aws

사용 설명서

Amazon Simple Storage Service



API 버전 2006-03-01

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Simple Storage Service: 사용 설명서

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon의 상표 및 브랜드 디자인은 Amazon 외 제품 또는 서비스와 함께, Amazon 브랜드 이미지를 떨어뜨리거나 고객에게 혼동을 일으킬 수 있는 방식으로 사용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 해당 상표의 소유자의 자산이며, 해당 상표의 소유자가 Amazon의 계열사이거나 Amazon과 제휴 관계에 있거나 Amazon의 후원을 받는 업체일 수 있습니다.

Table of Contents

Amazon S3란 무엇인가요? 1
Amazon S3의 기능 1
스토리지 클래스 1
스토리지 관리 2
액세스 관리 및 보안 2
데이터 처리 3
스토리지 로깅 및 모니터링 4
분석 및 인사이트 4
강력한 일관성 5
Amazon S3 작동 방식 5
버킷 6
Objects 8
키 8
S3 버전 관리 9
버전 ID 9
버킷 정책 9
S3 액세스 포인트 10
액세스 제어 목록(ACL) 10
리전 11
Amazon S3 데이터 일관성 모델 11
동시 애플리케이션 12
관련 서비스 14
Amazon S3 액세스 14
AWS Management Console 15
AWS Command Line Interface 15
AWS SDK 15
Amazon S3 REST API 15
Amazon S3 요금 지불 16
PCI DSS 준수 17
시작 18
설정 18
AWS 계정에 등록 19
관리자 액세스 권한이 있는 사용자 생성 19
1단계: 버킷 만들기 20

2단계: 객체 업로드 27
3단계: 객체 다운로드 27
S3 콘솔 사용 28
4단계: 객체 복사 29
5단계: 객체 및 버킷 삭제 30
객체 삭제 31
버킷 비우기 31
버킷 삭제 31
다음 단계 32
일반적인 사용 사례 이해 33
버킷 및 객체에 대한 액세스 제어 33
스토리지 보호 및 모니터링 34
Amazon S3를 사용한 개발 34
자습서로 배우기 36
교육 및 지원 살펴보기 37
AWS CLI에서 Amazon S3 사용 37
설정 38
1단계: 첫 번째 Amazon S3 버킷 생성 38
2단계: 버킷에 객체 업로드 45
3단계: 객체 다운로드 45
4단계: 폴더에 객체 복사 46
5단계: 객체 및 버킷 삭제 47
다음 단계 50
범용 버킷 작업 52
범용 버킷 개요 53
범용 버킷 개요 54
일반적인 범용 버킷 패턴 55
권한 55
범용 버킷에 대한 퍼블릭 액세스 관리 55
범용 버킷 구성 56
범용 버킷 작업 59
범용 버킷 성능 모니터링 59
일반적인 버킷 패턴 60
다중 테넌트 범용 버킷 패턴 61
사용당 버킷 패턴 61
이름 지정 규칙 62

범용 버킷 이름 지정 규칙 63
범용 버킷 이름 예시 64
모범 사례 65
버킷 이름에 GUID를 사용하는 버킷 생성 66
할당량, 규제 및 제한 67
버킷 할당량 67
객체 및 버킷 제한 68
버킷 이름 지정 규칙 68
버킷 액세스 68
..... 68
범용 버킷의 가상 호스팅 71
범용 버킷 생성 80
범용 버킷 설정 80
버킷 속성 보기 93
버킷 나열 96
범용 버킷 비우기 102
AWS CloudTrail가 구성된 범용 버킷 비우기 104
범용 버킷 삭제 105
Mountpoint for Amazon S3 110
Mountpoint 설치 111
Mountpoint 구성 및 사용 116
Mountpoint 문제 해결 122
Storage Browser for Amazon S3 122
Storage Browser for S3 사용 123
Storage Browser for S3 설치 124
Storage Browser for S3 설정 125
Storage Browser for S3 구성 138
Storage Browser for S3 문제 해결 139
Transfer Acceleration 구성 139
Transfer Acceleration을 사용하는 이유는 무엇입니까? 140
Transfer Acceleration을 사용하기 위한 요구 사항 140
시작 141
Transfer Acceleration 사용 설정 143
속도 비교 도구 150
요청자 지불 사용 151
요청자 지불 요금의 방식 152

요청자 지불 구성 ... 153
requestPayment 구성 검색 ... 155
요청자 지불 버킷에서 객체 다운로드 ... 155
객체 작업 ... 158
객체 개요 ... 159
하위 리소스 ... 160
객체 이름 지정 ... 161
객체 키 이름 선택 ... 161
객체 키 명명 지침 ... 162
메타데이터 작업 ... 166
시스템 정의 객체 메타데이터 ... 167
사용자 정의 객체 메타데이터 ... 170
객체 메타데이터 편집 ... 171
데이터 검색 가속화 ... 175
객체 업로드 ... 211
객체 업로드 ... 212
동일한 키 이름을 가진 객체 업로드 방지 ... 224
멀티파트 업로드 사용 ... 225
조건부 요청 생성 ... 297
메타데이터를 기반으로 객체를 검색하거나 복사하는 방법 ... 298
객체 덮어쓰기 방지 방법 ... 300
객체 복사, 이동, 이름 변경 ... 310
객체 복사 ... 313
객체를 이동하려면 ... 323
객체의 이름을 바꾸려면 ... 325
객체 다운로드 ... 326
객체 다운로드 ... 327
여러 객체 다운로드 ... 328
객체의 일부 다운로드 ... 330
다른 AWS 계정에서 객체 다운로드 ... 331
아카이브된 객체 다운로드 ... 332
메타데이터를 기반으로 한 객체 다운로드 ... 332
객체 다운로드 문제 해결 ... 332
Amazon S3에서 객체 무결성 확인 ... 333
지원되는 체크섬 알고리즘 사용 ... 333
전체 객체 및 복합 체크섬 유형 ... 334

멀티파트 업로드에 전체 객체 체크섬 사용 ... 336
멀티파트 업로드에 파트 수준의 체크섬 사용 ... 337
체크섬 작업 ... 338
객체를 업로드할 때 Content-MD5 사용 ... 346
Content-MD5 및 ETag를 사용하여 업로드된 객체 확인 ... 346
후행 체크섬 사용 ... 347
객체 삭제 ... 352
객체를 삭제하기 전에 고려해야 할 모범 사례 ... 353
버전 관리가 활성화된 버킷에서 객체 삭제 ... 354
버전 관리가 일시 중지된 버킷에서 객체 삭제 ... 354
버전이 지정되지 않은 버킷에서 객체 삭제 ... 354
MFA를 사용하는 버킷에서 객체 삭제 ... 355
단일 객체 삭제 ... 355
여러 객체 삭제 ... 367
개체 구성 및 나열 ... 370
접두어 사용 ... 370
객체 나열 ... 372
폴더 사용 ... 375
객체 속성 보기 ... 380
태그로 객체 분류 ... 381
미리 서명된 URL을 사용하여 객체 다운로드 및 업로드 ... 394
미리 서명된 URL을 생성할 수 있는 사용자 ... 395
미리 서명된 URL의 만료 시간 ... 396
미리 서명된 URL 기능 제한 ... 396
미리 서명된 URL을 통해 객체 공유 ... 398
미리 서명된 URL을 통해 객체 공유 ... 403
객체 변환 ... 406
객체 Lambda 액세스 포인트 생성 ... 408
Amazon S3 객체 Lambda 액세스 포인트 사용 ... 422
보안 고려 사항 ... 426
Lambda 함수 작성 ... 432
AWS 빌드 함수 사용 ... 463
S3 객체 Lambda에 대한 모범 사례 및 지침 ... 465
S3 객체 Lambda 자습서 ... 466
S3 Object Lambda 문제 해결 ... 502
대량으로 객체 작업 수행 ... 503

배치 작업 기본 사항 ... 504
S3 배치 작업 자습서 ... 505
권한 부여 ... 505
작업 생성 ... 516
지원되는 작업 ... 537
작업 관리 ... 577
작업 상태 및 완료 보고서 추적 ... 582
태그 사용 ... 596
Batch Operations를 사용하여 Object Lock 관리 ... 611
자습서: 비디오 일괄 트랜스코딩 ... 636
Batch Operations 문제 해결 ... 675
데이터 쿼리 적용 ... 677
요구 사항 및 제한 ... 678
요청의 구성 ... 679
오류 ... 680
S3 Select 예제 ... 680
SQL 참조 ... 685
디렉터리 버킷 작업 ... 724
디렉터리 버킷 이름 ... 725
디렉터리 ... 725
키 이름 ... 725
액세스 관리 ... 726
디렉터리 버킷 할당량 ... 726
디렉터리 버킷 생성 및 사용 ... 726
디렉터리 버킷 사용 사례 ... 727
고성능 워크로드 ... 728
데이터 레지던시 워크로드 ... 779
디렉터리 버킷의 차이점 ... 795
디렉터리 버킷의 차이점 ... 795
디렉터리 버킷에 지원되는 API 작업 ... 797
디렉터리 버킷에서 지원되지 않는 Amazon S3 기능 ... 797
디렉터리 버킷에 대한 네트워킹 ... 798
엔드포인트 ... 799
VPC 게이트웨이 엔드포인트 구성 ... 799
디렉터리 버킷 이름 지정 규칙 ... 799
속성 보기 ... 801

버킷 정책 관리 ... 801
버킷 정책 추가 ... 802
버킷 정책 보기 ... 805
버킷 정책 삭제 ... 805
디렉터리 버킷 비우기 ... 806
디렉터리 버킷 삭제 ... 807
디렉터리 버킷 나열 ... 810
버킷 액세스 가능 여부 결정 ... 813
디렉터리 버킷의 객체 작업 ... 814
디렉터리 버킷으로 객체 가져오기 ... 814
S3 수명 주기 작업 ... 816
S3 Express One Zone에서 배치 작업 사용 ... 828
객체에 데이터 추가 ... 831
객체 이름 바꾸기 ... 833
객체 업로드 ... 840
객체 복사 ... 871
객체 삭제 ... 877
객체 다운로드 ... 881
객체 공유를 위해 미리 서명된 URL 생성 ... 883
객체 메타데이터 검색 ... 883
객체 나열 ... 884
디렉터리 버킷의 보안 ... 885
데이터 보호 및 암호화 ... 885
요청 인증 및 권한 부여 ... 906
보안 모범 사례 ... 916
디렉터리 버킷에 대한 액세스 포인트 작업 ... 919
이름 지정 규칙, 제한 및 한계 ... 920
디렉터리 버킷의 액세스 포인트 참조 ... 922
액세스 포인트 객체 작업 ... 922
IAM 정책 구성 ... 923
모니터링 및 로깅 ... 929
디렉터리 버킷의 액세스 포인트 만들기 ... 930
액세스 포인트 관리 ... 934
AWS CloudTrail로 디렉터리 버킷에 로그인 ... 946
디렉터리 버킷에 대한 CloudTrail 관리 이벤트 ... 946
디렉터리 버킷에 대한 CloudTrail 데이터 이벤트 ... 947

..... 948
디렉터리 버킷 성능 최적화 955
세션 기반 인증 사용 955
S3 추가 체크섬 모범 사례 955
최신 버전의 AWS SDK 및 공용 런타임 라이브러리를 사용하세요. 956
디렉터리 버킷으로 배포 956
디렉터리 버킷에 대한 리전 및 영역 엔드포인트 956
S3 콘솔, AWS CLI, AWS SDK를 통한 디렉터리 버킷 작업 957
디렉터리 버킷 API 작업 959
디렉터리 버킷 태그 지정 961
태그 작동 방식 961
태그를 사용하는 일반적인 방법 962
태그 작업 967
태그가 있는 디렉터리 버킷 생성 968
디렉터리 버킷에 태그 추가 972
디렉터리 버킷 태그 보기 975
디렉터리 버킷에서 객체 삭제 978
Amazon S3 Tables 및 테이블 버킷 작업 981
S3 Tables의 특징 981
관련 서비스 983
자습서: S3 Tables 시작하기 983
1단계: 테이블 버킷 생성 및 AWS 분석 서비스와 통합 984
2단계: 테이블 네임스페이스 및 테이블 생성 985
(옵션) 3단계: 테이블에 대한 Lake Formation 권한 부여 988
4단계: Athena에서 SQL을 사용하여 데이터 쿼리 989
테이블 버킷 990
테이블 버킷 이름 지정 규칙 991
테이블 버킷 생성 993
테이블 버킷 삭제 995
테이블 버킷 세부 정보 보기 996
정책 관리 996
S3 Tables 유지 관리 997
S3 Tables 유지 관리 작업 상태 998
테이블 버킷 유지 관리 999
테이블 유지 관리 1001
고려 사항 및 제한 사항 1006

네임스페이스 1008
네임스페이스 생성 1008
네임스페이스 삭제 1010
표 1011
테이블 생성 1012
테이블 삭제 1017
정책 관리 1018
테이블에 액세스 1020
Amazon SageMaker Lakehouse 통합을 통해 테이블에 액세스 1020
테이블에 직접 액세스 1021
AWS 분석 서비스에서 S3 Tables 사용 1022
AWS Glue Iceberg REST 엔드포인트를 사용하여 테이블에 액세스 1034
Amazon S3 Tables Iceberg REST 엔드포인트를 사용하여 테이블에 액세스 1039
클라이언트 카탈로그를 사용하여 테이블에 액세스 1045
Amazon Athena 1047
Amazon Redshift 1049
Amazon EMR 1050
QuickSight 1054
Amazon Data Firehose 1056
AWS Glue ETL 1061
AWS 리전, 엔드포인트 및 할당량 1072
S3 Tables AWS 리전 및 엔드포인트 1072
S3 Tables 할당량 1077
S3 Tables의 보안 1077
암호화 1079
액세스 관리 1091
VPC 연결 1119
규제 및 제한 1122
S3 Tables에서 AWS CloudTrail을 사용하여 로깅 1123
S3 Tables에 대한 CloudTrail 관리 이벤트 1123
S3 Tables에 대한 CloudTrail 데이터 이벤트 1124
CloudTrail 로그 예제 1125
액세스 제어 1129
S3 리소스 1129
ID 1134
버킷 또는 리소스 소유자 1136

액세스 관리 도구 1136
작업 1141
액세스 관리 사용 사례 1142
액세스 관리 문제 해결 1149
Identity and Access Management(IAM) 1150
대상 1151
ID를 통한 인증 1151
정책을 사용하여 액세스 관리 1154
Amazon S3에서 IAM을 사용하는 방법 1157
권한 부여 요청 1182
S3 API 작업에 필요한 권한 1190
정책 및 권한 1224
버킷 정책 1227
ID 기반 정책 1272
정책을 사용하는 연습 1306
Amazon S3 스토리지 렌즈에 대한 서비스 연결 역할 사용 1344
Amazon S3 자격 증명 및 액세스 문제 해결 1348
AWS 관리형 정책 1371
액세스 포인트 작업 1373
이름 지정 규칙, 제한 및 한계 1374
액세스 포인트 참조 1377
액세스 포인트 호환성 1380
IAM 정책 구성 1383
모니터링 및 로깅 1390
액세스 포인트 생성 1392
액세스 포인트 관리 1400
액세스 포인트 사용 1405
S3 Access Grants를 통한 액세스 관리 1416
S3 Access Grants의 개념 1418
S3 Access Grants 및 기업 디렉터리 ID 1422
S3 Access Grants 시작하기 1430
S3 Access Grants 인스턴스 사용 1432
S3 Access Grants 위치 사용 1444
S3 Access Grants에서 권한 부여 사용 1466
액세스 권한 부여를 사용하여 S3 데이터 가져오기 1479
S3 Access Grants 크로스 계정 액세스 1494

S3 Access Grants에 AWS 태그 사용 1506
S3 Access Grants 한도 1509
S3 Access Grants 통합 1511
ACL을 사용한 액세스 관리 1512
ACL 개요 1513
ACL 구성 1531
정책 예시 1548
퍼블릭 액세스 차단 1552
퍼블릭 액세스 차단 설정 1554
액세스 포인트에서 퍼블릭 액세스 차단 작업 수행 1557
"퍼블릭"의 의미 1558
IAM Access Analyzer for S3를 사용하여 퍼블릭 버킷 검토 1561
권한 1562
퍼블릭 액세스 차단 구성 1562
계정 설정 구성 1563
버킷 및 액세스 포인트 설정 구성 1565
버킷 액세스 검토 1568
IAM Access Analyzer for S3는 어떤 정보를 제공합니까? 1570
IAM Access Analyzer for S3 활성화 1571
모든 퍼블릭 액세스 차단 1571
버킷 액세스 검토 및 변경 1572
버킷 결과 보관 1573
보관된 버킷 결과 활성화 1574
결과 세부 정보 보기 1574
IAM Access Analyzer for S3 보고서 다운로드 1574
버킷 소유권 확인 1575
버킷 소유자 조건을 사용해야 하는 경우 1576
버킷 소유자 확인 1576
예시 1577
규제 및 제한 1580
객체 소유권 제어 1580
객체 소유권 설정 1582
ACL 사용 중지로 인한 변경 사항 1583
ACL 사용 중지를 위한 사전 조건 1585
객체 소유권 권한 1586
모든 새 버킷에 대해 ACL 사용 중지 1586

복제 및 객체 소유권 1586
객체 소유권 설정 1586
ACL 사용 중지를 위한 사전 조건 1587
버킷 생성 1603
객체 소유권 설정 1611
객체 소유권 설정 보기 1614
모든 새 버킷에 대해 ACL 사용 중지 1616
문제 해결 1618
보안 1621
보안 모범 사례 1623
Amazon SNS 보안 모범 사례 1623
Amazon S3 모니터링 및 감사 모범 사례 1628
데이터 보안 모니터링 1632
데이터 보호 1635
데이터 암호화 1636
서버 측 암호화 1638
클라이언트 측 암호화 사용 1730
인터네트워크 트래픽 개인 정보 1731
서비스와 온프레미스 클라이언트 및 애플리케이션 간의 트래픽 1731
같은 리전에 있는 AWS 리소스 사이의 트래픽 1732
Amazon S3용 AWS PrivateLink 1732
규정 준수 확인 1749
복원성 1751
백업 암호화 1753
인프라 보안 1754
구성 및 취약성 분석 1755
액세스 관리 1755
데이터 보호 1758
리전 내 및 리전 간 객체 복제 1760
복제를 사용하는 이유 1761
교차 리전 복제를 사용하는 경우 1762
동일 리전 복제를 사용하는 경우 1762
양방향 복제를 사용해야 하는 경우 1763
S3 Batch Replication을 사용하는 경우 1763
워크로드 요구 사항 및 라이브 복제 1763
복제 가능한 객체 1764

복제본에 대한 요구 사항 및 고려 사항 1767
라이브 복제 설정 1771
라이브 복제 관리 또는 일시 중지 1860
기존 객체 복제 1861
복제 문제 해결 1875
진행 상황 모니터링 및 상태 확인 1882
다중 리전 트래픽 관리 1903
다중 리전 액세스 포인트 생성 1905
다중 리전 액세스 포인트 구성 1913
다중 리전 액세스 포인트 사용 1917
여러 버전의 객체 유지 1965
버전 관리 미사용 버킷, 버전 관리가 사용 설정된 버킷 및 버전 관리가 일시 중지된 버킷 1965
S3 수명 주기와 함께 S3 버전 관리 사용 1966
S3 버전 관리 1967
버킷에 버전 관리 사용 설정 1971
MFA Delete 구성 1978
버전 관리가 사용 설정된 객체 작업 1981
버전 관리가 일시 중지된 객체 작업 2009
버전 관리 문제 해결 2013
객체 잠금 2017
S3 객체 잠금 작동 방식 2018
객체 잠금 고려 사항 2022
객체 잠금 구성 2027
데이터 백업 2037
비용 최적화 2039
결제 및 사용 보고 2040
비용 할당 태그 사용 2040
결제 보고서 2042
사용 보고서 2045
결제 및 사용 보고서 이해 2047
Amazon S3 오류 응답에 대한 청구 2070
스토리지 클래스 이해 및 관리 2088
자주 액세스하는 객체 2088
변경되는 또는 알 수 없는 액세스 패턴으로 데이터 자동 최적화 2089
자주 액세스하지 않는 객체 2091
거의 액세스하지 않는 객체 2092

Outposts에서의 Amazon S3 ... 2093
스토리지 클래스 비교 ... 2094
객체의 스토리지 클래스 설정 ... 2095
스토리지 클래스 분석 ... 2099
Amazon S3 Intelligent-Tiering을 사용한 스토리지 비용 관리 ... 2106
Amazon S3 Glacier 스토리지 클래스 ... 2119
아카이브된 객체 작업 ... 2124
수명 주기 관리 ... 2136
객체의 전체 수명 주기 관리 ... 2138
객체 전환 ... 2139
객체 만료 ... 2146
수명 주기 구성 설정 ... 2149
다른 버킷 구성 사용 ... 2168
S3 수명 주기 이벤트 알림 구성 ... 2170
수명 주기 구성의 요소 ... 2172
수명 주기 구성의 충돌 ... 2189
S3 수명 주기 구성의 예제 ... 2194
수명 주기 문제 해결 ... 2208
로깅 및 모니터링 ... 2215
모니터링 도구 ... 2218
자동 도구 ... 2218
수동 도구 ... 2219
로깅 옵션 ... 2220
CloudTrail을 사용하여 로깅 ... 2222
Amazon S3 서버 액세스 로그 및 CloudWatch Logs와 함께 CloudTrail 로그 사용 ... 2224
Amazon S3 SOAP API 호출을 통한 CloudTrail 추적 ... 2225
CloudTrail 이벤트 ... 2226
로그 파일의 예 ... 2237
CloudTrail 사용 설정 ... 2241
S3 요청 식별 ... 2245
서버 액세스 로깅 ... 2252
로그 전송을 사용 설정하려면 어떻게 해야 합니까? ... 2252
로그 객체 키 형식 ... 2255
로그 전송 방법 ... 2255
서버 로그 전송이 항상 보장되지는 않음 ... 2256
버킷 로깅 상태 변경 시 일정 기간에 걸쳐 단계적으로 반영됨 ... 2257

서버 액세스 로깅 사용 설정 2257
로그 형식 2279
로그 파일 삭제 2293
S3 요청 식별 2294
서버 액세스 로깅 문제 해결 2302
CloudWatch를 사용한 지표 모니터링 2304
지표 및 차원 2306
CloudWatch 지표 액세스 2323
CloudWatch 지표 구성 2325
Amazon S3 이벤트 알림 2333
개요 2333
알림 유형 및 대상 2335
SQS, SNS 및 Lambda 사용 2342
EventBridge 사용 2371
스토리지 활동 및 사용량 시각화 2380
S3 스토리지 렌즈의 지표 및 기능 2381
S3 스토리지 렌즈 이해 2383
지표 용어집 2393
권한 설정 2429
S3 스토리지 렌즈 작업 2432
스토리지 지표 보기 2471
조직 작업 2524
S3 Storage Lens 그룹 사용 2535
데이터 카탈로그화 및 분석 2577
Amazon S3 인벤토리 버킷 2578
인벤토리 목록 2579
Amazon S3 인벤토리 구성 2584
인벤토리 찾기 2593
인벤토리 완료에 대한 알림 설정 2597
Athena로 인벤토리 쿼리 2597
빈 버전 ID 문자열을 null 문자열로 변환 2603
객체 ACL 필드 작업 2606
성능 최적화 2609
Amazon S3 성능 지침 2610
성능 측정 2610
수평 규모 조정 2611

바이트 범위 가져오기 사용 2611
요청 재시도 2612
동일한 리전에서 Amazon S3와 Amazon EC2 결합 2612
지연 시간을 최소화하기 위해 Transfer Acceleration 사용 2612
최신 AWS SDK 사용 2612
Amazon S3의 성능 디자인 패턴 2613
자주 액세스하는 콘텐츠 캐싱 2613
지연 시간에 민감한 앱의 제한 시간 및 재시도 2614
수평적 스케일링 및 요청 병렬화 2615
지리적으로 다른 데이터 전송 가속화 2616
정적 웹 사이트 호스팅 2617
웹 사이트 엔드포인트 2618
웹 사이트 엔드포인트 예제 2619
DNS CNAME 추가 2620
Route 53에서 사용자 지정 도메인 사용 2620
웹 사이트 엔드포인트와 REST API 엔드포인트 간의 주요 차이점 2620
웹 사이트 호스팅 사용 설정 2621
인덱스 문서 구성 2626
인덱스 문서 및 폴더 2627
인덱스 문서 구성 2628
사용자 지정 오류 문서 구성 2629
Amazon S3 HTTP 응답 코드 2630
사용자 지정 오류 문서 구성 2632
웹 사이트 액세스에 대한 권한 설정 2633
1단계: S3 퍼블릭 액세스 차단 설정 편집 2634
2단계: 버킷 정책 추가 2635
객체 ACL(액세스 제어 목록) 2637
웹 트래픽 로깅 2638
리디렉션 구성 2639
다른 호스트로 요청 리디렉션 2639
리디렉션 규칙 구성 2640
객체에 대한 요청 리디렉션 2648
CORS 사용 2650
CORS(Cross-Origin Resource Sharing): 사용 사례 시나리오 2650
Amazon S3은 버킷의 CORS 구성을 어떻게 평가하나요? 2651
객체 Lambda 액세스 포인트가 CORS를 지원하는 방법 2651

CORS 구성의 요소 2651
CORS 구성 2657
CORS 테스트 2665
CORS 문제 해결 2668
정적 웹 사이트 자습서 2672
비디오 스트리밍 호스팅 2674
정적 웹 사이트 구성 2692
사용자 지정 도메인을 사용하여 정적 웹 사이트 구성 2701
Amazon S3에서 Amplify에 정적 웹 사이트 배포 2724
할당량 2727
할당량 증가 2727
레퍼런스 2728
문서 기록 2729
이전 업데이트 2769

Amazon S3란 무엇인가요?

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

> **Note**
>
> Amazon S3 Express One Zone 스토리지 클래스를 디렉터리 버킷과 함께 사용하는 방법에 대한 자세한 내용은 S3 Express One Zone 및 디렉터리 버킷 작업 섹션을 참조하세요.

주제

- Amazon S3의 기능
- Amazon S3 작동 방식
- Amazon S3 데이터 일관성 모델
- 관련 서비스
- Amazon S3 액세스
- Amazon S3 요금 지불
- PCI DSS 준수

Amazon S3의 기능

스토리지 클래스

Amazon S3는 여러 사용 사례에 맞춰 설계된 다양한 스토리지 클래스를 제공합니다. 예를 들어 자주 액세스하기 위해 미션 크리티컬 프로덕션 데이터를 S3 Standard 또는 S3 Express One Zone에 저장하고, 액세스 빈도가 낮은 데이터를 S3 Standard-IA 또는 S3 One Zone-IA에 저장하여 비용을 절감하고, S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive에 가장 낮은 비용으로 데이터를 보관할 수 있습니다.

S3 Express One Zone은 지연 시간에 가장 민감한 애플리케이션에 대해 일관되게 10밀리초 미만의 데이터 액세스를 제공하도록 특별히 설계된 고성능 단일 영역 Amazon S3 스토리지 클래스입니다. S3 Express One Zone은 현재 사용 가능한 클라우드 객체 스토리지 클래스 중 지연 시간이 가장 낮으며, S3 Standard보다 데이터 액세스 속도는 최대 10배 빠르고 요청 비용은 50% 저렴합니다. S3 Express One Zone은 객체 스토리지를 컴퓨팅 리소스와 함께 배치하는 옵션을 제공하면서 단일 가용 영역 선택이 가능한 최초의 S3 스토리지 클래스로, 최고의 액세스 속도를 제공합니다. 또한 액세스 속도를 더욱 높이고 초당 수십만 건의 요청을 지원하기 위해 데이터는 새로운 버킷 유형인 Amazon S3 디렉터리 버킷에 저장됩니다. 자세한 내용은 S3 Express One Zone 및 디렉터리 버킷 작업(을)를 참조하세요.

액세스 패턴이 변경되거나 알 수 없는 액세스 패턴이 있는 데이터를 S3 Intelligent-Tiering에 저장할 수 있습니다. 이렇게 하면 액세스 패턴이 변경될 때 4개의 액세스 계층 간에 데이터를 자동으로 이동하여 스토리지 비용을 최적화할 수 있습니다. 4개의 액세스 계층에는 빈번한 액세스와 간헐적인 액세스에 최적화된 2개의 대기 시간이 짧은 액세스 계층과, 비동기 액세스용으로 설계되어 드문 액세스에 최적화된 2개의 옵트인 아카이브 액세스 계층이 포함되어 있습니다.

자세한 내용은 Amazon S3 스토리지 클래스 이해 및 관리 섹션을 참조하세요.

스토리지 관리

Amazon S3에는 비용 관리, 규정 요구 사항 충족, 대기 시간 단축, 규정 준수 요구 사항에 맞게 여러 개의 개별 데이터 복제본 저장을 수행할 수 있는 스토리지 관리 기능이 포함되어 있습니다.

- S3 수명 주기 - 수명 주기 구성을 설정하여 객체를 관리하고 수명 주기 동안 객체를 비용 효율적으로 저장할 수 있습니다. 객체를 다른 S3 스토리지 클래스로 전환하거나 수명이 다한 객체를 만료시킬 수 있습니다.
- S3 객체 잠금 - 고정된 시간 동안 또는 무기한으로 Amazon S3 객체의 삭제 또는 덮어쓰기를 방지할 수 있습니다. 객체 잠금을 사용하면 WORM(write-once-read-many) 스토리지가 필요한 규제 요구 사항을 충족하거나 객체 변경 및 삭제에 대한 보호 계층을 추가하는 데 도움이 됩니다.
- S3 복제 - 대기 시간 단축, 규정 준수, 보안 및 기타 사용 사례를 위해 객체, 객체의 각 메타데이터, 객체 태그를 동일하거나 다른 AWS 리전에 있는 하나 이상의 대상 버킷에 복제합니다.
- S3 배치 작업 - Amazon S3 콘솔에서 단일 S3 API 요청이나 몇 번의 클릭만으로 수십억 개의 객체를 대규모로 관리할 수 있습니다. 배치 작업(Batch Operations)을 사용하여 수백만 또는 수십억 개의 객체에 대해 복사, AWS Lambda 함수 호출 및 복원 등의 작업을 수행할 수 있습니다.

액세스 관리 및 보안

Amazon S3는 버킷 및 객체에 대한 액세스 감사 및 관리 기능을 제공합니다. 기본적으로 S3 버킷 및 객체는 프라이빗입니다. 생성한 S3 리소스에만 액세스할 수 있습니다. 특정 사용 사례를 지원하는 세분

화된 리소스 권한을 부여하거나 Amazon S3 리소스의 권한을 감사하기 위해 다음 기능을 사용할 수 있습니다.

- S3 퍼블릭 액세스 차단 - S3 버킷과 객체에 대한 퍼블릭 액세스를 차단합니다. 기본적으로 퍼블릭 액세스 차단 설정은 버킷 수준에서 켜져 있습니다. 특정 사용 사례에 대해 하나 이상의 설정을 해제해야 하는 경우가 아니라면 모든 퍼블릭 액세스 차단 설정을 활성화된 상태로 유지하는 것이 좋습니다. 자세한 내용은 S3 버킷에 대한 퍼블릭 액세스 차단 설정 구성 섹션을 참조하세요.
- AWS Identity and Access Management(IAM) – IAM은 Amazon S3 리소스를 포함하여 AWS 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 되는 웹 서비스입니다. IAM을 사용하면 사용자가 액세스할 수 있는 AWS 리소스를 제어하는 권한을 중앙에서 관리할 수 있습니다. IAM을 사용하여 리소스를 사용하도록 인증(로그인) 및 권한 부여(권한 있음)된 대상을 제어합니다.
- 버킷 정책 - IAM 기반 정책 언어를 사용하여 S3 버킷과 그 안에 있는 객체에 대한 리소스 기반 권한을 구성합니다.
- Amazon S3 액세스 포인트 – Amazon S3의 공유 데이터터 집합에 대한 데이터 액세스를 대규모로 관리하기 위해 전용 액세스 정책이 포함된 명명된 네트워크 엔드포인트를 구성합니다.
- 액세스 제어 목록(ACL) - 인증된 사용자에게 개별 버킷 및 객체에 대한 읽기 및 쓰기 권한을 부여합니다. 일반적으로 ACL 대신 액세스 제어를 위해 S3 리소스 기반 정책(버킷 정책 및 액세스 포인트 정책) 또는 IAM 사용자 정책을 사용하는 것이 좋습니다. 정책은 단순하고 더 유연한 액세스 제어 옵션입니다. 버킷 정책과 액세스 포인트 정책을 사용하면 Amazon S3 리소스에 대한 모든 요청에 광범위하게 적용되는 규칙을 정의할 수 있습니다. 리소스 기반 정책 또는 IAM 사용자 정책 대신 ACL을 사용하는 특정 사례에 대한 자세한 내용은 ACL을 사용한 액세스 관리 섹션을 참조하세요.
- S3 객체 소유권 - 버킷의 모든 객체에 대한 소유권을 가져와서 Amazon S3에 저장된 데이터에 대한 액세스 관리를 간소화합니다. S3 객체 소유권은 ACL을 비활성화 또는 활성화하는 데 사용할 수 있는 Amazon S3 버킷 수준 설정입니다. 기본적으로 ACL은 비활성화되어 있습니다. ACL이 비활성화되면 버킷 소유자는 버킷의 모든 객체를 소유하고 액세스 관리 정책을 사용하여 데이터에 대한 액세스를 독점적으로 관리합니다.
- IAM Access Analyzer for S3 - S3 버킷 액세스 정책을 평가 및 모니터링하여 정책이 S3 리소스에 대한 의도된 액세스만 제공하는지 확인합니다.

데이터 처리

데이터를 변환하고 워크플로를 트리거하여 다양한 다른 처리 작업을 대규모로 자동화하기 위해 다음 기능을 사용할 수 있습니다.

- S3 객체 Lambda - 자체 코드를 S3 GET, HEAD 및 LIST 요청에 추가하여 애플리케이션에 데이터가 반환될 때 데이터를 수정 및 처리할 수 있습니다. 행을 필터링하고, 이미지의 크기를 동적으로 조정하고, 기밀 데이터를 편집하는 등의 작업을 수행할 수 있습니다.
- 이벤트 알림 - S3 리소스가 변경되면 Amazon Simple Notification Service(Amazon SNS), Amazon Simple Queue Service(Amazon SQS) 및 AWS Lambda를 사용하는 워크플로를 트리거합니다.

스토리지 로깅 및 모니터링

Amazon S3는 Amazon S3 리소스가 사용되는 방식을 모니터링하고 제어하는 데 사용할 수 있는 로깅 및 모니터링 도구를 제공합니다. 자세한 내용은 모니터링 도구를 참조하세요.

자동 모니터링 도구

- Amazon S3용 Amazon CloudWatch 지표 - S3 리소스의 운영 상태를 추적하고 예상 요금이 사용자 정의 임계값에 도달하면 결제 알림을 구성합니다.
- AWS CloudTrail - Amazon S3 에서 사용자, 역할 또는 AWS 서비스에 의해 수행된 작업을 기록합니다. CloudTrail 로그는 S3 버킷 수준 및 객체 수준 작업에 대한 자세한 API 추적을 제공합니다.

수동 모니터링 도구

- 서버 액세스 로깅 - 버킷에 대해 이루어진 요청에 대한 상세 레코드를 제공합니다. 보안 및 액세스 감사 수행, 고객 기반 파악, Amazon S3 결제 내역 이해 등의 많은 사용 사례에 대해 서버 액세스 로그를 사용할 수 있습니다.
- AWS Trusted Advisor - AWS 인프라 최적화, 보안 및 성능 개선, 비용 절감, 서비스 할당량 모니터링 방법을 식별하기 위해 AWS 모범 사례를 확인하여 계정을 평가합니다. 그런 다음 권장 사항에 따라 서비스와 리소스를 최적화할 수 있습니다.

분석 및 인사이트

Amazon S3는 스토리지 사용량을 파악할 수 있는 기능을 제공하며, 이를 통해 규모에 따라 스토리지를 더 잘 이해하고 분석하며 최적화할 수 있습니다.

- Amazon S3 Storage Lens - 스토리지를 이해하고, 분석하며, 최적화할 수 있습니다. S3 Storage Lens는 60개 이상의 사용량 및 활동 지표와 대화형 대시보드를 제공하여 전체 조직, 특정 계정, AWS 리전, 버킷 또는 접두사에 대한 데이터를 집계합니다.

- 스토리지 클래스 분석 - 스토리지 액세스 패턴을 분석함으로써 데이터를 보다 비용 효율적인 스토리지 클래스로 이전할 시기를 결정할 수 있습니다.
- 인벤토리 보고서가 있는 S3 인벤토리 - 객체와 해당 메타데이터를 감사 및 보고하고 인벤토리 보고서에서 조치를 취하도록 다른 Amazon S3 기능을 구성합니다. 예를 들어 객체의 복제 및 암호화 상태를 보고할 수 있습니다. 인벤토리 보고서의 각 개체에 사용할 수 있는 모든 메타데이터 목록은 Amazon S3 인벤토리 목록을 참조하세요.

강력한 일관성

Amazon S3는 모든 AWS 리전의 Amazon S3 버킷에 있는 객체의 PUT 및 DELETE 요청에 대해 강력한 쓰기 후 읽기(read-after-write) 일관성을 제공합니다. 이는 새 객체에 대한 쓰기와, 기존 객체 및 DELETE 요청을 덮어쓰는 PUT 요청 모두에 적용됩니다. 또한 Amazon S3 Select, Amazon S3 액세스 제어 목록(ACL), Amazon S3 객체 태그, 객체 메타데이터(예: HEAD 객체)에 대한 읽기 작업은 매우 일관적입니다. 자세한 정보는 Amazon S3 데이터 일관성 모델을 참조하세요.

Amazon S3 작동 방식

Amazon S3는 데이터를 버킷 내에 객체, 계층적 데이터 또는 테이블 형식 데이터로 저장하는 객체 스토리지 서비스입니다. 객체는 해당 파일을 설명하는 모든 메타데이터입니다. 버킷은 객체에 대한 컨테이너입니다.

Amazon S3에 데이터를 저장하려면 먼저 버킷을 생성하고 버킷 이름 및 AWS 리전을 지정해야 합니다. 그런 다음 Amazon S3에서 객체로 해당 버킷에 데이터를 업로드합니다. 각 객체에는 키(또는 키 이름)가 있으며, 이는 버킷 내 객체에 대한 고유한 식별자입니다.

S3는 특정 사용 사례를 지원하도록 구성할 수 있는 기능을 제공합니다. 예를 들어, S3 버전 관리를 사용하여 동일한 버킷에 여러 버전의 객체를 보관하고, 실수로 삭제되거나 덮어쓰기된 객체를 복원할 수 있습니다.

버킷과 버킷의 객체는 프라이빗이며 액세스 권한을 명시적으로 부여한 경우에만 액세스할 수 있습니다. 버킷 정책, AWS Identity and Access Management(IAM) 정책, 액세스 제어 목록(ACL), S3 액세스 포인트를 사용하여 액세스를 관리할 수 있습니다.

주제

- 버킷
- Objects
- 키

- S3 버전 관리
- 버전 ID
- 버킷 정책
- S3 액세스 포인트
- 액세스 제어 목록(ACL)
- 리전

버킷

Amazon S3는 범용 버킷, 디렉터리 버킷, 테이블 버킷이라는 세 가지 유형의 버킷을 지원합니다. 각 버킷 유형은 다양한 사용 사례에 맞는 고유한 기능 세트를 제공합니다.

범용 버킷 - 범용 버킷은 대부분의 사용 사례 및 액세스 패턴에 권장되며 S3 버킷 본래의 유형입니다. 범용 버킷은 Amazon S3에 저장되는 객체의 컨테이너입니다. 버킷과 모든 스토리지 클래스(S3 Express One Zone 제외)에 원하는 수의 객체를 저장할 수 있으므로 여러 가용 영역에 걸쳐 객체를 중복해서 저장할 수 있습니다. 자세한 내용은 Amazon S3 범용 버킷 생성, 구성 및 작업을 참조하세요.

> **Note**
>
> 기본적으로 모든 범용 버킷은 프라이빗입니다. 하지만 범용 버킷에 대한 퍼블릭 액세스 권한을 부여할 수 있습니다. 버킷, 접두사(폴더) 또는 객체 태그 수준에서 범용 버킷에 대한 액세스를 제어할 수 있습니다. 자세한 내용은 Amazon S3의 액세스 제어를 참조하세요.

디렉터리 버킷 - 지연 시간이 짧은 사용 사례 및 데이터 레지던시 사용 사례에 권장됩니다. 기본적으로 AWS 계정에 최대 100개의 디렉터리 버킷을 생성할 수 있으며, 버킷에 저장할 수 있는 객체 수에는 제한이 없습니다. 디렉터리 버킷은 객체를 범용 버킷의 플랫 스토리지 구조 대신 계층적 디렉터리(접두사)로 구성합니다. 이 버킷 유형에는 접두사 제한이 없으며 개별 디렉터리의 규모를 수평적으로 조정할 수 있습니다. 자세한 내용은 디렉터리 버킷 작업을 참조하세요.

- 지연 시간이 짧은 사용 사례의 경우 단일 AWS 가용 영역에 디렉터리 버킷을 생성하여 데이터를 저장할 수 있습니다. 가용 영역의 디렉터리 버킷은 S3 Express One Zone 스토리지 클래스를 지원합니다. S3 Express One Zone 스토리지 클래스를 사용하면 단일 가용 영역 내의 여러 디바이스에 데이터가 중복으로 저장됩니다. S3 Express One Zone 스토리지 클래스는 애플리케이션이 성능에 민감하고 10밀리초 미만의 PUT 및 GET 지연 시간이 필요한 경우에 권장됩니다. 가용 영역에서 디렉터리 버킷을 생성하는 방법에 대한 자세한 내용은 고성능 워크로드를 참조하세요.

- 데이터 레지던시 사용 사례의 경우 단일 AWS 전용 로컬 영역(DLZ)에 디렉터리 버킷을 생성하여 데이터를 저장할 수 있습니다. 전용 로컬 영역에서는 S3 디렉터리 버킷을 생성하여 특정 데이터 경계 내에 데이터를 저장할 수 있으므로 데이터 레지던시 및 격리 사용 사례를 지원하는 데 도움이 됩니다. 로컬 영역의 디렉터리 버킷은 S3 One Zone-Infrequent Access(S3 One Zone-IA, Z-IA) 스토리지 클래스를 지원합니다. 로컬 영역에서 디렉터리 버킷을 생성하는 방법에 대한 자세한 내용은 데이터 레지던시 워크로드를 참조하세요.

> **Note**
>
> 디렉터리 버킷에는 기본적으로 모든 퍼블릭 액세스가 비활성화되어 있습니다. 이 동작은 변경할 수 없습니다. 디렉터리 버킷에 저장된 객체에는 액세스 권한을 부여할 수 없습니다. 디렉터리 버킷에만 액세스 권한을 부여할 수 있습니다. 자세한 내용은 요청 인증 및 권한 부여를 참조하세요.

테이블 버킷 - 일별 구매 트랜잭션, 스트리밍 센서 데이터 또는 광고 노출과 같은 테이블 형식 데이터를 저장하는 데 권장됩니다. 테이블 형식 데이터는 데이터베이스 테이블과 같이 열과 행의 데이터를 나타냅니다. 테이블 버킷은 쿼리 성능을 지속적으로 개선하고 테이블의 스토리지 비용을 절감하도록 설계된 기능이 내장되어 있으며 분석 및 기계 학습 워크로드에 최적화된 S3 스토리지를 제공합니다. S3 Tables는 Apache Iceberg 형식으로 테이블 형식 데이터를 저장하기 위해 특별히 설계되었습니다. Amazon Athena, Amazon Redshift 및 Apache Spark를 비롯한 인기 있는 쿼리 엔진을 사용하여 S3 Tables에서 테이블 형식 데이터를 쿼리할 수 있습니다. 기본적으로, AWS 리전별 AWS 계정당 최대 10개의 테이블 버킷과 테이블 버킷당 최대 10,000개의 테이블을 생성할 수 있습니다. 자세한 내용은 Amazon S3 Tables 및 테이블 버킷 작업을 참조하세요.

> **Note**
>
> 모든 테이블 버킷과 테이블은 비공개이며 공개될 수 없습니다. 이러한 리소스는 명시적으로 액세스 권한이 부여된 사용자만 액세스할 수 있습니다. 액세스 권한을 부여하려면 테이블 버킷 및 테이블에 대한 IAM 리소스 기반 정책과 사용자 및 역할에 대한 IAM ID 기반 정책을 사용할 수 있습니다. 자세한 내용은 S3 Tables의 보안을 참조하세요.

모든 버킷 유형에 대한 추가 정보

버킷을 생성할 때 버킷 이름을 입력하고 버킷이 속할 AWS 리전을 선택합니다. 버킷을 생성한 후에는 버킷 이름 또는 해당 리전을 변경할 수 없습니다. 버킷 이름은 다음 버킷 이름 지정 규칙을 따라야 합니다.

- 범용 버킷 이름 지정 규칙
- 디렉터리 버킷 이름 지정 규칙
- 테이블 버킷 이름 지정 규칙

또한 버킷은 다음과 같은 기능이 있습니다.

- Amazon S3 네임스페이스를 최상위 수준으로 구성합니다. 범용 버킷의 경우 이 네임스페이스는 `S3`입니다. 디렉터리 버킷의 경우 이 네임스페이스는 `s3express`입니다. 테이블 버킷의 경우 이 네임스페이스는 `s3tables`입니다.
- 스토리지 및 데이터 전송 요금을 담당하는 계정을 식별합니다.
- 사용량 보고를 위한 집계 단위로 사용됩니다.

Objects

객체는 Amazon S3에 저장되는 기본 개체입니다. 객체는 객체 데이터와 메타데이터로 구성됩니다. 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 지정 메타데이터를 지정할 수도 있습니다.

모든 객체는 어떤 버킷에 포함됩니다. 예를 들어 `photos/puppy.jpg`로 명명된 객체는 미국 서부(오리건) 리전의 `amzn-s3-demo-bucket` 범용 버킷에 저장되며 URL `https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/photos/puppy.jpg`를 사용하여 주소를 지정할 수 있습니다. 자세한 내용은 버킷 액세스를 참조하십시오.

객체는 키(이름) 및 버전 ID를 통해 버킷 내에서 고유하게 식별됩니다(버킷에서 S3 버전 관리가 사용 설정된 경우). 객체에 대한 자세한 내용은 Amazon S3 객체 개요 섹션을 참조하세요.

키

객체 키(또는 키 이름)는 버킷 내 객체에 대한 고유한 식별자입니다. 버킷 내 모든 객체는 정확히 하나의 키를 갖습니다. 버킷, 객체 키 및 선택적으로 버전 ID(버킷에 대해 S3 버전 관리가 사용 설정된 경우)

의 조합은 각 객체를 고유하게 식별합니다. 따라서 Amazon S3를 "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://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/photos/puppy.jpg` URL에서 *amzn-s3-demo-bucket*은 버킷의 이름이고 `photos/puppy.jpg`는 키입니다.

객체 키에 대한 자세한 내용은 Amazon S3 객체 이름 지정 섹션을 참조하십시오.

S3 버전 관리

S3 버전 관리를 사용하면 동일 버킷 내에 여러 개의 객체 변형을 보유할 수 있습니다. S3 버전 관리를 사용하여 버킷에 저장된 모든 버전의 모든 객체를 보존, 검색 및 복원할 수 있습니다. 또한 의도치 않은 사용자 작업 및 애플리케이션 장애로부터 쉽게 복구할 수 있습니다.

자세한 정보는 S3 버전 관리로 여러 버전의 객체 유지을 참조하세요.

버전 ID

버킷에 S3 버전 관리를 사용 설정하면 Amazon S3에서 버킷에 추가되는 각 객체에 고유한 버전 ID를 생성합니다. 버전 관리를 사용 설정할 때 버킷에 이미 존재하는 객체에는 `null`의 버전 ID가 있습니다. 이러한(또는 다른) 객체를 CopyObject 및 PutObject와 같은 기타 작업으로 수정하는 경우 새 객체가 고유한 버전 ID를 가집니다.

자세한 정보는 S3 버전 관리로 여러 버전의 객체 유지을 참조하세요.

버킷 정책

버킷 정책은 버킷과 해당 버킷의 객체에 액세스 권한을 부여할 수 있는 리소스 기반 AWS Identity and Access Management(IAM) 정책입니다. 버킷 소유자만 정책을 버킷에 연결할 수 있습니다. 버킷에 연결된 권한은 버킷 소유자가 소유한 모든 버킷의 객체에 적용됩니다. 버킷 정책은 크기가 20KB로 제한됩니다.

버킷 정책에는 AWS에서 표준인 JSON 기반 액세스 정책 언어가 사용됩니다. 버킷 정책을 사용하여 버킷의 객체에 대한 권한을 추가하거나 거부할 수 있습니다. 버킷 정책은 요청자, S3 작업, 리소스 및 요청의 측면 또는 조건(예: 요청을 수행하는 데 사용된 IP 주소)을 포함하여 정책의 요소를 기반으로 요청을 허용하거나 거부합니다. 예를 들어 버킷 소유자가 업로드된 객체를 완전히 제어할 수 있도록 하면서 S3 버킷에 객체를 업로드할 수 있는 교차 계정 권한을 부여하는 버킷 정책을 생성할 수 있습니다. 자세한 정보는 Amazon S3 버킷 정책 예시을 참조하세요.

버킷 정책에서는 Amazon 리소스 이름(ARN) 및 기타 값에 와일드카드 문자를 사용하여 객체의 하위 집합에 권한을 부여할 수 있습니다. 예를 들어, 공통 접두사로 시작하거나 `.html`과 같은 지정된 확장자로 끝나는 객체 그룹에 대한 액세스를 제어할 수 있습니다.

S3 액세스 포인트

Amazon S3 액세스 포인트는 해당 엔드포인트를 사용하여 데이터에 액세스하는 방법을 설명하는 전용 액세스 정책이 포함된 명명된 네트워크 엔드포인트입니다. 액세스 포인트는 `GetObject` 및 `PutObject`와 같은 S3 객체 작업을 수행하는 데 사용할 수 있는 범용 버킷, 디렉터리 버킷 또는 FSx for OpenZFS 볼륨과 같은 기본 데이터 소스에 연결됩니다. 액세스 포인트는 Amazon S3의 공유 데이터 세트에 대한 대규모 데이터 액세스 관리를 간소화합니다.

각 액세스 포인트에는 고유한 액세스 포인트 정책이 있습니다. 또한 버킷에 연결된 각 액세스 포인트에 대해 퍼블릭 액세스 차단 설정을 구성할 수 있습니다. 가상 프라이빗 클라우드(VPC)의 요청만 수락하도록 액세스 포인트를 구성하여 프라이빗 네트워크에 대한 Amazon S3 데이터 액세스를 제한할 수 있습니다.

범용 버킷의 액세스 포인트에 대한 자세한 내용은 액세스 포인트로 공유 데이터세트에 대한 액세스 관리 섹션을 참조하세요. 디렉터리 버킷의 액세스 포인트에 대한 자세한 내용은 액세스 포인트가 있는 디렉터리 버킷에서 공유 데이터세트 액세스 관리 섹션을 참조하세요.

액세스 제어 목록(ACL)

ACL을 사용하여 권한이 부여된 사용자에게 개별 범용 버킷 및 객체에 대한 읽기 및 쓰기 권한을 부여합니다. 각 범용 버킷과 객체마다 하위 리소스로서 연결되어 있는 ACL이 있습니다. ACL은 액세스를 허용할 AWS 계정 또는 그룹과 액세스 유형을 정의합니다. ACL은 IAM보다 먼저 적용되는 액세스 제어 메커니즘입니다. ACL에 대한 자세한 내용은 ACL(액세스 제어 목록) 개요 단원을 참조하십시오.

S3 객체 소유권은 버킷에 업로드되는 객체의 소유권을 제어하고 ACL을 비활성화 또는 활성화하는 데 사용할 수 있는 Amazon S3 버킷 수준 설정입니다. 기본적으로 객체 소유권은 버킷 소유자 적용 설정으로 설정되며 모든 ACL이 비활성화되어 있습니다. ACL이 비활성화되면 버킷 소유자는 버킷의 모든 객체를 소유하고 액세스 관리 정책을 사용하여 객체에 대한 액세스를 독점적으로 관리합니다.

Amazon S3의 최신 사용 사례 대부분은 더 이상 ACL을 사용할 필요가 없습니다. 각 객체에 대해 액세스를 개별적으로 제어할 필요가 있는 드문 상황을 제외하고는 ACL을 비활성화한 채로 두는 것이 좋습니다. ACL을 비활성화하면 누가 객체를 버킷에 업로드했는지에 관계없이 정책을 사용하여 버킷의 모든 객체에 대한 액세스를 제어할 수 있습니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

리전

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. AWS 리전에 저장된 객체는 사용자가 명시적으로 객체를 다른 리전으로 전송하거나 복제하지 않는 한 해당 리전을 벗어나지 않습니다. 예를 들어, 유럽(아일랜드) 리전에 저장된 객체는 유럽 밖으로 이동하지 않습니다.

> **Note**
>
> 계정에 대해 사용되는 AWS 리전에서만 Amazon S3에 액세스하고 해당 기능을 사용할 수 있습니다. 리전에서 AWS 리소스 생성 및 관리를 활성화하는 방법에 대한 자세한 내용은 AWS 일반 참조의 AWS 리전 관리를 참조하세요.

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

Amazon S3 데이터 일관성 모델

Amazon S3는 모든 AWS 리전의 Amazon S3 버킷에 있는 객체의 PUT 및 DELETE 요청에 대해 강력한 쓰기 후 읽기(read-after-write) 일관성을 제공합니다. 이는 새 객체에 대한 쓰기와, 기존 객체 및 DELETE 요청을 덮어쓰는 PUT 요청 모두에 적용됩니다. 또한 Amazon S3 Select, Amazon S3 액세스 제어 목록(ACL), Amazon S3 객체 태그, 객체 메타데이터(예: HEAD 객체)에 대한 읽기 작업은 매우 일관적입니다.

단일 키에 대한 업데이트는 원자성입니다. 예를 들어, 한 스레드에서 기존 키에 PUT 요청을 수행하고 두 번째 스레드에서 동일한 키에 GET 요청을 동시에 수행하면, 이전 데이터 또는 새 데이터는 얻지만 부분적 데이터나 손상된 데이터는 얻지 못합니다.

Amazon S3에서는 AWS 데이터 센터 내의 여러 서버로 데이터를 복제함으로써 고가용성을 구현합니다. PUT 요청이 성공하면 데이터가 안전하게 저장됩니다. 성공적인 PUT 응답을 받은 후 시작된 모든 읽기(GET 또는 LIST 요청)는 PUT 요청에 의해 쓰여진 데이터를 반환합니다. 다음은 이 동작의 예입니다.

- 프로세스가 Amazon S3로 새 객체를 쓰고 해당 버킷 내에 바로 키를 나열합니다. 새 객체가 목록에 나타납니다.
- 프로세스가 기존 객체를 대체하고 바로 읽기를 시도합니다. Amazon S3가 새 데이터를 반환합니다.

- 프로세스가 기존 객체를 삭제하고 바로 읽기를 시도합니다. 객체가 삭제되었으므로 Amazon S3는 데이터를 반환하지 않습니다.
- 프로세스가 기존 객체를 삭제하고 해당 버킷 내에 바로 키를 나열합니다. 객체가 목록에 나타나지 않습니다.

> **Note**
>
> - Amazon S3는 동시 작성자에 대한 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 구축해야 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.

버킷 구성에는 최종 일관성 모델이 있습니다. 특히 이는 다음을 의미합니다.

- 버킷을 삭제하고 즉시 모든 버킷을 나열하면 목록에 삭제된 버킷이 여전히 표시될 수 있습니다.
- 버킷에서 버전 관리를 처음으로 사용 설정하면 변경 사항이 완전히 전파되는 데 시간이 조금 걸릴 수 있습니다. 버전 관리를 사용 설정하고 나서 15분 정도 기다린 후, 버킷의 객체에 대해 쓰기 작업(PUT 또는 DELETE 요청)을 실행하는 것이 좋습니다.

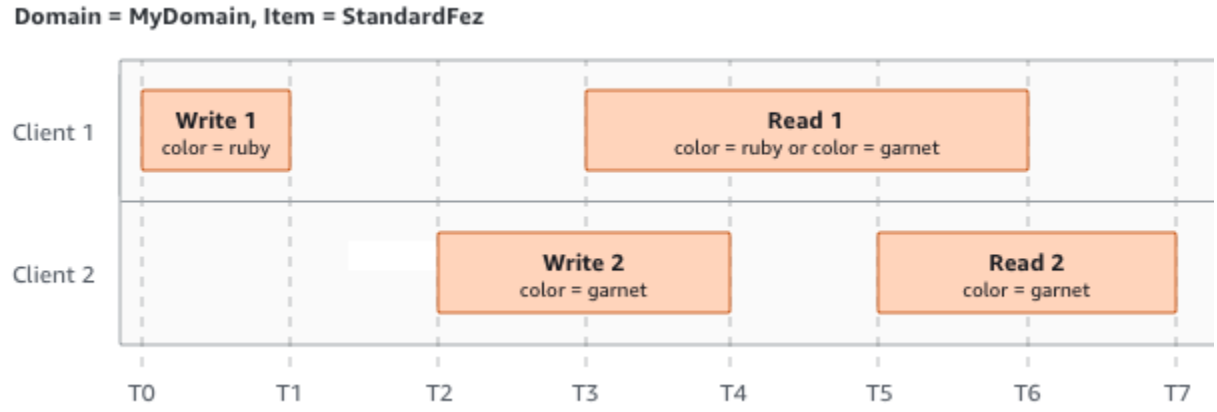
동시 애플리케이션

이 섹션에서는 여러 클라이언트가 동일한 항목에 쓸 때 Amazon S3에서 예상되는 동작의 예제를 제공합니다.

이 예제에서는 W1(쓰기 1)과 W2(쓰기 2) 모두 R1(읽기 1) 및 R2(읽기 2)가 시작되기 전에 완료됩니다. S3는 매우 일관적이기 때문에 R1과 R2는 모두 `color = ruby`을(를) 반환합니다.



다음 예제에서 W2는 R1이 시작되기 전에 완료되지 않습니다. 따라서 R1은 `color = ruby` 또는 `color = garnet`을(를) 반환할 수 있습니다. 그러나 R2가 시작되기 전에 W1과 W2가 완료되므로 R2는 `color = garnet`을(를) 반환합니다.



마지막 예시에서 W1이 승인을 받기 전에 W2가 시작되었습니다. 따라서 이러한 쓰기는 동시에 이루어진 것으로 간주됩니다. Amazon S3는 내부적으로 최종 작성자 인정(last-writer-wins) 의미를 사용하여 어떤 쓰기가 우선하는지 결정합니다. 그러나 Amazon S3가 요청을 수신하는 순서와 애플리케이션이 승인을 받는 순서는 네트워크 대기 시간 등의 다양한 요인으로 인해 예측할 수 없습니다. 예를 들어, W2는 동일한 리전의 Amazon EC2 인스턴스에 의해 시작되고 W1은 멀리 떨어진 호스트에 의해 시작될 수 있습니다. 최종 값을 결정하는 가장 좋은 방법은 두 쓰기가 모두 승인된 후 읽기를 수행하는 것입니다.



관련 서비스

데이터를 Amazon S3로 로드한 후에는 해당 데이터를 다른 AWS 서비스에 사용할 수 있습니다. 가장 자주 사용하게 될 서비스는 다음과 같습니다.

- Amazon Elastic Compute Cloud(Amazon EC2) – AWS 클라우드에서 안전하고 확장 가능한 컴퓨팅 용량을 제공합니다. Amazon EC2를 사용하면 하드웨어에 선투자할 필요가 없어 더 빠르게 애플리케이션을 개발하고 배포할 수 있습니다. Amazon EC2를 사용하여 원하는 수의 가상 서버를 빌드하고 보안 및 네트워킹을 구성하며 스토리지를 관리할 수 있습니다.
- Amazon EMR - 기업, 연구원, 데이터 분석가 및 개발자가 대량의 데이터를 쉽고 비용 효율적으로 처리할 수 있습니다. Amazon EMR은 호스팅된 Hadoop 프레임워크를 사용합니다. Hadoop 프레임워크는 Amazon EC2와 Amazon S3의 웹 규모 인프라에서 실행됩니다.
- AWS Snow 패밀리 - 데이터 센터가 아닌 까다로운 환경과, 일관된 네트워크 접속이 불가능한 장소에서 작업을 실행해야 하는 고객을 지원합니다. AWS Snow 패밀리 디바이스를 사용하면 인터넷에 연결할 수 없는 곳에서도 AWS 클라우드의 스토리지 및 컴퓨팅 파워를 로컬에서 비용 효율적으로 액세스할 수 있습니다.
- AWS Transfer Family - SSH(Secure Shell) 파일 전송 프로토콜(SFTP), SSL을 통한 파일 전송 프로토콜(FTPS) 및 FTP(파일 전송 프로토콜)를 사용하여 Amazon S3 또는 Amazon Elastic File System(Amazon EFS)에서 직접 송수신하는 파일 전송에 대한 완전관리형 지원을 제공합니다.

Amazon S3 액세스

다음 방법 중 하나를 사용하여 Amazon S3에서 작업할 수 있습니다.

AWS Management Console

이 콘솔은 Amazon S3 및 AWS 리소스를 관리하기 위한 웹 기반 사용자 인터페이스입니다. AWS 계정에 가입한 고객은 AWS Management Console에 로그인한 후 AWS Management Console 홈페이지에서 S3를 선택하여 Amazon S3 콘솔에 액세스할 수 있습니다.

AWS Command Line Interface

AWS 명령줄 도구를 통해 시스템 명령줄에서 명령을 실행하거나 스크립트를 빌드하여 AWS(S3 등) 작업을 수행할 수 있습니다.

AWS Command Line Interface(AWS CLI)은 광범위한 AWS 서비스의 명령을 제공합니다. AWS CLI는 Windows, macOS, Linux에서 지원됩니다. 시작하려면 AWS Command Line Interface 사용 설명서를 참조하세요. Amazon S3용 명령에 대한 자세한 내용은 AWS CLI 명령 참조의 s3api 및 s3control을 참조하세요.

AWS SDK

AWS에서는 다양한 프로그래밍 언어 및 플랫폼(Java, Python, Ruby, .NET, iOS, Android 등)을 위한 라이브러리와 샘플 코드로 구성된 소프트웨어 개발 키트(SDK)를 제공합니다. AWS SDK를 사용하면 편리하게 S3 및 AWS에 프로그래밍 방식으로 액세스할 수 있습니다. Amazon S3은 REST 서비스입니다. 프로그래밍 태스크를 간소화하기 위해 기본 Amazon S3 REST API를 래핑하는 AWS SDK 라이브러리를 사용하여 Amazon S3에 요청을 전송할 수 있습니다. 예를 들어 SDK는 서명 계산, 암호화 방식으로 요청 서명, 오류 관리 및 자동으로 요청 재시도와 같은 작업을 처리합니다. 다운로드 및 설치 방법을 비롯하여 AWS SDK에 대한 자세한 내용은 AWS용 도구를 참조하세요.

Amazon S3와의 모든 상호 작용은 인증을 거치거나 익명으로 할 수 있습니다. AWS SDK를 사용할 경우 라이브러리는 사용자가 제공하는 키로부터 인증을 위한 서명을 컴퓨팅합니다. Amazon S3에 요청하는 방법에 대한 자세한 내용은 요청을 참조하세요.

Amazon S3 REST API

Amazon S3 아키텍처는 AWS에서 지원하는 인터페이스를 사용하여 객체를 저장 및 검색하는 프로그래밍 언어 중립적 아키텍처로 설계되었습니다. Amazon S3 REST API를 사용하여 프로그래밍 방식으로 S3 및 AWS에 액세스할 수 있습니다. REST API는 Amazon S3에 대한 HTTP 인터페이스입니다. REST API를 통해 표준 HTTP 요청을 사용하여 버킷과 객체를 생성하고, 가져오며, 삭제할 수 있습니다.

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.

애플리케이션에서 직접 REST API를 호출할 경우 서명을 계산하는 코드를 작성하여 요청에 추가해야 합니다. Amazon S3에 요청하는 방법에 대한 자세한 내용은 Amazon S3 API 참조에서 요청을 참조하세요.

> **Note**
>
> HTTP를 통한 SOAP API 지원은 중단되었지만 HTTPS를 통해 계속해서 사용할 수 있습니다. 최신 Amazon S3 기능은 SOAP를 지원하지 않습니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

Amazon S3 요금 지불

Amazon S3의 요금은 애플리케이션의 스토리지 요구 사항에 맞춰 계획을 수립할 필요가 없도록 설계되었습니다. 대부분의 스토리지 공급자는 미리 정해진 양의 스토리지 및 네트워크 전송 용량을 구입하도록 요구합니다. 이 시나리오에서는 용량을 초과할 경우 서비스가 중단되거나 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. 이 모델은 AWS 인프라의 비용 이점을 활용하면서 사업과 함께 성장시킬 수 있는 가변 비용 서비스를 제공합니다. 자세한 내용은 Amazon S3 요금을 참조하세요.

AWS에 가입하면 Amazon S3를 포함하여 AWS의 모든 서비스에 AWS 계정이 자동으로 등록됩니다. 하지만 사용한 서비스에 대해서만 청구됩니다. 신규 Amazon S3 고객인 경우 Amazon S3를 무료로 시작할 수 있습니다. 자세한 내용은 AWS 프리 티어를 참조하세요.

청구 요금은 AWS 결제 및 비용 관리 콘솔의 결제 및 비용 관리(Billing and Cost Management) 대시보드에서 확인할 수 있습니다. AWS 계정 결제에 대한 자세한 내용은 AWS Billing 사용 설명서를 참조하세요. AWS 결제 및 AWS 계정에 관련된 질문은 AWS Support에 문의하세요.

PCI DSS 준수

Amazon S3에서는 전자 상거래 웹사이트 운영자 또는 서비스 공급자에 의한 신용카드 데이터의 처리, 저장 및 전송을 지원하며, Payment Card Industry(PCI) Data Security Standard(DSS) 준수를 검증받았습니다. AWS PCI 규정 준수 패키지의 사본을 요청하는 방법 등 PCI DSS에 대해 자세히 알아보려면 PCI DSS 레벨 1을 참조하십시오.

Amazon S3 시작하기

버킷과 객체를 사용하여 Amazon S3를 시작할 수 있습니다. 버킷은 객체에 대한 컨테이너입니다. 객체는 파일과 해당 파일을 설명하는 모든 메타데이터입니다.

Amazon S3에 객체를 저장하려면 버킷을 생성한 다음 버킷에 객체를 업로드합니다. 객체가 버킷에 있으면 객체를 열고 다운로드하고 이동할 수 있습니다. 객체나 버킷이 더 이상 필요하지 않은 경우 리소스를 정리할 수 있습니다.

Amazon S3에서는 사용한 만큼만 비용을 청구하며, Amazon S3 기능 및 요금에 대한 자세한 내용은 Amazon S3를 참조하십시오. 신규 Amazon S3 고객인 경우 Amazon S3를 무료로 시작할 수 있습니다. 자세한 내용은 AWS 프리 티어를 참조하십시오.

> **Note**
>
> Amazon S3 Express One Zone 스토리지 클래스를 디렉터리 버킷과 함께 사용하는 방법에 대한 자세한 내용은 S3 Express One Zone 및 디렉터리 버킷 작업 섹션을 참조하세요.

비디오: Amazon S3 시작하기

사전 조건

시작하기 전에 Amazon S3 설정의 단계를 완료했는지 확인합니다.

Amazon S3 설정

AWS에 가입하면 Amazon S3를 포함하여 AWS의 모든 서비스에 AWS 계정이 자동으로 등록됩니다. 사용한 서비스에 대해서만 청구됩니다.

Amazon S3에서는 사용한 만큼만 비용을 청구하며, Amazon S3 기능 및 요금에 대한 자세한 내용은 Amazon S3를 참조하십시오. 신규 Amazon S3 고객인 경우 Amazon S3를 무료로 시작할 수 있습니다. 자세한 내용은 AWS 프리 티어를 참조하십시오.

Amazon S3를 설정하려면 다음 섹션의 단계를 사용합니다.

AWS에 가입하고 Amazon S3를 설정할 때, 원하면 AWS Management Console에서 표시 언어를 변경할 수 있습니다. 자세한 내용은 AWS Management Console 시작하기 안내서의 AWS Management Console 언어 변경을 참조하세요.

주제

- AWS 계정에 등록
- 관리자 액세스 권한이 있는 사용자 생성

AWS 계정에 등록

AWS 계정이 없는 경우 다음 절차에 따라 계정을 생성합니다.

AWS 계정에 가입

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

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

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

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

관리자 액세스 권한이 있는 사용자 생성

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

귀하의 AWS 계정 루트 사용자 보호

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

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

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

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

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

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

관리 액세스 권한이 있는 사용자로 로그인

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

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

추가 사용자에게 액세스 권한 할당

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

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

   지침은 AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

1단계: 첫 번째 S3 버킷 생성

AWS에 가입한 후에는 AWS Management Console을 사용하여 Amazon S3에 버킷을 만들 수 있습니다. Amazon S3의 모든 객체는 버킷에 저장됩니다. 데이터를 Amazon S3에 저장하기 전에 버킷을 만들어야 합니다.

> **Note**
>
> Amazon S3 Express One Zone 스토리지 클래스를 디렉터리 버킷과 함께 사용하는 방법에 대한 자세한 내용은 S3 Express One Zone 및 디렉터리 버킷 작업 섹션을 참조하세요.

> **Note**
>
> 버킷을 만드는 데는 요금이 청구되지 않습니다. 객체를 버킷에 저장하거나 객체를 버킷과 주고받은 경우에만 요금이 청구됩니다. 이 설명서의 예제를 수행하는 동안 1 USD 미만의 최소 요금이 발생합니다. 스토리지 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 버킷을 생성하려는 리전을 선택합니다.

   > **Note**
   >
   > - 버킷을 만든 후에는 리전을 변경할 수 없습니다.
   > - 지연 시간과 요금을 최소화하고 규제 요건을 충족하려면 가장 가까운 리전을 선택하십시오. 특정 리전에 저장된 객체는 사용자가 명시적으로 객체를 다른 리전으로 전송하지 않는 한 해당 리전을 벗어나지 않습니다. Amazon S3 AWS 리전 목록은 Amazon Web Services 일반 참조의 AWS 서비스 엔드포인트를 참조하십시오.

3. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
4. 버킷 만들기를 선택합니다. 버킷 만들기 페이지가 열립니다.
5. [Bucket Name]에서 버킷 이름을 입력합니다.

   버킷 이름은 다음과 같아야 합니다.

   - 파티션 내에서 고유해야 합니다. 파티션은 리전의 그룹입니다. AWS에는 현재 `aws`(상용 리전), `aws-cn`(중국 리전) 및 `aws-us-gov`(AWS GovCloud (US) Regions), 이렇게 세 가지 파티션이 있습니다.
   - 3~63자 이내여야 합니다.

- 소문자, 숫자, 마침표(.) 및 하이픈(-)만 포함해야 합니다. 최상의 호환성을 위해 정적 웹 사이트 호스팅에만 사용되는 버킷을 제외하고 버킷 이름에 마침표(.)를 사용하지 않는 것이 좋습니다.
- 문자나 숫자로 시작하고 끝나야 합니다.
- 버킷 이름 지정 규칙의 전체 목록은 범용 버킷 이름 지정 규칙 섹션을 참조하세요.

> **Important**
>
> - 버킷을 생성한 후에는 해당 이름을 변경할 수 없습니다.
> - 버킷 이름에 민감한 정보가 포함되어서는 안 됩니다. 버킷 이름은 버킷의 객체를 가리키는 URL에 표시됩니다.

6. (선택 사항) 일반 구성 아래에서 기존 버킷의 설정을 새 버킷에 복사하도록 선택할 수 있습니다. 기존 버킷의 설정을 복사하지 않으려면 다음 단계로 건너뛰세요.

> **Note**
>
> 이 옵션:
>
> - AWS CLI에서는 사용할 수 없으며 Amazon S3 콘솔에서만 사용할 수 있습니다.
> - 기존 버킷의 버킷 정책을 새 버킷으로 복사하지 않습니다.

기존 버킷의 설정을 복사하려면 기존 버킷에서 설정 복사 아래에서 버킷 선택을 선택합니다. 버킷 선택 창이 열립니다. 복사하려는 설정이 있는 버킷을 찾아 버킷 선택을 선택합니다. 버킷 선택 창이 닫히고 버킷 생성 창이 다시 열립니다.

이제 기존 버킷에서 설정 복사 아래에 선택한 버킷의 이름이 표시됩니다. 이제 새 버킷의 설정이 선택한 버킷의 설정과 일치합니다. 복사된 설정을 제거하려면 기본값 복원을 선택합니다. 버킷 생성 페이지에서 나머지 버킷 설정을 검토하세요. 변경하지 않으려면 마지막 단계로 건너뛸 수 있습니다.

7. 객체 소유권(Object Ownership)에서 ACL을 사용 중지 또는 사용 설정하고 버킷에 업로드된 객체의 소유권을 제어하려면 다음 설정 중 하나를 선택합니다.

ACL 사용 중지됨

- 버킷 소유자 적용(기본값) – ACL이 비활성화되고 버킷 소유자는 범용 버킷의 모든 객체를 자동으로 소유하고 완전히 제어합니다. ACL은 더 이상 S3 범용 버킷의 데이터에 대한 권한에 영향을 주지 않습니다. 버킷은 정책을 독점적으로 사용하여 액세스 제어를 정의합니다.

  기본적으로 ACL은 비활성화되어 있습니다. Amazon S3의 최신 사용 사례 대부분은 더 이상 ACL을 사용할 필요가 없습니다. 각 객체에 대해 액세스를 개별적으로 제어해야 하는 드문 상황을 제외하고는 ACL을 비활성화한 채로 두는 것이 좋습니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

ACL 사용됨

- 버킷 소유자 기본(Bucket owner preferred) – 버킷 소유자가 `bucket-owner-full-control` 미리 제공 ACL을 사용하여 다른 계정이 버킷에 작성하는 새 객체를 소유하고 완전히 제어합니다.

  버킷 소유자 선호 설정을 적용하는 경우 모든 Amazon S3 업로드에 `bucket-owner-full-control` 미리 제공된 ACL을 포함하도록 요구하려면 이 ACL을 사용하는 객체 업로드만 허용하는 버킷 정책을 추가할 수 있습니다.
- 객체 작성자(Object writer) – 객체를 업로드하는 AWS 계정은 객체를 소유하고 완전히 제어하며 ACL을 통해 다른 사용자에게 이에 대한 액세스 권한을 부여할 수 있습니다.

> **Note**
>
> 기본 설정은 버킷 소유자 적용입니다. 기본 설정을 적용하고 ACL을 비활성화된 상태로 유지하려면 `s3:CreateBucket` 권한만 있으면 됩니다. ACL을 활성화하려면 `s3:PutBucketOwnershipControls` 권한이 있어야 합니다.

8. 퍼블릭 액세스 차단을 위한 버킷 설정에서 버킷에 적용할 퍼블릭 액세스 차단 설정을 선택합니다.

   기본적으로, 네 개의 퍼블릭 액세스 차단 설정이 모두 활성화되어 있습니다. 특정 사용 사례에 대해 하나 이상의 설정을 해제해야 하는 경우가 아니라면 모든 설정을 활성화된 상태로 유지하는 것이 좋습니다. 퍼블릭 액세스 차단에 대한 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하십시오.

> **Note**
>
> 모든 퍼블릭 액세스 차단 설정을 활성화하려면 `s3:CreateBucket` 권한만 있으면 됩니다. 퍼블릭 액세스 차단 설정을 해제하려면 `s3:PutBucketPublicAccessBlock` 권한이 있어야 합니다.

9. (선택 사항) 버킷 버전 관리는 기본으로 비활성화됩니다. 버전 관리는 동일 버킷 내에 여러 개의 객체 변형을 보유하는 것을 의미합니다. 버전 관리를 사용하여 버킷에 저장된 모든 버전의 객체를 모두 보존, 검색 및 복원할 수 있습니다. 또한, 의도치 않은 사용자 작업 및 애플리케이션 장애가 발생해도 쉽게 복구할 수 있습니다. 버전 관리에 대한 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하십시오.

   버킷에서 버전 관리를 활성화하려면 활성화를 선택합니다.

10. (선택 사항) Tags(태그)에서 버킷에 태그를 추가할 수 있습니다. AWS 비용 할당을 하면서 버킷 태그를 지정해 버킷 사용에 대한 요금 청구 주석을 달 수 있습니다. 태그는 버킷에 할당된 라벨을 나타내는 한 쌍의 키-값입니다. 자세한 내용은 the section called “비용 할당 태그 사용” 섹션을 참조하세요.

    버킷 태그를 추가하려면 Key(키) 및 원하는 경우 Value(값)를 입력하고 Add Tag(태그 추가)를 선택합니다.

11. 기본 암호화를 구성하려면 암호화 유형 아래에서 다음 중 하나를 선택합니다.

    - Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)
    - AWS Key Management Service 키를 사용한 서버 측 암호화(SSE-KMS)
    - AWS Key Management Service(AWS KMS) 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)

    > **Important**
    >
    > 기본 암호화 구성으로 SSE-KMS 또는 DSSE-KMS 옵션을 사용할 경우 AWS KMS의 초당 요청 수(RPS) 할당량이 적용됩니다. AWS KMS 할당량과 할당량 증대를 요청하는 방법에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 할당량을 참조하십시오.

버킷과 새 객체는 Amazon S3 관리형 키(SSE-S3)를 암호화 구성의 기본 수준으로 사용하는 서버 측 암호화로 암호화됩니다. 기본 암호화에 대한 자세한 내용은 Amazon S3 버킷에 대한 기본 서버 측 암호화 동작 설정 섹션을 참조하십시오. SSE-S3에 대한 자세한 내용은 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) 사용 섹션을 참조하세요.

서버 측 암호화를 사용하는 데이터 암호화에 대한 자세한 내용은 the section called “데이터 암호화” 섹션을 참조하세요.

12. Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) 또는 AWS Key Management Service(AWS KMS) 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)를 선택한 경우 다음을 수행하세요.

    a. AWS KMS 키에서 다음 방법 중 하나로 KMS 키를 지정합니다.

    - 사용 가능한 KMS 키 목록에서 AWS KMS keys 중에서 선택을 선택하고 사용 가능한 키 목록에서 KMS 키를 선택합니다.

      AWS 관리형 키(aws/s3)와 고객 관리형 키가 모두 목록에 표시됩니다. 고객 관리형 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 키 및 AWS 키를 참조하십시오.

    - KMS 키 ARN을 입력하려면 AWS KMS key ARN 입력을 선택하고 나타나는 필드에 KMS 키 ARN을 입력합니다.

    - AWS KMS 콘솔에서 고객 관리형 키를 생성하려면 KMS 키 생성을 선택합니다.

      AWS KMS key 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 키 생성을 참조하십시오.

> **Important**
>
> 버킷과 동일한 AWS 리전에서 사용할 수 있는 KMS 키만 사용 가능합니다. Amazon S3 콘솔은 버킷과 동일한 리전에 있는 처음 100개의 KMS 키만 나열합니다. 목록에 없는 KMS 키를 사용하려면 KMS 키 ARN을 입력해야 합니다. 다른 계정에서 소유한 KMS 키를 사용하려면 먼저 해당 키에 대한 사용 권한이 있어야 하고, 다음 단계로 KMS 키 ARN을 입력해야 합니다. KMS 키의 크로스 계정 권한에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 Creating KMS keys that other accounts can use를 참조하세요. SSE-KMS에 대한 자세한 내용은 AWS KMS(SSE-KMS)를 사용한 서버 측 암호화 지정 단원을 참조하십시오. DSSE-KMS에 대한 자세

한 내용은 the section called “이중 계층 서버 측 암호화(DSSE-KMS)” 섹션을 참조하세요.

Amazon S3에서 서버 측 암호화에 AWS KMS key을 사용하는 경우 대칭 암호화 KMS 키를 선택해야 합니다. Amazon S3는 대칭 암호화 KMS 키만 지원하며 비대칭 KMS 키는 지원하지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 대칭 및 비대칭 KMS 키 식별을 참조하십시오.

b. SSE-KMS와 함께 기본 암호화를 사용하도록 버킷을 구성할 때 S3 버킷 키를 사용할 수도 있습니다. S3 버킷 키를 사용하면 Amazon S3에서 AWS KMS로의 요청 트래픽이 줄어 암호화 비용이 절감됩니다. 자세한 내용은 Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감 섹션을 참조하세요. S3 버킷 키는 DSSE-KMS에서 지원되지 않습니다.

   기본적으로 S3 버킷 키는 Amazon S3 콘솔에서 활성화됩니다. 비용을 낮추려면 S3 버킷 키를 활성화된 상태로 두는 것이 좋습니다. 버킷에 대해 S3 버킷 키를 비활성화하려면 버킷 키 아래에서 비활성화를 선택합니다.

13. (선택 사항) S3 Object Lock을 사용하면 새 객체가 삭제되거나 덮어쓰이지 않도록 보호할 수 있습니다. 자세한 내용은 Object Lock으로 객체 잠금 섹션을 참조하세요. S3 Object Lock을 활성화하려면 다음을 수행합니다.

    a. 고급 설정(Advanced Settings)을 선택합니다.

    > **Important**
    >
    > Object Lock을 활성화하면 버킷의 버전 관리도 활성화됩니다. 버킷을 활성화하고 성공적으로 만든 후에는 버킷의 속성 탭에서 Object Lock 기본 보존 및 법적 보존 설정도 구성해야 합니다.

    b. 객체 잠금을 활성화하려면 Enable(활성화)을 선택하고 표시되는 경고를 읽고 확인합니다.

> **Note**
>
> Object Lock을 사용하는 버킷을 만들려면 `s3:CreateBucket`, `s3:PutBucketVersioning` 및 `s3:PutBucketObjectLockConfiguration` 권한이 있어야 합니다.

14. 버킷 만들기를 선택합니다.

Amazon S3에 버킷을 만들었습니다.

다음 단계

버킷에 객체를 추가하려면 2단계: 버킷에 객체 업로드 섹션을 참조하세요.

2단계: 버킷에 객체 업로드

Amazon S3에 버킷을 생성했으면 버킷에 객체를 업로드할 준비가 된 것입니다. 텍스트 파일, 사진, 동영상 및 기타 모든 종류의 파일이 객체가 될 수 있습니다.

> **Note**
>
> Amazon S3 Express One Zone 스토리지 클래스를 디렉터리 버킷과 함께 사용하는 방법에 대한 자세한 내용은 S3 Express One Zone 및 디렉터리 버킷 작업 섹션을 참조하세요.

버킷에 객체를 업로드하려면

1. https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 버킷(Buckets) 목록에서 객체를 업로드하려는 버킷 이름을 선택합니다.
3. 버킷의 객체(Objects) 탭에서 업로드(Upload)를 선택합니다.
4. 파일 및 폴더(Files and folders)에서 파일 추가(Add files)를 선택합니다.
5. 업로드할 파일을 선택한 후 열기를 선택합니다.
6. 업로드를 선택합니다.

객체를 버킷에 성공적으로 업로드했습니다.

다음 단계

객체를 확인하려면 3단계: 객체 다운로드 단원을 참조하세요.

3단계: 객체 다운로드

버킷에 객체를 업로드한 후에는 객체에 대한 정보를 보고, 로컬 컴퓨터로 객체를 다운로드할 수 있습니다.

> **Note**
>
> Amazon S3 Express One Zone 스토리지 클래스를 디렉터리 버킷과 함께 사용하는 방법에 대한 자세한 내용은 S3 Express One Zone 및 디렉터리 버킷 작업 섹션을 참조하세요.

S3 콘솔 사용

이 단원에서는 Amazon S3 콘솔을 사용하여 S3 버킷에서 단일 객체를 다운로드하는 방법을 설명합니다.

> **Note**
>
> - 한 번에 하나의 객체만 다운로드할 수 있습니다.
> - Amazon S3 콘솔을 사용하여 키 이름이 마침표(.)로 끝나는 객체를 다운로드하는 경우, 다운로드한 객체의 키 이름에서 마침표가 제거됩니다. 다운로드한 객체 이름 끝에 마침표를 유지하려면 AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용해야 합니다.

S3 버킷에서 객체 다운로드

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷 또는 디렉터리 버킷을 선택합니다.
3. 버킷 목록에서 객체를 다운로드할 버킷의 이름을 선택합니다.
4. 다음 방법 중 하나를 사용하여 S3 버킷에서 객체를 다운로드할 수 있습니다.
   - 객체 옆의 확인란을 선택하고 다운로드를 선택합니다. 특정 폴더에 객체를 다운로드하려면 작업 메뉴에서 다른 이름으로 다운로드를 선택합니다.
   - 특정 버전의 객체를 다운로드하려면 버전 표시(검색 상자 옆에 있음)를 켭니다. 원하는 객체 버전 옆의 확인란을 선택하고 다운로드를 선택합니다. 특정 폴더에 객체를 다운로드하려면 작업 메뉴에서 다른 이름으로 다운로드를 선택합니다.

객체를 성공적으로 다운로드했습니다.

다음 단계

Amazon S3 내에서 객체를 복사하여 붙여넣으려면 4단계: 폴더에 객체 복사 섹션을 참조하세요.

4단계: 폴더에 객체 복사

이미 버킷에 객체를 추가한 후 객체를 다운로드했습니다. 이제 폴더를 만들고 개체를 복사하여 폴더에 붙여 넣습니다.

> **Note**
>
> Amazon S3 Express One Zone 스토리지 클래스를 디렉터리 버킷과 함께 사용하는 방법에 대한 자세한 내용은 S3 Express One Zone 및 디렉터리 버킷 작업 섹션을 참조하세요.

객체를 폴더에 복사하려면

1. 버킷 목록에서 버킷 이름을 선택합니다.
2. 폴더 만들기(Create folder)를 선택하고 다음과 같이 폴더를 구성합니다.
   a. 폴더 이름(예: `favorite-pics`)을 입력합니다.
   b. 폴더 암호화 설정에서 비활성화(Disable)를 선택합니다.
   c. 저장을 선택합니다.
3. 복사할 객체가 포함된 Amazon S3 버킷 또는 폴더로 이동합니다.
4. 복사할 객체 이름 왼쪽에 있는 확인란을 선택합니다.
5. 작업(Actions)을 선택한 후 표시되는 옵션 목록에서 복사(Copy)를 선택합니다.

   또는 오른쪽 위에 있는 옵션에서 복사를 선택합니다.
6. 다음과 같이 대상 폴더를 선택합니다.
   a. S3 찾아보기(Browse S3)를 선택합니다.
   b. 폴더 이름 왼쪽에 있는 옵션 버튼을 선택합니다.

      폴더로 이동하여 하위 폴더를 대상으로 선택하려면 폴더 이름을 선택합니다.
   c. 목적지 선택(Choose destination)을 선택합니다.

대상 폴더의 경로가 대상(Destination) 상자에 표시됩니다. 대상(Destination)에서 대상 경로를 변경하여 입력할 수 있습니다(예: s3://*bucket-name*/*folder-name*/).

7. 오른쪽 하단에서 복사(Copy)를 선택합니다.

   Amazon S3가 객체를 대상 폴더에 복사합니다.

다음 단계

Amazon S3에서 객체 및 버킷을 삭제하려면 5단계: 객체 및 버킷 삭제 섹션을 참조하십시오.

5단계: 객체 및 버킷 삭제

객체 또는 버킷이 더 이상 필요하지 않은 경우 추가 요금이 부과되지 않도록 객체 또는 버킷을 삭제하는 것이 좋습니다. 실습용으로 이 시작 연습을 완료한 후 버킷 또는 객체를 사용할 계획이 없는 경우 요금이 더 이상 발생하지 않도록 버킷 및 객체를 삭제하는 것이 좋습니다.

버킷을 삭제하기 전에 버킷을 비우거나 버킷의 객체를 삭제합니다. 객체 및 버킷을 삭제한 후에는 더 이상 사용할 수 없습니다.

동일한 버킷 이름을 계속 사용하려면 객체를 삭제하거나 버킷을 비우고 버킷을 삭제하지 않는 것이 좋습니다. 버킷을 삭제하면 이름을 다시 사용할 수 있게 됩니다. 그러나 다른 AWS 계정에서 동일한 이름의 버킷을 생성한 후 다시 사용할 수 있습니다.

> **Note**
>
> Amazon S3 Express One Zone 스토리지 클래스를 디렉터리 버킷과 함께 사용하는 방법에 대한 자세한 내용은 S3 Express One Zone 및 디렉터리 버킷 작업 섹션을 참조하세요.

주제

- 객체 삭제
- 버킷 비우기
- 버킷 삭제

객체 삭제

버킷에서 모든 객체를 비우지 않고 삭제할 객체를 선택하려는 경우 객체를 삭제할 수 있습니다.

1. 버킷 목록에서 객체를 삭제하려는 버킷 이름을 선택합니다.
2. 삭제할 객체를 선택합니다.
3. 오른쪽 상단의 옵션에서 삭제를 선택합니다.
4. 객체 삭제 페이지에서 **delete**를 입력하여 객체 삭제를 확인합니다.
5. 객체 삭제를 선택합니다.

버킷 비우기

버킷을 삭제하려면 먼저 버킷을 비워야 합니다. 그러면 버킷의 모든 객체가 삭제됩니다.

버킷을 비우려면

1. 버킷 목록에서 비우려는 버킷을 선택한 다음 Empty(비우기)를 선택합니다.
2. 버킷을 비우고 버킷에 있는 모든 객체를 삭제하려면 버킷 비우기에 **permanently delete**를 입력합니다.

   > **Important**
   >
   > 버킷을 비우는 작업은 실행 취소할 수 없습니다. 버킷 비우기 작업이 진행되는 동안 버킷에 추가된 객체는 삭제됩니다.

3. 버킷을 비우고 버킷에 있는 모든 객체를 삭제하려면 Empty(비우기)를 선택합니다.

   객체 삭제 실패 및 성공에 대한 요약을 검토하는 데 사용할 수 있는 버킷 비우기: 상태 페이지가 열립니다.
4. 버킷 목록으로 돌아가려면 종료를 선택합니다.

버킷 삭제

버킷을 비우거나 버킷에서 모든 객체를 삭제한 후 버킷을 삭제할 수 있습니다.

1. 버킷을 삭제하려면 버킷 목록에서 버킷을 선택합니다.

2. 삭제를 선택합니다.
3. 삭제를 확인하려면 버킷 삭제에 버킷 이름을 입력합니다.

> **Important**
>
> 버킷 삭제는 실행 취소할 수 없습니다. 버킷 이름은 고유합니다. 버킷을 삭제하면 다른 AWS 사용자가 해당 이름을 사용할 수 있습니다. 같은 버킷 이름을 계속 사용하려면 버킷을 삭제하지 마십시오. 대신 버킷을 비우고 보관하십시오.

4. 버킷을 삭제하려면 버킷 삭제를 선택합니다.

다음 단계

이전 예제에서는 몇 가지 기본 Amazon S3 작업을 수행하는 방법을 알아보았습니다.

다음 주제에서는 사용자가 직접 애플리케이션에서 구현할 수 있도록 Amazon S3을 보다 심도 있게 이해할 수 있는 학습 경로를 설명합니다.

> **Note**
>
> Amazon S3 Express One Zone 스토리지 클래스를 디렉터리 버킷과 함께 사용하는 방법에 대한 자세한 내용은 S3 Express One Zone 및 디렉터리 버킷 작업 섹션을 참조하세요.

주제

- 일반적인 사용 사례 이해
- 버킷 및 객체에 대한 액세스 제어
- 스토리지 보호 및 모니터링
- Amazon S3를 사용한 개발
- 자습서로 배우기
- 교육 및 지원 살펴보기

일반적인 사용 사례 이해

Amazon S3을 사용하여 특정 사용 사례를 지원할 수 있습니다. 이 AWS 솔루션 라이브러리 및 AWS 블로그 에서는 정보 및 자습서별 사용 사례를 제공합니다. 다음은 Amazon S3의 몇 가지 일반적인 사용 사례입니다.

- 백업 및 스토리지 – Amazon S3 스토리지 관리 기능을 사용하여 비용 관리, 규정 요구 사항 충족, 지연 시간 단축, 규정 준수 요구 사항에 맞게 여러 개의 개별 데이터 복제본 저장을 수행합니다.
- 애플리케이션 호스팅 - 안정성과 확장성이 뛰어나며 저렴한 비용으로 웹 애플리케이션을 배포, 설치, 관리할 수 있습니다. 예를 들어 정적 웹 사이트를 호스팅하도록 Amazon S3 버킷을 구성할 수 있습니다. 자세한 내용은 Amazon S3를 사용하여 정적 웹 사이트 호스팅 섹션을 참조하세요.
- 미디어 호스팅 - 동영상, 사진 또는 음악 업로드 및 다운로드를 호스팅하는 가용성이 높은 인프라를 구축합니다.
- 소프트웨어 제공 - 고객이 다운로드할 수 있는 소프트웨어 애플리케이션을 호스팅합니다.

버킷 및 객체에 대한 액세스 제어

Amazon S3은 다양한 보안 기능 및 도구를 제공합니다. 개요는 Amazon S3의 액세스 제어 섹션을 참조하세요.

기본적으로 S3 버킷 및 객체는 프라이빗입니다. 생성한 S3 리소스에만 액세스할 수 있습니다. 다음 기능을 사용하여 특정 사용 사례를 지원하는 세분화된 리소스 권한을 부여하거나 Amazon S3 리소스의 권한을 감사할 수 있습니다.

- S3 퍼블릭 액세스 차단 - S3 버킷과 객체에 대한 퍼블릭 액세스를 차단합니다. 기본적으로 퍼블릭 액세스 차단 설정은 버킷 수준에서 켜져 있습니다.
- AWS Identity and Access Management(IAM) 자격 증명 – IAM 또는 AWS IAM Identity Center를 사용하여 AWS 계정에서 IAM 자격 증명을 생성하여 Amazon S3 리소스에 대한 액세스를 관리합니다. 예를 들어 IAM을 Amazon S3와 함께 사용하여 AWS 계정이 소유한 Amazon S3 버킷에 대해 사용자 또는 사용자 그룹이 보유한 액세스 유형을 제어할 수 있습니다. IAM 자격 증명 및 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM 자격 증명(사용자, 사용자 그룹 및 역할)을 참조하세요.
- 버킷 정책 - IAM 기반 정책 언어를 사용하여 S3 버킷과 그 안에 있는 객체에 대한 리소스 기반 권한을 구성합니다.
- 액세스 제어 목록(ACL) - 인증된 사용자에게 개별 버킷 및 객체에 대한 읽기 및 쓰기 권한을 부여합니다. 일반적으로 ACL 대신 액세스 제어를 위해 S3 리소스 기반 정책(버킷 정책 및 액세스 포인트 정책) 또는 IAM 사용자 정책을 사용하는 것이 좋습니다. 정책은 단순하고 더 유연한 액세스 제어 옵션

입니다. 버킷 정책과 액세스 포인트 정책을 사용하면 Amazon S3 리소스에 대한 모든 요청에 광범위하게 적용되는 규칙을 정의할 수 있습니다. 리소스 기반 정책 또는 IAM 사용자 정책 대신 ACL을 사용하는 특정 사례에 대한 자세한 내용은 Amazon S3의 ID 및 액세스 관리 섹션을 참조하세요.

- S3 객체 소유권 - 버킷의 모든 객체에 대한 소유권을 가져와서 Amazon S3에 저장된 데이터에 대한 액세스 관리를 간소화합니다. S3 객체 소유권은 ACL을 비활성화 또는 활성화하는 데 사용할 수 있는 Amazon S3 버킷 수준 설정입니다. 기본적으로 ACL은 비활성화되어 있습니다. ACL이 비활성화되면 버킷 소유자는 버킷의 모든 객체를 소유하고 액세스 관리 정책을 사용하여 데이터에 대한 액세스를 독점적으로 관리합니다.
- IAM Access Analyzer for S3 - S3 버킷 액세스 정책을 평가 및 모니터링하여 정책이 S3 리소스에 대한 의도된 액세스만 제공하는지 확인합니다.

스토리지 보호 및 모니터링

- 스토리지 보호 - Amazon S3에서 버킷을 만들고 객체를 업로드한 후에는 객체 스토리지를 보호할 수 있습니다. 예를 들어, 재해 복구를 위한 S3 버전 관리, S3 복제 및 다중 리전 액세스 포인트 장애 조치 제어, 데이터 백업을 위한 AWS Backup, 보존 기간 설정, 삭제 및 덮어쓰기 방지, 규정 준수 요구 사항 충족을 위한 S3 Object Lock을 사용할 수 있습니다.
- 스토리지 모니터링 - 모니터링은 Amazon S3와 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다. 스토리지 작업 및 비용을 모니터링할 수 있습니다. 또한 발생하는 다중 지점 실패를 보다 쉽게 디버깅할 수 있도록 AWS 솔루션의 모든 부분으로부터 모니터링 데이터를 수집하는 것이 좋습니다.

  Amazon S3의 분석 및 인사이트를 사용하여 스토리지 사용량을 이해하고, 분석하며, 최적화할 수도 있습니다. 예를 들어, Amazon S3 Storage Lens를 사용하여 스토리지를 이해하고, 분석하며, 최적화할 수 있습니다. S3 Storage Lens는 29개 이상의 사용량 및 활동 지표와 대화형 대시보드를 제공하여 전체 조직, 특정 계정, 리전, 버킷 또는 접두사에 대한 데이터를 집계합니다. 스토리지 클래스 분석을 사용하여 스토리지 액세스 패턴을 분석함으로써 데이터를 보다 비용 효율적인 스토리지 클래스로 이전할 시기를 결정할 수 있습니다. S3 수명 주기를 사용하여 비용을 관리할 수 있습니다.

Amazon S3를 사용한 개발

Amazon S3은 REST 서비스입니다. REST API 또는 프로그래밍 태스크를 간소화하기 위해 기본 Amazon S3 REST API를 래핑하는 AWS SDK 라이브러리를 사용하여 Amazon S3에 요청을 전송할 수 있습니다. AWS Command Line Interface(AWS CLI)를 사용하여 Amazon S3 API를 호출할 수도 있습니다. 자세한 내용은 Amazon S3 API 참조에서 요청을 참조하세요.

Amazon S3 REST API는 Amazon S3에 대한 HTTP 인터페이스입니다. REST API의 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. 자세한 내용은 Amazon S3 API 참조에서 Amazon S3를 사용한 개발을 참조하세요.

선택한 언어를 사용하여 애플리케이션을 구축할 수 있도록 다음 리소스를 제공합니다.

AWS CLI

AWS CLI를 사용하여 Amazon S3의 기능에 액세스할 수 있습니다. AWS CLI를 다운로드하고 구성하려면 Amazon S3 API 참조의 AWS CLI를 사용하여 Amazon S3에서 개발을 참조하세요.

AWS CLI는 Amazon S3에 액세스하기 위해 상위 수준(s3) 명령과 API 수준(s3api 및 `s3control`) 명령의 2가지 명령 계층을 제공합니다. 상위 수준 S3 명령은 객체 및 버킷 생성, 조작 및 삭제와 같이 일반적인 작업 수행을 간소화합니다. s3api 및 s3control 명령은 모든 Amazon S3 API 작업에 대한 직접 액세스를 노출하며, 이를 사용하여 상위 수준 명령만으로는 가능하지 않을 수 있는 고급 작업을 수행할 수 있습니다.

Amazon S3 AWS CLI 명령 목록에 대해서는 s3, s3api 및 s3control을 참조하세요.

AWS SDK 및 Explorer

Amazon S3로 애플리케이션을 개발할 때 AWS SDK를 사용할 수 있습니다. AWS SDK에서는 기본 REST API를 래핑하여 프로그래밍 태스크를 단순화합니다. 또한, AWS를 사용하여 커넥티드 모바일 애플리케이션과 웹 애플리케이션을 구축하는 데에 AWS Mobile SDK와 Amplify JavaScript 라이브러리를 사용할 수 있습니다.

AWS SDK 외에도, Visual Studio 및 Eclipse for Java IDE에서 AWS Explorer를 사용할 수 있습니다. 이 경우 SDK 및 Explorer가 AWS 도구 키트로 함께 번들됩니다.

자세한 내용은 Amazon S3 API 참조의 AWS SDK를 사용하여 Amazon S3에서 개발을 참조하세요.

샘플 코드 및 라이브러리

AWS 개발자 센터 및 AWS 코드 샘플 카탈로그에는 Amazon S3용으로 특별히 작성된 샘플 코드와 라이브러리가 있습니다. 이 샘플 코드를 사용하여 Amazon S3 API를 구현하는 방법을 이해할 수 있습니다. 또한 Amazon Simple Storage Service API Reference를 검토하여 Amazon S3 API 작업에 대해 자세히 이해할 수 있습니다.

자습서로 배우기

단계별 자습서를 통해 Amazon S3에 대해 자세히 알아볼 수 있습니다. 이러한 자습서는 가상 회사 이름 및 사용자 이름 등을 사용하여 랩 유형의 환경에 맞게 고안되었습니다. 그 목적은 일반적인 지침을 제공하는 것입니다. 조직의 고유한 요구 사항에 맞는지 주의 깊은 검토 및 조정 없이 프로덕션 환경에 바로 사용할 수 있도록 고안된 것이 아닙니다.

시작

- 자습서: Amazon S3를 사용하여 파일 저장 및 검색
- 자습서: S3 Intelligent-Tiering 사용 시작하기
- 자습서: Amazon S3 Glacier 스토리지 클래스 사용 시작하기

스토리지 비용 최적화

- 자습서: S3 Intelligent-Tiering 사용 시작하기
- 자습서: Amazon S3 Glacier 스토리지 클래스 사용 시작하기
- 자습서: S3 스토리지 렌즈로 비용 최적화 및 사용에 대한 가시성 확보

스토리지 관리

- 자습서: Amazon S3 다중 리전 액세스 포인트 시작하기
- 자습서: S3 배치 복제를 사용하여 Amazon S3 버킷의 기존 객체 복제

비디오 및 웹 사이트 호스팅

- 자습서: Amazon S3, Amazon CloudFront 및 Amazon Route 53로 온디맨드 스트리밍 비디오 호스팅
- 자습서: Amazon S3에서 정적 웹 사이트 구성
- 자습서: Route 53에 등록된 사용자 지정 도메인을 사용하여 정적 웹 사이트 구성

데이터 처리

- 자습서: S3 객체 Lambda를 사용하여 애플리케이션의 데이터 변환
- 자습서: S3 객체 Lambda 및 Amazon Comprehend를 사용하여 PII 데이터 감지 및 수정

- Tutorial: Using S3 Object Lambda to dynamically watermark images as they are retrieved(자습서: S3 객체 Lambda를 사용하여 이미지를 검색할 때 동적으로 워터마크 지정)
- 자습서: S3 Batch Operations를 통해 비디오 일괄 트랜스코딩

데이터 보호

- 자습서: 추가 체크섬을 사용하여 Amazon S3 내 데이터 무결성 확인
- 자습서: S3 복제를 사용하여AWS 리전 내 및 사이에 데이터 복제
- 자습서: S3 버전 관리, S3 객체 잠금, S3 복제를 사용하여 우발적인 삭제나 애플리케이션 버그로부터 Amazon S3의 데이터 보호
- 자습서: S3 배치 복제를 사용하여 Amazon S3 버킷의 기존 객체 복제

교육 및 지원 살펴보기

AWS 전문가의 도움을 받아 기술을 발전시키고 목표를 달성하는 데 도움을 받을 수 있습니다.

- 교육 - 교육 리소스는 Amazon S3 학습에 대한 실무적인 접근법을 제공합니다. 자세한 내용은 AWS 교육 및 자격증과 AWS 온라인 기술을 참조하세요.
- 토론 포럼 - 포럼에서 게시물을 검토하여 Amazon S3에서 수행 가능한 작업과 그렇지 않은 작업에 대해 이해할 수 있습니다. 질문을 게시할 수도 있습니다. 자세한 내용은 토론 포럼을 참조하세요.
- 기술 지원 - 추가 질문이 있는 경우 기술 지원에 연락할 수 있습니다.

AWS CLI를 사용하여 Amazon S3 시작하기

AWS Command Line Interface(AWS CLI)를 사용하여 버킷과 객체를 활용해 Amazon S3를 시작할 수 있습니다. 버킷은 객체에 대한 컨테이너입니다. 객체는 파일과 해당 파일을 설명하는 모든 메타데이터입니다.

Amazon S3에 객체를 저장하려면 버킷을 생성한 다음 버킷에 객체를 업로드합니다. 객체가 버킷에 있으면 객체를 열고 다운로드하고 이동할 수 있습니다. 객체나 버킷이 더 이상 필요하지 않은 경우 리소스를 정리할 수 있습니다.

Amazon S3에서는 사용한 만큼만 비용을 청구하며, Amazon S3 기능 및 요금에 대한 자세한 내용은 Amazon S3를 참조하십시오. 신규 Amazon S3 고객인 경우 Amazon S3를 무료로 시작할 수 있습니다. 자세한 내용은 AWS 프리 티어를 참조하십시오.

> **Note**
>
> Amazon S3 Express One Zone 스토리지 클래스를 디렉터리 버킷과 함께 사용하는 방법에 대한 자세한 내용은 튜토리얼: S3 Express One Zone 시작하기 및 디렉터리 버킷 작업 섹션을 참조하세요.

설정

Amazon S3에서 AWS CLI를 사용하기 전에 다음 사항을 충족해야 합니다.

- AWS 계정에 가입합니다. 지침은 Amazon S3 설정 섹션을 참조하세요.
- `s3:*` 권한이 있는 사용자를 생성합니다. 지침은 Amazon S3 설정 섹션을 참조하세요.
- AWS CLI를 설치하고 구성했습니다. 지침을 보려면 AWS Command Line Interface 사용 설명서의 AWS CLI의 최신 버전 설치 또는 업데이트를 참조하세요.

AWS CLI가 제대로 구성되어 있는지 확인하려면 다음 명령을 실행합니다.

```
aws sts get-caller-identity
```

자세한 내용은 AWS CLI Command Reference의 get-caller-identity를 참조하세요.

이 명령은 AWS CLI가 올바르게 설치 및 구성된 경우 사용 가능한 Amazon S3 명령 목록을 표시합니다.

1단계: 첫 번째 Amazon S3 버킷 생성

AWS CLI에 가입했다면 Amazon S3에 버킷을 만들 준비가 된 것입니다. Amazon S3의 모든 객체는 버킷에 저장됩니다. 데이터를 Amazon S3에 저장하기 전에 버킷을 만들어야 합니다.

> **Note**
>
> 버킷을 만드는 데는 요금이 청구되지 않습니다. 객체를 버킷에 저장하거나 객체를 버킷과 주고받은 경우에만 요금이 청구됩니다. 이 설명서의 예제를 수행하는 동안 1 USD 미만의 최소 요금이 발생합니다. 스토리지 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

버킷을 생성하려면

1. `s3api create-bucket` 명령을 사용하여 버킷을 만듭니다. *amzn-s3-demo-bucket*을 고유한 버킷 이름으로 바꾸고 `us-east-1`을 원하는 리전으로 바꿉니다.

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket --region us-east-1
```

us-east-1 이외의 리전의 경우 위치 제약 조건을 지정해야 합니다.

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket --region us-west-2 --create-
bucket-configuration LocationConstraint=us-west-2
```

> **Note**
>
> - 버킷을 만든 후에는 리전을 변경할 수 없습니다.
> - 지연 시간과 요금을 최소화하고 규제 요건을 충족하려면 가장 가까운 리전을 선택하십시오. 특정 리전에 저장된 객체는 사용자가 명시적으로 객체를 다른 리전으로 전송하지 않는 한 해당 리전을 벗어나지 않습니다. Amazon S3 AWS 리전 목록은 Amazon Web Services 일반 참조의 AWS 서비스 엔드포인트를 참조하세요.
> - 버킷 이름은 다음과 같아야 합니다.
>   - 파티션 내에서 고유해야 합니다. 파티션은 리전의 그룹입니다. AWS에는 `aws`(상용 리전), `aws-cn`(중국 리전) 및 `aws-us-gov`(AWS GovCloud (US) 리전), 이렇게 세 가지 파티션이 있습니다.
>   - 3~63자 이내여야 합니다.
>   - 소문자, 숫자, 마침표(.) 및 하이픈(-)만 포함해야 합니다. 최상의 호환성을 위해 정적 웹 사이트 호스팅에만 사용되는 버킷을 제외하고 버킷 이름에 마침표(.)를 사용하지 않는 것이 좋습니다.
>   - 문자나 숫자로 시작하고 끝나야 합니다.
> - 버킷을 생성한 후에는 해당 이름을 변경할 수 없습니다.
> - 버킷 이름에 민감한 정보가 포함되어서는 안 됩니다. 버킷 이름은 버킷의 객체를 가리키는 URL에 표시됩니다.

2. 모든 버킷을 나열하여 버킷이 생성되었는지 확인합니다.

```
aws s3 ls
```

3. 객체 소유권에서 ACL을 비활성화하거나 활성화하고 버킷에 업로드된 객체의 소유권을 제어할 수 있습니다.

   ACL 비활성화됨

   버킷 소유자 적용(기본값) 설정 - ACL이 비활성화되고 버킷 소유자가 범용 버킷의 모든 객체를 자동으로 소유하고 완전히 제어합니다.

   ```
   aws s3api put-bucket-ownership-controls --bucket amzn-s3-demo-bucket --ownership-controls="Rules=[{ObjectOwnership=BucketOwnerEnforced}]"
   ```

   > **Note**
   >
   > 기본적으로 ACL은 비활성화되어 있습니다. Amazon S3의 최신 사용 사례 대부분은 더 이상 ACL을 사용할 필요가 없습니다. 각 객체에 대해 액세스를 개별적으로 제어해야 하는 드문 상황을 제외하고는 ACL을 비활성화한 채로 두는 것이 좋습니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

   ACL 활성화됨

   - 버킷 소유자 기본 설정 – `bucket-owner-full-control` 미리 준비된 ACL을 사용하여 다른 계정이 버킷에 작성하는 새 객체를 버킷 소유자가 소유하고 완전히 제어합니다.

     ```
     aws s3api put-bucket-ownership-controls --bucket amzn-s3-demo-bucket --ownership-controls="Rules=[{ObjectOwnership=BucketOwnerPreferred}]"
     ```

     버킷 소유자 기본 설정을 적용하는 경우 모든 Amazon S3 업로드에 `bucket-owner-full-control` 미리 준비된 ACL을 포함하도록 요구하려면 이 ACL을 사용하는 객체 업로드만 허용하는 Amazon S3 PUT 작업에 bucket-owner-full-control 미리 제공된 ACL 요구(버킷 소유자 기본)를 수행할 수 있습니다.
   - 객체 라이터 설정 – 객체를 업로드하는 AWS 계정이 객체를 소유하고 완전히 제어하며 ACL을 통해 다른 사용자에게 이에 대한 액세스 권한을 부여할 수 있습니다.

     ```
     aws s3api put-bucket-ownership-controls --bucket amzn-s3-demo-bucket --ownership-controls="Rules=[{ObjectOwnership=ObjectWriter}]"
     ```

> **Note**
>
> 기본 설정은 버킷 소유자 적용입니다. 기본 설정을 적용하고 ACL을 비활성화된 상태로 유지하려면 `s3:CreateBucket` 권한만 있으면 됩니다. ACL을 활성화하려면 `s3:PutBucketOwnershipControls` 권한이 있어야 합니다.

버킷의 현재 객체 소유권 설정을 확인하는 방법:

```
aws s3api get-bucket-ownership-controls --bucket amzn-s3-demo-bucket
```

4. 퍼블릭 액세스 차단이 활성화되어 있는지 확인하는 방법(새 버킷에 대해 기본적으로 활성화되어 있음):

```
aws s3api get-public-access-block --bucket amzn-s3-demo-bucket
```

기본적으로, 신규 버킷에는 네 개의 퍼블릭 액세스 차단 설정이 모두 활성화되어 있습니다. 특정 사용 사례에 대해 하나 이상의 설정을 해제해야 하는 경우가 아니라면 모든 설정을 활성화된 상태로 유지하는 것이 좋습니다. 퍼블릭 액세스 차단에 대한 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하십시오.

퍼블릭 액세스 차단을 활성화해야 하는 경우 다음 명령을 사용합니다.

```
aws s3api put-public-access-block --bucket amzn-s3-demo-bucket --public-access-
block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=t
```

> **Note**
>
> 모든 퍼블릭 액세스 차단 설정을 활성화하려면 `s3:CreateBucket` 권한만 있으면 됩니다. 퍼블릭 액세스 차단 설정을 해제하려면 `s3:PutBucketPublicAccessBlock` 권한이 있어야 합니다.

5. 버킷의 버전 관리를 활성화하는 방법:

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-
configuration Status=Enabled
```

기본적으로, 버킷 버전 관리는 비활성화되어 있습니다. 버전 관리는 동일 버킷 내에 여러 개의 객체 변형을 보유하는 것을 의미합니다. 버전 관리를 사용하여 버킷에 저장된 모든 버전의 객체를 모두 보존, 검색 및 복원할 수 있습니다. 또한, 의도치 않은 사용자 작업 및 애플리케이션 장애가 발생해도 쉽게 복구할 수 있습니다. 버전 관리에 대한 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하십시오.

6. Amazon S3 Object Lock을 사용하면 새 객체가 삭제되거나 덮어쓰이지 않도록 보호할 수 있습니다. 자세한 내용은 Object Lock으로 객체 잠금 섹션을 참조하세요. Object Lock으로 객체 잠금을 활성화하는 방법(버킷 버전 관리 필요):

   새 버킷의 경우:

   ```
   aws s3api create-bucket --bucket amzn-s3-demo-bucket --region us-east-1 --object-
   lock-enabled-for-bucket
   ```

   기존 버킷의 경우:

   ```
   aws s3api put-object-lock-configuration --bucket amzn-s3-demo-bucket --object-lock-
   configuration '{"ObjectLockEnabled": "Enabled"}'
   ```

   Object Lock 활성화와 함께 기본값인 Object Lock으로 객체 잠금을 설정하려면 다음을 사용할 수 있습니다.

   ```
   aws s3api put-object-lock-configuration --bucket amzn-s3-demo-bucket --object-
   lock-configuration '{"ObjectLockEnabled":"Enabled","Rule":{"DefaultRetention":
   {"Mode":"COMPLIANCE","Days":30}}}'
   ```

   덜 제한적인 모드의 경우 "COMPLIANCE"를 "GOVERNANCE"로 바꾸고 필요에 따라 일수를 조정할 수 있습니다.

> **Note**
>
> Object Lock을 사용하는 버킷을 만들려면 `s3:CreateBucket`, `s3:PutBucketVersioning` 및 `s3:PutBucketObjectLockConfiguration` 권한이 있어야 합니다.

7. 버킷에 태그를 추가할 수 있습니다. AWS 비용 할당을 하면서 버킷 태그를 지정해 버킷 사용에 대한 요금 청구 주석을 달 수 있습니다. 태그는 버킷에 할당된 라벨을 나타내는 한 쌍의 키-값입니다. 자세한 내용은 비용 할당 S3 버킷 태그 사용 섹션을 참조하세요.

   버킷에 태그를 추가하는 방법:

   ```
   aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging
    'TagSet=[{Key=Purpose,Value=Testing},{Key=Environment,Value=Development}]'
   ```

8. 버킷과 새 객체는 Amazon S3 관리형 키(SSE-S3)를 암호화 구성의 기본 수준으로 사용하는 서버 측 암호화로 암호화됩니다. 버킷의 기본 암호화를 확인하려면 다음 명령을 사용합니다.

   ```
   aws s3api get-bucket-encryption --bucket amzn-s3-demo-bucket
   ```

   또한 AWS KMS 키(SSE-KMS)를 사용한 서버 측 암호화와 버킷의 AWS KMS 키(DSSE-KMS)를 사용한 이중 계층 서버 측 암호화를 구성할 수 있습니다. AWS 관리형 키(aws/s3)와 고객 관리형 키를 모두 SSE-KMS 및 DSSE-KMS 암호화 구성의 AWS KMS 키로 사용할 수 있습니다. 고객 관리형 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 키 및 AWS 키를 참조하십시오. AWS KMS 키 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 Creating keys를 참조하세요.

   > **Important**
   >
   > AWS KMS 키는 Amazon S3 버킷과 동일한 AWS 리전에 있어야 합니다. 교차 리전 KMS 키는 Amazon S3 버킷 암호화에 지원되지 않습니다.

   SSE-KMS를 통해 기본 암호화를 사용하도록 버킷을 구성할 때 Amazon S3 버킷 키를 사용할 수도 있습니다. Amazon S3 버킷 키를 사용하면 Amazon S3에서 AWS KMS로의 요청 트래픽이 줄어 암호화 비용이 절감됩니다. 자세한 내용은 Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감 섹션을 참조하세요. Amazon S3 버킷 키는 DSSE-KMS에서 지원되지 않습니다. AWS CLI에서

는 새 버킷을 생성할 때 Amazon S3 버킷 키가 기본적으로 활성화되지 않습니다. 기본적으로 활성화되어 있는 콘솔과는 다릅니다.

SSE-KMS를 구성하고 Amazon S3 버킷 키를 활성화하는 방법:

```
aws s3api put-bucket-encryption --bucket amzn-s3-demo-bucket --server-side-
encryption-configuration "{\"Rules\":[{\"ApplyServerSideEncryptionByDefault
\":{\"SSEAlgorithm\":\"aws:kms\",\"KMSMasterKeyID\":\"YOUR-KMS-KEY-ARN\"},
\"BucketKeyEnabled\":true}]}"
```

버킷에 Amazon S3 버킷 키가 활성화되어 있는지 확인하는 방법:

```
aws s3api get-bucket-encryption --bucket amzn-s3-demo-bucket
```

출력에는 `true` 또는 `false`로 설정된 `BucketKeyEnabled` 필드가 포함됩니다.

DSSE-KMS를 구성하려면 다음 명령을 사용합니다.

```
aws s3api put-bucket-encryption --bucket amzn-s3-demo-bucket --server-side-
encryption-configuration '{"Rules":[{"ApplyServerSideEncryptionByDefault":
{"SSEAlgorithm":"aws:kms:dsse","KMSMasterKeyID":"YOUR-KMS-KEY-ARN"}}]}'
```

기본 암호화에 대한 자세한 내용은 Amazon S3 버킷에 대한 기본 서버 측 암호화 동작 설정 섹션을 참조하십시오. SSE-S3에 대한 자세한 내용은 Amazon S3 관리형 키를 사용한 서버 측 암호화 (SSE-S3) 사용 섹션을 참조하세요.

> **Important**
>
> 기본 암호화 구성으로 SSE-KMS 또는 DSSE-KMS 옵션을 사용할 경우 AWS KMS의 초당 요청 수(RPS) 할당량이 적용됩니다. AWS KMS로 전송되는 요청 수를 줄이는 Amazon S3 버킷 키를 활성화하여 KMS API 직접 호출을 줄일 수 있습니다. AWS KMS 할당량과 할당량 증대를 요청하는 방법에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 할당량을 참조하십시오.

Amazon S3에 버킷을 만들었습니다. 다음 단계는 버킷에 객체를 업로드하는 것입니다.

2단계: 버킷에 객체 업로드

Amazon S3에 버킷을 생성했으면 버킷에 객체를 업로드할 준비가 된 것입니다. 텍스트 파일, 사진, 동영상 및 기타 모든 종류의 파일이 객체가 될 수 있습니다.

버킷에 객체를 업로드하려면

1. 업로드할 간단한 텍스트 파일을 생성합니다. 텍스트 편집기를 사용하거나 다음 명령을 실행할 수 있습니다.

   ```
   echo 'Hello, Amazon S3!' > example.txt
   ```

2. s3 cp 명령을 사용하여 파일을 버킷에 업로드합니다.

   ```
   aws s3 cp example.txt s3://amzn-s3-demo-bucket/
   ```

   업로드에 성공한 경우 다음과 유사한 출력 화면이 표시됩니다.

   ```
   upload: ./example.txt to s3://amzn-s3-demo-bucket/example.txt
   ```

3. 버킷의 콘텐츠를 나열하여 객체가 업로드되었는지 확인합니다.

   ```
   aws s3 ls s3://amzn-s3-demo-bucket/
   ```

객체를 버킷에 성공적으로 업로드했습니다. 다음 단계는 객체를 다운로드하는 것입니다.

3단계: 객체 다운로드

버킷에 객체를 업로드한 후에는 객체에 대한 정보를 보고, 로컬 컴퓨터로 객체를 다운로드할 수 있습니다.

Amazon S3 버킷에서 객체를 다운로드하는 방법

1. 객체에 대한 정보를 가져오는 방법:

   ```
   aws s3api head-object --bucket amzn-s3-demo-bucket --key example.txt
   ```

   이 명령은 콘텐츠 유형, 콘텐츠 길이, 마지막으로 수정된 날짜 등 객체에 대한 메타데이터를 반환합니다.

2. 로컬 컴퓨터에 객체를 다운로드합니다.

```
aws s3 cp s3://amzn-s3-demo-bucket/example.txt downloaded-example.txt
```

다운로드에 성공한 경우 다음과 유사한 출력 화면이 표시됩니다.

```
download: s3://amzn-s3-demo-bucket/example.txt to ./downloaded-example.txt
```

3. 다운로드한 파일의 콘텐츠를 확인합니다.

```
cat downloaded-example.txt
```

> **Note**
>
> - 콘솔과 달리 AWS CLI는 와일드카드 또는 `--recursive` 플래그를 사용하여 한 번에 여러 객체를 다운로드할 수 있습니다.
> - AWS CLI를 사용하여 객체를 다운로드할 때 객체 키 이름 끝에 있는 마침표(.)가 보존됩니다(콘솔에서는 제거됨). 이는 객체 키가 마침표로 끝나는 경우에 중요합니다.

여러 객체를 다운로드하는 예:

특정 파일 확장자가 있는 Amazon S3 버킷에서 여러 객체를 다운로드하려면 예시와 같이 `exclude` 및 `include` 필터와 함께 재귀 복사 명령을 사용합니다.

```
aws s3 cp s3://amzn-s3-demo-bucket/ . --recursive --exclude "*" --include "*.txt"
```

객체를 성공적으로 다운로드했습니다. 다음 단계는 폴더에 객체를 복사하는 것입니다.

4단계: 폴더에 객체 복사

이미 버킷에 객체를 추가한 후 객체를 다운로드했습니다. 이제 폴더를 만들고 객체를 폴더에 복사합니다.

객체를 폴더에 복사하려면

1. Amazon S3에서 폴더는 객체 키 이름의 접두사로 표시됩니다. 접두사가 있는 객체를 복사하여 '폴더'를 생성합니다.

```
aws s3 cp s3://amzn-s3-demo-source-bucket/example.txt s3://amzn-s3-demo-destination-bucket/favorite-files/example.txt
```

복사에 성공한 경우 다음과 유사한 출력 화면이 표시됩니다.

```
copy: s3://amzn-s3-demo-source-bucket/example.txt to s3://amzn-s3-demo-destination-bucket/favorite-files/example.txt
```

2. 폴더의 콘텐츠를 나열하여 객체가 복사되었는지 확인합니다.

```
aws s3 ls s3://amzn-s3-demo-destination-bucket/favorite-files/
```

객체를 폴더에 성공적으로 복사했습니다. 다음 단계는 객체 및 버킷을 삭제하는 것입니다.

5단계: 객체 및 버킷 삭제

객체 또는 버킷이 더 이상 필요하지 않은 경우 추가 요금이 부과되지 않도록 객체 또는 버킷을 삭제하는 것이 좋습니다. 실습용으로 이 시작 연습을 완료한 후 버킷 또는 객체를 사용할 계획이 없는 경우 요금이 더 이상 발생하지 않도록 버킷 및 객체를 삭제하는 것이 좋습니다.

버킷을 삭제하기 전에 버킷을 비우거나 버킷의 객체를 삭제합니다. 객체 및 버킷을 삭제한 후에는 더 이상 사용할 수 없습니다.

동일한 버킷 이름을 계속 사용하려면 객체를 삭제하거나 버킷을 비우고 버킷을 삭제하지 않는 것이 좋습니다. 버킷을 삭제하면 이름을 다시 사용할 수 있게 됩니다. 그러나 다른 AWS 계정에서 동일한 이름의 버킷을 생성한 후 다시 사용할 수 있습니다.

객체 삭제

버킷에서 모든 객체를 비우지 않고 삭제할 객체를 선택하려는 경우 객체를 삭제할 수 있습니다.

특정 객체 삭제:

```
aws s3 rm s3://amzn-s3-demo-bucket/example.txt
```

삭제에 성공한 경우 다음과 유사한 출력 화면이 표시됩니다.

```
delete: s3://amzn-s3-demo-bucket/example.txt
```

버킷 비우기

버킷을 삭제하려면 먼저 버킷을 비워야 합니다. 버킷을 비우면 버킷의 모든 객체, 버전, 삭제 마커가 삭제됩니다.

버킷을 비우려면

> **Important**
>
> 버킷을 비우는 작업은 실행 취소할 수 없습니다. 버킷 비우기 작업이 진행되는 동안 버킷에 추가된 객체는 삭제됩니다.

1. 옵션 1: 상대적으로 작은 버킷의 경우 `--recursive` 플래그와 함께 `rm` 명령을 사용하여 버킷의 모든 객체를 삭제합니다.

   ```
   aws s3 rm s3://amzn-s3-demo-bucket --recursive
   ```

   이 명령은 폴더의 객체를 포함하여 버킷의 모든 객체를 삭제합니다.

   > **Note**
   >
   > 버킷에 많은 객체 또는 큰 객체가 포함된 경우 이 명령이 시간 초과될 수 있습니다. 데이터가 많은 버킷의 경우 Amazon S3 수명 주기 규칙을 사용하여 버킷의 객체를 만료시킵니다.

   옵션 2: Amazon S3 수명 주기 규칙 사용(대규모 버킷에 권장)

   객체가 많거나 큰 객체가 있는 버킷의 경우 Amazon S3 수명 주기 규칙을 사용하여 모든 객체를 자동으로 만료시키고 삭제합니다. 수명 주기 규칙이 처리될 때까지 기다립니다(최대 24시간이 걸릴 수 있음). 버킷을 비우기 위해 수명 주기 규칙을 사용하는 방법에 대한 자세한 내용은 수명 주기 구성 규칙을 사용하여 Amazon S3 버킷을 비우려면 어떻게 해야 합니까?를 참조하세요.
2. 버킷이 비어 있는지 확인:

   ```
   aws s3 ls s3://amzn-s3-demo-bucket
   ```
3. 버킷에 버전 관리가 활성화되어 있는 경우 다음 명령을 사용하여 버전이 지정된 객체를 삭제하고 마커를 삭제합니다.

버전이 지정된 객체 제거:

```
aws s3api delete-objects --bucket amzn-s3-demo-bucket --delete "$(aws s3api list-
object-versions --bucket amzn-s3-demo-bucket --output json --query='{Objects:
 Versions[].{Key:Key,VersionId:VersionId}}')"
```

삭제 마커 제거:

```
aws s3api delete-objects --bucket amzn-s3-demo-bucket --delete "$(aws s3api list-
object-versions --bucket amzn-s3-demo-bucket --output json --query='{Objects:
 DeleteMarkers[].{Key:Key,VersionId:VersionId}}')"
```

4. 버킷에 객체 버전과 삭제 마커가 전혀 없는지 확인:

```
aws s3api list-object-versions --bucket amzn-s3-demo-bucket
```

출력에 버전이나 삭제 마커가 전혀 남아 있지 않아야 합니다.

버킷 삭제

버킷을 비우거나 버킷에서 모든 객체를 삭제한 후 버킷을 삭제할 수 있습니다.

> **Important**
>
> 버킷 삭제는 실행 취소할 수 없습니다. 버킷 이름은 고유합니다. 버킷을 삭제하면 다른 AWS 사용자가 해당 이름을 사용할 수 있습니다. 같은 버킷 이름을 계속 사용하려면 버킷을 삭제하지 마십시오. 대신 버킷을 비우고 보관하십시오.

버킷을 삭제하는 방법

1. 버킷 삭제:

```
aws s3api delete-bucket --bucket amzn-s3-demo-bucket
```

2. 모든 버킷을 나열하여 버킷이 삭제되었는지 확인합니다.

```
aws s3 ls
```

다음 단계

이전 예시에서는 AWS CLI를 사용하여 몇 가지 기본 Amazon S3 작업을 수행하는 방법을 알아보았습니다.

다음 주제에서는 사용자가 직접 애플리케이션에서 구현할 수 있도록 Amazon S3을 보다 심도 있게 이해할 수 있는 학습 경로를 설명합니다.

- 일반적인 사용 사례 이해
- 버킷 및 객체에 대한 액세스 제어
- 스토리지 보호 및 모니터링
- Amazon S3를 사용한 개발
- 일반적인 사용 사례 이해
- 교육 및 지원 살펴보기

다음 목록은 Amazon S3의 일반적인 AWS CLI 명령을 보여줍니다.

- cp - 로컬 파일 시스템과 Amazon S3 사이 또는 Amazon S3 위치 간에 파일 또는 객체를 복사합니다.
- ls - 지정된 버킷 및 접두사 아래에 Amazon S3 객체 및 공통 접두사를 나열합니다.
- mb - Amazon S3 버킷을 생성합니다.
- mv - 로컬 파일 시스템과 Amazon S3 사이 또는 Amazon S3 위치 간에 파일 또는 객체를 이동합니다.
- presign - AWS Amazon S3 객체에 대해 자격 증명 없이 임시 액세스를 허용하는 미리 서명된 URL을 생성합니다.
- rb - 빈 Amazon S3 버킷을 제거합니다. `--force` 플래그를 사용하여 하나의 명령으로 콘텐츠가 포함된 버킷을 자동으로 비우고 삭제할 수 있습니다. 이 작업은 실행 취소할 수 없습니다.
- rm - Amazon S3에서 객체를 삭제합니다.
- sync - 소스 디렉터리에서 대상으로 새 파일과 업데이트된 파일을 재귀적으로 복사하여 디렉터리와 Amazon S3 접두사를 동기화합니다.
- website - 버킷을 정적 웹 사이트로 구성합니다.

Amazon S3의 AWS CLI 명령에 대한 자세한 내용은 다음 리소스를 참조하세요.

- s3 - 일반적인 작업을 간소화하는 상위 수준의 Amazon S3 명령입니다.

- s3api - 모든 Amazon S3 API 작업에 직접 액세스합니다.
- s3control - 모든 Amazon S3 Control API 작업에 직접 액세스합니다.

Amazon S3 범용 버킷 생성, 구성 및 작업

Amazon S3에 데이터를 저장하려면 버킷 및 객체라는 리소스를 사용합니다. 버킷은 객체에 대한 컨테이너입니다. 객체는 파일과 해당 파일을 설명하는 모든 메타데이터입니다.

Amazon S3에 객체를 저장하려면 버킷을 생성한 다음 버킷에 객체를 업로드합니다. 객체가 버킷에 있으면 객체를 열고 다운로드하고 이동할 수 있습니다. 객체나 버킷이 더 이상 필요하지 않은 경우 리소스를 정리할 수 있습니다.

이 섹션의 항목에서는 Amazon S3의 범용 버킷 작업을 개괄적으로 설명합니다. 여기에는 범용 버킷 이름 지정, 생성, 액세스 및 삭제에 대한 정보가 포함됩니다. 버킷에서 객체 보기 또는 나열에 대한 자세한 내용은 개체 구성, 나열 및 작업 섹션을 참조하세요.

Amazon S3 버킷에는 여러 유형이 있습니다. 버킷을 만들기 전에 애플리케이션 및 성능 요구 사항에 가장 적합한 버킷 유형을 선택하세요. 다양한 버킷 유형과 각 버킷의 적절한 사용 사례에 대한 자세한 내용은 버킷 섹션을 참조하세요.

> **Note**
>
> Amazon S3 Express One Zone 스토리지 클래스를 디렉터리 버킷과 함께 사용하는 방법에 대한 자세한 내용은 S3 Express One Zone 및 디렉터리 버킷 작업 섹션을 참조하세요.

> **Note**
>
> Amazon S3에서는 사용한 만큼만 비용을 청구하며, Amazon S3 기능 및 요금에 대한 자세한 내용은 Amazon S3를 참조하십시오. 신규 Amazon S3 고객인 경우 Amazon S3를 무료로 시작할 수 있습니다. 자세한 내용은 AWS 프리 티어를 참조하십시오.

주제

- 범용 버킷 개요
- Amazon S3에서 애플리케이션을 빌드하는 일반적인 범용 버킷 패턴
- 범용 버킷 이름 지정 규칙
- 범용 버킷 할당량, 제한 및 규제
- Amazon S3 범용 버킷 액세스

- 범용 버킷 생성
- S3 범용 버킷에 대한 속성 보기
- Amazon S3 범용 버킷 나열
- 범용 버킷 비우기
- 범용 버킷 삭제
- Amazon S3 버킷을 로컬 파일 시스템으로 마운트
- Storage Browser for Amazon S3 작업
- Amazon S3 Transfer Acceleration을 사용하여 빠르고 안전한 파일 전송 구성
- 스토리지 전송 및 사용량에 대한 요청자 지불 범용 버킷 사용

범용 버킷 개요

Amazon S3에 데이터(사진, 동영상, 문서 등)를 업로드하려면 우선 하나의 AWS 리전에 S3 버킷을 만들어야 합니다.

Amazon S3 버킷에는 여러 유형이 있습니다. 버킷을 만들기 전에 애플리케이션 및 성능 요구 사항에 가장 적합한 버킷 유형을 선택하세요. 다양한 버킷 유형과 각 버킷의 적절한 사용 사례에 대한 자세한 내용은 버킷 섹션을 참조하세요.

다음 섹션에서는 버킷 이름 지정 규칙, 할당량 및 버킷 구성 세부 정보를 포함하여 범용 버킷에 대한 자세한 정보를 제공합니다. Amazon S3 버킷과 관련된 규제 및 제한 목록은 범용 버킷 할당량, 제한 및 규제 섹션을 참조하세요.

주제

- 범용 버킷 개요
- 일반적인 범용 버킷 패턴
- 권한
- 범용 버킷에 대한 퍼블릭 액세스 관리
- 범용 버킷 구성 옵션
- 범용 버킷 작업
- 범용 버킷 성능 모니터링

범용 버킷 개요

모든 객체는 어떤 버킷에 포함됩니다. 예를 들어 `photos/puppy.jpg`로 명명된 객체는 미국 서부(오리건) 리전의 `amzn-s3-demo-bucket` 범용 버킷에 저장되며 URL `https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/photos/puppy.jpg`를 사용하여 주소를 지정할 수 있습니다. 자세한 내용은 버킷 액세스를 참조하십시오.

- 상업 리전의 범용 버킷 할당량은 미국 동부(버지니아 북부)에서만 보고 관리할 수 있습니다.
- AWS GovCloud (US)에 대한 범용 버킷 할당량은 AWS GovCloud(미국 서부)에서만 보고 관리할 수 있습니다.

구현 측면에서 버킷과 객체는 AWS 리소스에 해당되며 Amazon S3는 이를 관리하기 위한 API를 제공합니다. 예를 들면 Amazon S3 API를 사용하여 버킷을 만들고 객체를 업로드할 수 있습니다. 이러한 작업은 Amazon S3 콘솔을 사용하여 수행할 수도 있습니다. 콘솔은 Amazon S3 API를 사용하여 요청을 Amazon S3로 보냅니다.

이 섹션에서는 범용 버킷 작업 방법에 대해 설명합니다. 객체 작업에 대한 자세한 내용은 Amazon S3 객체 개요을 참조하십시오.

Amazon S3는 글로벌 범용 버킷을 지원합니다. 즉, 각 버킷 이름은 파티션 내 모든 AWS 리전의 AWS 계정에서 고유해야 합니다. 파티션은 리전 그룹입니다. AWS에는 `aws`(표준 리전), `aws-cn`(중국 리전) 및 `aws-us-gov`(AWS GovCloud (US))의 세 가지 파티션이 있습니다.

범용 버킷이 생성된 후에는 해당 버킷이 삭제될 때까지 동일한 파티션의 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 범용 버킷 이름 지정 규칙을 참조하십시오.

Amazon S3는 사용자가 지정한 리전에 버킷을 만듭니다. 지연 시간을 줄이고, 비용을 최소화하며, 규제 요건을 해결하려면 지리적으로 가까운 AWS 리전을 선택합니다. 예를 들어 유럽에 거주할 경우, 유럽(아일랜드) 또는 유럽(프랑크푸르트) 리전에서 버킷을 만드는 것이 유리할 수 있습니다. Amazon S3 리전 목록은 AWS 일반 참조의 리전 및 엔드포인트를 참조하세요.

> **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.

일반적인 범용 버킷 패턴

Amazon S3에서 애플리케이션을 빌드할 때 고유한 범용 버킷을 사용하여 다양한 데이터세트 또는 워크로드를 분리할 수 있습니다. 사용 사례에 따라 범용 버킷을 사용하는 다양한 설계 패턴과 모범 사례가 있습니다. 자세한 내용은 Amazon S3에서 애플리케이션을 빌드하는 일반적인 범용 버킷 패턴 섹션을 참조하세요.

권한

AWS 계정 루트 사용자 자격 증명을 사용하여 범용 버킷을 생성하고 기타 Amazon S3 작업을 수행할 수 있습니다. 그러나 버킷 생성 등의 요청을 할 때는 AWS 계정의 루트 사용자 자격 증명을 사용하지 않는 것이 좋습니다. 대신 AWS Identity and Access Management(IAM) 사용자를 만들고 이 사용자에게 모든 액세스 권한을 부여합니다(기본적으로 사용자는 권한이 없음).

이러한 사용자를 관리자라고 합니다. 자체 계정의 루트 사용자 자격 증명 대신 관리자 사용자 자격 증명을 사용하여 AWS와 연동하면서 버킷 생성, 사용자 생성, 권한 부여 등의 태스크를 수행할 수 있습니다.

자세한 내용은 AWS 일반 참조의 AWS 계정 루트 사용자 자격 증명 및 IAM 사용자 자격 증명과 IAM 사용 설명서의 IAM 보안 모범 사례를 참조하세요.

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 ID 및 액세스 관리 단원을 참조하십시오.

범용 버킷에 대한 퍼블릭 액세스 관리

버킷 정책, 액세스 제어 목록(ACL) 또는 둘 다를 통해 범용 버킷 및 객체에 대한 퍼블릭 액세스 권한이 부여됩니다. Amazon S3 리소스에 대한 퍼블릭 액세스를 관리하는 데 도움을 주기 위해 Amazon S3는 퍼블릭 액세스 차단 설정을 제공합니다. Amazon S3에서는 이러한 리소스에 대한 퍼블릭 액세스에 대해 일정한 제한을 적용할 수 있도록 퍼블릭 액세스 차단 설정을 통해 ACL 및 버킷 정책을 재정의할 수 있습니다. 개별 버킷 또는 계정의 모든 버킷에 퍼블릭 액세스 차단 설정을 적용할 수 있습니다.

모든 Amazon S3 범용 버킷 및 객체의 퍼블릭 액세스가 차단되도록 하기 위해 새로운 버킷을 생성할 때 기본적으로 퍼블릭 액세스 차단 설정 네 개가 모두 활성화되어 있습니다. 계정에 대해 퍼블릭 액세스 차단 설정 네 개를 모두 켜두는 것이 좋습니다. 이러한 설정은 모든 현재 및 미래 버킷에 대한 모든 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 S3를 사용하여 정적 웹 사이트 호스팅에 설명된 대로 정적 웹 사이트를 호스팅하는 경우), 스토리지 사용 사례에 적합하도록 개별 설정을 사용자 지정할 수 있습니다. 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하세요.

그러나 퍼블릭 액세스 차단 설정을 활성화 상태로 유지하는 것을 권장합니다. 네 개의 퍼블릭 액세스 차단 설정을 모두 활성화하여 정적 웹 사이트를 호스팅하려는 경우 Amazon CloudFront 원본 액세스 제어(OAC)를 사용할 수 있습니다. Amazon CloudFront는 안전한 정적 웹 사이트를 설정하는 데 필요한 기능을 제공합니다. Amazon S3 정적 웹 사이트는 HTTP 엔드포인트만 지원합니다. Amazon CloudFront는 Amazon S3의 내구성 있는 스토리지를 사용하면서 HTTPS와 같은 추가 보안 헤더를 제공합니다. HTTPS는 일반적인 HTTP 요청을 암호화하고 일반적인 사이버 공격으로부터 보호함으로써 보안을 강화합니다.

자세한 내용은 Amazon CloudFront 개발자 안내서의 안전한 정적 웹 사이트 시작하기를 참조하십시오.

> **Note**
>
> 범용 버킷 및 퍼블릭 액세스 설정을 나열할 때 `Error`가 표시되면 필요한 권한이 없을 수 있습니다. 사용자 또는 역할 정책에 다음 권한이 추가되었는지 확인합니다.
>
> ```
> s3:GetAccountPublicAccessBlock
> s3:GetBucketPublicAccessBlock
> s3:GetBucketPolicyStatus
> s3:GetBucketLocation
> s3:GetBucketAcl
> s3:ListAccessPoints
> s3:ListAllMyBuckets
> ```
>
> 드문 경우지만 AWS 리전 중단으로 인해 요청이 실패할 수도 있습니다.

범용 버킷 구성 옵션

Amazon S3는 범용 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 사용하면 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 교차 오리진 리소스 공유(CORS) 사용 섹션을 참조하세요.
이벤트 알림	지정한 버킷 이벤트의 알림을 받도록 버킷을 설정할 수 있습니다. 자세한 내용은 Amazon S3 이벤트 알림 섹션을 참조하세요.
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 관리 섹션을 참조하세요.
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 고객 기반을 이해하고 Amazon S3 청구 비용을 파악할 수 있습니다. 자세한 내용은 서버 액세스 로깅을 사용한 요청 로깅 섹션을 참조하세요.
객체 잠금	S3 객체 잠금을 사용하려면 버킷에 대해 활성화해야 합니다. 버킷에 있는 새 객체에 적용할 기본 보관 모드와 기간을 선택적으로 구성할 수도 있습니다.

하위 리소스	설명
	자세한 내용은 Object Lock으로 객체 잠금 섹션을 참조하세요.
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 policy와 acl 하위 리소스에 권한 정보를 저장합니다. 자세한 내용은 Amazon S3의 ID 및 액세스 관리 섹션을 참조하세요.
복제	복제는 동일한 또는 서로 다른 AWS 리전의 버킷 간에 객체를 비동기식으로 자동 복사하는 것을 말합니다. 자세한 내용은 리전 내 및 리전 간 객체 복제 섹션을 참조하세요.
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 스토리지 전송 및 사용량에 대한 요청자 지불 범용 버킷 사용 섹션을 참조하세요.
tagging	비용 할당 태그를 버킷에 추가하여 AWS 비용을 분류하고 추적할 수 있습니다. 버킷의 태그를 저장 및 관리하기 위해 Amazon S3는 tagging 하위 리소스를 제공합니다. AWS에서는 버킷에 적용된 태그를 사용하여 사용 내역 및 비용을 태그별로 집계한 비용 할당 보고서를 만듭니다. 자세한 내용은 Amazon S3에 결제 및 사용 보고 섹션을 참조하세요.
전송 속도 향상	Transfer Acceleration을 사용하면 클라이언트와 S3 버킷 사이에서 파일을 빠르고 쉽고 안전하게 장거리 전송할 수 있습니다. Transfer Acceleration은 전 세계에 분산된 Amazon CloudFront의 엣지 로케이션을 활용합니다. 자세한 내용은 Amazon S3 Transfer Acceleration을 사용하여 빠르고 안전한 파일 전송 구성 섹션을 참조하세요.

하위 리소스	설명
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 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하세요.
웹 사이트	버킷을 정적 웹 사이트 호스팅용으로 구성할 수 있습니다. Amazon S3는 웹 사이트 하위 리소스를 만들어 이 구성을 저장합니다. 자세한 내용은 Amazon S3를 사용하여 정적 웹 사이트 호스팅 섹션을 참조하세요.

범용 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 범용 버킷을 생성, 삭제 또는 구성하지 않는 것이 좋습니다. 자주 실행하지 않는 별도의 초기화 루틴이나 설정 루틴에서 버킷을 생성, 삭제 또는 구성하는 것이 좋습니다.

범용 버킷 성능 모니터링

AWS 리소스를 사용하는 중요한 애플리케이션과 비즈니스 프로세스가 있는 경우 시스템을 모니터링하고 시스템 알림을 받는 것이 중요합니다. 데이터 모니터링은 Amazon S3와 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 도움이 됩니다. S3 버킷에 대한 지표와 로그를 수집하고 집계하는 데 사용할 수 있는 몇 가지 AWS 서비스가 있습니다.

사용 사례에 따라 문제를 디버깅하거나, 데이터를 모니터링하거나, 스토리지 비용을 최적화하거나, 다중 지점 문제를 해결하는 데 필요한 조직의 요구에 가장 적합한 AWS 서비스를 선택할 수 있습니다. 예시:

- S3를 사용하는 애플리케이션의 성능을 개선하려면 CloudWatch 경보를 설정하여 스토리지 데이터, 복제 지표 또는 요청 지표를 모니터링합니다.
- 스토리지 사용량을 계획하거나, 스토리지 비용을 최적화하거나, 조직 전체에서 보유한 스토리지의 양을 확인하려면 Amazon S3 Storage Lens를 사용합니다. 또는 고급 지표를 활성화하고 세부 상태

코드 지표를 사용하여 성공 또는 실패 요청의 수를 파악함으로써 S3 Storage Lens를 사용하여 데이터 성능을 개선할 수 있습니다.

- 운영 상태에 대한 통합 보기를 원하면 S3 Storage Lens 사용량 및 활동 지표를 Amazon CloudWatch 대시보드에 게시합니다.

> **Note**
>
> Amazon CloudWatch 게시 옵션은 고급 지표 및 권장 사항으로 업그레이드된 S3 Storage Lens 대시보드에서 사용할 수 있습니다. S3 스토리지 렌즈의 신규 또는 기존 대시보드 구성에 CloudWatch 게시 옵션을 사용하도록 설정할 수 있습니다.

- 사용자, 역할 또는 AWS 서비스가 수행한 작업의 레코드를 가져오려면 AWS CloudTrail 로그를 설정합니다. AWS CloudTrail 로그를 사용하여 Amazon S3에 대한 API 직접 호출을 이벤트로 검토할 수도 있습니다.
- S3 버킷에서 특정 이벤트가 발생할 때 알림을 받으려면 Amazon S3 이벤트 알림을 설정합니다.
- S3 버킷에 수행된 요청에 대한 자세한 레코드를 가져오려면 S3 액세스 로그를 설정합니다.

데이터를 모니터링하는 데 사용할 수 있는 다양한 AWS 서비스의 목록은 Amazon S3의 로깅 및 모니터링을 참조하세요.

Amazon S3에서 애플리케이션을 빌드하는 일반적인 범용 버킷 패턴

Amazon S3에서 애플리케이션을 빌드할 때 고유한 범용 버킷을 사용하여 다양한 데이터세트 또는 워크로드를 분리할 수 있습니다. 최종 사용자 또는 다른 사용자 그룹에 서비스를 제공하는 애플리케이션을 빌드할 때는 모범 사례 설계 패턴을 사용하여 Amazon S3 기능과 확장성을 최대한 활용할 수 있는 애플리케이션을 빌드합니다.

> **Important**
>
> 예측할 수 없는 범용 버킷 이름을 생성하는 것이 좋습니다. 이미 버킷을 생성한 경우를 제외하고 선택한 버킷 이름을 사용할 수 있다고 가정하고 코드를 작성하지 마세요. 예측할 수 없는 버킷 이름을 생성하는 한 가지 방법은 버킷 이름에 전역 고유 식별자(GUID)를 추가하는 것입니다(예: `amzn-s3-demo-bucket-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111`). 범용 버킷 이름 지정 규칙에 대한 자세한 내용은 범용 버킷 이름 지정 규칙 섹션을 참조하세요.

주제

- 다중 테넌트 범용 버킷 패턴
- 사용당 버킷 패턴

다중 테넌트 범용 버킷 패턴

다중 테넌트 버킷을 사용하여 팀 또는 워크로드에 대한 단일 범용 버킷을 생성합니다. 고유한 S3 접두사를 사용하여 버킷에 저장하는 객체를 구성합니다. 접두사는 객체 키 이름의 시작 부분에 있는 문자열입니다. 접두사는 임의의 길이일 수 있지만 객체 키 이름의 최대 길이(1,024바이트)를 초과할 수 없습니다. 접두사를 디렉터리와 비슷한 방식으로 데이터를 구성하는 방법으로 생각할 수 있습니다. 그러나 접두사는 디렉터리가 아닙니다.

도시에 대한 정보를 저장하기 위해 대륙, 국가 및 시/도를 기준으로 도시를 구성한다고 가정해 보겠습니다. 이러한 이름에는 보통 문장 부호가 사용되지 않으므로 슬래시(/)를 구분 기호로 선택할 수 있습니다. 다음 예제에서는 슬래시(/) 구분 기호를 사용하여 대륙, 국가, 시/도별로 도시 이름을 구성하는 데 사용되는 접두사를 보여줍니다.

- 유럽/프랑스/누벨아키텐/보르도
- 북미/캐나다/퀘벡/몬트리올
- 북미/미국/워싱턴/벨뷰
- 북미/미국/워싱턴/시애틀

이 패턴의 규모는 범용 버킷 내에 수백 개의 고유한 데이터세트가 있을 때 효과적으로 조정됩니다. 접두사를 사용하면 이러한 데이터세트를 쉽게 구성하고 그룹화할 수 있습니다.

그러나 다중 테넌트 범용 버킷 패턴의 한 가지 잠재적 단점은 기본 버킷 암호화, S3 버전 관리 및 S3 요청자 지불과 같은 많은 S3 버킷 수준 기능이 접두사 수준이 아닌 버킷 수준에서 설정된다는 것입니다. 다중 테넌트 버킷 내의 서로 다른 데이터세트에 고유한 요구 사항이 있을 경우 접두사 수준에서 많은 S3 버킷 수준 기능을 구성할 수 없기 때문에 각 데이터세트에 올바른 설정을 지정하기 어려울 수 있습니다. 또한 다중 테넌트 버킷에서는 특정 접두사와 관련된 스토리지, 요청 및 데이터 전송을 이해하는 과정에서 비용 할당이 복잡해질 수 있습니다.

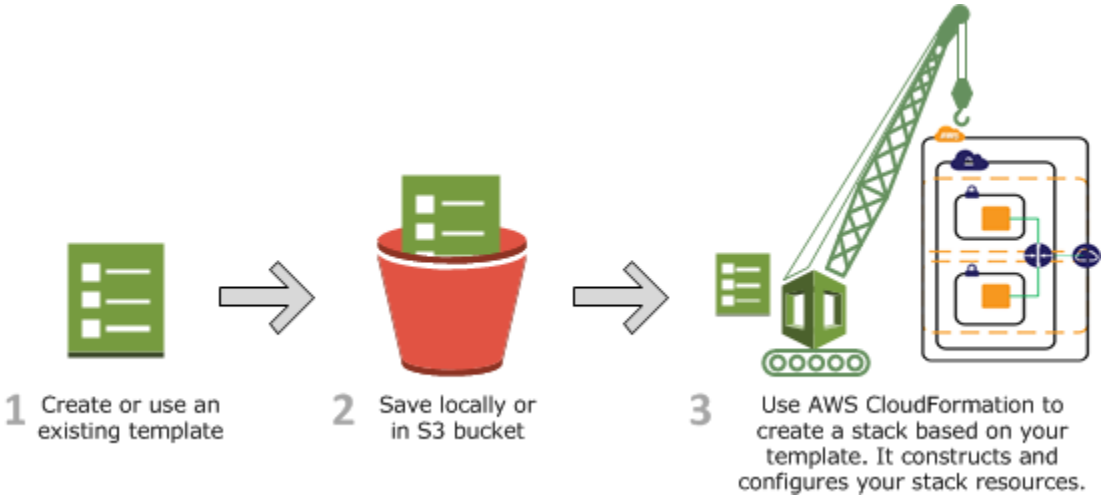
사용당 버킷 패턴

사용당 버킷 패턴을 사용하여 각 개별 데이터세트, 최종 사용자 또는 팀에 대한 범용 버킷을 생성합니다. 이러한 각 버킷에 대해 S3 버킷 수준 기능을 구성할 수 있으므로 이 패턴을 사용하여 고유한 버킷 수준 설정을 구성할 수 있습니다. 예를 들어 각 버킷의 데이터세트에 사용자 지정된 방식으로 기본 버킷 암호화, S3 버전 관리 및 S3 요청자 지불과 같은 기능을 구성할 수 있습니다. 각 개별 데이터세트, 최

종 사용자 또는 팀에 하나의 버킷을 사용하면 액세스 관리 및 비용 할당 전략 모두를 간소화하는 데 도움이 될 수 있습니다.

잠재적으로 수천 개의 버킷을 관리해야 한다는 점이 이 전략의 잠재적 단점입니다. 모든 AWS 계정의 기본 할당량은 범용 버킷 10,000개입니다. 할당량 증가 요청을 제출하여 계정의 버킷 할당량을 늘릴 수 있습니다. 범용 버킷의 증가를 요청하려면 Service Quotas 콘솔을 방문하세요.

AWS CloudFormation를 사용하면 사용별 버킷 패턴을 관리하고 인프라 관리를 간소화할 수 있습니다. S3 범용 버킷에 대해 원하는 모든 설정을 이미 정의한 패턴의 사용자 지정 AWS CloudFormation 템플릿을 생성하여 인프라에 대한 변경 사항을 쉽게 배포하고 추적할 수 있습니다. 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS::S3::Bucket 섹션을 참조하세요.



범용 버킷 이름 지정 규칙

범용 버킷을 만들 때는 버킷 이름의 길이, 유효한 문자, 형식 및 고유성을 고려해야 합니다. 다음 섹션에서는 이름 지정 규칙, 모범 사례, 전역 고유 식별자(GUID)가 포함된 이름으로 범용 버킷을 생성하는 예를 포함하여 범용 버킷 이름 지정에 대한 정보를 제공합니다.

객체 키 이름에 대한 자세한 내용은 Creating object key names를 참조하세요.

범용 버킷을 만들려면 the section called “범용 버킷 생성” 섹션을 참조하세요.

주제

- 범용 버킷 이름 지정 규칙
- 범용 버킷 이름 예시
- 모범 사례
- 버킷 이름에 GUID를 사용하는 버킷 생성

범용 버킷 이름 지정 규칙

범용 버킷에는 다음 이름 지정 규칙이 적용됩니다.

- 버킷 이름은 3자(최소)에서 63자(최대) 사이여야 합니다.
- 버킷 이름은 소문자, 숫자, 마침표(.) 및 하이픈(-)으로만 구성될 수 있습니다.
- 버킷 이름은 문자 또는 숫자로 시작하고 끝나야 합니다.
- 버킷 이름에 두 마침표를 나란히 붙여 사용하면 안 됩니다.
- 버킷 이름은 IP 주소 형식(예: 192.168.5.4)을 사용해서는 안 됩니다.
- 버킷 이름은 접두사 xn--로 시작해서는 안됩니다.
- 버킷 이름은 접두사 sthree-로 시작해서는 안됩니다.
- 버킷 이름은 접두사 amzn-s3-demo-로 시작해서는 안됩니다.
- 버킷 이름은 접미사 -s3alias로 끝나서는 안됩니다. 이 접미사는 액세스 포인트 별칭 이름 용도로 예약되어 있습니다. 자세한 내용은 액세스 포인트 별칭 섹션을 참조하세요.
- 버킷 이름은 접미사 --ol-s3로 끝나서는 안됩니다. 이 접미사는 객체 Lambda 액세스 포인트 별칭 이름 용도로 예약되어 있습니다. 자세한 내용은 S3 버킷 객체 Lambda 액세스 포인트에 버킷 스타일 별칭을 사용하는 방법 섹션을 참조하세요.
- 버킷 이름은 접미사 .mrap로 끝나서는 안됩니다. 이 접미사는 다중 리전 액세스 포인트 이름 용도로 예약되어 있습니다. 자세한 내용은 Amazon S3 다중 리전 액세스 포인트 이름 지정 규칙 섹션을 참조하세요.
- 버킷 이름은 접미사 --x-s3로 끝나서는 안됩니다. 이 접미사는 디렉터리 버킷용으로 예약되어 있습니다. 자세한 내용은 디렉터리 버킷 이름 지정 규칙 섹션을 참조하세요.
- 버킷 이름은 접미사 --table-s3로 끝나서는 안됩니다. 이 접미사는 S3 Tables 버킷용으로 예약되어 있습니다. 자세한 내용은 Amazon S3 테이블 버킷, 테이블 및 네임스페이스 이름 지정 규칙 섹션을 참조하세요.
- Amazon S3 Transfer Acceleration에 사용되는 버킷은 이름에 마침표(.)를 사용할 수 없습니다. Transfer Acceleration에 대한 자세한 내용은 Amazon S3 Transfer Acceleration을 사용하여 빠르고 안전한 파일 전송 구성 단원을 참조하세요.

Important

- 버킷 이름은 파티션 내 모든 AWS 리전의 모든 AWS 계정에서 고유해야 합니다. 파티션은 리전의 그룹입니다. AWS에는 aws(상용 리전), aws-cn(중국 리전) 및 aws-us-gov(AWS GovCloud (US) 리전), 이렇게 세 가지 파티션이 있습니다.

- 동일한 파티션의 다른 AWS 계정은 버킷이 삭제될 때까지 버킷 이름을 사용할 수 없습니다. 버킷을 삭제한 후에는 동일한 파티션의 다른 AWS 계정이 새 버킷에 대해 동일한 버킷 이름을 사용할 수 있으므로 삭제된 버킷에 대한 요청을 수신할 수도 있다는 점에 유의하세요. 이 현상을 방지하고 싶거나 같은 버킷 이름을 사용하려면 버킷을 삭제하지 마세요. 버킷을 비우고 보관한 다음 필요에 따라 버킷 요청을 차단하는 것이 좋습니다. 더 이상 적극적으로 사용되지 않는 버킷의 경우 버킷 자체를 유지하면서 비용을 최소화하기 위해 버킷에서 모든 객체를 비우는 것이 좋습니다.
- 범용 버킷을 만들 때 버킷의 이름과 버킷을 만들 AWS 리전을 선택할 수 있습니다. 범용 버킷을 만든 후에는 이름 또는 리전을 변경할 수 없습니다.
- 버킷 이름에 민감한 정보가 포함되어서는 안 됩니다. 버킷 이름은 버킷의 객체를 가리키는 URL에 표시됩니다.

Note

2018년 3월 1일 이전에 미국 동부(버지니아 북부) 리전에서 생성된 버킷의 이름은 최대 255자이며 대문자와 밑줄을 포함할 수 있었습니다. 2018년 3월 1일부터 미국 동부(버지니아 북부)의 새 버킷은 그 외 모든 리전에서 적용되는 것과 동일한 규칙을 준수해야 합니다.

범용 버킷 이름 예시

다음 버킷 이름은 범용 버킷 이름에 허용되는 문자의 예를 보여줍니다(a~z, 0~9 및 하이픈(-)). `amzn-s3-demo-` 예약 접두사는 여기에서 설명용으로만 사용되었습니다. 예약된 접두사이므로 `amzn-s3-demo-`로 시작하는 버킷 이름을 생성할 수 없습니다.

- `amzn-s3-demo-bucket1-a1b2c3d4-5678-90ab-cdef-example11111`
- `amzn-s3-demo-bucket`

다음 예시 버킷 이름은 유효하지만 마침표(`.`)가 포함되어 있으므로 정적 웹 사이트 호스팅 이외의 용도에는 권장되지 않습니다.

- `example.com`
- `www.example.com`
- `my.example.s3.bucket`

다음 예제 버킷 이름은 유효하지 않습니다.

- amzn_s3_demo_bucket(밑줄 포함)
- AmznS3DemoBucket(대문자 포함)
- amzn-s3-demo-bucket-(amzn-s3-demo- 접두사로 시작하고 하이픈으로 끝남)
- example..com(한 줄에 두 개의 마침표 포함)
- 192.168.5.4(IP 주소 형식과 일치)

모범 사례

범용 버킷 이름을 지정할 때는 다음 버킷 이름 지정 모범 사례를 고려하세요.

이름 충돌을 일으킬 가능성이 낮은 버킷 이름 지정 체계 선택

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 이름에 전역 고유 식별자(GUID) 추가

예측할 수 없는 버킷 이름을 생성하는 것이 좋습니다. 이미 버킷을 생성한 경우가 아니라면 선택한 버킷 이름을 사용할 수 있다고 가정하고 코드를 작성하지 마세요. 예측할 수 없는 버킷 이름을 생성하는 한 가지 방법은 버킷 이름에 전역 고유 식별자(GUID)를 추가하는 것입니다(예: amzn-s3-demo-bucket-a1b2c3d4-5678-90ab-cdef-example11111). 자세한 내용은 the section called “버킷 이름에 GUID를 사용하는 버킷 생성” 섹션을 참조하세요.

버킷 이름에 마침표(.) 사용하지 않기

최상의 호환성을 위해 정적 웹 사이트 호스팅에만 사용되는 버킷을 제외하고 버킷 이름에 마침표(.)를 사용하지 않는 것이 좋습니다. 버킷 이름에 마침표를 포함하는 경우 자체 인증서 검증을 수행하지 않는 한 HTTPS를 통해 가상 호스팅 방식 주소 지정을 사용할 수 없습니다. 버킷의 가상 호스팅에 사용되는 보안 인증서가 이름에 마침표가 있는 버킷에 대해 작동하지 않습니다.

정적 웹 사이트 호스팅은 HTTP를 통해서만 사용할 수 있기 때문에 이 제한은 정적 웹 사이트 호스팅에 사용되는 버킷에는 영향을 주지 않습니다. 가상 호스트 방식 주소 지정에 대한 자세한 내용은 범용 버킷의 가상 호스팅 단원을 참조하세요. 정적 웹 사이트 호스팅에 대한 자세한 내용은 Amazon S3를 사용하여 정적 웹 사이트 호스팅 단원을 참조하세요.

관련 이름 선택

버킷의 이름을 지정할 때는 사용자 또는 사용자의 회사와 관련된 이름을 선택하는 것이 좋습니다. 다른 사용자와 연결된 이름은 사용하면 안 됩니다. 예를 들어 버킷 이름에 AWS 또는 Amazon을 사용하지 마세요.

버킷 이름을 재사용할 수 있도록 버킷을 삭제하지 않기

버킷을 비웠으면 버킷을 삭제할 수 있습니다. 버킷을 삭제한 후 버킷 이름은 다시 사용할 수 있게 됩니다. 하지만 이름을 즉시 또는 언젠가 재사용할 수 있다는 보장이 없습니다. 버킷을 삭제하고 나면 이름을 다시 사용할 수 있을 때까지 약간의 시간이 걸릴 수 있습니다. 또한 이름을 재사용하기 전에 다른 AWS 계정에서 동일한 이름의 버킷을 생성할 수 있습니다.

범용 버킷을 삭제한 후에는 동일한 파티션의 다른 AWS 계정이 새 버킷에 대해 동일한 버킷 이름을 사용할 수 있으므로 삭제된 범용 버킷에 대한 요청을 수신할 수도 있다는 점에 유의하세요. 이 현상을 방지하고 싶거나 같은 범용 버킷 이름을 사용하려면 범용 버킷을 삭제하지 마세요. 버킷을 비우고 보관한 다음 필요에 따라 버킷 요청을 차단하는 것이 좋습니다.

버킷 이름에 GUID를 사용하는 버킷 생성

다음 예제에서는 버킷 이름 끝에 GUID를 사용하는 범용 버킷을 생성하는 방법을 설명합니다.

AWS CLI 사용

다음 AWS CLI 예시에서는 전역 고유 식별자(GUID)를 사용하는 버킷 이름의 예시를 사용하여 미국 서부(캘리포니아 북부) 리전(us-west-1)에 범용 버킷을 생성합니다. 이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3api create-bucket \
    --bucket amzn-s3-demo-bucket1$(uuidgen | tr -d - | tr '[:upper:]' '[:lower:]' ) \
    --region us-west-1 \
    --create-bucket-configuration LocationConstraint=us-west-1
```

Java용 AWS SDK 사용

다음 예시에서는 AWS SDK for Java를 사용하여 미국 동부(버지니아 북부) 리전(us-east-1)의 버킷 이름 끝에 GUID가 있는 버킷을 생성하는 방법을 설명합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다. 다른 AWS SDK에 대한 자세한 내용은 AWS 기반의 도구를 참조하세요.

