은 제어가 가능합니다. 예를 들어, 멀티파트 업로드를 일시 중지했다 다시 시작해야 하거나, 업로드 중에 부분 크기를 변경해야 하거나, 데이터 크기를 모르고 업로드를 시작할 경우 하위 수준 API를 사용할 수 있습니다. 이러한 요구 사항이 없는 경우 상위 수준 API를 사용하여 객체를 업로드합니다.

상위 수준 API

객체를 업로드할 경우, SDK에서는 `TransferManager` 클래스를 제공함으로써 더 높은 수준의 추상화를 제공합니다. 고위 수준 API는 더 간단한 API로, 단 몇 줄의 코드로 파일을 업로드하고 Amazon S3에 스트리밍할 수 있습니다. 앞서 하위 수준 API 단원에 설명된 대로 업로드를 제어해야 하는 경우가 아니라면 이 API를 사용하여 데이터를 업로드해야 합니다.

데이터 크기가 더 작은 경우 `TransferManager` API는 단일 작업으로 데이터를 업로드합니다. 그러나 데이터 크기가 특정 임계값에 도달하면 `TransferManager`는 멀티파트 업로드 API 사용으로 전환합니다. 가능한 경우 `TransferManager`에서는 여러 스레드를 사용하여 여러 부분을 동시에 업로드합니다. 한 부분의 업로드가 실패한 경우, API에서는 실패한 부분의 업로드를 최대 세 번까지 다시 시도합니다. 그러나 이는 `TransferManagerConfiguration` 클래스를 사용하여 구성 가능한 옵션입니다.

> **Note**
>
> 데이터 소스로 스트림을 사용하는 경우, `TransferManager` 클래스는 동시 업로드를 수행하지 않습니다.

Java API 구성

AWS SDK for Java의 다음 패키지에서는 이러한 API를 제공합니다.

- com.amazonaws.services.s3–Amazon S3 클라이언트를 생성하고 버킷 및 객체 사용에 필요한 API를 제공합니다. 예를 들어 이를 통해 버킷 생성, 객체 업로드, 객체 가져오기, 객체 삭제, 키 나열을 수행할 수 있습니다.
- com.amazonaws.services.s3.transfer–상위 수준 API 데이터 작업을 제공합니다.

  상위 수준 API는 Amazon S3와의 객체 전송을 간소화하기 위해 설계되었습니다. 여기에는 사용 방법, 쿼리 및 전송 조작을 위한 비동기식 메서드를 제공하는 `TransferManager` 클래스가 포함됩니다. 또한 파트 업로드를 위한 최소 부분 크기와 멀티파트 업로드를 사용하는 경우의 임계값 바이트를 구성하는 데 사용할 수 있는 `TransferManagerConfiguration` 클래스도 포함되어 있습니다.
- com.amazonaws.services.s3.model–하위 수준 API 클래스를 제공하여 요청을 생성하고 응답을 처리합니다. 예를 들어 객체 가져오기 요청을 형성하는 `GetObjectRequest` 클래스, 키 목록 작성 요청을 형성하는 `ListObjectsRequest` 클래스, 멀티파트 업로드를 생성하는 `InitiateMultipartUploadRequest` 클래스가 포함되어 있습니다.

AWS SDK for Java API에 대한 자세한 내용은 AWS SDK for Java API Reference을 참조하십시오.

Amazon S3 Java 코드 예제 테스트

이 설명서의 Java 예제는 AWS SDK for Java 버전 1.11.321과 호환됩니다. 코드 샘플 설정 및 실행에 대한 지침은 AWS SDK for Java 개발자 안내서의 AWS SDK for Java 시작하기를 참조하십시오.

.NET용 AWS SDK 사용

.NET용 AWS SDK에서는 Amazon S3 버킷 및 객체 작업용 API를 제공합니다. 객체 작업의 경우 단일 작업에서 객체를 업로드하기 위해 API를 제공하는 것에 추가로, SDK에서는 대용량 객체를 여러 부분으로 나누어 업로드하기 위해 이 API를 제공합니다(멀티파트 업로드 API를 사용한 객체 업로드 (p. 177) 참조).

주제

- .NET API 구성 (p. 735)
- Amazon S3 .NET 코드 예시 실행 (p. 736)

.NET용 AWS SDK에서는 고위 수준 또는 하위 수준의 API를 사용하는 옵션을 제공합니다.

하위 수준 API

하위 수준 API는 버킷과 객체에 적용되는 작업의 생성, 업데이트, 삭제를 비롯한 기본 Amazon S3 REST 작업에 적합합니다. 하위 수준 멀티파트 업로드 API(멀티파트 업로드 API를 사용한 객체 업로드 (p. 177) 참조)를 사용하여 큰 객체를 업로드하면 더 많은 제어가 가능합니다. 예를 들어, 멀티파트 업로드를 일시 중지했다 다시 시작해야 하거나, 업로드 중에 부분 크기를 변경해야 하거나, 데이터 크기를 모르고 업로드를 시작할 경우 하위 수준 API를 사용할 수 있습니다. 이러한 요구 사항이 없는 경우 객체 업로드에 상위 수준 API를 사용합니다.

상위 수준 API

객체를 업로드할 경우, SDK에서는 `TransferUtility` 클래스를 제공함으로써 더 높은 수준의 추상화를 제공합니다. 고위 수준 API는 더 간단한 API로, 단 몇 줄의 코드로 파일을 업로드하고 Amazon S3에 스트리밍할 수 있습니다. 앞서 하위 수준 API 단원에 설명된 대로 업로드를 제어해야 하는 경우가 아니라면 이 API를 사용하여 데이터를 업로드해야 합니다.

데이터 크기가 더 작은 경우 `TransferUtility` API는 단일 작업으로 데이터를 업로드합니다. 그러나 데이터 크기가 특정 임계값에 도달하면 `TransferUtility`는 멀티파트 업로드 API 사용으로 전환합니다. 기본적으로는 여러 스레드를 사용하여 여러 부분을 동시에 업로드합니다. 한 부분의 업로드가 실패한 경우, API에서는 실패한 부분의 업로드를 최대 세 번까지 다시 시도합니다. 그러나 이는 구성 가능한 옵션입니다.

> **Note**
>
> 데이터 소스로 스트림을 사용하는 경우, `TransferUtility` 클래스는 동시 업로드를 수행하지 않습니다.

.NET API 구성

.NET용 AWS SDK를 사용하여 Amazon S3 애플리케이션을 작성할 경우 `AWSSDK.dll`을 사용합니다. 이 조합에서의 다음 네임스페이스는 멀티파트 업로드 API를 제공합니다.

- Amazon.S3.Transfer–고위 수준 API를 제공하여 데이터를 파트로 업로드합니다.

  데이터를 업로드하기 위한 파일, 디렉터리 또는 스트림을 지정할 수 있는 `TransferUtility` 클래스가 포함되어 있습니다. 또한 동시 스레드 수, 파트 크기, 객체 메타데이터, 스토리지 클래스(STANDARD, REDUCED_REDUNDANCY) 및 객체 ACL(액세스 제어 목록) 등의 고급 설정을 구성하기 위한

`TransferUtilityUploadRequest` 및 `TransferUtilityUploadDirectoryRequest` 클래스도 포함되어 있습니다.

- Amazon.S3–하위 수준 API에 대한 구현을 제공합니다.

  Amazon S3 REST 멀티파트 업로드 API(멀티파트 업로드를 위해 REST API 사용 (p. 205) 참조)에 적합한 방법을 제공합니다.
- Amazon.S3.Model–하위 수준 API 클래스를 제공하여 요청을 생성하고 응답을 처리합니다. 예를 들어, 멀티파트 업로드를 초기화할 때 사용 가능한 `InitiateMultipartUploadRequest` 및 `InitiateMultipartUploadResponse` 클래스와 파트를 업로드할 때 사용 가능한 `UploadPartRequest` 및 `UploadPartResponse` 클래스를 제공합니다.
- Amazon.S3.Encryption–`AmazonS3EncryptionClient`를 제공합니다.
- Amazon.S3.Util–다양한 유틸리티 클래스를 제공합니다(예: `AmazonS3Util` 및 `BucketRegionDetector`).

.NET용 AWS SDK API에 대한 자세한 내용은 .NET용 AWS SDK 버전 3 API 참조를 참조하십시오.

Amazon S3 .NET 코드 예시 실행

이 설명서의 .NET 코드 예제는 .NET용 AWS SDK 버전 3.0과 호환됩니다. 코드 예정 설정 및 실행에 대한 자세한 내용은 .NET용 AWS SDK 개발자 안내서의 .NET용 AWS SDK 시작하기를 참조하십시오.

PHP용 AWS SDK 사용 및 PHP 예제 실행

PHP용 AWS SDK에서는 Amazon S3 버킷 및 객체 작업용 API에 대한 액세스를 제공합니다. SDK에서는 서비스의 하위 수준 API 또는 상위 수준 추상화를 사용하는 옵션을 제공합니다.

SDK는 PHP용 AWS SDK에서 제공됩니다. 여기에는 또한 SDK 설치 및 시작하기 관련 지침도 나와 있습니다.

PHP용 AWS SDK를 사용하기 위한 설정은 사용 환경과 애플리케이션을 실행하려는 방법에 따라 달라집니다. 이 설명서에 있는 예제를 실행하도록 환경을 설정하려면 PHP용 AWS SDK 시작 안내서를 참조하십시오.

주제

- PHP용 AWS SDK 수준 (p. 736)
- PHP 예제 실행 (p. 737)
- 관련 리소스 (p. 737)

PHP용 AWS SDK 수준

PHP용 AWS SDK에서는 고위 수준 또는 하위 수준의 API를 사용하는 옵션을 제공합니다.

하위 수준 API

하위 수준 API는 버킷 및 객체에 대한 작업의 생성, 업데이트, 삭제를 비롯하여 기본 Amazon S3 REST 작업에 적합합니다. 하위 수준 API에서는 이러한 작업에 대한 더 높은 수준의 제어를 제공합니다. 예를 들어, 요청을 일괄 처리하고 동시에 실행할 수 있습니다. 또는 멀티파트 업로드 API를 사용하면 객체 파트를 개별적으로 관리할 수 있습니다. 이러한 하위 수준 API 호출은 Amazon S3 응답의 세부 정보가 모두 포함된 결과를 반환합니다. 멀티파트 업로드 API에 대한 자세한 내용은 멀티파트 업로드 API를 사용한 객체 업로드 (p. 177) 단원을 참조하십시오.

상위 수준 추상화

상위 수준 추상화는 일반 사용 사례를 간소화하기 위해 마련되었습니다. 예를 들어 하위 수준 API를 사용하여 대용량 객체를 업로드하는 경우, 먼저 `Aws\S3\S3Client::createMultipartUpload()`를 호출한 다음 `Aws\S3\S3Client::uploadPart()` 메서드를 호출하여 객체 파트를 업로드한 뒤 `Aws\S3\S3Client::completeMultipartUpload()` 메서드를 호출하여 업로드를 완료합니다. 대신 멀티파트 업로드 생성을 간소화하는 상위 수준 `Aws\S3\\MultipartUploader` 객체를 사용할 수 있습니다.

또 다른 예로, 버킷의 객체를 열거할 때 버킷에 저장된 객체 수와 관계없이 PHP용 AWS SDK의 반복 기능을 사용하여 전체 객체 키를 반환할 수 있습니다. 하위 수준 API를 사용하는 경우 응답 시 최대 1,000의 키를 반환합니다. 버킷에 객체가 1,000개 이상인 경우 결과가 잘리므로 응답을 관리하고 잘렸는지 확인해야 합니다.

PHP 예제 실행

PHP용 AWS SDK 버전 3에 대한 Amazon S3 샘플을 설정 및 사용하려면 PHP용 AWS SDK Developer Guide의 설치를 참조하십시오.

관련 리소스

- Amazon S3용 PHP용 AWS SDK
- PHP용 AWS SDK 설명서
- Amazon S3의 PHP용 AWS SDK API

Ruby용 AWS SDK 버전 3 사용

Ruby용 AWS SDK에서는 Amazon S3 버킷 및 객체 작업용 API를 제공합니다. 객체 작업의 경우, API를 사용하여 단일 작업에서 객체를 업로드하거나 대용량 객체를 여러 부분으로 나누어 업로드할 수 있습니다(멀티파트 업로드를 위한 Ruby용 AWS SDK 사용 (p. 204) 참조). 그러나 단일 작업 업로드용 API에서도 마찬가지로 대용량 객체를 수용할 수 있으며, 백그라운드에서 파트의 업로드를 관리함으로써 쓰기 작업이 필요한 스크립트 수를 줄입니다.

Ruby API 구성

Ruby용 AWS SDK를 사용하여 Amazon S3 애플리케이션을 작성할 경우 Ruby용 SDK gem을 설치해야 합니다. 자세한 정보는 Ruby - 버전 3용 AWS SDK를 참조하십시오. 설치 후에는 다음 주요 클래스를 포함한 API에 액세스할 수 있습니다.

- Aws::S3::Resource–Ruby SDK용 Amazon S3 인터페이스를 표시하고 버킷을 생성 및 열거하는 메서드를 제공합니다.

  `S3` 클래스는 기존 버킷에 액세스하고 새 버킷을 생성하기 위한 `#buckets` 인스턴스 메서드를 제공합니다.
- Aws::S3::Bucket–Amazon S3 버킷을 표시합니다.

  `Bucket` 클래스는 버킷의 객체에 액세스하기 위한 `#object(key)` 및 `#objects` 메서드 및 버킷을 삭제하기 위한 메서드를 제공하며 버킷 정책과 같은 버킷에 대한 정보를 반환합니다.
- Aws::S3::Object–해당 키로 식별된 Amazon S3 객체를 표시합니다.

  `Object` 클래스는 객체 속성 가져오기 및 설정, 객체 저장을 위한 스토리지 클래스 지정, 액세스 통제 목록을 사용한 객체 권한 설정에 필요한 메서드를 제공합니다. 또한 `Object` 클래스에는 객체의 삭제, 업로드, 복사를 위한 메서드가 포함되어 있습니다. 객체를 여러 부분으로 업로드할 경우 이 클래스는 업로드된 파트의 순서와 파트 크기를 지정하는 옵션을 제공합니다.

Ruby API용 AWS SDK에 대한 자세한 정보는 Ruby - 버전 2용 AWS SDK를 참조하십시오.

Ruby 스크립트 예제 테스트

Ruby 스크립트 예제를 시작하는 가장 쉬운 방법은 최신 Ruby용 AWS SDK gem을 설치하는 것입니다. 최신 gem으로 설치 및 업데이트에 대한 자세한 정보는 Ruby - 버전 3용 AWS SDK를 참조하십시오. 다음 작업은 Ruby용 AWS SDK를 설치했다는 가정 하에 Ruby 스크립트 예제의 작성 및 테스트 과정을 안내합니다.

Ruby 스크립트 예제의 일반적인 작성 및 테스트 프로세스

1	AWS에 액세스하려면 Ruby용 SDK 애플리케이션에 대한 자격 증명 세트를 제공해야 합니다. 자세한 내용은 Ruby용 AWS SDK 구성을 참조하십시오.

2	새 Ruby용 SDK 스크립트를 만들어 다음 줄을 스크립트 맨 앞에 추가합니다.
	`#!/usr/bin/env ruby`
	`require 'rubygems'`
	`require 'aws-sdk-s3'`
	첫 번째 줄은 인터프리터 명령이며 두 개의 `require` 문은 필요한 gem 두 개를 스크립트로 가져옵니다.

3	읽고 있는 섹션에서 코드를 스크립트로 복사합니다.
4	필요한 데이터를 모두 제공하여 코드를 업데이트합니다. 예를 들어, 파일을 업로드하는 경우 파일 경로 및 버킷 이름을 제공합니다.
5	스크립트를 실행합니다. AWS Management 콘솔을 사용하여 버킷 및 객체에 대한 변경 사항을 확인합니다. AWS Management 콘솔에 대한 자세한 내용은 https://aws.amazon.com/console/을 참조하십시오.

Ruby 샘플

다음 링크에는 Ruby용 SDK 버전 3을 시작하는 데 도움이 되는 샘플이 있습니다.

- Ruby 버전 3용 AWS SDK 사용 (p. 61)
- Ruby용 AWS SDK를 사용하여 객체 업로드 (p. 176)

AWS SDK for Python (Boto) 사용

Boto는 Amazon S3.를 포함한 AWS에 인터페이스를 제공하는 Python 패키지입니다. Boto에 대한 자세한 내용은 AWS SDK for Python (Boto)을 참조하십시오. 이 페이지의 시작하기 링크를 선택하면 시작하기 위한 단계별 지침을 볼 수 있습니다.

iOS용 및 Android용 AWS Mobile SDK 사용

Android용 및 iOS용 AWS Mobile SDK를 사용하면 기존 모바일 앱에 견고한 클라우드 백엔드를 빠르고 쉽게 통합할 수 있습니다. AWS 전문가가 되지 않고도 사용자 로그인, 데이터베이스, 푸시 알림 등의 기능을 구성하고 시작할 수 있습니다.

AWS Mobile SDK는 Amazon S3 및 다른 많은 AWS 서비스에 쉽게 액세스할 수 있게 해줍니다. AWS Mobile SDK를 사용해 보려면 AWS Mobile SDK 시작하기를 참조하십시오.

추가 정보

AWS Amplify JavaScript 라이브러리 사용 (p. 739)

AWS Amplify JavaScript 라이브러리 사용

AWS Amplify는 클라우드 지원 애플리케이션을 빌드하는 웹 및 모바일 개발자를 위한 오픈 소스 JavaScript 라이브러리입니다. AWS Amplify는 사용자 지정 가능한 UI 구성 요소, S3 버킷과 연동하는 선언 인터페이스 및 AWS 서비스를 위한 기타 상위 범주를 제공합니다.

AWS Amplify JavaScript 라이브러리를 사용하여 시작하려면 다음 링크 중 하나를 선택합니다.

- 웹용 AWS Amplify 라이브러리 시작하기
- React Native용 AWS Amplify 라이브러리 시작하기

AWS Amplify에 대한 자세한 내용은 GitHub에서 AWS Amplify를 참조하십시오.

추가 정보

iOS용 및 Android용 AWS Mobile SDK 사용 (p. 738)

부록

이 Amazon Simple Storage Service 개발자 가이드 부록에는 다음과 같은 단원이 포함되어 있습니다.

주제

- 부록 A: SOAP API 사용 (p. 740)
- 부록 B: 요청 인증(AWS Signature 버전 2) (p. 743)

부록 A: SOAP API 사용

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

이 단원에는 Amazon S3 SOAP API에 관련된 정보가 포함되어 있습니다.

> **Note**
>
> SOAP 요청은 인증된 경우와 익명인 경우 모두 SSL을 사용하여 Amazon S3로 전송해야 합니다. HTTP로 SOAP 요청을 전송하면 Amazon S3가 오류를 반환합니다.

공통 SOAP API 요소

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

HTTP를 통한 SOAP 1.1을 사용하여 Amazon S3와 상호 작용할 수 있습니다. Amazon S3 API를 머신 판독 가능한 방법으로 설명하는 Amazon S3 WSDL이 다음 위치에 제공됩니다. https://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl. Amazon S3 스키마는 https://doc.s3.amazonaws.com/2006-03-01/AmazonS3.xsd에 제공됩니다.

대부분의 사용자는 해당 언어 및 개발 환경에 맞춘 SOAP 도구 키트를 사용하여 Amazon S3와 상호 작용합니다. 여러 가지 도구 키트에서 Amazon S3 API를 다양한 방식으로 제공합니다. 특정 도구 키트의 사용 방법을 알아보려면 해당 설명서를 참조하십시오. 이 단원에서는 웹에 표시되는 대로 XML 요청 및 응답을 나타냄으로써 도구 키트에 독립적인 방식으로 Amazon S3 SOAP 작업을 설명합니다.

공통 요소

다음 권한 부여 관련 요소를 SOAP 요청에 포함할 수 있습니다.

- `AWSAccessKeyId`: 요청자의 AWS 액세스 키 ID
- `Timestamp`: 시스템의 현재 시간
- `Signature`: 요청에 대한 서명

SOAP 요청 인증

Note

HTTP를 통한 SOAP 지원은 중단되었지만 HTTPS를 통해 계속해서 사용할 수 있습니다. 새로운 Amazon S3 기능은 SOAP에 대해 지원되지 않습니다. 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 (p. 10)을 참조하십시오.
- `Timestamp:` 반드시 `2009-01-01T12:00:00.000Z`와 같은 협정 세계시(그리니치 표준시) 시간대의 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 Secret Access Key를 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

SOAP 요청은 인증된 경우와 익명인 경우 모두 SSL을 사용하여 Amazon S3로 전송해야 합니다. HTTP로 SOAP 요청을 전송하면 Amazon S3가 오류를 반환합니다.

Important

시간 정밀도의 유효 자릿수에 대한 해석이 다양하기 때문에 .NET 사용자는 너무 자세한 타임스탬프를 Amazon S3(으)로 보내지 않도록 합니다. 밀리초 단위만 사용해 수동으로 `DateTime` 객체를 생성하면 됩니다.

SOAP를 사용한 액세스 정책 설정

Note

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

액세스 제어는 "AccessControlList" 요소를 `CreateBucket`, `PutObjectInline` 또는 `PutObject`에 대한 요청에 포함하여 버킷 또는 객체가 기록될 때 설정할 수 있습니다. AccessControlList 요소에 관한 설명은 Amazon S3의 Identity and Access Management (p. 299)에 나와 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>&quot828ef3fdfa96f00ad9f27c383fc9ac7f&quot</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.

부록 B: 요청 인증(AWS Signature 버전 2)

Important

이 단원에서는 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 내용은 Amazon S3에서 AWS 서명 버전 2 사용 중지 (p. 729) 단원을 참조하십시오.
서명 버전 4는 모든 AWS 리전에서 지원되며, 새 리전을 지원하는 유일한 버전입니다. 자세한 내용은 Amazon Simple Storage Service API Reference의 요청 인증(AWS 서명 버전 4)을 참조하십시오.
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 AWS CloudTrail를 사용하여 Amazon S3 서명 버전 2 요청 식별 (p. 689)에서 이러한 요청을 쿼리하고 식별하는 방법을 참조하십시오.
- Amazon S3 서버 액세스 로그를 사용하는 경우 Amazon S3 액세스 로그를 사용하여 서명 버전 2 요청 식별 (p. 724) 단원을 참조하십시오.

주제

- REST API를 사용하여 요청 인증 (p. 743)
- REST 요청 서명 및 인증 (p. 745)
- POST를 사용한 브라우저 기반 업로드(AWS 서명 버전 2) (p. 754)

REST API를 사용하여 요청 인증

REST를 사용하여 Amazon S3에 액세스할 때는 요청이 인증될 수 있도록 요청에 다음 항목을 제공해야 합니다.

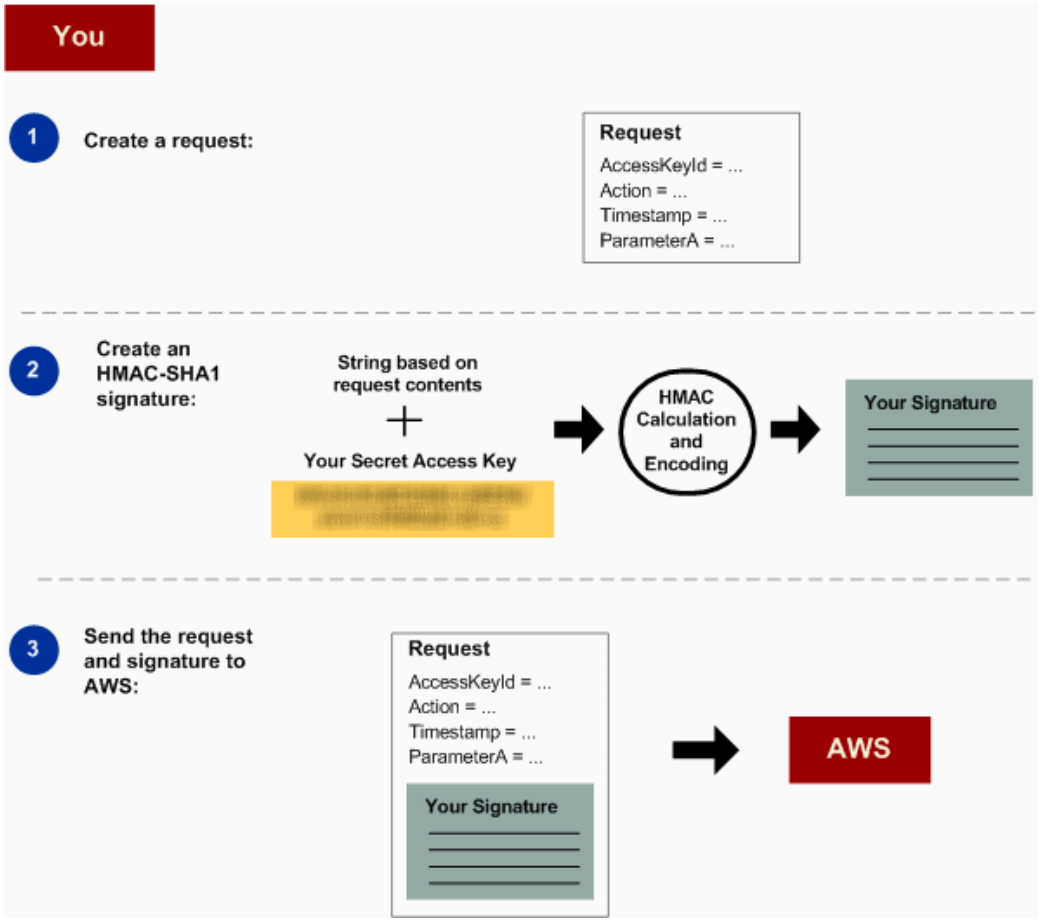
요청 요소

- 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.

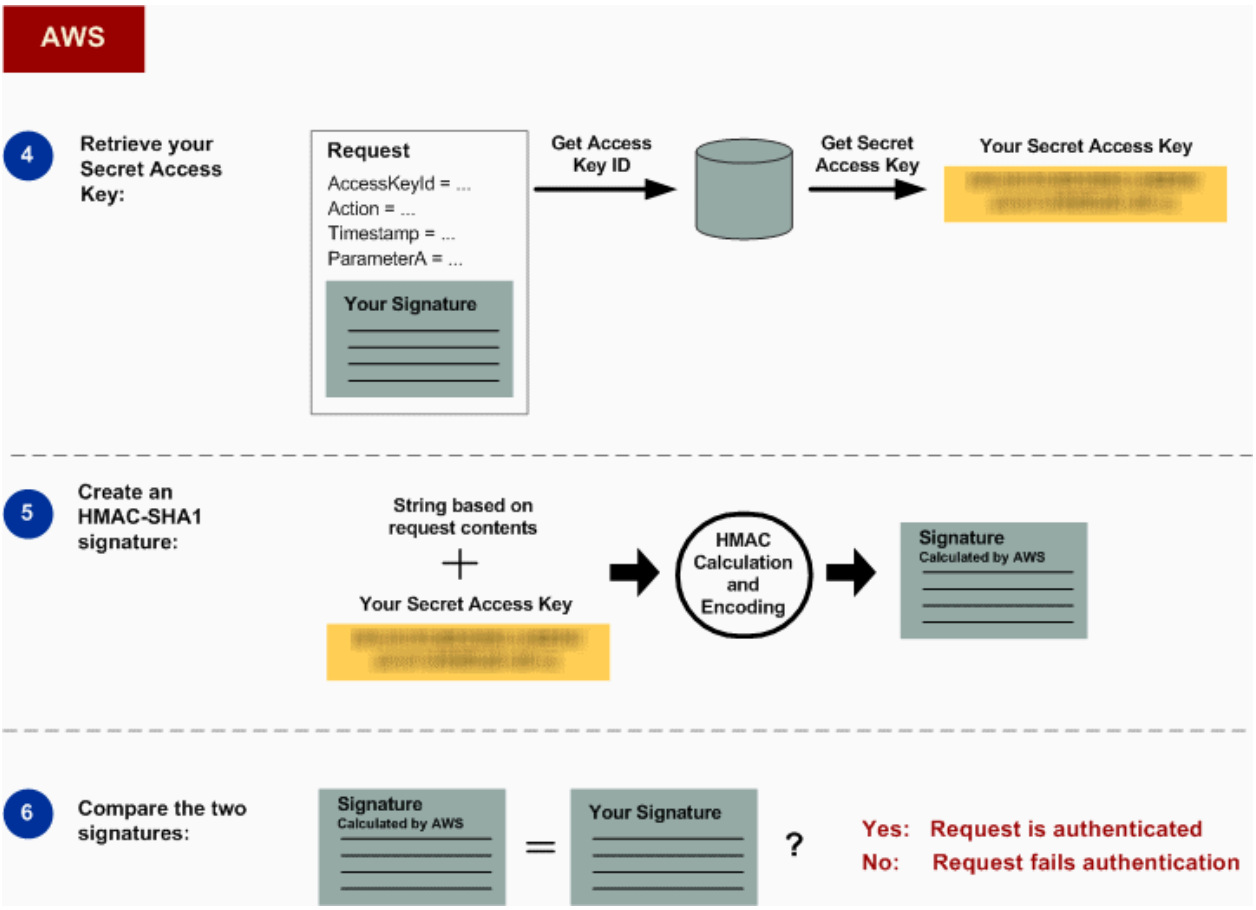
  요청 서명은 사용자와 AWS에만 알려지는 공유 암호인 보안 액세스 키를 사용하여 계산됩니다.
- 타임스탬프 – 각 요청에 요청이 생성된 날짜 및 시간을 UTC 문자열로 표기하여 포함해야 합니다.
- 날짜 – 각 요청에 요청의 타임스탬프를 포함해야 합니다.

  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.



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 세 단계를 수행합니다.



4	Amazon S3는 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 않을 경우 요청이 삭제되고 Amazon S3에서 오류 응답을 반환합니다.

세부 인증 정보

REST 인증에 대한 세부 정보는 REST 요청 서명 및 인증 (p. 745)을 참조하십시오.

REST 요청 서명 및 인증

주제

- 임시 보안 자격 증명 사용 (p. 746)
- Authentication 헤더 (p. 747)
- 서명에 대한 요청 정규화 (p. 747)
- CanonicalizedResource 요소 구성 (p. 747)
- CanonicalizedAmzHeaders 요소 구성 (p. 748)

- 위치 및 이름 지정된 HTTP 헤더 StringToSign 요소 (p. 749)
- 타임스탬프 요구 사항 (p. 749)
- 인증 예제 (p. 749)
- REST 요청 서명 문제 (p. 752)
- 쿼리 문자열 요청 인증 대체 방법 (p. 753)

> **Note**
>
> 이 주제에서는 서명 버전 2를 사용한 요청 인증을 설명합니다. Amazon S3는 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 Reference의 요청 인증(AWS 서명 버전 4)을 참조하십시오.

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) (p. 754) 단원을 참조하십시오.

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 보안 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 보안 액세스 키를 가져와 동일한 방식으로 사용하여 받은 메시지에 대한 서명을 계산합니다. 그런 다음 계산한 서명을 요청자가 제시한 서명과 비교합니다. 두 서명이 일치하면 요청자가 AWS 보안 액세스 키에 대한 액세스 권한을 가진고 있는 것으로 간주되어 키가 발행된 보안 주체의 권한으로 동작합니다. 두 서명이 일치하지 않으면 요청이 삭제되고 시스템에서 오류 메시지를 반환합니다.

Example 인증된 Amazon S3 REST 요청

```
GET /photos/puppy.jpg HTTP/1.1
Host: awsexamplebucket.s3.us-west-1.amazonaws.com
Date: Mon, 26 Mar 2007 19:37:58 +0000

Authorization: AWS AKIAIOSFODNN7EXAMPLE:frJIUN8DYpKDtOLCwo//yllqDzg=
```

임시 보안 자격 증명 사용

임시 보안 자격 증명을 사용하여 요청을 서명할 경우(요청 만들기 (p. 10) 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 Reference 가이드의 작업을 참조하십시오.

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 보안 액세스 키 ID와 AWS 보안 액세스 키를 발급받습니다. 요청 인증에 대해 `AWSAccessKeyId` 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 `Signature`와 일치하면 요청자는 AWS 보안 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" + " " + AWSAccessKeyId + ":" + Signature;

Signature = Base64( HMAC-SHA1( 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", "?logging", or "?torrent"];