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.Bucket;
import com.amazonaws.services.s3.model.CreateBucketRequest;

import java.util.List;
import java.util.UUID;

public class CreateBucketWithUUID {
    public static void main(String[] args) {
        final AmazonS3 s3 =
 AmazonS3ClientBuilder.standard().withRegion(Regions.US_EAST_1).build();
        String bucketName = "amzn-s3-demo-bucket" +
 UUID.randomUUID().toString().replace("-", "");
        CreateBucketRequest createRequest = new CreateBucketRequest(bucketName);
        System.out.println(bucketName);
        s3.createBucket(createRequest);
    }
}
```

범용 버킷 할당량, 제한 및 규제

Amazon S3 범용 버킷은 해당 버킷을 생성한 AWS 계정의 소유입니다. 버킷 소유권은 다른 계정으로 양도할 수 없습니다.

버킷 할당량

기본적으로 AWS 계정당 최대 범용 버킷 10,000개를 만들 수 있습니다. 범용 버킷의 할당량 증가를 요청하려면 Service Quotas 콘솔을 방문하세요.

> **Important**
>
> 페이지 매김된 `ListBuckets` 요청만 사용하는 것이 좋습니다. 페이지가 지정되지 않은 `ListBuckets` 요청은 기본 범용 버킷 할당량인 10,000개로 설정된 AWS 계정에 대해서만 지원됩니다. 승인된 범용 버킷 할당량이 10,000개를 초과할 경우 페이지 매김된 `ListBuckets` 요청을 전송하여 계정 버킷을 나열해야 합니다. 범용 버킷 할당량이 10,000개가 넘는 AWS 계정에 대한 페이지가 지정되지 않은 모든 `ListBuckets` 요청은 거부됩니다.

> **Note**
>
> 다음 AWS 리전을 사용하여 할당량, 버킷 사용률을 확인하거나 AWS 계정 내 범용 버킷 증대를 요청해야 합니다.
>
> - 상업 리전의 범용 버킷 할당량은 미국 동부(버지니아 북부)에서만 보고 관리할 수 있습니다.
> - AWS GovCloud (US)에 대한 범용 버킷 할당량은 AWS GovCloud(미국 서부)에서만 보고 관리할 수 있습니다.

서비스 할당량에 대한 정보는 Amazon Web Services 일반 참조의 AWS 서비스 할당량을 참조하세요.

객체 및 버킷 제한

버킷 크기, 즉 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 이름 지정 규칙

버킷을 만들 때 버킷의 이름과 버킷을 만들 AWS 리전을 선택할 수 있습니다. 버킷을 만든 후에는 이름 또는 리전을 변경할 수 없습니다. 버킷 이름 지정에 대한 자세한 내용은 범용 버킷 이름 지정 규칙 단원을 참조하세요.

Amazon S3 범용 버킷 액세스

Amazon S3 콘솔, AWS Command Line Interface, AWS SDK 또는 Amazon S3 REST API를 사용하여 Amazon S3 범용 버킷에 액세스할 수 있습니다. S3 범용 버킷에 액세스하는 각 방법은 특정 사용 사례에 적용됩니다. 자세한 내용은 다음 섹션을 참조하세요.

주제

- 사용 사례
- Amazon S3 콘솔
- AWS CLI
- AWS SDK
- Amazon S3 REST API

사용 사례

Amazon S3 범용 버킷의 사용 사례에 따라 버킷의 기본 데이터에 액세스하는 데 권장되는 다양한 방법이 있습니다. 다음 목록에는 데이터 액세스를 위한 일반적인 사용 사례가 포함되어 있습니다.

- 정적 웹 사이트 – Amazon S3를 사용하여 정적 웹 사이트를 호스팅할 수 있습니다. 이 사용 사례에서는 웹 사이트처럼 작동하도록 범용 S3 버킷을 구성할 수 있습니다. Amazon S3에서의 웹 사이트 호스팅 절차를 단계별로 안내하는 예제는 자습서: Amazon S3에서 정적 웹 사이트 구성을 참조하십시오.

  퍼블릭 액세스 차단과 같은 보안 설정이 활성화된 정적 웹 사이트를 호스팅하려면 원본 액세스 제어(OAC)와 함께 Amazon CloudFront를 사용하고 HTTPS와 같은 추가 보안 헤더를 구현하는 것이 좋습니다. 자세한 내용은 안전한 정적 웹 사이트 시작하기를 참조하십시오.

  > **Note**
  >
  > Amazon S3는 정적 웹 사이트 액세스에 대해 가상 호스팅 방식 URL과 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 범용 버킷 할당량, 제한 및 규제 섹션을 참조하세요.

- 공유 데이터세트 - Amazon S3를 기반으로 규모를 조정할 때 공유 범용 버킷 내의 고유한 접두사에 다양한 최종 고객 또는 사업부를 할당하는 다중 테넌트 모델을 채택하는 것이 일반적입니다. Amazon S3 액세스 포인트를 사용하면 공유 데이터 세트에 액세스해야 하는 각 애플리케이션에 대해 하나의 대형 버킷 정책을 별도의 개별 액세스 포인트 정책으로 나눌 수 있습니다. 이 접근 방식을 사용하면 공유 데이터 세트 내에서 다른 애플리케이션이 수행하는 작업을 방해하지 않으면서 애플리케이션에 적합한 액세스 정책을 빌드하는 데 더 쉽게 집중할 수 있습니다. 자세한 내용은 액세스 포인트로 공유 데이터세트에 대한 액세스 관리 섹션을 참조하세요.
- 높은 처리량 워크로드 – Mountpoint for Amazon S3는 Amazon S3 범용 버킷을 로컬 파일 시스템으로 마운트하는 데 사용되는 높은 처리량 성능의 오픈 소스 파일 클라이언트입니다. Mountpoint를 사용하면 애플리케이션에서 열기 및 읽기와 같은 파일 시스템 작업을 통해 Amazon S3에 저장된 객체에 액세스할 수 있습니다. Mountpoint는 이러한 작업을 S3 객체 API 호출로 자동 변환하여 애플리케이션이 파일 인터페이스를 통해 Amazon S3의 탄력적 스토리지 및 처리량을 이용할 수 있도록 합니다. 자세한 내용은 Amazon S3 버킷을 로컬 파일 시스템으로 마운트 섹션을 참조하세요.
- 다중 리전 애플리케이션 – Amazon S3 다중 리전 액세스 포인트는 애플리케이션이 여러 AWS 리전에 있는 S3 범용 버킷의 요청을 이행하는 데 사용할 수 있는 글로벌 엔드포인트를 제공합니다. 다중 리전 액세스 포인트를 사용하여 단일 리전에서 사용되는 것과 동일한 아키텍처로 다중 리전 애플리케이션을 빌드하면 전 세계 어디에서나 해당 애플리케이션을 실행할 수 있습니다. 다중 리전 액세스

포인트는 퍼블릭 인터넷을 통해 요청을 보내는 대신 Amazon S3에 대한 인터넷 기반 요청의 가속화를 통해 기본 제공 네트워크 복원력을 제공합니다. 자세한 내용은 다중 리전 액세스 포인트로 다중 리전 트래픽 관리 섹션을 참조하세요.

- Secure Shell(SSH) File Transfer 프로토콜(SFTP) - 인터넷을 통해 민감한 데이터를 안전하게 전송하려는 경우 Amazon S3 범용 버킷과 함께 SFTP 지원 서버를 사용할 수 있습니다. AWS SFTP는 SSH의 전체 보안 및 인증 기능을 지원하는 네트워크 프로토콜입니다. 이 프로토콜을 사용하면 사용자 ID, 권한 및 키를 세밀하게 제어하거나 IAM 정책을 사용하여 액세스를 관리할 수 있습니다. SFTP 지원 서버를 Amazon S3 버킷과 연결하려면 먼저 SFTP 지원 서버를 만들어야 합니다. 그런 다음 사용자 계정을 설정하고 서버를 Amazon S3 범용 버킷과 연결합니다. 이 프로세스에 대한 자세한 내용은 AWS 블로그에서 AWS Transfer for SFTP - Amazon S3용 완전 관리형 SFTP 서비스를 참조하십시오.

Amazon S3 콘솔

이 콘솔은 Amazon S3 및 AWS 리소스를 관리하기 위한 웹 기반 사용자 인터페이스입니다. Amazon S3 콘솔을 사용하면 쉽게 버킷에 액세스하고 버킷의 속성을 수정할 수 있습니다. 코드를 작성하지 않고 콘솔 UI를 사용하여 대부분의 버킷 작업을 수행할 수 있습니다.

AWS 계정에 가입한 경우 Amazon S3 콘솔에 로그인한 후 Amazon S3 콘솔 홈페이지에서 S3을 선택하여 Amazon S3 콘솔에 액세스할 수 있습니다. 이 링크를 사용하여 https://console.aws.amazon.com/s3/에 직접 액세스할 수도 있습니다.

AWS CLI

AWS CLI를 통해 시스템 명령줄에서 명령을 실행하거나 스크립트를 빌드하여 AWS(S3 등) 작업을 수행할 수 있습니다. 예를 들어 여러 버킷에 액세스해야 할 경우 AWS CLI를 사용하여 일반적이고 반복적인 작업을 자동화하여 시간을 절약할 수 있습니다. 조직이 확장됨에 따라 일반적인 작업에 대한 스크립팅 가능성과 반복성을 자주 고려합니다.

AWS CLI는 광범위한 AWS 서비스의 명령을 제공합니다. AWS CLI는 Windows, macOS, Linux에서 지원됩니다. 시작하려면 AWS Command Line Interface 사용 설명서를 참조하세요. Amazon S3용 명령에 대한 자세한 내용은 AWS CLI 명령 참조의 s3api 및 s3control을 참조하세요.

AWS SDK

AWS에서는 다양한 프로그래밍 언어 및 플랫폼(Java, Python, Ruby, .NET, iOS, Android 등)을 위한 라이브러리와 샘플 코드로 구성된 소프트웨어 개발 키트(SDK)를 제공합니다. AWS SDK를 사용하면 편리하게 S3 및 AWS에 프로그래밍 방식으로 액세스할 수 있습니다. Amazon S3은 REST 서비스입니다.

프로그래밍 태스크를 간소화하기 위해 기본 Amazon S3 REST API를 래핑하는 AWS SDK 라이브러리를 사용하여 Amazon S3에 요청을 전송할 수 있습니다. 예를 들어 SDK는 서명 계산, 암호화 방식으로 요청 서명, 오류 관리 및 자동으로 요청 재시도와 같은 작업을 처리합니다. 다운로드 및 설치 방법을 비롯하여 AWS SDK에 대한 자세한 내용은 AWS용 도구를 참조하세요.

Amazon S3와의 모든 상호 작용은 인증을 거치거나 익명으로 할 수 있습니다. AWS SDK를 사용할 경우 라이브러리는 사용자가 제공하는 키로부터 인증을 위한 서명을 컴퓨팅합니다. Amazon S3에 요청하는 방법에 대한 자세한 내용은 요청을 참조하세요.

Amazon S3 REST API

Amazon S3 아키텍처는 AWS에서 지원하는 인터페이스를 사용하여 객체를 저장 및 검색하는 프로그래밍 언어 중립적 아키텍처로 설계되었습니다. Amazon S3 REST API를 사용하여 프로그래밍 방식으로 S3 및 AWS에 액세스할 수 있습니다. REST API는 Amazon S3에 대한 HTTP 인터페이스입니다. REST API를 통해 표준 HTTP 요청을 사용하여 버킷과 객체를 생성하고, 가져오며, 삭제할 수 있습니다.

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.

애플리케이션에서 직접 REST API를 호출할 경우 서명을 계산하는 코드를 작성하여 요청에 추가해야 합니다. Amazon S3에 요청하는 방법에 대한 자세한 내용은 Amazon S3 API 참조에서 요청을 참조하세요.

범용 버킷의 가상 호스팅

가상 호스팅은 한 웹 서버에 있는 여러 웹 사이트에 서비스를 제공하는 기능입니다. Amazon S3 REST API 요청에서 사이트를 구별하는 한 가지 방법은 URI의 경로 이름 부분 대신 요청-URI의 명백한 호스트 이름을 사용하는 것입니다. 일반적인 Amazon S3 REST 요청은 요청-URI 경로의 슬래시로 구분된 구성 요소를 사용하여 버킷을 지정합니다. 대신 Amazon S3 가상 호스팅을 통해 HTTP `Host` 헤더를 사용하여 REST API 직접 호출에서 범용 버킷을 지정할 수 있습니다. 실제로 Amazon S3은 `Host`의 의미를 `https://`*`bucket-name`*`.s3.`*`region-code`*`.amazonaws.com`에서 대부분의 버킷이 제한된 요청 유형에 대해 자동으로 액세스가 가능하다고 해석합니다. Amazon S3 리전 및 엔드포인트의 전체 목록은 Amazon Web Services 일반 참조에서 Amazon S3 엔드포인트 및 할당량을 참조하세요.

가상 호스팅에는 다른 이점도 있습니다. 또한 등록된 도메인 이름 뒤에 버킷 이름을 지정하고 해당 이름을 Amazon S3용 DNS 별칭으로 만들어 Amazon S3 리소스의 URL을 `http://my.bucket-name.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.

> **Important**
>
> SSL과 함께 가상 호스팅 스타일 범용 버킷을 사용하는 경우 SSL 와일드카드 인증서는 점(.)이 포함되지 않은 버킷과만 일치합니다. 이 제한을 해결하려면 HTTP를 사용하거나 고유한 인증서 확인 논리를 직접 작성합니다. 자세한 내용은 AWS 뉴스 블로그에 있는 Amazon S3 경로 사용 중지 계획을 참조하세요.

주제

- 경로 방식 요청
- 가상 호스팅 방식 요청
- HTTP Host 헤더 버킷 사양
- 예시
- CNAME 레코드를 사용하여 Amazon S3 URL 사용자 지정
- 호스트 이름과 Amazon S3 버킷 연결
- 제한 사항
- 이전 버전과의 호환성

경로 방식 요청

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

Amazon S3에서 경로 방식 URL은 다음 형식을 사용합니다.

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

예를 들어 미국 서부(오리건) 리전에서 이름이 amzn-s3-demo-bucket1인 버킷을 생성하고 해당 버킷의 puppy.jpg 객체에 액세스하려는 경우 다음 경로 방식 URL을 사용할 수 있습니다.

```
https://s3.us-west-2.amazonaws.com/amzn-s3-demo-bucket1/puppy.jpg
```

Important

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

Warning

웹 브라우저에서 액세스할 웹 사이트 콘텐츠를 호스팅할 때는 경로 스타일 URL을 사용하지 마세요. 브라우저의 동일 오리진 보안 모델을 방해할 수 있습니다. 웹 사이트 콘텐츠를 호스팅하려면 S3 웹 사이트 엔드포인트 또는 CloudFront 배포를 사용하는 것이 좋습니다. 자세한 내용은 웹 사이트 엔드포인트 및 AWS 퍼스펙티브 지침 패턴에서 React 기반 단일 페이지 애플리케이션을 Amazon S3 및 CloudFront에 배포를 참조하세요.

가상 호스팅 방식 요청

가상 호스팅 방식의 URI에서 버킷 이름은 URL에서 도메인 이름의 일부입니다.

Amazon S3 가상 호스팅 방식 URL은 다음 형식을 사용합니다.

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

이 예제에서 amzn-s3-demo-bucket1은 버킷 이름이고, 미국 서부(오리건)는 리전이며, puppy.png는 키 이름입니다.

```
https://amzn-s3-demo-bucket1.s3.us-west-2.amazonaws.com/puppy.png
```

HTTP **Host** 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-code*.amazonaws.com이면, 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. Host 헤더를 생략하는 것은 HTTP 1.0 요청에만 유효합니다.
- 또는 Host 헤더 값이 .s3.*region-code*.amazonaws.com으로 끝나는 경우 버킷 이름이 Host 헤더 값에서 .s3.*region-code*.amazonaws.com까지의 선행 구성 요소가 됩니다. 요청에 대한 키는 요청-URI입니다. 이 해석은 이 단원의 세 번째와 네 번째 예제에 나와 있는 것처럼 버킷을 .s3.*region-code*.amazonaws.com의 하위 도메인으로 표시합니다.
- 그렇지 않은 경우 요청에 대한 버킷은 Host 헤더의 소문자 값이며 요청에 대한 키는 요청-URI입니다. 이 해석은 버킷 이름과 동일한 DNS 이름을 등록하고 이 이름을 Amazon S3용 표준 이름(CNAME) 별칭으로 구성한 경우에 유용합니다. 도메인 이름 등록 및 CNAME DNS 레코드 구성 절차는 본 가이드에서 다루지 않지만, 이 단원의 마지막 예제에 나와 있습니다.

예시

이 단원은 예제 URL 및 요청을 제공합니다.

Example - 경로 방식 URL 및 요청

이 예에서는 다음을 사용합니다.

- 버킷 이름 - example.com
- 리전 - 미국 동부(버지니아 북부)
- 키 이름 - homepage.html

URL은 다음과 같습니다.

```
http://s3.us-east-1.amazonaws.com/example.com/homepage.html
```

요청은 다음과 같습니다.

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

HTTP 1.0을 사용한 요청 및 Host 헤더 생략은 다음과 같습니다.

```
GET /example.com/homepage.html HTTP/1.0
```

DNS를 준수하는 이름에 대한 자세한 내용은 제한 사항을 참조하십시오. 키에 대한 자세한 내용은 키를 참조하십시오.

Example - 가상 호스팅 방식 URL 및 요청

이 예에서는 다음을 사용합니다.

- 버킷 이름 - `amzn-s3-demo-bucket1`
- 리전 - 유럽(아일랜드)
- 키 이름 - `homepage.html`

URL은 다음과 같습니다.

```
http://amzn-s3-demo-bucket1.s3.eu-west-1.amazonaws.com/homepage.html
```

요청은 다음과 같습니다.

```
GET /homepage.html HTTP/1.1
Host: amzn-s3-demo-bucket1.s3.eu-west-1.amazonaws.com
```

Example — CNAME 별칭 메서드

이 메서드를 사용하려면 DNS 이름을 *`bucket-name`*`.s3.us-east-1.amazonaws.com`의 CNAME 별칭으로 구성해야 합니다. 자세한 내용은 CNAME 레코드를 사용하여 Amazon S3 URL 사용자 지정 섹션을 참조하세요.

이 예에서는 다음을 사용합니다.

- 버킷 이름 - `example.com`
- 키 이름 - `homepage.html`

URL은 다음과 같습니다.

```
http://www.example.com/homepage.html
```

예제는 다음과 같습니다.

```
GET /homepage.html HTTP/1.1
Host: www.example.com
```

CNAME 레코드를 사용하여 Amazon S3 URL 사용자 지정

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

버킷 이름은 CNAME과 동일해야 합니다. 예를 들어 images.example.com을 images.example.com.s3.us-east-1.amazonaws.com으로 매핑하기 위한 CNAME을 만드는 경우 http://images.example.com/filename 및 http://images.example.com.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.example.com이고 버킷이 미국 동부(버지니아 북부) 리전에 있는 경우, CNAME 레코드의 별칭은 images.example.com.s3.us-east-1.amazonaws.com으로 지정되어야 합니다.

```
images.example.com CNAME    images.example.com.s3.us-east-1.amazonaws.com.
```

Amazon S3은 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. http://www.example.com을 액세스하는 경우, Amazon S3은 다음과 같은 요청을 받습니다.

Example

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

Amazon S3은 원래 호스트 이름 www.example.com만 볼 수 있으며 요청을 해결하는 데 사용한 CNAME 매핑을 알지 못합니다.

모든 Amazon S3 엔드포인트를 CNAME 별칭에서 사용할 수 있습니다. 예를 들어, s3.ap-southeast-1.amazonaws.com은 CNAME 별칭에서 사용될 수 있습니다. 엔드포인트에 대한 자세

한 내용은 Amazon S3 API 참조의 요청 엔드포인트를 참조하세요. 사용자 지정 도메인을 사용하여 정적 웹 사이트를 만들려면 자습서: Route 53에 등록된 사용자 지정 도메인을 사용하여 정적 웹 사이트 구성을 참조하세요.

> **Important**
>
> CNAME과 함께 사용자 지정 URL을 사용하는 경우 구성하는 CNAME 또는 별칭 레코드에 대해 일치하는 버킷이 있는지 확인해야 합니다. 예를 들어, S3를 사용하여 웹 콘텐츠를 게시하기 위해 `www.example.com` 및 `login.example.com`에 대한 DNS 항목을 생성하는 경우 버킷 `www.example.com` 및 `login.example.com`을 모두 생성해야 합니다.
> 일치하는 버킷이 없는 S3 엔드포인트를 가리키는 CNAME 또는 별칭 레코드가 구성된 경우 소유권이 동일하지 않더라도 모든 AWS 사용자는 해당 버킷을 생성하고 구성된 별칭으로 콘텐츠를 게시할 수 있습니다.
> 같은 이유로 버킷을 삭제할 때 해당 CNAME 또는 별칭을 변경하거나 제거하는 것이 좋습니다.

호스트 이름과 Amazon S3 버킷 연결

CNAME 별칭을 사용하여 호스트 이름과 Amazon S3 버킷 연결

1. 제어하는 도메인에 속한 호스트 이름을 선택합니다.

   이 예제는 `images` 도메인의 `example.com` 하위 도메인을 사용합니다.
2. 호스트 이름과 일치하는 버킷을 만듭니다.

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

   예:

   ```
   images.example.com CNAME images.example.com.s3.us-west-2.amazonaws.com
   ```

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

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

제한 사항

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

이전 버전과의 호환성

다음 섹션에서는 경로 스타일 및 가상 호스팅 스타일 URL 요청과 관련된 Amazon S3 이전 버전과의 다양한 호환성 측면을 다룹니다.

레거시 엔드포인트

일부 리전에서 레거시 엔드포인트를 지원합니다. 서버 액세스 로그 또는 AWS CloudTrail 로그에 이러한 엔드포인트가 표시될 수 있습니다. 자세한 내용은 아래 정보를 검토하세요. Amazon S3 리전 및 엔드포인트의 전체 목록은 Amazon Web Services 일반 참조에서 Amazon S3 엔드포인트 및 할당량을 참조하세요.

Important

로그에 레거시 엔드포인트가 표시될 수도 있지만 항상 표준 엔드포인트 구문을 사용하여 버킷에 액세스하는 것이 좋습니다.
Amazon S3 가상 호스팅 방식 URL은 다음 형식을 사용합니다.

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

Amazon S3에서 경로 방식 URL은 다음 형식을 사용합니다.

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

s3-리전

일부 이전 Amazon S3 리전은 점(예: `s3.us-west-2`) 대신 `s3`와 리전 코드(예: `s3-us-west-2`) 사이에 대시(`-`)가 포함된 엔드포인트를 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-code.amazonaws.com
```

이 예제에서 버킷 이름은 amzn-s3-demo-bucket1이고 리전은 미국 서부(오리건)입니다.

```
https://amzn-s3-demo-bucket1.s3-us-west-2.amazonaws.com
```

레거시 전역 엔드포인트

일부 리전의 경우 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 amzn-s3-demo-bucket1이며 레거시 전역 엔드포인트는 다음과 같습니다.

```
https://amzn-s3-demo-bucket1.s3.amazonaws.com
```

미국 동부(버지니아 북부)에 대한 가상 호스팅 방식 요청

레거시 전역 엔드포인트를 사용한 요청은 기본적으로 미국 동부(버지니아 북부) 리전으로 이동합니다. 따라서 레거시 전역 엔드포인트는 미국 동부(버지니아 북부)에 대한 리전 엔드포인트 대신 사용되기도 합니다. 미국 동부(버지니아 북부)에 버킷을 생성하고 전역 엔드포인트를 사용하는 경우 Amazon S3은 기본적으로 요청을 이 리전으로 라우팅합니다.

다른 리전에 대한 가상 호스팅 방식 요청

또한 레거시 전역 엔드포인트는 지원되는 다른 리전의 가상화 호스팅 방식 요청에도 사용됩니다. 2019년 3월 20일 이전에 시작된 리전에 버킷을 생성하고 레거시 전역 엔드포인트를 사용하는 경우 Amazon S3은 DNS 레코드를 업데이트하여 요청을 올바른 위치로 다시 라우팅하며, 이로 인해 시간이 걸릴 수 있습니다. 그 동안 기본 규칙이 적용되며 가상 호스팅 방식의 요청은 미국 동부(버지니아 북부) 리전으로 이동합니다. 그런 다음 Amazon S3이 HTTP 307 임시 리디렉션을 사용하여 올바른 리전으로 리디렉션합니다.

2019년 3월 20일 이후에 시작된 리전의 S3 버킷인 경우 DNS 서버는 버킷이 상주하는 AWS 리전으로 요청을 직접 라우팅하지 않습니다. 대신, HTTP 400 Bad Request 오류를 반환합니다. 자세한 내용은 Amazon S3 API 참조에서 요청을 참조하세요.

경로 방식 요청

미국 동부(버지니아 북부) 리전에서는 경로 스타일 요청에 레거시 전역 엔드포인트를 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 301 영구 리디렉션 오류와 리소스에 대한 올바른 URI를 나타내는 메시지가 표시됩니다. 예를 들어 미국 서부(오리건) 리전에 생성된 버킷에 `https://s3.amazonaws.com/bucket-name`을 사용하는 경우 HTTP 301 영구 리디렉션 오류가 표시됩니다.

범용 버킷 생성

Amazon S3에 데이터를 업로드하려면 먼저 AWS 리전 중 하나에서 Amazon S3 범용 버킷을 생성해야 합니다. 버킷을 생성하는 AWS 계정이 해당 버킷을 소유합니다. 버킷을 생성할 때 버킷 이름과 리전을 선택해야 합니다. 버킷을 만드는 프로세스 중에 버킷에 대해 다른 스토리지 관리 옵션을 선택할 수도 있습니다.

> **Important**
>
> 버킷을 만든 후에는 버킷 이름, 버킷 소유자 또는 리전을 변경할 수 없습니다. 버킷 이름 지정에 대한 자세한 내용은 the section called “이름 지정 규칙” 단원을 참조하세요.

기본적으로 AWS 계정당 최대 범용 버킷 10,000개를 만들 수 있습니다. 범용 버킷의 할당량 증가를 요청하려면 Service Quotas 콘솔을 방문하세요.

버킷에 저장할 수 있는 객체 수에는 제한이 없습니다. Amazon S3 범용 버킷과 관련된 규제 및 제한 목록은 범용 버킷 할당량, 제한 및 규제 섹션을 참조하세요.

범용 버킷 설정

범용 버킷을 생성할 때 다음 설정을 사용하여 버킷 동작의 다양한 측면을 제어할 수 있습니다.

- S3 객체 소유권 – S3 객체 소유권은 버킷에 업로드되는 객체의 소유권을 제어하고 액세스 제어 목록(ACL)을 비활성화 또는 활성화하는 데 사용할 수 있는 Amazon S3 버킷 수준 설정입니다. 기본적으로 객체 소유권은 버킷 소유자 적용 설정으로 설정되며 모든 ACL이 비활성화되어 있습니다. ACL이 비활성화되면 버킷 소유자는 버킷의 모든 객체를 소유하고 정책을 사용하여 데이터에 대한 액세스

를 독점적으로 관리합니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

- S3 Object Lock – S3 Object Lock을 사용하면 고정된 시간 동안 또는 무기한으로 Amazon S3 객체의 삭제 또는 덮어쓰기를 방지하는 데 도움이 됩니다. 객체 잠금은 WORM(Write Once Read Many) 모델을 사용하여 객체를 저장합니다. 객체 잠금을 사용하면 WORM 스토리지가 필요한 규제 요구 사항을 충족하거나 객체 변경 또는 삭제에 대한 보호 계층을 추가하는 데 도움이 됩니다. 자세한 내용은 the section called “객체 잠금” 섹션을 참조하세요.

범용 버킷을 생성한 후 또는 Amazon S3 콘솔을 사용하여 범용 버킷을 생성할 때 다음 설정을 사용하여 버킷 동작의 다른 측면을 제어할 수도 있습니다.

- S3 Block Public Access – Amazon S3 Block Public Access 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. 자세한 내용은 the section called “퍼블릭 액세스 차단” 섹션을 참조하세요.
- S3 버전 관리 – 버전 관리는 동일 버킷 내에 여러 개의 객체 변형을 보유하는 것을 의미합니다. 버전 관리를 사용하여 버킷에 저장된 모든 버전의 객체를 모두 보존, 검색 및 복원할 수 있습니다. 또한 의도치 않은 사용자 작업 및 애플리케이션 장애로부터 쉽게 복구할 수 있습니다. 버킷의 경우 버전 관리는 기본으로 비활성화됩니다. 자세한 내용은 the section called “여러 버전의 객체 유지” 섹션을 참조하세요.
- 기본 암호화 – 버킷의 모든 객체에 대해 기본 암호화 유형을 설정할 수 있습니다. Amazon S3 관리형 키(SSE-S3)를 사용한 서버 측 암호화가 Amazon S3 내 모든 버킷의 기본 수준 암호화 구성입니다. S3 버킷에 업로드된 모든 새 객체는 SSE-S3를 기본 암호화로 사용하여 자동으로 암호화됩니다. 다른 유형의 기본 암호화를 사용하려는 경우, AWS Key Management Service(AWS KMS) 키를 사용한 서버 측 암호화(SSE-KMS), AWS KMS 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS) 또는 고객 제공 키를 사용한 서버 측 암호화(SSE-C)를 지정하여 데이터를 암호화할 수 있습니다. 자세한 내용은 Amazon S3 버킷에 대한 기본 서버 측 암호화 동작 설정 섹션을 참조하세요.

Amazon S3 콘솔, Amazon S3 REST API, AWS Command Line Interface(AWS CLI) 또는 AWS SDK를 사용하여 범용 버킷을 생성할 수 있습니다. 범용 버킷을 생성하는 데 필요한 권한에 대한 자세한 내용은 Amazon Simple Storage Service API Reference의 CreateBucket을 참조하세요.

Amazon S3 버킷을 만드는 데 문제가 있는 경우, AWS re:Post의 Amazon S3 버킷을 생성하려고 할 때 발생하는 문제를 해결하려면 어떻게 해야 합니까?를 참조하세요.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 버킷을 생성하려는 리전을 선택합니다.

   > **Note**
   >
   > - 버킷을 만든 후에는 리전을 변경할 수 없습니다.
   > - 지연 시간과 요금을 최소화하고 규제 요건을 충족하려면 가장 가까운 리전을 선택하십시오. 특정 리전에 저장된 객체는 사용자가 명시적으로 객체를 다른 리전으로 전송하지 않는 한 해당 리전을 벗어나지 않습니다. Amazon S3 AWS 리전 목록은 Amazon Web Services 일반 참조의 AWS 서비스 엔드포인트를 참조하십시오.

3. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
4. 버킷 만들기를 선택합니다. 버킷 만들기 페이지가 열립니다.
5. [Bucket Name]에서 버킷 이름을 입력합니다.

   버킷 이름은 다음과 같아야 합니다.

   - 파티션 내에서 고유해야 합니다. 파티션은 리전의 그룹입니다. AWS에는 현재 `aws`(상용 리전), `aws-cn`(중국 리전) 및 `aws-us-gov`(AWS GovCloud (US) Regions), 이렇게 세 가지 파티션이 있습니다.
   - 3~63자 이내여야 합니다.
   - 소문자, 숫자, 마침표(`.`) 및 하이픈(`-`)만 포함해야 합니다. 최상의 호환성을 위해 정적 웹 사이트 호스팅에만 사용되는 버킷을 제외하고 버킷 이름에 마침표(`.`)를 사용하지 않는 것이 좋습니다.
   - 문자나 숫자로 시작하고 끝나야 합니다.
   - 버킷 이름 지정 규칙의 전체 목록은 범용 버킷 이름 지정 규칙 섹션을 참조하세요.

   > **Important**
   >
   > - 버킷을 생성한 후에는 해당 이름을 변경할 수 없습니다.

- 버킷 이름에 민감한 정보가 포함되어서는 안 됩니다. 버킷 이름은 버킷의 객체를 가리키는 URL에 표시됩니다.

6. (선택 사항) 일반 구성 아래에서 기존 버킷의 설정을 새 버킷에 복사하도록 선택할 수 있습니다. 기존 버킷의 설정을 복사하지 않으려면 다음 단계로 건너뛰세요.

> **Note**
>
> 이 옵션:
>
> - AWS CLI에서는 사용할 수 없으며 Amazon S3 콘솔에서만 사용할 수 있습니다.
> - 기존 버킷의 버킷 정책을 새 버킷으로 복사하지 않습니다.

기존 버킷의 설정을 복사하려면 기존 버킷에서 설정 복사 아래에서 버킷 선택을 선택합니다. 버킷 선택 창이 열립니다. 복사하려는 설정이 있는 버킷을 찾아 버킷 선택을 선택합니다. 버킷 선택 창이 닫히고 버킷 생성 창이 다시 열립니다.

이제 기존 버킷에서 설정 복사 아래에 선택한 버킷의 이름이 표시됩니다. 이제 새 버킷의 설정이 선택한 버킷의 설정과 일치합니다. 복사된 설정을 제거하려면 기본값 복원을 선택합니다. 버킷 생성 페이지에서 나머지 버킷 설정을 검토하세요. 변경하지 않으려면 마지막 단계로 건너뛸 수 있습니다.

7. 객체 소유권(Object Ownership)에서 ACL을 사용 중지 또는 사용 설정하고 버킷에 업로드된 객체의 소유권을 제어하려면 다음 설정 중 하나를 선택합니다.

ACL 사용 중지됨

- 버킷 소유자 적용(기본값) – ACL이 비활성화되고 버킷 소유자는 범용 버킷의 모든 객체를 자동으로 소유하고 완전히 제어합니다. ACL은 더 이상 S3 범용 버킷의 데이터에 대한 권한에 영향을 주지 않습니다. 버킷은 정책을 독점적으로 사용하여 액세스 제어를 정의합니다.

  기본적으로 ACL은 비활성화되어 있습니다. Amazon S3의 최신 사용 사례 대부분은 더 이상 ACL을 사용할 필요가 없습니다. 각 객체에 대해 액세스를 개별적으로 제어해야 하는 드문 상황을 제외하고는 ACL을 비활성화한 채로 두는 것이 좋습니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

ACL 사용됨

- 버킷 소유자 기본(Bucket owner preferred) – 버킷 소유자가 `bucket-owner-full-control` 미리 제공 ACL을 사용하여 다른 계정이 버킷에 작성하는 새 객체를 소유하고 완전히 제어합니다.

  버킷 소유자 선호 설정을 적용하는 경우 모든 Amazon S3 업로드에 `bucket-owner-full-control` 미리 제공된 ACL을 포함하도록 요구하려면 이 ACL을 사용하는 객체 업로드만 허용하는 버킷 정책을 추가할 수 있습니다.
- 객체 작성자(Object writer) – 객체를 업로드하는 AWS 계정은 객체를 소유하고 완전히 제어하며 ACL을 통해 다른 사용자에게 이에 대한 액세스 권한을 부여할 수 있습니다.

> **Note**
>
> 기본 설정은 버킷 소유자 적용입니다. 기본 설정을 적용하고 ACL을 비활성화된 상태로 유지하려면 `s3:CreateBucket` 권한만 있으면 됩니다. ACL을 활성화하려면 `s3:PutBucketOwnershipControls` 권한이 있어야 합니다.

8. 퍼블릭 액세스 차단을 위한 버킷 설정에서 버킷에 적용할 퍼블릭 액세스 차단 설정을 선택합니다.

   기본적으로, 네 개의 퍼블릭 액세스 차단 설정이 모두 활성화되어 있습니다. 특정 사용 사례에 대해 하나 이상의 설정을 해제해야 하는 경우가 아니라면 모든 설정을 활성화된 상태로 유지하는 것이 좋습니다. 퍼블릭 액세스 차단에 대한 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하십시오.

   > **Note**
   >
   > 모든 퍼블릭 액세스 차단 설정을 활성화하려면 `s3:CreateBucket` 권한만 있으면 됩니다. 퍼블릭 액세스 차단 설정을 해제하려면 `s3:PutBucketPublicAccessBlock` 권한이 있어야 합니다.

9. (선택 사항) 버킷 버전 관리는 기본으로 비활성화됩니다. 버전 관리는 동일 버킷 내에 여러 개의 객체 변형을 보유하는 것을 의미합니다. 버전 관리를 사용하여 버킷에 저장된 모든 버전의 객체를 모두 보존, 검색 및 복원할 수 있습니다. 또한, 의도치 않은 사용자 작업 및 애플리케이션 장애가 발생해도 쉽게 복구할 수 있습니다. 버전 관리에 대한 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하십시오.

버킷에서 버전 관리를 활성화하려면 활성화를 선택합니다.

10. (선택 사항) Tags(태그)에서 버킷에 태그를 추가할 수 있습니다. AWS 비용 할당을 하면서 버킷 태그를 지정해 버킷 사용에 대한 요금 청구 주석을 달 수 있습니다. 태그는 버킷에 할당된 라벨을 나타내는 한 쌍의 키-값입니다. 자세한 내용은 the section called “비용 할당 태그 사용” 섹션을 참조하세요.

    버킷 태그를 추가하려면 Key(키) 및 원하는 경우 Value(값)를 입력하고 Add Tag(태그 추가)를 선택합니다.

11. 기본 암호화를 구성하려면 암호화 유형 아래에서 다음 중 하나를 선택합니다.

    - Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)
    - AWS Key Management Service 키를 사용한 서버 측 암호화(SSE-KMS)
    - AWS Key Management Service(AWS KMS) 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)

    > **Important**
    >
    > 기본 암호화 구성으로 SSE-KMS 또는 DSSE-KMS 옵션을 사용할 경우 AWS KMS의 초당 요청 수(RPS) 할당량이 적용됩니다. AWS KMS 할당량과 할당량 증대를 요청하는 방법에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 할당량을 참조하십시오.

    버킷과 새 객체는 Amazon S3 관리형 키(SSE-S3)를 암호화 구성의 기본 수준으로 사용하는 서버 측 암호화로 암호화됩니다. 기본 암호화에 대한 자세한 내용은 Amazon S3 버킷에 대한 기본 서버 측 암호화 동작 설정 섹션을 참조하십시오. SSE-S3에 대한 자세한 내용은 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) 사용 섹션을 참조하세요.

    서버 측 암호화를 사용하는 데이터 암호화에 대한 자세한 내용은 the section called “데이터 암호화” 섹션을 참조하세요.

12. Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) 또는 AWS Key Management Service(AWS KMS) 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)를 선택한 경우 다음을 수행하세요.

    a. AWS KMS 키에서 다음 방법 중 하나로 KMS 키를 지정합니다.

- 사용 가능한 KMS 키 목록에서 AWS KMS keys 중에서 선택을 선택하고 사용 가능한 키 목록에서 KMS 키를 선택합니다.

  AWS 관리형 키(aws/s3)와 고객 관리형 키가 모두 목록에 표시됩니다. 고객 관리형 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 키 및 AWS 키를 참조하십시오.
- KMS 키 ARN을 입력하려면 AWS KMS key ARN 입력을 선택하고 나타나는 필드에 KMS 키 ARN을 입력합니다.
- AWS KMS 콘솔에서 고객 관리형 키를 생성하려면 KMS 키 생성을 선택합니다.

  AWS KMS key 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 키 생성을 참조하십시오.

> **Important**
>
> 버킷과 동일한 AWS 리전에서 사용할 수 있는 KMS 키만 사용 가능합니다. Amazon S3 콘솔은 버킷과 동일한 리전에 있는 처음 100개의 KMS 키만 나열합니다. 목록에 없는 KMS 키를 사용하려면 KMS 키 ARN을 입력해야 합니다. 다른 계정에서 소유한 KMS 키를 사용하려면 먼저 해당 키에 대한 사용 권한이 있어야 하고, 다음 단계로 KMS 키 ARN을 입력해야 합니다. KMS 키의 크로스 계정 권한에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 Creating KMS keys that other accounts can use를 참조하세요. SSE-KMS에 대한 자세한 내용은 AWS KMS(SSE-KMS)를 사용한 서버 측 암호화 지정 단원을 참조하십시오. DSSE-KMS에 대한 자세한 내용은 the section called “이중 계층 서버 측 암호화(DSSE-KMS)” 섹션을 참조하세요.
>
> Amazon S3에서 서버 측 암호화에 AWS KMS key을 사용하는 경우 대칭 암호화 KMS 키를 선택해야 합니다. Amazon S3는 대칭 암호화 KMS 키만 지원하며 비대칭 KMS 키는 지원하지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 대칭 및 비대칭 KMS 키 식별을 참조하십시오.

b. SSE-KMS와 함께 기본 암호화를 사용하도록 버킷을 구성할 때 S3 버킷 키를 사용할 수도 있습니다. S3 버킷 키를 사용하면 Amazon S3에서 AWS KMS로의 요청 트래픽이 줄어 암호화 비용이 절감됩니다. 자세한 내용은 Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감 섹션을 참조하세요. S3 버킷 키는 DSSE-KMS에서 지원되지 않습니다.

기본적으로 S3 버킷 키는 Amazon S3 콘솔에서 활성화됩니다. 비용을 낮추려면 S3 버킷 키를 활성화된 상태로 두는 것이 좋습니다. 버킷에 대해 S3 버킷 키를 비활성화하려면 버킷 키 아래에서 비활성화를 선택합니다.

13. (선택 사항) S3 Object Lock을 사용하면 새 객체가 삭제되거나 덮어쓰이지 않도록 보호할 수 있습니다. 자세한 내용은 Object Lock으로 객체 잠금 섹션을 참조하세요. S3 Object Lock을 활성화하려면 다음을 수행합니다.

    a. 고급 설정(Advanced Settings)을 선택합니다.

    > **Important**
    >
    > Object Lock을 활성화하면 버킷의 버전 관리도 활성화됩니다. 버킷을 활성화하고 성공적으로 만든 후에는 버킷의 속성 탭에서 Object Lock 기본 보존 및 법적 보존 설정도 구성해야 합니다.

    b. 객체 잠금을 활성화하려면 Enable(활성화)을 선택하고 표시되는 경고를 읽고 확인합니다.

    > **Note**
    >
    > Object Lock을 사용하는 버킷을 만들려면 `s3:CreateBucket`, `s3:PutBucketVersioning` 및 `s3:PutBucketObjectLockConfiguration` 권한이 있어야 합니다.

14. 버킷 만들기를 선택합니다.

AWS SDK 사용

AWS SDK를 사용하여 범용 버킷을 생성하는 경우, 클라이언트를 생성한 다음 이 클라이언트를 사용하여 버킷 생성 요청을 보냅니다. 모범 사례로서, 동일한 AWS 리전에 클라이언트 및 버킷을 만들어야 합니다. 클라이언트나 버킷을 생성할 때 리전을 지정하지 않으면 Amazon S3는 기본 리전인 미국 동부(버지니아 북부)를 사용합니다. 버킷 생성을 특정 AWS 리전 리전으로 제한하려면 LocationConstraint 조건 키를 사용하세요.

클라이언트를 생성하여 듀얼 스택 엔드포인트에 액세스하려면 AWS 리전을 지정해야 합니다. 자세한 내용은 Amazon S3 API 참조의 Amazon S3 듀얼 스택 엔드포인트 사용을 참조하세요. 사용 가능한 AWS 리전의 목록은 AWS 일반 참조의 Amazon Simple Storage Service endpoints and quotas를 참조하세요.

클라이언트를 만들면 리전은 리전별 엔드포인트로 매핑됩니다. 클라이언트는 `s3.region.amazonaws.com` 엔드포인트를 사용하여 Amazon S3와 통신합니다. 2019년 3월 20일 이후에 리전이 시작된 경우 클라이언트와 버킷이 동일한 리전에 있어야 합니다. 다만 미국 동부(버지니아 북부) 리전의 클라이언트를 사용하면 2019년 3월 20일 이전에 시작된 모든 리전에 버킷을 생성할 수 있습니다. 자세한 내용은 레거시 엔드포인트 섹션을 참조하세요.

이 AWS SDK 코드 예제는 다음 태스크를 수행합니다.

- AWS 리전을 명시적으로 지정하여 클라이언트 생성 - 이 예에서 클라이언트는 `s3.us-west-2.amazonaws.com` 엔드포인트를 사용하여 Amazon S3와 통신합니다. AWS 리전을 지정할 수 있습니다. AWS 리전 목록은 AWS 일반 참조의 리전 및 엔드포인트를 참조하세요.
- 버킷 이름만 지정하여 버킷 생성 요청 전송 - 클라이언트는 Amazon S3로 요청을 전송하여 클라이언트가 생성된 리전에 버킷을 생성합니다.
- 버킷의 위치에 대한 정보 검색 - Amazon S3는 버킷과 연결된 location 하위 리소스에 버킷 위치 정보를 저장합니다.

추가 AWS SDK 예제 및 다른 언어의 예제는 Amazon Simple Storage Service API 참조의 Use CreateBucket with an AWS SDK or CLI를 참조하세요.

Java

Example 버킷 이름에 전역 고유 식별자(GUID)를 사용하는 버킷 생성

다음 예시에서는 AWS SDK for Java를 사용하여 미국 동부(버지니아 북부) 리전(us-east-1)의 버킷 이름 끝에 GUID가 있는 버킷을 생성하는 방법을 설명합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다. 다른 AWS SDK에 대한 자세한 내용은 AWS 기반의 도구를 참조하세요.

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.Bucket;
import com.amazonaws.services.s3.model.CreateBucketRequest;

import java.util.List;
import java.util.UUID;

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

```
        final AmazonS3 s3 =
 AmazonS3ClientBuilder.standard().withRegion(Regions.US_EAST_1).build();
        String bucketName = "amzn-s3-demo-bucket" +
 UUID.randomUUID().toString().replace("-", "");
        CreateBucketRequest createRequest = new CreateBucketRequest(bucketName);
        System.out.println(bucketName);
        s3.createBucket(createRequest);
    }
}
```

Example 범용 버킷 생성

이 예제에서는 AWS SDK for Java를 사용하여 Amazon S3 버킷을 만드는 방법을 설명합니다. 실제 샘플을 만들고 테스트하는 방법에 대한 지침은 AWS SDK for Java 2.x 개발자 안내서를 참조하세요.

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

import java.io.IOException;

public class CreateBucket2 {

    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();
        }
    }
}
```

.NET

실제 샘플을 만들고 테스트하는 방법에 대한 자세한 내용은 AWS SDK for .NET 버전 3 API 참조를 참조하세요.

Example

```
using Amazon;
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

실제 샘플을 만들고 테스트하는 방법에 대한 자세한 내용은 AWS SDK for Ruby - 버전 3을 참조하세요.

Example

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
 This is a client-side object until
  #                                 create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
```

```
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      'None. You must create a bucket before you can get its location!'
    else
      @bucket.client.get_bucket_location(bucket: @bucket.name).location_constraint
    end
  rescue Aws::Errors::ServiceError => e
    "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("amzn-s3-demo-bucket-
#{Random.uuid}"))
  return unless wrapper.create?(region)

  puts "Created bucket #{wrapper.bucket.name}."
  puts "Your bucket's region is: #{wrapper.location}"
end

run_demo if $PROGRAM_NAME == __FILE__
```

AWS CLI 사용

다음 AWS CLI 예시에서는 전역 고유 식별자(GUID)를 사용하는 버킷 이름의 예시를 사용하여 미국 서부(캘리포니아 북부) 리전(us-west-1)에 범용 버킷을 생성합니다. 이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3api create-bucket \
    --bucket amzn-s3-demo-bucket1$(uuidgen | tr -d - | tr '[:upper:]' '[:lower:]' ) \
    --region us-west-1 \
    --create-bucket-configuration LocationConstraint=us-west-1
```

자세한 내용과 추가 예제는 AWS CLI 명령 참조의 create-bucket 섹션을 참조하세요.

S3 범용 버킷에 대한 속성 보기

소유한 Amazon S3 버킷의 속성을 볼 수 있습니다. 이러한 설정에는 다음이 포함됩니다.

- 버킷 버전 관리 – 버전 관리를 사용하여 하나의 범용 버킷에 여러 버전의 객체를 보관합니다. 새 버킷의 경우 버전 관리가 기본으로 비활성화됩니다. 버전 관리 사용에 대한 자세한 내용은 버킷에 버전 관리 사용 설정 단원을 참조하십시오.
- 태그 – AWS 비용 할당을 하면서 버킷 태그를 지정해 범용 버킷 사용에 대한 요금 청구 주석을 달 수 있습니다. 태그는 버킷에 할당된 라벨을 나타내는 한 쌍의 키-값입니다. 자세한 내용은 비용 할당 S3 버킷 태그 사용 섹션을 참조하세요.
- 기본 암호화 – 기본 암호화를 활성화하면 서버 측 자동 암호화가 제공됩니다. Amazon S3에서는 객체를 디스크에 저장하기 전에 암호화하고 객체를 다운로드할 때 이를 해독합니다. 자세한 내용은 Amazon S3 버킷에 대한 기본 서버 측 암호화 동작 설정 섹션을 참조하세요.
- 서버 액세스 로깅 – 서버 액세스 로깅을 사용하면 범용 버킷에 대해 이루어진 요청에 따른 상세 레코드를 가져올 수 있습니다. Amazon S3는 기본적으로 서버 액세스 로그를 수집하지 않습니다. 서버 액세스 로깅에 대한 자세한 내용은 Amazon S3 서버 액세스 로깅 사용 설정 단원을 참조하십시오.
- AWS CloudTrail 데이터 이벤트 – CloudTrail을 사용하여 데이터 이벤트를 기록합니다. 기본적으로 추적은 데이터 이벤트를 로깅하지 않습니다. 데이터 이벤트에는 추가 요금이 적용됩니다. 자세한 내용은 AWS CloudTrail 사용 설명서의 추적을 위해 데이터 이벤트 로깅을 참조하세요.
- 이벤트 알림 – 특정 Amazon S3 범용 버킷 이벤트를 활성화해 이벤트가 발생할 때마다 대상에 알림을 보낼 수 있습니다. 자세한 내용은 Amazon S3 콘솔을 사용하여 이벤트 알림 사용 설정 및 구성 섹션을 참조하세요.
- 전송 속도 향상 – 클라이언트와 S3 버킷 사이에서 파일을 빠르고 쉽고 안전하게 장거리 전송할 수 있습니다. 전송 속도 향상 활성화에 대한 자세한 내용은 S3 Transfer Acceleration 사용 설정 및 사용 단원을 참조하십시오.
- 객체 잠금 – S3 객체 잠금을 사용하면 일정한 시간 동안 또는 무기한으로 객체를 삭제하거나 덮어쓰지 않도록 할 수 있습니다. 자세한 내용은 Object Lock으로 객체 잠금 섹션을 참조하세요.
- 요청자 지불 – 범용 버킷 소유자가 아닌 요청자가 요청 및 데이터 전송에 대해 비용을 지불하도록 하려면 요청자 지불을 활성화하세요. 자세한 내용은 스토리지 전송 및 사용량에 대한 요청자 지불 범용 버킷 사용 섹션을 참조하세요.
- 정적 웹 사이트 호스팅 – Amazon S3에 정적 웹 사이트를 호스팅할 수 있습니다. 자세한 내용은 Amazon S3를 사용하여 정적 웹 사이트 호스팅 섹션을 참조하세요.

AWS Management Console, AWS CLI, 또는 AWS SDK를 사용하여 버킷 속성을 볼 수 있습니다.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.

2. 왼쪽 탐색 창에서 범용 버킷 또는 디렉터리 버킷을 선택합니다.
3. 버킷 목록에서 속성을 볼 해당 버킷의 이름을 선택합니다.
4. 속성(Properties) 탭을 선택합니다.
5. 속성 페이지에서 위의 버킷 속성을 구성할 수 있습니다.

AWS CLI 사용

AWS CLI를 사용하여 버킷 속성 보기

다음 명령은 AWS CLI를 사용하여 다양한 범용 버킷 속성을 나열하는 방법을 보여줍니다.

amzn-s3-demo-bucket1 버킷과 연결된 태그 세트를 반환합니다. 버킷 태그에 대한 자세한 내용은 비용 할당 S3 버킷 태그 사용 섹션을 참조하세요.

```
aws s3api get-bucket-tagging --bucket amzn-s3-demo-bucket1
```

자세한 내용과 예제는 AWS CLI 명령 참조에서 get-bucket-tagging를 참조하십시오.

다음은 *amzn-s3-demo-bucket1* 버킷의 버전 관리 상태를 반환합니다. 버킷 버전 관리에 대한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하세요.

```
aws s3api get-bucket-versioning --bucket amzn-s3-demo-bucket1
```

자세한 내용과 예제는 AWS CLI 명령 참조에서 get-bucket-versioning를 참조하십시오.

다음은 *amzn-s3-demo-bucket1* 버킷의 기본 암호화 구성을 반환합니다. 기본적으로 모든 버킷은 Amazon S3 관리형 키를 통한 서버 측 암호화(SSE-S3)를 사용하는 기본 암호화 구성을 가지고 있습니다. 버킷 기본 암호화에 대한 자세한 내용은 Amazon S3 버킷에 대한 기본 서버 측 암호화 동작 설정 섹션을 참조하세요.

```
aws s3api get-bucket-encryption --bucket amzn-s3-demo-bucket1
```

자세한 내용과 예제는 AWS CLI 명령 참조에서 get-bucket-encryption를 참조하십시오.

다음은 *amzn-s3-demo-bucket1* 버킷의 알림 구성을 반환합니다. 버킷 이벤트 알림에 대한 자세한 내용은 Amazon S3 이벤트 알림 섹션을 참조하세요.

```
aws s3api get-bucket-notification-configuration --bucket amzn-s3-demo-bucket1
```

자세한 내용과 예제는 AWS CLI 명령 참조에서 get-bucket-notification-configuration를 참조하십시오.

다음은 *amzn-s3-demo-bucket1* 버킷의 로깅 상태를 반환합니다. 버킷 로깅에 대한 자세한 내용은 서버 액세스 로깅을 사용한 요청 로깅 섹션을 참조하세요.

```
aws s3api get-bucket-logging --bucket amzn-s3-demo-bucket1
```

자세한 내용과 예제는 AWS CLI 명령 참조에서 get-bucket-logging를 참조하세요.

AWS SDK 사용

AWS SDK로 범용 버킷 속성(예: 버전 관리, 태그 등)을 반환하는 방법에 대한 예제는 Amazon S3 API 참조의 코드 예제를 참조하세요.

다양한 AWS SDK 사용에 대한 일반적인 정보는 Amazon S3 API 참조의 Developing with Amazon S3 using the AWS SDKs를 참조하세요.

Amazon S3 범용 버킷 나열

ListBuckets를 사용하여 직접 소유한 범용 버킷 목록을 반환할 수 있습니다. Amazon S3 콘솔, AWS Command Line Interface 또는 AWS SDK를 사용하여 버킷을 나열할 수 있습니다. 버킷(버킷 10,000개)에 대한 기본 서비스 할당량을 사용하는 AWS CLI, AWS SDK, Amazon S3 REST API, AWS 계정를 통한 `ListBuckets` 요청의 경우 페이지 매김 요청 및 페이지가 지정되지 않은 요청을 모두 지원합니다. 계정에 포함된 버킷 수와 상관없이 버킷 1개~10,000개 사이에서 페이지 크기를 만들어 모든 버킷을 나열할 수 있습니다. 페이지 매김 요청의 경우 `ListBuckets` 요청은 버킷 이름과 각 버킷에 해당하는 AWS 리전을 모두 반환합니다. 다음 AWS Command Line Interface 및 AWS SDK 예제에서는 `ListBuckets` 요청에 페이지 매김을 사용하는 방법을 설명합니다. 일부 AWS SDK는 페이지 매김을 지원한다는 점에 유의하세요.

권한

모든 범용 버킷을 나열하려면 `s3:ListAllMyBuckets` 권한이 있어야 합니다. `HTTP Access Denied (403 Forbidden)` 오류가 발생할 경우 Amazon S3의 액세스 거부(403 Forbidden) 오류 문제 해결 섹션을 참조하세요.

> **Important**
>
> 페이지 매김된 `ListBuckets` 요청만 사용하는 것이 좋습니다. 페이지가 지정되지 않은 `ListBuckets` 요청은 기본 범용 버킷 할당량인 10,000개로 설정된 AWS 계정에 대해서만 지

원됩니다. 승인된 범용 버킷 할당량이 10,000개를 초과할 경우 페이지 매김된 `ListBuckets` 요청을 전송하여 계정 버킷을 나열해야 합니다. 범용 버킷 할당량이 10,000개가 넘는 AWS 계정에 대한 페이지가 지정되지 않은 모든 `ListBuckets` 요청은 거부됩니다.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 범용 버킷 탭에서 범용 버킷 목록을 볼 수 있습니다.
4. 이름으로 버킷을 찾으려면 이름으로 버킷 찾기 필드에 버킷 이름을 입력합니다.

AWS CLI 사용

`ls` 또는 `list-buckets` 명령을 사용하면 AWS CLI를 사용하여 범용 버킷 목록을 생성할 수 있습니다. 다음 예제에서는 페이지 매김된 `list-buckets` 요청과 페이지가 지정되지 않은 `ls` 요청을 생성하는 방법을 보여줍니다. 해당 예제를 사용하려면 *`### ## ## ###`*를 바꾸세요.

Example - **`ls`**를 사용하여 계정의 모든 버킷 나열(페이지가 지정되지 않음)

다음 예제 명령은 페이지가 지정되지 않은 단일 직접 호출에서 계정의 모든 범용 버킷을 나열합니다. 이 직접 호출은 계정의 모든 버킷 목록을 반환합니다(최대 결과 10,000개).

```
$ aws s3 ls
```

자세한 내용 및 예제는 버킷 및 객체 목록을 참조하십시오.

Example - **`ls`**를 사용하여 계정의 모든 버킷 나열(페이지 매김됨)

다음 예제 명령은 페이지 매김된 직접 호출을 하나 이상 수행하여 계정의 모든 범용 버킷을 나열하고 페이지당 버킷 100개를 반환합니다.

```
$ aws s3 ls --page-size 100
```

자세한 내용 및 예제는 버킷 및 객체 목록을 참조하십시오.

Example - 계정의 모든 버킷 나열(페이지 매김됨)

다음 예제에서는 계정의 모든 범용 버킷을 나열하는 페이지 매김된 `list-buckets` 명령을 제공합니다. `--max-items` 및 `--page-size` 옵션은 나열된 버킷 수를 페이지당 100개로 제한합니다.

```
$ aws s3api list-buckets /
    --max-items 100 /
    --page-size 100
```

출력 항목 수(`--max-items`)가 기본 API 직접 호출에서 반환하는 전체 항목 수보다 적을 경우 출력에 `starting-token` 인수로 지정된 연속 토큰이 포함되며, 이를 후속 명령에 전달하여 다음 항목 세트를 검색할 수 있습니다. 다음 예제에서는 이전 예제에서 반환한 `starting-token` 값을 사용하는 방법을 설명합니다. `starting-code`를 지정하여 다음 버킷 100개를 검색할 수 있습니다.

```
$ aws s3api list-buckets /
    --max-items 100 /
    --page-size 100 /
    --starting-token eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ==
```

Example – AWS 리전의 모든 버킷 나열(페이지 매김됨)

다음 예제 명령은 `--bucket-region` 파라미터를 사용하여 `us-east-2` 리전에 있는 계정의 버킷을 최대 100개까지 나열합니다. `--bucket-region` 파라미터에 지정된 값과 다른 리전 엔드포인트에 대한 요청은 지원되지 않습니다. 예를 들어 `us-east-2`의 버킷에 대한 응답을 제한하려는 경우 `us-east-2`의 엔드포인트에 요청해야 합니다.

```
$ aws s3api list-buckets /
    --region us-east-2 /
    --max-items 100 /
    --page-size 100 /
    --bucket-region us-east-2
```

Example - 특정 버킷 이름 접두사로 시작하는 모든 버킷 나열(페이지 매김됨)

다음 예제 명령은 이름이 *amzn-s3-demo-bucket* 접두사로 시작하는 버킷을 최대 100개까지 나열합니다.

```
$ aws s3api list-buckets /
```

```
    --max-items 100 /
    --page-size 100 /
    --prefix amzn-s3-demo-bucket
```

AWS SDK 사용

다음 예제에서는 AWS SDK를 사용하여 범용 버킷을 나열하는 방법을 설명합니다.

SDK for Python

Example – ListBuckets 요청(페이지 매김됨)

```
import boto3

s3 = boto3.client('s3')
response = s3.list_buckets(MaxBuckets=100)
```

Example – ListBuckets 응답(페이지 매김됨)

```
import boto3

s3 = boto3.client('s3')
response =
 s3.list_buckets(MaxBuckets=1,ContinuationToken="eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZVS
```

SDK for Java

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.Bucket;
import com.amazonaws.services.s3.model.ListBucketsPaginatedRequest;
import com.amazonaws.services.s3.model.ListBucketsPaginatedResult;

import java.util.List;

public class ListBuckets {
    public static void main(String[] args) {
        final AmazonS3 s3 =
 AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
        ListBucketsPaginatedRequest listBucketsPaginatedRequest = new
 ListBucketsPaginatedRequest().withMaxBuckets(1);
```

```
        ListBucketsPaginatedResult listBucketsPaginatedResult =
 s3.listBuckets(listBucketsPaginatedRequest);
        List<Bucket> buckets = listBucketsPaginatedResult.getBuckets();
        System.out.println("Your Amazon S3 buckets are:");
        for (Bucket b : buckets) {
            System.out.println("* " + b.getName() + " region: " + b.getRegion());
        }
        System.out.println("continuation token: " +
 listBucketsPaginatedResult.getContinuationToken());
    }
}
```

SDK for Java 2.x

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;

import java.util.List;

public class ListBuckets {
    public static void main(String[] args) {
        // Create an S3 client
        S3Client s3 = S3Client.builder()
                .region(Region.US_EAST_1) // Replace with your preferred region
                .credentialsProvider(DefaultCredentialsProvider.create())
                .build();

        try {
            // List buckets
            ListBucketsRequest listBucketsRequest = ListBucketsRequest.builder()
                    .maxBuckets(10)
                    .build();
            ListBucketsResponse listBucketsResponse =
 s3.listBuckets(listBucketsRequest);
            List<Bucket> buckets = listBucketsResponse.buckets();

            // Print bucket names
            System.out.println("Your Amazon S3 buckets are:");
            for (Bucket bucket : buckets) {
```

```
                System.out.println(bucket.name());
                System.out.println(bucket.getBucketRegion());
            }

        } catch (Exception e) {
            System.err.println("Error listing buckets: " + e.getMessage());
            e.printStackTrace();
        } finally {
            // Close the S3 client to release resources
            s3.close();
        }
    }
}
```

SDK for Go

```
package main
import (
 "context"
 "fmt"
 "log"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)
func main() {
 cfg, err := config.LoadDefaultConfig(context.TODO(), config.WithRegion("us-
east-2"))
 if err != nil {
  log.Fatal(err)
 }
 client := s3.NewFromConfig(cfg)
 maxBuckets := 1000
 resp, err := client.ListBuckets(context.TODO(), management
 portals3.ListBucketsInput{MaxBuckets: aws.Int32(int32(maxBuckets))})
 if err != nil {
  log.Fatal(err)
 }
 fmt.Println("S3 Buckets:")
 for _, bucket := range resp.Buckets {
     fmt.Println("- Name:", *bucket.Name)
     fmt.Println("-BucketRegion", *bucket.BucketRegion)
 }
```

```
 fmt.Println(resp.ContinuationToken == nil)
 fmt.Println(resp.Prefix == nil)
}
```

범용 버킷 비우기

Amazon S3 콘솔, AWS SDK 또는 AWS Command Line Interface(AWS CLI)를 사용하여 범용 버킷의 콘텐츠를 비울 수 있습니다. 범용 버킷을 비우면 모든 객체가 삭제되지만 버킷은 유지됩니다. 버킷을 비운 후에는 실행 취소할 수 없습니다. 버킷 비우기 작업이 진행되는 동안 버킷에 추가된 객체도 삭제될 수 있습니다. 버킷 자체를 삭제하려면 먼저 버킷의 모든 객체(모든 객체 버전 및 삭제 마커 포함)를 삭제해야 합니다.

S3 버전 관리를 활성화한 상태로 또는 일시 중지된 상태로 범용 버킷을 비우면 버킷에 있는 모든 객체의 모든 버전이 삭제됩니다. 자세한 내용은 버전 관리가 사용 설정된 버킷의 객체 작업 섹션을 참조하세요.

버킷을 비우는 동안 미완료 멀티파트 업로드도 모두 제거하는 것이 좋습니다. 멀티파트 업로드를 사용하면 매우 큰 객체(최대 5TB)를 하나의 파트 집합으로 업로드하여 처리량을 개선하고 네트워크 문제를 신속하게 복구할 수 있습니다. 멀티파트 업로드 프로세스가 완료되지 않는 경우 미완료 파트가 (사용할 수 없는 상태로) 버킷에 남아 있게 됩니다. 이렇게 미완료 파트가 발생할 경우 업로드 프로세스가 완료되거나 미완료 파트가 제거될 때까지 스토리지 비용이 발생합니다. 자세한 내용은 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 섹션을 참조하세요.

특정 일수보다 오래된 객체 및 미완료 멀티파트 업로드를 만료시키도록 수명 주기 규칙을 구성하는 것이 가장 좋습니다. 미완료 멀티파트 업로드를 만료시키도록 수명 주기 규칙을 만들 때는 7일부터 시작하는 것이 좋습니다. 자세한 내용은 버킷에서 S3 수명 주기 구성 설정 섹션을 참조하세요.

수명 주기 만료는 비동기 프로세스이므로 버킷이 비워지기 전에 규칙이 실행되는 데 며칠이 걸릴 수 있습니다. Amazon S3가 처음 규칙을 실행하면 만료 대상인 모든 객체가 삭제 대상으로 표시됩니다. 삭제 대상으로 표시된 객체에 대해서는 더 이상 요금이 청구되지 않습니다. 자세한 내용은 수명 주기 구성 규칙을 사용하여 Amazon S3 버킷을 비우려면 어떻게 해야 합니까?를 참조하세요.

S3 콘솔 사용

Amazon S3 콘솔을 사용하여 범용 버킷을 비울 수 있습니다. 버킷을 비우면 버킷은 삭제되지 않고 버킷의 모든 객체가 삭제됩니다.

S3 버킷을 비우려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 비우려는 버킷의 이름 옆에 있는 옵션을 선택한 다음 비우기를 선택합니다.
4. 버킷 비우기 페이지에서 텍스트 필드에 버킷 이름을 입력하여 버킷을 비울 것인지 확인한 다음 Empty(비우기)를 선택합니다.
5. [버킷 비우기: 상태(Empty bucket: Status)] 페이지에서 버킷 비우기 프로세스의 진행 상황을 모니터링합니다.

AWS CLI 사용

버킷 버전 관리가 사용되지 않은 경우에만 AWS CLI를 사용하여 범용 버킷을 비울 수 있습니다. 버전 관리가 사용되지 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 내용은 AWS Command Line Interface 사용 설명서의 AWS CLI와 함께 상위 수준 S3 명령 사용을 참조하세요.

> **Note**
>
> 버전 관리가 활성화된 버킷에서는 객체를 제거할 수 없습니다. 이 명령을 사용하면 객체를 삭제할 때 Amazon S3가 삭제 마커를 추가합니다. S3 버킷 버전 관리에 대한 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하세요.

AWS SDK 사용

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

AWS SDK for Java 제품을 사용하여 버킷을 비우는 방법의 예는 범용 버킷 삭제 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 내용은 Amazon Web Services용 도구를 참조하세요.

수명 주기 구성 사용

대용량 범용 버킷을 비우려면 S3 수명 주기 구성 규칙을 사용하는 것이 좋습니다. 수명 주기 만료는 비동기 프로세스이므로 버킷이 비워지기 전에 규칙이 실행되는 데 며칠이 걸릴 수 있습니다. Amazon S3가 처음 규칙을 실행하면 만료 대상인 모든 객체가 삭제 대상으로 표시됩니다. 삭제 대상으로 표시된 객체에 대해서는 더 이상 요금이 청구되지 않습니다. 자세한 내용은 수명 주기 구성 규칙을 사용하여 Amazon S3 버킷을 비우려면 어떻게 해야 합니까?를 참조하세요.

수명 주기 구성을 사용하여 버킷을 비우는 경우, 수명 주기 구성에는 최신 버전, 최신이 아닌 버전, 삭제 마커, 불완전 멀티파트 업로드가 포함되어야 합니다.

수명 주기 구성 규칙을 추가하여 특정 키 이름 접두사를 가진 모든 객체 또는 객체의 하위 집합에 대해 만료를 지정할 수 있습니다. 예를 들어, 버킷의 모든 객체를 제거하기 위해서는 생성 후 하루가 지나면 객체가 만료되도록 수명 주기 규칙을 설정할 수 있습니다.

이제 Amazon S3는 시작된 후 지정 일수 내에 완료되지 않은 멀티파트 업로드를 중단하는 데 사용할 수 있는 버킷 수명 주기 규칙을 지원합니다. 스토리지 비용을 최소화하려면 이 수명 주기 규칙을 구성하는 것이 좋습니다. 자세한 내용은 불완전한 멀티파트 업로드를 삭제하도록 버킷 수명 주기 구성 설정 섹션을 참조하세요.

수명 주기 구성을 사용하여 버킷을 비우는 방법에 대한 자세한 내용은 버킷에서 S3 수명 주기 구성 설정 및 객체 만료 단원을 참조하세요.

AWS CloudTrail가 구성된 범용 버킷 비우기

AWS CloudTrail은 Amazon S3 범용 버킷의 객체 수준 데이터 이벤트(예: 객체 삭제)를 추적합니다. 범용 버킷을 대상으로 사용하여 CloudTrail 이벤트를 로깅하고 동일한 버킷에서 객체를 삭제하는 경우 버킷을 비우는 동안 새 객체가 생성될 수 있습니다. 이를 방지하려면 AWS CloudTrail 추적을 중지하세요.

CloudTrail 추적에서 이벤트 로깅을 중지하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 추적에 대해 로깅 비활성화를 참조하세요.

CloudTrail 추적이 버킷에 추가되는 것을 중지하는 또 다른 방법은 버킷 정책에 거부 `s3:PutObject` 문을 추가하는 것입니다. 나중에 버킷에 새 객체를 저장하려는 경우 이 거부 `s3:PutObject` 문을 제거해야 합니다. 자세한 내용은 IAM 사용 설명서의 객체 작업과 IAM JSON 정책 요소: 효과를 참조하세요.

범용 버킷 삭제

빈 Amazon S3 범용 버킷을 삭제할 수 있습니다. 범용 버킷 비우기에 대한 자세한 내용은 the section called “범용 버킷 비우기” 섹션을 참조하세요.

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하여 버킷을 삭제할 수 있습니다.

> **Important**
>
> 범용 버킷을 삭제하기 전에 다음 사항을 고려하세요.
>
> - 범용 버킷 이름은 글로벌 네임스페이스 내에서 고유합니다. 버킷을 삭제하면 다른 AWS 계정이 새 버킷에 대해 동일한 범용 버킷 이름을 사용할 수 있으므로 삭제된 버킷에 대한 요청을 수신할 수도 있다는 점에 유의하세요. 이 현상을 방지하고 싶거나 같은 버킷 이름을 사용하려면 버킷을 삭제하지 마세요. 버킷을 비우고 보관한 다음 필요에 따라 버킷 요청을 차단하는 것이 좋습니다. 더 이상 적극적으로 사용되지 않는 버킷의 경우 버킷 자체를 유지하면서 비용을 최소화하기 위해 버킷에서 모든 객체를 비우는 것이 좋습니다.
> - 범용 버킷을 삭제하면 버킷이 즉시 제거되지 않을 수 있습니다. 대신 Amazon S3는 삭제를 위해 버킷을 대기열에 넣습니다. Amazon S3는 AWS 리전 전반에 분산되므로 삭제 프로세스가 완전히 전파되고 시스템 전체에서 일관성을 달성하는 데 시간이 걸립니다.
> - 버킷이 정적 웹 사이트를 호스팅하고 자습서: Route 53에 등록된 사용자 지정 도메인을 사용하여 정적 웹 사이트 구성에 설명된 대로 Amazon Route 53 호스팅 영역을 생성 및 구성한 경우 버킷과 관련된 Route 53 호스팅 영역 설정을 정리해야 합니다. 자세한 내용은 2단계: Route 53 호스팅 영역 삭제 섹션을 참조하세요.
> - 버킷이 Elastic Load Balancing(ELB)에서 로그 데이터를 수신하는 경우 버킷을 삭제하기 전에 버킷으로 ELB 로그 전달을 중지하는 것이 좋습니다. 그렇지 않으면 버킷을 삭제했는데 다른 사용자가 이름이 같은 버킷을 생성하면 여러분의 로그 데이터가 해당 버킷으로 전달될 수 있습니다. ELB 액세스 로그에 대한 자세한 내용은 Classic Load Balancer 사용 설명서의

Access logs for your Classic Load Balancer 및 Application Load Balancer 사용 설명서의 Access logs for your Application Load Balancer를 참조하세요.

문제 해결

Amazon S3 범용 버킷을 삭제할 수 없는 경우, 다음을 고려하세요.

- 버킷이 비어 있는지 확인 - 객체가 없는 버킷만 삭제할 수 있습니다. 버킷이 비어 있는지 확인하세요. 버킷 비우기에 대한 자세한 내용은 the section called “범용 버킷 비우기” 섹션을 참조하세요.
- 연결된 액세스 포인트가 없는지 확인 - 동일한 계정 내에 연결된 S3 액세스 포인트 또는 다중 리전 액세스 포인트가 없는 경우에만 버킷을 삭제할 수 있습니다. 버킷을 삭제하기 전에 버킷에 연결된 같은 계정의 액세스 포인트를 모두 삭제하세요.
- **`s3:DeleteBucket`** 권한이 있어야 함 – 버킷을 삭제할 수 없는 경우 IAM 관리자와 협력하여 `s3:DeleteBucket` 권한을 확보합니다. IAM 권한을 확인하거나 업데이트하는 방법에 대한 자세한 내용은 IAM 사용 설명서에서 IAM 사용자의 권한 변경을 참조하세요. 문제 해결 정보는 the section called “액세스 거부 (403 Forbidden) 오류 문제 해결”를 참조하세요.
- AWS Organizations 서비스 제어 정책(SCP) 및 리소스 제어 정책(RCP)의 `s3:DeleteBucket` Deny 문 확인 – SCP 및 RCP는 버킷에 대한 삭제 권한을 거부할 수 있습니다. 자세한 내용은 AWS Organizations 사용 설명서의 서비스 제어 정책 및 리소스 제어 정책을 참조하세요.
- 버킷 정책의 **`s3:DeleteBucket Deny`** 문 확인 – IAM 사용자 또는 역할 정책에 `s3:DeleteBucket` 권한이 있는데 버킷을 삭제할 수 없는 경우, 버킷 정책에 `s3:DeleteBucket`에 대한 `Deny` 문이 포함되어 있을 수 있습니다. AWS Elastic Beanstalk에서 만든 버킷에는 기본적으로 이 문이 포함된 정책이 있습니다. 버킷을 삭제하려면 먼저 이 문 또는 버킷 정책을 삭제해야 합니다.

사전 조건

범용 버킷을 삭제하려면 버킷을 비워야 합니다. 버킷 비우기에 대한 자세한 내용은 the section called “범용 버킷 비우기” 섹션을 참조하세요.

S3 콘솔 사용

S3 버킷 삭제

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.

2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 삭제할 버킷 이름 옆에 있는 옵션 버튼을 선택한 후 페이지 상단에서 삭제를 선택합니다.
4. 버킷 삭제(Delete bucket) 페이지의 텍스트 필드에 버킷 이름을 입력하여 버킷의 삭제 여부를 확인한 다음 버킷 삭제(Delete bucket)를 선택합니다.

> **Note**
>
> 버킷에 객체가 포함된 경우 버킷을 삭제하기 전에 이 버킷은 비어 있지 않음 오류 알림의 버킷 비우기 버튼을 선택하고 버킷 비우기 페이지의 지침에 따라 버킷을 비웁니다. 그런 다음 버킷 삭제 페이지로 돌아가서 버킷을 삭제합니다.

5. 버킷을 삭제했는지 확인하려면 범용 버킷 목록을 열고 삭제한 버킷의 이름을 입력합니다. 버킷을 찾을 수 없다면 성공적으로 삭제한 것입니다.

Java용 AWS SDK 사용

다음 예시에서는 AWS SDK for Java를 사용하여 범용 버킷을 비우고 삭제하는 방법을 보여줍니다. 먼저 코드가 범용 버킷의 모든 객체를 삭제한 후 버킷을 삭제합니다.

다른 언어의 예제는 Amazon Simple Storage Service API 참조의 Use DeleteBucket with an AWS SDK or CLI를 참조하세요. 그 밖의 AWS SDK 사용에 대한 자세한 내용은 Amazon Web Services에서의 구축을 위한 도구를 참조하세요.

Java

다음 Java 예시는 객체가 포함된 버킷을 삭제합니다. 모든 객체를 삭제한 후 버킷을 삭제합니다. 이 예제는 버전 관리가 사용 설정되거나 사용 설정되지 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 샘플을 만들고 테스트하는 방법에 대한 지침은 AWS SDK for Java 2.x 개발자 안내서를 참조하세요.

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

import java.util.Iterator;

public class DeleteBucket2 {

    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();
        }
```

```
    }
}
```

AWS CLI 사용

버킷에 버전 관리가 활성화되지 않은 경우 AWS CLI를 사용하여 객체가 포함된 범용 버킷을 삭제할 수 있습니다. 객체가 포함된 버킷을 삭제하면 S3 Glacier Flexible Retrieval 스토리지 클래스로 전환된 객체를 포함해 버킷의 모든 객체가 영구적으로 삭제됩니다.

버킷에 버전 관리가 사용되지 않은 경우 `rb`(버킷 제거) AWS CLI 명령을 `--force` 파라미터와 함께 사용하여 버킷과 버킷의 모든 객체를 삭제할 수 있습니다. 이 명령은 먼저 모든 객체를 삭제한 후 버킷을 삭제합니다.

버전 관리가 활성화된 경우 `--force` 파라미터와 함께 `rb` 명령을 사용하면 버전이 지정된 객체가 삭제되지 않으므로 버킷이 비어 있지 않아 버킷 삭제가 실패합니다. 버전이 지정된 객체 삭제에 대한 자세한 내용은 객체 버전 삭제를 참조하세요.

다음 명령을 사용하려면 *amzn-s3-demo-bucket*을 삭제하려는 버킷의 이름으로 바꾸세요.

```
$ aws s3 rb s3://amzn-s3-demo-bucket --force
```

자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS Command Line Interface에서 상위 수준 S3 명령 사용을 참조하세요.

Amazon S3 버킷을 로컬 파일 시스템으로 마운트

Mountpoint for Amazon S3는 Amazon S3 버킷을 로컬 파일 시스템에 마운트하는 데 사용되는 높은 처리량 성능의 오픈 소스 파일 클라이언트입니다. Mountpoint를 사용하면 애플리케이션에서 열기 및 읽기와 같은 파일 시스템 작업을 통해 Amazon S3에 저장된 객체에 액세스할 수 있습니다. Mountpoint는 이러한 작업을 S3 객체 API 호출로 자동 변환하여 애플리케이션이 파일 인터페이스를 통해 Amazon S3의 탄력적 스토리지 및 처리량을 이용할 수 있도록 합니다.

Mountpoint for Amazon S3는 데이터 레이크, 기계 학습 훈련, 이미지 렌더링, 자율 주행 차량 시뮬레이션, 추출, 전환, 적재(ETL) 등 읽기 작업이 많은 대규모 애플리케이션에서 프로덕션용으로 사용 가능합니다.

Mountpoint는 기본적인 파일 시스템 작업을 지원하며 최대 5TB 크기의 파일을 읽을 수 있습니다. 기존 파일을 나열하고 읽을 수 있으며 새 파일을 만들 수도 있습니다. 기존 파일을 수정하거나 디렉터

리를 삭제할 수 없으며, 상징적 링크나 파일 잠금을 지원하지 않습니다. Mountpoint는 공유 파일 시스템의 모든 기능과 POSIX 스타일 권한이 필요하지는 않지만 대용량 S3 데이터 세트를 읽고 쓰기 위해 Amazon S3의 탄력적인 처리량이 필요한 애플리케이션에 이상적입니다. 자세한 내용은 GitHub의 Mountpoint 파일 시스템 동작을 참조하세요. 완전한 POSIX 지원이 필요한 워크로드의 경우 Amazon FSx for Lustre 및 해당 Amazon S3 버킷 연결 지원 기능이 권장됩니다.

Mountpoint for Amazon S3는 Linux 운영 체제에만 사용할 수 있습니다. Mountpoint를 사용하여 S3 Glacier Flexible Retrieval, S3 Glacier Deep Archive, S3 Intelligent-Tiering Archive Access Tier 및 S3 Intelligent-Tiering Deep Archive Access Tier를 제외한 모든 스토리지 클래스에서 S3 객체에 액세스할 수 있습니다.

주제

- Mountpoint 설치
- Mountpoint 구성 및 사용
- Mountpoint 문제 해결

Mountpoint 설치

명령줄을 사용하여 Mountpoint for Amazon S3의 사전 빌드된 패키지를 다운로드하고 설치할 수 있습니다. Mountpoint 다운로드 및 설치 지침은 사용 중인 Linux 운영 체제에 따라 다릅니다.

주제

- RPM 기반 배포판(Amazon Linux, Fedora, CentOS, RHEL)
- DEB 기반 배포판(Debian, Ubuntu)
- 기타 Linux 배포판
- Mountpoint for Amazon S3 패키지의 서명 확인

RPM 기반 배포판(Amazon Linux, Fedora, CentOS, RHEL)

1. 사용하는 아키텍처에 해당하는 다음 다운로드 URL을 복사합니다.

   x86_64:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.rpm
   ```

   ARM64 (Graviton):

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.rpm
```

2. Mountpoint for Amazon S3 패키지를 다운로드합니다. *download-link*를 앞 단계의 해당 다운로드 URL로 바꿉니다.

```
wget download-link
```

3. (선택 사항) 다운로드한 파일의 신뢰성 및 무결성을 확인합니다. 먼저 사용하는 아키텍처에 해당하는 서명 URL을 복사합니다.

   x86_64:

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.rpm.asc
```

   ARM64 (Graviton):

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.rpm.asc
```

   그런 다음, Mountpoint for Amazon S3 패키지의 서명 확인을 참조하세요.

4. 다음 명령을 사용하여 패키지를 설치합니다.

```
sudo yum install ./mount-s3.rpm
```

5. 다음 명령을 입력하여 Mountpoint가 성공적으로 설치되었는지 확인합니다.

```
mount-s3 --version
```

   다음과 유사한 출력 화면이 표시되어야 합니다.

```
mount-s3 1.3.1
```

DEB 기반 배포판(Debian, Ubuntu)

1. 사용하는 아키텍처에 해당하는 다운로드 URL을 복사합니다.

   x86_64:

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.deb
```

ARM64 (Graviton):

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.deb
```

2. Mountpoint for Amazon S3 패키지를 다운로드합니다. *download-link*를 앞 단계의 해당 다운로드 URL로 바꿉니다.

```
wget download-link
```

3. (선택 사항) 다운로드한 파일의 신뢰성 및 무결성을 확인합니다. 먼저 사용하는 아키텍처에 해당하는 서명 URL을 복사합니다.

   x86_64:

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.deb.asc
```

   ARM64 (Graviton):

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.deb.asc
```

   그런 다음, Mountpoint for Amazon S3 패키지의 서명 확인을 참조하세요.

4. 다음 명령을 사용하여 패키지를 설치합니다.

```
sudo apt-get install ./mount-s3.deb
```

5. 다음 명령을 실행하여 Mountpoint for Amazon S3가 성공적으로 설치되었는지 확인합니다.

```
mount-s3 --version
```

   다음과 유사한 출력 화면이 표시되어야 합니다.

```
mount-s3 1.3.1
```

기타 Linux 배포판

1. 운영 체제 설명서를 참조하여 필수 구성 요소인 FUSE 및 `libfuse2` 패키지를 설치합니다.
2. 사용하는 아키텍처에 해당하는 다운로드 URL을 복사합니다.

   x86_64:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.tar.gz
   ```

   ARM64 (Graviton):

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.tar.gz
   ```

3. Mountpoint for Amazon S3 패키지를 다운로드합니다. *`download-link`*를 앞 단계의 해당 다운로드 URL로 바꿉니다.

   ```
   wget download-link
   ```

4. (선택 사항) 다운로드한 파일의 신뢰성 및 무결성을 확인합니다. 먼저 사용하는 아키텍처에 해당하는 서명 URL을 복사합니다.

   x86_64:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.tar.gz.asc
   ```

   ARM64 (Graviton):

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.tar.gz.asc
   ```

   그런 다음, Mountpoint for Amazon S3 패키지의 서명 확인을 참조하세요.

5. 다음 명령을 사용하여 패키지를 설치합니다.

   ```
   sudo mkdir -p /opt/aws/mountpoint-s3 && sudo tar -C /opt/aws/mountpoint-s3 -xzf ./
   mount-s3.tar.gz
   ```

6. `PATH` 환경 변수에 `mount-s3` 바이너리를 추가합니다. `$HOME/.profile` 파일에 다음 줄을 추가합니다.

```
export PATH=$PATH:/opt/aws/mountpoint-s3/bin
```

`.profile` 파일을 저장하고 다음 명령을 실행합니다.

```
source $HOME/.profile
```

7. 다음 명령을 실행하여 Mountpoint for Amazon S3가 성공적으로 설치되었는지 확인합니다.

```
mount-s3 --version
```

다음과 유사한 출력 화면이 표시되어야 합니다.

```
mount-s3 1.3.1
```

Mountpoint for Amazon S3 패키지의 서명 확인

1. GnuPG를 설치합니다(gpg 명령). 다운로드한 Mountpoint for Amazon S3 패키지의 신뢰성 및 무결성을 확인하는 데 필요합니다. GnuPG는 Amazon Linux Amazon Machine Image(AMI)에 기본적으로 설치됩니다. GnuPG를 설치한 후 2단계로 진행합니다.
2. 다음 명령을 실행하여 Mountpoint 퍼블릭 키를 다운로드합니다.

```
wget https://s3.amazonaws.com/mountpoint-s3-release/public_keys/KEYS
```

3. 다음 명령을 실행하여 Mountpoint 퍼블릭 키를 키 링으로 가져옵니다.

```
gpg --import KEYS
```

4. 다음 명령을 실행하여 Mountpoint 퍼블릭 키의 지문을 확인합니다.

```
gpg --fingerprint mountpoint-s3@amazon.com
```

표시된 지문 문자열이 다음과 일치하는지 확인합니다.

```
673F E406 1506 BB46 9A0E  F857 BE39 7A52 B086 DA5A
```

지문 문자열이 일치하지 않는 경우, Mountpoint 설치를 완료하지 말고 AWS Support에 문의하세요.

5. 패키지 서명 파일을 다운로드합니다. *signature-link*를 앞 섹션에 나온 해당 서명 링크로 바꿉니다.

```
wget signature-link
```

6. 다음 명령을 실행하여 다운로드한 패키지의 서명을 확인합니다. *signature-filename*을 앞 단계에 나온 파일 이름으로 바꿉니다.

```
gpg --verify signature-filename
```

예를 들어 Amazon Linux가 포함된 RPM 기반 배포판의 경우 다음 명령을 실행합니다.

```
gpg --verify mount-s3.rpm.asc
```

7. 출력에 `Good signature`라는 문구가 포함되어야 합니다. 출력에 `BAD signature`라는 문구가 포함된 경우 Mountpoint 패키지 파일을 다시 다운로드하고 이러한 단계를 반복합니다. 문제가 지속될 경우 Mountpoint 설치를 마치지 말고 AWS Support에 문의하세요.

   출력에는 신뢰할 수 있는 서명에 대한 경고가 포함될 수 있습니다. 이는 문제가 있다는 뜻이 아니며, Mountpoint 퍼블릭 키를 독립적으로 확인하지 않았다는 의미일 뿐입니다.

Mountpoint 구성 및 사용

Mountpoint for Amazon S3를 사용하려면 호스트에 마운트하려는 하나 이상의 Amazon S3 버킷에 대한 액세스 권한이 있는 유효한 AWS 자격 증명이 필요합니다. 다양한 인증 방법은 GitHub의 AWS 자격 증명을 참조하세요.

예를 들어, 이 목적을 위해 새 AWS Identity and Access Management(IAM) 사용자 및 역할을 생성할 수 있습니다. 이 역할에 마운트하려는 하나 이상의 버킷에 대한 액세스 권한이 있는지 확인합니다. 인스턴스 프로파일을 사용하여 Amazon EC2 인스턴스에 IAM 역할을 전달할 수 있습니다.

주제

- Mountpoint for Amazon S3 사용
- Mountpoint에서 캐시 구성

Mountpoint for Amazon S3 사용

Mountpoint for Amazon S3를 사용하여 다음을 수행합니다.

1. Amazon S3 버킷을 마운트합니다.

   a. `mount-s3` 명령을 사용하여 Amazon S3 버킷을 수동으로 마운트할 수 있습니다.

   다음 예에서는 *amzn-s3-demo-bucket*을 S3 버킷 이름으로 바꾸고 *~/mnt*를 S3 버킷을 마운트할 호스트의 디렉터리로 바꿉니다.

   ```
   mkdir ~/mnt
   mount-s3 amzn-s3-demo-bucket ~/mnt
   ```

   Mountpoint 클라이언트는 기본적으로 백그라운드에서 실행되기 때문에 이제 *~/mnt* 디렉터리를 통해 Amazon S3 버킷의 객체에 액세스할 수 있습니다.

   b. 또는 인스턴스가 시작되거나 재부팅될 때 Amazon S3 버킷의 자동 마운트를 구성할 수 있습니다.

   기존 또는 실행 중인 Amazon EC2 인스턴스의 경우 Linux 시스템의 `/etc/fstab` 디렉터리에서 `fstab` 파일을 찾습니다. 그런 다음 `fstab` 파일에 줄을 추가합니다. 예를 들어 접두사 `example-prefix/`를 사용하여 시스템 경로 `/mnt/mountpoint`에 *amzn-s3-demo-bucket*을 마운트하려면 다음을 참조하세요. 다음 예시를 사용하려면 *### ## ## ###*를 실제 정보로 바꾸세요.

   ```
   s3://amzn-s3-demo-bucket/example-prefix/ /mnt/mountpoint mount-s3
    _netdev,nosuid,nodev,nofail,rw 0 0
   ```

   예시에 사용된 옵션에 대한 설명은 다음 표를 참조하세요.

옵션	설명
`_netdev`	파일 시스템을 마운트하려면 네트워킹이 필요하도록 지정합니다.
`nosuid`	파일 시스템에 설정된 사용자 ID 파일을 포함할 수 없도록 지정합니다.

옵션	설명
nodev	파일 시스템에 특수 디바이스를 포함할 수 없도록 지정합니다.
nofail	파일 시스템을 마운트하지 못하더라도 시스템이 부팅되도록 지정합니다.
rw	읽기 및 쓰기 권한으로 마운트 지점이 생성되도록 지정합니다. 또는 읽기 전용을 위해 ro를 사용합니다.

새 Amazon EC2 인스턴스의 경우 Amazon EC2 템플릿에서 사용자 데이터를 수정하고 다음과 같이 fstab 파일을 설정할 수 있습니다. 다음 예시를 사용하려면 *### ## ## ###*를 실제 정보로 바꾸세요.

```
#!/bin/bash -e
MP_RPM=$(mktemp --suffix=.rpm)
curl https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.rpm
 > $MP_RPM
yum install -y $MP_RPM
rm $MP_RPM

MNT_PATH=/mnt/mountpoint
echo "s3://amzn-s3-demo-bucket/ ${MNT_PATH} mount-s3
 _netdev,nosuid,nodev,rw,allow-other,nofail" >> /etc/fstab
mkdir $MNT_PATH

systemctl daemon-reload
mount -a
```

2. Mountpoint를 통해 Amazon S3 버킷의 객체에 액세스합니다.

   버킷을 로컬로 마운트 후 cat 또는 ls와 같은 일반적인 Linux 명령을 사용하여 S3 객체 작업을 수행할 수 있습니다. Mountpoint for Amazon S3는 Amazon S3 버킷의 키를 슬래시(/) 문자로 구분하여 파일 시스템 경로로 해석합니다. 예를 들어 버킷에 객체 키 Data/2023-01-01.csv가 있는 경우 Mountpoint 파일 시스템에 Data라는 디렉터리가 있고 그 안에 2023-01-01.csv라는 파일이 있습니다.

Mountpoint for Amazon S3는 의도적으로 파일 시스템에 대한 전체 POSIX 표준 사양을 구현하지 않습니다. Mountpoint는 파일 시스템 인터페이스를 통해 Amazon S3에 저장된 데이터에 대한 높은 처리량의 읽기 및 쓰기 액세스가 필요하지만 그 외에는 파일 시스템 기능에 의존하지 않는 워크로드에 최적화되어 있습니다. 자세한 내용은 GitHub의 Mountpoint for Amazon S3 파일 시스템 동작을 참조하세요. 보다 풍부한 파일 시스템 의미 체계가 필요한 고객은 Amazon Elastic File System(Amazon EFS) 또는 Amazon FSx 등의 다른 AWS 파일 서비스를 고려해야 합니다.

3. `umount` 명령을 사용하여 Amazon S3 버킷을 마운트 해제합니다. 이 명령은 S3 버킷을 마운트 해제하고 Mountpoint를 종료합니다.

   다음 예제 명령을 사용하려면 *~/mnt*를 S3 버킷이 마운트된 호스트의 디렉터리로 바꿉니다.

   ```
   umount ~/mnt
   ```

   > **Note**
   >
   > 이 명령의 옵션 목록을 보려면 `umount --help`를 실행합니다.

추가 Mountpoint 구성 세부 정보는 GitHub의 Amazon S3 버킷 구성 및 파일 시스템 구성을 참조하세요.

Mountpoint에서 캐시 구성

Mountpoint for Amazon S3는 다양한 유형의 데이터 캐싱을 지원합니다. 반복되는 읽기 요청을 가속화하려면 다음을 옵트인할 수 있습니다.

- 로컬 캐시 - Amazon EC2 인스턴스 스토리지 또는 Amazon Elastic Block Store 볼륨에서 로컬 캐시를 사용할 수 있습니다. 동일한 컴퓨팅 인스턴스에서 동일한 데이터를 반복적으로 읽고 반복 읽기 데이터세트에 대해 로컬 인스턴스 스토리지에 미사용 공간이 있는 경우 로컬 캐시에 옵트인해야 합니다.
- 공유 캐시 - S3 Express One Zone에서 공유 캐시를 사용할 수 있습니다. 여러 컴퓨팅 인스턴스에서 작은 객체를 반복적으로 읽거나 반복적으로 읽는 데이터세트의 크기를 모르고 캐시 크기의 탄력성을 활용하려는 경우 공유 캐시에 옵트인해야 합니다. 옵트인하면 Mountpoint는 S3 Express One Zone을 사용하는 디렉터리 버킷에 최대 1MB 크기의 객체를 유지합니다.
- 로컬 캐시 및 공유 캐시 결합 - 로컬 캐시에 미사용 공간이 있지만 여러 인스턴스에서 공유 캐시도 원할 경우 로컬 캐시와 공유 캐시를 모두 옵트인할 수 있습니다.

Mountpoint에서의 캐시는 여러 번 읽는 동안 변경되지 않는 동일한 데이터를 반복적으로 읽는 사용 사례에 적합합니다. 예를 들어, 모델 정확도를 높이기 위해 학습 데이터 세트를 여러 번 읽어야 하는 기계 학습 훈련 작업에 캐시를 사용할 수 있습니다.

Mountpoint에서 캐싱을 구성하는 방법에 대한 자세한 내용은 다음 예제를 참조하세요.

주제

- 로컬 캐시
- 공유 캐시
- 로컬 캐시 및 공유 캐시 결합

로컬 캐시

`--cache` *`CACHE_PATH`* 플래그를 사용하여 로컬 캐시에 옵트인할 수 있습니다. 다음 예시에서 데이터를 캐시하려는 디렉터리의 파일 경로로 *`CACHE_PATH`*를 바꾸세요. *`amzn-s3-demo-bucket`*을 Amazon S3 버킷 이름으로 바꾸고 *`~/mnt`*를 S3 버킷을 마운트할 호스트의 디렉터리로 바꿉니다.

```
mkdir ~/mnt
mount-s3 --cache CACHE_PATH amzn-s3-demo-bucket ~/mnt
```

Amazon S3 버킷을 마운트하고 로컬 캐싱을 옵트인하면 하위 디렉터리가 아직 없는 경우 Mountpoint는 구성된 캐시 위치에 빈 하위 디렉터리를 생성합니다. 버킷을 처음 마운트하고 마운트를 해제하면 Mountpoint는 로컬 캐시의 콘텐츠를 삭제합니다.

> **Important**
>
> 로컬 캐싱을 활성화할 경우 Mountpoint는 마운트 시 제공된 로컬 캐시 위치에 마운트된 Amazon S3 버킷의 암호화되지 않은 객체 콘텐츠를 보관합니다. 데이터를 보호하려면 파일 시스템 액세스 제어 메커니즘을 사용하여 데이터 캐시 위치에 대한 액세스를 제한해야 합니다.

공유 캐시

여러 컴퓨팅 인스턴스에서 작은 객체(최대 1MB)를 반복적으로 읽거나 반복적으로 읽는 데이터세트 크기가 로컬 캐시 크기를 초과하는 경우가 많을 경우 S3 Express One Zone에서 공유 캐시를 사용해야 합니다. 여러 인스턴스에서 동일한 데이터를 반복적으로 읽으면 마운트된 Amazon S3 버킷에 대한 중복 요청을 방지하여 지연 시간을 개선합니다.

공유 캐시에 옵트인하면 S3 Express One Zone의 디렉터리 버킷에 캐시된 데이터에 대한 비용을 지불합니다. 또한 S3 Express One Zone의 디렉터리 버킷에 있는 데이터에 대한 요청도 비용을 지불합니다. 자세한 내용은 Amazon S3 요금을 참조하십시오. Mountpoint는 디렉터리 버킷에서 캐시된 객체를 삭제하지 않습니다. 스토리지 비용을 관리하려면 디렉터리 버킷에 수명 주기 정책을 설정하여 Amazon S3가 지정한 기간 이후에 S3 Express One Zone에서 캐시된 데이터를 만료하도록 해야 합니다. 자세한 내용은 GitHub의 Mountpoint for Amazon S3 캐싱 구성을 참조하세요.

Amazon S3 버킷을 컴퓨팅 인스턴스에 마운트할 때 S3 Express One Zone에서 캐싱에 옵트인하려면 `--cache-xz` 플래그를 사용하고 디렉터리 버킷을 캐시 위치로 지정합니다. 다음 예제에서는 *### ## ## ###*를 바꿉니다.

```
mount-s3 amzn-s3-demo-bucket ~/mnt --cache-xz amzn-s3-demo-bucket--usw2-az1--x-s3
```

로컬 캐시 및 공유 캐시 결합

인스턴스에 미사용 공간이 있지만 여러 인스턴스에서 공유 캐시도 사용하려는 경우 로컬 캐시와 공유 캐시를 모두 옵트인할 수 있습니다. 이 캐싱 구성을 사용하면 필수 데이터가 로컬 스토리지에서 캐싱될 때 디렉터리 버킷의 공유 캐시에 대한 동일한 인스턴스의 중복 읽기 요청을 방지할 수 있습니다. 이렇게 하면 요청 비용을 줄이고 성능을 개선할 수 있습니다.

Amazon S3 버킷을 마운트할 때 로컬 캐시와 공유 캐시를 모두 옵트인하려면 `--cache` 및 `--cache-xz` 플래그를 사용하여 두 캐시 위치를 모두 지정합니다. 다음 예제를 사용하여 로컬 캐시와 공유 캐시를 모두 옵트인하려면 *### ## ## ###*를 바꿉니다.

```
mount -s3 amzn-s3-demo-bucket ~/mnt --cache /path/to/mountpoint/cache --cache -xz amzn-s3-demo-bucket--usw2-az1--x-s3
```

자세한 내용은 GitHub의 Mountpoint for Amazon S3 캐싱 구성을 확인하세요.

> **Important**
>
> 공유 캐싱을 활성화할 경우 Mountpoint는 마운트된 Amazon S3 버킷의 객체 콘텐츠를 공유 캐시 위치로 제공하는 S3 디렉터리 버킷으로 복사하므로 S3 디렉터리 버킷에 액세스할 수 있는 모든 호출자가 액세스할 수 있습니다. 캐시된 데이터를 보호하려면 Amazon S3의 보안 모범 사례에 따라 버킷이 올바른 정책을 사용하고 공개적으로 액세스할 수 없도록 해야 합니다. Mountpoint 공유 캐싱 전용 디렉터리 버킷을 사용하고 Mountpoint 클라이언트에만 액세스 권한을 부여해야 합니다.

Mountpoint 문제 해결

Mountpoint for Amazon S3는 지원에서 지원됩니다. 도움이 필요한 경우 AWS Support Center에 문의하세요.

GitHub에서 Mountpoint 문제를 검토하고 제출할 수도 있습니다.

이 프로젝트에서 잠재적인 보안 문제를 발견한 경우 취약성 보고 페이지를 통해 AWS Security에 알려주시기 바랍니다. 공개적으로 GitHub 문제를 작성하지 마세요.

애플리케이션이 Mountpoint에서 예기치 않게 동작하는 경우 로그 정보를 검토하여 문제를 진단할 수 있습니다.

로깅

기본적으로 Mountpoint는 심각도가 높은 로그 정보를 syslog에 내보냅니다.

Amazon Linux를 포함한 대부분의 최신 Linux 배포판에서 로그를 보려면 다음 `journald` 명령을 실행합니다.

```
journalctl -e SYSLOG_IDENTIFIER=mount-s3
```

다른 Linux 시스템에서는 `syslog` 항목이 `/var/log/syslog`와 같은 파일에 기록될 가능성이 높습니다.

이러한 로그를 사용하여 애플리케이션 문제를 해결할 수 있습니다. 예를 들어 애플리케이션이 기존 파일을 덮어쓰려고 하면 작업이 실패하고 로그에 다음과 비슷한 줄이 표시됩니다.

```
[WARN] open{req=12 ino=2}: mountpoint_s3::fuse: open failed: inode error: inode 2 (full
 key "README.md") is not writable
```

자세한 내용은 GitHub의 Mountpoint for Amazon S3 로깅을 참조하세요.

Storage Browser for Amazon S3 작업

Storage Browser for S3는 웹 애플리케이션에 추가하여 최종 사용자에게 Amazon S3에 저장된 데이터에 대한 간단한 그래픽 인터페이스를 제공할 수 있는 오픈 소스 구성 요소입니다. Storage Browser for S3를 사용하면 권한이 있는 최종 사용자에게 S3의 데이터를 자체 애플리케이션에서 직접 검색, 다운로드, 업로드, 복사 및 삭제할 수 있는 액세스 권한을 제공할 수 있습니다.

Storage Browser for S3는 파일에 대해 LIST, GET, PUT, COPY, UPLOAD 및 DELETE 작업을 지원합니다. 높은 처리량의 데이터 전송을 제공하기 위해 Storage Browser for S3는 최종 사용자에게 액세스 권한이 부여된 데이터만 표시하고 업로드 요청을 최적화합니다. 또한 Storage Browser는 더 빠른 로드 시간을 위해 성능을 최적화하고, 최종 사용자가 업로드하는 데이터의 체크섬을 계산하고, 퍼블릭 인터넷을 통해 전송 중에 데이터 무결성이 유지되었는지 확인한 후 객체를 수락합니다. AWS 보안 및 ID 서비스 또는 자체 관리형 서비스를 사용하여 최종 사용자의 ID를 기반으로 데이터에 대한 액세스를 제어할 수 있습니다. 기존 애플리케이션의 설계 및 브랜딩과 일치하도록 Storage Browser를 사용자 지정할 수도 있습니다.

Storage Browser for S3는 React 프레임워크의 웹 및 인트라넷 애플리케이션에서만 사용할 수 있습니다. Storage Browser를 사용하여 S3 Glacier Flexible Retrieval, S3 Glacier Deep Archive, S3 Intelligent-Tiering Archive Access 티어 및 S3 Intelligent-Tiering Deep Archive Access 티어를 제외한 모든 스토리지 클래스에서 Amazon S3 객체에 액세스할 수 있습니다.

Storage Browser for S3는 AWS Amplify React 라이브러리의 웹 애플리케이션과 함께 사용할 수 있습니다. Storage Browser에 대한 자세한 내용은 Storage Browser for S3를 참조하세요.

주제

- Storage Browser for S3 사용
- Storage Browser for S3 설치
- Storage Browser for S3 설정
- Storage Browser for S3 구성
- Storage Browser for S3 문제 해결

Storage Browser for S3 사용

Storage Browser for S3에서 위치는 S3 범용 버킷 또는 접두사로, 최종 사용자에게 S3 Access Grants, IAM 정책 또는 자체 관리형 권한 부여 서비스를 사용하여 위치에 대한 액세스 권한을 부여합니다. 특정 위치에 액세스하도록 최종 사용자에게 권한을 부여하면 최종 사용자가 해당 위치 내의 어떤 데이터로든 작업할 수 있습니다.

Storage Browser for S3 사용자 인터페이스에는 네 가지 기본 보기가 있습니다.

- 홈 페이지: 홈 페이지에는 사용자가 액세스할 수 있는 S3 위치와 각 위치에 대한 권한이 나열됩니다. 이는 최종 사용자가 액세스할 수 있는 루트 수준 S3 리소스와 각 S3 위치에 대한 권한(READ/WRITE/READWRITE)을 보여주는 사용자의 최초 보기입니다.

- 위치 세부 정보: 이 보기를 통해 사용자는 S3에서 파일 및 폴더를 둘러보고 파일을 업로드하거나 다운로드할 수 있습니다. (Storage Browser for S3에서 객체를 파일이라고 하고 접두사와 버킷을 폴더라고 합니다.)
- 위치 작업: 사용자가 Storage Browser에서 작업(예: 업로드)을 선택하면 파일 위치의 다른 보기가 열립니다.
- 세로 줄임표: 세로 줄임표 아이콘으로 작업 드롭다운 목록을 엽니다.

Storage Browser for S3를 사용할 때는 다음 제한 사항에 유의하세요.

- 점(.)으로 시작하거나 끝나는 폴더는 지원되지 않습니다.
- WRITE 전용 권한이 있는 S3 Access Grants는 지원되지 않습니다.
- Storage Browser for S3는 최대 160GB 크기의 파일에 대한 PUT 작업을 지원합니다.
- Storage Browser for S3는 5GB 미만의 파일에 대해 COPY 작업만 지원합니다. 파일 크기가 5GB를 초과하면 Storage Browser가 요청에 실패합니다.

Storage Browser for S3 설치

Storage Browser를 시작하는 가장 빠른 방법은 GitHub에서 샘플 프로젝트 중 하나를 복제하는 것입니다. 이러한 샘플 프로젝트를 사용하면 AWS Identity and Access Management에 대한 사전 설정된 AWS 서비스 통합을 통해 Storage Browser용 프로덕션 지원 웹 앱을 배포하여 승인된 최종 사용자를 S3의 데이터에 빠르게 연결할 수 있습니다.

자세한 내용은 Amplify 개발자 센터의 빠른 시작을 참조하세요.

GitHub에서 Storage Browser for S3 설치

또는 aws-amplify GitHub 리포지토리에 있는 최신 버전의 `aws-amplify/ui-react-storage` 및 `aws-amplify` 패키지에서 Storage Browser for S3를 설치하여 Storage Browser를 기존 애플리케이션에 통합할 수 있습니다. Storage Browser for S3를 설치할 때 `package.json` 파일에 다음 종속성을 추가해야 합니다.

```
"dependencies": {
    "aws-amplify/ui-react-storage": "latest",
    "aws-amplify": "latest",
  }
```

또는 Node Package Manager(NPM)를 사용하여 종속성을 추가할 수 있습니다.

```
npm i --save @aws-amplify/ui-react-storage aws-amplify
```

Storage Browser for S3 설정

최종 사용자를 Amazon S3 위치에 연결하려면 먼저 인증 및 권한 부여 방법을 설정해야 합니다. Storage Browser를 사용하여 인증 및 권한 부여 방법을 설정하는 방법에는 세 가지가 있습니다.

- 방법 1: 고객 및 타사 파트너의 데이터 액세스 관리
- 방법 2: AWS 계정의 IAM 위탁자에 대한 데이터 액세스 관리
- 방법 3: 대규모 데이터 액세스 관리

방법 1: 고객 및 타사 파트너의 데이터 액세스 관리

이 방법을 사용하면 AWS Amplify Auth를 사용하여 파일에 대한 액세스 제어 및 보안을 관리할 수 있습니다. 이 방법은 고객 또는 타사 파트너를 S3의 데이터와 연결하려는 경우에 적합합니다. 이 옵션을 사용하면 고객은 소셜 또는 엔터프라이즈 ID 제공업체를 사용하여 인증할 수 있습니다.

Amplify Storage를 사용하도록 구성된 S3 버킷과 함께 AWS Amplify Auth를 사용하여 최종 사용자 및 타사 파트너에게 IAM 자격 증명을 제공합니다. AWS Amplify Auth는 Amazon Cognito를 기반으로 구축되었습니다. 이 완전관리형 고객 ID 및 액세스 관리 서비스에서는 기본 제공 사용자 디렉터리 또는 엔터프라이즈 디렉터리 또는 소비자 ID 제공업체를 통해 사용자를 인증하고 권한을 부여할 수 있습니다. Amplify 권한 부여 모델은 현재 인증된 사용자가 액세스할 수 있는 접두사를 정의합니다. AWS Amplify에 대한 권한 부여를 설정하는 방법에 대한 자세한 내용은 Set up storage를 참조하세요.

Amplify 인증 및 스토리지 방법을 사용하여 구성 요소를 초기화하려면 웹 애플리케이션에 다음 코드 조각을 추가합니다.

```
import {
  createAmplifyAuthAdapter,
  createStorageBrowser,
} from '@aws-amplify/ui-react-storage/browser';
import "@aws-amplify/ui-react-storage/styles.css";

import config from './amplify_outputs.json';

Amplify.configure(config);

export const { StorageBrowser } = createStorageBrowser({
```

```
  config: createAmplifyAuthAdapter(),
});
```

방법 2: AWS 계정의 IAM 위탁자에 대한 데이터 액세스 관리

IAM 위탁자 또는 AWS 계정에 대한 액세스를 직접 관리하려면 GetDataAccess S3 API 작업을 간접 호출할 권한이 있는 IAM 역할을 생성하면 됩니다. 설정하려면 S3 범용 버킷 및 접두사에 대한 권한을 지정된 IAM ID에 매핑하는 S3 Access Grants 인스턴스를 생성해야 합니다. 그러면 Storage Browser 구성 요소(IAM 자격 증명을 얻은 후 클라이언트 측에서 직접 호출해야 함)가 ListCallerAccessGrants S3 API 작업을 간접 호출하여 ID 요청자에게 사용 가능한 권한을 가져오고 구성 요소의 위치를 채웁니다. `s3:GetDataAccess` 권한을 얻으면 Storage Browser 구성 요소에서 이러한 자격 증명을 사용하여 S3에 대한 데이터 액세스를 요청합니다.

```
import {
  createManagedAuthAdapter,
  createStorageBrowser,
} from '@aws-amplify/ui-react-storage/browser';
import "@aws-amplify/ui-react-storage/styles.css";

export const { StorageBrowser } = createStorageBrowser({
  config: createManagedAuthAdapter({
    credentialsProvider: async (options?: { forceRefresh?: boolean }) => {
      // return your credentials object
      return {
        credentials: {
          accessKeyId: 'my-access-key-id',
          secretAccessKey: 'my-secret-access-key',
          sessionToken: 'my-session-token',
          expiration: new Date()
        },
      }
    },
    // AWS `region` and `accountId`
    region: '',
    accountId: '',
    // call `onAuthStateChange` when end user auth state changes
    // to clear sensitive data from the `StorageBrowser` state
    registerAuthListener: (onAuthStateChange) => {},
  })
});
```

방법 3: 대규모 데이터 액세스 관리

보다 확장 가능한 솔루션(예: 회사 전체에 데이터 액세스 제공)을 위해 S3 Access Grants 인스턴스를 IAM Identity Center에 연결하려면 현재 인증된 사용자를 대신하여 Amazon S3에 데이터를 요청하면 됩니다. 예를 들어 회사 디렉터리의 사용자 그룹에 S3의 데이터에 대한 액세스 권한을 부여할 수 있습니다. 이 접근 방식을 사용하면 Microsoft Entra, Okta 등과 같은 외부 제공업체에서 호스팅되는 것을 포함한 사용자 및 그룹에 대한 S3 Access Grants 권한을 중앙에서 관리할 수 있습니다.

이 방법을 사용하는 경우 IAM Identity Center와의 통합을 통해 기존 사용자 디렉터리를 사용할 수 있습니다. IAM Identity Center 신뢰할 수 있는 ID 전파의 또 다른 이점은 AWS CloudTrail Amazon S3의 각 데이터 이벤트에 S3 데이터에 액세스한 최종 사용자 ID에 대한 직접 참조가 포함된다는 것입니다.

OAuth 2.0을 지원하는 애플리케이션이 있고 사용자가 이러한 애플리케이션에서 AWS 서비스에 액세스해야 하는 경우 신뢰할 수 있는 ID 전파를 사용하는 것이 좋습니다. 신뢰할 수 있는 ID 전파를 통해 사용자는 애플리케이션에 로그인할 수 있고, 해당 애플리케이션은 AWS 서비스의 데이터에 액세스하기 위한 요청에 사용자의 ID를 전달할 수 있습니다. 이 애플리케이션은 인증된 사용자를 대신하여 IAM Identity Center와 상호 작용합니다. 자세한 내용은 Using trusted identity propagation with customer managed applications을 참조하세요.

신뢰할 수 있는 ID 전파는 연결된 AWS 서비스의 관리자가 서비스 데이터에 대한 액세스 권한을 부여하고 감사하는 데 사용할 수 있는 AWS IAM Identity Center 기능입니다. 이 데이터에 대한 액세스는 그룹 연결과 같은 사용자 속성을 기반으로 합니다. 신뢰할 수 있는 ID 전파를 설정하려면 연결된 AWS 서비스의 관리자와 IAM Identity Center 관리자 간의 협업이 필요합니다. 자세한 내용은 사전 조건 및 고려 사항을 참조하세요.

설정

S3 Access Grants 및 IAM Identity Center 신뢰할 수 있는 자격 증명 전파를 AWS Management Console 사용하여 에서 Storage Browser 인증을 설정하려면 애플리케이션이 현재 인증된 사용자를 대신하여 Amazon S3에 데이터를 요청해야 합니다. 이 접근 방식을 사용하면 회사 디렉터리의 사용자 또는 사용자 그룹에 S3 버킷, 접두사 또는 객체에 대한 직접 액세스 권한을 부여할 수 있습니다. 즉, 애플리케이션에서 사용자를 IAM 위탁자에 매핑할 필요가 없습니다.

다음 워크플로에서는 IAM Identity Center 및 S3 Access Grants를 사용하여 Storage Browser for S3를 설정하는 단계를 간략하게 설명합니다.

단계	설명
1	AWS Organizations에 대해 IAM Identity Center 활성화

단계	설명
2	AWS Identity and Access Management Identity Center 페더레이션 구성
3	AWS Identity and Access Management Identity Center 콘솔에 신뢰할 수 있는 토큰 발급자 추가 신뢰할 수 있는 토큰 발급자는 IAM Identity Center 내의 외부 ID 제공업체(IdP)를 나타내며, 애플리케이션의 인증된 사용자의 ID 토큰을 인식할 수 있도록 합니다.
4	`bootstrap` 애플리케이션 및 `identity bearer`에 대한 IAM 역할 생성
5	IAM Identity Center에서 애플리케이션 생성 및 구성 이 애플리케이션은 인증된 사용자를 대신하여 IAM Identity Center와 상호 작용합니다.
6	ID 전파를 위한 신뢰할 수 있는 애플리케이션으로 S3 Access Grants 추가 이 단계에서는 애플리케이션이 인증된 사용자를 대신하여 S3 Access Grants에 요청할 수 있도록 애플리케이션을 S3 Access Grants에 연결합니다.
7	사용자 또는 그룹에 대한 권한 부여 생성 이 단계에서는 AWS Identity and Access Management Identity Center의 사용자를 System for Cross-domain Identity Management(SCIM)와 동기화합니다. SCIM은 IAM Identity Center ID를 ID 제공업체(idP)의 ID와 동기화합니다.
8	Storage Browser for S3 구성 요소 생성

AWS Organizations에 대해 IAM Identity Center 활성화

AWS Organizations에 대해 IAM Identity Center를 활성화하려면 다음 단계를 수행하세요.

1. 다음 방법 중 하나를 사용하여 AWS Management Console에 로그인합니다.

   1. AWS를 처음 사용하시나요?(루트 사용자) - 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력해 계정 소유자로 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.
   2. 이미 AWS를 사용하고 있음(IAM 자격 증명) - 관리자 권한이 있는 IAM 자격 증명을 사용하여 로그인합니다.

2. IAM Identity Center 콘솔을 엽니다.
3. IAM Identity Center 활성화에서 활성화를 선택합니다.

   > **Note**
   >
   > IAM Identity Center는 AWS Organizations 설정이 필요합니다. 조직을 설정하지 않은 경우, AWS에서 대신 생성하도록 선택할 수 있습니다. AWS 조직 생성을 선택하여 프로세스를 완료합니다.

4. AWS Organizations으로 활성화를 선택합니다.
5. 계속을 선택합니다.
6. (선택 사항) 이 조직 인스턴스와 연결하고자 하는 태그를 추가합니다.
7. (선택 사항) 위임 관리를 구성합니다.

   > **Note**
   >
   > 다중 계정을 사용하는 환경인 경우, 위임 관리를 구성하는 것을 권장합니다. 위임 관리를 사용하면 AWS Organizations의 관리 계정에 액세스해야 하는 사람의 수를 제한할 수 있습니다. 자세한 내용은 Delegated administration을 참조하세요.

8. 저장을 선택합니다.

AWS Organizations에서 관리 계정과 연결된 주소로 확인 이메일을 자동으로 전송합니다. 확인 이메일을 받기까지 어느 정도 시간이 걸릴 수 있습니다. 확인 이메일이 만료되기 전에 24시간 내에 이메일 주소를 확인해야 합니다.

AWS Identity and Access Management Identity Center 페더레이션 구성

회사 디렉터리 사용자와 함께 Storage Browser for S3를 사용하려면 외부 ID 제공업체(idP)를 사용하도록 IAM Identity Center를 구성해야 합니다. 원하는 ID 제공업체를 사용할 수 있습니다. 그러나 각 ID 제

공업체는 서로 다른 구성 설정을 사용한다는 것을 유의하세요. 각기 다른 외부 ID 제공업체 사용에 대한 자습서는 IAM Identity Center source tutorials을 참조하세요.

> **Note**
>
> 이후 단계에서 참조해야 하므로 구성한 ID 제공업체의 발급자 URL과 대상 속성을 기록해 두세요. IdP를 구성하는 데 필요한 액세스 권한이나 권한이 없는 경우 외부 IdP의 관리자에게 문의해 받아야 할 수 있습니다.

AWS Identity and Access Management Identity Center 콘솔에 신뢰할 수 있는 토큰 발급자 추가

신뢰할 수 있는 토큰 발급자는 AWS Identity and Access Management Identity Center의 외부 ID 제공업체를 나타내며 애플리케이션의 인증된 사용자에 대한 토큰을 인식합니다. AWS Organizations의 IAM Identity Center 인스턴스 계정 소유자는 다음 단계를 수행해야 합니다. 이러한 단계는 IAM Identity Center 콘솔에서 또는 프로그래밍 방식으로 수행할 수 있습니다.

AWS Identity and Access Management Identity Center 콘솔에서 신뢰할 수 있는 토큰 발급자를 추가하려면 다음 단계를 수행합니다.

1. IAM Identity Center 콘솔을 엽니다.
2. 설정을 선택합니다.
3. 인증 탭을 선택하세요.
4. 신뢰할 수 있는 토큰 발급자 섹션으로 이동하여 다음 세부 정보를 입력합니다.
    1. 발급자 URL에 신뢰할 수 있는 토큰 발급자 역할을 하는 외부 IdP의 URL을 입력합니다. 이 정보를 얻으려면 외부 IdP의 관리자에게 문의해야 할 수 있습니다. 자세한 내용은 Using applications with a trusted token issuer를 참조하세요.
    2. 신뢰할 수 있는 토큰 발급자 이름에 신뢰할 수 있는 토큰 발급자의 이름을 입력합니다. 이 이름은 애플리케이션 리소스가 자격 증명 전파를 위해 구성된 경우 8단계에서 선택할 수 있는 신뢰할 수 있는 토큰 발급자 목록에 표시됩니다.
5. 속성 매핑을 원하는 애플리케이션 속성으로 업데이트합니다. 여기서 각 ID 제공업체 속성은 IAM Identity Center 속성에 매핑됩니다. 예를 들어 애플리케이션 속성 `email`을 IAM Identity Center 사용자 속성 `email`에 매핑하고 싶을 수 있습니다. IAM Identity Center에서 허용되는 사용자 속성 목록을 보려면 Attribute mappings for AWS Managed Microsoft AD directory의 표를 참조하세요.
6. (선택 사항) 리소스 태그를 추가하려면 키와 값 페어를 입력합니다. 여러 리소스 태그를 추가하려면 새 태그 추가를 선택하여 새 항목을 생성하고 키 및 값 페어를 입력합니다.

7. 신뢰할 수 있는 토큰 발급자 생성을 선택합니다.
8. 신뢰할 수 있는 토큰 발급자 생성을 완료한 후에는 애플리케이션 관리자에게 문의하여 신뢰할 수 있는 토큰 발급자의 이름을 알리면 신뢰할 수 있는 토큰 발급자가 해당 콘솔에 표시되는지 확인할 수 있습니다.
9. 신뢰할 수 있는 ID 전파에 대해 구성된 애플리케이션에서 사용자가 애플리케이션에 액세스할 수 있도록 애플리케이션 관리자가 해당 콘솔에서 이 신뢰할 수 있는 토큰 발급자를 선택해야 합니다.

bootstrap 애플리케이션 및 **identity bearer**에 대한 IAM 역할 생성

bootstrap 애플리케이션과 identity bearer 사용자가 서로 올바르게 작업할 수 있도록 두 개의 IAM 역할을 생성해야 합니다. IAM 역할 하나는 bootstrap 애플리케이션에 필요하고, 다른 하나는 ID 보유자 또는 S3 Access Grants를 통해 액세스를 요청하는 웹 애플리케이션에 액세스하는 최종 사용자에게 사용해야 합니다. bootstrap 애플리케이션은 ID 제공업체가 발급한 토큰을 수신하고 CreateTokenWithIAM API를 간접 호출하여 이 토큰을 Identity Center에서 발급한 토큰과 교환합니다.

다음과 같은 권한을 사용하여 bootstrap-role과 같은 IAM 역할을 생성합니다. 다음 예시 IAM 정책은 bootstrap-role에 토큰 교환을 수행할 수 있는 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Action": [
            "sso-oauth:CreateTokenWithIAM",
        ],
        "Resource": "arn:${Partition}:sso::${AccountId}:application/${InstanceId}/
${ApplicationId}",
        "Effect": "Allow"
    },
    {
        "Action": [
            "sts:AssumeRole",
            "sts:SetContext"
        ],
        "Resource": "arn:aws:iam::${AccountId}:role/identity-bearer-role",
        "Effect": "Allow"
    }]
}
```

그런 다음 ID 브로커가 IAM 자격 증명을 생성하는 데 사용하는 두 번째 IAM 역할(예: `identity-bearer-role`)을 생성합니다. ID 브로커에서 웹 애플리케이션으로 반환되는 IAM 자격 증명은 Storage Browser for S3 구성 요소에서 S3 데이터에 대한 액세스를 허용하는 데 사용됩니다.

```
{
    "Action": [
        "s3:GetDataAccess",
        "s3:ListCallerAccessGrants",
    ],
    "Resource": "arn:${Partition}:s3:${Region}:${Account}:access-grants/default",
    "Effect": "Allow"
}
```

이 IAM 역할(`identity-bearer-role`)은 다음 문과 함께 신뢰 정책을 사용해야 합니다.

```
{
   "Effect": "Allow",
   "Principal": {
      "AWS": "arn:${Partition}:iam::${Account}:role/${RoleNameWithPath}"
   },
   "Action": [
       "sts:AssumeRole",
       "sts:SetContext"
   ]
}
```

IAM Identity Center에서 애플리케이션 생성 및 구성

> **Note**
>
> 시작하기 전에 이전 단계에서 필요한 IAM 역할을 생성했는지 확인합니다. 이 단계에서는 이러한 IAM 역할 중 하나를 지정해야 합니다.

AWS IAM Identity Center에서 고객 관리형 애플리케이션을 생성하고 구성하려면 다음 단계를 수행합니다.

1. IAM Identity Center 콘솔을 엽니다.
2. [Applications]를 선택합니다.
3. 고객 관리형 탭을 선택합니다.

4. 애플리케이션 추가를 선택합니다.
5. 애플리케이션 유형 선택 페이지의 설정 기본 설정에서 설정하고자 하는 애플리케이션이 있습니다를 선택합니다.
6. 애플리케이션 유형에서 OAuth 2.0을 선택합니다.
7. 다음을 선택합니다. 애플리케이션 지정 페이지가 표시됩니다.
8. 애플리케이션 이름 및 설명 섹션에서 애플리케이션의 표시 이름(예: **`storage-browser-oauth`**)을 입력합니다.
9. 설명(Description)을 입력합니다. 애플리케이션 설명은 IAM Identity Center 콘솔과 API 요청에는 표시되지만 AWS 액세스 포털에는 표시되지 않습니다.
10. 사용자 및 그룹 할당 방법에서 할당 필요 없음을 선택합니다. 이 옵션은 권한이 부여된 모든 IAM Identity Center 사용자 및 그룹이 이 애플리케이션에 액세스할 수 있도록 허용합니다.
11. AWS 액세스 포털에서 사용자가 애플리케이션에 액세스할 수 있는 애플리케이션 URL을 입력합니다.
12. (선택 사항) 리소스 태그를 추가하려면 키와 값 페어를 입력합니다. 여러 리소스 태그를 추가하려면 새 태그 추가를 선택하여 새 항목을 생성하고 키 및 값 페어를 입력합니다.
13. 다음을 선택합니다. 인증 지정 페이지가 표시됩니다.
14. 신뢰할 수 있는 토큰 발급자를 통한 인증에서 확인란을 사용하여 이전에 생성한 신뢰할 수 있는 토큰 발급자를 선택합니다.
15. 선택한 신뢰할 수 있는 토큰 발급자 구성에서 AUD 클레임을 입력합니다. AUD 클레임은 JSON 웹 토큰(JWT)의 대상을 식별하며, 신뢰할 수 있는 토큰 발급자가 이 애플리케이션을 식별하는 이름입니다.

    > **Note**
    >
    > 이 정보를 얻으려면 외부 IdP의 관리자에게 문의해야 할 수 있습니다.

16. 다음을 선택합니다. 인증 자격 증명 지정 페이지가 표시됩니다.
17. 구성 방법에서 IAM 역할 하나 이상 입력을 선택합니다.
18. IAM 역할 입력에서 ID 보유자 토큰의 IAM 역할 또는 Amazon 리소스 이름(ARN)을 추가합니다. 이전 단계에서 ID 브로커 애플리케이션에 대해 생성한 IAM 역할(예: **`bootstrap-role`**)을 입력해야 합니다.
19. 다음을 선택합니다.
20. 검토 및 구성 페이지에서 애플리케이션 구성의 세부 정보를 검토합니다. 설정을 수정해야 하는 경우 편집하고 변경하려는 섹션에서 편집을 선택합니다.

21. 제출을 선택합니다. 방금 추가한 애플리케이션의 세부 정보 페이지가 표시됩니다.

애플리케이션을 설정한 후 사용자는 생성된 권한 세트와 할당된 사용자 액세스에 따라 AWS 액세스 포털 내에서 애플리케이션에 액세스할 수 있습니다.

ID 전파를 위한 신뢰할 수 있는 애플리케이션으로 S3 Access Grants 추가

고객 관리형 애플리케이션을 설정한 후에 ID 전파를 위한 S3 Access Grants를 지정해야 합니다. S3 Access Grants는 사용자가 Amazon S3 데이터에 액세스할 수 있도록 자격 증명을 제공합니다. 고객 관리형 애플리케이션에 로그인하면 S3 Access Grants가 사용자 ID를 신뢰할 수 있는 애플리케이션에 전달합니다.

사전 조건: 다음 단계를 수행하기 전에 S3 Access Grants를 설정했는지 확인합니다(예: S3 Access Grants 인스턴스 생성 및 위치 등록). 자세한 내용은 S3 Access Grants 시작하기를 참조하세요.

고객 관리형 애플리케이션에 ID 전파를 위한 S3 Access Grants를 추가하려면 다음 단계를 수행합니다.

1. IAM Identity Center 콘솔을 엽니다.
2. [Applications]를 선택합니다.
3. 고객 관리형 탭을 선택합니다.
4. 고객 관리형 애플리케이션 목록에서 액세스 요청을 시작하려는 OAuth 2.0 애플리케이션을 선택합니다. 이 애플리케이션은 사용자가 로그인하는 애플리케이션입니다.
5. 세부 정보 페이지의 ID 전파에 신뢰할 수 있는 애플리케이션에서 신뢰할 수 있는 애플리케이션 지정을 선택합니다.
6. 설정 유형에서 개별 애플리케이션 및 액세스 지정을 선택하고, 다음을 선택합니다.
7. 서비스 선택 페이지에서 S3 Access Grants를 선택합니다. S3 Access Grants에는 신뢰할 수 있는 ID 전파를 위한 자체 웹 애플리케이션을 정의하는 데 사용할 수 있는 애플리케이션이 있습니다.
8. 다음을 선택합니다. 다음 단계에서 애플리케이션을 선택합니다.
9. 애플리케이션 선택 페이지에서 개별 애플리케이션을 선택하고, 액세스 요청을 수신할 수 있는 각 애플리케이션의 확인란을 선택한 후 다음을 선택합니다.
10. 액세스 구성 페이지의 구성 방법에서 다음 중 하나를 선택합니다.

    - 애플리케이션별 액세스 선택 - 이 옵션을 선택하여 각 애플리케이션에 대해 다른 액세스 수준을 구성합니다. 액세스 수준을 구성하려는 애플리케이션을 선택하고, 액세스 편집을 선택합니다. 적용할 액세스 수준에서 필요한 정도로 액세스 수준을 변경한 다음 변경 사항 저장을 선택합니다.

- 모든 애플리케이션에 동일한 수준의 액세스 적용 - 애플리케이션별로 액세스 수준을 구성할 필요가 없는 경우 이 옵션을 선택합니다.

11. 다음을 선택합니다.
12. 구성 검토 페이지에서 선택한 항목을 검토합니다.

> **Note**
>
> 사용자에게 `s3:access_grants:read_write` 권한이 부여되었는지 확인해야 합니다. 이 권한을 통해 사용자는 자격 증명을 검색하여 Amazon S3에 액세스할 수 있습니다. 이전에 생성한 IAM 정책 또는 S3 Access Grants를 사용하여 쓰기 작업에 대한 액세스를 제한해야 합니다.

13. 변경하려면 변경하려는 구성 섹션에서 편집을 선택합니다. 그런 다음, 변경해야 하는 부분을 변경하고 변경 사항 저장을 선택합니다.
14. 신뢰할 수 있는 애플리케이션을 선택하여 ID 전파를 위해 신뢰할 수 있는 애플리케이션을 추가합니다.

사용자 또는 그룹에 대한 권한 부여 생성

이 단계에서는 IAM Identity Center를 사용하여 사용자를 프로비저닝합니다. 사용자 및 그룹의 자동 또는 수동 프로비저닝에 SCIM을 사용할 수 있습니다. SCIM은 IAM Identity Center ID를 ID 제공업체(idP)의 ID와 동기화합니다. 여기에는 IdP와 IAM Identity Center 간의 모든 사용자 프로비저닝, 업데이트 및 디프로비저닝이 포함됩니다.

> **Note**
>
> IAM Identity Center에서 S3 Access Grants를 사용하는 경우 로컬 IAM Identity Center 사용자가 사용되지 않기 때문에 이 단계가 필요합니다. 대신 사용자는 ID 제공업체에서 IAM Identity Center와 동기화되어야 합니다.

ID 제공업체의 사용자를 IAM Identity Center와 동기화하려면 다음 단계를 수행합니다.

1. 자동 프로비저닝을 활성화합니다.
2. 액세스 토큰을 생성합니다.

특정 사용 사례에 맞게 IAM Identity Center에서 ID 제공업체를 설정하는 방법에 대한 예는 IAM Identity Center Identity source tutorials을 참조하세요.

Storage Browser for S3 구성 요소 생성

IAM Identity Center 인스턴스를 설정하고 S3 Access Grants에서 권한 부여를 생성한 후 React 애플리케이션을 엽니다. React 애플리케이션에서 `createManagedAuthAdapter`를 사용하여 권한 부여 규칙을 설정합니다. IAM Identity Center에서 획득한 자격 증명을 반환하려면 ID 제공업체를 제공해야 합니다. 그런 다음 `createStorageBrowser`를 직접 호출하여 Storage Browser for S3 구성 요소를 초기화할 수 있습니다.

```
import {
    createManagedAuthAdapter,
    createStorageBrowser,
} from '@aws-amplify/ui-react-storage/browser';
import '@aws-amplify/ui-react-storage/styles.css';

export const { StorageBrowser } = createStorageBrowser({
   config: createManagedAuthAdapter({
    credentialsProvider: async (options?: { forceRefresh?: boolean }) => {
      // return your credentials object
      return {
        credentials: {
          accessKeyId: 'my-access-key-id',
          secretAccessKey: 'my-secret-access-key',
          sessionToken: 'my-session-token',
          expiration: new Date(),
        },
      }
    },
    // AWS `region` and `accountId` of the S3 Access Grants Instance.
    region: '',
    accountId: '',
    // call `onAuthStateChange` when end user auth state changes
    // to clear sensitive data from the `StorageBrowser` state
    registerAuthListener: (onAuthStateChange) => {},
  })
});
```

다음으로, 웹 애플리케이션의 JSON 웹 토큰(JWT)을 IAM Identity Center의 IAM 자격 증명과 교환하는 메커니즘을 생성합니다. JWT를 교환하는 방법에 대한 자세한 내용은 다음 리소스를 참조하세요.

- AWS Storage Blog의 How to develop a user-facing data application with IAM Identity Center and S3 Access Grants 게시물
- AWS Storage Blog의 Scaling data access with S3 Access Grants 게시물
- AWS workshop studio의 S3 Access Grants workshop
- GitHub의 S3 Access Grants workshop

그런 다음 자격 증명 가져오기 요청을 처리하도록 API 엔드포인트를 설정합니다. JSON 웹 토큰(JWT) 교환을 검증하려면 다음 단계를 수행합니다.

1. 수신 요청에 대해 권한 부여 헤더에서 JSON 웹 토큰을 검색합니다.
2. 지정된 JSON 웹 키 세트(JWKS) URL의 퍼블릭 키를 사용하여 토큰을 검증합니다.
3. 토큰의 만료, 발급자, 제목 및 대상 클레임을 확인합니다.

ID 제공업체의 JSON 웹 토큰을 AWS IAM 자격 증명과 교환하려면 다음 단계를 수행합니다.

> **Tip**
>
> 민감한 정보가 로깅되지 않도록 해야 합니다. 누락된 권한 부여, 만료된 토큰 및 기타 예외에 대한 오류 처리 제어를 사용하는 것이 좋습니다. 자세한 내용은 AWS Compute Blog의 Implementing AWS Lambda error handling patterns 게시물을 참조하세요.

1. 필요한 권한 및 범위 파라미터가 요청에 제공되었는지 확인합니다.
2. CreateTokenWithIAM API를 사용하여 JSON 웹 토큰을 IAM Identity Center 토큰으로 교환합니다.

   > **Note**
   >
   > IdP JSON 웹 토큰을 사용한 후에는 다시 사용할 수 없습니다. IAM Identity Center와 교환하려면 새 토큰을 사용해야 합니다.

3. AssumeRole API 작업을 사용하여 IAM Identity Center 토큰을 사용하여 임시 역할을 수임합니다. 자격 증명을 요청하려면 자격 증명 컨텍스트(예: **`identity-bearer-role`**)를 포함하는 역할이라고도 하는 ID 보유자 역할을 사용해야 합니다.
4. IAM 자격 증명을 웹 애플리케이션에 반환합니다.

> **Note**
>
> 적절한 로깅 메커니즘을 설정했는지 확인합니다. 응답은 적절한 HTTP 상태 코드와 함께 표준화된 JSON 형식으로 반환됩니다.

Storage Browser for S3 구성

S3 버킷에 대한 Storage Browser for S3 액세스를 허용하기 위해 Storage Browser 구성 요소는 Amazon S3에 대한 REST API 직접 호출을 수행합니다. 기본적으로 교차 오리진 리소스 공유(CORS)는 S3 버킷에서 활성화되지 않습니다. 따라서 Storage Browser가 데이터에 액세스하는 각 S3 버킷에 대해 CORS를 활성화해야 합니다.

예를 들어 S3 버킷에서 CORS를 활성화하려면 다음과 같이 CORS 정책을 업데이트하면 됩니다.

```
[
    {
        "ID": "S3CORSRuleId1",
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "GET",
            "HEAD",
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": [
            "last-modified",
            "content-type",
            "content-length",
            "etag",
            "x-amz-version-id",
            "x-amz-request-id",
            "x-amz-id-2",
            "x-amz-cf-id",
            "x-amz-storage-class",
```

```
            "date",
            "access-control-expose-headers"
        ],
        "MaxAgeSeconds": 3000
    }
]
```

Storage Browser for S3 문제 해결

Storage Browser for S3에 문제가 있는 경우 다음 문제 해결 팁을 검토해야 합니다.

- 여러 요청에 동일한 토큰(`idToken` 또는 `accessToken`)을 사용하지 마세요. 토큰은 재사용할 수 없습니다. 그러면 요청 실패가 발생합니다.
- Storage Browser 구성 요소에 제공하는 IAM 자격 증명에 `s3:GetDataAccess` 작업을 간접 호출할 수 있는 권한이 포함되어 있는지 확인합니다. 포함되어 있지 않다면 최종 사용자가 데이터에 액세스할 수 없습니다.

또는 다음 리소스를 확인할 수 있습니다.

- Storage Browser for S3는 AWS Support에서 지원합니다. 도움이 필요한 경우 AWS Support Center에 문의하세요.
- Storage Browser for S3에 문제가 있거나 피드백을 제출하려면 Amplify GitHub 페이지를 방문하세요.
- 이 프로젝트에서 잠재적인 보안 문제를 발견한 경우 AWS 취약성 보고 페이지를 통해 AWS Security에 알려주시기 바랍니다.

Amazon S3 Transfer Acceleration을 사용하여 빠르고 안전한 파일 전송 구성

Amazon S3 Transfer Acceleration은 클라이언트와 S3 범용 버킷 간의 장거리 파일 전송을 파일을 빠르고 쉽고 안전하게 전송할 수 있는 버킷 수준 기능입니다. Transfer Acceleration은 전 세계에서 S3 범용 버킷으로 전송 속도를 최적화하도록 설계되었습니다. Transfer Acceleration은 Amazon CloudFront에서 전 세계에 분산된 엣지 로케이션을 활용합니다. 엣지 로케이션에 도착한 데이터는 최적화된 네트워크 경로를 통해 Amazon S3로 라우팅됩니다.

Transfer Acceleration을 사용하면 추가 데이터 전송 요금이 적용될 수 있습니다. 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

Transfer Acceleration을 사용하는 이유는 무엇입니까?

범용 버킷에서 Transfer Acceleration을 사용하는 이유는 다음과 같이 다양합니다.

- 전 세계 각지에서 중앙의 범용 버킷으로 업로드하는 고객이 있는 경우
- 전 세계에 정기적으로 수 기가바이트에서 수 테라바이트의 데이터를 전송할 경우
- Amazon S3에 업로드할 때 인터넷을 통해 사용 가능한 대역폭을 충분히 활용할 수 없는 경우

Transfer Acceleration을 사용하는 경우에 대한 자세한 내용은 Amazon S3 FAQ를 참조하십시오.

Transfer Acceleration을 사용하기 위한 요구 사항

S3 버킷에서 Transfer Acceleration을 사용하는 경우 다음이 필요합니다.

- Transfer Acceleration은 가상 호스팅 방식 요청에서만 지원됩니다. 가상 호스팅 스타일 요청에 대한 자세한 내용은 Amazon S3 API 참조의 REST API를 사용하여 요청을 참조하세요.
- Transfer Acceleration에 사용되는 버킷의 이름은 DNS를 따라야 하며 마침표(".")를 포함할 수 없습니다.
- 버킷에서 Transfer Acceleration을 사용 설정해야 합니다. 자세한 내용은 S3 Transfer Acceleration 사용 설정 및 사용 섹션을 참조하세요.

  버킷에서 Transfer Acceleration을 사용 설정하면 버킷으로의 데이터 전송 속도가 증가하기까지 최대 20분이 걸릴 수 있습니다.

> **Note**
>
> 현재 다음 리전에 위치한 버킷에 Transfer Acceleration이 지원됩니다.
>
> - 아시아 태평양(도쿄)(ap-northeast-1)
> - 아시아 태평양(서울)(ap-northeast-2)
> - 아시아 태평양(뭄바이)(ap-south-1)
> - 아시아 태평양(싱가포르)(ap-southeast-1)
> - 아시아 태평양(시드니)(ap-southeast-2)
> - 캐나다(중부)(ca-central-1)
> - 유럽(프랑크푸르트)(eu-central-1)
> - 유럽(아일랜드)(eu-west-1)
> - 유럽(런던) (eu-west-2)

  - 유럽(파리)(eu-west-3)
  - 남아메리카(상파울루)(sa-east-1)
  - 미국 동부(버지니아 북부)(us-east-1)
  - 미국 동부(오하이오)(us-east-2)
  - 미국 서부(캘리포니아 북부) (us-west-1)
  - 미국 서부(오리건)(us-west-2)

- Transfer Acceleration이 사용 설정된 버킷에 액세스하려면 *bucket-name*`.s3-accelerate.amazonaws.com` 엔드포인트를 사용해야 합니다. 또는 듀얼 스택 엔드포인트 *bucket-name*`.s3-accelerate.dualstack.amazonaws.com`에서 IPv6을 통해 해당 버킷에 액세스해야 합니다. 표준 데이터 전송을 위해 일반 엔드포인트를 계속 사용할 수 있습니다.
- 전송 가속 상태를 설정하려면 버킷 소유자여야 합니다. 버킷 소유자는 다른 사용자에게 버킷에 가속 상태를 설정할 수 있는 권한을 할당할 수 있습니다. 이 `s3:PutAccelerateConfiguration` 권한은 사용자가 버킷에서 Transfer Acceleration을 사용 설정하거나 사용 중지할 수 있도록 허용합니다. `s3:GetAccelerateConfiguration` 권한은 사용자가 버킷의 Transfer Acceleration 상태(`Enabled` 또는 `Suspended`.)를 반환할 수 있도록 허용합니다.

다음 섹션에서는 Amazon S3 Transfer Acceleration을 시작하고 데이터 전송에 사용하는 방법에 대해 설명합니다.

주제

- Amazon S3 Transfer Acceleration 시작하기
- S3 Transfer Acceleration 사용 설정 및 사용
- Amazon S3 Transfer Acceleration 속도 비교 도구 사용

Amazon S3 Transfer Acceleration 시작하기

Amazon S3 Transfer Acceleration을 사용하면 클라이언트와 S3 버킷 간에 파일을 빠르고 쉽고 안전하게 장거리 전송할 수 있습니다. Transfer Acceleration은 전 세계에 분산된 Amazon CloudFront의 엣지 로케이션을 데이터 전송에 사용합니다. 엣지 로케이션에 도착한 데이터는 최적화된 네트워크 경로를 통해 Amazon S3로 라우팅됩니다.

Amazon S3 Transfer Acceleration을 사용하려면 다음 단계를 수행합니다.

1. 버킷에서 Transfer Acceleration 사용 설정

버킷에서 다음 방법 중 하나를 사용하여 버킷에서 Transfer Acceleration을 설정할 수 있습니다.

- Amazon S3 콘솔을 사용합니다.
- REST API PutBucketAccelerateConfiguration 작업을 사용합니다.
- AWS CLI 및 AWS SDK를 사용합니다. 자세한 내용은 Amazon S3 API 참조의 AWS SDK를 사용하여 Amazon S3에서 개발을 참조하세요.

자세한 내용은 S3 Transfer Acceleration 사용 설정 및 사용 섹션을 참조하세요.

> **Note**
>
> 버킷에서 Transfer Acceleration이 작동하려면 버킷 이름이 DNS 이름 지정 요구 사항을 준수해야 하며 마침표(.)를 포함하면 안 됩니다.

2. 가속화가 사용 설정된 버킷의 데이터 전송

다음 `s3-accelerate` 엔드포인트 도메인 이름 중 하나를 사용합니다.

- 가속화가 사용 설정된 버킷에 액세스하려면 *`bucket-name`*`.s3-accelerate.amazonaws.com`을 사용합니다.
- IPv6를 통해 가속화가 사용 설정된 버킷에 액세스하려면 *`bucket-name`*`.s3-accelerate.dualstack.amazonaws.com`을 사용합니다.

  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 Amazon S3 API 참조의 IPv6을 통해 Amazon S3에 요청 및 Amazon S3 API 참조의 Amazon S3 듀얼 스택 엔드포인트 사용을 참조하세요.

> **Note**
>
> 데이터 전송 애플리케이션은 더 빠른 데이터 전송을 위해 다음 두 가지 유형의 엔드포인트 중 하나를 사용하여 버킷에 액세스해야 합니다. 듀얼 스택 엔드포인트의 경우 `.s3-accelerate.amazonaws.com` 또는 `.s3-accelerate.dualstack.amazonaws.com`입니다. 표준 데이터 전송을 사용하려는 경우 일반 엔드포인트를 계속 사용하면 됩니다.

Transfer Acceleration을 사용 설정한 후에는 Amazon S3 PUT 객체 및 GET 객체 요청의 대상을 s3-accelerate 엔드포인트 도메인 이름으로 지정할 수 있습니다. 예를 들어 PUT 요청에 *amzn-s3-demo-bucket*.s3.us-east-1.amazonaws.com 호스트 이름을 사용하는 PutObject를 사용하는 REST API 애플리케이션이 있다고 가정합니다. PUT 요청을 가속화하려면 요청의 호스트 이름을 *amzn-s3-demo-bucket*.s3-accelerate.amazonaws.com으로 변경합니다. 표준 업로드 속도로 돌아가려면 이름을 다시 *amzn-s3-demo-bucket*.s3.us-east-1.amazonaws.com으로 변경합니다.

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 *bucket-name*.s3-accelerate.amazonaws.com(으)로 명시적으로 설정할 필요가 없습니다. 가속 엔드포인트 클라이언트 구성 플래그를 사용하는 방법의 예를 보려면 S3 Transfer Acceleration 사용 설정 및 사용 단원을 참조하십시오.

전송 가속 엔드포인트에서 다음을 제외한 모든 Amazon S3 작업을 수행할 수 있습니다.

- ListBuckets
- CreateBucket
- DeleteBucket

또한 Amazon S3 Transfer Acceleration은 CopyObject를 사용하는 교차 리전 복사본을 지원하지 않습니다.

S3 Transfer Acceleration 사용 설정 및 사용

Amazon S3 Transfer Acceleration을 사용하여 클라이언트와 S3 범용 버킷 간의 장거리 파일 전송을 빠르고 안전하게 수행할 수 있습니다. S3 콘솔, AWS Command Line Interface(AWS CLI), API 또는 AWS SDK를 사용하여 Transfer Acceleration을 사용할 수 있습니다.

이 섹션에서는 버킷에서 Amazon S3 Transfer Acceleration을 설정하는 방법과 설정된 버킷에 대해 가속 엔드포인트를 사용하는 방법의 예를 설명합니다.

Transfer Acceleration 요구 사항에 대한 자세한 내용은 Amazon S3 Transfer Acceleration을 사용하여 빠르고 안전한 파일 전송 구성 섹션을 참조하세요.

S3 콘솔 사용

> **Note**
>
> 가속화된 업로드 속도와 가속화되지 않은 업로드 속도를 비교하려면 Amazon S3 Transfer Acceleration 속도 비교 도구를 엽니다.
> 이 속도 비교 도구는 멀티파트 업로드를 통해 Amazon S3 Transfer Acceleration을 사용하거나 사용하지 않으면서 브라우저에서 여러 AWS 리전으로 파일을 전송합니다. 직접 업로드의 업로드 속도를 비교하고 리전별로 가속화된 업로드를 전송할 수 있습니다.

S3 범용 버킷에 전송 가속화를 활성화하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 범용 버킷 목록에서 전송 가속화를 활성화하려는 버킷의 이름을 선택합니다.
4. 속성을 선택합니다.
5. Transfer Acceleration에서 편집(Edit)을 선택합니다.
6. 사용 설정를 선택하고 변경 사항 저장을 선택합니다.

가속화된 데이터 전송에 액세스

1. Amazon S3의 버킷에 Transfer Acceleration을 사용 설정한 후 버킷의 [속성(Properties)] 탭을 확인합니다.
2. Transfer Acceleration에서 가속 엔드포인트(Accelerated endpoint)는 버킷의 Transfer Acceleration 엔드포인트를 표시합니다. 이 엔드포인트를 사용하여 버킷과 주고받을 때 가속화된 데이터 전송에 액세스할 수 있습니다.

   Transfer Acceleration을 중지하면 가속 엔드포인트는 더 이상 작동하지 않습니다.

AWS CLI 사용

다음은 Transfer Acceleration에 사용되는 AWS CLI 명령의 예입니다. AWS CLI 설정에 대한 지침은 Amazon S3 API 참조의 AWS CLI를 사용하여 Amazon S3에서 개발을 참조하세요.

버킷에서 Transfer Acceleration 사용 설정

AWS CLI put-bucket-accelerate-configuration 명령을 사용하여 버킷에서 Transfer Acceleration을 활성화하거나 일시 중지합니다.

다음은 *amzn-s3-demo-bucket*이라는 버킷에서 Transfer Acceleration을 활성화하기 위해 `Status=Enabled`를 설정하는 예제입니다. Transfer Acceleration을 일시 중지하려면 `Status=Suspended`를 사용합니다.

Example

```
$ aws s3api put-bucket-accelerate-configuration --bucket amzn-s3-demo-bucket --
accelerate-configuration Status=Enabled
```

Transfer Acceleration 활성화

`s3` 및 `s3api` AWS CLI 명령으로 구성된 모든 Amazon S3 요청을 가속 엔드포인트 `s3-accelerate.amazonaws.com`으로 보낼 수 있습니다. 이렇게 하려면 AWS Config 파일의 프로파일에서 구성 값 `use_accelerate_endpoint`를 `true`로 설정합니다. 가속 엔드포인트를 사용하려면 버킷에서 Transfer Acceleration을 사용하도록 설정해야 합니다.

모든 요청은 가상의 버킷 주소 지정 방식(*amzn-s3-demo-bucket*`.s3-accelerate.amazonaws.com`)을 사용하여 전송됩니다. `ListBuckets`, `CreateBucket` 및 `DeleteBucket` 요청은 엔드포인트에서 이러한 작업을 지원하지 않으므로 가속 엔드포인트로 전송되지 않습니다.

`use_accelerate_endpoint`에 대한 자세한 내용은 AWS CLI 명령 참조의 AWS CLI S3 구성을 참조하세요.

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.

- `--endpoint-url` 파라미터를 `https://s3-accelerate.amazonaws.com`으로 설정하여 `s3` 또는 `s3api` 명령에 가속 엔드포인트를 사용합니다.
- AWS Config 파일에 별도의 프로파일을 설정합니다. 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 Acceleration이 사용 설정된 버킷에 객체 업로드

다음은 가속 엔드포인트를 사용하도록 구성된 기본 프로파일을 사용하여 Transfer Acceleration이 설정된 *amzn-s3-demo-bucket*이라는 버킷에 파일을 업로드하는 예제입니다.

Example

```
$ aws s3 cp file.txt s3://amzn-s3-demo-bucket/key-name --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://amzn-s3-demo-bucket/key-name --region region --endpoint-url
 https://s3-accelerate.amazonaws.com
```

AWS SDK 사용

다음은 Transfer Acceleration을 사용하여 AWS SDK를 통해 Amazon S3에 객체를 업로드하는 예제입니다. 일부 AWS SDK 지원 언어(예: Java, .NET)는 가속 엔드포인트 클라이언트 구성 플래그를 사용하므로, Transfer Acceleration의 엔드포인트를 *bucket-name*`.s3-accelerate.amazonaws.com`으로 명시적으로 설정할 필요가 없습니다.

Java

Example

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

- 가속 엔드포인트를 사용하도록 구성된 AmazonS3Client를 만듭니다. 클라이언트가 액세스하는 모든 버킷에 Transfer Acceleration이 사용 설정되어 있어야 합니다.
- 지정된 버킷에 대해 Transfer Acceleration을 사용 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 대한 자세한 내용은 Amazon S3 Transfer Acceleration 시작하기 단원을 참조하십시오. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 AWS SDK for Java 개발자 안내서의 시작하기 섹션을 참조하세요.

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

.NET

다음 예제에서는 AWS SDK for .NET를 사용하여 버킷에서 Transfer Acceleration을 사용하는 방법을 보여 줍니다. 코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

Example

```
using Amazon;
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 사용 설정된 버킷에 객체를 업로드하는 경우 클라이언트를 생성할 때 가속 엔드포인트 사용을 지정합니다.

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

JavaScript

AWS SDK for JavaScript를 사용하여 Transfer Acceleration을 활성화하는 예제는 AWS SDK for JavaScript API 참조의 PutBucketAccelerateConfiguration 명령을 참조하세요.

Python(Boto)

SDK for Python을 사용하여 Transfer Acceleration을 활성화하는 예제는 AWS SDK for Python(Boto3) API 참조의 put_bucket_accelerate_configuration 섹션을 참조하세요.

Other

다른 AWS SDK 사용에 대한 자세한 내용은 샘플 코드 및 라이브러리 단원을 참조하세요.

REST API 사용

REST API `PutBucketAccelerateConfiguration` 작업을 사용하여 기존 버킷에서 구성을 가속화할 수 있습니다.

자세한 내용은 Amazon Simple Storage Service API 참조에서 PutBucketAccelerateConfiguration를 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. 이 속도 비교 도구는 Transfer Acceleration을 사용하거나 사용하지 않고 멀티파트 업로드를 통해 브라우저에서 여러 Amazon S3 리전으로 파일을 전송합니다.

속도 비교 도구는 다음과 같은 방법으로 이용할 수 있습니다.

- 다음 URL을 브라우저 창에 복사합니다. *region*을 현재 사용하는AWS 리전(예: us-west-2)으로 바꾸고, *amzn-s3-demo-bucket*을 평가하려는 버킷의 이름으로 바꿉니다.

  ```
  https://s3-accelerate-speedtest.s3-accelerate.amazonaws.com/en/
  accelerate-speed-comparsion.html?region=region&origBucketName=amzn-s3-
  demo-bucket
  ```

  Amazon S3에서 지원하는 리전 목록은 AWS 일반 참조의 Amazon S3 엔드포인트 및 할당량을 참조하세요.
- Amazon S3 콘솔을 사용합니다.

스토리지 전송 및 사용량에 대한 요청자 지불 범용 버킷 사용

일반적으로 버킷과 연결된 모든 Amazon S3 스토리지 및 데이터 전송 비용은 버킷 소유자가 지불합니다. 그러나 범용 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 버킷을 요청자 지불 버킷으로 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**
>
> 범용 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 수임할 때는 해당 역할이 속한 계정에 요청에 대한 요금이 부과됩니다. IAM 역할에 대한 자세한 내용은 IAM 사용 설명서의 IAM 역할 단원을 참조하세요.

버킷을 요청자 지불 버킷으로 구성한 후에는 요청자가 요청 및 데이터 다운로드에 요금이 부과된다는 사실을 이해한다는 의사를 표시해야 합니다. 요금 부과에 동의를 표하려면 요청자는 DELETE, GET, HEAD, POST 및 PUT 요청에 대한 API 요청에 `x-amz-request-payer`를 헤더로 포함하거나 REST 요청에 `RequestPayer` 파라미터를 추가해야 합니다. CLI 요청의 경우 요청자는 `--request-payer` 파라미터를 사용할 수 있습니다.

Example - 객체 삭제 시 요청자 지불 사용

다음 DeleteObjectVersion API 예제를 사용하려면 *`user input placeholders`*를 실제 정보로 대체합니다.

```
DELETE /Key+?versionId=VersionId HTTP/1.1
Host: Bucket.s3.amazonaws.com
x-amz-mfa: MFA
x-amz-request-payer: RequestPayer
x-amz-bypass-governance-retention: BypassGovernanceRetention
x-amz-expected-bucket-owner: ExpectedBucketOwner
```

요청자가 RestoreObject API를 사용하여 객체를 복원하는 경우, 요청에 `x-amz-request-payer` 헤더 또는 `RequestPayer` 파라미터가 있는 한 요청자 지불이 지원되지만 요청자는 요청 비용만 지불합니다. 버킷 소유자가 검색 요금을 지불합니다.

요청자 지불 버킷은 다음을 지원하지 않습니다.

- 익명 요청
- SOAP 요청
- 요청자 지불 버킷을 최종 사용자 로깅의 대상 버킷으로 사용할 수 있으며, 그 반대의 경우도 마찬가지입니다. 그러나 대상 버킷이 요청자 지불 버킷이 아니면 요청자 지불 버킷에서 최종 사용자 로깅을 설정할 수 있습니다.

요청자 지불 요금의 방식

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

- 이 요청은 `AccessDenied`(HTTP `403 Forbidden`) 오류를 반환하고 버킷 소유자의 개별 AWS 계정 또는 AWS 조직 내에서 시작됩니다.

- 요청이 SOAP 요청인 경우

요청자 지불액에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- 버킷에서 요청자 지불 구성
- REST API를 사용하여 requestPayment 구성 검색
- 요청자 지불 버킷에서 객체 다운로드

버킷에서 요청자 지불 구성

Amazon S3 버킷을 요청자 지불 버킷으로 구성하여 요청자가 버킷 소유자 대신 요청 및 데이터 다운로드 비용을 지불하도록 할 수 있습니다.

이 섹션에서는 콘솔과 REST API를 사용하여 Amazon S3 버킷에서 요청자 지불을 구성하는 방법에 대한 예제를 제공합니다.

S3 콘솔 사용

S3 범용 버킷에 대한 요청자 지불을 활성화하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 범용 버킷 목록에서 요청자 지불을 활성화할 버킷의 이름을 선택합니다.
4. [속성(Properties)]을 선택합니다.
5. 요청자 지불에서 편집을 선택합니다.
6. 활성화를 선택하고 변경 사항 저장을 선택합니다.

   Amazon S3는 버킷에 대한 요청자 지불을 활성화하고 버킷 개요를 표시합니다. 요청자 지불(Requester pays) 아래에 활성화(Enabled)가 표시됩니다.

REST API 사용

버킷 소유자만 버킷의 `RequestPaymentConfiguration.payer` 구성 값을 `BucketOwner`(기본값) 또는 `Requester`(으)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 BucketOwner를 사용하며 버킷에 객체를 게시하기 전에 이 값을 Requester로 설정합니다.

requestPayment를 설정하려면

- PUT 요청을 사용하여 지정된 버킷에 대해 Payer 값을 Requester로 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 있습니다. 버킷 내의 특정 객체에 대해서는 요청자 지불을 설정할 수 없습니다.

언제든지 버킷을 BucketOwner 또는 Requester로 구성할 수 있습니다. 그러나 새 구성 값이 적용되기까지 몇 분 정도 걸릴 수 있습니다.

> **Note**
>
> 미리 서명된 URL을 제공하는 버킷 소유자는 버킷을 요청자 지불로 구성하기 전에 신중하게 고려해야 합니다. 특히 수명 주기가 긴 URL의 경우에는 더욱 신중해야 합니다. 요청자가 버킷 소

유자의 자격 증명을 사용하는 미리 서명된 URL을 사용할 때마다 버킷 소유자에게 요금이 부과됩니다.

REST API를 사용하여 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 Requester로 설정되었음을 보여 줍니다.

요청자 지불 버킷에서 객체 다운로드

요청자 지불 버킷에서 데이터를 다운로드하면 요청자가 요금을 지불하기 때문에, 요청자는 특수 파라미터인 x-amz-request-payer을(를) 포함시킴으로써 요청자에게 해당 다운로드에 대한 요금이 부

과됨을 인지했다고 밝혀야 합니다. 요청자 지불 버킷의 객체를 액세스하려면 요청에 다음 중 하나가 포함되어야 합니다.

- DELETE, GET, HEAD, POST 및 PUT 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 Amazon S3 API 참조의 Using an Authorization Header를 참조하세요.

REST API 사용

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 Amazon Simple Storage Service API Reference의 오류 응답을 참조하세요.

AWS CLI 사용

AWS CLI를 사용하여 요청자 지불액 버킷의 객체를 다운로드하려면 `get-object` 요청의 일부로 `--request-payer requester`를 지정합니다. 자세한 내용은 AWS CLI 참조에서 get-object를 참조하세요.

Amazon S3의 객체 작업

Amazon S3에 데이터를 저장하려면 버킷 및 객체라는 리소스를 사용합니다. 버킷은 객체에 대한 컨테이너입니다. 객체는 파일과 해당 파일을 설명하는 모든 메타데이터입니다.

Amazon S3에 객체를 저장하려면 버킷을 생성한 다음 버킷에 객체를 업로드합니다. 객체가 버킷에 있으면 객체를 열고 다운로드하고 복사할 수 있습니다. 객체 또는 버킷이 더 이상 필요하지 않은 경우 이러한 리소스를 정리할 수 있습니다.

> **Note**
>
> Amazon S3 Express One Zone 스토리지 클래스를 디렉터리 버킷과 함께 사용하는 방법에 대한 자세한 내용은 S3 Express One Zone 및 디렉터리 버킷 작업 섹션을 참조하세요.

> **Important**
>
> Amazon S3 콘솔에서 객체에 대해 Open(열기) 또는 Download As(다운로드 형식)를 선택하면 이러한 작업이 미리 서명된 URL을 생성합니다. 5분 동안 이러한 미리 서명된 URL에 액세스할 수 있는 사람은 누구나 객체에 액세스할 수 있습니다. 미리 서명된 URL에 대한 자세한 내용은 미리 서명된 URL 사용 섹션을 참조하십시오.

Amazon S3에서는 사용한 만큼만 비용을 청구하며, Amazon S3 기능 및 요금에 대한 자세한 내용은 Amazon S3를 참조하십시오. 신규 Amazon S3 고객인 경우 Amazon S3를 무료로 시작할 수 있습니다. 자세한 내용은 AWS 프리 티어를 참조하십시오.

주제

- Amazon S3 객체 개요
- Amazon S3 객체 이름 지정
- 객체 메타데이터 작업
- 객체 업로드
- 조건부 요청을 통해 S3 작업에 사전 조건 추가
- 객체 복사, 이동, 이름 변경
- 객체 다운로드

- Amazon S3에서 객체 무결성 확인
- Amazon S3 객체 삭제
- 개체 구성, 나열 및 작업
- 미리 서명된 URL을 통해 객체 다운로드 및 업로드
- S3 객체 Lambda를 사용하여 객체 변환
- Batch Operations를 사용하여 대량으로 객체 작업 수행
- Amazon S3 Select를 사용하여 데이터 쿼리 적용

Amazon S3 객체 개요

Amazon S3는 고유한 키-값을 사용하여 원하는 수만큼 객체를 저장하는 객체 스토어입니다. 이러한 객체는 하나 이상의 버킷에 저장되며 각 객체의 크기는 5TB까지 가능합니다. 객체는 다음과 같은 요소로 구성됩니다.

키

객체에 할당한 이름입니다. 객체 키를 사용하여 객체를 검색합니다. 자세한 내용은 객체 메타데이터 작업을 참조하세요.

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 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하세요.

값

저장하는 콘텐츠입니다.

객체 값은 임의의 바이트 시퀀스입니다. 객체 크기는 0TB 이상 5TB까지 가능합니다. 자세한 내용은 객체 업로드 섹션을 참조하세요.

Metadata

객체 관련 정보를 저장하기 위한 이름-값 페어의 세트입니다. Amazon S3의 객체에 사용자 정의 메타데이터라고 하는 메타데이터를 지정할 수 있습니다. 또한 Amazon S3는 이러한 객체의 관리에 사용되는 시스템 메타데이터를 객체에 지정합니다. 자세한 내용은 객체 메타데이터 작업을 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 섹션을 참조하세요.

액세스 제어 정보

Amazon S3에 저장하는 객체에 대한 액세스를 제어할 수 있습니다. Amazon S3는 ACL(액세스 제어 목록), 버킷 정책 등의 리소스 기반 액세스 제어와 사용자 기반 액세스 제어를 모두 지원합니다. 액세스 제어에 대한 자세한 내용은 다음을 참조하십시오.

- Amazon S3의 액세스 제어
- Amazon S3의 ID 및 액세스 관리
- ACL 구성

Amazon S3 리소스(예: 버킷, 객체)는 기본적으로 비공개입니다. 명시적으로 권한을 부여해야 다른 사용자가 이러한 리소스에 액세스할 수 있습니다. 객체 공유에 대한 자세한 내용은 미리 서명된 URL을 통해 객체 공유 단원을 참조하십시오.

Tags

태그를 사용하여 액세스 제어 또는 비용 할당을 위해 저장된 객체를 분류할 수 있습니다. 자세한 내용은 태그를 사용하여 스토리지 분류 섹션을 참조하세요.

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 표에는 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 업데이트하

하위 리소스	설명
	려면 기존 ACL을 교체해야 합니다. ACL에 대한 자세한 내용은 ACL(액세스 제어 목록) 개요 단원을 참조하십시오.

Amazon S3 객체 이름 지정

객체 키(또는 키 이름)는 Amazon S3 버킷 내 객체를 고유하게 식별합니다. 객체를 만들 때 키 이름을 지정합니다. 예를 들어, Amazon S3 콘솔에서 버킷을 선택하면 버킷 내 객체의 목록이 표시됩니다. 이러한 이름이 객체 키입니다.

객체 키 이름은 UTF-8로 인코딩된 유니코드 문자 시퀀스로 구성되며, 최대 길이는 1,024바이트 또는 약 1,024개의 라틴 문자입니다. 일부 로캘에서는 인코딩을 위해 단일 문자에 2바이트가 필요할 수 있습니다. 객체의 이름을 지정할 때는 다음 사항에 유의하세요.

- 객체 키 이름은 대/소문자를 구분합니다.
- 객체 키 이름에는 접두사(콘솔에서 폴더라고 함)가 포함됩니다. 예를 들어 `Development/Projects.xls`는 `Development` 접두사(또는 폴더) 내에 있는 `Projects.xls` 객체의 전체 객체 키 이름입니다. 접두사, 구분 기호(/) 및 객체 이름은 객체 키 이름의 1,024바이트 제한에 포함됩니다. 접두사 및 폴더에 대한 자세한 정보는 the section called “객체 키 이름 선택” 섹션을 참조하세요.
- 특정 문자는 객체 키 이름에 사용될 때 특별한 처리가 필요할 수 있습니다. 자세한 내용은 the section called “객체 키 명명 지침” 섹션을 참조하세요.

> **Note**
>
> 값이 `"soap"`인 객체 키 이름은 가상 호스팅 방식 요청에 지원되지 않습니다. `"soap"`이 사용되는 객체 키 이름 값의 경우 경로 스타일 URL을 대신 사용해야 합니다.

객체 키 이름 선택

Amazon S3 데이터 모델은 단순한 구조를 가지고 있습니다. 사용자가 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 추론할 수 있습니다. Amazon S3 콘솔

은 폴더 개념을 지원합니다. Amazon S3 콘솔에서 메타데이터를 편집하는 방법에 대한 자세한 내용은 Amazon S3 콘솔에서 객체 메타데이터 편집 섹션을 참조하십시오.

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/, Private/) 및 구분 기호(/)를 사용하여 폴더 구조를 표현합니다. s3-dg.pdf 키에는 슬래시로 구분된 접두사가 없으므로 해당 객체는 버킷의 루트 수준에 곧바로 표시됩니다. Development/ 폴더를 열면 그 안에 Projects.xlsx 객체가 표시됩니다.

- Amazon S3는 버킷과 객체를 지원하며 계층 구조가 없습니다. 다만 객체 키 이름에 접두사와 구분 기호를 사용하면 Amazon S3 콘솔과 AWS SDK에서 계층 구조를 추론하고 폴더 개념을 도입할 수 있습니다.
- Amazon S3 콘솔은 폴더 접두사 및 구분 기호 값을 키로 사용하여 0바이트 객체를 만들어 폴더 객체 생성을 구현합니다. 이러한 폴더 객체는 콘솔에 표시되지 않습니다. 그렇지 않으면 다른 객체처럼 동작하며 REST API, AWS CLI 및 AWS SDK를 통해 볼 수 있고 조작도 할 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 조합은 일반적으로 키 이름으로 사용해도 문제가 되지 않습니다.

Alphanumeric characters	• 0~9 • a-z • A-Z
Special characters	• 느낌표(!)

- 하이픈(-)
- 밑줄(_)
- 마침표(.)
- 별표(*)
- 작은따옴표(')
- 여는 괄호(()
- 닫는 괄호())

다음은 유효한 객체 키 이름의 예입니다.

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

> **Note**
>
> Amazon S3 콘솔을 사용하여 키 이름이 마침표(.)로 끝나는 객체를 다운로드하는 경우, 다운로드한 객체의 키 이름에서 마침표가 제거됩니다. 다운로드한 객체 이름 끝에 마침표를 유지하려면 AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용해야 합니다.
> 또한 다음 접두사 제한 사항을 숙지해야 합니다.
>
> - 접두사가 `./`인 객체는 AWS CLI, AWS SDK 또는 REST API를 사용하여 업로드하거나 다운로드해야 합니다. Amazon S3 콘솔을 사용하여 이러한 객체를 업로드할 수 없습니다.
> - 상대 경로 요소(예: `../`)가 포함된 객체 키는 왼쪽에서 오른쪽으로 구문 분석할 때 상대 경로 세그먼트의 누적 수가 상대 경로가 아닌 요소의 수를 초과하지 않는 경우 유효합니다. 이 규칙은 Amazon S3 콘솔, Amazon S3 REST API, AWS CLI 및 AWS SDK를 사용하여 이루어진 모든 요청에 적용됩니다.
>
>   예시:
>
>   - `videos/2014/../../video1.wmv`는 유효합니다.
>   - `videos/../../video1.wmv`는 유효하지 않습니다.
>   - `videos/../../2014/video1.wmv`는 유효하지 않습니다.

특별한 처리가 필요한 문자

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

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

피해야 하는 문자

모든 애플리케이션 간에 일관되지 않은 상당한 특수 문자 처리로 인해 다음과 같은 문자는 키 이름에 사용하지 않는 것이 좋습니다.

- 백슬래시(\)
- 왼쪽 중괄호({)
- 인쇄되지 않는 ASCII 문자(128~255 사이의 10진수)
- 캐럿 또는 곡절(^)
- 오른쪽 중괄호(})
- 백분율 문자(%)
- 억음 부호(`)
- 오른쪽 대괄호(])
- 따옴표(")

- 보다 큼 기호(>)
- 왼쪽 대괄호([)
- 물결표(~)
- 보다 작음 기호(<)
- 파운드 기호(#)
- 세로 막대 또는 파이프(|)

XML 관련 객체 키 제한 사항

줄 끝 처리의 XML 표준에서 지정한 대로 모든 XML 텍스트는 단일 캐리지 리턴(ASCII 코드 13) 및 캐리지 리턴 바로 뒤에 오는 줄 바꿈(ASCII 코드 10), 즉 새 줄 바꿈 문자가 단일 줄 바꿈 문자로 대체되도록 정규화됩니다. XML 요청에서 객체 키를 올바르게 구문 분석하려면 캐리지 리턴 및 기타 특수 문자가 XML 태그 내에 삽입될 때 해당 XML 엔터티 코드로 대체되어야 합니다.

다음은 이러한 특수 문자 및 대응하는 XML 엔터티 코드의 목록입니다.

- 아포스트로피(')를 `'`로 바꿔야 함
- 따옴표(")를 `"`로 바꿔야 함
- 앰퍼샌드(&)를 `&`로 바꿔야 함
- 보다 작음 기호(<)를 `<`로 바꿔야 함
- 보다 큼 기호(>)를 `>`로 바꿔야 함
- 캐리지 리턴(\r)을 `
` 또는 `
`로 바꿔야 함
- 줄 바꿈(\n)을 `
` 또는 `
`로 바꿔야 함

Example

다음 예에서는 캐리지 리턴의 대체물로 XML 엔터티 코드를 사용하는 방법을 보여줍니다. 이 `DeleteObjects` 요청은 `key` 파라미터 `/some/prefix/objectwith\rcarriagereturn`으로 객체를 삭제합니다(여기서 `\r`은 캐리지 리턴).

```
<Delete xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Object>
    <Key>/some/prefix/objectwith&#13;carriagereturn</Key>
  </Object>
</Delete>
```

객체 메타데이터 작업

Amazon S3에는 시스템 정의 메타데이터와 사용자 정의 메타데이터라는 두 가지 종류의 객체 메타데이터가 있습니다. 시스템 정의 메타데이터에는 객체의 생성 날짜, 크기 및 스토리지 클래스와 같은 메타데이터가 포함됩니다. 사용자 정의 메타데이터는 객체를 업로드할 때 설정할 수 있는 메타데이터입니다. 이 사용자 정의 메타데이터는 이름-값 페어의 집합입니다. 자세한 내용은 the section called “시스템 정의 객체 메타데이터” 및 the section called “사용자 정의 객체 메타데이터”(을)를 참조하세요.

객체를 만들 때 Amazon S3 버킷 내 객체를 고유하게 식별하는 객체 키(또는 키 이름)을 지정합니다. 자세한 내용은 Amazon S3 객체 이름 지정 섹션을 참조하세요. 또한 객체를 업로드할 때 Amazon S3에서 사용자 정의 메타데이터를 설정할 수 있습니다.

객체를 업로드한 후에는 이 사용자 정의 메타데이터를 수정할 수 없습니다. 이 메타데이터를 수정할 수 있는 유일한 방법은 객체 복사본을 만든 후 메타데이터를 설정하는 것입니다. Amazon S3 콘솔을 사용한 메타데이터 편집에 대한 자세한 내용은 Amazon S3 콘솔에서 객체 메타데이터 편집 섹션을 참조하세요.

S3 메타데이터를 사용하여 메타데이터 쿼리 및 데이터 검색 가속화

S3 객체에 대한 메타데이터를 쉽게 찾고 저장하고 쿼리하려면 S3 메타데이터를 사용할 수 있습니다. S3 메타데이터를 사용하면 비즈니스 분석, 콘텐츠 검색, 인공 지능 및 기계 학습(AI/ML) 모델 훈련 등에 사용할 데이터를 빠르게 준비할 수 있습니다.

S3 메타데이터는 범용 버킷의 객체에 대한 메타데이터를 자동으로 캡처하고, 쿼리할 수 있는 읽기 전용 완전관리형 Apache Iceberg 테이블에 저장하여 데이터 검색을 가속화합니다. 이러한 읽기 전용 테이블을 메타데이터 테이블이라고 합니다. 객체가 범용 버킷에 추가, 업데이트 및 제거되면 S3 메타데이터는 해당 메타데이터 테이블을 자동으로 새로 고쳐 최신 변경 사항을 반영합니다.

기본적으로 S3 메타데이터는 객체의 생성 시간 및 스토리지 클래스와 같은 시스템 정의 객체 메타데이터와 객체 업로드 중에 포함된 태그 및 사용자 정의 메타데이터와 같은 사용자 지정 메타데이터를 제공합니다. 또한 S3 메타데이터는 객체가 업데이트되거나 삭제되는 시기 및 요청을 수행한 AWS 계정과 같은 이벤트 메타데이터를 제공합니다.

메타데이터 테이블은 테이블 형식 데이터에 최적화된 스토리지를 제공하는 S3 테이블 버킷에 저장됩니다. 메타데이터를 쿼리하려면 테이블 버킷을 Amazon Athena, Amazon Redshift, Amazon QuickSight와 같은 AWS 분석 서비스와 통합하면 됩니다.

S3 메타데이터에 대한 자세한 정보는 the section called “데이터 검색 가속화” 섹션을 참조하세요.

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 생성 날짜와 크기 메타데이터를 유지하며 객체 관리의 일환으로 이 정보를 사용합니다.

시스템 메타데이터에는 다음 2가지 카테고리가 있습니다.

- 시스템이 제어함 - 객체 생성 날짜와 같은 메타데이터는 시스템이 제어하므로 Amazon S3만 날짜 값을 수정할 수 있습니다.
- 사용자가 제어함 - 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 버킷의 경우 다른 페이지 또는 외부 URL 중 하나로 페이지 요청을 리디렉션해야 할 때가 있습니다. 이 경우 웹 페이지는 버킷의 객체가 됩니다. Amazon S3는 페이지 리디렉션 값을 시스템 메타데이터로 저장하며, 이것을 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 이해 및 관리를 참조하십시오.

  Amazon S3는 AWS KMS 키를 사용하여 Amazon S3 객체를 암호화합니다. AWS KMS는 객체 데이터만 암호화합니다. 체크섬과 지정된 알고리즘은 객체 메타데이터의 일부로 저장됩니다. 객체에 서버 측 암호화가 요청된 경우 체크섬은 암호화된 형식으로 저장됩니다. 서버 측 암호화에 대한 자세한 정보는 암호화로 데이터 보호 섹션을 참조하십시오.

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

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

이름	설명	사용자의 값 수정 여부
Date	현재 날짜 및 시간.	아니요
Cache-Control	캐싱 정책을 지정하는 데 사용되는 일반 헤더 필드입니다.	예
Content-Disposition	객체 표현 정보입니다.	예
Content-Length	객체 크기(바이트).	아니요
Content-Type	객체 유형입니다.	예
Last-Modified	객체 생성일 또는 최종 수정일 중 최근 날짜. 멀티파트 업로드의 경우 객체 생성 날짜는 멀티파트 업로드 시작 날짜입니다.	아니요
ETag	객체의 특정 버전을 나타내는 엔터티 태그(ETag). 멀티파트 업로드로 업로드되지 않고 암호화되지 않거나 Amazon S3 관리형 키(SSE-S3)를 사용하는 서버 측 암호화를 통해 암호화된 객체의 경우 ETag는 데이터의 MD5 다이제스트입니다.	아니요
x-amz-server-side-encryption	객체에 대한 서버 측 암호화 사용 여부 및 해당 암호화 유형이 AWS Key Management Service(AWS KMS) 키(SSE-KMS)와 Amazon S3 관리형 암호화 키(SSE-S3)중 무엇인지를 나타내는 헤더. 자세한 내용은 서버 측 암호화를 사용하여 데이터 보호 섹션을 참조하세요.	예
x-amz-checksum-crc64nvme , x-amz-checksum-crc32 , x-amz-checksum-crc32c , x-amz-checksum-sha1 , x-	객체의 체크섬 또는 다이제스트를 포함하는 헤더. 대부분의 경우 Amazon S3에서 사용하도록 하는 체크섬 알고리즘에 따라 이러한 헤더 중 하나가 한 번에 설정됩니다. 체크섬 알고리즘 선택에 대한 자세한 내용은 Amazon S3에서 객체 무결성 확인 단원을 참조하십시오.	아니요

이름	설명	사용자의 값 수정 여부
amz-checksum-sha256		
x-amz-checksum-type	멀티파트 객체에 대한 객체 수준 체크섬을 생성하기 위해 파트 수준 체크섬을 결합하는 방법을 결정하는 체크섬 유형입니다.	예
x-amz-version-id	객체 버전. 버전 관리를 사용하는 버킷의 경우 Amazon S3는 버킷에 추가된 객체에 버전 ID를 지정합니다. 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하세요.	아니요
x-amz-delete-marker	객체가 삭제 마커인지를 나타내는 부울 마커. 이 마커는 버전 관리가 활성화된 버킷에서만 사용됩니다.	아니요
x-amz-storage-class	객체 저장에 사용된 스토리지 클래스. 자세한 내용은 Amazon S3 스토리지 클래스 이해 및 관리 섹션을 참조하세요.	예
x-amz-website-redirect-location	관련 객체에 대한 요청을 동일한 버킷의 다른 객체 또는 외부 URL로 리디렉션하는 헤더. 자세한 내용은 (선택 사항) 웹 페이지 리디렉션 구성 섹션을 참조하세요.	예
x-amz-server-side-encryption-aws-kms-key-id	객체를 암호화하는 데 사용된 AWS KMS 대칭 암호화 KMS 키의 ID를 나타내는 헤더. 이 헤더는 x-amz-server-side-encryption 헤더가 존재하고 값이 aws:kms인 경우에만 사용됩니다.	예
x-amz-server-side-encryption-customer-algorithm	고객 제공 암호화 키(SSE-C)를 사용하는 서버 측 암호화가 활성화되었는지를 나타내는 헤더. 자세한 내용은 고객 제공 키(SSE-C)로 서버 측 암호화 사용 섹션을 참조하세요.	예
x-amz-tagging	객체에 대한 태그 집합입니다. 태그 집합은 URL 쿼리 매개변수로 인코딩되어야 합니다.	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**
>
> SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료(EOL)됩니다. 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. `HeadObject` 작업은 객체 자체를 반환하지 않고 객체에서 메타데이터를 검색합니다. 이 작업은 객체의 메타데이터에만 관심이 있는 경우에 유용합니다. `HEAD`를 사용하려면 객체에 대한 `READ` 액세스 권한이 있어야 합니다. 자세한 내용은 Amazon Simple Storage Service API Reference의 HeadObject를 참조하십시오.

사용자 정의 메타데이터는 키-값 페어의 집합입니다. Amazon S3는 사용자 정의 메타데이터 키를 소문자로 저장합니다.

Amazon S3는 메타데이터 값에 임의의 유니코드 문자를 허용합니다.

이러한 메타데이터 값 표시와 관련한 문제를 방지하려면 REST를 사용할 경우 US-ASCII 문자를 사용하고, SOAP를 사용하거나 `POST`를 통해 브라우저 기반 업로드를 사용할 경우 UTF-8을 사용해야 합니다.

메타데이터 값에 US-ASCII 문자가 아닌 문자를 사용하는 경우, 제공된 유니코드 문자열에서 US-ASCII 문자가 아닌 문자가 있는지 검사합니다. 이러한 헤더의 값은 저장 및 인코딩하기 전에 RFC 2047에 따라 문자 디코딩되고, 반환되기 전에 메일이 안전하도록 RFC 2047에 따라 인코딩됩니다. 문자열에 US-ASCII 문자만 포함되어 있으면 그대로 표시됩니다.

다음은 예입니다.

```
PUT /Key HTTP/1.1
Host: amzn-s3-demo-bucket.s3.amazonaws.com
x-amz-meta-nonascii: ÄMÄZÕÑ S3

HEAD /Key HTTP/1.1
Host: amzn-s3-demo-bucket.s3.amazonaws.com
x-amz-meta-nonascii: =?UTF-8?B?w4PChE3Dg8KEWsODwpXDg8KRIFMz?=

PUT /Key HTTP/1.1
Host: amzn-s3-demo-bucket.s3.amazonaws.com
x-amz-meta-ascii: AMAZONS3

HEAD /Key HTTP/1.1
Host: amzn-s3-demo-bucket.s3.amazonaws.com
x-amz-meta-ascii: AMAZONS3
```

> **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 이후에 객체의 복사본을 만들어서 수정하고 이전 객체를 교체하거나 새 버전을 만드는 방법으로 객체의 메타데이터를 변경하는 내용은 Amazon S3 콘솔에서 객체 메타데이터 편집 섹션을 참조하십시오.

Amazon S3 콘솔에서 객체 메타데이터 편집

Amazon S3 콘솔에서 복사 작업을 사용하여 기존 S3 객체의 메타데이터를 편집할 수 있습니다. 메타데이터를 편집하려면 객체를 동일한 대상으로 복사하고 적용하려는 새 메타데이터를 지정하면 객체의 이전 메타데이터가 대체됩니다. 일부 메타데이터는 객체를 업로드할 때 Amazon S3에서 설정합니다. 예를 들어 `Content-Length` 및 `Last-Modified`는 사용자가 수정할 수 없는 시스템 정의 객체 메타데이터 필드입니다.

객체를 업로드할 때 사용자 정의 메타데이터를 설정하고 필요에 따라 변경할 수도 있습니다. 예를 들어 처음에 `STANDARD` 스토리지 클래스에 객체 세트를 저장했습니다. 시간이 지나면 이 데이터의 고가용성이 더 이상 필요하지 않을 수도 있습니다. 따라서 `x-amz-storage-class` 키의 값을 `STANDARD`에서 `GLACIER`로 바꾸면 스토리지 클래스를 `GLACIER`로 변경할 수 있습니다.

Note

Amazon S3에서 객체 메타데이터를 바꿀 때는 다음 사항을 고려해야 합니다.

- 유지하려는 기존 메타데이터, 추가하려는 메타데이터 및 편집하려는 메타데이터를 지정해야 합니다.
- 객체가 5GB 미만인 경우 S3 콘솔에서 복사 작업을 사용하여 객체 메타데이터를 바꿀 수 있습니다. 객체가 5GB보다 큰 경우, AWS CLI 또는 AWS SDK를 사용하여 여러 부분으로 업로드하는 객체를 복사할 때 객체 메타데이터를 대체할 수 있습니다. 자세한 내용은 멀티파트 업로드를 사용한 객체 복사 섹션을 참조하세요.
- 자세한 내용은 멀티파트 업로드를 사용하여 the section called “S3 API 작업에 필요한 권한”를 참조하세요. 이 권한을 부여하는 정책의 예시는 the section called “자격 증명 기반 정책 예제” 단원을 참조하세요.
- 이 작업을 수행하면 업데이트된 설정과 마지막 수정 날짜가 포함된 객체의 복사본이 만들어집니다. S3 버전 관리가 사용 설정된 경우 객체의 새 버전이 생성되고 기존 객체는 이전 버전이 됩니다. S3 버전 관리가 활성화되지 않은 경우 객체의 새 복사본이 원본 객체를 대체합니다. 또한 속성을 변경하는 IAM 역할과 연결된 AWS 계정도 새 객체(또는 객체 버전)의 소유자가 됩니다.
- 메타데이터를 편집하면 기존 키 이름의 값이 바뀝니다
- 고객 제공 암호화 키(SSE-C)로 암호화된 객체는 콘솔을 사용하여 복사할 수 없습니다. AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용해야 합니다.
- Amazon S3 콘솔을 사용하여 객체를 복사할 때 "복사된 메타데이터를 확인할 수 없습니다."라는 오류 메시지가 표시될 수 있습니다. 콘솔은 헤더를 사용하여 객체에 대한 메타데이터를 검색하고 설정합니다. 네트워크 또는 브라우저 구성이 네트워크 요청을 수정하면 이 동작으로 인해 복사된 객체에 의도하지 않은 메타데이터(수정된 `Cache-Control` 헤더 등)가 기록될 수 있습니다. Amazon S3는 이 의도하지 않은 메타데이터를 확인할 수 없습니다.

  이 문제를 해결하려면 네트워크 및 브라우저 구성을 확인하여 `Cache-Control`과 같은 헤더를 수정하지 않는지 확인하세요. 자세한 내용은 The Shared Responsibility Model을 참조하세요.

> **Warning**
>
> 폴더의 메타데이터를 바꿀 때는 복사 작업이 완료될 때까지 기다렸다가 폴더에 새 객체를 추가하세요. 그렇지 않으면 새 객체도 편집될 수 있습니다.

다음 주제에서는 Amazon S3 콘솔에서 복사 작업을 사용하여 객체의 메타데이터를 바꾸는 방법에 대해 설명합니다.

시스템 정의 메타데이터 바꾸기

S3 객체에 대해 몇 가지 시스템 정의 메타데이터를 바꿀 수 있습니다. 시스템 정의 메타데이터 및 수정할 수 있는 값 목록은 시스템 정의 객체 메타데이터를 참조하세요.

객체의 시스템 정의 메타데이터 바꾸기

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷 또는 디렉터리 버킷을 선택합니다.
3. 버킷 목록에서 변경할 객체가 들어 있는 버킷 이름을 선택합니다.
4. 변경하려는 객체의 확인란을 선택합니다.
5. 작업 메뉴에 표시되는 옵션 목록에서 복사를 선택합니다.
6. 대상 경로를 지정하려면 S3 찾아보기를 선택하고 소스 객체와 동일한 대상으로 이동한 다음 대상 확인란을 선택합니다. 오른쪽 하단 모서리에서 대상 선택(Choose destination)을 선택합니다.

   또는 대상 경로를 입력합니다.
7. 버킷 버전 관리를 사용 설정하지 않은 경우, 의도치 않게 객체를 덮어쓰거나 삭제하는 것을 방지하기 위해 버킷 버전 관리를 사용하도록 권장하는 경고 메시지가 표시됩니다. 객체의 모든 버전을 이 버킷에 보관하려면 버킷 버전 관리 사용(Enable Bucket Versioning)을 선택합니다. 대상 세부 사항에서 기본 암호화 및 Object Lock 속성을 볼 수도 있습니다.
8. 추가 복사 설정 아래에서 설정 지정을 선택하여 메타데이터에 대한 설정을 지정합니다.
9. 메타데이터 섹션으로 스크롤한 다음 모든 메타데이터 바꾸기를 선택합니다.
10. [메타데이터 추가(Add metadata)]를 선택합니다.
11. 메타데이터 유형(Type)에서 시스템 정의(System-defined)를 선택합니다.
12. 고유 키 및 메타데이터 값을 지정합니다.

13. 추가 메타데이터를 편집하려면 메타데이터 추가(Add metadata)를 선택합니다. [제거(Remove)]를 선택하여 유형-키-값 세트를 제거할 수도 있습니다.
14. 복사를 선택합니다. Amazon S3는 메타데이터 변경 사항을 저장합니다.

사용자 정의 메타데이터 바꾸기

메타데이터 접두사 `x-amz-meta-`와 사용자 지정 키를 만들기 위해 선택한 이름을 결합하여 객체의 사용자 정의 메타데이터를 바꿀 수 있습니다. 예를 들어, `alt-name`을 사용자 이름으로 추가하면 메타데이터 키는 `x-amz-meta-alt-name`이 됩니다.

사용자 정의 메타데이터의 최대 크기는 총 2KB입니다. 사용자 정의 메타데이터의 총 크기를 계산하려면 각 키와 값에 대한 UTF-8 인코딩의 바이트 수를 합산합니다. 키와 값 모두 US-ASCII 표준에 부합해야 합니다. 자세한 내용은 사용자 정의 객체 메타데이터 섹션을 참조하세요.

객체의 사용자 정의 메타데이터 바꾸기

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 탐색 창에서 버킷을 선택한 다음 범용 버킷 또는 디렉토리 버킷 탭을 선택합니다. 변경할 객체가 포함된 Amazon S3 버킷 또는 폴더로 이동합니다.
3. 변경하려는 객체의 확인란을 선택합니다.
4. 작업 메뉴에 표시되는 옵션 목록에서 복사를 선택합니다.
5. 대상 경로를 지정하려면 S3 찾아보기를 선택하고 소스 객체와 동일한 대상으로 이동한 다음 대상 확인란을 선택합니다. 목적지 선택(Choose destination)을 선택합니다.

   또는 대상 경로를 입력합니다.
6. 버킷 버전 관리를 사용 설정하지 않은 경우, 의도치 않게 객체를 덮어쓰거나 삭제하는 것을 방지하기 위해 버킷 버전 관리를 사용하도록 권장하는 경고 메시지가 표시됩니다. 객체의 모든 버전을 이 버킷에 보관하려면 버킷 버전 관리 사용(Enable Bucket Versioning)을 선택합니다. 대상 세부 사항에서 기본 암호화 및 Object Lock 속성을 볼 수도 있습니다.
7. 추가 복사 설정 아래에서 설정 지정을 선택하여 메타데이터에 대한 설정을 지정합니다.
8. 메타데이터 섹션으로 스크롤한 다음 모든 메타데이터 바꾸기를 선택합니다.
9. [메타데이터 추가(Add metadata)]를 선택합니다.
10. 메타데이터의 [유형(Type)]에 대해 [사용자 정의(User-defined)]를 선택합니다.
11. `x-amz-meta-` 다음에 고유한 사용자 지정 키를 입력합니다. 메타데이터 값도 입력합니다.

12. 메타데이터를 추가하려면 메타데이터 추가(Add metadata)를 선택합니다. [제거(Remove)]를 선택하여 유형-키-값 세트를 제거할 수도 있습니다.
13. 복사를 선택합니다. Amazon S3는 메타데이터 변경 사항을 저장합니다.

S3 메타데이터를 사용하여 데이터 검색 가속화

Amazon S3 메타데이터는 범용 버킷의 객체에 대한 메타데이터를 자동으로 캡처하고, 쿼리할 수 있는 읽기 전용 완전관리형 Apache Iceberg 테이블에 저장하여 데이터 검색을 가속화합니다. 이러한 읽기 전용 테이블을 메타데이터 테이블이라고 합니다. 객체가 범용 버킷에 추가, 업데이트 및 제거되면 S3 메타데이터는 해당 메타데이터 테이블을 자동으로 새로 고쳐 최신 변경 사항을 반영합니다.

기본적으로 S3 메타데이터는 세 가지 유형의 메타데이터를 제공합니다.

- 객체의 생성 시간 및 스토리지 클래스와 같은 시스템 정의 메타데이터
- 객체 업로드 중에 포함된 태그 및 사용자 정의 메타데이터와 같은 사용자 지정 메타데이터
- 객체가 업데이트되거나 삭제되는 시기 및 요청을 수행한 AWS 계정과 같은 이벤트 메타데이터

메타데이터 테이블에 저장되는 데이터에 대한 자세한 내용은 the section called “메타데이터 테이블 스키마” 섹션을 참조하세요.

S3 메타데이터를 사용하면 S3 객체에 대한 메타데이터를 쉽게 찾고 저장하고 쿼리할 수 있으므로 비즈니스 분석, 콘텐츠 검색, 인공 지능 및 기계 학습(AI/ML) 모델 훈련 등에 사용할 데이터를 빠르게 준비할 수 있습니다.

메타데이터 테이블은 테이블 형식 데이터에 최적화된 스토리지를 제공하는 S3 테이블 버킷에 저장됩니다. 메타데이터를 쉽게 쿼리하기 위해 테이블 버킷을 AWS Glue Data Catalog와 통합할 수 있습니다. 테이블 버킷이 AWS Glue Data Catalog와 통합되면 Amazon Athena, Amazon EMR, Amazon Redshift, Apache Spark 및 Trino와 같은 쿼리 엔진을 사용하여 메타데이터 테이블을 직접 쿼리할 수 있습니다 Apache Iceberg 형식을 지원하는 다른 애플리케이션을 사용하여 메타데이터 테이블을 쿼리할 수도 있습니다. 메타데이터 테이블에서 대시보드를 생성하려면 Amazon QuickSight를 사용합니다.

S3 메타데이터 요금은 Amazon S3 요금을 참조하세요.

메타데이터 테이블 작동 방식

메타데이터 테이블은 Amazon S3에서 관리하며 Amazon S3 외부의 IAM 위탁자는 수정할 수 없습니다. (하지만 사용자의 메타데이터 테이블은 사용자가 삭제할 수 있습니다.) 따라서 메타데이터 테이블은 읽기 전용이므로 버킷의 콘텐츠를 올바르게 반영할 수 있습니다.

Apache Iceberg 메타데이터 테이블 성능을 최상의 상태로 유지하기 위해 Amazon S3는 테이블에 대해 압축 및 참조되지 않은 파일 제거와 같은 유지 관리 활동을 정기적으로 수행합니다. 이러한 유지 관리 활동은 메타데이터 테이블 저장 비용을 최소화하고 쿼리 성능을 최적화하는 데 도움이 됩니다. 이 테이블 유지 관리는 자동으로 수행되므로 사용자가 옵트인하거나 지속적으로 관리할 필요가 없습니다. 그러나 필요한 경우 이러한 테이블 유지 관리 활동을 구성할 수 있습니다. 자세한 내용은 Table bucket maintenance를 참조하세요.

> **Note**
>
> S3 메타데이터는 사용자가 범용 버킷을 변경할 때 메타데이터 테이블에 지속적으로 추가하도록 설계되었습니다. 각 업데이트는 메타데이터 테이블의 새 버전인 스냅샷을 생성합니다. 메타데이터 테이블은 읽기 전용이므로 메타데이터 테이블에서 레코드를 삭제할 수 없습니다. 또한 S3 Tables의 스냅샷 만료 기능을 사용하여 메타데이터 테이블의 이전 스냅샷을 만료시킬 수 없습니다.
>
> 비용을 최소화하기 위해 메타데이터 테이블 구성과 메타데이터 테이블을 주기적으로 삭제한 다음 다시 생성할 수 있습니다. 자세한 내용은 the section called “메타데이터 테이블 구성 삭제” 및 the section called “메타데이터 테이블 삭제”(을)를 참조하세요.

S3 관리형 메타데이터 테이블에 객체 메타데이터를 생성하고 저장하려면 범용 버킷에 대한 메타데이터 테이블 구성을 생성합니다. Amazon S3는 버킷에서 구성이 활성 상태인 한 메타데이터 테이블을 지속적으로 업데이트하여 데이터에 대한 최신 변경 사항을 반영하도록 설계되었습니다.

메타데이터 테이블 구성을 생성하려면 메타데이터 테이블을 생성하고 관리하는 데 필요한 AWS Identity and Access Management(IAM) 권한이 있는지 확인해야 합니다. 자세한 내용은 the section called “메타데이터 테이블에 대한 권한” 섹션을 참조하세요. 또한 메타데이터 테이블을 저장할 S3 테이블 버킷을 생성하거나 지정해야 합니다. 이 테이블 버킷은 범용 버킷과 동일한 AWS 리전 및 계정에 있어야 합니다. 테이블 버킷 생성에 대한 자세한 내용은 Creating table buckets을 참조하세요.

> **Note**
>
> S3 메타데이터는 메타데이터 테이블 구성을 생성하기 전에 범용 버킷에 이미 존재한 객체에는 적용되지 않습니다. 즉, S3 메타데이터는 메타데이터 테이블 구성을 생성한 후에 발생하는 변경 이벤트(예: 업로드, 업데이트 및 삭제)에 대한 메타데이터만 캡처합니다.

메타데이터 테이블 구성에 대한 업데이트를 모니터링하려면 AWS CloudTrail을 사용할 수 있습니다. 자세한 내용은 the section called “CloudTrail 로깅을 통해 추적되는 Amazon S3 버킷 수준 작업” 섹션을 참조하세요.

주제

- 메타데이터 테이블의 한계 및 제한
- S3 메타데이터 테이블 스키마
- 메타데이터 테이블 구성
- 메타데이터 테이블 쿼리

메타데이터 테이블의 한계 및 제한

메타데이터 테이블 구성을 생성하기 전에 다음 한계 및 제한에 유의하세요.

- S3 메타데이터는 현재 미국 동부(버지니아 북부), 미국 동부(오하이오) 및 미국 서부(오리건) 리전에서만 사용할 수 있습니다.
- S3 메타데이터는 다음을 제외한 모든 스토리지 클래스를 지원합니다.
  - S3 Express One Zone 스토리지 클래스
  - 로컬 영역의 디렉터리 버킷에 있는 S3 One Zone-Infrequent Access(S3 One Zone-IA, Z-IA) 스토리지 클래스

> **Note**
>
> S3 Intelligent-Tiering 스토리지 클래스의 경우 메타데이터 테이블에 특정 티어가 표시되지 않습니다.

- 메타데이터 테이블 구성을 생성하려면 메타데이터 테이블을 저장할 S3 테이블 버킷을 생성하거나 지정해야 합니다. 이 테이블 버킷은 범용 버킷과 동일한 AWS 리전 및 AWS 계정에 있어야 합니다.
- 디렉터리 버킷 또는 테이블 버킷에는 S3 메타데이터가 지원되지 않습니다. 범용 버킷에 대해서만 메타데이터 테이블 구성을 생성할 수 있습니다.
- S3 메타데이터는 메타데이터 테이블 구성을 생성하기 전에 범용 버킷에 이미 존재한 객체에는 적용되지 않습니다. 즉, S3 메타데이터는 메타데이터 테이블 구성을 생성한 후에 발생하는 변경 이벤트(예: 업로드, 업데이트 및 삭제)에 대한 메타데이터만 캡처합니다.
- S3 메타데이터는 사용자가 범용 버킷을 변경할 때 메타데이터 테이블에 지속적으로 추가하도록 설계되었습니다. 각 업데이트는 메타데이터 테이블의 새 버전인 스냅샷을 생성합니다. 메타데이터 테

이블은 읽기 전용이므로 메타데이터 테이블에서 레코드를 삭제할 수 없습니다. 또한 S3 Tables의 스냅샷 만료 기능을 사용하여 메타데이터 테이블의 이전 스냅샷을 만료시킬 수 없습니다.

비용을 최소화하기 위해 메타데이터 테이블 구성과 메타데이터 테이블을 주기적으로 삭제한 다음 다시 생성할 수 있습니다. 자세한 내용은 the section called “메타데이터 테이블 구성 삭제” 및 the section called “메타데이터 테이블 삭제”(을)를 참조하세요.

- 테이블 버킷 또는 테이블 정책을 생성하거나 업데이트할 때 Amazon S3가 테이블 버킷 또는 메타데이터 테이블에 쓰는 것을 제한하지 않아야 합니다. Amazon S3가 테이블 버킷 또는 메타데이터 테이블에 쓸 수 없는 경우 메타데이터 테이블 구성과 메타데이터 테이블을 삭제한 다음 새 구성을 생성하여 새 메타데이터 테이블을 생성해야 합니다.
- 메타데이터 테이블을 삭제하려면 먼저 범용 버킷에서 연결된 메타데이터 테이블 구성을 삭제해야 합니다.
- 전체 범용 버킷에 대해서만 메타데이터 테이블 구성을 생성할 수 있습니다. 접두사 수준에서는 메타데이터 테이블 구성을 적용할 수 없습니다.
- 메타데이터 테이블에 대한 업데이트를 일시 중지하고 재개할 수 없습니다. 대신 연결된 메타데이터 테이블 구성을 삭제하여 메타데이터 테이블의 업데이트를 중지할 수 있습니다. 업데이트 수신을 다시 시작하려면 새 메타데이터 테이블 구성을 생성해야 합니다. 그러면 새 메타데이터 테이블이 생성됩니다.
- 메타데이터 테이블에는 S3 인벤토리 또는 Amazon S3 REST API를 통해 사용할 수 있는 메타데이터와 동일한 메타데이터가 모두 포함되어 있지는 않습니다. 예를 들어 메타데이터 테이블에서는 다음 정보를 사용할 수 없습니다.
  - S3 수명 주기 만료 또는 전환 상태
  - Object Lock 보존 기간 또는 거버넌스 모드
  - 객체 액세스 제어 목록(ACL) 정보
  - 복제 상태
- 메타데이터 테이블에 대한 파티셔닝 또는 정렬은 조정할 수 없습니다. 따라서 일부 쿼리에는 테이블 스캔이 필요할 수 있으므로 효율성이 떨어질 수 있습니다.

S3 메타데이터 테이블 스키마

Amazon S3 메타데이터 테이블에는 행과 열이 포함됩니다. 각 행은 범용 버킷에서 객체를 생성, 업데이트 또는 삭제한 변형 이벤트를 나타냅니다. 이러한 이벤트의 대부분은 다양한 사용자 작업의 결과이지만, 이러한 이벤트 중 일부는 Amazon S3 수명 주기 만료 또는 스토리지 클래스 전환과 같이 Amazon S3가 사용자를 대신하여 수행한 작업의 결과입니다.

S3 메타데이터는 메타데이터 테이블을 최종적으로 범용 버킷에서 발생한 변경 사항과 일치시키도록 설계된 이벤트 처리 파이프라인입니다. 객체가 생성되거나 업데이트되었다는 알림이 S3 메타데이터에 수신될 때쯤에는 해당 객체가 버킷에서 이미 덮어쓰였거나 삭제되었을 수 있습니다. 기본적으로 각 S3 버킷 작업에 대해 테이블 행이 생성됩니다. 그러나 객체 메타데이터가 삭제되었거나 덮어쓰였거나 객체를 더 이상 검색할 수 없는 경우 일부 열에는 누락된 메타데이터 스키마를 나타내는 NULL 값이 표시될 수 있습니다.

다음은 `amzn-s3-demo-bucket:`이라는 범용 버킷에 대한 메타데이터 테이블의 예입니다.

```
bucket                key                        sequence_number
                                                                  record_type
 record_timestamp           version_id   is_delete_marker   size   last_modified_date
   e_tag                            storage_class  is_multipart   encryption_status
 is_bucket_key_enabled   kms_key_arn
                checksum_algorithm   object_tags   user_metadata

    requester      source_ip_address   request_id
amzn-s3-demo-bucket   Finance/statement1.pdf
 80e737d8b4d82f776affffffffffffffff006737d8b4d82f776a00000000000000000000000000000000000000000
   CREATE        2024-11-15 23:26:44.899                 FALSE
 6223   11/15/2024 23:26     e131b86632dda753aac4018f72192b83    STANDARD
 FALSE          SSE-KMS             FALSE                   arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890df   SSECRC32             {}
            {count -> Asia, customs -> false, family -> true, location -> Mary, name ->
 football, user -> United States}                       111122223333   192.0.2.1
    CVK8FWYRW0M9JW65
amzn-s3-demo-bucket   s3-dg.pdf
 80e737d8b4e39f1dbdffffffffffffffff006737d8b4e39f1dbd00000000000000000000000000000000000000000
   CREATE        2024-11-15 23:26:44.942                 FALSE
 3554   11/15/2024 23:26     9bb49efc2d92c05558ddffbbde8636d5    STANDARD
 FALSE          DSSE-KMS            FALSE                   arn:aws:kms:us-
east-1:936810216292:key/0dcebce6-49fd-4cae-b2e2-5512ad281afd   SSESHA1              {}
           {}
                                                      111122223333   192.0.2.1
   CVKAQDRAZEG7KXAY
amzn-s3-demo-bucket   Development/Projects.xls
 80e737d8b4ed9ac5c6ffffffffffffffff006737d8b4ed9ac5c60000000000000000000000000000000000000000
   CREATE        2024-11-15 23:26:44.966                 FALSE              7746
 11/15/2024 23:26     729a6863e47fb9955b31bfabce984908    STANDARD   FALSE
 SSE-S3              FALSE                   NULL
                                     SSECRC32             {}            {count -> Asia,
```

```
customs -> Canada, family -> Billiards, filter -> true, location -> Europe, name ->
Asia, user -> United States}   111122223333   192.0.2.1          CVK7Z6XQTQ90BSRV
```

메타데이터 테이블에는 다음과 같은 스키마가 있습니다.

열 이름	필수?	데이터 유형	
bucket	예	String	범용 버킷 이름입니다. 자세한 내용은 the section called “이름 지정 규칙” 섹션을 참조하세요.
key	예	String	버킷에 있는 객체를 고유하게 식별하는 객체 키 이름(또는 키)입니다. 자세한 내용은 the section called “객체 이름 지정” 섹션을 참조하세요.
sequence_number	예	String	주어진 객체의 레코드에 포함된 서수인 시퀀스 번호입니다. 동일한 버킷 및 키의 레코드를 정렬하려면 sequence_number 를 기준으로 정렬할 수 있습니다. 주어진 버킷과 키에서 사전을 기준으로 sequence_number 값이 크면 레코드가 더 최근에 버킷에 기록되었음을 의미합니다.
record_type	예	String	이 레코드의 유형으로, CREATE, UPDATE_ME

열 이름	필수?	데이터 유형	
			TADATA 또는 DELETE 중 하나입니다. CREATE 레코드는 새 객체(또는 객체의 새 버전)가 버킷에 기록되었음을 나타냅니다. UPDATE_METADATA 는 스토리지 클래스 또는 태그와 같은 기존 객체의 변경 가능한 메타데이터에 대한 변경 사항을 캡처합니다. DELETE는 이 객체(또는 객체의 이 버전)가 삭제되었음을 나타냅니다. 버전 관리가 활성화된 경우 DELETE는 삭제 마커 또는 영구 삭제를 나타냅니다. 삭제 마커의 record_type 값은 DELETE이고 is_delete_marker 값은 True입니다. 영구 삭제 레코드는 bucket, key, sequence_number , record_type , record_timestamp ,

열 이름	필수?	데이터 유형	
			`version_id` 를 제외한 다른 모든 열에 Null 값을 갖습니다. 자세한 내용은 the section called “객체 버전 삭제” 섹션을 참조하세요.
`record_timestamp`	예	타임스탬프 NTZ(표준 시간대 없음)	이 레코드와 연결된 타임스탬프입니다.
`version_id`	아니요	String	객체의 버전 ID입니다. 버전 관리를 사용하는 버킷의 경우 Amazon S3는 버킷에 추가된 객체에 버전 번호를 지정합니다. 자세한 내용은 the section called “여러 버전의 객체 유지” 섹션을 참조하세요. 버전 관리 상태를 설정하기 전에 버킷에 저장된 객체의 버전 ID는 Null입니다.

열 이름	필수?	데이터 유형	
is_delete_marker	아니요	불	객체의 삭제 마커 상태입니다. 객체가 삭제 마커인 경우 이 값은 True입니다. 그렇지 않은 경우 False입니다. 자세한 내용은 the section called “삭제 마커를 통한 작업” 섹션을 참조하세요. **Note** 삭제 마커에 추가된 행의 record_type 값은 UPDATE_METADATA 가 아니라 DELETE입니다. S3 수명 주기 만료의 결과로 삭제 마커가 생성된 경우 requester 값은 s3.amazonaws.com 입니다.

열 이름	필수?	데이터 유형	
size	아니요	Long	객체 크기(바이트)를 나타내며, 불완전한 멀티파트 업로드 또는 객체 메타데이터의 크기는 포함되지 않습니다. is_delete_marker 가 True인 경우 크기는 0입니다. 자세한 내용은 the section called “시스템 정의 객체 메타데이터” 섹션을 참조하세요.
last_modified_date	아니요	타임스탬프 NTZ(표준 시간대 없음)	객체 생성일 또는 최종 수정일 중 최근 날짜. 멀티파트 업로드의 경우 객체 생성 날짜는 멀티파트 업로드 시작 날짜입니다. 자세한 내용은 the section called “시스템 정의 객체 메타데이터” 섹션을 참조하세요.

열 이름	필수?	데이터 유형	
e_tag	아니요	String	엔터티 태그(ETag)로, 객체의 해시입니다. ETag는 객체의 콘텐츠에 대한 변경 사항만 반영하고 메타데이터에 대한 변경을 반영하지 않습니다. ETag는 객체 데이터의 MD5 다이제스트일 수 있습니다. ETag가 MD5 다이제스트인지는 객체 생성 방식 및 암호화 방식에 달려 있습니다. 자세한 내용은 Amazon S3 API 참조의 Object 섹션을 참조하세요.

열 이름	필수?	데이터 유형	
storage_class	아니요	String	객체 저장에 사용된 스토리지 클래스입니다. STANDARD, REDUCED_REDUNDANCY, STANDARD_IA, ONEZONE_IA, INTELLIGENT_TIERING, GLACIER, DEEP_ARCHIVE 또는 GLACIER_IR 중 하나입니다. 자세한 내용은 the section called “스토리지 클래스 이해 및 관리” 섹션을 참조하세요.
is_multipart	아니요	불	객체의 업로드 유형입니다. 객체가 멀티파트 업로드로 업로드된 경우 이 값은 True입니다. 그렇지 않은 경우 False입니다. 자세한 내용은 the section called “멀티파트 업로드 사용” 섹션을 참조하세요.

열 이름	필수?	데이터 유형	
`encryption_status`	아니요	String	Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3), AWS Key Management Service(AWS KMS) 키를 사용한 서버 측 암호화(SSE-KMS), AWS KMS keys KMS 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS), 고객 제공 키를 사용한 서버 측 암호화(SSE-C) 등 사용되는 암호화 키의 종류에 따른 객체의 서버 측 암호화 상태입니다. 객체가 암호화되지 않은 경우 이 값은 Null입니다. 가능한 값은 `SSE-S3`, `SSE-KMS`, `DSSE-KMS`, `SSE-C` 또는 Null입니다. 자세한 내용은 the section called “데이터 암호화” 섹션을 참조하세요.

열 이름	필수?	데이터 유형	
`is_bucket_key_enabled`	아니요	불	객체의 S3 버킷 키 활성화 상태입니다. 객체가 SSE-KMS에 S3 버킷 키를 사용하는 경우 이 값은 `True`입니다. 그렇지 않은 경우 `False`입니다. 자세한 내용은 the section called “객체에 대한 S3 버킷 키 구성” 섹션을 참조하세요.

열 이름	필수?	데이터 유형	
kms_key_arn	아니요	String	객체 암호화에 사용된 KMS 키의 Amazon 리소스 이름(ARN)으로, encryption_status 가 SSE-KMS 또는 DSSE-KMS인 행의 경우에 해당합니다. 객체가 SSE-KMS 또는 DSSE-KMS로 암호화되지 않은 경우 값은 Null입니다. 자세한 내용은 the section called “AWS KMS에 저장된 KMS 키(SSE-KMS)” 및 the section called “이중 계층 서버 측 암호화(DSSE-KMS)”(을)를 참조하세요. **Note** 행이 삭제 또는 덮어쓰기 이벤트가 처리된 시점에 더 이상 존재하지 않았던 객체 버전을 나타내는 경우 encryption_status 열 값이 SSE-KMS 또는

열 이름	필수?	데이터 유형	
			DSSE-KMS인 경우에도 kms_key_arn 에는 Null 값이 포함됩니다.
checksum_algorithm	아니요	String	객체에 대한 체크섬을 생성하는 데 사용되는 알고리즘으로, CRC64NVME , CRC32, CRC32C, SHA1 또는 SHA256 중 하나입니다. 체크섬이 없는 경우 이 값은 Null입니다. 자세한 내용은 the section called “지원되는 체크섬 알고리즘 사용” 섹션을 참조하세요.

열 이름	필수?	데이터 유형	
object_tags	아니요	Map <String, String>	객체와 연결된 객체 태그입니다. 객체 태그는 키-값 페어의 맵으로 저장됩니다. 객체에 객체 태그가 없는 경우 빈 맵({})이 저장됩니다. 자세한 내용은 the section called “태그로 객체 분류” 섹션을 참조하세요. **Note** record_type 값이 DELETE이면 object_tags 열에 Null 값이 포함됩니다. record_type 값이 CREATE 또는 UPDATE_METADATA 인 경우 삭제 또는 덮어쓰기 이벤트가 처리된 시점에 더 이상 존재하지 않았던 객체 버전을 나타내는 행에는 object_tags 열에 Null

열 이름	필수?	데이터 유형	
			값이 포함됩니다.

열 이름	필수?	데이터 유형	
user_metadata	아니요	Map <String, String>	객체와 연결된 사용자 메타데이터입니다. 사용자 메타데이터는 키-값 페어의 맵으로 저장됩니다. 객체에 사용자 메타데이터가 없는 경우 빈 맵({})이 저장됩니다. 자세한 내용은 the section called “사용자 정의 객체 메타데이터” 섹션을 참조하세요. **Note** record_type 값이 DELETE이면 user_metadata 열에 Null 값이 포함됩니다. record_type 값이 CREATE 또는 UPDATE_METADATA 인 경우 삭제 또는 덮어쓰기 이벤트가 처리된 시점에 더 이상 존재하지 않았던 객체 버전을 나타내는 행에는

열 이름	필수?	데이터 유형	
			`user_metadata` 열에 Null 값이 포함됩니다.
`requester`	아니요	String	요청을 수행한 요청자 또는 AWS 서비스 위탁자의 AWS 계정 ID입니다.
`source_ip_address`	아니요	String	요청의 소스 IP 주소입니다. 사용자 요청에 의해 생성된 레코드의 경우 이 열에는 요청의 소스 IP 주소가 포함됩니다. 사용자를 대신해 Amazon S3 또는 다른 AWS 서비스가 수행한 작업의 경우 이 열에는 Null 값이 포함됩니다.
`request_id`	아니요	String	요청과 연결된 요청 ID입니다.

메타데이터 테이블 구성

Amazon S3 메타데이터는 범용 버킷의 객체에 대한 메타데이터를 자동으로 캡처하고, 쿼리할 수 있는 읽기 전용 완전관리형 Apache Iceberg 테이블에 저장하여 데이터 검색을 가속화합니다. 이러한 읽기 전용 테이블을 메타데이터 테이블이라고 합니다. 객체가 범용 버킷에 추가, 업데이트 및 제거되면 S3 메타데이터는 해당 메타데이터 테이블을 자동으로 새로 고쳐 최신 변경 사항을 반영합니다.

S3 메타데이터를 사용하면 S3 객체에 대한 메타데이터를 쉽게 찾고 저장하고 쿼리할 수 있으므로 비즈니스 분석, 인공 지능 및 기계 학습(AI/ML) 모델 훈련 등에 사용할 데이터를 빠르게 준비할 수 있습니다.

S3 관리형 메타데이터 테이블에 객체 메타데이터를 생성하고 저장하려면 범용 버킷에 대한 메타데이터 테이블 구성을 생성합니다. Amazon S3는 버킷에서 구성이 활성 상태인 한 메타데이터 테이블을 지속적으로 업데이트하여 데이터에 대한 최신 변경 사항을 반영하도록 설계되었습니다. 또한 Amazon S3는 메타데이터 테이블을 지속적으로 최적화하여 스토리지 비용을 줄이고 분석 쿼리 성능을 개선하는 데 도움을 줍니다.

메타데이터 테이블 구성을 생성하려면 메타데이터 테이블을 생성하고 관리하는 데 필요한 AWS Identity and Access Management (IAM) 권한이 있는지 확인하세요. 또한 메타데이터 테이블을 저장할 S3 테이블 버킷을 생성하거나 지정해야 합니다. 이 테이블 버킷은 범용 버킷과 동일한 AWS 리전 및 AWS 계정에 있어야 합니다.

메타데이터 테이블 구성에 대한 업데이트를 모니터링하려면 AWS CloudTrail을 사용할 수 있습니다. 자세한 내용은 the section called “CloudTrail 로깅을 통해 추적되는 Amazon S3 버킷 수준 작업” 섹션을 참조하세요.

주제

- 메타데이터 테이블 구성에 대한 권한 설정
- 메타데이터 테이블 구성 생성
- 메타데이터 테이블에 대한 액세스 제어
- 메타데이터 테이블 구성 삭제
- 메타데이터 테이블 삭제

메타데이터 테이블 구성에 대한 권한 설정

메타데이터 테이블 구성을 생성하려면 메타데이터 테이블 구성을 생성 및 관리하고 메타데이터 테이블과 메타데이터 테이블이 저장되는 테이블 버킷을 생성 및 관리하는 데 필요한 AWS Identity and Access Management(IAM) 권한이 있어야 합니다.

메타데이터 테이블 구성을 생성하고 관리하려면 다음 권한이 있어야 합니다.

- `s3:CreateBucketMetadataTableConfiguration` - 이 권한을 사용하면 범용 버킷에 대한 메타데이터 테이블 구성을 생성할 수 있습니다.
- `s3:GetBucketMetadataTableConfiguration` - 이 권한을 사용하면 메타데이터 테이블 구성에 대한 정보를 검색할 수 있습니다.
- `s3:DeleteBucketMetadataTableConfiguration` - 이 권한을 사용하면 메타데이터 테이블 구성을 삭제할 수 있습니다.

테이블 및 테이블 버킷을 생성하고 작업하려면 특정 s3tables 권한이 있어야 합니다. 메타데이터 테이블 구성을 생성하려면 최소한 다음 s3tables 권한이 있어야 합니다.

- s3tables:CreateNamespace - 이 권한을 사용하면 테이블 버킷에 네임스페이스를 생성할 수 있습니다. 메타데이터 테이블은 기본 aws_s3_metadata 네임스페이스를 사용합니다.
- s3tables:GetTable - 이 권한을 사용하면 메타데이터 테이블에 대한 정보를 검색할 수 있습니다.
- s3tables:CreateTable - 이 권한을 사용하면 메타데이터 테이블을 생성할 수 있습니다.
- s3tables:PutTablePolicy - 이 권한을 사용하면 메타데이터 테이블 정책을 추가하거나 업데이트할 수 있습니다.

모든 테이블 및 테이블 버킷 권한에 대한 자세한 내용은 Access management for S3 Tables을 참조하세요.

> **Important**
>
> 또한 메타데이터 테이블을 쿼리할 수 있도록 테이블 버킷을 AWS 분석 서비스와 통합하려면 추가 권한이 필요합니다. 자세한 내용은 Integrating Amazon S3 Tables with AWS analytics services를 참조하세요.

SSE-KMS에 대한 권한

AWS Key Management Service(AWS KMS) 키를 통한 서버 측 암호화(SSE-KMS)를 사용하는 S3 테이블 버킷 또는 S3 테이블에 액세스하려면 추가 권한을 포함해야 합니다.

1. 사용자 또는 IAM 역할에 다음 권한이 필요합니다. IAM 콘솔(https://console.aws.amazon.com/iam/)을 사용해 이러한 권한을 부여할 수 있습니다.
   a. 테이블 암호화 구성: s3tables:PutTableEncryption
   b. 테이블 버킷 암호화 구성: s3tables:PutTableBucketEncryption
   c. 사용된 AWS KMS 키의 kms:DescribeKey
2. KMS 키에 대한 리소스 정책에는 다음 권한이 필요합니다. KMS 콘솔(https://console.aws.amazon.com/kms)을 사용하여 이러한 권한을 부여할 수 있습니다.
   a. metadata.s3.amazonaws.com 및 maintenance.s3tables.amazonaws.com에 대한 kms:GenerateDataKey 권한
   b. metadata.s3.amazonaws.com 및 maintenance.s3tables.amazonaws.com에 대한 kms:Decrypt 권한

c. AWS 위탁자 간접 호출에 대한 `kms:DescribeKey` 권한

S3 메타데이터 서비스에 필요한 권한을 부여하는 방법에 대한 자세한 내용은 Granting the S3 Metadata service principal permissions to use your KMS key에 대한 설명을 참조하세요.

예제 정책

메타데이터 테이블 및 테이블 버킷을 생성하고 작업하려면 다음 예시 정책을 사용할 수 있습니다. 이 정책에서는 메타데이터 테이블 구성을 적용하는 범용 버킷을 *amzn-s3-demo-source-bucket*이라고 합니다. 메타데이터 테이블을 저장하는 테이블 버킷을 *amzn-s3-demo-bucket*이라고 합니다. 이 정책을 사용하려면 두 버킷 이름과 *user input placeholders*를 실제 정보로 대체하세요.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"PermissionsToWorkWithMetadataTables",
         "Effect":"Allow",
         "Action":[
            "s3:CreateBucketMetadataTableConfiguration",
            "s3:GetBucketMetadataTableConfiguration",
            "s3:DeleteBucketMetadataTableConfiguration",
            "s3tables:*"
         ],
         "Resource":[
            "arn:aws:s3:::bucket/amzn-s3-demo-source-bucket",
            "arn:aws:s3tables:region:account_id:bucket/amzn-s3-demo-bucket",
            "arn:aws:s3tables:region:account_id:bucket/amzn-s3-demo-bucket/table/*"
         ]
       }
    ]
}
```

메타데이터 테이블 구성 생성

완전관리형 Apache Iceberg 메타데이터 테이블에 Amazon S3 메타데이터를 생성하고 저장하려면 범용 버킷에 대한 메타데이터 테이블 구성을 생성합니다. Amazon S3는 버킷에서 구성이 활성 상태인 한 메타데이터 테이블을 지속적으로 업데이트하여 데이터에 대한 최신 변경 사항을 반영하도록 설계되었습니다. 또한 Amazon S3는 메타데이터 테이블을 지속적으로 최적화하여 스토리지 비용을 줄이고 분석 쿼리 성능을 개선하는 데 도움을 줍니다.

메타데이터 테이블의 Amazon 리소스 이름(ARN) 형식은 다음과 같습니다.

```
arn:aws:s3tables:region-code:account-id:bucket/table-bucket-name/
table/metadata_table_name
```

Amazon S3 완전관리형 메타데이터 테이블은 테이블 버킷의 `aws_s3_metadata` 네임스페이스에 저장됩니다. 테이블 버킷의 네임스페이스에 대한 자세한 내용은 Table namespaces를 참조하세요.

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하여 메타데이터 테이블 구성을 생성할 수 있습니다.

사전 조건

메타데이터 테이블 구성을 생성하려면 먼저 다음을 수행해야 합니다.

- 메타데이터 테이블을 생성하고 관리하는 데 필요한 AWS Identity and Access Management(IAM) 권한이 있는지 확인합니다. 자세한 내용은 the section called “메타데이터 테이블에 대한 권한” 섹션을 참조하세요.
- 메타데이터 테이블을 저장할 S3 테이블 버킷을 생성합니다. 이 테이블 버킷은 범용 버킷과 동일한 AWS 리전 및 AWS 계정에 있어야 합니다. 테이블 버킷 생성에 대한 자세한 내용은 Creating table buckets을 참조하세요. Amazon S3 콘솔을 사용하여 구성을 생성하는 경우 이 프로세스의 일부로 이 단계를 수행할 수 있습니다.
- Amazon Athena, Amazon EMR, Amazon Redshift, Apache Spark, Apache Trino 및 Apache Iceberg 형식을 지원하는 기타 애플리케이션과 같은 쿼리 엔진으로 메타데이터 테이블을 직접 쿼리할 수 있도록 테이블 버킷을 AWS Glue Data Catalog와 통합합니다. 자세한 내용은 the section called “AWS 분석 서비스를 사용하여 메타데이터 테이블 쿼리” 섹션을 참조하세요.

메타데이터 테이블 구성 생성

S3 콘솔 사용

메타데이터 테이블 구성을 생성하는 방법

메타데이터 테이블 구성을 생성하기 전에 사전 조건을 검토하고 충족했는지 확인하고 the section called “한계 및 제한” 섹션을 검토합니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.

3. 메타데이터 테이블 구성을 생성할 범용 버킷을 선택합니다.

> **Note**
>
> 이 범용 버킷이 테이블 버킷을 사용할 수 있는 AWS 리전인지 확인합니다. 테이블 버킷은 미국 동부(버지니아 북부), 미국 동부(오하이오) 및 미국 서부(오리건) 리전에서만 사용할 수 있습니다.

4. 버킷의 세부 정보 페이지에서 메타데이터 탭을 선택합니다.
5. 메타데이터 탭에서 메타데이터 구성 생성을 선택합니다.
6. 메타데이터 구성 생성 페이지의 대상 테이블 버킷에서 메타데이터 테이블을 저장할 테이블 버킷을 지정합니다. 테이블 버킷은 범용 버킷과 동일한 AWS 계정 및 AWS 리전에 있어야 합니다.

   테이블 버킷이 아직 없는 경우 테이블 버킷 생성을 선택합니다. 테이블 버킷 이름은 3~63자여야 하며 선택한 AWS 리전의 AWS 계정 내에서 고유해야 합니다. 유효한 문자는 a~z, 0~9 및 하이픈(-)입니다. 테이블 버킷 생성에 대한 자세한 내용은 Creating table buckets을 참조하세요.

   테이블 버킷을 생성할 때 AWS Glue Data Catalog와 통합해야 합니다. 자세한 내용은 the section called “AWS 분석 서비스를 사용하여 메타데이터 테이블 쿼리” 섹션을 참조하세요.
7. 메타데이터 테이블 이름에서 테이블의 이름을 지정합니다. 메타데이터 테이블 이름은 1~255자여야 하며 테이블 버킷의 `aws_s3_metadata` 네임스페이스 내에서 고유해야 합니다. 유효한 문자는 소문자, 숫자 및 밑줄(_)입니다.
8. 메타데이터 테이블 구성 생성을 선택합니다.

메타데이터 테이블 구성이 성공하면 메타데이터 테이블의 ARN이 지정된 테이블 버킷 및 메타데이터 테이블 이름과 함께 메타데이터 탭에 표시됩니다.

메타데이터 테이블 구성에 대한 업데이트를 모니터링하려면 AWS CloudTrail을 사용할 수 있습니다. 자세한 내용은 the section called “CloudTrail 로깅을 통해 추적되는 Amazon S3 버킷 수준 작업” 섹션을 참조하세요.

AWS CLI 사용

다음 명령을 실행하려면 AWS CLI를 설치하고 구성해야 합니다. AWS CLI를 설치하지 않은 경우 AWS Command Line Interface 사용 설명서에서 최신 버전의 AWS CLI 설치 또는 업데이트를 참조하세요.

또는 AWS CloudShell을 사용하여 콘솔에서 AWS CLI 명령을 실행할 수 있습니다. AWS CloudShell은 브라우저 기반의 사전 인증된 쉘로, AWS Management Console에서 직접 시작할 수 있습니다. 자세

한 내용은 AWS CloudShell 사용 설명서에서 CloudShell이란 무엇인가요? 및 AWS CloudShell 시작하기를 참조하세요.

AWS CLI를 사용하여 메타데이터 테이블 구성을 생성하는 방법

메타데이터 테이블 구성을 생성하기 전에 사전 조건을 검토하고 충족했는지 확인하고 the section called "한계 및 제한" 섹션을 검토합니다.

다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

1. 테이블 버킷이 아직 없는 경우 다음 명령을 사용하여 메타데이터 테이블을 저장할 테이블 버킷을 생성합니다. 테이블 버킷이 메타데이터 테이블 구성을 생성하려는 범용 버킷과 동일한 AWS 리전에 있는지 확인합니다.

   ```
   aws s3tables create-table-bucket --name amzn-s3-demo-bucket --region us-east-2
   ```

2. 테이블 버킷이 생성되었는지 확인하려면 다음 명령을 사용합니다.

   ```
   aws s3tables list-table-buckets --region us-east-2
   ```

3. 메타데이터 테이블 구성이 포함된 JSON 파일을 생성하고 저장합니다(예: `metadata-config.json`). 다음은 샘플 구성입니다.

   테이블 버킷 이름은 3~63자여야 하며 선택한 AWS 리전의 AWS 계정 내에서 고유해야 합니다. 유효한 문자는 a~z, 0~9 및 하이픈(-)입니다. 테이블 버킷 생성에 대한 자세한 내용은 Creating table buckets을 참조하세요.

   메타데이터 테이블 이름은 1~255자여야 하며 테이블 버킷의 `aws_s3_metadata` 네임스페이스 내에서 고유해야 합니다. 유효한 문자는 소문자, 숫자 및 밑줄(_)입니다.

   ```
   {
     "S3TablesDestination": {
       "TableBucketArn": "arn:aws:s3tables:us-east-2:111122223333:bucket/amzn-s3-demo-bucket",
       "TableName": "test_metadata_table"
     }
   }
   ```

4. 다음 명령을 사용하여 메타데이터 테이블 구성을 범용 버킷에 적용합니다(예: *amzn-s3-demo-source-bucket*).

```
aws s3api create-bucket-metadata-table-configuration \
--bucket amzn-s3-demo-source-bucket \
--metadata-table-configuration  file://./metadata-config.json \
--region us-east-2
```

5. 구성이 추가되었는지 확인하려면 다음 명령을 사용합니다.

```
aws s3api get-bucket-metadata-table-configuration \
--bucket amzn-s3-demo-source-bucket \
--region us-east-2
```

메타데이터 테이블 구성에 대한 업데이트를 모니터링하려면 AWS CloudTrail을 사용할 수 있습니다. 자세한 내용은 the section called “CloudTrail 로깅을 통해 추적되는 Amazon S3 버킷 수준 작업” 섹션을 참조하세요.

REST API 사용

REST 요청을 전송하여 메타데이터 테이블 구성을 생성할 수 있습니다. 자세한 내용은 Amazon S3 API 참조의 CreateBucketMetadataTableConfiguration 섹션을 참조하세요.

AWS SDK 사용

AWS SDK를 사용하여 Amazon S3에서 메타데이터 테이블 구성을 생성할 수 있습니다. 자세한 내용은 Amazon S3 API 참조의 지원되는 SDK 목록을 참조하세요.

메타데이터 테이블에 대한 액세스 제어

Amazon S3 메타데이터 테이블에 대한 액세스를 제어하기 위해 테이블 버킷 및 메타데이터 테이블에 연결된 AWS Identity and Access Management(IAM) 리소스 기반 정책을 사용할 수 있습니다. 즉, 테이블 버킷 수준과 테이블 수준 모두에서 메타데이터 테이블에 대한 액세스를 제어할 수 있습니다.

테이블 버킷 및 테이블에 대한 액세스 제어에 대한 자세한 내용은 Access management for S3 Tables을 참조하세요.

> **Important**
>
> Amazon S3가 테이블 버킷 또는 메타데이터 테이블에 쓰는 것을 제한하지 않아야 합니다. Amazon S3가 테이블 버킷 또는 메타데이터 테이블에 쓸 수 없는 경우 메타데이터 테이블 구성을 삭제한 다음 새 구성을 생성하여 새 메타데이터 테이블을 생성해야 합니다.

AWS Lake Formation을 통해 메타데이터 테이블의 행 및 열에 대한 액세스를 제어할 수도 있습니다. 자세한 내용은 AWS Lake Formation 개발자 안내서의 Managing Lake Formation permissions 및 Data filtering and cell-level security in Lake Formation을 참조하세요.

메타데이터 테이블 구성 삭제

Amazon S3 범용 버킷에 대한 메타데이터 테이블 구성 업데이트를 중지하려면 버킷에 연결된 메타데이터 테이블 구성을 삭제하면 됩니다. 메타데이터 테이블 구성을 삭제하면 구성만 삭제됩니다. 메타데이터 테이블 구성을 삭제하더라도 테이블 버킷과 메타데이터 테이블은 여전히 존재합니다. 그러나 메타데이터 테이블은 더 이상 업데이트되지 않습니다.

메타데이터 테이블을 삭제하려면 the section called “메타데이터 테이블 삭제” 섹션을 참조하세요. 테이블 버킷을 삭제하려면 Deleting table buckets 및 Amazon S3 API 참조의 DeleteTableBucket 섹션을 참조하세요.

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하여 메타데이터 테이블 구성을 삭제할 수 있습니다.

메타데이터 테이블 구성 삭제

S3 콘솔 사용

메타데이터 테이블 구성을 삭제하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 메타데이터 테이블 구성을 제거할 범용 버킷을 선택합니다.
4. 버킷의 세부 정보 페이지에서 메타데이터 탭을 선택합니다.
5. 메타데이터 탭에서 삭제를 선택합니다.
6. 메타데이터 구성 삭제 대화 상자에서 **`confirm`**을 입력하여 구성 삭제를 확정합니다. 그런 다음 삭제를 선택합니다.

AWS CLI 사용

다음 명령을 실행하려면 AWS CLI를 설치하고 구성해야 합니다. AWS CLI를 설치하지 않은 경우 AWS Command Line Interface 사용 설명서에서 최신 버전의 AWS CLI 설치 또는 업데이트를 참조하세요.

또는 AWS CloudShell을 사용하여 콘솔에서 AWS CLI 명령을 실행할 수 있습니다. AWS CloudShell은 브라우저 기반의 사전 인증된 쉘로, AWS Management Console에서 직접 시작할 수 있습니다. 자세

한 내용은 AWS CloudShell 사용 설명서에서 CloudShell이란 무엇인가요? 및 AWS CloudShell 시작하기를 참조하세요.

AWS CLI를 사용하여 메타데이터 테이블 구성을 삭제하는 방법

다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

1. 다음 명령을 사용하여 메타데이터 테이블 구성을 범용 버킷에서 삭제합니다(예: *amzn-s3-demo-source-bucket*).

```
aws s3api delete-bucket-metadata-table-configuration \
--bucket amzn-s3-demo-source-bucket \
--region us-east-2
```

2. 구성이 삭제되었는지 확인하려면 다음 명령을 사용합니다.

```
aws s3api get-bucket-metadata-table-configuration \
--bucket amzn-s3-demo-source-bucket \
--region us-east-2
```

REST API 사용

REST 요청을 전송하여 메타데이터 테이블 구성을 삭제할 수 있습니다. 자세한 내용은 DeleteBucketMetadataTableConfiguration 섹션을 참조하세요.

AWS SDK 사용

AWS SDK를 사용하여 Amazon S3에서 메타데이터 테이블 구성을 삭제할 수 있습니다. 자세한 내용은 지원되는 SDK 목록을 참조하세요.

메타데이터 테이블 삭제

Amazon S3 범용 버킷에 대해 생성한 메타데이터 테이블을 삭제하려면 테이블 버킷에서 메타데이터 테이블을 삭제하면 됩니다.

> **Note**
>
> 메타데이터 테이블을 삭제하려면 먼저 범용 버킷에서 연결된 메타데이터 테이블 구성을 삭제하는 것이 좋습니다. 자세한 내용은 the section called “메타데이터 테이블 구성 삭제” 섹션을 참조하세요.

테이블 버킷을 삭제하려면 Deleting table buckets 및 Amazon S3 API 참조의 DeleteTableBucket 섹션을 참조하세요.

AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하여 메타데이터 테이블을 삭제할 수 있습니다.

메타데이터 테이블 삭제

AWS CLI 사용

다음 명령을 실행하려면 AWS CLI를 설치하고 구성해야 합니다. AWS CLI를 설치하지 않은 경우 AWS Command Line Interface 사용 설명서에서 최신 버전의 AWS CLI 설치 또는 업데이트를 참조하세요.

또는 AWS CloudShell을 사용하여 콘솔에서 AWS CLI 명령을 실행할 수 있습니다. AWS CloudShell은 브라우저 기반의 사전 인증된 쉘로, AWS Management Console에서 직접 시작할 수 있습니다. 자세한 내용은 AWS CloudShell 사용 설명서에서 CloudShell이란 무엇인가요? 및 AWS CloudShell 시작하기를 참조하세요.

AWS CLI를 사용하여 메타데이터 테이블 구성을 삭제하는 방법

다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

1. 다음 명령을 사용하여 메타데이터 테이블을 테이블 버킷에서 삭제합니다(예: *amzn-s3-demo-bucket*).

```
aws s3tables delete-table \
--table-bucket-arn arn:aws:s3tables:us-east-2:111122223333:bucket/amzn-s3-demo-bucket \
--namespace aws_s3_metadata \
--name test_metadata_table \
--region us-east-2
```

2. 테이블이 삭제되었는지 확인하려면 다음 명령을 사용합니다.

```
aws s3tables get-table \
--table-bucket-arn arn:aws:s3tables:us-east-2:111122223333:bucket/amzn-s3-demo-bucket \
--namespace aws_s3_metadata \
--name test_metadata_table \
--region us-east-2
```

REST API 사용

REST 요청을 전송하여 메타데이터 테이블 구성을 삭제할 수 있습니다. 자세한 내용은 Amazon S3 API 참조의 DeleteTable 섹션을 참조하세요.

AWS SDK 사용

AWS SDK를 사용하여 Amazon S3에서 메타데이터 테이블 구성을 삭제할 수 있습니다. 자세한 내용은 Amazon S3 API 참조의 지원되는 SDK 목록을 참조하세요.

메타데이터 테이블 쿼리

Amazon S3 메타데이터를 사용하면 Apache Iceberg 형식을 지원하는 모든 쿼리 엔진을 사용하여 S3 관리형 메타데이터 테이블을 분석할 수 있습니다. S3 테이블 버킷을 AWS 분석 서비스와 통합할 때 Amazon Athena, Amazon Redshift 등과 같은 서비스를 사용하여 메타데이터 테이블로 다음을 수행할 수 있습니다.

- 스토리지 사용 패턴 및 추세 알아보기
- 객체 전반에서 AWS Key Management Service(AWS KMS) 암호화 키 사용량 감사
- 사용자 정의 메타데이터 및 객체 태그로 객체 검색
- 시간 경과에 따른 객체 메타데이터 변경 사항 이해
- 요청을 수행한 AWS 계정 ID 또는 IP 주소를 포함하여 객체가 업데이트되거나 삭제되는 시점 알아보기

테이블 버킷을 AWS 분석 서비스와 통합한 후 메타데이터 테이블을 쿼리할 수 있습니다. 여기에는 S3 관리형 메타데이터 테이블과 사용자 지정 메타데이터 테이블의 조인이 포함되며, 메타데이터 테이블이 동일한 테이블 버킷에 저장되어 있는 한 여러 데이터세트에서 쿼리할 수 있습니다.

여기에서 Amazon QuickSight를 사용하여 쿼리 데이터로 대화형 대시보드를 생성할 수 있습니다.

쿼리 요금 고려 사항

메타데이터 테이블에서 쿼리를 실행하는 경우 추가 요금이 적용됩니다. 자세한 내용은 사용 중인 쿼리 엔진의 요금 정보를 참조하세요.

쿼리의 비용 효과를 높이는 방법에 대한 자세한 내용은 메타데이터 테이블 쿼리 성능 최적화 섹션을 참조하세요.

주제

- AWS 분석 서비스를 사용하여 메타데이터 테이블 쿼리
- 오픈 소스 쿼리 엔진을 사용하여 메타데이터 테이블 쿼리
- 메타데이터 테이블 쿼리 성능 최적화
- 메타데이터 테이블 쿼리 예시
- S3 메타데이터 테이블과 사용자 지정 메타데이터 조인
- Amazon QuickSight를 사용하여 메타데이터 테이블 데이터 시각화

AWS 분석 서비스를 사용하여 메타데이터 테이블 쿼리

Amazon Athena, Amazon Redshift 및 Amazon EMR과 같은 AWS 분석 서비스를 사용하여 S3 관리형 메타데이터 테이블을 쿼리할 수 있습니다.

쿼리를 실행하려면 먼저 AWS 계정 및 리전의 S3 테이블 버킷을 AWS 분석 서비스와 통합해야 합니다.

Amazon Athena를 사용하여 메타데이터 테이블 쿼리

S3 테이블 버킷을 AWS 분석 서비스와 통합한 후 Athena에서 메타데이터 테이블 쿼리를 시작할 수 있습니다. 쿼리에서 카탈로그를 `s3tablescatalog`로 지정하고 데이터베이스를 `aws_s3_metadata`(메타데이터 테이블의 네임스페이스)로 지정합니다. 자세한 내용은 Querying Amazon S3 tables with Athena를 참조하세요.

Amazon Redshift를 사용하여 메타데이터 테이블 쿼리

S3 테이블 버킷을 AWS 분석 서비스와 통합한 후 메타데이터 테이블 네임스페이스(`aws_s3_metadata`)에 대한 리소스 링크를 생성합니다. 완료되면 Amazon Redshift 콘솔에서 메타데이터 테이블 쿼리를 시작할 수 있습니다. 자세한 내용은 Accessing Amazon S3 tables with Amazon Redshift를 참조하세요.

Amazon EMR을 사용하여 메타데이터 테이블 쿼리

Amazon EMR을 사용하여 메타데이터 테이블을 쿼리하려면 Apache Iceberg용으로 구성된 Amazon EMR 클러스터를 생성하고 Apache Spark를 사용하여 메타데이터 테이블에 연결합니다. S3 테이블 버킷을 AWS 분석 서비스와 통합하거나 오픈 소스 Amazon S3 Tables Catalog for Iceberg 클라이언트 카탈로그를 사용하여 이를 설정할 수 있습니다.

자세한 내용은 Accessing Amazon S3 tables with Amazon EMR을 참조하세요.

오픈 소스 쿼리 엔진을 사용하여 메타데이터 테이블 쿼리

Apache Spark와 같은 오픈 소스 쿼리 엔진을 사용하여 S3 관리형 메타데이터 테이블을 쿼리할 수 있습니다. 메타데이터 테이블을 쿼리하려면 Amazon S3 Tables Catalog for Apache Iceberg 클라이언트 카탈로그(AWS Labs에서 호스팅하는 오픈 소스 라이브러리)가 필요합니다.

자세한 내용은 Querying Amazon S3 tables with Apache Spark를 참조하세요.

메타데이터 테이블 쿼리 성능 최적화

S3 메타데이터는 Apache Iceberg 테이블 형식을 기반으로 하므로 특정 시간 범위를 사용하여 메타데이터 테이블 쿼리의 성능과 비용을 최적화할 수 있습니다.

예를 들어 다음 SQL 쿼리는 S3 범용 버킷에 있는 새 객체의 민감도 수준을 제공합니다.

```
SELECT key, object_tags['SensitivityLevel']
FROM aws_s3_metadata.my_metadata_table
WHERE record_type = 'CREATE'
GROUP BY object_tags['SensitivityLevel']
```

이 쿼리는 전체 메타데이터 테이블을 스캔하므로 실행하는 데 시간이 오래 걸릴 수 있습니다. 성능을 개선하기 위해 `record_timestamp` 열을 포함하여 특정 시간 범위에 집중할 수 있습니다. 다음은 지난 달의 새 객체를 살펴보는 이전 쿼리의 업데이트된 버전입니다.

```
SELECT key, object_tags['SensitivityLevel']
FROM aws_s3_metadata.my_metadata_table
WHERE record_type = 'CREATE'
AND record_timestamp > (CURRENT_TIMESTAMP - interval '1' month)
GROUP BY object_tags['SensitivityLevel']
```

메타데이터 테이블 쿼리 예시

다음 예시에서는 표준 SQL 쿼리를 사용하여 S3 메타데이터 테이블에서 다양한 유형의 정보를 가져오는 방법을 보여줍니다.

다음 예시를 사용할 때는 다음 사항에 유의하세요.

- 이 예시는 Amazon Athena에서 작동하도록 작성되었습니다. 다른 쿼리 엔진으로 작업하려면 예시를 수정해야 할 수 있습니다.
- 쿼리를 최적화하는 방법을 이해해야 합니다.

- *amzn-s3-demo-bucket*을 메타데이터 테이블을 저장하는 S3 테이블 버킷의 이름으로 바꿉니다.
- *my_metadata_table*을 쿼리하려는 메타데이터 테이블의 이름으로 바꿉니다.
- 지원되는 열의 전체 목록은 S3 메타데이터 테이블 스키마 섹션을 참조하세요.

파일 확장명으로 객체 찾기

다음 쿼리는 특정 파일 확장명(이 경우 .jpg)이 있는 객체를 반환합니다.

```
SELECT key FROM "s3tablescatalog/amzn-s3-demo-
bucket"."aws_s3_metadata"."my_metadata_table"
WHERE key LIKE '%.jpg'
AND record_type = 'CREATE'
```

객체 삭제 나열

다음 쿼리는 요청을 수행한 AWS 계정 ID 또는 AWS 서비스 위탁자를 포함하여 객체 삭제 이벤트를 반환합니다.

```
SELECT DISTINCT bucket, key, sequence_number, record_type, record_timestamp, requester,
 source_ip_address, version_id
FROM "s3tablescatalog/amzn-s3-demo-bucket"."aws_s3_metadata"."my_metadata_table"
WHERE record_type = 'DELETE';
```

객체에서 사용하는 AWS KMS 암호화 키 나열

다음 쿼리는 객체를 암호화하는 AWS Key Management Service(AWS KMS) 키의 ARN을 반환합니다.

```
SELECT DISTINCT kms_key_arn
FROM "s3tablescatalog/amzn-s3-demo-bucket"."aws_s3_metadata"."my_metadata_table";
```

KMS 키를 사용하지 않는 객체 나열

다음 쿼리는 AWS KMS 키로 암호화되지 않은 객체를 반환합니다.

```
SELECT DISTINCT kms_key_arn
FROM "s3tablescatalog/amzn-s3-demo-bucket"."aws_s3_metadata"."my_metadata_table"
WHERE encryption_status NOT IN ('SSE-KMS', 'DSSE-KMS')
AND record_type = 'CREATE';
```

Amazon Bedrock에서 제공하는 메타데이터 보기

일부 AWS 서비스(예: Amazon Bedrock)는 Amazon S3에 객체를 업로드합니다. 이러한 서비스에서 제공하는 객체 메타데이터를 쿼리할 수 있습니다. 예를 들어, 다음 쿼리에는 Amazon Bedrock에서 범용 버킷에 업로드한 객체가 있는지 확인하는 `user_metadata` 열이 포함됩니다.

```
SELECT DISTINCT bucket, key, sequence_number, record_type, record_timestamp,
 user_metadata
FROM "s3tablescatalog/amzn-s3-demo-bucket"."aws_s3_metadata"."my_metadata_table"
WHERE record_type = 'CREATE'
AND user_metadata['content-source'] = 'AmazonBedrock';
```

Amazon Bedrock이 버킷에 객체를 업로드한 경우 쿼리 결과에 객체와 연결된 다음 메타데이터가 `user_metadata` 열에 표시됩니다.

```
user_metadata
{content-additional-params -> requestid="CVK8FWYRW0M9JW65",
 signedContentSHA384="38b060a751ac96384cd9327eb1b1e36a21fdb71114be07434c0cc7bf63f6e1da274edebfe
 content-model-id -> bedrock-model-arn, content-source -> AmazonBedrock}
```

현재 객체 상태의 이해

다음 쿼리는 객체의 현재 상태를 판단하는 데 도움이 될 수 있습니다. 쿼리는 각 객체의 최신 버전을 식별하고, 삭제된 객체를 필터링하고, 시퀀스 번호를 기반으로 각 객체의 최신 버전을 표시합니다. 결과는 `bucket`, `key` 및 `sequence_number` 열별로 정렬됩니다.

```
WITH records_of_interest as (
   -- Start with a query that can narrow down the records of interest.
    SELECT * from "s3tablescatalog/amzn-s3-demo-
bucket"."aws_s3_metadata"."my_metadata_table"
),

version_stacks as (
   SELECT *,
          -- Introduce a column called 'next_sequence_number', which is the next larger
          -- sequence_number for the same key version_id in sorted order.
          LEAD(sequence_number, 1) over (partition by (bucket, key,
 coalesce(version_id, '')) order by sequence_number ASC) as next_sequence_number
   from records_of_interest
),
```

```
-- Pick the 'tip' of each version stack triple: (bucket, key, version_id).
-- The tip of the version stack is the row of that triple with the largest sequencer.
-- Selecting only the tip filters out any row duplicates.
-- This isn't typical, but some events can be delivered more than once to the table
-- and include rows that might no longer exist in the bucket (since the
-- table contains rows for both extant and extinct objects).
-- In the next subquery, eliminate the rows that contain deleted objects.
current_versions as (
    SELECT * from version_stacks where next_sequence_number is NULL
),

-- Eliminate the rows that are extinct from the bucket by filtering with
-- record_type. An object version has been deleted from the bucket if its tip is
-- record_type==DELETE.
existing_current_versions as (
    SELECT * from current_versions where not (record_type = 'DELETE' and
 is_delete_marker = FALSE)
),

-- Optionally, to determine which of several object versions is the 'latest',
-- you can compare their sequence numbers. A version_id is the latest if its
-- tip's sequencer is the largest among all other tips in the same key.
with_is_latest as (
    SELECT *,
           -- Determine if the sequence_number of this row is the same as the largest
 sequencer for the key that still exists.
           sequence_number = (MAX(sequence_number) over (partition by (bucket, key)))
 as is_latest_version
    FROM existing_current_versions
)

SELECT * from with_is_latest
ORDER BY bucket, key, sequence_number;
```

S3 메타데이터 테이블과 사용자 지정 메타데이터 조인

S3 관리형 메타데이터 테이블과 고객(자체 관리형) 메타데이터 테이블의 데이터를 분석할 수 있습니다. 표준 SQL `JOIN` 연산자를 사용하여 이러한 여러 소스에서 데이터를 쿼리할 수 있습니다.

다음 예시 SQL 쿼리는 S3 관리형 메타데이터 테이블(*`my_s3_metadata_table`*)과 자체 관리형 메타데이터 테이블(*`my_self_managed_metadata_table`*) 간에 일치하는 레코드를 찾습니다. 또한 쿼리는 새 객체(또는 객체의 새 버전)가 버킷에 작성되었음을 나타내는 `CREATE` 이벤트를 기반으로 정보를 필터링합니다. 자세한 내용은 S3 메타데이터 테이블 스키마 단원을 참조하십시오.

```
SELECT *
FROM aws_s3_metadata.my_s3_metadata_table a
JOIN my_namespace.my_self_managed_metadata_table b
ON a.bucket = b.bucket AND a.key = b.key AND a.version_id = b.version_id
WHERE a.record_type = 'CREATE';
```

Amazon QuickSight를 사용하여 메타데이터 테이블 데이터 시각화

Amazon QuickSight를 사용하면 대화형 대시보드를 생성하여 S3 관리형 메타데이터 테이블에 대한 SQL 쿼리 결과를 분석하고 시각화할 수 있습니다. QuickSight 대시보드를 사용하면 통계를 모니터링하고, 변경 사항을 추적하고, 메타데이터 테이블에 대한 운영 인사이트를 얻을 수 있습니다.

메타데이터 테이블에 대한 대시보드에 다음이 표시될 수 있습니다.

- 서로 다른 스토리지 클래스에 있는 객체의 수
- 스토리지 데이터에 있는 객체 중 상대적으로 작은 객체의 비율
- 버킷에 있는 객체의 유형
- 삭제 대비 객체 업로드의 비율

S3 테이블 버킷을 AWS 분석 서비스와 통합한 후 메타데이터 테이블에서 데이터세트를 생성하고 SPICE를 사용하거나 쿼리 엔진에서 직접 SQL 쿼리를 사용하여 Amazon QuickSight에서 작업할 수 있습니다. QuickSight는 Amazon Athena 및 Amazon Redshift를 데이터 소스로 지원합니다.

자세한 내용은 Visualizing table data with Amazon QuickSight 섹션을 참조하세요.

객체 업로드

Amazon S3에 파일을 업로드하면 이 파일은 S3 객체로 저장됩니다. 객체는 파일 데이터 및 그 객체를 설명하는 메타데이터로 구성됩니다. 또한 버킷에 저장할 수 있는 객체 수에는 제한이 없습니다. Amazon S3 버킷에 파일을 업로드하려면 해당 버킷에 대한 쓰기 권한이 있어야 합니다. 액세스 권한에 대한 자세한 내용은 Amazon S3의 ID 및 액세스 관리을 참조하십시오.

이미지, 백업, 데이터, 동영상 등 모든 유형의 파일을 S3 버킷에 업로드할 수 있습니다. Amazon S3 콘솔을 사용하여 업로드할 수 있는 파일의 최대 크기는 160GB입니다. 160GB가 넘는 파일을 업로드하려면 AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하십시오.

버전 관리를 사용하는 버킷에 이미 키 이름이 있는 객체를 업로드하면 Amazon S3는 기존 객체를 대체하는 대신 객체의 다른 버전을 만듭니다. 버전 관리 활성화에 대한 자세한 내용은 버킷에 버전 관리 사용 설정 섹션을 참조하세요.

업로드하는 데이터의 크기에 따라 Amazon S3는 다음과 같은 옵션을 제공합니다.

- AWS SDK, REST API 또는 AWS CLI를 사용하여 단일 작업으로 객체 업로드 - 단일 PUT 작업으로 최대 5GB 크기의 단일 객체를 업로드할 수 있습니다.
- Amazon S3 콘솔을 사용하여 단일 객체 업로드 - Amazon S3 콘솔을 사용하면 최대 160GB 크기의 단일 객체를 업로드할 수 있습니다.
- AWS SDK, REST API 또는 AWS CLI를 사용하여 부분으로 나누어 객체 업로드 - 멀티파트 업로드 API 작업을 사용하여 최대 5TB 크기의 단일 대형 객체를 업로드할 수 있습니다.

  멀티파트 업로드 API 작업은 대용량 객체의 업로드 경험을 개선하기 위해 설계되었습니다. 객체를 부분별로 업로드할 수 있습니다. 이러한 객체 부분은 임의의 순서로 독립적으로, 그리고 병렬적으로 업로드할 수 있습니다. 크기가 5MB에서 5TB까지인 객체에 대해 멀티파트 업로드를 사용할 수 있습니다. 자세한 내용은 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 섹션을 참조하세요.

객체를 업로드할 때 기본적으로 Amazon S3 관리형 키(SSE-S3)를 통한 서버 측 암호화를 사용하여 객체가 자동으로 암호화됩니다. 다운로드하면 객체의 암호가 해독됩니다. 자세한 내용은 Amazon S3 버킷에 대한 기본 서버 측 암호화 동작 설정 및 암호화로 데이터 보호(을)를 참조하세요.

객체를 업로드할 때 다른 유형의 기본 암호화를 사용하려는 경우 S3 PUT 요청에서 AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 사용하여 서버 측 암호화를 지정하거나 SSE-KMS를 사용하여 데이터를 암호화하도록 대상 버킷의 기본 암호화 구성을 설정할 수도 있습니다. SSE-KMS에 대한 자세한 내용은 AWS KMS(SSE-KMS)를 사용한 서버 측 암호화 지정 단원을 참조하십시오. 다른 계정에서 소유한 KMS 키를 사용하려면 먼저 해당 키에 대한 사용 권한이 있어야 합니다. KMS 키의 크로스 계정 권한에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서에서 다른 계정에서 사용할 수 있는 KMS 키 만들기를 참조하십시오.

Amazon S3에서 액세스 거부됨(403 Forbidden) 오류가 발생하는 경우 Amazon S3의 액세스 거부(403 Forbidden) 오류 문제 해결 섹션에서 일반적인 원인에 대해 자세히 알아보세요.

객체 업로드

S3 콘솔 사용

이 절차는 콘솔을 사용하여 Amazon S3 버킷에 객체 및 폴더를 업로드하는 방법을 설명합니다.

객체를 업로드할 때 객체 키 이름은 파일 이름과 선택적 접두사입니다. Amazon S3 콘솔에서 폴더를 생성하여 객체를 구성할 수 있습니다. Amazon S3에서 폴더는 객체 키 이름에 나타나는 접두사로 표시됩니다. Amazon S3 콘솔의 폴더에 개별 객체를 업로드하면 객체 키 이름에 폴더 이름이 포함됩니다.

예를 들어 이름이 sample1.jpg인 객체를 이름이 backup인 폴더에 업로드하는 경우 키 이름은 backup/sample1.jpg입니다. 그러나 콘솔에는 해당 객체가 sample1.jpg 폴더의 backup로 표시됩니다. 키 이름에 대한 자세한 내용은 객체 메타데이터 작업 단원을 참조하십시오.

> **Note**
>
> Amazon S3 콘솔에서 객체 이름을 바꾸거나 스토리지 클래스, 암호화 또는 메타데이터 등의 속성을 변경하면 새 객체가 생성되어 이전 객체를 대체합니다. S3 버전 관리가 사용 설정된 경우 객체의 새 버전이 생성되고 기존 객체는 이전 버전이 됩니다. 또한 속성을 변경하는 역할도 새 객체(또는 객체 버전)의 소유자가 됩니다.

폴더를 업로드하면 Amazon S3는 지정된 폴더의 모든 파일과 하위 폴더를 버킷으로 업로드합니다. 그러면 업로드된 파일 이름과 폴더 이름을 조합하여 객체 키 이름이 지정됩니다. 예를 들어 /images 및 sample1.jpg라는 두 개의 파일을 포함하는 폴더 sample2.jpg를 업로드하는 경우 Amazon S3는 파일을 업로드한 후 해당 키 이름인 images/sample1.jpg 및 images/sample2.jpg를 할당합니다. 키 이름에는 폴더 이름이 접두사로 포함됩니다. Amazon S3 콘솔에는 마지막 / 이후의 키 이름 부분만 표시됩니다. 예를 들어, images 폴더에서 images/sample1.jpg 및 images/sample2.jpg 객체는 sample1.jpg 및 sample2.jpg로 표시됩니다.

S3 버킷에 폴더 및 파일 업로드

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 목록에서 폴더 또는 파일을 업로드할 버킷 이름을 선택합니다.
4. 업로드를 선택합니다.
5. 업로드 창에서 다음 중 하나를 수행합니다.
   - 파일 및 폴더를 업로드 창으로 끌어다 놓습니다.
   - 파일 추가 또는 폴더 추가를 선택하고 업로드할 파일 또는 폴더를 선택한 후 열기를 선택합니다.
6. 버전 관리를 사용 설정하려면 대상에서 버킷 버전 관리 사용 설정를 선택합니다.
7. 추가 업로드 옵션을 구성하지 않고 나열된 파일 및 폴더를 업로드하려면 페이지 하단에서 업로드를 선택합니다.

   Amazon S3가 객체와 폴더를 업로드합니다. 업로드가 완료되면 업로드: 상태 페이지에서 성공 메시지를 볼 수 있습니다.

추가 객체 속성 구성

1. 액세스 제어 목록 권한을 변경하려면 권한(Permissions)을 선택합니다.
2. 액세스 제어 목록(ACL)(Access control list (ACL))에서 권한을 편집합니다.

   객체 액세스 권한에 대한 자세한 내용은 S3 콘솔을 사용하여 객체에 대한 ACL 권한 설정를 참조하십시오. 현재 업로드 중인 모든 파일에 대해 대중(전 세계 모든 사람)에게 객체에 대한 읽기 액세스 권한을 부여할 수 있습니다. 그러나 퍼블릭 읽기 액세스의 기본 설정은 변경하지 않는 것이 좋습니다. 퍼블릭 읽기 액세스 권한 부여는 웹 사이트에 버킷을 사용하는 경우와 같은 소수의 사용 사례에 적용될 수 있습니다. 객체를 업로드한 후에도 언제든 객체 권한을 변경할 수 있습니다.
3. 다른 추가 속성을 구성하려면 속성(Properties)을 선택합니다.
4. 스토리지 클래스에서 업로드하려는 파일의 스토리지 클래스를 선택합니다.

   스토리지 클래스에 대한 자세한 정보는 Amazon S3 스토리지 클래스 이해 및 관리를 참조하세요.
5. 객체에 대한 암호화 설정을 업데이트하려면 서버 측 암호화 설정에서 다음을 수행합니다.

   a. 암호화 키 지정(Specify an encryption key)을 선택합니다.

   b. 암호화 설정에서 버킷 기본 암호화 설정 사용 또는 기본 암호화에 버킷 설정 재정의를 선택합니다.

   c. 기본 암호화에 버킷 설정 재정의를 선택한 경우 다음과 같은 암호화 설정을 구성해야 합니다.

      - Amazon S3에서 관리하는 키를 사용하여 업로드된 파일을 암호화하려면 Amazon S3 관리형 키(SSE-S3)를 선택합니다.

        자세한 내용은 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) 사용 섹션을 참조하세요.
      - AWS Key Management Service(AWS KMS)에 저장된 키를 사용하여 업로드된 파일을 암호화하려면 AWS Key Management Service 키(SSE-KMS)를 선택합니다. 그리고 나서 AWS KMS 키에 다음 옵션 중 하나를 선택합니다.
        - 사용 가능한 KMS 키 목록에서 AWS KMS keys 중에서 선택을 선택한 다음, 사용 가능한 키 목록에서 KMS 키를 선택합니다.

          AWS 관리형 키(aws/s3)와 고객 관리형 키가 모두 목록에 표시됩니다. 고객 관리형 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 키 및 AWS 키를 참조하세요.
        - KMS 키 ARN을 입력하려면 AWS KMS key ARN 입력을 선택한 다음 나타나는 필드에 KMS 키 ARN을 입력합니다.

- AWS KMS 콘솔에서 고객 관리형 키를 생성하려면 KMS 키 생성을 선택합니다.

  AWS KMS key 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 키 생성을 참조하십시오.

> **Important**
>
> 버킷과 동일한 AWS 리전에서 사용할 수 있는 KMS 키만 사용 가능합니다. Amazon S3 콘솔은 버킷과 동일한 리전에 있는 처음 100개의 KMS 키만 나열합니다. 목록에 없는 KMS 키를 사용하려면 KMS 키 ARN을 입력해야 합니다. 다른 계정에서 소유한 KMS 키를 사용하려면 먼저 해당 키에 대한 사용 권한이 있어야 하고, 다음 단계로 KMS 키 ARN을 입력해야 합니다.
> Amazon S3는 대칭 암호화 KMS 키만 지원하며 비대칭 KMS 키는 지원하지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 대칭 및 비대칭 KMS 키 식별을 참조하세요.

6. 추가 체크섬을 사용하려면 켜기(On)를 선택합니다. 그런 다음 체크섬 함수(Checksum function)에서 사용할 함수를 선택합니다. Amazon S3는 전체 객체를 수신한 후 체크섬 값을 계산하고 저장합니다. 미리 계산된 값(Precalculated value) 상자를 사용하여 미리 계산된 값을 제공할 수 있습니다. 이렇게 하면 Amazon S3가 제공한 값과 계산한 값을 비교합니다. 두 값이 일치하지 않으면 Amazon S3가 오류를 생성합니다.

   추가 체크섬을 사용하면 데이터를 확인하는 데 사용할 체크섬 알고리즘을 지정할 수 있습니다. 추가 체크섬에 대한 자세한 내용은 Amazon S3에서 객체 무결성 확인 단원을 참조하십시오.

7. 업로드 중인 모든 객체에 태그를 추가하려면 태그 추가를 선택합니다. 키 필드에 태그 이름을 입력합니다. 태그 값을 입력합니다.

   객체 태그 지정을 통해 스토리지를 분류할 수 있습니다. 각 태그는 키-값 페어입니다. 키와 태그 값은 대/소문자를 구분합니다. 객체마다 태그를 10개까지 포함할 수 있습니다. 태그 키는 최대 128개 길이의 유니코드 문자이며, 태그 값은 최대 255개의 유니코드 문자입니다. 객체 태그에 대한 자세한 내용은 태그를 사용하여 스토리지 분류 섹션을 참조하세요.

8. 메타데이터를 추가하려면 메타데이터 추가를 선택합니다.

   a. 유형에서 시스템 정의 또는 사용자 정의를 선택합니다.

      시스템 정의 메타데이터의 경우, 콘텐츠 유형 및 콘텐츠 처리 같은 일반적인 HTTP 헤더를 선택할 수 있습니다. 시스템 정의 메타데이터 목록과 값 추가 가능 여부를 확인하려면 시스템 정의 객체 메타데이터 단원을 참조하십시오. 접두사 `x-amz-meta-`로 시작하는 모든 메타데이

터는 사용자 정의 메타데이터로 처리됩니다. 사용자 정의 메타데이터는 객체와 함께 저장되었다가 해당 객체를 다운로드할 때 반환됩니다. 키와 값 모두 US-ASCII 표준에 부합해야 합니다. 사용자 정의 메타데이터의 최대 크기는 2KB입니다. 시스템 정의 메타데이터와 사용자 정의 메타데이터에 대한 자세한 내용은 객체 메타데이터 작업 단원을 참조하십시오.

b. 키에 대해 키를 선택합니다.

c. 키 값을 입력합니다.

9. 객체를 업로드하려면 업로드를 선택합니다.

   Amazon S3가 객체를 업로드합니다. 업로드가 완료되면 업로드: 상태 페이지에서 성공 메시지를 볼 수 있습니다.

10. 종료를 선택합니다.

AWS CLI 사용

PUT 요청을 보내 한 번의 작업으로 최대 5GB의 객체를 업로드할 수 있습니다. 자세한 내용은 AWS CLI 명령 참조의 `PutObject` 예시를 참조하십시오.

REST API 사용

객체를 업로드하기 위해 REST 요청을 보낼 수 있습니다. PUT 요청을 전송하여 단일 작업에서 데이터를 업로드할 수 있습니다. 자세한 내용은 PUT Object 단원을 참조하십시오.

AWS SDK 사용

Amazon S3에서 AWS SDK를 사용하여 객체를 업로드할 수 있습니다. SDK는 데이터를 간편하게 업로드할 수 있는 래퍼 라이브러리를 제공합니다. 자세한 내용은 지원되는 SDK 목록을 참조하십시오.

다음은 엄선된 SDK가 포함된 예입니다.

.NET

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.

코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

```
using Amazon;
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");
                PutObjectResponse response2 = await
 client.PutObjectAsync(putRequest2);
            }
            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);
            }
        }
    }
}
```

Java

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 AWS SDK for Java 개발자 안내서의 시작하기 섹션을 참조하세요.

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

JavaScript

다음 예시에서는 특정 리전의 Amazon S3 버킷에 기존 파일을 업로드합니다.

```
import { readFile } from "node:fs/promises";

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

/**
 * Upload a file to an S3 bucket.
 * @param {{ bucketName: string, key: string, filePath: string }}
 */
export const main = async ({ bucketName, key, filePath }) => {
  const client = new S3Client({});
  const command = new PutObjectCommand({
    Bucket: bucketName,
    Key: key,
    Body: await readFile(filePath),
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
```

```
      caught.name === "EntityTooLarge"
    ) {
      console.error(
        `Error from S3 while uploading object to ${bucketName}. \
The object was too large. To upload objects larger than 5GB, use the S3 console
 (160GB max) \
or the multipart upload API (5TB max).`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while uploading object to ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

PHP

이 예시는 AWS SDK for PHP의 클래스를 사용하여 최대 5GB 크기의 객체를 업로드하는 방법을 설명합니다. 더 큰 파일의 경우 멀티파트 업로드 API 작업을 사용해야 합니다. 자세한 내용은 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 섹션을 참조하세요.

AWS SDK for Ruby API에 대한 자세한 내용은 AWS SDK for Ruby – 버전 2를 참조하세요.

Example - 데이터를 업로드하여 Amazon S3 버킷에 객체 생성

다음 PHP 예제는 `putObject()` 메서드를 사용한 데이터 업로드를 통해 지정된 버킷에 객체를 생성합니다.

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

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

$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;
}
```

Ruby

AWS SDK for Ruby - 버전 3은 Amazon S3로 객체를 업로드하는 2가지 방법을 제공합니다. 첫 번째 방법은 더 간편하게 디스크에서 모든 크기의 파일을 업로드할 수 있는 관리형 파일 업로더를 사용합니다. 관리형 파일 업로더 방법 사용:

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'

# Wraps Amazon S3 object actions.
class ObjectUploadFileWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end
```

```
  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

  wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  return unless wrapper.upload_file(file_path)

  puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

AWS SDK for Ruby 버전 3에서 객체를 업로드하는 두 번째 방법은 #put의 `Aws::S3::Object` 메서드를 사용하는 것입니다. 객체가 문자열이거나 디스크에 저장된 파일이 아닌 I/O 객체인 경우 이 방법이 유용합니다. 이 방법 사용:

1. `Aws::S3::Resource` 클래스의 인스턴스를 만듭니다.
2. 버킷 이름과 키로 대상 객체를 참조합니다.
3. 문자열 또는 I/O 객체를 전달하는 #put을 호출합니다.

Example

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

```
# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, 'rb') do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)
  return unless success

  puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

동일한 키 이름을 가진 객체 업로드 방지

업로드 시 조건부 쓰기 작업을 사용하여 객체를 생성하기 전에 버킷에 객체가 있는지 확인할 수 있습니다. 이렇게 하면 기존 데이터의 덮어쓰기를 방지할 수 있습니다. 조건부 쓰기는 업로드하는 동안 버킷에 이미 같은 키 이름을 가진 기존 객체가 없는지 확인합니다.

PutObject 또는 CompleteMultipartUpload 요청에 조건부 쓰기를 사용할 수 있습니다.

조건부 요청에 대한 자세한 내용은 조건부 요청을 통해 S3 작업에 사전 조건 추가 섹션을 참조하세요.

Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사

멀티파트 업로드를 사용하면 단일 객체를 여러 파트의 집합으로 Amazon S3에 업로드할 수 있습니다. 각 부분은 객체 데이터의 연속적인 부분입니다. 이러한 객체 파트는 독립적으로 그리고 임의의 순서로 업로드할 수 있습니다. 업로드의 경우 업데이트된 AWS 클라이언트는 객체의 체크섬을 자동으로 계산하여 요청의 일부로 객체 크기와 함께 Amazon S3로 전송합니다. 부분의 전송이 실패할 경우 다른 부분에 영향을 주지 않고도 해당 부분을 재전송할 수 있습니다. 객체의 모든 파트가 업로드되면 Amazon S3가 이 파트를 수집하여 객체를 생성합니다. 100MB 이상의 객체인 경우 단일 작업으로 업로드하는 대신 멀티파트 업로드를 사용하는 것이 좋습니다.

멀티파트 업로드 사용은 다음 이점을 제공합니다.

- 개선된 처리량 개선 - 부분을 병렬적으로 업로드하여 처리량을 개선할 수 있습니다.
- 네트워크 문제로부터 빠른 복구 - 더 작아진 부분 크기는 네트워크 오류로 인해 실패한 업로드 재시작의 영향을 최소화합니다.
- 객체 업로드 일시 중지 및 재개 – 객체 부분을 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 생성하면서 업로드할 수 있습니다.

다음 방법으로 멀티파트 업로드를 사용하는 것이 좋습니다.

- 안정적인 높은 대역폭 네트워크를 통해 큰 객체를 업로드하는 경우, 멀티파트 업로드를 사용하여 멀티스레드 성능을 위해 여러 객체 파트를 동시에 업로드함으로써 대역폭 사용을 극대화합니다.
- 불규칙한 네트워크를 통해 업로드하는 경우, 멀티파트 업로드를 사용하여 업로드가 다시 시작되는 것을 방지하여 네트워크 오류에 대한 복원력을 높입니다. 멀티파트 업로드를 사용하는 경우, 업로드 중에 중단된 파트만 다시 업로드를 시도하면 됩니다. 객체 업로드를 처음부터 다시 시작하지 않아도 됩니다.

> **Note**
>
> Amazon S3 Express One Zone 스토리지 클래스를 디렉터리 버킷과 함께 사용하는 방법에 대한 자세한 내용은 S3 Express One Zone 및 디렉터리 버킷 작업 섹션을 참조하세요. S3

Express One Zone 및 디렉터리 버킷에서 멀티파트 업로드를 사용하는 방법에 대한 자세한 내용은 디렉터리 버킷에 멀티파트 업로드 사용 섹션을 참조하세요.

멀티파트 업로드 프로세스

멀티파트 업로드는 3단계 프로세스입니다. 업로드를 시작하고, 객체 파트를 업로드하고, 모두 업로드했으면 멀티파트 업로드를 완료합니다. 멀티파트 업로드 완료 요청이 전송되면 Amazon S3는 업로드된 각 파트를 모아 완전한 객체를 구성합니다. 버킷의 다른 객체와 마찬가지로 이 객체에 액세스할 수 있습니다.

진행 중인 모든 멀티파트 업로드 작업의 목록 또는 특정 멀티파트 업로드에 대해 업로드한 부분의 목록을 확인할 수 있습니다. 다음 섹션에서 이러한 각 작업에 대해 자세히 설명합니다.

멀티파트 업로드 시작

멀티파트 업로드 시작 요청을 보낼 때는 체크섬 유형을 지정해야 합니다. 그런 다음 Amazon S3는 멀티파트 업로드의 고유 식별자인 업로드 ID가 포함된 응답을 반환합니다. 파트를 업로드하거나 파트를 나열하거나 업로드를 완료하거나 업로드를 중지할 때 이 업로드 ID가 필요합니다. 업로드되는 객체를 설명하는 메타데이터를 제공하려는 경우 멀티파트 업로드 시작 요청에서 제공해야 합니다. 익명 사용자는 멀티파트 업로드를 시작할 수 없습니다.

파트 업로드

파트를 업로드할 때 업로드 ID와 함께 파트 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 파트와 동일한 파트 번호로 새 파트를 업로드할 경우 이전에 업로드한 파트를 덮어쓰게 된다는 것을 유의하세요.

파트를 업로드하면 Amazon S3는 각 파트의 체크섬 값과 함께 체크섬 알고리즘 유형을 응답의 헤더로 반환합니다. 각 부분 업로드에 대해 부분 번호와 ETag 값을 기록해야 합니다. 이후 멀티파트 업로드를 완료하기 위한 요청에 이러한 값을 포함해야 하기 때문입니다. 업로드 시 각 부분에는 고유한 ETag가 있습니다. 하지만 멀티파트 업로드가 완료되고 모든 파트가 통합되면 모든 파트는 체크섬의 체크섬으로 하나의 ETag에 속하게 됩니다.

Important

멀티파트 업로드를 시작하여 하나 이상의 파트를 업로드한 후에는 멀티파트 업로드를 완료하거나 중지해야 업로드된 파트의 스토리지 비용이 청구되지 않습니다. 멀티파트 업로드를 완료

또는 중단한 후에만 Amazon S3가 파트 스토리지를 비우고 파트 스토리지에 대한 비용 청구를 중지합니다.
멀티파트 업로드를 중지한 후에는 해당 업로드 ID로 다시 파트를 업로드할 수 없습니다. 파트 업로드가 진행 중일 때 멀티파트 업로드를 중지하더라도 진행 중인 파트 업로드는 성공하거나 실패할 수 있습니다. 따라서 각 파트에서 사용하고 있는 스토리지를 모두 비우려면 모든 파트 업로드가 완료된 후에 멀티파트 업로드를 중지해야 합니다.

멀티파트 업로드 완료

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 연결합니다. 성공적으로 완료 요청이 수행되면 부분은 더 이상 존재하지 않습니다.

멀티파트 업로드 완료 요청에는 업로드 ID와 각 파트 번호 및 해당 ETag 값의 목록이 포함되어야 합니다. Amazon S3 응답에는 결합된 객체 데이터를 고유하게 식별하는 ETag가 포함됩니다. 이 ETag가 반드시 객체 데이터의 MD5 해시여야 하는 것은 아닙니다.

멀티파트 업로드 중에 전체 객체 체크섬을 제공하면 AWS SDK는 체크섬을 Amazon S3에 전달하고 S3는 서버 측의 객체 무결성을 검증하여 수신된 값과 비교합니다. 그런 다음 값이 일치하면 S3가 객체를 저장합니다. 두 값이 일치하지 않으면 Amazon S3에서 `BadDigest` 오류와 함께 요청이 실패합니다. 객체의 체크섬은 나중에 객체의 데이터 무결성을 검증하는 데 사용할 객체 메타데이터에도 저장됩니다.

멀티파트 업로드 호출 샘플

이 예시에서는 100GB 파일에 대해 멀티파트 업로드를 생성한다고 가정합니다. 이 경우 전체 프로세스를 위해 다음과 같은 API 호출이 이루어집니다. 총 1,002개의 API 직접 호출이 수행됩니다.

- 프로세스를 시작하는 `CreateMultipartUpload` 호출.
- 총 100GB 크기에 대해 각각 100MB인 파트를 업로드하는 1,000개의 개별 `UploadPart` 직접 호출
- 프로세스를 완료하기 위한 `CompleteMultipartUpload` 직접 호출

멀티파트 업로드 나열

특정 멀티파트 업로드 또는 진행 중인 모든 멀티파트 업로드에 대해 부분 목록을 확인할 수 있습니다. 파트 나열 작업은 특정 멀티파트 업로드에 대해 업로드한 파트의 정보를 반환합니다. 각 부분 목록 조

회 요청에 대해 Amazon S3는 특정 멀티파트 업로드에서 최대 1,000개의 부분에 대해 부분 정보를 반환합니다. 멀티파트 업로드에서 1,000개 이상의 파트가 있을 경우 모든 파트를 검색하려면 파트 나열 요청을 여러 번 반복해야 합니다. 반환된 부분 목록에는 업로드가 완료되지 않은 부분이 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.

> **Important**
>
> 멀티파트 업로드 완료 요청을 전송할 때 이 나열 결과를 사용하면 안 됩니다. 그 대신, 부분을 업로드할 때 지정한 부분 번호 및 Amazon S3가 반환한 해당 ETag 값의 목록을 보관해야 합니다.

멀티파트 업로드 작업을 사용한 체크섬

Amazon S3에 객체를 업로드할 때 Amazon S3에서 사용할 체크섬 알고리즘을 지정할 수 있습니다. 기본적으로 AWS SDK 및 S3 콘솔은 모든 객체 업로드에 알고리즘을 사용하며, 이를 재정의할 수 있습니다. 이전 SDK를 사용하고 업로드된 객체에 지정된 체크섬이 없는 경우 Amazon S3는 CRC-64/NVME(CRC64NVME) 체크섬 알고리즘을 자동으로 사용합니다. (효율적인 데이터 무결성 확인을 위한 권장 옵션이기도 합니다.) CRC-64/NVME를 사용할 때 Amazon S3는 멀티파트 또는 단일 파트 업로드가 완료된 후 전체 객체의 체크섬을 계산합니다. CRC-64/NVME 체크섬 알고리즘은 각 개별 파트에 대해 전체 객체의 직접 체크섬 또는 체크섬의 체크섬을 계산하는 데 사용됩니다.

멀티파트 업로드를 사용하여 객체를 S3에 업로드한 후 Amazon S3는 각 파트 또는 전체 객체에 대한 체크섬 값을 계산하고 값을 저장합니다. S3 API 또는 AWS SDK를 사용하여 다음과 같은 방법으로 체크섬 값을 검색할 수 있습니다.

- 개별 파트의 경우 `GetObject` 또는 `HeadObject`를 사용할 수 있습니다. 아직 진행 중인 멀티파트 업로드의 개별 파트에 대한 체크섬 값을 검색하려면 `ListParts`를 사용하면 됩니다.
- 전체 객체에 대해 `PutObject`를 사용할 수 있습니다. 전체 객체 체크섬으로 멀티파트 업로드를 수행하려면 전체 객체 체크섬 유형을 지정하여 `CreateMultipartUpload` 및 `CompleteMultipartUpload`를 사용합니다. 전체 객체의 체크섬 값을 검증하거나 멀티파트 업로드에서 사용 중인 체크섬 유형을 확인하려면 `ListParts`를 사용합니다.

> **Important**
>
> 체크섬과 함께 멀티파트 업로드를 사용하는 경우 멀티파트 업로드에서 각 파트 업로드에 대한 파트 번호는 연속 파트 번호를 사용해야 하고 1로 시작해야 합니다. 체크섬을 사용할 때 연속되지 않는 파트 번호로 멀티파트 업로드 요청을 완료하려고 하면 Amazon S3가 `HTTP 500 Internal Server` 오류를 생성합니다.

멀티파트 업로드 객체를 통한 체크섬 작업에 대한 자세한 내용은 Amazon S3에서 객체 무결성 확인 섹션을 참조하세요.

추가 체크섬이 포함된 멀티파트 업로드를 사용하여 객체를 업로드하는 전체 절차는 튜토리얼: 멀티파트 업로드를 통한 객체 업로드 및 데이터 무결성 확인 섹션을 참조하세요.

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 후 Amazon S3가 객체를 생성하도록 업로드 완료 요청을 전송할 수 있습니다. 버킷에서 S3 버전 관리를 사용할 경우 멀티파트 업로드를 완료하면 항상 새 버전이 생성됩니다. 버전 관리가 활성화된 버킷에서 동일한 객체 키를 사용하는 멀티파트 업로드를 여러 개 시작하는 경우, 가장 최근에 시작된 업로드(`createdDate`)에 따라 객체의 현재 버전이 결정됩니다.

예를 들어 오전 10시에 객체에 대한 `CreateMultipartUpload` 요청을 시작합니다. 그런 다음 오전 11시에 같은 객체에 대한 두 번째 `CreateMultipartUpload` 요청을 제출합니다. 두 번째 요청이 가장 최근에 제출되었으므로 첫 번째 업로드가 두 번째 업로드 후에 완료되더라도 오전 11시 요청에 의해 업로드된 객체가 현재 버전이 됩니다. 버전 관리가 활성화되지 않은 버킷의 경우 멀티파트 업로드의 시작과 완료 사이에 전송된 다른 요청을 우선 수행할 수 있습니다.

동시 멀티파트 업로드 요청이 우선할 수 있는 또 다른 예는 키를 사용하여 멀티파트 업로드를 시작한 후 다른 작업이 해당 키를 삭제하는 경우입니다. 작업을 완료하기 전에 전체 멀티파트 업로드 응답은 객체를 보지 못한 상태에서도 객체를 성공적으로 생성했다고 나타낼 수 있습니다.

멀티파트 업로드 중에 키 이름이 동일한 객체 업로드 방지

업로드 시 조건부 쓰기 작업을 사용하여 객체를 생성하기 전에 버킷에 객체가 있는지 확인할 수 있습니다. 이렇게 하면 기존 데이터의 덮어쓰기를 방지할 수 있습니다. 조건부 쓰기는 업로드하는 동안 버킷에 이미 같은 키 이름을 가진 기존 객체가 없는지 확인합니다.

PutObject 또는 CompleteMultipartUpload 요청에 조건부 쓰기를 사용할 수 있습니다.

조건부 요청에 대한 자세한 내용은 조건부 요청을 통해 S3 작업에 사전 조건 추가 섹션을 참조하세요.

멀티파트 업로드 및 요금

멀티파트 업로드가 시작되면 Amazon S3는 업로드가 완료되거나 중단될 때까지 모든 부분을 계속 유지합니다. 수명 주기가 끝날 때까지 이 멀티파트 업로드와 관련 부분의 모든 스토리지, 대역폭 및 요청에 대해 비용이 청구됩니다.

이러한 파트는 파트를 업로드할 때 지정한 스토리지 클래스에 따라 요금이 부과됩니다. 그러나 이러한 파트가 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive에 업로드된 경우 해당 파트에 대한 요금이 부과되지 않습니다. S3 Glacier Flexier Flexible Retrieval 스토리지 클래스로의 PUT 요청에 멀티파트 파트가 진행 중인 경우, 업로드가 완료될 때까지 S3 Glacier Flexible Retrieval 스테이징 스토리지에 대해 S3 Standard 스토리지 요금이 부과됩니다. 또한 `CreateMultipartUpload` 및 `UploadPart` 모두 S3 Standard 요금이 부과됩니다. `CompleteMultipartUpload` 요청만 S3 Glacier Flexible Retrieval 요금이 부과됩니다. 마찬가지로, S3 Glacier Deep Archive 스토리지 클래스로의 PUT에 대한 진행 중인 멀티파트 파트는 업로드가 완료될 때까지 S3 Glacier Flexible Retrieval 스테이징 스토리지에 대해 S3 Standard 스토리지 요금이 부과되며, `CompleteMultipartUpload` 요청만 S3 Glacier Deep Archive 요금이 부과됩니다.

멀티파트 업로드를 중단하면 Amazon S3가 아티팩트 업로드 및 업로드된 모든 파트를 삭제합니다. 이러한 아티팩트에 대해서는 요금이 부과되지 않습니다. 지정한 스토리지 클래스에 관계없이 불완전한 멀티파트 업로드를 삭제하는 경우 조기 삭제 요금이 부과되지 않습니다. 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

> **Note**
>
> 스토리지 비용을 최소화하려면 `AbortIncompleteMultipartUpload` 작업을 사용하여 특정 기간이 지나면 불완전한 멀티파트 업로드를 삭제하는 수명 주기 규칙을 구성하는 것이 좋습니다. 불완전한 멀티파트 업로드를 삭제하기 위한 수명 주기 규칙을 만드는 방법과 관련하여 자세한 내용은 불완전한 멀티파트 업로드를 삭제하도록 버킷 수명 주기 구성 설정을 참조하십시오.

멀티파트 업로드를 위한 API 지원

Amazon Simple Storage Service API 참조의 다음 섹션에서는 멀티파트 업로드를 위한 REST API에 대해 설명합니다.

AWS Lambda 함수를 사용하는 멀티파트 업로드 방법에 대한 자세한 내용은 Uploading large objects to Amazon S3 using multipart upload and transfer acceleration을 참조하세요.

- 멀티파트 업로드 생성
- 부분 업로드
- 부분 업로드(복사)
- 멀티파트 업로드 완료
- 멀티파트 업로드 중단
- 부분 목록 조회
- 멀티파트 업로드 목록 조회

멀티파트 업로드를 위한 AWS Command Line Interface 지원

AWS Command Line Interface의 다음 주제에서는 멀티파트 업로드를 위한 작업을 설명합니다.

- 멀티파트 업로드 시작
- 부분 업로드
- 부분 업로드(복사)
- 멀티파트 업로드 완료
- 멀티파트 업로드 중단
- 부분 목록 조회
- 멀티파트 업로드 목록 조회

멀티파트 업로드에 대한 AWS SDK 지원

AWS SDK를 사용하여 객체를 여러 부분으로 나누어 업로드할 수 있습니다. API 작업에서 지원하는 AWS SDK 목록은 다음을 참조하십시오.

- 멀티파트 업로드 생성
- 부분 업로드
- 부분 업로드(복사)
- 멀티파트 업로드 완료
- 멀티파트 업로드 중단
- 부분 목록 조회
- 멀티파트 업로드 목록 조회

멀티파트 업로드 API 및 권한

멀티파트 업로드 작업을 수행하려면 필수 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.

작업	필수 권한
멀티파트 업로드 생성	멀티파트 업로드를 생성하려면 객체에 대해 `s3:PutObject` 작업을 수행할 수 있어야 합니다. 버킷 소유자는 다른 주체가 `s3:PutObject` 작업을 수행하도록 허가할 수 있습니다.
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.

작업	필수 권한
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.
멀티파트 업로드 중지	멀티파트 업로드를 중단하려면 객체에 대해 s3:AbortMultipartUpload 작업을 수행할 수 있어야 합니다. 기본적으로 버킷 소유자와 멀티파트 업로드를 시작한 사용자는 이 작업을 IAM 및 S3 버킷 정책의 일환으로 수행하도록 허용됩니다. 시작한 사용자가 IAM 사용자인 경우 해당 사용자의 AWS 계정도 이 멀티파트 업로드를 중단할 수 있어야 합니다. VPC 엔드포인트 정책의 경우 멀티파트 업로드를 시작한 사용자는 s3:AbortMultipartUpload 작업을 수행할 권한을 자동으로 얻지 못합니다. 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 사용자를 제어하는 AWS 계정도 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.

작업	필수 권한
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) KMS 키를 통한 암호화를 사용해 멀티파트 업로드를 수행하려면 요청자에게 다음 권한이 있어야 합니다. • 키에 대한 `kms:Decrypt` 및 `kms:GenerateDataKey` 작업 • CreateMultipartUpload API에 대한 `kms:GenerateDataKey` 작업 • UploadPart 및 UploadPartCopy API에 대한 `kms:Decrypt` 작업 이러한 권한이 필요한 이유는 Amazon S3이 멀티파트 업로드를 완료하기 전에 암호화된 파일 부분에서 데이터를 암호 해독하고 읽어야 하기 때문입니다. 객체의 체크섬 값을 얻으려면 `kms:Decrypt` 권한과 고객이 제공한 암호화 키를 사용한 서버 측 암호화도 필요합니다. CompleteMultipartUpload API를 사용할 때 이러한 필수 권한이 없는 경우 체크섬 값 없이 객체가 생성됩니다. IAM 사용자 또는 역할이 KMS 키와 동일한 AWS 계정에 있는 경우 키 및 IAM 정책 둘 다에 대한 권한이 있는지 확인하세요. IAM 사용자 또는 역할이 KMS 키와 다른 계정에 속한 경우, 키 정책과 IAM 사용자 또는 역할 모두에 대한 권한이 있어야 합니다.
SSE-C(고객 제공 암호화 키를 사용한 서버 측 암호화)	CompleteMultipartUpload API를 사용하는 경우 SSE-C(고객 제공 암호화 키를 사용한 서버 측 암호화)를 제공해야 합니다. 그렇지 않으면 체크섬 없이 객체가 생성되고 체크섬 값이 반환되지 않습니다.

ACL 권한 및 액세스 정책의 권한 간 관계에 대한 자세한 내용은 ACL 권한과 액세스 정책 권한의 매핑 단원을 참조하십시오. IAM 사용자, 역할 및 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM 자격 증명(사용자, 사용자 그룹 및 역할)을 참조하십시오.

멀티파트 업로드 작업을 사용한 체크섬

실제 멀티파트 업로드를 수행하는 데 사용되는 세 가지 Amazon S3 API는 CreateMultipartUpload, UploadPart, CompleteMultipartUpload입니다. 다음 표는 각 API에 제공해야 하는 체크섬 헤더 및 값을 나타냅니다.

체크섬 알고리즘	체크섬 유형	**CreateMultipartUpload**	**UploadPart**	**CompleteMultipartUpload**

CRC-64/NVME(CRC64NVME)	전체 객체	필수 헤더:
		`x-amz-checksum-algorithm`
			선택적 헤더:
			`x-amz-checksum-crc64nvme`
				선택적 헤더:
				`x-amz-checksum-algorithm`
				`x-amz-crc64`

CRC-32(CRC32)
CRC 32-C(CRC32C)
	전체 객체	필수 헤더:
		`x-amz-checksum-algorithm`
		`x-amz-checksum-type`
			선택적 헤더:
			`x-amz-checksum-crc64nvme`
				선택적 헤더:
				`x-amz-checksum-algorithm`
				`x-amz-crc32`
				`x-amz-crc32c`

CRC-32(CRC32)
CRC-32C(CRC32C)
SHA-1(SHA1)
SHA-256(SHA256)
	복합	필수 헤더:
		`x-amz-checksum-algorithm`
			필수 헤더:
			`x-amz-checksum-crc32`
			`x-amz-checksum-crc32c`
				필수 헤더:
				모든 파트 수준 체크섬을 CompleteMultiPartUpload 요청에 포함해야 합니다.
				선택적 헤더:

체크섬 알고리즘	체크섬 유형	**CreateMultipartUpload**	**UploadPart**	**CompleteMultipartUpoad**

			`x-amz-checksum-sha1`
			`x-amz-checksum-sha256`
				`x-amz-crc32`
				`x-amz-crc32c`
				`x-amz-sha1`
				`x-amz-sha256`

주제

- 불완전한 멀티파트 업로드를 삭제하도록 버킷 수명 주기 구성 설정
- 멀티파트 업로드를 사용한 객체 업로드
- 상위 수준 .NET TransferUtility 클래스를 사용하여 디렉터리 업로드
- 멀티파트 업로드 나열
- AWS SDK를 사용하여 멀티파트 업로드 추적
- 멀티파트 업로드 중단
- 멀티파트 업로드를 사용한 객체 복사
- 튜토리얼: 멀티파트 업로드를 통한 객체 업로드 및 데이터 무결성 확인
- Amazon S3 멀티파트 업로드 제한

불완전한 멀티파트 업로드를 삭제하도록 버킷 수명 주기 구성 설정

모범 사례로, 저장 비용을 최소화하도록 `AbortIncompleteMultipartUpload` 작업을 사용하여 수명 주기 규칙을 구성할 것을 권장합니다. 멀티파트 업로드 중단에 대한 자세한 내용은 멀티파트 업로드 중단 단원을 참조하십시오.

이제 Amazon S3는 시작된 후 지정 일수 내에 완료되지 않은 멀티파트 업로드를 중단하도록 Amazon S3에 지시하는 데 사용할 수 있는 버킷 수명 주기 규칙을 지원합니다. 지정된 기간 내에 멀티파트 업로드가 완료되지 않으면 중단 작업을 수행할 수 있습니다. 그러면 Amazon S3가 멀티파트 업로드를 중단하고 멀티파트 업로드와 관련된 모든 부분을 삭제합니다. 이 규칙은 기존 멀티파트 업로드와 나중에 생성하는 멀티파트 업로드 모두에 적용됩니다.

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 규칙이 적용됩니다. 시작된 후 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 단원을 참조하십시오.

> **Note**
>
> 규칙에 지정된 일수 내에 멀티파트 업로드가 완료되면 AbortIncompleteMultipartUpload 수명 주기 작업은 적용되지 않습니다(즉, Amazon S3가 어떤 작업도 하지 않음). 또한 이 작업이 객체에도 적용되지 않습니다. 이 수명 주기 작업으로 객체가 삭제되지는 않습니다. 또한 불완전한 멀티파트 업로드 부분을 제거하더라도 S3 수명 주기에 대한 조기 삭제 요금이 발생하지 않습니다.

S3 콘솔 사용

불완전한 멀티파트 업로드를 자동으로 관리하려면, 지정된 일수가 지난 후에 버킷에서 미완료 멀티파트 업로드 바이트를 만료시키는 수명 주기 규칙을 S3 콘솔을 사용하여 만들면 됩니다. 다음 절차는 7일 후 미완료 멀티파트 업로드를 삭제하기 위한 수명 주기 규칙을 추가하는 방법을 보여줍니다. 수명 주기 규칙에 대한 자세한 내용은 버킷에서 S3 수명 주기 구성 설정 페이지를 참조하십시오.

7일 이상 지난 미완료 멀티파트 업로드를 중단하기 위한 수명 주기 규칙을 추가하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 버킷(Buckets) 목록에서 수명 주기 규칙을 생성할 버킷의 이름을 선택합니다.
3. 관리(Management) 탭을 선택하고 수명 주기 규칙 생성(Create lifecycle rule)을 선택합니다.
4. 수명 주기 규칙 이름(Lifecycle rule name)에 규칙의 이름을 입력합니다.

   단, 버킷 내에서 고유한 이름을 갖도록 합니다.
5. 수명 주기 규칙의 범위를 선택합니다.

   - 특정 접두사가 있는 모든 객체에 수명 주기 규칙을 생성하려면 Limit the scope of this rule using one or more filters(하나 이상의 필터를 사용하여 이 규칙의 범위 제한)를 선택하고 Prefix(접두사) 필드에 접두사를 입력합니다.
   - 이 수명 주기 규칙을 버킷의 모든 객체에 적용하려면 This rule applies to all objects in the bucket(이 규칙이 버킷의 모든 객체에 적용됨)을 선택하고 I acknowledge that this rule applies to all objects in the bucket(이 규칙이 버킷의 모든 객체에 적용됨을 확인합니다)을 선택합니다.
6. Lifecycle rule actions(수명 주기 규칙 작업)에서 Delete expired object delete markers or incomplete multipart uploads(만료된 객체 삭제 마커 또는 미완료 멀티파트 업로드 삭제)를 선택합니다.
7. Delete expired object delete markers or incomplete multipart uploads(만료된 객체 삭제 마커 또는 미완료 멀티파트 업로드 삭제)에서 Delete incomplete multipart uploads(미완료 멀티파트 업로드 삭제)를 선택합니다.
8. Number of days(일수) 필드에, 미완료 멀티파트 업로드를 삭제하기까지의 경과 일수(이 예에서는 7일)를 입력합니다.
9. 규칙 생성을 선택합니다.

AWS CLI 사용

다음`put-bucket-lifecycle-configuration` AWS Command Line Interface(AWS CLI) 명령은 지정된 버킷에 수명 주기 구성을 추가합니다. 이 명령을 사용하려면 *`user input placeholders`*를 자체 정보로 대체합니다.

```
aws s3api put-bucket-lifecycle-configuration  \
        --bucket amzn-s3-demo-bucket  \
        --lifecycle-configuration filename-containing-lifecycle-configuration
```

다음 예제는 AWS CLI를 사용하여 미완료 멀티파트 업로드를 중단하는 수명 주기 규칙을 추가하는 방법을 보여줍니다. 여기에는 7일 이상 경과된 미완료 멀티파트 업로드를 중단하기 위한 JSON 수명 주기 구성 예제가 포함되어 있습니다.

이 예제에서 CLI 명령을 사용하려면 *user input placeholders*를 자체 정보로 대체합니다.

미완료 멀티파트 업로드를 중단하기 위한 수명 주기 규칙을 추가하려면

1. AWS CLI를 설정합니다. 자세한 내용은 Amazon S3 API 참조의 AWS CLI를 사용하여 Amazon S3에서 개발을 참조하세요.
2. 다음 수명 주기 구성 예제를 파일(예: *lifecycle.json*)에 저장합니다. 이 예제 구성에서는 빈 접두사가 지정되어 있으므로, 버킷에 있는 모든 객체에 적용됩니다. 구성을 객체의 하위 집합에만 제한적으로 적용하려면 접두사를 지정하면 됩니다.

```
{
    "Rules": [
        {
            "ID": "Test Rule",
            "Status": "Enabled",
            "Filter": {
                "Prefix": ""
            },
            "AbortIncompleteMultipartUpload": {
                "DaysAfterInitiation": 7
            }
        }
    ]
}
```

3. 다음 CLI 명령을 사용하여 버킷에서 이 수명 주기 구성을 설정합니다.

```
aws s3api put-bucket-lifecycle-configuration   \
--bucket amzn-s3-demo-bucket  \
--lifecycle-configuration file://lifecycle.json
```

4. 버킷에 수명 주기 구성이 설정되었는지 확인하려면 다음 `get-bucket-lifecycle` 명령을 사용하여 수명 주기 구성을 검색합니다.

```
aws s3api get-bucket-lifecycle  \
--bucket amzn-s3-demo-bucket
```

5. 수명 주기 구성을 삭제하려면 다음 `delete-bucket-lifecycle` 명령을 사용합니다.

```
aws s3api delete-bucket-lifecycle \
--bucket amzn-s3-demo-bucket
```

멀티파트 업로드를 사용한 객체 업로드

멀티파트 업로드를 사용하여 프로그래밍 방식으로 단일 객체를 Amazon S3에 업로드할 수 있습니다. 각 객체는 여러 파트의 집합으로 업로드됩니다. 각 부분은 객체 데이터의 연속적인 부분입니다. 이러한 객체 부분은 독립적으로 그리고 임의의 순서로 업로드할 수 있습니다. 부분의 전송이 실패할 경우 다른 부분에 영향을 주지 않고도 해당 부분을 재전송할 수 있습니다. 객체의 모든 부분이 업로드되면 Amazon S3가 이들 부분을 수집하여 객체를 생성합니다. 익명 사용자는 멀티파트 업로드를 시작할 수 없습니다.

추가 체크섬이 포함된 멀티파트 업로드를 사용하여 객체를 업로드하는 전체 절차는 튜토리얼: 멀티파트 업로드를 통한 객체 업로드 및 데이터 무결성 확인 섹션을 참조하세요.

다음 섹션에서는 AWS Command Line Interface 및 AWS SDK와 함께 멀티파트 업로드를 사용하는 방법을 보여줍니다.

S3 콘솔 사용

이미지, 백업, 데이터, 동영상 등 모든 유형의 파일을 S3 버킷에 업로드할 수 있습니다. Amazon S3 콘솔을 사용하여 업로드할 수 있는 파일의 최대 크기는 160GB입니다. 160GB가 넘는 파일을 업로드하려면 AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하십시오.

AWS Management Console을 통한 객체 업로드 지침은 객체 업로드 섹션을 참조하세요.

AWS CLI 사용

다음은 AWS CLI를 사용하여 멀티파트 업로드를 위한 Amazon S3 작업을 설명합니다.

- 멀티파트 업로드 시작
- 부분 업로드
- 부분 업로드(복사)
- 멀티파트 업로드 완료
- 멀티파트 업로드 중단
- 부분 목록 조회
- 멀티파트 업로드 목록 조회

REST API 사용

Amazon Simple Storage Service API 참조의 다음 섹션에서는 멀티파트 업로드를 위한 REST API에 대해 설명합니다.

- 멀티파트 업로드 시작
- 부분 업로드
- 멀티파트 업로드 완료
- 멀티파트 업로드 중지
- 부분 목록 조회
- 멀티파트 업로드 목록 조회

AWS SDK 사용(상위 수준 API)

일부 AWS SDK는 멀티파트 업로드를 완료하는 데 필요한 다양한 API 작업을 단일 작업으로 결합하여 멀티파트 업로드를 간소화하는 상위 수준 API를 제공합니다. 자세한 내용은 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 섹션을 참조하세요.

멀티파트 업로드를 일시 중지했다 다시 시작해야 하거나 업로드 중에 파트의 크기를 변경해야 하거나 혹은 데이터 크기를 미리 확인하지 않은 경우 하위 수준 API를 사용합니다. 멀티파트 업로드를 위한 하위 수준 API 메서드에서 제공하는 추가 기능에 대한 자세한 내용은 AWS SDK 사용(하위 수준 API) 섹션을 참조하세요.

.NET

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.

> **Note**
>
> 데이터 소스로 스트림을 사용하는 경우, `TransferUtility` 클래스는 동시 업로드를 수행하지 않습니다.

다음 C# 예제는 파일을 여러 파트로 나누어 Amazon S3 버킷에 업로드합니다. 다양한 `TransferUtility.Upload` 오버로드를 사용하여 파일을 업로드하는 방법을 보여줍니다. 업로드에 대해 연속적으로 수행되는 각 호출이 이전 업로드를 대체합니다. 코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

```
using Amazon;
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);
            }

        }
    }
```

```
}
```

JavaScript

Example

대용량 파일을 업로드합니다.

```
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

import {
  ProgressBar,
  logger,
} from "@aws-doc-sdk-examples/lib/utils/util-log.js";

const twentyFiveMB = 25 * 1024 * 1024;

export const createString = (size = twentyFiveMB) => {
  return "x".repeat(size);
};

/**
 * Create a 25MB file and upload it in parts to the specified
 * Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const str = createString();
  const buffer = Buffer.from(str, "utf8");
  const progressBar = new ProgressBar({
    description: `Uploading "${key}" to "${bucketName}"`,
    barLength: 30,
  });

  try {
    const upload = new Upload({
      client: new S3Client({}),
      params: {
        Bucket: bucketName,
        Key: key,
        Body: buffer,
      },
    });
```

```
    upload.on("httpUploadProgress", ({ loaded, total }) => {
      progressBar.update({ current: loaded, total });
    });

    await upload.done();
  } catch (caught) {
    if (caught instanceof Error && caught.name === "AbortError") {
      logger.error(`Multipart upload was aborted. ${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

Example

대용량 파일을 다운로드합니다.

```
import { fileURLToPath } from "node:url";
import { GetObjectCommand, NoSuchKey, S3Client } from "@aws-sdk/client-s3";
import { createWriteStream, rmSync } from "node:fs";

const s3Client = new S3Client({});
const oneMB = 1024 * 1024;

export const getObjectRange = ({ bucket, key, start, end }) => {
  const command = new GetObjectCommand({
    Bucket: bucket,
    Key: key,
    Range: `bytes=${start}-${end}`,
  });

  return s3Client.send(command);
};

/**
 * @param {string | undefined} contentRange
 */
export const getRangeAndLength = (contentRange) => {
  const [range, length] = contentRange.split("/");
  const [start, end] = range.split("-");
  return {
```

```
    start: Number.parseInt(start),
    end: Number.parseInt(end),
    length: Number.parseInt(length),
  };
};

export const isComplete = ({ end, length }) => end === length - 1;

const downloadInChunks = async ({ bucket, key }) => {
  const writeStream = createWriteStream(
    fileURLToPath(new URL(`./${key}`, import.meta.url)),
  ).on("error", (err) => console.error(err));

  let rangeAndLength = { start: -1, end: -1, length: -1 };

  while (!isComplete(rangeAndLength)) {
    const { end } = rangeAndLength;
    const nextRange = { start: end + 1, end: end + oneMB };

    const { ContentRange, Body } = await getObjectRange({
      bucket,
      key,
      ...nextRange,
    });
    console.log(`Downloaded bytes ${nextRange.start} to ${nextRange.end}`);

    writeStream.write(await Body.transformToByteArray());
    rangeAndLength = getRangeAndLength(ContentRange);
  }
};

/**
 * Download a large object from and Amazon S3 bucket.
 *
 * When downloading a large file, you might want to break it down into
 * smaller pieces. Amazon S3 accepts a Range header to specify the start
 * and end of the byte range to be downloaded.
 *
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  try {
    await downloadInChunks({
      bucket: bucketName,
```

```
      key: key,
    });
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(`Failed to download object. No such key "${key}".`);
      rmSync(key);
    }
  }
};
```

Go

멀티파트 업로드를 위한 Go 코드 예제에 대한 자세한 내용은 AWS SDK를 사용하여 Amazon S3에 대용량 파일 업로드 또는 다운로드를 참조하세요.

Example

업로드 관리자를 사용하여 데이터를 부분으로 나누고 동시에 업로드하여 큰 객체를 업로드합니다.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
```

```
 S3Client *s3.Client
}
```

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

 return err
}
```

Example

다운로드 관리자를 사용하여 데이터를 부분으로 나누고 동시에 다운로드하여 큰 객체를 다운로드 합니다.

```
// DownloadLargeObject uses a download manager to download an object from a bucket.
// The download manager gets the data in parts and writes them to a buffer until all
 of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
 string, objectKey string) ([]byte, error) {
 var partMiBs int64 = 10
 downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader) {
  d.PartSize = partMiBs * 1024 * 1024
 })
 buffer := manager.NewWriteAtBuffer([]byte{})
 _, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return buffer.Bytes(), err
}
```

PHP

이 주제에서는 멀티파트 파일 업로드를 위해 AWS SDK for PHP의 상위 수준 `Aws\S3\Model\MultipartUpload\UploadBuilder` 클래스를 사용하는 방법을 설명합니다. AWS SDK for Ruby API에 대한 자세한 내용은 AWS SDK for Ruby – 버전 2를 참조하세요.

다음은 Amazon S3 버킷에 파일을 업로드하는 PHP 예제입니다. 이 예제는 `MultipartUploader` 객체에 대한 파라미터를 설정하는 방법을 보여 줍니다.

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

use Aws\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;
}
```

Python

다음 예제에서는 상위 수준 멀티파트 업로드 Python API(`TransferManager` 클래스)를 사용하여 객체를 업로드합니다.

```
import sys
import threading

import boto3
from boto3.s3.transfer import TransferConfig


MB = 1024 * 1024
s3 = boto3.resource("s3")


class TransferCallback:
    """
```

```
    Handle callbacks from the transfer manager.

    The transfer manager periodically calls the __call__ method throughout
    the upload and download process so that it can take action, such as
    displaying progress to the user and collecting data about the transfer.
    """

    def __init__(self, target_size):
        self._target_size = target_size
        self._total_transferred = 0
        self._lock = threading.Lock()
        self.thread_info = {}

    def __call__(self, bytes_transferred):
        """
        The callback method that is called by the transfer manager.

        Display progress during file transfer and collect per-thread transfer
        data. This method can be called by multiple threads, so shared instance
        data is protected by a thread lock.
        """
        thread = threading.current_thread()
        with self._lock:
            self._total_transferred += bytes_transferred
            if thread.ident not in self.thread_info.keys():
                self.thread_info[thread.ident] = bytes_transferred
            else:
                self.thread_info[thread.ident] += bytes_transferred

            target = self._target_size * MB
            sys.stdout.write(
                f"\r{self._total_transferred} of {target} transferred "
                f"({(self._total_transferred / target) * 100:.2f}%)."
            )
            sys.stdout.flush()


def upload_with_default_configuration(
    local_file_path, bucket_name, object_key, file_size_mb
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, using the default
    configuration.
    """
```

```
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_chunksize_and_meta(
    local_file_path, bucket_name, object_key, file_size_mb, metadata=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart chunk size and adding metadata to the Amazon S3 object.

    The multipart chunk size controls the size of the chunks of data that are
    sent in the request. A smaller chunk size typically results in the transfer
    manager using more threads for the upload.

    The metadata is a set of key-value pairs that are stored with the object
    in Amazon S3.
    """
    transfer_callback = TransferCallback(file_size_mb)

    config = TransferConfig(multipart_chunksize=1 * MB)
    extra_args = {"Metadata": metadata} if metadata else None
    s3.Bucket(bucket_name).upload_file(
        local_file_path,
        object_key,
        Config=config,
        ExtraArgs=extra_args,
        Callback=transfer_callback,
    )
    return transfer_callback.thread_info


def upload_with_high_threshold(local_file_path, bucket_name, object_key,
 file_size_mb):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard upload instead of
    a multipart upload.
```

```
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_sse(
    local_file_path, bucket_name, object_key, file_size_mb, sse_key=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, adding server-side
    encryption with customer-provided encryption keys to the object.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)
    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey": sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, ExtraArgs=extra_args,
 Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_default_configuration(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using the
    default configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

```
def download_with_single_thread(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using a
    single thread.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(use_threads=False)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_high_threshold(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard download instead
    of a multipart download.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_sse(
    bucket_name, object_key, download_file_path, file_size_mb, sse_key
):
    """
    Download a file from an Amazon S3 bucket to a local folder, adding a
    customer-provided encryption key to the request.

    When this kind of encryption is specified, Amazon S3 encrypts the object
```

```
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)

    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey": sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, ExtraArgs=extra_args, Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

AWS SDK 사용(하위 수준 API)

AWS SDK는 멀티파트 업로드를 위해 Amazon S3 REST API와 매우 유사한 하위 수준 API를 표시합니다(Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 참조). 멀티파트 업로드를 일시 중지한 후 다시 시작해야 하거나 업로드 중에 부분 크기를 변경해야 하거나 업로드 데이터 크기를 미리 확인하지 않은 경우 하위 수준 API를 사용합니다. 이러한 요구 사항이 없는 경우에는 상위 수준 API를 사용합니다(AWS SDK 사용(상위 수준 API) 참조).

Java

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 AWS SDK for Java 개발자 안내서의 시작하기 섹션을 참조하세요.

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

.NET

다음은 하위 수준 SDK for .NET 멀티파트 업로드 API를 사용하여 S3 버킷에 파일을 업로드하는 방법을 보여주는 C# 예제입니다. Amazon S3 멀티파트 업로드에 대한 자세한 내용은 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 단원을 참조하십시오.

> **Note**
>
> SDK for .NET 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. 코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

```
using Amazon;
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);
        }
    }
}
```

PHP

이 주제는 AWS SDK for PHP 버전 3에서 하위 수준 `uploadPart` 메서드를 사용하여 파일을 여러 부분으로 업로드하는 방법을 보여 줍니다. AWS SDK for Ruby API에 대한 자세한 내용은 AWS SDK for Ruby – 버전 2를 참조하세요.

다음은 하위 수준 PHP API 멀티파트 업로드를 사용하여 Amazon S3 버킷에 파일을 업로드하는 PHP 예제입니다.

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

use Aws\S3\Exception\S3Exception;
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',
    '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;
```

AWS SDK for Ruby 사용

AWS SDK for Ruby 버전 3은 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 원본으로 업로드합니다.
- complete_multipart_upload – 이전에 업로드된 파트를 어셈블하여 멀티파트 업로드를 완료합니다.
- abort_multipart_upload – 멀티파트 업로드를 중단합니다.

상위 수준 .NET TransferUtility 클래스를 사용하여 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 S3는 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 보여 줍니다. 업로드에 대해 연속적으로 수행되는 각 호출이 이전 업로드를 대체합니다. 코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

```
using Amazon;
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);
            }
        }
    }
}
```

멀티파트 업로드 나열

AWS CLI, REST API 또는 AWS SDK를 사용하여 Amazon S3에서 진행 중인 멀티파트 업로드 목록을 검색할 수 있습니다. 멀티파트 업로드를 사용하여 프로그래밍 방식으로 단일 객체를 Amazon S3에 업로드할 수 있습니다. 멀티파트 업로드는 한 번에 객체 데이터를 일부분씩 옮겨 객체를 Amazon S3로 이동합니다. 멀티파트 업로드에 대한 일반적인 내용을 자세히 알아보려면 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 섹션을 참조하세요.

추가 체크섬이 포함된 멀티파트 업로드를 사용하여 객체를 업로드하는 전체 절차는 튜토리얼: 멀티파트 업로드를 통한 객체 업로드 및 데이터 무결성 확인 섹션을 참조하세요.

다음 섹션에서는 AWS Command Line Interface, Amazon S3 REST API, AWS SDK를 사용하여 진행 중인 멀티파트 업로드를 나열하는 방법을 보여줍니다.

AWS CLI를 사용한 멀티파트 업로드 나열

AWS Command Line Interface의 다음 섹션에서는 멀티파트 업로드를 나열하는 작업을 설명합니다.

- list-parts - 특정 멀티파트 업로드의 업로드된 부분을 나열합니다.
- list-multipart-uploads - 진행 중인 멀티파트 업로드를 나열합니다.

REST API를 사용한 멀티파트 업로드 나열

Amazon Simple Storage Service API 참조의 다음 섹션에서는 멀티파트 업로드 나열을 위한 REST API에 대해 설명합니다.

- ListParts - 특정 멀티파트 업로드의 업로드된 부분을 나열합니다.
- ListMultipartUploads - 진행 중인 멀티파트 업로드를 나열합니다.

AWS SDK(하위 수준 API)를 사용한 멀티파트 업로드 나열

Java

다음 작업은 버킷에서 진행 중인 모든 멀티파트 업로드를 나열하기 위해 하위 수준 Java 클래스를 사용하는 방법을 보여줍니다.

하위 수준 API 멀티파트 업로드 목록 조회 프로세스

1	`ListMultipartUploadsRequest` 클래스의 인스턴스를 만들고 버킷 이름을 제공합니다.
2	`AmazonS3Client.listMultipartUploads` 메서드를 실행합니다. 이 메서드는 진행 중인 멀티파트 업로드에 대한 정보를 제공하는 `MultipartUploadListing` 클래스의 인스턴스를 반환합니다.

다음은 위에서 설명한 작업을 실행하는 Java 코드 예제입니다.

Example

```
ListMultipartUploadsRequest allMultpartUploadsRequest =
     new ListMultipartUploadsRequest(existingBucketName);
MultipartUploadListing multipartUploadListing =
     s3Client.listMultipartUploads(allMultpartUploadsRequest);
```

.NET

특정 버킷에 대해 진행 중인 멀티파트 업로드를 모두 나열하려면 SDK for .NET 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 중지되지 않은 멀티파트 업로드를 말합니다. Amazon S3 멀티파트 업로드에 대한 자세한 내용은 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 단원을 참조하십시오.

다음 C# 예제는 SDK for .NET를 사용하여 버킷에 대해 진행 중인 모든 멀티파트 업로드를 나열하는 방법을 보여줍니다. 코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

```
ListMultipartUploadsRequest request = new ListMultipartUploadsRequest
{
  BucketName = bucketName // Bucket receiving the uploads.
};

ListMultipartUploadsResponse response = await
 AmazonS3Client.ListMultipartUploadsAsync(request);
```

PHP

이 주제에서는 AWS SDK for PHP 버전 3의 하위 수준 API 클래스를 사용하여 버킷에서 진행 중인 멀티파트 업로드를 모두 나열하는 방법을 보여줍니다. AWS SDK for Ruby API에 대한 자세한 내용은 AWS SDK for Ruby – 버전 2를 참조하세요.