CanonicalizedAmzHeaders = <described below>
```

HMAC-SHA1은 메시지 인증에 대한 RFC 2104 - 키-해싱으로 정의되는 알고리즘입니다. 알고리즘은 키와 메시지로 구성된 2바이트 문자열을 입력으로 사용합니다. Amazon S3 인증 요청의 경우, AWS 보안 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 서명을 계산해야 합니다. 서명 정규화를 위해 요청을 합의된 형식에 추가(put)하는 프로세스를 호출합니다.

CanonicalizedResource 요소 구성

`CanonicalizedResource`는 요청 대상인 Amazon S3 리소스를 나타냅니다. REST 요청의 경우 다음과 같이 작성합니다.

프로세스 시작

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 (p. 43)을 참조하십시오. 가상 호스팅 방식 요청 "https://awsexamplebucket.s3.us-west-1.amazonaws.com/photos/puppy.jpg“의 경우 `CanonicalizedResource`는 “/awsexamplebucket”입니다. 경로 방식 요청인 "https://s3.us-west-1.amazonaws.com/awsexamplebucket/photos/puppy.jpg"에서 `CanonicalizedResource`는 ""입니다.
3	디코딩되지 않는 HTTP 요청 URI의 경로 부분을 쿼리 문자열 직전까지 추가합니다. 가상 호스팅 방식 요청 "https://awsexamplebucket.s3.us-west-1.amazonaws.com/photos/puppy.jpg“의 경우 `CanonicalizedResource`는 “/awsexamplebucket/photos/puppy.jpg”입니다. 경로 방식 요청인 "https://s3.us-west-1.amazonaws.com/awsexamplebucket/photos/puppy.jpg“에서 `CanonicalizedResource`는 “/awsexamplebucket/photos/puppy.jpg”입니다. 이 시점에서 `CanonicalizedResource`는 가상 호스팅 방식과 경로 방식 요청에서 동일합니다. GET Service와 같이 버킷을 다루지 않는 요청의 경우, "/"를 추가합니다.
4	요청이 `?versioning`, `?location`, `?acl`, `?torrent`, `?lifecycle` 또는 `?versionid`와 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, torrent, uploadId, uploads, versionId, versioning, versions, website입니다. 요청이 응답 헤더 값을 재정의하 쿼리 문자열 파라미터를 지정할 경우(객체 Get 참조), 쿼리 문자열 파라미터와 그 값을 추가합니다. 서명 시에는 이러한 값을 인코딩하지 않지만, 요청을 할 때에는 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-encoding`이 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.

`CanonicalizedResource`는 HTTP 요청-URI와는 다를 수 있습니다. 특히, 요청이 HTTP `Host` 헤더를 사용하여 버킷을 지정할 경우에는 HTTP 요청-URI에 버킷이 표시되지 않습니다. 하지만 `CanonicalizedResource`에는 계속 버킷이 포함됩니다. 쿼리 문자열 파라미터는 요청-URI에 표시될 수 있지만 `CanonicalizedResource`에는 포함되지 않습니다. 자세한 내용은 버킷의 가상 호스팅 (p. 43) 단원을 참조하십시오.

CanonicalizedAmzHeaders 요소 구성

`StringToSign`의 CanonicalizedAmzHeaders 부분을 작성하려면 'x-amz-'(대/소문자 구분 없는 비교 사용)로 시작하는 모든 HTTP 요청 헤더를 선택한 후 다음 프로세스를 사용합니다.

CanonicalizedAmzHeaders 프로세스

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 사용자에 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 (p. 753) 단원을 참조하십시오.

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 Date을 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
AWSSecretAccessKey	wJalrXUtnFEMI/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 awsexamplebucket 버킷에서 객체를 가져옵니다.

요청	StringToSign

`GET /photos/puppy.jpg HTTP/1.1`
`Host: awsexamplebucket.us-west-1.s3.amazonaws.com`
`Date: Tue, 27 Mar 2007 19:36:42 +0000`
 	
`Authorization: AWS AKIAIOSFODNN7EXAMPLE:bWq2s1WEIj+Ydj0vQ697zp+IXMU=`
	`GET\n`
	`\n`
	`\n`
	`Tue, 27 Mar 2007 19:36:42 +0000\n`
	`/awsexamplebucket/photos/puppy.jpg`

CanonicalizedResource는 버킷 이름을 포함하지만 HTTP 요청-URI는 그렇지 않습니다. 버킷은 호스트 헤더가 지정합니다.

객체 PUT

이 예제는 awsexamplebucket 버킷으로 객체를 가져옵니다.

요청	StringToSign

`PUT /photos/puppy.jpg HTTP/1.1`
`Content-Type: image/jpeg`
`Content-Length: 94328`
`Host: awsexamplebucket.s3.us-west-1.amazonaws.com`
`Date: Tue, 27 Mar 2007 21:15:45 +0000`
 	
`Authorization: AWS AKIAIOSFODNN7EXAMPLE:MyyxeRY7whkBe+bq8fHCL/2kKUg=`
	`PUT\n`
	`\n`
	`image/jpeg\n`
	`Tue, 27 Mar 2007 21:15:45 +0000\n`
	`/awsexamplebucket/photos/puppy.jpg`

요청과 StringToSign의 Content-Type 헤더에 유의합니다. 또한 Content-MD5는 요청에 나타나지 않기 때문에 StringToSign에서 공백이어야 합니다.

나열

이 예제는 awsexamplebucket 버킷의 내용을 나열합니다.

요청	StringToSign

`GET /?prefix=photos&max-keys=50&marker=puppy HTTP/1.1`
`User-Agent: Mozilla/5.0`
`Host: awsexamplebucket.s3.us-west-1.amazonaws.com`
`Date: Tue, 27 Mar 2007 19:42:41 +0000`
 	
`Authorization: AWS AKIAIOSFODNN7EXAMPLE:htDYFYduRNen8P9ZfE/s9SuKy0U=`
	`GET\n`
	`\n`
	`\n`
	`Tue, 27 Mar 2007 19:42:41 +0000\n`
	`/awsexamplebucket/`

CanonicalizedResource 뒤에 슬래시가 오고, 쿼리 문자열 파라미터가 없다는 점에 유의합니다.

가져오기

이 예제는 'awsexamplebucket' 버킷에 대한 액세스 제어 정책 하위 리소스를 가져옵니다.

요청	StringToSign

`GET /?acl HTTP/1.1`
`Host: awsexamplebucket.s3.us-west-1.amazonaws.com`
`Date: Tue, 27 Mar 2007 19:44:46 +0000`
 	
`Authorization: AWS AKIAIOSFODNN7EXAMPLE:`
`c2WLPFtWHVgbEmeEG93a4cG37dM=`
	`GET\n`
	`\n`
	`\n`
	`Tue, 27 Mar 2007 19:44:46`
	` +0000\n`
	`/awsexamplebucket/?acl`

CanonicalizedResource에 하위 리소스 쿼리 문자열 파라미터가 어떻게 포함되는지를 살펴보십시오.

삭제

이 예제는 경로 방식 및 Date 대체 헤더를 사용하여 'awsexamplebucket' 버킷에서 객체를 제거합니다.

요청	StringToSign

`DELETE /awsexamplebucket/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:lx3byBScXR6KzyMaifNkardMwNk=`
	`DELETE\n`
	`\n`
	`\n`
	`Tue, 27 Mar 2007 21:20:26 +0000\n`
	`/awsexamplebucket/photos/puppy.jpg`

클라이언트 라이브러리로 인해 날짜를 설정할 수 없어서 'x-amz-date' 메서드를 대신 사용하여 어떻게 날짜를 지정했는지 살펴보십시오. 이 경우 `x-amz-date`는 `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

`PUT /db-backup.dat.gz HTTP/1.1`
`User-Agent: curl/7.15.5`
`Host: static.awsexamplebucket.net: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@awsexamplebucket.net`
`X-Amz-Meta-ReviewedBy:`
` jane@awsexamplebucket.net`
`X-Amz-Meta-FileChecksum: 0x02661779`
	`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@awsexamplebucket.net,jane@awsexamplebucket.net`
	`\n`

요청	StringToSign

`X-Amz-Meta-ChecksumAlgorithm: crc32`
`Content-Disposition: attachment;`
` filename=database.dat`
`Content-Encoding: gzip`
`Content-Length: 5913339`
 	
`Authorization: AWS AKIAIOSFODNN7EXAMPLE:`
`ilyl83RwaSoYIEdixDQcA4OnAnc=`
	`/static.awsexamplebucket.net/db-`
	`backup.dat.gz`

'x-amz-' 헤더가 어떻게 정렬되고, 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`StringToSign`에 표시되는 것에 유의합니다. 다른 `Content-*` 엔터티 헤더는 표시되지 않습니다.

즉, `CanonicalizedResource`에는 버킷 이름이 포함되지만, HTTP 요청-URI에는 포함되지 않습니다. 버킷은 호스트 헤더가 지정합니다.

내 버킷 모두 나열하기

요청	StringToSign

`GET / HTTP/1.1`
`Host: s3.us-west-1.amazonaws.com`
`Date: Wed, 28 Mar 2007 01:29:59 +0000`
 	
`Authorization: AWS`
` AKIAIOSFODNN7EXAMPLE:qGdzdERIC03wnaRNKh6OqZehG9s=`
	`GET\n`
	`\n`
	`\n`
	`Wed, 28 Mar 2007 01:29:59`
	` +0000\n`
	`/`

유니코드 키

요청	StringToSign

`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:DNEZGsoieTZ92F3bUfSPQcbGmlM=`
	`GET\n`
	`\n`
	`\n`
	`Wed, 28 Mar 2007 01:49:49 +0000\n`
	`/dictionary/fran%C3%A7ais/pr%c3%a9f`
	`%c3%a8re`

> **Note**
>
> 요청-URI에서 파생된 `StringToSign`의 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 Ethereal 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 Amazon S3 비공개 데이터에 타사 브라우저가 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.

> **Note**
>
> AWS SDK를 사용하여 미리 서명된 URL을 생성하는 예제는 다른 사용자와 객체 공유 (p. 170) 단원을 참조하십시오.

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: awsexamplebucket.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	예제 값	설명
`AWSAccessKeyId`	`AKIAIOSFODNN7EXAMPLE`	AWS 액세스 키 ID. 요청에 서명하는 데 사용되는 AWS 보안 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;
```

`YourSecretAccessKey`는 Amazon 웹 서비스 개발자로 등록할 때 Amazon에서 할당한 AWS 보안 액세스 키 ID입니다. `Signature`가 쿼리 문자열에 배치될 수 있도록 URL 인코딩된 것에 유의하십시오. 또한 `StringToSign`에서 HTTP `Date` 위치 요소가 `Expires`로 교체되었음에 유의합니다. `CanonicalizedAmzHeaders`와 `CanonicalizedResource`는 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

`GET /photos/puppy.jpg?`
`AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&`
`  Signature=NpgCjnDzrM`
`%2BWFzoENXmpNDUsSn8%3D&`
`  Expires=1175139620 HTTP/1.1`
 	
`Host: awsexamplebucket.s3.us-`
`west-1.amazonaws.com`
	`GET\n`
	`\n`
	`\n`
	`1175139620\n`
	
	`/awsexamplebucket/photos/puppy.jpg`

브라우저가 GET 요청을 할 때 Content-MD5 또는 Content-Type 헤더를 제공하지 않으며 어떠한 x-amz- 헤더도 설정하지 않으며 따라서 `StringToSign`의 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.

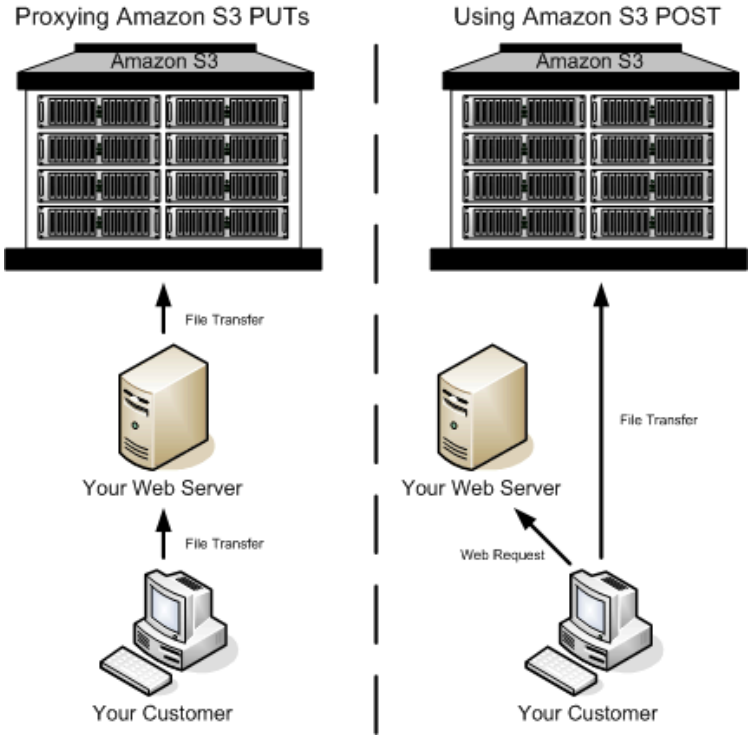
Base64 방식의 인코딩은 Amazon S3 인증 예제 (p. 749)를 참조하십시오.

POST를 사용한 브라우저 기반 업로드(AWS 서명 버전 2)

Amazon S3는 POST를 지원하므로 사용자가 곧바로 Amazon S3에 콘텐츠를 업로드할 수 있습니다. POST는 Amazon S3에 데이터를 업로드하여 저장하는 애플리케이션에서 업로드를 간소화하고 업로드 지연 시간을 줄이고 비용을 절약하도록 설계되었습니다.

> **Note**
>
> 이 단원에서 다루는 요청 인증은 AWS 서비스에 대한 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 Reference의 Authenticating Requests in Browser-Based Uploads Using POST (AWS Signature Version 4)를 참조하십시오.

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 (p. 756)
- HTML 양식 선언 (p. 756)
- HTML 양식 필드 (p. 757)
- 정책 구성 (p. 759)
- 서명 생성 (p. 762)
- 리디렉션 (p. 762)

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**
>
> HTTP를 통한 SOAP 지원은 중단되었지만 HTTPS를 통해 계속해서 사용할 수 있습니다. 새로운 Amazon S3 기능은 SOAP에 대해 지원되지 않습니다. 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 `awsexamplebucket`이고 리전이 미국 서부(캘리포니아 북부 지역)인 경우 URL은 `https://awsexamplebucket.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-data로 설정되어야 합니다. 자세한 내용은 RFC 1867를 참조하십시오.

Example

다음은 버킷 "awsexamplebucket"에 대한 양식 선언 예제입니다.

```
<form action="https://awsexamplebucket.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. ACL에 대한 자세한 내용은 액세스 통제 목록 (p. 7)을 참조하십시오. 유형: 문자열 기본: 프라이빗 Valid Values: `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 PUT Object 단원을 참조하십시오.	아니요
`key`	업로드된 키의 이름입니다. 사용자가 제공한 파일 이름을 사용하려면 ${filename} 변수를 사용하십시오. 예를 들어, 사용자 Betty가 파일 lolcatz.jpg을 업로드하고 경로가 /user/betty/${filename}으로 지정돼 있는 경우, 파일은 /user/betty/lolcatz.jpg로 저장됩니다. 자세한 내용은 객체 키와 메타데이터 (p. 102) 단원을 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 S3는 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 해당 필드는 무시됩니다. 업로드가 실패하는 경우 Amazon S3가 오류를 표시하고 사용자는 어떤 URL로도 리디렉션되지 않습니다. 자세한 내용은 리디렉션 (p. 762) 단원을 참조하십시오. Note 리디렉션 필드 이름을 더 이상 사용하지 않는 경우 해당 리디렉션 필드 이름에 대한 지원이 조만간 제거됩니다.	아니요
`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 않는 경우 Amazon S3은 상태 코드가 200 또는 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 AWSAccessKeyId에 해당하는 보안 액세스 키를 사용하여 생성하는 HMAC 서명입니다. 요청과 함께 정책 문서가 포함되는 경우 이 필드는 필수입니다. 자세한 내용은 Using Auth Access 단원을 참조하십시오.	조건
`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 증명 단원을 참조하십시오.	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

- 만료 (p. 760)
- 조건 (p. 760)
- 조건 매칭 (p. 761)
- 문자 이스케이프 (p. 762)

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": "awsexamplebucket" },

    ["starts-with", "$key", "user/eric/"],

  ]

}
```

정책 문서는 만료 및 조건을 포함합니다.

만료

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 포함돼야 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,Stallman`이라 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 (p. 761) 단원을 참조하십시오.

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.
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.

요소 이름	설명
	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 하여 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.
	
	`{"acl": "public-read" }`
	
	이 예제를 통해 ACL이 public-read로 설정되어야 함을 알 수 있습니다.
	
	`[ "eq", "$acl", "public-read" ]`

Starts With(다음으로 시작)	값이 특정 값으로 시작돼야 하는 경우 starts-with를 사용합니다. 이 예제는 키가 user/betty로 시작돼야 한다는 것을 나타냅니다. `["starts-with", "$key", "user/betty/"]`
내용 무관 매칭	필드에 들어 있는 모든 내용을 허용하도록 구성하려면 값이 비어 있는 starts-with를 사용합니다. 이 예제에서는 어떤 success_action_redirect도 허용됩니다. `["starts-with", "$success_action_redirect", ""]`

Condition	설명
범위 지정	범위를 허용하는 필드의 경우 상위 및 하위 범위를 쉼표로 구분하십시오. 이 예제에서는 1~10MB의 파일 크기가 허용됩니다. `["content-length-range", 1048579, 10485760]`

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
\u*xxxx*	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 정보는 Using Auth Access 단원을 참조하십시오.

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 (p. 763)
- 텍스트 영역 업로드 (p. 766)

> **Note**
>
> 이 단원에서 다루는 요청 인증은 AWS 서비스에 대한 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 Reference의 Examples: Browser-Based Upload using HTTP POST (Using AWS Signature Version 4) 단원을 참조하십시오.

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 awsexamplebucket 버킷의 Amazon S3에 대한 업로드를 지원합니다.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "awsexamplebucket"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "https://awsexamplebucket.s3.us-west-1.amazonaws.com/
successful_upload.html"},
    ["starts-with", "$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 awsexamplebucket 버킷에 업로드되어야 합니다.
- 키가 "user/eric/"로 시작해야 합니다.

- ACL이 public-read로 설정되었습니다.
- success_action_redirect가 https://awsexamplebucket.s3.us-west-1.amazonaws.com/successful_upload.html로 설정되어 있습니다.
- 객체가 이미지 파일입니다.
- x-amz-meta-uuid 태그가 14365123651274로 설정되어야 합니다.
- x-amz-meta-tag는 아무 값도 포함할 수 없습니다.

다음은 이 정책이 Base64로 인코딩된 버전입니다.

```
eyAiZXhwaXJhdGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWNrZXQiOiA
```

자격 증명을 사용하여 예컨대 앞서 다룬 정책 문서의 서명인 `0RavWzkygo6QX9caELEqKi9kDbU=`와 같은 서명을 만듭니다.

다음 양식은 이 정책을 사용하는 awsexamplebucket.net 버킷에 대한 POST 요청을 지원합니다.

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
  ...
  <form action="https://awsexamplebucket.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://
awsexamplebucket.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 />
    <input type="hidden" name="AWSAccessKeyId" value="AKIAIOSFODNN7EXAMPLE" />
    <input type="hidden" name="Policy" value="POLICY" />
    <input type="hidden" name="Signature" value="SIGNATURE" />
    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>
```

예제 요청

이 요청은 업로드된 이미지가 117,108바이트라 가정하며 이때 이미지 데이터는 포함되지 않습니다.

```
POST / HTTP/1.1
Host: awsexamplebucket.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://awsexamplebucket.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"

eyAiZXhwaXJhdGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWNrZXQiOiA
--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--
```

예제 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://awsexamplebucket.s3.us-west-1.amazonaws.com/
successful_upload.html?bucket=awsexamplebucket&key=user/eric/
MyPicture.jpg&etag=&quot;39d459dfbc0faabbb5e179358dfb94c3&quot;
Server: AmazonS3
```

텍스트 영역 업로드

주제

- 정책 및 양식 생성 (p. 766)
- 예제 요청 (p. 767)
- 예제 응답 (p. 768)

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 awsexamplebucket 버킷의 Amazon S3 대한 텍스트 영역 업로드를 지원합니다.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "awsexamplebucket"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "https://awsexamplebucket.s3.us-west-1.amazonaws.com/
new_post.html"},
    ["eq", "$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 awsexamplebucket 버킷에 업로드되어야 합니다.
- 키가 "user/eric/"로 시작해야 합니다.
- ACL이 public-read로 설정되었습니다.
- success_action_redirect가 https://awsexamplebucket.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.

다음은 이 정책이 Base64로 인코딩된 버전입니다.


```
eyAiZXhwaXJhdGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXR
pb25zIjogWwogICAgeyJidWNrZXQiOiAiam9obNtaXRoIn0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiRrZXkiLCAidXNlci9lcmljLy
LAogICAgeyJhY2wiOiAicHVibGljLXJlYWQifSwKICAgIHsic3VjY2Vzc19hY3Rpb25fcmVkaXJlY3QiOiAiaHR0cDovL2pvaG5zbWl
C5zMy5hbWF6b25hd3MuY29tL25ld19wb3N0Lmh0bWwifSwKICAgIFsiZXEiLCAiJENvbnRlbnQtVHlwZSIsICJ0ZXh0L2h0bWwiXSwK
CAgIHsieC1hbXotbWV0YS11dWlkIjogIjE0MzY1MTIzNjUxMjc0In0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiR4LWFtei1tZXRhLXRh
IsICIiXQogIF0KfQo=
```


자격 증명을 사용하여 서명을 만듭니다. 예를 들면, `qA7FWXKq6VvU68lI9KdveT1cWgF=`가 앞서 다룬 정책 문서의 서명입니다.