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());
```

AWS SDK를 사용하여 멀티파트 업로드 추적

수신 대기 인터페이스를 사용하여 Amazon S3에 대한 객체 업로드 진행 상황을 추적할 수 있습니다. 고급 멀티파트 업로드 API는 `ProgressListener`라는 수신 대기 인터페이스를 제공합니다. 진행률 이벤트는 주기적으로 발생하며 바이트가 전송되었음을 리스너에게 통보합니다. 멀티파트 업로드에 대

한 일반적인 내용을 자세히 알아보려면 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 섹션을 참조하세요.

추가 체크섬이 포함된 멀티파트 업로드를 사용하여 객체를 업로드하는 전체 절차는 튜토리얼: 멀티파트 업로드를 통한 객체 업로드 및 데이터 무결성 확인 섹션을 참조하세요.

다음 섹션에서는 AWS SDK와 함께 멀티파트 업로드를 추적하는 방법을 보여줍니다.

Java

Example

```
TransferManager tm = new TransferManager(new ProfileCredentialsProvider());

PutObjectRequest request = new PutObjectRequest(
    existingBucketName, keyName, new File(filePath));

// Subscribe to the event and provide event handler.
request.setProgressListener(new ProgressListener() {
   public void progressChanged(ProgressEvent event) {
    System.out.println("Transferred bytes: " +
      event.getBytesTransfered());
            }
});
```

Example

다음 Java 코드는 파일을 업로드하고 ProgressListener을(를) 사용하여 업로드 진행률을 추적합니다. 실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 AWS SDK for Java 개발자 안내서에서 시작하기 섹션을 참조하세요.

```
import java.io.File;

import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.event.ProgressEvent;
import com.amazonaws.event.ProgressListener;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.Upload;

public class TrackMPUProgressUsingHighLevelAPI {
```

```
    public static void main(String[] args) throws Exception {
        String existingBucketName = "*** Provide bucket name ***";
        String keyName            = "*** Provide object key ***";
        String filePath           = "*** file to upload ***";

        TransferManager tm = new TransferManager(new ProfileCredentialsProvider());


        // For more advanced uploads, you can create a request object
        // and supply additional request parameters (ex: progress listeners,
        // canned ACLs, etc.)
        PutObjectRequest request = new PutObjectRequest(
          existingBucketName, keyName, new File(filePath));

        // You can ask the upload for its progress, or you can
        // add a ProgressListener to your request to receive notifications
        // when bytes are transferred.
        request.setGeneralProgressListener(new ProgressListener() {
   @Override
   public void progressChanged(ProgressEvent progressEvent) {
    System.out.println("Transferred bytes: " +
      progressEvent.getBytesTransferred());
   }
  });

        // TransferManager processes all transfers asynchronously,
        // so this call will return immediately.
        Upload upload = tm.upload(request);

        try {
         // You can block and wait for the upload to finish
         upload.waitForCompletion();
        } catch (AmazonClientException amazonClientException) {
         System.out.println("Unable to upload file, upload aborted.");
         amazonClientException.printStackTrace();
        }
    }
}
```

.NET

다음 C# 예제는 TransferUtility 클래스를 사용하여 S3 버킷에 파일을 업로드하고 업로드 진행률을 추적합니다. 코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

```
using Amazon;
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);
        }
    }
}
```

멀티파트 업로드 중단

멀티파트 업로드를 개시한 후에 부분 업로드를 시작합니다. Amazon S3는 부분들을 저장하며, 모든 부분을 업로드하고 멀티파트 업로드 완료 요청을 보낸 후에만 객체를 생성합니다. 완전한 멀티파트 업로드 요청을 수신하면 그 즉시 Amazon S3는 부분들을 결합해 객체를 생성합니다. 완전한 멀티파트 업로드 요청을 성공적으로 전송하지 못하면 S3는 부분들을 결합하지 않고 어떤 객체도 생성하지 않습니다. 부분들을 업로드한 후 멀티파트 업로드를 완료하지 않으려면 멀티파트 업로드를 중단해야 합니다.

업로드된 부분에 연결된 모든 스토리지에 대해 요금이 청구됩니다. 항상 멀티파트 업로드를 완료하거나 멀티파트 업로드를 중지하여 업로드된 부분을 제거하는 것이 좋습니다. 요금에 대한 자세한 내용은 멀티파트 업로드 및 요금 단원을 참조하세요.

버킷 수명 주기 구성을 사용하여 불완전한 멀티파트 업로드를 중지할 수도 있습니다. 자세한 내용은 불완전한 멀티파트 업로드를 삭제하도록 버킷 수명 주기 구성 설정 섹션을 참조하세요.

다음 섹션에서는 AWS Command Line Interface, REST API 또는 AWS SDK를 사용하여 Amazon S3에서 진행 중인 멀티파트 업로드를 중지하는 방법을 보여줍니다.

AWS CLI 사용

AWS CLI를 사용하여 멀티파트 업로드를 중지하는 방법에 대한 자세한 내용은 AWS CLI 명령 참조의 abort-multipart-upload를 참조하십시오.

REST API 사용

REST API를 사용하여 멀티파트 업로드를 중지하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 AbortMultipartUpload를 참조하십시오.

AWS SDK 사용(상위 수준 API)

Java

`TransferManager` 클래스는 진행 중인 멀티파트 업로드를 중지하는 `abortMultipartUploads` 메서드를 제공합니다. 사용자가 업로드를 시작한 후 완료 또는 중지할 때까지 업로드가 진행 중인 것으로 간주됩니다. 사용자가 `Date` 값을 제공하면 API가 해당 버킷에서 지정된 `Date` 이전에 시작하여 여전히 진행 중인 멀티파트 업로드를 모두 중지합니다.

다음 작업은 상위 수준 Java 클래스를 사용하여 파일을 중지하는 방법을 보여줍니다.

상위 수준 API 멀티파트 업로드 중지 프로세스

1	`TransferManager` 클래스의 인스턴스를 만듭니다.
2	버킷 이름과 `TransferManager.abortMultipartUploads` 값을 전달하여 `Date` 메소드를 실행합니다.

다음 Java 코드는 1주일 이상 전에 특정 버킷에서 시작되어 진행 중인 멀티파트 업로드를 모두 중지합니다. 실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 AWS SDK for Java 개발자 안내서에서 시작하기 섹션을 참조하세요.

```
import java.util.Date;

import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.transfer.TransferManager;

public class AbortMPUUsingHighLevelAPI {

    public static void main(String[] args) throws Exception {
        String existingBucketName = "*** Provide existing bucket name ***";

        TransferManager tm = new TransferManager(new ProfileCredentialsProvider());


        int sevenDays = 1000 * 60 * 60 * 24 * 7;
  Date oneWeekAgo = new Date(System.currentTimeMillis() - sevenDays);

        try {
         tm.abortMultipartUploads(existingBucketName, oneWeekAgo);
        } catch (AmazonClientException amazonClientException) {
         System.out.println("Unable to upload file, upload was aborted.");
         amazonClientException.printStackTrace();
        }
    }
}
```

> **Note**
>
> 특정 멀티파트 업로드를 중지할 수도 있습니다. 자세한 내용은 AWS SDK 사용(하위 수준 API) 섹션을 참조하세요.

.NET

다음 C# 예제는 1주일 이상 전에 특정 버킷에서 시작되어 진행 중인 멀티파트 업로드를 모두 중지합니다. 코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

```
using Amazon;
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 중지할 수도 있습니다. 자세한 내용은 AWS SDK 사용(하위 수준 API) 섹션을 참조하세요.

AWS SDK 사용(하위 수준 API)

`AmazonS3.abortMultipartUpload` 메서드를 호출하여 진행 중인 멀티파트 업로드를 중단할 수 있습니다. 이 메서드는 Amazon S3에 업로드된 모든 부분을 삭제하므로 가용 리소스가 늘어나게 됩니다. 업로드 ID, 버킷 이름 및 키 이름을 제공해야 합니다. 다음은 진행 중인 멀티파트 업로드를 중지하는 방법을 보여주는 Java 코드 예제입니다.

멀티파트 업로드를 중지하려면 업로드에 사용된 업로드 ID, 버킷 및 키 이름을 제공해야 합니다. 멀티파트 업로드를 중지하면 해당 업로드 ID를 사용해 추가 파트를 업로드할 수 없습니다. Amazon S3 멀티파트 업로드에 대한 자세한 내용은 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 단원을 참조하십시오.

Java

다음 Java 코드 예제에서는 진행 중인 멀티파트 업로드를 중지합니다.

Example

```
InitiateMultipartUploadRequest initRequest =
    new InitiateMultipartUploadRequest(existingBucketName, keyName);
InitiateMultipartUploadResult initResponse =
               s3Client.initiateMultipartUpload(initRequest);

AmazonS3 s3Client = new AmazonS3Client(new ProfileCredentialsProvider());
s3Client.abortMultipartUpload(new AbortMultipartUploadRequest(
            existingBucketName, keyName, initResponse.getUploadId()));
```

> **Note**
>
> 특정 멀티파트 업로드 대신, 특정 시간 이전에 시작되어 아직 진행 중인 모든 멀티파트 업로드를 중지할 수 있습니다. 이 정리 작업은 시작되었지만 완료되거나 중지되지 않은 이전 멀티파트 업로드를 중지하는 데 유용합니다. 자세한 내용은 AWS SDK 사용(상위 수준 API) 섹션을 참조하세요.

.NET

다음 C# 예제는 멀티파트 업로드를 중지하는 방법을 보여줍니다. 다음 코드가 포함된 전체 C# 예제는 AWS SDK 사용(하위 수준 API) 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 AWS SDK 사용(상위 수준 API) 섹션을 참조하세요.

PHP

이 예제에서는 진행 중인 멀티파트 업로드를 중지하기 위해 AWS SDK for PHP 버전 3의 클래스를 사용하는 방법을 보여줍니다. AWS SDK for Ruby API에 대한 자세한 내용은 AWS SDK for Ruby – 버전 2를 참조하세요. 예를 들어 `abortMultipartUpload()` 메서드입니다.

AWS SDK for Ruby API에 대한 자세한 내용은 AWS SDK for Ruby – 버전 2를 참조하세요.

```
 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,
]);
```

멀티파트 업로드를 사용한 객체 복사

멀티파트 업로드를 통해 객체를 여러 부분의 집합으로 복사할 수 있습니다. 이 섹션의 예제는 멀티파트 업로드 API를 사용하여 5GB보다 큰 객체를 복사하는 방법을 보여 줍니다. 멀티파트 업로드에 대한 내용은 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 섹션을 참조하세요.

멀티파트 업로드 API를 사용하지 않고 단일 작업으로는 5GB보다 작은 객체를 복사할 수 있습니다. AWS Management Console, AWS CLI, REST API 또는 AWS SDK를 사용하여 5GB보다 작은 객체를 복사할 수 있습니다. 자세한 내용은 객체 복사, 이동, 이름 변경 섹션을 참조하세요.

추가 체크섬이 포함된 멀티파트 업로드를 사용하여 객체를 업로드하는 전체 절차는 튜토리얼: 멀티파트 업로드를 통한 객체 업로드 및 데이터 무결성 확인 섹션을 참조하세요.

다음 섹션에서는 REST API 또는 AWS SDK를 사용하여 멀티파트 업로드로 객체를 복사하는 방법을 보여줍니다.

REST API 사용

Amazon Simple Storage Service API 참조의 다음 섹션에서는 멀티파트 업로드를 위한 REST API에 대해 설명합니다. 기존 객체를 복사하려면, 파트 업로드(복사) API를 사용하고 요청에 `x-amz-copy-source` 요청 헤더를 추가하여 소스 객체를 지정합니다.

- 멀티파트 업로드 시작
- 부분 업로드
- 부분 업로드(복사)
- 멀티파트 업로드 완료
- 멀티파트 업로드 중단
- 부분 목록 조회
- 멀티파트 업로드 목록 조회

이러한 API를 사용하여 REST 요청을 만들거나, 제공된 SDK 중 하나를 사용할 수 있습니다. AWS CLI에서 멀티파트 업로드를 사용하는 방법에 대한 자세한 내용은 AWS CLI 사용 단원을 참조하십시오. SDK에 대한 자세한 내용은 멀티파트 업로드에 대한 AWS SDK 지원을 참조하십시오.

AWS SDK 사용

하위 수준 API를 사용하여 객체를 복사하려면 다음을 수행합니다.

- `AmazonS3Client.initiateMultipartUpload()` 메서드를 호출하여 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 번호화 `ETag` 값이 포함되어 있습니다. 멀티파트 업로드를 완료하려면 이러한 정보가 필요합니다.
- `AmazonS3Client.completeMultipartUpload()` 메서드를 호출하여 복사 작업을 완료합니다.

Java

Example

다음 예제에서는 Amazon S3 하위 수준 Java API를 사용하여 멀티파트 복사를 수행하는 방법을 보여줍니다. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 AWS SDK for Java 개발자 안내서의 시작하기 섹션을 참조하세요.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.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;
    }
}
```

.NET

다음 C# 예제는 SDK for .NET을 사용하여 5GB보다 큰 Amazon S3 객체를 하나의 소스 위치에서 다른 위치(예: 한 버킷에서 다른 버킷으로)로 복사합니다. 5GB보다 작은 객체를 복사하려면 AWS SDK 사용에 설명된 단일 작업 복사 프로시저를 사용하십시오. Amazon S3 멀티파트 업로드에 대한 자세한 내용은 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 단원을 참조하십시오.

이 예제에서는 SDK for .NET 멀티파트 업로드 API를 사용하여 S3 버킷 간에 5GB보다 큰 Amazon S3 객체를 복사하는 방법을 보여 줍니다.

```
using Amazon;
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);
            }
        }
    }
}
```

튜토리얼: 멀티파트 업로드를 통한 객체 업로드 및 데이터 무결성 확인

멀티파트 업로드를 사용하면 단일 객체를 여러 부분의 집합으로 업로드할 수 있습니다. 각 부분은 객체 데이터의 연속적인 부분입니다. 이러한 객체 부분은 독립적으로 그리고 임의의 순서로 업로드할 수 있습니다. 부분의 전송이 실패할 경우 다른 부분에 영향을 주지 않고도 해당 부분을 재전송할 수 있습니다. 객체의 모든 부분이 업로드되면 Amazon S3가 이들 부분을 수집하여 객체를 생성합니다. 일반적으로 객체 크기가 100MB에 근접할 경우, 단일 작업에서 객체를 업로드하는 대신 멀티파트 업로드 사용을 고려해 봐야 합니다. 멀티파트 업로드에 대한 자세한 내용은 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 섹션을 참조하십시오. 멀티파트 업로드와 관련된 제한은 Amazon S3 멀티파트 업로드 제한 섹션을 참조하세요.

체크섬을 사용하여 자산을 복사할 때 자산이 변경되지 않았는지 확인할 수 있습니다. 체크섬을 수행하는 것은 알고리즘을 사용하여 파일의 모든 바이트를 순차적으로 반복하는 작업으로 구성됩니다. Amazon S3는 데이터 무결성을 검사하기 위한 여러 체크섬 옵션을 제공합니다. 내구성 모범 사례로 이러한 무결성 검사를 수행하고 모든 바이트가 변경 없이 전송되는지 확인하는 것이 좋습니다. Amazon S3는 SHA-1, SHA-256, CRC32, CRC32C 등의 알고리즘도 지원합니다. Amazon S3는 이러한 알고리즘을 하나 이상 사용하여 추가 체크섬 값을 계산하고 객체 메타데이터의 일부로 저장합니다. 체크섬에 대한 자세한 내용은 Amazon S3에서 객체 무결성 확인 섹션을 참조하십시오.

목표

이 튜토리얼에서는 AWS 명령줄 인터페이스(AWS CLI)를 통해 멀티파트 업로드 및 추가 SHA-256 체크섬을 사용하여 Amazon S3에 객체를 업로드하는 방법을 알아봅니다. 또한 업로드한 객체의 MD5 해시 및 SHA-256 체크섬을 계산하여 객체의 데이터 무결성을 검사하는 방법도 배우게 됩니다.

주제

- 사전 조건
- 1단계: 대용량 파일 생성
- 2단계: 파일을 여러 파일로 분할
- 3단계: 추가 체크섬을 사용하여 멀티파트 업로드 생성
- 4단계: 멀티파트 업로드의 부분 업로드
- 5단계: 멀티파트 업로드의 모든 부분 나열
- 6단계: 멀티파트 업로드 완료
- 7단계: 객체가 버킷에 업로드되었는지 확인
- 8단계: MD5 체크섬으로 객체 무결성 확인
- 9단계: 추가 체크섬으로 객체 무결성 확인

- 10단계: 리소스 정리

사전 조건

- 이 튜토리얼을 시작하기 전에 업로드할 수 있는 Amazon S3 버킷에 대한 액세스 권한이 있는지 확인합니다. 자세한 내용은 범용 버킷 생성 섹션을 참조하세요.
- AWS CLI를 설치하고 구성해야 합니다. AWS CLI를 설치하지 않은 경우 AWS Command Line Interface 사용 설명서에서 최신 버전의 AWS CLI 설치 또는 업데이트를 참조하세요.
- 또는 AWS CloudShell을 사용하여 콘솔에서 AWS CLI 명령을 실행할 수 있습니다. AWS CloudShell은 브라우저 기반의 사전 인증된 쉘로, AWS Management Console에서 직접 시작할 수 있습니다. 자세한 내용은 AWS CloudShell 사용 설명서에서 CloudShell이란 무엇인가요? 및 AWS CloudShell 시작하기를 참조하세요.

1단계: 대용량 파일 생성

업로드할 파일이 이미 있으면 이 튜토리얼에 해당 파일을 사용할 수 있습니다. 그렇지 않은 경우 다음 단계를 사용하여 15MB 파일을 생성합니다. 멀티파트 업로드와 관련된 제한은 Amazon S3 멀티파트 업로드 제한 섹션을 참조하세요.

대용량 파일을 생성하려면

사용 중인 운영 체제에 따라 다음 명령 중 하나로 파일을 생성합니다.

Linux 또는 macOS

15MB 파일을 생성하려면 로컬 터미널을 열고 다음 명령을 실행합니다.

```
dd if=/dev/urandom of=census-data.bin bs=1M count=15
```

이 명령은 크기가 15MB인 무작위 바이트로 채워진 `census-data.bin` 파일을 생성합니다.

Windows

15MB 파일을 생성하려면 로컬 터미널을 열고 다음 명령을 실행합니다.

```
fsutil file createnew census-data.bin 15728640
```

이 명령은 크기가 15MB((15728640바이트)인 임의 데이터로 이루어진 `census-data.bin` 파일을 생성합니다.

2단계: 파일을 여러 파일로 분할

멀티파트 업로드를 수행하려면 대용량 파일을 더 작은 부분으로 분할해야 합니다. 그런 다음 멀티파트 업로드 프로세스를 사용하여 작은 부분을 업로드할 수 있습니다. 이 단계에서는 1단계에서 생성한 대용량 파일을 작은 부분으로 분할하는 방법을 보여줍니다. 다음 예는 `census-data.bin`이라는 15MB 파일을 사용합니다.

대용량 파일을 여러 부분으로 분할하려면

Linux 또는 macOS

대용량 파일을 5MB 부분으로 나누려면 `split` 명령을 사용합니다. 터미널을 열고 다음 명령을 실행합니다.

```
split -b 5M -d census-data.bin census-part
```

이 명령은 `census-data.bin`을 `census-part**`라는 5MB 부분으로 나눕니다. 여기서 `**`는 `00`으로 시작하는 숫자 접미사입니다.

Windows

대용량 파일을 분할하려면 PowerShell을 사용합니다. PowerShell을 열고 다음 스크립트를 실행합니다.

```
$inputFile = "census-data.bin"
$outputFilePrefix = "census-part"
$chunkSize = 5MB

$fs = [System.IO.File]::OpenRead($inputFile)
$buffer = New-Object byte[] $chunkSize
$fileNumber = 0

while ($fs.Position -lt $fs.Length) {
$bytesRead = $fs.Read($buffer, 0, $chunkSize)
$outputFile = "{0}{1:D2}" -f $outputFilePrefix, $fileNumber
$fileStream = [System.IO.File]::Create($outputFile)
$fileStream.Write($buffer, 0, $bytesRead)
$fileStream.Close()
$fileNumber++
}
```

```
$fs.Close()
```

이 PowerShell 스크립트는 5MB 청크로 대용량 파일을 읽고 각 청크를 숫자 접미사가 있는 새 파일에 씁니다.

적절한 명령을 실행한 후에는 명령을 실행한 디렉터리에 해당 부분이 표시됩니다. 각 부분에는 부분 번호에 해당하는 접미사가 붙습니다. 예를 들면 다음과 같습니다.

```
census-part00 census-part01 census-part02
```

3단계: 추가 체크섬을 사용하여 멀티파트 업로드 생성

멀티파트 업로드 프로세스를 시작하려면 멀티파트 업로드 요청을 생성해야 합니다. 이 단계에는 멀티파트 업로드를 시작하고 데이터 무결성을 위한 추가 체크섬을 지정하는 작업이 포함됩니다. 다음 예에서는 SHA-256 체크섬을 사용합니다. 업로드할 객체에 메타데이터를 제공하려는 경우 멀티파트 업로드 시작 요청에서 메타데이터를 제공해야 합니다.

> **Note**
>
> 이 단계와 후속 단계에서 이 튜토리얼은 SHA-256 추가 알고리즘을 사용합니다. 필요에 따라 이러한 단계에 CRC32, CRC32C 또는 SHA-1을 비롯한 다른 추가 체크섬을 사용할 수도 있습니다. 다른 알고리즘을 사용하는 경우 튜토리얼 단계 전체에서 사용해야 합니다.

멀티파트 업로드를 시작하려면

터미널에서 다음 `create-multipart-upload` 명령을 사용하여 버킷의 멀티파트 업로드를 시작합니다. *amzn-s3-demo-bucket1*을 실제 버킷 이름으로 바꿉니다. 또한 `census_data_file`을 선택한 파일 이름으로 바꿉니다. 업로드가 완료되면 이 파일 이름이 객체 키가 됩니다.

```
aws s3api create-multipart-upload --bucket amzn-s3-demo-bucket1 --key
 'census_data_file' --checksum-algorithm sha256
```

요청이 성공하면 다음과 같은 JSON 결과가 표시됩니다.

```
{
    "ServerSideEncryption": "AES256",
    "ChecksumAlgorithm": "SHA256",
```

```
    "Bucket": "amzn-s3-demo-bucket1",
    "Key": "census_data_file",
    "UploadId":
 "cNV6KCSNANFZapz1LUGPC5XwUVi1n6yUoIeSP138sNOKPeMhpKQRrbT9k0ePmgoOTCj9K83T4e2Gb5hQvNoNpCKqyb8m3
}
```

> **Note**
>
> 멀티파트 업로드 시작 요청을 전송하면 Amazon S3는 멀티파트 업로드에 대한 고유 식별자인 업로드 ID와 함께 응답을 반환합니다. 부분 업로드, 부분 목록 확인, 업로드 완료 또는 업로드 중단 요청 시 항상 이 업로드 ID를 포함해야 합니다. 이후 단계에서 `UploadId`, `Key`, `Bucket` 값을 사용해야 하므로 이러한 값을 저장해 둡니다.
> 추가 체크섬과 함께 멀티파트 업로드를 사용하는 경우 부분 번호는 연속된 번호여야 합니다. 연속되지 않은 부분 번호를 사용하는 경우 `complete-multipart-upload` 요청에서 `HTTP 500 Internal Server Error`가 반환될 수 있습니다.

4단계: 멀티파트 업로드의 부분 업로드

이 단계에서는 S3 버킷에 멀티파트 업로드의 부분들을 업로드합니다. `upload-part` 명령을 사용하여 각 부분을 개별적으로 업로드합니다. 이 프로세스를 수행하려면 업로드 ID, 부분 번호, 각 부분에 업로드할 파일을 지정해야 합니다.

부분을 업로드하려면

1. 부분을 업로드할 때 업로드 ID와 함께 `--part-number` 인수를 사용하여 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 연속 시퀀스로 선택해야 합니다(예: 1, 2 또는 3일 수 있음). 이전에 업로드한 부분과 동일한 부분 번호로 새 부분을 업로드할 경우 이전에 업로드한 부분을 덮어쓰게 됩니다.
2. `upload-part` 명령을 사용하여 멀티파트 업로드의 각 부분을 업로드합니다. `--upload-id`는 3단계에서 `create-multipart-upload` 명령으로 생성된 출력과 동일합니다. 데이터의 첫 부분을 업로드하려면 다음 명령을 사용합니다.

```
aws s3api upload-part --bucket amzn-s3-demo-bucket1 --key
 'census_data_file' --part-number 1 --body census-part00 --upload-id
 "cNV6KCSNANFZapz1LUGPC5XwUVi1n6yUoIeSP138sNOKPeMhpKQRrbT9k0ePmgoOTCj9K83T4e2Gb5hQvNoNpCKqy
 --checksum-algorithm SHA256
```

각 upload-part 명령을 완료하면 다음 예와 같은 출력이 표시됩니다.

```
{
    "ServerSideEncryption": "AES256",
    "ETag": "\"e611693805e812ef37f96c9937605e69\"",
    "ChecksumSHA256": "QLl8R4i4+SaJlrl8ZIcutc5TbZtwt2NwB8lTXkd3GH0="
}
```

3. 후속 부분의 경우 부분 번호를 적절히 증가시킵니다.

```
aws s3api upload-part --bucket amzn-s3-demo-bucket1 --key 'census_data_file' --
part-number <part-number> --body <file-path> --upload-id "<your-upload-id>" --
checksum-algorithm SHA256
```

예를 들어, 다음 명령을 사용하여 두 번째 부분을 업로드합니다.

```
aws s3api upload-part --bucket amzn-s3-demo-bucket1 --key
 'census_data_file' --part-number 2 --body census-part01 --upload-id
 "cNV6KCSNANFZapz1LUGPC5XwUVi1n6yUoIeSP138sNOKPeMhpKQRrbT9k0ePmgoOTCj9K83T4e2Gb5hQvNoNpCKqy
 --checksum-algorithm SHA256
```

Amazon S3는 업로드된 각 부분에 대한 엔터티 태그(ETag)와 추가 체크섬을 응답의 헤더로 반환합니다.

4. 객체의 모든 부분을 업로드할 때까지 upload-part 명령을 계속 사용합니다.

5단계: 멀티파트 업로드의 모든 부분 나열

멀티파트 업로드를 완료하려면 해당 멀티파트 업로드를 위해 업로드된 모든 부분의 목록이 필요합니다. list-parts 명령의 출력에는 버킷 이름, 키, 업로드 ID, 파트 번호, ETag, 추가 체크섬 등과 같은 정보가 제공됩니다. 멀티파트 업로드 프로세스를 완료할 때 다음 단계에서 사용할 수 있도록 이 출력을 파일에 저장하는 것이 좋습니다. 다음 방법을 사용하여 parts.json이라는 JSON 출력 파일을 생성할 수 있습니다.

모든 부분이 나열된 파일을 생성하려면

1. 업로드된 모든 부분의 세부 정보가 포함된 JSON 파일을 생성하려면 다음 list-parts 명령을 사용합니다. ***amzn-s3-demo-bucket1***을 실제 버킷 이름 및 **<your-upload-id>**(3단계에서 받은

업로드 ID)로 바꿉니다 `list-parts` 명령에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 list-parts를 참조하세요.

```
aws s3api list-parts --bucket amzn-s3-demo-bucket1 --key 'census_data_file' --
upload-id <your-upload-id> --query '{Parts: Parts[*].{PartNumber: PartNumber, ETag:
 ETag, ChecksumSHA256: ChecksumSHA256}}' --output json > parts.json
```

`parts.json`이라는 새 파일이 생성됩니다. 이 파일에는 업로드한 모든 부분에 대한 JSON 형식의 정보가 들어 있습니다. `parts.json` 파일에는 멀티파트 업로드 프로세스를 완료하는 데 필요한 부분 번호, 해당 ETag 값 등 멀티파트 업로드의 각 부분에 대한 필수 정보가 포함되어 있습니다.

2. 텍스트 편집기를 사용하거나 터미널을 통해 `parts.json`을 열 수 있습니다. 예시 출력은 다음과 같습니다.

```
{
    "Parts": [
        {
            "PartNumber": 1,
            "ETag": "\"3c3097f89e2a2fece47ac54b243c9d97\"",
            "ChecksumSHA256": "fTPVHfyNHdv5VkR4S3EewdyioXECv7JBxN+d4FXYYTw="
        },
        {
            "PartNumber": 2,
            "ETag": "\"03c71cc160261b20ab74f6d2c476b450\"",
            "ChecksumSHA256": "VDWTa8enjOvULBAO3W2a6C+5/7ZnNjrnLApa1QVc3FE="
        },
        {
            "PartNumber": 3,
            "ETag": "\"81ae0937404429a97967dffa7eb4affb\"",
            "ChecksumSHA256": "cVVkXehUlzcwrBrXgPIM+EKQXPUvWist8mlUTCs4bg8="
        }
    ]
}
```

6단계: 멀티파트 업로드 완료

멀티파트 업로드의 모든 부분을 업로드하고 나열한 후, 마지막 단계는 멀티파트 업로드를 완료하는 것입니다. 이 단계는 업로드된 모든 부분을 S3 버킷의 단일 객체로 병합합니다.

> **Note**
>
> 요청에 `--checksum-sha256`을 포함하여 `complete-multipart-upload`를 직접적으로 호출하기 전에 객체 체크섬을 계산할 수 있습니다. 체크섬이 일치하지 않으면 Amazon S3가 요청에 실패합니다. 자세한 내용은 AWS Command Line Interface 사용 설명서의 complete-multipart-upload를 참조하세요.

멀티파트 업로드를 완료하려면

멀티파트 업로드를 완료하려면 `complete-multipart-upload` 명령을 사용합니다. 이 명령에는 5단계에서 생성한 `parts.json` 파일, 버킷 이름, 업로드 ID가 필요합니다. **<*amzn-s3-demo-bucket1*>**을 해당 버킷 이름으로 바꾸고 **<your-upload-id>**를 `parts.json`의 업로드 ID로 바꿉니다.

```
aws s3api complete-multipart-upload --multipart-upload file://parts.json --bucket amzn-s3-demo-bucket1 --key 'census_data_file' --upload-id <your-upload-id>
```

예시 출력은 다음과 같습니다.

```
{
    "ServerSideEncryption": "AES256",
    "Location": "https://amzn-s3-demo-bucket1.s3.us-east-2.amazonaws.com/census_data_file",
    "Bucket": "amzn-s3-demo-bucket1",
    "Key": "census_data_file",
    "ETag": "\"f453c6dccca969c457efdf9b1361e291-3\"",
    "ChecksumSHA256": "aI8EoktCdotjU8Bq46DrPCxQCGuGcPIhJ51noWs6hvk=-3"
}
```

> **Note**
>
> 개별 부분 파일은 아직 삭제하지 마세요. 함께 병합된 객체의 무결성을 확인하기 위해 해당 부분에 대해 체크섬을 수행하려면 개별 부분이 필요합니다.

7단계: 객체가 버킷에 업로드되었는지 확인

멀티파트 업로드를 완료한 후 객체가 S3 버킷에 성공적으로 업로드되었는지 확인할 수 있습니다. 버킷의 객체를 나열하고 새로 업로드한 파일이 있는지 확인하려면 `list-objects-v2` 명령을 사용합니다.

업로드된 객체를 나열하려면

버킷의 객체를 나열하려면 `list-objects-v2` 명령을 사용합니다. ***amzn-s3-demo-bucket1***을 실제 버킷 이름으로 바꿉니다.

```
aws s3api list-objects-v2 --bucket amzn-s3-demo-bucket1
```

이 명령은 버킷의 객체 목록을 반환합니다. 객체 목록에서 업로드한 파일(예:`census_data_file`)을 찾아보세요.

자세한 내용은 AWS Command Line Interface 사용 설명서에서 `list-objects-v2` 명령에 대한 예제 섹션을 참조하세요.

8단계: MD5 체크섬으로 객체 무결성 확인

객체를 업로드할 때 Amazon S3에서 사용할 체크섬 알고리즘을 지정할 수 있습니다. 기본적으로 Amazon S3는 MD5 바이트 다이제스트를 객체의 ETag로 저장합니다. 멀티파트 업로드의 경우 ETag는 전체 객체의 체크섬이 아니라 각 개별 부분에 대한 체크섬의 복합체입니다.

MD5 체크섬을 사용하여 객체 무결성을 확인하려면

1. 업로드된 객체의 ETag를 검색하려면 `head-object` 요청을 수행합니다.

   ```
   aws s3api head-object --bucket amzn-s3-demo-bucket1 --key census_data_file
   ```

   예시 출력은 다음과 같습니다.

   ```
   {
       "AcceptRanges": "bytes",
       "LastModified": "2024-07-26T19:04:13+00:00",
       "ContentLength": 16106127360,
       "ETag": "\"f453c6dccca969c457efdf9b1361e291-3\"",
       "ContentType": "binary/octet-stream",
       "ServerSideEncryption": "AES256",
       "Metadata": {}
   ```

```
}
```

이 ETag의 끝에는 '-3'이 추가되어 있습니다. 이는 멀티파트 업로드를 사용하여 객체가 세 부분으로 업로드되었음을 나타냅니다.

2. 그런 다음 `md5sum` 명령을 사용하여 각 부분의 MD5 체크섬을 계산합니다. 부분 파일에 올바른 경로를 입력했는지 확인합니다.

```
md5sum census-part*
```

예시 출력은 다음과 같습니다.

```
e611693805e812ef37f96c9937605e69 census-part00
63d2d5da159178785bfd6b6a5c635854 census-part01
95b87c7db852451bb38b3b44a4e6d310 census-part02
```

3. 이 단계에서는 MD5 해시를 수동으로 하나의 문자열로 결합합니다. 그러고 나서 다음 명령을 실행하여 문자열을 바이너리로 변환하고 바이너리 값의 MD5 체크섬을 계산합니다.

```
echo
 "e611693805e812ef37f96c9937605e6963d2d5da159178785bfd6b6a5c63585495b87c7db852451bb38b3b44a
 | xxd -r -p | md5sum
```

예시 출력은 다음과 같습니다.

```
f453c6dccca969c457efdf9b1361e291 -
```

이 해시 값은 `census_data_file` 객체의 무결성을 검증하는 1단계의 원래 ETag 값의 해시 값과 일치해야 합니다.

Amazon S3에게 추가 체크섬을 사용하도록 하면 Amazon S3가 각 부분에 대한 체크섬 값을 계산하고 저장합니다. 아직 진행 중인 멀티파트 업로드의 개별 부분에 대한 체크섬 값을 검색하려면 `list-parts`를 사용하면 됩니다.

멀티파트 업로드 객체를 통한 체크섬 작업에 대한 자세한 내용은 Amazon S3에서 객체 무결성 확인 섹션을 참조하세요.

9단계: 추가 체크섬으로 객체 무결성 확인

이 단계에서 이 튜토리얼은 SHA-256을 추가 체크섬으로 사용하여 객체 무결성을 검증합니다. 다른 추가 체크섬을 사용한 경우 해당 체크섬 값을 대신 사용하세요.

SHA256을 사용하여 객체 무결성을 확인하려면

1. 터미널에서 `--checksum-mode enabled` 인수를 포함해 다음 명령을 실행하여 객체의 ChecksumSHA256 값을 표시합니다.

```
aws s3api head-object --bucket amzn-s3-demo-bucket1 --key census_data_file --
checksum-mode enabled
```

예시 출력은 다음과 같습니다.

```
{
    "AcceptRanges": "bytes",
    "LastModified": "2024-07-26T19:04:13+00:00",
    "ContentLength": 16106127360,
    "ChecksumSHA256": "aI8EoktCdotjU8Bq46DrPCxQCGuGcPIhJ51noWs6hvk=-3",
    "ETag": "\"f453c6dccca969c457efdf9b1361e291-3\"",
    "ContentType": "binary/octet-stream",
    "ServerSideEncryption": "AES256",
    "Metadata": {}
}
```

2. 다음 명령을 사용하여 개별 부분의 ChecksumSHA256 값을 base64로 디코딩하고 `outfile`이라는 바이너리 파일에 저장합니다. 이러한 값은 `parts.json` 파일에서 찾을 수 있습니다. 예제 base64 문자열을 실제 ChecksumSHA256 값으로 바꿉니다.

```
echo "QLl8R4i4+SaJlrl8ZIcutc5TbZtwt2NwB8lTXkd3GH0=" | base64 --decode >> outfile
echo "xCdgs1K5Bm4jWETYw/CmGYr+m6O2DcGfpckx5NVokvE=" | base64 --decode >> outfile
echo "f5wsfsa5bB+yXuwzqG1Bst91uYneqGD3CCidpb54mAo=" | base64 --decode >> outfile
```

3. 다음 명령을 실행하여 `outfile`의 SHA256 체크섬을 계산합니다.

```
sha256sum outfile
```

예시 출력은 다음과 같습니다.

```
688f04a24b42768b6353c06ae3a0eb3c2c50086b8670f221279d67a16b3a86f9 outfile
```

다음 단계에서는 해시 값을 가져와 이진 값으로 변환합니다. 이 이진 값은 1단계의 ChecksumSHA256 값과 일치해야 합니다.

4. 3단계의 SHA256 체크섬을 이진수로 변환한 다음 base64로 인코딩하여 1단계의 ChecksumSHA256 값과 일치하는지 확인합니다.

```
echo "688f04a24b42768b6353c06ae3a0eb3c2c50086b8670f221279d67a16b3a86f9" | xxd -r -p
 | base64
```

예시 출력은 다음과 같습니다.

```
aI8EoktCdotjU8Bq46DrPCxQCGuGcPIhJ51noWs6hvk=
```

이 출력은 base64 출력이 `head-object` 명령 출력의 ChecksumSHA256 값과 일치하는지 확인합니다. 출력이 체크섬 값과 일치하면 객체가 유효합니다.

Important

- Amazon S3에게 추가 체크섬을 사용하도록 하면 Amazon S3가 각 부분에 대한 체크섬 값을 계산하고 저장합니다.
- 아직 진행 중인 멀티파트 업로드의 개별 부분에 대한 체크섬 값을 검색하려면 `list-parts` 명령을 사용하면 됩니다.

10단계: 리소스 정리

이 튜토리얼에서 생성된 파일을 정리하려면 다음 방법을 사용합니다. S3 버킷에 업로드된 파일을 삭제하는 방법에 대한 지침은 Amazon S3 객체 삭제 섹션을 참조하세요.

1단계에서 생성한 로컬 파일 삭제:

멀티파트 업로드를 위해 생성한 파일을 제거하려면 작업 디렉터리에서 다음 명령을 실행합니다.

```
rm census-data.bin census-part* outfile parts.json
```

Amazon S3 멀티파트 업로드 제한

멀티파트 업로드를 사용하면 단일 객체를 여러 부분의 집합으로 업로드할 수 있습니다. 각 부분은 객체 데이터의 연속적인 부분입니다. 객체의 모든 부분이 업로드되면 Amazon S3가 이들 부분을 수집하여 객체를 생성합니다. 일반적으로 객체 크기가 100MB에 근접할 경우, 단일 작업에서 객체를 업로드하는 대신 멀티파트 업로드 사용을 고려해 봐야 합니다. 멀티파트 업로드에 대한 자세한 내용은 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 섹션을 참조하십시오.

다음 표에 멀티파트 업로드의 주요 사양이 나와 있습니다. 여기에는 최대 객체 크기, 최대 부분 수, 최대 부분 크기 등이 포함됩니다. 멀티파트 업로드의 마지막 부분에는 최소 크기 제한이 없습니다.

항목	사양
최대 객체 크기	5TiB
업로드당 최대 부분 개수	10,000개
부분 번호	1 ~ 10,000(포함)
부분 크기	5MiB~5GiB. 멀티파트 업로드의 마지막 부분에는 최소 크기 제한이 없습니다.
부분 목록 조회 요청에 대해 반환되는 최대 부분 개수	1000
멀티파트 업로드 나열 요청에서 반환되는 최대 멀티파트 업로드 개수	1000

조건부 요청을 통해 S3 작업에 사전 조건 추가

조건부 요청을 사용하여 S3 작업에 사전 조건을 추가할 수 있습니다. 조건부 요청을 사용하려면 Amazon S3 API 작업에 헤더를 추가합니다. 이 헤더는 충족되지 않으면 S3 작업이 실패할 수 있는 조건을 지정합니다.

GET, HEAD 및 COPY 요청에 대해 조건부 읽기가 지원됩니다. 엔터티 태그(ETag) 또는 마지막 수정 날짜를 기준으로 객체를 반환하거나 복사하기 위한 사전 조건을 추가할 수 있습니다. 이렇게 하면 S3 작

업을 지정된 날짜 이후에 업데이트된 객체로 제한할 수 있습니다. 또한, S3 작업을 특정 ETag로 제한할 수 있습니다. 이렇게 하면 특정 객체 버전만 반환하거나 복사할 수 있습니다. 객체 메타데이터에 대한 자세한 내용은 객체 메타데이터 작업 단원을 참조하십시오.

조건부 쓰기를 사용하면 PUT 작업 중에 버킷에 동일한 키 이름을 가진 기존 객체가 없는지 확인할 수 있습니다. 이렇게 하면 기존 객체를 동일한 키 이름으로 덮어쓰지 않도록 방지할 수 있습니다. 마찬가지로 조건부 쓰기를 사용하여 객체를 업데이트하기 전에 객체의 ETag가 변경되지 않았는지 확인할 수 있습니다. 이렇게 하면 콘텐츠의 상태를 모르는 상태에서 객체를 의도하지 않게 덮어쓰는 것을 방지할 수 있습니다. PutObject 또는 CompleteMultipartUpload 요청에 조건부 쓰기를 사용할 수 있습니다. 키 이름에 대한 자세한 내용은 Amazon S3 객체 이름 지정 단원을 참조하십시오.

조건부 읽기 또는 조건부 쓰기에는 추가 요금이 부과되지 않습니다. 실패한 요청을 포함하여 해당 요청에 대한 기존 요금만 청구됩니다. Amazon S3 기능 및 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

주제

- 조건부 읽기를 사용하여 메타데이터를 기반으로 객체를 검색하거나 복사하는 방법
- 조건부 쓰기를 통해 객체 덮어쓰기를 방지하는 방법

조건부 읽기를 사용하여 메타데이터를 기반으로 객체를 검색하거나 복사하는 방법

조건부 읽기를 사용하면 읽기 요청에 헤더를 추가하여 S3 작업에 사전 조건을 추가할 수 있습니다. 이러한 사전 조건이 충족되지 않을 경우 읽기 요청이 실패합니다.

GET, HEAD 또는 COPY 요청에 대한 조건부 읽기를 사용하여 해당 메타데이터에 기반한 객체만 반환할 수 있습니다.

객체를 업로드하는 경우 Amazon S3는 S3만 수정할 수 있는 시스템 제어 메타데이터를 생성합니다. 엔터티 태그(ETag)와 Last-Modified를 시스템 제어 메타데이터의 예로 들 수 있습니다. 객체의 ETag는 객체의 특정 버전을 나타내는 문자열입니다. Last-Modified 날짜는 객체의 생성 날짜 또는 최종 수정 날짜 중 가장 최근 날짜를 나타내는 메타데이터입니다.

조건부 읽기를 사용하면 객체의 ETag 또는 최종 수정 날짜를 기준으로 객체를 반환할 수 있습니다. 요청에 ETag 값을 지정하고 ETag 값이 일치하는 경우에만 객체를 반환할 수 있습니다. 이렇게 하면 객체의 특정 버전만 반환하거나 복사할 수 있습니다. 읽기 요청에서 Last-Modified 값을 지정하고 입력한 날짜 이후에 해당 객체가 수정된 경우에만 객체를 반환할 수 있습니다.

지원되는 API

다음 S3 API는 조건부 읽기 사용을 지원합니다.

- GetObject
- HeadObject
- CopyObject

다음 헤더를 사용하여 엔터티 태그(ETag) 또는 마지막 수정 날짜에 따라 객체를 반환할 수 있습니다. ETag와 Last-Modified 등의 객체 메타데이터에 대한 자세한 내용은 the section called “시스템 정의 객체 메타데이터” 섹션을 참조하세요.

GetObject

- `If-Match` - ETag가 제공된 것과 일치하는 경우에만 객체를 반환합니다.
- `If-Modified-Since` - 지정된 시간 이후에 수정된 경우에만 객체를 반환합니다.
- `If-None-Match` - ETag가 제공된 것과 일치하지 않는 경우에만 객체를 반환합니다.
- `If-Unmodified-Since` - 지정된 시간 이후에 수정되지 않은 경우에만 객체를 반환합니다.

이러한 헤더, 반환된 오류, S3가 단일 요청에서 여러 조건부 헤더를 처리하는 순서에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 GetObject 섹션을 참조하세요.

HeadObject

- `If-Match` - ETag가 제공된 것과 일치하는 경우에만 객체를 반환합니다.
- `If-Modified-Since` - 지정된 시간 이후에 수정된 경우에만 객체를 반환합니다.
- `If-None-Match` - ETag가 제공된 것과 일치하지 않는 경우에만 객체를 반환합니다.
- `If-Unmodified-Since` - 지정된 시간 이후에 수정되지 않은 경우에만 객체를 반환합니다.

이러한 헤더, 반환된 오류, S3가 단일 요청에서 여러 조건부 헤더를 처리하는 순서에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 HeadObject 섹션을 참조하세요.

CopyObject

- `x-amz-copy-source-if-match` - ETag가 제공된 것과 일치하는 경우에만 소스 객체를 복사합니다.
- `x-amz-copy-source-if-modified-since` - 지정된 시간 이후에 수정된 경우에만 소스 객체를 복사합니다.
- `x-amz-copy-source-if-none-match` - ETag가 제공된 것과 일치하지 않는 경우에만 소스 객체를 복사합니다.
- `x-amz-copy-source-if-unmodified-since` - 지정된 시간 이후에 수정되지 않은 경우에만 소스 객체를 복사합니다.

이러한 헤더, 반환된 오류, S3가 단일 요청에서 여러 조건부 헤더를 처리하는 순서에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 CopyObject 섹션을 참조하세요.

조건부 쓰기를 통해 객체 덮어쓰기를 방지하는 방법

조건부 쓰기를 사용하면 `WRITE` 요청에 헤더를 더 추가하여 Amazon S3 작업에 사전 조건을 지정할 수 있습니다. 조건부로 객체를 작성하려면 HTTP `If-None-Match` 또는 `If-Match` 헤더를 추가합니다.

`If-None-Match` 헤더는 버킷에 이미 같은 키 이름을 가진 객체가 없는지 확인하여 기존 데이터를 덮어쓰지 않도록 방지합니다.

또는 객체를 쓰기 전에 `If-Match` 헤더를 추가하여 객체의 엔터티 태그(ETag)를 확인할 수 있습니다. 이 헤더를 사용하여 Amazon S3는 제공된 ETag 값을 S3에 있는 객체의 ETag 값과 비교합니다. 이때 ETag 값이 일치하지 않으면 작업이 실패합니다.

버킷 소유자는 버킷 정책을 사용하여 업로드된 객체에 조건부 쓰기를 강제 적용할 수 있습니다. 자세한 내용은 the section called “조건부 쓰기 강제 적용” 섹션을 참조하세요.

> **Note**
>
> 조건부 쓰기를 사용하려면 HTTPS(TLS)를 통해 요청을 수행하거나 AWS Signature Version 4를 사용하여 요청에 서명해야 합니다.

주제

- 키 이름을 기반으로 객체 덮어쓰기를 방지하는 방법
- 객체가 변경된 경우 덮어쓰기를 방지하는 방법
- 조건부 쓰기 동작

- 조건부 쓰기 시나리오
- Amazon S3 버킷에 조건부 쓰기 강제 적용

키 이름을 기반으로 객체 덮어쓰기를 방지하는 방법

HTTP `If-None-Match` 조건부 헤더를 사용하여 객체를 생성하기 전에 키 이름을 기반으로 지정된 버킷에 객체가 이미 존재하는지 확인할 수 있습니다. Amazon S3에 객체를 업로드할 때 키 이름, 즉 버킷에 있는 객체의 고유한 식별자(대소문자 구분)를 지정합니다. HTTP `If-None-Match` 헤더가 없으면 버전이 지정되지 않았거나 버전 지정이 일시 중지된 버킷에 동일한 키 이름을 가진 객체를 업로드하면 해당 객체가 덮어쓰입니다. 버전이 지정된 버킷에서는 가장 최근에 업로드한 객체가 최신 버전이 됩니다. HTTP `If-None-Match` 헤더가 포함된 조건부 쓰기는 `WRITE` 작업 중에 객체의 존재 여부를 확인합니다. 버킷에 동일한 키 이름이 있는 경우 작업이 실패합니다. 키 이름 사용에 대한 자세한 내용은 the section called “객체 이름 지정” 섹션을 참조하세요.

HTTP `If-None-Match` 헤더가 포함된 조건부 쓰기를 수행하려면 `s3:PutObject` 권한이 있어야 합니다. 이를 통해 호출자는 버킷에 객체가 있는지 확인할 수 있습니다. `If-None-Match` 헤더에는 *(별표) 값이 필요합니다.

`If-None-Match` 헤더를 다음 API와 함께 사용할 수 있습니다.

- PutObject
- CompleteMultipartUpload

AWS CLI 사용

다음 `put-object` 예시 명령은 키 이름이 `dir-1/my_images.tar.bz2`인 객체에 대해 조건부 쓰기를 수행하려고 시도합니다.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key dir-1/my_images.tar.bz2 --body
 my_images.tar.bz2 --if-none-match "*"
```

자세한 정보는 AWS CLI 명령 참조의 put-object 섹션을 참조하세요.

AWS CLI에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS Command Line Interface란 무엇입니까?를 참조하세요.

객체가 변경된 경우 덮어쓰기를 방지하는 방법

객체의 ETag는 객체에 고유한 문자열이며 객체의 콘텐츠에 대한 변경 사항을 반영합니다. `If-Match` 헤더를 사용하여 Amazon S3 버킷에 있는 객체의 ETag 값을 `WRITE` 작업 중에 제공한 값과 비교할 수 있습니다. 이때 ETag 값이 일치하지 않으면 작업이 실패합니다. ETag에 대한 자세한 내용은 the section called “Content-MD5 및 ETag를 사용하여 업로드된 객체 확인” 섹션을 참조하세요.

HTTP `If-Match` 헤더가 포함된 조건부 쓰기를 수행하려면 `s3:PutObject` 및 `s3:GetObject` 권한이 있어야 합니다. 그러면 호출자가 ETag를 확인하고 버킷에 있는 객체의 상태를 확인할 수 있습니다. `If-Match` 헤더에는 ETag 값이 문자열로 필요합니다.

`If-Match` 헤더를 다음 API와 함께 사용할 수 있습니다.

- PutObject
- CompleteMultipartUpload

AWS CLI 사용

다음 `put-object` 예시 명령은 제공된 ETag 값 `6805f2cfc46c0f04559748bb039d69ae`를 사용해 조건부 쓰기를 수행하려고 시도합니다.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key dir-1/my_images.tar.bz2 --body
 my_images.tar.bz2 --if-match "6805f2cfc46c0f04559748bb039d69ae"
```

자세한 정보는 AWS CLI 명령 참조의 put-object 섹션을 참조하세요.

AWS CLI에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS Command Line Interface란 무엇입니까?를 참조하세요.

조건부 쓰기 동작

`If-None-Match` 헤더가 포함된 조건부 쓰기

`If-None-Match` 헤더가 포함된 조건부 쓰기는 버킷의 기존 객체와 비교해 평가합니다. 버킷에 동일한 키 이름을 가진 기존 객체가 없는 경우 쓰기 작업이 성공하여 `200 OK` 응답이 반환됩니다. 기존 객체가 있는 경우 쓰기 작업이 실패하여 `412 Precondition Failed` 응답이 반환됩니다.

버전 관리가 활성화된 버킷의 경우 이름이 같은 현재 객체 버전이 없거나 현재 객체 버전이 삭제 마커인 경우 쓰기 작업이 성공합니다. 그렇지 않으면 `412 Precondition Failed` 응답과 함께 쓰기 작업이 실패하게 됩니다.

동일한 객체 이름에 대해 조건부 쓰기가 여러 번 발생하는 경우 가장 먼저 완료되는 쓰기 작업이 성공합니다. 그런 다음 Amazon S3에서 `412 Precondition Failed` 응답과 함께 후속 쓰기 작업이 실패합니다.

요청이 동시에 발생하는 경우에도 객체에 대한 조건부 쓰기 작업이 완료되기 전에 객체에 대한 삭제 요청이 성공하면 `409 Conflict` 응답이 반환될 수 있습니다. `PutObject`와 함께 조건부 쓰기를 사용하는 경우 `409 Conflict` 오류가 발생한 후 업로드가 재시도될 수 있습니다. `CompleteMultipartUpload`를 사용하는 경우 `409 Conflict` 오류가 발생한 후 객체를 다시 업로드하려면 `CreateMultipartUpload`로 멀티파트 업로드를 다시 시작해야 합니다.

`If-Match` 헤더가 포함된 조건부 쓰기

`If-Match` 헤더는 버킷의 기존 객체와 비교해 평가합니다. 버킷에 동일한 키 이름 및 일치하는 ETag를 가진 기존 객체가 있는 경우 쓰기 작업이 성공하여 `200 OK` 응답이 반환됩니다. ETag가 일치하지 않으면 `412 Precondition Failed` 응답과 함께 쓰기 작업이 실패합니다.

동시 요청의 경우에도 `409 Conflict` 응답을 받을 수 있습니다.

객체에 대한 조건부 쓰기 작업이 완료되기 전에 해당 객체에 대한 동시 삭제 요청이 성공하는 경우 객체 키가 더 이상 존재하지 않으므로 `404 Not Found` 응답이 반환됩니다. `404 Not Found` 응답을 받으면 객체를 다시 업로드해야 합니다.

이름이 같은 현재 객체 버전이 없거나 현재 객체 버전이 삭제 마커인 경우 작업이 `404 Not Found` 오류와 함께 실패합니다.

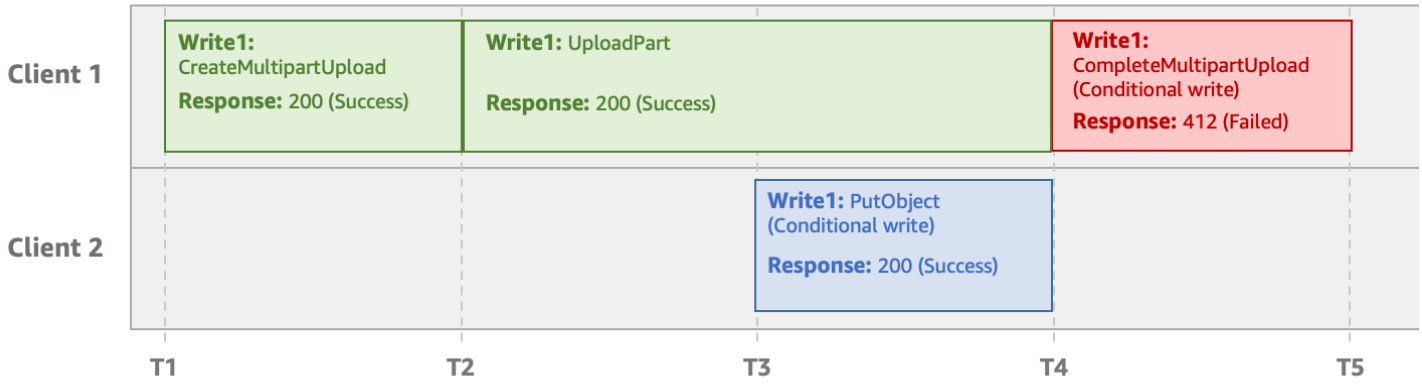
조건부 쓰기 시나리오

두 클라이언트가 같은 버킷에서 작업을 실행하는 다음 시나리오를 생각해 보세요.

멀티파트 업로드 중 조건부 쓰기

진행 중인 멀티파트 업로드 요청은 아직 완전히 쓴 객체가 아니므로 조건부 쓰기는 진행 중인 멀티파트 업로드 요청을 고려하지 않습니다. 클라이언트 1이 멀티파트 업로드를 사용하여 객체를 업로드하는 다음 예를 살펴보세요. 멀티파트 업로드 중에 클라이언트 2는 조건부 쓰기 작업을 통해 동일한 객체를 성공적으로 쓸 수 있습니다. 이후 클라이언트 1이 조건부 쓰기를 사용하여 멀티파트 업로드를 완료하려고 하면 업로드가 실패합니다.

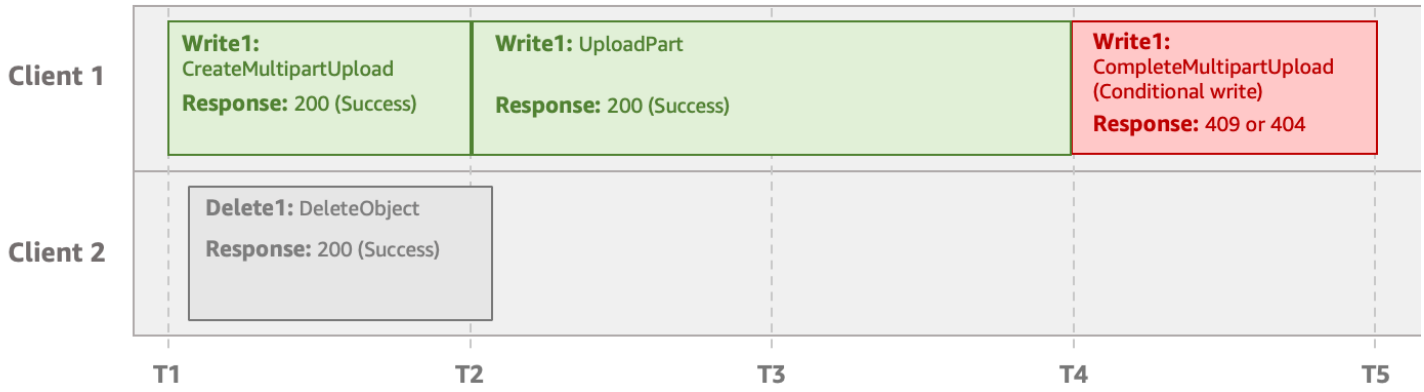
> **Note**
>
> 이 시나리오에서는 `If-None-Match` 및 `If-Match` 헤더 모두에 대해 `412 Precondition Failed` 응답이 반환됩니다.



멀티파트 업로드 중 동시 삭제

조건부 쓰기 요청이 완료되기 전에 삭제 요청이 성공하면 Amazon S3는 쓰기 작업에 대해 `409 Conflict` 또는 `404 Not Found` 응답을 반환합니다. 이전에 시작된 삭제 요청이 조건부 쓰기 작업보다 우선하기 때문입니다. 이 경우 새 멀티파트 업로드를 시작해야 합니다.

> **Note**
>
> 이 시나리오에서 `If-None-Match` 헤더에 대해서는 `409 Conflict` 응답이, `If-Match` 헤더에 대해서는 `404 Not Found` 응답이 반환됩니다.



> **Note**
>
> 스토리지 비용을 최소화하려면 `AbortIncompleteMultipartUpload` 작업을 사용하여 특정 기간이 지나면 불완전한 멀티파트 업로드를 삭제하는 수명 주기 규칙을 구성하는 것이 좋습니다. 불완전한 멀티파트 업로드를 삭제하기 위한 수명 주기 규칙을 만드는 방법과 관련하여 자세한 내용은 불완전한 멀티파트 업로드를 삭제하도록 버킷 수명 주기 구성 설정을 참조하십시오.

Amazon S3 버킷에 조건부 쓰기 강제 적용

Amazon S3 버킷 정책을 사용하면 범용 버킷의 객체 업로드에 조건부 쓰기를 강제 적용할 수 있습니다.

버킷 정책은 Amazon S3 버킷과 그 내부의 객체에 대한 액세스 권한을 부여할 수 있는 리소스 기반 정책입니다. 버킷 소유자만 정책을 버킷에 연결할 수 있습니다. 버킷 정책에 대한 자세한 내용은 the section called “버킷 정책”를 참조하십시오.

조건 키 `s3:if-match` 또는 `s3:if-none-match`를 선택적 `Condition` 요소 또는 `Condition` 블록을 사용하여 정책이 적용되는 시점을 지정할 수 있습니다. 멀티파트 업로드의 경우 `CreateMultipartUpload`, `UploadPart` 및 `UploadPartCopy` 작업을 제외하려면 `s3:ObjectCreationOperation` 조건 키를 지정해야 합니다. 이러한 API는 조건부 헤더를 허용하지 않기 때문입니다. 버킷 정책에서 조건을 사용하는 것에 대한 자세한 내용은 the section called “조건 키 예시” 섹션을 참조하세요.

> **Note**
>
> 버킷 정책을 사용하여 조건부 쓰기를 강제 적용하는 경우 버킷 정책에 지정된 버킷 또는 접두사에 복사 작업을 수행할 수 없습니다. `If-None-Match` 또는 `If-Match` HTTP 헤더가 없는 `CopyObject` 요청은 `403 Access Denied` 오류와 함께 실패합니다. 이러한 HTTP 헤더가 포함된 `CopyObject` 요청은 `501 Not Implemented` 응답과 함께 실패합니다.

다음 예시에서는 버킷 정책에서 조건을 사용하여 클라이언트가 `If-None-Match` 또는 `If-Match` HTTP 헤더를 사용하도록 강제하는 방법을 보여줍니다.

주제

- 예시 1: if-none-match 헤더를 포함하는 PutObject 및 CompleteMultipartUpload 요청만 사용하여 객체 업로드 허용
- 예시 2: if-match 헤더를 포함하는 PutObject 및 CompleteMultipartUpload 요청만 사용하여 객체 업로드 허용
- 예시 3: if-none-match 또는 if-match 헤더가 포함된 객체 업로드 요청만 허용

예시 1: **`if-none-match`** 헤더를 포함하는 **`PutObject`** 및 **`CompleteMultipartUpload`** 요청만 사용하여 객체 업로드 허용

이 정책은 요청에 `if-none-match` 헤더가 포함된 경우 계정 111122223333, 사용자 Alice가 *`amzn-s3-demo-bucket1`* 버킷에 쓸 수 있도록 허용하여 객체 키가 버킷에 이미 존재하지 않도록 합니다. 지정된 버킷에 대한 모든 `PutObject` 및 `CompleteMultipartUpload` 요청이 성공하려면 `if-none-match` 헤더가 포함되어야 합니다. 이 헤더를 사용하면 객체 키가 버킷에 없는 경우에만 이 버킷에 쓸 수 있습니다.

> **Note**
>
> 또한 이 정책은 `CreateMultipartUpload`, `UploadPart` 및 `UploadPartCopy` API를 사용하여 멀티파트 업로드를 허용하는 `s3:ObjectCreationOperation` 조건 키를 설정합니다.

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

```
            "Sid": "AllowConditionalPut",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Alice"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "Null": {
                    "s3:if-none-match": "false"
                }
            }
        },
        {
            "Sid": "AllowConditionalPutwithMPUs",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Alice"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "Bool": {
                    "s3:ObjectCreationOperation": "false"
                }
            }
        }
    ]
}
```

예시 2: **if-match** 헤더를 포함하는 **PutObject** 및 **CompleteMultipartUpload** 요청만 사용하여 객체 업로드 허용

이 정책은 계정 111122223333, 사용자 Alice가 요청에 `if-match` 헤더가 포함된 경우에만 *amzn-s3-demo-bucket1*에 쓸 수 있도록 허용합니다. 이 헤더는 S3에 있는 객체의 ETag 값을 `WRITE` 작업 중에 제공한 값과 비교합니다. ETag 값이 일치하지 않으면 작업이 실패합니다. 지정된 버킷에 대한 모든 `PutObject` 및 `CompleteMultipartUpload` 요청이 성공하려면 `if-match` 헤더가 포함되어야 합니다.

> **Note**
>
> 또한 이 정책은 `CreateMultipartUpload`, `UploadPart` 및 `UploadPartCopy` API를 사용하여 멀티파트 업로드를 허용하는 `s3:ObjectCreationOperation` 조건 키를 설정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutObject",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Alice"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
        },
        {
            "Sid": "BlockNonConditionalObjectCreation",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Alice"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "Null": {
                    "s3:if-match": "true"
                },
                "Bool": {
                    "s3:ObjectCreationOperation": "true"
                }
            }
        },
        {
            "Sid": "AllowGetObjectBecauseConditionalPutIfMatchETag",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Alice"
            },
            "Action": "s3:GetObject",
```

```
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*"
        }
    ]
}
```

예시 3: **if-none-match** 또는 **if-match** 헤더가 포함된 객체 업로드 요청만 허용

이 정책은 요청에 `if-none-match` 또는 `if-match` 헤더가 포함된 경우 계정 111122223333, 사용자 Alice가 *amzn-s3-demo-bucket1*에 쓸 수 있도록 허용합니다. 이렇게 하면 키 이름이 버킷에 존재하지 않을 경우 Alice가 객체를 업로드할 수 있습니다. 또는 키 이름이 존재할 경우 객체 ETag가 PUT 요청에 제공된 ETag와 일치하면 Alice가 객체를 덮어쓸 수 있습니다.

> **Note**
>
> 또한 이 정책은 `CreateMultipartUpload`, `UploadPart` 및 `UploadPartCopy` API를 사용하여 멀티파트 업로드를 허용하는 `s3:ObjectCreationOperation` 조건 키를 설정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": " AllowConditionalPutifAbsent",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Alice"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "Null": {
                    "s3:if-none-match": "false"
                }
            }
        },
        {
            "Sid": "AllowConditionalPutIfMatchEtag",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Alice"
            },
```

```
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "Null": {
                    "s3:if-match": "false"
                }
            }
        },
        {
            "Sid": "AllowConditionalObjectCreation",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Alice"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "Bool": {
                    "s3:ObjectCreationOperation": "false"
                }
            }
        },
        {
            "Sid": " AllowGetObjectBecauseConditionalPutIfMatchETag",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Alice"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*"
        }
    ]
}
```

객체 복사, 이동, 이름 변경

`CopyObject` 작업은 Amazon S3에 이미 저장되어 있는 객체의 복사본을 만듭니다.

단일 원자성 작업으로 최대 5GB의 객체 복사본을 만들 수 있습니다. 그러나 5GB보다 큰 객체를 복사하려면 AWS CLI 또는 AWS SDK를 사용하여 멀티파트 업로드를 사용해야 합니다. 자세한 내용은 멀티파트 업로드를 사용한 객체 복사 섹션을 참조하세요.

> **Note**
>
> 멀티파트 업로드를 사용하여 업로드한 객체의 성능상 이점을 그대로 누리려면 S3 콘솔 대신 AWS CLI 또는 AWS SDK를 사용하여 멀티파트 업로드를 사용해 객체를 복사해야 합니다. 자세한 내용은 멀티파트 업로드를 사용한 객체 복사 섹션을 참조하세요.

`CopyObject`를 사용하여 다음과 같은 작업을 수행할 수 있습니다.

- 객체의 추가 복사본을 생성합니다.
- 객체를 복사하여 이름을 변경한 후 원본을 삭제합니다.
- AWS 리전 간(예: `us-west-1`에서 `eu-west-2`로)을 포함하여 한 버킷에서 다른 버킷으로 객체를 복사 또는 이동합니다. 객체를 이동할 경우 Amazon S3는 객체를 지정된 대상으로 복사한 다음 원본 객체를 삭제합니다.

  > **Note**
  >
  > AWS 리전 간에 객체를 복사하거나 이동할 경우 대역폭 비용이 청구됩니다. 자세한 내용은 Amazon S3 요금을 참조하세요.

- 객체 메타데이터를 변경합니다. 각 Amazon S3 객체에는 메타데이터가 있습니다. 이 메타데이터는 이름-값 페어의 집합입니다. 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. 이렇게 하려면 복사 작업의 원본과 대상을 동일한 객체로 설정합니다.

  객체 메타데이터에는 시스템 메타데이터와 사용자 정의 메타데이터의 두 종류가 있습니다. 일부 시스템 메타데이터를 제어할 수 있습니다. 예를 들어, 객체에 사용할 스토리지 클래스와 서버 측 암호화 유형을 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 요청에서는 이러한 시스템 제어 메타데이터 값을 설정할 필요가 없습니다.

  객체를 복사할 때 일부 메타데이터 값을 업데이트할 수 있습니다. 예를 들어, 원본 객체가 S3 Standard 스토리지를 사용하도록 구성된 경우 객체 복사에 대해 S3 Intelligent-Tiering을 사용하도록 선택할 수 있습니다. 또한 원본 객체에 설정된 일부 사용자 정의 메타데이터 값을 변경할 수도 있습니다. 복사 과정에서 사용자가 구성 가능한 객체 메타데이터(시스템 또는 사용자 정의 메타데이터)

를 업데이트하려면 메타데이터 값 하나만 변경하려는 경우에도 요청의 원본 객체에 설정된 사용자가 구성 가능한 모든 메타데이터를 명시적으로 지정해야 합니다.

> **Note**
>
> Amazon S3 콘솔을 사용하여 객체를 복사할 때 "복사된 메타데이터를 확인할 수 없습니다."라는 오류 메시지가 표시될 수 있습니다. 콘솔은 헤더를 사용하여 객체에 대한 메타데이터를 검색하고 설정합니다. 네트워크 또는 브라우저 구성이 네트워크 요청을 수정하면 이 동작으로 인해 복사된 객체에 의도하지 않은 메타데이터(수정된 `Cache-Control` 헤더 등)가 기록될 수 있습니다. Amazon S3는 이 의도하지 않은 메타데이터를 확인할 수 없습니다. 이 문제를 해결하려면 네트워크 및 브라우저 구성을 확인하여 `Cache-Control`과 같은 헤더를 수정하지 않는지 확인하세요. 자세한 내용은 The Shared Responsibility Model을 참조하세요.

객체 메타데이터에 대한 자세한 정보는 객체 메타데이터 작업 섹션을 참조하십시오.

아카이브 및 복원된 객체 복사

소스 객체가 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive에 아카이브되어 있는 경우 객체를 다른 버킷에 복사하기 전에 먼저 임시 복사본을 복원해야 합니다. 객체 보관에 대한 자세한 내용은 아카이브된 객체 작업 단원을 참조하십시오.

Amazon S3 콘솔의 복사 작업은 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스의 복원된 객체에 대해 지원되지 않습니다. 이러한 복원된 객체를 복사하려면 AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용합니다.

암호화된 객체 복사

Amazon S3는 S3 버킷에 복사되는 모든 새 객체를 자동으로 암호화합니다. 복사 요청에서 암호화 정보를 지정하지 않은 경우 대상 객체의 암호화 설정이 대상 버킷의 기본 암호화 구성으로 설정됩니다. 기본적으로 모든 버킷은 Amazon S3 관리형 키(SSE-S3)를 통한 서버 측 암호화를 사용하는 기본 수준의 암호화 구성을 가지고 있습니다. 대상 버킷에 AWS Key Management Service(AWS KMS) 키(SSE-KMS) 또는 고객 제공 암호화 키(SSE-C)를 통한 서버 측 암호화를 사용하는 기본 암호화 구성이 있는 경우, Amazon S3는 해당 KMS 키 또는 고객이 제공한 키를 사용하여 대상 객체 사본을 암호화합니다.

객체를 복사할 때 대상 객체에 다른 유형의 암호화 설정을 사용하려면 Amazon S3가 KMS 키, Amazon S3 관리형 키 또는 고객 제공 키를 사용하여 대상 객체를 암호화하도록 요청할 수 있습니다. 요청의 암호화 설정이 대상 버킷의 기본 암호화 구성과 다른 경우 요청의 암호화 설정이 우선합니다. 사본의 원

본 객체가 SSE-C를 사용하여 암호화된 경우, Amazon S3가 복사를 위해 객체를 해독할 수 있도록 필요한 암호화 정보를 요청에 시 제공해야 합니다. 자세한 내용은 암호화로 데이터 보호 섹션을 참조하세요.

객체 복사 시 체크섬 사용

객체를 복사할 때 객체에 대해 다른 체크섬 알고리즘을 사용하도록 선택할 수 있습니다. 동일한 알고리즘을 사용하든 새 알고리즘을 사용하든 Amazon S3는 객체를 복사한 후 새 체크섬 값을 계산합니다. Amazon S3에서는 체크섬 값을 직접 복사하지 않습니다. 체크섬 및 지정된 대상 체크섬 알고리즘이 없이 복사된 모든 객체는 `CRC-64NVME` 체크섬 알고리즘을 자동으로 얻습니다. 체크섬 계산에 대한 자세한 내용은 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 단원을 참조하십시오.

단일 요청으로 여러 객체 복사

단일 요청으로 둘 이상의 Amazon S3 객체를 복사하기 위해 S3 배치 작업을 사용할 수도 있습니다. 작업할 객체 목록을 S3 배치 작업에 제공합니다. S3 배치 작업은 지정된 작업을 수행하기 위해 각 API 작업을 호출합니다. 단일 배치 작업 건으로 엑사바이트 규모의 데이터가 포함된 수십억 개의 객체에서 지정된 작업을 수행할 수 있습니다.

S3 배치 작업 기능은 진행 상황을 추적하고 알림을 보내며 모든 작업에 대한 자세한 완료 보고서를 저장하여 감사 가능한 완전관리형 서버리스 환경을 제공합니다. Amazon S3 콘솔, AWS CLI, AWS SDK 또는 REST API를 통해 S3 배치 작업을 사용할 수 있습니다. 자세한 내용은 the section called “배치 작업 기본 사항” 섹션을 참조하세요.

디렉터리 버킷에 객체 복사

객체를 디렉터리 버킷으로 복사하는 방법에 대한 자세한 내용은 디렉터리 버킷에서 또는 디렉터리 버킷에 객체 복사 단원을 참조하세요. Amazon S3 Express One Zone 스토리지 클래스를 디렉터리 버킷과 함께 사용하는 방법에 대한 자세한 내용은 S3 Express One Zone 및 디렉터리 버킷 작업 단원을 참조하세요.

객체 복사

객체를 복사하려면 다음 메서드를 사용합니다.

S3 콘솔 사용

> **Note**
>
> 콘솔을 사용하여 객체를 복사할 때의 규제 및 제한 사항은 다음과 같습니다.

- 객체가 5GB 미만인 경우 객체를 복사할 수 있습니다. 객체가 5GB보다 큰 경우, 객체를 복사하려면 AWS CLI 또는 AWS SDK를 사용해야 합니다.
- 객체를 복사하는 데 필요한 추가 권한 목록은 the section called “S3 API 작업에 필요한 권한”을 참조하세요. 이 권한을 부여하는 정책의 예시는 the section called “자격 증명 기반 정책 예제” 단원을 참조하세요.
- Copy 작업은 지정된 폴더 내의 모든 객체(접두사)에 적용됩니다. 작업이 진행되는 동안 이 폴더에 추가된 객체가 영향을 받을 수 있습니다.
- Amazon S3 콘솔에서는 SSE-KMS로 암호화된 객체의 교차 리전 복사가 지원되지 않습니다. SSE-KMS로 암호화된 객체를 리전 간에 복사하려면 AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용합니다.
- 고객 제공 암호화 키(SSE-C)로 암호화된 객체는 S3 콘솔을 사용하여 복사할 수 없습니다. SSE-C로 암호화된 객체를 복사하려면 AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용합니다.
- 복사된 객체에는 원본 객체의 Object Lock 설정이 유지되지 않습니다.
- 객체를 복사하는 버킷이 S3 객체 소유권에 대해 버킷 소유자 적용 설정을 사용하는 경우 객체 ACL은 지정된 대상에 복사되지 않습니다.
- S3 객체 소유권에 대한 버킷 소유자 적용 설정을 사용하는 버킷에 객체를 복사하려면 소스 버킷도 버킷 소유자 적용 설정을 사용하거나 다른 AWS 계정 및 그룹에 부여된 객체 ACL을 제거해야 합니다.

객체 복사

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷 또는 디렉터리 버킷을 선택합니다.
3. 버킷 목록에서 복사할 객체가 들어 있는 버킷 이름을 선택합니다.
4. 복사할 객체 이름 왼쪽에 있는 확인란을 선택합니다.
5. 작업 메뉴에 표시되는 옵션 목록에서 복사를 선택합니다.
6. 대상 유형과 대상 계정을 선택합니다. 대상 경로를 지정하려면 S3 찾아보기를 선택하고 대상으로 이동한 후 대상 왼쪽에 있는 확인란을 선택합니다. 오른쪽 하단 모서리에서 대상 선택(Choose destination)을 선택합니다.

   또는 대상 경로를 입력합니다.

7. 버킷 버전 관리를 사용 설정하지 않은 경우, 의도치 않게 객체를 덮어쓰거나 삭제하는 것을 방지하기 위해 버킷 버전 관리를 사용하도록 권장하는 경고 메시지가 표시됩니다. 객체의 모든 버전을 이 버킷에 보관하려면 버킷 버전 관리 사용(Enable Bucket Versioning)을 선택합니다. 대상 세부 사항에서 기본 암호화 및 S3 Object Lock 속성을 볼 수도 있습니다.
8. 추가 복사 설정에서 소스 설정 복사, 설정 지정 안 함 또는 설정 지정 중 원하는 것을 선택합니다. 소스 설정 복사가 기본 옵션입니다. 소스 설정 속성 없이 객체만 복사하려면 설정 지정 안 함을 선택합니다. 스토리지 클래스, ACL, 객체 태그, 메타데이터, 서버 측 암호화 및 추가 체크섬에 대한 설정을 지정하려면 설정 지정을 선택합니다.
9. 오른쪽 하단 모서리에서 복사(Copy)를 선택합니다. Amazon S3가 객체를 대상에 복사합니다.

AWS SDK 사용

이 섹션의 예제에서는 단일 작업으로 최대 5GB의 객체를 복사하는 방법을 보여줍니다. 5GB보다 큰 객체를 복사하려면 멀티파트 업로드를 사용해야 합니다. 자세한 내용은 멀티파트 업로드를 사용한 객체 복사 섹션을 참조하세요.

Java

Example

다음 예제에서는 AWS SDK for Java를 사용하여 Amazon S3에서 객체를 복사합니다. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 AWS SDK for Java 개발자 안내서의 시작하기 섹션을 참조하세요.

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

.NET

다음 C# 예제에서는 상위 수준의 SDK for .NET을 사용하여 단일 작업으로 최대 5GB의 객체를 복사합니다. 5GB보다 큰 객체의 경우 멀티파트 업로드를 사용한 객체 복사에 설명된 멀티파트 업로드 복사 예제를 참조하십시오.

이 예제에서는 최대 5GB의 객체 복사본을 만듭니다. 코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

```
using Amazon;
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);
            }
        }
    }
}
```

PHP

이 주제에서는 AWS SDK for PHP 버전 3의 클래스를 사용하여 Amazon S3 내 단일 객체 및 여러 객체를 한 버킷에서 다른 버킷으로 또는 동일한 버킷 내에서 복사하는 방법을 보여줍니다.

AWS SDK for Ruby API에 대한 자세한 내용은 AWS SDK for Ruby – 버전 2를 참조하세요.

다음 PHP 예시는 `copyObject()` 메서드를 사용하여 Amazon S3 내에서 단일 객체를 복사하는 과정을 보여 줍니다. `getcommand()` 메서드를 사용하여 `CopyObject`를 일괄적으로 직접 호출함으로써 객체의 여러 복사본을 만드는 방법도 보여 줍니다.

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 직접 호출을 실행합니다. 첫 번째 인수로 `CopyObject` 명령을, 두 번째 인수로 원본 버킷, 원본 키 이름, 대상 버킷, 대상 키 이름이 포함된 배열을 제공합니다.

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

use Aws\CommandPool;
use Aws\Exception\AwsException;
use Aws\ResultInterface;
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
}
```

Python

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
 Boto3
```

```
                        that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key
```

```
    def copy(self, dest_object):
        """
        Copies the object to another bucket.

        :param dest_object: The destination object initialized with a bucket and
key.
                            This is a Boto3 Object resource.
        """
        try:
            dest_object.copy_from(
                CopySource={"Bucket": self.object.bucket_name, "Key":
self.object.key}
            )
            dest_object.wait_until_exists()
            logger.info(
                "Copied object from %s:%s to %s:%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
        except ClientError:
            logger.exception(
                "Couldn't copy object from %s/%s to %s/%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
            raise
```

Ruby

다음 작업은 Ruby 클래스를 사용하여 한 버킷에서 다른 버킷으로 또는 동일한 버킷 내에서 Amazon S3의 객체를 복사하는 방법을 보여 줍니다.

객체 복사

1	AWS SDK for Ruby 버전 3에 대해 모듈화된 Amazon S3 gem을 사용하고 `aws-sdk-s3` 를 요구하고 AWS 자격 증명을 제공합니다. 자격 증명을 제공하는 방법에 대한 자세한 내용은 Amazon S3 API 참조에서 AWS 계정 또는 IAM 사용자 자격 증명을 사용하여 요청하기를 참조하세요.
2	원본 버킷 이름, 원본 키 이름, 대상 버킷 이름, 대상 키 등의 요청 정보를 제공합니다.

다음 Ruby 코드 예시는 `#copy_object` 메서드를 사용하여 앞서 설명한, 한 버킷에서 다른 버킷으로 객체를 복사하는 작업을 수행하는 방법을 보여 줍니다.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
 target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
  end
end
```

```
# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

REST API 사용

이 예시는 Amazon S3 REST API를 사용하여 객체를 복사하는 방법을 보여 줍니다. REST API에 대한 자세한 내용은 CopyObject 단원을 참조하십시오.

이 예제는 메타데이터를 유지하면서 `flotsam` 버킷의 *amzn-s3-demo-bucket1* 객체를 `jetsam` 버킷의 *amzn-s3-demo-bucket2* 객체로 복사합니다.

```
PUT /jetsam HTTP/1.1
Host: amzn-s3-demo-bucket2.s3.amazonaws.com
x-amz-copy-source: /amzn-s3-demo-bucket1/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:/amzn-s3-demo-bucket1/flotsam\r\n
```

```
/amzn-s3-demo-bucket2/jetsam
```

Amazon S3는 객체의 ETag와 최종 수정 일시를 지정하는 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>
```

AWS CLI 사용

또한 AWS Command Line Interface(AWS CLI)를 사용하여 S3 객체를 복사할 수도 있습니다. 자세한 정보는 AWS CLI 명령 참조의 copy-object 섹션을 참조하세요.

AWS CLI에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS Command Line Interface란 무엇입니까?를 참조하세요.

객체를 이동하려면

객체를 복사하려면 다음 메서드를 사용합니다.

S3 콘솔 사용

> **Note**
>
> - 객체가 5GB 미만인 경우 객체를 이동할 수 있습니다. 객체가 5GB보다 큰 경우, 객체를 이동하려면 AWS CLI 또는 AWS SDK를 사용해야 합니다.
> - 객체를 이동하는 데 필요한 추가 권한 목록은 the section called “S3 API 작업에 필요한 권한”을 참조하세요. 이 권한을 부여하는 정책의 예시는 the section called “자격 증명 기반 정책 예제” 단원을 참조하세요.

- 고객 제공 암호화 키(SSE-C)로 암호화된 객체는 S3 콘솔을 사용하여 이동할 수 없습니다. SSE-C로 암호화된 객체를 이동하려면 AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용합니다.
- 폴더를 이동할 때 폴더를 추가로 변경하기 전에 이동 작업이 완료될 때까지 기다립니다.
- Amazon S3 콘솔에서는 S3 액세스 포인트 별칭을 이동 작업의 원본 또는 대상으로 사용할 수 없습니다.

객체를 이동하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다. 이동할 객체가 포함된 Amazon S3 버킷 또는 폴더로 이동합니다.
3. 이동하려는 객체의 확인란을 선택합니다.
4. 작업 메뉴에서 이동을 선택합니다.
5. 대상 경로를 지정하려면 S3 찾아보기를 선택하고 대상으로 이동한 후 대상 확인란을 선택합니다. 목적지 선택(Choose destination)을 선택합니다.

   또는 대상 경로를 입력합니다.
6. 버킷 버전 관리를 사용 설정하지 않은 경우, 의도치 않게 객체를 덮어쓰거나 삭제하는 것을 방지하기 위해 버킷 버전 관리를 사용하도록 권장하는 경고 메시지가 표시됩니다. 객체의 모든 버전을 이 버킷에 보관하려면 버킷 버전 관리 사용(Enable Bucket Versioning)을 선택합니다. 대상 세부 사항에서 기본 암호화 및 Object Lock 속성을 볼 수도 있습니다.
7. 추가 복사 설정에서 소스 설정 복사, 설정 지정 안 함 또는 설정 지정 중 원하는 것을 선택합니다. 소스 설정 복사가 기본 옵션입니다. 소스 설정 속성 없이 객체만 복사하려면 설정 지정 안 함을 선택합니다. 스토리지 클래스, ACL, 객체 태그, 메타데이터, 서버 측 암호화 및 추가 체크섬에 대한 설정을 지정하려면 설정 지정을 선택합니다.
8. 오른쪽 하단 모서리에서 이동(Move)을 선택합니다. Amazon S3가 객체를 대상으로 이동합니다.

AWS CLI 사용

또한 AWS Command Line Interface(AWS CLI)를 사용하여 S3 객체를 이동할 수도 있습니다. 자세한 정보는 AWS CLI 명령 참조의 mv 섹션을 참조하세요.

AWS CLI에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS Command Line Interface란 무엇입니까?를 참조하세요.

객체의 이름을 바꾸려면

객체의 이름을 변경하려면 다음 절차를 따르세요.

> **Note**
>
> - 객체가 5GB 미만인 경우 객체의 이름을 바꿀 수 있습니다. 5GB를 초과하는 객체의 이름을 변경하려면 AWS CLI 또는 AWS SDK를 사용하여 새 이름으로 객체를 복사한 다음 원본 객체를 삭제해야 합니다.
> - 객체를 복사하는 데 필요한 추가 권한 목록은 the section called “S3 API 작업에 필요한 권한”을 참조하세요. 이 권한을 부여하는 정책의 예시는 the section called “자격 증명 기반 정책 예제” 단원을 참조하세요.
> - 객체 이름을 변경하면 새 마지막 수정 날짜로 객체 사본이 만들어지고 원본 객체에 삭제 마커가 추가됩니다.
> - 기본 암호화에 대한 버킷 설정은 암호화되지 않은 지정된 모든 객체에 자동으로 적용됩니다.
> - 고객 제공 암호화 키(SSE-C)를 사용하는 객체는 Amazon S3 콘솔을 사용하여 이름을 바꿀 수 없습니다. SSE-C로 암호화된 객체의 이름을 변경하려면, AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용하여 새 이름으로 해당 객체를 복사합니다.
> - 이 버킷에서 S3 객체 소유권에 대해 버킷 소유자 적용 설정을 사용하는 경우 객체 액세스 제어 목록(ACL)은 복사되지 않습니다.

객체의 이름을 바꾸려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 탐색 창에서 버킷을 선택하고 범용 버킷 탭을 선택합니다. 이름을 변경할 객체가 포함된 Amazon S3 버킷 또는 폴더로 이동합니다.
3. 이름을 바꾸려는 객체의 확인란을 선택합니다.
4. 작업 메뉴에서 객체 이름 바꾸기를 선택합니다.
5. 새 객체 이름 상자에 객체의 새 이름을 입력합니다.
6. 추가 복사 설정에서 소스 설정 복사, 설정 지정 안 함 또는 설정 지정 중 원하는 것을 선택합니다. 소스 설정 복사가 기본 옵션입니다. 소스 설정 속성 없이 객체만 복사하려면 설정 지정 안 함을 선

택합니다. 스토리지 클래스, ACL, 객체 태그, 메타데이터, 서버 측 암호화 및 추가 체크섬에 대한 설정을 지정하려면 설정 지정을 선택합니다.

7. 변경 사항 저장을 선택합니다. Amazon S3가 객체의 이름을 바꿉니다.

객체 다운로드

이 섹션에서는 Amazon S3 버킷에서 객체를 다운로드하는 방법을 설명합니다. Amazon S3를 사용하면 하나 이상의 버킷에 객체를 저장할 수 있으며, 각 객체의 크기는 최대 5TB까지 가능합니다. 아카이브되지 않은 모든 Amazon S3 객체는 실시간으로 액세스할 수 있습니다. 그러나 아카이브된 객체는 다운로드하기 전에 복원해야 합니다. 아카이브된 객체 다운로드에 대한 자세한 내용은 the section called "아카이브된 객체 다운로드"를 참조하십시오.

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하여 단일 객체를 다운로드할 수 있습니다. 코드를 작성하거나 명령을 실행하지 않고 S3에서 객체를 다운로드하려면 S3 콘솔을 사용하십시오. 자세한 내용은 the section called "객체 다운로드" 섹션을 참조하세요.

여러 개의 객체를 다운로드하려면 AWS CloudShell, AWS CLI 또는 AWS SDK를 사용하십시오. 자세한 내용은 the section called "여러 객체 다운로드" 섹션을 참조하세요.

객체의 일부만 다운로드해야 하는 경우, AWS CLI 또는 REST API에서 추가 파라미터를 사용하여 다운로드할 바이트만 지정할 수 있습니다. 자세한 내용은 the section called "객체의 일부 다운로드" 섹션을 참조하세요.

소유하지 않은 객체를 다운로드해야 하는 경우, 객체 소유자에게 객체를 다운로드할 수 있는 미리 지정된 URL을 생성해 달라고 요청하십시오. 자세한 내용은 the section called "다른 AWS 계정에서 객체 다운로드" 섹션을 참조하세요.

AWS 네트워크 외부에서 객체를 다운로드하는 경우 데이터 전송 수수료가 적용됩니다. 동일한 AWS 리전 내에서는 AWS 네트워크 내 데이터 전송은 무료이지만 모든 GET 요청에 대해서는 요금이 부과됩니다. 데이터 전송 비용 및 데이터 검색 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

주제

- 객체 다운로드
- 여러 객체 다운로드
- 객체의 일부 다운로드
- 다른 AWS 계정에서 객체 다운로드

- 아카이브된 객체 다운로드
- 메타데이터를 기반으로 한 객체 다운로드
- 객체 다운로드 문제 해결

객체 다운로드

Amazon S3 콘솔, AWS CLI, AWS SDK 또는 REST API를 사용하여 객체를 다운로드할 수 있습니다.

S3 콘솔 사용

이 단원에서는 Amazon S3 콘솔을 사용하여 S3 버킷에서 단일 객체를 다운로드하는 방법을 설명합니다.

> **Note**
>
> - 한 번에 하나의 객체만 다운로드할 수 있습니다.
> - Amazon S3 콘솔을 사용하여 키 이름이 마침표(.)로 끝나는 객체를 다운로드하는 경우, 다운로드한 객체의 키 이름에서 마침표가 제거됩니다. 다운로드한 객체 이름 끝에 마침표를 유지하려면 AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용해야 합니다.

S3 버킷에서 객체 다운로드

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷 또는 디렉터리 버킷을 선택합니다.
3. 버킷 목록에서 객체를 다운로드할 버킷의 이름을 선택합니다.
4. 다음 방법 중 하나를 사용하여 S3 버킷에서 객체를 다운로드할 수 있습니다.
   - 객체 옆의 확인란을 선택하고 다운로드를 선택합니다. 특정 폴더에 객체를 다운로드하려면 작업 메뉴에서 다른 이름으로 다운로드를 선택합니다.
   - 특정 버전의 객체를 다운로드하려면 버전 표시(검색 상자 옆에 있음)를 켭니다. 원하는 객체 버전 옆의 확인란을 선택하고 다운로드를 선택합니다. 특정 폴더에 객체를 다운로드하려면 작업 메뉴에서 다른 이름으로 다운로드를 선택합니다.

AWS CLI 사용

다음 `get-object` 예제 명령은 AWS CLI를 사용하여 Amazon S3에서 객체를 다운로드하는 방법을 보여줍니다. 이 명령은 *amzn-s3-demo-bucket1* 버킷에서 객체 *folder/my_image*를 가져옵니다. 다운로드한 객체의 파일 이름인 `outfile`을 포함해야 합니다(예: *my_downloaded_image.jpg*).

```
aws s3api get-object --bucket amzn-s3-demo-bucket1 --key folder/
my_image my_downloaded_image.jpg
```

자세한 내용과 예제는 AWS CLI 명령 참조에서 get-object를 참조하세요.

AWS SDK 사용

AWS SDK를 사용하여 객체를 다운로드하는 방법에 대한 예제는 Amazon S3 API 참조의 코드 예제를 참조하세요.

다양한 AWS SDK 사용에 대한 일반적인 정보는 Amazon S3 API 참조의 AWS SDK를 사용하여 Amazon S3에서 개발을 참조하세요.

REST API 사용

REST API를 사용하여 Amazon S3에서 객체를 검색할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 GetObject를 참조하십시오.

여러 객체 다운로드

AWS CloudShell, AWS CLI 또는 AWS SDK를 사용하여 여러 객체를 다운로드할 수 있습니다.

AWS Management Console에서 AWS CloudShell 사용

AWS CloudShell은 브라우저 기반의 사전 인증된 쉘로, AWS Management Console에서 바로 시작할 수 있습니다.

AWS CloudShell에 대한 자세한 내용은 AWS CloudShell 사용 안내서에서 CloudShell이란 무엇인가요?를 참조하십시오.

> **Important**
>
> AWS CloudShell을 사용하면 홈 디렉터리에는 AWS 리전당 최대 1GB의 저장 공간이 있습니다. 따라서 객체가 이 용량을 초과하면 동기화할 수 없습니다. 자세한 제한 사항은 AWS CloudShell 사용 안내서에서 서비스 할당량 및 제한 사항을 참조하십시오.

AWS CloudShell을 사용하여 객체를 다운로드하려면 다음을 수행하십시오.

1. AWS Management Console에 로그인하고 CloudShell 콘솔(https://console.aws.amazon.com/cloudshell/)을 엽니다.
2. 다음 명령을 실행하여 버킷의 객체를 CloudShell에 동기화합니다. 다음 명령은 *amzn-s3-demo-bucket1*이라는 버킷의 객체를 동기화하고 CloudShell에 *temp*라는 폴더를 생성합니다. CloudShell은 이 폴더에 객체를 동기화합니다. 이 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체합니다.

   ```
   aws s3 sync s3://amzn-s3-demo-bucket1 ./temp
   ```

   > **Note**
   >
   > sync 명령은 디렉터리 버킷과 호환되지 않습니다.
   > 특정 객체를 제외하거나 포함하기 위해 패턴 일치를 수행하려면 sync 명령과 함께 --exclude "*value*" 및 --include "*value*" 파라미터를 사용할 수 있습니다.

3. 다음 명령을 실행하여 *temp*라는 폴더에 있는 객체를 *temp.zip*이라는 파일로 압축합니다.

   ```
   zip temp.zip -r temp/
   ```

4. 작업을 선택한 다음 파일 다운로드를 선택합니다.
5. 파일 이름 **temp.zip**을 입력한 다음 다운로드를 선택합니다.
6. (선택 사항) CloudShell에서 *temp.zip* 파일과 *temp* 폴더에 동기화된 객체를 삭제합니다. AWS CloudShell을 사용하면 각 AWS 리전에 대해 최대 1GB의 영구 스토리지가 제공됩니다.

   다음 예제 명령을 사용하여 .zip 파일과 폴더를 삭제할 수 있습니다. 이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

   ```
   rm temp.zip && rm -rf temp/
   ```

AWS CLI 사용

다음 예제는 AWS CLI를 사용하여 지정된 디렉토리 또는 접두사 아래에 있는 모든 파일 또는 객체를 다운로드하는 방법을 보여줍니다. 이 명령은 *amzn-s3-demo-bucket1* 버킷의 모든 객체를 현재 디렉터리로 복사합니다. 이 예제 명령을 사용하려면 *amzn-s3-demo-bucket1* 대신 버킷 이름을 사용하십시오.

```
aws s3 cp s3://amzn-s3-demo-bucket1 . --recursive
```

다음 명령은 *amzn-s3-demo-bucket1* 버킷의 접두사 *logs*에 해당하는 모든 객체를 현재 디렉터리로 다운로드합니다. 또한 `--exclude` 및 `--include` 파라미터를 사용하여 접미사가 *.log*인 객체만 복사합니다. 이 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하십시오.

```
aws s3 cp s3://amzn-s3-demo-bucket1/logs/ . --recursive --exclude "*" --include "*.log"
```

자세한 내용과 예제는 AWS CLI 명령 참조에서 cp를 참조하세요.

AWS SDK 사용

AWS SDK를 사용하여 Amazon S3 버킷의 모든 객체를 다운로드하는 방법에 대한 예제는 Amazon S3 API 참조의 코드 예제를 참조하세요.

다양한 AWS SDK 사용에 대한 일반적인 정보는 Amazon S3 API 참조의 AWS SDK를 사용하여 Amazon S3에서 개발을 참조하세요.

객체의 일부 다운로드

AWS CLI 또는 REST API를 사용하여 객체의 일부를 다운로드할 수 있습니다. 이렇게 하려면 추가 파라미터를 사용하여 다운로드할 객체의 일부를 지정합니다.

AWS CLI 사용

다음 예제 명령은 *amzn-s3-demo-bucket1*이라는 버킷에 있는 *folder/my_data*라는 객체의 바이트 범위에 대한 GET 요청을 수행합니다. 요청에서 바이트 범위 앞에 `bytes=`를 붙여야 합니다. 부분 객체는 *my_data_range*라는 출력 파일로 다운로드됩니다. 이 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하십시오.

```
aws s3api get-object --bucket amzn-s3-demo-bucket1 --key folder/my_data --range
 bytes=0-500 my_data_range
```

자세한 내용과 예제는 AWS CLI 명령 참조에서 get-object를 참조하십시오.

HTTP Range 헤더에 대한 자세한 내용은 RFC 편집기 웹 사이트에서 RFC 9110을 참조하십시오.

> **Note**
>
> Amazon S3는 단일 GET 요청에서 여러 범위의 데이터 검색을 지원하지 않습니다.

REST API 사용

REST API의 `partNumber` 및 `Range` 파라미터를 사용하여 Amazon S3에서 객체 파트를 검색할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 GetObject를 참조하십시오.

다른 AWS 계정에서 객체 다운로드

미리 지정된 URL을 사용하여 버킷 정책을 업데이트하지 않고도 다른 사람에게 객체에 대한 시간 제한 액세스를 허용할 수 있습니다.

미리 지정된 URL은 브라우저에 입력하거나 프로그램에서 객체를 다운로드하는 데 사용할 수 있습니다. URL에 사용되는 자격 증명은 URL을 생성한 AWS 사용자의 자격 증명입니다. URL이 생성된 후에는 URL이 만료될 때까지 미리 지정된 URL을 가진 사람은 누구나 해당 객체를 다운로드할 수 있습니다.

S3 콘솔에서 미리 지정된 URL 사용

다음 단계에 따라 Amazon S3 콘솔을 사용하여 범용 버킷에서 객체를 공유하기 위해 미리 서명된 URL을 생성할 수 있습니다. 콘솔을 사용할 때 미리 지정된 URL의 최대 만료 시간은 생성 시점으로부터 12시간입니다.

Amazon S3 콘솔을 사용하여 미리 서명된 URL을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 미리 서명된 URL을 사용할 객체가 있는 버킷 이름을 선택합니다.
4. 객체(Objects) 목록에서 미리 서명된 URL을 생성할 객체를 선택합니다.
5. 객체 작업 메뉴에서 미리 서명된 URL 공유를 선택합니다.
6. 미리 서명된 URL의 유효 기간을 지정합니다.
7. 미리 서명된 URL 생성을 선택합니다.
8. 확인 메시지가 나타나면 URL이 클립보드에 자동으로 복사됩니다. 미리 서명된 URL을 다시 복사해야 하는 경우 미리 서명된 URL을 복사하는 버튼이 표시됩니다.
9. 다시 복사해야 하는 경우 미리 지정한 URL을 복사하는 버튼이 표시됩니다.

미리 지정된 URL 및 기타 생성 방법에 대한 자세한 내용은 미리 서명된 URL을 통해 객체 다운로드 및 업로드을 참조하십시오.

아카이브된 객체 다운로드

자주 액세스하지 않는 객체의 스토리지 비용을 줄이려면 해당 객체를 아카이브합니다. 객체를 아카이브하면 저비용 스토리지로 이동되므로 실시간으로 액세스할 수 없습니다. 아카이브된 객체를 다운로드하려면 먼저 복원해야 합니다.

스토리지 클래스에 따라 몇 분 또는 몇 시간 내에 아카이브된 객체를 복원할 수 있습니다. Amazon S3 콘솔, S3 배치 작업, Amazon S3 REST API, AWS SDK 및 AWS Command Line Interface(AWS CLI)를 사용하여 아카이브된 객체를 복원할 수 있습니다.

지침은 아카이브된 객체 복원 섹션을 참조하세요. 아카이브된 객체를 복원하고 나면 다운로드할 수 있습니다.

메타데이터를 기반으로 한 객체 다운로드

조건부 읽기 요청을 사용하여 메타데이터를 기반으로 객체를 다운로드하기 위한 사전 조건을 추가할 수 있습니다. 엔터티 태그(ETag) 또는 마지막 수정 날짜를 기준으로 객체를 반환할 수 있습니다. 이렇게 하면 S3 작업을 지정된 날짜 이후에 업데이트된 객체로 제한하거나 특정 객체 버전만 반환할 수 있습니다.

GetObject 또는 HeadObject 요청에 대해 조건부 쓰기를 사용할 수 있습니다.

조건부 요청에 대한 자세한 내용은 조건부 요청을 통해 S3 작업에 사전 조건 추가 섹션을 참조하세요.

객체 다운로드 문제 해결

권한이 충분하지 않거나 버킷 또는 AWS Identity and Access Management(IAM) 사용자 정책이 잘못되면 Amazon S3에서 객체를 다운로드하려고 할 때 오류가 발생할 수 있습니다. 이러한 문제로 인해 Amazon S3에서 리소스에 대한 액세스를 허용할 수 없는 액세스 거부(403 금지됨) 오류가 자주 발생할 수 있습니다.

액세스 거부(403 금지됨) 오류의 일반적인 원인에 대해서는 Amazon S3의 액세스 거부(403 Forbidden) 오류 문제 해결을 참조하십시오.

객체에 액세스하려고 할 때 404 NoSuchKey 오류가 발생하는 경우 AWS re:Post 지식 센터의 Amazon S3의 404 NoSuchKey 오류를 해결하려면 어떻게 해야 합니까?를 참조하세요.

Amazon S3에서 객체 무결성 확인

Amazon S3는 체크섬 값을 사용하여 업로드하거나 다운로드하는 데이터의 무결성을 확인합니다. 또한 Amazon S3에 저장하는 모든 객체에 대해 다른 체크섬 값을 계산하도록 요청할 수 있습니다. 데이터를 업로드, 복사 또는 일괄 복사할 때 사용할 체크섬 알고리즘을 선택할 수 있습니다.

데이터를 업로드하면 Amazon S3는 서버 측에서 체크섬을 계산하기 위해 선택한 알고리즘을 사용하고, 객체를 저장하고 체크섬을 객체 메타데이터의 일부로 저장하기 전에 제공된 값으로 검증합니다. 이 검증은 단일 파트 및 멀티파트 업로드 모두에 대해 암호화 모드, 객체 크기 및 스토리지 클래스에서 일관되게 작동합니다. 그러나 데이터를 복사하거나 일괄 복사하면 Amazon S3는 소스 객체의 체크섬을 계산하여 대상 객체로 이동합니다.

> **Note**
>
> 단일 파트 또는 멀티파트 업로드를 수행할 때 필요에 따라 사전 계산된 체크섬을 요청의 일부로 포함하고 전체 객체 체크섬 유형을 사용할 수 있습니다. 여러 객체에서 사전 계산된 값을 사용하려면 AWS CLI 또는 AWS SDK를 사용합니다.

지원되는 체크섬 알고리즘 사용

Amazon S3를 사용하면 업로드 중에 데이터를 검증할 체크섬 알고리즘을 선택할 수 있습니다. 그러면 지정된 체크섬 알고리즘이 객체와 함께 저장되며 다운로드 중에 데이터 무결성을 검증하는 데 사용할 수 있습니다. 다음 보안 해시 알고리즘(SHA) 또는 순환 중복 검사(CRC) 체크섬 알고리즘 중 하나를 선택해 체크섬 값을 계산할 수 있습니다.

- CRC-64/NVME(CRC64NVME)
- CRC-32(CRC32)
- CRC-32C(CRC32C)
- SHA-1(SHA1)
- SHA-256(SHA256)

또한 Content-MD5 헤더를 사용하여 각 요청에 체크섬을 제공할 수 있습니다.

객체를 업로드할 때 사용할 알고리즘을 지정합니다.

- AWS Management Console을 사용하는 경우 사용할 체크섬 알고리즘을 선택합니다. 필요에 따라 객체의 체크섬 값을 지정할 수 있습니다. Amazon S3가 객체를 수신하면 지정한 알고리즘을 사용하여 체크섬 값을 계산합니다. 두 체크섬 값이 일치하지 않으면 Amazon S3가 오류를 생성합니다.
- SDL를 사용할 경우 다음 사항에 유의해야 합니다.
  - `ChecksumAlgorithm` 파라미터를 Amazon S3에서 사용할 알고리즘으로 설정합니다. 사전 계산된 체크섬이 이미 있는 경우 체크섬 값을 AWS SDK에 전달하면 SDK가 요청에 값을 포함합니다. 체크섬 값을 전달하지 않거나 체크섬 알고리즘을 지정하지 않으면 SDK는 자동으로 체크섬 값을 계산하고 무결성 보호를 제공하기 위해 요청과 함께 체크섬 값을 포함합니다. 개별 체크섬 값이 체크섬 알고리즘의 설정 값과 일치하지 않는 경우 Amazon S3에서 `BadDigest` 오류와 함께 요청이 실패합니다.
  - 업그레이드된 AWS SDK를 사용하는 경우 SDK가 체크섬 알고리즘을 선택합니다. 그러나 이 체크섬 알고리즘을 재정의할 수 있습니다.
  - 체크섬 알고리즘을 지정하지 않고 SDK도 체크섬을 계산하지 않는 경우 S3는 CRC-64/NVME(`CRC64NVME`) 체크섬 알고리즘을 자동으로 선택합니다.
- REST API를 사용하는 경우 `x-amz-sdk-checksum-algorithm` 파라미터를 사용하면 안 됩니다. 그 대신 알고리즘별 헤더 중 하나를 사용합니다(예: `x-amz-checksum-crc32`).

Amazon S3에 이미 업로드된 객체에 이러한 체크섬 값을 적용하려면 객체를 복사하고 기존 체크섬 알고리즘을 사용할지, 새 체크섬 알고리즘을 사용할지 지정하면 됩니다. 알고리즘을 지정하지 않으면 S3는 기존 알고리즘을 사용합니다. 소스 객체에 지정된 체크섬 알고리즘 또는 체크섬 값이 없는 경우 Amazon S3는 CRC-64/NVME 알고리즘을 사용하여 대상 객체의 체크섬 값을 계산합니다. S3 배치 작업을 포함하여 객체를 복사할 때도 체크섬 알고리즘을 지정할 수 있습니다.

> **Important**
>
> 복합(또는 파트 수준) 체크섬에 대해 체크섬을 통해 멀티파트 업로드를 사용하는 경우 멀티파트 업로드 파트 번호는 연속적이어야 하고 1로 시작해야 합니다. 연속되지 않는 파트 번호로 멀티파트 업로드 요청을 완료하려고 하면 Amazon S3에서 `HTTP 500 Internal Server` 오류가 생성됩니다.

전체 객체 및 복합 체크섬 유형

Amazon S3에는 지원되는 체크섬 유형이 두 가지입니다.

- 전체 객체 체크섬: 전체 객체 체크섬은 멀티파트 업로드의 모든 콘텐츠를 기반으로 계산되며, 첫 번째 파트의 첫 번째 바이트부터 마지막 파트의 마지막 바이트까지 모든 데이터를 포함합니다.

> **Note**
>
> 모든 PUT 요청에는 전체 객체 체크섬 유형이 필요합니다.

- 복합 체크섬: 복합 체크섬은 멀티파트 업로드에 있는 각 파트의 개별 체크섬을 기반으로 계산됩니다. 이 접근 방식은 모든 데이터 콘텐츠를 기반으로 체크섬을 계산하는 대신 파트 수준 체크섬(첫 번째 파트부터 마지막 파트까지)을 집계하여 전체 객체에 대한 하나의 결합된 체크섬을 생성합니다.

> **Note**
>
> 객체가 멀티파트 업로드로 업로드되면 객체의 엔터티 태그(ETag)는 전체 객체의 MD5 다이제스트가 아닙니다. 대신, Amazon S3는 업로드될 때 각 개별 파트의 MD5 다이제스트를 계산합니다. MD5 다이제스트는 최종 객체의 ETag를 결정하는 데 사용됩니다. Amazon S3는 MD5 다이제스트에 대한 바이트를 함께 연결한 다음 이러한 연결된 값의 MD5 다이제스트를 계산합니다. ETag 생성의 마지막 단계에서 Amazon S3는 끝에 총 파트의 수와 함께 대시를 추가합니다.

Amazon S3는 다음과 같은 전체 객체 및 복합 체크섬 알고리즘 유형을 지원합니다.

- CRC-64/NVME(CRC64NVME): 전체 객체 알고리즘 유형만 지원합니다.
- CRC-32(CRC32): 전체 객체 및 복합 알고리즘 유형을 모두 지원합니다.
- CRC-32C(CRC32C): 전체 객체 및 복합 알고리즘 유형을 모두 지원합니다.
- SHA-1(SHA1): 전체 객체 및 복합 알고리즘 유형을 모두 지원합니다.
- SHA-256(SHA256): 전체 객체 및 복합 알고리즘 유형을 모두 지원합니다.

> **Note**
>
> 멀티파트 업로드에 SHA-1 또는 SHA-256 체크섬 알고리즘을 사용하는 경우 복합 알고리즘 유형을 사용해야 합니다. 단일 파트 업로드에 SHA-1 또는 SHA-256 체크섬 알고리즘을 사용하는 경우 전체 객체 알고리즘 유형만 지원됩니다.

단일 파트 업로드

단일 파트(PutObject 사용)로 업로드된 객체의 체크섬은 전체 객체 체크섬으로 처리됩니다. Amazon S3 콘솔에서 객체를 업로드할 때 S3에서 사용할 체크섬 알고리즘을 선택하고 원한다면 사전 계산된 값도 제공할 수 있습니다. 그러면 Amazon S3는 객체와 해당 체크섬 값을 저장하기 전에 이 체크섬을 검증합니다. 객체 다운로드 중에 체크섬 값을 요청할 때 객체의 데이터 무결성을 확인할 수 있습니다.

멀티파트 업로드

MultipartUpload API를 사용하여 객체를 여러 파트로 업로드할 때 Amazon S3에서 사용할 체크섬 알고리즘과 체크섬 유형(전체 객체 또는 복합)을 지정할 수 있습니다.

다음 표는 멀티파트 업로드에서 각 체크섬 알고리즘에 대해 지원되는 체크섬 알고리즘 유형을 나타냅니다.

체크섬 알고리즘	전체 객체	복합
CRC-64/NVME(CRC64NVME)	예	아니요
CRC-32(CRC32)	예	예
CRC-32C(CRC32C)	예	예
SHA-1(SHA1)	아니요	예
SHA-256(SHA256)	아니요	예

멀티파트 업로드에 전체 객체 체크섬 사용

멀티파트 업로드를 생성하거나 수행할 때 업로드 시 검증에 전체 객체 체크섬을 사용할 수 있습니다. 즉, 업로드된 객체에 대한 파트 경계를 더 이상 추적할 필요가 없으므로 MultipartUpload API에 대한 체크섬 알고리즘을 제공하여 무결성 검증 도구를 간소화할 수 있습니다. CompleteMultipartUpload 요청에서 전체 객체의 체크섬을 객체 크기와 함께 제공할 수 있습니다.

멀티파트 업로드 중에 전체 객체 체크섬을 제공하면 AWS SDK는 체크섬을 Amazon S3에 전달하고 S3는 서버 측의 객체 무결성을 검증하여 수신된 값과 비교합니다. 그런 다음 값이 일치하면 Amazon S3가 객체를 저장합니다. 두 값이 일치하지 않으면 S3에서 BadDigest 오류와 함께 요청이 실패합니다. 객체의 체크섬은 나중에 객체의 데이터 무결성을 검증하는 데 사용할 객체 메타데이터에도 저장됩니다.

전체 객체 체크섬의 경우 S3에서 CRC-64/NVME(CRC64NVME), CRC-32(CRC32) 또는 CRC-32C(CRC32C) 체크섬 알고리즘을 사용할 수 있습니다. 멀티파트 업로드의 전체 객체 체크섬은 전체 객체 체크섬으로 선형화할 수 있으므로 CRC 기반 체크섬에서만 사용할 수 있습니다. 이 선형화를 통해 Amazon S3는 성능 향상을 위한 요청을 병렬화할 수 있습니다. 특히 S3는 파트 수준 체크섬에서 전체 객체의 체크섬을 계산할 수 있습니다. 이 검증 유형은 SHA 및 MD5와 같은 다른 알고리즘에는 사용할 수 없습니다. S3에는 기본 무결성 보호 기능이 있으므로 객체가 체크섬 없이 업로드되면 S3는 권장되는 전체 객체 CRC-64/NVME(CRC64NVME) 체크섬 알고리즘을 객체에 자동으로 연결합니다.

> **Note**
>
> 멀티파트 업로드를 시작하려면 체크섬 알고리즘과 전체 객체 체크섬 유형을 지정하면 됩니다. 체크섬 알고리즘과 전체 객체 체크섬 유형을 지정한 후 멀티파트 업로드에 대한 전체 객체 체크섬 값을 제공할 수 있습니다.

멀티파트 업로드에 파트 수준의 체크섬 사용

객체를 Amazon S3에 업로드할 때 단일 객체로 업로드하거나 멀티파트 업로드 프로세스를 통해 여러 파트로 업로드할 수 있습니다. 멀티파트 업로드에 대한 체크섬 유형을 선택할 수 있습니다. 멀티파트 업로드 파트 수준 체크섬(또는 복합 체크섬)의 경우 Amazon S3는 지정된 체크섬 알고리즘을 사용하여 각 개별 파트의 체크섬을 계산합니다. `UploadPart`를 사용하여 각 파트의 체크섬 값을 제공할 수 있습니다. Amazon S3 콘솔에서 업로드하려는 객체가 CRC-64/NVME(CRC64NVME) 체크섬 알고리즘을 사용하도록 설정되어 있고 16MB를 초과하는 경우 전체 객체 체크섬으로 자동 지정됩니다.

그런 다음 Amazon S3는 저장된 파트 수준 체크섬 값을 사용하여 각 파트가 올바르게 업로드되었는지 확인합니다. 전체 객체에 대해 각 파트의 체크섬이 제공되면 S3는 각 파트의 저장된 체크섬 값을 사용하여 전체 객체 체크섬을 내부적으로 계산하여 제공된 체크섬 값과 비교합니다. 이 방법으로 S3가 파트의 체크섬을 사용하여 전체 객체의 체크섬을 계산할 수 있으므로 컴퓨팅 비용이 최소화됩니다. 멀티파트 업로드에 대한 자세한 내용은 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 및 멀티파트 업로드에 전체 객체 체크섬 사용 섹션을 참조하세요.

객체가 완전히 업로드되면 계산된 최종 체크섬을 사용하여 객체의 데이터 무결성을 확인할 수 있습니다.

멀티파트 업로드의 일부를 업로드할 때는 다음 사항에 유의하세요.

- 전체 객체를 구성하는 파트의 수를 포함하여 객체에 대한 정보를 검색하려면 `GetObjectAttributes` 작업을 사용하면 됩니다. 추가 체크섬을 사용하면 파트의 체크섬 값을 비롯한 각 개별 파트에 대한 정보를 복구할 수도 있습니다.

- 완료된 업로드의 경우, `GetObject` 또는 `HeadObject` 작업을 사용하고 단일 파트에 해당하는 파트 번호나 바이트 범위를 지정하여 개별 파트의 체크섬을 가져올 수도 있습니다. 아직 진행 중인 멀티파트 업로드의 개별 부분에 대한 체크섬 값을 검색하려면 `ListParts`를 사용하면 됩니다.
- Amazon S3가 멀티파트 객체에 대한 체크섬을 계산하는 방식 때문에 객체를 복사할 경우 객체의 체크섬 값이 변경될 수 있습니다. SDK 또는 REST API를 사용하고 있으며 `CopyObject`를 직접 호출하는 경우 Amazon S3는 `CopyObject` API 작업의 크기 한도에 도달할 때까지 객체를 복사합니다. Amazon S3는 객체가 단일 요청으로 업로드되었는지 또는 멀티파트 업로드의 일부로 업로드되었는지 여부에 관계없이 이 복사 작업을 단일 작업으로 수행합니다. 복사 명령을 사용하면 객체의 체크섬은 전체 객체의 직접 체크섬입니다. 객체가 원래 멀티파트 업로드를 사용하여 업로드된 경우 데이터가 변경되지 않아도 체크섬 값이 변경됩니다.
- `CopyObject` API 작업의 크기 한도보다 큰 객체는 멀티파트 업로드 복사 명령을 사용해야 합니다.
- AWS Management Console을 사용하여 일부 작업을 수행할 때 객체의 크기가 16MB를 초과하는 경우 Amazon S3는 멀티파트 업로드를 사용합니다.

체크섬 작업

객체를 업로드한 후 체크섬 값을 가져와 사전 계산된 체크섬 값 또는 동일한 알고리즘 유형의 이전에 저장된 체크섬 값과 비교할 수 있습니다. 다음 예시에서는 데이터 무결성을 확인하는 데 사용할 수 있는 체크섬 작업 또는 메서드를 보여줍니다.

S3 콘솔 사용

콘솔 사용 방법과 객체를 업로드할 때 사용할 체크섬 알고리즘을 지정하는 방법을 자세히 알아보려면 객체 업로드 및 자습서: 추가 체크섬을 사용하여 Amazon S3 내 데이터 무결성 확인을 참조하십시오.

AWS SDK 사용

다음 예시는 AWS SDK를 사용하여 멀티파트 업로드로 대량의 파일을 업로드 및 다운로드하고, 파일 검증에 SHA-256을 사용하여 멀티파트 업로드 파일을 검증하는 방법을 보여줍니다.

Java

Example 예제: SHA-256을 사용하여 대량의 파일 업로드, 다운로드 및 확인

실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 AWS SDK for Java 개발자 안내서의 시작하기 섹션을 참조하세요.

```
import software.amazon.awssdk.auth.credentials.AwsCredentials;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
```

```
    import software.amazon.awssdk.core.ResponseInputStream;
    import software.amazon.awssdk.core.sync.RequestBody;
    import software.amazon.awssdk.regions.Region;
    import software.amazon.awssdk.services.s3.S3Client;
    import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
    import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
    import software.amazon.awssdk.services.s3.model.ChecksumMode;
    import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
    import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadResponse;
    import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
    import software.amazon.awssdk.services.s3.model.CompletedPart;
    import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
    import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
    import software.amazon.awssdk.services.s3.model.GetObjectAttributesRequest;
    import software.amazon.awssdk.services.s3.model.GetObjectAttributesResponse;
    import software.amazon.awssdk.services.s3.model.GetObjectRequest;
    import software.amazon.awssdk.services.s3.model.GetObjectResponse;
    import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
    import software.amazon.awssdk.services.s3.model.ObjectAttributes;
    import software.amazon.awssdk.services.s3.model.PutObjectTaggingRequest;
    import software.amazon.awssdk.services.s3.model.Tag;
    import software.amazon.awssdk.services.s3.model.Tagging;
    import software.amazon.awssdk.services.s3.model.UploadPartRequest;
    import software.amazon.awssdk.services.s3.model.UploadPartResponse;

    import java.io.File;
    import java.io.FileInputStream;
    import java.io.FileOutputStream;
    import java.io.IOException;
    import java.io.InputStream;
    import java.io.OutputStream;
    import java.nio.ByteBuffer;
    import java.security.MessageDigest;
    import java.security.NoSuchAlgorithmException;
    import java.util.ArrayList;
    import java.util.Base64;
    import java.util.List;

    public class LargeObjectValidation {
        private static String FILE_NAME = "sample.file";
        private static String BUCKET = "sample-bucket";
        //Optional, if you want a method of storing the full multipart object
checksum in S3.
        private static String CHECKSUM_TAG_KEYNAME = "fullObjectChecksum";
```

```
        //If you have existing full-object checksums that you need to validate
 against, you can do the full object validation on a sequential upload.
        private static String SHA256_FILE_BYTES = "htCM5g7ZNdoSw8bN/
mkgiAhXt5MFoVowVg+LE9aIQmI=";
        //Example Chunk Size - this must be greater than or equal to 5MB.
        private static int CHUNK_SIZE = 5 * 1024 * 1024;

        public static void main(String[] args) {
            S3Client s3Client = S3Client.builder()
                    .region(Region.US_EAST_1)
                    .credentialsProvider(new AwsCredentialsProvider() {
                        @Override
                        public AwsCredentials resolveCredentials() {
                            return new AwsCredentials() {
                                @Override
                                public String accessKeyId() {
                                    return Constants.ACCESS_KEY;
                                }

                                @Override
                                public String secretAccessKey() {
                                    return Constants.SECRET;
                                }
                            };
                        }
                    })
                    .build();
            uploadLargeFileBracketedByChecksum(s3Client);
            downloadLargeFileBracketedByChecksum(s3Client);
            validateExistingFileAgainstS3Checksum(s3Client);
        }

        public static void uploadLargeFileBracketedByChecksum(S3Client s3Client) {
            System.out.println("Starting uploading file validation");
            File file = new File(FILE_NAME);
            try (InputStream in = new FileInputStream(file)) {
                MessageDigest sha256 = MessageDigest.getInstance("SHA-256");
                CreateMultipartUploadRequest createMultipartUploadRequest =
 CreateMultipartUploadRequest.builder()
                        .bucket(BUCKET)
                        .key(FILE_NAME)
                        .checksumAlgorithm(ChecksumAlgorithm.SHA256)
                        .build();
```

```
                CreateMultipartUploadResponse createdUpload =
s3Client.createMultipartUpload(createMultipartUploadRequest);
                List<CompletedPart> completedParts = new ArrayList<CompletedPart>();
                int partNumber = 1;
                byte[] buffer = new byte[CHUNK_SIZE];
                int read = in.read(buffer);
                while (read != -1) {
                    UploadPartRequest uploadPartRequest =
UploadPartRequest.builder()

.partNumber(partNumber).uploadId(createdUpload.uploadId()).key(FILE_NAME).bucket(BUCKET).ch
                    UploadPartResponse uploadedPart =
s3Client.uploadPart(uploadPartRequest,
RequestBody.fromByteBuffer(ByteBuffer.wrap(buffer, 0, read)));
                    CompletedPart part =
CompletedPart.builder().partNumber(partNumber).checksumSHA256(uploadedPart.checksumSHA256()
                    completedParts.add(part);
                    sha256.update(buffer, 0, read);
                    read = in.read(buffer);
                    partNumber++;
                }
                String fullObjectChecksum =
Base64.getEncoder().encodeToString(sha256.digest());
                if (!fullObjectChecksum.equals(SHA256_FILE_BYTES)) {
                    //Because the SHA256 is uploaded after the part is uploaded; the
upload is bracketed and the full object can be fully validated.

s3Client.abortMultipartUpload(AbortMultipartUploadRequest.builder().bucket(BUCKET).key(FILE
                    throw new IOException("Byte mismatch between stored checksum and
upload, do not proceed with upload and cleanup");
                }
                CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder().parts(completedParts).build();
                CompleteMultipartUploadResponse completedUploadResponse =
s3Client.completeMultipartUpload(

CompleteMultipartUploadRequest.builder().bucket(BUCKET).key(FILE_NAME).uploadId(createdUplo
                Tag checksumTag =
Tag.builder().key(CHECKSUM_TAG_KEYNAME).value(fullObjectChecksum).build();
                //Optionally, if you need the full object checksum stored with the
file; you could add it as a tag after completion.

s3Client.putObjectTagging(PutObjectTaggingRequest.builder().bucket(BUCKET).key(FILE_NAME).t
            } catch (IOException | NoSuchAlgorithmException e) {
```

```
                e.printStackTrace();
            }
            GetObjectAttributesResponse
                    objectAttributes =
s3Client.getObjectAttributes(GetObjectAttributesRequest.builder().bucket(BUCKET).key(FILE_N
                    .objectAttributes(ObjectAttributes.OBJECT_PARTS,
ObjectAttributes.CHECKSUM).build());
            System.out.println(objectAttributes.objectParts().parts());
            System.out.println(objectAttributes.checksum().checksumSHA256());
        }

        public static void downloadLargeFileBracketedByChecksum(S3Client s3Client) {
            System.out.println("Starting downloading file validation");
            File file = new File("DOWNLOADED_" + FILE_NAME);
            try (OutputStream out = new FileOutputStream(file)) {
                GetObjectAttributesResponse
                        objectAttributes =
s3Client.getObjectAttributes(GetObjectAttributesRequest.builder().bucket(BUCKET).key(FILE_N
                        .objectAttributes(ObjectAttributes.OBJECT_PARTS,
ObjectAttributes.CHECKSUM).build());
                //Optionally if you need the full object checksum, you can grab a
tag you added on the upload
                List<Tag> objectTags =
s3Client.getObjectTagging(GetObjectTaggingRequest.builder().bucket(BUCKET).key(FILE_NAME).b
                String fullObjectChecksum = null;
                for (Tag objectTag : objectTags) {
                    if (objectTag.key().equals(CHECKSUM_TAG_KEYNAME)) {
                        fullObjectChecksum = objectTag.value();
                        break;
                    }
                }
                MessageDigest sha256FullObject =
MessageDigest.getInstance("SHA-256");
                MessageDigest sha256ChecksumOfChecksums =
MessageDigest.getInstance("SHA-256");

                //If you retrieve the object in parts, and set the ChecksumMode to
enabled, the SDK will automatically validate the part checksum
                for (int partNumber = 1; partNumber <=
objectAttributes.objectParts().totalPartsCount(); partNumber++) {
                    MessageDigest sha256Part = MessageDigest.getInstance("SHA-256");
                    ResponseInputStream<GetObjectResponse> response =
s3Client.getObject(GetObjectRequest.builder().bucket(BUCKET).key(FILE_NAME).partNumber(part
                    GetObjectResponse getObjectResponse = response.response();
```

```
                    byte[] buffer = new byte[CHUNK_SIZE];
                    int read = response.read(buffer);
                    while (read != -1) {
                        out.write(buffer, 0, read);
                        sha256FullObject.update(buffer, 0, read);
                        sha256Part.update(buffer, 0, read);
                        read = response.read(buffer);
                    }
                    byte[] sha256PartBytes = sha256Part.digest();
                    sha256ChecksumOfChecksums.update(sha256PartBytes);
                    //Optionally, you can do an additional manual validation again
 the part checksum if needed in addition to the SDK check
                    String base64PartChecksum =
 Base64.getEncoder().encodeToString(sha256PartBytes);
                    String base64PartChecksumFromObjectAttributes =
 objectAttributes.objectParts().parts().get(partNumber - 1).checksumSHA256();
                    if (!
base64PartChecksum.equals(getObjectResponse.checksumSHA256()) || !
base64PartChecksum.equals(base64PartChecksumFromObjectAttributes)) {
                        throw new IOException("Part checksum didn't match for the
 part");
                    }
                    System.out.println(partNumber + " " + base64PartChecksum);
                }
                //Before finalizing, do the final checksum validation.
                String base64FullObject =
 Base64.getEncoder().encodeToString(sha256FullObject.digest());
                String base64ChecksumOfChecksums =
 Base64.getEncoder().encodeToString(sha256ChecksumOfChecksums.digest());
                if (fullObjectChecksum != null && !
fullObjectChecksum.equals(base64FullObject)) {
                    throw new IOException("Failed checksum validation for full
 object");
                }
                System.out.println(fullObjectChecksum);
                String base64ChecksumOfChecksumFromAttributes =
 objectAttributes.checksum().checksumSHA256();
                if (base64ChecksumOfChecksumFromAttributes != null && !
base64ChecksumOfChecksums.equals(base64ChecksumOfChecksumFromAttributes)) {
                    throw new IOException("Failed checksum validation for full
 object checksum of checksums");
                }
                System.out.println(base64ChecksumOfChecksumFromAttributes);
                out.flush();
```

```
            } catch (IOException | NoSuchAlgorithmException e) {
                //Cleanup bad file
                file.delete();
                e.printStackTrace();
            }
        }

        public static void validateExistingFileAgainstS3Checksum(S3Client s3Client)
{
            System.out.println("Starting existing file validation");
            File file = new File("DOWNLOADED_" + FILE_NAME);
            GetObjectAttributesResponse
                    objectAttributes =
s3Client.getObjectAttributes(GetObjectAttributesRequest.builder().bucket(BUCKET).key(FILE_N
                    .objectAttributes(ObjectAttributes.OBJECT_PARTS,
ObjectAttributes.CHECKSUM).build());
            try (InputStream in = new FileInputStream(file)) {
                MessageDigest sha256ChecksumOfChecksums =
MessageDigest.getInstance("SHA-256");
                MessageDigest sha256Part = MessageDigest.getInstance("SHA-256");
                byte[] buffer = new byte[CHUNK_SIZE];
                int currentPart = 0;
                int partBreak =
objectAttributes.objectParts().parts().get(currentPart).size();
                int totalRead = 0;
                int read = in.read(buffer);
                while (read != -1) {
                    totalRead += read;
                    if (totalRead >= partBreak) {
                        int difference = totalRead - partBreak;
                        byte[] partChecksum;
                        if (totalRead != partBreak) {
                            sha256Part.update(buffer, 0, read - difference);
                            partChecksum = sha256Part.digest();
                            sha256ChecksumOfChecksums.update(partChecksum);
                            sha256Part.reset();
                            sha256Part.update(buffer, read - difference,
difference);
                        } else {
                            sha256Part.update(buffer, 0, read);
                            partChecksum = sha256Part.digest();
                            sha256ChecksumOfChecksums.update(partChecksum);
                            sha256Part.reset();
                        }
```

```
                        String base64PartChecksum =
 Base64.getEncoder().encodeToString(partChecksum);
                        if (!
base64PartChecksum.equals(objectAttributes.objectParts().parts().get(currentPart).checksumSH
 {
                            throw new IOException("Part checksum didn't match S3");
                        }
                        currentPart++;
                        System.out.println(currentPart + " " + base64PartChecksum);
                        if (currentPart <
 objectAttributes.objectParts().totalPartsCount()) {
                            partBreak +=
 objectAttributes.objectParts().parts().get(currentPart - 1).size();
                        }
                    } else {
                        sha256Part.update(buffer, 0, read);
                    }
                    read = in.read(buffer);
                }
                if (currentPart != objectAttributes.objectParts().totalPartsCount())
 {
                    currentPart++;
                    byte[] partChecksum = sha256Part.digest();
                    sha256ChecksumOfChecksums.update(partChecksum);
                    String base64PartChecksum =
 Base64.getEncoder().encodeToString(partChecksum);
                    System.out.println(currentPart + " " + base64PartChecksum);
                }

                String base64CalculatedChecksumOfChecksums =
 Base64.getEncoder().encodeToString(sha256ChecksumOfChecksums.digest());
                System.out.println(base64CalculatedChecksumOfChecksums);
                System.out.println(objectAttributes.checksum().checksumSHA256());
                if (!
base64CalculatedChecksumOfChecksums.equals(objectAttributes.checksum().checksumSHA256()))
 {
                    throw new IOException("Full object checksum of checksums don't
 match S3");
                }

            } catch (IOException | NoSuchAlgorithmException e) {
                e.printStackTrace();
            }
        }
```

```
    }
```

REST API 사용

REST 요청을 전송하고 체크섬 값을 사용하여 객체를 업로드하여 PutObject로 데이터의 무결성을 확인할 수 있습니다. GetObject 또는 HeadObject를 사용하여 객체의 체크섬 값을 검색할 수도 있습니다.

AWS CLI 사용

PUT 요청을 보내 한 번의 작업으로 최대 5GB의 객체를 업로드할 수 있습니다. 자세한 내용은 AWS CLI 명령 참조의 `put-object`를 참조하십시오. `get-object` 및 `head-object`를 사용하여 이미 업로드된 객체의 체크섬을 검색함으로써 데이터 무결성을 확인할 수도 있습니다.

자세한 내용은 AWS Command Line Interface 사용 설명서의 Amazon S3 CLI FAQ를 참조하세요.

객체를 업로드할 때 Content-MD5 사용

업로드 후 객체의 무결성을 확인하는 또 다른 방법은 객체를 업로드할 때 객체의 MD5 다이제스트를 제공하는 것입니다. 객체에 대한 MD5 다이제스트를 계산하면 `Content-MD5` 헤더를 사용하여 PUT 명령으로 다이제스트를 제공할 수 있습니다.

객체를 업로드한 후 Amazon S3는 객체의 MD5 다이제스트를 계산하여 사용자가 제공한 값과 비교합니다. 두 다이제스트가 일치하는 경우에만 요청이 성공합니다.

MD5 다이제스트를 제공할 필요는 없지만 업로드 프로세스의 일부로 객체의 무결성을 확인하는 데 사용할 수 있습니다.

Content-MD5 및 ETag를 사용하여 업로드된 객체 확인

객체의 엔터티 태그(ETag)는 해당 객체의 특정 버전을 나타냅니다. ETag는 객체의 콘텐츠에 대한 변경 사항만 반영하고 메타데이터에 대한 변경 사항은 반영하지 않는다는 점을 명심하세요. 객체의 메타데이터만 변경되는 경우 ETag는 동일하게 유지됩니다.

객체에 따라 객체의 ETag가 객체 데이터의 MD5 다이제스트일 수 있습니다.

- 객체가 `PutObject`, `PostObject` 또는 `CopyObject` 작업으로, 또는 AWS Management Console을 통해 생성되며 해당 객체가 일반 텍스트 또는 Amazon S3 관리형 키(SSE-S3)를 통한 서버 측 암호화를 사용하여 암호화되는 경우 해당 객체에는 객체 데이터의 MD5 다이제스트인 ETag가 있습니다.

- 객체가 PutObject, PostObject 또는 CopyObject 작업으로, 또는 AWS Management Console을 통해 생성되며 해당 객체가 고객 제공 키를 통한 서버 측 암호화(SSE-C) 또는 AWS Key Management Service(AWS KMS) 키를 통한 서버 측 암호화(SSE-KMS)를 사용하여 암호화되는 경우 해당 객체에는 객체 데이터의 MD5 다이제스트가 아닌 ETag가 있습니다.
- 멀티파트 업로드 프로세스 또는 UploadPartCopy 작업을 통해 객체가 생성되는 경우 암호화 방법에 관계없이 객체의 ETag는 MD5 다이제스트가 아닙니다. 객체가 16MB보다 큰 경우 AWS Management Console은 해당 객체를 멀티파트 업로드로 업로드하거나 복사하므로 ETag가 MD5 다이제스트가 아닙니다.

ETag가 객체의 Content-MD5 다이제스트인 객체의 경우 해당 객체의 ETag 값을 계산된 또는 이전에 저장된 Content-MD5 다이제스트와 비교할 수 있습니다.

후행 체크섬 사용

Amazon S3에 객체를 업로드할 때 객체에 대해 사전 계산된 체크섬을 제공하거나 AWS SDK를 사용하여 사용자를 대신하여 청크 업로드에 대해 후행 체크섬을 자동으로 생성하도록 할 수 있습니다. 후행 체크섬을 사용하는 경우 Amazon S3는 지정된 알고리즘을 사용해 자동으로 체크섬을 생성하여 업로드 시 청크 업로드에 속한 객체의 무결성을 검증합니다.

AWS SDK를 사용할 때 후행 체크섬을 생성하려면 ChecksumAlgorithm 파라미터를 원하는 알고리즘으로 채웁니다. SDK는 해당 알고리즘을 사용하여 객체(또는 객체 파트)에 대한 체크섬을 계산하고 청크 업로드 요청 끝에 자동으로 추가합니다. 이 동작은 Amazon S3가 단일 패스로 데이터의 확인 및 업로드를 모두 수행하므로 시간을 절약할 수 있습니다.

> **Important**
>
> S3 객체 Lambda를 사용하는 경우 S3 객체 Lambda에 대한 모든 요청은 s3 대신 s3-object-lambda를 사용하여 서명됩니다. 이 동작은 후행 체크섬 값의 서명에 영향을 줍니다. S3 객체 Lambda에 대한 자세한 내용은 S3 객체 Lambda를 사용하여 객체 변환 섹션을 참조하십시오.

후행 체크섬 헤더

청크 콘텐츠 인코딩 요청을 수행하려면 Amazon S3에서 요청을 올바르게 구문 분석하기 위해 클라이언트에 몇 가지 헤더를 포함할 것을 요구합니다. 클라이언트에 다음 헤더를 포함해야 합니다.

- **x-amz-decoded-content-length**: 이 헤더는 요청과 함께 Amazon S3에 업로드되는 실제 데이터의 일반 텍스트 크기를 나타냅니다.

- **`x-amz-content-sha256`**: 이 헤더는 요청에 포함된 청크 업로드 유형을 나타냅니다. 후행 체크섬이 있는 청크 업로드의 경우 헤더 값은 페이로드 서명을 사용하지 않는 요청의 경우 `STREAMING-UNSIGNED-PAYLOAD-TRAILER`이고 SigV4 페이로드 서명을 사용하는 요청의 경우 `STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER`입니다. (서명된 페이로드를 구현하는 방법에 대한 자세한 내용은 Signature calculations for the authorization header: Transferring a payload in multiple chunks를 참조하세요.)
- **`x-amz-trailer`**: 이 헤더는 요청에 있는 후행 헤더의 이름을 나타냅니다. 후행 체크섬이 있는 경우(AWS SDK가 인코딩된 요청 본문에 체크섬을 추가함) `x-amz-trailer` 헤더 값에 `x-amz-checksum-` 접두사가 포함되고 알고리즘 이름으로 끝납니다. 현재 다음 `x-amz-trailer` 값이 지원됩니다.
  - `x-amz-checksum-crc32`
  - `x-amz-checksum-crc32c`
  - `x-amz-checksum-crc64nvme`
  - `x-amz-checksum-sha1`
  - `x-amz-checksum-sha256`

> **Note**
>
> 요청에 청크 값과 함께 `Content-Encoding` 헤더를 포함할 수도 있습니다. 이 헤더는 필수는 아니지만 이 헤더를 포함하면 인코딩된 데이터를 전송할 때 HTTP 프록시 문제를 최소화할 수 있습니다. 요청에 다른 `Content-Encoding` 헤더(예: gzip)가 있는 경우 `Content-Encoding` 헤더는 쉼표로 구분된 인코딩 목록에 청크 값을 포함합니다. 예를 들어 `Content-Encoding: aws-chunked, gzip`입니다.

청크 파트

청크 인코딩을 사용하여 Amazon S3에 객체를 업로드하는 경우 업로드 요청에는 다음 유형의 청크(목록에 나열된 순서대로 형식이 지정됨)가 포함됩니다.

- 객체 본문 청크: 청크 업로드 요청과 연결된 본문 청크가 하나, 여러 개 또는 0개 있을 수 있습니다.
- 완료 청크: 청크 업로드 요청과 연결된 본문 청크가 하나, 여러 개 또는 0개 있을 수 있습니다.
- 후행 청크: 후행 체크섬은 완료 청크 뒤에 나열됩니다. 후행 청크는 하나만 허용됩니다.

> **Note**
>
> 모든 청크 업로드는 요청의 종료를 나타내기 위해 최종 CRLF(예: \r\n)로 끝나야 합니다.

청크 형식 지정의 예는 예: 후행 체크섬이 있는 청크 업로드 섹션을 참조하세요.

객체 본문 청크

객체 본문 청크는 S3에 업로드되는 실제 객체 데이터가 포함된 청크입니다. 이러한 청크에는 일관된 크기 및 형식 제약 조건이 있습니다.

객체 본문 청크 크기

이러한 청크에는 최소 8,192바이트(즉, 8KiB)의 객체 데이터가 포함되어야 합니다. 단, 최종 본문 청크는 더 작을 수 있습니다. 명시적인 최대 청크 크기는 없지만 모든 청크가 5GB의 최대 업로드 크기보다 작을 것으로 예상할 수 있습니다. 청크 크기는 클라이언트 서버 구현에 따라 청크마다 다를 수 있습니다.

객체 본문 청크 형식

객체 본문 청크는 객체 본문 청크의 바이트 수에 대한 16진수 인코딩으로 시작되며, 그 뒤에 CRLF(캐리지 리턴 라인 피드), 해당 청크의 객체 바이트 및 다른 CRLF가 옵니다.

예시:

```
hex-encoding-of-object-bytes-in-chunk\r\n
chunk-object-bytes\r\n
```

그러나 청크에 서명하면 객체 본문 청크는 다른 형식을 따르며, 여기서 서명은 세미콜론 구분 기호를 사용하여 청크 크기에 추가됩니다. 예시:

```
hex-encoding-of-object-bytes-in-chunk;chunk-signature\r\n
 chunk-object-bytes\r\n
```

청크 서명에 대한 자세한 내용은 Signature calculations for the Authorization Header: Transferring a payload in multiple chunks (AWS Signature Version 4)를 참조하세요. 청크 형식 지정에 대한 자세한 내용은 RFC Editor 웹 사이트의 Chunked transfer encoding을 참조하세요.

완료 청크

완료 청크는 모든 청크 업로드의 최종 객체 본문 청크여야 합니다. 완료 청크의 형식은 본문 청크와 비슷하지만 항상 0바이트의 객체 데이터를 포함합니다. (객체 데이터의 0바이트는 모든 데이터가 업로드되었음을 나타냅니다.) 청크 업로드에는 다음과 같은 형식을 따라 최종 객체 본문 청크로 완료 청크가 포함되어야 합니다.

```
0\r\n
```

그러나 콘텐츠 인코딩 요청이 페이로드 서명을 사용하는 경우 대신 다음 형식을 따릅니다.

```
0;chunk-signature\r\n
```

트레일러 청크

트레일러 청크는 모든 S3 업로드 요청에 대해 계산된 체크섬을 보유합니다. 트레일러 청크에는 헤더 이름 필드 하나와 헤더 값 필드 하나가 포함됩니다. 업로드 요청의 헤더 이름 필드는 `x-amz-trailer` 요청 헤더에 전달된 값과 일치해야 합니다. 예를 들어 요청에 `x-amz-trailer: x-amz-checksum-crc32`가 포함되어 있고 트레일러 청크의 헤더 이름이 `x-amz-checksum-sha1`인 경우 요청이 실패합니다. 트레일러 청크의 값 필드에는 해당 객체에 대한 빅엔디언 체크섬 값의 base64 인코딩이 포함됩니다. (빅엔디언 순서는 가장 중요한 바이트의 데이터를 가장 낮은 메모리 주소에 저장하고 가장 덜 중요한 바이트를 가장 큰 메모리 주소에 저장합니다.) 이 체크섬을 계산하는 데 사용되는 알고리즘은 헤더 이름의 접미사(예: `crc32`)와 동일합니다.

트레일러 청크 형식

트레일러 청크는 서명되지 않은 페이로드 요청에 대해 다음 형식을 사용합니다.

```
x-amz-checksum-lowercase-checksum-algorithm-name:base64-checksum-value\n\r\n\r\n
```

SigV4 서명 페이로드가 있는 요청의 경우 트레일러 청크에는 트레일러 청크 뒤에 트레일러 서명이 포함됩니다.

```
trailer-checksum\n\r\n
trailer-signature\r\n
```

base64 체크섬 값의 끝에 CRLF를 직접 추가할 수도 있습니다. 예시:

```
x-amz-checksum-lowercase-checksum-algorithm-name:base64-checksum-value\r\n\r\n
```

예: 후행 체크섬이 있는 청크 업로드

Amazon S3는 PutObject에 aws-chunked 콘텐츠 인코딩을 사용하는 청크 업로드와 후행 체크섬이 있는 UploadPart 요청을 지원합니다.

Example 1 - 후행 CRC-32 체크섬이 있는 서명되지 않은 청크 **PutObject** 요청

다음은 후행 CRC-32 체크섬이 있는 청크 PutObject 요청의 예입니다. 이 예시에서 클라이언트는 서명되지 않은 세 개의 청크로 17KB 객체를 업로드하고 x-amz-checksum-crc32 헤더를 사용하여 후행 CRC-32 체크섬 청크를 추가합니다.

```
PUT /Key+ HTTP/1.1
Host: amzn-s3-demo-bucket
Content-Encoding: aws-chunked
x-amz-decoded-content-length: 17408
x-amz-content-sha256: STREAMING-UNSIGNED-PAYLOAD-TRAILER
x-amz-trailer: x-amz-checksum-crc32

2000\r\n                                    // Object body chunk 1 (8192 bytes)
object-bytes\r\n
2000\r\n                                    // Object body chunk 2 (8192 bytes)
object-bytes\r\n
400\r\n                                     // Object body chunk 3 (1024 bytes)
object-bytes\r\n
0\r\n                                       // Completion chunk
x-amz-checksum-crc32:YABb/g==\n\r\n\r\n     // Trailer chunk (note optional \n
 character)
\r\n                                          // CRLF
```

다음은 응답의 예시입니다.

```
HTTP/1.1 200
ETag: ETag
x-amz-checksum-crc32: YABb/g==
```

> **Note**
>
> 체크섬 값 끝에 있는 라인 피드 \n 사용 여부는 클라이언트마다 다를 수 있습니다.

Example 2 - 후행 CRC-32(**CRC32**) 체크섬이 있는 SigV4 서명 청크 **PutObject** 요청

다음은 후행 CRC-32 체크섬이 있는 청크 PutObject 요청의 예입니다. 이 요청은 SigV4 페이로드 서명을 사용합니다. 이 예시에서 클라이언트는 17KB 객체를 세 개의 서명된 청크로 업로드합니다. object body 청크 외에 completion chunk 및 trailer chunk 도 서명됩니다.

```
PUT /Key+ HTTP/1.1
Host: amzn-s3-demo-bucket.s3.amazonaws.com
Content-Encoding: aws-chunked
x-amz-decoded-content-length: 17408
x-amz-content-sha256: STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER
x-amz-trailer: x-amz-checksum-crc32

authorization-code                              // SigV4 headers authorization

2000;chunk-signature=signature-value...\r\n   // Object body chunk 1 (8192 bytes)
object-bytes\r\n
2000;chunk-signature\r\n                        // Object body chunk 2 (8192 bytes)
object-bytes\r\n
400;chunk-signature\r\n                         // Object body chunk 3 (1024 bytes)
object-bytes\r\n
0;chunk-signature\r\n                           // Completion chunk
x-amz-checksum-crc32:YABb/g==\n\r\n             // Trailer chunk (note optional \n
 character)
trailer-signature\r\n
\r\n                                            // CRLF
```

다음은 응답의 예시입니다.

```
HTTP/1.1 200
ETag: ETag
x-amz-checksum-crc32: YABb/g==
```

Amazon S3 객체 삭제

Amazon S3 콘솔, AWS SDK, AWS Command Line Interface(AWS CLI) 또는 REST API를 사용하여 Amazon S3에서 직접 하나 이상의 객체를 삭제할 수 있습니다. 예를 들어 로그 파일을 수집하는 경우, 더 이상 필요가 없는 로그 파일은 삭제하는 것이 좋습니다. 로그 파일과 같은 객체를 자동으로 삭제하도록 S3 수명 주기 규칙을 설정할 수 있습니다.

객체를 삭제하려면 다음 API 작업 중 하나를 사용하면 됩니다.

- 단일 객체 삭제 – Amazon S3는 단일 HTTP 요청으로 하나의 객체를 삭제할 수 있는 DELETE(DeleteObject) API를 제공합니다.
- 여러 객체 삭제 – Amazon S3는 단일 HTTP 요청에서 최대 1,000개의 객체를 삭제하는 데 사용할 수 있는 Multi-Object Delete(DeleteObjects) API를 제공합니다.

버전 관리가 활성화되지 않은 버킷에서 객체를 삭제할 때는 객체 키 이름만 제공합니다. 그러나 버전 관리가 활성화된 버킷에서 객체를 삭제할 때는 객체의 버전 ID를 제공하여 특정 버전의 객체를 삭제할 수 있습니다.

객체를 삭제하기 전에 고려해야 할 모범 사례

객체를 삭제하기 전에 다음 모범 사례를 고려하세요.

- 버킷 버전 관리를 활성화합니다. S3 버전 관리는 단순 DeleteObject 요청에 대한 보호 기능을 추가하여 실수로 인한 삭제를 방지합니다. 버전이 지정된 버킷의 경우, 객체의 현재 버전을 삭제하거나 삭제 요청에 특정 버전 ID가 지정되지 않았다면 Amazon S3는 객체를 영구적으로 삭제하지 않습니다. 대신 S3가 삭제 마커를 추가하여 객체를 소프트 삭제합니다. 그러면 삭제 마커는 새 버전 ID를 가진 객체의 현재(또는 최신) 버전이 됩니다. 자세한 내용은 버전 관리가 사용 설정된 버킷에서 객체 버전 삭제를 참조하세요.
- 다수의 객체를 삭제하거나 객체를 만든 날짜를 기준으로 프로그래밍 방식으로 객체를 삭제하려면 버킷에 S3 수명 주기 구성을 설정하세요. 이러한 삭제를 모니터링하려면 S3 수명 주기 이벤트 알림을 사용하는 것이 좋습니다. S3 수명 주기 알림을 구성하면 버킷의 객체가 삭제될 때 s3:LifecycleExpiration:Delete 이벤트 유형이 알려줍니다. S3 수명 주기 구성에 의해 객체 버전이 영구적으로 삭제될 때도 알려줍니다. s3:LifecycleExpiration:DeleteMarkerCreated 이벤트 유형은 S3 수명 주기에서 삭제 마커를 생성할 때 알려줍니다. 삭제 마커는 버전이 지정된 버킷에 있는 객체의 현재 버전이 삭제될 때 생성됩니다.
- S3 수명 주기 구성을 업데이트하기 전에 수명 주기가 의도한 모든 객체에 대한 작업을 완료했는지 확인하세요. 자세한 내용은 버킷에서 S3 수명 주기 구성 설정의 수명 주기 규칙 업데이트, 사용 중지 또는 삭제 섹션을 참조하세요.

> **Note**
>
> 의도하지 않은 삭제를 방지하려면 S3 수명 주기 규칙을 올바른 객체 하위 집합에 적용해야 합니다. 수명 주기 규칙을 만들 때 접두사, 객체 태그 또는 객체 크기별로 객체를 필터링할 수 있습니다.

- 사용자가 버킷에서 객체를 제거하거나 삭제하는 것을 제한해 보세요. 사용자를 제한하려면 Amazon S3 버킷 정책에서 다음 작업에 대한 사용자 권한을 명시적으로 거부해야 합니다.
  - `s3:DeleteObject`, `s3:DeleteObjectVersion`(API 요청을 사용하여 객체를 삭제할 수 있는 사람을 제어하기 위한 목적)
  - `s3:PutLifecycleConfiguration`(S3 수명 주기 만료 규칙을 추가할 수 있는 사람을 제어하기 위한 목적)
- S3 복제를 사용하여 여러 데이터 사본을 생성하고 이를 다양한 위치에 한 번에 복제하는 것을 고려해 보세요. 필요한 만큼 대상 버킷을 선택할 수 있습니다. 또한, 객체가 실수로 삭제되더라도 데이터 사본은 계속 남아 있습니다.

버전 관리가 활성화된 버킷에서 객체 삭제

버전 관리를 사용하는 버킷에서는 동일한 객체에 여러 버전이 존재할 수 있습니다. 버전 관리가 활성화된 버킷에서 작업할 때 `Delete` API 작업은 다음과 같은 옵션을 제공합니다.

- 버전이 지정되지 않은 삭제 요청 지정 - 버전 ID가 아닌 객체의 키만 지정합니다. 이 경우 Amazon S3는 현재 버전의 객체에 대해 삭제 마커를 생성하고 응답에서 버전 ID를 반환합니다. 그 결과 객체가 버킷에서 보이지 않습니다. 객체 버전 관리 및 삭제 마커 개념에 대한 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 단원을 참조하십시오.
- 버전이 지정된 삭제 요청 지정 - 키와 버전 ID를 모두 지정합니다. 이 경우 다음과 같은 결과가 나올 수 있습니다.
  - 버전 ID가 특정 객체 버전과 일치할 경우 Amazon S3는 해당 특정 버전의 객체를 삭제합니다.
  - 버전 ID가 객체의 삭제 마커와 일치할 경우 Amazon S3는 삭제 마커를 삭제합니다. 삭제 마커가 삭제되면 객체가 버킷에 다시 나타납니다.

버전 관리가 일시 중지된 버킷에서 객체 삭제

버킷의 버전 관리가 일시 중단된 경우, `Delete` API 작업은 버전 관리가 활성화된 버킷에 대해 동일한 방식으로 작동합니다(현재 버전의 버전 ID가 null인 경우는 제외). 자세한 내용은 버전 관리가 일시 중지된 버킷에서 객체 삭제 섹션을 참조하세요.

버전이 지정되지 않은 버킷에서 객체 삭제

버킷에 버전이 지정되지 않은 경우 `Delete` API 작업에서 객체 키를 지정할 수 있으며 Amazon S3가 객체를 영구적으로 삭제할 수 있습니다. 객체의 영구 삭제를 방지하려면 버킷 버전 관리를 활성화하세요.

MFA를 사용하는 버킷에서 객체 삭제

멀티 팩터 인증(MFA)을 사용하는 버킷에서 객체를 삭제할 경우 다음과 같은 사항에 주의합니다.

- 유효하지 않은 MFA 토큰을 제공할 경우 요청은 항상 실패합니다.
- MFA가 사용된 버킷에서 버전을 지정한 삭제를 요청(객체 키 및 버전 ID 제공)할 경우 MFA 토큰이 올바르지 않으면 요청에 실패합니다. 또한, MFA가 활성화된 버킷에서 다중 객체 `Delete` API 작업을 사용할 때 버전이 지정된 삭제 요청(객체 키 및 버전 ID 제공) 시 MFA 토큰을 제공하지 않으면 요청이 모두 실패로 끝납니다.

그러나 다음과 같은 경우에는 요청이 성공합니다.

- MFA가 활성화된 버킷에서 버전이 지정되지 않은 삭제를 요청하고(버전이 지정된 객체를 삭제하지 않음) MFA 토큰을 제공하지 않으면 삭제가 성공합니다.
- Multi-Object Delete 요청 시 MFA가 활성화된 버킷에서 버전이 지정되지 않은 객체들만 삭제하도록 지정하고 MFA 토큰을 제공하지 않으면 삭제가 성공합니다.

MFA 삭제에 대한 자세한 내용은 MFA Delete 구성 섹션을 참조하십시오.

주제

- 단일 객체 삭제
- 여러 객체 삭제

단일 객체 삭제

Amazon S3 콘솔 또는 DELETE API를 사용하여 S3 버킷에서 단일의 기존 객체를 삭제할 수 있습니다. Amazon S3에서 객체 삭제에 대한 자세한 내용은 Amazon S3 객체 삭제 단원을 참조하십시오.

S3 버킷에 있는 모든 객체에 스토리지 비용이 발생하기 때문에 더 이상 필요하지 않은 객체는 삭제해야 합니다. 예를 들어 로그 파일을 수집하는 경우, 더 이상 필요가 없는 로그 파일은 삭제하는 것이 좋습니다. 로그 파일과 같은 객체를 자동으로 삭제하도록 수명 주기 규칙을 설정할 수 있습니다. 자세한 내용은 the section called “수명 주기 구성 설정” 섹션을 참조하세요.

Amazon S3 기능 및 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하세요.

S3 콘솔 사용

다음 단계에 따라 Amazon S3 콘솔을 사용하여 버킷에서 단일 객체를 삭제합니다.

> **Warning**
>
> Amazon S3 콘솔에서 객체 또는 지정된 객체 버전을 영구 삭제하는 경우 이를 실행 취소할 수 없습니다.

버전 관리가 활성화되거나 일시 중단된 객체를 삭제하려면

> **Note**
>
> 버전 관리가 일시 중단된 버킷에 있는 객체의 버전 ID가 NULL로 표시되는 경우에는 이전 버전이 없으므로 S3는 해당 객체를 영구 삭제합니다. 그러나 버전 관리가 일시 중단된 버킷의 해당 객체에 대한 유효한 버전 ID가 있는 경우 S3는 삭제된 객체에 대한 삭제 마커를 생성하고 객체의 이전 버전을 그대로 유지합니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷 또는 디렉터리 버킷을 선택합니다.
3. 버킷 목록에서 객체를 삭제하려는 버킷 이름을 선택합니다.
4. 객체를 선택한 다음 삭제를 선택합니다.
5. 개체를 삭제할까요? 텍스트 상자의 지정된 객체 아래에 있는 객체 목록의 삭제를 확인하려면 **`delete`**를 입력합니다.

버전 관리가 활성화된 버킷에서 특정 객체 버전을 영구적으로 삭제하려면

> **Warning**
>
> Amazon S3에서 지정된 객체 버전을 영구 삭제하는 경우 이를 실행 취소할 수 없습니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 버킷 이름 목록에서 객체를 삭제하려는 버킷 이름을 선택합니다.
3. 삭제할 객체를 선택합니다.

4. 버전 표시 토글을 선택합니다.
5. 객체 버전을 선택한 다음 삭제를 선택합니다.
6. 객체를 삭제할까요? 텍스트 상자의 지정된 객체 아래에 있는 특정 객체 버전의 영구 삭제를 확인하려면 영구 삭제를 입력합니다. Amazon S3가 특정 객체 버전을 영구적으로 삭제합니다.

버전 관리가 활성화되지 않은 Amazon S3 버킷의 객체를 영구적으로 삭제하려면

> **Warning**
>
> Amazon S3에서 객체를 영구 삭제하는 경우 이를 실행 취소할 수 없습니다. 또한 버전 관리가 활성화되지 않은 버킷의 경우 삭제는 영구적입니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 객체를 삭제하려는 버킷 이름을 선택합니다.
4. 객체를 선택한 다음 삭제를 선택합니다.
5. 객체를 삭제할까요? 텍스트 상자의 지정된 객체 아래에 있는 객체의 영구 삭제를 확인하려면 영구 삭제를 입력합니다.

> **Note**
>
> 객체 삭제와 관련하여 문제가 발생하는 경우 버전이 지정된 객체를 영구적으로 삭제하고 싶습니다. 섹션을 참조하세요.

AWS CLI 사용

요청당 하나의 객체를 삭제하려면 DELETE API를 사용합니다. 자세한 내용은 DELETE Object(객체 삭제) 단원을 참조하십시오. CLI를 사용하여 객체를 삭제하는 방법에 대한 자세한 내용은 delete-object를 참조하십시오.

REST API 사용

AWS SDK를 사용하여 객체를 삭제할 수 있습니다. 하지만 애플리케이션에서 요구할 경우 REST 요청을 직접 전송할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조의 DELETE Object를 참조하십시오.

AWS SDK 사용

다음 예제에서는 AWS SDK를 사용하여 버킷에서 객체를 삭제하는 방법을 보여줍니다. 자세한 내용은 Amazon Simple Storage Service API 참조의 DELETE Object를 참조하십시오.

S3 버전 관리를 사용하는 버킷의 경우 다음 옵션을 사용할 수 있습니다.

- 버전 ID를 지정하여 특정 객체 버전을 삭제합니다.
- 버전 ID를 지정하지 않고 객체를 삭제합니다. 이 경우 Amazon S3에서는 객체에 삭제 마커를 추가합니다.

S3 버전 관리에 대한 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하십시오.

더 많은 예제와 다른 언어의 예제는 Amazon S3 API 참조의 AWS SDK 또는 CLI와 함께 `DeleteObject` 사용을 참조하세요.

Java

Example 예제 1: 객체 삭제(버전이 지정되지 않은 버킷)

다음 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 AWS SDK for Java 개발자 안내서의 시작하기 섹션을 참조하세요.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
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. Amazon S3에서는 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 S3에서는 그 객체를 완전히 삭제합니다. 그렇지 않은 경우 Amazon S3에서는 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();
        }
    }
}
```

.NET

다음 예제는 버전이 지정된 버킷과 버전이 지정되지 않는 버킷 둘 다에서 객체를 삭제하는 방법을 보여줍니다. S3 버전 관리에 대한 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하십시오.

Example 버전이 지정되지 않은 버킷에서 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.

코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

```
using Amazon;
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 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 사용 설정합니다(이미 S3 버전 관리를 사용하는 경우 이 작업은 아무런 효과가 없음).
2. 버킷에 샘플 객체를 추가합니다. 응답으로 Amazon S3는 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 });
> ```

코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

```
using Amazon;
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;
        }
    }
}
```

PHP

이 예제에서는 AWS SDK for PHP 버전 3의 클래스를 사용하여 버전이 지정되지 않은 버킷에서 단일 객체를 삭제하는 방법을 보여 줍니다. 버전이 지정된 버킷에서 객체를 삭제하는 방법에 대한 자세한 내용은 REST API 사용 섹션을 참조하십시오.

AWS SDK for Ruby API에 대한 자세한 내용은 AWS SDK for Ruby – 버전 2를 참조하세요.

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.

```
<?php

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'
]);

// 1. Delete the object from the bucket.
try
{
    echo 'Attempting to delete ' . $keyname . '...' . PHP_EOL;

    $result = $s3->deleteObject([
        'Bucket' => $bucket,
        'Key'    => $keyname
    ]);
```

```
    if ($result['DeleteMarker'])
    {
        echo $keyname . ' was deleted or does not exist.' . PHP_EOL;
    } else {
        exit('Error: ' . $keyname . ' was not deleted.' . PHP_EOL);
    }
}
catch (S3Exception $e) {
    exit('Error: ' . $e->getAwsErrorMessage() . PHP_EOL);
}

// 2. Check to see if the object was deleted.
try
{
    echo 'Checking to see if ' . $keyname . ' still exists...' . PHP_EOL;

    $result = $s3->getObject([
        'Bucket' => $bucket,
        'Key'    => $keyname
    ]);

    echo 'Error: ' . $keyname . ' still exists.';
}
catch (S3Exception $e) {
    exit($e->getAwsErrorMessage());
}
```

Javascript

```
import { DeleteObjectCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js" // Helper function that creates Amazon
 S3 service client module.

export const bucketParams = { Bucket: "BUCKET_NAME", Key: "KEY" };

export const run = async () => {
  try {
    const data = await s3Client.send(new DeleteObjectCommand(bucketParams));
    console.log("Success. Object deleted.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
```

```
};
run();
```

여러 객체 삭제

S3 버킷에 있는 모든 객체에 스토리지 비용이 발생하기 때문에 더 이상 필요하지 않은 객체는 삭제해야 합니다. 예를 들어 로그 파일을 수집하는 경우, 더 이상 필요가 없는 로그 파일은 삭제하는 것이 좋습니다. 로그 파일과 같은 객체를 자동으로 삭제하도록 수명 주기 규칙을 설정할 수 있습니다. 자세한 내용은 the section called “수명 주기 구성 설정” 섹션을 참조하세요.

Amazon S3 기능 및 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

Amazon S3 콘솔, AWS SDK 또는 REST API를 사용하여 S3 버킷에서 여러 객체를 동시에 삭제할 수 있습니다.

S3 콘솔 사용

Amazon S3 콘솔을 사용하여 버킷에서 여러 객체를 삭제하려면 다음 단계를 수행합니다.

Warning

- 지정된 객체를 삭제하면 실행 취소할 수 없습니다.
- 이 작업은 지정된 모든 객체를 삭제합니다. 폴더를 삭제할 때 폴더에 새 객체를 추가하기 전에 삭제 작업이 완료될 때까지 기다립니다. 그러지 않으면 새 객체도 삭제될 수 있습니다.
- 디렉터리 버킷을 비롯해 버전 관리가 활성화되지 않은 버킷에서 객체를 삭제하는 경우 Amazon S3가 객체를 영구적으로 삭제합니다.
- 버킷 버전 관리가 활성화 또는 일시 중지된 버킷의 객체를 삭제하는 경우 Amazon S3에서 삭제 마커를 생성합니다. 자세한 내용은 삭제 마커를 통한 작업을 참조하십시오.

버전 관리가 활성화되거나 일시 중지된 객체를 삭제하는 방법

Note

버전 관리가 일시 중지된 버킷에 있는 객체의 버전 ID가 NULL로 표시되는 경우에는 이전 버전이 없으므로 S3는 해당 객체를 영구 삭제합니다. 그러나 버전 관리가 일시 중지된 버킷의 해당

객체에 대한 유효한 버전 ID가 있는 경우 S3는 삭제된 객체에 대한 삭제 마커를 생성하고 객체의 이전 버전을 그대로 유지합니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 객체를 삭제하려는 버킷 이름을 선택합니다.
4. 객체를 선택한 다음 삭제를 선택합니다.
5. 개체를 삭제할까요? 텍스트 상자의 지정된 객체 아래에 있는 객체 목록의 삭제를 확인하려면 **`delete`**를 입력합니다.

버전 관리가 활성화된 버킷에서 특정 객체 버전을 영구적으로 삭제하는 방법

Warning

Amazon S3에서 지정된 객체 버전을 영구 삭제하는 경우 이를 실행 취소할 수 없습니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 객체를 삭제하려는 버킷 이름을 선택합니다.
4. 삭제할 객체를 선택합니다.
5. 버전 표시 토글을 선택합니다.
6. 객체 버전을 선택한 다음 삭제를 선택합니다.
7. 객체를 삭제할까요? 텍스트 상자의 지정된 객체 아래에 있는 특정 객체 버전의 영구 삭제를 확인하려면 영구 삭제를 입력합니다. Amazon S3가 특정 객체 버전을 영구적으로 삭제합니다.

버전 관리가 활성화되지 않은 Amazon S3 버킷의 객체를 영구적으로 삭제하는 방법

> **Warning**
>
> Amazon S3에서 객체를 영구 삭제하는 경우 이를 실행 취소할 수 없습니다. 또한 디렉터리 버킷을 비롯해 버전 관리가 활성화되지 않은 버킷의 경우 삭제는 영구적입니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷 또는 디렉터리 버킷을 선택합니다.
3. 버킷 목록에서 객체를 삭제하려는 버킷 이름을 선택합니다.
4. 객체를 선택한 다음 삭제를 선택합니다.
5. 객체를 삭제할까요? 텍스트 상자의 지정된 객체 아래에 있는 객체의 영구 삭제를 확인하려면 permanently delete를 입력합니다.

> **Note**
>
> 객체 삭제와 관련하여 문제가 발생하는 경우 버전이 지정된 객체를 영구적으로 삭제하고 싶습니다. 섹션을 참조하세요.

AWS SDK 사용

AWS SDK를 사용하여 여러 객체를 삭제하는 방법에 대한 예제는 Amazon S3 API 참조의 코드 예제를 참조하세요.

다양한 AWS SDK 사용에 대한 일반적인 정보는 Amazon S3 API 참조의 AWS SDK를 사용하여 Amazon S3에서 개발을 참조하세요.

REST API 사용

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 참조의 여러 객체 삭제를 참조하십시오.

개체 구성, 나열 및 작업

Amazon S3에서는 접두사를 사용하여 스토리지를 구성할 수 있습니다. 접두사는 버킷에 있는 객체를 논리적으로 그룹화한 것입니다. 비슷한 데이터를 버킷의 동일한 디렉터리에 저장할 수 있도록 접두사 값은 디렉터리 이름과 비슷합니다. 프로그래밍 방식으로 객체를 업로드하는 경우 접두사를 사용하여 데이터를 구성할 수 있습니다.

Amazon S3 콘솔에서는 접두사를 폴더라고 합니다. S3 콘솔에서 버킷으로 이동하여 모든 객체와 폴더를 볼 수 있습니다. 또한 객체 속성과 같은 각 객체에 대한 정보를 볼 수 있습니다.

Amazon S3에서 데이터를 나열하고 구성하는 방법에 대한 자세한 내용은 다음 주제를 참조하십시오.

주제

- 접두어를 사용한 객체 구성
- 프로그래밍 방식으로 객체 키 나열
- Amazon S3 콘솔에서 폴더를 사용하여 객체 구성
- Amazon S3 콘솔에서 객체 속성 보기
- 태그를 사용하여 스토리지 분류

접두어를 사용한 객체 구성

접두사를 사용하여 Amazon S3 버킷에 저장할 데이터를 구성할 수 있습니다. 접두사는 객체 키 이름의 시작 부분에 있는 문자열입니다. 접두사는 임의의 길이일 수 있지만 객체 키 이름의 최대 길이(1,024바이트)를 초과할 수 없습니다. 접두사를 디렉터리와 비슷한 방식으로 데이터를 구성하는 방법으로 생각할 수 있습니다. 그러나 접두사는 디렉터리가 아닙니다.

접두사로 검색하면 지정된 접두사로 시작하는 키로 결과가 제한됩니다. 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. 다른 문자를 구분 기호로 사용할 수 있습니다. 슬래시(/) 문자에 특별한 점은 없지만 매우 일반적인 접두사 구분 기호입니다. 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. 이러한 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 주를 나열하려면 `Delimiter='/'` 및 `Prefix='North America/USA/'`를 설정합니다. 데이터가 저장된 캐나다의 모든 지방을 나열하려면 `Delimiter='/'` 및 `Prefix='North America/Canada/'`를 설정합니다.

구분자, 접두사 및 중첩 폴더에 대한 자세한 내용은 접두사와 중첩 폴더의 차이를 참조하십시오.

접두사 및 구분 기호를 사용한 객체 나열

구분 기호를 사용한 나열 요청을 실행하는 경우 각 수준별로 계층 구조를 탐색할 수 있어 하위 수준의 수백만 키를 건너뛸 수 있습니다. 예를 들어 다음 키가 있는 버킷(*amzn-s3-demo-bucket*)이 있다고 가정합니다.

```
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 슬래시(/) 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 `CommonPrefixes` 접두사 값을 가진 단일 `photos/` 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>amzn-s3-demo-bucket</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>
```

프로그래밍 방식으로 객체 키를 나열하는 방법에 대한 자세한 내용은 프로그래밍 방식으로 객체 키 나열 섹션을 참조하십시오.

프로그래밍 방식으로 객체 키 나열

Amazon S3에서는 접두사별로 키를 나열할 수 있습니다. 관련 키 이름에 대한 공통 접두사를 선택하고 계층 구조를 구분하는 특수 문자로 이러한 키를 표시할 수 있습니다. 그런 다음 나열 작업을 사용하여 계층적으로 키를 선택하고 찾아볼 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**
>
> SOAP API는 신규 고객에게 제공되지 않으며 2025년 8월 31일에 서비스 종료(EOL)됩니다. 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: "`French/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 접두어를 사용한 객체 구성 섹션을 참조하십시오.

REST API

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 Amazon Simple Storage Service API 참조의 GET Bucket(List Objects) 버전 2를 참조하십시오.

효율적인 나열 기능 구현

나열 기능의 성능은 버킷의 총 키 수에 크게 영향을 받지 않습니다. 또한 `prefix`, `marker`, `maxkeys` 또는 `delimiter` 인수의 유무에도 영향을 받지 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 그 조합을 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.

예시

버킷의 모든 객체를 나열할 때는 `s3:ListBucket` 권한이 있어야 한다는 점에 유의하십시오.

CLI

객체 나열

다음 예시에서는 `list-objects` 명령을 사용하여 지정된 버킷에 있는 모든 객체의 이름을 표시합니다.

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size: Size}'
```

이 예시에서는 `--query` 인수를 사용하여 `list-objects`의 출력을 각 객체의 키 값 및 크기로 필터링합니다.

객체에 대한 자세한 내용은 Amazon S3의 객체 작업 섹션을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListObjects를 참조하세요.

ls

다음 예제에서는 `ls` 명령을 사용하여 버킷에 있는 모든 객체와 접두사를 나열합니다.

이 예제 명령을 사용하려면 *amzn-s3-demo-bucket*을 버킷의 이름으로 바꿉니다.

```
$ aws s3 ls s3://amzn-s3-demo-bucket
```

- 상위 수준 명령 `ls`에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 버킷 및 객체 나열 섹션을 참조하세요.

PowerShell

Tools for PowerShell V4

예시 1: 이 명령은 “test-files” 버킷에 있는 모든 항목에 대한 정보를 검색합니다.

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

예시 2: 이 명령은 "test-files" 버킷에서 "sample.txt" 항목에 대한 정보를 검색합니다.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

예시 3: 이 명령은 “test-files” 버킷에서 접두사가 “sample”인 모든 항목에 대한 정보를 검색합니다.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListObjects를 참조하세요.

Tools for PowerShell V5

예시 1: 이 명령은 “test-files” 버킷에 있는 모든 항목에 대한 정보를 검색합니다.

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

예시 2: 이 명령은 "test-files" 버킷에서 "sample.txt" 항목에 대한 정보를 검색합니다.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

예시 3: 이 명령은 “test-files” 버킷에서 접두사가 “sample”인 모든 항목에 대한 정보를 검색합니다.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet Reference (V5)의 ListObjects를 참조하세요.

Amazon S3 콘솔에서 폴더를 사용하여 객체 구성

Amazon S3 범용 버킷에서 객체는 기본 리소스이며 객체는 버킷에 저장됩니다. Amazon S3 범용 버킷은 파일 시스템에서와 같이 계층 대신 단순한 구조를 가지고 있습니다. 하지만 간결한 구성을 위해 Amazon S3 콘솔에서는 객체를 그룹화하는 수단으로 폴더 개념을 지원합니다. 콘솔은 그룹화된 객체에 공유 이름 접두사를 사용하여 이 작업을 수행합니다. 즉, 그룹화된 객체는 공통 문자열로 시작하는 이름을 갖습니다. 이 공통 문자열 또는 공유 접두사는 폴더 이름입니다. 객체 이름을 키 이름이라고도 합니다.

예를 들어 콘솔에서 범용 버킷 안에 `photos`라는 폴더를 만들고 그 안에 `myphoto.jpg`라는 객체를 저장할 수 있습니다. 그러면 객체가 키 이름 `photos/myphoto.jpg`와 함께 저장됩니다. 여기서 `photos/`는 접두사입니다.

아래에 두 가지 예가 더 있습니다.

- 범용 버킷에 세 개의 객체(`logs/date1.txt`, `logs/date2.txt` 및 `logs/date3.txt`)가 있다면 콘솔은 `logs`라는 폴더를 표시합니다. 콘솔에서 폴더를 열면 세 객체 `date1.txt`, `date2.txt` 및 `date3.txt`가 표시됩니다.
- `photos/2017/example.jpg`라는 객체가 있는 경우 콘솔에는 `2017` 폴더가 포함된 `photos` 폴더가 표시됩니다. `2017` 폴더에는 `example.jpg` 객체가 포함되어 있습니다.

폴더 안에 폴더를 만들 수 있지만 버킷 안에 버킷을 만들 수는 없습니다. 객체를 폴더로 직접 업로드 또는 복사할 수 있습니다. 폴더를 생성하고 삭제하고 퍼블릭으로 만들 수 있지만 폴더 이름을 바꿀 수는 없습니다. 객체를 다른 폴더로 복사할 수 있습니다.

> **Important**
>
> Amazon S3 콘솔에서 폴더를 생성하면 S3가 0바이트 객체를 생성합니다. 이 객체 키는 제공한 폴더 이름에 후행 슬래시(/) 문자가 추가되어 설정됩니다. 예를 들어 Amazon S3 콘솔에서 버킷에 `photos`라는 폴더를 만드는 경우 Amazon S3 콘솔이 `photos/` 키를 사용하여 0바이트 객체를 생성합니다. 콘솔은 폴더에 대한 아이디어를 지원하기 위해 이 객체를 만듭니다.
> 또한 이름에 후행 슬래시 문자(/)가 포함된 기존 객체는 Amazon S3 콘솔에 폴더로 표시됩니다. 예를 들어 키 이름이 `examplekeyname/`인 객체는 Amazon S3 콘솔에서 객체가 아닌 폴더로 표시됩니다. 그렇지 않으면 다른 객체처럼 동작하며 AWS Command Line Interface(AWS CLI), AWS SDK 또는 REST API를 통해 볼 수 있고 조작도 할 수 있습니다. 또한 Amazon S3 콘솔을 사용하여 키 이름에 후행 슬래시(/) 문자가 포함된 객체를 업로드할 수 없습니다. 이름에 후행 슬래시(/) 문자가 포함된 객체는 AWS Command Line Interface(AWS CLI), AWS SDK 또는 REST API를 사용하여 업로드할 수 있습니다.
> 또한 Amazon S3 콘솔에는 다른 객체와는 달리 폴더 객체의 콘텐츠와 메타데이터가 표시되지 않습니다. 콘솔을 사용하여 이름에 후행 슬래시(/) 문자가 포함된 객체를 복사할 경우 대상 위치에 새 폴더가 생성되지만 객체의 데이터와 메타데이터는 복사되지 않습니다. 또한 객체 키 이름에 슬래시(/)를 사용하려면 특별한 처리가 필요할 수 있습니다. 자세한 내용은 Amazon S3 객체 이름 지정 섹션을 참조하세요.

디렉터리 버킷에서 폴더를 생성하려면 폴더를 업로드합니다. 자세한 내용은 디렉터리 버킷에 객체 업로드 섹션을 참조하세요.

주제

- 폴더 생성
- 퍼블릭 폴더 설정

- 폴더 크기 계산
- 폴더 삭제

폴더 생성

이 섹션에서는 Amazon S3 콘솔을 사용하여 폴더를 만드는 방법을 설명합니다.

> **Important**
>
> 버킷 정책에서 태그, 메타데이터 또는 액세스 제어 목록(ACL) 권한 부여자없이 이 버킷에 객체를 업로드할 수 없도록 하는 경우에는 다음 절차를 사용하여 폴더를 만들 수 없습니다. 그 대신 빈 폴더를 업로드하고 업로드 구성에서 다음 설정을 지정합니다.

폴더 생성

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 폴더를 만들 버킷의 이름을 선택합니다.
4. 객체 탭에서 폴더 생성을 선택합니다.
5. 폴더의 이름을 입력합니다(예: **`favorite-pics`**).

   > **Note**
   >
   > 폴더 이름에는 특정 제한 사항과 지침이 적용되며, 1,024바이트로 제한되는 객체의 객체 키 이름의 일부로 간주됩니다. 자세한 내용은 the section called “객체 이름 지정” 섹션을 참조하세요.

6. (선택 사항) 버킷 정책에 따라 특정 암호화 키로 객체를 암호화해야 하는 경우 서버 측 암호화에서 암호화 키 지정을 선택하고 폴더를 생성할 때 동일한 암호화 키를 지정해야 합니다. 동일한 암호화 키를 지정하지 않으면 폴더 생성이 실패합니다.
7. 폴더 생성을 선택합니다.

퍼블릭 폴더 설정

퍼블릭 폴더 또는 버킷이 특별히 필요하지 않은 경우에는 Amazon S3 폴더 및 버킷에 대한 모든 퍼블릭 액세스를 차단하는 것이 좋습니다. 폴더를 퍼블릭으로 설정하면 인터넷에서 누구나 해당 폴더에 있는 그룹화된 모든 객체를 볼 수 있습니다.

Amazon S3 콘솔에서 폴더를 퍼블릭으로 설정할 수 있습니다. 또한 접두사별로 데이터 액세스를 제한하는 버킷 정책을 생성하여 폴더를 퍼블릭으로 설정할 수도 있습니다. 자세한 내용은 Amazon S3의 ID 및 액세스 관리 섹션을 참조하세요.

> **Warning**
>
> Amazon S3 콘솔에서 폴더를 퍼블릭으로 설정한 후에는 다시 프라이빗으로 설정할 수 없습니다. 대신에, 객체에 대한 퍼블릭 액세스가 허용되지 않도록 퍼블릭 폴더에 있는 각 개별 객체에 대한 권한을 설정해야 합니다. 자세한 내용은 ACL 구성 섹션을 참조하세요.

주제

- 폴더 크기 계산
- 폴더 삭제

폴더 크기 계산

이 섹션에서는 Amazon S3 콘솔을 사용하여 폴더의 크기를 계산하는 방법을 설명합니다.

폴더 크기를 계산하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 범용 버킷 목록에서 폴더가 저장된 버킷의 이름을 선택합니다.
4. 객체 목록에서 폴더 이름 옆의 확인란을 선택합니다.
5. Actions(작업)를 선택한 다음 Calculate total size(총 크기 계산)를 선택합니다.

> **Note**
>
> 페이지에서 다른 곳으로 이동하면 폴더 정보(총 크기 포함)를 더 이상 사용할 수 없습니다. 다시 보려면 총 크기를 다시 계산해야 합니다.

> **Important**
>
> 버킷 내 지정된 객체 또는 폴더에 대해 Calculate total size(총 크기 계산) 작업을 사용하면 Amazon S3에서 총 객체 수와 총 스토리지 크기를 계산합니다. 하지만 완료되지 않았거나 진행 중인 멀티파트 업로드와 현재 버전이 아닌 버전이나 이전 버전은 총 객체 수 또는 총 크기에 계산되지 않습니다. 이 작업은 버킷에 저장된 각 객체의 현재 또는 최신 버전만 총 객체 수와 총 크기를 계산합니다.
>
> 예를 들어, 버킷에 객체의 버전이 두 개 있는 경우 Amazon S3 스토리지 계산기는 이를 하나의 객체로 계산합니다. 따라서 Amazon S3 콘솔에서 계산되는 총 객체 수는 S3 Storage Lens에 표시된 객체 수 지표 및 Amazon CloudWatch 지표 `NumberOfObjects`에서 보고한 객체 수와 다를 수 있습니다. 마찬가지로, 총 스토리지 크기도 S3 Storage Lens에 표시된 Total Storage(총 스토리지) 지표 및 CloudWatch에 표시된 `BucketSizeBytes` 지표와 다를 수 있습니다.

폴더 삭제

이 섹션에서는 Amazon S3 콘솔을 사용하여 S3 버킷에서 폴더를 삭제하는 방법을 설명합니다.

Amazon S3 기능 및 요금에 대한 자세한 내용은 Amazon S3를 참조하십시오.

S3 버킷에서 폴더 삭제

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 범용 버킷 목록에서 폴더를 삭제할 해당 버킷의 이름을 선택합니다.
4. 객체 목록에서 삭제할 폴더 및 객체 옆의 확인란을 선택합니다.
5. 삭제를 선택합니다.
6. 객체 삭제 페이지에서 삭제하기 위해 선택한 폴더와 객체의 이름이 지정된 객체에 나열되어 있는지 확인합니다.

7. [객체 삭제(Delete objects)] 상자에 **delete**를 입력하고 [객체 삭제(Delete objects)]를 선택합니다.

> **Warning**
>
> 이 작업은 지정된 모든 객체를 삭제합니다. 폴더를 삭제할 때 폴더에 새 객체를 추가하기 전에 삭제 작업이 완료될 때까지 기다립니다. 그러지 않으면 새 객체도 삭제될 수 있습니다.

Amazon S3 콘솔에서 객체 속성 보기

Amazon S3 콘솔을 사용하여 스토리지 클래스, 암호화 설정, 태그 및 메타데이터와 같은 객체의 속성을 볼 수 있습니다.

객체의 속성 보기

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷 또는 디렉터리 버킷을 선택합니다.
3. 버킷 목록에서 객체가 포함된 버킷의 이름을 선택합니다.
4. [객체(Objects)] 목록에서 속성을 보려는 객체의 이름을 선택합니다.

   객체에 대한 [객체 개요(Object overview)]가 열립니다. 아래로 스크롤하여 객체 속성을 볼 수 있습니다.
5. 객체 개요 페이지에서 다음과 같은 객체 속성을 보거나 구성할 수 있습니다.

> **Note**
>
> - 스토리지 클래스, 암호화, 메타데이터 속성을 변경하면 새 객체가 생성되어 이전 객체를 대체합니다. S3 버전 관리가 사용 설정된 경우 객체의 새 버전이 생성되고 기존 객체는 이전 버전이 됩니다. 또한 속성을 변경하는 역할도 새 객체(또는 객체 버전)의 소유자가 됩니다.
> - 사용자 정의 태그가 있는 객체의 스토리지 클래스, 암호화 또는 메타데이터 속성을 변경하는 경우 `s3:GetObjectTagging` 권한이 있어야 합니다. 사용자 정의 태그는 없지만 크기가 16MB를 초과하는 객체의 이러한 속성을 변경하는 경우 `s3:GetObjectTagging` 권한도 있어야 합니다.

대상 버킷 정책에서 `s3:GetObjectTagging` 작업을 거부하는 경우 객체의 이러한 속성은 업데이트되지만 사용자 정의 태그는 객체에서 제거되며 오류가 발생합니다.

a. 스토리지 클래스 - Amazon S3의 각 객체에는 연결된 스토리지 클래스가 있습니다. 해당 객체에 액세스하는 빈도에 따라 사용할 스토리지 클래스를 선택합니다. 범용 버킷에 있는 S3 객체의 기본 스토리지 클래스는 Standard입니다. 디렉터리 버킷에 있는 S3 객체의 기본 스토리지 클래스는 S3 Express One Zone입니다. 객체를 업로드할 때 사용할 스토리지 클래스를 선택하면 됩니다. 스토리지 클래스에 대한 자세한 정보는 Amazon S3 스토리지 클래스 이해 및 관리를 참조하십시오.

   범용 버킷에 객체를 업로드한 뒤에 스토리지 클래스를 변경하려면 스토리지 클래스를 선택합니다. 원하는 스토리지 클래스를 선택한 다음 저장을 선택합니다.

   > **Note**
   >
   > 디렉터리 버킷에 있는 객체의 스토리지 클래스는 변경할 수 없습니다.

b. [서버 측 암호화 설정(Server-side encryption settings)] - 서버 측 암호화를 사용하여 S3 객체를 암호화할 수 있습니다. 자세한 내용은 AWS KMS(SSE-KMS)를 사용한 서버 측 암호화 지정 또는 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) 지정을 참조하세요.

c. 메타데이터 - Amazon S3의 각 객체에는 해당 메타데이터를 나타내는 이름-값 페어 세트가 있습니다. S3 객체에 메타데이터를 추가하는 방법에 대한 자세한 내용은 Amazon S3 콘솔에서 객체 메타데이터 편집 섹션을 참조하십시오.

d. 태그 - 범용 버킷에 있는 S3 객체에 태그를 추가하여 스토리지를 분류합니다. 자세한 내용은 태그를 사용하여 스토리지 분류 섹션을 참조하세요.

e. Object Lock 법적 보존 및 보존 - 범용 버킷의 객체가 삭제되는 것을 방지할 수 있습니다. 자세한 내용은 Object Lock으로 객체 잠금 섹션을 참조하세요.

태그를 사용하여 스토리지 분류

객체 태그 지정을 이용하여 스토리지를 분류합니다. 각 태그는 키-값 페어입니다.

새 객체를 업로드할 때 태그를 덧붙이거나 기존 객체에 태그를 덧붙일 수 있습니다.

- 한 객체에 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. Amazon S3 객체 태그는 내부적으로 UTF-16 형식으로 표시됩니다. 참고로 UTF-16에서 문자는 1 또는 2자 위치를 차지합니다.
- 키와 값은 대/소문자를 구분합니다.
- 태그 제한에 대한 자세한 내용은 AWS 결제 및 비용 관리 사용 설명서의 사용자 정의 태그 제한을 참조하세요. 기본 태그 제한에 대해서는 Amazon EC2 사용 설명서의 태그 제한을 참조하세요.

예시

태그 지정에 대한 다음 예시를 참고하십시오.

Example PHI 정보

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
```

Example 프로젝트 파일

S3 버킷에 프로젝트 파일을 저장한다고 가정합시다. 아래와 같이 `Project`라는 키와 값으로 이 객체들에 태그를 지정할 수 있습니다.

```
Project=Blue
```

Example 다중 태그

다음과 같이 한 객체에 태그를 여러 개 덧붙일 수 있습니다.

```
Project=x
Classification=confidential
```

키 이름 접두사 및 태그

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.

추가 혜택

태그 지정은 데이터 분류 뿐만 아니라 다음과 같은 이점도 제공합니다.

- 객체 태그를 통해 권한에 대한 액세스 제어를 세부적으로 수행할 수 있습니다. 예를 들어, 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 여러 Amazon S3 객체에 객체 태그 세트 추가

단일 요청으로 둘 이상의 Amazon S3 객체에 객체 태그 세트를 추가하려면 S3 배치 작업을 사용하면 됩니다. 작업할 객체 목록을 S3 배치 작업에 제공합니다. S3 배치 작업은 지정된 작업을 수행하기 위해 각 API 작업을 호출합니다. 단일 배치 작업 건으로 엑사바이트 규모의 데이터가 포함된 수십억 개의 객체에서 지정된 작업을 수행할 수 있습니다.

S3 배치 작업 기능은 진행 상황을 추적하고 알림을 보내며 모든 작업에 대한 자세한 완료 보고서를 저장하여 감사 가능한 완전관리형 서버리스 환경을 제공합니다. Amazon S3 콘솔, AWS CLI, AWS SDK 또는 REST API를 통해 S3 배치 작업을 사용할 수 있습니다. 자세한 내용은 the section called “배치 작업 기본 사항” 섹션을 참조하세요.

객체 태그에 대한 자세한 내용은 객체 태그 관리 섹션을 참조하세요.

객체 태그 지정과 관련된 API 작업

Amazon S3은 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 세트로 이 요청을 전송하는 경우 Amazon S3은 객체에 있는 기존 태그 세트를 삭제합니다. 이 방법을 사용하면 티어 1 요청(PUT)에 대한 요금이 부과됩니다. 자세한 내용은 Amazon S3 요금을 참조하세요.
    > DELETE Object 태그 지정 요청은 요금이 부과되지 않고 동일한 결과를 얻으므로 더 유용합니다.

- GET Object 태그 지정 - 객체에 연결된 태그 세트를 반환합니다. Amazon S3은 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 S3은 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 S3이 태그를 복사(기본 동작)하거나 요청 시 제공되는 새로운 태그 세트로 태그를 교체하도록 유도할 수 있습니다.

다음을 참조하십시오.

- S3 객체 태그 지정은 매우 일관적입니다. 자세한 내용은 Amazon S3 데이터 일관성 모델 섹션을 참조하세요.

추가 구성

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 가정해 보겠습니다. 이 객체들에 다음과 같이 태그를 지정할 수 있습니다.

```
phototype=raw
or
phototype=finished
```

원본 사진을 생성한 후 언젠가 S3 Glacier에 아카이브하는 것을 고려해 볼 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 섹션을 참조하세요.

객체 태그 지정 및 복제

버킷에서 복제를 구성한 경우, Amazon S3에 태그를 읽을 수 있는 권한을 부여하면 Amazon S3에서 태그를 복제합니다. 자세한 내용은 라이브 복제 설정 개요 섹션을 참조하세요.

객체 태깅 이벤트 알림

객체 태그가 객체에서 추가되거나 삭제될 때 알림을 받도록 Amazon S3 이벤트 알림을 설정할 수 있습니다. `s3:ObjectTagging:Put` 이벤트 유형은 태그가 객체에 PUT되거나 기존 태그가 업데이트될 때 알려줍니다. `s3:ObjectTagging:Delete` 이벤트 유형은 객체에서 태그가 제거될 때 알려줍니다. 자세한 내용은 이벤트 알림 사용 설정를 참조하세요.

객체 태깅에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- 태그 지정 및 액세스 제어 정책
- 객체 태그 관리

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
- 버킷 작업

객체 태그를 통해 권한 관리에 대한 액세스 제어를 세부적으로 수행할 수 있습니다. 객체 태그를 바탕으로 조건적인 권한을 부여할 수 있습니다. Amazon S3에서는 객체 태그 기반의 조건부 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 키 전체 목록은 조건 키를 사용한 버킷 정책 예시 단원을 참조하세요. 다음 권한 정책들은 객체 태그 지정을 통해 어떻게 세부적인 액세스 권한 관리가 가능한지 보여줍니다.

Example 1: 사용자가 특정 태그 및 키 값이 있는 객체만 읽도록 허용

다음 권한 정책은 `environment: production` 태그 키 및 값이 있는 객체만 사용자가 읽을 수 있도록 제한합니다. 이 정책은 `s3:ExistingObjectTag` 조건 키를 사용하여 태그 키 및 값을 지정합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
  {
    "Principal": {
      "AWS": [
        "arn:aws:iam::111122223333:role/JohnDoe"
      ]
    },
    "Effect": "Allow",
    "Action": ["s3:GetObject", "s3:GetObjectVersion"],
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
    "Condition": {
      "StringEquals":
        {"s3:ExistingObjectTag/environment": "production"}
    }
  }
  ]
}
```

Example 2. 사용자가 추가할 수 있는 객체 태그 키 제한

다음 권한 정책은 사용자에게 `s3:PutObjectTagging` 작업을 수행할 수 있는 권한을 부여합니다. 이에 따라 사용자는 기존 객체에 태그를 붙일 수 있습니다. 조건이 `s3:RequestObjectTagKeys` 조건 키를 사용하여 `Owner`, `CreationDate` 등의 허용 태그 키 세트를 지정합니다. 자세한 내용은 IAM 사용 설명서의 다수의 키 또는 값을 사용하는 조건 생성을 참조하십시오.

요청에 지정된 모든 태그 키가 인증된 태그 키임을 정책이 보장합니다. 지정된 값 중 최소 1개가 요청에 존재함을 조건 내의 ForAnyValue 한정자가 보장합니다.

```
{
   "Version": "2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
            "arn:aws:iam::111122223333:role/JohnDoe"
          ]
       },
 "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ],
      "Condition": {"ForAnyValue:StringEquals": {"s3:RequestObjectTagKeys": [
            "Owner",
            "CreationDate"
          ]
        }
      }
    }
  ]
}
```

Example 3. 사용자가 객체 태그를 추가할 수 있게 하려면 특정 태그 키 및 값 필요

다음 권한 정책은 s3:PutObjectTagging 작업을 수행할 수 있는 권한을 사용자에게 부여하여, 사용자가 기존 객체에 태그를 추가할 수 있게 합니다. 이 조건은 값이 *X*로 설정된 (*Project* 등의) 특정 태그 키를 포함하도록 사용자에게 요구합니다.

```
{
   "Version": "2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
       "arn:aws:iam::111122223333:user/JohnDoe"
         ]
       },
      "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
```

```
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ],
      "Condition": {"StringEquals": {"s3:RequestObjectTag/Project": "X"
        }
      }
    }
  ]
}
```

객체 태그 관리

이 섹션에서는 AWS SDK for Java 및 .NET 또는 Amazon S3 콘솔을 사용하여 객체 태그를 관리하는 방법에 대해 설명합니다.

객체 태그 지정을 통해 범용 버킷에서 스토리지를 분류할 수 있습니다. 모든 태그는 다음 규칙이 적용되는 키-값 페어입니다.

- 한 객체에 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. Amazon S3 객체 태그는 내부적으로 UTF-16 형식으로 표시됩니다. 참고로 UTF-16에서 문자는 1 또는 2자 위치를 차지합니다.
- 키와 값은 대/소문자를 구분합니다.

객체 태그에 대한 자세한 내용은 태그를 사용하여 스토리지 분류 섹션을 참조하세요. 태그 제한에 대한 자세한 내용은 AWS 결제 및 비용 관리 사용 설명서의 사용자 정의 태그 제한을 참조하세요.

S3 콘솔 사용

객체에 태그 추가

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 객체가 포함된 버킷의 이름을 선택합니다.
4. 변경할 객체의 이름 왼쪽에 있는 확인란을 선택합니다.

5. 작업(Actions) 메뉴에서 태그 편집(Edit tags)을 선택합니다.
6. 나열된 객체를 검토하고 태그 추가(Add tags)를 선택합니다.
7. 각 객체 태그는 키-값 페어입니다. 키와 값을 입력합니다. 태그를 더 추가하려면 태그 추가(Add Tag)를 선택합니다.

   한 개체에 태그를 최대 10개까지 입력할 수 있습니다.
8. [변경 사항 저장(Save changes)]을 선택합니다.

   Amazon S3는 지정된 객체에 태그를 추가합니다.

자세한 내용은 이 설명서의 Amazon S3 콘솔에서 객체 속성 보기 및 객체 업로드 단원을 참조하세요.

AWS SDK 사용

Java

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 태그를 사용하여 스토리지 분류 단원을 참조하십시오. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 AWS SDK for Java 개발자 안내서의 시작하기 섹션을 참조하세요.

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

.NET

다음 예제는 AWS SDK for .NET를 사용하여 새 객체에 대한 태그를 설정하고 기존 객체에 대한 태그를 가져오거나 대체하는 방법을 보여줍니다. 객체 태그 지정에 대한 자세한 정보는 태그를 사용하여 스토리지 분류 단원을 참조하십시오.

코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

```
using Amazon;
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);
            }
        }
    }
}
```

미리 서명된 URL을 통해 객체 다운로드 및 업로드

미리 서명된 URL을 사용하여 버킷 정책을 업데이트하지 않고도 Amazon S3의 객체에 시간 제한 액세스를 부여할 수 있습니다. 미리 서명된 URL은 브라우저에 입력하거나 프로그램에서 객체를 다운로드하는 데 사용할 수 있습니다. 미리 서명된 URL에서 사용하는 자격 증명은 URL을 생성한 AWS 사용자의 자격 증명입니다.

또한 미리 서명된 URL을 사용하여 다른 사람이 특정 객체를 Amazon S3 버킷에 업로드하도록 허용할 수도 있습니다. 이 경우 상대방에게 AWS 보안 자격 증명이나 권한이 없어도 업로드할 수 있습니다. 미리 서명된 URL에 지정된 것과 동일한 키를 가진 객체가 버킷에 이미 존재하는 경우 Amazon S3는 기존 객체를 업로드한 객체로 대체합니다.

미리 서명된 URL은 만료 날짜 및 시간까지 여러 번 사용할 수 있습니다.

미리 서명된 URL을 생성하면 자격 증명을 제공한 후 다음을 지정해야 합니다.

- Amazon S3 버킷
- 객체 키(이 객체를 다운로드하면 Amazon S3 버킷, 업로드하면 업로드할 파일 이름)
- HTTP 메서드(객체 다운로드를 위한 GET, 업로드를 위한 PUT, 객체 메타데이터를 읽기 위한 HEAD 등)
- 만료 시간 간격

현재 Amazon S3에 미리 서명된 URL은 객체를 업로드할 때 다음과 같은 데이터 무결성 체크섬 알고리즘(CRC32, CRC32C, SHA-1, SHA-256) 사용을 지원하지 않습니다. 업로드 후 객체의 무결성을 확인하려면 미리 서명된 URL과 함께 객체를 업로드할 때 객체의 MD5 다이제스트를 제공하는 것입니다. 객체 무결성에 대한 자세한 내용은 Amazon S3에서 객체 무결성 확인 섹션을 참조하십시오.

주제

- 미리 서명된 URL을 생성할 수 있는 사용자
- 미리 서명된 URL의 만료 시간
- 미리 서명된 URL 기능 제한
- 미리 서명된 URL을 통해 객체 공유
- 미리 서명된 URL을 통해 객체 공유

미리 서명된 URL을 생성할 수 있는 사용자

유효한 보안 자격 증명을 가진 사용자는 누구나 미리 서명된 URL을 만들 수 있습니다. 미리 서명된 URL에서 제공하려는 작업을 수행할 권한이 있는 사용자가 생성해야 이 URL을 통해 성공적으로 객체에 액세스할 수 있습니다.

다음은 미리 서명된 URL을 생성하는 데 사용할 수 있는 자격 증명 유형입니다.

- IAM 인스턴스 프로파일: 최대 6시간 동안 유효함.
- AWS Security Token Service - 장기간 자격 증명으로 서명한 경우 최대 36시간 또는 임시 자격 증명 중 먼저 종료되는 기간까지 유효합니다.
- IAM 사용자: AWS 서명 버전 4를 사용할 경우 최대 7일 동안 유효함.

  최대 7일 동안 유효한 미리 서명된 URL을 생성하려면 먼저 미리 서명된 URL을 만드는 데 사용하는 방법으로 IAM 사용자 자격 증명(액세스 키 및 비밀 키)을 위임합니다.

> **Note**
>
> 임시 자격 증명을 사용하여 미리 서명된 URL을 생성하면 자격 증명이 만료되면 이 URL도 만료됩니다. 일반적으로 미리 서명된 URL은 URL을 만드는 데 사용한 자격 증명이 취소, 삭제 또는 비활성화될 때 만료됩니다. 이 URL이 자격 증명보다 이후의 만료 시간으로 생성된 경우에도 마찬가지입니다. 임시 자격 증명의 수명은 IAM 사용 설명서의 AWS STS API 작업 비교를 참조하십시오.

미리 서명된 URL의 만료 시간

미리 서명된 URL은 URL이 생성될 때 지정된 기간 동안 유효합니다. Amazon S3 콘솔로 미리 서명된 URL을 생성하는 경우 만료 시간은 1분~12시간 사이로 설정할 수 있습니다. AWS CLI또는 AWS SDK를 사용하는 경우 만료 시간을 최대 7일로 설정할 수 있습니다.

임시 토큰을 사용하여 미리 서명된 URL을 생성한 경우 토큰이 만료되면 이 URL도 만료됩니다. 일반적으로 미리 서명된 URL은 URL을 만드는 데 사용한 자격 증명이 취소, 삭제 또는 비활성화될 때 만료됩니다. 이 URL이 자격 증명보다 이후의 만료 시간으로 생성된 경우에도 마찬가지입니다. 사용하는 자격 증명이 만료 시간에 미치는 영향에 대한 자세한 내용은 미리 서명된 URL을 생성할 수 있는 사용자 섹션을 참조하십시오.

Amazon S3는 HTTP 요청이 있을 때 서명된 URL의 만료 날짜 및 시간을 확인합니다. 예를 들어 클라이언트가 만료 시간 직전에 대용량 파일을 다운로드하기 시작한 경우, 다운로드 중에 만료 시간이 경과해도 다운로드는 진행됩니다. 단, 연결이 끊어진 경우 클라이언트가 만료 시간 이후에 다운로드를 다시 시작하는 것은 불가능합니다.

미리 서명된 URL 기능 제한

미리 서명된 URL의 기능은 이를 만든 사용자의 권한에 의해 제한됩니다. 미리 서명된 URL은 기본적으로 이를 소유한 사용자에게 액세스 권한을 부여하는 보유자 토큰입니다. 따라서 이러한 URL은 적절하게 보호하는 것이 좋습니다. 미리 서명된 URL의 사용을 제한할 때 사용할 수 있는 몇 가지 방법은 다음과 같습니다.

AWS Signature Version 4(SigV4)

사전 서명된 URL 요청이 AWS Signature Version 4(SigV4)를 사용하여 인증될 때 특정 동작을 적용하기 위해 버킷 정책 및 액세스 포인트 정책에서 조건 키를 사용할 수 있습니다. 예를 들어, 다음 버킷 정책은 서명이 10분 이상 지난 경우 `s3:signatureAge` 조건을 사용하여 *amzn-s3-demo-bucket* 버

킷의 객체에 대한 Amazon S3 미리 서명된 URL 요청을 거부합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Deny a presigned URL request if the signature is more than 10 min
 old",
         "Effect": "Deny",
         "Principal": {"AWS":"*"},
         "Action": "s3:*",
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
         "Condition": {
            "NumericGreaterThan": {
               "s3:signatureAge": 600000
            }
         }
      }
   ]
}
```

AWS 서명 버전 4와 관련된 정책 키에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 AWS 서명 버전 4 인증을 참조하십시오.

네트워크 경로 제한

미리 서명된 URL의 사용과 모든 Amazon S3 액세스를 특정 네트워크 경로로 제한하려는 경우, AWS Identity and Access Management(IAM) 정책을 작성할 수 있습니다. 이러한 정책은 호출을 하는 IAM 보안 주체, Amazon S3 버킷 또는 그 두 가지 모두에 설정할 수 있습니다.

IAM 보안 주체에 대한 네트워크 경로 제한은 해당 자격 증명의 사용자가 지정된 네트워크에서 요청해야 합니다. 버킷 또는 액세스 포인트에 대한 제한은 해당 리소스에 대한 모든 요청이 지정된 네트워크에서 시작되도록 요구합니다. 이러한 제한은 미리 서명된 URL 시나리오 외부에서도 적용됩니다.

사용하는 IAM 전역 조건 키는 엔드포인트 유형에 따라 달라집니다. Amazon S3용 퍼블릭 엔드포인트를 사용하는 경우 `aws:SourceIp`를 사용합니다. Amazon S3에 대한 가상 프라이빗 클라우드(VPC) 엔드포인트를 사용하는 경우 `aws:SourceVpc` 또는 `aws:SourceVpce`를 사용하십시오.

다음 IAM 정책 설명에서는 보안 주체가 지정된 네트워크 범위에서만 AWS에 액세스해야 합니다. 이 정책 설명을 사용하면 모든 액세스가 해당 범위에서 시작되어야 합니다. 여기에는 Amazon S3

에 대해 미리 서명된 URL을 사용하는 사례가 포함됩니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
{
    "Sid": "NetworkRestrictionForIAMPrincipal",
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
        "NotIpAddressIfExists": {"aws:SourceIp": "IP-address-range"},
        "BoolIfExists": {"aws:ViaAWSService": "false"}
    }
}
```

미리 서명된 URL을 통해 객체 공유

기본적으로 모든 Amazon S3 객체는 프라이빗이며 객체 소유자만 액세스할 수 있는 권한이 있습니다. 그러나 객체 소유자는 미리 서명된 URL을 생성하여 다른 사람과 객체를 공유할 수 있습니다. 미리 서명된 URL은 보안 자격 증명을 사용하여 객체를 다운로드할 수 있는 시간 제한 권한을 부여합니다. 해당 URL은 브라우저에 입력하거나 프로그램에서 객체를 다운로드하는 데 사용할 수 있습니다. 미리 서명된 URL에서 사용하는 자격 증명은 URL을 생성한 AWS 사용자의 자격 증명입니다.

미리 서명된 URL에 대한 일반적인 내용은 미리 서명된 URL을 통해 객체 다운로드 및 업로드 섹션을 참조하세요.

Amazon S3 콘솔, AWS Explorer for Visual Studio(Windows) 또는 AWS Toolkit for Visual Studio Code을 사용하여 코드를 작성하지 않고도 객체를 공유하기 위한 미리 서명된 URL을 생성할 수 있습니다. 또한 AWS Command Line Interface(AWS CLI) 또는 AWS SDK를 사용하여 프로그래밍 방식으로 미리 서명된 URL을 생성할 수 있습니다.

S3 콘솔 사용

다음 단계에 따라 Amazon S3 콘솔을 사용하여 객체 공유를 위해 미리 서명된 URL을 생성할 수 있습니다. 콘솔을 사용할 때 미리 서명된 URL의 최대 만료 시간은 생성 시점으로부터 12시간입니다.

Amazon S3 콘솔을 사용하여 미리 서명된 URL을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.

3. 범용 버킷 목록에서 미리 서명된 URL을 사용할 객체가 있는 범용 버킷 이름을 선택합니다.
4. 객체(Objects) 목록에서 미리 서명된 URL을 생성할 객체를 선택합니다.
5. 객체 작업 메뉴에서 미리 서명된 URL 공유를 선택합니다.
6. 미리 서명된 URL의 유효 기간을 지정합니다.
7. 미리 서명된 URL 생성을 선택합니다.
8. 확인 메시지가 표시되면 URL이 클립보드로 자동으로 복사됩니다. 미리 서명된 URL을 다시 복사해야 하는 경우 미리 서명된 URL을 복사하는 버튼이 표시됩니다.

AWS CLI 사용

다음 예제 AWS CLI 명령은 Amazon S3 버킷에서 객체를 공유하기 위한 미리 서명된 URL을 생성합니다. AWS CLI을 사용하면 미리 서명된 URL의 최대 만료 시간은 생성 시점으로부터 7일입니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws s3 presign s3://amzn-s3-demo-bucket/mydoc.txt --expires-in 604800
```

> **Note**
>
> 2019년 3월 20일 이후에 출시된 모든 AWS 리전 리전의 경우 요청에 엔드포인트 `endpoint-url`과 `AWS ## 리전`을 지정해야 합니다. 모든 Amazon S3 리전 및 엔드포인트 목록은 AWS 일반 참조의 리전 및 엔드포인트를 참조하세요.

```
aws s3 presign s3://amzn-s3-demo-bucket/mydoc.txt --expires-in 604800 --region af-south-1 --endpoint-url https://s3.af-south-1.amazonaws.com
```

자세한 내용은 AWS CLI 명령 레퍼런스의 presign 섹션을 참조하십시오.

AWS SDK 사용

AWS SDK를 사용하여 프로그래밍 방식으로 미리 서명된 URL을 생성할 수 있습니다.

Python

다음 Python 스크립트는 객체를 공유하기 위해 `GET` 미리 서명된 URL을 생성합니다.

1. 스크립트의 내용을 복사하여 '*get-only-url.py*' 파일로 저장합니다. 다음 예시를 사용하려면 *user input placeholders*를 실제 정보(예: 파일 이름)로 바꾸세요.

```
import argparse
import boto3
from botocore.exceptions import ClientError

def generate_presigned_url(s3_client, client_method, method_parameters,
 expires_in):
    """
    Generate a presigned Amazon S3 URL that can be used to perform an action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds the presigned URL is valid for.
    :return: The presigned URL.
    """
    try:
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method,
            Params=method_parameters,
            ExpiresIn=expires_in
        )
    except ClientError:
        print(f"Couldn't get a presigned URL for client method
 '{client_method}'.")
        raise
    return url

def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("bucket", help="The name of the bucket.")
    parser.add_argument(
        "key", help="The key (path and filename) in the S3 bucket.",
    )
    args = parser.parse_args()

    # By default, this will use credentials from ~/.aws/credentials
    s3_client = boto3.client("s3")

    # The presigned URL is specified to expire in 1000 seconds
    url = generate_presigned_url(
```

```
        s3_client,
        "get_object",
        {"Bucket": args.bucket, "Key": args.key},
        1000
    )
    print(f"Generated GET presigned URL: {url}")

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

2. 파일을 공유하기 위해 GET 미리 서명된 URL을 생성하려면 버킷 이름과 원하는 객체 경로로 다음 스크립트를 실행합니다.

   다음 명령에서는 예제 값을 사용합니다. *user input placeholders*를 사용자의 정보로 대체합니다.

   ```
   python get-only-url.py amzn-s3-demo-bucket <object-path>
   ```

   스크립트는 GET 미리 서명된 URL을 출력합니다.

   ```
   Generated GET presigned URL: https://amzn-s3-demo-
   bucket.s3.amazonaws.com/object.txt?
   AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Signature=vjbyNxybdZaMmLa
   %2ByT372YEAiv4%3D&Expires=1741978496
   ```

3. 다음 curl을 통해 생성된 미리 서명된 URL을 사용하여 파일을 다운로드할 수 있습니다.

   ```
   curl -X GET "generated-presigned-url" -o "path/to/save/file"
   ```

AWS SDK를 사용하여 객체를 공유하기 위한 미리 서명된 URL을 생성하는 더 많은 예시는 Create a presigned URL for Amazon S3 by using an AWS SDK를 참조하세요.

> **Note**
>
> 2019년 3월 20일 이후에 출시된 모든 AWS 리전 리전의 경우 요청에 엔드포인트 endpoint-url과 AWS ## 리전을 지정해야 합니다. 모든 Amazon S3 리전 및 엔드포인트 목록은 AWS 일반 참조의 리전 및 엔드포인트를 참조하십시오.

> **Note**
>
> AWS SDK를 사용할 때 태그 지정 속성은 쿼리 파라미터가 아닌 헤더여야 합니다. 다른 모든 속성은 미리 서명된 URL의 파라미터로 전달할 수 있습니다.

사용 AWS Toolkit for Visual Studio(Windows)

> **Note**
>
> 현재 AWS Toolkit for Visual Studio는 Mac용 Visual Studio를 지원하지 않습니다.

1. AWS Toolkit for Visual Studio 사용 설명서의 Installing and setting up the Toolkit for Visual Studio 지침에 따라 AWS Toolkit for Visual Studio를 설치합니다.
2. 다음 단계, AWS Toolkit for Visual Studio 사용 설명서의 AWS에 연결 단계를 사용하여 AWS에 연결합니다.
3. AWS Explorer라는 레이블이 붙은 왼쪽 패널에서 객체가 들어 있는 버킷을 두 번 클릭합니다.
4. 미리 서명된 URL을 생성하려는 객체를 마우스 오른쪽 버튼으로 클릭하고 미리 서명된 URL 생성...을 선택합니다.
5. 팝업 창에서 미리 서명된 URL의 만료 날짜와 시간을 설정합니다.
6. 객체 키는 선택한 객체를 기반으로 미리 채워집니다.
7. GET을 선택하여 이 미리 서명된 URL이 객체를 다운로드하는 데 사용되도록 지정합니다.
8. 생성 버튼을 선택합니다.
9. URL을 클립보드에 복사하려면 복사를 선택합니다.
10. 생성된 미리 서명된 URL을 사용하려면 URL을 브라우저에 붙여 넣습니다.

AWS Toolkit for Visual Studio Code 사용하기

Visual Studio Code를 사용할 경우 코드를 작성하지 않고도 공유할 수 있는 AWS Toolkit for Visual Studio Code를 사용하여 미리 서명된 객체 URL을 만들 수 있습니다. 일반적인 정보는 AWS Toolkit for Visual Studio Code 사용 설명서의 AWS Toolkit for Visual Studio Code 섹션을 참조하십시오.

AWS Toolkit for Visual Studio Code을 설치하는 방법에 대한 지침은 AWS Toolkit for Visual Studio Code 사용 설명서의 AWS Toolkit for Visual Studio Code 설치를 참조하십시오.

1. 다음 단계, AWS Toolkit for Visual Studio Code 사용 설명서의 AWS Toolkit for Visual Studio Code에 연결 단계를 사용하여 AWS에 연결합니다.
2. Visual Studio Code의 왼쪽 패널에서 AWS 로고를 선택합니다.
3. Explorer에서 S3를 선택합니다.
4. 버킷과 파일을 선택하고 컨텍스트 메뉴를 엽니다(마우스 오른쪽 버튼 클릭).
5. 미리 서명된 URL 생성을 선택한 다음 만료 시간(분 단위)을 설정합니다.
6. Enter 키를 누르면 미리 서명된 URL이 클립보드에 복사됩니다.

미리 서명된 URL을 통해 객체 공유

미리 서명된 URL을 사용하여 다른 사람이 Amazon S3 버킷에 객체를 업로드하도록 허용할 수 있습니다. 미리 서명된 URL을 사용하면 상대방에게 AWS 보안 자격 증명이나 권한이 없어도 업로드할 수 있습니다. 미리 서명된 URL은 이를 생성하는 사용자의 권한에 따라 제한됩니다. 즉, 객체를 업로드하기 위해 미리 서명된 URL을 수신하는 경우, URL의 생성자가 해당 객체를 업로드하는 데 필요한 권한을 보유하는 경우에만 객체를 업로드할 수 있습니다.

사용자가 URL을 사용하여 객체를 업로드하는 경우 Amazon S3는 지정된 버킷에 객체를 생성합니다. 미리 서명된 URL에 지정된 것과 동일한 키를 사용하는 객체가 이미 버킷에 있다면 Amazon S3는 업로드된 객체로 기존 객체를 바꿉니다. 업로드 후에는 버킷 소유자가 객체를 소유하게 됩니다.

미리 서명된 URL에 대한 일반적인 내용은 미리 서명된 URL을 통해 객체 다운로드 및 업로드 섹션을 참조하십시오.

AWS Explorer for Visual Studio를 사용하여 코드를 작성할 필요 없이 객체 업로드를 위해 미리 서명된 URL을 만들 수 있습니다. AWS SDK를 사용하여 프로그래밍 방식으로 미리 서명된 URL을 생성할 수도 있습니다.

> **Note**
>
> 현재 AWS Toolkit for Visual Studio는 Mac용 Visual Studio를 지원하지 않습니다.

사용 AWS Toolkit for Visual Studio(Windows)

1. AWS Toolkit for Visual Studio 사용 설명서의 Installing and setting up the Toolkit for Visual Studio 지침에 따라 AWS Toolkit for Visual Studio를 설치합니다.

2. 다음 단계, AWS Toolkit for Visual Studio 사용 설명서의 AWS에 연결 단계를 사용하여 AWS에 연결합니다.
3. AWS Explorer라는 레이블이 붙은 왼쪽 패널에서 객체를 업로드하려는 버킷을 마우스 오른쪽 버튼으로 클릭합니다.
4. 미리 서명된 URL 생성...을 선택합니다.
5. 팝업 창에서 미리 서명된 URL의 만료 날짜와 시간을 설정합니다.
6. 객체 키의 경우 업로드할 파일의 이름을 설정합니다. 업로드하는 파일은 이 이름과 정확히 일치해야 합니다. 동일한 객체 키를 가진 객체가 버킷에 이미 존재하는 경우 Amazon S3는 기존 객체를 새로 업로드한 객체로 대체합니다.
7. PUT을 선택하여 이 미리 서명된 URL이 객체를 업로드하는 데 사용되도록 지정합니다.
8. 생성 버튼을 선택합니다.
9. URL을 클립보드에 복사하려면 복사를 선택합니다.
10. 이 URL을 사용하려면 `curl` 명령으로 PUT 요청을 전송합니다. 파일의 전체 경로와 미리 서명된 URL 자체를 포함하세요.

```
curl -X PUT -T "/path/to/file" "presigned URL"
```

AWS SDK를 사용하여 파일을 업로드하기 위한 **PUT** 미리 서명된 URL을 생성합니다.

제한된 시간 동안 S3 작업을 수행할 수 있는 미리 서명된 URL을 생성할 수 있습니다.

> **Note**
>
> AWS CLI 또는 AWS SDK를 사용하는 경우 미리 서명된 URL의 만료 시간을 최대 7일로 설정할 수 있습니다. 자세한 내용은 Expiration time for presigned URLs를 참조하세요.

Python

다음 Python 스크립트는 S3 범용 버킷에 객체를 업로드하기 위해 PUT 미리 서명된 URL을 생성합니다.

1. 스크립트의 내용을 복사하여 '*put-only-url.py*' 파일로 저장합니다. 다음 예시를 사용하려면 *user input placeholders*를 실제 정보(예: 파일 이름)로 바꾸세요.

```
import argparse
```

```
import boto3
from botocore.exceptions import ClientError

def generate_presigned_url(s3_client, client_method, method_parameters,
 expires_in):
    """
    Generate a presigned Amazon S3 URL that can be used to perform an action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds the presigned URL is valid for.
    :return: The presigned URL.
    """
    try:
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method,
            Params=method_parameters,
            ExpiresIn=expires_in
        )
    except ClientError:
        print(f"Couldn't get a presigned URL for client method
 '{client_method}'.")
        raise
    return url

def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("bucket", help="The name of the bucket.")
    parser.add_argument(
        "key", help="The key (path and filename) in the S3 bucket.",
    )
    args = parser.parse_args()

    # By default, this will use credentials from ~/.aws/credentials
    s3_client = boto3.client("s3")

    # The presigned URL is specified to expire in 1000 seconds
    url = generate_presigned_url(
        s3_client,
        "put_object",
        {"Bucket": args.bucket, "Key": args.key},
        1000
    )
```

```
    print(f"Generated PUT presigned URL: {url}")

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

2. 파일을 업로드하기 위해 PUT 미리 서명된 URL을 생성하려면 버킷 이름과 원하는 객체 경로로 다음 스크립트를 실행합니다.

   다음 명령에서는 예제 값을 사용합니다. *user input placeholders*를 사용자의 정보로 대체합니다.

   ```
   python put-only-url.py amzn-s3-demo-bucket <object-path>
   ```

   스크립트는 PUT 미리 서명된 URL을 출력합니다.

   ```
   Generated PUT presigned URL: https://amzn-s3-demo-
   bucket.s3.amazonaws.com/object.txt?
   AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Signature=vjbyNxybdZaMmLa
   %2ByT372YEAiv4%3D&Expires=1741978496
   ```

3. 이제 다음 curl을 통해 생성된 미리 서명된 URL을 사용하여 파일을 업로드할 수 있습니다.

   ```
   curl -X PUT -T "path/to/your/local/file" "generated-presigned-url"
   ```

AWS SDK를 사용하여 객체를 업로드하기 위한 미리 서명된 URL을 생성하는 추가 예제는 Create a presigned URL for Amazon S3 by using an AWS SDK를 참조하세요.

S3 객체 Lambda를 사용하여 객체 변환

Amazon S3 객체 Lambda를 통해 자체 코드를 Amazon S3 GET, LIST, HEAD 요청에 추가하여 애플리케이션으로 데이터가 반환될 때 데이터를 수정 및 처리할 수 있습니다. 사용자 지정 코드를 사용하여 행을 필터링하고 동적으로 이미지 크기를 조정하고 워터마크를 삽입하고 기밀 데이터를 교정하는 등 S3 GET 요청에서 반환한 데이터를 수정할 수 있습니다. 또한, S3 객체 Lambda를 사용하여 S3 LIST 요청의 출력을 수정하여 버킷 내 모든 객체의 사용자 지정 보기를 생성하고, S3 HEAD 요청의 출력을 수정하여 객체 이름 및 크기와 같은 객체 메타데이터를 수정할 수 있습니다. S3 객체 Lambda를 Amazon CloudFront 배포의 오리진으로 사용하여 이미지 크기 자동 조정, 이전 형식 트랜스코딩(예: JPEG에서 WebP로), 메타데이터 스트리핑과 같이 최종 사용자를 위해 데이터를 조정할 수 있습니다. 자세한 내용은 Amazon CloudFront와 함께 Amazon S3 객체 Lambda 사용하기 AWS 블로그 게시물을

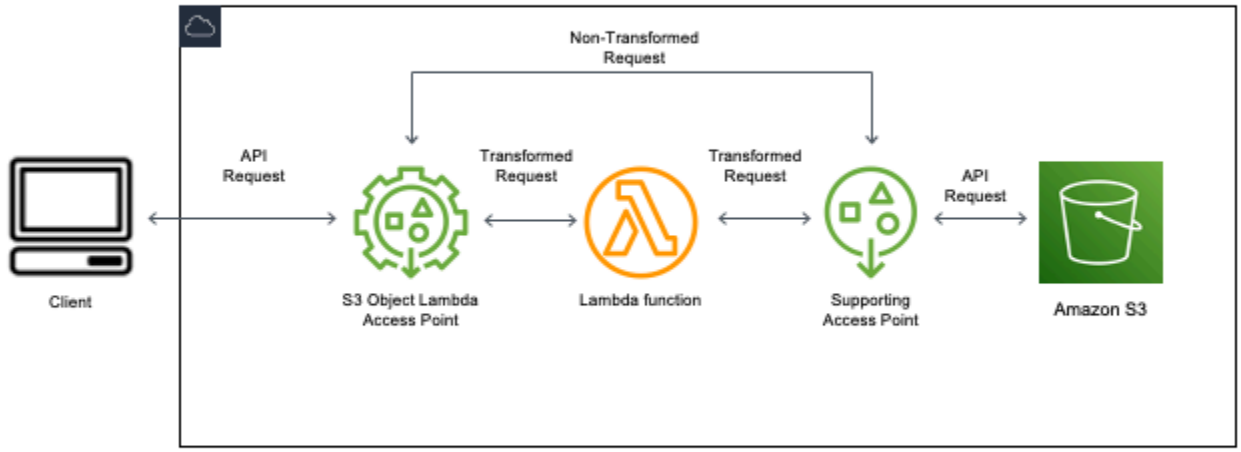
참조하십시오. AWS Lambda 함수로 구동되는 코드는 AWS에서 완전히 관리되는 인프라에서 실행됩니다. S3 객체 Lambda를 사용하면 데이터의 파생 사본을 생성 및 저장하거나 프록시를 실행하지 않아도 되며, 애플리케이션을 변경하지 않아도 됩니다.

S3 객체 Lambda 작동 방식

S3 객체 Lambda는 AWS Lambda 함수를 사용하여 표준 S3 GET, LIST 또는 HEAD 요청의 출력을 자동으로 처리합니다. AWS Lambda는 기본 컴퓨팅 리소스를 관리할 필요 없이 고객 정의 코드를 실행하는 서버리스 컴퓨팅 서비스입니다. 자체 사용자 지정 Lambda 함수를 작성 및 실행하여 특정 사용 사례에 맞게 데이터 변환을 조정할 수 있습니다.

Lambda 함수를 구성한 후에 객체 Lambda 액세스 포인트라고도 하는 S3 객체 Lambda 서비스 엔드포인트에 연결합니다. Object Lambda 액세스 포인트는 지원 액세스 포인트라고도 하는 표준 S3 액세스 포인트를 사용하여 데이터에 액세스합니다.

객체 Lambda 액세스 포인트로 요청을 보낼 때 Amazon S3가 자동으로 Lambda 함수를 호출합니다. 객체 Lambda 액세스 포인트를 통해 S3 GET, LIST 또는 HEAD 요청을 사용하여 검색되는 모든 데이터는 변환된 결과를 다시 애플리케이션으로 반환합니다. 다른 모든 요청은 다음 다이어그램과 같이 일반적으로 처리됩니다.



이 섹션의 주제에서는 S3 객체 Lambda를 사용한 작업 방법에 대해 설명합니다.

주제

- 객체 Lambda 액세스 포인트 생성
- Amazon S3 객체 Lambda 액세스 포인트 사용
- S3 객체 Lambda 액세스 포인트에 대한 보안 고려 사항
- S3 객체 Lambda 액세스 포인트에 대한 Lambda 함수 작성
- AWS 빌드 Lambda 함수 사용
- S3 객체 Lambda에 대한 모범 사례 및 지침
- S3 객체 Lambda 자습서
- S3 Object Lambda 디버깅 및 문제 해결

객체 Lambda 액세스 포인트 생성

Object Lambda 액세스 포인트는 생성 중에 지정하는 정확히 하나의 표준 액세스 포인트와 연결됩니다. 객체 Lambda 액세스 포인트를 생성하려면 다음 리소스가 필요합니다.

- 표준 S3 액세스 지점. Object Lambda 액세스 포인트를 사용할 때 이 표준 액세스 포인트를 지원 액세스 포인트라고 하며, S3 버킷 또는 Amazon FSx for OpenZFS 볼륨에 연결됩니다. 표준 액세스 지점 생성에 대한 자세한 내용은 the section called “액세스 포인트 생성” 섹션을 참조하세요.
- AWS Lambda 함수. 직접 Lambda 함수를 만들 수도 있고 사전 빌드된 함수를 사용할 수도 있습니다. Lambda 함수 생성에 대한 자세한 내용은 the section called “Lambda 함수 작성” 섹션을 참조하세요. 미리 빌드된 함수에 대한 자세한 내용은 AWS 빌드 Lambda 함수 사용 섹션을 참조하십시오.
- (선택 사항) AWS Identity and Access Management (IAM) 정책. Amazon S3 액세스 지점은 리소스, 사용자 또는 기타 조건별로 액세스 지점 사용을 제어할 수 있는 IAM 리소스 정책을 지원합니다. 이러한 정책 생성에 대한 자세한 내용은 the section called “IAM 정책 구성” 섹션을 참조하십시오.

다음 섹션에서는 다음을 사용하여 객체 Lambda 액세스 포인트를 생성하는 방법에 대해 설명합니다.

- AWS Management Console은
- AWS Command Line Interface(AWS CLI)
- AWS CloudFormation 템플릿
- AWS 클라우드 개발 키트 (AWS CDK)은

REST API를 사용하여 객체 Lambda 액세스 포인트를 생성하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 `CreateAccessPointForObjectLambda` 섹션을 참조하십시오.

Object Lambda 액세스 포인트 생성

다음 절차 중 하나를 사용하여 객체 Lambda 액세스 포인트를 생성합니다.

S3 콘솔 사용

콘솔을 사용하여 객체 Lambda 액세스 포인트 생성

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 전환하려는 리전을 선택합니다.
3. 왼쪽 탐색 창에서 객체 Lambda 액세스 포인트(Object Lambda Access Points)를 선택합니다.
4. 객체 Lambda 액세스 포인트(Object Lambda Access Points) 페이지에서 객체 Lambda 액세스 포인트 생성(Create Object Lambda Access Point)을 선택합니다.
5. Object Lambda access point name(객체 Lambda 액세스 지점 이름)에 액세스 지점에 사용할 이름을 입력합니다.

   표준 액세스 포인트와 마찬가지로 객체 Lambda 액세스 포인트에 대한 명명 규칙이 있습니다. 자세한 내용은 액세스 포인트 이름 지정 규칙 섹션을 참조하세요.
6. Supporting Access Point(지원 액세스 지점)에 사용하려는 표준 액세스 지점을 입력하거나 찾아봅니다. 액세스 포인트는 변환하려는 객체와 동일한 AWS 리전에 있어야 합니다. 표준 액세스 지점 생성에 대한 자세한 내용은 the section called “액세스 포인트 생성” 섹션을 참조하세요.
7. 변환 구성에서 객체 Lambda 액세스 포인트의 데이터를 변환하는 함수를 추가할 수 있습니다. 다음 중 하나를 수행합니다.

   - 계정에 이미 AWS Lambda 함수가 있는 경우 Invoke Lambda function(Lambda 함수 호출)에서 해당 함수를 선택할 수 있습니다. 여기에서 AWS 계정에 있는 Lambda 함수의 Amazon 리소스 이름(ARN)을 입력하거나 드롭다운 메뉴에서 Lambda 함수를 선택할 수 있습니다.
   - AWS에서 빌드한 함수를 사용하려면 AWS built function(빌드 함수)에서 함수 이름을 선택하고 Create Lambda function(Lambda 함수 생성)을 선택합니다. 그러면 빌드된 함수를 AWS 계정에 배포할 수 있는 Lambda 콘솔로 이동합니다. 빌드된 함수에 대한 자세한 내용은 AWS 빌드 Lambda 함수 사용 섹션을 참조하십시오.

   S3 APIs(S3 API)에서 호출할 API 작업을 하나 이상 선택합니다. 선택한 각 API에 대해 호출할 Lambda 함수를 지정해야 합니다.

8. (선택 사항) Payload(페이로드) 아래에서 Lambda 함수에 입력으로 제공할 JSON 텍스트를 추가합니다. 동일한 Lambda 함수를 호출하는 여러 객체 Lambda 액세스 포인트에 대해 서로 다른 파라미터를 사용하여 페이로드를 구성할 수 있으므로 Lambda 함수의 유연성을 높일 수 있습니다.

   > **Important**
   >
   > 객체 Lambda 액세스 포인트를 사용할 때 페이로드에는 기밀 정보를 포함하지 않아야 합니다.

9. (선택 사항) Range and part number(범위 및 부분 번호)의 경우 범위 및 부분 번호 헤더가 있는 `GET` 및 `HEAD` 요청을 처리하려면 이 옵션을 활성화해야 합니다. 이 옵션을 사용하면 Lambda 함수로 이러한 요청을 인식하고 처리할 수 있음을 확인하는 것입니다. 범위 헤더 및 부분 번호에 대한 자세한 내용은 Range 및 partNumber 헤더 작업 섹션을 참조하십시오.

10. (선택 사항) 요청 지표에서 활성화 또는 비활성화를 선택하여 객체 Lambda 액세스 포인트에 Amazon S3 모니터링을 추가합니다. 요청 지표는 표준 Amazon CloudWatch 요금으로 청구됩니다.

11. (선택 사항) Object Lambda Access Point Policy(객체 Lambda 액세스 지점 정책)에서 리소스 정책을 설정합니다. 리소스 정책은 지정된 객체 Lambda 액세스 포인트에 대한 권한을 부여하며 리소스, 사용자 또는 기타 조건별로 액세스 포인트 사용을 제어할 수 있습니다. 객체 Lambda 액세스 지점 리소스 정책에 대한 자세한 내용은 객체 Lambda 액세스 포인트에 대한 IAM 정책 구성 섹션을 참조하십시오.

12. Block Public Access settings for this Object Lambda Access Point(이 객체 Lambda 액세스 포인트에 대한 퍼블릭 액세스 차단 설정)에서, 적용할 퍼블릭 액세스 차단 설정을 선택합니다. 새 객체 Lambda 액세스 지점에는 기본적으로 모든 퍼블릭 액세스 차단 설정이 활성화되어 있으며 기본 설정을 그대로 유지하는 것이 좋습니다. Amazon S3에서는 현재 객체 Lambda 액세스 지점이 생성된 후 객체 Lambda 액세스 지점의 퍼블릭 액세스 차단 설정을 변경하도록 지원하지 않습니다.

    Amazon S3 퍼블릭 액세스 차단 사용에 대한 자세한 내용은 범용 버킷의 액세스 포인트에 대한 퍼블릭 액세스 관리 섹션을 참조하십시오.

13. Create Object Lambda Access Point(객체 Lambda 액세스 지점 생성)를 선택합니다.

AWS CLI 사용

AWS CloudFormation 템플릿을 사용하여 객체 Lambda 액세스 포인트 생성

> **Note**
>
> 다음 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

1. S3 객체 Lambda 기본 구성에서 AWS Lambda 함수 배포 패키지 `s3objectlambda_deployment_package.zip`을 다운로드합니다.
2. 다음 `put-object` 명령을 실행하여 Amazon S3 버킷에 패키지를 업로드합니다.

```
aws s3api put-object --bucket Amazon S3 bucket name --key
 s3objectlambda_deployment_package.zip --body release/
s3objectlambda_deployment_package.zip
```

3. S3 객체 Lambda 기본 구성에서 AWS CloudFormation 템플릿 `s3objectlambda_defaultconfig.yaml`을 다운로드합니다.
4. 다음 `deploy` 명령을 실행하여 템플릿을 AWS 계정에 배포합니다.

```
aws cloudformation deploy --template-file s3objectlambda_defaultconfig.yaml \
 --stack-name AWS CloudFormation stack name \
 --parameter-overrides ObjectLambdaAccessPointName=Object Lambda Access Point name
 \
  SupportingAccessPointName=Amazon S3 access point S3BucketName=Amazon S3 bucket \
  LambdaFunctionS3BucketName=Amazon S3 bucket containing your Lambda package \
  LambdaFunctionS3Key=Lambda object key LambdaFunctionS3ObjectVersion=Lambda object
 version \
  LambdaFunctionRuntime=Lambda function runtime --capabilities capability_IAM
```

`GET`, `HEAD` 및 `LIST` API 작업에 대해 Lambda를 호출하도록 이 AWS CloudFormation 템플릿을 구성할 수 있습니다. 템플릿의 기본 구성 수정에 대한 자세한 내용은 the section called “AWS CloudFormation을 사용하여 S3 객체 Lambda 설정 자동화” 섹션을 참조하십시오.

AWS CLI를 사용하여 객체 Lambda 액세스 포인트 생성

> **Note**
>
> 다음 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하십시오.

다음 예시에서는 계정 *111122223333*의 *amzn-s3-demo-bucket1* 버킷에 대해 이름이 *my-object-lambda-ap*인 객체 Lambda 액세스 포인트를 생성합니다. 이 예제에서는 이름이 *example-ap*인 표준 액세스 포인트 정책이 이미 생성되어 있다고 가정합니다. 표준 액세스 포인트 생성에 대한 자세한 내용은 the section called “액세스 포인트 생성” 섹션을 참조하십시오.

이 예제에서는 AWS의 미리 빌드된 함수 `decompress`를 사용합니다. 미리 빌드된 함수에 대한 자세한 내용은 the section called “AWS 빌드 함수 사용” 섹션을 참조하십시오.

1. 버킷을 만듭니다. 이 예제에서는 *amzn-s3-demo-bucket1*을 사용합니다. 버킷 생성에 대한 자세한 내용은 the section called “범용 버킷 생성” 섹션을 참조하십시오.
2. 표준 액세스 포인트를 생성하여 버킷에 연결합니다. 이 예제에서는 *example-ap*을 사용합니다. 표준 액세스 지점 생성에 대한 자세한 내용은 the section called “액세스 포인트 생성” 섹션을 참조하십시오.
3. 다음 중 하나를 수행합니다.
   - 계정에 Amazon S3 객체를 변환하는 데 사용할 Lambda 함수를 생성합니다. Lambda 함수 생성에 대한 자세한 내용은 the section called “Lambda 함수 작성” 섹션을 참조하십시오. AWS CLI에서 사용자 지정 함수를 사용하려면 AWS Lambda 개발자 안내서의 AWS CLI에서 Lambda 사용을 참조하십시오.
   - AWS의 사전 빌드된 Lambda 함수를 사용합니다. 미리 빌드된 함수에 대한 자세한 내용은 AWS 빌드 Lambda 함수 사용 섹션을 참조하십시오.
4. 이름이 `my-olap-configuration.json`인 JSON 구성 파일을 생성합니다. 이 구성에서는 이전 단계에서 생성한 Lambda 함수의 지원 액세스 지점과 Amazon 리소스 이름(ARN) 또는 사용 중인 사전 빌드된 함수의 ARN을 제공합니다.

   Example

```
{
    "SupportingAccessPoint" : "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
    "TransformationConfigurations": [{
```

```
        "Actions" : ["GetObject", "HeadObject", "ListObjects", "ListObjectsV2"],
        "ContentTransformation" : {
            "AwsLambda": {
                "FunctionPayload" : "{\"compressionType\":\"gzip\"}",
                "FunctionArn" : "arn:aws:lambda:us-east-1:111122223333:function/
compress"
            }
        }
    }]
}
```

5. `create-access-point-for-object-lambda` 명령을 실행하여 객체 Lambda 액세스 포인트를 생성합니다.

```
aws s3control create-access-point-for-object-lambda --account-id 111122223333 --
name my-object-lambda-ap --configuration file://my-olap-configuration.json
```

6. (선택 사항) 이름이 `my-olap-policy.json`인 JSON 정책 파일을 생성합니다.

   객체 Lambda 액세스 지점 리소스 정책을 추가하면 리소스, 사용자 또는 기타 조건별로 액세스 지점 사용을 제어할 수 있습니다. 이 리소스 정책은 계정 *444455556666*에 대한 `GetObject` 권한을 객체 Lambda 액세스 포인트에 부여합니다.

   Example

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "Grant account 444455556666 GetObject access",
            "Effect": "Allow",
            "Action": "s3-object-lambda:GetObject",
            "Principal": {
                "AWS": "arn:aws:iam::444455556666:root"
            },
            "Resource": "your-object-lambda-access-point-arn"
        }
    ]
}
```

7. (선택 사항) `put-access-point-policy-for-object-lambda` 명령을 실행하여 리소스 정책을 설정합니다.

```
aws s3control put-access-point-policy-for-object-lambda --account-id 111122223333
 --name my-object-lambda-ap --policy file://my-olap-policy.json
```

8. (선택 사항) 페이로드를 지정합니다.

   페이로드는 AWS Lambda 함수에 입력으로 제공할 수 있는 선택적 JSON입니다. 동일한 Lambda 함수를 호출하는 여러 객체 Lambda 액세스 포인트에 대해 서로 다른 파라미터를 사용하여 페이로드를 구성할 수 있으므로 Lambda 함수의 유연성을 높일 수 있습니다.

   다음 객체 Lambda 액세스 포인트 구성은 파라미터 2개가 있는 페이로드를 보여줍니다.

```
{
 "SupportingAccessPoint": "AccessPointArn",
 "CloudWatchMetricsEnabled": false,
 "TransformationConfigurations": [{
  "Actions": ["GetObject", "HeadObject", "ListObjects", "ListObjectsV2"],
  "ContentTransformation": {
   "AwsLambda": {
    "FunctionArn": "FunctionArn",
    "FunctionPayload": "{\"res-x\": \"100\",\"res-y\": \"100\"}"
   }
  }
 }]
}
```

   다음 객체 Lambda 액세스 포인트 구성은 파라미터 1개가 있고 `GetObject-Range`, `GetObject-PartNumber`, `HeadObject-Range` 및 `HeadObject-PartNumber`가 활성화된 페이로드를 보여줍니다.

```
{
    "SupportingAccessPoint":"AccessPointArn",
    "CloudWatchMetricsEnabled": false,
    "AllowedFeatures": ["GetObject-Range", "GetObject-PartNumber", "HeadObject-
Range", "HeadObject-PartNumber"],
    "TransformationConfigurations": [{
        "Action": ["GetObject", "HeadObject", "ListObjects", "ListObjectsV2"],
        "ContentTransformation": {
            "AwsLambda": {
                "FunctionArn":"FunctionArn",
                "FunctionPayload": "{\"compression-amount\": \"5\"}"
            }
```

```
        }
    }]
}
```

> **Important**
>
> 객체 Lambda 액세스 포인트를 사용할 때 페이로드에는 기밀 정보를 포함하지 않아야 합니다.

AWS CloudFormation 콘솔 및 템플릿 사용

Amazon S3에서 제공하는 기본 구성을 사용하여 객체 Lambda 액세스 포인트를 생성할 수 있습니다. GitHub 리포지토리에서 AWS CloudFormation 템플릿 및 Lambda 함수 소스 코드를 다운로드하고 이러한 리소스를 배포하여 기능적 객체 Lambda 액세스 포인트를 설정할 수 있습니다.

AWS CloudFormation 템플릿의 기본 구성 수정에 대한 내용은 the section called “AWS CloudFormation을 사용하여 S3 객체 Lambda 설정 자동화” 섹션을 참조하십시오.

템플릿 없이 AWS CloudFormation을 사용하여 객체 Lambda 액세스 포인트를 구성하는 방법에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 `AWS::S3ObjectLambda::AccessPoint`를 참조하십시오.

Lambda 함수 배포 패키지 업로드

1. S3 객체 Lambda 기본 구성에서 AWS Lambda 함수 배포 패키지 `s3objectlambda_deployment_package.zip`을 다운로드합니다.
2. Amazon S3 버킷에 패키지를 업로드합니다.

AWS CloudFormation 콘솔을 사용하여 객체 Lambda 액세스 포인트 생성

1. S3 객체 Lambda 기본 구성에서 AWS CloudFormation 템플릿 `s3objectlambda_defaultconfig.yaml`을 다운로드합니다.
2. AWS 관리 콘솔에 로그인하고 https://console.aws.amazon.com/cloudformation에서 AWS CloudFormation 콘솔을 엽니다.
3. 다음 중 하나를 수행하십시오.

   - AWS CloudFormation을 한 번도 사용해 본 적이 없다면 AWS CloudFormation 홈페이지에서 Create stack(스택 생성)을 선택합니다.

- AWS CloudFormation을 이전에 사용한 적이 있다면 왼쪽 탐색 창에서 Stacks(스택)를 선택합니다. Create stack(스택 생성)을 선택한 다음 With new resources (standard)새 리소스 사용(표준)를 선택합니다.

4. 사전 조건 - 템플릿 준비(Prerequisite - Prepare template)에서 템플릿 준비 완료를 선택합니다.
5. Specify template(템플릿 지정)에서 Upload a template file(템플릿 파일 업로드)를 선택하고 `s3objectlambda_defaultconfig.yaml`을 업로드합니다.
6. 다음을 선택합니다.
7. 스택 세부 정보 지정(Specify stack details) 페이지에서 스택 이름을 입력합니다.
8. Parameters(파라미터) 섹션에서 스택 템플릿에 정의된 다음 파라미터를 지정합니다.

   a. CreateNewSupportingAccessPoint의 경우 다음 중 하나를 수행합니다.

      - 템플릿을 업로드한 S3 버킷에 대한 지원 액세스 지점이 이미 있는 경우 false(거짓)를 선택합니다.
      - 이 버킷에 대한 새 액세스 지점을 생성하려면 true(참)를 선택합니다.

   b. EnableCloudWatchMonitoring의 경우 Amazon CloudWatch 요청 지표 및 경보를 활성화할지 여부에 따라 true(참) 또는 false(거짓)를 선택합니다.

   c. (선택 사항) LambdaFunctionPayload에서 Lambda 함수에 입력으로 제공할 JSON 텍스트를 추가합니다. 동일한 Lambda 함수를 호출하는 여러 객체 Lambda 액세스 포인트에 대해 서로 다른 파라미터를 사용하여 페이로드를 구성할 수 있으므로 Lambda 함수의 유연성을 높일 수 있습니다.

      > **Important**
      >
      > 객체 Lambda 액세스 포인트를 사용할 때 페이로드에는 기밀 정보를 포함하지 않아야 합니다.

   d. LambdaFunctionRuntime에 Lambda 함수의 원하는 런타임을 입력합니다. 사용 가능한 선택 항목은 `nodejs14.x`, `python3.9`, `java11`입니다.

   e. LambdaFunctionS3BucketName에 배포 패키지를 업로드한 Amazon S3 버킷 이름을 입력합니다.

   f. LambdaFunctionS3Key에 배포 패키지를 업로드한 Amazon S3 객체 키를 입력합니다.

   g. LambdaFunctionS3ObjectVersion에 배포 패키지를 업로드한 Amazon S3 객체 버전을 입력합니다.

   h. ObjectLambdaAccessPointName에 객체 Lambda 액세스 포인트의 이름을 입력합니다.

i. S3BucketName에 객체 Lambda 액세스 포인트와 연결될 Amazon S3 버킷 이름을 입력합니다.

j. SupportingAccessPointName에 지원 액세스 지점의 이름을 입력합니다.

> **Note**
>
> 이것이 이전 단계에서 선택한 Amazon S3 버킷과 연결된 액세스 지점입니다. Amazon S3 버킷과 연결된 액세스 지점이 없는 경우 CreateNewSupportingAccessPoint에서 true(참)를 선택하여 액세스 지점을 생성하도록 템플릿을 구성할 수 있습니다.

9. 다음을 선택합니다.
10. Configure stack options(스택 옵션 구성) 페이지에서 Next(다음)를 선택합니다.

    이 페이지의 선택적 설정에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS CloudFormation 스택 옵션 설정을 참조하십시오.
11. 검토 페이지에서 스택 생성을 선택합니다.

AWS 클라우드 개발 키트 (AWS CDK) 사용

AWS CDK을 사용하여 객체 Lambda 액세스 포인트를 구성하는 방법에 대한 자세한 내용은 AWS 클라우드 개발 키트 (AWS CDK) API 참조의 `AWS::S3ObjectLambda` 구성 라이브러리를 참조하십시오.

CloudFormation 템플릿을 사용하여 S3 객체 Lambda 설정 자동화

AWS CloudFormation 템플릿을 사용하여 Amazon S3 객체 Lambda 액세스 포인트를 빠르게 생성할 수 있습니다. CloudFormation 템플릿은 관련 리소스를 자동으로 생성하고, AWS Identity and Access Management(IAM) 역할을 구성하고, 객체 Lambda 액세스 포인트를 통해 요청을 자동으로 처리하는 AWS Lambda 함수를 설정합니다. CloudFormation 템플릿을 사용하면 모범 사례를 구현하고 보안 태세를 개선하고 수동 프로세스로 인한 오류를 줄일 수 있습니다.

이 GitHub 리포지토리에는 CloudFormation 템플릿 및 Lambda 함수 소스 코드가 포함되어 있습니다. 스키마 템플릿을 사용하는 방법에 대한 지침은 the section called “객체 Lambda 액세스 포인트 생성” 섹션을 참조하십시오.

템플릿에 제공된 Lambda 함수는 변환을 실행하지 않습니다. 대신 기본 데이터 소스에서 있는 그대로 객체를 반환합니다. 함수를 복제하고 고유한 변환 코드를 추가하여 애플리케이션에 반환되는 데이터

를 수정하고 처리할 수 있습니다. 함수 수정에 대한 자세한 내용은 the section called “Lambda 함수 수정” 및 the section called “Lambda 함수 작성” 섹션을 참조하십시오.

템플릿 수정

새로운 지원 액세스 지점 생성

S3 객체 Lambda는 객체 Lambda 액세스 포인트와 지원 액세스 포인트라고도 하는 표준 S3 액세스 포인트라는 두 가지 액세스 포인트를 사용합니다. 객체 Lambda 액세스 포인트에 요청을 수행할 때 S3는 S3 객체 Lambda 구성에 따라 사용자를 대신하여 Lambda를 호출하거나 지원 액세스 포인트에 요청을 위임합니다. 템플릿을 배포할 때 `aws cloudformation deploy` 명령의 일부로 다음 파라미터를 전달하여 새로운 지원 액세스 지점을 생성할 수 있습니다.

```
CreateNewSupportingAccessPoint=true
```

함수 페이로드 구성

템플릿을 배포할 때 `aws cloudformation deploy` 명령의 일부로 다음 파라미터를 전달하여 Lambda 함수에 추가 데이터를 제공하도록 페이로드를 구성할 수 있습니다.

```
LambdaFunctionPayload="format=json"
```

Amazon CloudWatch 모니터링 사용

템플릿을 배포할 때 `aws cloudformation deploy` 명령의 일부로 다음 파라미터를 전달하여 CloudWatch 모니터링을 사용 설정할 수 있습니다.

```
EnableCloudWatchMonitoring=true
```

이 파라미터는 Amazon S3 요청 지표에 대한 객체 Lambda 액세스 포인트를 활성화하고 클라이언트 측 오류와 서버 측 오류를 모니터링하는 2개의 CloudWatch 경보를 생성합니다.

> **Note**
>
> Amazon CloudWatch를 사용하면 추가 비용이 발생합니다. Amazon S3 요청 지표에 대한 자세한 내용은 액세스 포인트 모니터링 및 로깅 섹션을 참조하십시오.
> 요금에 대한 자세한 내용은 CloudWatch 요금을 참조하십시오.

프로비저닝된 동시성 구성

지연 시간을 줄이려면 Resources 아래에 다음 줄을 포함하도록 템플릿을 편집하여 객체 Lambda 액세스 포인트를 지원하는 Lambda 함수에 대해 프로비저닝된 동시성을 구성합니다.

```
LambdaFunctionVersion:
      Type: AWS::Lambda::Version
      Properties:
        FunctionName: !Ref LambdaFunction
        ProvisionedConcurrencyConfig:
            ProvisionedConcurrentExecutions: Integer
```

> **Note**
>
> 동시성 프로비저닝에 대한 추가 요금이 발생합니다. 프로비저닝된 동시성에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 Lambda 프로비저닝된 동시성 관리를 참조하십시오. 요금에 대한 자세한 내용은 AWS Lambda 요금을 참조하십시오.

Lambda 함수 수정

GetObject 요청의 헤더 값 변경

기본적으로 Lambda 함수는 미리 서명된 URL 요청에서 GetObject 클라이언트로 Content-Length 및 ETag를 제외한 모든 헤더를 전달합니다. Lambda 함수의 변환 코드를 기반으로 GetObject 클라이언트에 새 헤더 값을 보내도록 선택할 수 있습니다.

WriteGetObjectResponse API 작업에서 새 헤더 값을 전달하여 Lambda 함수를 업데이트하여 새 헤더 값을 보낼 수 있습니다.

예를 들어 Lambda 함수가 Amazon S3 객체의 텍스트를 다른 언어로 번역하는 경우 Content-Language 헤더에 새 값을 전달할 수 있습니다. 다음과 같이 writeResponse 함수를 수정하면 됩니다.

```
async function writeResponse (s3Client: S3, requestContext: GetObjectContext,
 transformedObject: Buffer,
  headers: Headers): Promise<PromiseResult<{}, AWSError>> {
  const { algorithm, digest } = getChecksum(transformedObject);

  return s3Client.writeGetObjectResponse({
```

```
    RequestRoute: requestContext.outputRoute,
    RequestToken: requestContext.outputToken,
    Body: transformedObject,
    Metadata: {
      'body-checksum-algorithm': algorithm,
      'body-checksum-digest': digest
    },
    ...headers,
    ContentLanguage: 'my-new-language'
  }).promise();
}
```

지원되는 헤더의 전체 목록은 Amazon Simple Storage Service API Reference의 WriteGetObjectResponse 섹션을 참조하십시오.

메타데이터 헤더 반환

WriteGetObjectResponse API 작업 요청에서 새 헤더 값을 전달하여 Lambda 함수를 업데이트하여 새 헤더 값을 보낼 수 있습니다.

```
async function writeResponse (s3Client: S3, requestContext: GetObjectContext,
 transformedObject: Buffer,
  headers: Headers): Promise<PromiseResult<{}, AWSError>> {
  const { algorithm, digest } = getChecksum(transformedObject);

  return s3Client.writeGetObjectResponse({
    RequestRoute: requestContext.outputRoute,
    RequestToken: requestContext.outputToken,
    Body: transformedObject,
    Metadata: {
      'body-checksum-algorithm': algorithm,
      'body-checksum-digest': digest,
      'my-new-header': 'my-new-value'
    },
    ...headers
  }).promise();
}
```

새 상태 코드 반환

사용자 정의 상태 코드를 WriteGetObjectResponse API 작업 요청에서 전달하여 GetObject 클라이언트에 반환할 수 있습니다.

```
async function writeResponse (s3Client: S3, requestContext: GetObjectContext,
 transformedObject: Buffer,
  headers: Headers): Promise<PromiseResult<{}, AWSError>> {
  const { algorithm, digest } = getChecksum(transformedObject);

  return s3Client.writeGetObjectResponse({
    RequestRoute: requestContext.outputRoute,
    RequestToken: requestContext.outputToken,
    Body: transformedObject,
    Metadata: {
      'body-checksum-algorithm': algorithm,
      'body-checksum-digest': digest
    },
    ...headers,
    StatusCode: Integer
  }).promise();
}
```

지원되는 상태 코드의 전체 목록은 Amazon Simple Storage Service API Reference의 *WriteGetObjectResponse* 섹션을 참조하십시오.

원본 객체에 **Range** 및 **partNumber** 적용

기본적으로 CloudFormation 템플릿에서 생성된 객체 Lambda 액세스 포인트는 Range 및 partNumber 파라미터를 처리할 수 있습니다. Lambda 함수는 요청된 범위 또는 부품 번호를 변환된 객체에 적용합니다. 이를 위해 이 함수는 전체 객체를 다운로드하고 변환을 실행해야 합니다. 경우에 따라 변환된 객체 범위가 원본 객체 범위에 정확히 매핑될 수 있습니다. 즉, 원본 객체에서 바이트 범위 A-B를 요청하고 변환을 실행하면 전체 객체를 요청하고 변환을 실행하고 변환된 객체에서 바이트 범위 A-B를 반환하는 것과 동일한 결과를 얻을 수 있습니다.

이러한 경우 Lambda 함수 구현을 변경하여 범위 또는 부품 번호를 원본 객체에 직접 적용할 수 있습니다. 이 방법을 통해 전체 함수 지연 시간과 필요한 메모리가 줄어듭니다. 자세한 내용은 the section called “Range 및 partNumber 헤더 작업” 섹션을 참조하세요.

Range 및 **partNumber** 처리 사용 중지

기본적으로 CloudFormation 템플릿에서 생성된 객체 Lambda 액세스 포인트는 Range 및 partNumber 파라미터를 처리할 수 있습니다. 이 동작이 필요하지 않은 경우 템플릿에서 다음 줄을 제거하여 이를 비활성화할 수 있습니다.

```
AllowedFeatures:
```

```
  - GetObject-Range
  - GetObject-PartNumber
  - HeadObject-Range
  - HeadObject-PartNumber
```

대형 객체 변환

기본적으로 Lambda 함수는 S3 객체 Lambda에 대한 응답 스트리밍을 시작하기 전에 메모리에서 전체 객체를 처리합니다. 변환을 수행할 때 응답을 스트리밍하도록 함수를 수정할 수 있습니다. 이렇게 하면 변환 대기 시간과 Lambda 함수 메모리 크기를 줄이는 데 도움이 됩니다. 구현 예는 압축된 콘텐츠 스트리밍 예를 참조하십시오.

Amazon S3 객체 Lambda 액세스 포인트 사용

S3 객체 Lambda 액세스 포인트를 통해 요청하는 것은 다른 액세스 포인트를 통해 요청하는 것과 동일하게 작동합니다. 액세스 포인트를 통해 요청하는 방법에 대한 자세한 내용은 범용 버킷의 Amazon S3 Access Points 사용 섹션을 참조하십시오. AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하여 객체 Lambda 액세스 포인트를 통해 요청할 수 있습니다.

> **Important**
>
> 객체 Lambda 액세스 포인트에 대한 Amazon 리소스 이름(ARN)은 `s3-object-lambda`의 서비스 이름을 사용합니다. 따라서 객체 Lambda 액세스 포인트 ARN은 다른 액세스 포인트에서 사용되는 `arn:aws:s3` 대신 `arn:aws:s3-object-lambda`로 시작됩니다.

객체 Lambda 액세스 포인트에 대한 ARN을 찾는 방법

AWS CLI 또는 AWS SDK에서 객체 Lambda 액세스 포인트를 사용하려면 객체 Lambda 액세스 포인트의 Amazon 리소스 이름(ARN)을 알아야 합니다. 다음 예시에서는 Amazon S3 콘솔 또는 AWS CLI를 사용하여 객체 Lambda 액세스 포인트에 대한 ARN을 찾는 방법을 보여줍니다.

S3 콘솔 사용

콘솔을 사용하여 객체 Lambda 액세스 포인트에 대한 ARN을 찾는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 객체 Lambda 액세스 포인트(Object Lambda Access Points)를 선택합니다.

3. ARN을 복사할 객체 Lambda 액세스 포인트 옆에 있는 옵션 버튼을 선택합니다.
4. ARN 복사(Copy ARN)를 선택합니다.

AWS CLI 사용

AWS CLI를 사용하여 객체 Lambda 액세스 포인트에 대한 ARN을 찾는 방법

1. AWS 계정와 연결된 객체 Lambda 액세스 포인트의 목록을 검색하려면 다음 명령을 실행합니다. 명령을 실행하기 전에 계정 ID *111122223333*을 자신의 AWS 계정 ID로 바꿉니다.

```
aws s3control list-access-points-for-object-lambda --account-id 111122223333
```

2. 명령 출력을 검토하여 사용할 객체 Lambda 액세스 포인트 ARN을 찾습니다. 이전 명령의 출력이 다음 예제와 비슷해야 합니다.

```
{
    "ObjectLambdaAccessPointList": [
        {
            "Name": "my-object-lambda-ap",
            "ObjectLambdaAccessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/my-object-lambda-ap"
        },
        ...
    ]
}
```

S3 버킷 객체 Lambda 액세스 포인트에 버킷 스타일 별칭을 사용하는 방법

객체 Lambda 액세스 포인트를 생성할 때 Amazon S3는 객체 Lambda 액세스 포인트에 대한 고유한 별칭을 자동으로 생성합니다. 액세스 포인트 데이터 영역 작업 요청에 Amazon S3 버킷 이름이나 객체 Lambda 액세스 포인트 Amazon 리소스 이름(ARN) 대신 이 별칭을 사용할 수 있습니다. 해당 작업의 목록은 액세스 포인트 호환성을(를) 참조하십시오.

객체 Lambda 액세스 포인트 별칭 이름은 Amazon S3 버킷과 동일한 네임스페이스 내에서 생성됩니다. 이 별칭 이름은 자동으로 생성되며 변경할 수 없습니다. 기존 객체 Lambda 액세스 포인트의 경우 사용할 별칭이 자동으로 할당됩니다. 객체 Lambda 액세스 포인트 별칭 이름은 유효한 Amazon S3 버킷 이름의 모든 요구 사항을 충족하며 다음의 파트로 구성됩니다.

Object Lambda Access Point name prefix-*metadata*--ol-s3

> **Note**
>
> `--ol-s3` 접미사는 객체 Lambda 액세스 포인트 별칭 이름용으로 예약되어 있으며 버킷 또는 객체 Lambda 액세스 포인트 이름에는 사용할 수 없습니다. Amazon S3 버킷 이름 지정 규칙에 대한 자세한 내용은 범용 버킷 이름 지정 규칙 섹션을 참조하십시오.

다음 예시는 이름이 *my-object-lambda-access-point*인 객체 Lambda 액세스 포인트에 대한 ARN 및 객체 Lambda 액세스 포인트 별칭을 보여줍니다.

- ARN-`arn:aws:s3-object-lambda:`*region*`:`*account-id*`:accesspoint/`*my-object-lambda-access-point*
- 객체 Lambda 액세스 포인트 별칭 - *my-object-lambda-acc-1a4n8yjrb3kda96f67zwrwiiuse1a*`--ol-s3`

객체 Lambda 액세스 포인트를 사용하면 코드를 크게 변경하지 않고도 객체 Lambda 액세스 포인트 별칭 이름을 사용할 수 있습니다.

객체 Lambda 액세스 포인트를 삭제하면 객체 Lambda 액세스 포인트 별칭 이름이 비활성화되고 프로비저닝이 해제됩니다.

객체 Lambda 액세스 포인트에 대한 별칭을 찾는 방법

S3 콘솔 사용

콘솔을 사용하여 객체 Lambda 액세스 포인트에 대한 별칭을 찾는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 객체 Lambda 액세스 포인트(Object Lambda Access Points)를 선택합니다.
3. 사용하려는 객체 Lambda 액세스 포인트를 위해 객체 Lambda 액세스 포인트 별칭 값을 복사합니다.

AWS CLI 사용

객체 Lambda 액세스 포인트를 생성하면 Amazon S3는 다음 예시 명령과 같이 객체 Lambda 액세스 포인트 별칭 이름을 자동으로 생성합니다. 이 명령을 실행하려면 *user input placeholders*를 사용

자의 정보로 대체합니다. AWS CLI를 사용하여 객체 Lambda 액세스 포인트를 생성하는 방법에 대한 자세한 내용은 AWS CLI를 사용하여 객체 Lambda 액세스 포인트 생성 섹션을 참조하십시오.

```
aws s3control create-access-point-for-object-lambda --account-id 111122223333 --
name my-object-lambda-access-point --configuration file://my-olap-configuration.json
{
    "ObjectLambdaAccessPointArn": "arn:aws:s3:region:111122223333:accesspoint/my-
access-point",
    "Alias": {
        "Value": "my-object-lambda-acc-1a4n8yjrb3kda96f67zwrwiiuse1a--ol-s3",
        "Status": "READY"
    }
}
```

생성된 객체 Lambda 액세스 포인트 별칭 이름에는 두 개의 필드가 있습니다.

- `Value` 필드는 객체 Lambda 액세스 포인트의 별칭 값입니다.
- `Status` 필드는 객체 Lambda 액세스 포인트 별칭의 상태입니다. 상태가 `PROVISIONING`인 경우 Amazon S3가 객체 Lambda 액세스 포인트 별칭을 프로비저닝하고 있으며 아직 별칭을 사용할 준비가 되지 않았다는 뜻입니다. 상태가 `READY`인 경우 객체 Lambda 액세스 포인트 별칭이 성공적으로 프로비저닝되어 사용할 준비가 된 것입니다.

REST API의 `ObjectLambdaAccessPointAlias` 데이터 유형에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 CreateAccessPointForObjectLambda 및 ObjectLambdaAccessPointAlias 섹션을 참조하십시오.

객체 Lambda 액세스 포인트 별칭을 사용하는 방법

액세스 포인트 호환성에 나열된 작업에 Amazon S3 버킷 이름 대신 객체 Lambda 액세스 포인트 별칭을 사용할 수 있습니다.

`get-bucket-location` 명령에 대한 다음 AWS CLI 예시에서는 버킷의 액세스 포인트 별칭을 사용하여 버킷이 있는 AWS 리전을 반환합니다. 이 명령을 실행하려면 *`user input placeholders`*를 사용자의 정보로 대체합니다.

```
aws s3api get-bucket-location --bucket my-object-lambda-
acc-w7i37nq6xuzgax3jw3oqtifiusw2a--ol-s3

{
    "LocationConstraint": "us-west-2"
```

```
}
```

요청의 객체 Lambda 액세스 포인트 별칭이 유효하지 않은 경우 `InvalidAccessPointAliasError` 오류 코드가 반환됩니다. `InvalidAccessPointAliasError`에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 오류 코드 목록을 참조하십시오.

객체 Lambda 액세스 포인트 별칭의 제한 사항은 액세스 포인트 별칭의 제한 사항과 동일합니다. 액세스 포인트 별칭의 제한 사항에 대한 자세한 내용은 액세스 포인트 별칭 제한 사항 섹션을 참조하십시오.

S3 객체 Lambda 액세스 포인트에 대한 보안 고려 사항

Amazon S3 객체 Lambda를 사용하면 AWS Lambda의 규모와 유연성을 컴퓨팅 플랫폼으로 사용하여 Amazon S3에서 나오는 데이터에 대해 사용자 지정 변환을 수행할 수 있습니다. S3와 Lambda는 기본적으로 보안을 유지하지만 이 보안을 유지하기 위해 Lambda 함수 작성자가 특별히 고려해야 할 것이 있습니다. S3 객체 Lambda를 사용하려면 모든 액세스가 인증된 보안 주체(익명 액세스 불가) 및 HTTPS를 통해 이루어져야 합니다.

보안 위험을 완화하기 위해 다음 작업을 수행하는 것이 좋습니다.

- Lambda 실행 역할의 범위를 가능한 가장 작은 권한 집합으로 한정합니다.
- 가능하면 Lambda 함수가 제공된 미리 서명된 URL을 통해 Amazon S3에 액세스하도록 합니다.

IAM 정책 구성

S3 액세스 포인트는 리소스, 사용자 또는 기타 조건별로 액세스 포인트 사용을 제어할 수 있는 AWS Identity and Access Management(IAM) 리소스 정책을 지원합니다. 자세한 내용은 객체 Lambda 액세스 포인트에 대한 IAM 정책 구성 섹션을 참조하세요.

암호화 동작

객체 Lambda 액세스 포인트에는 Amazon S3와 AWS Lambda가 모두 사용되기 때문에 암호화 동작에 차이가 있습니다. 기본 S3 암호화 동작에 대한 자세한 내용은 Amazon S3 버킷에 대한 기본 서버 측 암호화 동작 설정 섹션을 참조하십시오.

- 객체 Lambda 액세스 포인트와 함께 S3 서버 측 암호화를 사용하는 경우 객체는 Lambda로 전송되기 전에 해독됩니다. Lambda로 전송된 객체는 암호화되지 않은 상태로 처리됩니다(`GET` 또는 `HEAD` 요청의 경우).

- 암호화 키의 로깅을 방지하기 위해 S3는 고객이 제공한 키(SSE-C)를 사용하여 서버 측 암호화를 통해 암호화된 객체에 대한 GET 및 HEAD 요청을 거부합니다. 그러나 Lambda 함수에서 클라이언트가 제공한 키에 액세스할 수 있는 경우에는 여전히 이러한 객체를 검색할 수 있습니다.
- 객체 Lambda 액세스 포인트와 함께 S3 클라이언트측 암호화를 사용하는 경우 Lambda가 객체를 해독하고 다시 암호화할 수 있도록 암호화 키에 액세스할 수 있는지 확인해야 합니다.

액세스 포인트

S3 객체 Lambda는 객체 Lambda 액세스 포인트와 지원 액세스 포인트라고도 하는 표준 S3 액세스 포인트라는 두 가지 액세스 포인트를 사용합니다. 객체 Lambda 액세스 포인트에 요청을 수행할 때 S3는 S3 객체 Lambda 구성에 따라 사용자를 대신하여 Lambda를 호출하거나 지원 액세스 포인트에 요청을 위임합니다. 요청에 대해 Lambda가 호출되면 S3는 지원 액세스 지점을 통해 사용자를 대신하여 객체에 대해 미리 서명된 URL을 생성합니다. Lambda 함수가 호출될 때 이 URL을 입력으로 수신합니다.

S3를 직접 호출하는 대신 이 미리 서명된 URL을 사용하여 원본 객체를 검색하도록 Lambda 함수를 설정할 수 있습니다. 이 모델을 사용하면 객체에 더 나은 보안 경계를 적용할 수 있습니다. S3 버킷 또는 S3 액세스 포인트를 통해 제한된 IAM 역할 또는 사용자 집합으로 직접 객체 액세스를 제한할 수 있습니다. 이 방법은 또한 호출자와 다른 사용 권한을 가진 잘못 구성된 함수가 의도치 않게 객체에 대한 액세스를 허용 또는 거부할 수 있는 혼동된 대리인 문제의 영향을 받지 않도록 Lambda 함수를 보호합니다.

객체 Lambda 액세스 포인트 퍼블릭 액세스

Amazon S3가 S3 객체 Lambda 요청을 완료하기 위해 사용자의 자격 증명을 인증해야 하기 때문에 S3 객체 Lambda는 익명 또는 퍼블릭 액세스를 허용하지 않습니다. 객체 Lambda 액세스 포인트를 통해 요청을 호출하는 경우, 구성된 Lambda 함수에 대한 `lambda:InvokeFunction` 권한이 필요합니다 마찬가지로 객체 Lambda 액세스 포인트를 통해 다른 API 작업을 호출할 때 필요한 `s3:*` 권한이 있어야 합니다.

이러한 권한이 없으면 Lambda를 호출하거나 S3에 위임하라는 요청이 HTTP 403(금지) 오류로 실패합니다. 모든 액세스는 인증된 보안 주체에 의해 이루어져야 합니다. 퍼블릭 액세스가 필요한 경우 Lambda@Edge를 대안으로 사용할 수 있습니다. 자세한 내용은 Amazon CloudFront 개발자 안내서에서 Lambda@Edge를 사용하여 엣지에서 사용자 지정을 참조하십시오.

객체 Lambda 액세스 포인트 IP 주소

`describe-managed-prefix-lists` 서브넷은 게이트웨이 가상 프라이빗 클라우드(VPC) 엔드포인트를 지원하며, VPC 엔드포인트의 라우팅 테이블과 관련됩니다. 객체 Lambda 액세스 포인트는 게이트웨이 VPC를 지원하지 않으므로 IP 범위가 누락되었습니다. 누락된 범위는 Amazon S3에 속하지만

게이트웨이 VPC 엔드포인트에서는 지원되지 않습니다. `describe-managed-prefix-lists`에 대한 자세한 내용은 Amazon EC2 API 참조의 DescribeManagedPrefixLists 및 AWS 일반 참조의 AWS IP 주소 범위를 참조하십시오.

객체 Lambda 액세스 포인트에 대한 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. 객체 Lambda 액세스 포인트의 선택적 리소스 정책이나 지원 액세스 포인트의 리소스 정책을 통해 액세스를 제어할 수 있습니다. 단계별 예제는 자습서: S3 객체 Lambda를 사용하여 애플리케이션의 데이터 변환 및 자습서: S3 객체 Lambda 및 Amazon Comprehend를 사용하여 PII 데이터 감지 및 수정 단원을 참조하십시오.

다음 4개 리소스에 객체 Lambda 액세스 포인트로 작업할 수 있는 권한을 부여해야 합니다.

- 사용자 또는 역할과 같은 IAM 자격 증명입니다. IAM 자격 증명 및 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM 자격 증명(사용자, 사용자 그룹 및 역할)을 참조하십시오.
- S3 버킷 또는 Amazon FSx for OpenZFS 볼륨과 같은 기본 데이터 소스에 연결된 표준 액세스 포인트입니다. 객체 Lambda 액세스 포인트로 작업할 때 이 표준 액세스 포인트를 지원 액세스 포인트라고 합니다.
- 객체 Lambda 액세스 포인트
- AWS Lambda 함수

> **Important**
>
> 정책을 저장하기 전에 AWS Identity and Access Management Access Analyzer의 보안 경고, 오류, 일반 경고 및 제안 사항을 해결해야 합니다. IAM Access Analyzer는 정책 확인은 실행하여 IAM 정책 문법 및 모범 사례에 대해 정책을 검증합니다. 이러한 확인은 결과를 생성하고 보안 모범 사례를 준수하고 작동하는 정책을 작성하는 데 도움이 되는 실행 가능한 권장 사항을 제공합니다.
>
> IAM Access Analyzer를 사용한 정책 검증에 대한 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer 정책 검증을 참조하십시오. IAM Access Analyzer에서 반환된 경고, 오류 및 제안 사항 목록을 보려면 IAM Access Analyzer 정책 확인 참조를 참조하십시오.

다음 정책 예시에서는 다음과 같은 리소스가 있다고 가정합니다.

- Amazon S3 버킷에는 다음과 같은 Amazon S3 리소스 이름(ARN)이 포함되어 있습니다.

`arn:aws:s3:::amzn-s3-demo-bucket1`

- 다음 ARN이 있는 이 버킷의 Amazon S3 Standard 액세스 포인트

`arn:aws:s3:us-east-1:111122223333:accesspoint/my-access-point`

- 다음 ARN이 있는 객체 Lambda 액세스 포인트:

`arn:aws:s3-object-lambda:us-east-1:111122223333:accesspoint/my-object-lambda-ap`

- 다음 ARN을 사용하는 AWS Lambda 함수

`arn:aws:lambda:us-east-1:111122223333:function:MyObjectLambdaFunction`

> **Note**
>
> 계정에서 Lambda 함수를 사용하는 경우 정책 문에 특정 함수 버전을 포함해야 합니다. 다음 예시 ARN에서 버전은 *1*로 표시됩니다.
> `arn:aws:lambda:us-east-1:111122223333:function:MyObjectLambdaFunction:1`
> Lambda는 버전 $LATEST에 IAM 정책을 추가하는 것을 지원하지 않습니다. Lambda 함수 버전에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 Lambda 함수 버전을 참조하십시오.

Example - 표준 액세스 지점에 액세스 제어를 위임하는 버킷 정책

다음 S3 버킷 정책 예는 버킷에 대한 액세스 제어를 버킷의 표준 액세스 지점에 위임합니다. 이 정책은 버킷 소유자의 계정이 소유한 모든 액세스 지점에 대한 전체 액세스를 허용합니다. 따라서 이 버킷에 대한 모든 액세스는 해당 액세스 지점에 연결된 정책에 의해 제어됩니다. 사용자는 액세스 지점을 통해서만 버킷에서 읽을 수 있으므로 액세스 지점을 통해서만 작업이 호출될 수 있습니다. 자세한 내용은 액세스 포인트에 액세스 제어 위임 섹션을 참조하세요.

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

```
            "arn:aws:s3:::amzn-s3-demo-bucket1",
            "arn:aws:s3:::amzn-s3-demo-bucket1/*"
        ],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointAccount" : "Bucket owner's account
 ID" }
        }
    }]
}
```

Example - 사용자에게 객체 Lambda 액세스 포인트를 사용하는 데 필요한 권한을 부여하는 IAM 정책

다음 IAM 정책은 사용자에게 Lambda 함수, 표준 액세스 포인트 및 객체 Lambda 액세스 포인트에 대한 권한을 부여합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowLambdaInvocation",
      "Action": [
        "lambda:InvokeFunction"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:lambda:us-
east-1:111122223333:function:MyObjectLambdaFunction:1",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "s3-object-lambda.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "AllowStandardAccessPointAccess",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:us-east-1:111122223333:accesspoint/my-access-point/*",
      "Condition": {
```

```
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "s3-object-lambda.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "AllowObjectLambdaAccess",
      "Action": [
        "s3-object-lambda:Get*",
        "s3-object-lambda:List*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3-object-lambda:us-east-1:111122223333:accesspoint/my-object-lambda-ap"
    }
  ]
}
```

Lambda 실행 역할에 대한 권한 활성화

객체 Lambda 액세스 포인트에 GET 요청이 있을 때 Lambda 함수는 S3 객체 Lambda 액세스 포인트에 데이터를 보낼 수 있는 권한이 필요합니다. 이 권한은 Lambda 함수의 실행 역할에 대한 `s3-object-lambda:WriteGetObjectResponse` 권한을 활성화하여 제공됩니다. 새로운 실행 역할을 생성하거나 기존 역할을 업데이트할 수 있습니다.

> **Note**
>
> GET 요청인 경우에만 함수에 `s3-object-lambda:WriteGetObjectResponse` 권한이 필요합니다.

IAM 콘솔에서 실행 역할을 생성하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 역할을 선택합니다.
3. 역할 생성을 선택합니다.
4. Common use cases(일반 사용 사례)에서 Lambda를 선택합니다.
5. 다음을 선택합니다.

6. Add permissions(권한 추가) 페이지에서 AWS 관리형 정책인 AmazonS3ObjectLambdaExecutionRolePolicy를 검색한 다음 정책 이름 옆의 확인란을 선택합니다.

   이 정책에는 `s3-object-lambda:WriteGetObjectResponse` 작업이 포함되어야 합니다.
7. 다음을 선택합니다.
8. Name, review, and create(이름, 검토 및 생성) 페이지에서 Role name(역할 이름)에 **`s3-object-lambda-role`**을 입력합니다.
9. (선택 사항) 이 역할에 대한 설명 및 태그를 추가합니다.
10. 역할 생성을 선택합니다.
11. 새로 생성된 **`s3-object-lambda-role`**을 Lambda 함수의 실행 역할로 적용합니다. 이 작업은 Lambda 콘솔에서 Lambda 함수를 생성하는 동안 또는 이후에 수행할 수 있습니다.

실행 역할에 대한 자세한 내용은 AWS Lambda 개발자 가이드의 Lambda 실행 역할을 참조하십시오.

객체 Lambda 액세스 포인트에 컨텍스트 키 사용

S3 객체 Lambda는 요청의 연결 서명과 관련된 `s3-object-lambda:TlsVersion` 또는 `s3-object-lambda:AuthType`과 같은 컨텍스트 키를 평가합니다. `s3:prefix`와 같은 다른 모든 컨텍스트 키는 Amazon S3에 의해 평가됩니다.

객체 Lambda 액세스 포인트 CORS 지원

브라우저로부터 요청을 받거나 요청에 `Origin` 헤더가 포함된 경우 S3 Object Lambda는 항상 `"AllowedOrigins":"*"` 헤더 필드를 추가합니다.

자세한 내용은 교차 오리진 리소스 공유(CORS) 사용 섹션을 참조하세요.

S3 객체 Lambda 액세스 포인트에 대한 Lambda 함수 작성

이 섹션에서는 Amazon S3 객체 Lambda 액세스 포인트에 사용할 AWS Lambda 함수를 작성하는 방법에 대해 자세히 설명합니다.

일부 S3 객체 Lambda 태스크 절차를 처음부터 끝까지 완전히 알아보려면 다음을 참조하십시오.

- 자습서: S3 객체 Lambda를 사용하여 애플리케이션의 데이터 변환
- 자습서: S3 객체 Lambda 및 Amazon Comprehend를 사용하여 PII 데이터 감지 및 수정
- Tutorial: Using S3 Object Lambda to dynamically watermark images as they are retrieved(자습서: S3 객체 Lambda를 사용하여 이미지를 검색할 때 동적으로 워터마크 지정)

주제

- Lambda에서 GetObject 요청 작업
- Lambda에서 HeadObject 요청 작업
- Lambda에서 ListObjects 요청 작업
- Lambda에서 ListObjectsV2 요청 작업
- 이벤트 컨텍스트 형식 및 사용법
- Range 및 partNumber 헤더 작업

Lambda에서 **GetObject** 요청 작업

이 섹션에서는 객체 Lambda 액세스 포인트가 GetObject에 대한 Lambda 함수를 호출하도록 구성되어 있다고 가정합니다. S3 객체 Lambda에는 Lambda 함수에서 GetObject 호출자에게 사용자 지정된 데이터 및 응답 헤더를 제공할 수 있는 Amazon S3 API 작업인 WriteGetObjectResponse가 포함되어 있습니다.

WriteGetObjectResponse는 처리 요구 사항에 따라 상태 코드, 응답 헤더 및 응답 본문에 대한 광범위한 제어 기능을 제공합니다. WriteGetObjectResponse를 사용하면 변환된 전체 객체, 변환된 객체의 일부 또는 애플리케이션의 컨텍스트에 따라 다른 응답으로 응답할 수 있습니다. 다음 섹션에서는 WriteGetObjectResponse API 작업을 사용하는 고유한 예를 보여줍니다.

- 예시 1: HTTP 상태 코드 403(금지)으로 응답
- 예제 2: 변환된 이미지로 응답
- 예제 3: 압축된 콘텐츠 스트리밍

예시 1: HTTP 상태 코드 403(금지)으로 응답

WriteGetObjectResponse를 사용하면 객체의 콘텐츠를 기반으로 HTTP 상태 코드 403(금지)으로 응답할 수 있습니다.

Java

```
package com.amazon.s3.objectlambda;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.events.S3ObjectLambdaEvent;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
```

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

import java.io.ByteArrayInputStream;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class Example1 {

    public void handleRequest(S3ObjectLambdaEvent event, Context context) throws
 Exception {
        AmazonS3 s3Client = AmazonS3Client.builder().build();

        // Check to see if the request contains all of the necessary information.
        // If it does not, send a 4XX response and a custom error code and message.
        // Otherwise, retrieve the object from S3 and stream it
        // to the client unchanged.
        var tokenIsNotPresent = !
event.getUserRequest().getHeaders().containsKey("requiredToken");
        if (tokenIsNotPresent) {
            s3Client.writeGetObjectResponse(new WriteGetObjectResponseRequest()
                    .withRequestRoute(event.outputRoute())
                    .withRequestToken(event.outputToken())
                    .withStatusCode(403)
                    .withContentLength(0L).withInputStream(new
 ByteArrayInputStream(new byte[0]))
                    .withErrorCode("MissingRequiredToken")
                    .withErrorMessage("The required token was not present in the
 request."));
            return;
        }

        // Prepare the presigned URL for use and make the request to S3.
        HttpClient httpClient = HttpClient.newBuilder().build();
        var presignedResponse = httpClient.send(
                HttpRequest.newBuilder(new URI(event.inputS3Url())).GET().build(),
                HttpResponse.BodyHandlers.ofInputStream());

        // Stream the original bytes back to the caller.
        s3Client.writeGetObjectResponse(new WriteGetObjectResponseRequest()
                .withRequestRoute(event.outputRoute())
                .withRequestToken(event.outputToken())
                .withInputStream(presignedResponse.body()));
```

```
    }
}
```

Python

```
import boto3
import requests

def handler(event, context):
    s3 = boto3.client('s3')

    """
    Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    should be delivered and contains a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    The 'userRequest' object has information related to the user who made this
 'GetObject' request to
    S3 Object Lambda.
    """
    get_context = event["getObjectContext"]
    user_request_headers = event["userRequest"]["headers"]

    route = get_context["outputRoute"]
    token = get_context["outputToken"]
    s3_url = get_context["inputS3Url"]

    # Check for the presence of a 'CustomHeader' header and deny or allow based on
 that header.
    is_token_present = "SuperSecretToken" in user_request_headers

    if is_token_present:
        # If the user presented our custom 'SuperSecretToken' header, we send the
 requested object back to the user.
        response = requests.get(s3_url)
        s3.write_get_object_response(RequestRoute=route, RequestToken=token,
 Body=response.content)
    else:
        # If the token is not present, we send an error back to the user.
        s3.write_get_object_response(RequestRoute=route, RequestToken=token,
 StatusCode=403,
        ErrorCode="NoSuperSecretTokenFound", ErrorMessage="The request was not
 secret enough.")
```

```
    # Gracefully exit the Lambda function.
    return { 'status_code': 200 }
```

Node.js

```
const { S3 } = require('aws-sdk');
const axios = require('axios').default;

exports.handler = async (event) => {
    const s3 = new S3();

    // Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    // should be delivered and contains a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    // The 'userRequest' object has information related to the user who made this
 'GetObject' request to S3 Object Lambda.
    const { userRequest, getObjectContext } = event;
    const { outputRoute, outputToken, inputS3Url } = getObjectContext;

    // Check for the presence of a 'CustomHeader' header and deny or allow based on
 that header.
    const isTokenPresent = Object
        .keys(userRequest.headers)
        .includes("SuperSecretToken");

    if (!isTokenPresent) {
        // If the token is not present, we send an error back to the user. The
 'await' in front of the request
        // indicates that we want to wait for this request to finish sending before
 moving on.
        await s3.writeGetObjectResponse({
            RequestRoute: outputRoute,
            RequestToken: outputToken,
            StatusCode: 403,
            ErrorCode: "NoSuperSecretTokenFound",
            ErrorMessage: "The request was not secret enough.",
        }).promise();
    } else {
        // If the user presented our custom 'SuperSecretToken' header, we send the
 requested object back to the user.
        // Again, note the presence of 'await'.
```

```
        const presignedResponse = await axios.get(inputS3Url);
        await s3.writeGetObjectResponse({
            RequestRoute: outputRoute,
            RequestToken: outputToken,
            Body: presignedResponse.data,
        }).promise();
    }

    // Gracefully exit the Lambda function.
    return { statusCode: 200 };
}
```

예제 2: 변환된 이미지로 응답

이미지 변환을 수행할 때는 소스 객체의 전체 바이트가 준비되어야 처리를 시작할 수 있습니다. 이 경우 `WriteGetObjectResponse`는 요청한 애플리케이션에 전체 객체를 한 번의 호출로 반환합니다.

Java

```
package com.amazon.s3.objectlambda;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.events.S3ObjectLambdaEvent;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.WriteGetObjectResponseRequest;

import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.awt.Image;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class Example2 {

    private static final int HEIGHT = 250;
    private static final int WIDTH = 250;
```

```
    public void handleRequest(S3ObjectLambdaEvent event, Context context) throws
 Exception {
        AmazonS3 s3Client = AmazonS3Client.builder().build();
        HttpClient httpClient = HttpClient.newBuilder().build();

        // Prepare the presigned URL for use and make the request to S3.
        var presignedResponse = httpClient.send(
                HttpRequest.newBuilder(new URI(event.inputS3Url())).GET().build(),
                HttpResponse.BodyHandlers.ofInputStream());

        // The entire image is loaded into memory here so that we can resize it.
        // Once the resizing is completed, we write the bytes into the body
        // of the WriteGetObjectResponse request.
        var originalImage = ImageIO.read(presignedResponse.body());
        var resizingImage = originalImage.getScaledInstance(WIDTH, HEIGHT,
 Image.SCALE_DEFAULT);
        var resizedImage = new BufferedImage(WIDTH, HEIGHT,
 BufferedImage.TYPE_INT_RGB);
        resizedImage.createGraphics().drawImage(resizingImage, 0, 0, WIDTH, HEIGHT,
 null);

        var baos = new ByteArrayOutputStream();
        ImageIO.write(resizedImage, "png", baos);

        // Stream the bytes back to the caller.
        s3Client.writeGetObjectResponse(new WriteGetObjectResponseRequest()
                .withRequestRoute(event.outputRoute())
                .withRequestToken(event.outputToken())
                .withInputStream(new ByteArrayInputStream(baos.toByteArray())));
    }
}
```

Python

```
import boto3
import requests
import io
from PIL import Image

def handler(event, context):
    """
    Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
```

```
    should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    The 'userRequest' object has information related to the user who made this
 'GetObject' request to
    S3 Object Lambda.
    """
    get_context = event["getObjectContext"]
    route = get_context["outputRoute"]
    token = get_context["outputToken"]
    s3_url = get_context["inputS3Url"]

    """
    In this case, we're resizing .png images that are stored in S3 and are
 accessible through the presigned URL
    'inputS3Url'.
    """
    image_request = requests.get(s3_url)
    image = Image.open(io.BytesIO(image_request.content))
    image.thumbnail((256,256), Image.ANTIALIAS)

    transformed = io.BytesIO()
    image.save(transformed, "png")

    # Send the resized image back to the client.
    s3 = boto3.client('s3')
    s3.write_get_object_response(Body=transformed.getvalue(), RequestRoute=route,
 RequestToken=token)

    # Gracefully exit the Lambda function.
    return { 'status_code': 200 }
```

Node.js

```
const { S3 } = require('aws-sdk');
const axios = require('axios').default;
const sharp = require('sharp');

exports.handler = async (event) => {
    const s3 = new S3();

    // Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
```

```
    // should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    const { getObjectContext } = event;
    const { outputRoute, outputToken, inputS3Url } = getObjectContext;

    // In this case, we're resizing .png images that are stored in S3 and are
 accessible through the presigned URL
    // 'inputS3Url'.
    const { data } = await axios.get(inputS3Url, { responseType: 'arraybuffer' });

    // Resize the image.
    const resized = await sharp(data)
        .resize({ width: 256, height: 256 })
        .toBuffer();

    // Send the resized image back to the client.
    await s3.writeGetObjectResponse({
        RequestRoute: outputRoute,
        RequestToken: outputToken,
        Body: resized,
    }).promise();

    // Gracefully exit the Lambda function.
    return { statusCode: 200 };
}
```

예제 3: 압축된 콘텐츠 스트리밍

객체를 압축할 때 압축된 데이터는 점진적으로 생성됩니다. 따라서 `WriteGetObjectResponse` 요청을 사용하여 압축된 데이터가 준비되는 즉시 압축된 데이터를 반환할 수 있습니다. 이 예제에서 볼 수 있듯이, 완료된 변환의 길이를 알아야 할 필요가 없습니다.

Java

```
package com.amazon.s3.objectlambda;

import com.amazonaws.services.lambda.runtime.events.S3ObjectLambdaEvent;
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.WriteGetObjectResponseRequest;
```

```
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class Example3 {

    public void handleRequest(S3ObjectLambdaEvent event, Context context) throws
 Exception {
        AmazonS3 s3Client = AmazonS3Client.builder().build();
        HttpClient httpClient = HttpClient.newBuilder().build();

        // Request the original object from S3.
        var presignedResponse = httpClient.send(
                HttpRequest.newBuilder(new URI(event.inputS3Url())).GET().build(),
                HttpResponse.BodyHandlers.ofInputStream());

        // Consume the incoming response body from the presigned request,
        // apply our transformation on that data, and emit the transformed bytes
        // into the body of the WriteGetObjectResponse request as soon as they're
 ready.
        // This example compresses the data from S3, but any processing pertinent
        // to your application can be performed here.
        var bodyStream = new GZIPCompressingInputStream(presignedResponse.body());

        // Stream the bytes back to the caller.
        s3Client.writeGetObjectResponse(new WriteGetObjectResponseRequest()
                .withRequestRoute(event.outputRoute())
                .withRequestToken(event.outputToken())
                .withInputStream(bodyStream));
    }

}
```

Python

```
import boto3
import requests
import zlib
from botocore.config import Config


"""
```

```
A helper class to work with content iterators. Takes an interator and compresses the
 bytes that come from it. It
implements 'read' and '__iter__' so that the SDK can stream the response.
"""
class Compress:
    def __init__(self, content_iter):
        self.content = content_iter
        self.compressed_obj = zlib.compressobj()

    def read(self, _size):
        for data in self.__iter__()
            return data

    def __iter__(self):
        while True:
            data = next(self.content)
            chunk = self.compressed_obj.compress(data)
            if not chunk:
                break

            yield chunk

        yield self.compressed_obj.flush()


def handler(event, context):
    """
    Setting the 'payload_signing_enabled' property to False allows us to send a
 streamed response back to the client.
    in this scenario, a streamed response means that the bytes are not buffered into
 memory as we're compressing them,
    but instead are sent straight to the user.
    """
    my_config = Config(
        region_name='eu-west-1',
        signature_version='s3v4',
        s3={
            "payload_signing_enabled": False
        }
    )
    s3 = boto3.client('s3', config=my_config)

    """
```

```
    Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    The 'userRequest' object has information related to the user who made this
 'GetObject' request to S3 Object Lambda.
    """
    get_context = event["getObjectContext"]
    route = get_context["outputRoute"]
    token = get_context["outputToken"]
    s3_url = get_context["inputS3Url"]

    # Compress the 'get' request stream.
    with requests.get(s3_url, stream=True) as r:
        compressed = Compress(r.iter_content())

        # Send the stream back to the client.
        s3.write_get_object_response(Body=compressed, RequestRoute=route,
 RequestToken=token, ContentType="text/plain",
                                     ContentEncoding="gzip")

    # Gracefully exit the Lambda function.
    return {'status_code': 200}
```

Node.js

```
const { S3 } = require('aws-sdk');
const axios = require('axios').default;
const zlib = require('zlib');

exports.handler = async (event) => {
    const s3 = new S3();

    // Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    // should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    const { getObjectContext } = event;
    const { outputRoute, outputToken, inputS3Url } = getObjectContext;

    // Download the object from S3 and process it as a stream, because it might be a
 huge object and we don't want to
```

```
    // buffer it in memory. Note the use of 'await' because we want to wait for
 'writeGetObjectResponse' to finish
    // before we can exit the Lambda function.
    await axios({
        method: 'GET',
        url: inputS3Url,
        responseType: 'stream',
    }).then(
        // Gzip the stream.
        response => response.data.pipe(zlib.createGzip())
    ).then(
        // Finally send the gzip-ed stream back to the client.
        stream => s3.writeGetObjectResponse({
            RequestRoute: outputRoute,
            RequestToken: outputToken,
            Body: stream,
            ContentType: "text/plain",
            ContentEncoding: "gzip",
        }).promise()
    );

    // Gracefully exit the Lambda function.
    return { statusCode: 200 };
}
```

> **Note**
>
> S3 객체 Lambda를 사용할 때는 `WriteGetObjectResponse` 요청을 통해 최대 60초간 호출자에게 전체 응답을 전송할 수 있지만 실제 사용 가능한 시간은 더 적을 수 있습니다. 예를 들어 Lambda 함수 제한 시간이 60초 미만일 수 있습니다. 호출자의 제한 시간이 더 엄격한 경우도 있을 수 있습니다.

원래 호출자가 HTTP 상태 코드500(내부 서버 오류) 외의 응답을 수신하려면 `WriteGetObjectResponse` 호출이 완료되어야 합니다. 예외나 다른 이유로 `WriteGetObjectResponse` API 작업이 호출되기 전에 Lambda 함수가 반환되면 원래 호출자가 500(내부 서버 오류) 응답을 수신합니다. 응답을 완료하는 데 걸리는 시간 동안 예외가 발생하면 호출자가 잘린 응답을 수신하게 됩니다. Lambda 함수가 `WriteGetObjectResponse` API 호출에서 HTTP 상태 코드 200(OK) 응답을 수신하는 경우 원래 호출자의 전체 요청이 전송된 것입니다. Lambda 함수의 응답은 예외가 발생했는지 여부에 관계없이 S3 객체 Lambda에 의해 무시됩니다.

`WriteGetObjectResponse` API 작업을 호출할 때 Amazon S3에는 이벤트 컨텍스트의 라우팅 및 요청 토큰이 필요합니다. 자세한 내용은 이벤트 컨텍스트 형식 및 사용법 섹션을 참조하세요.

라우팅 및 요청 토큰 파라미터는 원래 호출자와 `WriteGetObjectResult` 응답을 연결하는 데 필요합니다. 500(내부 서버 오류) 응답이 수신되는 경우 다시 시도하는 것이 항상 적절하지만 요청 토큰은 1회 사용 토큰이기 때문에 이후 사용을 시도할 경우 HTTP 상태 코드 400(잘못된 요청) 응답이 발생할 수 있습니다. 라우팅 및 요청 토큰을 사용한 `WriteGetObjectResponse` 호출을 호출된 Lambda 함수에서 수행할 필요는 없지만 동일한 계정의 자격 증명으로 수행되어야 합니다. 또한 Lambda 함수가 실행을 완료하기 전에 호출이 완료되어야 합니다.

Lambda에서 **`HeadObject`** 요청 작업

이 섹션에서는 객체 Lambda 액세스 포인트가 `HeadObject`에 대한 Lambda 함수를 호출하도록 구성되어 있다고 가정합니다. Lambda는 `headObjectContext`라는 키가 포함된 JSON 페이로드를 수신합니다. 컨텍스트 내에는 `HeadObject`의 지원 액세스 지점에 대해 미리 서명된 URL인 `inputS3Url`이라는 하나의 속성이 있습니다.

미리 서명된 URL에는 지정된 경우 다음과 같은 속성이 포함됩니다.

- `versionId`(쿼리 파라미터에 포함됨)
- `requestPayer`(`x-amz-request-payer` 헤더에 포함됨)
- `expectedBucketOwner`(`x-amz-expected-bucket-owner` 헤더에 포함됨)

다른 속성은 미리 서명되지 않으므로 포함되지 않습니다. `userRequest` 헤더에 없는 미리 서명된 URL을 호출할 때 헤더로 전송된 서명되지 않은 옵션을 요청에 수동으로 추가할 수 있습니다. 서버 측 암호화 옵션은 `HeadObject`에 지원되지 않습니다.

요청 구문 URI 파라미터는 Amazon Simple Storage Service API 참조의 `HeadObject` 섹션을 참조하세요.

다음 예는 `HeadObject`에 대한 Lambda JSON 입력 페이로드를 보여줍니다.

```
{
  "xAmzRequestId": "requestId",
  "**headObjectContext**": {
    "**inputS3Url**": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/example?X-Amz-Security-Token=<snip>"
  },
  "configuration": {
```

```
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
        "payload": "{}"
    },
    "userRequest": {
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
    },
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "principalId",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
             "attributes": {
             "mfaAuthenticated": "false",
             "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
        },
        "sessionIssuer": {
             "type": "Role",
             "principalId": "principalId",
             "arn": "arn:aws:iam::111122223333:role/Admin",
             "accountId": "111122223333",
             "userName": "Admin"
             }
        }
     },
    "protocolVersion": "1.00"
}
```

Lambda 함수는 HeadObject 호출에 대해 반환될 헤더와 값이 포함된 JSON 객체를 반환해야 합니다.

다음 예는 HeadObject에 대한 Lambda 응답 JSON의 구조를 보여줍니다.

```
{
```

```
    "statusCode": <number>; // Required
    "errorCode": <string>;
    "errorMessage": <string>;
    "headers": {
        "Accept-Ranges": <string>,
        "x-amz-archive-status": <string>,
        "x-amz-server-side-encryption-bucket-key-enabled": <boolean>,
        "Cache-Control": <string>,
        "Content-Disposition": <string>,
        "Content-Encoding": <string>,
        "Content-Language": <string>,
        "Content-Length": <number>, // Required
        "Content-Type": <string>,
        "x-amz-delete-marker": <boolean>,
        "ETag": <string>,
        "Expires": <string>,
        "x-amz-expiration": <string>,
        "Last-Modified": <string>,
        "x-amz-missing-meta": <number>,
        "x-amz-object-lock-mode": <string>,
        "x-amz-object-lock-legal-hold": <string>,
        "x-amz-object-lock-retain-until-date": <string>,
        "x-amz-mp-parts-count": <number>,
        "x-amz-replication-status": <string>,
        "x-amz-request-charged": <string>,
        "x-amz-restore": <string>,
        "x-amz-server-side-encryption": <string>,
        "x-amz-server-side-encryption-customer-algorithm": <string>,
        "x-amz-server-side-encryption-aws-kms-key-id": <string>,
        "x-amz-server-side-encryption-customer-key-MD5": <string>,
        "x-amz-storage-class": <string>,
        "x-amz-tagging-count": <number>,
        "x-amz-version-id": <string>,
        <x-amz-meta-headers>: <string>, // user-defined metadata
        "x-amz-meta-meta1": <string>, // example of the user-defined metadata header,
 it will need the x-amz-meta prefix
        "x-amz-meta-meta2": <string>
        ...
    };
}
```

다음 예는 미리 서명된 URL을 사용하여 JSON을 반환하기 전에 필요에 따라 헤더 값을 수정하여 응답을 채우는 방법을 보여줍니다.

Python

```
import requests

def lambda_handler(event, context):
    print(event)

    # Extract the presigned URL from the input.
    s3_url = event["headObjectContext"]["inputS3Url"]

    # Get the head of the object from S3.
    response = requests.head(s3_url)

    # Return the error to S3 Object Lambda (if applicable).
    if (response.status_code >= 400):
        return {
            "statusCode": response.status_code,
            "errorCode": "RequestFailure",
            "errorMessage": "Request to S3 failed"
    }

    # Store the headers in a dictionary.
    response_headers = dict(response.headers)

    # This obscures Content-Type in a transformation, it is optional to add
    response_headers["Content-Type"] = ""

    # Return the headers to S3 Object Lambda.
    return {
        "statusCode": response.status_code,
        "headers": response_headers
        }
```

Lambda에서 **`ListObjects`** 요청 작업

이 섹션에서는 객체 Lambda 액세스 포인트가 `ListObjects`에 대한 Lambda 함수를 호출하도록 구성되어 있다고 가정합니다. Lambda는 `listObjectsContext`라는 새 객체가 포함된 JSON 페이로드를 수신합니다. `listObjectsContext`는 `ListObjects`에 대한 지원 액세스 지점의 미리 서명된 URL인 하나의 속성, `inputS3Url`을 포함합니다.

`GetObject` 및 `HeadObject`와는 달리 미리 서명된 URL에는 지정된 경우 다음과 같은 속성이 포함됩니다.

- 모든 쿼리 파라미터
- `requestPayer`(`x-amz-request-payer` 헤더에 포함됨)
- `expectedBucketOwner`(`x-amz-expected-bucket-owner` 헤더에 포함됨)

요청 구문 URI 파라미터는 Amazon Simple Storage Service API 참조의 `ListObjects` 섹션을 참조하세요.

> **Important**
>
> 애플리케이션을 개발할 때 최신 버전인 ListObjectsV2를 사용하는 것이 좋습니다. 이전 버전과의 호환성을 위해 Amazon S3는 `ListObjects`를 계속 지원합니다.

다음 예는 `ListObjects`에 대한 Lambda JSON 입력 페이로드를 보여줍니다.

```
{
    "xAmzRequestId": "requestId",
     "**listObjectsContext**": {
     "**inputS3Url**": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/?X-Amz-Security-Token=<snip>",
     },
    "configuration": {
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
        "payload": "{}"
    },
    "userRequest": {
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
    },
```

```
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "principalId",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "principalId",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            }
        }
    },
  "protocolVersion": "1.00"
}
```

Lambda 함수는 S3 객체 Lambda에서 반환되는 상태 코드, 목록 XML 결과 또는 오류 정보가 포함된 JSON 객체를 반환해야 합니다.

S3 객체 Lambda는 `listResultXml`을 처리하거나 검증하지 않고, 그 대신 `ListObjects` 호출자에게 전달합니다. `listBucketResult`의 경우, 지정된 유형에 따라 특정 속성이 있을 것으로 S3 객체 Lambda가 예상하여 이를 파싱할 수 없는 경우 예외를 발생시킵니다. `listResultXml` 및 `listBucketResult`는 동시에 제공할 수 없습니다.

다음 예는 미리 서명된 URL을 사용하여 Amazon S3를 호출하고 그 결과를 사용하여 오류 검사를 비롯한 응답을 채우는 방법을 보여줍니다.

Python

```
import requests
import xmltodict

def lambda_handler(event, context):
    # Extract the presigned URL from the input.
    s3_url = event["listObjectsContext"]["inputS3Url"]
```

```
    # Get the head of the object from Amazon S3.
    response = requests.get(s3_url)

    # Return the error to S3 Object Lambda (if applicable).
    if (response.status_code >= 400):
        error = xmltodict.parse(response.content)
        return {
            "statusCode": response.status_code,
            "errorCode": error["Error"]["Code"],
            "errorMessage": error["Error"]["Message"]
        }

    # Store the XML result in a dict.
    response_dict = xmltodict.parse(response.content)

    # This obscures StorageClass in a transformation, it is optional to add
    for item in response_dict['ListBucketResult']['Contents']:
        item['StorageClass'] = ""

    # Convert back to XML.
    listResultXml = xmltodict.unparse(response_dict)

    # Create response with listResultXml.
    response_with_list_result_xml = {
        'statusCode': 200,
        'listResultXml': listResultXml
    }

    # Create response with listBucketResult.
    response_dict['ListBucketResult'] =
 sanitize_response_dict(response_dict['ListBucketResult'])
    response_with_list_bucket_result = {
        'statusCode': 200,
        'listBucketResult': response_dict['ListBucketResult']
    }

    # Return the list to S3 Object Lambda.
    # Can return response_with_list_result_xml or response_with_list_bucket_result
    return response_with_list_result_xml

# Converting the response_dict's key to correct casing
def sanitize_response_dict(response_dict: dict):
    new_response_dict = dict()
```

```
    for key, value in response_dict.items():
        new_key = key[0].lower() + key[1:] if key != "ID" else 'id'
        if type(value) == list:
            newlist = []
            for element in value:
                if type(element) == type(dict()):
                    element = sanitize_response_dict(element)
                newlist.append(element)
            value = newlist
        elif type(value) == dict:
            value = sanitize_response_dict(value)
        new_response_dict[new_key] = value
    return new_response_dict
```

다음 예는 ListObjects에 대한 Lambda 응답 JSON의 구조를 보여줍니다.

```
{
  "statusCode": <number>; // Required
  "errorCode": <string>;
  "errorMessage": <string>;
  "listResultXml": <string>; // This can also be Error XML string in case S3 returned
 error response when calling the pre-signed URL

  "listBucketResult": {  // listBucketResult can be provided instead of listResultXml,
 however they can not both be provided in the JSON response
        "name": <string>,  // Required for 'listBucketResult'
        "prefix": <string>,
        "marker": <string>,
        "nextMarker": <string>,
        "maxKeys": <int>,   // Required for 'listBucketResult'
        "delimiter": <string>,
        "encodingType": <string>
        "isTruncated": <boolean>,  // Required for 'listBucketResult'
        "contents": [  {
            "key": <string>,  // Required for 'content'
            "lastModified": <string>,
            "eTag": <string>,
            "checksumAlgorithm": <string>,   // CRC32,  CRC32C,  SHA1,  SHA256
            "size": <int>,   // Required for 'content'
            "owner": {
                "displayName": <string>,  // Required for 'owner'
                "id": <string>,  // Required for 'owner'
```

```
            },
            "storageClass": <string>
            },
        ...
        ],
        "commonPrefixes": [  {
            "prefix": <string>   // Required for 'commonPrefix'
        },
        ...
        ],
    }
}
```

Lambda에서 **ListObjectsV2** 요청 작업

이 섹션에서는 객체 Lambda 액세스 포인트가 `ListObjectsV2`에 대한 Lambda 함수를 호출하도록 구성되어 있다고 가정합니다. Lambda는 `listObjectsV2Context`라는 새 객체가 포함된 JSON 페이로드를 수신합니다. `listObjectsV2Context`는 `ListObjectsV2`에 대한 지원 액세스 지점의 미리 서명된 URL인 하나의 속성, `inputS3Url`을 포함합니다.

`GetObject` 및 `HeadObject`와는 달리 미리 서명된 URL에는 지정된 경우 다음과 같은 속성이 포함됩니다.

- 모든 쿼리 파라미터
- `requestPayer`(`x-amz-request-payer` 헤더에 포함됨)
- `expectedBucketOwner`(`x-amz-expected-bucket-owner` 헤더에 포함됨)

요청 구문 URI 파라미터는 Amazon Simple Storage Service API 참조의 `ListObjectsV2` 섹션을 참조하십시오.

다음 예는 `ListObjectsV2`에 대한 Lambda JSON 입력 페이로드를 보여줍니다.

```
{
    "xAmzRequestId": "requestId",
     "**listObjectsV2Context**": {
     "**inputS3Url**": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/?list-type=2&X-Amz-Security-Token=<snip>",
     },
    "configuration": {
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
```

```
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
        "payload": "{}"
    },
    "userRequest": {
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
    },
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "principalId",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "principalId",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            }
        }
    },
  "protocolVersion": "1.00"
}
```

Lambda 함수는 S3 객체 Lambda에서 반환되는 상태 코드, 목록 XML 결과 또는 오류 정보가 포함된 JSON 객체를 반환해야 합니다.

S3 객체 Lambda는 `listResultXml`을 처리하거나 검증하지 않고, 그 대신 `ListObjectsV2` 호출자에게 전달합니다. `listBucketResult`의 경우, 지정된 유형에 따라 특정 속성이 있을 것으로

S3 객체 Lambda가 예상하여 이를 파싱할 수 없는 경우 예외를 발생시킵니다. `listResultXml` 및 `listBucketResult`는 동시에 제공할 수 없습니다.

다음 예는 미리 서명된 URL을 사용하여 Amazon S3를 호출하고 그 결과를 사용하여 오류 검사를 비롯한 응답을 채우는 방법을 보여줍니다.

Python

```
import requests
import xmltodict

def lambda_handler(event, context):
    # Extract the presigned URL from the input.
    s3_url = event["listObjectsV2Context"]["inputS3Url"]


    # Get the head of the object from Amazon S3.
    response = requests.get(s3_url)

    # Return the error to S3 Object Lambda (if applicable).
    if (response.status_code >= 400):
        error = xmltodict.parse(response.content)
        return {
            "statusCode": response.status_code,
            "errorCode": error["Error"]["Code"],
            "errorMessage": error["Error"]["Message"]
        }

    # Store the XML result in a dict.
    response_dict = xmltodict.parse(response.content)

    # This obscures StorageClass in a transformation, it is optional to add
    for item in response_dict['ListBucketResult']['Contents']:
        item['StorageClass'] = ""

    # Convert back to XML.
    listResultXml = xmltodict.unparse(response_dict)

    # Create response with listResultXml.
    response_with_list_result_xml = {
        'statusCode': 200,
        'listResultXml': listResultXml
    }
```

```
    # Create response with listBucketResult.
    response_dict['ListBucketResult'] =
 sanitize_response_dict(response_dict['ListBucketResult'])
    response_with_list_bucket_result = {
        'statusCode': 200,
        'listBucketResult': response_dict['ListBucketResult']
    }

    # Return the list to S3 Object Lambda.
    # Can return response_with_list_result_xml or response_with_list_bucket_result
    return response_with_list_result_xml

# Converting the response_dict's key to correct casing
def sanitize_response_dict(response_dict: dict):
    new_response_dict = dict()
    for key, value in response_dict.items():
        new_key = key[0].lower() + key[1:] if key != "ID" else 'id'
        if type(value) == list:
            newlist = []
            for element in value:
                if type(element) == type(dict()):
                    element = sanitize_response_dict(element)
                newlist.append(element)
            value = newlist
        elif type(value) == dict:
            value = sanitize_response_dict(value)
        new_response_dict[new_key] = value
    return new_response_dict
```

다음 예는 ListObjectsV2에 대한 Lambda 응답 JSON의 구조를 보여줍니다.

```
{
    "statusCode": <number>; // Required
    "errorCode": <string>;
    "errorMessage": <string>;
    "listResultXml": <string>; // This can also be Error XML string in case S3 returned
 error response when calling the pre-signed URL

    "listBucketResult": {  // listBucketResult can be provided instead of
 listResultXml, however they can not both be provided in the JSON response
        "name": <string>, // Required for 'listBucketResult'
```

```
        "prefix": <string>,
        "startAfter": <string>,
        "continuationToken": <string>,
        "nextContinuationToken": <string>,
        "keyCount": <int>, // Required for 'listBucketResult'
        "maxKeys": <int>, // Required for 'listBucketResult'
        "delimiter": <string>,
        "encodingType": <string>
        "isTruncated": <boolean>, // Required for 'listBucketResult'
        "contents": [ {
            "key": <string>, // Required for 'content'
            "lastModified": <string>,
            "eTag": <string>,
            "checksumAlgorithm": <string>, // CRC32, CRC32C, SHA1, SHA256
            "size": <int>, // Required for 'content'
            "owner": {
                "displayName": <string>, // Required for 'owner'
                "id": <string>, // Required for 'owner'
            },
            "storageClass": <string>
            },
            ...
        ],
        "commonPrefixes": [ {
            "prefix": <string> // Required for 'commonPrefix'
            },
        ...
        ],
    }
}
```

이벤트 컨텍스트 형식 및 사용법

Amazon S3 객체 Lambda는 AWS Lambda 함수에 전달된 이벤트에서 수행되는 요청에 대한 컨텍스트를 제공합니다. 다음 그림에서는 요청 예시를 보여줍니다. 필드에 대한 설명은 예시 뒤에 포함되어 있습니다.

```
{
    "xAmzRequestId": "requestId",
    "getObjectContext": {
        "inputS3Url": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/example?X-Amz-Security-Token=<snip>",
        "outputRoute": "io-use1-001",
```

```
        "outputToken": "OutputToken"
    },
    "configuration": {
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
        "payload": "{}"
    },
    "userRequest": {
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
    },
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "principalId",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "principalId",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            }
        }
    },
    "protocolVersion": "1.00"
}
```


요청에는 다음 필드가 포함됩니다.

- `xAmzRequestId` – 이 요청에 대한 Amazon S3 요청 ID입니다. 디버깅에 도움이 되도록 이 값을 기록하는 것이 좋습니다.
- `getObjectContext` – Amazon S3 및 S3 객체 Lambda 연결에 대한 입력 및 출력 세부 정보입니다.
  - `inputS3Url` – Amazon S3에서 원본 객체를 가져오는 데 사용할 수 있는 미리 서명된 URL입니다. URL은 원래 호출자의 ID를 사용하여 서명되며 URL이 사용될 때 해당 사용자의 권한이 적용됩니다. URL에 서명된 헤더가 있는 경우 Lambda 함수가 Amazon S3에 대한 호출에 이러한 헤더를 포함해야 합니다(`Host` 헤더 제외).
  - `outputRoute` - Lambda 함수에서 `WriteGetObjectResponse`를 호출할 때 S3 객체 Lambda URL에 추가되는 라우팅 토큰입니다.
  - `outputToken` – S3 객체 Lambda에서 원래 호출자와 일치하는 `WriteGetObjectResponse` 호출을 찾을 때 사용되는 불투명 토큰입니다.
- `configuration` – 객체 Lambda 액세스 포인트에 대한 구성 정보입니다.
  - `accessPointArn` – 이 요청을 수신한 객체 Lambda 액세스 포인트의 Amazon 리소스 이름(ARN)입니다.
  - `supportingAccessPointArn` – 객체 Lambda 액세스 포인트 구성에 지정된 지원 액세스 포인트의 ARN입니다.
  - `payload` – 객체 Lambda 액세스 포인트 구성에 적용되는 사용자 지정 데이터입니다. S3 객체 Lambda는 이 데이터를 불투명 한 문자열로 취급하므로 사용하기 전에 데이터를 디코딩해야 할 수도 있습니다.
- `userRequest` – S3 객체 Lambda에 대한 원래 호출에 대한 정보입니다.
  - `url` – 권한 부여 관련 쿼리 파라미터를 제외하고 S3 객체 Lambda에서 수신한 요청의 디코딩된 URL입니다.
  - `headers` – 권한 부여 관련 헤더를 제외하고 원래 호출의 HTTP 헤더와 해당 값을 포함하는 문자열에 대한 문자열 맵입니다. 동일한 헤더가 여러 번 나타나는 경우 동일한 헤더를 갖는 각 인스턴스의 값은 쉼표로 구분된 목록으로 결합됩니다. 이 맵에는 원래 헤더의 대/소문자가 유지됩니다.
- `userIdentity` – S3 객체 Lambda를 호출한 ID에 대한 세부 정보입니다. 자세한 내용은 AWS CloudTrail 사용 설명서의 추적을 위해 데이터 이벤트 로깅을 참조하세요.
  - `type` – 자격 증명의 유형입니다.
  - `accountId` – 자격 증명이 속한 AWS 계정 계정입니다.
  - `userName` – 호출을 수행한 자격 증명의 표시 이름입니다.
  - `principalId` – 호출을 수행한 자격 증명에 대한 고유 식별자입니다.
  - `arn` – 호출을 수행한 보안 주체의 ARN입니다. ARN의 마지막 섹션에는 호출을 수행한 사용자 또는 역할이 포함됩니다.

  - `sessionContext` – 임시 보안 자격 증명으로 요청이 이루어진 경우 이 요소는 해당 자격 증명을 위해 생성된 세션에 관한 정보를 제공합니다.
  - `invokedBy` – 요청을 수행한 AWS 서비스의 이름입니다(예: Amazon EC2 Auto Scaling 또는 AWS Elastic Beanstalk).
  - `sessionIssuer` – 임시 보안 자격 증명으로 요청이 이루어진 경우 이 요소는 자격 증명을 획득하는 방법에 관한 정보를 제공합니다.
- `protocolVersion` – 제공된 컨텍스트의 버전 ID입니다. 이 필드의 형식은 `{Major Version}.{Minor Version}`입니다. 마이너 버전 번호는 항상 2자리 숫자입니다. 필드의 의미 체계를 제거하거나 변경하면 메이저 버전 범프가 필요하며 활성 옵트인이 요구됩니다. Amazon S3는 언제든지 새 필드를 추가할 수 있으며, 이 시점에서 마이너 버전 범프가 발생할 수 있습니다. 소프트웨어 롤아웃의 특성으로 인해 한 번에 여러 개의 마이너 버전이 표시될 수 있습니다.

Range 및 partNumber 헤더 작업

Amazon S3 객체 Lambda에서 대용량 객체로 작업하는 경우 `Range` HTTP 헤더를 사용하면 객체에서 지정된 바이트 범위를 다운로드할 수 있습니다. 동일한 객체 내에서 서로 다른 바이트 범위를 가져오려면 Amazon S3에 대한 동시 연결을 사용하면 됩니다. 객체에서 지정된 부분에 대해 범위가 지정된 요청을 수행하는 `partNumber` 파라미터(1~10,000 사이의 정수)를 지정할 수도 있습니다.

`Range` 또는 `partNumber` 파라미터를 포함하는 요청을 처리할 수 있는 여러 방법이 있기 때문에 S3 객체 Lambda는 이러한 파라미터를 변환된 객체에 적용하지 않습니다. 대신 AWS Lambda 함수가 애플리케이션에 필요한 경우 이 기능을 구현해야 합니다.

`Range` 및 `partNumber` 파라미터를 S3 객체 Lambda와 함께 사용하려면 다음을 수행하십시오.

- 객체 Lambda 액세스 포인트 구성에서 이러한 파라미터를 활성화합니다.
- 이러한 파라미터를 포함하는 요청을 처리할 수 있는 Lambda 함수를 작성합니다.

다음 단계에서 이 작업을 수행하는 방법을 설명합니다.

1단계: 객체 Lambda 액세스 포인트 구성

기본적으로 객체 Lambda 액세스 포인트는 헤더 또는 쿼리 파라미터에 `Range` 또는 `partNumber` 파라미터가 포함되는 모든 `GetObject` 또는 `HeadObject` 요청에 HTTP 상태 코드 501(구현되지 않음) 오류로 응답합니다.

객체 Lambda 액세스 포인트가 이러한 요청을 수락하도록 설정하려면 객체 Lambda 액세스 포인트 구성의 `AllowedFeatures` 섹션에 `GetObject-Range`, `GetObject-PartNumber`, `HeadObject-`

Range 또는 HeadObject-PartNumber를 포함해야 합니다. 객체 Lambda 액세스 포인트 구성 업데이트에 대한 자세한 내용은 객체 Lambda 액세스 포인트 생성 섹션을 참조하십시오.

2단계: Lambda 함수에서 **Range** 또는 **partNumber** 처리 구현

객체 Lambda 액세스 포인트가 범위가 지정된 GetObject 또는 HeadObject 요청을 사용하여 Lambda 함수를 호출할 경우 Range 또는 partNumber 파라미터가 이벤트 컨텍스트에 포함됩니다. 다음 테이블에 설명된 대로 이벤트 컨텍스트에서 파라미터의 위치는 객체 Lambda 액세스 포인트에 대한 원래 요청에서 어떤 파라미터가 사용되었는지와 파라미터가 포함된 방식에 따라 달라집니다.

파라미터	이벤트 컨텍스트 위치
Range(헤더)	userRequest.headers.Range
Range(쿼리 파라미터)	userRequest.url (쿼리 파라미터 Range)
partNumber	userRequest.url (쿼리 파라미터 partNumber)

> **Important**
>
> 객체 Lambda 액세스 포인트에 제공된 미리 서명된 URL에는 원래 요청의 Range 또는partNumber 파라미터가 포함되어 있지 않습니다. AWS Lambda 함수에서 이러한 파라미터를 처리하는 방법은 다음 옵션을 참조하세요.

Range 또는 partNumber 값을 추출한 후 애플리케이션의 요구 사항에 따라 다음 방법 중 하나를 사용할 수 있습니다.

A. 요청된 **Range** 또는 **partNumber**를 변환된 객체에 매핑합니다(권장).

Range 또는 partNumber 요청을 처리하는 가장 신뢰할 수 있는 방법은 다음을 수행하는 것입니다.

- Amazon S3에서 전체 객체를 검색합니다.
- 객체를 변환합니다.
- 요청된 Range 또는 partNumber 파라미터를 변환된 객체에 적용합니다.

이 작업을 수행하려면 제공된 미리 서명된 URL을 사용하여 Amazon S3에서 전체 객체를 가져온 다음 필요에 따라 객체를 처리합니다. 이 방식으로 Range 파라미터를 처리하는 Lambda 함수의 예제에 대해서는 AWS 샘플 GitHub 리포지토리의 이 샘플을 참조하십시오.

B. 요청된 **Range**를 미리 서명된 URL에 매핑합니다.

경우에 따라 Lambda 함수가 요청된 Range를 미리 서명된 URL에 직접 매핑하여 Amazon S3에서 객체의 일부만 검색할 수 있습니다. 이 접근 방식은 변환이 다음 두 기준을 모두 충족하는 경우에만 적합합니다.

1. 변환 함수를 부분 객체 범위에 적용할 수 있습니다.
2. 변환 함수 앞 또는 뒤에 Range 파라미터를 사용하면 동일한 변환된 객체가 생성됩니다.

예를 들어 ASCII로 인코딩된 객체의 모든 문자를 대문자로 변환하는 변환 함수는 이전 두 조건을 모두 충족합니다. 변환을 객체의 일부에 적용할 수 있고 Range 파라미터를 변환 전에 적용하는 것과 파라미터를 변환 후에 적용하는 것이 동일한 결과를 얻습니다.

이와 달리 ASCII로 인코딩된 객체의 문자 순서를 반대로 바꾸는 함수는 이러한 조건을 충족하지 못합니다. 이러한 함수는 부분 객체 범위에 적용할 수 있기 때문에 조건 1을 충족합니다. 그러나 Range 파라미터를 변환 전에 적용하는 것과 파라미터를 변환 후에 적용하는 것이 서로 다른 결과를 얻기 때문에 조건 2를 충족하지 못합니다.

콘텐츠가 abcdefg인 객체의 처음 세 문자에 함수를 적용하는 요청을 가정합니다. Range 파라미터를 변환 전에 적용하면 abc만 가져오며 다음에 데이터의 순서를 반대로 바꾸면 cba를 반환합니다. 그러나 파라미터를 변환 후에 적용하면 함수가 전체 객체를 가져온 다음 데이터의 순서를 바꾸고 Range 파라미터를 적용하여 gfe를 반환합니다. 두 결과가 다르기 때문에 이 함수는 Amazon S3에서 객체를 가져올 때 Range 파라미터를 적용해서는 안 됩니다. 따라서 이 함수는 전체 객체를 가져와 변환을 수행한 다음 Range 파라미터를 적용해야만 합니다.

> **Warning**
>
> 많은 경우에 Range 파라미터를 미리 서명된 URL에 적용하면 Lambda 함수 또는 요청을 하는 클라이언트에서 예기치 않은 동작이 발생합니다. Amazon S3에서 부분 객체만 가져올 때 애플리케이션이 제대로 작동한다는 확신이 없는 경우 앞서 방법 A에서 설명한 대로 전체 객체를 가져오고 변환하는 것이 좋습니다.

애플리케이션이 B 방법에서 설명한 조건을 충족하는 경우 요청된 객체 범위만 가져온 다음 해당 범위에서 변환을 실행하여 AWS Lambda 함수를 간소화할 수 있습니다.

다음 Java 코드 예시는 다음 작업을 실행하는 방법을 설명한 것입니다.

- `GetObject` 요청에서 `Range` 헤더를 검색합니다.
- `Range` 헤더를 Lambda가 Amazon S3에서 요청된 범위를 가져오는 데 사용할 수 있는 미리 서명된 URL에 추가합니다.