다음 양식은 이 정책을 사용하는 awsexamplebucket.net 버킷에 대한 POST 요청을 지원합니다.

```
<html>
```

```
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
  ...
  <form action="https://awsexamplebucket.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://
awsexamplebucket.s3.us-west-1.amazonaws.com/new_post.html" />
    <input type="hidden" name="Content-Type" value="text/html" />
    <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
    Tags for File: <input type="input" name="x-amz-meta-tag" value="" /><br />
    <input type="hidden" name="AWSAccessKeyId" value="AKIAIOSFODNN7EXAMPLE" />
    <input type="hidden" name="Policy" value="POLICY" />
    <input type="hidden" name="Signature" value="SIGNATURE" />
    Entry: <textarea name="file" cols="60" rows="10">

Your blog post goes here.

    </textarea><br />
    <!-- The elements after this will be ignored -->
    <input type="submit" name="submit" value="Upload to Amazon S3" />
  </form>
  ...
</html>
```

예제 요청

이 요청은 업로드된 이미지가 117,108바이트라 가정하며 이때 이미지 데이터는 포함되지 않습니다.

```
POST / HTTP/1.1
Host: awsexamplebucket.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://awsexamplebucket.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"
eyAiZXhwaXJhdGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWNrZXQiOiA

--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--
```

예제 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://awsexamplebucket.s3.us-west-1.amazonaws.com/new_post.html?
bucket=awsexamplebucket&key=user/eric/NewEntry.html&etag=40c3271af26b7f1672e41b8a274d28d4
Server: AmazonS3
```

Adobe Flash의 POST

이 단원에서는 Adobe Flash에서 `POST`를 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 Player을 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_status`를 201로 설정합니다. 그러면 Amazon S3에서 상태 코드가 201인 XML 문서를 반환합니다. XML 문서의 내용에 대한 자세한 내용은 POST Object 단원을 참조하십시오. 양식 필드에 대한 자세한 내용은 HTML 양식 필드 (p. 757) 단원을 참조하십시오.

Amazon S3 리소스

다음은 이 서비스를 이용할 때 유용하게 참조할 수 있는 관련 리소스가 나와 있는 표입니다.

리소스	설명
Amazon Simple Storage Service 시작 안내서	이 시작 안내서에서는 간단한 사용 사례를 기반으로 빠른 서비스 자습서를 제공합니다.
Amazon Simple Storage Service API Reference	이 API 참조에서는 Amazon S3 작업에 대해 자세히 설명합니다.
Amazon S3 기술 FAQ	FAQ는 이 제품에 대해 개발자들이 가장 많이 질문한 상위 개 내용을 소개합니다.
AWS Developer Resource Center	설명서, 코드 샘플, 릴리스 정보 및 AWS를 사용하여 혁신적인 애플리케이션을 구현하는 데 도움이 되는 기타 정보를 이 한 곳에서 찾을 수 있습니다.
AWS Management Console	이 콘솔을 사용하면 프로그래밍 없이 Amazon S3 기능 중 대부분을 수행할 수 있습니다.
https://forums.aws.amazon.com/	AWS와 관련된 기술적 질문에 대해 토론할 수 있는 개발자를 위한 커뮤니티 기반 포럼입니다.
AWS 지원 센터	AWS Technical Support 홈 페이지입니다. 이 페이지에서 개발자 포럼, 기술 FAQ, 서비스 상태 페이지, Premium Support에 액세스할 수 있습니다.
AWS Premium Support	AWS Premium Support에 대한 기본 웹 페이지입니다. 일대일의 신속한 응대 지원 채널을 통해 AWS Infrastructure Services에서 애플리케이션을 구축하고 실행하도록 지원합니다.
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.

Amazon S3 Select와 S3 Glacier Select의 SQL 참조

이 참조는 Amazon S3 Select와 S3 Glacier Select에서 지원하는 구조화 질의 언어(SQL) 요소에 대한 설명을 포함합니다.

주제

- SELECT 명령 (p. 771)
- 데이터 형식 (p. 776)
- 연산자 (p. 777)
- 예약어 (p. 779)
- SQL 함수 (p. 782)

SELECT 명령

Amazon S3 Select와 S3 Glacier Select는 `SELECT` SQL 명령만 지원합니다. `SELECT`에서 지원되는 ANSI 표준 절은 다음과 같습니다.

- `SELECT` 계정의
- `FROM` 절
- `WHERE` 절
- `LIMIT` 절(Amazon S3 Select만 해당)

> **Note**
>
> Amazon S3 Select와 S3 Glacier Select 쿼리는 현재 하위 쿼리 또는 조인을 지원하지 않습니다.

목록 SELECT

`SELECT` 목록은 쿼리에서 반환하려는 열, 함수 및 표현식의 이름을 지정합니다. 목록은 쿼리의 출력을 나타냅니다.

```
SELECT *
SELECT projection [ AS column_alias | column_alias ] [, ...]
```

*(별표)가 있는 첫 번째 형식은 `WHERE` 절을 통과한 모든 행을 그대로 반환합니다. 두 번째 형식은 각 열마다 사용자 정의 출력 스칼라 표현식 투영이 포함된 행을 생성합니다.

FROM 절

Amazon S3 Select와 S3 Glacier Select는 다음 형식의 `FROM` 절을 지원합니다.

```
FROM table_name
```

```
FROM table_name alias
FROM table_name AS alias
```

여기서 `table_name`은 쿼리되는 아카이브를 참조하는 `S3Object`(Amazon S3 Select의 경우) 또는 `ARCHIVE` 또는 `OBJECT`(S3 Glacier Select의 경우) 중 하나입니다. 기존의 관계형 데이터베이스를 사용해오던 사용자의 경우, 이것을 단일 테이블에 대해 다수의 보기를 포함하는 데이터베이스 스키마로 생각할 수 있습니다.

표준 SQL에 따라 `FROM` 절은 `WHERE` 절에서 필터링되고 `SELECT` 목록에서 예상되는 행을 생성합니다.

Amazon S3 Select에 저장된 JSON 객체에 대해 다음과 같은 `FROM` 절 양식도 사용할 수 있습니다.

```
FROM S3Object[*].path
FROM S3Object[*].path alias
FROM S3Object[*].path AS alias
```

이 양식의 `FROM` 절을 사용하면 JSON 객체 내에서 배열 또는 객체를 선택할 수 있습니다. 다음 양식 중 하나를 사용하여 `path`를 지정할 수 있습니다.

- 이름별(객체 내): `.name` 또는 `['name']`
- 인덱스로(배열 내): `[index]`
- 와일드카드(객체 내): `.*`
- 와일드카드(배열 내): `[*]`

Note

- 이 양식의 `FROM` 절은 JSON 객체에만 사용할 수 있습니다.
- 와일드카드는 항상 하나 이상의 레코드를 출력합니다. 일치하는 레코드가 없을 경우 Amazon S3 Select가 `MISSING` 값을 출력합니다. 출력 직렬화 도중(쿼리 완료 후), Amazon S3 Select가 `MISSING` 값을 빈 레코드로 대체합니다.
- 집계 함수(`AVG`, `COUNT`, `MAX`, `MIN` 및 `SUM`)는 `MISSING` 값을 건너뜁니다.
- 와일드카드를 사용할 때 별칭을 제공하지 않는 경우 경로의 마지막 요소를 사용하여 행을 참조할 수 있습니다. 예를 들어 `SELECT price FROM S3Object[*].books[*].price` 쿼리를 사용하여 서적 목록에서 모든 가격을 선택할 수 있습니다. 경로가 이름이 아니라 와일드카드로 끝날 경우 값 `_1`을 사용하여 행을 참조할 수 있습니다. 예를 들면 `SELECT price FROM S3Object[*].books[*].price` 대신 `SELECT _1.price FROM S3Object[*].books[*]`를 사용할 수 있습니다.
- Amazon S3 Select는 항상 JSON 문서를 루트 레벨 값의 배열로 취급합니다. 따라서 쿼리하는 JSON 객체가 하나의 루트 요소만 가지더라도 `FROM` 절이 `S3Object[*]`로 시작해야 합니다. 하지만 호환성 때문에 Amazon S3 Select는 사용자가 경로를 포함하지 않을 경우 와일드카드를 생략하도록 허용합니다. 따라서 전체 절 `FROM S3Object`는 `FROM S3Object[*] as S3Object`와 같습니다. 경로를 포함하는 경우에는 와일드카드도 사용해야 합니다. 그러므로 `FROM S3Object` 및 `FROM S3Object[*].path`는 모두 유효한 절이지만, `FROM S3Object.path`는 유효하지 않습니다.

Example

예제:

예제 #1

이 예제에서는 다음 데이터 세트 및 쿼리를 사용한 결과를 보여 줍니다.

```
{ "Rules": [ {"id": "1"}, {"expr": "y > x"}, {"id": "2", "expr": "z = DEBUG"} ]}
```

```
{ "created": "June 27", "modified": "July 6" }
```

```
SELECT id FROM S3Object[*].Rules[*].id
```

```
{"id":"1"}
{}
{"id":"2"}
{}
```

Amazon S3 Select가 각 결과를 반환한 이유는 다음과 같습니다.

- {"id":"id-1"} – S3Object0.Rules0.id는 일치하는 레코드가 1개 있습니다.
- {} – S3Object0.Rules1.id는 일치하는 레코드가 없으므로 Amazon S3 Select가 `MISSING`을 출력했습니다. 이 값은 출력 직렬화 과정에서 빈 레코드로 변환 후 반환되었습니다.
- {"id":"id-2"} – S3Object0.Rules2.id는 일치하는 레코드가 1개 있습니다.
- {} – S3Object1는 `Rules`에서 일치하는 레코드가 없으므로 Amazon S3 Select가 `MISSING`을 출력했습니다. 이 값은 출력 직렬화 과정에서 빈 레코드로 변환 후 반환되었습니다.

Amazon S3 Select가 일치 레코드를 찾지 못할 경우 빈 레코드를 반환하지 않게 하려면 값 `MISSING`에 대해 테스트할 수 있습니다. 다음 쿼리는 이전 쿼리와 동일한 결과를 반환하지만 빈 값이 생략되었습니다.

```
SELECT id FROM S3Object[*].Rules[*].id WHERE id IS NOT MISSING
```

```
{"id":"1"}
{"id":"2"}
```

예제 #2

이 예제에서는 다음 데이터 세트 및 쿼리를 사용한 결과를 보여 줍니다.

```
{ "created": "936864000", "dir_name": "important_docs", "files": [ { "name": "." },
 { "name": ".." }, { "name": ".aws" }, { "name": "downloads" } ], "owner": "AWS S3" }
{ "created": "936864000", "dir_name": "other_docs", "files": [ { "name": "." }, { "name":
 ".." }, { "name": "my stuff" }, { "name": "backup" } ], "owner": "User" }
```

```
SELECT d.dir_name, d.files FROM S3Object[*] d
```

```
{"dir_name":"important_docs","files":[{"name":"."},{"name":".."},{"name":".aws"},
{"name":"downloads"}]}
{"dir_name":"other_docs","files":[{"name":"."},{"name":".."},{"name":"my stuff"},
{"name":"backup"}]}
```

```
SELECT _1.dir_name, _1.owner FROM S3Object[*]
```

```
{"dir_name":"important_docs","owner":"AWS S3"}
{"dir_name":"other_docs","owner":"User"}
```

WHERE 절

`WHERE` 절은 다음 구문에 따릅니다.

```
WHERE condition
```

WHERE 절은 조건에 기반하여 행을 필터링합니다. 조건은 부울 결과가 있는 표현식입니다. 조건이 TRUE로 평가되는 행만이 결과에서 반환됩니다.

LIMIT 절(Amazon S3 Select만 해당)

LIMIT 절은 다음 구문에 따릅니다.

```
LIMIT number
```

LIMIT 절은 쿼리가 반환하게 할 레코드의 수를 숫자에 기초하여 제한합니다.

> **Note**
>
> S3 Glacier Select는 LIMIT 절을 지원하지 않습니다.

속성 액세스

SELECT와 WHERE 절은 쿼리되는 파일이 CSV 형식인지 JSON 형식인지에 따라 다음 단원에서 소개하는 메서드 중 하나를 사용하여 레코드 데이터를 참조할 수 있습니다.

CSV

- 열 번호 – 열 이름 _N으로 특정 행의 N번째 열을 참조할 수 있습니다(여기서 N은 열의 위치입니다). 위치 카운트는 1에서 시작합니다. 예를 들어, 첫 번째 열의 이름은 _1이고, 두 번째 열의 이름은 _2입니다.

  열을 _N 또는 alias._N이라고 부를 수 있습니다. 예를 들어, _2와 myAlias._2는 모두 SELECT 목록과 WHERE 절에 있는 열을 참조하는 유효한 방법입니다.
- 열 머리글 – 머리글이 있는 CSV 형식의 객체들은 그 머리글이 SELECT 목록과 WHERE 절에 제공됩니다. 특히 기존의 SQL과 같이, SELECT와 WHERE 절 표현식에서 alias.column_name 또는 column_name으로 열을 참조할 수 있습니다.

JSON(Amazon S3 Select만 해당)

- 문서 – alias.name으로 JSON 문서 필드에 액세스할 수 있습니다. 중첩 필드에도 액세스할 수 있습니다(예: alias.name1.name2.name3).
- 목록 – [] 연산자가 있는 0 기반 인덱스를 사용하여 JSON 목록의 요소들에 액세스할 수 있습니다. 예를 들어, 목록의 두 번째 요소에 alias[1]로 액세스할 수 있습니다. 목록 요소에 대한 액세스를 alias.name1.name2[1].name3의 필드와 결합할 수 있습니다.
- 예제: 이 JSON 객체를 샘플 데이터 세트로 간주합니다.

```
{"name": "Susan Smith",
"org": "engineering",
"projects":
    [
     {"project_name":"project1", "completed":false},
     {"project_name":"project2", "completed":true}
    ]
}
```

예제 #1

다음 쿼리는 다음과 같은 결과를 반환합니다.

```
Select s.name from S3Object s
```

```
{"name":"Susan Smith"}
```

예제 #2

다음 쿼리는 다음과 같은 결과를 반환합니다.

```
Select s.projects[0].project_name from S3Object s
```

```
{"project_name":"project1"}
```

머리글/속성 이름의 대/소문자 구분

Amazon S3 Select와 S3 Glacier Select에서 큰 따옴표를 사용하여 열 머리글(CSV 객체)과 속성(JSON 객체)이 대/소문자를 구분한다는 것을 표시할 수 있습니다. 큰 따옴표가 없으면 객체 머리글/속성이 대/소문자를 구분하지 않는 것입니다. 모호한 경우에는 오류가 발생합니다.

다음 예제는 1) 열 머리글이 지정되어 있고 쿼리 요청에 대해 `FileHeaderInfo`가 "Use(사용)"로 설정되어 있는 CSV 형식의 하나의 Amazon S3 또는 S3 Glacier 객체, 또는 2) 속성이 지정된 JSON 형식의 하나의 Amazon S3 객체입니다.

예제 #1: 쿼리되는 객체가 머리글/속성 "NAME"을 갖습니다.

- 다음 표현식은 객체에서 값을 성공적으로 반환합니다(따옴표 없음: 대/소문자를 구분하지 않음).

```
SELECT s.name from S3Object s
```

- 다음 표현식의 결과는 400 오류 `MissingHeaderName`입니다(따옴표: 대/소문자를 구분함).

```
SELECT s."name" from S3Object s
```

예제 #2: 쿼리되는 Amazon S3 객체가 "NAME"이라는 한 머리글/속성과 "name"이라는 또 다른 머리글/속성을 갖습니다.

- 다음 표현식의 결과는 400 오류 `AmbiguousFieldName`입니다(따옴표 없음: 대/소문자를 구분하지 않으나 일치하는 것이 2개가 있음).

```
SELECT s.name from S3Object s
```

- 다음 표현식은 객체에서 값을 성공적으로 반환합니다(따옴표: 대/소문자를 구분하므로 모호함이 해결됨).

```
SELECT s."NAME" from S3Object s
```

예약어를 사용자 정의 용어로 사용

Amazon S3 Select와 S3 Glacier Select에는 객체 콘텐츠를 쿼리하는 데 사용되는 SQL 표현식을 실행하기 위해 필요한 일련의 예약어가 있습니다. 예약어에는 함수 이름, 데이터 유형, 연산자 등이 포함됩니다. 어떤 경우에는 열 머리글(CSV 파일의 경우)이나 속성(JSON 객체의 경우) 같은 사용자 정의 용어가 예약어와 충

돌할 수가 있습니다. 이 경우에는 예약어와 충돌하는 사용자 정의 용어를 일부러 사용하고 있다는 것을 큰 따옴표를 사용하여 표시해야 합니다. 이렇게 하지 않으면 400 구문 분석 오류가 발생할 것입니다.

예약어의 전체 목록은 예약어 (p. 779) 단원을 참조하십시오.

다음의 예제는 1) 열 머리글이 지정되어 있고 쿼리 요청에 대해 `FileHeaderInfo`가 "Use(사용)"로 설정되어 있는 CSV 형식의 하나의 Amazon S3 또는 S3 Glacier 객체, 또는 2) 속성이 지정된 JSON 형식의 하나의 Amazon S3 객체입니다.

예제: 쿼리되는 객체가 "CAST"(예약어임)라는 머리글/속성을 갖습니다.

- 다음 표현식은 객체에서 값을 성공적으로 반환합니다(따옴표: 사용자 정의 머리글/속성을 사용함).

```
SELECT s."CAST" from S3Object s
```

- 다음 표현식의 결과는 400 구문 분석 오류입니다(따옴표 없음: 예약어와 충돌함).

```
SELECT s.CAST from S3Object s
```

스칼라 표현식

`WHERE` 절과 `SELECT` 목록 내에서 스칼라 값을 반환하는 표현식인 SQL 스칼라 표현식을 사용할 수 있습니다. 스칼라 표현식의 형식은 다음과 같습니다.

- literal

  SQL 리터럴.
- column_reference

  column_name 또는 alias.column_name 형식의 열에 대한 참조.
- unary_op 표현식

  여기서 unary_op 단항은 SQL 단항 연산자입니다.
- 표현식 binary_op 표현식

  여기서 binary_op는 SQL 이항 연산자입니다.
- func_name

  여기서 func_name은 호출할 스칼라 함수의 이름입니다.
- 표현식 `[ NOT ] BETWEEN` 표현식 `AND` 표현식
- 표현식 `LIKE` 표현식 [`ESCAPE` 표현식]

데이터 형식

Amazon S3 Select와 S3 Glacier Select는 몇 가지 기본 데이터 형식을 지원합니다.

데이터 형식 변환

일반적 규칙은 `CAST` 함수에 따르는 것입니다(정의된 경우). `CAST`를 정의하지 않는 경우, 모든 입력 데이터를 문자열로 취급합니다. 필요할 경우, 관련 데이터 형식으로 캐스팅되어야 합니다.

`CAST` 함수에 대한 자세한 내용은 CAST (p. 785) 단원을 참조하십시오.

지원되는 데이터 형식

Amazon S3 Select와 S3 Glacier Select는 다음과 같은 기본 데이터 형식 하위 집합을 지원합니다.

이름	설명	예제
bool	TRUE 또는 FALSE	`FALSE`
int, integer	-9,223,372,036,854,775,808부터 9,223,372,036,854,775,807 사이의 8바이트 정수입니다.	`100000`
string	UTF8로 인코딩된 가변 길이 문자열입니다. 기본 한도는 1자입니다. 최대 문자 한도는 2,147,483,647입니다.	`'xyz'`
float	8바이트 부동 소수점 숫자입니다.	`CAST(0.456 AS FLOAT)`
decimal, numeric	최대 정밀도(최대 유효 숫자 수)가 38이고, 스케일 범위가 -2^{31}~2^{31}-1(십진수 지수)인 10진수.	`123.456`
타임스탬프	타임스탬프는 시간상 특정 순간을 나타내며, 항상 로컬 오프셋을 포함하고, 임의 정밀도가 가능합니다. 텍스트 형식에서 타임스탬프는 날짜 및 시간 형식에 대한 W3C 노트를 따르지만 적어도 종일 정밀도가 아니라면 리터럴 "T"로 끝나야 합니다. 소수부 초는 한 자리 이상의 정밀도와 무한대의 최대값을 포함하여 허용됩니다. 로컬 시간 오프셋은 UTC로부터의 hour:minute 오프셋이나 UTC의 로컬 시간을 가리키는 리터럴 "Z"로 나타낼 수 있습니다. 로컬 시간 오프셋은 시간이 포함된 타임스탬프에서 필수이며, 날짜 값에서는 허용되지 않습니다.	`CAST('2007-04-05T14:30Z' AS TIMESTAMP)`

연산자

Amazon S3 Select와 S3 Glacier Select는 다음 연산자만 지원합니다.

논리 연산자

- `AND`
- `NOT`
- `OR`

비교 연산자

- `<`
- `>`
- `<=`
- `>=`
- `=`
- `<>`
- `!=`

- `BETWEEN`
- `IN` – 예: `IN ('a', 'b', 'c')`

패턴 일치 연산자

- `LIKE`
- _(모든 문자를 일치시킵니다.)
- %(모든 문자 시퀀스를 일치시킵니다.)

단일 연산자

- IS NULL
- IS NOT NULL

수학 연산자

더하기, 빼기, 곱하기, 나누기 및 모듈로가 지원됩니다.

- +
- -
- *
- %

연산자 우선 순위

다음 표에서는 연산자들의 우선 순위를 내림차순으로 보여줍니다.

연산자/원소	연결성	필수
-	오른쪽	단항 마이너스
*, /, %	왼쪽	곱하기, 나누기, 모듈로
+, -	왼쪽	더하기, 빼기
IN		집합 소속
BETWEEN		범위 제한
LIKE		문자열 패턴 일치
<>		보다 작음, 보다 큼
=	오른쪽	같음, 대입
NOT	오른쪽	논리 부정
AND	왼쪽	논리 결합

연산자/원소	연결성	필수
또는	왼쪽	논리 분리

예약어

Amazon S3 Select와 S3 Glacier Select에서 사용되는 예약어들을 아래에 나열하였습니다. 여기에는 객체 콘텐츠를 쿼리할 때 사용되는 SQL 표현식을 실행하는 데 필요한 함수 이름, 데이터 형식, 연산자 등이 포함되어 있습니다.