```
private HttpRequest.Builder applyRangeHeader(ObjectLambdaEvent event,
 HttpRequest.Builder presignedRequest) {
    var header = event.getUserRequest().getHeaders().entrySet().stream()
            .filter(e -> e.getKey().toLowerCase(Locale.ROOT).equals("range"))
            .findFirst();

    // Add check in the query string itself.
    header.ifPresent(entry -> presignedRequest.header(entry.getKey(),
 entry.getValue()));
    return presignedRequest;
}
```

AWS 빌드 Lambda 함수 사용

AWS는 Amazon S3 객체 Lambda와 함께 사용하여 개인 식별 정보(PII)를 검색 후 교정하고 S3 객체의 압축을 풀 수 있는 미리 빌드된 AWS Lambda 함수를 제공합니다. 이러한 Lambda 함수는 AWS Serverless Application Repository에서 사용할 수 있습니다. 객체 Lambda 액세스 포인트를 생성할 때 AWS Management Console을 통해 이러한 함수를 선택할 수 있습니다.

AWS Serverless Application Repository에서 서버리스 애플리케이션을 배포하는 방법에 대한 자세한 내용은 AWS Serverless Application Repository 개발자 안내서에서 애플리케이션 배포를 참조하십시오.

> **Note**
>
> 다음 예시는 `GetObject` 요청에만 사용할 수 있습니다.

예시 1: PII 액세스 제어

이 Lambda 함수는 기계 학습을 사용하는 자연어 처리(NLP) 서비스인 Amazon Comprehend를 사용하여 텍스트에서 인사이트와 관계를 찾습니다. 이 함수는 Amazon S3 버킷의 문서에 있는 이름, 주소, 날짜, 신용 카드 번호, 주민등록번호와 같은 개인 식별 정보(PII)를 자동으로 교정합니다. 버킷에 PII가 포

함된 문서가 있는 경우 이러한 PII 엔터티 유형을 감지하고 권한이 없는 사용자에 대한 액세스를 제한하는 PII 액세스 제어 함수를 구성할 수 있습니다.

시작하려면 계정에 다음 Lambda 함수를 배포하고 객체 Lambda 액세스 포인트 구성에 함수의 Amazon 리소스 이름(ARN)을 추가하면 됩니다.

다음은 이 함수의 ARN 예시입니다.

```
arn:aws:serverlessrepo:us-east-1:111122223333:applications/
ComprehendPiiAccessControlS3ObjectLambda
```

다음 AWS Serverless Application Repository 링크를 사용하여 AWS Management Console에서 이 함수를 추가하거나 볼 수 있습니다. ComprehendPiiAccessControlS3ObjectLambda

GitHub에서 이 함수를 보려면 Amazon Comprehend S3 객체 Lambda에 관한 문서를 참조하세요.

예시 2: PII 교정

이 Lambda 함수는 기계 학습을 사용하는 자연어 처리(NLP) 서비스인 Amazon Comprehend를 사용하여 텍스트에서 인사이트와 관계를 찾습니다. 이 함수는 Amazon S3 버킷의 문서에서 이름, 주소, 날짜, 신용 카드 번호, 주민등록번호와 같은 개인 식별 정보(PII)를 자동으로 교정합니다.

버킷에 신용 카드 번호 또는 은행 계좌 정보와 같은 정보가 포함된 문서가 있는 경우 PII를 감지한 다음 PII 엔터티 유형이 교정되는 문서의 복사본을 반환하는 PII 교정 S3 객체 Lambda 함수를 구성할 수 있습니다.

시작하려면 계정에 다음 Lambda 함수를 배포하고 객체 Lambda 액세스 포인트 구성에 함수의 ARN을 추가하면 됩니다.

다음은 이 함수의 ARN 예시입니다.

```
arn:aws:serverlessrepo:us-east-1:111122223333::applications/
ComprehendPiiRedactionS3ObjectLambda
```

다음 AWS Serverless Application Repository 링크를 사용하여 AWS Management Console에서 이 함수를 추가하거나 볼 수 있습니다. ComprehendPiiRedactionS3ObjectLambda

GitHub에서 이 함수를 보려면 Amazon Comprehend S3 객체 Lambda에 관한 문서를 참조하십시오.

PII 교정에서의 일부 S3 객체 Lambda 태스크 절차를 처음부터 끝까지 완전히 알아보려면 자습서: S3 객체 Lambda 및 Amazon Comprehend를 사용하여 PII 데이터 감지 및 수정 페이지를 참조하십시오.

예제 3: 압축 풀기

S3ObjectLambdaDecompression Lambda 함수는 6가지 압축 파일 형식(bzip2, gzip, snappy, zlib, zstandard, ZIP) 중 하나로 Amazon S3에 저장되는 객체의 압축을 풀 수 있습니다.

시작하려면 계정에 다음 Lambda 함수를 배포하고 객체 Lambda 액세스 포인트 구성에 함수의 ARN을 추가하면 됩니다.

다음은 이 함수의 ARN 예시입니다.

```
arn:aws:serverlessrepo:us-east-1:111122223333::applications/S3ObjectLambdaDecompression
```

다음 AWS Serverless Application Repository 링크를 사용하여 AWS Management Console에서 이 함수를 추가하거나 볼 수 있습니다. S3ObjectLambdaDecompression

GitHub에서 이 함수를 보려면 S3 Object Lambda Decompression을 참조하십시오.

S3 객체 Lambda에 대한 모범 사례 및 지침

S3 객체 Lambda를 사용하는 경우 다음 모범 사례 및 지침에 따라 작업 및 성능을 최적화하십시오.

주제

- S3 객체 Lambda 작업
- S3 객체 Lambda와 관련하여 사용되는 AWS 서비스
- Range 및 partNumber 헤더
- expiry-date 변환
- AWS CLI 및 AWS SDK 작업

S3 객체 Lambda 작업

S3 객체 Lambda는 GET, LIST, HEAD 요청 처리만 지원합니다. 다른 요청은 AWS Lambda을 반환하지 않고 변환되지 않은 표준 API 응답을 반환합니다. AWS 계정에 대해 리전당 최대 1,000개의 객체 Lambda 액세스 포인트를 생성할 수 있습니다. 사용하는 AWS Lambda 함수는 객체 Lambda 액세스 포인트와 동일한 AWS 계정 및 리전에 있어야 합니다.

S3 객체 Lambda는 최대 60초 동안 호출자에게 전체 응답을 스트리밍할 수 있습니다. 함수에는 AWS Lambda 기본 할당량도 적용됩니다. 자세한 내용은 AWS Lambda 개발자 안내서에서 Lambda 할당량을 참조하십시오.

S3 객체 Lambda가 지정된 Lambda 함수를 호출할 때 지정된 Lambda 함수 또는 애플리케이션에 의해 Amazon S3에서 덮어쓰거나 삭제되는 데이터가 의도된 것이고 올바른지 확인할 책임은 사용자에게 있습니다.

S3 객체 Lambda는 객체에 대한 작업을 수행하는 데만 사용할 수 있습니다. S3 객체 Lambda를 사용하여 버킷 수정 또는 삭제와 같은 다른 Amazon S3 작업을 수행할 수 없습니다. 액세스 포인트를 지원하는 전체 S3 작업 목록은 S3 작업과의 액세스 포인트 호환성 단원을 참조하십시오.

이 목록 외에도 객체 Lambda 액세스 포인트는 `POST Object`, `CopyObject`(소스) 및 `SelectObjectContent` API 작업을 지원하지 않습니다.

S3 객체 Lambda와 관련하여 사용되는 AWS 서비스

S3 객체 Lambda는 Amazon S3, AWS Lambda, 선택적으로 원하는 다른 AWS 서비스를 연결하여 요청 애플리케이션과 관련된 객체를 전달합니다. S3 객체 Lambda와 함께 사용되는 모든 AWS 서비스는 각 서비스 수준에 관한 계약(SLA)에 따라 규제됩니다. 예를 들어, 한 AWS 서비스가 서비스 약정을 충족하지 못하는 경우 해당 서비스의 SLA에 명시된 서비스 크레딧을 받을 수 있습니다.

`Range` 및 **`partNumber`** 헤더

대용량 객체로 작업하는 경우 `Range` HTTP 헤더를 사용하면 객체에서 지정된 바이트 범위를 다운로드할 수 있습니다. `Range` 헤더를 사용하는 경우 요청은 객체의 지정된 부분만 가져옵니다. `partNumber` 헤더를 사용하여 객체에서 지정된 부분에 대해 범위가 지정된 요청을 수행할 수도 있습니다.

자세한 내용은 Range 및 partNumber 헤더 작업 섹션을 참조하세요.

`expiry-date` 변환

AWS Management Console의 객체 Lambda 액세스 포인트에서 변환된 객체를 열거나 다운로드할 수 있습니다. 이러한 객체는 만료되지 않은 상태여야 합니다. Lambda 함수가 객체의 `expiry-date`를 변환하는 경우 열거나 다운로드할 수 없는 만료된 객체를 볼 수 있습니다. 이 동작은 S3 Glacier Deep Retrieval 및 S3 Glacier Deep Archive 복원된 객체에만 적용됩니다.

AWS CLI 및 AWS SDK 작업

AWS Command Line Interface(AWS CLI) S3 하위 명령(`cp`, `mv`, `sync`) 및 AWS SDK for Java `TransferManager` 클래스 사용은 S3 객체 Lambda와 함께 사용할 수 없습니다.

S3 객체 Lambda 자습서

다음 자습서에서는 일부 S3 객체 Lambda 태스크를 처음부터 끝까지 수행하는 절차를 보여줍니다.

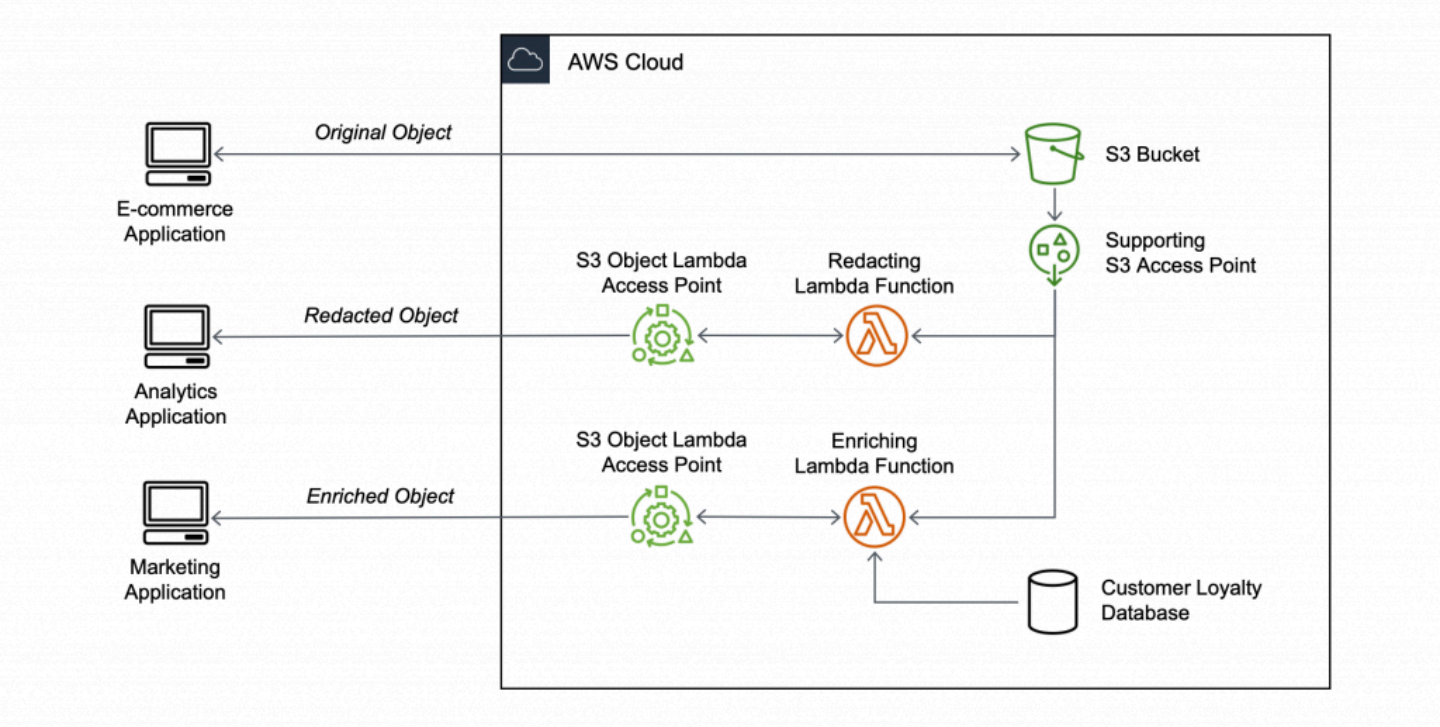
S3 객체 Lambda를 사용하여 자체 코드를 추가하여 S3에서 검색한 데이터를 애플리케이션으로 반환하기 전에 처리할 수 있습니다. 다음 자습서를 통해 기존 객체를 변경하거나 여러 데이터 사본을 유지하지 않고도 Amazon S3에서 데이터를 검색할 때 데이터를 수정할 수 있습니다. 첫 번째 자습서에서는 S3에서 검색된 객체를 수정하기 위해 S3 GET 요청에 AWS Lambda 함수를 추가하는 방법을 안내합니다. 두 번째 자습서에서는 애플리케이션으로 반환하기 전에 Amazon Comprehend에서 제공하는 사전 빌드된 Lambda 함수를 사용하여 S3에서 검색된 개인 식별 정보(PII)를 보호하는 방법을 보여줍니다. 세 번째 자습서에서는 Amazon S3에서 이미지를 검색할 때 S3 Object Lambda를 사용하여 이미지에 워터마크를 추가합니다.

- 자습서: S3 객체 Lambda를 사용하여 애플리케이션의 데이터 변환
- 자습서: S3 객체 Lambda 및 Amazon Comprehend를 사용하여 PII 데이터 감지 및 수정
- Tutorial: Using S3 Object Lambda to dynamically watermark images as they are retrieved(자습서: S3 객체 Lambda를 사용하여 이미지를 검색할 때 동적으로 워터마크 지정)

자습서: S3 객체 Lambda를 사용하여 애플리케이션의 데이터 변환

데이터를 Amazon S3 저장하면 여러 애플리케이션에서 사용하기 위해 데이터를 쉽게 공유할 수 있습니다. 그러나 각 애플리케이션에는 고유한 데이터 형식 요구 사항이 있을 수 있으며 특정 사용 사례에 따라 데이터를 수정하거나 처리해야 할 수 있습니다. 예를 들어 전자 상거래 애플리케이션에서 생성한 데이터 집합에는 개인 식별 정보(PII)가 포함될 수 있습니다. 분석을 위해 동일한 데이터가 처리되는 경우 이 PII가 필요하지 않으므로 수정해야 합니다. 그러나 마케팅 캠페인에 동일한 데이터 집합이 사용되는 경우 고객 충성도 데이터베이스의 정보와 같은 추가 세부 정보로 데이터를 보강해야 할 수 있습니다.

S3 객체 Lambda를 사용하여 자체 코드를 추가하여 S3에서 검색한 데이터를 애플리케이션으로 반환하기 전에 처리할 수 있습니다. 특히, AWS Lambda 함수를 구성하여 S3 객체 Lambda 액세스 포인트에 연결할 수 있습니다. 애플리케이션에서 S3 Object Lambda 액세스 포인트를 통해 표준 S3 GET 요청을 전송하면 지정된 Lambda 함수가 간접 호출되어 지원 S3 액세스 포인트를 통해 기본 데이터 소스에서 검색된 데이터를 처리합니다. 그런 다음 S3 객체 Lambda 액세스 포인트가 변환된 결과를 애플리케이션에 다시 반환합니다. 애플리케이션을 변경할 필요 없이 S3 객체 Lambda 데이터 변환을 특정 사용 사례에 맞게 조정하여 자체 사용자 지정 Lambda 함수를 작성 및 실행할 수 있습니다.



목표

이 자습서에서는 표준 S3 GET 요청에 사용자 지정 코드를 추가하여, 요청하는 클라이언트 또는 애플리케이션의 요구 사항에 맞게 S3에서 검색된 요청된 객체를 수정하는 방법에 대해 알아봅니다. 특히 S3 버킷에 저장된 원본 객체의 모든 텍스트를 S3 Object Lambda를 통해 대문자로 변환하는 방법에 대해 알아봅니다.

> **Note**
>
> 이 튜토리얼에서는 Python 코드를 사용하여 데이터를 변환합니다. 다른 AWS SDK를 사용하는 예는 AWS SDK 코드 예제 라이브러리의 S3 객체 Lambda를 사용하여 애플리케이션의 데이터 변환을 참조하세요.

주제

- 사전 조건
- 1단계: S3 버킷 생성
- 2단계: S3 버킷에 파일 업로드
- 3단계: S3 액세스 포인트 생성
- 4단계: Lambda 함수 생성

- 5단계: Lambda 함수의 실행 역할에 대한 IAM 정책 구성
- 6단계: S3 객체 Lambda 액세스 포인트 생성
- 7단계: 변환된 데이터 보기
- 8단계: 정리
- 다음 단계

사전 조건

이 자습서를 시작하기 전에 올바른 권한이 있는 AWS Identity and Access Management(IAM) 사용자로 로그인할 수 있는 AWS 계정이 있어야 합니다. 또한 Python 버전 3.8 이상을 설치해야 합니다.

하위 단계

- AWS 계정 권한이 있는 IAM 사용자 생성(콘솔)
- 로컬 시스템에 Python 3.8 이상을 설치합니다.

AWS 계정 권한이 있는 IAM 사용자 생성(콘솔)

그 대신 자습서의 IAM 사용자 자격 증명을 사용하면 됩니다. 이 자습서를 완료하려면 IAM 사용자가 다음 IAM 정책을 연결하여 관련 AWS 리소스에 액세스하고 특정 작업을 수행해야 합니다. IAM 사용자를 생성하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 사용자 생성(콘솔)을 참조하십시오.

IAM 사용자에게는 다음의 정책이 필요합니다.

- AmazonS3FullAccess - 객체 Lambda 액세스 포인트를 생성하고 사용할 수 있는 권한을 포함하여 모든 Amazon S3 작업에 권한을 부여합니다.
- AWSLambda_FullAccess - 모든 Lambda 작업에 권한을 부여합니다.
- IAMFullAccess - 모든 IAM 작업에 권한을 부여합니다.
- IAMAccessAnalyzerReadOnlyAccess - IAM 액세스 분석기에서 제공하는 모든 액세스 정보를 읽을 수 있는 권한을 부여합니다.
- CloudWatchLogsFullAccess – CloudWatch Logs에 대한 전체 액세스 권한을 부여합니다.

> **Note**
>
> 편의상 이 자습서에서는 IAM 사용자를 생성해 사용합니다. 이 자습서를 완료하면 IAM 사용자 삭제를 수행해야 합니다. 프로덕션 용도로는 IAM 사용 설명서에서 IAM의 보안 모범 사례를 따

르는 것이 좋습니다. 가장 좋은 방법은 인간 사용자가 ID 제공업체와의 페더레이션을 사용하여 임시 보안 인증으로 AWS에 액세스하도록 하는 것입니다. 워크로드에서 IAM 역할과 함께 임시 보안 인증을 사용하여 AWS에 액세스하도록 하는 것도 좋은 방법입니다. AWS IAM Identity Center를 사용하여 임시 보안 인증으로 사용자를 생성하는 방법에 대해 알아보려면 AWS IAM Identity Center 사용 설명서의 Getting started(시작하기)를 참조하십시오.
이 자습서에서는 편의상 전체 액세스 AWS 관리형 정책을 사용합니다. 프로덕션 사용 용도의 경우 보안 모범 사례에 따라 사용 사례에 필요한 최소한의 권한만 부여하는 것이 좋습니다.

로컬 시스템에 Python 3.8 이상을 설치합니다.

로컬 시스템에 Python 3.8 이상을 설치하려면 다음 절차를 따르십시오. 설치 지침은 Python 초급 가이드의 Python 다운로드 페이지를 참조하십시오.

1. 로컬 터미널 또는 셸을 열고 다음 명령을 실행하여 Python이 이미 설치되어 있는지 여부를 확인하고, 설치되어 있는 경우에는 어떤 버전이 설치되어 있는지 확인합니다.

   ```
   python --version
   ```

2. Python 3.8 이상의 버전이 없는 경우 로컬 시스템에 적합한 Python 3.8 이상의 공식 설치 프로그램을 다운로드합니다.
3. 다운로드한 파일을 두 번 클릭하여 설치 프로그램을 실행하고 단계에 따라 설치를 완료합니다.

   Windows 사용자의 경우 먼저 설치 마법사에서 Python 3.X를 경로에 추가(Add Python 3.X to PATH)를 선택한 후 지금 설치(Install Now)를 선택합니다.
4. 터미널을 닫았다가 다시 열어서 다시 시작합니다.
5. Python 3.8 이상이 제대로 설치되었는지 확인하려면 다음 명령을 실행합니다.

   macOS 사용자의 경우 이 명령을 실행합니다.

   ```
   python3 --version
   ```

   Windows 사용자의 경우 이 명령을 실행합니다.

   ```
   python --version
   ```

6. pip3 패키지 관리자가 설치되었는지 확인하려면 다음 명령을 실행합니다. 명령 응답에 pip 버전 번호와 python 3.8 이상이 표시되면 pip3 패키지 관리자가 성공적으로 설치되었음을 의미합니다.

```
pip --version
```

1단계: S3 버킷 생성

버킷을 생성하여 변환하려는 원본 데이터를 저장합니다.

> **Note**
>
> Amazon FSx for OpenZFS 볼륨과 같은 다른 데이터 소스에 액세스 포인트를 연결할 수 있지만 이 자습서에서는 S3 버킷에 연결된 지원 액세스 포인트를 사용합니다.

버킷을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 만들기를 선택합니다.

   버킷 만들기 페이지가 열립니다.
4. 버킷 이름(Bucket Name)에서 버킷 이름을 입력합니다(예: **`tutorial-bucket`**).

   Amazon S3의 버킷 이름 지정 규칙에 대한 자세한 내용은 범용 버킷 이름 지정 규칙 섹션을 참조하세요.
5. 리전(Region)에서 버킷이 속할 AWS 리전을 선택합니다.

   버킷 리전에 대한 자세한 내용은 범용 버킷 개요 섹션을 참조하세요.
6. 이 버킷에 대한 퍼블릭 액세스 차단 설정(Block Public Access settings for this bucket)에서 해당 설정을 기본값으로 유지합니다(모든 퍼블릭 액세스 차단(Block all public access)이 활성화됨).

   해당 사용 사례에 대해 하나 이상의 설정을 해제해야 하는 경우가 아니라면 모든 설정을 퍼블릭 액세스 차단 설정 활성화 상태로 유지하는 것이 좋습니다. 퍼블릭 액세스 차단에 대한 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하십시오.
7. 나머지 설정은 기본값으로 유지합니다.

   특정 사용 사례에 대한 추가 버킷 설정을 구성하려면 범용 버킷 생성 섹션을 참조하십시오(선택 사항).

8. 버킷 생성을 선택합니다.

2단계: S3 버킷에 파일 업로드

텍스트 파일을 S3 버킷에 업로드합니다. 이 텍스트 파일에는 이 자습서의 뒷부분에서 대문자로 변환할 원본 데이터가 들어 있습니다.

예를 들어, 다음 텍스트가 포함된 `tutorial.txt` 파일을 업로드할 수 있습니다.

```
Amazon S3 Object Lambda Tutorial:
You can add your own code to process data retrieved from S3 before
returning it to an application.
```

버킷에 파일을 업로드하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷(Buckets) 목록에서, 1단계에서 생성한 버킷의 이름(예: **`tutorial-bucket`**)을 선택하여 파일을 업로드합니다.
4. 버킷의 객체(Objects) 탭에서 업로드(Upload)를 선택합니다.
5. 업로드(Upload) 페이지의 파일 및 폴더(Files and Folders)에서 파일 추가(Add Files)를 선택합니다.
6. 업로드할 파일을 선택한 후 열기를 선택합니다. 예를 들어, 앞에서 언급한 `tutorial.txt` 파일 예제를 업로드할 수 있습니다.
7. 업로드를 선택합니다.

3단계: S3 액세스 포인트 생성

S3 객체 Lambda 액세스 포인트를 사용하여 원본 데이터에 액세스하고 변환하려면 S3 액세스 포인트를 생성하고 1단계에서 생성한 S3 버킷에 연결해야 합니다. 액세스 포인트는 변환하려는 객체와 동일한 AWS 리전에 있어야 합니다.

이 자습서의 뒷부분에서 이 액세스 포인트를 객체 Lambda 액세스 포인트에 대한 지원 액세스 포인트로 사용합니다.

액세스 포인트 생성

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 탐색 창에서 액세스 포인트(Access Points)를 선택합니다.
3. 액세스 포인트(Access Points) 페이지에서 액세스 포인트 생성(Create access point)을 선택합니다.
4. 액세스 포인트 이름(Access point name) 필드에 액세스 포인트 이름을 입력합니다(예: **`tutorial-access-point`**).

   액세스 포인트 명명에 대한 자세한 내용은 액세스 포인트 이름 지정 규칙 섹션을 참조하십시오.
5. 데이터 소스 필드에서 1단계에서 생성한 버킷의 이름을 입력합니다(예: **`tutorial-bucket`**). S3는 액세스 포인트를 이 버킷에 연결합니다.

   (선택 사항) S3 찾아보기(Browse S3)를 선택하여 계정의 버킷을 찾아보고 검색할 수 있습니다. S3 찾아보기(Browse S3)를 선택할 경우 원하는 버킷을 선택한 후 경로 선택(Choose path)을 선택하여 버킷 이름(Bucket name) 필드를 해당 버킷의 이름으로 채웁니다.
6. 네트워크 오리진(Network origin)에서 인터넷(Internet)을 선택합니다.

   액세스 포인트의 네트워크 오리진에 대한 자세한 내용은 가상 프라이빗 클라우드(VPC)로 제한된 액세스 포인트 생성 섹션을 참조하십시오.
7. 액세스 포인트에는 기본값으로 모든 퍼블릭 액세스 차단(Block all public access) 설정이 활성화되어 있습니다. 모든 퍼블릭 액세스 차단(Block all public access) 설정을 활성화 상태로 유지하는 것이 좋습니다.

   자세한 정보는 범용 버킷의 액세스 포인트에 대한 퍼블릭 액세스 관리을 참조하세요.
8. 다른 모든 액세스 포인트 설정의 경우 기본 설정을 그대로 유지합니다.

   (선택 사항) 사용 사례를 지원하도록 액세스 포인트 설정을 수정할 수 있습니다. 이 자습서에서는 기본 설정을 그대로 유지하는 것이 좋습니다.

   (선택 사항) 액세스 포인트에 대한 액세스를 관리해야 하는 경우 액세스 포인트 정책을 지정할 수 있습니다. 자세한 정보는 액세스 포인트의 정책 예시을 참조하세요.
9. 액세스 포인트 생성을 선택합니다.

4단계: Lambda 함수 생성

원본 데이터를 변환하려면 S3 객체 Lambda 액세스 포인트에 사용할 Lambda 함수를 생성합니다.

하위 단계

- Lambda 함수 코드 작성 및 가상 환경의 배포 패키지 생성
- 실행 역할을 사용하여 Lambda 함수 생성(콘솔)
- .zip 파일 아카이브를 사용하여 Lambda 함수 코드 배포 및 Lambda 함수 구성(콘솔)

Lambda 함수 코드 작성 및 가상 환경의 배포 패키지 생성

1. 로컬 시스템에서, 이 자습서의 뒷부분에서 사용할 가상 환경의 `object-lambda` 폴더 이름으로 폴더를 생성합니다.
2. `object-lambda` 폴더에서 원본 객체의 모든 텍스트를 대문자로 변경하는 Lambda 함수를 사용하여 파일을 생성합니다. 예를 들어 Python으로 작성된 다음 함수를 사용할 수 있습니다. 이 함수를 `transform.py` 파일에 저장합니다.

```
import boto3
import requests
from botocore.config import Config

# This function capitalizes all text in the original object
def lambda_handler(event, context):
    object_context = event["getObjectContext"]
    # Get the presigned URL to fetch the requested original object
    # from S3
    s3_url = object_context["inputS3Url"]
    # Extract the route and request token from the input context
    request_route = object_context["outputRoute"]
    request_token = object_context["outputToken"]

    # Get the original S3 object using the presigned URL
    response = requests.get(s3_url)
    original_object = response.content.decode("utf-8")

    # Transform all text in the original object to uppercase
    # You can replace it with your custom code based on your use case
    transformed_object = original_object.upper()

    # Write object back to S3 Object Lambda
```

```
    s3 = boto3.client('s3', config=Config(signature_version='s3v4'))
    # The WriteGetObjectResponse API sends the transformed data
    # back to S3 Object Lambda and then to the user
    s3.write_get_object_response(
        Body=transformed_object,
        RequestRoute=request_route,
        RequestToken=request_token)

    # Exit the Lambda function: return the status code
    return {'status_code': 200}
```

> **Note**
>
> 앞의 예제 Lambda 함수는 요청된 객체 전체를 메모리에 로드한 다음 변환하여 클라이언트에 반환합니다. 또는 S3에서 객체를 스트리밍하여 전체 객체를 메모리에 로드하지 않도록 할 수 있습니다. 이 접근 방법은 큰 객체 작업 시 유용할 수 있습니다. 객체 Lambda 액세스 포인트를 사용하여 응답을 스트리밍하는 방법에 대한 자세한 내용은 Lambda에서 GetObject 요청 작업 섹션을 참조하십시오.

S3 객체 Lambda 액세스 포인트와 함께 사용할 Lambda 함수를 작성할 때 이 함수는 S3 객체 Lambda 함수가 Lambda 함수에 제공하는 입력 이벤트 컨텍스트를 기반으로 합니다. 이벤트 컨텍스트는 S3 객체 Lambda에서 Lambda로 전달된 이벤트에서 수행되는 요청에 대한 정보를 제공합니다. 여기에는 Lambda 함수를 생성하는 데 사용하는 파라미터가 포함되어 있습니다.

앞의 Lambda 함수를 생성하는 데 사용된 필드는 다음과 같습니다.

`getObjectContext` 필드는 Amazon S3 및 S3 객체 Lambda 연결에 대한 입력 및 출력 세부 정보입니다. 해당 필드에는 다음 필드가 포함되어 있습니다.

- `inputS3Url` - Lambda 함수가 지원 액세스 포인트에서 원본 객체를 다운로드하는 데 사용할 수 있는 미리 서명된 URL입니다. 미리 서명된 URL을 사용하면 Lambda 함수가 원본 객체를 검색하기 위해 Amazon S3 읽기 권한을 가질 필요가 없으며 각 호출에 의해 처리된 객체에만 액세스할 수 있습니다.
- `outputRoute` - 변환된 객체를 다시 전송하기 위해 Lambda 함수에서 `WriteGetObjectResponse`를 호출할 때 S3 객체 Lambda URL에 추가되는 라우팅 토큰입니다.
- `outputToken` - 변환된 객체를 다시 전송할 때 `WriteGetObjectResponse` 호출을 원래 발신자와 일치시키기 위해 S3 객체 Lambda에서 사용하는 토큰입니다.

이벤트 컨텍스트의 모든 필드에 대한 자세한 내용은 이벤트 컨텍스트 형식 및 사용법 및 S3 객체 Lambda 액세스 포인트에 대한 Lambda 함수 작성 섹션을 참조하십시오.

3. 로컬 터미널에서 `virtualenv` 패키지를 설치하려면 다음 명령을 입력합니다.

```
python -m pip install virtualenv
```

4. 로컬 터미널에서, 이전에 생성된 `object-lambda` 폴더를 연 후 다음 명령을 입력하여 venv 가상 환경을 생성하고 초기화합니다.

```
python -m virtualenv venv
```

5. 가상 환경을 활성화하려면 다음 명령을 입력하여 환경의 폴더에서 `activate` 파일을 실행합니다.

   macOS 사용자의 경우 이 명령을 실행합니다.

```
source venv/bin/activate
```

   Windows 사용자의 경우 이 명령을 실행합니다.

```
.\venv\Scripts\activate
```

   이제 명령 프롬프트가 (venv)를 표시하도록 변경되어 가상 환경이 활성임을 보여 줍니다.

6. 필요한 라이브러리를 설치하려면 venv 가상 환경에서 다음 명령을 한 줄씩 실행합니다.

   이러한 명령은 `lambda_handler` Lambda 함수의 종속성에 대한 업데이트된 버전을 설치합니다. 이러한 종속성은 AWS SDK for Python(Boto3) 및 요청 모듈입니다.

```
pip3 install boto3
```

```
pip3 install requests
```

7. 가상 환경을 비활성화하려면 다음 명령을 실행합니다.

```
deactivate
```

8. 설치된 라이브러리를 `object-lambda` 디렉터리의 루트에 `lambda.zip`이라는 `.zip` 파일로 사용하여 배포 패키지를 생성하려면 로컬 터미널에서 다음 명령줄을 한 줄씩 실행합니다.

> **Tip**
>
> 특정 환경에서 작동하도록 다음 명령을 조정해야 할 수 있습니다. 예를 들어 라이브러리는 `site-packages` 또는 `dist-packages`에 표시될 수 있으며, 첫 번째 폴더는 `lib` 또는 `lib64`에 표시될 수 있습니다. 또한 `python` 폴더의 이름은 다른 Python 버전으로 명명될 수 있습니다. 특정 패키지를 찾으려면 `pip show` 명령을 사용합니다.

macOS 사용자의 경우 다음 명령을 실행합니다.

```
cd venv/lib/python3.8/site-packages
```

```
zip -r ../../../../lambda.zip .
```

Windows 사용자의 경우 다음 명령을 실행합니다.

```
cd .\venv\Lib\site-packages\
```

```
powershell Compress-Archive * ../../../lambda.zip
```

마지막 명령은 배포 패키지를 `object-lambda` 디렉터리의 루트에 저장합니다.

9. 배포 패키지 루트에 함수 코드 파일 `transform.py`를 추가합니다.

macOS 사용자의 경우 다음 명령을 실행합니다.

```
cd ../../../../
```

```
zip -g lambda.zip transform.py
```

Windows 사용자의 경우 다음 명령을 실행합니다.

```
cd ..\..\..\
```

```
powershell Compress-Archive -update transform.py lambda.zip
```

이 단계를 완료한 후에는 다음과 같은 디렉터리 구조가 있어야 합니다.

```
lambda.zip$
  # transform.py
  # __pycache__
  | boto3/
  # certifi/
  # pip/
  # requests/
  ...
```

실행 역할을 사용하여 Lambda 함수 생성(콘솔)

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/lambda/에서 AWS Lambda 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 함수(Functions)를 선택합니다.
3. 함수 생성을 선택합니다.
4. 새로 작성을 선택합니다.
5. 기본 정보에서 다음과 같이 합니다.
   a. [함수 이름(Function name)]에 **tutorial-object-lambda-function**을 입력합니다.
   b. 런타임(Runtime)에서 Python 3.8 이상 버전을 선택합니다.
6. 기본 실행 역할 변경(Change default execution role) 섹션을 펼칩니다. 실행 역할(Execution role)에서 기본 Lambda 권한을 가진 새 역할 생성(Create a new role with basic Lambda permissions)을 선택합니다.

   이 자습서 뒷부분의 5단계에서 AmazonS3ObjectLambdaExecutionRolePolicy를 이 Lambda 함수의 실행 역할에 연결합니다.
7. 나머지 설정은 기본값으로 유지합니다.
8. 함수 생성을 선택합니다.

.zip 파일 아카이브를 사용하여 Lambda 함수 코드 배포 및 Lambda 함수 구성(콘솔)

1. https://console.aws.amazon.com/lambda/에 있는 AWS Lambda 콘솔의 왼쪽 탐색 창에서 함수(Functions)를 선택합니다.

2. 전에 생성해 둔 Lambda 함수를 선택합니다(예: **`tutorial-object-lambda-function`**).
3. Lambda 함수의 세부 정보 페이지에서 코드(Code) 탭을 선택합니다. 코드 소스(Code Source) 섹션에서 업로드 원본(Upload from)을 선택한 다음 .zip 파일(.zip file)을 선택합니다.
4. 업로드(Upload)를 선택하여 로컬 `.zip` 파일을 선택합니다.
5. 이전에 생성한 `lambda.zip` 파일을 선택한 후 열기(Open)를 선택합니다.
6. 저장을 선택합니다.
7. 런타임 설정(Runtime settings) 섹션에서 편집(Edit)을 선택합니다.
8. 런타임 설정 편집(Edit runtime settings) 페이지에서 런타임(Runtime)이 Python 3.8 이상 버전으로 설정되었는지 확인합니다.
9. Lambda 함수 코드에서 호출할 핸들러 메서드를 Lambda 런타임에 알리기 위해 핸들러(Handler)에 **`transform.lambda_handler`**를 입력합니다.

   Python에서 함수를 구성할 때, 핸들러 설정의 값은 파일의 이름과 핸들러 모듈의 이름이며 점으로 구분됩니다. 예를 들어 `transform.lambda_handler`는 `transform.py` 파일에 정의된 `lambda_handler` 메서드를 호출합니다.
10. 저장을 선택합니다.
11. (선택 사항) Lambda 함수의 세부 정보 페이지에서 구성(Configuration) 탭을 선택합니다. 왼쪽 탐색 창에서 일반 구성(General configuration)을 선택한 후 편집(Edit)을 선택합니다. 제한 시간(Timeout) 필드에 **`1`**분 **`0`**초를 입력합니다. 나머지 설정은 기본값으로 유지하고 저장(Save)을 선택합니다.

    제한 시간(Timeout)은 Lambda가 함수를 중지하기까지 호출 실행을 허용하는 시간입니다. 기본값은 3초입니다. S3 객체 Lambda에서 사용하는 Lambda 함수의 최대 지속 시간은 60초입니다. 요금은 구성된 메모리 양과 코드가 실행되는 시간을 기준으로 책정됩니다.

5단계: Lambda 함수의 실행 역할에 대한 IAM 정책 구성

Lambda 함수가 사용자 지정된 데이터 및 응답 헤더를 `GetObject` 발신자에게 제공하도록 설정하려면 Lambda 함수의 실행 역할에 `WriteGetObjectResponse` API를 호출할 수 있는 IAM 권한이 있어야 합니다.

IAM 정책을 Lambda 함수 역할에 연결하려면

1. https://console.aws.amazon.com/lambda/에 있는 AWS Lambda 콘솔의 왼쪽 탐색 창에서 함수(Functions)를 선택합니다.

2. 4단계에서 생성한 함수(예: **tutorial-object-lambda-function**)를 선택합니다.
3. Lambda 함수의 세부 정보 페이지에서 구성(Configuration) 탭을 선택한 후 왼쪽 탐색 창에서 권한(Permissions)을 선택합니다.
4. 실행 역할(Execution role)에서 역할 이름(Role name) 링크를 선택합니다. 그러면 IAM 콘솔이 열립니다.
5. Lambda 함수의 실행 역할에 대한 IAM 콘솔의 Summary(요약) 페이지에서 Permissions(권한) 탭을 선택합니다. 그런 다음, 권한 추가 메뉴에서 정책 연결을 선택합니다.
6. 권한 연결(Attach Permissions) 페이지에서 검색 필드에 **AmazonS3ObjectLambdaExecutionRolePolicy**를 입력하여 정책 목록을 필터링합니다. AmazonS3ObjectLambdaExecutionRolePolicy 정책 이름 옆의 확인란을 선택합니다.
7. 정책 연결을 선택합니다.

6단계: S3 객체 Lambda 액세스 포인트 생성

S3 객체 Lambda 액세스 포인트는 해당 함수가 S3 액세스 포인트에서 검색된 데이터를 처리할 수 있도록 S3 GET 요청에서 직접 Lambda 함수를 호출할 수 있는 유연성을 제공합니다. S3 객체 Lambda 액세스 포인트를 생성하고 구성하는 경우, Lambda 함수를 지정하여 Lambda가 사용할 사용자 지정 파라미터로 JSON 형식의 이벤트 컨텍스트를 호출하고 제공해야 합니다.

S3 객체 Lambda 액세스 포인트를 생성하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 객체 Lambda 액세스 포인트(Object Lambda Access Points)를 선택합니다.
3. 객체 Lambda 액세스 포인트(Object Lambda Access Points) 페이지에서 객체 Lambda 액세스 포인트 생성(Create Object Lambda Access Point)을 선택합니다.
4. 객체 Lambda 액세스 포인트 이름에서 객체 Lambda 액세스 포인트에 사용할 이름을 입력합니다(예: **tutorial-object-lambda-accesspoint**).
5. 지원 액세스 포인트(Supporting Access Point)에서, 3단계에서 생성한 표준 액세스 포인트(예: **tutorial-access-point**)를 입력하거나 검색한 후 지원 액세스 포인트 선택(Choose supporting Access Point).을 선택합니다.
6. S3 API의 경우, Lambda 함수가 처리할 S3 버킷에서 객체를 검색하려면 GetObject를 선택합니다.
7. Lambda 함수 호출(Invoke Lambda function)에서, 이 자습서의 다음 두 가지 옵션 중 하나를 선택할 수 있습니다.

- 계정의 함수에서 선택(Choose from functions in your account)을 선택한 후 Lambda 함수(Lambda function) 드롭다운 목록의 4단계에서 생성한 Lambda 함수(예: **`tutorial-object-lambda-function`**)를 선택합니다.
- ARN 입력(Enter ARN)을 선택한 후, 4단계에서 생성한 Lambda 함수의 Amazon 리소스 이름(ARN)을 입력합니다.

8. Lambda 함수 버전(Lambda function version)에서 $LATEST(4단계에서 생성한 Lambda 함수의 최신 버전)를 선택합니다.
9. (선택 사항) Lambda 함수가 범위 및 부분 번호 헤더가 있는 GET 요청을 인식하고 처리해야 하는 경우 Lambda 함수는 범위를 사용하여 요청을 지원합니다(Lambda function supports requests using range) 및 Lambda 함수는 부분 번호를 사용하여 요청을 지원합니다(Lambda function supports requests using part numbers)를 선택합니다. 그렇지 않으면 이 두 확인란의 선택을 취소합니다.

   S3 객체 Lambda에서 범위 또는 부분 번호를 사용하는 방법에 대한 자세한 내용은 Range 및 partNumber 헤더 작업 섹션을 참조하십시오.
10. (선택 사항) 페이로드 - 선택 사항(Payload - optional)에서, Lambda 함수에 추가 정보를 제공하는 JSON 텍스트를 추가합니다.

    페이로드는 특정 S3 객체 Lambda 액세스 포인트에서 오는 모든 호출에 대한 입력으로 Lambda 함수에 제공할 수 있는 선택적 JSON 텍스트입니다. 동일한 Lambda 함수를 호출하는 여러 객체 Lambda 액세스 포인트에 대해 사용자 지정하기 위해 서로 다른 파라미터를 사용하여 페이로드를 구성할 수 있으므로 Lambda 함수의 유연성을 높일 수 있습니다.

    페이로드에 대한 자세한 내용은 이벤트 컨텍스트 형식 및 사용법 섹션을 참조하십시오.
11. 요청 지표 - 선택 사항에서 비활성화 또는 활성화를 선택하여 객체 Lambda 액세스 포인트에 Amazon S3 모니터링을 추가합니다(선택 사항). 요청 지표는 표준 Amazon CloudWatch 요금으로 청구됩니다. 자세한 내용은 CloudWatch 요금을 참조하십시오.
12. 객체 Lambda 액세스 포인트 정책 - 선택 사항(Object Lambda Access Point policy - optional)에서 기본 설정을 그대로 유지합니다.

    리소스 정책을 설정할 수 있습니다(선택 사항). 이 리소스 정책은 지정된 객체 Lambda 액세스 포인트를 사용할 수 있는 권한을 `GetObject` API에 부여합니다.
13. 나머지 설정은 기본값으로 유지하고 객체 Lambda 액세스 포인트 생성(Create Object Lambda Access Point)을 선택합니다.

7단계: 변환된 데이터 보기

이제 S3 Object Lambda가 사용 사례에 맞게 데이터를 변환할 준비가 되었습니다. 이 자습서에서는 S3 객체 Lambda가 객체의 모든 텍스트를 대문자로 변환합니다.

하위 단계

- S3 객체 Lambda 액세스 포인트에서 변환된 데이터 보기
- Python 스크립트를 실행하여 원본 및 변환된 데이터를 인쇄합니다.

S3 객체 Lambda 액세스 포인트에서 변환된 데이터 보기

S3 객체 Lambda 액세스 포인트를 통해 파일을 검색하도록 요청할 때 S3 객체 Lambda에 대한 `GetObject` API 호출을 수행합니다. S3 객체 Lambda는 Lambda 함수를 호출하여 데이터를 변환하고 변환된 데이터를 표준 S3 `GetObject` API 호출에 대한 응답으로 반환합니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 객체 Lambda 액세스 포인트(Object Lambda Access Points)를 선택합니다.
3. 객체 Lambda 액세스 포인트 페이지에서 6단계에서 생성한 S3 객체 Lambda 액세스 포인트를 선택합니다(예: **`tutorial-object-lambda-accesspoint`**).
4. S3 객체 Lambda 액세스 포인트의 객체 탭에서 2단계에서 S3 버킷에 업로드한 파일과 이름이 같은 파일을 선택합니다(예: `tutorial.txt`)

   이 파일에는 변환된 모든 데이터가 포함되어야 합니다.
5. 변환된 데이터를 보려면 열기(Open) 또는 다운로드(Download)를 선택합니다.

Python 스크립트를 실행하여 원본 및 변환된 데이터를 인쇄합니다.

S3 객체 Lambda를 기존 애플리케이션에서 사용할 수 있습니다. 이 작업을 수행하려면 6단계에서 생성한 새로운 S3 객체 Lambda 액세스 포인트 ARN을 사용하여 S3에서 데이터를 검색하도록 애플리케이션 구성을 업데이트합니다.

다음 예시 Python 스크립트는 S3 버킷의 원본 데이터와 S3 객체 Lambda 액세스 포인트에서 변환된 데이터를 모두 인쇄합니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.

2. 왼쪽 탐색 창에서 객체 Lambda 액세스 포인트(Object Lambda Access Points)를 선택합니다.
3. 객체 Lambda 액세스 포인트 페이지에서 6단계에서 생성한 S3 객체 Lambda 액세스 포인트 왼쪽에 있는 라디오 버튼을 선택합니다(예: **tutorial-object-lambda-accesspoint**).
4. ARN 복사(Copy ARN)를 선택합니다.
5. 나중에 사용할 수 있도록 ARN을 저장합니다.
6. 로컬 시스템에서 Python 스크립트를 작성하여 S3 버킷의 원본 데이터(예 :tutorial.txt)와 S3 객체 Lambda 액세스 포인트의 변환된 데이터(예: tutorial.txt)를 모두 인쇄합니다. 다음 예제 스크립트를 사용할 수 있습니다.

```
import boto3
from botocore.config import Config

s3 = boto3.client('s3', config=Config(signature_version='s3v4'))

def getObject(bucket, key):
    objectBody = s3.get_object(Bucket = bucket, Key = key)
    print(objectBody["Body"].read().decode("utf-8"))
    print("\n")

print('Original object from the S3 bucket:')
# Replace the two input parameters of getObject() below with
# the S3 bucket name that you created in Step 1 and
# the name of the file that you uploaded to the S3 bucket in Step 2
getObject("tutorial-bucket",
          "tutorial.txt")

print('Object transformed by S3 Object Lambda:')
# Replace the two input parameters of getObject() below with
# the ARN of your S3 Object Lambda Access Point that you saved earlier and
# the name of the file with the transformed data (which in this case is
# the same as the name of the file that you uploaded to the S3 bucket
# in Step 2)
getObject("arn:aws:s3-object-lambda:us-west-2:111122223333:accesspoint/tutorial-
object-lambda-accesspoint",
          "tutorial.txt")
```

7. 로컬 시스템의 4단계에서 생성한 폴더(예: object-lambda)에 Python 스크립트를 사용자 지정 이름(예: tutorial_print.py)으로 저장합니다.
8. 로컬 터미널에서, 4단계에서 생성한 디렉터리 루트에서 다음 명령을 실행합니다(예: object-lambda).

```
python3 tutorial_print.py
```

터미널을 통해 원본 데이터와 변환된 데이터(대문자로 된 모든 텍스트)가 모두 표시되어야 합니다. 예를 들어 다음 텍스트와 같은 내용이 표시되어야 합니다.

```
Original object from the S3 bucket:
Amazon S3 Object Lambda Tutorial:
You can add your own code to process data retrieved from S3 before
returning it to an application.

Object transformed by S3 Object Lambda:
AMAZON S3 OBJECT LAMBDA TUTORIAL:
YOU CAN ADD YOUR OWN CODE TO PROCESS DATA RETRIEVED FROM S3 BEFORE
RETURNING IT TO AN APPLICATION.
```

8단계: 정리

실습용으로만 S3 객체 Lambda를 통해 데이터를 변환한 경우에는 요금이 발생하지 않도록 할당한 AWS 리소스를 삭제합니다.

하위 단계

- 객체 Lambda 액세스 포인트 삭제
- S3 액세스 포인트 삭제
- Lambda 함수용 실행 역할 삭제
- Lambda 함수 삭제
- CloudWatch 로그 그룹 삭제
- S3 소스 버킷에서 원본 파일 삭제
- S3 소스 버킷 삭제
- IAM 사용자 삭제

객체 Lambda 액세스 포인트 삭제

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 객체 Lambda 액세스 포인트(Object Lambda Access Points)를 선택합니다.

3. 객체 Lambda 액세스 포인트 페이지에서 6단계에서 생성한 S3 객체 Lambda 액세스 포인트 왼쪽에 있는 라디오 버튼을 선택합니다(예: **tutorial-object-lambda-accesspoint**).
4. 삭제를 선택합니다.
5. 표시되는 텍스트 필드에 객체 Lambda 액세스 포인트 이름을 입력하여 삭제 여부를 확인하고 삭제를 선택합니다.

S3 액세스 포인트 삭제

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 탐색 창에서 액세스 포인트(Access Points)를 선택합니다.
3. 3단계에서 생성한 액세스 포인트(예: **tutorial-access-point**)로 이동한 후 액세스 포인트 이름 옆에 있는 라디오 버튼을 선택합니다.
4. 삭제를 선택합니다.
5. 표시되는 텍스트 필드에 액세스 포인트 이름을 입력하여 삭제 여부를 확인하고 삭제(Delete)를 선택합니다.

Lambda 함수용 실행 역할 삭제

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/lambda/에서 AWS Lambda 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 함수(Functions)를 선택합니다.
3. 4단계에서 생성한 함수(예: **tutorial-object-lambda-function**)를 선택합니다.
4. Lambda 함수의 세부 정보 페이지에서 구성(Configuration) 탭을 선택한 후 왼쪽 탐색 창에서 권한(Permissions)을 선택합니다.
5. 실행 역할(Execution role)에서 역할 이름(Role name) 링크를 선택합니다. 그러면 IAM 콘솔이 열립니다.
6. Lambda 함수의 실행 역할에 대한 IAM 콘솔 요약(Summary) 페이지에서 역할 삭제(Delete role)를 선택합니다.
7. 역할 삭제(Delete role) 대화 상자에서 예, 삭제(Yes, Delete)를 선택합니다.

Lambda 함수 삭제

1. https://console.aws.amazon.com/lambda/에 있는 AWS Lambda 콘솔의 왼쪽 탐색 창에서 함수(Functions)를 선택합니다.
2. 4단계에서 생성한 함수 이름 왼쪽에 있는 확인란을 선택합니다(예: **`tutorial-object-lambda-function`**).
3. 작업을 선택한 후 삭제를 선택합니다.
4. 함수 삭제(Delete function) 대화 상자에서 삭제(Delete)를 선택합니다.

CloudWatch 로그 그룹 삭제

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그 그룹(Log groups)을 선택합니다.
3. 이름이 4단계에서 생성한 Lambda 함수로 끝나는 로그 그룹을 찾습니다(예: **`tutorial-object-lambda-function`**).
4. 로그 그룹 이름 왼쪽에 있는 확인란을 선택합니다.
5. 작업(Actions)을 선택한 후 로그 그룹 삭제(Delete log group(s))를 선택합니다.
6. 로그 그룹 삭제(Delete log group(s)) 대화 상자에서 삭제(Delete)를 선택합니다.

S3 소스 버킷에서 원본 파일 삭제

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 이름(Bucket name) 목록에서, 2단계에서 원본 파일을 업로드한 버킷의 이름을 선택합니다(예: **`tutorial-bucket`**).
4. 삭제할 객체의 이름 왼쪽에 있는 확인란을 선택합니다(예: `tutorial.txt`).
5. 삭제를 선택합니다.
6. 객체 삭제(Delete objects) 페이지의 객체를 영구적으로 삭제하시겠습니까?(Permanently delete objects?) 섹션에서 텍스트 상자에 **`permanently delete`**를 입력하여 이 객체의 삭제 여부를 확인합니다.
7. 객체 삭제를 선택합니다.

S3 소스 버킷 삭제

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷(Buckets) 목록에서, 1단계에서 생성한 버킷의 이름 옆에 있는 라디오 버튼을 선택합니다(예: **`tutorial-bucket`**).
4. 삭제를 선택합니다.
5. 버킷 삭제(Delete bucket) 페이지의 텍스트 필드에 버킷 이름을 입력하여 버킷의 삭제 여부를 확인한 다음 버킷 삭제(Delete bucket)를 선택합니다.

IAM 사용자 삭제

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 사용자(Users)를 선택한 후 삭제할 사용자 이름 옆에 있는 확인란을 선택합니다.
3. 페이지 상단에서 삭제(Delete)를 선택합니다.
4. ### ##을 삭제하시겠습니까?(Delete user name?) 대화 상자에서 텍스트 입력 필드에 역할 이름을 입력하여 사용자의 삭제 여부를 확인합니다. 삭제를 선택합니다.

다음 단계

이 자습서를 완료한 후 사용 사례에 따라 Lambda 함수를 사용자 지정하여 표준 S3 GET 요청에서 반환된 데이터를 수정할 수 있습니다.

다음은 S3 객체 Lambda의 일반적인 사용 사례 목록입니다.

- 보안 및 규정 준수를 위해 민감한 데이터를 마스킹합니다.

  자세한 정보는 자습서: S3 객체 Lambda 및 Amazon Comprehend를 사용하여 PII 데이터 감지 및 수정을 참조하세요.
- 특정 정보를 제공하기 위해 특정 데이터 행을 필터링합니다.
- 다른 서비스 또는 데이터베이스의 정보로 데이터를 보강합니다.
- 애플리케이션 호환성을 위해 XML을 JSON으로 변환하는 등 데이터 형식 간에 변환합니다.
- 다운로드 중일 때 파일을 압축 또는 압축 해제합니다.
- 이미지 크기를 조정하고 워터마킹을 수행합니다.

자세한 내용은 Tutorial: Using S3 Object Lambda to dynamically watermark images as they are retrieved(자습서: S3 객체 Lambda를 사용하여 이미지를 검색할 때 동적으로 워터마크 지정)를 참조하십시오.

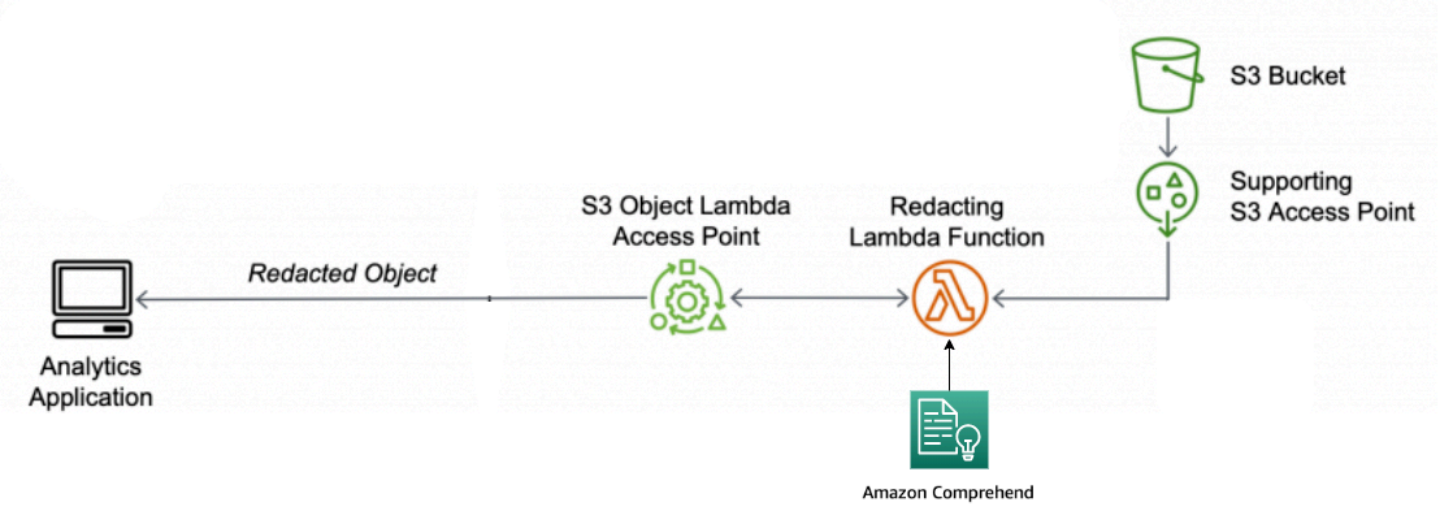
- 사용자 지정 권한 부여 규칙을 구현하여 데이터에 액세스합니다.

S3 객체 Lambda에 대한 자세한 내용은 S3 객체 Lambda를 사용하여 객체 변환 섹션을 참조하세요.

자습서: S3 객체 Lambda 및 Amazon Comprehend를 사용하여 PII 데이터 감지 및 수정

여러 애플리케이션과 사용자가 액세스할 수 있도록 공유 데이터 집합에 Amazon S3를 사용하는 경우 개인 식별 정보(PII)와 같이 권한 있는 정보를 승인된 엔터티로만 제한하는 것이 중요합니다. 예를 들어 마케팅 애플리케이션에서 PII가 포함된 일부 데이터를 사용하는 경우 데이터 개인 정보 보호 요구 사항을 충족하기 위해 먼저 PII 데이터를 마스킹해야 할 수 있습니다. 또한 분석 애플리케이션이 생산 주문 인벤토리 데이터 집합을 사용하는 경우 의도하지 않은 데이터 유출을 방지하기 위해 먼저 고객 신용 카드 정보를 수정해야 할 수 있습니다.

S3 객체 Lambda와 Amazon Comprehend에서 제공하는 사전 빌드된 AWS Lambda 함수를 사용하면 애플리케이션으로 반환하기 전에 S3에서 검색된 PII 데이터를 보호할 수 있습니다. 특히 사전 빌드된 Lambda 함수를 수정 함수로 사용하여 S3 객체 Lambda 액세스 포인트에 연결할 수 있습니다. 애플리케이션(예: 분석 애플리케이션)이 표준 S3 GET 요청을 전송하는 경우, S3 Object Lambda 액세스 포인트를 통해 수행된 이러한 요청은 사전 빌드된 수정 Lambda 함수를 간접 호출하여 지원 S3 액세스 포인트를 통해 기본 데이터 소스에서 검색된 PII 데이터를 감지하고 수정합니다. 그런 다음 S3 객체 Lambda 액세스 포인트가 수정된 결과를 애플리케이션에 다시 반환합니다.



프로세스에서 사전 빌드된 Lambda 함수는 자연어 처리(NLP) 서비스인 Amazon Comprehend를 사용하여 PII가 텍스트에 존재하는 방식(예: 숫자 또는 단어와 숫자의 조합)에 관계없이 PII가 표현되는

방식의 변형을 캡처할 수 있습니다. Amazon Comprehend는 텍스트에서 컨텍스트를 사용하여 4자리 숫자가 PIN인지, SSN(사회 보장 번호)의 마지막 4자리 숫자나 연도인지 파악할 수도 있습니다. Amazon Comprehend는 모든 텍스트 파일을 UTF-8 형식으로 처리하며 정확성에 영향을 주지 않고 대규모로 PII를 보호할 수 있습니다. 자세한 내용은 Amazon Comprehend 개발자 가이드의 Amazon Comprehend란 무엇입니까?를 참조하십시오.

목표

이 자습서에서는 사전 빌드된 Lambda 함수 `ComprehendPiiRedactionS3ObjectLambda`와 함께 S3 객체 Lambda를 사용하는 방법에 대해 알아봅니다. 이 함수는 Amazon Comprehend를 사용하여 PII 엔터티를 감지합니다. 그런 다음 별표로 대체하여 이러한 엔티티를 수정합니다. PII를 수정하면 보안 및 규정 준수에 도움이 되는 민감한 데이터를 숨길 수 있습니다.

또한 쉬운 배포를 위해 S3 Object Lambda와 함께 작동하도록 AWS Serverless Application Repository에서 사전 빌드된 AWS Lambda 함수를 사용하고 구성하는 방법에 대해 학습합니다.

주제

- 사전 조건: 권한이 있는 IAM 사용자 생성
- 1단계: S3 버킷 생성
- 2단계: S3 버킷에 파일 업로드
- 3단계: S3 액세스 포인트 생성
- 4단계: 사전 빌드된 Lambda 함수 구성 및 배포
- 5단계: S3 객체 Lambda 액세스 포인트 생성
- 6단계: S3 객체 Lambda 액세스 포인트를 사용하여 수정된 파일 검색
- 7단계: 정리
- 다음 단계

사전 조건: 권한이 있는 IAM 사용자 생성

이 자습서를 시작하기 전에 올바른 권한이 있는 AWS Identity and Access Management 사용자(IAM 사용자)로 로그인할 수 있는 AWS 계정이 있어야 합니다.

그 대신 자습서의 IAM 사용자 자격 증명을 사용하면 됩니다. 이 자습서를 완료하려면 IAM 사용자가 다음 IAM 정책을 연결하여 관련 AWS 리소스에 액세스하고 특정 작업을 수행해야 합니다.

> **Note**
>
> 편의상 이 자습서에서는 IAM 사용자를 생성해 사용합니다. 이 자습서를 완료하면 IAM 사용자 삭제를 수행해야 합니다. 프로덕션 용도로는 IAM 사용 설명서에서 IAM의 보안 모범 사례를 따르는 것이 좋습니다. 가장 좋은 방법은 인간 사용자가 ID 제공업체와의 페더레이션을 사용하여 임시 보안 인증으로 AWS에 액세스하도록 하는 것입니다. 워크로드에서 IAM 역할과 함께 임시 보안 인증을 사용하여 AWS에 액세스하도록 하는 것도 좋은 방법입니다. AWS IAM Identity Center를 사용하여 임시 보안 인증으로 사용자를 생성하는 방법에 대해 알아보려면 AWS IAM Identity Center 사용 설명서의 Getting started(시작하기)를 참조하십시오.
> 이 자습서에서는 편의상 전체 액세스 정책을 사용합니다. 프로덕션 사용 용도의 경우 보안 모범 사례에 따라 사용 사례에 필요한 최소한의 권한만 부여하는 것이 좋습니다.

IAM 사용자에게는 다음의 AWS 관리형 정책이 필요합니다.

- AmazonS3FullAccess - 객체 Lambda 액세스 포인트를 생성하고 사용할 수 있는 권한을 포함하여 모든 Amazon S3 작업에 권한을 부여합니다.
- AWSLambda_FullAccess - 모든 Lambda 작업에 권한을 부여합니다.
- AWSCloudFormationFullAccess - 모든 AWS CloudFormation 작업에 권한을 부여합니다.
- IAMFullAccess - 모든 IAM 작업에 권한을 부여합니다.
- IAMAccessAnalyzerReadOnlyAccess - IAM 액세스 분석기에서 제공하는 모든 액세스 정보를 읽을 수 있는 권한을 부여합니다.

IAM 사용자를 생성할 때 이러한 기존 정책을 직접 연결할 수 있습니다. IAM 사용자를 생성하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 사용자 생성(콘솔)을 참조하십시오.

또한 IAM 사용자에게는 고객 관리형 정책이 필요합니다. IAM 사용자에게 AWS Serverless Application Repository 리소스 및 작업에 대한 모든 권한을 부여하려면 IAM 정책을 생성하고 이 정책을 IAM 사용자에게 연결해야 합니다.

IAM 정책을 생성하여 IAM 사용자에게 연결하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 정책을 선택합니다.
3. 정책 생성을 선택합니다.

4. 시각적 편집기(Visual editor) 탭의 서비스에서 서비스 선택(Choose a service)을 선택합니다. 그런 다음 Serverless Application Repository를 선택합니다.
5. 작업(Actions)의 수동 작업(Manual actions)에서 이 자습서의 모든 Serverless Application Repository 작업(serverlessrepo:*)을 선택합니다.

   보안 모범 사례에 따라 사용 사례에 필요한 작업과 리소스에만 권한을 허용해야 합니다. 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하십시오.
6. 리소스(Resources)에서, 이 자습서에 대해 모든 리소스(All resources)를 선택합니다.

   최선의 결과를 위해 특정 계정의 특정 리소스에 대한 권한만 정의해야 합니다. 또는 조건 키를 사용하여 최소 권한을 부여할 수도 있습니다. 자세한 내용은 IAM 사용 설명서의 최소 권한 부여를 참조하십시오.
7. 다음: 태그를 선택합니다.
8. 다음: 검토를 선택합니다.
9. 정책 검토(Review policy) 페이지에서, 생성하는 정책에 대한 이름(예: **`tutorial-serverless-application-repository`**)과 설명(선택 사항)을 입력합니다. 정책 요약을 검토하여 의도한 권한이 부여되었는지 확인한 다음 정책 생성을 선택하여 새 정책을 저장합니다.
10. 왼쪽 탐색 창에서 사용자를 선택합니다. 그런 다음 이 자습서에 대해 IAM 사용자를 선택합니다.
11. 선택한 사용자의 요약(Summary) 페이지에서 권한(Permissions)탭을 선택한 후 권한 추가(Add permissions)를 선택합니다.
12. 권한 부여(Grant permissions)에서 기존 정책 직접 연결(Attach existing policies directly)을 선택합니다.
13. 앞서 생성한 정책(예: **`tutorial-serverless-application-repository`**) 옆에 있는 확인란을 선택하고 다음: 검토(Next: Review)를 선택합니다.
14. 정책 요약(Permissions summary)에서 요약을 검토하여 의도한 정책을 연결했는지 확인합니다. 그런 다음 권한 추가(Add permissions)를 선택합니다.

1단계: S3 버킷 생성

버킷을 생성하여 변환하려는 원본 데이터를 저장합니다.

> **Note**
>
> Amazon FSx for OpenZFS 볼륨과 같은 다른 데이터 소스에 액세스 포인트를 연결할 수 있지만 이 자습서에서는 S3 버킷에 연결된 지원 액세스 포인트를 사용합니다.

버킷을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 만들기를 선택합니다.

   버킷 만들기 페이지가 열립니다.
4. 버킷 이름(Bucket Name)에서 버킷 이름을 입력합니다(예: **`tutorial-bucket`**).

   Amazon S3의 버킷 이름 지정 규칙에 대한 자세한 내용은 범용 버킷 이름 지정 규칙 섹션을 참조하세요.
5. 리전(Region)에서 버킷이 속할 AWS 리전을 선택합니다.

   버킷 리전에 대한 자세한 내용은 범용 버킷 개요 섹션을 참조하세요.
6. 이 버킷에 대한 퍼블릭 액세스 차단 설정(Block Public Access settings for this bucket)에서 해당 설정을 기본값으로 유지합니다(모든 퍼블릭 액세스 차단(Block all public access)이 활성화됨).

   해당 사용 사례에 대해 하나 이상의 설정을 해제해야 하는 경우가 아니라면 모든 설정을 퍼블릭 액세스 차단 설정 활성화 상태로 유지하는 것이 좋습니다. 퍼블릭 액세스 차단에 대한 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하십시오.
7. 나머지 설정은 기본값으로 유지합니다.

   특정 사용 사례에 대한 추가 버킷 설정을 구성하려면 범용 버킷 생성 섹션을 참조하십시오(선택 사항).
8. 버킷 생성을 선택합니다.

2단계: S3 버킷에 파일 업로드

이름, 은행 정보, 전화 번호, SSN 등 다양한 유형의 알려진 PII 데이터가 포함된 텍스트 파일을, 이 자습서의 뒷부분에서 PII를 수정할 원본 데이터로 S3 버킷에 업로드합니다.

예를 들어, 다음의 `tutorial.txt` 파일을 업로드할 수 있습니다. 이 파일은 Amazon Comprehend의 입력 파일의 예입니다.

```
Hello Zhang Wei, I am John. Your AnyCompany Financial Services,
LLC credit card account 1111-0000-1111-0008 has a minimum payment
of $24.53 that is due by July 31st. Based on your autopay settings,
```

```
we will withdraw your payment on the due date from your
bank account number XXXXXX1111 with the routing number XXXXX0000.

Your latest statement was mailed to 100 Main Street, Any City,
WA 98121.
After your payment is received, you will receive a confirmation
text message at 206-555-0100.
If you have questions about your bill, AnyCompany Customer Service
is available by phone at 206-555-0199 or
email at support@anycompany.com.
```

S3 버킷에 파일을 업로드하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷(Buckets) 목록에서, 1단계에서 생성한 버킷의 이름(예: **`tutorial-bucket`**)을 선택하여 파일을 업로드합니다.
4. 버킷의 객체(Objects) 탭에서 업로드(Upload)를 선택합니다.
5. 업로드(Upload) 페이지의 파일 및 폴더(Files and Folders)에서 파일 추가(Add Files)를 선택합니다.
6. 업로드할 파일을 선택한 후 열기를 선택합니다. 예를 들어, 앞에서 언급한 `tutorial.txt` 파일 예제를 업로드할 수 있습니다.
7. 업로드를 선택합니다.

3단계: S3 액세스 포인트 생성

S3 객체 Lambda 액세스 포인트를 사용하여 원본 데이터에 액세스하고 변환하려면 S3 액세스 포인트를 생성하고 1단계에서 생성한 S3 버킷에 연결해야 합니다. 액세스 포인트는 변환하려는 객체와 동일한 AWS 리전에 있어야 합니다.

이 자습서의 뒷부분에서 이 액세스 포인트를 객체 Lambda 액세스 포인트에 대한 지원 액세스 포인트로 사용합니다.

액세스 포인트 생성

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.

2. 탐색 창에서 액세스 포인트(Access Points)를 선택합니다.
3. 액세스 포인트(Access Points) 페이지에서 액세스 포인트 생성(Create access point)을 선택합니다.
4. 액세스 포인트 이름(Access point name) 필드에 액세스 포인트 이름을 입력합니다(예: **`tutorial-pii-access-point`**).

   액세스 포인트 명명에 대한 자세한 내용은 액세스 포인트 이름 지정 규칙 섹션을 참조하십시오.
5. 데이터 소스 필드에서 1단계에서 생성한 버킷의 이름을 입력합니다(예: **`tutorial-bucket`**). S3는 액세스 포인트를 이 버킷에 연결합니다.

   (선택 사항) S3 찾아보기(Browse S3)를 선택하여 계정의 버킷을 찾아보고 검색할 수 있습니다. S3 찾아보기(Browse S3)를 선택할 경우 원하는 버킷을 선택한 후 경로 선택(Choose path)을 선택하여 버킷 이름(Bucket name) 필드를 해당 버킷의 이름으로 채웁니다.
6. 네트워크 오리진(Network origin)에서 인터넷(Internet)을 선택합니다.

   액세스 포인트의 네트워크 오리진에 대한 자세한 내용은 가상 프라이빗 클라우드(VPC)로 제한된 액세스 포인트 생성 섹션을 참조하십시오.
7. 액세스 포인트에는 기본값으로 모든 퍼블릭 액세스 차단 설정이 활성화되어 있습니다. 모든 퍼블릭 액세스 차단(Block all public access) 설정을 활성화 상태로 유지하는 것이 좋습니다. 자세한 정보는 범용 버킷의 액세스 포인트에 대한 퍼블릭 액세스 관리을 참조하세요.
8. 다른 모든 액세스 포인트 설정의 경우 기본 설정을 그대로 유지합니다.

   (선택 사항) 사용 사례를 지원하도록 액세스 포인트 설정을 수정할 수 있습니다. 이 자습서에서는 기본 설정을 그대로 유지하는 것이 좋습니다.

   (선택 사항) 액세스 포인트에 대한 액세스를 관리해야 하는 경우 액세스 포인트 정책을 지정할 수 있습니다. 자세한 정보는 액세스 포인트의 정책 예시을 참조하세요.
9. 액세스 포인트 생성을 선택합니다.

4단계: 사전 빌드된 Lambda 함수 구성 및 배포

PII 데이터를 수정하려면 S3 객체 Lambda 액세스 포인트와 함께 사용할 사전 빌드된 AWS Lambda 함수 `ComprehendPiiRedactionS3ObjectLambda`를 구성하고 배포합니다.

Lambda 함수를 구성하고 배포하려면

1. AWS Management Console에 로그인한 후에 AWS Serverless Application Repository에서 `ComprehendPiiRedactionS3ObjectLambda` 함수를 확인합니다.

2. 애플리케이션 설정(Application settings)의 애플리케이션 이름(Application name)에서 이 자습서의 기본 값(`ComprehendPiiRedactionS3ObjectLambda`)을 그대로 유지합니다.

   (선택 사항) 이 애플리케이션에 부여할 이름을 입력할 수 있습니다. 동일한 공유 데이터 집합에 대한 서로 다른 액세스 요구에 대해 여러 Lambda 함수를 구성하려는 경우 이 작업을 수행할 수 있습니다.

3. MaskCharacter에서 기본값(*)을 그대로 유지합니다. 마스크 문자는 수정된 PII 엔터티의 각 문자를 대체합니다.

4. MaskMode에서 기본값(MASK)을 그대로 유지합니다. MaskMode 값은 PII 엔터티가 `MASK`문자 또는 `PII_ENTITY_TYPE` 값으로 수정되는지 여부를 지정합니다.

5. 지정된 유형의 데이터를 수정하기 위해 PiiEntityTypes에서 기본값(모두(ALL))을 그대로 유지합니다. PiiEntityTypes 값은 수정을 고려할 PII 엔터티 유형을 지정합니다.

   지원되는 PII 엔터티 유형 목록에 대한 자세한 내용은 Amazon Comprehend 개발자 가이드의 개인 식별 정보(PII) 감지를 참조하십시오.

6. 나머지 설정은 기본값으로 유지합니다.

   특정 사용 사례에 대한 추가 설정을 구성하려면 페이지 왼쪽의 Readme 파일 섹션을 참조하십시오.

7. I acknowledge that this app creates custom IAM roles(이 앱이 사용자 지정 IAM 역할을 생성하는 것을 확인) 옆 확인란을 선택합니다.

8. 배포(Deploy)를 선택합니다.

9. 새 애플리케이션 페이지의 리소스(Resources)에서, Lambda 함수 페이지에서 함수를 검토하기 위해 배포한 Lambda 함수의 논리 ID(Logical ID)를 선택합니다.

5단계: S3 객체 Lambda 액세스 포인트 생성

S3 객체 Lambda 액세스 포인트는 해당 함수가 S3 액세스 포인트에서 검색된 PII 데이터를 수정할 수 있도록 S3 GET 요청에서 직접 Lambda 함수를 호출할 수 있는 유연성을 제공합니다. S3 객체 Lambda 액세스 포인트를 생성하고 구성하는 경우, 수정 Lambda 함수를 지정하여 Lambda가 사용할 사용자 지정 파라미터로 JSON 형식의 이벤트 컨텍스트를 호출하고 제공해야 합니다.

이벤트 컨텍스트는 S3 객체 Lambda에서 Lambda로 전달된 이벤트에서 수행되는 요청에 대한 정보를 제공합니다. 이벤트 컨텍스트의 모든 필드에 대한 자세한 내용은 이벤트 컨텍스트 형식 및 사용법 섹션을 참조하십시오.

S3 객체 Lambda 액세스 포인트를 생성하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 객체 Lambda 액세스 포인트(Object Lambda Access Points)를 선택합니다.
3. 객체 Lambda 액세스 포인트(Object Lambda Access Points) 페이지에서 객체 Lambda 액세스 포인트 생성(Create Object Lambda Access Point)을 선택합니다.
4. 객체 Lambda 액세스 포인트 이름에서 객체 Lambda 액세스 포인트에 사용할 이름을 입력합니다(예: **`tutorial-pii-object-lambda-accesspoint`**).
5. 지원 액세스 포인트(Supporting Access Point)에서, 3단계에서 생성한 표준 액세스 포인트(예: **`tutorial-pii-access-point`**)를 입력하거나 검색한 후 지원 액세스 포인트 선택(Choose supporting Access Point).을 선택합니다.
6. S3 API의 경우, Lambda 함수가 처리할 S3 버킷에서 객체를 검색하려면 GetObject를 선택합니다.
7. Lambda 함수 호출(Invoke Lambda function)에서, 이 자습서의 다음 두 가지 옵션 중 하나를 선택할 수 있습니다.
   - 계정의 함수에서 선택(Choose from functions in your account)을 선택하고, Lambda 함수(Lambda function) 드롭다운 목록에서, 4단계에서 배포한 Lambda 함수(예: **`serverlessrepo-ComprehendPiiRedactionS3ObjectLambda`**)를 선택합니다.
   - ARN 입력(Enter ARN)을 선택한 후, 4단계에서 생성한 Lambda 함수의 Amazon 리소스 이름(ARN)을 입력합니다.
8. Lambda 함수 버전(Lambda function version)에서, $LATEST(4단계에서 배포한 Lambda 함수의 최신 버전)를 선택합니다.
9. (선택 사항) Lambda 함수가 범위 및 부분 번호 헤더가 있는 GET 요청을 인식하고 처리해야 하는 경우 Lambda 함수는 범위를 사용하여 요청을 지원합니다(Lambda function supports requests using range) 및 Lambda 함수는 부분 번호를 사용하여 요청을 지원합니다(Lambda function supports requests using part numbers)를 선택합니다. 그렇지 않으면 이 두 확인란의 선택을 취소합니다.

   S3 객체 Lambda에서 범위 또는 부분 번호를 사용하는 방법에 대한 자세한 내용은 Range 및 partNumber 헤더 작업 섹션을 참조하십시오.
10. (선택 사항) 페이로드 - 선택 사항(Payload - optional)에서, Lambda 함수에 추가 정보를 제공하는 JSON 텍스트를 추가합니다.

    페이로드는 특정 S3 객체 Lambda 액세스 포인트에서 오는 모든 호출에 대한 입력으로 Lambda 함수에 제공할 수 있는 선택적 JSON 텍스트입니다. 동일한 Lambda 함수를 호출하는 여러 객체

Lambda 액세스 포인트에 대해 사용자 지정하기 위해 서로 다른 파라미터를 사용하여 페이로드를 구성할 수 있으므로 Lambda 함수의 유연성을 높일 수 있습니다.

페이로드에 대한 자세한 내용은 이벤트 컨텍스트 형식 및 사용법 섹션을 참조하십시오.

11. 요청 지표 - 선택 사항에서 비활성화 또는 활성화를 선택하여 객체 Lambda 액세스 포인트에 Amazon S3 모니터링을 추가합니다(선택 사항). 요청 지표는 표준 Amazon CloudWatch 요금으로 청구됩니다. 자세한 내용은 CloudWatch 요금을 참조하십시오.
12. 객체 Lambda 액세스 포인트 정책 - 선택 사항(Object Lambda Access Point policy - optional)에서 기본 설정을 그대로 유지합니다.

    리소스 정책을 설정할 수 있습니다(선택 사항). 이 리소스 정책은 지정된 객체 Lambda 액세스 포인트를 사용할 수 있는 권한을 `GetObject` API에 부여합니다.
13. 나머지 설정은 기본값으로 유지하고 객체 Lambda 액세스 포인트 생성(Create Object Lambda Access Point)을 선택합니다.

6단계: S3 객체 Lambda 액세스 포인트를 사용하여 수정된 파일 검색

이제 S3 객체 Lambda가 원본 파일에서 PII 데이터를 수정할 준비가 되었습니다.

S3 객체 Lambda 액세스 포인트를 사용하여 수정된 파일을 검색하려면

S3 객체 Lambda 액세스 포인트를 통해 파일을 검색하도록 요청할 때 S3 객체 Lambda에 대한 `GetObject` API 호출을 수행합니다. S3 객체 Lambda는 Lambda 함수를 호출하여 PII 데이터를 수정하고 변환된 데이터를 표준 S3 `GetObject` API 호출에 대한 응답으로 반환합니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 객체 Lambda 액세스 포인트(Object Lambda Access Points)를 선택합니다.
3. 객체 Lambda 액세스 포인트 페이지에서 5단계에서 생성한 S3 객체 Lambda 액세스 포인트를 선택합니다(예: **`tutorial-pii-object-lambda-accesspoint`**).
4. S3 객체 Lambda 액세스 포인트의 객체 탭에서 2단계에서 S3 버킷에 업로드한 파일과 이름이 같은 파일을 선택합니다(예: `tutorial.txt`)

    이 파일에는 변환된 모든 데이터가 포함되어야 합니다.
5. 변환된 데이터를 보려면 열기(Open) 또는 다운로드(Download)를 선택합니다.

    다음 예제와 같이 수정된 파일이 표시되어야 합니다.

```
Hello *********. Your AnyCompany Financial Services,
LLC credit card account ******************* has a minimum payment
of $24.53 that is due by *********. Based on your autopay settings,
we will withdraw your payment on the due date from your
bank account ********** with the routing number *********.

Your latest statement was mailed to ********************************.
After your payment is received, you will receive a confirmation
text message at ************.
If you have questions about your bill, AnyCompany Customer Service
is available by phone at ************ or
email at **********************.
```

7단계: 정리

실습용으로만 S3 객체 Lambda를 통해 데이터를 수정한 경우에는 요금이 발생하지 않도록 할당한 AWS 리소스를 삭제합니다.

하위 단계

- 객체 Lambda 액세스 포인트 삭제
- S3 액세스 포인트 삭제
- Lambda 함수 삭제
- CloudWatch 로그 그룹 삭제
- S3 소스 버킷에서 원본 파일 삭제
- S3 소스 버킷 삭제
- Lambda 함수용 IAM 역할 삭제
- IAM 사용자용 고객 관리형 정책 삭제
- IAM 사용자 삭제

객체 Lambda 액세스 포인트 삭제

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 객체 Lambda 액세스 포인트(Object Lambda Access Points)를 선택합니다.

3. 객체 Lambda 액세스 포인트 페이지에서 5단계에서 생성한 S3 객체 Lambda 액세스 포인트 왼쪽에 있는 옵션 버튼을 선택합니다(예: **tutorial-pii-object-lambda-accesspoint**).
4. 삭제를 선택합니다.
5. 표시되는 텍스트 필드에 객체 Lambda 액세스 포인트 이름을 입력하여 삭제 여부를 확인하고 삭제를 선택합니다.

S3 액세스 포인트 삭제

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 탐색 창에서 액세스 포인트(Access Points)를 선택합니다.
3. 3단계에서 생성한 액세스 포인트(예: **tutorial-pii-access-point**)로 이동한 후 액세스 포인트 이름 옆에 있는 옵션 버튼을 선택합니다.
4. 삭제를 선택합니다.
5. 표시되는 텍스트 필드에 액세스 포인트 이름을 입력하여 삭제 여부를 확인하고 삭제(Delete)를 선택합니다.

Lambda 함수 삭제

1. https://console.aws.amazon.com/lambda/에 있는 AWS Lambda 콘솔의 왼쪽 탐색 창에서 함수(Functions)를 선택합니다.
2. 4단계에서 생성한 함수(예: **serverlessrepo-ComprehendPiiRedactionS3ObjectLambda**)를 선택합니다.
3. 작업을 선택한 후 삭제를 선택합니다.
4. 함수 삭제(Delete function) 대화 상자에서 삭제(Delete)를 선택합니다.

CloudWatch 로그 그룹 삭제

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그 그룹(Log groups)을 선택합니다.
3. 이름이 4단계에서 생성한 Lambda 함수로 끝나는 로그 그룹을 찾습니다(예: **serverlessrepo-ComprehendPiiRedactionS3ObjectLambda**).
4. 작업(Actions)을 선택한 후 로그 그룹 삭제(Delete log group(s))를 선택합니다.

5. 로그 그룹 삭제(Delete log group(s)) 대화 상자에서 삭제(Delete)를 선택합니다.

S3 소스 버킷에서 원본 파일 삭제

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 이름(Bucket name) 목록에서, 2단계에서 원본 파일을 업로드한 버킷의 이름을 선택합니다(예: **`tutorial-bucket`**).
4. 삭제할 객체의 이름 왼쪽에 있는 확인란을 선택합니다(예: `tutorial.txt`).
5. 삭제를 선택합니다.
6. 객체 삭제(Delete objects) 페이지의 객체를 영구적으로 삭제하시겠습니까?(Permanently delete objects?) 섹션에서 텍스트 상자에 **`permanently delete`**를 입력하여 이 객체의 삭제 여부를 확인합니다.
7. 객체 삭제를 선택합니다.

S3 소스 버킷 삭제

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷(Buckets) 목록에서, 1단계에서 생성한 버킷의 이름 옆에 있는 옵션 버튼을 선택합니다(예: **`tutorial-bucket`**).
4. 삭제를 선택합니다.
5. 버킷 삭제(Delete objects) 페이지의 텍스트 필드에 버킷 이름을 입력하여 버킷의 삭제 여부를 확인한 다음 버킷 삭제(Delete bucket)를 선택합니다.

Lambda 함수용 IAM 역할 삭제

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할(Roles)을 선택한 후 삭제할 역할 이름 옆에 있는 확인란을 선택합니다. 역할 이름은 4단계에서 배포한 Lambda 함수의 이름으로 시작합니다(예: **`serverlessrepo-ComprehendPiiRedactionS3ObjectLambda`**).

3. 삭제를 선택합니다.
4. 삭제(Delete) 대화 상자에서 텍스트 입력 필드에 역할 이름을 입력하여 삭제 여부를 확인합니다. 그런 다음 삭제를 선택합니다.

IAM 사용자용 고객 관리형 정책 삭제

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 정책을 선택합니다.
3. 정책(Policies) 페이지의 검색 상자에서, 사전 조건(Prerequisites)에서 생성한 고객 관리형 정책의 이름(예: **`tutorial-serverless-application-repository`**)을 입력하여 정책 목록을 필터링합니다. 삭제할 정책 이름 옆에 있는 옵션 버튼을 선택합니다.
4. 작업을 선택한 후 삭제를 선택합니다.
5. 표시되는 텍스트 필드에 해당 이름을 입력하여 이 정책의 삭제 여부를 확인하고 삭제(Delete)를 선택합니다.

IAM 사용자 삭제

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 사용자(Users)를 선택한 후 삭제할 사용자 이름 옆에 있는 확인란을 선택합니다.
3. 페이지 상단에서 삭제(Delete)를 선택합니다.
4. **`### ##`**을 삭제하시겠습니까?(Delete user name?) 대화 상자에서 텍스트 입력 필드에 역할 이름을 입력하여 사용자의 삭제 여부를 확인합니다. 삭제를 선택합니다.

다음 단계

이 자습서를 완료한 후에는 다음과 같은 관련 사용 사례를 더 자세히 탐색할 수 있습니다.

- 여러 S3 객체 Lambda 액세스 포인트를 생성하고 데이터 접근자의 비즈니스 요구에 따라 특정 유형의 PII를 수정하도록 다르게 구성된 사전 빌드된 Lambda 함수를 사용하여 이 액세스 포인트를 활성화할 수 있습니다.

  각 유형의 사용자는 IAM 역할을 수임하고 하나의 S3 객체 Lambda 액세스 포인트(IAM 정책을 통해 관리됨)에만 액세스할 수 있습니다. 그런 다음 다른 수정 사용 사례에 대해 구성된 각 `ComprehendPiiRedactionS3ObjectLambda` Lambda 함수를 다른 S3 객체 Lambda 액세스 포

인트에 연결합니다. 각 S3 객체 Lambda 액세스 포인트에 대해 공유 데이터 집합을 저장하는 S3 버킷에서 데이터를 읽을 수 있는 지원 S3 액세스 포인트를 소유할 수 있습니다.

사용자가 S3 액세스 포인트를 통해서만 버킷에서 읽을 수 있도록 허용하는 S3 버킷 정책을 생성하는 방법에 대한 자세한 내용은 액세스 포인트를 사용하도록 IAM 정책 구성 섹션을 참조하십시오.

Lambda 함수, S3 액세스 포인트 및 S3 객체 Lambda 액세스 포인트에 액세스할 수 있는 권한을 사용자에게 부여하는 방법에 대한 자세한 내용은 객체 Lambda 액세스 포인트에 대한 IAM 정책 구성 섹션을 참조하십시오.

- 자체 Lambda 함수를 빌드하고 사용자 지정된 Lambda 함수와 함께 S3 객체 Lambda를 사용하여 특정 데이터 요구 사항을 충족할 수 있습니다.

  예를 들어 다양한 데이터 값을 탐색하려면 S3 객체 Lambda와 추가적인 Amazon Comprehend 기능(예: 엔터티 인식, 핵심 구문 인식, 감정 분석, 문서 분류)을 사용하는 자체 Lambda 함수를 사용하여 데이터를 처리할 수 있습니다. 또한 S3 객체 Lambda를 Amazon Comprehend Medical(HIPAA 적격 NLP 서비스)과 함께 사용하여 컨텍스트 인식 방식으로 데이터를 분석하고 추출할 수 있습니다.

  S3 객체 Lambda 및 자체 Lambda 함수를 사용하여 데이터를 변환하는 방법에 대한 자세한 내용은 자습서: S3 객체 Lambda를 사용하여 애플리케이션의 데이터 변환 섹션을 참조하십시오.

S3 Object Lambda 디버깅 및 문제 해결

Amazon S3 객체 Lambda 액세스 지점에 대한 요청에서 Lambda 함수 호출 또는 실행에 문제가 발생하면 새로운 오류 응답이 발생할 수 있습니다. 이러한 오류는 표준 Amazon S3 오류와 동일한 형식을 따릅니다. S3 객체 Lambda 오류에 대한 자세한 내용은 Amazon Simple Storage Service API 참조에서 S3 객체 Lambda 오류 코드 목록을 참조하십시오.

일반적인 Lambda 함수 디버깅에 대한 자세한 내용은 AWS Lambda 개발자 안내서에서 Lambda 애플리케이션 모니터링 및 문제 해결을 참조하십시오.

표준 Amazon S3 오류에 대한 자세한 내용은 Amazon Simple Storage Service API 참조에서 오류 응답을 참조하십시오.

Amazon CloudWatch에서 객체 Lambda 액세스 포인트에 대한 요청 지표를 활성화할 수 있습니다. 이러한 지표는 액세스 지점의 운영 성능을 모니터링하는 데 도움이 됩니다. 객체 Lambda 액세스 포인트를 생성하는 동안 또는 생성한 후에 요청 지표를 활성화할 수 있습니다. 자세한 내용은 CloudWatch의 S3 객체 Lambda 요청 지표 섹션을 참조하세요.

객체 Lambda 액세스 포인트에 수행된 요청에 대한 보다 세부적인 로깅 정보를 얻으려면 AWS CloudTrail 데이터 이벤트를 활성화할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서의 추적을 위해 데이터 이벤트 로깅을 참조하십시오.

S3 객체 Lambda 자습서의 경우 다음을 참조하십시오.

- 자습서: S3 객체 Lambda를 사용하여 애플리케이션의 데이터 변환
- 자습서: S3 객체 Lambda 및 Amazon Comprehend를 사용하여 PII 데이터 감지 및 수정
- Tutorial: Using S3 Object Lambda to dynamically watermark images as they are retrieved(자습서: S3 객체 Lambda를 사용하여 이미지를 검색할 때 동적으로 워터마크 지정)

표준 액세스 포인트에 대한 자세한 내용은 액세스 포인트로 공유 데이터세트에 대한 액세스 관리 섹션을 참조하십시오.

버킷 작업에 대한 자세한 내용은 범용 버킷 개요 단원을 참조하십시오. 객체 작업에 대한 자세한 내용은 Amazon S3 객체 개요을 참조하십시오.

Batch Operations를 사용하여 대량으로 객체 작업 수행

S3 배치 작업을 사용하여 Amazon S3 객체에 대해 대규모 배치 작업을 수행할 수 있습니다. S3 배치 작업은 지정된 Amazon S3 객체 목록에 대해 단일 작업을 수행할 수 있습니다. 단일 작업으로 엑사바이트 규모의 데이터가 포함된 수십억 개의 객체에서 지정된 작업을 수행할 수 있습니다. Amazon S3는 진행 상황을 추적하고 알림을 보내며 모든 작업에 대한 자세한 완료 보고서를 저장하여 감사 가능한 완전관리형 서버리스 환경을 제공합니다. Amazon S3 콘솔, AWS CLI, AWS SDK 또는 Amazon S3 REST API를 통해 S3 Batch Operations를 사용할 수 있습니다.

S3 배치 작업을 사용하여 객체를 복사하고 객체 태그 또는 액세스 제어 목록(ACL)을 설정합니다. S3 Glacier Flexible Retrieval에서 객체 복원을 시작하거나 AWS Lambda 함수를 호출하여 객체로 사용자 정의 작업을 수행할 수도 있습니다. 사용자 지정 객체 목록에서 이러한 작업을 수행하거나 Amazon S3 인벤토리 보고서를 사용하여 객체 목록을 손쉽게 생성할 수 있습니다. Amazon S3 Batch Operations는 이미 Amazon S3에서 사용하고 있는 것과 동일한 Amazon S3 API 작업을 사용합니다.

> **Note**
>
> Amazon S3 Express One Zone 스토리지 클래스를 디렉터리 버킷과 함께 사용하는 방법에 대한 자세한 내용은 S3 Express One Zone 및 디렉터리 버킷 작업 섹션을 참조하세요. S3

Express One Zone 및 디렉터리 버킷에서 배치 작업을 사용하는 방법에 대한 자세한 내용은 디렉터리 버킷에 배치 작업 사용 섹션을 참조하세요.

S3 배치 작업 기본 사항

S3 배치 작업을 사용하여 Amazon S3 객체에 대해 대규모 배치 작업을 수행할 수 있습니다. S3 배치 작업은 지정된 Amazon S3 객체 목록에 대해 단일 작업을 실행할 수 있습니다.

용어

이 섹션에서는 매니페스트, 작업(jobs), 작업(operations) 및 태스크라는 용어를 사용하며, 이 용어는 다음과 같이 정의됩니다.

매니페스트

매니페스트는 Amazon S3가 작업을 수행할 객체 키를 포함하는 Amazon S3 객체입니다. Batch Operations 작업을 만들려면 매니페스트를 제공해야 합니다. 사용자 생성 매니페스트에는 버킷 이름, 객체 키, 각 객체의 객체 버전(선택 사항)이 포함되어야 합니다. 사용자 생성 매니페스트를 제공하는 경우 Amazon S3 인벤토리 보고서 또는 CSV 파일 형식이어야 합니다.

또한. 작업을 만들 때 지정하는 객체 필터 기준에 따라 Amazon S3에서 자동으로 매니페스트를 생성하도록 할 수도 있습니다. 이 옵션은 Amazon S3 콘솔에서 만드는 S3 Batch Replication 작업이나 AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하여 만드는 작업 유형에 사용할 수 있습니다.

작업

작업(job)은 S3 배치 작업의 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 수행합니다.

작업

작업(operation)은 배치 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. 태스크는 하나의 객체에 대해 작업(job)의 작업(operation)을 수행하기 위한 Amazon S3 또는 AWS Lambda API 작업에 대한 단일 호출을 나타냅니다. 작업 수명 주기 동안 S3 배치 작업은 매니페스트에 지정된 각 객체에 대해 하나의 태스크(task)를 생성합니다.

S3 배치 작업 건의 작동 방식

작업(job)은 S3 배치 작업의 기본 작업 단위입니다. 작업은 객체 목록에 대해 지정된 작업을 실행하는 데 필요한 모든 정보를 포함합니다. 작업을 생성하려면 S3 배치 작업에 객체 목록을 제공하고 해당 객체에서 수행할 작업을 지정합니다.

S3 배치 작업에서 지원하는 작업에 대한 자세한 내용은 S3 배치 작업에서 지원하는 작업 섹션을 참조하세요.

배치 작업은 매니페스트에 포함된 모든 객체에 대해 지정된 작업을 수행합니다. 매니페스트는 배치 작업에서 처리할 객체를 나열하며 버킷에 객체로 저장됩니다. CSV(쉼표로 구분된 값) 형식의 S3 Inventory를 사용한 데이터 카탈로그화 및 분석 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 수행된 각 작업(task)의 결과를 설명합니다. 작업 모니터링에 대한 자세한 내용은 S3 배치 작업 건 관리 섹션을 참조하세요.

S3 Batch Operations과 관련된 비용이 있습니다. 완료 전에 취소된 작업을 포함하여 Batch Operations 생성에 대해 요금이 청구됩니다. 자세한 내용은 Amazon S3 요금을 참조하십시오.

S3 Batch Operations 작업은 기본적으로 모든 작업에서 최대 40억 개의 객체를 처리할 수 있습니다. 특히 복사, 객체 태그 지정, Object Lock, AWS Lambda 함수 간접 호출 및 배치 복제 작업은 최대 200억 개의 객체를 지원할 수 있습니다. AWS 계정당 활성 배치 복제 작업 수는 6개로 제한됩니다. 배치 작업 만들기를 시작하려면 Creating an S3 Batch Operations job을 참조하세요.

S3 배치 작업 자습서

다음 자습서에서는 일부 배치 작업 태스크를 처음부터 끝까지 수행하는 절차를 보여줍니다.

- 자습서: S3 Batch Operations를 통해 비디오 일괄 트랜스코딩

배치 작업에 대한 권한 부여

S3 배치 작업을 생성 및 실행하기 전에, 필요한 권한을 부여해야 합니다. Amazon S3 배치 작업을 생성하려면 `s3:CreateJob` 사용자 권한이 필요합니다. 작업을 생성한 동일한 엔터티에는 작업

에 대해 지정된 AWS Identity and Access Management(IAM) 역할을 Batch Operations에 전달하는 `iam:PassRole` 권한도 있어야 합니다.

IAM 리소스 지정에 대한 일반적인 정보는 IAM 사용 설명서의 IAM JSON 정책인 리소스 요소를 참조하세요. 다음 섹션에서는 IAM 역할 생성 및 정책 연결에 대한 정보를 제공합니다.

주제

- S3 배치 작업 IAM 역할 생성
- 권한 연결 정책

S3 배치 작업 IAM 역할 생성

Amazon S3에는 사용자를 대신해 S3 배치 작업을 수행할 권한이 있어야 합니다. AWS Identity and Access Management(IAM) 역할을 통해 이러한 권한을 부여합니다. 이 섹션에서는 IAM 역할을 생성할 때 사용하는 신뢰 및 권한 정책의 예제를 제공합니다. 자세한 내용은 IAM 사용 설명서에서 IAM 역할을 참조하세요. 예제는 작업 태그를 사용하여 Batch Operations에 대한 권한 제어 및 S3 배치 작업을 사용하여 객체 복사 섹션을 참조하세요.

IAM 정책에서 조건 키를 사용하여 S3 배치 작업 건에 대한 액세스 권한을 필터링할 수도 있습니다. Amazon S3에 사용되는 조건 키의 전체 목록과 자세한 내용은 서비스 승인 참조에서 Amazon S3에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

신뢰 정책

S3 배치 작업 서비스 보안 주체가 IAM 역할을 맡도록 허용하려면 다음 신뢰 정책을 역할에 연결합니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
```

```
    ]
}
```

권한 연결 정책

작업(operation) 유형에 따라 다음 정책 중 하나를 연결할 수 있습니다.

사용 권한을 구성하기 전에 다음 사항에 유의하세요.

- 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 S3 배치 작업에서는 manifest.json 객체 및 연결된 모든 CSV 데이터 파일을 읽을 수 있는 권한이 필요합니다.
- 객체의 버전 ID를 지정할 때는 `s3:GetObjectVersion`과 같은 버전 관련 권한만 필요합니다.
- 암호화된 객체에 대해 S3 배치 작업을 실행하는 경우 IAM 역할에도 해당 객체를 암호화하는 데 사용되는 AWS KMS 키에 대한 액세스 권한이 있어야 합니다.
- AWS KMS로 암호화된 인벤토리 보고서 매니페스트를 제출하는 경우 IAM 정책에 manifest.json 객체 및 연결된 모든 CSV 데이터 파일에 대한 `"kms:Decrypt"` 및 `"kms:GenerateDataKey"` 권한이 포함되어 있어야 합니다.
- 액세스 제어 목록(ACL)이 활성화되어 있고 다른 AWS 계정에 있는 버킷의 매니페스트를 Batch Operations 작업에서 생성하는 경우, 배치 작업에 대해 구성된 IAM 역할의 IAM 정책에서 `s3:PutObjectAcl` 권한을 부여해야 합니다. 이 권한을 포함하지 않은 경우 배치 작업은 `Error occurred when preparing manifest: Failed to write manifest` 오류와 함께 실패합니다.

객체 복사: PutObject

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectTagging"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
```

```
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectTagging",
                "s3:ListBucket"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-source-bucket",
                "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*"
            ]
        }
    ]
}
```

객체 태그 지정 대체: PutObjectTagging

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

```
      "Action":[
        "s3:PutObjectTagging",
        "s3:PutObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject"
      ],
      "Resource":[
        "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*"
      ]
    }
  ]
}
```

객체 태그 지정 삭제: DeleteObjectTagging

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "s3:DeleteObjectTagging",
              "s3:DeleteObjectVersionTagging"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
            ]
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*"
            ]
        }
    ]
}
```

액세스 제어 목록 대체: PutObjectAcl

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObjectAcl",
        "s3:PutObjectVersionAcl"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
```

```
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject"
      ],
      "Resource":[
        "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*"
      ]
    }
  ]
}
```

객체 복원: RestoreObject

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
          "s3:RestoreObject"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject"
      ],
      "Resource":[
        "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*"
```

```
      ]
    }
  ]
}
```

Object Lock 보존 적용: PutObjectRetention

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetBucketObjectLockConfiguration",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObjectRetention",
                "s3:BypassGovernanceRetention"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
```

```
                "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*"
            ]
        }
    ]
}
```

Object Lock 법적 보존 적용: PutObjectLegalHold

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetBucketObjectLockConfiguration",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "s3:PutObjectLegalHold",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*"
            ]
```

```
        }
    ]
}
```

기존 객체 복제: S3 생성 매니페스트를 사용한 InitiateReplication

S3 생성 매니페스트를 사용 및 저장하는 경우 이 정책을 사용합니다. 기존 객체 복제에 Batch Operations를 사용하는 방법에 대한 자세한 내용은 배치 복제를 사용한 기존 객체 복제 섹션을 참조하세요.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:InitiateReplication"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
         ]
      },
      {
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:PutInventoryConfiguration"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket"
         ]
      },
      {
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
         ]
      },
      {
```

```
         "Effect":"Allow",
         "Action":[
            "s3:PutObject"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*",
            "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
         ]
      }
   ]
}
```

기존 객체 복제: 사용자 매니페스트를 사용한 InitiateReplication

사용자 제공 매니페스트를 사용하는 경우 이 정책을 사용합니다. 기존 객체 복제에 Batch Operations를 사용하는 방법에 대한 자세한 내용은 배치 복제를 사용한 기존 객체 복제 섹션을 참조하세요.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:InitiateReplication"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
         ]
      },
      {
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject"
```

```
            ],
            "Resource":[
               "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*"
            ]
         }
      ]
}
```

S3 배치 작업 건 생성

Amazon S3 배치 작업을 사용하면 특정 Amazon S3 객체 목록에서 대규모 배치 작업을 수행할 수 있습니다. 이 섹션에서는 S3 배치 작업을 생성하는 데 필요한 정보와 `CreateJob` 요청의 결과를 설명합니다. 또한 Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 및 AWS SDK for Java를 사용하여 배치 작업 생성 지침도 제공합니다.

S3 배치 작업을 생성할 때 모든 작업 또는 실패한 작업에 대한 완료 보고서를 요청할 수 있습니다. 하나 이상의 작업이 성공적으로 호출되면 S3 배치 작업은 완료, 실패 또는 취소된 작업에 대한 보고서를 생성합니다. 자세한 내용은 예: S3 배치 작업 완료 보고서 섹션을 참조하세요.

주제

- 배치 작업 요청 요소
- 매니페스트 지정

배치 작업 요청 요소

S3 배치 작업 건을 생성하려면 다음 정보를 제공해야 합니다.

작업

매니페스트 내 객체에 대해 S3 배치 작업이 실행할 작업을 지정합니다. 각 작업 유형은 해당 작업과 관련된 파라미터를 허용합니다. 배치 작업을 사용하면 작업을 대량으로 수행함으로써 각 객체에 해당 작업을 하나씩 수행할 때와 같은 결과를 얻을 수 있습니다.

매니페스트

매니페스트는 S3 배치 작업에서 지정된 작업을 실행해야 하는 모든 객체의 목록입니다. 다음 방법을 사용하여 배치 작업 건의 매니페스트를 지정할 수 있습니다.

- CSV 형식의 사용자 지정된 객체 목록을 수동으로 만듭니다.
- 기존 CSV 형식의 S3 Inventory를 사용한 데이터 카탈로그화 및 분석 보고서를 선택합니다.

- 작업을 생성할 때 지정한 객체 필터 기준에 따라 자동으로 매니페스트를 생성하도록 배치 작업에 지시합니다. 이 옵션은 Amazon S3 콘솔에서 생성하는 배치 복제 작업 또는 AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용하여 생성하는 작업 유형에 사용할 수 있습니다.

> **Note**
>
> - 매니페스트를 지정하는 방법과 관계없이 목록 자체는 범용 버킷에 저장해야 합니다. 배치 작업은 디렉터리 버킷에서 기존 매니페스트를 가져올 수 없으며 생성된 매니페스트를 디렉터리 버킷에 저장할 수 없습니다. 하지만 매니페스트에 설명된 객체는 디렉터리 버킷에 저장할 수 있습니다. 자세한 내용은 디렉터리 버킷을 참조하세요.
> - 매니페스트 내 객체가 버전이 지정된 버킷에 있는 경우, 객체의 버전 ID를 지정하면 배치 작업이 지정된 버전에서 실행됩니다. 버전 ID를 지정하지 않는 경우 배치 작업 시 객체의 마지막 버전에서 작업이 실행됩니다. 매니페스트에 버전 ID 필드가 포함된 경우, 매니페스트 내 모든 객체에 대한 버전 ID를 제공해야 합니다.

자세한 내용은 매니페스트 지정 섹션을 참조하세요.

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. 사용자는 자신에게 적합한 번호 지정 시스템을 선택할 수 있습니다. 예를 들어 모든 복원(`RestoreObject`) 작업에 1의 우선순위를, 모든 복사(`CopyObject`) 작업에 2의 우선순위를, 모든 액세스 제어 목록(ACL) 교체(`PutObjectAcl`) 작업에 3의 우선순위를 할당할 수 있습니다.

S3 배치 작업은 우선순위 번호에 따라 작업의 우선순위를 부여하며 다만 정확한 순서가 보장되지는 않습니다. 따라서 어떤 작업이 다른 작업보다 먼저 시작되거나 끝나도록 하기 위해 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 해당 작업 건(job)에 지정된 작업(operation)을 수행하기에 충분한 권한이 있어야 합니다. 예를 들어 `CopyObject` 작업을 실행하려면 IAM 역할이 소스 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 역할 단원을 참조하세요.

Amazon S3 권한에 대한 자세한 내용은 Amazon S3의 정책 작업 섹션을 참조하세요.

> **Note**
>
> 디렉터리 버킷에서 작업을 수행하는 배치 작업 건에는 특정 권한이 필요합니다. 자세한 내용은 AWS Identity and Access Management (IAM) for S3 Express One Zone을 참조하세요.

Report

S3 배치 작업에서 완료 보고서를 생성할지 여부를 지정합니다. 작업 완료 보고서를 요청하면 이 요소에서 보고서에 대한 파라미터도 제공해야 합니다. 필요한 정보에는 다음이 포함됩니다.

- 보고서를 저장할 버킷

  > **Note**
  >
  > 보고서는 범용 버킷에 저장해야 합니다. 배치 작업에서는 보고서를 디렉터리 버킷에 저장할 수 없습니다. 자세한 내용은 디렉터리 버킷을 참조하세요.

- 보고서의 형식
- 보고서에 모든 작업 또는 실패한 작업의 세부 정보만 포함할지 여부
- 선택적 접두사 문자열

> **Note**
>
> 완료 보고서는 항상 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)로 암호화됩니다.

태그(선택 사항)

태그를 추가하여 S3 배치 작업 건에 대한 레이블을 지정하고 액세스를 제어할 수 있습니다. 배치 작업 건의 책임자를 식별하거나 사용자가 배치 작업 건과 상호작용하는 방법을 제어하기 위해 태그를 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. 예를 들어,

"Department=Finance" 태그로 작업이 생성된 경우 CreateJob 작업을 호출할 수 있는 권한을 사용자에게 부여할 수 있습니다.

태그가 연결된 작업을 생성할 수 있으며 작업을 생성한 후 작업에 태그를 추가할 수 있습니다.

자세한 내용은 the section called “태그 사용” 섹션을 참조하세요.

Description(선택 사항)

작업 건을 추적하고 모니터링하기 위해 최대 256자의 설명을 제공할 수도 있습니다. Amazon S3는 작업 건에 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.

매니페스트 지정

매니페스트는 Amazon S3가 작업을 수행할 객체 키를 포함하는 Amazon S3 객체입니다. 다음 방법 중 하나를 사용하여 매니페스트를 제공할 수 있습니다.

- 새 매니페스트 파일을 수동으로 생성합니다.
- 기존 매니페스트를 사용합니다.
- 작업을 생성할 때 지정한 객체 필터 기준에 따라 자동으로 매니페스트를 생성하도록 배치 작업에 지시합니다. 이 옵션은 Amazon S3 콘솔에서 생성하는 배치 복제 작업 또는 AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용하여 생성하는 작업 유형에 사용할 수 있습니다.

> **Note**
>
> - Amazon S3 Batch Operations는 리전 간 매니페스트 생성을 지원하지 않습니다.
> - 매니페스트를 지정하는 방법과 관계없이 목록 자체는 범용 버킷에 저장해야 합니다. 배치 작업은 디렉터리 버킷에서 기존 매니페스트를 가져올 수 없으며 생성된 매니페스트를 디렉터리 버킷에 저장할 수 없습니다. 하지만 매니페스트에 설명된 객체는 디렉터리 버킷에 저장할 수 있습니다. 자세한 내용은 디렉터리 버킷을 참조하세요.

매니페스트 파일 생성

수동으로 매니페스트 파일을 생성하려면 매니페스트 객체 키, ETag(엔터티 태그) 및 선택적으로 버전 ID를 CSV 형식의 목록으로 지정합니다. 매니페스트의 콘텐츠는 URL로 인코딩되어야 합니다.

기본적으로 Amazon S3가 Amazon S3 관리형 키(SSE-S3)를 통한 서버 측 암호화를 사용하여 Amazon S3 버킷에 업로드된 매니페스트를 암호화합니다. 고객 제공 키를 사용한 서버 측 암호화(SSE-C) 키를 사용한 매니페스트는 지원되지 않습니다. AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 통한 서버 측 암호화를 사용하는 매니페스트는 CSV 형식의 인벤토리 보고서를 사용할 때만 지원됩니다. AWS KMS로 수동으로 생성된 매니페스트를 사용하는 것은 지원되지 않습니다.

매니페스트에는 버킷 이름, 객체 키, 각 객체의 객체 버전(선택 사항)이 포함되어야 합니다. 매니페스트의 다른 모든 필드는 S3 배치 작업에서 사용되지 않습니다.

> **Note**
>
> 매니페스트 내 객체가 버전이 지정된 버킷에 있는 경우, 객체의 버전 ID를 지정하면 배치 작업이 지정된 버전에서 실행됩니다. 버전 ID를 지정하지 않는 경우 배치 작업 시 객체의 마지막 버전에서 작업이 실행됩니다. 매니페스트에 버전 ID 필드가 포함된 경우, 매니페스트 내 모든 객체에 대한 버전 ID를 제공해야 합니다.

다음은 버전 ID가 없는 CSV 형식의 매니페스트 예입니다.

```
amzn-s3-demo-bucket1,objectkey1
amzn-s3-demo-bucket1,objectkey2
amzn-s3-demo-bucket1,objectkey3
amzn-s3-demo-bucket1,photos/jpgs/objectkey4
amzn-s3-demo-bucket1,photos/jpgs/newjersey/objectkey5
amzn-s3-demo-bucket1,object%20key%20with%20spaces
```

다음은 버전 ID를 포함하는 CSV 형식의 매니페스트 예시입니다.

```
amzn-s3-demo-bucket1,objectkey1,PZ9ibn9D5lP6p298B7S9_ceqx1n5EJ0p
amzn-s3-demo-bucket1,objectkey2,YY_ouuAJByNW1LRBfFMfxMge7XQWxMBF
amzn-s3-demo-bucket1,objectkey3,jbo9_jhdPEyB4RrmOxWS0kU0EoNrU_oI
amzn-s3-demo-bucket1,photos/jpgs/objectkey4,6EqlikJJxLTsHsnbZbSRffn24_eh5Ny4
amzn-s3-demo-bucket1,photos/jpgs/newjersey/objectkey5,imHf3FAiRsvBW_EHB8GOu.NHunHO1gVs
amzn-s3-demo-bucket1,object%20key%20with%20spaces,9HkPvDaZY5MVbMhn6TMn1YTb5ArQAo3w
```

기존 매니페스트 파일 지정

다음 두 형식 중 하나를 사용하여 작업 생성 요청에 매니페스트 파일을 지정할 수 있습니다.

- Amazon S3 인벤토리 보고서 - CSV 형식의 Amazon S3 인벤토리 보고서여야 합니다. 인벤토리 보고서와 연결된 `manifest.json` 파일을 지정해야 합니다. 인벤토리 보고서에 대한 자세한 내용은 S3 Inventory를 사용한 데이터 카탈로그화 및 분석 섹션을 참조하세요. 인벤토리 보고서가 버전 ID를 포함할 경우, S3 배치 작업이 특정 객체 버전에만 작용합니다.

  > **Note**
  >
  > - S3 배치 작업은 SSE-KMS로 암호화된 CSV 인벤토리 보고서를 지원합니다.
  > - SSE-KMS로 암호화된 인벤토리 보고서 매니페스트를 제출하는 경우 IAM 정책에 `manifest.json` 객체 및 연결된 모든 CSV 데이터 파일에 대한 `"kms:Decrypt"` 및 `"kms:GenerateDataKey"` 권한이 포함되어 있어야 합니다.

- 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 참조에서 JobManifestSpec을 참조하세요.

  > **Note**
  >
  > S3 배치 작업은 SSE-KMS로 암호화된 CSV 매니페스트 파일을 지원하지 않습니다.

> **Important**
>
> 수동으로 생성된 매니페스트와 버전이 지정된 버킷을 사용하는 경우 객체의 버전 ID를 지정하는 것이 좋습니다. 작업을 생성하면 S3 배치 작업이 작업을 실행하기 전에 전체 매니페스트를 구문 분석합니다. 하지만 버킷 상태 "스냅샷"을 생성하지는 않습니다.
> 매니페스트에는 수십억 개의 객체가 포함될 수 있으므로 작업 실행에 오랜 시간이 걸릴 수 있으며, 그러한 경우 작업이 실행되는 객체의 버전에 영향을 미칠 수 있습니다. 작업이 실행되는 동안 객체를 새 버전으로 덮어쓰고 그 객체에 버전 ID를 지정하지 않았다고 가정해 봅시다. 이 경우 Amazon S3는 작업 생성 시 존재했던 버전이 아니라 객체의 최신 버전에서 작업을 수행합니다. 이 문제를 피하는 유일한 방법은 매니페스트에 나열된 객체에 버전 ID를 지정하는 것입니다.

매니페스트 자동 생성

작업을 생성할 때 지정한 객체 필터 기준에 따라 자동으로 매니페스트를 생성하도록 Amazon S3에 지시할 수 있습니다. 이 옵션은 Amazon S3 콘솔에서 생성하는 배치 복제 작업 또는 AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용하여 생성하는 작업 유형에 사용할 수 있습니다. Batch Replication에 대한 자세한 내용은 배치 복제를 사용한 기존 객체 복제 섹션을 참조하세요.

매니페스트를 자동으로 생성하려면 작업 생성 요청의 일부로 다음 요소를 지정합니다.

- 소스 객체를 포함하는 버킷에 대한 정보(버킷 소유자 및 Amazon 리소스 이름(ARN) 등)
- 매니페스트 출력에 대한 정보(매니페스트 파일 생성 플래그, 출력 버킷 소유자, ARN, 접두사, 파일 형식, 암호화 유형 등)
- 만든 날짜, 키 이름, 크기 및 스토리지 클래스별로 객체를 필터링하는 선택적 기준입니다. 복제 작업의 경우 태그를 사용하여 객체를 필터링할 수도 있습니다.

객체 필터 기준

자동으로 생성된 매니페스트에 포함할 객체 목록을 필터링하려면 다음 기준을 지정할 수 있습니다. 자세한 내용은 Amazon S3 API 참조의 JobManifestGeneratorFilter 섹션을 참조하세요.

CreatedAfter

제공된 경우 생성된 매니페스트에는 이 시점 후에 생성된 소스 버킷 객체만 포함됩니다.

CreatedBefore

제공된 경우 생성된 매니페스트에는 이 시점 전에 생성된 소스 버킷 객체만 포함됩니다.

EligibleForReplication

제공된 경우 생성된 매니페스트에는 소스 버킷의 복제 구성에 따라 복제에 적합한 객체만 포함됩니다.

KeyNameConstraint

제공된 경우 생성된 매니페스트에는 MatchAnySubstring, MatchAnyPrefix 및 MatchAnySuffix에 지정된 문자열 제약 조건과 객체 키가 일치하는 소스 버킷 객체만 포함됩니다.

MatchAnySubstring - 제공된 경우 생성된 매니페스트에는 지정된 문자열이 객체 키 문자열 내에 있으면 객체가 포함됩니다.

MatchAnyPrefix - 제공된 경우 생성된 매니페스트에는 지정된 문자열이 객체 키 문자열의 시작 부분에 있으면 객체가 포함됩니다.

MatchAnySuffix - 제공된 경우 생성된 매니페스트에는 지정된 문자열이 객체 키 문자열의 끝 부분에 있으면 객체가 포함됩니다.

MatchAnyStorageClass

제공된 경우 생성된 매니페스트에는 지정된 스토리지 클래스와 함께 저장된 소스 버킷 객체만 포함됩니다.

ObjectReplicationStatuses

제공된 경우 생성된 매니페스트에는 지정된 복제 상태 중 하나를 가진 소스 버킷 객체만 포함됩니다.

ObjectSizeGreaterThanBytes

제공된 경우 생성된 매니페스트에는 파일 크기가 지정된 바이트 수보다 큰 소스 버킷 객체만 포함됩니다.

ObjectSizeLessThanBytes

제공된 경우 생성된 매니페스트에는 파일 크기가 지정된 바이트 수보다 작은 소스 버킷 객체만 포함됩니다.

> **Note**
>
> 매니페스트를 자동으로 생성한 대부분의 작업은 복제할 수 없습니다. `KeyNameConstraint`, `MatchAnyStorageClass`, `ObjectSizeGreaterThanBytes` 또는 `ObjectSizeLessThanBytes` 매니페스트 필터 기준을 사용하는 경우를 제외하고 배치 복제 작업을 복제할 수 있습니다.

매니페스트 기준을 지정하는 구문은 작업을 생성하는 데 사용하는 방법에 따라 달라집니다. 예시는 작업 생성 섹션을 참조하세요.

작업 생성

Amazon S3 콘솔, AWS CLI, AWS SDK 또는 Amazon REST API를 사용하여 S3 배치 작업을 생성할 수 있습니다.

작업 요청 생성에 대한 자세한 내용은 배치 작업 요청 요소 섹션을 참조하세요.

사전 조건

배치 작업 건을 생성하기 전에 관련 권한을 구성했는지 확인합니다. 자세한 내용은 배치 작업에 대한 권한 부여 섹션을 참조하세요.

S3 콘솔 사용

배치 작업 생성

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 작업을 생성하려는 리전을 선택합니다.

   > **Note**
   >
   > 복사 작업의 경우 대상 버킷과 동일한 리전에서 작업을 생성해야 합니다. 기타 작업의 경우 매니페스트에 있는 객체와 동일한 리전에서 작업을 생성해야 합니다.

3. Amazon S3 콘솔의 왼쪽 탐색 창에서 배치 작업을 선택합니다.
4. 작업 생성을 선택합니다.
5. 작업을 생성하려는 AWS 리전을 확인합니다.
6. 매니페스트 형식에서 사용할 매니페스트 객체의 형식을 선택합니다.

   - S3 인벤토리 보고서를 선택하는 경우 Amazon S3가 CSV 형식 인벤토리 보고서의 일부로 생성한 manifest.json 객체의 경로를 입력하고 최신 버전이 아닌 다른 버전을 사용하려는 경우 매니페스트 객체의 버전 ID를 선택적으로 지정합니다.
   - CSV를 선택하는 경우 CSV 형식 매니페스트 객체의 경로를 입력합니다. 매니페스트 객체는 콘솔에 설명된 형식을 따라야 합니다. 선택적으로 최신 버전이 아닌 버전을 사용하려는 경우 매니페스트 객체의 버전 ID를 포함시킬 수 있습니다.

   > **Note**
   >
   > Amazon S3 콘솔은 배치 복제 작업에 대해서만 자동 매니페스트 생성을 지원합니다. 다른 모든 작업 유형의 경우 지정한 필터 기준에 따라 Amazon S3에서 매니페스트를 자동으로 생성하도록 하려면 AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용하여 작업을 구성해야 합니다.

7. 다음을 선택합니다.

8. 작업에서 매니페스트에 나열된 모든 객체에 수행할 작업을 선택합니다. 선택한 작업에 대한 정보를 입력하고 다음을 선택합니다.
9. Configure additional options(추가 옵션 구성)에 대한 정보를 입력하고 다음을 선택합니다.
10. 검토에서 설정을 확인합니다. 설정을 변경하려면 이전을 선택합니다 또는 작업 생성을 선택합니다.

AWS CLI 사용

AWS CLI를 사용하여 Batch Operations 작업을 만들려면 기존 매니페스트를 지정할지 아니면 매니페스트를 자동으로 생성할지에 따라 다음 예제 중 하나를 선택하세요.

Specify manifest

다음 예제는 AWS CLI를 사용하여 기존 매니페스트 파일에 나열된 객체에 대해 수행되는 S3 Batch Operations `S3PutObjectTagging` 작업을 만드는 방법을 보여줍니다.

매니페스트를 지정하여 Batch Operations **`S3PutObjectTagging`** 작업을 만들려면

1. 다음 명령을 사용하여 AWS Identity and Access Management(IAM) 역할을 생성한 다음 IAM 정책을 생성하여 관련 권한을 할당합니다. 다음 역할 및 정책은 Amazon S3에 객체 태그를 추가할 수 있는 권한을 부여합니다. 이 권한은 후속 단계에서 작업을 생성할 때 필요합니다.

   a. 다음 예시 명령을 사용하여 배치 작업에서 사용할 IAM 역할을 생성합니다. 이 예시 명령을 사용하려면 *`S3BatchJobRole`*을 역할에 지정할 이름으로 변경하세요.

```
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. 다음 예시 명령을 사용하여 필요한 권한이 포함된 IAM 정책을 생성하고 이전 단계에서 생성한 IAM 역할에 연결합니다. 필요한 권한에 대한 자세한 내용은 배치 작업에 대한 권한 부여 섹션을 참조하세요.

> **Note**
>
> 디렉터리 버킷에서 작업을 수행하는 배치 작업 건에는 특정 권한이 필요합니다. 자세한 내용은 AWS Identity and Access Management (IAM) for S3 Express One Zone을 참조하세요.

이 예시 명령을 사용하려면 *user input placeholders*를 다음과 같이 바꾸세요.

- *S3BatchJobRole*을 IAM 역할의 이름으로 바꿉니다. 이 이름이 이전에 사용한 이름과 일치하는지 확인하세요.
- *PutObjectTaggingBatchJobPolicy*를 IAM 정책에 부여하려는 이름으로 바꿉니다.
- *amzn-s3-demo-destination-bucket*을 태그를 적용하려는 객체를 포함하는 버킷의 이름으로 바꿉니다.
- *amzn-s3-demo-manifest-bucket*을 매니페스트를 포함하는 버킷의 이름으로 바꿉니다.
- *amzn-s3-demo-completion-report-bucket*을 완료 보고서를 전송하려는 버킷 이름으로 바꿉니다.

```
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:::amzn-s3-demo-destination-bucket/*"
```

```
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-manifest-bucket",
        "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject",
        "s3:GetBucketLocation"
      ],
      "Resource":[
        "arn:aws:s3:::amzn-s3-demo-completion-report-bucket",
        "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*"
      ]
    }
  ]
}'
```

2. 다음 예시 명령을 사용하여 S3PutObjectTagging 작업을 생성합니다.

manifest.csv 파일은 일련의 버킷 및 객체 키 값을 제공합니다. 이 작업은 매니페스트에서 식별된 객체에 지정된 태그를 적용합니다. ETag는 manifest.csv 객체의 ETag이며 Amazon S3 콘솔에서 가져올 수 있습니다. 이 요청은 no-confirmation-required 파라미터를 지정하므로 update-job-status 명령으로 파라미터를 확인하지 않고도 작업을 실행할 수 있습니다. 자세한 정보는 AWS CLI 명령 참조의 create-job 섹션을 참조하세요.

이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요. *IAM-role*을 이전에 생성한 IAM 역할의 ARN으로 바꿉니다.

```
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:::amzn-s3-demo-manifest-
bucket/manifest.csv","ETag":"60e460c9d1046e73f7dde5043ac3ae85"}}' \
    --report '{"Bucket":"arn:aws:s3:::amzn-s3-demo-
completion-report-bucket","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
```

이에 대한 응답으로 Amazon S3은 작업 ID(예: `00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c`)를 반환합니다. 작업을 식별, 모니터링 및 수정하려면 작업 ID가 필요합니다.

Generate manifest

다음 예시는 객체 필터 기준에 따라 매니페스트를 자동으로 생성하는 `S3DeleteObjectTagging` S3 배치 작업 건을 생성하는 방법을 보여줍니다. 이 기준에는 생성 날짜, 키 이름, 크기, 스토리지 클래스 및 태그가 포함됩니다.

매니페스트를 생성하여 Batch Operations **`S3DeleteObjectTagging`** 작업을 만들려면

1. 다음 명령을 사용하여 AWS Identity and Access Management(IAM) 역할을 생성한 다음 IAM 정책을 생성하여 권한을 할당합니다. 다음 역할 및 정책은 Amazon S3에 객체 태그를 삭제할 수 있는 권한을 부여합니다. 이 권한은 후속 단계에서 작업을 생성할 때 필요합니다.

   a. 다음 예시 명령을 사용하여 배치 작업에서 사용할 IAM 역할을 생성합니다. 이 예시 명령을 사용하려면 *`S3BatchJobRole`*을 역할에 지정할 이름으로 변경하세요.

   ```
   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. 다음 예시 명령을 사용하여 필요한 권한이 포함된 IAM 정책을 생성하고 이전 단계에서 생성한 IAM 역할에 연결합니다. 필요한 권한에 대한 자세한 내용은 배치 작업에 대한 권한 부여 섹션을 참조하세요.

> **Note**
>
> 디렉터리 버킷에서 작업을 수행하는 배치 작업 건에는 특정 권한이 필요합니다. 자세한 내용은 AWS Identity and Access Management (IAM) for S3 Express One Zone을 참조하세요.

이 예시 명령을 사용하려면 *user input placeholders*를 다음과 같이 바꾸세요.

- *S3BatchJobRole*을 IAM 역할의 이름으로 바꿉니다. 이 이름이 이전에 사용한 이름과 일치하는지 확인하세요.
- *DeleteObjectTaggingBatchJobPolicy*를 IAM 정책에 부여하려는 이름으로 바꿉니다.
- *amzn-s3-demo-destination-bucket*을 태그를 적용하려는 객체를 포함하는 버킷의 이름으로 바꿉니다.
- *amzn-s3-demo-manifest-bucket*을 매니페스트를 저장하려는 버킷의 이름으로 바꿉니다.
- *amzn-s3-demo-completion-report-bucket*을 완료 보고서를 전송하려는 버킷 이름으로 바꿉니다.

```
aws iam put-role-policy \
  --role-name S3BatchJobRole \
  --policy-name DeleteObjectTaggingBatchJobPolicy \
  --policy-document '{
  "Version":"2012-10-17",
  "Statement":[
    {
```

```
      "Effect":"Allow",
      "Action":[
        "s3:DeleteObjectTagging",
        "s3:DeleteObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutInventoryConfiguration"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-manifest-bucket",
        "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject",
        "s3:ListBucket"
      ],
      "Resource":[
        "arn:aws:s3:::amzn-s3-demo-completion-report-bucket",
        "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
      ]
    }
  ]
}'
```

2. 다음 예시 명령을 사용하여 `S3DeleteObjectTagging` 작업을 생성합니다.

이 예시에서 `--report` 섹션의 값은 생성될 작업 보고서의 버킷, 접두사, 형식 및 범위를 지정합니다. `--manifest-generator` 섹션은 작업이 실행될 객체를 포함하는 소스 버킷에 대한 정보, 작업에 대해 생성될 매니페스트 출력 목록에 대한 정보, 매니페스트에 포함할 객체의 범위를 생성 날짜, 이름 제약 조건, 크기, 스토리지 클래스별로 좁히는 필터 기준이 지정됩니다. 또한 이 명령은 작업의 우선순위, IAM 역할, AWS 리전을 지정합니다.

자세한 정보는 AWS CLI 명령 참조의 create-job 섹션을 참조하세요.

이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요. *IAM-role*을 이전에 생성한 IAM 역할의 ARN으로 바꿉니다.

```
aws s3control create-job \
    --account-id 012345678901 \
    --operation '{
        "S3DeleteObjectTagging": {}
    }' \
    --report '{
        "Bucket":"arn:aws:s3:::amzn-s3-demo-completion-report-bucket",
        "Prefix":"reports",
        "Format":"Report_CSV_20180820",
        "Enabled":true,
        "ReportScope":"AllTasks"
    }' \
    --manifest-generator '{
        "S3JobManifestGenerator": {
          "ExpectedBucketOwner": "012345678901",
          "SourceBucket": "arn:aws:s3:::amzn-s3-demo-source-bucket",
          "EnableManifestOutput": true,
          "ManifestOutputLocation": {
            "ExpectedManifestBucketOwner": "012345678901",
            "Bucket": "arn:aws:s3:::amzn-s3-demo-manifest-bucket",
            "ManifestPrefix": "prefix",
            "ManifestFormat": "S3InventoryReport_CSV_20211130"
          },
          "Filter": {
            "CreatedAfter": "2023-09-01",
            "CreatedBefore": "2023-10-01",
            "KeyNameConstraint": {
              "MatchAnyPrefix": [
                "prefix"
              ],
              "MatchAnySuffix": [
```

```
              "suffix"
            ]
          },
          "ObjectSizeGreaterThanBytes": 100,
          "ObjectSizeLessThanBytes": 200,
          "MatchAnyStorageClass": [
            "STANDARD",
            "STANDARD_IA"
          ]
        }
      }
    }' \
    --priority 2 \
    --role-arn IAM-role \
    --region us-east-1
```

이에 대한 응답으로 Amazon S3은 작업 ID(예: `00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c`)를 반환합니다. 작업을 식별, 모니터링 또는 수정하려면 이 작업 ID가 필요합니다.

AWS SDK for Java 사용

AWS SDK for Java를 사용하여 Batch Operations 작업을 만들려면 기존 매니페스트를 지정할지 아니면 매니페스트를 자동으로 생성할지에 따라 다음 예제 중 하나를 선택하세요.

Specify manifest

다음 예시에서는 기존 매니페스트 파일에 나열된 객체에 수행되는 `S3PutObjectTagging` S3 배치 작업 건을 생성하는 방법을 보여줍니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

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:::amzn-s3-demo-completion-report-
bucket";
        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();
        }
    }
}
```

Generate manifest

다음 예시는 생성 날짜, 키 이름, 크기 등의 객체 필터 기준에 따라 매니페스트를 자동으로 생성하는 s3PutObjectCopy S3 배치 작업 건을 생성하는 방법을 보여줍니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

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.CreateJobRequest;
    import com.amazonaws.services.s3control.model.CreateJobResult;
    import com.amazonaws.services.s3control.model.JobManifestGenerator;
```

```
    import com.amazonaws.services.s3control.model.JobManifestGeneratorFilter;
    import com.amazonaws.services.s3control.model.JobOperation;
    import com.amazonaws.services.s3control.model.JobReport;
    import com.amazonaws.services.s3control.model.KeyNameConstraint;
    import com.amazonaws.services.s3control.model.S3JobManifestGenerator;
    import com.amazonaws.services.s3control.model.S3ManifestOutputLocation;
    import com.amazonaws.services.s3control.model.S3SetObjectTaggingOperation;
    import com.amazonaws.services.s3control.model.S3Tag;

    import java.time.Instant;
    import java.util.Date;
    import java.util.UUID;
    import java.util.ArrayList;

    import static com.amazonaws.regions.Regions.US_WEST_2;

    public class test {
        public static void main(String[] args) {
            String accountId = "012345678901";
            String iamRoleArn = "arn:aws:iam::012345678901:role/ROLE";
            String sourceBucketName = "arn:aws:s3:::amzn-s3-demo-source-bucket";
            String reportBucketName = "arn:aws:s3:::amzn-s3-demo-completion-report-
bucket";
            String manifestOutputBucketName = "arn:aws:s3:::amzn-s3-demo-manifest-
bucket";
            String uuid = UUID.randomUUID().toString();
            long minimumObjectSize = 100L;

            ArrayList<S3Tag> tagSet = new ArrayList<>();
            tagSet.add(new S3Tag().withKey("keyOne").withValue("ValueOne"));

            ArrayList<String> prefixes = new ArrayList<>();
            prefixes.add("s3KeyStartsWith");

            try {
                JobOperation jobOperation = new JobOperation()
                        .withS3PutObjectTagging(new S3SetObjectTaggingOperation()
                                .withTagSet(tagSet)
                        );
                S3ManifestOutputLocation manifestOutputLocation = new
 S3ManifestOutputLocation()
                        .withBucket(manifestOutputBucketName)
                        .withManifestPrefix("manifests")
                        .withExpectedManifestBucketOwner(accountId)
```

```
                        .withManifestFormat("S3InventoryReport_CSV_20211130");

                JobManifestGeneratorFilter jobManifestGeneratorFilter = new
JobManifestGeneratorFilter()
                        .withEligibleForReplication(true)
                        .withKeyNameConstraint(
                                new KeyNameConstraint()
                                        .withMatchAnyPrefix(prefixes))
                        .withCreatedBefore(Date.from(Instant.now()))
                        .withObjectSizeGreaterThanBytes(minimumObjectSize);

                S3JobManifestGenerator s3JobManifestGenerator = new
S3JobManifestGenerator()
                        .withEnableManifestOutput(true)
                        .withManifestOutputLocation(manifestOutputLocation)
                        .withFilter(jobManifestGeneratorFilter)
                        .withSourceBucket(sourceBucketName);

                JobManifestGenerator jobManifestGenerator = new
JobManifestGenerator()
                        .withS3JobManifestGenerator(s3JobManifestGenerator);

                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();

                CreateJobResult createJobResult = s3ControlClient.createJob(new
CreateJobRequest()
                        .withAccountId(accountId)
                        .withOperation(jobOperation)
                        .withManifestGenerator(jobManifestGenerator)
                        .withReport(jobReport)
                        .withPriority(42)
                        .withRoleArn(iamRoleArn)
                        .withClientRequestToken(uuid)
                        .withDescription("job description")
```

```
                        .withConfirmationRequired(true)
                );

                System.out.println("Created job " + createJobResult.getJobId());

            } 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();
            }
        }
    }
```

REST API 사용

REST API를 사용하여 배치 작업을 생성할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 CreateJob를 참조하십시오.

작업 응답

`CreateJob` 요청이 성공하면 Amazon S3가 작업 ID를 반환합니다. 작업 ID는 Amazon S3가 자동으로 생성하는 고유한 식별자이므로 배치 작업 건을 식별하고 그 상태를 모니터링할 수 있습니다.

AWS CLI, AWS SDK 또는 REST API를 통해 작업을 생성할 때 자동으로 작업 처리를 시작하도록 S3 배치 작업을 설정할 수 있습니다. 작업은 우선순위가 더 높은 작업을 기다리지 않고 준비가 완료되는 즉시 실행됩니다.

Amazon S3 콘솔을 통해 작업을 생성할 때 작업 세부 정보를 검토하고 배치 작업이 처리를 시작하기 전에 작업을 실행할 것인지 확인해야 합니다. 작업이 30일 이상 일시 중지된 상태로 있으면 작업이 실패합니다.

S3 배치 작업에서 지원하는 작업

S3 배치 작업을 사용하여 Amazon S3 객체에 대해 대규모 배치 작업을 수행할 수 있습니다. S3 배치 작업은 지정된 Amazon S3 객체 목록에 대해 단일 작업을 수행할 수 있습니다. 단일 작업으로 엑사바이트 규모의 데이터가 포함된 수십억 개의 객체에서 지정된 작업을 수행할 수 있습니다. Amazon S3

는 진행 상황을 추적하고 알림을 보내며 모든 작업에 대한 자세한 완료 보고서를 저장하여 감사 가능한 완전관리형 서버리스 환경을 제공합니다. Amazon S3 콘솔, AWS CLI, AWS SDK 또는 Amazon S3 REST API를 통해 S3 Batch Operations를 사용할 수 있습니다.

S3 배치 작업은 다음 작업(operation)을 지원합니다.

객체 복사

Amazon S3 Batch Operations를 사용하여 Amazon S3 객체에 대해 대규모 배치 작업을 수행할 수 있습니다. Batch Operations 복사 작업은 매니페스트에 지정된 각 객체를 복사합니다. 객체를 동일한 AWS 리전의 버킷 또는 다른 리전의 버킷으로 복사할 수 있습니다. S3 배치 작업은 객체를 복사할 때 Amazon S3를 통해 사용 가능한 대부분의 옵션을 지원합니다. 이러한 옵션에는 객체 메타데이터 설정, 권한 설정, 객체 스토리지 클래스 변경이 포함됩니다.

또한, 복사 작업을 사용하여 암호화되지 않은 기존 객체를 복사하고 암호화된 객체로 동일한 버킷에 다시 쓸 수 있습니다. 자세한 내용은 Amazon S3 배치 작업에서 객체 암호화를 참조하세요.

객체를 복사할 때 객체의 체크섬을 계산하는 데 사용되는 체크섬 알고리즘을 변경할 수 있습니다. 객체에 계산된 추가 체크섬이 없는 경우 Amazon S3에서 사용할 체크섬 알고리즘을 지정하여 체크섬을 추가할 수도 있습니다. 자세한 내용은 Amazon S3에서 객체 무결성 확인 섹션을 참조하세요.

Amazon S3의 객체 복사와 필수 및 선택적 파라미터에 대한 자세한 내용은 이 가이드의 객체 복사, 이동, 이름 변경 섹션과 Amazon Simple Storage Service API 참조의 CopyObject 섹션을 참조하세요.

규제 및 제한

Batch Operations 복사 작업을 사용하는 경우 다음과 같은 규제 및 제한이 적용됩니다.

- 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.
- S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 클래스에서 S3 Standard 스토리지 클래스로 객체를 복사하려는 경우 먼저 객체를 복원해야 합니다. 자세한 내용은 아카이브된 객체 복원 섹션을 참조하세요.
- 객체를 복사하려는 리전인 대상 리전에 Batch Operations 복사 작업을 만들어야 합니다.
- 엔터티 태그(ETag)의 조건부 검사와 고객 제공 암호화 키를 사용한 서버 측 암호화(SSE-C)를 제외한 모든 `CopyObject` 옵션이 지원됩니다.

- 대상 버킷의 버전이 지정되지 않는 경우 키 이름이 같은 모든 객체를 덮어쓰게 됩니다.
- 객체는 반드시 매니페스트에 나타나는 것과 동일한 순서로 복사되는 것은 아닙니다. 버전이 지정된 버킷의 경우 최신 또는 비최신 버전 순서를 유지하는 것이 중요하다면 모든 비최신 버전을 먼저 복사합니다. 그런 다음 첫 번째 작업이 완료된 후 후속 작업에서 현재 버전을 복사합니다.
- Reduced Redundancy Storage(RRS) 클래스로 객체를 복사하는 것은 지원되지 않습니다.
- 단일 Batch Operations 복사 작업은 최대 200억 개의 객체가 포함된 매니페스트를 지원할 수 있습니다.

S3 배치 작업을 사용하여 객체 복사

Amazon S3 Batch Operations를 사용하여 Amazon S3 객체에 대해 대규모 배치 작업을 수행할 수 있습니다. S3 Batch Operations를 사용하여 복사(`CopyObject`) 작업을 만들어 동일한 계정 내에서 또는 다른 대상 계정으로 객체를 복사할 수 있습니다.

다음 예제는 다른 계정에 있는 매니페스트를 저장하고 사용하는 방법을 보여줍니다. 첫 번째 예제는 Amazon S3 Inventory를 사용하여 작업을 만드는 동안 사용할 수 있도록 인벤토리 보고서를 대상 계정으로 전달하는 방법을 보여줍니다. 두 번째 예제에서는 소스 또는 대상 계정에서 쉼표로 구분된 값(CSV) 매니페스트를 사용하는 방법을 보여줍니다. 세 번째 예제에서는 AWS Key Management Service(AWS KMS) 키를 사용한 서버 측 암호화(SSE-KMS)를 통해 암호화된 기존 객체에 대해 S3 버킷 키를 활성화하도록 복사 작업을 사용하는 방법을 보여줍니다.

복사 작업 예

- 인벤토리 보고서를 사용하여 AWS 계정 간에 객체 복사
- CSV 매니페스트를 사용하여 AWS 계정 간에 객체 복사
- Batch Operations를 사용하여 SSE-KMS용 S3 버킷 키 활성화

인벤토리 보고서를 사용하여 AWS 계정 간에 객체 복사

Amazon S3 Batch Operations를 사용하여 Amazon S3 객체에 대해 대규모 배치 작업을 수행할 수 있습니다. S3 Batch Operations를 사용하여 복사(`CopyObject`) 작업을 만들어 동일한 계정 내에서 또는 다른 대상 계정으로 객체를 복사할 수 있습니다.

Amazon S3 Inventory를 사용하여 인벤토리 보고서를 만들고 이 보고서를 사용하여 S3 Batch Operations로 복사할 객체 목록(매니페스트)을 만들 수 있습니다. 소스 또는 대상 계정에서 CSV 매니페스트를 사용하는 방법에 대한 자세한 내용은 the section called “CSV 매니페스트를 사용하여 AWS 계정 간에 객체 복사” 섹션을 참조하세요.

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 계정에 전송하도록 인벤토리 보고서를 구성할 수 있습니다. 그러면 대상 계정에서 작업이 생성될 때 S3 Batch Operations에서 인벤토리 보고서를 읽을 수 있습니다.

Amazon S3 인벤토리 소스 및 대상 버킷에 대한 자세한 내용은 원본 및 대상 버킷 섹션을 참조하세요.

인벤토리를 설정하는 가장 쉬운 방법은 Amazon S3 콘솔을 사용하는 것이지만, Amazon S3 REST API, AWS Command Line Interface(AWS CLI) 또는 AWS SDK를 사용할 수도 있습니다.

다음 콘솔 절차에는 S3 배치 작업에 대한 권한을 설정하기 위한 상위 단계가 포함되어 있습니다. 이 절차에서는 원본 계정에서 대상 계정으로 객체를 복사하고, 대상 계정에 인벤토리 보고서를 저장합니다.

다른 계정이 소유한 원본 버킷과 대상 버킷에 Amazon S3 인벤토리를 설정하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 인벤토리 보고서를 저장할 대상 매니페스트 버킷을 결정하거나 만듭니다. 이 절차에서 대상 계정은 대상 매니페스트 버킷과 객체가 복사되는 버킷을 모두 소유한 계정입니다.
4. 소스 버킷에 대한 인벤토리 보고서를 구성합니다. 콘솔을 사용하여 인벤토리 목록을 구성하는 방법 또는 인벤토리 목록 파일을 암호화하는 방법에 대한 자세한 내용은 Amazon S3 인벤토리 구성 페이지를 참조하세요.

   인벤토리 보고서를 구성할 때 목록을 저장할 대상 버킷을 지정합니다. 원본 버킷에 대한 인벤토리 보고서가 대상 버킷에 게시됩니다. 이 절차에서 원본 계정은 원본 버킷을 소유한 계정입니다.

   출력 형식으로 CSV를 선택하세요.

   대상 버킷에 대한 정보를 입력할 때 다른 계정의 버킷을 선택하세요. 그런 다음 대상 매니페스트 버킷의 이름을 입력합니다. 선택적으로 대상 계정의 계정 ID를 입력할 수 있습니다.

   인벤토리 구성을 저장하면 콘솔에 다음과 유사한 메시지가 표시됩니다.

   Amazon S3은 대상 버킷에 버킷 정책을 생성할 수 없습니다. Amazon S3이 대상 버킷에 데이터를 저장할 수 있도록 다음 버킷 정책을 추가하려면 대상 버킷 소유자에게 요청하세요.

그러면 콘솔에 대상 버킷에 사용할 수 있는 버킷 정책이 표시됩니다.

5. 콘솔에 표시되는 대상 버킷 정책을 복사하세요.
6. 대상 계정에서 인벤토리 보고서가 저장된 대상 매니페스트 버킷에 복사한 버킷 정책을 추가합니다.
7. S3 배치 작업 신뢰 정책을 기반으로 하는 대상 계정에 역할을 생성합니다. 신뢰 정책에 대한 자세한 내용은 신뢰 정책 섹션을 참조하세요.

   역할 만들기에 대한 자세한 내용은 IAM 사용 설명서의 역할을 만들어 AWS 서비스에 권한 위임을 참조하세요.

   역할의 이름을 입력하세요(다음 예제 역할은 이름 *BatchOperationsDestinationRoleCOPY* 사용). S3 서비스를 선택한 다음, 신뢰 정책을 역할에 적용하는 S3 Batch Operations 사용 사례를 선택합니다.

   그리고 정책 생성을 선택하여 다음 정책을 역할에 연결하세요. 이 정책을 사용하려면 *user input placeholders*를 실제 정보로 대체합니다.

```
{
  "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:::amzn-s3-demo-destination-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-source-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
```

```
      ]
    }
  ]
}
```

이 역할은 정책을 사용하여 대상 버킷의 매니페스트를 읽을 수 있는 `batchoperations.s3.amazonaws.com` 권한을 부여합니다. 소스 객체 버킷의 GET 객체, 액세스 제어 목록(ACL), 태그 및 버전에 대한 권한도 부여합니다. 또한, 대상 객체 버킷의 PUT 객체, ACL, 태그 및 버전에 대한 권한을 부여합니다.

8. 소스 계정에서, 이전 단계에서 만든 역할에 소스 버킷의 GET 객체, ACL, 태그 및 버전에 대한 권한을 부여하는 소스 버킷에 대한 버킷 정책을 만듭니다. 이 단계에서는 신뢰할 수 있는 역할을 통해 S3 배치 작업이 원본 버킷에서 객체를 가져올 수 있습니다.

   다음은 원본 계정에 대한 버킷 정책의 예입니다. 이 정책을 사용하려면 *user input placeholders*를 실제 정보로 대체합니다.

```
{
  "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:::amzn-s3-demo-source-bucket/*"
    }
  ]
}
```

9. 인벤토리 보고서를 사용할 수 있게 되면 대상 계정에 S3 Batch Operations 복사(`CopyObject`) 작업을 생성하고 대상 매니페스트 버킷에서 인벤토리 보고서를 선택합니다. 대상 계정에서 만든 IAM 역할에 대한 ARN이 필요합니다.

   작업 생성에 대한 일반적인 정보는 S3 배치 작업 건 생성 섹션을 참조하세요.

   콘솔을 사용하여 작업을 만드는 방법에 대한 자세한 내용은 S3 배치 작업 건 생성 섹션을 참조하세요.

CSV 매니페스트를 사용하여 AWS 계정 간에 객체 복사

Amazon S3 Batch Operations를 사용하여 Amazon S3 객체에 대해 대규모 배치 작업을 수행할 수 있습니다. S3 Batch Operations를 사용하여 복사(`CopyObject`) 작업을 만들어 동일한 계정 내에서 또는 다른 대상 계정으로 객체를 복사할 수 있습니다.

소스 계정에 저장된 CSV 매니페스트를 사용하여 S3 배치 작업에서 AWS 계정 간에 객체를 복사할 수 있습니다. S3 인벤토리 보고서를 매니페스트로 사용하려면 the section called “인벤토리 보고서를 사용하여 AWS 계정 간에 객체 복사” 섹션을 참조하세요.

매니페스트 파일의 CSV 형식 예는 the section called “매니페스트 파일 생성” 섹션을 참조하세요.

다음 절차에서는 S3 배치 작업을 사용하여 소스 계정의 객체를 소스 계정에 저장된 CSV 매니페스트 파일이 있는 대상 계정으로 복사할 때 권한을 설정하는 방법을 보여줍니다.

CSV 매니페스트를 사용하여 AWS 계정 간에 객체를 복사하려면

1. S3 Batch Operations 신뢰 정책을 기반으로 하는 대상 계정에서 AWS Identity and Access Management(IAM) 역할을 만듭니다. 이 절차에서 대상 계정은 객체가 복사되는 계정입니다.

   신뢰 정책에 대한 자세한 내용은 신뢰 정책 섹션을 참조하세요.

   역할 만들기에 대한 자세한 내용은 IAM 사용 설명서의 역할을 만들어 AWS 서비스에 권한 위임을 참조하세요.

   콘솔을 사용하여 역할을 만드는 경우 역할의 이름을 입력합니다(다음 예제 역할은 이름 *`BatchOperationsDestinationRoleCOPY`* 사용). S3 서비스를 선택한 다음, 신뢰 정책을 역할에 적용하는 S3 Batch Operations 사용 사례를 선택합니다.

   그리고 정책 생성을 선택하여 다음 정책을 역할에 연결하세요. 이 정책을 사용하려면 *`user input placeholders`*를 실제 정보로 대체합니다.

```
{
  "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:::amzn-s3-demo-destination-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-source-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
      ]
    }
  ]
}
```

이 역할은 정책을 사용하여 원본 매니페스트 버킷의 매니페스트를 읽을 수 있는 batchoperations.s3.amazonaws.com 권한을 부여합니다. 소스 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.

2. 소스 계정에서 매니페스트가 포함된 버킷에 대한 버킷 정책을 생성하여 이전 단계에서 생성한 역할에 소스 매니페스트 버킷의 GET 객체 및 버전에 대한 권한을 부여합니다.

   이 단계에서는 S3 배치 작업이 신뢰할 수 있는 역할을 사용하여 매니페스트를 읽을 수 있습니다. 매니페스트가 포함된 버킷에 버킷 정책을 적용합니다.

   다음은 원본 매니페스트 버킷에 적용할 버킷 정책의 예입니다. 이 정책을 사용하려면 *user input placeholders*를 실제 정보로 대체합니다.

```
{
  "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:::amzn-s3-demo-manifest-bucket/*"
    }
  ]
}
```

또한 이 정책은 대상 계정에서 작업을 생성하는 콘솔 사용자가 동일한 버킷 정책을 통해 원본 매니페스트 버킷의 동일한 권한을 사용할 수 있는 권한을 부여합니다.

3. 소스 계정에서, 생성한 역할에 소스 객체 버킷의 GET 객체, ACL, 태그 및 버전에 대한 권한을 부여하는 버킷 정책을 소스 버킷에 대해 생성합니다. 그런 다음 S3 배치 작업은 신뢰할 수 있는 역할을 통해 원본 버킷에서 객체를 가져올 수 있습니다.

   다음은 원본 객체가 포함된 버킷에 대한 버킷 정책의 예입니다. 이 정책을 사용하려면 *user input placeholders*를 실제 정보로 대체합니다.

```
{
  "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:::amzn-s3-demo-source-bucket/*"
    }
  ]
}
```

4. 대상 계정에서 S3 배치 작업을 생성합니다. 대상 계정에서 생성한 역할에 대한 Amazon 리소스 이름(ARN)이 필요합니다. 작업 생성에 대한 자세한 내용은 S3 배치 작업 건 생성 단원을 참조하세요.

Batch Operations를 사용하여 SSE-KMS용 S3 버킷 키 활성화

S3 버킷 키는 Amazon S3에서 AWS KMS로 가는 요청 트래픽을 줄여 AWS Key Management Service(AWS KMS)를 사용한 서버 측 암호화(SSE-KMS) 비용을 절감합니다. 자세한 내용은 Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감 및 새 객체에 SSE-KMS와 함께 S3 버킷 키를 사용하도록 버킷 구성(을)를 참조하세요. REST API, AWS SDK 또는 AWS CLI를 사용하여 `CopyObject` 작업을 수행할 때 `x-amz-server-side-encryption-bucket-key-enabled` 요청 헤더에 `true` 또는 `false` 값을 추가하여 객체 수준에서 S3 버킷 키를 활성화하거나 비활성화할 수 있습니다.

`CopyObject` 작업을 사용하여 객체에 대한 S3 버킷 키를 구성하면 Amazon S3는 해당 객체에 대한 설정만 업데이트합니다. 대상 버킷에 대한 S3 버킷 키 설정은 변경되지 않습니다. S3 버킷 키가 활성화된 버킷에 AWS KMS로 암호화된 객체에 대한 `CopyObject` 요청을 제출하면 요청 헤더의 키를 비활성화하지 않는 이상 객체 수준 작업에서 자동으로 S3 버킷 키가 사용됩니다. 객체에 S3 버킷 키를 지정하지 않으면 Amazon S3가 대상 버킷에 대한 S3 버킷 키 설정을 객체에 적용합니다.

기존 Amazon S3 객체를 암호화하기 위해 S3 Batch Operations를 사용할 수 있습니다. 배치 작업 복사 작업을 사용하여 암호화되지 않은 기존 객체를 복사하고 암호화된 새로운 객체를 동일한 버킷에 작성할 수 있습니다. 자세한 내용은 AWS 스토리지 블로그의 Amazon S3 Batch Operations로 객체 암호화를 참조하세요.

다음 예제에서는 Batch Operations 복사 작업을 사용하여 기존 객체에서 S3 버킷 키를 활성화합니다. 자세한 내용은 the section called “객체에 대한 S3 버킷 키 구성” 섹션을 참조하세요.

주제

- S3 Batch Operations를 사용하여 S3 버킷 키가 활성화된 객체를 암호화할 때 고려해야 할 사항
- 사전 조건
- 1단계: Amazon S3 인벤토리를 사용하여 객체 목록 가져오기
- 2단계: S3 Select를 사용하여 객체 목록 필터링
- 3단계: S3 배치 작업 설정 및 실행

S3 Batch Operations를 사용하여 S3 버킷 키가 활성화된 객체를 암호화할 때 고려해야 할 사항

S3 Batch Operations를 사용하여 S3 버킷 키가 활성화된 객체를 암호화할 때는 다음 문제를 고려합니다.

- 데이터 전송, 요청 및 기타 요금을 포함하여 S3 배치 작업에서 사용자를 대신해 수행하는 작업과 연관된 요금 외에도 S3 배치 작업, 객체 및 요청에 대한 요금이 청구됩니다. 자세한 내용은 Amazon S3 요금을 참조하세요.
- 버전 관리 버킷을 사용하는 경우 수행된 각 S3 배치 작업은 암호화된 새 객체 버전을 생성합니다. 또한, S3 버킷 키가 구성되지 않은 이전 버전도 유지 관리합니다. 이전 버전을 삭제하려면 수명 주기 구성의 요소에서 설명한 대로, 최신 버전이 아닌 버전에 대해 S3 수명 주기 만료 정책을 설정합니다.
- 복사 작업은 새 생성 날짜에 새 객체를 생성하므로 아카이브와 같은 수명 주기 작업에 영향을 줄 수 있습니다. 버킷의 모든 객체를 복사하는 경우, 모든 새 사본의 생성 날짜가 동일하거나 비슷합니다. 이러한 객체를 추가로 식별하고 다양한 데이터 하위 집합에 대해 서로 다른 수명 주기 규칙을 생성하려면 객체 태그 사용을 고려합니다.

사전 조건

S3 버킷 키를 사용하도록 객체를 구성하기 전에 S3 버킷 키를 사용 설정하기 전에 유의할 변경 사항 섹션을 검토하세요.

이 예제를 사용하려면 AWS 계정 및 작업 파일과 암호화된 결과를 보관할 수 있는 S3 버킷이 하나 이상 있어야 합니다. 또한, 다음 항목을 포함하여 기존 S3 배치 작업 설명서의 대부분을 유용하게 사용할 수 있습니다.

- S3 배치 작업 기본 사항
- S3 배치 작업 건 생성
- S3 배치 작업에서 지원하는 작업
- S3 배치 작업 건 관리

1단계: Amazon S3 인벤토리를 사용하여 객체 목록 가져오기

시작하려면 암호화할 객체를 포함하는 S3 버킷을 식별하고 해당 콘텐츠 목록을 가져옵니다. Amazon S3 인벤토리 보고서는 이 작업을 수행하는 가장 편리하면서도 경제적인 방법입니다. 이 보고서는 연결된 메타데이터와 함께 버킷에 있는 객체 목록을 제공합니다. 이 단계에서 소스 버킷은 인벤토리가 있는 버킷이고 대상 버킷은 인벤토리 보고서 파일을 저장하는 버킷입니다. Amazon S3 인벤토리 소스 및 대상 버킷에 대한 자세한 내용은 S3 Inventory를 사용한 데이터 카탈로그화 및 분석 섹션을 참조하세요.

인벤토리를 설정하는 가장 쉬운 방법은 AWS Management Console을 사용하는 것입니다. 하지만 REST API, AWS Command Line Interface(AWS CLI) 또는 AWS SDK를 사용할 수도 있습니다. 이 단계를 수행하기 전에 콘솔에 로그인하고 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다. . 권한 거부 오류가 발생하면 대상 버킷에 버킷 정책을 추가합니다. 자세한 내용은 S3 인벤토리 및 S3 분석 권한 부여 섹션을 참조하세요.

S3 인벤토리를 사용하여 객체 목록 가져오기

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷을 선택하고 암호화할 객체를 포함하는 버킷을 선택합니다.
3. [관리(Management)] 탭에서 [인벤토리 구성(Inventory configurations)] 섹션으로 이동하고 [인벤토리 구성 작성(Create inventory configuration)]을 선택합니다.
4. 새 인벤토리에 이름을 지정하고 대상 S3 버킷의 이름을 입력한 다음, 선택적으로 Amazon S3 대상 접두사를 생성하여 해당 버킷에서 객체를 할당합니다.
5. [출력 형식(Output format)]으로 [CSV]를 선택합니다.
6. (선택 사항) 추가 필드 - 선택 사항 섹션에서 암호화 및 원하는 기타 보고서 필드를 선택합니다. 첫 번째 보고서가 버킷에 더 빨리 제공되도록 보고서 제공 빈도를 [일별(Daily)]로 설정합니다.
7. [생성(Create)]을 선택하여 구성을 저장합니다.