```
absolute
action
add
all
allocate
alter
and
any
are
as
asc
assertion
at
authorization
avg
bag
begin
between
bit
bit_length
blob
bool
boolean
both
by
cascade
cascaded
case
cast
catalog
char
char_length
character
character_length
check
clob
close
coalesce
collate
collation
column
commit
connect
connection
constraint
constraints
continue
convert
corresponding
count
create
```

```
cross
current
current_date
current_time
current_timestamp
current_user
cursor
date
day
deallocate
dec
decimal
declare
default
deferrable
deferred
delete
desc
describe
descriptor
diagnostics
disconnect
distinct
domain
double
drop
else
end
end-exec
escape
except
exception
exec
execute
exists
external
extract
false
fetch
first
float
for
foreign
found
from
full
get
global
go
goto
grant
group
having
hour
identity
immediate
in
indicator
initially
inner
input
insensitive
insert
int
integer
intersect
```

```
interval
into
is
isolation
join
key
language
last
leading
left
level
like
limit
list
local
lower
match
max
min
minute
missing
module
month
names
national
natural
nchar
next
no
not
null
nullif
numeric
octet_length
of
on
only
open
option
or
order
outer
output
overlaps
pad
partial
pivot
position
precision
prepare
preserve
primary
prior
privileges
procedure
public
read
real
references
relative
restrict
revoke
right
rollback
rows
schema
```

```
scroll
second
section
select
session
session_user
set
sexp
size
smallint
some
space
sql
sqlcode
sqlerror
sqlstate
string
struct
substring
sum
symbol
system_user
table
temporary
then
time
timestamp
timezone_hour
timezone_minute
to
trailing
transaction
translate
translation
trim
true
tuple
union
unique
unknown
unpivot
update
upper
usage
user
using
value
values
varchar
varying
view
when
whenever
where
with
work
write
year
zone
```

SQL 함수

Amazon S3 Select와 S3 Glacier Select는 몇 가지 SQL 함수만 지원합니다.

주제

- 집계 함수(Amazon S3 Select만 해당) (p. 783)
- 조건 함수 (p. 783)
- 변환 함수 (p. 784)
- Date 함수 (p. 785)
- 문자열 함수 (p. 790)

집계 함수(Amazon S3 Select만 해당)

Amazon S3 Select는 다음과 같은 집계 함수를 지원합니다.

> **Note**
>
> S3 Glacier Select는 집계 함수를 지원하지 않습니다.

기능	인수 형식	반환 유형
`AVG(expression)`	INT, FLOAT, DECIMAL	INT 인수의 경우 DECIMAL, 부동 소수점 인수의 경우 FLOAT, 그 밖의 경우에는 인수 데이터 형식과 동일합니다.
`COUNT`	-	INT
`MAX(expression)`	INT, DECIMAL	인수 형식과 동일합니다.
`MIN(expression)`	INT, DECIMAL	인수 형식과 동일합니다.
`SUM(expression)`	INT, FLOAT, DOUBLE, DECIMAL	INT 인수의 경우 INT, 부동 소수점 인수의 경우 FLOAT, 그 밖의 경우에는 인수 데이터 형식과 동일합니다.

조건 함수

Amazon S3 Select와 S3 Glacier Select는 다음과 같은 조건 함수를 지원합니다.

주제

- COALESCE (p. 783)
- NULLIF (p. 784)

COALESCE

인수들을 순서대로 평가하여 알 수 없는 인수가 아닌 첫 번째 인수, 즉 null이 아니거나 누락되지 않은 첫 번째 인수를 반환합니다. 이 함수는 null과 누락을 전파하지 않습니다.

구문

```
COALESCE ( expression, expression, ... )
```

파라미터

표현식

함수가 실행되는 대상 표현식입니다.

예제

```
COALESCE(1)                -- 1
COALESCE(null)             -- null
COALESCE(null, null)       -- null
COALESCE(missing)          -- null
COALESCE(missing, missing) -- null
COALESCE(1, null)          -- 1
COALESCE(null, null, 1)    -- 1
COALESCE(null, 'string')   -- 'string'
COALESCE(missing, 1)       -- 1
```

NULLIF

두 개의 표현식이 주어질 경우 그 두 표현식이 같은 값으로 평가된다면 NULL을 반환하고, 그렇지 않다면 첫 번째 표현식을 평가한 결과를 반환합니다.

구문

```
NULLIF ( expression1, expression2 )
```

파라미터

expression1, expression2

함수가 실행되는 대상 표현식들입니다.

예제

```
NULLIF(1, 1)             -- null
NULLIF(1, 2)             -- 1
NULLIF(1.0, 1)           -- null
NULLIF(1, '1')           -- 1
NULLIF([1], [1])         -- null
NULLIF(1, NULL)          -- 1
NULLIF(NULL, 1)          -- null
NULLIF(null, null)       -- null
NULLIF(missing, null)    -- null
NULLIF(missing, missing) -- null
```

변환 함수

Amazon S3 Select와 S3 Glacier Select는 다음과 같은 변환 함수를 지원합니다.

주제

- CAST (p. 785)

CAST

`CAST` 함수는 단일 값으로 평가되는 표현식 같은 엔터티의 형식을 다른 형식으로 변환합니다.

구문

```
CAST ( expression AS data_type )
```

파라미터

표현식

하나 이상의 값, 연산자 및 값으로 평가되는 SQL 함수의 조합입니다.

data_type

`INT`와 같이 표현식을 캐스팅할 대상 데이터 형식입니다. 지원되는 데이터 형식의 목록은 데이터 형식 (p. 776) 단원을 참조하십시오.

예제

```
CAST('2007-04-05T14:30Z' AS TIMESTAMP)
CAST(0.456 AS FLOAT)
```

Date 함수

Amazon S3 Select와 S3 Glacier Select는 다음과 같은 날짜 함수를 지원합니다.

주제

- DATE_ADD (p. 785)
- DATE_DIFF (p. 786)
- EXTRACT (p. 787)
- TO_STRING (p. 787)
- TO_TIMESTAMP (p. 790)
- UTCNOW (p. 790)

DATE_ADD

날짜 부분, 양, 그리고 타임스탬프가 주어질 경우 그 날짜 부분을 그 수량으로 변환하여 업데이트된 타임스탬프를 반환합니다.

구문

```
DATE_ADD( date_part, quantity, timestamp )
```

파라미터

date_part

날짜의 어느 부분을 수정할지 지정합니다. 이것은 다음 중 하나가 될 수 있습니다.

- 년
- 개월
- day
- 시간
- minute
- 초

양

업데이트된 타임스탬프에 적용되는 값입니다. 이 양이 양의 값이면 타임스탬프의 날짜 부분이 증가하고 음의 값이면 감소합니다.

타임스탬프

함수가 실행되는 대상 타임스탬프입니다.

예제

```
DATE_ADD(year, 5, `2010-01-01T`)                -- 2015-01-01 (equivalent to 2015-01-01T)
DATE_ADD(month, 1, `2010T`)                     -- 2010-02T (result will add precision as
 necessary)
DATE_ADD(month, 13, `2010T`)                    -- 2011-02T
DATE_ADD(day, -1, `2017-01-10T`)                -- 2017-01-09 (equivalent to 2017-01-09T)
DATE_ADD(hour, 1, `2017T`)                      -- 2017-01-01T01:00-00:00
DATE_ADD(hour, 1, `2017-01-02T03:04Z`)          -- 2017-01-02T04:04Z
DATE_ADD(minute, 1, `2017-01-02T03:04:05.006Z`) -- 2017-01-02T03:05:05.006Z
DATE_ADD(second, 1, `2017-01-02T03:04:05.006Z`) -- 2017-01-02T03:04:06.006Z
```

DATE_DIFF

날짜 부분 1개와 유효한 타임스탬프 2개가 주어질 경우, 날짜 부분들의 차이점을 반환합니다. `date_part`의 `timestamp1` 값이 `date_part`의 `timestamp2` 값보다 더 클 경우 반환 값은 음의 정수입니다. `date_part`의 `timestamp1` 값이 `date_part`의 `timestamp2` 값보다 더 작을 경우 반환 값은 양의 정수입니다.

구문

```
DATE_DIFF( date_part, timestamp1, timestamp2 )
```

파라미터

date_part

타임스탬프들의 어느 부분을 비교할지 지정합니다. `date_part`의 정의에 대해서는 DATE_ADD (p. 785) 단원을 참조하십시오.

timestamp1

비교할 첫 번째 타임스탬프입니다.

timestamp2

비교할 두 번째 타임스탬프입니다.

예제

```
DATE_DIFF(year, `2010-01-01T`, `2011-01-01T`)            -- 1
DATE_DIFF(year, `2010T`, `2010-05T`)                     -- 4 (2010T is equivalent to
 2010-01-01T00:00:00.000Z)
```

```
DATE_DIFF(month, `2010T`, `2011T`)                        -- 12
DATE_DIFF(month, `2011T`, `2010T`)                        -- -12
DATE_DIFF(day, `2010-01-01T23:00T`, `2010-01-02T01:00T`) -- 0 (need to be at least 24h
 apart to be 1 day apart)
```

EXTRACT

날짜 부분 1개와 타임스탬프 1개가 주어질 경우, 타임스탬프의 날짜 부분 값을 반환합니다.

구문

```
EXTRACT( date_part FROM timestamp )
```

파라미터

date_part

타임스탬프들의 어느 부분을 추출할지 지정합니다. 이것은 다음 중 하나가 될 수 있습니다.

- 년
- 개월
- day
- 시간
- minute
- 초
- timezone_hour
- timezone_minute

타임스탬프

함수가 실행되는 대상 타임스탬프입니다.

예제

```
EXTRACT(YEAR FROM `2010-01-01T`)                          -- 2010
EXTRACT(MONTH FROM `2010T`)                               -- 1 (equivalent to
 2010-01-01T00:00:00.000Z)
EXTRACT(MONTH FROM `2010-10T`)                            -- 10
EXTRACT(HOUR FROM `2017-01-02T03:04:05+07:08`)            -- 3
EXTRACT(MINUTE FROM `2017-01-02T03:04:05+07:08`)          -- 4
EXTRACT(TIMEZONE_HOUR FROM `2017-01-02T03:04:05+07:08`)   -- 7
EXTRACT(TIMEZONE_MINUTE FROM `2017-01-02T03:04:05+07:08`) -- 8
```

TO_STRING

타임스탬프와 형식 패턴이 주어질 경우, 타임스탬프의 문자열 표현을 그 형식으로 반환합니다.

구문

```
TO_STRING ( timestamp time_format_pattern )
```

파라미터

타임스탬프

함수가 실행되는 대상 타임스탬프입니다.

time_format_pattern

문자에 대해 다음과 같이 특수하게 해석하는 문자열입니다.

형식	예	설명
yy	69	2자리 연도
y	1969	4자리 연도
yyyy	1969	제로 패딩된 4자리 연도
M	1	연 기준 월
MM	01	제로 패딩된 연 기준 월
MMM	Jan	약어 형태의 월 이름
MMMM	January	월의 전체 이름
MMMMM	J	월의 첫 번째 글자(참고: to_timestamp 함수와 사용할 경우 유효하지 않음)
d	2	일(1~31)
dd	02	제로 패딩된 일(01~31)
a	AM	하루 중 AM 또는 PM
h	3	시간(1~12)
hh	03	제로 패딩된 시간(01~12)
H	3	시간(0~23)
HH	03	제로 패딩된 시간(00~23)
m	4	분(0~59)
mm	04	제로 패딩된 분(00~59)
s	5	초(0~59)
ss	05	제로 패딩된 초(00~59)
S	0	몇 분의 1초(정밀도: 0.1, 범위: 0.0~0.9)

형식	예	설명
SS	6	몇 분의 1초(정밀도: 0.01, 범위: 0.0~0.99)
SSS	60	몇 분의 1초(정밀도: 0.001, 범위: 0.0~0.999)
…	…	…
SSSSSSSSS	60000000	몇 분의 1초(최대 정밀도: 1 나노초, 범위: 0.0~0.999999999)
n	60000000	나노초
X	+07 or Z	시간 단위로 표시한 오프셋. 오프셋이 0인 경우 "Z"
XX or XXXX	+0700 or Z	시간과 분 단위로 표시한 오프셋. 오프셋이 0인 경우 "Z"
XXX or XXXXX	+07:00 or Z	시간과 분 단위로 표시한 오프셋. 오프셋이 0인 경우 "Z"
x	7	시간 단위로 표시한 오프셋
xx or xxxx	700	시간과 분 단위로 표시한 오프셋
xxx or xxxxx	+07:00	시간과 분 단위로 표시한 오프셋

예제

```
TO_STRING(`1969-07-20T20:18Z`,  'MMMM d, y')                    -- "July 20, 1969"
TO_STRING(`1969-07-20T20:18Z`, 'MMM d, yyyy')                   -- "Jul 20, 1969"
TO_STRING(`1969-07-20T20:18Z`, 'M-d-yy')                        -- "7-20-69"
TO_STRING(`1969-07-20T20:18Z`, 'MM-d-y')                        -- "07-20-1969"
TO_STRING(`1969-07-20T20:18Z`, 'MMMM d, y h:m a')               -- "July 20, 1969 8:18 PM"
TO_STRING(`1969-07-20T20:18Z`, 'y-MM-dd''T''H:m:ssX')           -- "1969-07-20T20:18:00Z"
TO_STRING(`1969-07-20T20:18+08:00Z`, 'y-MM-dd''T''H:m:ssX')     -- "1969-07-20T20:18:00Z"
TO_STRING(`1969-07-20T20:18+08:00`, 'y-MM-dd''T''H:m:ssXXXX')   --
 "1969-07-20T20:18:00+0800"
TO_STRING(`1969-07-20T20:18+08:00`, 'y-MM-dd''T''H:m:ssXXXXX')  --
 "1969-07-20T20:18:00+08:00"
```

TO_TIMESTAMP

문자열이 주어질 경우, 이것을 타임스탬프로 변환합니다. 이것은 TO_STRING의 역연산입니다.

구문

```
TO_TIMESTAMP ( string )
```

파라미터

string

함수가 실행되는 대상 문자열입니다.

예제

```
TO_TIMESTAMP('2007T')                          -- `2007T`
TO_TIMESTAMP('2007-02-23T12:14:33.079-08:00') -- `2007-02-23T12:14:33.079-08:00`
```

UTCNOW

현재 시간(UTC)을 타임스탬프로 반환합니다.

구문

```
UTCNOW()
```

파라미터

none

예제

```
UTCNOW() -- 2017-10-13T16:02:11.123Z
```

문자열 함수

Amazon S3 Select와 S3 Glacier Select는 다음과 같은 문자열 함수를 지원합니다.

주제

- CHAR_LENGTH, CHARACTER_LENGTH (p. 790)
- LOWER (p. 791)
- SUBSTRING (p. 791)
- TRIM (p. 792)
- UPPER (p. 792)

CHAR_LENGTH, CHARACTER_LENGTH

지정된 문자열의 문자 수를 계수합니다.

Note

`CHAR_LENGTH`와 `CHARACTER_LENGTH`는 동의어입니다.

구문

```
CHAR_LENGTH ( string )
```

파라미터

string

함수가 실행되는 대상 문자열입니다.

예제

```
CHAR_LENGTH('')          -- 0
CHAR_LENGTH('abcdefg')   -- 7
```

LOWER

문자열이 주어지면 모든 대문자를 소문자로 변환합니다. 대문자로 변환되지 않는 문자는 변경되지 않고 그대로 유지됩니다.

구문

```
LOWER ( string )
```

파라미터

string

함수가 실행되는 대상 문자열입니다.

예제

```
LOWER('AbCdEfG!@#$') -- 'abcdefg!@#$'
```

SUBSTRING

문자열, 시작 인덱스, 그리고 때로는 길이가 주어질 경우 시작 인덱스부터 그 문자열의 마지막까지, 또는 제공된 길이까지 하위 문자열을 반환합니다.

Note

입력 문자열의 첫 번째 문자가 인덱스 1을 갖습니다. `start`가 < 1인 경우, 이것은 1로 설정됩니다.

구문

```
SUBSTRING( string FROM start [ FOR length ] )
```

파라미터

string

함수가 실행되는 대상 문자열입니다.

start

문자열의 시작 위치입니다.

length

반환할 하위 문자열의 길이입니다. 존재하지 않는 경우, 문자열의 끝으로 넘어갑니다.

예제

```
SUBSTRING("123456789", 0)      -- "123456789"
SUBSTRING("123456789", 1)      -- "123456789"
SUBSTRING("123456789", 2)      -- "23456789"
SUBSTRING("123456789", -4)     -- "123456789"
SUBSTRING("123456789", 0, 999) -- "123456789"
SUBSTRING("123456789", 1, 5)   -- "12345"
```

TRIM

문자열에서 앞에 오거나 뒤에 오는 문자들을 잘라냅니다. 제거할 기본 문자는 ''입니다.

구문

```
TRIM ( [[LEADING | TRAILING | BOTH remove_chars] FROM] string )
```

파라미터

string

함수가 실행되는 대상 문자열입니다.

선행 | 후행 | 둘 다

선행 문자를 잘라내거나 후행 문자를 잘라내거나 아니면 두 문자를 모두 잘라냅니다.

remove_chars

제거할 문자들의 집합입니다. `remove_chars`는 길이가 > 1인 문자열이 될 수 있다는 점에 유의하십시오. 이 함수는 제거된 문자열의 시작 또는 끝에 `remove_chars`의 문자가 포함된 문자열을 반환합니다.

예제

```
TRIM('       foobar         ')              -- 'foobar'
TRIM('      \tfoobar\t         ')           -- '\tfoobar\t'
TRIM(LEADING FROM '       foobar         ')  -- 'foobar         '
TRIM(TRAILING FROM '       foobar         ') -- '       foobar'
TRIM(BOTH FROM '       foobar         ')     -- 'foobar'
TRIM(BOTH '12' FROM '1112211foobar22211122') -- 'foobar'
```

UPPER

문자열이 주어지면 모든 소문자를 대문자로 변환합니다. 소문자로 변환되지 않는 문자는 변경되지 않고 그대로 유지됩니다.

구문

```
UPPER ( string )
```

파라미터

string

함수가 실행되는 대상 문자열입니다.

예제

```
UPPER('AbCdEfG!@#$') -- 'ABCDEFG!@#$'
```

문서 이력

- 설명서 최종 업데이트: 2019년 9월 18일
- 현재 API 버전: 2006-03-01

다음 표는 2018년 6월 19일 및 이후 Amazon Simple Storage Service 개발자 가이드의 각 릴리스에서 변경된 중요 사항에 대해 설명합니다. 이 설명서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

update-history-change	update-history-description	update-history-date
Amazon S3 Batch Operations에 대한 작업 태그 (p. 794)	Amazon S3 Batch Operations 작업에 태그를 추가하여 해당 작업을 제어하고 레이블을 지정할 수 있습니다. 자세한 내용은 Amazon S3 Batch Operations 작업 태그를 참조하십시오.	March 16, 2020
동일 리전 복제 (p. 794)	동일 리전 복제(SRR)는 동일한 AWS 리전의 Amazon S3 버킷에서 객체를 복사하는 데 사용됩니다. 교차 리전 및 동일 리전 복제에 관한 자세한 내용은 복제 단원을 참조하십시오.	September 18, 2019
교차 리전 복제가 객체 잠금 지원 (p. 794)	교차 리전 복제가 이제 Amazon S3 객체 잠금을 지원합니다. 자세한 내용은 교차 리전 복제 및 Amazon S3가 복제하는 대상 단원을 참조하십시오.	May 28, 2019
Amazon S3 Batch Operations (p. 794)	Amazon S3 Batch Operations을 사용하여 Amazon S3 객체에 대해 대규모 배치 작업을 수행할 수 있습니다. Amazon S3 Batch Operations은 지정된 객체의 목록에 대해 단일 작업을 실행할 수 있습니다. 단일 작업으로 엑사바이트 규모의 데이터가 포함된 수십억 개의 객체에서 지정된 작업을 수행할 수 있습니다. 자세한 내용은 배치 작업 수행을 참조하십시오.	April 30, 2019
아시아 태평양(홍콩) 리전 (p. 794)	이제 아시아 태평양(홍콩) 리전에서 Amazon S3을 사용할 수 있습니다. Amazon S3 리전 및 엔드포인트에 대한 자세한 정보는 AWS General Reference의 리전 및 엔드포인트 단원을 참조하십시오.	April 24, 2019
서버 액세스 로그에 새 필드가 추가됨 (p. 794)	Amazon S3은 서버 액세스 로그에 TLS(전송 계층 보안) 버전인 새 필드를 추가했습니다. 자세한 내용은 Amazon S3 서버 액세스 로그 형식 항목을 참조하십시오.	March 28, 2019

새 아카이브 스토리지 클래스 (p. 794)	Amazon S3은 거의 액세스하지 않는 객체를 저장하기 위한 새로운 아카이브 스토리지 클래스인 DEEP_ARCHIVE를 제공합니다. 자세한 내용은 스토리지 클래스를 참조하십시오.	March 27, 2019
서버 액세스 로그에 새 필드가 추가됨 (p. 794)	Amazon S3은 서버 액세스 로그에 호스트 ID, 서명 버전, 암호 모음, 인증 유형 및 호스트 헤더와 같은 새 필드를 추가했습니다. 자세한 내용은 Amazon S3 서버 액세스 로그 형식 항목을 참조하십시오.	March 5, 2019
Parquet 형식 Amazon S3 인벤토리 파일 지원 (p. 794)	이제 Amazon S3는 인벤토리 출력 파일에 대해 Apache ORC(Optimized Row Columnar) 및 쉼표로 구분된 값(CSV) 파일 형식 이외에 Apache Parquet(Parquet) 형식도 지원합니다. 자세한 내용은 Amazon S3 인벤토리를 참조하십시오.	December 4, 2018
새 스토리지 클래스 (p. 794)	Amazon S3는 이제 변경되고 있거나 알 수 없는 액세스 패턴으로 수명이 긴 데이터용으로 설계된 INTELLIGENT_TIERING이라는 새로운 스토리지 클래스를 제공합니다. 자세한 내용은 스토리지 클래스를 참조하십시오.	November 26, 2018
복원 이벤트 알림 (p. 794)	Amazon S3 이벤트 알림은 이제 S3 Glacier 스토리지 클래스에서 객체를 복원할 때 초기화 및 완료 이벤트를 지원합니다. 자세한 내용은 이벤트 알림을 참조하십시오.	November 26, 2018
복원 속도 업그레이드 (p. 794)	Amazon S3 복원 속도 업그레이드를 사용하면 복원이 진행되는 동안 S3 Glacier 스토리지 클래스에서 더 빠른 속도로 복원 속도를 변경할 수 있습니다. 자세한 내용은 보관된 객체의 복원을 참조하십시오.	November 26, 2018

S3 Glacier 스토리지 클래스에 직접 PUT (p. 794)	Amazon S3 PUT 작업은 이제 객체를 생성할 때 S3 Glacier를 스토리지 클래스로 지정하는 것을 지원합니다. 이전에는 다른 Amazon S3 스토리지 클래스에서 S3 Glacier 스토리지 클래스로 객체를 전환해야 했습니다. 또한 S3 교차 리전 복제(CRR)를 사용할 때 이제 복제된 객체의 스토리지 클래스로 S3 Glacier를 지정할 수 있습니다. S3 Glacier 스토리지 클래스에 대한 자세한 내용은 스토리지 클래스를 참조하십시오. 복제된 객체에 대한 스토리지 클래스 지정에 대한 자세한 내용은 복제 구성 개요를 참조하십시오. S3 Glacier REST API 변경 사항에 직접 PUT에 대한 자세한 내용은 문서 이력: S3 Glacier에 직접 PUT을 참조하십시오.	November 26, 2018
Amazon S3 객체 잠금 (p. 794)	Amazon S3는 Amazon S3 객체에 대한 Write Once Read Many(WORM) 보호를 제공하는 객체 잠금 기능을 제공합니다. 자세한 내용은 객체 잠금을 참조하십시오.	November 26, 2018
Amazon S3 퍼블릭 액세스 차단 (p. 794)	Amazon S3는 이제 버킷 및 객체에 대한 퍼블릭 액세스를 버킷당 또는 계정 전체 기준으로 차단하는 기능이 포함됩니다. 자세한 내용은 Amazon S3 퍼블릭 액세스 차단 사용 단원을 참조하십시오.	November 15, 2018
교차 리전 복제(CRR) 규칙에서 필터링 개선 (p. 794)	CRR 규칙 구성에서 객체 필터를 지정하여 규칙을 적용할 객체의 하위 집합을 선택할 수 있습니다. 이전에는 객체 키 접두사만 필터링할 수 있었습니다. 이 릴리스에서는 객체 키 접두사, 하나 이상의 객체 태그 또는 이 두 가지를 모두에 대해 필터를 지정할 수 있습니다. 자세한 내용은 CRR 설정: 복제 구성 개요 단원을 참조하십시오.	September 19, 2018
새로운 Amazon S3 Select 기능 (p. 794)	Amazon S3 Select는 이제 Apache Parquet 입력, 중첩 JSON 객체에 대한 쿼리, 두 개의 새로운 Amazon CloudWatch 모니터링 지표(`SelectScannedBytes` 및 `SelectReturnedBytes`)를 지원합니다.	September 5, 2018

RSS에서 현재 사용 가능한 업데이트 (p. 794)	이제 RSS 피드를 구독하여 Amazon Simple Storage Service 개발자 가이드 설명서 업데이트에 대한 알림을 받을 수 있습니다.	June 19, 2018

이전 업데이트

다음 표에서는 2018년 6월 19일 이전 Amazon Simple Storage Service 개발자 가이드의 각 릴리스에서 변경된 중요 사항에 대해 설명합니다.

변경 사항	설명	날짜
코드 예제 업데이트	다음과 같이 코드 예제가 업데이트되었습니다. • C#–작업 기반 비동기 패턴을 사용하도록 모든 예제가 업데이트되었습니다. 자세한 내용은 .NET용 AWS SDK 개발자 안내서의 Amazon Web Services Asynchronous APIs for .NET 단원을 참조하십시오. 코드 예제는 이제 버전 3과 호환됩니다. • Java–클라이언트 빌더 모델을 사용하도록 모든 예제가 업데이트되었습니다. 클라이언트 빌더 모델에 대한 자세한 내용은 서비스 클라이언트 생성을 참조하십시오. • PHP–PHP용 AWS SDK 3.0을 사용하도록 모든 예제가 업데이트되었습니다. PHP용 AWS SDK 3.0에 대한 자세한 내용은 PHP용 AWS SDK 단원을 참조하십시오. • Ruby–Ruby용 AWS SDK 버전 3과 작동하도록 예제 코드가 업데이트되었습니다.	2018년 30월 4일

이제 Amazon S3이 Amazon CloudWatch Logs 스토리지 지표에 S3 Glacier 및 ONEZONE_IA 스토리지 클래스를 보고합니다.	이러한 스토리지 지표는 실제 바이트 수를 보고하는 것 이외에 적용 가능한 스토리지 클래스(ONEZONE_IA, STANDARD_IA 및 S3 Glacier)에 대한 객체당 오버헤드 바이트 수를 포함합니다.
	• ONEZONE_IA 및 STANDARD_IA 스토리지 클래스 객체의 경우 Amazon S3는 128KB보다 작은 객체를 128KB로 보고합니다. 자세한 내용은 Amazon S3 스토리지 클래스 (p. 107) 단원을 참조하십시오.
	• S3 Glacier 스토리지 클래스 객체의 경우 스토리지 지표는 다음 오버헤드를 보고합니다.
	  • 객체당 32KB 오버헤드(S3 Glacier 스토리지 클래스 요금으로 청구됨)
	  • 객체당 8KB 오버헤드(STANDARD 스토리지 클래스 요금으로 청구됨)
	  자세한 내용은 Amazon S3 수명 주기를 사용하여 객체 이전 (p. 124) 단원을 참조하십시오.
	스토리지 측정치에 대한 자세한 내용은 Amazon CloudWatch로 지표 모니터링 (p. 669) 단원을 참조하십시오.
		2018년 30월 4일

새 스토리지 클래스	Amazon S3는 이제 객체 저장을 위한 새 스토리지 클래스인 ONEZONE_IA(IA, 자주 액세스하지 않는 경우)를 제공합니다. 자세한 내용은 Amazon S3 스토리지 클래스 (p. 107) 단원을 참조하십시오.	2018년 4월 4일

변경 사항	설명	날짜
Amazon S3 Select	Amazon S3가 이제 SQL 표현식에 기초하여 객체 콘텐츠를 검색하는 기능을 지원합니다. 자세한 내용은 객체에서 콘텐츠 선택 (p. 243) 단원을 참조하십시오.	2018년 4월 4일
아시아 태평양(오사카-로컬) 리전	이제 아시아 태평양(오사카-로컬) 리전에서 Amazon S3을 사용할 수 있습니다. Amazon S3 리전 및 엔드포인트에 대한 자세한 정보는 AWS General Reference의 리전 및 엔드포인트 단원을 참조하십시오. Important 아시아 태평양(오사카-로컬) 리전은 반드시 아시아 태평양(도쿄) 리전과 함께 사용해야 합니다. 리전에 대한 액세스를 요청하려면 영업 담당자에게 문의하십시오.	2018년 2월 12일
Amazon S3 인벤토리 생성 타임스탬프	이제 Amazon S3 인벤토리에 Amazon S3 인벤토리 보고서를 생성한 날짜 및 시작 시간을 나타내는 타임스탬프가 포함됩니다. 타임스탬프를 통해 인벤토리 보고서가 생성된 시작 시간을 알 수 있으므로 Amazon S3 스토리지가 변경되었는지를 확인할 수 있습니다.	2018년 1월 16일
유럽(파리) 리전	이제 유럽(파리) 리전에서 Amazon S3을 사용할 수 있습니다. Amazon S3 리전 및 엔드포인트에 대한 자세한 정보는 AWS General Reference의 리전 및 엔드포인트 단원을 참조하십시오.	2017년 12월 18일
중국(닝샤) 리전	이제 중국(닝샤) 리전에서 Amazon S3을 사용할 수 있습니다. Amazon S3 리전 및 엔드포인트에 대한 자세한 정보는 AWS General Reference의 리전 및 엔드포인트 단원을 참조하십시오.	2017년 12월 11일
SQL로 아카이브 쿼리	Amazon S3는 이제 SQL을 사용한 S3 Glacier 데이터 아카이브 쿼리를 지원합니다. 자세한 내용은 보관된 객체의 쿼리 (p. 251) 단원을 참조하십시오.	2017년 11월 29일
ORC 형식 Amazon S3 인벤토리 파일 지원	이제 Amazon S3는 인벤토리 출력 파일에 대해 쉼표로 구분된 값(CSV) 파일 형식 이외에 Apache ORC(Optimized Row Columnar) 형식도 지원합니다. 또한 이제 Amazon Athena, Amazon Redshift Spectrum 및 기타 도구(예: Presto, Apache Hive, Apache Spark)를 사용하여 표준 SQL을 통해 Amazon S3 인벤토리를 쿼리할 수 있습니다. 자세한 내용은 Amazon S3 인벤토리 (p. 459) 단원을 참조하십시오.	2017년 11월 17일
S3 버킷에 대한 기본 암호화	Amazon S3 기본 암호화를 사용하면 S3 버킷의 기본 암호화 동작을 설정할 수 있습니다. 버킷에 저장되는 모든 객체를 암호화하도록 버킷에 대한 기본 암호화를 설정할 수 있습니다. Amazon S3 관리형 키(SSE-S3) 또는 AWS KMS 관리형 키(SSE-KMS)로 서버 측 암호화를 사용하여 객체를 암호화합니다. 자세한 내용은 S3 버킷에 대한 Amazon S3 기본 암호화 (p. 65) 단원을 참조하십시오.	2017년 11월 06일
Amazon S3 인벤토리의 암호화 상태	Amazon S3는 이제 Amazon S3 인벤토리의 암호화 상태를 지원하므로 규정 준수 감사 또는 기타 목적을 위해 저장된 객체가 어떻게 암호화되었는지 확인할 수 있습니다. 또한 모든 인벤토리 파일이 적절히 암호화되도록 서버 측 암호화(SSE) 또는 SSE-KMS를 사용하여 S3 인벤토리를 암호화하도록 구성할 수 있습니다. 자세한 내용은 Amazon S3 인벤토리 (p. 459) 단원을 참조하십시오.	2017년 11월 06일

변경 사항	설명	날짜
교차 리전 복제(CRR) 기능 향상	이제 교차 리전 복제에서는 다음을 지원합니다. • 교차 계정 시나리오에서 복제본 소유권을 대상 버킷을 소유한 AWS 계정으로 변경하도록 CRR 구성을 추가할 수 있습니다. 자세한 내용은 복제본 소유자 변경 (p. 626) 단원을 참조하십시오. • 기본적으로 Amazon S3에서는 AWS KMS에 저장된 키를 사용하는 서버 측 암호화를 사용하여 생성되는 객체를 원본 버킷에 복제하지 않습니다. 이제 CRR 구성에서 이러한 객체를 복제하도록 Amazon S3에 지시할 수 있습니다. 자세한 내용은 AWS KMS에 저장된 암호화 키를 사용한 서버 측 암호화(SSE)로 생성된 객체 복제 (p. 629) 단원을 참조하십시오.	2017년 11월 06일
유럽(런던) 리전	이제 유럽(런던) 리전에서 Amazon S3을 사용할 수 있습니다. Amazon S3 리전 및 엔드포인트에 대한 자세한 정보는 AWS General Reference의 리전 및 엔드포인트 단원을 참조하십시오.	2016년 13월 12일
캐나다(중부) 리전	이제 캐나다(중부) 리전에서 Amazon S3을 사용할 수 있습니다. Amazon S3 리전 및 엔드포인트에 대한 자세한 정보는 AWS General Reference의 리전 및 엔드포인트 단원을 참조하십시오.	2016년 12월 8일
객체 태그 지정	Amazon S3가 이제 객체 태그 지정을 지원합니다. 객체 태그 지정을 통해 스토리지를 분류할 수 있습니다. 객체 키 이름 접두사로도 스토리지를 분류할 수 있지만, 객체 태그 지정은 여기에 새로운 차원을 더합니다. 태그 지정은 더욱 다양한 혜택을 제공합니다. 다음이 포함됩니다. • 객체 태그는 정교한 권한 액세스 제어를 구현합니다(예를 들어, 특정 태그로 IAM 사용자에 읽기 전용 객체에 대한 권한을 부여할 수 있습니다). • 수명 주기 구성 지정으로 정교한 통제를 구현합니다. 태그를 지정해 수명 주기 규칙을 적용할 객체 하위 집합을 선택할 수 있습니다. • 교차 리전 복제(CRR)를 구성했다면, Amazon S3가 태그를 복제할 수 있습니다. 객체 복제를 맡기려면 Amazon S3에 대해 생성된 IAM 역할에 필요한 권한을 부여해야 합니다. • 또한 CloudWatch 지표 및 CloudTrail 이벤트를 사용자 지정하여 특정 태그 필터로 걸러 정보를 표시할 수도 있습니다. 자세한 내용은 객체 태그 지정 (p. 114) 단원을 참조하십시오.	2016년 11월 29일