Amazon S3에서 첫 번째 보고서를 제공하는 데 최대 48시간이 걸릴 수 있으므로 첫 번째 보고서가 도착하면 다시 확인합니다. 첫 번째 보고서를 받은 후 다음 단계로 이동하여 S3 Inventory 보고서의 콘텐츠를 필터링합니다. 이 버킷에 대한 인벤토리 보고서를 더 이상 받지 않으려면 S3 인벤토리 구성을 삭제합니다. 그렇지 않으면 Amazon S3는 매일 또는 매주 일정에 따라 보고서를 계속 제공합니다.

인벤토리 목록은 모든 객체의 단일 시점 보기가 아닙니다. 인벤토리 목록은 최종 일관성을 지닌 버킷 항목의 롤링 스냅샷입니다(예: 목록에 최근에 추가되거나 삭제된 객체가 포함되지 않을 수 있음). S3 인벤토리와 S3 배치 작업을 결합하면 정적 객체 또는 2일 이상 전에 생성한 객체 세트로 작업할 때 가장 효과적입니다. 최신 데이터로 작업하려면 ListObjectsV2(GET 버킷) API 작업을 사용하여 수동으로

객체 목록을 작성합니다. 필요한 경우 다음 며칠 동안 또는 인벤토리 보고서에 모든 객체의 원하는 상태가 표시될 때까지 이 프로세스를 반복합니다.

2단계: S3 Select를 사용하여 객체 목록 필터링

S3 Inventory 보고서를 받은 후 활성화된 S3 버킷 키로 암호화되지 않은 객체만 나열하도록 보고서의 콘텐츠를 필터링할 수 있습니다. 활성화된 S3 버킷 키로 모든 버킷의 객체를 암호화하려는 경우 이 단계는 무시할 수 있습니다. 그러나 이 단계에서 S3 Inventory 보고서를 필터링하면 이전에 활성화된 S3 버킷 키로 암호화한 객체를 다시 암호화하는 데 드는 시간과 비용을 절약할 수 있습니다.

다음 단계에서는 Amazon S3 Select를 사용하여 필터링하는 방법을 보여주지만, Amazon Athena를 사용할 수도 있습니다. 사용할 도구를 결정하려면 S3 인벤토리 보고서의`manifest.json` 파일을 검토합니다. 이 파일은 해당 보고서에 연결된 데이터 파일의 수를 나열합니다. 많은 경우 Amazon Athena를 사용합니다. 여러 S3 객체에서 실행되기 때문입니다. 반면 S3 Select는 한 번에 하나의 객체에서 작동합니다. Amazon S3와 Athena를 함께 사용하는 방법에 대한 자세한 내용은 AWS 스토리지 블로그 게시물 Amazon S3 Batch Operations로 객체 암호화에서 Amazon Athena로 Amazon S3 인벤토리 쿼리 및 'Athena 사용'을 참조하세요.

S3 Select를 사용하여 S3 Inventory 보고서를 필터링하려면

1. 인벤토리 보고서에서 `manifest.json` 파일을 열고 JSON의 `fileSchema` 섹션을 참조하세요. 이 데이터에서 실행하는 쿼리를 알립니다.

   다음 JSON 버전 관리가 사용되는 버킷에서 CSV 형식 인벤토리의 `manifest.json` 파일에 대한 예제입니다. 인벤토리 보고서를 구성한 방법에 따라 매니페스트가 다를 수 있습니다.

```
  {
    "sourceBucket": "batchoperationsdemo",
    "destinationBucket": "arn:aws:s3:::amzn-s3-demo-destination-bucket",
    "version": "2021-05-22",
    "creationTimestamp": "1558656000000",
    "fileFormat": "CSV",
    "fileSchema": "Bucket, Key, VersionId, IsLatest, IsDeleteMarker,
 BucketKeyStatus",
    "files": [
      {
        "key": "demoinv/batchoperationsdemo/DemoInventory/data/009a40e4-
f053-4c16-8c75-6100f8892202.csv.gz",
        "size": 72691,
        "MD5checksum": "c24c831717a099f0ebe4a9d1c5d3935c"
      }
```

```
    ]
  }
```

버킷에서 버전 관리가 활성화되지 않았거나 최신 버전에 대한 보고서를 실행하도록 선택한 경우 `fileSchema`는 `Bucket`, `Key` 및 `BucketKeyStatus`입니다.

버전 관리가 활성화된 경우 인벤토리 보고서를 설정하는 방법에 따라 `fileSchema`에는 `Bucket`, `Key`, `VersionId`, `IsLatest`, `IsDeleteMarker`, `BucketKeyStatus`가 포함될 수 있습니다. 따라서 쿼리를 실행할 때는 1, 2, 3, 6 열에 주의하세요.

S3 배치 작업에는 `BucketKeyStatus`인 검색 기준 필드 외에, 작업을 수행하기 위한 입력으로 버킷, 키 및 버전 ID가 필요합니다. `VersionID` 필드가 필요하지는 않지만, 버전이 지정된 버킷에서 작업할 때 `VersionID` 필드를 지정하는 데 도움이 됩니다. 자세한 내용은 버전 관리가 사용 설정된 버킷의 객체 작업 섹션을 참조하세요.

2. 인벤토리 보고서의 데이터 파일을 찾습니다. `manifest.json` 객체는 files 아래 데이터 파일을 나열합니다..
3. S3 콘솔에서 데이터 파일을 찾아 선택한 후 [작업(Actions)]을 선택한 다음 [S3 Select에서 쿼리(Query with S3 Select)]를 선택합니다.
4. 사전 설정된 [CSV], [쉼표(Comma)] 및 [GZIP] 필드를 선택한 상태로 두고 [다음(Next)]을 선택합니다.
5. 계속하기 전에 인벤토리 보고서의 형식을 검토하려면 [파일 미리 보기 표시(Show file preview)]를 선택합니다.
6. SQL 표현식 필드에 참조할 열을 입력하고 SQL 실행(Run SQL)을 선택합니다. 다음 표현식은 S3 버킷 키가 구성되지 않은 모든 객체에 대해 1~3번째 열을 반환합니다.

```
select s._1, s._2, s._3 from s3object s where s._6 = 'DISABLED'
```

다음은 예 결과입니다.

```
      batchoperationsdemo,0100059%7Ethumb.jpg,lsrtIxksLu0R0ZkYPL.LhgD5caTYn6vu
      batchoperationsdemo,0100074%7Ethumb.jpg,sd2M60g6Fdazoi6D5kNARIE7KzUibmHR
      batchoperationsdemo,0100075%7Ethumb.jpg,TLYESLnl1mXD5c4BwiOIinqFrktddkoL
      batchoperationsdemo,0200147%7Ethumb.jpg,amufzfMi_fEw0Rs99rxR_HrDFlE.l3Y0
      batchoperationsdemo,0301420%7Ethumb.jpg,9qGU2SEscL.C.c_sK89trmXYIwooABSh
      batchoperationsdemo,0401524%7Ethumb.jpg,ORnEWNuB1QhHrrYAGFsZhbyvEYJ3DUor
      batchoperationsdemo,200907200065HQ
%7Ethumb.jpg,d8LgvIVjbDR5mUVwW6pu9ahTfReyn5V4
      batchoperationsdemo,200907200076HQ
%7Ethumb.jpg,XUT25d7.gK40u_GmnupdaZg3BVx2jN40
```

```
      batchoperationsdemo,201103190002HQ
%7Ethumb.jpg,z.2sVRh0myqVi0BuIrngWlsRPQdb7qOS
```

7. 결과를 다운로드하여 CSV 형식으로 저장한 다음 S3 배치 작업의 객체 목록으로 Amazon S3에 업로드합니다.
8. 매니페스트 파일이 여러 개인 경우 [S3 Select에서 쿼리(Query with S3 Select)도 실행합니다. 결과의 크기에 따라 목록을 결합하여 단일 S3 배치 작업을 실행하거나 각 목록을 별도의 작업으로 실행할 수 있습니다. 실행할 작업 수를 결정하려면 각 S3 Batch Operations 작업의 실행 요금을 고려합니다.

3단계: S3 배치 작업 설정 및 실행

S3 객체의 CSV 목록을 필터링했으므로, S3 Batch Operations 작업을 시작하여 활성화된 S3 버킷 키로 객체를 암호화할 수 있습니다.

작업은 제공된 객체의 목록(매니페스트), 수행된 작업 및 지정된 파라미터를 총체적으로 나타냅니다. 활성화된 S3 버킷 키로 객체 집합을 암호화하는 가장 쉬운 방법은 복사 작업을 사용하고 매니페스트에 나열된 객체와 동일한 대상 접두사를 지정하는 것입니다. 버전이 지정되지 않은 버킷에서 이 작업은 기존 객체를 덮어씁니다. 버전 관리가 설정된 버킷에서 이 작업은 암호화된 최신 버전의 객체를 생성합니다.

객체를 복사하는 과정에서 Amazon S3가 SSE-KMS 암호화로 객체를 암호화하도록 지정합니다. 이 작업은 객체를 복사하므로, 처음에 Amazon S3에 추가한 시점과 관계없이 모든 객체가 완료 시 업데이트된 생성 날짜를 표시합니다. 또한, 객체 태그와 스토리지 클래스를 포함하여 S3 배치 작업의 일부로 객체 집합에 대한 다른 속성을 지정합니다.

하위 단계

- IAM 정책 설정
- 배치 작업 IAM 역할 설정
- 기존 버킷에 대해 S3 버킷 키 활성화
- 배치 작업 생성
- 배치 작업 실행

IAM 정책 설정

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 정책을 선택한 다음 정책 만들기를 선택합니다.

3. [JSON] 탭을 선택합니다. [정책 편집(Edit policy)]을 선택하고 다음 코드 블록에 표시되는 IAM 정책 예제를 추가합니다.

   정책 예제를 IAM 콘솔에 복사한 후 다음을 바꿉니다.

   a. *amzn-s3-demo-source-bucket*을 객체를 복사할 소스 버킷의 이름으로 바꿉니다.

   b. *amzn-s3-demo-destination-bucket*을 객체를 복사할 대상 버킷의 이름으로 바꿉니다.

   c. *amzn-s3-demo-manifest-bucket*/*manifest-key*를 매니페스트 객체의 이름으로 바꿉니다.

   d. *amzn-s3-demo-completion-report-bucket*을 완료 보고서를 저장하려는 버킷의 이름으로 바꿉니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CopyObjectsToEncrypt",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectTagging",
        "s3:PutObjectAcl",
        "s3:PutObjectVersionTagging",
        "s3:PutObjectVersionAcl",
        "s3:GetObject",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersion",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-source-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
      ]
    },
    {
      "Sid": "ReadManifest",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
```

```
        "s3:GetObjectVersion"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-manifest-bucket/manifest-key"
    },
    {
      "Sid": "WriteReport",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*"
    }
  ]
}
```

4. 다음: 태그를 선택합니다.
5. 선택적으로 원하는 태그를 추가하고 [다음: 검토(Next: Review)]를 선택합니다.
6. 정책 이름과 설명(선택 사항)을 추가하고 [정책 생성(Create policy)]을 선택합니다.
7. [정책 검토(Review policy)] 및 [변경 사항 저장(Save changes)]을 선택합니다.
8. 이제 S3 배치 작업 정책이 완료되면 콘솔이 IAM 정책 페이지로 이동합니다. 정책 이름을 필터링하고 정책 이름 왼쪽에 있는 버튼을 선택한 다음 [정책 작업(Policy actions)]을 선택하고 [연결(Attach)]을 선택합니다.

   새로 생성된 정책을 IAM 역할에 연결하려면 계정에서 적절한 사용자, 그룹 또는 역할을 선택하고 [정책 연결(Attach policy)]을 선택합니다. 그러면 IAM 콘솔로 돌아갑니다.

배치 작업 IAM 역할 설정

1. IAM 콘솔의 탐색 창에서 역할을 선택하고 역할 생성을 선택합니다.
2. AWS 서비스, S3 및 S3 배치 작업을 선택합니다. 그런 다음 [Next: Permissions]를 선택합니다.
3. 먼저 방금 생성한 IAM 정책 이름을 입력합니다. 정책 이름이 나타나면 해당 확인란을 선택하고 [다음: 태그(Next: Tags)]를 선택합니다.
4. (선택 사항) 이 연습에서는 태그를 추가하거나 키 및 값 필드를 비워 둡니다. 다음: 검토를 선택합니다.
5. 역할 이름을 입력하고 기본 설명을 그대로 사용하거나 직접 추가합니다. 역할 생성을 선택합니다.
6. 작업을 생성하는 사용자에게 다음 예제의 권한이 있는지 확인합니다.

AWS 계정 ID 및 *IAM-role-name*의 *account-id*를 나중에 배치 작업 생성 단계에서 생성할 IAM 역할에 적용하려는 이름으로 바꿉니다. 자세한 내용은 배치 작업에 대한 권한 부여 섹션을 참조하세요.

```
{
"Sid": "AddIamPermissions",
"Effect": "Allow",
"Action": [
"iam:GetRole",
"iam:PassRole"
],
"Resource": "arn:aws:iam::account-id:role/IAM-role-name"
}
```

기존 버킷에 대해 S3 버킷 키 활성화

1. https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 버킷 목록에서 S3 버킷 키를 활성화할 버킷을 선택합니다.
3. [속성(Properties)]을 선택합니다.
4. 기본 암호화에서 편집을 선택합니다.
5. 암호화 유형에서 Amazon S3 관리형 키(SSE-S3) 또는 AWS Key Management Service 키(SSE-KMS)를 선택할 수 있습니다.
6. AWS Key Management Service 키(SSE-KMS)를 선택한 경우, AWS KMS key에서 다음 옵션 중 하나를 통해 AWS KMS 키를 지정할 수 있습니다.
   - 사용 가능한 KMS 키 목록에서 선택하려면 AWS KMS 키에서 선택을 선택합니다. 사용 가능한 키 목록에서 버킷과 동일한 리전의 대칭 암호화 KMS 키를 선택합니다. AWS 관리형 키(aws/s3)와 고객 관리형 키가 모두 목록에 표시됩니다.
   - KMS 키 ARN을 입력하려면 AWS KMS 키 ARN 입력을 선택한 다음 나타나는 필드에 KMS 키 ARN을 입력합니다.
   - AWS KMS 콘솔에서 고객 관리형 키를 생성하려면 KMS 키 생성을 선택합니다.
7. [버킷 키(Bucket Key)] 아래에서 [사용(Enable)]을 선택하고 [변경 사항 저장(Save changes)]을 선택합니다.

버킷 수준에서 S3 버킷 키가 활성화되었으므로, 이 버킷으로 업로드, 수정 또는 복사된 객체는 기본적으로 이 암호화 구성을 상속합니다. 여기에는 Amazon S3 배치 작업을 사용하여 복사된 객체가 포함됩니다.

배치 작업 생성

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 탐색 창에서 배치 작업(Batch Operations)을 선택하고 작업 생성(Create Job)을 선택합니다.
3. 객체를 저장할 [리전(Region)]을 선택하고 매니페스트 유형으로 [CSV]를 선택합니다.
4. 경로를 입력하거나 이전에 S3 Select(또는 Athena) 결과에서 생성한 CSV 매니페스트 파일로 이동합니다. 매니페스트에 버전 ID가 있으면 해당 상자를 선택합니다. [Next]를 선택합니다.
5. [복사(Copy)] 작업을 선택하고 복사 대상 버킷을 선택합니다. 서버 측 암호화를 사용하지 않도록 설정할 수 있습니다. 버킷 대상에서 S3 버킷 키가 사용되는 한, 복사 작업은 대상 버킷에 S3 버킷 키를 적용합니다.
6. (선택 사항) 스토리지 클래스와 다른 파라미터를 원하는 대로 선택합니다. 이 단계에서 지정하는 파라미터는 매니페스트에 나열된 객체에서 수행된 모든 작업에 적용됩니다. 다음을 선택합니다.
7. 서버 측 암호화를 구성하려면 다음 단계를 따르세요.

   a. 서버 측 암호화에서 다음 중 하나를 선택합니다.

      - Amazon S3에 객체를 저장할 때 객체의 기본 서버 측 암호화에 대한 버킷 설정을 유지하려면 암호화 키를 지정하지 마십시오를 선택합니다. 버킷 대상에서 S3 버킷 키가 사용되는 한 복사 작업은 대상 버킷에 S3 버킷 키를 적용합니다.

        > **Note**
        >
        > 지정된 대상의 버킷 정책에서 Amazon S3에 저장하기 전에 객체를 암호화해야 하는 경우 암호화 키를 지정해야 합니다. 지정하지 않으면 대상에 대한 객체 복사가 실패합니다.

      - Amazon S3에 저장하기 전에 객체를 암호화하려면 암호화 키 지정을 선택합니다.

   b. 암호화 설정에서 암호화 키 지정을 선택하는 경우, 기본 암호화에 대상 버킷 설정 사용 또는 기본 암호화에 대상 버킷 설정 재정의 중 하나를 선택해야 합니다.

   c. 기본 암호화에 대상 버킷 설정 재정의를 선택하는 경우 다음과 같은 암호화 설정을 구성해야 합니다.

i. 암호화 유형에서 Amazon S3 관리형 키(SSE-S3) 또는 AWS Key Management Service 키(SSE-KMS) 중 하나를 선택해야 합니다. SSE-S3는 가장 강력한 블록 암호 중 하나인 256비트 Advanced Encryption Standard(AES-256)을 사용하여 각 객체를 암호화합니다. SSE-KMS는 키에 대한 더 많은 제어 기능을 제공합니다. 자세한 내용은 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) 사용 및 AWS KMS 키를 사용한 서버 측 암호화(SSE-KMS) 사용(을)를 참조하세요.

ii. AWS Key Management Service 키(SSE-KMS)를 선택하는 경우, AWS KMS key에서 다음 옵션 중 하나를 통해 AWS KMS key를 지정할 수 있습니다.

- 사용 가능한 KMS 키 목록에서 선택하려면 AWS KMS keys 중에서 선택을 선택한 다음, 버킷과 동일한 리전에서 대칭 암호화 KMS 키를 선택합니다. AWS 관리형 키(aws/s3)와 고객 관리형 키가 모두 목록에 표시됩니다.
- KMS 키 ARN을 입력하려면 AWS KMS 키 ARN 입력을 선택하고 나타나는 필드에 KMS 키 ARN을 입력합니다.
- AWS KMS 콘솔에서 고객 관리형 키를 생성하려면 KMS 키 생성을 선택합니다.

iii. 버킷 키(Bucket Key)에서 사용(Enable)을 선택합니다. 복사 작업은 대상 버킷에서 S3 버킷 키를 적용합니다.

8. 작업에 설명을 제공하거나 기본값을 유지하고 우선 순위 수준을 설정하며, 보고서 유형을 선택하고, [완료 보고서 대상 경로(Path to completion report destination)]를 지정합니다.
9. 권한(Permissions) 섹션에서 앞서 정의한 배치 작업 IAM 역할을 선택해야 합니다. [Next]를 선택합니다.
10. [검토(Review)]에서 설정을 확인합니다. 변경하려면 [이전(Previous)]을 선택합니다 배치 작업 설정을 확인한 후 작업 생성(Create job)을 선택합니다.

    자세한 내용은 S3 배치 작업 건 생성 섹션을 참조하세요.

배치 작업 실행

설치 마법사가 Amazon S3 콘솔의 S3 배치 작업 섹션으로 자동으로 돌아갑니다. 새 작업은 S3에서 프로세스를 시작할 때 [신규(New)] 상태에서 [준비 중(Preparing)] 상태로 전환됩니다. 준비 중 상태에서는 S3가 작업의 매니페스트를 읽고 오류를 확인하며 객체 수를 계산합니다.

1. Amazon S3 콘솔에서 새로 고침 버튼을 선택하여 진행 상태를 확인합니다. 매니페스트의 크기에 따라 읽는 데 몇 분 또는 몇 시간이 걸릴 수 있습니다.

2. S3가 작업 매니페스트 읽기를 마치면 작업이 [확인을 기다리는 중(Awaiting your confirmation)] 상태로 이전됩니다. 작업 ID 왼쪽에 있는 옵션 버튼을 선택하고 [작업 실행(Run job)]을 선택합니다.
3. 작업에 대한 설정을 확인하고 오른쪽 하단에 있는 [작업 실행(Run job)]을 선택합니다.

   작업 실행이 시작된 후 특정 작업을 선택하거나 새로 고침 버튼을 선택하여 콘솔 대시보드 보기를 통해 진행 상태를 확인할 수 있습니다.
4. 작업이 완료되면 [성공(Successful)] 및 [실패(Failed)] 객체 개수를 보고 모든 작업이 예상대로 수행되었는지 확인할 수 있습니다. 작업 보고서를 사용하는 경우 보고서에서 실패한 작업의 정확한 원인을 확인합니다.

   AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용해서도 이 단계를 수행할 수 있습니다. 작업 상태 및 완료 보고서 추적에 대한 자세한 내용은 작업 상태 및 완료 보고서 추적 섹션을 참조하세요.

AWS CLI 및 AWS SDK for Java에서 태그를 사용한 복사 작업을 보여 주는 예제는 레이블 지정에 작업 태그를 사용하는 배치 작업 생성 섹션을 참조하세요.

AWS Lambda 함수 호출

Amazon S3 Batch Operations를 사용하여 Amazon S3 객체에 대해 대규모 배치 작업을 수행할 수 있습니다. AWS Lambda 함수 간접 호출 Batch Operations 작업은 매니페스트에 나열된 객체에 대한 사용자 지정 작업을 수행할 AWS Lambda 함수를 시작합니다. 이 섹션에서는 S3 배치 작업에 사용할 Lambda 함수 생성 방법과 함수 호출 작업 생성 방법을 설명합니다. S3 배치 작업은 `LambdaInvoke` 작업을 사용하여 매니페스트에 나열된 모든 객체에 대해 Lambda 함수를 실행합니다.

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하여 S3 Batch Operations와 함께 작업할 수 있습니다. Lambda 사용에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 AWS Lambda 시작하기를 참조하세요.

다음 섹션에서는 S3 배치 작업을 Lambda와 함께 사용하여 시작할 수 있는 방법에 대해 설명합니다.

주제

- Batch Operations에 Lambda 사용
- S3 배치 작업에서 사용할 Lambda 함수 생성
- Lambda 함수를 호출하는 S3 배치 작업 건 생성
- Lambda 매니페스트에서 태스크 수준 정보 제공
- S3 배치 작업 자습서

Batch Operations에 Lambda 사용

S3 배치 작업을 AWS Lambda와 함께 사용할 때 특별히 S3 배치 작업에 사용할 새로운 Lambda 함수를 생성해야 합니다. 기존의 Amazon S3 이벤트 기반 함수를 S3 배치 작업에 재사용할 수는 없습니다. 이벤트 함수는 메시지만 받을 수 있습니다. 메시지를 반환하지 않습니다. S3 배치 작업에 사용되는 Lambda 함수는 메시지를 수락하고 반환해야 합니다. Amazon S3 이벤트와 함께 Lambda를 사용하는 방법에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 Amazon S3에서 AWS Lambda 사용 섹션을 참조하세요.

Lambda 함수를 호출하는 S3 배치 처리 작업을 생성합니다. 이 작업은 매니페스트에 나열된 모든 객체에 대해 동일한 Lambda 함수를 실행합니다. 매니페스트의 객체를 처리하는 동안 사용할 Lambda 함수의 버전을 제어할 수 있습니다. S3 배치 작업은 정규화되지 않은 Amazon 리소스 이름(ARN), 별칭 및 특정 버전을 지원합니다. 자세한 내용은 AWS Lambda 개발자 안내서의 AWS Lambda 버전 관리 소개를 참조하세요.

별칭 또는 `$LATEST` 한정어를 사용하는 함수 ARN에 S3 배치 작업을 제공하고 해당 버전 중 하나를 업데이트하면 S3 배치 작업에서 Lambda 함수의 새 버전 호출을 시작합니다. 이는 라지 작업을 통해 기능 부분을 업데이트하고자 할 때 유용할 수 있습니다. S3 Batch Operations에서 사용되는 버전을 변경하지 않으려면 작업을 만들 때 `FunctionARN` 파라미터에 특정 버전을 제공합니다.

S3 Batch Operations를 사용한 단일 AWS Lambda 작업은 최대 200억 개의 객체가 포함된 매니페스트를 지원할 수 있습니다.

디렉터리 버킷에 Lambda 및 Batch Operations 사용

디렉터리 버킷은 일관되게 10밀리초 미만의 지연 시간이 필요한 워크로드 또는 성능이 중요한 애플리케이션용으로 설계된 Amazon S3 버킷의 유형입니다. 자세한 내용은 디렉터리 버킷을 참조하세요.

배치 작업을 사용하여 디렉터리 버킷에서 작동하는 Lambda 함수를 호출하려면 특별한 요구 사항이 있습니다. 예를 들어, 업데이트된 JSON 스키마를 사용하여 Lambda 요청을 구조화하고 작업을 만들 때 InvocationSchemaVersion 2.0(1.0 아님)을 지정해야 합니다. 이 업데이트된 스키마를 통해 기존 Lambda 함수의 특정 파라미터를 수정하는 데 사용할 수 있는 UserArguments에 대한 선택적 키-값 쌍을 지정할 수 있습니다. 자세한 내용은 AWS 스토리지 블로그에서 Automate object processing in Amazon S3 directory buckets with S3 Batch Operations and AWS Lambda를 참조하세요.

응답 및 결과 코드

S3 배치 작업은 하나 이상의 키를 사용하여 Lambda 함수를 간접 호출하며, 각각에는 연결된 `TaskID`가 있습니다. S3 배치 작업은 Lambda 함수로부터 키별 결과 코드를 기대합니다. 요청으로 전송되었지만 키별 결과 코드와 함께 반환되지 않은 모든 태스크 ID에는 `treatMissingKeysAs`

필드의 결과 코드가 제공됩니다. `treatMissingKeysAs`는 선택적 요청 필드이며 기본값은 `TemporaryFailure`입니다. 다음 표에는 `treatMissingKeysAs` 필드에 사용할 수 있는 다른 결과 코드와 값이 나와 있습니다.

응답 코드	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.

S3 배치 작업에서 사용할 Lambda 함수 생성

이 섹션에서는 Lambda 함수와 함께 사용해야 하는 AWS Identity and Access Management(IAM) 권한 예제를 제공합니다. 또한 S3 배치 작업에 사용할 예제 Lambda 함수도 포함되어 있습니다. 이전에 Lambda 함수를 생성한 적이 없다면 AWS Lambda 개발자 안내서의 자습서: Amazon S3에서 AWS Lambda 사용을 참조하세요.

S3 배치 작업 전용으로 Lambda 함수를 만들어야 합니다. S3 Batch Operations에 사용되는 Lambda 함수는 특수 데이터 필드를 받아들여 반환해야 하기 때문에 기존 Amazon S3 이벤트 기반 Lambda 함수를 다시 사용할 수 없습니다.

> **Important**
>
> Java로 작성된 AWS Lambda 함수는 RequestHandler 또는 RequestStreamHandler 핸들러 인터페이스를 받아들입니다. 그러나 S3 배치 작업 요청 및 응답 형식을 지원하려면 AWS Lambda에 요청 및 응답의 사용자 지정 직렬화 및 역직렬화를 위한 `RequestStreamHandler` 인터페이스가 필요합니다. 이 인터페이스를 통해 Lambda에서 InputStream 및 OutputStream을 Java `handleRequest` 메서드에 전달할 수 있습니다.

S3 배치 작업에 Lambda 함수를 사용할 때는 반드시 `RequestStreamHandler` 인터페이스를 사용해야 합니다. `RequestHandler` 인터페이스를 사용하는 경우 배치 작업이 실패하고 완료 보고서에 "잘못된 JSON이 Lambda 페이로드로 반환됨"이라고 표시됩니다.
자세한 내용은 AWS Lambda 사용 설명서의 핸들러 인터페이스를 참조하세요.

예제 IAM 권한

다음은 Lambda 함수를 S3 배치 작업에 사용하는 데 필요한 IAM 권한의 예제입니다.

Example - 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 읽을 수 있는 S3 배치 작업 권한을 부여하는 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:amzn-s3-demo-bucket1"
        }
    ]
}
```

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\"]"
    }
  ]
}
```

S3 배치 작업에 대한 Lambda 함수 예제

다음 예제 Python Lambda는 버전이 지정된 객체에서 삭제 마커를 제거합니다.

예제에서 볼 수 있듯이 S3 배치 작업의 키는 URL로 인코딩됩니다. Amazon S3를 다른 AWS 서비스와 함께 사용하려면 S3 배치 작업에서 전달된 키를 URL로 디코딩해야 합니다.

```
import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.client("s3")


def lambda_handler(event, context):
    """
    Removes a delete marker from the specified versioned object.

    :param event: The S3 batch event that contains the ID of the delete marker
                  to remove.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of the
             operation. When the result code is TemporaryFailure, S3 retries the
             operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None

    task = event["tasks"][0]
    task_id = task["taskId"]
```

```
    try:
        obj_key = parse.unquote(task["s3Key"], encoding="utf-8")
        obj_version_id = task["s3VersionId"]
        bucket_name = task["s3BucketArn"].split(":")[-1]

        logger.info(
            "Got task: remove delete marker %s from object %s.", obj_version_id,
obj_key
        )

        try:
            # If this call does not raise an error, the object version is not a delete
            # marker and should not be deleted.
            response = s3.head_object(
                Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
            )
            result_code = "PermanentFailure"
            result_string = (
                f"Object {obj_key}, ID {obj_version_id} is not " f"a delete marker."
            )

            logger.debug(response)
            logger.warning(result_string)
        except ClientError as error:
            delete_marker = error.response["ResponseMetadata"]["HTTPHeaders"].get(
                "x-amz-delete-marker", "false"
            )
            if delete_marker == "true":
                logger.info(
                    "Object %s, version %s is a delete marker.", obj_key,
obj_version_id
                )
                try:
                    s3.delete_object(
                        Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
                    )
                    result_code = "Succeeded"
                    result_string = (
                        f"Successfully removed delete marker "
                        f"{obj_version_id} from object {obj_key}."
                    )
                    logger.info(result_string)
                except ClientError as error:
```

```
                    # Mark request timeout as a temporary failure so it will be
retried.
                    if error.response["Error"]["Code"] == "RequestTimeout":
                        result_code = "TemporaryFailure"
                        result_string = (
                            f"Attempt to remove delete marker from  "
                            f"object {obj_key} timed out."
                        )
                        logger.info(result_string)
                    else:
                        raise
            else:
                raise ValueError(
                    f"The x-amz-delete-marker header is either not "
                    f"present or is not 'true'."
                )
    except Exception as error:
        # Mark all other exceptions as permanent failures.
        result_code = "PermanentFailure"
        result_string = str(error)
        logger.exception(error)
    finally:
        results.append(
            {
                "taskId": task_id,
                "resultCode": result_code,
                "resultString": result_string,
            }
        )
    return {
        "invocationSchemaVersion": invocation_schema_version,
        "treatMissingKeysAs": "PermanentFailure",
        "invocationId": invocation_id,
        "results": results,
    }
```

Lambda 함수를 호출하는 S3 배치 작업 건 생성

Lambda 함수를 호출하기 위해 S3 배치 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`

S3 배치 작업 생성에 대한 자세한 내용은 S3 배치 작업 건 생성 및 S3 배치 작업에서 지원하는 작업 섹션을 참조하세요.

다음 예제에서는 AWS CLI를 사용하여 Lambda 함수를 간접 호출하는 S3 Batch Operations 작업을 생성합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws s3control create-job
    --account-id account-id
    --operation  '{"LambdaInvoke": { "FunctionArn": "arn:aws:lambda:region:account-
id:function:LambdaFunctionName" } }'
    --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
["Bucket","Key"]},"Location":{"ObjectArn":"arn:aws:s3:::amzn-s3-demo-manifest-
bucket","ETag":"ManifestETag"}}'
    --report '{"Bucket":"arn:aws:s3:::amzn-s3-demo-
bucket","Format":"Report_CSV_20180820","Enabled":true,"Prefix":"ReportPrefix","ReportScope":"Al
    --priority 2
    --role-arn arn:aws:iam::account-id:role/BatchOperationsRole
    --region region
    --description "Lambda Function"
```

Lambda 매니페스트에서 태스크 수준 정보 제공

S3 Batch Operations에서 AWS Lambda 함수를 사용할 때는 작업 중인 각 태스크나 키에 추가 데이터를 포함해야 할 수도 있습니다. 예를 들어, 소스 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 Batch Operations는 작업에 대해 입력 매니페스트의 대상 버킷과 소스 키 목록만 지정할 수 있도록 합니다. 아래 예제에는 보다 복잡한 Lambda 함수를 실행할 수 있도록 매니페스트에 추가 데이터를 포함시킬 수 있는 방법이 나와 있습니다.

S3 배치 작업 매니페스트에 Lambda 함수 코드에서 사용할 키별 파라미터를 지정하려면 다음과 같이 URL 인코딩된 JSON 형식을 사용하세요. `key` 필드는 마치 Amazon S3 객체 키인 것처럼 Lambda 함수로 전달됩니다. 그러나 아래 예제와 같이 Lambda 함수로 다른 값이나 여러 키를 포함하도록 해석할 수 있습니다.

> **Note**
>
> 매니페스트에서 key 필드의 최대 문자 수는 1,024자입니다.

Example - 'Amazon S3 키'를 JSON 문자열로 대체하는 매니페스트

URL 인코딩 버전을 S3 배치 작업에 제공해야 합니다.

```
amzn-s3-demo-bucket,{"origKey": "object1key", "newKey": "newObject1Key"}
amzn-s3-demo-bucket,{"origKey": "object2key", "newKey": "newObject2Key"}
amzn-s3-demo-bucket,{"origKey": "object3key", "newKey": "newObject3Key"}
```

Example - URL 인코딩된 매니페스트

이러한 URL 인코딩 버전을 S3 배치 작업에 제공해야 합니다. URL 인코딩 버전이 아니면 작동하지 않습니다.

```
amzn-s3-demo-bucket,%7B%22origKey%22%3A%20%22object1key%22%2C%20%22newKey%22%3A
%20%22newObject1Key%22%7D
amzn-s3-demo-bucket,%7B%22origKey%22%3A%20%22object2key%22%2C%20%22newKey%22%3A
%20%22newObject2Key%22%7D
amzn-s3-demo-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

URL로 인코딩된 이 매니페스트 예시에는 다음 Lambda 함수에서 구문 분석을 할 수 있는 파이프로 구분된 객체 키가 포함되어 있습니다.

```
amzn-s3-demo-bucket,object1key%7Clower
amzn-s3-demo-bucket,object2key%7Cupper
amzn-s3-demo-bucket,object3key%7Creverse
amzn-s3-demo-bucket,object4key%7Cdelete
```

이 Lambda 함수는 파이프로 제한된 태스크를 S3 Batch Operations 매니페스트에 인코딩하여 구문 분석하는 방법을 보여줍니다. 이 태스크는 지정된 객체에 적용되는 개정 작업을 나타냅니다.

```
import logging
from urllib import parse
import boto3
```

```
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.resource("s3")


def lambda_handler(event, context):
    """
    Applies the specified revision to the specified object.

    :param event: The Amazon S3 batch event that contains the ID of the object to
                  revise and the revision type to apply.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of the
             operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None

    task = event["tasks"][0]
    task_id = task["taskId"]
    # The revision type is packed with the object key as a pipe-delimited string.
    obj_key, revision = parse.unquote(task["s3Key"], encoding="utf-8").split("|")
    bucket_name = task["s3BucketArn"].split(":")[-1]

    logger.info("Got task: apply revision %s to %s.", revision, obj_key)

    try:
        stanza_obj = s3.Bucket(bucket_name).Object(obj_key)
        stanza = stanza_obj.get()["Body"].read().decode("utf-8")
        if revision == "lower":
            stanza = stanza.lower()
        elif revision == "upper":
            stanza = stanza.upper()
        elif revision == "reverse":
            stanza = stanza[::-1]
        elif revision == "delete":
```

```
            pass
        else:
            raise TypeError(f"Can't handle revision type '{revision}'.")

        if revision == "delete":
            stanza_obj.delete()
            result_string = f"Deleted stanza {stanza_obj.key}."
        else:
            stanza_obj.put(Body=bytes(stanza, "utf-8"))
            result_string = (
                f"Applied revision type '{revision}' to " f"stanza {stanza_obj.key}."
            )

        logger.info(result_string)
        result_code = "Succeeded"
    except ClientError as error:
        if error.response["Error"]["Code"] == "NoSuchKey":
            result_code = "Succeeded"
            result_string = (
                f"Stanza {obj_key} not found, assuming it was deleted "
                f"in an earlier revision."
            )
            logger.info(result_string)
        else:
            result_code = "PermanentFailure"
            result_string = (
                f"Got exception when applying revision type '{revision}' "
                f"to {obj_key}: {error}."
            )
            logger.exception(result_string)
    finally:
        results.append(
            {
                "taskId": task_id,
                "resultCode": result_code,
                "resultString": result_string,
            }
        )
    return {
        "invocationSchemaVersion": invocation_schema_version,
        "treatMissingKeysAs": "PermanentFailure",
        "invocationId": invocation_id,
        "results": results,
    }
```

S3 배치 작업 자습서

다음 자습서는 Lambda를 사용한 일부 배치 작업 태스크 절차를 끝까지 완전히 보여줍니다. 이 자습서에서는 S3 소스 버킷에 저장된 비디오의 일괄 트랜스코딩을 위해 Lambda 함수를 간접 호출하도록 Batch Operations를 설정하는 방법에 대해 알아봅니다. Lambda 함수는 AWS Elemental MediaConvert를 직접 호출하여 비디오를 트랜스코딩합니다.

- 자습서: S3 Batch Operations를 통해 비디오 일괄 트랜스코딩

모든 객체 태그 교체

Amazon S3 Batch Operations를 사용하여 Amazon S3 객체에 대해 대규모 배치 작업을 수행할 수 있습니다. 모든 객체 태그 교체 작업은 매니페스트에 나열된 모든 객체의 객체 태그를 교체합니다. 객체 태그는 객체에 대한 메타데이터를 저장하는 데 사용할 수 있는 문자열의 키-값 쌍입니다.

모든 객체 태그 교체 작업을 만들려면 적용할 태그 집합을 제공합니다. S3 배치 작업은 모든 객체에 동일한 태그 세트를 적용합니다. 사용자가 제공한 태그 세트는 매니페스트에서 이미 객체에 연결된 모든 태그 세트를 대체합니다. S3 Batch Operations는 기존 태그는 그대로 두고 객체에 태그를 추가하는 것은 지원하지 않습니다.

매니페스트의 객체가 버전 지정된 버킷에 있는 경우 모든 객체의 특정 버전에 태그 세트를 적용할 수 있습니다. 그렇게 하려면 매니페스트의 모든 객체에 대한 버전 ID를 지정합니다. 객체에 대한 버전 ID를 포함하지 않을 경우 S3 Batch Operations가 모든 객체의 최신 버전에 태그 집합을 적용합니다. 배치 작업 매니페스트에 대한 자세한 내용은 매니페스트 지정 섹션을 참조하세요.

객체 태그 지정에 대한 자세한 내용을 알아보려면 이 가이드의 태그를 사용하여 스토리지 분류 섹션을 살펴보고, Amazon Simple Storage Service API 참조의 PutObjectTagging, GetObjectTagging, DeleteObjectTagging 섹션을 참조하세요.

콘솔을 사용하여 모든 객체 태그 교체 작업을 만들려면 S3 배치 작업 건 생성 섹션을 참조하세요.

규제 및 제한

Batch Operations를 사용하여 객체 태그를 교체하는 경우 다음 규제와 제한 사항이 적용됩니다.

- Batch Operations 작업을 실행하기 위해 지정하는 AWS Identity and Access Management(IAM) 역할에는 기본 `PutObjectTagging` 작업을 수행할 수 있는 권한이 있어야 합니다. 필요한 권한에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 PutObjectTagging 섹션을 참조하세요.
- S3 Batch Operations는 Amazon S3 PutObjectTagging 작업을 사용하여 매니페스트 내 각 객체에 태그를 적용합니다. 기본 작업에 적용되는 모든 규제 및 제한은 S3 배치 작업에도 적용됩니다.
- 단일 모든 객체 태그 교체 작업은 최대 200억 개의 객체가 포함된 매니페스트를 지원할 수 있습니다.

모든 객체 태그 삭제

Amazon S3 Batch Operations를 사용하여 Amazon S3 객체에 대해 대규모 배치 작업을 수행할 수 있습니다. 모든 객체 태그 삭제 작업은 매니페스트에 나열된 객체와 현재 연결된 모든 Amazon S3 객체 태그 세트를 제거합니다. S3 Batch Operations는 다른 태그를 그대로 남겨 두면서 객체에서 태그를 삭제하는 작업을 지원하지 않습니다.

매니페스트의 객체가 버전 지정된 버킷에 있는 경우, 특정 버전의 객체에서 태그 세트를 제거할 수 있습니다. 그렇게 하려면 매니페스트의 모든 객체에 대한 버전 ID를 지정해야 합니다. 객체에 대한 버전 ID를 포함하지 않으면 S3 배치 작업이 모든 객체의 최신 버전에서 태그 세트를 제거합니다. 배치 작업 매니페스트에 대한 자세한 내용은 매니페스트 지정 섹션을 참조하세요.

객체 태그 지정에 대한 자세한 내용을 알아보려면 이 가이드의 태그를 사용하여 스토리지 분류 섹션을 살펴보고, Amazon Simple Storage Service API 참조의 PutObjectTagging, GetObjectTagging, DeleteObjectTagging 섹션을 참조하세요.

> **Warning**
>
> 이 작업을 실행하면 매니페스트에 나열된 모든 객체의 모든 객체 태그 세트가 제거됩니다.

콘솔을 사용하여 모든 객체 태그 삭제 작업을 만들려면 S3 배치 작업 건 생성 섹션을 참조하세요.

규제 및 제한

Batch Operations를 사용하여 객체 태그를 삭제하는 경우 다음 규제와 제한 사항이 적용됩니다.

- 작업을 실행하기 위해 지정하는 AWS Identity and Access Management(IAM) 역할에는 기본 Amazon S3 `DeleteObjectTagging` 작업을 수행할 수 있는 권한이 있어야 합니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 DeleteObjectTagging를 참조하십시오.

- S3 Batch Operations는 Amazon S3 DeleteObjectTagging 작업을 사용하여 매니페스트의 모든 객체에서 태그 집합을 제거합니다. 기본 작업에 적용되는 모든 규제 및 제한은 S3 배치 작업에도 적용됩니다.
- 단일 객체 태그 삭제 작업은 최대 200억 개의 객체가 포함된 매니페스트를 지원할 수 있습니다.

액세스 제어 목록(ACL) 대체

Amazon S3 Batch Operations를 사용하여 Amazon S3 객체에 대해 대규모 배치 작업을 수행할 수 있습니다. 액세스 제어 목록(ACL) 교체 작업은 매니페스트에 나열된 모든 객체에 대해 액세스 제어 목록(ACL)을 교체합니다. ACL을 통해 사용자가 액세스할 수 있는 객체와 수행할 수 있는 작업을 정의할 수 있습니다.

> **Note**
>
> Amazon S3의 최신 사용 사례 대부분은 더 이상 ACL을 사용할 필요가 없습니다. 각 객체에 대해 액세스를 개별적으로 제어할 필요가 있는 드문 상황을 제외하고는 ACL을 비활성화한 채로 두는 것이 좋습니다. ACL을 비활성화하면 누가 객체를 버킷에 업로드했는지에 관계없이 정책을 사용하여 버킷의 모든 객체에 대한 액세스를 제어할 수 있습니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

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 모든 객체의 특정 버전에 ACL을 적용할 수 있습니다. 그렇게 하려면 매니페스트의 모든 객체에 대한 버전 ID를 지정합니다. 객체에 대한 버전 ID를 포함하지 않을 경우 S3 Batch Operations가 해당 객체의 최신 버전에 ACL을 적용합니다.

Amazon S3의 ACL에 대한 자세한 내용은 ACL(액세스 제어 목록) 개요 섹션을 참조하세요.

S3 Block Public Access

버킷의 모든 객체에 대한 퍼블릭 액세스를 제한하려면 S3 Batch Operations를 사용하여 ACL을 적용하는 대신 Amazon S3 Block Public Access를 사용하는 것이 좋습니다. 퍼블릭 액세스 차단은 빠르게 효과를 발휘하는 간단한 단일 작업으로 버킷별로 또는 계정 전체에서 퍼블릭 액세스를 제한할 수 있습니다. 이러한 동작으로 인해 버킷이나 계정의 모든 객체에 대한 퍼블릭 액세스를 제어하는 것이 목표일 경우 Amazon S3 Block Public Access가 더 나은 선택이 됩니다. 매니페스트의 모든 객체에 대해 사용자 지정된 ACL을 적용해야 할 경우에는 S3 Batch Operations만 사용합니다. S3 퍼블릭 액세스 차단에 대한 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하세요.

S3 객체 소유권

매니페스트의 객체가 객체 소유권에 대한 버킷 소유자 적용 설정을 사용하는 버킷에 있는 경우, 액세스 제어 목록(ACL) 교체 작업은 버킷 소유자에게 전체 제어권을 부여하는 객체 ACL만 지정할 수 있습니다. 이 경우 액세스 제어 목록(ACL) 교체 작업은 다른 AWS 계정 또는 그룹에 객체 ACL 권한을 부여할 수 없습니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

규제 및 제한

Batch Operations를 사용하여 ACL을 교체하는 경우 다음 규제와 제한 사항이 적용됩니다.

- 액세스 제어 목록(ACL) 교체 작업을 실행하기 위해 지정하는 AWS Identity and Access Management(IAM) 역할에는 기본 Amazon S3 `PutObjectAcl` 작업을 수행할 수 있는 권한이 있어야 합니다. 필요한 권한에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 PutObjectAcl 섹션을 참조하세요.
- S3 배치 작업은 Amazon S3 `PutObjectAcl` 작업을 사용하여 매니페스트의 모든 객체에 지정된 ACL을 적용합니다. 따라서 기본 `PutObjectAcl` 작업에 적용되는 모든 규제 및 제한은 S3 Batch Operations 액세스 제어 목록(ACL) 교체 작업에도 적용됩니다.
- 단일 액세스 제어 목록 교체 작업은 최대 200억 개의 객체가 포함된 매니페스트를 지원할 수 있습니다.

배치 작업을 통한 객체 복원

Amazon S3 Batch Operations를 사용하여 Amazon S3 객체에 대해 대규모 배치 작업을 수행할 수 있습니다. 복원 작업은 매니페스트에 나열되어 있는 아카이브된 객체에 대한 복원 요청을 시작합니다. 다음 아카이브된 객체는 먼저 복원해야 실시간으로 액세스할 수 있습니다.

- S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스에 아카이브된 객체
- Archive Access 또는 Deep Archive Access 계층의 S3 Intelligent-Tiering 스토리지 클래스를 통해 보관된 객체

S3 Batch Operations 작업에서 복원(S3InitiateRestoreObjectOperation) 작업을 사용하면 매니페스트에 지정된 모든 객체에 대해 `RestoreObject` 요청이 이루어집니다.

> **Important**
>
> 복원 작업은 객체 복원 요청만 시작합니다. 각 객체에 대해 요청이 시작되면 S3 배치 작업이 각 객체에 대해 작업을 완료로 보고합니다. Amazon S3는 객체 복원 시 작업을 업데이트하거나 알

림을 제공하지 않습니다. 하지만 S3 이벤트 알림을 사용하여 Amazon S3에서 객체를 사용할 수 있을 때 알림을 받을 수 있습니다. 자세한 내용은 Amazon S3 이벤트 알림 섹션을 참조하세요.

복원 작업을 만들 때 다음 인수를 사용할 수 있습니다.

ExpirationInDays

이 인수는 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 객체를 Amazon S3에서 사용할 수 있는 기간을 지정합니다. S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive 객체를 대상으로 하는 객체 복원 작업에는 1 이상으로 설정한 `ExpirationInDays`가 필요합니다.

> **Important**
>
> S3 Intelligent-Tiering Archive Access 및 Deep Archive Access 계층 객체를 대상으로 하는 복원 작업을 만들 때 `ExpirationInDays`를 설정하지 마세요. S3 Intelligent-Tiering Archive Access 계층의 객체는 복원 만료 대상이 아니므로, `ExpirationInDays`를 지정하면 `RestoreObject` 요청이 실패합니다.

GlacierJobTier

Amazon S3는 3가지 검색 계층(`EXPEDITED`, `STANDARD`, `BULK`) 중 하나를 사용하여 객체를 복원할 수 있습니다. 그러나 S3 배치 작업 기능은 `STANDARD` 및 `BULK` 검색 계층만 지원합니다. 검색 계층 간의 차이점에 대한 자세한 내용은 아카이브 가져오기 옵션 이해 섹션을 참조하세요.

각 계층의 요금에 대한 자세한 내용은 Amazon S3 요금 페이지의 요청 및 데이터 검색 섹션을 참조하세요.

S3 Glacier와 S3 Intelligent-Tiering에서 복원할 때의 차이점

S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스에서 아카이브된 파일을 복원하는 것은 Archive Access 또는 Deep Archive Access 계층의 S3 Intelligent-Tiering 스토리지 클래스에서 파일을 복원하는 것과 다릅니다.

- S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive에서 복원하면 객체의 임시 사본이 생성됩니다. Amazon S3는 `ExpirationInDays` 인수에 지정한 값이 경과된 후 이 사본을 삭제합니다. 이 임시 사본이 삭제된 후, 사용자는 추가 복원 요청을 제출하여 객체에 액세스해야 합니다.

- 아카이브된 S3 Intelligent-Tiering 객체를 복원할 때는 `ExpirationInDays` 인수를 지정하지 마세요. S3 Intelligent-Tiering Archive Access 또는 Deep Archive Access 계층에서 객체를 복원하면 객체가 S3 Intelligent-Tiering Frequent Access 계층으로 다시 전환됩니다. 액세스하지 않은 기간이 최소 연속 90일이 넘으면 객체가 자동으로 Archive Access 계층으로 전환됩니다. 액세스하지 않은 기간이 최소 연속 180일이 넘으면 객체가 자동으로 Deep Archive Access 계층으로 이동합니다.
- 배치 작업 건은 S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive 스토리지 클래스 객체 또는 S3 Intelligent-Tiering Archive Access 및 Deep Archive Access 스토리지 계층 객체에서 작동할 수 있습니다. 배치 작업은 동일한 작업 건에 있는 두 가지 유형의 아카이브된 객체 모두에서 작동할 수 없습니다. 두 유형의 객체를 복원하려면 별도의 배치 작업을 생성해야 합니다.

중복 복원

S3InitiateRestoreObjectOperation 작업이 이미 복원 과정 중에 있는 객체를 복원하려고 시도할 경우 S3 Batch Operations가 다음과 같이 진행합니다.

다음 조건 중 하나가 true일 경우 해당 객체의 복원 작업은 성공합니다.

- 이미 진행 중인 복원 요청과 비교해, 이 작업의 `ExpirationInDays` 값이 동일하고 `GlacierJobTier` 값이 더 빠른 경우.
- 이전 복원 요청이 이미 완료되었으며 현재 객체를 사용할 수 있습니다. 이 경우, 배치 작업은 복원된 객체의 만료 날짜를 진행 중인 복원 요청에 지정된 `ExpirationInDays` 값과 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` 값으로 지정됨)이 진행 중인 복원 요청에 지정된 복원 기간과 다릅니다.
- 이 작업의 복원 계층(`GlacierJobTier` 값으로 지정됨)이 이미 진행 중인 복원 요청에 지정된 복원 계층보다 느리거나 같습니다.

제한 사항

`S3InitiateRestoreObjectOperation` 작업에는 다음과 같은 제한 사항이 있습니다.

- 보관된 객체와 동일한 리전에서 작업을 생성해야 합니다.
- S3 Batch Operations는 `EXPEDITED` 검색 계층을 지원하지 않습니다.

- 단일 Batch Operations 복원 작업은 최대 40억 개의 객체가 포함된 매니페스트를 지원할 수 있습니다.

객체 복원에 대한 자세한 내용은 아카이브된 객체 복원 섹션을 참조하세요.

S3 객체 잠금 보존

Amazon S3 Batch Operations를 사용하여 Amazon S3 객체에 대해 대규모 배치 작업을 수행할 수 있습니다. Object Lock 보존 작업을 사용하여 거버넌스 모드 또는 규정 준수 모드를 통해 객체에 대한 보존 날짜를 적용할 수 있습니다. 이러한 보관 모드는 다양한 수준의 보호를 적용합니다. 두 보관 모드를 모든 객체 버전에 적용할 수 있습니다. 법적 보존과 마찬가지로 보관 날짜를 사용하면 객체를 덮어쓰거나 삭제할 수 없습니다. Amazon S3는 객체의 메타데이터에 지정된 보존 종료일을 저장하고 보존 기간이 만료될 때까지 지정된 버전의 객체 버전을 보호합니다.

Object Lock을 적용한 S3 Batch Operations를 사용하여 여러 Amazon S3 객체의 보존 날짜를 한 번에 관리할 수 있습니다. 매니페스트에서 대상 객체 목록을 지정하고 매니페스트를 Batch Operations에 제출하여 완료할 수 있습니다. 자세한 내용은 S3 객체 잠금 the section called “보관 기간” 섹션을 참조하세요.

보관 날짜가 있는 S3 배치 작업은 완료될 때까지, 취소될 때까지 또는 실패 상태에 도달할 때까지 실행됩니다. 단일 요청으로 많은 객체에 대한 보존 날짜를 추가, 변경 또는 제거하려는 경우 S3 Batch Operations 및 S3 Object Lock 보존을 사용하는 것이 좋습니다.

배치 작업은 매니페스트의 키를 처리하기 전에 버킷에서 객체 잠금이 사용 설정되어 있는지 확인합니다. 작업 및 유효성 검사를 수행하려면 Batch Operations가 사용자를 대신하여 Object Lock을 직접 호출할 수 있도록 Batch Operations에 AWS Identity and Access Management(IAM) 역할의 `s3:GetBucketObjectLockConfiguration` 및 `s3:PutObjectRetention` 권한이 필요합니다. 자세한 내용은 the section called “객체 잠금 고려 사항” 섹션을 참조하세요.

REST API에서 이 작업을 사용하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 CreateJob 작업에 있는 `S3PutObjectRetention` 섹션을 참조하세요.

이 작업을 사용하는 AWS Command Line Interface(AWS CLI) 예제는 the section called “객체 잠금 보존과 함께 배치 작업 사용” 섹션을 참조하세요. AWS SDK for Java 예제는 the section called “객체 잠금 보존과 함께 배치 작업 사용” 섹션을 참조하세요.

규제 및 제한

Batch Operations를 사용하여 Object Lock 보존 기간을 적용하는 경우 다음과 같은 규제와 제한 사항이 적용됩니다.

- S3 Batch Operations는 버킷 수준을 변경하지 않습니다.
- 버전 관리 및 S3 객체 잠금은 작업이 수행되는 버킷에서 구성해야 합니다.
- 매니페스트에 나열된 모든 객체는 동일한 버킷에 있어야 합니다.
- 매니페스트에 버전이 명시적으로 지정되지 않은 경우 이 작업은 최신 버전의 객체에서 작동합니다.
- Object Lock 보존 작업을 사용하려면 IAM 역할의 `s3:PutObjectRetention` 권한이 필요합니다.
- 작업을 수행 중인 S3 버킷에 대해 Object Lock이 활성화되어 있는지 확인하려면 `s3:GetBucketObjectLockConfiguration` IAM 권한이 필요합니다.
- `COMPLIANCE` 모드 보존 날짜가 적용된 객체의 보존 기간만 연장할 수 있으며, 이 보존 기간은 단축할 수 없습니다.
- 단일 S3 Object Lock 보존 작업은 최대 200억 개의 객체가 포함된 매니페스트를 지원할 수 있습니다.

S3 객체 잠금 법적 보존

Amazon S3 Batch Operations를 사용하여 Amazon S3 객체에 대해 대규모 배치 작업을 수행할 수 있습니다. Object Lock 법적 보존 작업을 사용하여 객체 버전에 법적 보존을 설정할 수 있습니다. 보관 기간 설정과 마찬가지로 법적 보존을 사용하면 객체 버전을 덮어쓰거나 삭제할 수 없습니다. 그러나 법적 보존에는 연결된 보존 기간이 없고, 제거될 때까지 유효합니다.

객체 잠금을 적용한 S3 배치 작업을 사용하여 법적 보존을 한 번에 여러 Amazon S3 객체에 추가할 수 있습니다. 그렇게 하려면 매니페스트에서 대상 객체 목록을 지정하고 해당 목록을 Batch Operations에 제출하면 됩니다. S3 Batch Operations Object Lock 법적 보존 작업은 완료, 취소 또는 실패 상태에 도달할 때까지 실행됩니다.

S3 Batch Operations는 매니페스트의 객체를 처리하기 전에 S3 버킷에서 Object Lock이 활성화되어 있는지 확인합니다. 객체 작업 및 버킷 수준 검증을 수행하려면 S3 Batch Operations의 AWS Identity and Access Management(IAM) 역할에 `s3:PutObjectLegalHold` 및 `s3:GetBucketObjectLockConfiguration`이 필요합니다. 이러한 권한을 통해 S3 Batch Operations는 사용자를 대신하여 S3 Object Lock을 직접 호출할 수 있습니다.

법적 보존을 제거하기 위해 S3 Batch Operations 작업을 만들 때 법적 보존 상태로 `Off`를 지정하면 됩니다. 자세한 내용은 the section called “객체 잠금 고려 사항” 섹션을 참조하세요.

Amazon S3 REST API에서 이 작업을 사용하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 CreateJob 작업에 있는 `S3PutObjectLegalHold` 섹션을 참조하세요.

이 작업을 사용하는 예제는 Java용 AWS SDK 사용 섹션을 참조하세요.

규제 및 제한

Batch Operations를 사용하여 Object Lock 법적 보존을 적용하거나 제거하는 경우 다음과 같은 규제와 제한 사항이 적용됩니다.

- S3 Batch Operations는 버킷 수준을 변경하지 않습니다.
- 매니페스트에 나열된 모든 객체는 동일한 버킷에 있어야 합니다.
- 버전 관리 및 S3 객체 잠금은 작업이 수행되는 버킷에서 구성해야 합니다.
- 매니페스트에 버전이 명시적으로 지정되지 않은 경우 Object Lock 법적 보존 작업은 최신 버전의 객체에서 작동합니다.
- 객체에 대한 법적 보존을 추가하거나 제거하려면 IAM 역할에 `s3:PutObjectLegalHold` 권한이 필요합니다.
- 작업이 수행되는 S3 버킷에 대해 S3 Object Lock이 활성화되어 있는지 확인하려면 `s3:GetBucketObjectLockConfiguration` IAM 권한이 필요합니다.
- 단일 S3 Object Lock 법적 보존 작업은 최대 200억 개의 객체가 포함된 매니페스트를 지원할 수 있습니다.

- 객체 복사
- AWS Lambda 함수 호출
- 모든 객체 태그 교체
- 모든 객체 태그 삭제
- 액세스 제어 목록(ACL) 대체
- 배치 작업을 통한 객체 복원
- S3 객체 잠금 보존
- S3 객체 잠금 법적 보존
- 배치 복제를 사용한 기존 객체 복제

S3 배치 작업 건 관리

Amazon S3는 S3 배치 작업을 생성한 후 관리하는 데 도움이 되는 강력한 도구 세트를 제공합니다. 이 섹션에서는 Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하여 작업을 관리하고 추적하는 데 사용할 수 있는 작업에 대해 설명합니다.

주제

- Amazon S3 콘솔을 사용하여 S3 배치 작업 관리
- 작업 나열
- 작업 세부 정보 보기
- 작업 우선 순위 지정

Amazon S3 콘솔을 사용하여 S3 배치 작업 관리

콘솔을 사용하여 S3 배치 작업을 관리할 수 있습니다. 예를 들어, 다음을 수행할 수 있습니다.

- 활성 및 대기 중인 작업 보기
- 작업의 우선 순위 변경
- 작업 확인 및 실행
- 작업 복제
- 작업 취소

콘솔을 사용하여 배치 작업 관리

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 배치 작업을 선택합니다.
3. 관리하려는 특정 작업을 선택합니다.

작업 나열

S3 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.

`Active` 및 `Complete` 작업 목록 가져오기

다음 AWS CLI 예제에서는 `Active` 및 `Complete` 작업의 목록을 가져옵니다. 이 예제를 사용하려면 *`user input placeholders`*를 사용자의 정보로 대체합니다.

```
aws s3control list-jobs \
```

```
    --region us-west-2 \
    --account-id account-id \
    --job-statuses '["Active","Complete"]' \
    --max-results 20
```

자세한 내용과 예제는 https://awscli.amazonaws.com/v2/documentation/api/latest/reference/s3control/list-jobs.htmllist-jobs 명령 참조에서 AWS CLI를 참조하세요.

작업 세부 정보 보기

작업을 나열하여 검색할 수 있는 것보다 Amazon S3 Batch Operations 작업에 대한 더 많은 정보를 원하는 경우, 단일 작업에 대한 모든 세부 정보를 확인하면 됩니다. 아직 완료되지 않은 작업이나 지난 90일 이내에 완료된 작업에 대한 세부 정보를 볼 수 있습니다. 작업 목록에서 반환되는 정보 이외에 단일 작업의 세부 정보는 다른 항목도 포함합니다.

- 작업 파라미터
- 매니페스트에 대한 세부 정보
- 완료 보고서에 대한 정보(작업을 생성할 때 보고서를 구성한 경우)
- 작업을 실행하도록 할당된 AWS Identity and Access Management(IAM) 사용자 역할의 Amazon 리소스 이름(ARN)

개별 작업의 세부 정보를 보면서 작업의 전체 구성에 액세스할 수 있습니다. 작업의 세부 정보를 보려면 Amazon S3 콘솔 또는 AWS Command Line Interface(AWS CLI)를 사용할 수 있습니다.

Amazon S3 콘솔에서 S3 배치 작업에 대한 작업 설명 가져오기

콘솔을 사용하여 배치 작업에 대한 작업 설명을 보는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 배치 작업을 선택합니다.
3. 세부 정보를 보려면 특정 작업의 작업 ID를 선택합니다.

AWS CLI에서 S3 배치 작업에 대한 작업 설명 가져오기

다음 예시에서는 AWS CLI를 사용하여 S3 배치 작업에 대한 작업 설명을 가져옵니다. 다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3control describe-job \
--region us-west-2 \
--account-id account-id \
--job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c
```

자세한 내용과 예제는 https://awscli.amazonaws.com/v2/documentation/api/latest/reference/s3control/describe-job.htmldescribe-job 명령 참조에서 AWS CLI를 참조하세요.

작업 우선 순위 지정

각 Amazon S3 Batch Operations 작업에 우선순위 번호(임의의 양수)를 지정할 수 있습니다. S3 Batch Operations는 할당된 우선순위에 따라 작업의 우선순위를 지정합니다. 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 우선순위를 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.

작업의 우선순위를 변경해도 작업 처리 속도에 영향을 미치지 않습니다.

> **Note**
>
> S3 Batch Operations는 최대 효과에 기초하여 작업 우선순위를 준수합니다. 우선순위가 높은 작업이 일반적으로 우선순위가 낮은 작업보다 우선하지만, Amazon S3는 엄격한 작업 순서를 보장하지 않습니다.

S3 콘솔 사용

Amazon S3 콘솔에서 작업 우선순위를 업데이트하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 배치 작업을 선택합니다.
3. 관리할 특정 작업을 선택합니다.
4. 작업을 선택합니다. 드롭다운 목록에서 업데이트 우선 순위(Update priority)를 선택합니다.

AWS CLI 사용

다음 예제에서는 AWS CLI를 사용하여 작업의 우선순위를 업데이트합니다. 숫자가 높을 수록 실행 우선 순위가 높아집니다. 다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3control update-job-priority \
    --region us-west-2 \
    --account-id account-id \
    --priority 98 \
    --job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c
```

AWS SDK for Java 사용

다음 예제에서는 AWS SDK for Java를 사용하여 S3 배치 작업의 우선 순위를 업데이트합니다.

작업 우선 순위에 대한 자세한 내용은 작업 우선 순위 지정 섹션을 참조하세요.

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();
        }
    }
}
```

작업 상태 및 완료 보고서 추적

S3 배치 작업에서는 작업 상태를 보고 업데이트하고, 알림 및 로깅을 추가하고, 작업 실패를 추적하고, 완료 보고서를 생성할 수 있습니다.

주제

- 작업 상태
- 작업 상태 업데이트
- 알림 및 로깅
- 작업 실패 추적
- 완료 보고서
- 예: AWS CloudTrail을(를) 통해 Amazon EventBridge에서 S3 배치 작업 추적
- 예: S3 배치 작업 완료 보고서

작업 상태

작업을 생성하고 실행하면 작업은 일련의 상태를 거칩니다. 다음 표에서는 상태 및 상태 간의 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 확인할 때까지 Suspended 상태로 유지됩니다.
Suspended	작업이 확인을 필요로 하지만, 사용자가 아직 실행을 확인하지 않았습니다. Amazon S3 콘솔을 사용하여 생성하는 작업만 확인이 필요합니다. 콘솔을 통해 만든 작업은 Suspended 상태에서 바로 Preparing 상태로 들어갑니다. 사용자가 작업 실행을 확인하여 Ready 상태가 된 이후에는 작업이 절대로 Suspended 상태로 되돌아가지 않습니다.	작업 실행을 확인하면 작업 상태가 Ready로 변경됩니다.

상태	설명	Transitions
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. 이 동작은 작업이 성공적으로 완료되거나 실패한 경우와 같이 자동으로 발생할 수 있습니다. 또는 작업 취소와 같이 사용자 작업의 결과로 동작이 발생할 수 있습니다. 이후 작업이 이동하는 상태는 전환 이유에 따라 다릅니다.
Pausing	작업이 다른 상태에서 Paused 상태로 전환합니다.	Paused 단계가 완료되면 작업이 자동으로 Pausing 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.

상태	설명	Transitions
`Complete`	작업이 매니페스트의 모든 객체에 대해 요청된 작업을 완료했습니다. 모든 객체에 대해 작업이 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.
`Cancelling`	작업이 `Cancelled` 상태로 전환 중입니다.	`Cancelled` 단계가 완료되면 작업이 자동으로 `Cancelling` 상태로 이동합니다.
`Cancelled`	사용자가 작업 취소를 요청했고 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.	`Failed` 단계가 완료되면 작업이 자동으로 `Failing` 상태로 이동합니다.
`Failed`	작업이 실패했고 더 이상 실행되지 않습니다. 작업 실패에 대한 자세한 내용은 작업 실패 추적 섹션을 참조하세요.	`Failed`는 최종 상태입니다. 작업이 `Failed` 상태에 도달하면 다른 어떤 상태로도 전환하지 않습니다.

작업 상태 업데이트

다음 AWS CLI 및 AWS SDK for Java 예제는 Batch Operations 작업의 상태를 업데이트합니다. Amazon S3 콘솔을 사용하여 Batch Operations 작업을 관리하는 방법에 대한 자세한 내용은 Amazon S3 콘솔을 사용하여 S3 배치 작업 관리 섹션을 참조하세요.

AWS CLI 사용

다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

- create-job 명령에 --no-confirmation-required 파라미터를 지정하지 않았다면 작업은 사용자가 그 상태를 Ready로 설정하여 확인할 때까지 보류 상태로 유지됩니다. 그러면 Amazon S3이 작업을 실행 가능하도록 만듭니다.

```
aws s3control update-job-status \
    --region us-west-2 \
    --account-id 123456789012 \
    --job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c \
    --requested-job-status 'Ready'
```

- 작업 상태를 Cancelled로 설정하여 작업을 취소합니다.

```
aws s3control update-job-status \
     --region us-west-2 \
     --account-id 123456789012 \
     --job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c \
     --status-update-reason "No longer needed" \
     --requested-job-status Cancelled
```

Java용 AWS SDK 사용

다음 예제에서는 AWS SDK for Java를 사용하여 S3 Batch Operations 작업의 상태를 업데이트합니다.

작업 상태에 대한 자세한 내용은 작업 상태 및 완료 보고서 추적 섹션을 참조하세요.

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();
        }
    }
}
```

알림 및 로깅

AWS CloudTrail을 사용하여 완료 보고서를 요청할 뿐만 아니라 Batch Operations 활동을 캡처, 검토 및 감사할 수 있습니다. Batch Operations는 기존 Amazon S3 API 작업을 사용하여 태스크를 수행하기 때문에 이러한 태스크 역시 직접 호출한 경우와 동일한 이벤트를 방출합니다. 따라서 이미 Amazon S3에서 사용하는 것과 동일한 알림, 로깅 및 감사 도구와 프로세스를 사용하여 작업 진행률 및 모든 태스크를 추적하고 기록할 수 있습니다. 자세한 내용은 다음 섹션의 예제를 참조하세요.

> **Note**
>
> Batch Operations는 작업 실행 중에 CloudTrail에서 관리 이벤트와 데이터 이벤트를 둘 다 생성합니다. 이러한 이벤트의 볼륨은 각 작업의 매니페스트에 있는 키의 수에 따라 규모가 조정

됩니다. 자세한 내용은 계정에 구성한 추적 수에 따라 달라지는 요금 변경의 예제가 포함된 CloudTrail 요금 페이지를 참조하세요. 필요에 따라 이벤트를 구성하고 기록하는 방법을 알아보려면 AWS CloudTrail 사용 설명서의 첫 번째 추적 생성을 참조하세요.

Amazon S3 이벤트에 대한 자세한 정보는 Amazon S3 이벤트 알림 섹션을 참조하세요.

작업 실패 추적

S3 배치 작업 건이 지정된 매니페스트를 읽을 수 없는 경우처럼 성공적으로 실행되지 못하게 하는 문제가 발생하면 해당 작업이 실패합니다. 작업이 실패하면 하나 이상의 실패 코드 또는 실패 이유를 생성합니다. S3 Batch Operations는 작업의 세부 정보를 요청하여 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 작업에 태스크(task) 실패 임계값을 부여합니다. 하나의 작업이 최소 1,000개의 태스크를 실행하면 Amazon S3가 태스크 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**
>
> S3 Batch Operations는 비동기식으로 작동하며, 반드시 매니페스트에 객체가 나열된 순서대로 태스크를 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.

완료 보고서

작업을 생성할 때 완료 보고서를 요청할 수 있습니다. S3 Batch Operations가 하나 이상의 태스크를 성공적으로 간접 호출하는 한, Amazon S3가 작업이 태스크 실행을 마치거나 실패하거나 취소된 후 완료 보고서를 생성합니다. 모든 작업 또는 실패한 작업만 포함하도록 완료 보고서를 구성할 수 있습니다.

완료 보고서에는 작업 구성, 상태, 그리고 객체 키와 버전, 상태, 오류 코드 및 오류 설명 등 각 태스크의 정보가 포함됩니다. 완료 보고서를 사용하면 추가 설정을 하지 않아도 통합된 형식으로 작업 결과를 손

쉽게 볼 수 있습니다. 완료 보고서는 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)를 통해 암호화됩니다. 완료 보고서의 예는 예: S3 배치 작업 완료 보고서 섹션을 참조하세요.

완료 보고서를 구성하지 않은 경우에도 여전히 CloudTrail 및 Amazon CloudWatch를 사용하여 작업 및 그 태스크를 모니터링하고 감사할 수 있습니다. 자세한 정보는 다음의 주제를 참조하세요.

주제

- 예: AWS CloudTrail을(를) 통해 Amazon EventBridge에서 S3 배치 작업 추적
- 예: S3 배치 작업 완료 보고서

예: AWS CloudTrail을(를) 통해 Amazon EventBridge에서 S3 배치 작업 추적

Amazon S3 배치 작업 활동은 AWS CloudTrail에 이벤트로 기록됩니다. Amazon EventBridge에서 사용자 정의 규칙을 생성하고 이러한 이벤트를 원하는 대상 알림 리소스(예: Amazon Simple Notification Service(Amazon SNS))로 보낼 수 있습니다.

> **Note**
>
> Amazon EventBridge는 이벤트를 관리하는 데 선호되는 방법입니다. Amazon CloudWatch Events와 EventBridge는 기본 서비스 및 API가 동일하지만 EventBridge가 더 많은 기능을 제공합니다. CloudWatch 또는 EventBridge에서 변경한 내용은 각 콘솔에 나타납니다. 자세한 내용은 Amazon EventBridge 사용 설명서를 참조하세요.

추적 예

- CloudTrail에 기록된 S3 배치 작업 이벤트
- S3 배치 작업 이벤트 추적을 위한 EventBridge 규칙

CloudTrail에 기록된 S3 배치 작업 이벤트

배치 작업이 생성되면 CloudTrail에 `JobCreated` 이벤트로 기록됩니다. 작업이 실행되면 처리 중에 상태가 변경되고 다른 `JobStatusChanged` 이벤트가 CloudTrail에 기록됩니다. CloudTrail 콘솔에서 이러한 이벤트를 볼 수 있습니다. CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

> **Note**
>
> S3 배치 작업 `status-change` 이벤트만 CloudTrail에 기록됩니다.

Example - CloudTrail에 의해 기록된 S3 배치 작업 완료 이벤트

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "accountId": "123456789012",
        "invokedBy": "s3.amazonaws.com"
    },
    "eventTime": "2020-02-05T18:25:30Z",
    "eventSource": "s3.amazonaws.com",
    "eventName": "JobStatusChanged",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "s3.amazonaws.com",
    "userAgent": "s3.amazonaws.com",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "f907577b-bf3d-4c53-b9ed-8a83a118a554",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "recipientAccountId": "123412341234",
    "serviceEventDetails": {
        "jobId": "d6e58ec4-897a-4b6d-975f-10d7f0fb63ce",
        "jobArn": "arn:aws:s3:us-west-2:181572960644:job/
d6e58ec4-897a-4b6d-975f-10d7f0fb63ce",
        "status": "Complete",
        "jobEventId": "b268784cf0a66749f1a05bce259804f5",
        "failureCodes": [],
        "statusChangeReason": []
    }
}
```

S3 배치 작업 이벤트 추적을 위한 EventBridge 규칙

다음 예제에서는 Amazon EventBridge에서 규칙을 생성하여 AWS CloudTrail에 의해 기록된 S3 배치 작업 이벤트를 선택한 대상으로 캡처하는 방법을 보여 줍니다.

이렇게 하려면 이벤트에 반응하는 EventBridge 규칙 생성의 모든 단계에 따라 규칙을 생성합니다. 해당하는 경우 다음 S3 배치 작업 사용자 지정 이벤트 패턴 정책을 붙여넣고 원하는 대상 서비스를 선택합니다.

S3 배치 작업 사용자 지정 이벤트 패턴 정책

```
{
    "source": [
        "aws.s3"
    ],
    "detail-type": [
        "AWS Service Event via CloudTrail"
    ],
    "detail": {
        "eventSource": [
            "s3.amazonaws.com"
        ],
        "eventName": [
            "JobCreated",
            "JobStatusChanged"
        ]
    }
}
```

다음 예제는 EventBridge 이벤트 규칙에서 Amazon Simple Queue Service(Amazon SQS)로 전송된 두 개의 배치 작업 이벤트입니다. 배치 작업은 처리 중에 여러 가지 상태(`New`, `Preparing`, `Active` 등)를 거치므로 각 작업에 대해 여러 메시지를 수신할 수 있습니다.

Example - JobCreated 샘플 이벤트

```
{
    "version": "0",
    "id": "51dc8145-541c-5518-2349-56d7dffdf2d8",
    "detail-type": "AWS Service Event via CloudTrail",
    "source": "aws.s3",
    "account": "123456789012",
    "time": "2020-02-27T15:25:49Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "eventVersion": "1.05",
        "userIdentity": {
```

```
            "accountId": "11112223334444",
            "invokedBy": "s3.amazonaws.com"
        },
        "eventTime": "2020-02-27T15:25:49Z",
        "eventSource": "s3.amazonaws.com",
        "eventName": "JobCreated",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "s3.amazonaws.com",
        "userAgent": "s3.amazonaws.com",
        "eventID": "7c38220f-f80b-4239-8b78-2ed867b7d3fa",
        "readOnly": false,
        "eventType": "AwsServiceEvent",
        "serviceEventDetails": {
            "jobId": "e849b567-5232-44be-9a0c-40988f14e80c",
            "jobArn": "arn:aws:s3:us-east-1:181572960644:job/
e849b567-5232-44be-9a0c-40988f14e80c",
            "status": "New",
            "jobEventId": "f177ff24f1f097b69768e327038f30ac",
            "failureCodes": [],
            "statusChangeReason": []
        }
    }
}
```

Example - JobStatusChanged 작업 완료 이벤트

```
{
  "version": "0",
  "id": "c8791abf-2af8-c754-0435-fd869ce25233",
  "detail-type": "AWS Service Event via CloudTrail",
  "source": "aws.s3",
  "account": "123456789012",
  "time": "2020-02-27T15:26:42Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "eventVersion": "1.05",
    "userIdentity": {
      "accountId": "1111222233334444",
      "invokedBy": "s3.amazonaws.com"
    },
    "eventTime": "2020-02-27T15:26:42Z",
    "eventSource": "s3.amazonaws.com",
```

```
    "eventName": "JobStatusChanged",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "s3.amazonaws.com",
    "userAgent": "s3.amazonaws.com",
    "eventID": "0238c1f7-c2b0-440b-8dbd-1ed5e5833afb",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "serviceEventDetails": {
      "jobId": "e849b567-5232-44be-9a0c-40988f14e80c",
      "jobArn": "arn:aws:s3:us-east-1:181572960644:job/
e849b567-5232-44be-9a0c-40988f14e80c",
      "status": "Complete",
      "jobEventId": "51f5ac17dba408301d56cd1b2c8d1e9e",
      "failureCodes": [],
      "statusChangeReason": []
    }
  }
}
```

예: S3 배치 작업 완료 보고서

S3 배치 작업을 생성할 때 모든 작업 또는 실패한 작업에 대한 완료 보고서를 요청할 수 있습니다. 하나 이상의 작업이 성공적으로 호출되면 S3 배치 작업은 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.

> **Note**
>
> 완료 보고서는 항상 Amazon S3 관리형 키(SSE-S3)를 사용하여 암호화됩니다.

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 태스크 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.

```
amzn-s3-demo-bucket1,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""}"
```

```
amzn-s3-demo-bucket1,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""}"
amzn-s3-demo-bucket1,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""}"
amzn-s3-demo-bucket1,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""}"
amzn-s3-demo-bucket1,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""}"
amzn-s3-demo-bucket1,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 태스크 보고서

성공한 작업 보고서에는 성공적인 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 S3 배치 작업으로 다시 전달된 다음 최종 완료 보고서에 기록됩니다.

```
amzn-s3-demo-bucket1,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'}}}"
amzn-s3-demo-bucket1,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'}}}"
amzn-s3-demo-bucket1,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'}}}"
```

태그를 사용하여 액세스 제어 및 작업 레이블 지정

태그를 추가하여 S3 배치 작업 건에 대한 레이블을 지정하고 액세스를 제어할 수 있습니다. 태그는 배치 작업 건을 담당하는 사용자를 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 만들 수 있으며 작업을 만든 후 작업에 태그를 추가할 수 있습니다. 각 태그는 작업을 생성하거나 나중에 업데이트할 때 포함할 수 있는 키-값 페어입니다.

> **Warning**
>
> 작업 태그에 기밀 정보나 개인 데이터가 포함되어 있지 않은지 확인하세요.

다음 태깅 예를 참조하세요. 재무 부서에서 배치 작업을 생성하려고 한다고 가정합니다. `Finance` 값이 할당된 `Department` 태그로 작업을 생성하는 경우 사용자가 `CreateJob`을 호출할 수 있도록 허용

하는 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 “권한 제어” 섹션을 참조하세요.

새 S3 배치 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 결제 및 비용 관리 사용 설명서의 사용자 정의 태그 제한을 참조하세요.

S3 배치 작업 건 태깅과 관련된 API 작업

Amazon S3는 S3 배치 작업 건 태깅을 위해 다음 API 작업을 지원합니다.

- GetJobTagging – Batch Operations 작업과 연결된 태그 집합을 반환합니다.
- PutJobTagging – 작업과 연결된 태그 집합을 대체합니다. 이 API 작업을 사용하는 S3 배치 작업 건 태그 관리에 대한 시나리오 두 가지가 있습니다.
  - 작업에 태그가 없는 경우 - 작업에 태그 집합을 추가할 수 있습니다(작업에 이전 태그가 없음).
  - 작업에 기존 태그 집합이 있는 경우 - 기존 태그 집합을 수정하려면 기존 태그 집합을 완전히 바꾸거나, GetJobTagging을 사용하여 기존 태그 집합을 검색한 다음 기존 태그 집합 내에서 변경하고, 해당 태그 집합을 수정하고, 이 API 작업을 사용하여 수정한 태그 집합으로 바꿉니다.

> **Note**
>
> 빈 태그 세트로 이 요청을 전송하는 경우 S3 배치 작업은 객체에 있는 기존 태그 세트를 삭제합니다. 이 방법을 사용하면 계층 1 요청(PUT)에 대한 요금이 부과됩니다. 자세한 내용은 Amazon S3 요금을 참조하세요.
>
> 배치 작업에 대한 기존 태그를 삭제하려면 비용이 발생하지 않고 동일한 결과를 얻을 수 있는 `DeleteJobTagging` 작업이 선호됩니다.

- DeleteJobTagging - Batch Operations 작업과 연결된 태그 집합을 삭제합니다.

레이블 지정에 작업 태그를 사용하는 배치 작업 생성

태그를 추가하여 Amazon S3 Batch Operations 작업에 대한 레이블을 지정하고 액세스를 제어할 수 있습니다. 태그는 배치 작업 건을 담당하는 사용자를 식별하는 데 사용할 수 있습니다. 태그가 연결된 작업을 생성할 수 있으며 작업을 생성한 후 작업에 태그를 추가할 수 있습니다. 자세한 내용은 the section called “태그 사용” 섹션을 참조하세요.

AWS CLI 사용

다음 AWS CLI 예제에서는 작업 태그를 작업의 레이블로 사용하여 S3 배치 작업 S3PutObjectCopy 작업을 생성합니다.

1. 배치 작업에서 수행하려는 작업 또는 OPERATION을 선택하고 TargetResource를 선택합니다.

```
read -d '' OPERATION <<EOF
{
  "S3PutObjectCopy": {
    "TargetResource": "arn:aws:s3:::amzn-s3-demo-destination-bucket"
  }
}
EOF
```

2. 작업에 대해 원하는 작업 TAGS를 식별합니다. 이 경우 두 개의 태그 department 및 FiscalYear를 각각 Marketing 및 2020 값과 함께 적용합니다.

```
read -d '' TAGS <<EOF
[
  {
    "Key": "department",
    "Value": "Marketing"
  },
  {
    "Key": "FiscalYear",
    "Value": "2020"
  }
]
EOF
```

3. 배치 작업에 대해 MANIFEST를 지정합니다.

```
read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "EXAMPLE_S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::amzn-s3-demo-manifest-bucket/example_manifest.csv",
    "ETag": "example-5dc7a8bfb90808fc5d546218"
  }
}
EOF
```

4. 배치 작업에 대해 REPORT를 구성합니다.

```
read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::amzn-s3-demo-completion-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 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";
```

Java용 AWS SDK 사용

Example

다음 예제에서는 AWS SDK for Java를 사용하여 태그가 있는 S3 배치 작업을 생성합니다.

```
public String createJob(final AWSS3ControlClient awss3ControlClient) {
    final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-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:::amzn-s3-demo-completion-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();
}
```

S3 배치 작업에서 태그 삭제

다음 예제를 사용하여 Amazon S3 Batch Operations 작업에서 태그를 삭제할 수 있습니다.

AWS CLI 사용

다음 예제에서는 AWS CLI를 사용하여 배치 작업에서 태그를 삭제합니다.

```
aws \
    s3control delete-job-tagging \
    --account-id 123456789012 \
    --job-id Example-e25a-4ed2-8bee-7f8ed7fc2f1c \
    --region us-east-1
```

배치 작업의 작업 태그 삭제

Example

다음 예제에서는 AWS SDK for Java를 사용하여 S3 배치 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);
}
```

기존 Batch Operations 작업에 작업 태그 추가

PutJobTagging API 작업을 사용하여 기존 Amazon S3 Batch Operations 작업에 작업 태그를 추가할 수 있습니다. 자세한 정보는 다음 예를 참조하세요.

AWS CLI 사용

다음은 AWS CLI에서 `s3control put-job-tagging`을 사용하여 S3 Batch Operations 작업에 작업 태그를 추가하는 예제입니다. 예제를 사용하려면 *`user input placeholders`*를 실제 정보로 대체하세요.

> **Note**
>
> 빈 태그 집합으로 이 요청을 전송하는 경우 Batch Operations는 객체에 있는 기존 태그 집합을 삭제합니다. 그러나 이 접근 방식을 사용하는 경우 계층 1 요청(PUT)에 대한 요금이 부과됩니다. 자세한 내용은 Amazon S3 요금을 참조하십시오.
> 대신 Batch Operations 작업에 대한 기존 태그를 삭제하려면 비용을 부담하지 않고 동일한 결과를 얻을 수 있으므로 `DeleteJobTagging` 작업을 사용하는 것이 좋습니다.

1. 작업에 대해 원하는 작업 TAGS를 식별합니다. 이 경우 두 개의 태그 *`department`* 및 *`FiscalYear`*를 각각 *`Marketing`* 및 *`2020`* 값과 함께 적용합니다.

```
read -d '' TAGS <<EOF
[
  {
    "Key": "department",
    "Value": "Marketing"
  },
  {
    "Key": "FiscalYear",
    "Value": "2020"
```

```
  }
]
EOF
```

2. 필요한 파라미터를 사용하여 다음 `put-job-tagging` 명령을 실행합니다.

```
aws \
    s3control put-job-tagging \
    --account-id 123456789012 \
    --tags "${TAGS//$'\n'/}" \
    --job-id Example-e25a-4ed2-8bee-7f8ed7fc2f1c \
    --region us-east-1
```

Java용 AWS SDK 사용

Example

다음 예제에서는 AWS SDK for Java를 사용하여 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);
}
```

S3 배치 작업의 작업 태그 가져오기

Amazon S3 Batch Operations 작업의 태그를 검색하려면 `GetJobTagging` API 작업을 사용하면 됩니다. 자세한 정보는 다음 예를 참조하세요.

AWS CLI 사용

다음 예제에서는 AWS CLI를 사용하여 배치 작업의 태그를 가져옵니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws \
    s3control get-job-tagging \
    --account-id 123456789012 \
    --job-id Example-e25a-4ed2-8bee-7f8ed7fc2f1c \
    --region us-east-1
```

Java용 AWS SDK 사용

Example

다음 예제에서는 AWS SDK for Java를 사용하여 S3 배치 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;
}
```

작업 태그를 사용하여 Batch Operations에 대한 권한 제어

Amazon S3 Batch Operations 작업 관리를 돕기 위해 작업 태그를 추가할 수 있습니다. 작업 태그를 사용하여 배치 작업에 대한 액세스를 제어하고 작업이 생성될 때 태그를 적용할 수 있습니다.

각 배치 작업에 최대 50개의 작업 태그를 적용할 수 있습니다. 태그를 통해 세분화된 정책을 설정하여 작업을 편집할 수 있는 사용자 집합을 제한할 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. AWS Identity and Access Management (IAM) 정책 언어를 사용하여 이러한 모든 조건을 표현할 수 있습니다. 자세한 정보는 서비스 승인 참조에서 Amazon S3에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

다음 예제에서는 S3 배치 작업 태그를 사용하여 특정 부서(예: 재무 또는 규정 준수 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 IAM 정책의 S3 Batch Operations 작업 태그를 사용하여 사용자에게 부서 내에서 실행 중인 작업만 만들고 편집할 수 있는 권한을 부여합니다. QA 또는 Production 등 관련된 개발의 단계에 따라 작업을 할당합니다.

다음 예제에서는 다음 부서를 사용하며 각 부서는 Batch Operations를 다양한 방식으로 활용합니다.

- 재무
- 규정 준수
- 비즈니스 인텔리전스
- 엔지니어링

주제

- 사용자 및 리소스에 태그를 할당하여 액세스 제어
- 단계별 배치 작업 태그 지정 및 작업 우선 순위 제한 적용

사용자 및 리소스에 태그를 할당하여 액세스 제어

이 시나리오에서는 관리자가 속성 기반 액세스 제어(ABAC)를 사용하고 있습니다. ABAC는 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 사용하여 재무 부서의 사용자에게 department=Finance 태그를 사용자와 연결하여 부서 내에서 S3 배치 작업을 생성하고 관리할 수 있는 권한을 부여합니다.

또한 회사의 모든 사용자가 해당 부서 내에서 S3 배치 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 작업을 생성할 수 있도록 허용합니다. 이는 정책 평가 시 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 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 태그 지정 및 작업 우선 순위 제한 적용

모든 S3 배치 작업에는 Amazon S3이 작업을 실행할 순서를 결정하는 데 사용하는 숫자 우선 순위가 있습니다. 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 S3 배치 작업 생성 및 업데이트에 대한 두 가지 새로운 제한을 적용합니다.

- 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 섹션의 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"
                    ]
                }
            }
        }
    ]
}
```

S3 배치 작업을 사용하여 S3 객체 잠금 관리

S3 배치 작업을 사용하여 Amazon S3 객체에 대해 대규모 배치 작업을 수행할 수 있습니다. S3 배치 작업은 지정된 Amazon S3 객체 목록에 대해 단일 작업을 수행할 수 있습니다. 단일 작업으로 엑사바

이트 규모의 데이터가 포함된 수십억 개의 객체에서 지정된 작업을 수행할 수 있습니다. Amazon S3는 진행 상황을 추적하고 알림을 보내며 모든 작업에 대한 자세한 완료 보고서를 저장하여 감사 가능한 완전관리형 서버리스 환경을 제공합니다. Amazon S3 콘솔, AWS CLI, AWS SDK 또는 Amazon S3 REST API를 통해 S3 Batch Operations를 사용할 수 있습니다.

S3 객체 잠금을 사용하면 객체 버전에 법적 보존을 적용할 수도 있습니다. 보관 기간 설정과 마찬가지로 법적 보존을 사용하면 객체 버전을 덮어쓰거나 삭제할 수 없습니다. 그러나 법적 보존에는 연결된 보존 기간이 없고, 법적 보존이 제거될 때까지 유효합니다. 자세한 내용은 S3 객체 잠금 법적 보존 섹션을 참조하세요.

Object Lock과 함께 S3 Batch Operations를 사용하여 많은 Amazon S3 객체에 법적 보존을 한 번에 추가하려면 다음 주제를 참조하세요.

주제

- S3 배치 작업을 사용하여 S3 객체 잠금 사용 설정
- 배치 작업을 사용한 객체 잠금 보존 설정
- S3 객체 잠금 보존 규정 준수 모드에서 S3 배치 작업 사용
- S3 객체 잠금 보존 거버넌스 모드에서 S3 배치 작업 사용
- S3 Batch Operations를 사용하여 S3 Object Lock 법적 보존 해제

S3 배치 작업을 사용하여 S3 객체 잠금 사용 설정

S3 Object Lock과 함께 Amazon S3 Batch Operations를 사용하여 보존을 관리하거나 여러 Amazon S3 객체에 대해 한 번에 법적 보존을 활성화할 수 있습니다. 매니페스트에서 대상 객체 목록을 지정하고 완료를 위해 배치 작업에 제출합니다. 자세한 내용은 the section called “객체 잠금 보존” 및 the section called “객체 잠금 법적 보존”(을)를 참조하세요.

다음 예제에서는 Object Lock을 활성화하는 작업을 만들기 위해 S3 Batch Operations 권한이 있는 AWS Identity and Access Management(IAM) 역할을 만들고 역할 권한을 업데이트하는 방법을 보여줍니다. S3 Batch Operations 작업에 대한 객체를 식별하는 CSV 매니페스트도 있어야 합니다. 자세한 내용은 the section called “매니페스트 지정” 섹션을 참조하세요.

다음 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

AWS CLI 사용

1. IAM 역할을 생성하고 실행할 S3 배치 작업 권한을 할당합니다.

   이 단계는 모든 S3 배치 작업에 필요합니다.

```
export AWS_PROFILE='aws-user'

read -d '' batch_operations_trust_policy <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "batchoperations.s3.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
aws iam create-role --role-name batch_operations-objectlock \
--assume-role-policy-document "${batch_operations_trust_policy}"
```

2. S3 객체 잠금을 사용하여 S3 배치 작업을 실행하도록 설정합니다.

   이 단계에서는 역할에서 다음 작업을 수행하도록 허용합니다.

   a. 배치 작업을 실행할 대상 객체가 포함된 S3 버킷에서 객체 잠금을 실행합니다.

   b. 매니페스트 CSV 파일과 객체가 있는 S3 버킷을 읽습니다.

   c. S3 배치 작업의 결과를 보고 버킷에 기록합니다.

```
read -d '' batch_operations_permissions <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetBucketObjectLockConfiguration",
            "Resource": [
                "arn:aws:s3:::{{amzn-s3-demo-manifest-bucket}}"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::{{amzn-s3-demo-manifest-bucket}}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::{{amzn-s3-demo-completion-report-bucket}}/*"
            ]
        }
    ]
}
EOF

aws iam put-role-policy --role-name batch_operations-objectlock \
--policy-name object-lock-permissions \
--policy-document "${batch_operations_permissions}"
```

Java용 AWS SDK 사용

다음 예제에서는 AWS SDK for Java를 사용하여 Object Lock을 활성화하는 작업을 만들기 위해 S3 Batch Operations 권한이 있는 IAM 역할을 만들고 역할 권한을 업데이트하는 방법을 보여줍니다. S3 배치 작업에 대한 객체를 식별하는 CSV 매니페스트도 있어야 합니다. 자세한 내용은 the section called “매니페스트 지정” 섹션을 참조하세요.

다음 단계를 수행합니다.

1. IAM 역할을 생성하고 실행할 S3 배치 작업 권한을 할당합니다. 이 단계는 모든 S3 배치 작업에 필요합니다.
2. S3 객체 잠금을 사용하여 S3 배치 작업을 실행하도록 설정합니다.

   역할에서 다음 작업을 수행하도록 허용합니다.

1. 배치 작업을 실행할 대상 객체가 포함된 S3 버킷에서 객체 잠금을 실행합니다.
2. 매니페스트 CSV 파일과 객체가 있는 S3 버킷을 읽습니다.
3. S3 배치 작업의 결과를 보고 버킷에 기록합니다.

```
public void createObjectLockRole() {
    final String roleName = "batch_operations-object-lock";

    final String trustPolicy = "{" +
            "  \"Version\": \"2012-10-17\", " +
            "  \"Statement\": [ " +
            "    { " +
            "      \"Effect\": \"Allow\", " +
            "      \"Principal\": { " +
            "        \"Service\": [" +
            "          \"batchoperations.s3.amazonaws.com\"" +
            "        ]" +
            "      }, " +
            "      \"Action\": \"sts:AssumeRole\" " +
            "    } " +
            "  ]" +
            "}";

    final String bopsPermissions = "{" +
            "    \"Version\": \"2012-10-17\"," +
            "    \"Statement\": [" +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": \"s3:GetBucketObjectLockConfiguration\"," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::amzn-s3-demo-manifest-bucket\"" +
            "            ]" +
            "        }," +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:GetObject\"," +
            "                \"s3:GetObjectVersion\"," +
            "                \"s3:GetBucketLocation\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::amzn-s3-demo-manifest-bucket/*\"" +
```

```
            "            ]" +
            "        }," +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:PutObject\"," +
            "                \"s3:GetBucketLocation\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*\""
 +
            "            ]" +
            "        }" +
            "    ]" +
            "}";

    final AmazonIdentityManagement iam =
            AmazonIdentityManagementClientBuilder.defaultClient();

    final CreateRoleRequest createRoleRequest = new CreateRoleRequest()
            .withAssumeRolePolicyDocument(bopsPermissions)
            .withRoleName(roleName);

    final CreateRoleResult createRoleResult = iam.createRole(createRoleRequest);

    final PutRolePolicyRequest putRolePolicyRequest = new PutRolePolicyRequest()
            .withPolicyDocument(bopsPermissions)
            .withPolicyName("batch_operations-permissions")
            .withRoleName(roleName);

    final PutRolePolicyResult putRolePolicyResult =
 iam.putRolePolicy(putRolePolicyRequest);
}
```

배치 작업을 사용한 객체 잠금 보존 설정

S3 Object Lock과 함께 Amazon S3 Batch Operations를 사용하여 여러 Amazon S3 객체에 대한 보존을 한 번에 관리할 수 있습니다. 매니페스트에서 대상 객체 목록을 지정하고 완료를 위해 배치 작업에 제출합니다. 자세한 내용은 the section called “객체 잠금 보존” 및 the section called “객체 잠금 법적 보존”(을)를 참조하세요.

다음 예제에서는 매니페스트 버킷에서 객체에 대한 S3 Object Lock 보존을 실행할 수 있도록 S3 Batch Operations 권한이 있는 AWS Identity and Access Management(IAM) 역할을 만들고

`s3:PutObjectRetention` 권한이 포함되도록 역할 권한을 업데이트하는 방법을 보여줍니다. S3 Batch Operations 작업에 대한 객체를 식별하는 CSV 매니페스트도 있어야 합니다. 자세한 내용은 the section called “매니페스트 지정” 섹션을 참조하세요.

다음 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

AWS CLI 사용

다음 AWS CLI 예제에서는 Batch Operations를 사용하여 여러 객체에 S3 Object Lock 보존을 적용하는 방법을 보여줍니다.

```
export AWS_PROFILE='aws-user'

read -d '' retention_permissions <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObjectRetention"
            ],
            "Resource": [
                "arn:aws:s3:::{{amzn-s3-demo-manifest-bucket}}/*"
            ]
        }
    ]
}
EOF

aws iam put-role-policy --role-name batch_operations-objectlock --policy-
name retention-permissions --policy-document "${retention_permissions}"
```

Java용 AWS SDK 사용

다음 AWS SDK for Java 예제에서는 Batch Operations를 사용하여 여러 객체에 S3 Object Lock 보존을 적용하는 방법을 보여줍니다.

```
public void allowPutObjectRetention() {
    final String roleName = "batch_operations-object-lock";

    final String retentionPermissions = "{" +
            "    \"Version\": \"2012-10-17\"," +
```

```
            "    \"Statement\": [" +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:PutObjectRetention\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::amzn-s3-demo-manifest-bucket*\"" +
            "            ]" +
            "        }" +
            "    ]" +
            "}";

    final AmazonIdentityManagement iam =
            AmazonIdentityManagementClientBuilder.defaultClient();

    final PutRolePolicyRequest putRolePolicyRequest = new PutRolePolicyRequest()
            .withPolicyDocument(retentionPermissions)
            .withPolicyName("retention-permissions")
            .withRoleName(roleName);

    final PutRolePolicyResult putRolePolicyResult =
 iam.putRolePolicy(putRolePolicyRequest);
}
```

S3 객체 잠금 보존 규정 준수 모드에서 S3 배치 작업 사용

다음 예제는 신뢰 정책을 생성하고 객체에 대해 S3 배치 작업 및 S3 객체 잠금 구성 권한을 설정하는 이전 예제를 기반으로 합니다. 다음 예시는 보존 모드를 COMPLIANCE로 설정하고 `retain until date`를 2025년 1월 1일로 설정합니다. 이 예제에서는 매니페스트 버킷의 객체를 대상으로 하는 작업을 만들고 사용자가 식별한 보고서 버킷에 결과를 보고합니다.

다음 예제를 사용하려면 *`user input placeholders`*를 사용자의 정보로 대체합니다.

AWS CLI 사용

다음 AWS CLI 예제에서는 Batch Operations를 사용하여 여러 객체에 S3 Object Lock 보존 규정 준수 모드를 적용하는 방법을 보여줍니다.

Example - 여러 객체에 대해 S3 Object Lock 보존 규정 준수 모드 설정

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
```

```
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/batch_operations-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "Retention": {
      "RetainUntilDate":"2025-01-01T00:00:00",
      "Mode":"COMPLIANCE"
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::amzn-s3-demo-manifest-bucket/compliance-objects-
manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::ReportBucket",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "amzn-s3-demo-completion-report-bucket/compliance-objects-batch-
operations",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
```

```
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Set compliance retain-until to 1 Jul 2030";
```

Example - **COMPLIANCE** 모드의 **retain until date**를 2025년 1월 15일로 연장

다음 예제에서는 COMPLIANCE 모드의 retain until date을(를) 2025년 1월 15일로 연장합니다.

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/batch_operations-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "Retention": {
      "RetainUntilDate":"2025-01-15T00:00:00",
      "Mode":"COMPLIANCE"
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::amzn-s3-demo-manifest-bucket/compliance-objects-
manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
```

```
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::amzn-s3-demo-completion-report-bucket",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/compliance-objects-batch_operations",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Extend compliance retention to 15 Jan 2025";
```

Java용 AWS SDK 사용

다음 AWS SDK for Java 예제에서는 Batch Operations를 사용하여 여러 객체에 S3 Object Lock 보존 규정 준수 모드를 적용하는 방법을 보여줍니다.

Example - 보존 모드를 COMPLIANCE로 설정하고 보존 종료 날짜를 2025년 1월 1일로 설정

```
public String createComplianceRetentionJob(final AWSS3ControlClient awss3ControlClient)
 throws ParseException {
    final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-bucket/
compliance-objects-manifest.csv";
    final String manifestObjectVersionId = "your-object-version-Id";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
```

```
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::amzn-s3-demo-completion-report-
bucket";
    final String jobReportPrefix = "reports/compliance-objects-bops";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
    final Date janFirst = format.parse("01/01/2025");

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectRetention(new S3SetObjectRetentionOperation()
                    .withRetention(new S3Retention()
                            .withMode(S3ObjectLockRetentionMode.COMPLIANCE)
                            .withRetainUntilDate(janFirst)));

    final String roleArn = "arn:aws:iam::123456789012:role/batch_operations-object-
lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Set compliance retain-until to 1 Jan 2025")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);
```

```
    return result.getJobId();
}
```

Example - **COMPLIANCE** 모드의 **retain until date** 연장

다음 예제에서는 COMPLIANCE 모드의 `retain until date`를 2025년 1월 15일로 연장합니다.

```
public String createExtendComplianceRetentionJob(final AWSS3ControlClient
 awss3ControlClient) throws ParseException {
    final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-bucket/
compliance-objects-manifest.csv";
    final String manifestObjectVersionId = "15ad5ba069e6bbc465c77bf83d541385";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::amzn-s3-demo-completion-report-
bucket";
    final String jobReportPrefix = "reports/compliance-objects-batch_operations";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
    final Date jan15th = format.parse("15/01/2025");

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectRetention(new S3SetObjectRetentionOperation()
                    .withRetention(new S3Retention()
                            .withMode(S3ObjectLockRetentionMode.COMPLIANCE)
```

```
                            .withRetainUntilDate(jan15th)));

    final String roleArn = "arn:aws:iam::123456789012:role/batch_operations-object-
lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Extend compliance retention to 15 Jan 2025")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

S3 객체 잠금 보존 거버넌스 모드에서 S3 배치 작업 사용

다음 예제는 신뢰 정책을 생성하고 S3 배치 작업 및 S3 객체 잠금 구성 권한을 설정하는 이전 예제를 기반으로 합니다. 이 예제에서는 여러 객체에 대해 `retain until date`가 2025년 1월 30일인 S3 Object Lock 보존 거버넌스를 적용하는 방법을 보여줍니다. 매니페스트 버킷을 사용하는 배치 작업을 생성하고 보고서 버킷에 결과를 보고합니다.

다음 예제를 사용하려면 *`user input placeholders`*를 사용자의 정보로 대체합니다.

AWS CLI 사용

다음 AWS CLI 예제에서는 Batch Operations를 사용하여 여러 객체에 S3 Object Lock 보존 거버넌스 모드를 적용하는 방법을 보여줍니다.

Example - 여러 객체에 보존 종료 날짜가 2025년 1월 30일인 S3 Object Lock 보존 거버넌스 적용

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/batch_operations-objectlock'
```

```
read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "Retention": {
      "RetainUntilDate":"2025-01-30T00:00:00",
      "Mode":"GOVERNANCE"
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::amzn-s3-demo-manifest-bucket/governance-objects-
manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::amzn-s3-demo-completion-report-bucketT",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/governance-objects",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
```

```
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Put governance retention";
```

Example - 여러 객체에 대한 보존 거버넌스 무시

다음 예제는 신뢰 정책을 생성하고 S3 배치 작업 및 S3 객체 잠금 구성 권한을 설정하는 이전 예제를 기반으로 합니다. 여러 객체에 대해 보존 거버넌스를 무시한 다음, 매니페스트 버킷을 사용하는 배치 작업을 생성하고 보고서 버킷에 결과를 보고하는 방법을 보여줍니다.

```
export AWS_PROFILE='aws-user'

read -d '' bypass_governance_permissions <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:BypassGovernanceRetention"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
            ]
        }
    ]
}
EOF

aws iam put-role-policy --role-name batch-operations-objectlock --policy-name bypass-
governance-permissions --policy-document "${bypass_governance_permissions}"

export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/batch_operations-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "BypassGovernanceRetention": true,
```

```
    "Retention": {
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::amzn-s3-demo-manifest-bucket/governance-objects-
manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::amzn-s3-demo-completion-report-bucket",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/batch_operations-governance",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
```

```
    --description "Remove governance retention";
```

Java용 AWS SDK 사용

다음 예제는 신뢰 정책을 생성하고 S3 배치 작업 및 S3 객체 잠금 구성 권한을 설정하는 이전 예제를 기반으로 합니다. 이 예제에서는 여러 객체에 대해 `retain until date`가 2025년 1월 30일로 설정된 S3 Object Lock 보존 거버넌스를 적용하는 방법을 보여줍니다. 이 예제에서는 매니페스트 버킷을 사용하는 Batch Operations 작업을 만들고 보고서 버킷에 결과를 보고합니다.

Example - 여러 객체에 보존 종료 날짜가 2025년 1월 30일인 S3 Object Lock 보존 거버넌스 적용

```
public String createGovernanceRetentionJob(final AWSS3ControlClient awss3ControlClient)
 throws ParseException {
    final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-bucket/
governance-objects-manifest.csv";
    final String manifestObjectVersionId = "15ad5ba069e6bbc465c77bf83d541385";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::amzn-s3-demo-completion-report-
bucket";
    final String jobReportPrefix = "reports/governance-objects";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
    final Date jan30th = format.parse("30/01/2025");
```

```
    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectRetention(new S3SetObjectRetentionOperation()
                    .withRetention(new S3Retention()
                            .withMode(S3ObjectLockRetentionMode.GOVERNANCE)
                            .withRetainUntilDate(jan30th)));

    final String roleArn = "arn:aws:iam::123456789012:role/batch_operations-object-
lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Put governance retention")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

Example - 여러 객체에 대한 보존 거버넌스 무시

다음 예제는 신뢰 정책을 생성하고 S3 배치 작업 및 S3 객체 잠금 구성 권한을 설정하는 이전 예제를 기반으로 합니다. 이 예제에서는 여러 객체에 대해 보존 거버넌스를 무시한 다음, 매니페스트 버킷을 사용하는 Batch Operations 작업을 만들고 보고서 버킷에 결과를 보고하는 방법을 보여줍니다.

```
public void allowBypassGovernance() {
    final String roleName = "batch_operations-object-lock";

    final String bypassGovernancePermissions = "{" +
            "    \"Version\": \"2012-10-17\"," +
            "    \"Statement\": [" +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:BypassGovernanceRetention\"" +
```

```
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::amzn-s3-demo-manifest-bucket/*\"" +
            "            ]" +
            "        }" +
            "    ]" +
            "}";

    final AmazonIdentityManagement iam =
            AmazonIdentityManagementClientBuilder.defaultClient();

    final PutRolePolicyRequest putRolePolicyRequest = new PutRolePolicyRequest()
            .withPolicyDocument(bypassGovernancePermissions)
            .withPolicyName("bypass-governance-permissions")
            .withRoleName(roleName);

    final PutRolePolicyResult putRolePolicyResult =
 iam.putRolePolicy(putRolePolicyRequest);
}
public String createRemoveGovernanceRetentionJob(final AWSS3ControlClient
 awss3ControlClient) {
    final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-bucket/
governance-objects-manifest.csv";
    final String manifestObjectVersionId = "15ad5ba069e6bbc465c77bf83d541385";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::amzn-s3-demo-completion-report-
bucket";
    final String jobReportPrefix = "reports/batch_operations-governance";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
```

```
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectRetention(new S3SetObjectRetentionOperation()
                    .withRetention(new S3Retention()));

    final String roleArn = "arn:aws:iam::123456789012:role/batch_operations-object-
lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Remove governance retention")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

S3 Batch Operations를 사용하여 S3 Object Lock 법적 보존 해제

다음 예제는 신뢰 정책을 생성하고 S3 배치 작업 및 S3 객체 잠금 구성 권한을 설정하는 이전 예제를 기반으로 합니다. 이 예제에서는 Batch Operations를 사용하여 객체에 대한 Object Lock 법적 보존을 비활성화하는 방법을 보여줍니다.

이 예제에서는 먼저 역할을 업데이트하여 `s3:PutObjectLegalHold` 권한을 부여하고, 매니페스트에서 식별된 객체의 법적 보존을 해제(제거)하는 배치 작업을 생성한 다음 보고합니다.

다음 예제를 사용하려면 *`user input placeholders`*를 사용자의 정보로 대체합니다.

AWS CLI 사용

다음 AWS CLI 예제에서는 Batch Operations를 사용하여 여러 객체에서 S3 Object Lock의 법적 보존을 해제하는 방법을 보여줍니다.

Example - **s3:PutObjectLegalHold** 권한을 부여하도록 역할 업데이트

```
export AWS_PROFILE='aws-user'

read -d '' legal_hold_permissions <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObjectLegalHold"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
            ]
        }
    ]

EOF

aws iam put-role-policy --role-name batch_operations-objectlock --policy-name legal-hold-permissions --policy-document "${legal_hold_permissions}"
```

Example - 법적 보존 해제

다음 예제에서는 법적 보존을 해제합니다.

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/batch_operations-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectLegalHold": {
    "LegalHold": {
      "Status":"OFF"
```

```
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::amzn-s3-demo-manifest-bucket/legalhold-object-
manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::amzn-s3-demo-completion-report-bucket",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/legalhold-objects-batch_operations",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Turn off legal hold";
```

Java용 AWS SDK 사용

다음 AWS SDK for Java 예제에서는 Batch Operations를 사용하여 여러 객체에서 S3 Object Lock의 법적 보존을 해제하는 방법을 보여줍니다.

Example - **s3:PutObjectLegalHold** 권한을 부여하도록 역할 업데이트

```
public void allowPutObjectLegalHold() {
    final String roleName = "batch_operations-object-lock";

    final String legalHoldPermissions = "{" +
            "    \"Version\": \"2012-10-17\"," +
            "    \"Statement\": [" +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:PutObjectLegalHold\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::amzn-s3-demo-manifest-bucket/*\"" +
            "            ]" +
            "        }" +
            "    ]" +
            "}";

    final AmazonIdentityManagement iam =
            AmazonIdentityManagementClientBuilder.defaultClient();

    final PutRolePolicyRequest putRolePolicyRequest = new PutRolePolicyRequest()
            .withPolicyDocument(legalHoldPermissions)
            .withPolicyName("legal-hold-permissions")
            .withRoleName(roleName);

    final PutRolePolicyResult putRolePolicyResult =
 iam.putRolePolicy(putRolePolicyRequest);
}
```

Example - 법적 보존 해제

법적 보존을 해제하려면 아래 예제를 사용하세요.

```
public String createLegalHoldOffJob(final AWSS3ControlClient awss3ControlClient) {
```

```
    final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-bucket/
legalhold-object-manifest.csv";
    final String manifestObjectVersionId = "15ad5ba069e6bbc465c77bf83d541385";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::amzn-s3-demo-completion-report-
bucket";
    final String jobReportPrefix = "reports/legalhold-objects-batch_operations";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectLegalHold(new S3SetObjectLegalHoldOperation()
                    .withLegalHold(new S3ObjectLockLegalHold()
                            .withStatus(S3ObjectLockLegalHoldStatus.OFF)));

    final String roleArn = "arn:aws:iam::123456789012:role/batch_operations-object-
lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Turn off legal hold")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
```

```
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

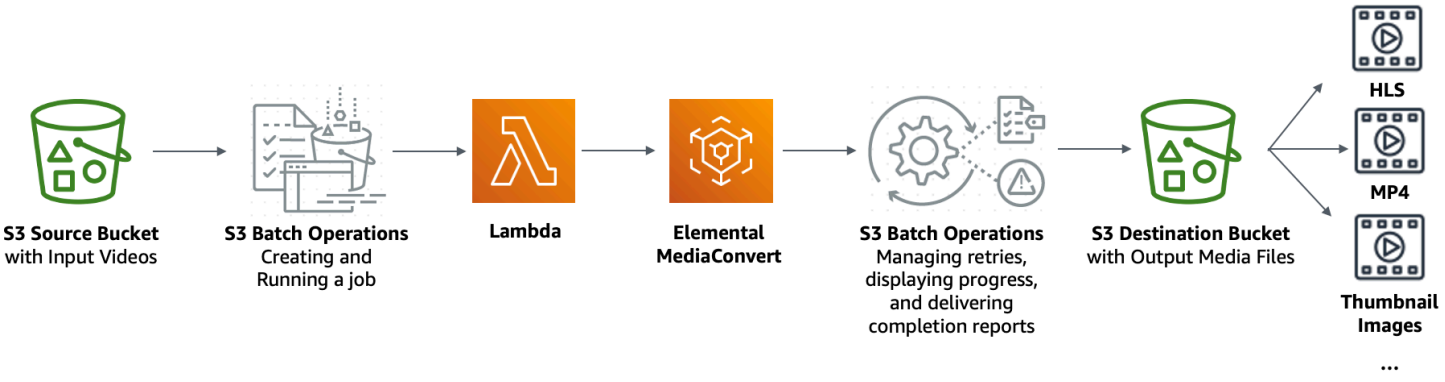
자습서: S3 Batch Operations를 통해 비디오 일괄 트랜스코딩

비디오 소비자는 모든 모양, 크기 및 빈티지의 디바이스를 사용하여 미디어 콘텐츠를 즐길 수 있습니다. 이 다수의 디바이스는 콘텐츠 제작자 및 배포자에게 어려운 과제를 제기합니다. 비디오는 하나의 크기에 맞는 형식이 아닌 다양한 크기, 형식 및 비트레이트로 변환되어야 합니다. 이 변환 작업은 변환해야 하는 비디오가 많을 때 더욱 까다롭습니다.

AWS에서는 다음을 수행하는 확장 가능한 분산 아키텍처를 빌드하는 방법을 제공합니다.

- 입력 비디오 수집
- 다양한 장치에서 재생할 수 있도록 비디오를 처리
- 트랜스코딩된 미디어 파일 저장
- 수요를 충족하기 위해 출력 미디어 파일 제공

Amazon S3에 저장된 광범위한 비디오 리포지토리가 있는 경우, 이러한 비디오를 소스 형식에서 (특정 비디오 플레이어 또는 디바이스에 필요한 크기, 해상도 및 형식의) 여러 파일 형식으로 트랜스코딩할 수 있습니다. 특히, S3 배치 작업은 S3 소스 버킷의 기존 입력 비디오에 대한 AWS Lambda 함수를 호출할 수 있는 솔루션을 제공합니다. 그런 다음 Lambda 함수는 AWS Elemental MediaConvert를 호출하여 대규모 비디오 트랜스코딩 작업을 수행할 수 있습니다. 변환된 출력 미디어 파일은 S3 대상 버킷에 저장됩니다.



목표

이 자습서에서는 S3 소스 버킷에 저장된 비디오의 일괄 트랜스코딩을 위해 Lambda 함수를 호출하도록 S3 배치 작업을 설정하는 방법에 대해 알아봅니다. Lambda 함수는 MediaConvert를 호출하여 비디오를 트랜스코딩합니다. S3 소스 버킷의 각 비디오에 대한 출력은 다음과 같습니다.

- 다양한 크기의 디바이스 및 다양한 대역폭에서 재생할 수 있는 적응형 비트레이트 스트림인 HLS(HTTP Live Streaming)
- MP4 비디오 파일
- 간격에 따라 수집된 썸네일 이미지

주제

- 사전 조건
- 1단계: 출력 미디어 파일용 S3 버킷 생성
- 2단계: MediaConvert용 IAM 역할 생성
- 3단계: Lambda 함수용 IAM 역할 생성
- 4단계: 비디오 트랜스코딩용 Lambda 함수 생성
- 5단계: S3 소스 버킷용 Amazon S3 인벤토리 구성
- 6단계: S3 배치 작업용 IAM 역할 생성
- 7단계: S3 배치 작업 생성 및 실행
- 8단계: S3 대상 버킷의 출력 미디어 파일 확인
- 9단계: 정리
- 다음 단계

사전 조건

이 자습서를 시작하려면 트랜스코딩할 비디오가 이미 저장된 Amazon S3 소스 버킷(예: *amzn-s3-demo-source-bucket*)이 있어야 합니다.

원하는 경우 버킷에 다른 이름을 지정할 수 있습니다. Amazon S3 버킷 이름 지정 규칙에 대한 자세한 내용은 범용 버킷 이름 지정 규칙 섹션을 참조하십시오.

S3 소스 버킷의 경우 이 버킷에 대한 퍼블릭 액세스 차단 설정과 관련된 설정을 기본값으로 유지합니다(모든 퍼블릭 액세스 차단이 사용 설정됨). 자세한 내용은 범용 버킷 생성 섹션을 참조하세요.

S3 소스 버킷에 비디오를 업로드하는 방법에 대한 자세한 내용은 객체 업로드 단원을 참조하십시오. S3에 많은 대용량 비디오파일을 업로드하는 경우 Amazon S3 Transfer Acceleration을 사용하여 빠르고 안전한 파일 전송을 구성할 수 있습니다. Transfer Acceleration을 사용하면 S3 버킷에 비디오를 빠르게 업로드하여 대용량의 비디오를 장거리 전송할 수 있습니다. 자세한 내용은 Amazon S3 Transfer Acceleration을 사용하여 빠르고 안전한 파일 전송 구성 섹션을 참조하세요.

1단계: 출력 미디어 파일용 S3 버킷 생성

이 단계에서는 변환된 출력 미디어 파일을 저장할 S3 대상 버킷을 생성합니다. 또한 교차 오리진 리소스 공유(CORS) 구성을 생성하여 S3 대상 버킷에 저장된 트랜스코딩된 미디어 파일에 대한 교차 오리진 액세스를 허용합니다.

하위 단계

- 출력 미디어 파일용 버킷 생성
- S3 출력 버킷에 CORS 구성 추가

출력 미디어 파일용 버킷 생성

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 생성을 선택합니다.
4. 버킷 이름에 버킷 이름을 입력합니다(예: *amzn-s3-demo-destination-bucket1*).
5. 리전(Region)에서 버킷이 속할 AWS 리전을 선택합니다.
6. 출력 미디어 파일에 대한 퍼블릭 액세스를 보장하려면 이 버킷에 대한 퍼블릭 액세스 차단 설정(Block Public Access settings for this bucket)에서 모든 퍼블릭 액세스 차단(Block all public access)을 선택 취소합니다.

> **Warning**
>
> 이 단계를 완료하기 전에 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단을 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.
> 퍼블릭 액세스 차단 설정을 지우지 않으려면 Amazon CloudFront를 사용하여 트랜스코딩된 미디어 파일을 뷰어(최종 사용자)에게 전달할 수 있습니다. 자세한 내용은 자습서:

> Amazon S3, Amazon CloudFront 및 Amazon Route 53로 온디맨드 스트리밍 비디오 호스팅 섹션을 참조하세요.

7. 현재 설정으로 인해 이 버킷과 그 안에 포함된 객체가 퍼블릭 상태가 될 수 있음을 알고 있습니다(I acknowledge that the current settings might result in this bucket and the objects within becoming public) 옆의 확인란을 선택합니다.
8. 나머지 설정은 기본값으로 유지합니다.
9. 버킷 생성을 선택합니다.

S3 출력 버킷에 CORS 구성 추가

JSON CORS 구성은 하나의 도메인에서 로드되는 클라이언트 웹 애플리케이션(여기서는 비디오 플레이어)이 다른 도메인에 있는 트랜스코딩된 출력 미디어 파일을 재생하는 방법을 정의합니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 목록에서 이전에 생성한 버킷의 이름을 선택합니다(예: *amzn-s3-demo-destination-bucket1*).
4. 권한 탭을 선택합니다.
5. CORS(Cross-Origin 리소스 공유) 섹션에서 편집을 선택합니다.
6. CORS 구성 텍스트 상자에서 다음의 CORS 구성을 복사하여 붙여 넣습니다.

   CORS 구성은 JSON 형식이어야 합니다. 이 예제에서 `AllowedOrigins` 속성은 와일드카드 문자(*)를 사용하여 모든 오리진을 지정합니다. 특정 오리진을 알고 있는 경우 `AllowedOrigins` 속성을 특정 플레이어 URL로 제한할 수 있습니다. 이 속성과 다른 속성의 구성에 대한 자세한 내용은 CORS 구성의 요소을(를) 참조하십시오.

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

```
            "*"
        ],
        "ExposeHeaders": []

    }
]
```

7. 변경 사항 저장을 선택합니다.

2단계: MediaConvert용 IAM 역할 생성

AWS Elemental MediaConvert를 사용하여 S3 버킷에 저장된 입력 비디오를 트랜스코딩하려면 S3 소스 및 대상 버킷에서/으로 파일을 읽고 쓸 수 있는 MediaConvert 권한을 부여하는 AWS Identity and Access Management(IAM) 서비스 역할이 있어야 합니다. 트랜스코딩 작업을 실행하면 MediaConvert 콘솔에서 이 역할을 사용합니다.

MediaConvert용 IAM 역할 생성

1. 선택한 역할 이름을 사용하여 IAM 역할을 생성합니다(예: **`tutorial-mediaconvert-role`**). 이 역할을 생성하려면 AWS Elemental MediaConvert 사용 설명서의 IAM(콘솔)에서 MediaConvert 역할 생성에 있는 절차를 따릅니다.
2. MediaConvert에 대한 IAM 역할을 생성한 후 역할(Roles) 목록에서 생성한 MediaConvert에 대한 역할 이름을 선택합니다(예: **`tutorial-mediaconvert-role`**).
3. 요약 페이지에서, `arn:aws:iam::`으로 시작되는 역할 ARN을 복사하여 나중에 사용할 수 있도록 해당 ARN을 저장합니다.

   ARN에 대한 자세한 내용은 AWS 일반 참조의 Amazon 리소스 이름(ARN)을 참조하십시오.

3단계: Lambda 함수용 IAM 역할 생성

MediaConvert 및 S3 Batch Operations를 사용하여 비디오를 일괄 트랜스코딩하려면 비디오를 변환하기 위해 이러한 두 서비스를 연결하는 Lambda 함수가 있어야 합니다. 이 Lambda 함수에는 MediaConvert 및 S3 Batch Operations에 액세스할 수 있는 Lambda 함수 권한을 부여하는 IAM 역할이 있어야 합니다.

하위 단계

- Lambda 함수용 IAM 역할 생성
- Lambda 함수의 IAM 역할에 대한 인라인 정책 포함

Lambda 함수용 IAM 역할 생성

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 역할을 선택한 다음, 역할 생성을 선택합니다.
3. AWS 서비스 역할 유형을 선택한 후 일반적인 사용 사례에서 Lambda를 선택합니다.
4. 다음: 권한을 선택합니다.
5. 권한 정책 연결 페이지에서 필터 정책에 **AWSLambdaBasicExecutionRole**을(를) 입력합니다. 관리형 정책 AWSLambdaBasicExecutionRole을 이 역할에 연결하여 Amazon CloudWatch Logs에 쓰기 권한을 부여하려면, AWSLambdaBasicExecutionRole 옆의 확인란을 선택합니다.
6. 다음을 선택합니다.
7. 역할 이름에 **tutorial-lambda-transcode-role**을 입력합니다.
8. (선택 사항) 관리형 정책에 태그를 추가합니다.
9. 역할 생성을 선택합니다.

Lambda 함수의 IAM 역할에 대한 인라인 정책 포함

Lambda 함수를 실행하는 데 필요한 MediaConvert 리소스에 권한을 부여하려면 인라인 정책을 사용해야 합니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 역할을 선택합니다.
3. 역할 목록에서, Lambda 함수에 대해 이전에 생성한 IAM 역할의 이름을 선택합니다(예: **tutorial-lambda-transcode-role**).
4. 권한 탭을 선택합니다.
5. 인라인 정책 추가(Add inline policy)를 선택합니다.
6. JSON 탭을 선택한 후 다음 JSON 정책을 붙여 넣습니다.

   JSON 정책에서 `Resource`의 예제 ARN 값을 2단계에서 생성한 MediaConvert에 대한 IAM 역할의 역할 ARN으로 바꿉니다(예: **tutorial-mediaconvert-role**).

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

```
        {
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "Logging"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::111122223333:role/tutorial-mediaconvert-role"
            ],
            "Effect": "Allow",
            "Sid": "PassRole"
        },
        {
            "Action": [
                "mediaconvert:*"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "MediaConvertService"
        },
        {
            "Action": [
                "s3:*"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "S3Service"
        }
    ]
}
```

7. 정책 검토를 선택합니다.

8. 이름에서 **tutorial-lambda-policy**을 입력합니다.
9. 정책 생성을 선택합니다.

   인라인 정책을 생성하면 이 정책이 Lambda 함수의 IAM 역할에 자동으로 포함됩니다.

4단계: 비디오 트랜스코딩용 Lambda 함수 생성

자습서의 이 섹션에서는 S3 Batch Operations 및 MediaConvert와 통합하기 위해 SDK for Python을 사용하여 Lambda 함수를 구축해야 합니다. S3 소스 버킷에 이미 저장된 비디오를 트랜스코딩하려면 S3 소스 버킷의 각 비디오에 대해 Lambda 함수를 직접 호출하는 S3 Batch Operations 작업을 실행합니다. 그런 다음 Lambda 함수는 각 비디오에 대한 트랜스코딩 작업을 MediaConvert에 제출합니다.

하위 단계

- Lambda 함수 코드 작성 및 배포 패키지 생성
- 실행 역할을 사용하여 Lambda 함수 생성(콘솔)
- .zip 파일 아카이브를 사용하여 Lambda 함수를 배포하고 Lambda 함수 구성(콘솔)

Lambda 함수 코드 작성 및 배포 패키지 생성

1. 로컬 시스템에 `batch-transcode`이라는 폴더를 생성하십시오.
2. `batch-transcode` 폴더에서 JSON 작업 설정을 사용하여 파일을 생성합니다. 예를 들어 이 섹션에 제공된 설정을 사용하고 파일 이름을 `job.json`으로 지정할 수 있습니다.

   `job.json` 파일이 지정하는 것은 다음과 같습니다.

   - 트랜스코딩할 파일
   - 입력 비디오를 트랜스코딩하는 방법
   - 생성하려는 출력 미디어 파일
   - 트랜스코딩된 파일에 지정할 이름
   - 트랜스코딩된 파일을 저장할 위치
   - 적용할 고급 기능 등

   이 자습서에서는 다음 `job.json` 파일을 사용하여 S3 소스 버킷의 각 비디오에 대해 다음과 같은 출력을 생성합니다.

- 다양한 크기의 디바이스 및 다양한 대역폭에서 재생할 수 있는 적응형 비트레이트 스트림인 HLS(HTTP Live Streaming)
- MP4 비디오 파일
- 간격에 따라 수집된 썸네일 이미지

이 예제 `job.json` 파일은 품질 정의 가변 비트레이트(QVCR)를 사용하여 비디오 품질을 최적화합니다. HLS 출력은 Apple과 호환됩니다(비디오에서 혼합되지 않은 오디오, 6초의 세그먼트 지속 및 자동 QVBR을 통한 최적화된 비디오 품질).

여기에 제공된 예제 설정을 사용하지 않으려는 경우 사용 사례를 기반으로 `job.json` 사양을 생성할 수 있습니다. 출력 간에 일관성을 유지하려면 입력 파일의 비디오 및 오디오 구성이 비슷한지 확인합니다. 다른 비디오 및 오디오 구성을 가진 입력 파일에 대해 별도의 자동화(고유 `job.json` 설정)를 생성할 수 있습니다. 자세한 내용은 AWS Elemental MediaConvert 사용 설명서의 JSON의 예제 AWS Elemental MediaConvert 작업 설정을 참조하십시오.

```
{
  "OutputGroups": [
    {
      "CustomName": "HLS",
      "Name": "Apple HLS",
      "Outputs": [
        {
          "ContainerSettings": {
            "Container": "M3U8",
            "M3u8Settings": {
              "AudioFramesPerPes": 4,
              "PcrControl": "PCR_EVERY_PES_PACKET",
              "PmtPid": 480,
              "PrivateMetadataPid": 503,
              "ProgramNumber": 1,
              "PatInterval": 0,
              "PmtInterval": 0,
              "TimedMetadata": "NONE",
              "VideoPid": 481,
              "AudioPids": [
                482,
                483,
                484,
                485,
                486,
```

```
                487,
                488,
                489,
                490,
                491,
                492
              ]
            }
          },
          "VideoDescription": {
            "Width": 640,
            "ScalingBehavior": "DEFAULT",
            "Height": 360,
            "TimecodeInsertion": "DISABLED",
            "AntiAlias": "ENABLED",
            "Sharpness": 50,
            "CodecSettings": {
              "Codec": "H_264",
              "H264Settings": {
                "InterlaceMode": "PROGRESSIVE",
                "NumberReferenceFrames": 3,
                "Syntax": "DEFAULT",
                "Softness": 0,
                "GopClosedCadence": 1,
                "GopSize": 2,
                "Slices": 1,
                "GopBReference": "DISABLED",
                "MaxBitrate": 1200000,
                "SlowPal": "DISABLED",
                "SpatialAdaptiveQuantization": "ENABLED",
                "TemporalAdaptiveQuantization": "ENABLED",
                "FlickerAdaptiveQuantization": "DISABLED",
                "EntropyEncoding": "CABAC",
                "FramerateControl": "INITIALIZE_FROM_SOURCE",
                "RateControlMode": "QVBR",
                "CodecProfile": "MAIN",
                "Telecine": "NONE",
                "MinIInterval": 0,
                "AdaptiveQuantization": "HIGH",
                "CodecLevel": "AUTO",
                "FieldEncoding": "PAFF",
                "SceneChangeDetect": "TRANSITION_DETECTION",
                "QualityTuningLevel": "SINGLE_PASS_HQ",
                "FramerateConversionAlgorithm": "DUPLICATE_DROP",
```

```
                "UnregisteredSeiTimecode": "DISABLED",
                "GopSizeUnits": "SECONDS",
                "ParControl": "INITIALIZE_FROM_SOURCE",
                "NumberBFramesBetweenReferenceFrames": 2,
                "RepeatPps": "DISABLED"
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          },
          "OutputSettings": {
            "HlsSettings": {
              "AudioGroupId": "program_audio",
              "AudioRenditionSets": "program_audio",
              "SegmentModifier": "$dt$",
              "IFrameOnlyManifest": "EXCLUDE"
            }
          },
          "NameModifier": "_360"
        },
        {
          "ContainerSettings": {
            "Container": "M3U8",
            "M3u8Settings": {
              "AudioFramesPerPes": 4,
              "PcrControl": "PCR_EVERY_PES_PACKET",
              "PmtPid": 480,
              "PrivateMetadataPid": 503,
              "ProgramNumber": 1,
              "PatInterval": 0,
              "PmtInterval": 0,
              "TimedMetadata": "NONE",
              "TimedMetadataPid": 502,
              "VideoPid": 481,
              "AudioPids": [
                482,
                483,
                484,
                485,
                486,
                487,
                488,
```

```
                489,
                490,
                491,
                492
              ]
            }
          },
          "VideoDescription": {
            "Width": 960,
            "ScalingBehavior": "DEFAULT",
            "Height": 540,
            "TimecodeInsertion": "DISABLED",
            "AntiAlias": "ENABLED",
            "Sharpness": 50,
            "CodecSettings": {
              "Codec": "H_264",
              "H264Settings": {
                "InterlaceMode": "PROGRESSIVE",
                "NumberReferenceFrames": 3,
                "Syntax": "DEFAULT",
                "Softness": 0,
                "GopClosedCadence": 1,
                "GopSize": 2,
                "Slices": 1,
                "GopBReference": "DISABLED",
                "MaxBitrate": 3500000,
                "SlowPal": "DISABLED",
                "SpatialAdaptiveQuantization": "ENABLED",
                "TemporalAdaptiveQuantization": "ENABLED",
                "FlickerAdaptiveQuantization": "DISABLED",
                "EntropyEncoding": "CABAC",
                "FramerateControl": "INITIALIZE_FROM_SOURCE",
                "RateControlMode": "QVBR",
                "CodecProfile": "MAIN",
                "Telecine": "NONE",
                "MinIInterval": 0,
                "AdaptiveQuantization": "HIGH",
                "CodecLevel": "AUTO",
                "FieldEncoding": "PAFF",
                "SceneChangeDetect": "TRANSITION_DETECTION",
                "QualityTuningLevel": "SINGLE_PASS_HQ",
                "FramerateConversionAlgorithm": "DUPLICATE_DROP",
                "UnregisteredSeiTimecode": "DISABLED",
                "GopSizeUnits": "SECONDS",
```

```
                "ParControl": "INITIALIZE_FROM_SOURCE",
                "NumberBFramesBetweenReferenceFrames": 2,
                "RepeatPps": "DISABLED"
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          },
          "OutputSettings": {
            "HlsSettings": {
              "AudioGroupId": "program_audio",
              "AudioRenditionSets": "program_audio",
              "SegmentModifier": "$dt$",
              "IFrameOnlyManifest": "EXCLUDE"
            }
          },
          "NameModifier": "_540"
        },
        {
          "ContainerSettings": {
            "Container": "M3U8",
            "M3u8Settings": {
              "AudioFramesPerPes": 4,
              "PcrControl": "PCR_EVERY_PES_PACKET",
              "PmtPid": 480,
              "PrivateMetadataPid": 503,
              "ProgramNumber": 1,
              "PatInterval": 0,
              "PmtInterval": 0,
              "TimedMetadata": "NONE",
              "VideoPid": 481,
              "AudioPids": [
                482,
                483,
                484,
                485,
                486,
                487,
                488,
                489,
                490,
                491,
```

```
            492
          ]
        }
      },
      "VideoDescription": {
        "Width": 1280,
        "ScalingBehavior": "DEFAULT",
        "Height": 720,
        "TimecodeInsertion": "DISABLED",
        "AntiAlias": "ENABLED",
        "Sharpness": 50,
        "CodecSettings": {
          "Codec": "H_264",
          "H264Settings": {
            "InterlaceMode": "PROGRESSIVE",
            "NumberReferenceFrames": 3,
            "Syntax": "DEFAULT",
            "Softness": 0,
            "GopClosedCadence": 1,
            "GopSize": 2,
            "Slices": 1,
            "GopBReference": "DISABLED",
            "MaxBitrate": 5000000,
            "SlowPal": "DISABLED",
            "SpatialAdaptiveQuantization": "ENABLED",
            "TemporalAdaptiveQuantization": "ENABLED",
            "FlickerAdaptiveQuantization": "DISABLED",
            "EntropyEncoding": "CABAC",
            "FramerateControl": "INITIALIZE_FROM_SOURCE",
            "RateControlMode": "QVBR",
            "CodecProfile": "MAIN",
            "Telecine": "NONE",
            "MinIInterval": 0,
            "AdaptiveQuantization": "HIGH",
            "CodecLevel": "AUTO",
            "FieldEncoding": "PAFF",
            "SceneChangeDetect": "TRANSITION_DETECTION",
            "QualityTuningLevel": "SINGLE_PASS_HQ",
            "FramerateConversionAlgorithm": "DUPLICATE_DROP",
            "UnregisteredSeiTimecode": "DISABLED",
            "GopSizeUnits": "SECONDS",
            "ParControl": "INITIALIZE_FROM_SOURCE",
            "NumberBFramesBetweenReferenceFrames": 2,
            "RepeatPps": "DISABLED"
```

```
            }
          },
          "AfdSignaling": "NONE",
          "DropFrameTimecode": "ENABLED",
          "RespondToAfd": "NONE",
          "ColorMetadata": "INSERT"
        },
        "OutputSettings": {
          "HlsSettings": {
            "AudioGroupId": "program_audio",
            "AudioRenditionSets": "program_audio",
            "SegmentModifier": "$dt$",
            "IFrameOnlyManifest": "EXCLUDE"
          }
        },
        "NameModifier": "_720"
      },
      {
        "ContainerSettings": {
          "Container": "M3U8",
          "M3u8Settings": {}
        },
        "AudioDescriptions": [
          {
            "AudioSourceName": "Audio Selector 1",
            "CodecSettings": {
              "Codec": "AAC",
              "AacSettings": {
                "Bitrate": 96000,
                "CodingMode": "CODING_MODE_2_0",
                "SampleRate": 48000
              }
            }
          }
        ],
        "OutputSettings": {
          "HlsSettings": {
            "AudioGroupId": "program_audio",
            "AudioTrackType": "ALTERNATE_AUDIO_AUTO_SELECT_DEFAULT"
          }
        },
        "NameModifier": "_audio"
      }
    ],
```

```
      "OutputGroupSettings": {
        "Type": "HLS_GROUP_SETTINGS",
        "HlsGroupSettings": {
          "ManifestDurationFormat": "INTEGER",
          "SegmentLength": 6,
          "TimedMetadataId3Period": 10,
          "CaptionLanguageSetting": "OMIT",
          "Destination": "s3://EXAMPLE-BUCKET/HLS/",
          "DestinationSettings": {
            "S3Settings": {
              "AccessControl": {
                "CannedAcl": "PUBLIC_READ"
              }
            }
          },
          "TimedMetadataId3Frame": "PRIV",
          "CodecSpecification": "RFC_4281",
          "OutputSelection": "MANIFESTS_AND_SEGMENTS",
          "ProgramDateTimePeriod": 600,
          "MinSegmentLength": 0,
          "DirectoryStructure": "SINGLE_DIRECTORY",
          "ProgramDateTime": "EXCLUDE",
          "SegmentControl": "SEGMENTED_FILES",
          "ManifestCompression": "NONE",
          "ClientCache": "ENABLED",
          "StreamInfResolution": "INCLUDE"
        }
      }
    },
    {
      "CustomName": "MP4",
      "Name": "File Group",
      "Outputs": [
        {
          "ContainerSettings": {
            "Container": "MP4",
            "Mp4Settings": {
              "CslgAtom": "INCLUDE",
              "FreeSpaceBox": "EXCLUDE",
              "MoovPlacement": "PROGRESSIVE_DOWNLOAD"
            }
          },
          "VideoDescription": {
            "Width": 1280,
```

```
            "ScalingBehavior": "DEFAULT",
            "Height": 720,
            "TimecodeInsertion": "DISABLED",
            "AntiAlias": "ENABLED",
            "Sharpness": 100,
            "CodecSettings": {
              "Codec": "H_264",
              "H264Settings": {
                "InterlaceMode": "PROGRESSIVE",
                "ParNumerator": 1,
                "NumberReferenceFrames": 3,
                "Syntax": "DEFAULT",
                "Softness": 0,
                "GopClosedCadence": 1,
                "HrdBufferInitialFillPercentage": 90,
                "GopSize": 2,
                "Slices": 2,
                "GopBReference": "ENABLED",
                "HrdBufferSize": 10000000,
                "MaxBitrate": 5000000,
                "ParDenominator": 1,
                "EntropyEncoding": "CABAC",
                "RateControlMode": "QVBR",
                "CodecProfile": "HIGH",
                "MinIInterval": 0,
                "AdaptiveQuantization": "AUTO",
                "CodecLevel": "AUTO",
                "FieldEncoding": "PAFF",
                "SceneChangeDetect": "ENABLED",
                "QualityTuningLevel": "SINGLE_PASS_HQ",
                "UnregisteredSeiTimecode": "DISABLED",
                "GopSizeUnits": "SECONDS",
                "ParControl": "SPECIFIED",
                "NumberBFramesBetweenReferenceFrames": 3,
                "RepeatPps": "DISABLED",
                "DynamicSubGop": "ADAPTIVE"
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          },
          "AudioDescriptions": [
```

```
            {
              "AudioTypeControl": "FOLLOW_INPUT",
              "AudioSourceName": "Audio Selector 1",
              "CodecSettings": {
                "Codec": "AAC",
                "AacSettings": {
                  "AudioDescriptionBroadcasterMix": "NORMAL",
                  "Bitrate": 160000,
                  "RateControlMode": "CBR",
                  "CodecProfile": "LC",
                  "CodingMode": "CODING_MODE_2_0",
                  "RawFormat": "NONE",
                  "SampleRate": 48000,
                  "Specification": "MPEG4"
                }
              },
              "LanguageCodeControl": "FOLLOW_INPUT",
              "AudioType": 0
            }
          ]
        }
      ],
      "OutputGroupSettings": {
        "Type": "FILE_GROUP_SETTINGS",
        "FileGroupSettings": {
          "Destination": "s3://EXAMPLE-BUCKET/MP4/",
          "DestinationSettings": {
            "S3Settings": {
              "AccessControl": {
                "CannedAcl": "PUBLIC_READ"
              }
            }
          }
        }
      }
    },
    {
      "CustomName": "Thumbnails",
      "Name": "File Group",
      "Outputs": [
        {
          "ContainerSettings": {
            "Container": "RAW"
          },
```

```
          "VideoDescription": {
            "Width": 1280,
            "ScalingBehavior": "DEFAULT",
            "Height": 720,
            "TimecodeInsertion": "DISABLED",
            "AntiAlias": "ENABLED",
            "Sharpness": 50,
            "CodecSettings": {
              "Codec": "FRAME_CAPTURE",
              "FrameCaptureSettings": {
                "FramerateNumerator": 1,
                "FramerateDenominator": 5,
                "MaxCaptures": 500,
                "Quality": 80
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          }
        }
      ],
      "OutputGroupSettings": {
        "Type": "FILE_GROUP_SETTINGS",
        "FileGroupSettings": {
          "Destination": "s3://EXAMPLE-BUCKET/Thumbnails/",
          "DestinationSettings": {
            "S3Settings": {
              "AccessControl": {
                "CannedAcl": "PUBLIC_READ"
              }
            }
          }
        }
      }
    }
  ],
  "AdAvailOffset": 0,
  "Inputs": [
    {
      "AudioSelectors": {
        "Audio Selector 1": {
          "Offset": 0,
```

```
          "DefaultSelection": "DEFAULT",
          "ProgramSelection": 1
        }
      },
      "VideoSelector": {
        "ColorSpace": "FOLLOW"
      },
      "FilterEnable": "AUTO",
      "PsiControl": "USE_PSI",
      "FilterStrength": 0,
      "DeblockFilter": "DISABLED",
      "DenoiseFilter": "DISABLED",
      "TimecodeSource": "EMBEDDED",
      "FileInput": "s3://EXAMPLE-INPUT-BUCKET/input.mp4"
    }
  ]
}
```

3. batch-transcode 폴더에서 Lambda 함수를 사용하여 파일을 생성합니다. 다음 Python 예제를 사용하여 convert.py 파일 이름을 지정할 수 있습니다.

   S3 Batch Operations는 특정 태스크 데이터를 Lambda 함수로 전송하고 결과 데이터를 다시 요구합니다. Lambda 함수에 대한 요청 및 응답 예제, 응답 및 결과 코드에 대한 정보, S3 Batch Operations의 Lambda 함수 예제는 AWS Lambda 함수 호출을(를) 참조하십시오.

```
import json
import os
from urllib.parse import urlparse
import uuid
import boto3

"""
When you run an S3 Batch Operations job, your job
invokes this Lambda function. Specifically, the Lambda function is
invoked on each video object listed in the manifest that you specify
for the S3 Batch Operations job in Step 5.

Input parameter "event": The S3 Batch Operations event as a request
                         for the Lambda function.

Input parameter "context": Context about the event.

Output: A result structure that Amazon S3 uses to interpret the result
```

```
        of the operation. It is a job response returned back to S3 Batch
 Operations.
"""
def handler(event, context):

    invocation_schema_version = event['invocationSchemaVersion']
    invocation_id = event['invocationId']
    task_id = event['tasks'][0]['taskId']

    source_s3_key = event['tasks'][0]['s3Key']
    source_s3_bucket = event['tasks'][0]['s3BucketArn'].split(':::')[-1]
    source_s3 = 's3://' + source_s3_bucket + '/' + source_s3_key

    result_list = []
    result_code = 'Succeeded'
    result_string = 'The input video object was converted successfully.'

    # The type of output group determines which media players can play
    # the files transcoded by MediaConvert.
    # For more information, see Creating outputs with AWS Elemental MediaConvert.
    output_group_type_dict = {
        'HLS_GROUP_SETTINGS': 'HlsGroupSettings',
        'FILE_GROUP_SETTINGS': 'FileGroupSettings',
        'CMAF_GROUP_SETTINGS': 'CmafGroupSettings',
        'DASH_ISO_GROUP_SETTINGS': 'DashIsoGroupSettings',
        'MS_SMOOTH_GROUP_SETTINGS': 'MsSmoothGroupSettings'
    }

    try:
        job_name = 'Default'
        with open('job.json') as file:
            job_settings = json.load(file)

        job_settings['Inputs'][0]['FileInput'] = source_s3

        # The path of each output video is constructed based on the values of
        # the attributes in each object of OutputGroups in the job.json file.
        destination_s3 = 's3://{0}/{1}/{2}' \
            .format(os.environ['amzn-s3-demo-destination-bucket'],
                    os.path.splitext(os.path.basename(source_s3_key))[0],
                    os.path.splitext(os.path.basename(job_name))[0])

        for output_group in job_settings['OutputGroups']:
            output_group_type = output_group['OutputGroupSettings']['Type']
```

```
            if output_group_type in output_group_type_dict.keys():
                output_group_type = output_group_type_dict[output_group_type]
                output_group['OutputGroupSettings'][output_group_type]
['Destination'] = \
                    "{0}{1}".format(destination_s3,
                                    urlparse(output_group['OutputGroupSettings']
[output_group_type]['Destination']).path)
            else:
                raise ValueError("Exception: Unknown Output Group Type {}."
                                 .format(output_group_type))

        job_metadata_dict = {
            'assetID': str(uuid.uuid4()),
            'application': os.environ['Application'],
            'input': source_s3,
            'settings': job_name
        }

        region = os.environ['AWS_DEFAULT_REGION']
        endpoints = boto3.client('mediaconvert', region_name=region) \
            .describe_endpoints()
        client = boto3.client('mediaconvert', region_name=region,
                              endpoint_url=endpoints['Endpoints'][0]['Url'],
                              verify=False)

        try:
            client.create_job(Role=os.environ['MediaConvertRole'],
                              UserMetadata=job_metadata_dict,
                              Settings=job_settings)
        # You can customize error handling based on different error codes that
        # MediaConvert can return.
        # For more information, see MediaConvert error codes.
        # When the result_code is TemporaryFailure, S3 Batch Operations retries
        # the task before the job is completed. If this is the final retry,
        # the error message is included in the final report.
        except Exception as error:
            result_code = 'TemporaryFailure'
            raise

    except Exception as error:
        if result_code != 'TemporaryFailure':
            result_code = 'PermanentFailure'
        result_string = str(error)
```

```
    finally:
        result_list.append({
            'taskId': task_id,
            'resultCode': result_code,
            'resultString': result_string,
        })

    return {
        'invocationSchemaVersion': invocation_schema_version,
        'treatMissingKeyAs': 'PermanentFailure',
        'invocationId': invocation_id,
        'results': result_list
    }
```

4. convert.py 및 job.json이(가) 포함된 배포 패키지를 lambda.zip라고 하는 .zip 파일로 생성하려면, 로컬 터미널에서 이전에 생성한 batch-transcode 폴더를 열고 다음 명령을 실행합니다.

   macOS 사용자의 경우 다음 명령을 실행합니다.

   ```
   zip -r lambda.zip convert.py job.json
   ```

   Windows 사용자의 경우 다음 명령을 실행합니다.

   ```
   powershell Compress-Archive convert.py lambda.zip
   ```

   ```
   powershell Compress-Archive -update job.json lambda.zip
   ```

실행 역할을 사용하여 Lambda 함수 생성(콘솔)

1. https://console.aws.amazon.com/lambda/에서 AWS Lambda 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 함수(Functions)를 선택합니다.
3. 함수 생성을 선택합니다.
4. 새로 작성을 선택합니다.
5. 기본 정보에서 다음과 같이 합니다.

   a. [함수 이름(Function name)]에 **tutorial-lambda-convert**을 입력합니다.

b. 런타임에서 Python 3.13을 선택합니다.

6. 기본 실행 역할 변경(Change default execution role)을 선택하고 실행 역할(Execution role)에서 기존 역할 사용(Use an existing role)을 선택합니다.
7. 기존 역할에서, 3단계에서 Lambda 함수용으로 생성한 IAM 역할의 이름을 선택합니다(예: **`tutorial-lambda-transcode-role`**).
8. 나머지 설정은 기본값으로 유지합니다.
9. 함수 생성을 선택합니다.

.zip 파일 아카이브를 사용하여 Lambda 함수를 배포하고 Lambda 함수 구성(콘솔)

1. 생성한 Lambda 함수(예: **`tutorial-lambda-convert`**)에 대한 페이지의 코드 소스 섹션에서 업로드 원본을 선택한 후 .zip 파일을 선택합니다.
2. 업로드(Upload)를 선택하여 로컬 `.zip` 파일을 선택합니다.
3. 이전에 생성한 `lambda.zip` 파일을 선택하고 열기를 선택합니다.
4. 저장을 선택합니다.
5. 런타임 설정(Runtime settings) 섹션에서 편집(Edit)을 선택합니다.
6. Lambda 함수 코드에서 호출할 핸들러 메서드를 Lambda 런타임에 알리기 위해 핸들러 필드에 **`convert.handler`**을(를) 입력합니다.

   Python에서 함수를 구성할 때, 핸들러 설정의 값은 파일의 이름과 핸들러 모듈의 이름이며 점(`.`)으로 구분됩니다. 예를 들어 `convert.handler`는 `convert.py` 파일에 정의된 `handler` 메서드를 호출합니다.
7. 저장을 선택합니다.
8. Lambda 함수 페이지에서 구성(Configuration) 탭을 선택합니다. 구성 탭의 왼쪽 탐색 창에서 환경 변수를 선택한 다음 편집을 선택합니다.
9. Add environment variable(환경 변수 추가)을 선택합니다. 그런 후, 다음의 각 환경 변수에 대해 지정된 키와 값을 입력합니다.

   - 키: **`DestinationBucket`** 값: **`amzn-s3-demo-destination-bucket1`**

     이 값은 1단계에서 생성한 출력 미디어 파일의 S3 버킷입니다.
   - 키: **`MediaConvertRole`** 값: **`arn:aws:iam::`*`111122223333`*`:role/tutorial-mediaconvert-role`**

이 값은 2단계에서 생성한 MediaConvert용 IAM 역할의 ARN입니다. 이 ARN을 IAM 역할의 실제 ARN으로 바꿔야 합니다.

- 키: **`Application`** 값: **`Batch-Transcoding`**

  이 값은 애플리케이션의 이름입니다.

10. 저장을 선택합니다.
11. (선택 사항) 구성(Configuration) 탭에서, 왼쪽 탐색 창의 일반 구성(General configuration) 섹션에서 편집(Edit)을 선택합니다. 제한 시간(Timeout) 필드에 **`2`**분 **`0`**초를 입력합니다. 그런 다음 저장을 선택합니다.

    제한 시간(Timeout)은 Lambda가 함수를 중지하기까지 호출 실행을 허용하는 시간입니다. 기본값은 3초입니다. 요금은 구성된 메모리 양과 코드가 실행되는 시간을 기준으로 책정됩니다. 자세한 내용은 AWS Lambda 요금을 참조하십시오.

5단계: S3 소스 버킷용 Amazon S3 인벤토리 구성

트랜스코딩 Lambda 함수를 설정한 후 비디오 세트를 트랜스코딩하기 위해 S3 Batch Operations 작업을 생성합니다. 먼저 S3 배치 작업에서 지정된 트랜스코딩 작업을 실행할 입력 비디오 객체 목록이 필요합니다. 입력 비디오 객체 목록을 가져오기 위해 S3 소스 버킷에 대한 S3 인벤토리 보고서를 생성할 수 있습니다(예: *`amzn-s3-demo-source-bucket`*).

하위 단계

- 입력 비디오의 S3 인벤토리 보고서용 버킷 생성 및 구성
- S3 비디오 소스 버킷용 Amazon S3 인벤토리 구성
- S3 비디오 소스 버킷에 대한 인벤토리 보고서 확인

입력 비디오의 S3 인벤토리 보고서용 버킷 생성 및 구성

S3 소스 버킷의 객체를 나열하는 S3 인벤토리 보고서를 저장하려면 S3 인벤토리 대상 버킷을 만들고 S3 소스 버킷에 인벤토리 파일을 기록하도록 버킷에 대한 버킷 정책을 구성해야 합니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 생성을 선택합니다.

4. 버킷 이름에 버킷 이름을 입력합니다(예: *amzn-s3-demo-destination-bucket2*).
5. AWS 리전은(는) 버킷이 속할 AWS 리전을(를) 선택합니다.

   인벤토리 대상 버킷은 S3 인벤토리를 설정하는 소스 버킷과 동일한 AWS 리전에 있어야 합니다. 인벤토리 대상 버킷은 다른 AWS 계정에 있을 수 있습니다.
6. 이 버킷에 대한 퍼블릭 액세스 차단 설정에서 기본값 설정을 유지합니다(모든 퍼블릭 액세스 차단이 사용 설정됨).
7. 나머지 설정은 기본값으로 유지합니다.
8. 버킷 생성을 선택합니다.
9. 버킷(Buckets) 목록에서 방금 생성한 버킷의 이름을 선택합니다(예: *amzn-s3-demo-destination-bucket2*).
10. 인벤토리 보고서에 대한 데이터를 S3 인벤토리 대상 버킷에 작성할 수 있는 권한을 Amazon S3에 부여하려면 권한 탭을 선택합니다.
11. 버킷 정책 섹션으로 스크롤을 내린 다음 편집을 선택합니다. 이버킷 정책 페이지가 열립니다.
12. S3 인벤토리에 대한 권한을 부여하려면 정책 필드에서 다음 버킷 정책을 붙여 넣습니다.

    세 가지 예제 값을 다음 값으로 바꿉니다.

    - 인벤토리 보고서를 저장하기 위해 생성한 버킷의 이름(예: *amzn-s3-demo-destination-bucket2*)
    - 입력 비디오를 저장하는 소스 버킷의 이름입니다(예: *amzn-s3-demo-source-bucket*).
    - S3 비디오 소스 버킷을 만드는 데 사용한 AWS 계정 ID입니다(예: *111122223333*).

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"InventoryAndAnalyticsExamplePolicy",
      "Effect":"Allow",
      "Principal": {"Service": "s3.amazonaws.com"},
      "Action":"s3:PutObject",
      "Resource":["arn:aws:s3:::amzn-s3-demo-destination-bucket2/*"],
      "Condition": {
          "ArnLike": {
              "aws:SourceArn": "arn:aws:s3:::amzn-s3-demo-source-bucket"
           },
         "StringEquals": {
```

```
                "aws:SourceAccount": "111122223333",
                "s3:x-amz-acl": "bucket-owner-full-control"
            }
        }
    }
  ]
}
```

13. 변경 사항 저장을 선택합니다.

S3 비디오 소스 버킷용 Amazon S3 인벤토리 구성

비디오 객체 및 메타데이터의 플랫 파일 목록을 생성하려면 S3 비디오 소스 버킷에 대한 S3 인벤토리를 구성해야 합니다. 이러한 예약된 인벤토리 보고서에는 버킷의 모든 객체 또는 공유 접두사로 그룹화된 객체가 포함될 수 있습니다. 이 자습서에서, S3 인벤토리 보고서에는 S3 소스 버킷의 모든 비디오 객체가 포함됩니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. S3 소스 버킷에서 입력 비디오에 대한 S3 인벤토리 보고서를 구성하려면 버킷 목록에서 S3 소스 버킷의 이름을 선택합니다(예: *amzn-s3-demo-source-bucket*).
4. [Management] 탭을 선택한 후
5. 아래의 인벤토리 구성 섹션으로 스크롤하고 인벤토리 구성 생성을 선택합니다.
6. 인벤토리 구성 이름(Inventory configuration name)에 이름을 입력합니다(예: **tutorial-inventory-config**).
7. 인벤토리 범위의 객체 버전에서 현재 버전만을 선택하고 다른 인벤토리 범위 설정을 이 자습서의 기본값으로 유지합니다.
8. 보고서 세부 정보 섹션의 대상 버킷에서 이 계정을 선택합니다.
9. 대상에서 S3 찾아보기를 선택하고, 이전에 생성한 대상 버킷을 선택하여 인벤토리 보고서 저장용으로 선택합니다(예: *amzn-s3-demo-destination-bucket2*). 그런 다음 경로 선택을 선택합니다.

   인벤토리 대상 버킷은 S3 인벤토리를 설정하는 소스 버킷과 동일한 AWS 리전에 있어야 합니다. 인벤토리 대상 버킷은 다른 AWS 계정에 있을 수 있습니다.

대상(Destination) 버킷 필드에서, 대상 버킷 권한(Destination bucket permission)이 인벤토리 대상 버킷 정책에 추가되어 Amazon S3가 해당 인벤토리 대상 버킷에 데이터를 저장할 수 있습니다. 자세한 내용은 대상 버킷 정책 생성 섹션을 참조하세요.

10. 빈도는 일별을 선택합니다.
11. [출력 형식(Output format)]으로 [CSV]를 선택합니다.
12. 상태는 활성을 선택합니다.
13. 서버 측 암호화 섹션에서 이 자습서에 대해 사용 중지를 선택합니다.

    자세한 내용은 S3 콘솔을 사용하여 인벤토리 구성 및 암호화에 고객 관리형 키를 사용하도록 Amazon S3에 권한 부여 단원을 참조하세요.
14. 추가 필드 - 선택 사항 섹션에서 크기, 최종 수정 날짜 및 스토리지 클래스를 선택합니다.
15. 생성(Create)을 선택합니다.

자세한 내용은 S3 콘솔을 사용하여 인벤토리 구성 섹션을 참조하세요.

S3 비디오 소스 버킷에 대한 인벤토리 보고서 확인

인벤토리 보고서가 게시될 때 매니페스트 파일은 S3 인벤토리 대상 버킷으로 전송됩니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷(Buckets) 목록에서 비디오 소스 버킷의 이름을 선택합니다(예: *amzn-s3-demo-source-bucket*).
4. 관리를 선택합니다.
5. S3 인벤토리 보고서가 준비되어 7단계의 S3 Batch Operations 작업을 생성할 수 있는지 확인하려면, 인벤토리 구성에서 매니페스트에서 작업 생성 버튼이 사용 설정되어 있는지 확인합니다.

> **Note**
>
> 첫 번째 인벤토리 보고서를 전달하는 데 최대 48시간이 걸릴 수 있습니다. 매니페스트에서 작업 생성(Create job from manifest) 버튼이 사용 중지되어 있으면 첫 번째 인벤토리 보고서가 전달되지 않은 것입니다. 첫 번째 인벤토리 보고서가 전달되고 매니페스트에서 작업 생성 버튼이 사용 설정될 때까지 기다린 후 7단계의 S3 Batch Operations 작업을 생성합니다.

6. S3 인벤토리 보고서(`manifest.json`)를 확인하려면 대상 열에서 인벤토리 보고서 저장을 위해 이전에 생성한 인벤토리 대상 버킷의 이름을 선택합니다(예: *amzn-s3-demo-destination-bucket2*).
7. 객체 탭에서 S3 소스 버킷의 이름을 가진 기존 폴더를 선택합니다(예:*amzn-s3-demo-source-bucket*). 그런 다음 인벤토리 구성을 이전에 생성했을 때 인벤토리 구성 이름에 입력한 이름을 선택합니다(예:**`tutorial-inventory-config`**).

   보고서의 생성 날짜가 해당 이름으로 포함된 폴더 목록을 볼 수 있습니다.
8. 특정 날짜의 일별 S3 인벤토리 보고서를 확인하려면 해당 생성 날짜 이름이 있는 폴더를 선택한 후 `manifest.json`을(를) 선택합니다.
9. 특정 날짜의 인벤토리 보고서에 대한 세부 정보를 확인하려면 manifest.json 페이지에서 다운로드(Download) 또는 열기(Open)를 선택합니다.

6단계: S3 배치 작업용 IAM 역할 생성

S3 Batch Operations를 사용하여 일괄 트랜스코딩을 수행하려면 먼저 Amazon S3에 S3 Batch Operations를 수행할 수 있는 권한을 부여하록 IAM 역할을 생성해야 합니다.

하위 단계

- S3 배치 작업용 IAM 정책 생성
- S3 Batch Operation IAM 역할 생성 및 권한 정책 연결

S3 배치 작업용 IAM 정책 생성

입력 매니페스트를 읽고 Lambda 함수를 호출하며 S3 Batch Operations 작업 완료 보고서를 작성할 수 있는 권한을 S3 Batch Operations에 부여하는 IAM 정책을 만들어야 합니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 정책을 선택합니다.
3. 정책 생성을 선택합니다.
4. JSON 탭을 선택합니다.
5. JSON 텍스트 필드에 다음 JSON 정책을 붙여 넣습니다.

   JSON 정책에서 네 가지 예제 값을 다음 값으로 바꿉니다.

- 입력 비디오를 저장하는 소스 버킷의 이름입니다(예: *amzn-s3-demo-source-bucket*).
- `manifest.json` 파일을 저장하기 위해 5단계에서 생성한 인벤토리 대상 버킷의 이름(예: *amzn-s3-demo-destination-bucket2*)
- 출력 미디어 파일을 저장하기 위해 1단계에서 생성한 버킷의 이름입니다(예: *amzn-s3-demo-destination-bucket1*). 이 자습서에서는 출력 미디어 파일의 대상 버킷에 작업 완료 보고서를 저장합니다.
- 4단계에서 생성한 Lambda 함수의 역할 ARN Lambda 함수의 역할 ARN을 찾아 복사하려면 다음을 수행합니다.
  - 새 브라우저 탭에서 https://console.aws.amazon.com/lambda/home#/functions의 Lambda 콘솔에서 함수 페이지를 엽니다.
  - 함수 목록에서, 4단계에서 생성한 Lambda 함수의 이름을 선택합니다(예, **tutorial-lambda-convert**).
  - ARN 복사(Copy ARN)를 선택합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3Get",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-source-bucket/*",
                "arn:aws:s3:::amzn-s3-demo-destination-bucket2/*"
            ]
        },
        {
            "Sid": "S3PutJobCompletionReport",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket1/*"
        },
        {
            "Sid": "S3BatchOperationsInvokeLambda",
            "Effect": "Allow",
```

```
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-west-2:111122223333:function:tutorial-lambda-
convert"
            ]
        }
    ]
}
```

6. 다음: 태그를 선택합니다.
7. 다음: 검토를 선택합니다.
8. 이름 필드에 **tutorial-s3batch-policy**을 입력합니다.
9. 정책 생성을 선택합니다.