Amazon S3 수명 주기가 이제 태그 기반 필터를 지원	Amazon S3가 이제 수명 주기 구성에서 태그 기반 필터링을 지원합니다. 이제 키 접두사, 하나 이상의 객체 태그 또는 이들의 조합을 지정해 수명 주기 규칙을 적용할 객체 하위 집단을 선택할 수 있는 수명 주기 규칙을 지정할 수 있습니다. 자세한 내용은 객체 수명 주기 관리 (p. 123) 단원을 참조하십시오.	2016년 11월 29일
버킷에 대한 CloudWatch 요청 지표	Amazon S3가 이제 버킷에 대한 요청에서 CloudWatch 지표를 지원합니다. 버킷에 대한 이러한 지표를 활성화하면, 지표는 1분 간격으로 보고됩니다. 버킷 내 어떤 객체가 이러한 요청 지표를 보고할지 구성할 수도 있습니다. 자세한 내용은 Amazon CloudWatch로 지표 모니터링 (p. 669) 단원을 참조하십시오.	2016년 11월 29일

변경 사항	설명	날짜
Amazon S3 인벤토리	Amazon S3가 이제 스토리지 인벤토리를 지원합니다. Amazon S3 인벤토리는 S3 버킷 또는 공유 접두사(공통 문자열로 시작하는 이름을 가진 객체)에 대해 일일 또는 주간 기준으로 객체 및 해당 메타데이터의 플랫 파일 출력을 가능하게 합니다. 자세한 내용은 Amazon S3 인벤토리 (p. 459) 단원을 참조하십시오.	2016년 11월 29일
Amazon S3 분석 – 스토리지 클래스 분석	새로운 Amazon S3 분석 기능은 데이터 액세스 패턴을 관찰해 자주 액세스하지 않는 STANDARD 스토리지를 STANDARD_IA (IA는 자주 액세스하지 않는다는 뜻입니다) 스토리지 클래스로 옮길 시점을 알려줍니다. 스토리지 클래스 분석 결과 필터링 데이터 세트가 일정 시간 동안 액세스 빈도가 떨어지는 패턴을 보인다면, 분석 결과를 이용해 수명 주기 정책을 개선할 수 있습니다. 또한 이 기능은 S3 버킷으로 내보낼 수 있는 지정된 버킷, 접두사 또는 태그 레벨에서의 스토리지 사용에 대한 상세한 일일 분석도 포함합니다. 자세한 내용은 Amazon Simple Storage Service 개발자 가이드의 Amazon S3 분석 – 스토리지 클래스 분석 (p. 255)를 참조하십시오.	2016년 11월 29일
S3 Glacier에 저장된 객체를 복원할 때 새로운 신속 처리 및 벌크 데이터를 검색합니다.	Amazon S3가 이제 S3 Glacier에 저장된 객체를 복원할 때 표준 검색 외에 신속 처리 및 벌크 데이터 검색도 지원합니다. 자세한 내용은 보관된 객체의 복원 (p. 247) 단원을 참조하십시오.	2016년 11월 21일
CloudTrail 객체 로깅	CloudTrail은 `GetObject`, `PutObject`, `DeleteObject` 같은 Amazon S3 객체 레벨 API 작업 로깅을 지원합니다. 이벤트 셀렉터를 구성하여 객체 레벨 API 작업 로깅이 가능합니다. 자세한 내용은 AWS CloudTrail을 사용하여 Amazon S3 API 호출 로깅 (p. 679) 단원을 참조하십시오.	2016년 11월 21일
미국 동부(오하이오) 리전	이제 미국 동부(오하이오) 리전에서 Amazon S3을 사용할 수 있습니다. Amazon S3 리전 및 엔드포인트에 대한 자세한 정보는 AWS General Reference의 리전 및 엔드포인트 단원을 참조하십시오.	2016년 10월 17일
Amazon S3 Transfer Acceleration에 대한 IPv6 지원	Amazon S3가 이제 Amazon S3 Transfer Acceleration에 대해 인터넷 프로토콜 버전 6(IPv6)을 지원합니다. Transfer Acceleration 엔드포인트에 대해 새로운 듀얼 스택을 사용하여 IPv6으로 Amazon S3에 연결할 수 있습니다. 자세한 내용은 Amazon S3 Transfer Acceleration 시작하기 (p. 68) 단원을 참조하십시오.	2016년 10월 6일
IPv6 지원	Amazon S3가 이제 인터넷 프로토콜 버전 6(IPv6)을 지원합니다. 듀얼 스택 엔드포인트를 사용하여 IPv6을 통해 Amazon S3에 액세스할 수 있습니다. 자세한 내용은 IPv6을 통해 Amazon S3에 요청 (p. 11) 단원을 참조하십시오.	2016년 8월 11일
아시아 태평양(뭄바이) 리전	이제 아시아 태평양(뭄바이) 리전에서 Amazon S3을 사용할 수 있습니다. Amazon S3 리전 및 엔드포인트에 대한 자세한 정보는 AWS General Reference의 리전 및 엔드포인트 단원을 참조하십시오.	2016년 6월 27일

변경 사항	설명	날짜
Amazon S3 Transfer Acceleration	Amazon S3 Transfer Acceleration은 거리가 먼 클라이언트와 S3 버킷 간에 파일을 빠르고, 쉽고, 안전하게 전송할 수 있게 해 줍니다. Transfer Acceleration은 전 세계적으로 분산되어 있는 Amazon CloudFront의 엣지 로케이션을 활용합니다. 자세한 내용은 Amazon S3 Transfer Acceleration (p. 67) 단원을 참조하십시오.	2016년 4월 19일
만료된 객체 삭제 마커를 제거하기 위한 수명 주기 지원	이제 수명 주기 구성 `Expiration` 작업을 통해, 버전이 지정된 버킷에서 만료된 객체 삭제 마커를 제거하도록 Amazon S3에게 지시할 수 있습니다. 자세한 내용은 수명 주기 작업을 설명할 요소 (p. 133) 단원을 참조하십시오.	2016년 3월 16일
이제 버킷 수명 주기 구성은 미완료 멀티파트 업로드 중단 작업을 지원합니다	이제 버킷 수명 주기 구성은 시작된 후 지정 일수 내에 완료되지 않은 멀티파트 업로드를 중단하도록 Amazon S3에게 지시하는 데 사용할 수 있는 `AbortIncompleteMultipartUpload` 작업을 지원합니다. 멀티파트 업로드가 중단 작업을 할 수 있는 조건을 갖춘 경우, Amazon S3는 업로드된 부분을 제거하고 멀티파트 업로드를 중단합니다. 개념에 대한 설명은 Amazon Simple Storage Service 개발자 가이드에서 다음 주제를 참조하십시오. • 버킷 수명 주기 정책을 사용한 미완료 멀티파트 업로드 중단 (p. 180) • 수명 주기 작업을 설명할 요소 (p. 133) 다음 API 작업들은 새로운 작업을 지원하도록 업데이트되었습니다. • PUT 버킷 수명 주기 – 이제 XML 구성을 통해 수명 주기 구성 규칙에서 `AbortIncompleteMultipartUpload` 작업을 지정할 수 있습니다. • 부분 나열 및 멀티파트 업로드 시작 – 이제 이 두 API 작업은 버킷에 `AbortIncompleteMultipartUpload` 작업을 지정하는 수명 주기 규칙이 있는 경우, 2개의 응답 헤더(`x-amz-abort-date` 및 `x-amz-abort-rule-id`)를 추가로 반환합니다. 이 응답 헤더들은 시작된 멀티파트 업로드가 중단 작업을 할 수 있는 조건이 되는 시점 및 적용 가능한 수명 주기 규칙을 알려줍니다.	2016년 3월 16일
아시아 태평양(서울) 리전	이제 아시아 태평양(서울) 리전에서 Amazon S3을 사용할 수 있습니다. Amazon S3 리전 및 엔드포인트에 대한 자세한 정보는 AWS General Reference의 리전 및 엔드포인트 단원을 참조하십시오.	2016년 1월 6일
새로운 조건 키 및 멀티파트 업로드 변경	IAM 정책은 현재 Amazon S3 `s3:x-amz-storage-class` 조건 키를 지원합니다. 자세한 내용은 Amazon S3 조건 키 (p. 345) 단원을 참조하십시오. 이제 파트를 업로드하고 업로드를 완료할 목적으로 직접 멀티파트 업로드를 시작할 필요가 없습니다. 자세한 내용은 멀티파트 업로드 API 및 권한 (p. 182) 단원을 참조하십시오.	2015년 12월 14일

변경 사항	설명	날짜
미국 표준 리전의 이름 변경	리전 이름 문자열이 "미국 표준"에서 "미국 동부(버지니아 북부)"로 바뀌었습니다. 리전 이름만 업데이트되었을 뿐 기능은 바뀌지 않았습니다.	2015년 12월 11일
새 스토리지 클래스	Amazon S3는 이제 객체 저장을 위한 새 스토리지 클래스인 STANDARD_IA(IA, 자주 액세스하지 않는 경우)를 제공합니다. 이 스토리지 클래스는 수명이 길고 액세스 빈도가 낮은 데이터에 최적화되었습니다. 자세한 내용은 Amazon S3 스토리지 클래스 (p. 107) 단원을 참조하십시오. 이제 수명 주기 구성 기능 업데이트를 사용하여 객체를 STANDARD_IA 스토리지 클래스로 전환할 수 있습니다. 자세한 내용은 객체 수명 주기 관리 (p. 123) 단원을 참조하십시오. 이전의 교차 리전 복제 기능에서는 객체 복제본에 대해 원본 객체의 스토리지 클래스를 사용했습니다. 이제 교차 리전 복제를 구성할 때 대상 버킷에 생성되는 객체 복제본에 대한 스토리지 클래스를 지정할 수 있습니다. 자세한 내용은 복제 (p. 607) 단원을 참조하십시오.	2015년 9월 16일
AWS CloudTrail 통합	새로운 AWS CloudTrail 통합을 통해 S3 버킷의 Amazon S3 API 작업을 기록할 수 있습니다. CloudTrail을 사용하여 S3 버킷의 생성 또는 삭제, 액세스 제어 수정 또는 수명 주기 변경을 추적할 수 있습니다. 자세한 내용은 AWS CloudTrail을 사용하여 Amazon S3 API 호출 로깅 (p. 679) 단원을 참조하십시오.	2015년 9월 1일
버킷 한도 향상	Amazon S3는 이제 버킷 한도 향상을 지원합니다. 기본적으로 고객은 AWS 계정당 최대 100개의 버킷을 생성할 수 있습니다. 추가로 버킷이 필요한 고객은 서비스 한도 향상을 제출하여 버킷 한도를 늘릴 수 있습니다. 버킷 한도를 늘리는 방법은 AWS General Reference의 AWS Service Limits를 참조하십시오. 자세한 내용은 버킷 생성 (p. 51) 및 버킷 규제 및 제한 (p. 56) 단원을 참조하십시오.	2015년 8월 4일
일관성 모델 업데이트	Amazon S3는 이제 미국 동부(버지니아 북부) 리전에서 Amazon S3에 추가된 새 객체에 대해 읽기 후 쓰기 일관성을 지원합니다. 이 업데이트 전에는 미국 동부(버지니아 북부)를 제외한 모든 리전에서 Amazon S3에 업로드된 새로운 객체에 대해 읽기 후 쓰기 일관성을 지원했습니다. 이러한 개선을 통해 Amazon S3는 이제 모든 리전에서 Amazon S3에 추가된 새 객체에 대해 읽기 후 쓰기 일관성을 지원합니다. 읽기 후 쓰기 일관성이 지원되면 Amazon S3에 객체를 생성한 후 즉시 객체를 검색할 수 있습니다. 자세한 내용은 Regions (p. 4) 단원을 참조하십시오.	2015년 8월 4일
이벤트 알림	Amazon S3 이벤트 알림이 객체 삭제 시의 알림을 추가하고, 접두사와 접미사가 일치하는 객체 이름에 대한 필터링을 추가하도록 업데이트되었습니다. 자세한 내용은 Amazon S3 이벤트 알림 구성 (p. 587) 단원을 참조하십시오.	2015년 7월 28일
Amazon CloudWatch 통합	Amazon CloudWatch가 새롭게 통합됨에 따라 Amazon S3에 대한 CloudWatch 측정치를 통해 Amazon S3 사용량을 모니터링하고 경보를 설정할 수 있습니다. 지원되는 지표에는 표준 스토리지의 총 바이트 수, RRS(Reduced-Redundancy Storage)의 총 바이트 수, 특정 S3 버킷의 총 객체 수가 포함됩니다. 자세한 내용은 Amazon CloudWatch로 지표 모니터링 (p. 669) 단원을 참조하십시오.	2015년 7월 28일

변경 사항	설명	날짜
비어 있지 않은 버킷의 삭제 및 비우기 지원	Amazon S3에서는 이제 비어 있지 않은 버킷의 삭제와 비우기가 지원됩니다. 자세한 내용은 버킷 삭제 또는 비우기 (p. 61) 단원을 참조하십시오.	2015년 7월 16일
Amazon VPC 종단점의 버킷 정책	Amazon S3에 Amazon Virtual Private Cloud(Amazon VPC) 엔드포인트의 버킷 정책에 대한 지원이 추가되었습니다. S3 버킷 정책을 사용하여 특정 Amazon VPC 엔드포인트 또는 특정 VPC의 버킷에 대한 액세스를 제어할 수 있습니다. VPC 엔드포인트는 게이트웨이 또는 NAT 인스턴스가 없어도 구성하기가 쉽고 높은 안정성을 자랑하며 Amazon S3에 대한 안전한 연결을 제공합니다. 자세한 내용은 Amazon S3의 VPC 엔드포인트에 대한 예제 버킷 정책 (p. 414) 단원을 참조하십시오.	2015년 4월 29일
이벤트 알림	AWS Lambda 기능에 대해 리소스 기반 권한으로 전환하는 작업을 지원하기 위해 Amazon S3 이벤트 알림이 업데이트되었습니다. 자세한 내용은 Amazon S3 이벤트 알림 구성 (p. 587) 단원을 참조하십시오.	2015년 4월 9일
리전 간 복제	Amazon S3에서는 이제 교차 리전 복제가 지원됩니다. 교차 리전 복제는 서로 다른 AWS 리전의 버킷 간 객체를 비동기식으로 자동 복사하는 것을 말합니다. 자세한 내용은 복제 (p. 607) 단원을 참조하십시오.	2015년 3월 24일
이벤트 알림	Amazon S3의 버킷 알림 구성에서 이제 새로운 이벤트 유형 및 대상이 지원됩니다. 이 릴리스 이전에는 Amazon S3에서 s3:ReducedRedundancyLostObject 이벤트 유형 및 Amazon SNS 주제만 대상으로 지원되었습니다. 새로운 이벤트 유형에 대한 자세한 내용은 Amazon S3 이벤트 알림 구성 (p. 587)을 참조하십시오.	2014년 11월 13일
고객 제공 암호화 키를 사용한 서버 측 암호화	AWS Key Management Service(AWS KMS)를 사용한 서버 측 암호화 Amazon S3에서는 이제 AWS KMS를 사용한 서버 측 암호화를 지원합니다. 이 기능을 사용하면 AWS KMS를 통해 엔벌로프 키를 관리할 수 있습니다. Amazon S3는 AWS KMS를 호출하여 사용자가 설정한 권한 내에서 엔벌로프 키에 액세스합니다. AWS KMS를 사용한 서버 측 암호화에 대한 자세한 내용은 Protecting Data Using Server-Side Encryption with AWS Key Management Service 단원을 참조하십시오.	2014년 11월 12일
유럽(프랑크푸르트) 리전	이제 유럽(프랑크푸르트) 리전에서 Amazon S3을 사용할 수 있습니다.	2014년 10월 23일
고객 제공 암호화 키를 사용한 서버 측 암호화	Amazon S3에서는 이제 고객 제공 암호화 키를 사용한 서버 측 암호화(SSE-C)가 지원됩니다. 서버 측 암호화를 사용하면 유휴 데이터를 암호화할 것을 Amazon S3에 요청할 수 있습니다. SSE-C를 사용하면 Amazon S3에서 사용자가 제공하는 사용자 정의 암호화 키로 객체를 암호화합니다. Amazon S3에서 자동으로 암호화가 수행되므로 사용자는 자신만의 암호화 코드를 작성하거나 실행하는 데 비용을 들이지 않고 자신만의 암호화 키를 사용하는 혜택을 얻게 됩니다. SSE-C에 대한 자세한 내용은 Server-Side Encryption (Using Customer-Provided Encryption Keys)을 참조하십시오.	2014년 6월 12일

변경 사항	설명	날짜
버전 관리를 위한 수명 주기 지원	이 릴리스 이전에는 버전 관리를 사용하지 않는 버킷에서만 수명 주기 구성이 지원되었습니다. 이제는 버전 관리를 사용하지 않는 버킷과 버전 관리가 활성화된 버킷 모두에서 수명 주기를 구성할 수 있습니다. 자세한 내용은 객체 수명 주기 관리 (p. 123) 단원을 참조하십시오.	2014년 5월 20일
액세스 제어 주제 개정	Amazon S3 액세스 제어 설명서가 개정되었습니다. 자세한 내용은 Amazon S3의 Identity and Access Management (p. 299) 단원을 참조하십시오.	2014년 4월 15일
서버 액세스 로깅 주제 개정	서버 액세스 로깅 설명서가 개정되었습니다. 자세한 내용은 Amazon S3 서버 액세스 로깅 (p. 705) 단원을 참조하십시오.	2013년 11월 26일
버전 2.0으로 .NET SDK 샘플 업데이트	이 안내서의 .NET SDK 샘플은 이제 버전 2.0에 맞습니다.	2013년 11월 26일
HTTP를 통한 SOAP 지원 중단	HTTP를 통한 SOAP 지원은 중단되었지만 HTTPS를 통해 계속해서 사용할 수 있습니다. 새로운 Amazon S3 기능은 SOAP에 대해 지원되지 않습니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.	2013년 9월 20일
IAM 정책 변수 지원	IAM 액세스 정책 언어는 이제 변수를 지원합니다. 정책이 평가되면 모든 정책 변수가 인증된 사용자의 세션에서 컨텍스트 기반 정보에 의해 제공되는 값으로 대체됩니다. 정책 변수를 사용하여 정책의 모든 구성 요소를 명시적으로 나열하지 않고 범용 정책을 정의할 수 있습니다. 정책 변수에 대한 자세한 내용은 IAM 사용 설명서의 IAM 정책 변수 개요 단원을 참조하십시오. Amazon S3의 정책 변수 예제는 사용자 정책 예제 (p. 416)를 참조하십시오.	2013년 4월 3일
요청자 지불에 대한 콘솔 지원	이제 Amazon S3 콘솔을 사용하여 요청자 지불에 대한 버킷을 구성할 수 있습니다. 자세한 내용은 Amazon S3 콘솔을 사용한 요청자 지불 구성 (p. 74) 단원을 참조하십시오.	2012년 12월 31일
웹 사이트 호스팅에 대한 루트 도메인 지원	Amazon S3에서는 이제 루트 도메인에서의 정적 웹 사이트 호스팅이 지원됩니다. 웹 사이트 방문자는 웹 주소에 "www"를 지정하지 않고(예: "example.com") 자신의 브라우저에서 사이트에 액세스할 수 있습니다. 많은 고객이 이미 Amazon S3에서 "www" 하위 도메인(예: "www.example.com")을 통해 액세스 가능한 정적 웹 사이트를 호스팅하고 있습니다. 이전에는 루트 도메인 액세스를 지원하기 위해 자신만의 웹 서버를 실행해야 브라우저에서 Amazon S3에 있는 자신의 웹 사이트로 루트 도메인을 프록시할 수 있었습니다. 요청을 프록시하기 위해 웹 서버를 실행하면 추가적인 비용, 운영 부담 및 또 다른 잠재적 장애 지점이 생깁니다. 이제는 "www"와 루트 도메인 주소 모두에 대해 Amazon S3의 고가용성 및 내구성을 활용할 수 있습니다. 자세한 내용은 Amazon S3 정적 웹 사이트 호스팅 (p. 548) 단원을 참조하십시오.	2012년 12월 27일
콘솔 개정	Amazon S3 콘솔이 업데이트되었습니다. 콘솔을 참조하는 설명서 주제도 이에 따라 개정되었습니다.	2012년 12월 14일

변경 사항	설명	날짜
S3 Glacier로의 데이터 보관 지원	Amazon S3에서는 이제 데이터 보관에 S3 Glacier의 저렴한 스토리지 서비스를 활용할 수 있게 해 주는 스토리지 옵션이 지원됩니다. 객체를 보관하려면 객체를 식별하는 보관 규칙과 Amazon S3에서 이러한 객체를 S3 Glacier에 보관하도록 할 타임라인을 정의합니다. Amazon S3 콘솔을 사용하거나 Amazon S3 API 또는 AWS SDK를 사용하여 프로그래밍 방식으로 버킷에 대한 규칙을 쉽게 설정할 수 있습니다. 자세한 내용은 객체 수명 주기 관리 (p. 123) 단원을 참조하십시오.	2012년 11월 13일
웹 사이트 페이지 리디렉션에 대한 지원	웹 사이트로 구성되는 버킷의 경우 Amazon S3에서는 이제 객체에 대한 요청을 같은 버킷의 다른 객체 또는 외부 URL로 리디렉션하는 작업이 지원됩니다. 자세한 내용은 (선택 사항) 웹 페이지 리디렉션 구성 (p. 559) 단원을 참조하십시오. 웹 사이트 호스팅에 대한 자세한 내용은 Amazon S3 정적 웹 사이트 호스팅 (p. 548)을 참조하십시오.	2012년 10월 4일
교차 원본 리소스 공유(CORS)에 대한 지원	Amazon S3에서는 이제 교차 원본 리소스 공유(CORS)가 지원됩니다. CORS는 한 도메인에서 로드되는 클라이언트 웹 애플리케이션이 다른 도메인에 있는 리소스와 상호 작용하거나 해당 리소스에 액세스할 수 있는 방법을 정의합니다. Amazon S3의 CORS 지원을 통해 Amazon S3를 기반으로 풍부한 클라이언트 측 웹 애플리케이션을 구축하고 Amazon S3 리소스에 대한 교차 도메인 액세스를 선택적으로 허용할 수 있습니다. 자세한 내용은 CORS(Cross-Origin Resource Sharing) (p. 154) 단원을 참조하십시오.	2012년 8월 31일
비용 할당 태그에 대한 지원	Amazon S3에서는 이제 비용 할당 태그 지정이 지원됩니다. 비용 할당 태그 지정을 사용하면 프로젝트 또는 기타 기준에 대해 보다 쉽게 비용을 추적할 수 있도록 S3 버킷에 레이블을 지정할 수 있습니다. 버킷에 태그 지정 사용에 대한 자세한 내용은 비용 할당 S3 버킷 태그 사용 (p. 88)을 참조하십시오.	2012년 8월 21일
버킷 정책에서의 MFA 보호 API 액세스에 대한 지원	Amazon S3에서는 이제 MFA 보호 API 액세스가 지원됩니다. MFA 보호 API 액세스는 Amazon S3 리소스에 액세스할 때 보안 수준을 높이기 위해 AWS Multi-Factor Authentication을 적용할 수 있는 기능입니다. 이는 사용자가 유효한 MFA 코드를 제공하여 MFA 디바이스를 물리적으로 가지고 있음을 증명하게 하는 보안 기능입니다. 자세한 내용은 AWS Multi-Factor Authentication을 참조하십시오. 이제 Amazon S3 리소스에 액세스하기 위한 모든 요청에 대해 MFA 인증을 요구할 수 있습니다. MFA 인증을 적용하기 위해 Amazon S3에서는 이제 버킷 정책에 `aws:MultiFactorAuthAge` 키가 지원됩니다. 버킷 정책 예제는 MFA가 필요한 버킷 정책 추가 (p. 411)를 참조하십시오.	2012년 7월 10일
객체 만료 지원	객체 만료 기능을 사용하여 구성된 시간 후에 데이터가 자동으로 제거되도록 예약할 수 있습니다. 수명 주기 구성을 버킷에 추가하여 객체 만료를 설정합니다.	2011년 12월 27일
새로운 리전 지원	Amazon S3에서는 이제 남아메리카(상파울루) 리전을 지원합니다. 자세한 내용은 버킷 액세스 (p. 53) 단원을 참조하십시오.	2011년 12월 14일

변경 사항	설명	날짜
다중 객체 삭제	Amazon S3에서는 이제 단일 요청으로 다중 객체를 삭제할 수 있게 하는 다중 객체 삭제 API가 지원됩니다. 이 기능을 사용하면 개별 DELETE 요청을 여러 개 사용하는 것보다 더 빨리 Amazon S3에서 많은 수의 객체를 제거할 수 있습니다. 자세한 내용은 객체 삭제 (p. 225) 단원을 참조하십시오.	2011년 12월 7일
새로운 리전 지원	Amazon S3에서는 이제 미국 서부(오레곤) 리전을 지원합니다. 자세한 내용은 버킷 및 리전 (p. 53) 단원을 참조하십시오.	2011년 11월 8일
설명서 업데이트	설명서 버그 수정	2011년 11월 8일
설명서 업데이트	이 릴리스에는 설명서 버그 수정 사항 외에 다음과 같은 향상된 내용도 포함되어 있습니다. • PHP용 AWS SDK(PHP용 AWS SDK를 사용하여 서버 측 암호화 지정 (p. 274) 참조) 및 Ruby용 AWS SDK(Ruby용 AWS SDK를 사용하여 서버 측 암호화 지정 (p. 276) 참조)를 사용한 서버 측 암호화에 대한 새로운 단원 • Ruby 샘플 생성 및 테스트에 대한 새로운 단원(Ruby용 AWS SDK 버전 3 사용 (p. 737) 참조)	2011년 10월 17일
서버 측 암호화 지원	Amazon S3에서는 이제 서버 측 암호화가 지원됩니다. 이 기능을 사용하면 유휴 데이터를 암호화할 것을 Amazon S3에 요청할 수 있습니다. 즉, Amazon S3에서 해당 데이터 센터의 디스크에 사용자 데이터를 쓸 때 객체 데이터를 암호화할 것을 요청할 수 있습니다. AWS SDK for Java 및 .NET은 REST API 업데이트 외에 서버 측 암호화를 요청하는 데 필요한 기능도 제공합니다. AWS Management Console을 사용하여 객체를 업로드할 때도 서버 측 암호화를 요청할 수 있습니다. 데이터 암호화에 대해 자세히 알아보려면 Using Data Encryption을 참조하십시오.	2011년 10월 4일
설명서 업데이트	이 릴리스에는 설명서 버그 수정 사항 외에 다음과 같은 향상된 내용도 포함되어 있습니다. • 요청 만들기 (p. 10) 단원에 Ruby 및 PHP 샘플을 추가했습니다. • 미리 서명된 URL을 생성하고 사용하는 방법에 대해 설명하는 단원을 추가했습니다. 자세한 내용은 다른 사용자와 객체 공유 (p. 170) 및 미리 서명된 URL을 사용하여 객체 업로드 (p. 205) 단원을 참조하십시오. • Eclipse 및 Visual Studio용 AWS Explorer를 소개하기 위해 기존 단원을 업데이트했습니다. 자세한 내용은 AWS SDK, CLI 및 Explorer 사용 (p. 727) 단원을 참조하십시오.	2011년 9월 22일

변경 사항	설명	날짜
임시 보안 자격 증명을 사용하여 요청하기 보내기 기능에 대한 지원	AWS 계정 및 IAM 사용자 보안 자격 증명을 사용하여 인증된 요청을 Amazon S3로 보내는 것 외에도 이제 AWS Identity and Access Management(IAM)에서 가져온 임시 보안 자격 증명을 사용하여 요청을 보낼 수 있습니다. AWS Security Token Service API 또는 AWS SDK 래퍼 라이브러리를 사용하여 IAM에서 이러한 임시 자격 증명을 요청할 수 있습니다. 직접 사용하기 위해 또는 연합된 사용자 및 애플리케이션에 배포하기 위해 이러한 임시 보안 자격 증명을 요청할 수 있습니다. 이 기능을 사용하면 AWS 외부에서 사용자를 관리하고 이러한 사용자에게 AWS 리소스에 액세스하기 위한 임시 보안 자격 증명을 제공할 수 있습니다. 자세한 내용은 요청 만들기 (p. 10) 단원을 참조하십시오. 임시 보안 자격 증명에 대한 IAM 지원에 관한 자세한 내용은 IAM 사용 설명서의 임시 보안 자격 증명 단원을 참조하십시오.	2011년 8월 3일
최대 5TB의 객체를 복사할 수 있도록 멀티파트 업로드 API 확장	이 릴리스 이전에는 Amazon S3 API에서 최대 5GB 크기의 객체 복사를 지원했습니다. 5GB를 초과하는 객체를 복사할 수 있도록 Amazon S3에서는 이제 새로운 작업인 `Upload Part (Copy)`로 멀티파트 업로드 API를 확장합니다. 이 멀티파트 업로드 작업을 사용하면 최대 5TB 크기의 객체를 복사할 수 있습니다. 자세한 내용은 객체 복사 (p. 209) 단원을 참조하십시오. 멀티파트 업로드 API에 대한 개념적 정보는 멀티파트 업로드 API를 사용한 객체 업로드 (p. 177)를 참조하십시오.	2011년 6월 21일
HTTP를 통한 SOAP API 호출 비활성화	보안을 높이기 위해 HTTP를 통한 SOAP API 호출이 비활성화되었습니다. 인증된 SOAP 요청 및 익명 SOAP 요청은 SSL을 사용하여 Amazon S3로 보내야 합니다.	2011년 6월 6일
IAM에서 교차 계정 위임 허용	이전에는 Amazon S3 리소스에 액세스하기 위해 IAM 사용자가 상위 AWS 계정과 Amazon S3 리소스 소유자 모두에게서 허가를 받아야 했습니다. 이제 교차 계정 액세스 기능을 사용하면 IAM 사용자가 소유자 계정으로부터만 허가를 받으면 됩니다. 즉, 리소스 소유자가 AWS 계정에 액세스 권한을 부여할 경우 해당 AWS 계정은 이제 자신의 IAM 사용자에게 이러한 리소스에 대한 액세스 권한을 부여할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 역할을 만들어 IAM 사용자에게 권한 위임 단원을 참조하십시오. 버킷 정책에 보안 주체 지정에 대한 자세한 내용은 Principal (p. 341)을 참조하십시오.	2011년 6월 6일
새로운 링크	이제 이 서비스의 엔드포인트 정보는 AWS General Reference에서 찾을 수 있습니다. 자세한 내용은 AWS General Reference의 Regions and Endpoints를 참조하십시오.	2011년 3월 1일

변경 사항	설명	날짜
Amazon S3에서의 정적 웹 사이트 호스팅에 대한 지원	Amazon S3에 정적 웹 사이트 호스팅에 대한 향상된 지원 기능이 도입되었습니다. 여기에는 인덱스 문서 및 사용자 정의 오류 문서에 대한 지원이 포함됩니다. 이러한 기능을 사용하면 버킷 또는 하위 폴더(예: `http://mywebsite.com/subfolder`)의 루트에 대한 요청 시 버킷의 객체 목록 대신 인덱스 문서가 반환됩니다. 오류가 발생하는 경우 Amazon S3에서 Amazon S3 오류 메시지 대신 사용자 정의 오류 메시지를 반환합니다. 자세한 내용은 Amazon S3 정적 웹 사이트 호스팅 (p. 548) 단원을 참조하십시오.	2011년 17월 2일
응답 헤더 API 지원	이제 GET Object REST API를 사용하면 각 요청에 대해 REST GET Object 요청의 응답 헤더를 변경할 수 있습니다. 즉, 객체 자체를 변경할 필요 없이 응답에서 객체 메타데이터를 변경할 수 있습니다. 자세한 내용은 객체 가져오기 (p. 164) 단원을 참조하십시오.	2011년 1월 14일
대형 객체 지원	Amazon S3에서 S3 버킷에 저장할 수 있는 최대 객체 크기를 5GB에서 5TB로 늘렸습니다. REST API를 사용하는 경우 단일 PUT 작업으로 최대 5GB 크기의 객체를 업로드할 수 있습니다. 더 큰 객체의 경우 멀티파트 업로드 REST API를 사용하여 객체를 여러 부분으로 나누어 업로드해야 합니다. 자세한 내용은 멀티파트 업로드 API를 사용한 객체 업로드 (p. 177) 단원을 참조하십시오.	2010년 12월 9일
멀티파트 업로드	멀티파트 업로드 기능을 사용하면 Amazon S3로 더 빠르고 더 유연하게 항목을 업로드할 수 있습니다. 또한 단일 객체를 여러 부분으로 구성된 한 세트로 업로드할 수 있습니다. 자세한 내용은 멀티파트 업로드 API를 사용한 객체 업로드 (p. 177) 단원을 참조하십시오.	2010년 11월 10일
버킷 정책에서의 정식 ID 지원	이제 버킷 정책에서 정식 ID를 지정할 수 있습니다. 자세한 내용은 Amazon S3의 정책 및 권한 (p. 339) 단원을 참조하십시오.	2010년 9월 17일
Amazon S3가 IAM과 함께 작동함	이 서비스는 이제 AWS Identity and Access Management(IAM)와 통합되었습니다. 자세한 내용은 IAM 사용 설명서의 IAM으로 작업하는 AWS 서비스 단원을 참조하십시오.	2010년 9월 2일
알림	Amazon S3 알림 기능을 사용하면 Amazon S3에서 버킷에 대한 주요 이벤트를 감지할 때 Amazon S3에서 Amazon SNS(Amazon Simple Notification Service) 주제에 대한 메시지를 게시하도록 버킷을 구성할 수 있습니다. 자세한 내용은 Setting Up Notification of Bucket Events (p. 587)를 참조하십시오.	2010년 7월 14일
버킷 정책	버킷 정책은 버킷, 객체 및 객체 세트에 대한 액세스 권한을 설정하기 위해 사용하는 액세스 관리 시스템입니다. 이 기능은 액세스 통제 목록을 보완하며, 많은 경우 이를 대체합니다. 자세한 내용은 버킷 정책 및 사용자 정책 사용 (p. 338) 단원을 참조하십시오.	2010년 7월 6일
모든 리전에서 경로 방식의 구문 사용 가능	Amazon S3에서는 이제 미국 클래식 리전의 모든 버킷에 대해, 또는 버킷이 요청의 엔드포인트와 같은 리전에 있는 경우에 경로 방식의 구문이 지원됩니다. 자세한 내용은 Virtual Hosting (p. 43)을 참조하십시오.	2010년 6월 9일

변경 사항	설명	날짜
유럽(아일랜드)의 새 엔드포인트	이제 Amazon S3가 유럽(아일랜드)에 엔드포인트를 제공합니다. `http://s3-eu-west-1.amazonaws.com`.	2010년 6월 9일
콘솔	이제 AWS Management 콘솔을 통해 Amazon S3를 사용할 수 있습니다. Amazon Simple Storage Service 콘솔 사용 설명서에서 콘솔의 모든 Amazon S3 기능에 대해 읽어 볼 수 있습니다.	2010년 6월 9일
중복성 감소	Amazon S3에서는 이제 중복성이 낮은 Amazon S3에 객체를 저장하여 스토리지 비용을 절감할 수 있습니다. 자세한 내용은 Reduced Redundancy Storage (p. 6)를 참조하십시오.	2010년 5월 12일
새로운 리전 지원	Amazon S3에서는 이제 아시아 태평양(싱가포르) 리전을 지원합니다. 자세한 내용은 버킷 및 리전 (p. 53) 단원을 참조하십시오.	2010년 4월 28일
객체 버전 관리	이 릴리스에서는 객체 버전 관리가 도입되었습니다. 이제 모든 객체에 키와 버전을 지정할 수 있습니다. 버킷에 대해 버전 관리를 활성화하면 Amazon S3에서 버킷에 추가되는 모든 객체에 고유한 버전 ID를 지정합니다. 이 기능을 사용하면 의도하지 않은 덮어쓰기 및 삭제로부터 복구할 수 있습니다. 자세한 내용은 버전 관리 (p. 7) 및 Using Versioning (p. 468)을 참조하십시오.	2010년 2월 8일
새로운 리전 지원	Amazon S3에서는 이제 미국 서부(캘리포니아 북부 지역) 리전을 지원합니다. 이 리전에 대한 요청의 새로운 엔드포인트는 s3-us-west-1.amazonaws.com입니다. 자세한 내용은 Buckets and Regions (p. 53)를 참조하십시오.	2009년 12월 2일
.NET용 AWS SDK	AWS에서는 이제 REST나 SOAP 대신 .NET 언어별 API 작업를 사용하여 애플리케이션을 빌드하는 것을 선호하는 소프트웨어 개발자를 위해 라이브러리, 샘플 코드, 자습서 및 기타 리소스를 제공합니다. 이러한 라이브러리에서는 쉽게 시작할 수 있도록 요청 인증, 요청 재시도 및 오류 처리와 같이 REST 또는 SOAP API에 없는 기본 기능을 제공합니다. 언어별 라이브러리 및 리소스에 대한 자세한 내용은 AWS SDK, CLI 및 Explorer 사용 (p. 727)을 참조하십시오.	2009년 11월 11일

AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the AWS General Reference.