S3 Batch Operation IAM 역할 생성 및 권한 정책 연결

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 역할을 선택한 다음, 역할 생성을 선택합니다.
3. AWS 서비스 역할 유형을 선택한 다음, S3 서비스를 선택합니다.
4. 사용 사례 선택(Select your use case)에서 S3 배치 작업(S3 Batch Operations)을 선택합니다.
5. 다음을 선택합니다.
6. 권한 연결에서 검색 상자에 이전에 생성한 IAM 정책의 이름(예:**tutorial-s3batch-policy**)을 입력하여 정책 목록을 필터링합니다. 정책 이름 옆에 있는 확인란을 선택합니다(예:**tutorial-s3batch-policy**).
7. 다음을 선택합니다.
8. 역할 이름에 **tutorial-s3batch-role**을 입력합니다.
9. 역할 생성을 선택합니다.

   S3 Batch Operations용 IAM 역할을 생성한 후 다음 신뢰 정책이 해당 역할에 자동으로 연결됩니다. 이 신뢰 정책은 S3 Batch Operations 서비스 보안 주체가 IAM 역할을 수임하도록 허용합니다.

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

```
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

7단계: S3 배치 작업 생성 및 실행

S3 소스 버킷의 입력 비디오를 처리하기 위해 S3 Batch Operations 작업을 만들려면 이 특정 작업에 대한 파라미터를 지정해야 합니다.

> **Note**
>
> S3 Batch Operations 작업 생성을 시작하려면 매니페스트에서 작업 생성 버튼이 사용 설정되어 있는지 확인합니다. 자세한 내용은 S3 비디오 소스 버킷에 대한 인벤토리 보고서 확인 섹션을 참조하세요. 매니페스트에서 작업 생성 버튼이 사용 중지되어 있으면 첫 번째 인벤토리 보고서가 전달되지 않은 것이며, 버튼이 사용 설정될 때까지 기다려야 합니다. 5단계에서 S3 소스 버킷에 대한 Amazon S3 인벤토리를 구성한 후, 첫 번째 인벤토리 보고서를 전달하는 데 최대 48시간이 걸릴 수 있습니다.

하위 단계

- S3 배치 작업 생성
- S3 배치 작업을 실행하여 Lambda 함수 호출
- (선택 사항) 완료 보고서 확인
- (선택 사항) Lambda 콘솔에서 각 Lambda 호출 모니터링
- (선택 사항) MediaConvert 콘솔에서 각 MediaConvert 비디오 트랜스코딩 작업 모니터링

S3 배치 작업 생성

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.

2. 왼쪽 탐색 창에서 Batch Operations를 선택합니다.
3. 작업 생성을 선택합니다.
4. AWS 리전은 작업을 생성하려는 리전을 선택합니다.

   이 자습서에서 S3 Batch Operations 작업을 사용하여 Lambda 함수를 호출하려면 매니페스트에서 참조된 객체가 있는 S3 비디오 소스 버킷과 동일한 리전에서 작업을 생성해야 합니다.
5. 매니페스트 섹션에서 다음을 수행합니다.

   a. 매니페스트 형식(Manifest format)에서 S3 인벤토리 보고서(S3 Inventory report) (manifest.json)를 선택합니다.

   b. 매니페스트 객체에서 S3 찾아보기를 선택하여 인벤토리 보고서 저장을 위해 5단계에서 생성한 버킷을 찾습니다(예: *amzn-s3-demo-destination-bucket2*). 매니페스트 객체 페이지에서 특정 날짜에 대한 `manifest.json` 파일을 찾을 때까지 객체 이름을 탐색합니다. 이 파일에는 일괄 트랜스코딩하려는 모든 비디오에 대한 정보가 나열됩니다. 사용할 `manifest.json` 파일을 찾으면, 옆에 있는 옵션 버튼을 선택합니다. 그런 다음 경로 선택을 선택합니다.

   c. (선택 사항) 최신 버전이 아닌 버전을 사용하려는 경우, 매니페스트 객체 버전 ID - 선택 사항에서 매니페스트 객체의 버전 ID를 입력합니다.
6. 다음을 선택합니다.
7. Lambda 함수를 사용하여 선택된 `manifest.json` 파일에 나열된 모든 객체를 트랜스코딩하려면 작업 유형에서 AWS Lambda 함수 호출을 선택합니다.
8. Lambda 함수 호출 섹션에서 다음을 수행합니다.

   a. 계정의 함수에서 선택(Choose from functions in your account)을 선택합니다.

   b. Lambda 함수에서, 4단계에서 생성한 Lambda 함수를 선택합니다(예: **`tutorial-lambda-convert`**).

   c. Lambda 함수 버전에서, 기본값 $LATEST를 유지합니다.
9. 다음을 선택합니다. 추가 옵션 구성 페이지가 열립니다.
10. 추가 옵션 섹션에서 기본값 설정을 유지합니다.

    이러한 옵션에 대한 자세한 내용은 배치 작업 요청 요소 섹션을 참조하십시오.
11. 보고서 완료섹션에서 완료 보고서 대상 경로는 S3 찾아보기를 선택합니다. 1단계에서 출력 미디어 파일에 대해 생성한 버킷을 찾습니다(예: *amzn-s3-demo-destination-bucket1*). 해당 버킷 이름 옆에 있는 옵션 버튼을 선택합니다. 그런 다음 경로 선택을 선택합니다.

나머지 완료 보고서 설정은 기본값을 유지합니다. 완료 보고서 설정에 대한 자세한 내용은 배치 작업 요청 요소 섹션을 참조하십시오. 완료 보고서는 작업 세부 정보 및 수행된 작업의 레코드를 유지 관리합니다.

12. 권한 섹션에서, 기존 IAM 역할에서 선택을 선택합니다. IAM 역할(IAM role)에서, 6단계에서 생성한 S3 배치 작업의 IAM 역할을 선택합니다(예: **`tutorial-s3batch-role`**).
13. 다음을 선택합니다.
14. 검토 페이지에서 설정을 검토합니다. 그런 다음 작업 생성을 선택합니다.

    S3가 S3 Batch Operations 작업의 매니페스트 읽기를 마치면, 작업의 상태를 실행 확인을 기다리는 중으로 설정합니다. 작업 상태에 대한 업데이트를 보려면 페이지를 새로 고칩니다. 작업의 상태가 실행 확인을 기다리는 중일 때까지 작업을 실행할 수 없습니다.

S3 배치 작업을 실행하여 Lambda 함수 호출

배치 작업을 실행하여 비디오 트랜스코딩을 위한 Lambda 함수를 호출합니다. 작업이 실패하면 완료 보고서를 확인하여 원인을 파악할 수 있습니다.

S3 배치 작업 실행

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Batch Operations를 선택합니다.
3. 작업 목록의 첫 번째 행에서, 이전에 생성한 S3 Batch Operations 작업의 작업 ID를 선택합니다.
4. 작업 실행(Run job)을 선택합니다.
5. 작업 파라미터를 다시 검토하고 매니페스트에 나열된 총 객체(Total objects listed in manifest)의 값이 매니페스트의 객체 수와 동일한지 확인합니다. 그런 다음 작업 실행을 선택합니다.

   S3 배치 작업 페이지가 열립니다.
6. 작업 실행이 시작된 후 작업 페이지의 상태(Status)에서 상태, % 완료, 총 성공(비율), 총 실패(비율), 종료 날짜, 종료 이유와 같은 S3 배치 작업 진행 상황을 확인합니다.

   S3 Batch Operations 작업이 완료되면 작업 페이지의 데이터를 보고 작업이 예상대로 완료되었는지 확인합니다.

1,000개 이상의 작업이 시도된 후 S3 Batch Operations 작업의 객체 작업 중 50 퍼센트 이상이 실패하면 작업이 자동으로 실패합니다. 완료 보고서를 확인하여 실패의 원인을 파악하려면 아래의 선택적 절차를 사용하십시오.

(선택 사항) 완료 보고서 확인

완료 보고서를 사용하여 실패한 객체와 실패의 원인을 확인할 수 있습니다.

실패한 객체에 대한 세부 정보를 완료 보고서에서 확인하려면 다음을 수행합니다.

1. S3 Batch Operations 작업 페이지에서 아래 완료 보고서 섹션으로 스크롤하고 완료 보고서 대상의 링크를 선택합니다.

   S3 출력 대상 버킷 페이지가 열립니다.
2. 객체 탭에서, 이전에 생성한 S3 Batch Operations 작업의 작업 ID로 끝나는 이름이 있는 폴더를 선택합니다.
3. results/를 선택합니다.
4. `.csv` 파일 옆의 확인란을 선택합니다.
5. 작업 보고서를 보려면 열기 또는 다운로드를 선택합니다.

(선택 사항) Lambda 콘솔에서 각 Lambda 호출 모니터링

S3 Batch Operations 작업 실행이 시작되면 작업이 각 입력 비디오 객체에 대해 Lambda 함수를 호출합니다. S3는 각 Lambda 호출 로그를 CloudWatch Logs에 기록합니다. Lambda 콘솔의 모니터링 대시보드를 사용하여 Lambda 함수를 모니터링할 수 있습니다.

1. https://console.aws.amazon.com/lambda/에서 AWS Lambda 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 함수(Functions)를 선택합니다.
3. 함수 목록에서, 4단계에서 생성한 Lambda 함수의 이름을 선택합니다(예, **`tutorial-lambda-convert`**).
4. 모니터링 탭을 선택합니다.
5. 지표(Metrics)에서 Lambda 함수의 런타임 지표를 참조하십시오.
6. 로그(Logs)에서, CloudWatch Logs 인사이트를 통해 각 Lambda 호출에 대한 로그 데이터를 봅니다.

> **Note**
>
> Lambda 함수와 함께 S3 배치 작업을 사용하면 각 객체에서 Lambda 함수가 호출됩니다. S3 배치 작업이 큰 경우 동시에 여러 Lambda 함수를 호출하여 Lambda 동시성이 급증할 수 있습니다.
> 각 AWS 계정은(는) 리전당 Lambda 동시성 할당량이 있습니다. 자세한 내용은 AWS Lambda개발자 안내서의 AWS Lambda함수 크기 조정을 참조하십시오. S3 배치 작업에 Lambda 함수를 사용하는 모범 사례는 Lambda 함수 자체에 동시성 제한을 설정하는 것입니다. 동시성 제한을 설정하면 작업에서 대부분의 Lambda 동시성을 소비하여 잠재적으로 계정의 다른 함수를 제한하는 상황을 방지할 수 있습니다. 자세한 내용은 AWS Lambda개발자 안내서의 Lambda 예약된 동시성 관리 단원을 참조하십시오.

(선택 사항) MediaConvert 콘솔에서 각 MediaConvert 비디오 트랜스코딩 작업 모니터링

MediaConvert 작업은 미디어 파일을 트랜스코딩하는 작업을 수행합니다. S3 Batch Operations 작업에서 각 비디오에 대해 Lambda 함수를 호출하면 각 Lambda 함수 호출은 각 입력 비디오에 대해 MediaConvert 트랜스코딩 작업을 만듭니다.

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/mediaconvert/에서 MediaConvert 콘솔을 엽니다.
2. MediaConvert 소개 페이지가 나타나면 시작하기(Get Started)를 선택합니다.
3. 작업(Jobs) 목록에서 각 행을 확인하여 각 입력 비디오의 트랜스코딩 작업을 모니터링합니다.
4. 확인하려는 작업의 행을 식별하고 작업 ID 링크를 선택하여 작업 세부 정보 페이지를 엽니다.
5. 작업 요약(Job summary) 페이지의 출력(Outputs)에서, 브라우저에서 지원되는 항목에 따라 HLS, MP4 또는 썸네일 출력에 대한 링크를 선택하여 출력 미디어 파일의 S3 대상 버킷으로 이동합니다.
6. S3 출력 대상 버킷의 해당 폴더(HLS, MP4 또는 썸네일)에서 출력 미디어 파일 객체의 이름을 선택합니다.

   객체 세부 정보 페이지가 열립니다.
7. 객체 세부 정보 페이지의 객체 개요에서 객체 URL의 링크를 선택하여 트랜스코딩된 출력 미디어 파일을 봅니다.

8단계: S3 대상 버킷의 출력 미디어 파일 확인

S3 대상 버킷의 출력 미디어 파일을 확인하려면 다음을 수행합니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 목록에서, 1단계에서 생성한 출력 미디어 파일에 대한 S3 대상 버킷의 이름을 선택합니다(예: *amzn-s3-demo-destination-bucket1*).
4. 객체(Objects) 탭에서, 각 입력 비디오에는 입력 비디오의 이름이 있는 폴더가 있습니다. 각 폴더에는 입력 비디오의 트랜스코딩된 출력 미디어 파일이 포함되어 있습니다.

   입력 비디오의 출력 미디어 파일을 확인하려면 다음을 수행합니다.

   a. 확인하려는 입력 비디오의 이름이 있는 폴더를 선택합니다.
   b. Default/ 폴더를 선택합니다.
   c. 트랜스코딩된 형식의 폴더(이 자습서의 HLS, MP4 또는 썸네일)를 선택합니다.
   d. 출력 미디어 파일의 이름을 선택합니다.
   e. 객체 세부 정보 페이지에서 트랜스코딩된 파일을 보려면 객체 URL의 링크를 선택합니다.

      HLS 형식의 출력 미디어 파일은 짧은 세그먼트로 분할됩니다. 이 비디오를 재생하려면, 호환 플레이어에 `.m3u8` 파일의 객체 URL을 포함합니다.

9단계: 정리

S3 배치 작업, Lambda 및 MediaConvert를 학습 연습으로만 사용하여 비디오를 트랜스코딩한 경우 더 이상 요금이 발생하지 않도록 할당한 AWS 리소스를 삭제합니다.

하위 단계

- S3 소스 버킷의 S3 인벤토리 구성 삭제
- Lambda 함수를 삭제하려면
- CloudWatch 로그 그룹 삭제
- IAM 역할에 대한 인라인 정책과 함께 IAM 역할 삭제
- 고객 관리형 IAM 정책 삭제
- S3 버킷 비우기

- S3 버킷 삭제

S3 소스 버킷의 S3 인벤토리 구성 삭제

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷(Buckets) 목록에서 소스 버킷의 이름을 선택합니다(예: *amzn-s3-demo-source-bucket*).
4. [Management] 탭을 선택한 후
5. 인벤토리 구성 섹션에서, 5단계에서 생성한 인벤토리 구성 옆의 옵션 버튼을 선택합니다(예: **tutorial-inventory-config**).
6. 삭제(Delete)를 선택한 후 확인(Confirm)을 선택합니다.

Lambda 함수를 삭제하려면

1. https://console.aws.amazon.com/lambda/에서 AWS Lambda 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 함수(Functions)를 선택합니다.
3. 4단계에서 생성한 함수 옆의 확인란을 선택합니다(예: **tutorial-lambda-convert**).
4. 작업을 선택한 후 삭제를 선택합니다.
5. 함수 삭제(Delete function) 대화 상자에서 삭제(Delete)를 선택합니다.

CloudWatch 로그 그룹 삭제

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 로그를 선택한 다음, 로그 그룹을 선택합니다.
3. 4단계에서 생성한 Lambda 함수로 끝나는 이름의 로그 그룹 옆에 있는 확인란을 선택합니다(예: **tutorial-lambda-convert**).
4. 작업(Actions)을 선택한 후 로그 그룹 삭제(Delete log group(s))를 선택합니다.
5. 로그 그룹 삭제(Delete log group(s)) 대화 상자에서 삭제(Delete)를 선택합니다.

IAM 역할에 대한 인라인 정책과 함께 IAM 역할 삭제

2단계, 3단계 및 6단계에서 생성한 IAM 역할을 삭제하려면 다음을 수행합니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 역할을 선택한 후 삭제할 역할 이름 옆에 있는 확인란을 선택합니다.
3. 페이지 상단에서 삭제(Delete)를 선택합니다.
4. 확인 대화 상자에서, 안내 메시지에 따라 필요한 응답을 텍스트 입력 필드에 입력한 후 삭제(Delete)를 선택합니다.

고객 관리형 IAM 정책 삭제

6단계에서 생성한 고객 관리형 IAM 정책을 삭제하려면 다음을 수행합니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 정책을 선택합니다.
3. 6단계에서 생성한 정책 옆에 있는 옵션 버튼을 선택합니다(예: **`tutorial-s3batch-policy`**). 검색 상자를 사용하여 정책 목록을 필터링할 수 있습니다.
4. 작업을 선택한 후 삭제를 선택합니다.
5. 텍스트 필드에 해당 이름을 입력하여 이 정책을 삭제할 것인지 확인한 후 삭제를 선택합니다.

S3 버킷 비우기

사전 조건, 1단계 및 5단계에서 생성한 S3 버킷을 비우려면 다음을 수행합니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 목록에서, 비우려는 버킷의 이름 옆에 있는 옵션 버튼을 선택한 후 비우기를 선택합니다.
4. 버킷 비우기 페이지에서 텍스트 필드에 **`permanently delete`**를 입력하여 해당 버킷 비우기를 확인한 후 비우기를 선택합니다.

S3 버킷 삭제

사전 조건, 1단계 및 5단계에서 생성한 S3 버킷을 삭제하려면 다음을 수행합니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.

2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 목록에서, 삭제할 버킷의 이름 옆에 있는 옵션 버튼을 선택합니다.
4. 삭제를 선택합니다.
5. 버킷 삭제(Delete bucket) 페이지의 텍스트 필드에 버킷 이름을 입력하여 버킷의 삭제 여부를 확인한 다음 버킷 삭제(Delete bucket)를 선택합니다.

다음 단계

이 자습서를 완료한 후 다른 관련 사용 사례를 자세히 살펴볼 수 있습니다.

- Amazon CloudFront를 사용하여 트랜스코딩된 미디어 파일을 전 세계 뷰어에게 스트리밍할 수 있습니다. 자세한 내용은 자습서: Amazon S3, Amazon CloudFront 및 Amazon Route 53로 온디맨드 스트리밍 비디오 호스팅 섹션을 참조하세요.
- 동영상을 S3 소스 버킷에 업로드하는 순간 트랜스코딩할 수 있습니다. 이렇게 하려면 Lambda 함수를 자동으로 호출하여 S3의 새 객체를 MediaConvert로 트랜스코딩하는 Amazon S3 이벤트 트리거를 구성할 수 있습니다. 자세한 내용은 AWS Lambda 개발자 가이드의 자습서: Amazon S3 트리거를 사용하여 Lambda 함수 호출을 참조하십시오.

Batch Operations 문제 해결

다음 주제에는 Amazon S3 Batch Operations 사용 중에 발생할 수 있는 문제를 해결하는 데 도움이 되는 일반적인 오류가 나열되어 있습니다.

S3 배치 복제와 관련된 문제를 해결하려면 the section called “배치 복제 오류” 섹션을 참조하세요.

일반적인 오류

- 권한 문제가 있거나 S3 Object Lock 보존 모드가 활성화된 경우 작업 보고서가 전달되지 않음
- 400 InvalidRequest: VersionId를 찾을 수 없어 태스크 실패 오류와 함께 Batch Operations 객체 실패
- 작업 태그 옵션이 활성화된 상태에서 작업 생성 실패
- 매니페스트 읽기에 대한 액세스 거부

권한 문제가 있거나 S3 Object Lock 보존 모드가 활성화된 경우 작업 보고서가 전달되지 않음

대상 버킷에 필요한 권한이 누락되었거나 대상 버킷에서 Object Lock 보존 모드(거버넌스 모드 또는 규정 준수 모드)가 활성화되어 있는 경우 다음 오류가 발생합니다.

오류: 실패 이유. 작업 보고서를 보고서 버킷에 쓸 수 없습니다. 권한을 확인하세요.

보고서가 전달될 버킷의 PUT 객체에 대한 Batch Operations의 `s3:PutObject` 권한을 허용하도록 AWS Identity and Access Management(IAM) 역할 및 신뢰 정책을 구성해야 합니다. 이러한 필수 권한이 누락되면 작업 보고서 전달에 실패합니다.

보존 모드가 활성화된 경우 버킷이 write-once-read-many(WORM)로 보호됩니다. 대상 버킷에서 보존 모드가 활성화된 Object Lock이 지원되지 않으므로, 작업 완료 보고서 전달 시도가 실패합니다. 이 문제를 해결하려면 작업 완료 보고서를 위해 객체 잠금 보존 모드가 활성화되지 않은 대상 버킷을 선택하세요.

400 InvalidRequest: VersionId를 찾을 수 없어 태스크 실패 오류와 함께 Batch Operations 객체 실패

다음 예시 오류는 배치 작업이 버전이 지정된 버킷의 객체에 대해 작업을 수행하다가 매니페스트에서 버전 ID 필드가 비어 있는 객체를 발견하면 발생합니다.

오류: *`bucket_name,prefix/file_name`*,failed,400,InvalidRequest,VersionId를 찾을 수 없어 태스크 실패

이 오류는 매니페스트의 버전 ID 필드가 리터럴 `null` 문자열이 아니라 빈 문자열이기 때문에 발생합니다.

배치 작업은 전체 작업이 아니라 특정 객체에 대해 실패합니다. 이 문제는 작업 중에 매니페스트 형식이 버전 ID를 사용하도록 구성된 경우 발생합니다. 버전이 지정되지 않은 작업은 각 객체의 최신 버전에서만 작동하고 매니페스트의 버전 ID를 무시하므로 이 문제가 발생하지 않습니다.

이 문제를 해결하려면 빈 버전 ID를 `null` 문자열로 변환하세요. 자세한 내용은 the section called “빈 버전 ID 문자열을 null 문자열로 변환” 섹션을 참조하세요.

작업 태그 옵션이 활성화된 상태에서 작업 생성 실패

`s3:PutJobTagging` 권한 없이 작업 태그 옵션을 활성화한 상태로 배치 작업을 만들면 `403 access denied` 오류가 발생합니다.

작업 태그 옵션이 활성화된 상태에서 배치 작업을 생성하려면 배치 작업을 생성하는 AWS Identity and Access Management(IAM) 사용자에게 `s3:CreateJob` 권한 외에도 `s3:PutJobTagging` 권한이 있어야 합니다.

배치 작업에 필요한 권한에 대한 자세한 내용은 the section called “권한 부여” 섹션을 참조하세요.

매니페스트 읽기에 대한 액세스 거부

배치 작업을 만들려고 할 때 배치 작업에서 매니페스트 파일을 읽을 수 없는 경우 다음 오류가 발생할 수 있습니다.

AWS CLI

실패 이유. 매니페스트 읽기가 금지됨: AccessDenied

Amazon S3 콘솔

경고: 매니페스트 객체의 ETag를 가져올 수 없습니다. 계속하려면 다른 객체를 지정하세요.

이 문제를 해결하려면 다음을 수행하세요.

- Batch Operations 작업을 만드는 데 사용한 AWS 계정의 IAM 역할에 `s3:GetObject` 권한이 있는지 확인합니다. 계정의 IAM 역할에는 Batch Operations에서 매니페스트 파일을 읽을 수 있는 `s3:GetObject` 권한이 있어야 합니다.

  배치 작업에 필요한 권한에 대한 자세한 내용은 the section called “권한 부여” 섹션을 참조하세요.
- 매니페스트 객체의 메타데이터에 S3 객체 소유권과 액세스 불일치가 있는지 확인합니다. S3 객체 소유권에 대한 자세한 내용은 the section called “객체 소유권 제어” 섹션을 참조하세요.
- AWS Key Management Service(AWS KMS) 키가 매니페스트 파일을 암호화하는 데 사용되는지 확인합니다.

  Batch Operations는 AWS KMS로 암호화된 CSV 인벤토리 보고서를 지원합니다. 그러나 Batch Operations는 AWS KMS로 암호화된 CSV 매니페스트 파일을 지원하지 않습니다. 자세한 내용은 Amazon S3 인벤토리 구성 및 매니페스트 지정(을)를 참조하세요.

Amazon S3 Select를 사용하여 데이터 쿼리 적용

> **Important**
>
> 신규 고객은 더 이상 Amazon S3 Select를 사용할 수 없습니다. Amazon S3 Select 기존 고객은 평소처럼 이 기능을 계속 사용할 수 있습니다. 자세히 알아보기

Amazon S3 Select에서는 구조화 질의 언어(SQL) 문을 사용하여 Amazon S3 객체의 콘텐츠를 필터링하고 필요한 데이터의 하위 집합만 검색할 수 있습니다. Amazon S3 Select를 사용하여 이 데이터를 필

터링하면 Amazon S3가 전송하는 데이터의 양을 줄일 수 있으며 이 데이터를 검색하는 데 드는 비용과 지연 시간이 감소됩니다.

Amazon S3 Select에서는 한 번에 하나의 객체만 쿼리할 수 있습니다. 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 Select에 대한 SQL 참조 단원을 참조하십시오.

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), `SelectObjectContent` REST API 작업 또는 AWS SDK를 사용하여 SQL 쿼리를 수행할 수 있습니다.

> **Note**
>
> 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.
- 쿼리하려는 객체가 고객 제공 키(SSE-C)를 사용한 서버 측 암호화로 암호화되어 있는 경우 `https`를 사용해야 하며, 요청에서 암호화 키를 제공해야 합니다.

Amazon S3 Select를 사용할 때는 다음 제한이 적용됩니다.

- S3 Select에서는 요청당 하나의 객체만 쿼리할 수 있습니다.
- SQL 표현식의 최대 길이는 256KB입니다.
- 입력 또는 결과에 표시되는 레코드의 최대 길이는 1MB입니다.
- Amazon S3 Select는 JSON 출력 형식을 사용하여 중첩 데이터만 내보낼 수 있습니다.
- S3 Glacier Flexible Retrieval, S3 Glacier Deep Archive 또는 Redundancy Reduction Storage(RRS) 스토리지 클래스에 저장된 객체를 쿼리할 수 없습니다. 또한 S3 Intelligent-Tiering Archive Access

계층 또는 S3 Intelligent-Tiering Deep Archive Access 계층에 저장된 객체를 쿼리할 수 없습니다. 스토리지 클래스에 대한 자세한 정보는 Amazon S3 스토리지 클래스 이해 및 관리를 참조하십시오.

Amazon S3 Select를 Parquet 객체와 함께 사용할 경우 다음과 같은 추가 제한 사항이 적용됩니다.

- 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 512MB입니다.
- 객체의 스키마에 지정되어 있는 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 요청을 구성하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service API 참조에서 SelectObjectContent를 참조하세요. 또한, 다음 단원에서 SDK 코드의 예시 중 하나를 확인할 수 있습니다.

스캔 범위를 사용하여 요청

Amazon S3 Select를 이용하면 쿼리할 바이트의 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 병렬로 처리할 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 해당 스캔 범위를 넘어서는 레코드는 쿼리에 의해 처리됩니다. 예를 들어 다음은 줄로 구분된 CSV 형식으로 된 일련의 레코드를 포함하는 Amazon S3 객체를 보여줍니다.

```
A,B
C,D
```

```
D,E
E,F
G,H
I,J
```

Amazon S3 Select ScanRange 파라미터를 사용하여 (바이트) 1에서 시작하고 (바이트) 4에서 종료한다고 가정해 보겠습니다. 스캔 범위는 ","에서 시작하여 C로 시작하는 레코드가 끝날 때까지 스캔합니다. 스캔 범위 요청은 레코드가 끝나는 지점이기 때문에 결과 C, D를 반환합니다.

Amazon S3 Select 스캔 범위 요청은 Parquet, CSV(따옴표로 묶은 구분자 사용하지 않음) 또는 JSON 객체(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 AWS CLI, Amazon S3 API 및 AWS SDK에서 사용할 수 있습니다. 이 기능을 위한 Amazon S3 Select 요청의 ScanRange 파라미터를 사용할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 SelectObjectContent를 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 참조에서 오류 응답 페이지의 SELECT 객체 콘텐츠 오류 코드 목록 단원을 참조하십시오.

Amazon S3 Select에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- 객체에 Amazon S3 Select를 사용하는 예시
- Amazon S3 Select에 대한 SQL 참조

객체에 Amazon S3 Select를 사용하는 예시

> **Important**
>
> 신규 고객은 더 이상 Amazon S3 Select를 사용할 수 없습니다. Amazon S3 Select 기존 고객은 평소처럼 이 기능을 계속 사용할 수 있습니다. 자세히 알아보기

Amazon S3 콘솔, REST API 및 AWS SDK를 통해 S3 Select를 사용하여 단일 객체에서 콘텐츠를 선택할 수 있습니다.

S3 Select에서 지원되는 SQL 함수에 대한 자세한 내용은 SQL 함수 섹션을 참조하세요.

S3 콘솔 사용

Amazon S3 콘솔의 객체에서 콘텐츠를 선택하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 콘텐츠를 선택하려는 객체를 포함하는 버킷을 선택한 다음 객체의 이름을 선택합니다.
4. 객체 작업을 선택하고 S3 Select에서 쿼리를 선택합니다.
5. 입력 데이터의 형식에 따라 입력 설정을 구성합니다.
6. 수신하려는 출력 형식에 따라 출력 설정을 구성합니다.
7. 선택한 객체에서 레코드를 추출하려면 SQL 쿼리에서 SELECT SQL 명령을 입력합니다. SQL 명령 작성 방법에 대한 자세한 내용은 Amazon S3 Select에 대한 SQL 참조를 참조하세요.
8. SQL 쿼리를 입력한 후 SQL 쿼리 실행을 선택합니다. 그런 다음 쿼리 결과에서 SQL 쿼리 결과를 확인할 수 있습니다.

REST API 사용

AWS SDK를 사용하여 한 객체에서 콘텐츠를 선택할 수 있습니다. 하지만 애플리케이션에서 요구할 경우 REST 요청을 직접 전송할 수 있습니다. 요청 및 응답 형식에 대한 자세한 내용은 SelectObjectContent를 참조하세요.

AWS SDK 사용

Amazon S3 Select를 통해 `selectObjectContent` 메서드를 사용하여 한 객체의 일부 콘텐츠를 선택할 수 있습니다. 이 메서드가 성공하면 SQL 표현식의 결과를 반환합니다.

Java

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 AWS SDK for Java 개발자 안내서의 시작하기 섹션을 참조하세요.

```
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;
    }
}
```

JavaScript

S3 `SelectObjectContent` API와 함께 AWS SDK for JavaScript를 사용하여 Amazon S3에 저장된 JSON 및 CSV 파일에서 레코드를 선택하는 JavaScript 예제는 블로그 게시물 AWS SDK for JavaScript에서 Amazon S3 Select에 대한 지원 소개를 참조하세요.

Python

S3 Select를 사용하여 Amazon S3에 로드된 데이터를 쉼표로 구분된 값(CSV) 파일로 검색하기 위해 SQL 쿼리를 사용하는 Python 예제에 대해서는 블로그 게시물 Amazon S3 Select를 사용하여 서버나 데이터베이스 없이 데이터 쿼리를 참조하세요.

Amazon S3 Select에 대한 SQL 참조

> **Important**
>
> 신규 고객은 더 이상 Amazon S3 Select를 사용할 수 없습니다. Amazon S3 Select 기존 고객은 평소처럼 이 기능을 계속 사용할 수 있습니다. 자세히 알아보기

이 참조는 Amazon S3 Select에서 지원하는 구조화 질의 언어(SQL) 요소들에 대한 설명을 포함합니다.

주제

- SELECT 명령
- 데이터 타입
- 연산자
- 예약어
- SQL 함수

SELECT 명령

> **Important**
>
> 신규 고객은 더 이상 Amazon S3 Select를 사용할 수 없습니다. Amazon S3 Select 기존 고객은 평소처럼 이 기능을 계속 사용할 수 있습니다. 자세히 알아보기

Amazon S3 Select는 SELECT SQL 명령만 지원합니다. SELECT에서 지원되는 ANSI 표준 절은 다음과 같습니다.

- SELECT 계정의
- FROM 절
- WHERE 절
- LIMIT 절

> **Note**
>
> Amazon S3 Select 쿼리는 현재 하위 쿼리 또는 조인을 지원하지 않습니다.

SELECT 계정의

SELECT 목록은 쿼리에서 반환하려는 열, 함수 및 표현식의 이름을 지정합니다. 목록은 쿼리의 출력을 나타냅니다.

```
SELECT *
SELECT projection1 AS column_alias_1, projection2 AS column_alias_2
```

*(별표)가 있는 SELECT의 첫 번째 형식은 WHERE 절을 통과한 모든 행을 있는 그대로 반환합니다. SELECT의 두 번째 형식은 각 열에 대해 사용자 정의 출력 스칼라 표현식 ***projection1*** 및 ***projection2***를 사용하여 행을 생성합니다.

FROM 절

Amazon S3 Select에서는 다음과 같은 형식의 FROM 절을 지원합니다.

```
FROM table_name
FROM table_name alias
FROM table_name AS alias
```

FROM 절의 각 형식에서 table_name은 쿼리되는 S3Object입니다. 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 객체의 경우 다음과 같은 형식의 FROM 절을 사용할 수도 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 path를 지정할 수 있습니다.

- 이름별(객체 내): .*name* 또는 ['*name*']

- 인덱스로(배열 내): [*index*]
- 와일드카드 문자(객체 내): .*
- 와일드카드 문자(배열 내): [*]

> **Note**
>
> - 이 양식의 FROM 절은 JSON 객체에만 사용할 수 있습니다.
> - 와일드카드 문자는 항상 하나 이상의 레코드를 출력합니다. 일치하는 레코드가 없을 경우 Amazon S3 Select가 MISSING 값을 출력합니다. 출력 직렬화 도중(쿼리 실행이 완료된 후) Amazon S3 Select는 MISSING 값을 빈 레코드로 대체합니다.
> - 집계 함수(AVG, COUNT, MAX, MIN 및 SUM)는 MISSING 값을 건너뜁니다.
> - 와일드카드 문자를 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 이름 대신 와일드카드 문자로 끝날 경우 값 _1을 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 JSON 문서를 항상 루트 수준 값의 배열로 취급합니다. 따라서 쿼리하는 JSON 객체가 하나의 루트 요소만 가지더라도 FROM 절이 S3Object[*]로 시작해야 합니다. 하지만 사용자가 경로를 포함하지 않을 경우 Amazon S3 Select는 호환성을 이유로 와일드카드 문자를 생략하도록 허용합니다. 따라서 전체 절 FROM S3Object는 FROM S3Object[*] as S3Object와 같습니다. 경로를 포함하는 경우에는 와일드카드 문자도 사용해야 합니다. 그러므로 FROM S3Object 및 FROM S3Object[*].*path*는 모두 유효한 절이지만, FROM S3Object.*path*는 유효하지 않습니다.

Example

예:

예제 #1

이 예제에서는 다음 데이터 세트 및 쿼리를 사용할 때의 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"}` – `S3Object[0].Rules[0].id`가 일치하는 레코드를 생성했습니다.
- `{}` – `S3Object[0].Rules[1].id`는 일치하는 레코드가 없으므로 Amazon S3 Select가 `MISSING`을 출력했습니다. 이 값은 출력 직렬화 과정에서 빈 레코드로 변환 후 반환되었습니다.
- `{"id":"id-2"}` – `S3Object[0].Rules[2].id`가 일치하는 레코드를 생성했습니다.
- `{}` – `S3Object[1]`는 `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 않도록 값 `MISSING`에 대해 테스트를 할 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 사용할 때의 결과를 보여 줍니다.

```
{ "created": "936864000", "dir_name": "important_docs", "files": [ { "name": "." },
 { "name": ".." }, { "name": ".aws" }, { "name": "downloads" } ], "owner": "Amazon
 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":"Amazon S3"}
{"dir_name":"other_docs","owner":"User"}
```

WHERE 절

WHERE 절은 다음 구문에 따릅니다.

```
WHERE condition
```

WHERE 절은 *condition*에 기반하여 행을 필터링합니다. 조건은 부울 결과가 있는 표현식입니다. 조건이 TRUE로 평가되는 행만이 결과에서 반환됩니다.

LIMIT 절

LIMIT 절은 다음 구문에 따릅니다.

```
LIMIT number
```

LIMIT 절은 쿼리가 반환하게 할 레코드의 수를 *number*에 기초하여 제한합니다.

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 *_N*으로 특정 행의 N번째 열을 참조할 수 있습니다(여기서 *N*은 열의 위치입니다). 위치 카운트는 1에서 시작합니다. 예를 들어, 첫 번째 열의 이름은 _1이고, 두 번째 열의 이름은 _2입니다.

  열을 *_N* 또는 *alias*.*_N*이라고 부를 수 있습니다. 예를 들어, _2와 myAlias._2는 모두 SELECT 목록과 WHERE 절에 있는 열을 참조하는 유효한 방법입니다.

- 열 머리글 – 머리글 행이 있는 CSV 형식의 객체들은 그 머리글이 SELECT 목록과 WHERE 절에 제공됩니다. 특히 기존의 SQL과 같이, SELECT와 WHERE 절 표현식에서 *alias*.*column_name* 또는 *column_name*으로 열을 참조할 수 있습니다.

JSON

- 문서 – *alias*.*name*으로 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. 액세스 목록 요소를 필드와 결합할 수 있습니다 (예: *alias*.*name1*.*name2*[1].*name3*).
- 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 및 속성 이름의 대/소문자 구분

Amazon S3 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 및 속성이 대/소문자를 구분하지 않는 것입니다. 모호한 경우에는 오류가 발생합니다.

다음 예제는 1) 지정된 열 헤더가 있고 쿼리 요청에 대해 `FileHeaderInfo`가 `""Use""`으로 설정된 CSV 형식의 Amazon S3 객체 또는 2) 지정된 속성이 있는 JSON 형식의 Amazon S3 객체입니다.

예제 #1: 쿼리되는 객체가 머리글 또는 속성 `NAME`을 갖습니다.

- 다음 표현식은 객체에서 값을 성공적으로 반환합니다. 따옴표가 없으므로 쿼리는 대소문자를 구분하지 않습니다.

```
SELECT s.name from S3Object s
```

- 다음 표현식은 400 오류 `MissingHeaderName`을 발생시킵니다. 따옴표가 있으므로 쿼리는 대소문자를 구분합니다.

```
SELECT s."name" from S3Object s
```

예제 #2: 쿼리되는 Amazon S3 객체에 `NAME`이 있는 머리글 또는 속성이 하나 있고 `name`이 있는 머리글 또는 속성이 또 있습니다.

- 다음 표현식은 400 오류 `AmbiguousFieldName`을 발생시킵니다. 따옴표가 없으므로 대소문자를 구분하지 않는 쿼리지만 일치하는 항목이 두 개 있으므로 오류가 발생합니다.

```
SELECT s.name from S3Object s
```

- 다음 표현식은 객체에서 값을 성공적으로 반환합니다. 따옴표가 있어 쿼리는 대/소문자를 구분하므로 모호하지 않습니다.

```
SELECT s."NAME" from S3Object s
```

예약어를 사용자 정의 용어로 사용

Amazon S3 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) 또는 속성(JSON 객체의 경우)과 같은 사용자 정의 용어가 예약어와 충돌

할 수 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 단원을 참조하세요.

다음 예제는 1) 열 머리글이 지정되어 있고 쿼리 요청에 대해 `FileHeaderInfo`가 `"Use"`으로 설정되어 있는 CSV 형식의 Amazon S3 객체 또는 2) 속성이 지정된 JSON 형식의 Amazon S3 객체 중 하나입니다.

예제: 쿼리되는 객체에는 예약어인 `CAST`라는 헤더 또는 속성이 있습니다.

- 다음 표현식은 객체에서 값을 성공적으로 반환합니다. 쿼리에 따옴표가 사용되므로 S3 Select는 사용자 정의 머리글 또는 속성을 사용합니다.

```
SELECT s."CAST" from S3Object s
```

- 다음 표현식은 400 오류 을 발생시킵니다. 쿼리에는 따옴표가 사용되지 않으므로 예약된 키워드와 `CAST`가 충돌합니다.

```
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 참조입니다.

- ***unary_op expression***

  이 경우 ***unary_op***는 SQL 단항 연산자입니다.

- ***expression binary_op expression***

  이 경우 ***binary_op***는 SQL 이항 연산자입니다.

- ***func_name***

이 경우 ***func_name***은 호출할 스칼라 함수의 이름입니다.

- ***expression*** [NOT] BETWEEN ***expression*** AND ***expression***
- ***expression*** LIKE ***expression*** [ESCAPE ***expression***]

데이터 타입

> **Important**
>
> 신규 고객은 더 이상 Amazon S3 Select를 사용할 수 없습니다. Amazon S3 Select 기존 고객은 평소처럼 이 기능을 계속 사용할 수 있습니다. 자세히 알아보기

Amazon S3 Select에서는 몇 가지 기본 데이터 형식을 지원합니다.

데이터 형식 변환

일반적 규칙은 CAST 함수에 따르는 것입니다(정의된 경우). CAST를 정의하지 않는 경우, 모든 입력 데이터를 문자열로 취급합니다. 이 경우 필요한 경우 입력 데이터를 관련 데이터 유형으로 캐스팅해야 합니다.

CAST 함수에 대한 자세한 내용은 CAST 단원을 참조하십시오.

지원되는 데이터 유형

Amazon S3 Select에서는 다음과 같은 일련의 기본 데이터 형식들을 지원합니다.

명칭	설명	예시
bool	부울 값으로 TRUE 또는 FALSE 둘 중 하나입니다.	FALSE
int, integer	9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807 범위의 8바이트 부호 있는 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)

명칭	설명	예시
`decimal`, `numeric`	최대 정밀도(최대 유효 숫자 수)가 38이고, 스케일 범위가 -2^{31}~2^{31}-1(십진수 지수)인 10진수입니다. **Note** 사용자가 그 두 가지를 동시에 제공하면 Amazon S3 Select는 스케일과 정밀도를 무시합니다.	`123.456`
`timestamp`	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)`

지원되는 Parquet 유형

Amazon S3 Select는 다음과 같은 Parquet 유형을 지원합니다.

- `DATE`
- `DECIMAL`
- `ENUM`
- `INT(8)`
- `INT(16)`
- `INT(32)`
- `INT(64)`
- `LIST`

> **Note**
>
> LIST Parquet 유형 출력의 경우, Amazon S3 Select는 JSON 형식만 지원합니다. 단, 쿼리에서 데이터를 단순 값으로 제한하는 경우 LIST Parquet 유형도 CSV 형식으로 쿼리할 수 있습니다.

- STRING
- TIMESTAMP 지원되는 정밀도(MILLIS/MICROS/NANOS)

> **Note**
>
> INT(96)로 저장된 타임스탬프는 지원되지 않습니다.
> INT(64) 유형의 범위로 인해 NANOS 단위를 사용하는 타임스탬프는 1677-09-21 00:12:43과 2262-04-11 23:47:16 사이의 값만 나타낼 수 있습니다. 이 범위를 벗어난 값은 NANOS 단위로 나타낼 수 없습니다.

Amazon S3 Select에서 Parquet 유형을 지원되는 데이터 유형에 매핑

Parquet 유형	지원되는 데이터 유형
DATE	timestamp
DECIMAL	decimal, numeric
ENUM	string
INT(8)	int, integer
INT(16)	int, integer
INT(32)	int, integer
INT(64)	decimal, numeric

Parquet 유형	지원되는 데이터 유형
LIST	목록의 각 Parquet 유형은 해당 데이터 유형에 매핑됩니다.
STRING	string
TIMESTAMP	timestamp

연산자

> **Important**
>
> 신규 고객은 더 이상 Amazon S3 Select를 사용할 수 없습니다. Amazon S3 Select 기존 고객은 평소처럼 이 기능을 계속 사용할 수 있습니다. 자세히 알아보기

Amazon S3 Select에서는 다음과 같은 연산자들을 지원합니다.

논리 연산자

- AND
- NOT
- OR

비교 연산자

- <
- >
- <=
- >=
- =
- <>
- !=

- BETWEEN
- IN예:IN ('a', 'b', 'c').

패턴 일치 연산자

- LIKE
- _(모든 문자를 일치시킵니다.)
- %(모든 문자 시퀀스를 일치시킵니다.)

단일 연산자

- IS NULL
- IS NOT NULL

수학 연산자

다음과 같이 더하기, 빼기, 곱하기, 나누기 및 모듈로가 지원됩니다.

- +
- -
- *
- /
- %

연산자 우선순위

다음 표에서는 연산자들의 우선 순위를 내림차순으로 보여줍니다.

연산자 또는 원소	연결성	필수
-	오른쪽	단항 마이너스
*, /, %	왼쪽	곱하기, 나누기, 모듈로

연산자 또는 원소	연결성	필수
+, -	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
OR	왼쪽	논리 분리

예약어

> **Important**
>
> 신규 고객은 더 이상 Amazon S3 Select를 사용할 수 없습니다. Amazon S3 Select 기존 고객은 평소처럼 이 기능을 계속 사용할 수 있습니다. 자세히 알아보기

다음은 Amazon S3 Select에 대한 예약 키워드 목록입니다. 이러한 키워드에는 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

> **Important**
>
> 신규 고객은 더 이상 Amazon S3 Select를 사용할 수 없습니다. Amazon S3 Select 기존 고객은 평소처럼 이 기능을 계속 사용할 수 있습니다. 자세히 알아보기

Amazon S3 Select는 다음 SQL 함수를 지원합니다.

주제

- 집계 함수
- 조건 함수
- 변환 함수
- 날짜 함수
- 문자열 함수

집계 함수

> **Important**
>
> 신규 고객은 더 이상 Amazon S3 Select를 사용할 수 없습니다. Amazon S3 Select 기존 고객은 평소처럼 이 기능을 계속 사용할 수 있습니다. 자세히 알아보기

Amazon S3 Select는 다음과 같은 집계 함수를 지원합니다.

함수	인수 유형	반환 타입
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.

SUM 예제

S3 Inventory 보고서에서 폴더의 총 객체 크기를 집계하려면 SUM 표현식을 사용하세요.

다음 S3 Inventory 보고서는 GZIP으로 압축된 CSV 파일입니다. 세 개의 열이 있습니다.

- 첫 번째 열은 S3 Inventory 보고서의 대상인 S3 버킷(*DOC-EXAMPLE-BUCKET*)의 이름입니다.
- 두 번째 열은 버킷의 객체를 고유하게 식별하는 객체 키 이름입니다.

  첫 번째 행의 *example-folder/* 값은 *example-folder* 폴더에 대한 값입니다. Amazon S3에서 버킷에 폴더를 만들면 S3는 제공한 폴더 이름으로 설정된 키가 있는 0바이트의 객체를 생성합니다.

  두 번째 행의 *example-folder/object1* 값은 *example-folder* 폴더에 있는 *object1* 객체에 대한 값입니다.

  세 번째 행의 *example-folder/object2* 값은 *example-folder* 폴더에 있는 *object2* 객체에 대한 값입니다.

  폴더에 대한 자세한 내용은 Amazon S3 콘솔에서 폴더를 사용하여 객체 구성을 참조하세요.
- 세 번째 열은 객체 크기(바이트)입니다.

```
"DOC-EXAMPLE-BUCKET","example-folder/","0"
"DOC-EXAMPLE-BUCKET","example-folder/object1","2011267"
"DOC-EXAMPLE-BUCKET","example-folder/object2","1570024"
```

SUM 표현식을 사용하여 *example-folder* 폴더의 전체 크기를 계산하려면 Amazon S3 Select에서 SQL 쿼리를 실행합니다.

```
SELECT SUM(CAST(_3 as INT)) FROM s3object s WHERE _2 LIKE 'example-folder/%' AND _2 !=
 'example-folder/';
```

쿼리 결과:

```
3581291
```

조건 함수

> **Important**
>
> 신규 고객은 더 이상 Amazon S3 Select를 사용할 수 없습니다. Amazon S3 Select 기존 고객은 평소처럼 이 기능을 계속 사용할 수 있습니다. 자세히 알아보기

Amazon S3 Select에서는 다음과 같은 조건부 함수를 지원합니다.

주제

- CASE
- COALESCE
- NULLIF

CASE

CASE 표현식은 다른 언어에서 볼 수 있는 `if/then/else` 문과 유사한 조건부 표현식입니다. CASE는 여러 조건이 있을 때 결과를 지정하는 데 사용됩니다. CASE 표현식은 단순(simple)과 검색(searched), 두 가지 유형이 있습니다.

단순 CASE 표현식에서는 표현식과 값을 비교합니다. 이때 일치하는 부분이 발견되면 THEN 절에서 지정된 작업이 적용됩니다. 일치하는 부분이 발견되지 않으면 ELSE 절에서 지정된 작업이 적용됩니다.

검색 CASE 표현식에서는 각 CASE가 부울 표현식에 따라 평가되고, CASE 문이 처음 일치하는 CASE를 반환합니다. WHEN 절 사이에서 일치하는 CASE이 발견되지 않으면 ELSE 절의 작업이 반환됩니다.

구문

> **Note**
>
> 현재 Amazon S3 Select는 ORDER BY 또는 새 줄이 포함된 쿼리를 지원하지 않습니다. 줄 바꿈 없이 쿼리를 사용해야 합니다.

다음은 조건을 일치시키는 데 사용되는 단순 CASE 문입니다.

```
CASE expression WHEN value THEN result [WHEN...] [ELSE result] END
```

다음은 각 조건을 평가하는 데 사용되는 검색 CASE 문입니다.

```
CASE WHEN boolean condition THEN result [WHEN ...] [ELSE result] END
```

예시

> **Note**
>
> Amazon S3 콘솔을 사용하여 다음 예제를 실행하고 CSV 파일에 헤더 행이 포함된 경우CSV 데이터의 첫 번째 줄 제외를 선택하세요.

예제 1: 쿼리에서 단순 CASE 표현식을 사용하여 New York City를 Big Apple로 바꿉니다. 그 밖의 도시 이름은 모두 other로 변경합니다.

```
SELECT venuecity, CASE venuecity WHEN 'New York City' THEN 'Big Apple' ELSE 'other' END
 FROM S3Object;
```

쿼리 결과:

```
venuecity        |   case
-----------------+-----------
Los Angeles      | other
New York City    | Big Apple
San Francisco    | other
Baltimore        | other
...
```

예제 2: 검색 CASE 표현식을 사용하여 개별 티켓 판매에 대한 pricepaid 값을 기준으로 그룹 번호를 할당합니다.

```
SELECT pricepaid, CASE WHEN CAST(pricepaid as FLOAT) < 10000 THEN 'group 1' WHEN
 CAST(pricepaid as FLOAT) > 10000 THEN 'group 2' ELSE 'group 3' END FROM S3Object;
```

쿼리 결과:

```
pricepaid |  case
----------+---------
12624.00 | group 2
```

```
10000.00 | group 3
10000.00 | group 3
9996.00 | group 1
9988.00 | group 1
...
```

COALESCE

COALESCE는 인수를 순서대로 평가하여 알 수 없는 첫 번째 값, 즉 첫 번째 null이 아니거나 누락되지 않은 값을 반환합니다. 이 함수는 null과 누락 값을 전파하지 않습니다.

구문

```
COALESCE ( expression, expression, ... )
```

파라미터

expression

함수가 실행되는 대상 표현식입니다.

예시

```
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 NULLIF는 NULL을 반환하고, 그렇지 않다면 NULLIF는 첫 번째 표현식을 평가한 결과를 반환합니다.

구문

```
NULLIF ( expression1, expression2 )
```

파라미터

expression1, *expression2*

함수가 실행되는 대상 표현식들입니다.

예시

```
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

> **Important**
>
> 신규 고객은 더 이상 Amazon S3 Select를 사용할 수 없습니다. Amazon S3 Select 기존 고객은 평소처럼 이 기능을 계속 사용할 수 있습니다. 자세히 알아보기

Amazon S3 Select에서는 다음과 같은 변환 함수를 지원합니다.

주제

- CAST

CAST

CAST 함수는 단일 값으로 평가되는 표현식 같은 엔터티의 형식을 다른 형식으로 변환합니다.

구문

```
CAST ( expression AS data_type )
```

파라미터

expression

하나 이상의 값, 연산자 및 값으로 평가되는 SQL 함수의 조합입니다.

data_type

INT와 같이 표현식을 캐스팅할 대상 데이터 형식입니다. 지원되는 데이터 형식의 목록은 데이터 타입 단원을 참조하십시오.

예시

```
CAST('2007-04-05T14:30Z' AS TIMESTAMP)
CAST(0.456 AS FLOAT)
```

날짜 함수

> **Important**
>
> 신규 고객은 더 이상 Amazon S3 Select를 사용할 수 없습니다. Amazon S3 Select 기존 고객은 평소처럼 이 기능을 계속 사용할 수 있습니다. 자세히 알아보기

Amazon S3 Select는 다음과 같은 날짜 함수를 지원합니다.

주제

- DATE_ADD
- DATE_DIFF
- EXTRACT
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

DATE_ADD

날짜 부분, 양, 그리고 타임스탬프가 주어질 경우 DATE_ADD는 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

quantity

업데이트된 타임스탬프에 적용되는 값입니다. 이 *quantity*가 양의 값이면 타임스탬프의 날짜 부분이 증가하고 음의 값이면 감소합니다.

timestamp

함수가 실행되는 대상 타임스탬프입니다.

예시

```
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, DATE_DIFF는 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_part의 정의에 대해서는 DATE_ADD 단원을 참조하십시오.

timestamp1

비교할 첫 번째 타임스탬프입니다.

timestamp2

비교할 두 번째 타임스탬프입니다.

예시

```
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, `2010-01-01T23:00`, `2010-01-02T01:00`) -- 0 (need to be at least 24h
 apart to be 1 day apart)
```

EXTRACT

날짜 부분 1개와 타임스탬프 1개가 주어질 경우, EXTRACT는 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.

- YEAR
- MONTH
- DAY
- HOUR
- MINUTE
- SECOND
- TIMEZONE_HOUR
- TIMEZONE_MINUTE

timestamp

함수가 실행되는 대상 타임스탬프입니다.

예시

```
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, TO_STRING은 타임스탬프의 문자열 표현을 그 형식으로 반환합니다.

구문

```
TO_STRING ( timestamp time_format_pattern )
```

파라미터

timestamp

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: 이 형식은 TO_TIMESTAMP 함수와 사용할 경우 유효하지 않음)

형식	예	설명
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)
SS	6	몇 분의 1초(정밀도: 0.01, 범위: 0.0~0.99)

형식	예	설명
SSS	60	몇 분의 1초(정밀도: 0.001, 범위: 0.0~0.999)
…	…	…
SSSSSSSSS	60000000	몇 분의 1초(최대 정밀도: 1나노초, 범위: 0.0~0.999999999)
n	60000000	나노초
X	+07 또는 Z	시간 단위로 표시한 오프셋. 오프셋이 0인 경우 Z
XX 또는 XXXX	+0700 또는 Z	시간과 분 단위로 표시한 오프셋. 오프셋이 0인 경우 Z
XXX 또는 XXXXX	+07:00 또는 Z	시간과 분 단위로 표시한 오프셋. 오프셋이 0인 경우 Z
x	7	시간 단위로 표시한 오프셋
xx 또는 xxxx	700	시간과 분 단위로 표시한 오프셋

형식	예	설명
xxx 또는 xxxxx	+07:00	시간과 분 단위로 표시한 오프셋

예시

```
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 TO_TIMESTAMP는 문자열을 타임스탬프로 변환합니다. TO_TIMESTAMP는 TO_STRING의 역연산입니다.

구문

```
TO_TIMESTAMP ( string )
```

파라미터

string

함수가 실행되는 대상 문자열입니다.

예시

```
TO_TIMESTAMP('2007T')                          -- `2007T`
TO_TIMESTAMP('2007-02-23T12:14:33.079-08:00') -- `2007-02-23T12:14:33.079-08:00`
```

UTCNOW

UTCNOW는 현재 시간(UTC)을 타임스탬프로 반환합니다.

구문

```
UTCNOW()
```

파라미터

UTCNOW는 파라미터를 사용하지 않습니다.

예시

```
UTCNOW() -- 2017-10-13T16:02:11.123Z
```

문자열 함수

> **Important**
>
> 신규 고객은 더 이상 Amazon S3 Select를 사용할 수 없습니다. Amazon S3 Select 기존 고객은 평소처럼 이 기능을 계속 사용할 수 있습니다. 자세히 알아보기

Amazon S3 Select는 다음과 같은 문자열 함수를 지원합니다.

주제

- CHAR_LENGTH, CHARACTER_LENGTH
- LOWER
- SUBSTRING
- TRIM
- UPPER

CHAR_LENGTH, CHARACTER_LENGTH

CHAR_LENGTH(또는 CHARACTER_LENGTH)는 지정된 문자열의 문자 수를 계수합니다.

> **Note**
>
> CHAR_LENGTH와 CHARACTER_LENGTH는 동의어입니다.

구문

```
CHAR_LENGTH ( string )
```

파라미터

string

함수가 실행되는 대상 문자열입니다.

예시

```
CHAR_LENGTH('')          -- 0
CHAR_LENGTH('abcdefg')   -- 7
```

LOWER

문자열이 주어지면 LOWER는 모든 대문자를 소문자로 변환합니다. 대문자로 변환되지 않는 문자는 변경되지 않고 그대로 유지됩니다.

구문

```
LOWER ( string )
```

파라미터

string

함수가 실행되는 대상 문자열입니다.

예시

```
LOWER('AbCdEfG!@#$') -- 'abcdefg!@#$'
```

SUBSTRING

문자열, 시작 인덱스, 그리고 때로는 길이가 주어질 경우 SUBSTRING은 시작 인덱스부터 그 문자열의 마지막까지, 또는 제공된 길이까지 하위 문자열을 반환합니다.

> **Note**
>
> 입력 문자열의 첫 번째 문자가 인덱스 위치가 1입니다.
>
> - `start`가 < 1 이고 길이가 지정되지 않았으면 인덱스 위치가 1로 설정됩니다.
> - `start`가 < 1 이고 길이가 지정되지 않았으면 인덱스 위치가 `start + length -1`로 설정됩니다.
> - `start + length -1`이 < 0이면 빈 문자열이 반환됩니다.
> - `start + length -1`이 > = 0이면 길이가 `start + length - 1`이고 인덱스 위치가 1로 시작하는 하위 문자열이 반환됩니다.

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.

예시

```
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 공백입니다(' ').

구문

```
TRIM ( [[LEADING | TRAILING | BOTH remove_chars] FROM] string )
```

파라미터

string

함수가 실행되는 대상 문자열입니다.

LEADING | TRAILING | BOTH

이 파라미터는 선행 또는 후행 문자를 다듬을지, 아니면 선행 및 후행 문자를 모두 다듬을지를 나타냅니다.

remove_chars

제거할 문자들의 집합입니다. *remove_chars*은 길이가 > 1인 문자열이 될 수 있습니다. 이 함수는 제거된 문자열의 시작 또는 끝에 *remove_chars*의 문자가 포함된 문자열을 반환합니다.

예시

```
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 UPPER는 모든 소문자를 대문자로 변환합니다. 소문자로 변환되지 않는 문자는 변경되지 않고 그대로 유지됩니다.

구문

```
UPPER ( string )
```

파라미터

string

함수가 실행되는 대상 문자열입니다.

예시

```
UPPER('AbCdEfG!@#$') -- 'ABCDEFG!@#$'
```

디렉터리 버킷 작업

디렉터리 버킷은 범용 버킷의 플랫 스토리지 구조와 달리 데이터를 계층적으로 디렉터리에 구성합니다. 디렉터리 버킷에는 접두사 제한이 없으며 개별 디렉터리는 수평적으로 확장할 수 있습니다.

각 AWS 계정에 최대 100개의 디렉터리 버킷을 생성할 수 있으며, 버킷에 저장할 수 있는 객체 수에는 제한이 없습니다. 버킷 할당량은 AWS 계정의 각 리전에 적용됩니다. 애플리케이션에서 이 한도를 늘려야 하는 경우 지원에 문의하세요.

> **Important**
>
> 가용 영역에서 최소 90일 동안 요청 활동이 없는 디렉터리 버킷은 비활성 상태로 전환됩니다. 비활성 상태에서는 디렉터리 버킷에 일시적으로 읽기 및 쓰기가 불가능합니다. 비활성 버킷은 모든 스토리지, 객체 메타데이터, 버킷 메타데이터를 보존합니다. 비활성 버킷에는 기존 스토리지 요금이 적용됩니다. 비활성 버킷에 대한 액세스 요청 시 버킷은 일반적으로 몇 분 이내에 활성 상태로 전환됩니다. 이 전환 기간 동안 읽기 및 쓰기 시 HTTP `503 (Service Unavailable)` 오류 코드가 반환됩니다. 로컬 영역의 버킷에는 적용되지 않습니다.

Amazon S3 버킷에는 여러 유형이 있습니다. 버킷을 만들기 전에 애플리케이션 및 성능 요구 사항에 가장 적합한 버킷 유형을 선택하세요. 다양한 버킷 유형과 각 버킷의 적절한 사용 사례에 대한 자세한 내용은 버킷 섹션을 참조하세요.

다음 주제에서는 디렉터리 버킷에 대한 정보를 제공합니다. 범용 버킷에 대한 자세한 내용은 범용 버킷 개요 섹션을 참조하세요.

디렉터리 버킷에 대한 자세한 내용은 다음 주제를 참조하세요.

- 디렉터리 버킷 이름
- 디렉터리
- 키 이름
- 액세스 관리

디렉터리 버킷 이름

디렉터리 버킷 이름은 사용자가 제공하는 기본 이름과 버킷이 위치한 영역(가용 영역 또는 로컬 영역)의 ID가 포함된 접미사로 구성됩니다. 디렉터리 버킷 이름은 다음 형식을 따라야 하며 디렉터리 버킷 이름 지정 규칙을 준수해야 합니다.

```
bucket-base-name--zone-id--x-s3
```

예를 들어, 다음 디렉터리 버킷 이름에는 가용 영역 ID인 `usw2-az1`이 포함되어 있습니다.

```
bucket-base-name--usw2-az1--x-s3
```

자세한 내용은 디렉터리 버킷 이름 지정 규칙 섹션을 참조하세요.

디렉터리

디렉터리 버킷은 범용 버킷의 플랫 스토리지 구조와 달리 데이터를 계층적으로 디렉터리에 구성합니다.

계층적 네임스페이스에서는 객체 키의 구분 기호가 중요합니다. 유일하게 지원되는 구분 기호는 슬래시(/)입니다. 디렉터리는 구분 기호 경계로 결정됩니다. 예를 들어, `dir1/dir2/file1.txt` 객체 키로 인해 `dir1/` 디렉터리가 만들어지고 `dir2/`가 자동으로 생성되며 `file1.txt` 객체는 `dir1/dir2/file1.txt` 경로의 `/dir2` 디렉터리에 추가됩니다.

디렉터리 버킷 인덱싱 모델은 `ListObjectsV2` API 작업에 대해 정렬되지 않은 결과를 반환합니다. 결과를 버킷의 하위 섹션으로 제한해야 하는 경우 `prefix` 파라미터에 하위 디렉터리 경로(예: `prefix=dir1/`)를 지정할 수 있습니다.

키 이름

디렉터리 버킷의 경우 여러 객체 키에 공통되는 하위 디렉터리가 첫 번째 객체 키로 생성됩니다. 동일한 하위 디렉터리의 추가 객체 키에는 이전에 만든 하위 디렉터리가 사용됩니다. 이 모델을 사용하면 밀집도가 낮은 디렉터리와 높은 디렉터리를 동일하게 지원하므로 애플리케이션에 가장 적합한 객체 키를 유연하게 선택할 수 있습니다.

액세스 관리

디렉터리 버킷에는 모든 S3 퍼블릭 액세스 차단 설정이 기본적으로 버킷 수준에서 활성화되어 있습니다. S3 객체 소유권은 버킷 소유자 적용으로 설정되며 액세스 제어 목록(ACL)이 비활성화되어 있습니다. 이 설정은 수정할 수 없습니다.

기본적으로 사용자는 디렉터리 버킷에 대한 권한이 없습니다. 디렉터리 버킷에 대한 액세스 권한을 부여하려면 IAM을 사용하여 사용자, 그룹 또는 역할을 생성하고 해당 ID에 권한을 연결하면 됩니다. 자세한 내용은 IAM을 사용한 리전 엔드포인트 API 권한 부여 섹션을 참조하세요.

액세스 포인트를 통해 디렉터리 버킷에 대한 액세스를 제어할 수도 있습니다. 액세스 포인트는 Amazon S3의 공유 데이터 세트에 대한 대규모 데이터 액세스 관리를 간소화합니다. 액세스 포인트는 액세스 포인트를 통해 이루어진 모든 요청에 대해 고유한 권한 및 네트워크 제어를 적용하기 위해 만드는 고유한 호스트 이름입니다. 자세한 내용은 액세스 포인트가 있는 디렉터리 버킷에서 공유 데이터세트 액세스 관리 섹션을 참조하세요.

디렉터리 버킷 할당량

한도라고도 하는 할당량은 AWS 계정의 최대 서비스 리소스 수 또는 작업 수입니다. 다음은 디렉터리 버킷의 할당량입니다. Amazon S3의 할당량에 대한 자세한 내용은 Amazon S3 quotas를 참조하세요.

명칭	기본값	조정 가능	설명
디렉터리 버킷	각 계정: 100개	예	한 계정에서 만들 수 있는 Amazon S3 디렉터리 버킷의 수입니다.
디렉터리 버킷당 읽기 TPS	각 디렉터리 버킷: 최대 200,000개의 읽기 TPS	할당량 증가를 요청하려면 Support에 문의하세요.	각 디렉터리 버킷의 초당 GET/HEAD 요청 수입니다.
디렉터리 버킷당 쓰기 TPS	각 디렉터리 버킷: 최대 100,000개의 쓰기 TPS	할당량 증가를 요청하려면 Support에 문의하세요.	각 디렉터리 버킷의 초당 PUT/DELETE 요청 수입니다.

디렉터리 버킷 생성 및 사용

디렉터리 버킷 작업에 대한 자세한 내용은 다음 주제를 참조하세요.

- 디렉터리 버킷 사용 사례
- 디렉터리 버킷의 차이점
- 디렉터리 버킷에 대한 네트워킹
- 디렉터리 버킷 이름 지정 규칙
- 디렉터리 버킷 속성 보기
- 디렉터리 버킷 정책 관리
- 디렉터리 버킷 비우기
- 디렉터리 버킷 삭제
- 디렉터리 버킷 나열
- 디렉터리 버킷 액세스 가능 여부 결정
- 디렉터리 버킷의 객체 작업
- 디렉터리 버킷의 보안
- 액세스 포인트가 있는 디렉터리 버킷에서 공유 데이터세트 액세스 관리
- 디렉터리 버킷 성능 최적화
- 디렉터리 버킷으로 배포
- S3 디렉터리 버킷에서 태그 사용

디렉터리 버킷 사용 사례

디렉터리 버킷은 가용 영역 또는 로컬 영역과 같은 버킷 위치 유형에서 버킷 생성을 지원합니다.

지연 시간이 짧은 사용 사례의 경우 단일 가용 영역에 디렉터리 버킷을 생성하여 데이터를 저장할 수 있습니다. 가용 영역의 디렉터리 버킷은 S3 Express One Zone 스토리지 클래스를 지원합니다. S3 Express One Zone 스토리지 클래스는 애플리케이션이 성능에 민감하고 10밀리초 미만의 PUT 및 GET 지연 시간이 필요한 경우에 권장됩니다. 가용 영역에서 디렉터리 버킷을 생성하는 방법에 대한 자세한 내용은 고성능 워크로드 섹션을 참조하세요.

데이터 레지던시 사용 사례의 경우 단일 AWS 전용 로컬 영역(DLZ)에 디렉터리 버킷을 생성하여 데이터를 저장할 수 있습니다. 로컬 영역의 디렉터리 버킷은 S3 One Zone-Infrequent Access(S3 One Zone-IA, Z-IA) 스토리지 클래스를 지원합니다. 로컬 영역에서 디렉터리 버킷을 생성하는 방법에 대한 자세한 내용은 데이터 레지던시 워크로드 섹션을 참조하세요.

주제

- 고성능 워크로드
- 데이터 레지던시 워크로드

고성능 워크로드

S3 Express One Zone

고성능 워크로드에 Amazon S3 Express One Zone을 사용할 수 있습니다. S3 Express One Zone은 객체 스토리지를 컴퓨팅 리소스와 함께 배치하는 옵션을 제공하면서 단일 가용 영역 선택이 가능한 최초의 S3 스토리지 클래스로, 최고의 액세스 속도를 제공합니다. S3 Express One Zone의 객체는 가용 영역에 위치한 디렉터리 버킷에 저장됩니다. 디렉터리 버킷에 대한 자세한 내용은 디렉터리 버킷을 참조하세요.

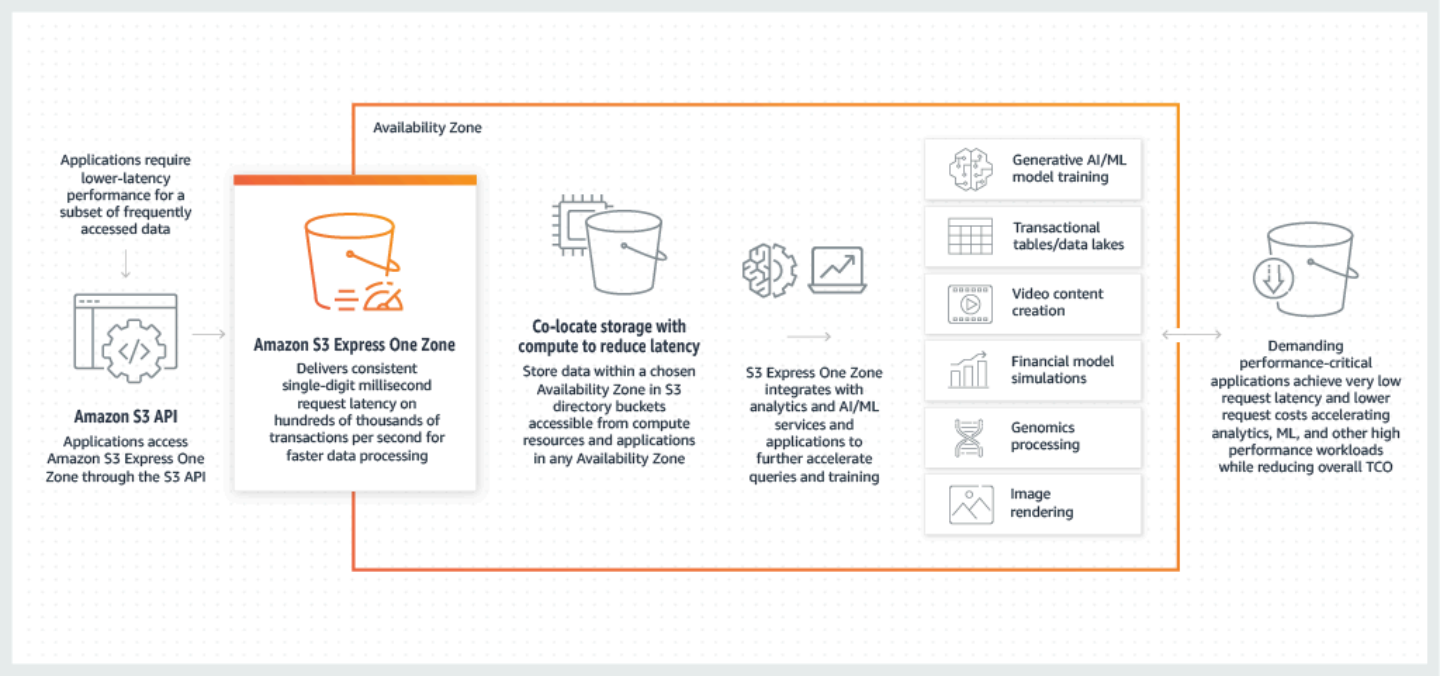
S3 Express One Zone은 지연 시간에 가장 민감한 애플리케이션에 대해 일관되게 10밀리초 미만의 데이터 액세스를 제공하도록 특별히 설계된 고성능 단일 영역 Amazon S3 스토리지 클래스입니다. S3 Express One Zone은 현재 사용 가능한 클라우드 객체 스토리지 클래스 중 지연 시간이 가장 낮으며, S3 Standard보다 데이터 액세스 속도는 최대 10배 빠르고 요청 비용은 50% 저렴합니다. 요청이 최대 10배 더 빠르게 완료되므로 애플리케이션이 즉시 이점을 누릴 수 있습니다. S3 Express One Zone은 다른 S3 스토리지 클래스와 유사한 성능 탄력성을 제공합니다. S3 Express One Zone은 일관된 10밀리초 미만의 지연 시간이 필요한 워크로드 또는 성능이 중요한 애플리케이션에 사용됩니다.

다른 Amazon S3 스토리지 클래스와 마찬가지로 용량 또는 처리량 요구 사항을 미리 계획하거나 프로비저닝할 필요가 없습니다. 필요에 따라 스토리지를 확장하거나 축소하고 Amazon S3 API를 통해 데이터에 액세스할 수 있습니다.

Amazon S3 Express One Zone 스토리지 클래스는 단일 가용 영역 내에서 99.95%의 가용성을 제공하도록 설계되었으며 Amazon S3 서비스 수준 계약의 지원을 받습니다. S3 Express One Zone 스토리지 클래스를 사용하면 단일 가용 영역 내의 여러 디바이스에 데이터가 중복으로 저장됩니다. S3 Express One Zone은 손실된 중복성을 신속하게 탐지하고 복구하여 동시 디바이스 오류를 처리하도록 설계되었습니다. 기존 디바이스에서 장애가 발생하는 경우 S3 Express One Zone은 자동으로 요청을 가용 영역 내의 새 디바이스로 이전합니다. 이러한 중복성은 가용 영역 내의 데이터에 대한 중단 없는 액세스를 보장하는 데 도움이 됩니다.

S3 Express One Zone은 객체 액세스에 필요한 지연 시간을 최소화하는 것이 중요한 모든 애플리케이션에 적합합니다. 크리에이티브 전문가가 사용자 인터페이스에서 콘텐츠에 신속하게 액세스해야 하는 비디오 편집과 같은 인적 상호작용 워크플로를 이러한 애플리케이션의 예로 들 수 있습니다. 또한 S3 Express One Zone은 데이터에 대한 응답성 요구 사항이 유사한 분석 및 기계 학습 워크로드, 특히 소규모 액세스가 많거나 임의 액세스가 많은 워크로드에 유용합니다. S3 Express One Zone을 다

른 AWS 서비스와 함께 사용하여 분석 및 인공 지능 및 기계 학습(AI/ML) 워크로드(예: Amazon EMR, Amazon SageMaker AI, Amazon Athena)를 지원할 수 있습니다.



S3 Express One Zone 스토리지 클래스를 사용하는 디렉터리 버킷의 경우 데이터는 단일 가용 영역 내의 여러 디바이스에 데이터를 저장하지만 서로 다른 가용 영역에 데이터를 중복으로 저장하지는 않습니다. 디렉터리 버킷을 생성하여 S3 Express One Zone 스토리지 클래스를 사용할 때 Amazon EC2, Amazon Elastic Kubernetes Service 또는 Amazon Elastic Container Service(Amazon ECS) 컴퓨팅 인스턴스와 같은 위치의 로컬 가용 영역과 AWS 리전을 지정하여 성능을 최적화하는 것이 좋습니다.

S3 Express One Zone을 사용하면 게이트웨이 VPC 엔드포인트를 사용하여 가상 프라이빗 클라우드(VPC)의 디렉터리 버킷과 상호작용할 수 있습니다. 게이트웨이 엔드포인트를 사용하면 인터넷 게이트웨이 또는 VPC의 NAT 디바이스를 사용하지 않고 추가 비용 없이 VPC에서 S3 Express One Zone 디렉터리 버킷 액세스할 수 있습니다.

범용 버킷 및 기타 스토리지 클래스에서 사용하는 디렉터리 버킷과 동일한 Amazon S3 API 작업 및 기능을 여러 개 사용할 수 있습니다. 여기에는 Mountpoint for Amazon S3, Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3), AWS Key Management Service(AWS KMS) 키를 사용한 서버 측 암호화(SSE-KMS), S3 배치 운영, S3 퍼블릭 액세스 차단이 포함됩니다. Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), AWS SDK 및 Amazon S3 REST API를 사용하여 S3 Express One Zone 스토리지 클래스에 액세스할 수 있습니다.

S3 Express One Zone에 대한 자세한 내용은 다음 주제를 참조하세요.

- 개요

- S3 Express One Zone의 기능
- 관련 서비스
- 다음 단계

개요

성능을 최적화하고 지연 시간을 줄이기 위해 S3 Express One Zone에 다음과 같은 새로운 개념을 도입했습니다.

가용 영역

Amazon S3 Express One Zone 스토리지 클래스는 단일 가용 영역 내에서 99.95%의 가용성을 제공하도록 설계되었으며 Amazon S3 서비스 수준 계약의 지원을 받습니다. S3 Express One Zone 스토리지 클래스를 사용하면 단일 가용 영역 내의 여러 디바이스에 데이터가 중복으로 저장됩니다. S3 Express One Zone은 손실된 중복성을 신속하게 탐지하고 복구하여 동시 디바이스 오류를 처리하도록 설계되었습니다. 기존 디바이스에서 장애가 발생하는 경우 S3 Express One Zone은 자동으로 요청을 가용 영역 내의 새 디바이스로 이전합니다. 이러한 중복성은 가용 영역 내의 데이터에 대한 중단 없는 액세스를 보장하는 데 도움이 됩니다.

가용 영역은 AWS 리전에 중복 전원, 네트워킹 및 연결이 있는 하나 이상의 개별 데이터 센터입니다. 디렉터리 버킷을 생성할 때 버킷이 위치할 가용 영역과 AWS 리전을 선택합니다.

단일 가용 영역

디렉터리 버킷을 생성할 때 가용 영역과 AWS 리전을 선택합니다.

디렉터리 버킷은 성능에 민감한 애플리케이션에서 사용하도록 빌드된 S3 Express One Zone 스토리지 클래스를 사용합니다. S3 Express One Zone은 객체 스토리지를 컴퓨팅 리소스와 함께 배치하는 옵션을 제공하면서 단일 가용 영역 선택이 가능한 최초의 S3 스토리지 클래스로, 최고의 액세스 속도를 제공합니다.

S3 Express One Zone 스토리지 클래스를 사용하면 단일 가용 영역 내의 여러 디바이스에 데이터가 중복으로 저장됩니다. S3 Express One Zone은 단일 가용 영역 내에서 99.95%의 가용성을 제공하도록 설계되었으며 Amazon S3 서비스 수준 계약의 지원을 받습니다. 자세한 내용은 가용 영역 섹션을 참조하세요.

엔드포인트 및 게이트웨이 VPC 엔드포인트

디렉터리 버킷의 버킷 관리 API 작업은 리전 엔드포인트를 통해 사용할 수 있으며 이를 리전 엔드포인트 API 작업이라고 합니다. 리전 엔드포인트 API 작업의 예로는 `CreateBucket` 및 `DeleteBucket`이

있습니다. 디렉터리 버킷을 생성한 후 영역 엔드포인트 API 작업을 사용하여 디렉터리 버킷에 객체를 업로드하고 관리할 수 있습니다. 영역 엔드포인트 API 작업은 영역 엔드포인트를 통해 사용할 수 있습니다. 영역 엔드포인트 API 작업의 예로는 `PutObject` 및 `CopyObject`가 있습니다.

VPC에서 게이트웨이 VPC 엔드포인트를 사용하여 S3 Express One Zone에 액세스할 수 있습니다. 게이트웨이 엔드포인트를 생성한 후 VPC에서 S3 Express One Zone으로 전송되는 트래픽에 대해 해당 엔드포인트를 라우팅 테이블의 대상으로 추가할 수 있습니다. Amazon S3와 마찬가지로 게이트웨이 엔드포인트 사용에 따르는 추가 요금은 없습니다. 게이트웨이 VPC 엔드포인트를 구성하는 방법에 대한 자세한 내용은 디렉터리 버킷에 대한 네트워킹 섹션을 참조하세요.

세션 기반 권한 부여

S3 Express One Zone을 사용하면 지연 시간을 최소화하도록 최적화된 새로운 세션 기반 메커니즘을 통해 요청을 인증하고 권한을 부여할 수 있습니다. `CreateSession`을 사용하여 지연 시간이 짧은 버킷 액세스를 제공하는 임시 자격 증명을 요청할 수 있습니다. 이러한 임시 자격 증명의 범위는 특정 S3 디렉터리 버킷으로 지정됩니다. 세션 토큰은 영역(객체 수준) 작업(CopyObject 제외)에만 사용됩니다. 자세한 내용은 `CreateSession`을 사용하여 영역 엔드포인트 API 작업 권한 부여 섹션을 참조하세요.

지원되는 S3 Express One Zone용 AWS SDK는 사용자를 대신하여 세션 설정 및 새로 고침을 처리합니다. 세션을 보호하기 위해 임시 자격 증명은 5분 후에 만료됩니다. AWS SDK를 다운로드하여 설치하고 필요한 AWS Identity and Access Management(IAM) 권한을 구성한 후에는 즉시 API 작업을 사용할 수 있습니다.

S3 Express One Zone의 기능

S3 Express One Zone에서 사용할 수 있는 S3 기능은 다음과 같습니다. 지원되는 API 작업 및 지원되지 않는 기능의 전체 목록은 디렉터리 버킷의 차이점 섹션을 참조하세요.

액세스 관리 및 보안

다음 기능을 사용하여 액세스를 감사하고 관리할 수 있습니다. 기본적으로 디렉터리 버킷은 프라이빗이며 액세스 권한이 명시적으로 부여된 사용자만 액세스할 수 있습니다. 버킷, 접두사 또는 객체 태그 수준에서 액세스 제어 경계를 설정할 수 있는 범용 버킷과 달리 디렉터리 버킷의 액세스 제어 경계는 버킷 수준에서만 설정됩니다. 자세한 내용은 IAM을 사용한 리전 엔드포인트 API 권한 부여 섹션을 참조하세요.

- S3 퍼블릭 액세스 차단 - 모든 S3 퍼블릭 액세스 차단 설정은 기본적으로 버킷 수준에서 활성화됩니다. 이 기본 설정은 수정할 수 없습니다.
- S3 객체 소유권(버킷 소유자가 기본적으로 적용됨) - 액세스 제어 목록(ACL)은 디렉터리 버킷에 지원되지 않습니다. 디렉토리 버킷은 S3 객체 소유권에 대해 버킷 소유자 적용 설정을 자동으로 사용

합니다. 버킷 소유자 시행은 ACL이 비활성화되고 버킷 소유자는 버킷의 모든 객체를 자동으로 소유하고 완전히 제어한다는 의미입니다. 이 기본 설정은 수정할 수 없습니다.

- AWS Identity and Access Management(IAM) - IAM을 사용하면 디렉터리 버킷에 대한 액세스를 안전하게 제어할 수 있습니다. IAM을 사용하면 `s3express:CreateSession` 작업을 통해 버킷 관리(리전) API 작업 및 객체 관리(영역) API 작업에 대한 액세스 권한을 부여할 수 있습니다. 자세한 내용은 IAM을 사용한 리전 엔드포인트 API 권한 부여 섹션을 참조하세요. 객체 관리 작업과 달리 버킷 관리 작업은 계정 간에 수행할 수 없습니다. 버킷 소유자만 이러한 작업을 수행할 수 있습니다.
- 버킷 정책 - IAM 기반 정책 언어를 사용하여 디렉터리 버킷에 대한 리소스 기반 권한을 구성합니다. 또한 IAM을 사용하여 `CreateSession` API 작업에 대한 액세스를 제어할 수 있으며, 이를 통해 리전 또는 객체 관리 API 작업을 사용할 수 있습니다. 영역 API 작업에 대한 동일 계정 또는 크로스 계정 액세스 권한을 부여할 수 있습니다. S3 Express One Zone 권한 및 정책에 대한 자세한 내용은 IAM을 사용한 리전 엔드포인트 API 권한 부여 섹션을 참조하세요.
- IAM Access Analyzer for S3 - 액세스 정책을 평가 및 모니터링하여 정책이 S3 리소스에 대한 의도된 액세스만 제공하는지 확인합니다.

로깅 및 모니터링

S3 Express One Zone은 리소스가 사용되는 방식을 모니터링하고 제어하는 데 사용할 수 있는 다음의 S3 로깅 및 모니터링 도구를 제공합니다.

- Amazon CloudWatch 지표 - CloudWatch를 사용하여 지표를 수집 및 추적하는 방식으로 AWS 리소스 및 애플리케이션을 모니터링합니다. S3 Express One Zone은 다른 Amazon S3 스토리지 클래스(`AWS/S3`)와 동일한 CloudWatch 네임스페이스를 사용하며 디렉터리 버킷 `BucketSizeBytes` 및 `NumberOfObjects`에 대해 일일 스토리지 지표를 지원합니다. 자세한 내용은 Amazon CloudWatch를 사용한 지표 모니터링 섹션을 참조하세요.
- AWS CloudTrail 로그 - AWS CloudTrail은 사용자, 역할 또는 AWS 서비스가 수행한 작업을 기록하여 AWS 계정의 운영 및 위험 감사, 거버넌스, 규정 준수를 구현하는 데 도움이 되는 AWS 서비스입니다. S3 Express One Zone의 경우 CloudTrail은 리전 엔드포인트 API 작업(예: `CreateBucket` 및 `PutBucketPolicy`)을 관리 이벤트로 캡처하고 영역 API 작업(예: `GetObject` 및 `PutObject`)을 데이터 이벤트로 캡처합니다. 이러한 이벤트에는AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDK 및 AWS API 작업에서 수행된 작업이 포함됩니다. 자세한 내용은 S3 Express One Zone에 대한 AWS CloudTrail 로깅을 참조하세요.

> **Note**
>
> Amazon S3 서버 액세스 로그는 S3 Express One Zone에서 지원되지 않습니다.

객체 관리

Amazon S3 콘솔, AWS SDK 및 AWS CLI를 사용하여 객체 스토리지를 관리할 수 있습니다. S3 Express One Zone에서 객체 관리에 사용할 수 있는 기능은 다음과 같습니다.

- S3 배치 작업 - 배치 작업을 사용하여 디렉터리 버킷의 객체에 배치 작업을 수행합니다(예: 복사 및 AWS Lambda 함수 호출). 예를 들어 배치 작업를 사용하여 디렉터리 버킷과 범용 버킷 간에 객체를 복사할 수 있습니다. 배치 작업을 사용하면 AWS SDK 또는 AWS CLI를 사용하거나 Amazon S3 콘솔에서 몇 번의 클릭을 통해 하나의 S3 요청으로 수십억 개의 객체를 대규모로 관리할 수 있습니다.
- 가져오기 - 디렉터리 버킷을 생성한 후 Amazon S3 콘솔의 가져오기 기능을 사용하여 버킷을 객체로 채울 수 있습니다. 가져오기는 범용 버킷에서 디렉터리 버킷으로 객체를 복사하는 배치 작업 건을 생성하는 간소화된 방법입니다.

AWS SDK 및 클라이언트 라이브러리

AWS SDK 및 클라이언트 라이브러리를 사용하여 객체 스토리지를 관리할 수 있습니다.

- Mountpoint for Amazon S3 - Mountpoint for Amazon S3는 높은 처리량 액세스를 제공하여 Amazon S3의 데이터 레이크에 대한 컴퓨팅 비용을 낮추는 오픈 소스 파일 클라이언트입니다. Mountpoint for Amazon S3는 로컬 파일 시스템 API 직접 호출을 `GET` 및 `LIST`와 같은 S3 객체 API 직접 호출로 변환합니다. 페타바이트 규모의 데이터를 처리하고 수천 개의 인스턴스로 확장 및 축소하기 위해 Amazon S3에서 제공하는 탄력적인 처리량이 필요한 읽기 중심의 데이터 레이크 워크로드에 적합합니다.
- S3A - S3A는 Amazon S3의 데이터 스토어에 액세스하는 데 권장되는 Hadoop 호환 인터페이스입니다. S3A는 S3N Hadoop 파일 시스템 클라이언트를 대체합니다.
- PyTorch on AWS - PyTorch on AWS는 기계 학습 모델을 쉽게 개발하고 프로덕션에 배포하는 데 사용되는 오픈 소스 딥 러닝 프레임워크입니다.
- AWS SDK - Amazon S3로 애플리케이션을 개발할 때 AWS SDK를 사용할 수 있습니다. AWS SDK에서는 기본 Amazon S3 REST API를 래핑하여 프로그래밍 태스크를 단순화합니다. S3 Express One Zone에서 AWS SDK를 사용하는 방법에 대한 자세한 내용은 the section called “AWS SDK” 섹션을 참조하세요.

암호화 및 데이터 보호

S3 Express One Zone의 객체는 Amazon S3 관리형 키를 통한 서버 측 암호화(SSE-S3)를 통해 자동으로 암호화됩니다. S3 Express One Zone은 AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 사용한 서버 측 암호화도 지원합니다. S3 Express One Zone은 고객 제공 암호화 키를 사용한 서버 측 암호화(SSE-C) 또는 AWS KMS keys를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)를 지원하지 않습니다. 자세한 내용은 데이터 보호 및 암호화 섹션을 참조하세요.

S3 Express One Zone은 업로드 또는 다운로드 중에 데이터를 검증하는 데 사용하는 체크섬 알고리즘을 선택할 수 있는 옵션을 제공합니다. SHA(보안 해시 알고리즘) 또는 CRC(순환 중복 검사) 데이터 무결성 확인 알고리즘(CRC32, CRC32C, SHA-1, SHA-256) 중 하나를 선택할 수 있습니다. MD5 기반 체크섬은 S3 Express One Zone 스토리지 클래스에서 지원되지 않습니다.

자세한 내용은 S3 추가 체크섬 모범 사례 섹션을 참조하세요.

AWS 서명 버전 4(SigV4)

S3 Express One Zone은 AWS Signature Version 4(SigV4)를 사용합니다. SigV4는 HTTPS를 통해 Amazon S3에 대한 요청을 인증하는 데 사용되는 서명 프로토콜입니다. S3 Express One Zone은 AWS Sigv4를 사용하여 요청에 서명합니다. 자세한 내용은 Amazon Simple Storage Service API 참조의 요청 인증(AWS 서명 버전 4)을 참조하세요.

강력한 일관성

S3 Express One Zone은 모든 AWS 리전의 디렉터리 버킷에 있는 객체의 PUT 및 DELETE 요청에 대해 강력한 쓰기 후 읽기(read-after-write) 일관성을 제공합니다. 자세한 내용은 Amazon S3 데이터 일관성 모델 섹션을 참조하세요.

관련 서비스

S3 Express One Zone 스토리지 클래스와 함께 다음 AWS 서비스를 사용하여 지연 시간이 짧은 구체적인 사용 사례를 지원할 수 있습니다.

- Amazon Elastic Compute Cloud(Amazon EC2) – Amazon EC2는 AWS 클라우드에서 안전하고 확장 가능한 컴퓨팅 용량을 제공합니다. Amazon EC2를 사용하면 하드웨어에 선투자할 필요성이 감소되어 더 빠르게 애플리케이션을 개발하고 배포할 수 있습니다. Amazon EC2를 사용하여 원하는 수의 가상 서버를 빌드하고 보안 및 네트워킹을 구성하며 스토리지를 관리할 수 있습니다.
- AWS Lambda - Lambda는 서버를 프로비저닝하거나 관리하지 않고도 코드를 실행할 수 있게 해주는 컴퓨팅 서비스입니다. 버킷에 알림 설정을 구성하고 Amazon S3에 함수의 리소스 기반 권한 정책에 따라 함수를 간접 호출할 수 있는 권한을 부여합니다.

- Amazon Elastic Kubernetes Service(Amazon EKS) - Amazon EKS는 AWS에 자체 Kubernetes 컨트롤 플레인을 설치, 운영 및 유지 관리할 필요가 없는 관리형 서비스입니다. Kubernetes는 컨테이너화된 애플리케이션의 관리, 규모 조정 및 배포를 자동화하는 오픈 소스 시스템입니다.
- Amazon Elastic Container Service(Amazon ECS) Amazon ECS는 컨테이너화된 애플리케이션을 쉽게 배포, 관리, 규모 조정할 수 있도록 도와주는 완전 관리형 컨테이너 오케스트레이션 서비스입니다.
- AWS Key Management Service(AWS KMS) – AWS Key Management Service(AWS KMS)는 데이터 암호화에 사용하는 암호화 키를 쉽게 생성하고 제어할 수 있게 해주는 AWS 관리형 서비스입니다. AWS KMS에서 생성하는 AWS KMS 키는 FIPS 140-2 검증 하드웨어 보안 모듈(HSM)로 보호됩니다. KMS 키를 사용하거나 관리하려면 AWS KMS와 상호 작용합니다.
- Amazon Athena – Amazon Athena는 표준 SQL을 사용해 Amazon S3에서 직접 데이터를 간편하게 분석할 수 있는 대화식 쿼리 서비스입니다. 또한 Athena를 사용하면 리소스를 계획, 구성 또는 관리할 필요 없이 Apache Spark를 사용하여 데이터 분석을 대화식으로 실행할 수 있습니다. Athena에서 Apache Spark 애플리케이션을 실행하는 경우 처리를 위해 Spark 코드를 제출하고 결과를 직접 수신합니다.
- Amazon SageMaker 훈련 - AWS Marketplace의 기본 제공 알고리즘, 사용자 지정 알고리즘, 라이브러리 및 모델 등 Amazon SageMaker를 사용하여 모델을 훈련하는 옵션을 검토합니다.
- AWS Glue - AWS Glue는 분석 사용자가 여러 소스의 데이터를 쉽게 검색, 준비, 이동, 통합할 수 있도록 하는 서버리스 데이터 통합 서비스입니다. 분석, 기계 학습 및 애플리케이션 개발에 AWS Glue를 사용할 수 있습니다. AWS Glue에도 작성, 작업 실행, 비즈니스 워크플로 구현을 위한 추가 생산성 및 데이터 운영 도구가 있습니다.
- Amazon EMR – Amazon EMR은 AWS에서 Apache Hadoop 및 Apache Spark와 같은 빅 데이터 프레임워크 실행을 단순화하여 방대한 양의 데이터를 처리하고 분석하는 관리형 클러스터 플랫폼입니다.
- AWSCloudTrail – AWSCloudTrail은 AWS 계정의 운영 및 위험 감사, 거버넌스, 규정 준수를 지원하는 AWS 서비스입니다. 사용자, 역할 또는 AWS 서비스가 수행하는 작업은 CloudTrail에 이벤트로 기록됩니다. 이벤트에는 AWS Management Console, AWS Command Line Interface, AWS SDK 및 API에서 수행되는 작업이 포함됩니다.
- AWS CloudFormation – AWS 리소스를 모델링하고 설정하여 리소스 관리 시간을 줄이고 AWS에서 실행되는 애플리케이션에 더 많은 시간을 사용하도록 해 주는 서비스입니다. 필요한 모든 AWS 리소스(예: Amazon EC2 인스턴스 또는 Amazon RDS DB 인스턴스)를 설명하는 템플릿을 생성하면 CloudFormation이 해당 리소스의 프로비저닝과 구성을 담당합니다. AWS 리소스를 개별적으로 생성하고 구성할 필요가 없으며 어떤 것이 무엇에 의존하는지 파악할 필요도 없습니다. CloudFormation에서 모든 것을 처리합니다.

다음 단계

S3 Express One Zone 스토리지 클래스 및 디렉터리 버킷 사용에 대한 자세한 내용은 다음 주제를 참조하세요.

- 튜토리얼: S3 Express One Zone 시작하기
- S3 Express One Zone 가용 영역 및 리전
- 가용 영역의 디렉터리 버킷에 대한 네트워킹
- 가용 영역에서 디렉터리 버킷 생성
- 가용 영역의 디렉터리 버킷에 대한 리전 및 영역 엔드포인트
- S3 Express One Zone 성능 최적화

튜토리얼: S3 Express One Zone 시작하기

Amazon S3 Express One Zone은 객체 스토리지를 컴퓨팅 리소스와 함께 배치하는 옵션을 제공하면서 단일 가용 영역 선택이 가능한 최초의 S3 스토리지 클래스로, 최고의 액세스 속도를 제공합니다. S3 Express One Zone에 있는 데이터는 가용 영역에 위치한 디렉터리 버킷에 저장됩니다. 디렉터리 버킷에 대한 자세한 내용은 디렉터리 버킷을 참조하세요.

S3 Express One Zone은 요청 지연 시간을 최소화하는 것이 중요한 모든 애플리케이션에 적합합니다. 크리에이티브 전문가가 사용자 인터페이스에서 콘텐츠에 신속하게 액세스해야 하는 비디오 편집과 같은 인적 상호작용 워크플로를 이러한 애플리케이션의 예로 들 수 있습니다. 또한 S3 Express One Zone은 데이터에 대한 응답성 요구 사항이 유사한 분석 및 기계 학습 워크로드, 특히 소규모 액세스가 많거나 임의 액세스가 많은 워크로드에 유용합니다. S3 Express One Zone을 다른 AWS 서비스(예: Amazon EMR, Amazon Athena, AWS Glue Data Catalog, Amazon SageMaker Model Training)와 함께 사용하여 분석 및 인공 지능 및 기계 학습(AI/ML) 워크로드를 지원할 수 있습니다. Amazon S3 콘솔, AWS SDK, AWS 명령줄 인터페이스(AWS CLI), Amazon S3 REST API를 사용하여 S3 Express One Zone 스토리지 클래스 및 디렉터리 버킷을 사용할 수 있습니다. 자세한 내용은 S3 Express One Zone이란? 및 S3 Express One Zone의 차이점을 참조하세요.
이는 S3 Express One Zone 워크플로 다이어그램입니다.

목표

이 튜토리얼에서는 게이트웨이 엔드포인트를 생성하고, IAM 정책을 생성 및 연결하고, 디렉터리 버킷을 생성한 다음, 가져오기 작업을 사용하여 범용 버킷에 현재 저장된 객체로 디렉터리 버킷을 채우는 방법을 알아봅니다. 디렉터리 버킷에 객체를 수동으로 업로드할 수 있습니다.

주제

- 사전 조건
- 1단계: S3 Express One Zone 디렉터리 버킷에 도달하도록 게이트웨이 VPC 엔드포인트 구성
- 2단계: S3 Express One Zone 디렉터리 버킷 생성
- 3단계: S3 Express One Zone 디렉터리 버킷으로 데이터 가져오기
- 4단계: S3 Express One Zone 디렉터리 버킷에 객체를 수동으로 업로드
- 5단계: S3 Express One Zone 디렉터리 버킷 비우기
- 6단계: S3 Express One Zone 디렉터리 버킷 삭제
- 다음 단계

사전 조건

이 자습서를 시작하기 전에 올바른 권한이 있는 AWS Identity and Access Management(IAM) 사용자로 로그인할 수 있는 AWS 계정이 있어야 합니다.

하위 단계

- AWS 계정 생성
- AWS 계정에서 IAM 사용자 생성(콘솔)
- IAM 정책을 생성하여 IAM 사용자 또는 역할에 연결(콘솔)

AWS 계정 생성

이 자습서를 완료하려면 AWS 계정이 필요합니다. AWS에 가입하면 Amazon S3를 포함하여 AWS의 모든 서비스에 AWS 계정이 자동으로 등록됩니다. 사용자에게는 사용한 서비스에 대해서만 요금이 청구됩니다. 요금에 대한 자세한 내용은 S3 요금을 참조하세요.

AWS 계정에서 IAM 사용자 생성(콘솔)

AWS Identity and Access Management(IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 있도록 지원하는 AWS 서비스입니다. IAM 관리자는 어떤 사용자가 S3 Express One Zone의 객체에 액세스하고 디렉터리 버킷을 사용할 수 있도록 인증(로그인)되고 권한이 부여(권한 보유)될 수 있는지 제어합니다. IAM은 추가 요금 없이 사용할 수 있습니다.

기본적으로 사용자는 디렉터리 버킷에 액세스하고 S3 Express One Zone 작업을 수행할 수 있는 권한이 없습니다. 디렉터리 버킷 및 S3 Express One Zone 작업에 대한 액세스 권한을 부여하려면 IAM을 사용하여 사용자 또는 역할을 생성하고 해당 ID에 권한을 연결하면 됩니다. IAM 사용자를 생성하는 방

법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 사용자 생성(콘솔)을 참조하십시오. IAM 역할을 생성하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 사용자에게 권한을 위임하는 역할 생성을 참조하세요.

편의상 이 자습서에서는 IAM 사용자를 생성해 사용합니다. 이 자습서를 완료하면 IAM 사용자 삭제를 수행해야 합니다. 프로덕션 용도로는 IAM 사용 설명서에서 IAM의 보안 모범 사례를 따르는 것이 좋습니다. 가장 좋은 방법은 인간 사용자가 ID 제공업체와의 페더레이션을 사용하여 임시 보안 인증으로 AWS에 액세스하도록 하는 것입니다. 워크로드에서 IAM 역할과 함께 임시 보안 인증을 사용하여 AWS에 액세스하도록 하는 것도 좋은 방법입니다. AWS IAM Identity Center을 사용하여 임시 자격 증명으로 사용자를 생성하는 방법은 AWS IAM Identity Center 사용 설명서의 시작하기를 참조하세요.

> **Warning**
>
> IAM 사용자는 장기 자격 증명을 가지므로 보안 위험이 있습니다. 이 위험을 줄이려면 이러한 사용자에게 작업을 수행하는 데 필요한 권한만 제공하고 더 이상 필요하지 않을 경우 이러한 사용자를 제거하는 것이 좋습니다.

IAM 정책을 생성하여 IAM 사용자 또는 역할에 연결(콘솔)

기본적으로 사용자는 디렉터리 버킷 및 S3 Express One Zone 작업에 대한 권한이 없습니다. 디렉터리 버킷에 대한 액세스 권한을 부여하려면 IAM을 사용하여 사용자, 그룹 또는 역할을 생성하고 해당 ID에 권한을 연결하면 됩니다. 디렉터리 버킷은 S3 Express One Zone 액세스에 대한 버킷 정책 또는 IAM ID 정책에 포함할 수 있는 유일한 리소스입니다.

S3 Express One Zone에서 리전 엔드포인트 API 작업(버킷 수준 또는 컨트롤 플레인 작업)을 사용하려면 세션 관리가 필요하지 않은 IAM 권한 부여 모델을 사용해야 합니다. 작업에 대한 권한은 개별적으로 부여됩니다. 영역 엔드포인트 API 작업(객체 수준 또는 데이터 영역 작업)을 사용하려면 `CreateSession`을 사용하여 지연 시간이 짧은 데이터 요청 권한 부여에 최적화된 세션을 생성하고 관리합니다. 세션 토큰을 검색하고 사용하려면 ID 기반 정책 또는 버킷 정책에서 디렉터리 버킷에 대한 `s3express:CreateSession` 작업을 허용해야 합니다. Amazon S3 콘솔에서, AWS 명령줄 인터페이스(AWS CLI)를 통해 또는 AWS SDK를 사용하여 S3 Express One Zone에 액세스하는 경우, S3 Express One Zone은 사용자를 대신하여 세션을 생성합니다. 자세한 내용은 `CreateSession` authorization 및 AWS Identity and Access Management (IAM) for S3 Express One Zone을 참조하세요.

IAM 정책을 생성하여 IAM 사용자(또는 역할)에 정책 연결

1. AWS Management Console에 로그인하여 IAM 관리 콘솔을 엽니다.

2. 탐색 창에서 Policies를 선택합니다.
3. 정책 생성을 선택하세요.
4. JSON을 선택합니다.
5. 아래 정책을 정책 편집기 창에 복사합니다. 디렉터리 버킷을 생성하거나 S3 Express One Zone을 사용하려면 먼저 AWS Identity and Access Management(IAM) 역할 또는 사용자에게 필요한 권한을 부여해야 합니다. 이 예시 정책은 `CreateSession` API 작업(다른 영역 또는 객체 수준 API 작업에 사용) 및 모든 리전 엔드포인트(버킷 수준) API 작업에 대한 액세스를 허용합니다. 이 정책은 모든 디렉터리 버킷에서 `CreateSession` API 작업을 사용할 수 있도록 허용하지만, 리전 엔드포인트 API 작업은 지정된 디렉터리 버킷에만 사용할 수 있습니다. 이 정책 예를 사용하려면 *`user input placeholders`*를 실제 정보로 바꾸세요.

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Sid": "AllowAccessRegionalEndpointAPIs",
             "Effect": "Allow",
             "Action": [
                 "s3express:DeleteBucket",
                 "s3express:DeleteBucketPolicy",
                 "s3express:CreateBucket",
                 "s3express:PutBucketPolicy",
                 "s3express:GetBucketPolicy",
                 "s3express:ListAllMyDirectoryBuckets"
             ],

             "Resource": "arn:aws:s3express:region:account_id:bucket/bucket-base-name--zone-id--x-s3/*"
         },
         {
             "Sid": "AllowCreateSession",
             "Effect": "Allow",
             "Action": "s3express:CreateSession",
             "Resource": "*"
         }
     ]
 }
```

6. 다음을 선택합니다.
7. 정책 이름을 지정합니다.

> **Note**
>
> 버킷 태그는 S3 Express One Zone에서 지원되지 않습니다.

8. 정책 생성을 선택합니다.
9. IAM 정책을 생성했으면 IAM 사용자에게 연결할 수 있습니다. 탐색 창에서 Policies를 선택합니다.
10. 검색 상자에 정책 이름을 입력합니다.
11. 작업 메뉴에서 연결을 선택합니다.
12. 엔터티 유형별 필터링에서 IAM 사용자 또는 역할을 선택합니다.
13. 검색 필드에 사용하려는 사용자 또는 역할의 이름을 입력합니다.
14. 정책 연결을 선택하세요.

1단계: S3 Express One Zone 디렉터리 버킷에 도달하도록 게이트웨이 VPC 엔드포인트 구성

게이트웨이 가상 프라이빗 클라우드(VPC) 엔드포인트를 통해 영역 및 리전 API 작업에 모두 액세스할 수 있습니다. 게이트웨이 엔드포인트를 사용하면 트래픽이 NAT 게이트웨이를 통과하지 않고도 S3 Express One Zone에 도달할 수 있습니다. 게이트웨이 엔드포인트는 S3 Express One Zone을 사용할 때 최적의 네트워킹 경로를 제공하므로 게이트웨이 엔드포인트를 사용하는 것이 좋습니다. 인터넷 게이트웨이 또는 VPC의 NAT 디바이스를 사용하지 않고 추가 비용 없이 VPC에서 S3 Express One Zone 디렉터리 버킷에 액세스할 수 있습니다. 다음 절차를 따라 S3 Express One Zone 스토리지 클래스 객체와 디렉터리 버킷에 연결하는 게이트웨이 엔드포인트를 구성합니다.

S3 Express One Zone에 액세스하려면 표준 Amazon S3 엔드포인트와 다른 리전 및 영역 엔드포인트를 사용합니다. 사용하는 Amazon S3 API 작업에 따라 영역 엔드포인트 또는 리전 엔드포인트가 필요합니다. 엔드포인트 유형별로 지원되는 API 작업의 전체 목록은 S3 Express One Zone에서 지원되는 API 작업을 참조하세요. 게이트웨이 가상 프라이빗 클라우드(VPC) 엔드포인트를 통해 영역 및 리전 엔드포인트에 액세스해야 합니다.

다음 절차를 따라 S3 Express One Zone 스토리지 클래스 객체와 디렉터리 버킷에 연결하는 게이트웨이 엔드포인트를 생성합니다.

게이트웨이 VPC 엔드포인트를 구성하는 방법

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. 측면 탐색 모음의 가상 프라이빗 클라우드에서 엔드포인트를 선택합니다.
3. Create endpoint(엔드포인트 생성)을 선택합니다.

4. 엔드포인트의 이름을 지정합니다.
5. 서비스 범주(Service category)에서 AWS 서비스를 선택합니다.
6. 서비스에서 유형=게이트웨이 필터를 사용하여 검색한 다음 com.amazonaws.***region***.s3express 옆의 옵션 버튼을 선택합니다.
7. VPC에서 엔드포인트를 생성할 VPC를 선택합니다.
8. 라우팅 테이블(Route tables)에서 엔드포인트에서 사용할 라우팅 테이블을 선택합니다. Amazon VPC가 서버로 전송되는 트래픽을 가리키는 라우팅을 엔드포인트 네트워크 인터페이스에 자동으로 추가합니다.
9. 정책에서 모든 액세스를 선택하여 VPC 엔드포인트를 통한 모든 리소스에 대한 모든 보안 주체의 모든 작업을 허용합니다. 또는 사용자 지정을 선택하여 VPC 엔드포인트를 통해 리소스에 대한 작업을 수행하기 위해 보안 주체에 필요한 권한을 제어하는 VPC 엔드포인트 정책을 연결합니다.
10. Create endpoint(엔드포인트 생성)을 선택합니다.

게이트웨이 엔드포인트를 생성한 후에는 리전 API 엔드포인트와 영역 API 엔드포인트를 사용하여 Amazon S3 Express One Zone 스토리지 클래스 객체 및 디렉터리 버킷에 액세스할 수 있습니다.

2단계: S3 Express One Zone 디렉터리 버킷 생성

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 버킷을 생성하려는 리전을 선택합니다.

   > **Note**
   >
   > 지연 시간과 요금을 최소화하고 규제 요건을 충족하려면 가장 가까운 리전을 선택하십시오. 특정 리전에 저장된 객체는 사용자가 명시적으로 객체를 다른 리전으로 전송하지 않는 한 해당 리전을 벗어나지 않습니다. Amazon S3 AWS 리전 목록은 Amazon Web Services 일반 참조의 AWS 서비스 엔드포인트를 참조하십시오.

3. 왼쪽 탐색 창에서 디렉터리 버킷을 선택합니다.
4. 버킷 만들기를 선택합니다. 버킷 만들기 페이지가 열립니다.
5. 일반 구성에서 버킷이 생성될 AWS 리전을 확인합니다.

   버킷 유형에서 디렉터리를 선택합니다.

> **Note**
>
> - 디렉터리 버킷을 지원하지 않는 리전을 선택한 경우 버킷 유형 옵션이 사라지고 버킷 유형이 범용 버킷으로 기본 설정됩니다. 디렉터리 버킷을 생성하려면 지원되는 리전을 선택해야 합니다. 지원 디렉터리 버킷 및 Amazon S3 Express One Zone 스토리지 클래스를 지원하는 리전 목록은 the section called “S3 Express One Zone 가용 영역 및 리전” 섹션을 참조하세요.
> - 버킷을 생성한 후에는 버킷 유형을 변경할 수 없습니다.

> **Note**
>
> 버킷 생성 후에는 가용 영역을 변경할 수 없습니다.

6. 가용 영역의 경우 컴퓨팅 서비스의 로컬 가용 영역을 선택합니다. 지원 디렉터리 버킷 및 S3 Express One Zone 스토리지 클래스를 지원하는 가용 영역의 목록은 the section called “S3 Express One Zone 가용 영역 및 리전” 섹션을 참조하세요.

   가용 영역에서 확인란을 선택하여 가용 영역 중단 시 데이터를 사용할 수 없거나 데이터가 손실될 수 있음을 확인합니다.

   > **Important**
   >
   > 디렉터리 버킷은 단일 가용 영역 내의 여러 디바이스에 걸쳐 저장되지만, 디렉터리 버킷은 가용 영역 간에 데이터를 중복으로 저장하지는 않습니다.

7. 버킷 이름에서 디렉터리 버킷 이름을 입력합니다.

   디렉터리 버킷에는 다음 이름 지정 규칙이 적용됩니다.

   - 선택한 영역(AWS 가용 영역 또는 AWS 로컬 영역) 내에서 고유해야 합니다.
   - 이름은 최소 3자에서 최대 63자 사이여야 합니다(접미사 포함).
   - 소문자, 숫자, 하이픈(-)으로만 구성해야 합니다.
   - 문자나 숫자로 시작하고 끝나야 합니다.
   - `--`*`zone-id`*`--x-s3`을 접미사로 포함해야 합니다.
   - 버킷 이름은 접두사 `xn--`로 시작해서는 안됩니다.

- 버킷 이름은 접두사 `sthree-`로 시작해서는 안됩니다.
- 버킷 이름은 접두사 `sthree-configurator`로 시작해서는 안됩니다.
- 버킷 이름은 접두사 `amzn-s3-demo-`로 시작해서는 안됩니다.
- 버킷 이름은 접미사 `-s3alias`로 끝나서는 안됩니다. 이 접미사는 액세스 포인트 별칭 이름 용도로 예약되어 있습니다. 자세한 내용은 액세스 포인트 별칭 섹션을 참조하세요.
- 버킷 이름은 접미사 `--ol-s3`로 끝나서는 안됩니다. 이 접미사는 객체 Lambda 액세스 포인트 별칭 이름 용도로 예약되어 있습니다. 자세한 내용은 S3 버킷 객체 Lambda 액세스 포인트에 버킷 스타일 별칭을 사용하는 방법 섹션을 참조하세요.
- 버킷 이름은 접미사 `.mrap`로 끝나서는 안됩니다. 이 접미사는 다중 리전 액세스 포인트 이름 용도로 예약되어 있습니다. 자세한 내용은 Amazon S3 다중 리전 액세스 포인트 이름 지정 규칙 섹션을 참조하세요.

콘솔을 사용하여 디렉터리 버킷을 생성하면 제공하는 기본 이름에 접미사가 자동으로 추가됩니다. 이 접미사에는 선택한 가용 영역의 가용 영역 ID가 포함됩니다.

버킷을 생성한 후에는 해당 이름을 변경할 수 없습니다. 버킷 이름 지정에 대한 자세한 내용은 범용 버킷 이름 지정 규칙 섹션을 참조하십시오.

> **Important**
>
> 버킷 이름에 계정 번호와 같은 중요한 정보를 포함하지 마세요. 버킷 이름은 버킷의 객체를 가리키는 URL에 표시됩니다.

8. 객체 소유권에서 버킷 소유자 적용 설정이 자동으로 활성화되고 모든 액세스 제어 목록(ACL)이 비활성화됩니다. 디렉터리 버킷의 경우 ACL을 활성화할 수 없습니다.

   버킷 소유자 적용(기본값) – ACL이 비활성화되고 버킷 소유자는 범용 버킷의 모든 객체를 자동으로 소유하고 완전히 제어합니다. ACL은 더 이상 S3 범용 버킷의 데이터에 대한 권한에 영향을 주지 않습니다. 버킷은 정책을 독점적으로 사용하여 액세스 제어를 정의합니다.

9. 이 버킷의 퍼블릭 액세스 차단 설정에서 디렉터리 버킷의 퍼블릭 액세스 차단 설정이 자동으로 활성화됩니다. 디렉터리 버킷의 경우 이 설정을 수정할 수 없습니다. 퍼블릭 액세스 차단에 대한 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하십시오.
10. 기본 암호화를 구성하려면 암호화 유형 아래에서 다음 중 하나를 선택합니다.

    - Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)
    - AWS Key Management Service 키를 사용한 서버 측 암호화(SSE-KMS)

Amazon S3 서버 측 암호화를 사용하는 데이터 암호화에 대한 자세한 내용은 데이터 보호 및 암호화 섹션을 참조하십시오.

> **Important**
>
> 기본 암호화 구성에 대해 SSE-KMS 옵션을 사용할 경우 AWS KMS의 초당 요청 수(RPS) 제한이 적용됩니다. AWS KMS 할당량과 할당량 증대를 요청하는 방법에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 할당량을 참조하십시오.
> 기본 암호화를 활성화할 때 버킷 정책을 업데이트해야 할 수 있습니다. 자세한 내용은 크로스 계정 작업에 SSE-KMS 암호화 사용 섹션을 참조하세요.

11. Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)를 선택한 경우 버킷 키 아래에 활성화됨이 나타납니다. SSE-S3를 통한 기본 암호화를 사용하도록 디렉터리 버킷을 구성할 때 S3 버킷 키는 항상 활성화됩니다. S3 버킷 키는 디렉터리 버킷의 GET 및 PUT 작업에 대해 항상 활성화되며 비활성화할 수 없습니다. CopyObject, UploadPartCopy, 배치 운영의 Copy 작업 또는 import 작업을 통해 SSE-KMS로 암호화된 객체를 범용 버킷에서 디렉터리 버킷으로, 디렉터리 버킷에서 범용 버킷으로 또는 디렉터리 버킷 간에 복사하는 경우 S3 버킷 키는 지원되지 않습니다. 이 경우 Amazon S3는 KMS 암호화 개체에 대한 사본 요청이 있을 때마다 AWS KMS를 직접적으로 호출합니다.

    S3 버킷 키를 사용하면 Amazon S3에서 AWS KMS로의 요청 트래픽이 줄어 암호화 비용이 절감됩니다. 자세한 내용은 Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감 섹션을 참조하세요.

12. AWS Key Management Service 키를 사용한 서버 측 암호화(SSE-KMS)를 선택한 경우 AWS KMS 키에서 다음 방법 중 하나로 AWS Key Management Service 키를 지정하거나 새 키를 생성합니다.

    - 사용 가능한 KMS 키 목록에서 중에서 선택하려면 AWS KMS keys에서 선택을 선택하고 사용 가능한 AWS KMS keys에서 KMS 키를 선택합니다.

      고객 관리형 키만 이 목록에 표시됩니다. AWS 관리형 키(aws/s3)는 디렉터리 버킷에서 지원되지 않습니다. 고객 관리형 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 키 및 AWS 키를 참조하십시오.

    - KMS 키 ARN 또는 별칭을 입력하려면 AWS KMS key ARN 입력을 선택하고 KMS 키 ARN 또는 별칭을 AWS KMS key ARN에 입력합니다.

    - AWS KMS 콘솔에서 고객 관리형 키를 생성하려면 KMS 키 생성을 선택합니다.

AWS KMS key 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 키 생성을 참조하십시오.

> **Important**
>
> - SSE-KMS 구성은 버킷 수명 기간 동안 디렉터리 버킷당 고객 관리형 키 1개만 지원할 수 있습니다. AWS 관리형 키(`aws/s3`)는 지원되지 않습니다. 또한, SSE-KMS에 고객 관리형 키를 지정한 후에는 버킷의 SSE-KMS 구성을 위한 고객 관리형 키를 재정의할 수 없습니다.
>
> 다음과 같은 방법으로 버킷의 SSE-KMS 구성에 대해 지정한 고객 관리형 키를 식별할 수 있습니다.
>
> - 응답에서 `x-amz-server-side-encryption-aws-kms-key-id`의 값을 찾기 위해 `HeadObject` API 작업을 요청합니다.
>
> 데이터에 새 고객 관리형 키를 사용하려면 최신 고객 관리형 키가 있는 새 디렉터리 버킷에 기존 객체를 복사하는 것이 좋습니다.
> - 버킷과 동일한 AWS 리전에서 사용할 수 있는 KMS 키만 사용 가능합니다. Amazon S3 콘솔은 버킷과 동일한 리전에 있는 처음 100개의 KMS 키만 나열합니다. 목록에 없는 KMS 키를 사용하려면 KMS 키 ARN을 입력해야 합니다. 다른 계정에서 소유한 KMS 키를 사용하려면 먼저 해당 키에 대한 사용 권한이 있어야 하고, 다음 단계로 KMS 키 ARN을 입력해야 합니다. KMS 키의 크로스 계정 권한에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서에서 다른 계정에서 사용할 수 있는 KMS 키 만들기를 참조하십시오. SSE-KMS에 대한 자세한 내용은 디렉터리 버킷의 새 객체 업로드에 AWS KMS(SSE-KMS)를 사용한 서버 측 암호화 지정 섹션을 참조하십시오.
> - 디렉터리 버킷에서 서버 측 암호화에 AWS KMS key를 사용하는 경우 대칭 암호화 KMS 키를 선택해야 합니다. Amazon S3는 대칭 암호화 KMS 키만 지원하며 비대칭 KMS 키는 지원하지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 대칭 및 비대칭 KMS 키 식별을 참조하십시오.

Amazon S3에서 AWS KMS을(를) 사용하는 방법에 대한 자세한 내용은 디렉터리 버킷에서 AWS KMS 키(SSE-KMS)를 사용한 서버 측 암호화 사용 단원을 참조하십시오.

13. 버킷 생성을 선택합니다. 버킷을 생성한 후 버킷에 파일 및 폴더를 추가할 수 있습니다. 자세한 내용은 the section called “디렉터리 버킷의 객체 작업” 섹션을 참조하세요.

3단계: S3 Express One Zone 디렉터리 버킷으로 데이터 가져오기

이 단계를 완료하려면 객체를 포함하고 디렉터리 버킷과 같은 AWS 리전에 있는 범용 버킷이 있어야 합니다.

Amazon S3에서 디렉터리 버킷을 생성한 후 Amazon S3 콘솔에서 가져오기 작업을 사용하여 새 버킷에 데이터를 채울 수 있습니다. 가져오기를 사용하면 복사할 모든 객체를 개별적으로 지정할 필요 없이 데이터를 가져올 접두사 또는 범용 버킷을 선택할 수 있으므로 데이터를 쉽게 디렉터리 버킷으로 복사할 수 있습니다. 가져오기에서는 선택한 접두사 또는 범용 버킷의 객체를 복사하는 S3 배치 작업이 사용됩니다. S3 배치 작업의 작업 세부 정보 페이지를 통해 가져오기 복사 작업의 진행 상황을 모니터링할 수 있습니다.

가져오기 작업을 사용하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 다음으로 디렉터리 버킷이 위치한 가용 영역과 연결된 리전을 선택합니다.
3. 왼쪽 탐색 창에서 디렉터리 버킷을 선택합니다.
4. 객체를 가져올 대상 버킷 이름 옆에 있는 옵션 버튼을 선택합니다.
5. 가져오기를 선택합니다.
6. 소스에는 가져오려는 객체가 들어 있는 범용 버킷(또는 접두사를 포함한 버킷 경로)을 입력합니다. 목록에서 기존의 범용 버킷을 선택하려면 S3 찾아보기를 선택합니다.
7. 권한 섹션에서 IAM 역할을 자동 생성하도록 선택할 수 있습니다. 또는 목록에서 IAM 역할을 선택하거나, IAM 역할 ARN을 직접 입력할 수 있습니다.

   - Amazon S3가 대신 새 IAM 역할을 생성하도록 허용하려면 새 IAM 역할 생성을 선택합니다.

     > **Note**
     >
     > 소스 객체가 AWS Key Management Service(AWS KMS) 키를 통한 서버 측 암호화(SSE-KMS)로 암호화된 경우, 새 IAM 역할 생성 옵션을 선택하지 마세요. 대신 `kms:Decrypt` 권한이 있는 기존 IAM 역할을 지정하세요.
     > Amazon S3는 이 권한을 사용하여 객체를 복호화합니다. 그런 다음 가져오기 프로세스 중에 Amazon S3가 Amazon S3 관리형 키를 통한 서버 측 암호화(SSE-S3)를 사용하여 해당 객체를 다시 암호화합니다.

   - 목록에서 기존 IAM 역할을 선택하려면 기존 IAM 역할에서 선택을 선택합니다.

- Amazon 리소스 이름(ARN)을 입력하여 기존 IAM 역할을 지정하려면 IAM 역할 ARN 입력을 선택한 다음 해당 필드에 ARN을 입력합니다.

8. 대상 및 복사된 객체 설정 섹션에 표시된 정보를 검토합니다. 대상 섹션의 정보가 정확하면 가져오기를 선택하여 복사 작업을 시작합니다.

   Amazon S3 콘솔은 배치 작업 페이지에 새 작업의 상태를 표시합니다. 작업에 대한 자세한 내용을 보려면 작업 이름 옆의 옵션 버튼을 선택한 다음 작업 메뉴에서 세부 정보 보기를 선택합니다. 객체를 가져올 대상 디렉터리 버킷을 열려면 가져오기 대상 보기를 선택합니다.

4단계: S3 Express One Zone 디렉터리 버킷에 객체를 수동으로 업로드

디렉터리 버킷에 객체를 수동으로 업로드할 수도 있습니다.

객체를 수동으로 업로드하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 오른쪽 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 다음으로 디렉터리 버킷이 위치한 가용 영역과 연결된 리전을 선택합니다.
3. 왼쪽 탐색 창에서 디렉터리 버킷을 선택합니다.
4. 파일 및 폴더를 업로드하려는 버킷의 이름을 선택합니다.

   > **Note**
   >
   > 이 튜토리얼의 이전 단계에서 사용한 것과 동일한 디렉터리 버킷을 선택한 경우 디렉터리 버킷에는 가져오기 도구에서 업로드한 객체가 포함됩니다. 이러한 객체는 이제 S3 Express One Zone 스토리지 클래스에 저장되어 있습니다.

5. 객체 탭에서 업로드를 선택합니다.
6. 업로드 페이지에서 다음 중 하나를 수행합니다.

   - 파일 및 폴더를 점선으로 표시된 업로드 영역으로 끌어다 놓습니다.
   - 파일 추가 또는 폴더 추가를 선택하고 업로드할 파일 또는 폴더를 선택한 후 열기 또는 업로드를 선택합니다.

7. 체크섬에서 사용하려는 체크섬 함수를 선택합니다.

> **Note**
>
> S3 Express One Zone 스토리지 클래스에서 최상의 성능을 위해 CRC32 및 CRC32C를 사용하는 것이 좋습니다. 자세한 내용은 S3 추가 체크섬 모범 사례를 참조하세요.

(선택 사항) 크기가 16MB 미만인 단일 객체를 업로드하는 경우 미리 계산된 체크섬 값을 지정할 수도 있습니다. 미리 계산된 값을 제공하면 Amazon S3는 선택한 체크섬 함수를 사용하여 계산한 값과 비교합니다. 값이 일치하지 않으면 업로드가 시작되지 않습니다.

8. 권한 및 속성 섹션의 옵션은 자동으로 기본 설정으로 설정되며 수정할 수 없습니다. 퍼블릭 액세스 차단은 자동으로 활성화되며 디렉터리 버킷에 대해서는 S3 버전 관리 및 S3 객체 잠금을 활성화할 수 없습니다.

   (선택 사항) 키-값 쌍의 메타데이터를 객체에 추가하려면 속성 섹션을 확장한 다음 메타데이터 섹션에서 메타데이터 추가를 선택합니다.

9. 나열된 파일을 업로드하려면 업로드를 선택합니다.

   Amazon S3가 객체와 폴더를 업로드합니다. 업로드가 완료되면 업로드: 상태 페이지에서 성공 메시지를 볼 수 있습니다.

   디렉터리 버킷을 성공적으로 생성하고 객체를 버킷에 업로드했습니다.

5단계: S3 Express One Zone 디렉터리 버킷 비우기

Amazon S3 콘솔을 사용하여 Amazon S3 디렉터리 버킷을 비울 수 있습니다.

디렉터리 버킷을 비우려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 오른쪽 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 다음으로 디렉터리 버킷이 위치한 가용 영역과 연결된 리전을 선택합니다.
3. 왼쪽 탐색 창에서 디렉터리 버킷을 선택합니다.
4. 비우려는 버킷의 이름 옆에 있는 옵션 버튼을 선택한 후 비우기를 선택합니다.
5. 버킷 비우기 페이지에서 텍스트 필드에 **`permanently delete`**를 입력하여 해당 버킷 비우기를 확인한 후 비우기를 선택합니다.

6. 버킷 비우기: 상태 페이지에서 버킷 비우기 프로세스의 진행 상황을 모니터링합니다.

6단계: S3 Express One Zone 디렉터리 버킷 삭제

디렉터리 버킷을 비우고 진행 중인 모든 멀티파트 업로드를 중단한 후 Amazon S3 콘솔을 사용하여 버킷을 삭제할 수 있습니다.

디렉터리 버킷을 삭제하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 오른쪽 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 다음으로 디렉터리 버킷이 위치한 가용 영역과 연결된 리전을 선택합니다.
3. 왼쪽 탐색 창에서 디렉터리 버킷을 선택합니다.
4. 디렉터리 버킷 목록에서, 삭제할 버킷의 이름 옆에 있는 옵션 버튼을 선택합니다.
5. 삭제를 선택합니다.
6. 버킷 삭제 페이지에서 텍스트 필드에 버킷 이름을 입력하여 버킷 삭제를 확인합니다.

> **Important**
>
> 디렉터리 버킷 삭제는 실행 취소할 수 없습니다.

7. 디렉터리 버킷을 삭제하려면 버킷 삭제를 선택합니다.

다음 단계

이 튜토리얼에서는 디렉터리 버킷을 생성하고 S3 Express One Zone 스토리지 클래스를 사용하는 방법을 살펴봤습니다. 이 튜토리얼을 완료한 후에는 S3 Express One Zone 스토리지 클래스에서 사용할 관련 AWS 서비스를 살펴볼 수 있습니다.

S3 Express One Zone 스토리지 클래스와 함께 다음 AWS 서비스를 사용하여 지연 시간이 짧은 구체적인 사용 사례를 지원할 수 있습니다.

- Amazon Elastic Compute Cloud(Amazon EC2) – Amazon EC2는 AWS 클라우드에서 안전하고 확장 가능한 컴퓨팅 용량을 제공합니다. Amazon EC2를 사용하면 하드웨어에 선투자할 필요성이 감소되어 더 빠르게 애플리케이션을 개발하고 배포할 수 있습니다. Amazon EC2를 사용하여 원하는 수의 가상 서버를 빌드하고 보안 및 네트워킹을 구성하며 스토리지를 관리할 수 있습니다.

- AWS Lambda - Lambda는 서버를 프로비저닝하거나 관리하지 않고도 코드를 실행할 수 있게 해주는 컴퓨팅 서비스입니다. 버킷에 알림 설정을 구성하고 Amazon S3에 함수의 리소스 기반 권한 정책에 따라 함수를 간접 호출할 수 있는 권한을 부여합니다.
- Amazon Elastic Kubernetes Service(Amazon EKS) - Amazon EKS는 AWS에 자체 Kubernetes 컨트롤 플레인을 설치, 운영 및 유지 관리할 필요가 없는 관리형 서비스입니다. Kubernetes는 컨테이너화된 애플리케이션의 관리, 규모 조정 및 배포를 자동화하는 오픈 소스 시스템입니다.
- Amazon Elastic Container Service(Amazon ECS) Amazon ECS는 컨테이너화된 애플리케이션을 쉽게 배포, 관리, 규모 조정할 수 있도록 도와주는 완전 관리형 컨테이너 오케스트레이션 서비스입니다.
- Amazon EMR – Amazon EMR은 AWS에서 Apache Hadoop 및 Apache Spark와 같은 빅 데이터 프레임워크 실행을 단순화하여 방대한 양의 데이터를 처리하고 분석하는 관리형 클러스터 플랫폼입니다.
- Amazon Athena – Amazon Athena는 표준 SQL을 사용해 Amazon S3에서 직접 데이터를 간편하게 분석할 수 있는 대화식 쿼리 서비스입니다. 또한 Athena를 사용하면 리소스를 계획, 구성 또는 관리할 필요 없이 Apache Spark를 사용하여 데이터 분석을 대화식으로 실행할 수 있습니다. Athena에서 Apache Spark 애플리케이션을 실행하는 경우 처리를 위해 Spark 코드를 제출하고 결과를 직접 수신합니다.
- AWS Glue Data Catalog - AWS Glue는 분석 사용자가 여러 소스의 데이터를 쉽게 검색, 준비, 이동, 통합할 수 있도록 하는 서버리스 데이터 통합 서비스입니다. 분석, 기계 학습 및 애플리케이션 개발에 AWS Glue를 사용할 수 있습니다. AWS Glue Data Catalog는 조직의 데이터 세트에 대한 메타데이터를 저장하는 중앙 집중식 리포지토리입니다. 즉, 데이터 소스의 위치, 스키마 및 런타임 지표에 대한 인덱스로 동작합니다.
- Amazon SageMaker Runtime 모델 훈련 - Amazon SageMaker Runtime은 완전관리형 기계 학습 서비스입니다. 데이터 과학자와 개발자들은 SageMaker Runtime으로 기계 학습 모델을 빠르고 쉽게 빌드하고 훈련시킬 수 있으며, 그런 다음 모델을 프로덕션 지원 호스팅 환경에 직접 배포할 수 있습니다.

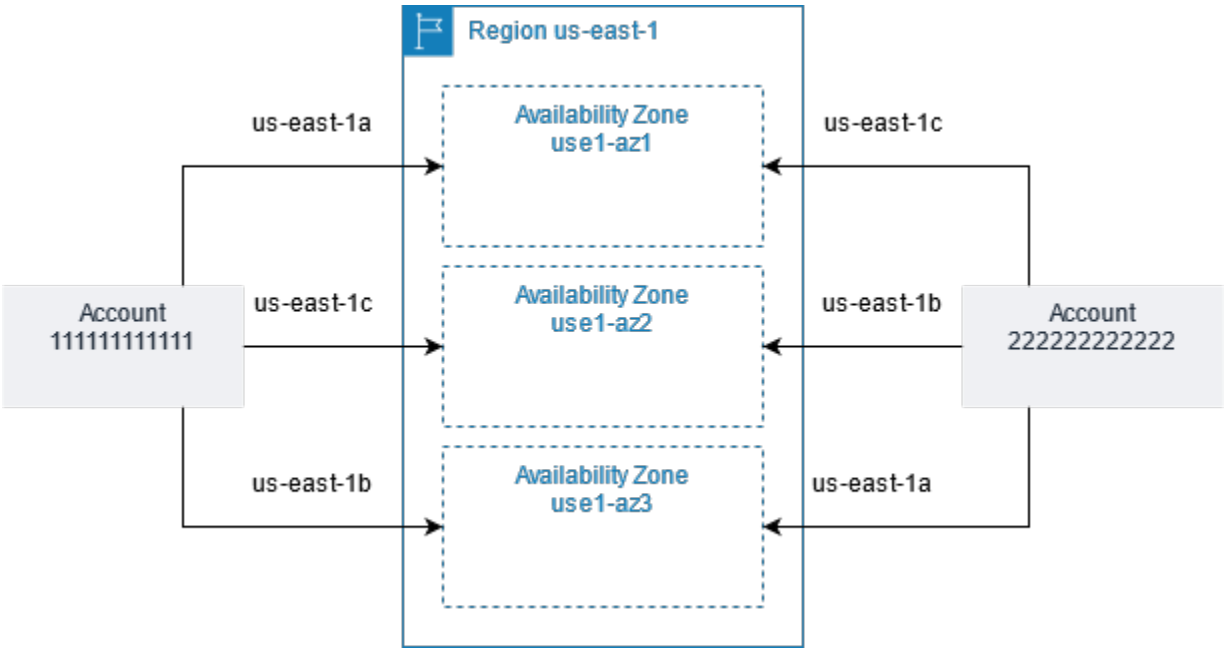
S3 Express One Zone에 대한 자세한 내용은 S3 Express One Zone이란? 및 S3 Express One Zone의 차이점을 참조하세요.

S3 Express One Zone 가용 영역 및 리전

가용 영역은 AWS 리전에 중복 전원, 네트워킹 및 연결이 있는 하나 이상의 개별 데이터 센터입니다. 지연 시간이 짧은 검색을 최적화하기 위해 Amazon S3 Express One Zone 스토리지 클래스의 객체는 컴퓨팅 워크로드와 동일한 위치에 있는 로컬 단일 가용 영역의 S3 디렉터리 버킷에 중복 저장됩니다. 디렉터리 버킷을 생성할 때 버킷이 위치할 가용 영역과 AWS 리전을 선택합니다.

AWS는 물리적 가용 영역을 각 AWS 계정의 가용 영역 이름에 무작위로 매핑합니다. 이 접근 방식을 사용하면 리소스를 각 리전의 첫 번째 가용 영역에 집중적으로 배치하는 대신 AWS 리전 리전의 가용 영역 전체에 분산시킬 수 있습니다. 따라서 AWS 계정의 `us-east-1a` 가용 영역은 다른 AWS 계정의 `us-east-1a`와 동일한 물리적 위치를 나타내지 않을 수 있습니다. 자세한 내용은 Amazon EC2 사용 설명서에서 리전 및 가용 영역을 참조하세요.

계정에 대해 가용 영역을 조정하려면 가용 영역에 대한 고유하고 일관된 식별자인 AZ ID를 사용해야 합니다. 예를 들어 `use1-az1`은 `us-east-1` 리전의 AZ ID이고, 모든 AWS 계정에서 물리적 위치가 동일합니다. 다음 그림은 가용 영역 이름은 계정마다 다르게 매핑될 수 있지만 AZ ID는 모든 계정에서 동일하다는 것을 보여줍니다.



S3 Express One Zone 스토리지 클래스를 사용하면 단일 가용 영역 내의 여러 디바이스에 데이터가 중복으로 저장됩니다. S3 Express One Zone은 단일 가용 영역 내에서 99.95%의 가용성을 제공하도록 설계되었으며 Amazon S3 서비스 수준 계약의 지원을 받습니다. 자세한 내용은 가용 영역 섹션을 참조하세요.

다음 테이블에는 S3 Express One Zone에서 지원되는 리전 및 가용 영역이 나와 있습니다.

리전 이름	리전 코드	가용 영역 ID				
미국 동부(버지니아 북부)	`us-east-1`	`use1-az4`				
		`use1-az5`				

리전 이름	리전 코드	가용 영역 ID
		use1-az6
미국 동부(오하이오)	us-east-2	use2-az1
		use2-az2
미국 서부(오리건)	us-west-2	usw2-az1
		usw2-az3
		usw2-az4
아시아 태평양(뭄바이)	ap-south-1	aps1-az1
		aps1-az3
아시아 태평양(도쿄)	ap-northeast-1	apne1-az1
		apne1-az4
유럽(아일랜드)	eu-west-1	euw1-az1
		euw1-az3
유럽(스톡홀름)	eu-north-1	eun1-az1
		eun1-az2
		eun1-az3

가용 영역의 디렉터리 버킷에 대한 네트워킹

다음 주제에서는 게이트웨이 VPC 엔드포인트를 사용하여 S3 Express One Zone에 액세스하기 위한 네트워킹 요구 사항을 설명합니다.

주제

- 가용 영역의 디렉터리 버킷에 대한 엔드포인트
- VPC 게이트웨이 엔드포인트 구성

가용 영역의 디렉터리 버킷에 대한 엔드포인트

다음 테이블에는 각 리전 및 가용 영역에서 사용할 수 있는 리전 및 영역 API 엔드포인트가 나와 있습니다.

VPC 게이트웨이 엔드포인트 구성

다음 절차를 따라 Amazon S3 Express One Zone 스토리지 클래스 객체와 디렉터리 버킷에 연결하는 게이트웨이 엔드포인트를 생성합니다.

게이트웨이 VPC 엔드포인트를 구성하는 방법

1. Amazon VPC 콘솔을 엽니다.
2. 탐색 창에서 엔드포인트를 선택합니다.
3. Create endpoint(엔드포인트 생성)을 선택합니다.
4. 엔드포인트의 이름을 지정합니다.
5. 서비스 범주(Service category)에서 AWS 서비스를 선택합니다.
6. 서비스에서 유형=게이트웨이 필터를 추가한 다음 com.amazonaws.***region***.s3express 옆의 옵션 버튼을 선택합니다.
7. VPC에서 엔드포인트를 생성할 VPC를 선택합니다.
8. 라우팅 테이블(Route tables)에서 엔드포인트에서 사용할 라우팅 테이블을 선택합니다. Amazon VPC가 서버로 전송되는 트래픽을 가리키는 라우팅을 엔드포인트 네트워크 인터페이스에 자동으로 추가합니다.
9. 정책에서 모든 액세스를 선택하여 VPC 엔드포인트를 통한 모든 리소스에 대한 모든 보안 주체의 모든 작업을 허용합니다. 또는 사용자 지정을 선택하여 VPC 엔드포인트를 통해 리소스에 대한 작업을 수행하기 위해 보안 주체에 필요한 권한을 제어하는 VPC 엔드포인트 정책을 연결합니다.
10. (선택 사항) 태그를 추가하려면 새 태그 추가를 선택하고 태그 키와 태그 값을 입력합니다.
11. Create endpoint(엔드포인트 생성)을 선택합니다.

게이트웨이 엔드포인트를 생성한 후에는 리전 API 엔드포인트와 영역 API 엔드포인트를 사용하여 Amazon S3 Express One Zone 스토리지 클래스 객체 및 디렉터리 버킷에 액세스할 수 있습니다.

가용 영역에서 디렉터리 버킷 생성

Amazon S3 Express One Zone 스토리지 클래스 사용을 시작하기 위해 디렉터리 버킷을 생성합니다. S3 Express One Zone 스토리지 클래스는 디렉터리 버킷에만 사용할 수 있습니다. S3 Express One Zone 스토리지 클래스는 지연 시간이 짧은 사용 사례를 지원하고 단일 가용 영역 내에서 더 빠른 데이

터 처리를 제공합니다. 애플리케이션이 성능에 민감하고 10밀리초 미만의 PUT 및 GET 지연 시간이 필요한 경우 S3 Express One Zone 스토리지 클래스를 사용할 수 있도록 디렉터리 버킷을 생성하는 것이 좋습니다.

Amazon S3 버킷에는 범용 버킷과 디렉터리 버킷이라는 두 가지 유형이 있습니다. 애플리케이션 및 성능 요구 사항에 가장 적합한 버킷 유형을 선택해야 합니다. 범용 버킷은 S3 버킷 본래의 유형입니다. 범용 버킷은 대부분의 사용 사례와 액세스 패턴에 권장되며 S3 Express One Zone을 제외한 모든 스토리지 클래스에 객체를 저장할 수 있습니다. 범용 버킷에 대한 자세한 내용은 범용 버킷 개요 섹션을 참조하세요.

디렉터리 버킷은 S3 Express One Zone 스토리지 클래스를 사용합니다. S3 Express One Zone 스토리지 클래스는 일관되게 10밀리초 미만의 지연 시간이 필요한 워크로드 또는 성능이 중요한 애플리케이션에 사용하도록 설계되었습니다. S3 Express One Zone은 객체 스토리지를 컴퓨팅 리소스와 함께 배치하는 옵션을 제공하면서 단일 가용 영역 선택이 가능한 최초의 S3 스토리지 클래스로, 최고의 액세스 속도를 제공합니다. 디렉터리 버킷을 생성할 때 Amazon EC2, Amazon Elastic Kubernetes Service 또는 Amazon Elastic Container Service(Amazon ECS) 컴퓨팅 인스턴스와 같은 위치의 로컬 가용 영역과 AWS 리전을 선택적으로 지정하여 성능을 최적화할 수 있습니다.

S3 Express One Zone 스토리지 클래스를 사용하면 단일 가용 영역 내의 여러 디바이스에 데이터가 중복으로 저장됩니다. S3 Express One Zone은 단일 가용 영역 내에서 99.95%의 가용성을 제공하도록 설계되었으며 Amazon S3 서비스 수준 계약의 지원을 받습니다. 자세한 내용은 가용 영역 섹션을 참조하세요.

디렉터리 버킷은 범용 버킷의 플랫 스토리지 구조와 달리 데이터를 계층적으로 디렉터리에 구성합니다. 디렉터리 버킷에는 접두사 제한이 없으며 개별 디렉터리는 수평적으로 확장할 수 있습니다.

디렉터리 버킷에 대한 자세한 내용은 디렉터리 버킷 작업 섹션을 참조하세요.

디렉터리 버킷 이름

디렉터리 버킷 이름은 이 형식을 따라야 하며 디렉터리 버킷 이름 지정 규칙을 준수해야 합니다.

```
bucket-base-name--zone-id--x-s3
```

예를 들어, 다음 디렉터리 버킷 이름에는 가용 영역 ID인 usw2-az1이 포함되어 있습니다.

```
bucket-base-name--usw2-az1--x-s3
```

디렉터리 버킷 이름 지정 규칙에 대한 자세한 내용은 디렉터리 버킷 이름 지정 규칙 섹션을 참조하세요.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 버킷을 생성하려는 리전을 선택합니다.

   > **Note**
   >
   > 지연 시간과 요금을 최소화하고 규제 요건을 충족하려면 가장 가까운 리전을 선택하십시오. 특정 리전에 저장된 객체는 사용자가 명시적으로 객체를 다른 리전으로 전송하지 않는 한 해당 리전을 벗어나지 않습니다. Amazon S3 AWS 리전 목록은 Amazon Web Services 일반 참조의 AWS 서비스 엔드포인트를 참조하십시오.

3. 왼쪽 탐색 창에서 디렉터리 버킷을 선택합니다.
4. 버킷 만들기를 선택합니다. 버킷 만들기 페이지가 열립니다.
5. 일반 구성에서 버킷이 생성될 AWS 리전을 확인합니다.

   버킷 유형에서 디렉터리를 선택합니다.

   > **Note**
   >
   > - 디렉터리 버킷을 지원하지 않는 리전을 선택한 경우 버킷 유형 옵션이 사라지고 버킷 유형이 범용 버킷으로 기본 설정됩니다. 디렉터리 버킷을 생성하려면 지원되는 리전을 선택해야 합니다. 지원 디렉터리 버킷 및 Amazon S3 Express One Zone 스토리지 클래스를 지원하는 리전 목록은 the section called “S3 Express One Zone 가용 영역 및 리전” 섹션을 참조하세요.
   > - 버킷을 생성한 후에는 버킷 유형을 변경할 수 없습니다.

   > **Note**
   >
   > 버킷 생성 후에는 가용 영역을 변경할 수 없습니다.

6. 가용 영역의 경우 컴퓨팅 서비스의 로컬 가용 영역을 선택합니다. 지원 디렉터리 버킷 및 S3 Express One Zone 스토리지 클래스를 지원하는 가용 영역의 목록은 the section called “S3 Express One Zone 가용 영역 및 리전” 섹션을 참조하세요.

가용 영역에서 확인란을 선택하여 가용 영역 중단 시 데이터를 사용할 수 없거나 데이터가 손실될 수 있음을 확인합니다.

> **Important**
>
> 디렉터리 버킷은 단일 가용 영역 내의 여러 디바이스에 걸쳐 저장되지만, 디렉터리 버킷은 가용 영역 간에 데이터를 중복으로 저장하지는 않습니다.

7. 버킷 이름에서 디렉터리 버킷 이름을 입력합니다.

   디렉터리 버킷에는 다음 이름 지정 규칙이 적용됩니다.

   - 선택한 영역(AWS 가용 영역 또는 AWS 로컬 영역) 내에서 고유해야 합니다.
   - 이름은 최소 3자에서 최대 63자 사이여야 합니다(접미사 포함).
   - 소문자, 숫자, 하이픈(-)으로만 구성해야 합니다.
   - 문자나 숫자로 시작하고 끝나야 합니다.
   - `--`*`zone-id`*`--x-s3`을 접미사로 포함해야 합니다.
   - 버킷 이름은 접두사 `xn--`로 시작해서는 안됩니다.
   - 버킷 이름은 접두사 `sthree-`로 시작해서는 안됩니다.
   - 버킷 이름은 접두사 `sthree-configurator`로 시작해서는 안됩니다.
   - 버킷 이름은 접두사 `amzn-s3-demo-`로 시작해서는 안됩니다.
   - 버킷 이름은 접미사 `-s3alias`로 끝나서는 안됩니다. 이 접미사는 액세스 포인트 별칭 이름 용도로 예약되어 있습니다. 자세한 내용은 액세스 포인트 별칭 섹션을 참조하세요.
   - 버킷 이름은 접미사 `--ol-s3`로 끝나서는 안됩니다. 이 접미사는 객체 Lambda 액세스 포인트 별칭 이름 용도로 예약되어 있습니다. 자세한 내용은 S3 버킷 객체 Lambda 액세스 포인트에 버킷 스타일 별칭을 사용하는 방법 섹션을 참조하세요.
   - 버킷 이름은 접미사 `.mrap`로 끝나서는 안됩니다. 이 접미사는 다중 리전 액세스 포인트 이름 용도로 예약되어 있습니다. 자세한 내용은 Amazon S3 다중 리전 액세스 포인트 이름 지정 규칙 섹션을 참조하세요.

   콘솔을 사용하여 디렉터리 버킷을 생성하면 제공하는 기본 이름에 접미사가 자동으로 추가됩니다. 이 접미사에는 선택한 가용 영역의 가용 영역 ID가 포함됩니다.

   버킷을 생성한 후에는 해당 이름을 변경할 수 없습니다. 버킷 이름 지정에 대한 자세한 내용은 범용 버킷 이름 지정 규칙 섹션을 참조하십시오.

> **Important**
>
> 버킷 이름에 계정 번호와 같은 중요한 정보를 포함하지 마세요. 버킷 이름은 버킷의 객체를 가리키는 URL에 표시됩니다.

8. 객체 소유권에서 버킷 소유자 적용 설정이 자동으로 활성화되고 모든 액세스 제어 목록(ACL)이 비활성화됩니다. 디렉터리 버킷의 경우 ACL을 활성화할 수 없습니다.

   버킷 소유자 적용(기본값) – ACL이 비활성화되고 버킷 소유자는 범용 버킷의 모든 객체를 자동으로 소유하고 완전히 제어합니다. ACL은 더 이상 S3 범용 버킷의 데이터에 대한 권한에 영향을 주지 않습니다. 버킷은 정책을 독점적으로 사용하여 액세스 제어를 정의합니다.

9. 이 버킷의 퍼블릭 액세스 차단 설정에서 디렉터리 버킷의 퍼블릭 액세스 차단 설정이 자동으로 활성화됩니다. 디렉터리 버킷의 경우 이 설정을 수정할 수 없습니다. 퍼블릭 액세스 차단에 대한 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하십시오.

10. 기본 암호화를 구성하려면 암호화 유형 아래에서 다음 중 하나를 선택합니다.

    - Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)
    - AWS Key Management Service 키를 사용한 서버 측 암호화(SSE-KMS)

    Amazon S3 서버 측 암호화를 사용하는 데이터 암호화에 대한 자세한 내용은 데이터 보호 및 암호화 섹션을 참조하십시오.

    > **Important**
    >
    > 기본 암호화 구성에 대해 SSE-KMS 옵션을 사용할 경우 AWS KMS의 초당 요청 수(RPS) 제한이 적용됩니다. AWS KMS 할당량과 할당량 증대를 요청하는 방법에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 할당량을 참조하십시오.
    > 기본 암호화를 활성화할 때 버킷 정책을 업데이트해야 할 수 있습니다. 자세한 내용은 크로스 계정 작업에 SSE-KMS 암호화 사용 섹션을 참조하세요.

11. Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)를 선택한 경우 버킷 키 아래에 활성화됨이 나타납니다. SSE-S3를 통한 기본 암호화를 사용하도록 디렉터리 버킷을 구성할 때 S3 버킷 키는 항상 활성화됩니다. S3 버킷 키는 디렉터리 버킷의 GET 및 PUT 작업에 대해 항상 활성화되며 비활성화할 수 없습니다. CopyObject, UploadPartCopy, 배치 운영의 Copy 작업 또는 import 작업을 통해 SSE-KMS로 암호화된 객체를 범용 버킷에서 디렉터리 버킷으로, 디렉터리 버킷에서 범용 버킷으로 또는 디렉터리 버킷 간에 복사하는 경우 S3 버킷 키는 지원되지 않습니다. 이 경우

Amazon S3는 KMS 암호화 개체에 대한 사본 요청이 있을 때마다 AWS KMS를 직접적으로 호출합니다.

S3 버킷 키를 사용하면 Amazon S3에서 AWS KMS로의 요청 트래픽이 줄어 암호화 비용이 절감됩니다. 자세한 내용은 Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감 섹션을 참조하세요.

12. AWS Key Management Service 키를 사용한 서버 측 암호화(SSE-KMS)를 선택한 경우 AWS KMS 키에서 다음 방법 중 하나로 AWS Key Management Service 키를 지정하거나 새 키를 생성합니다.

    - 사용 가능한 KMS 키 목록에서 중에서 선택하려면 AWS KMS keys에서 선택을 선택하고 사용 가능한 AWS KMS keys에서 KMS 키를 선택합니다.

      고객 관리형 키만 이 목록에 표시됩니다. AWS 관리형 키(aws/s3)는 디렉터리 버킷에서 지원되지 않습니다. 고객 관리형 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 키 및 AWS 키를 참조하십시오.

    - KMS 키 ARN 또는 별칭을 입력하려면 AWS KMS key ARN 입력을 선택하고 KMS 키 ARN 또는 별칭을 AWS KMS key ARN에 입력합니다.

    - AWS KMS 콘솔에서 고객 관리형 키를 생성하려면 KMS 키 생성을 선택합니다.

      AWS KMS key 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 키 생성을 참조하십시오.

    Important

    - SSE-KMS 구성은 버킷 수명 기간 동안 디렉터리 버킷당 고객 관리형 키 1개만 지원할 수 있습니다. AWS 관리형 키(aws/s3)는 지원되지 않습니다. 또한, SSE-KMS에 고객 관리형 키를 지정한 후에는 버킷의 SSE-KMS 구성을 위한 고객 관리형 키를 재정의할 수 없습니다.

      다음과 같은 방법으로 버킷의 SSE-KMS 구성에 대해 지정한 고객 관리형 키를 식별할 수 있습니다.

      - 응답에서 `x-amz-server-side-encryption-aws-kms-key-id`의 값을 찾기 위해 `HeadObject` API 작업을 요청합니다.

      데이터에 새 고객 관리형 키를 사용하려면 최신 고객 관리형 키가 있는 새 디렉터리 버킷에 기존 객체를 복사하는 것이 좋습니다.

- 버킷과 동일한 AWS 리전에서 사용할 수 있는 KMS 키만 사용 가능합니다. Amazon S3 콘솔은 버킷과 동일한 리전에 있는 처음 100개의 KMS 키만 나열합니다. 목록에 없는 KMS 키를 사용하려면 KMS 키 ARN을 입력해야 합니다. 다른 계정에서 소유한 KMS 키를 사용하려면 먼저 해당 키에 대한 사용 권한이 있어야 하고, 다음 단계로 KMS 키 ARN을 입력해야 합니다. KMS 키의 크로스 계정 권한에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서에서 다른 계정에서 사용할 수 있는 KMS 키 만들기를 참조하십시오. SSE-KMS에 대한 자세한 내용은 디렉터리 버킷의 새 객체 업로드에 AWS KMS(SSE-KMS)를 사용한 서버 측 암호화 지정 섹션을 참조하십시오.
- 디렉터리 버킷에서 서버 측 암호화에 AWS KMS key를 사용하는 경우 대칭 암호화 KMS 키를 선택해야 합니다. Amazon S3는 대칭 암호화 KMS 키만 지원하며 비대칭 KMS 키는 지원하지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 대칭 및 비대칭 KMS 키 식별을 참조하십시오.

Amazon S3에서 AWS KMS을(를) 사용하는 방법에 대한 자세한 내용은 디렉터리 버킷에서 AWS KMS 키(SSE-KMS)를 사용한 서버 측 암호화 사용 단원을 참조하십시오.

13. 버킷 생성을 선택합니다. 버킷을 생성한 후 버킷에 파일 및 폴더를 추가할 수 있습니다. 자세한 내용은 the section called “디렉터리 버킷의 객체 작업” 섹션을 참조하세요.

AWS SDK 사용

SDK for Go

이 예시는 AWS SDK for Go를 사용하여 디렉터리 버킷을 생성하는 방법을 보여줍니다.

Example

```
var bucket = "..."

func runCreateBucket(c *s3.Client) {
    resp, err := c.CreateBucket(context.Background(), &s3.CreateBucketInput{
        Bucket: &bucket,
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            Location: &types.LocationInfo{
                Name: aws.String("usw2-az1"),
                Type: types.LocationTypeAvailabilityZone,
            },
            Bucket: &types.BucketInfo{
                DataRedundancy: types.DataRedundancySingleAvailabilityZone,
```

```
                Type:           types.BucketTypeDirectory,
            },
        },
    })
    var terr *types.BucketAlreadyOwnedByYou
    if errors.As(err, &terr) {
        fmt.Printf("BucketAlreadyOwnedByYou: %s\n", aws.ToString(terr.Message))
        fmt.Printf("noop...\n")
        return
    }
    if err != nil {
        log.Fatal(err)
    }

    fmt.Printf("bucket created at %s\n", aws.ToString(resp.Location))
}
```

SDK for Java 2.x

이 예시는 AWS SDK for Java 2.x를 사용하여 디렉터리 버킷을 생성하는 방법을 보여줍니다.

Example

```
public static void createBucket(S3Client s3Client, String bucketName) {

    //Bucket name format is {base-bucket-name}--{az-id}--x-s3
    //example: doc-example-bucket--usw2-az1--x-s3 is a valid name for a directory
 bucket created in
    //Region us-west-2, Availability Zone 2

    CreateBucketConfiguration bucketConfiguration =
 CreateBucketConfiguration.builder()
             .location(LocationInfo.builder()
                     .type(LocationType.AVAILABILITY_ZONE)
                     .name("usw2-az1").build()) //this must match the Region and
 Availability Zone in your bucket name
             .bucket(BucketInfo.builder()
                    .type(BucketType.DIRECTORY)
                    .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                    .build()).build();
    try {

             CreateBucketRequest bucketRequest =
 CreateBucketRequest.builder().bucket(bucketName).createBucketConfiguration(bucketConfigurat
```

```
            CreateBucketResponse response = s3Client.createBucket(bucketRequest);
            System.out.println(response);
    }

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

AWS SDK for JavaScript

이 예시는 AWS SDK for JavaScript을 사용하여 디렉터리 버킷을 생성하는 방법을 보여줍니다.

Example

```
// file.mjs, run with Node.js v16 or higher
// To use with the preview build, place this in a folder
// inside the preview build directory, such as /aws-sdk-js-v3/workspace/

import { S3 } from "@aws-sdk/client-s3";

const region = "us-east-1";
const zone = "use1-az4";
const suffix = `${zone}--x-s3`;

const s3 = new S3({ region });

const bucketName = `...--${suffix}`;

const createResponse = await s3.createBucket(
    { Bucket: bucketName,
      CreateBucketConfiguration: {Location: {Type: "AvailabilityZone", Name: zone},
      Bucket: { Type: "Directory", DataRedundancy: "SingleAvailabilityZone" }}
    }
   );
```

SDK for .NET

이 예시는 SDK for .NET을 사용하여 디렉터리 버킷을 생성하는 방법을 보여줍니다.

Example

```
using (var amazonS3Client = new AmazonS3Client())
{
    var putBucketResponse = await amazonS3Client.PutBucketAsync(new PutBucketRequest
    {

       BucketName = "DOC-EXAMPLE-BUCKET--usw2-az1--x-s3",
       PutBucketConfiguration = new PutBucketConfiguration
       {
         BucketInfo = new BucketInfo { DataRedundancy =
 DataRedundancy.SingleAvailabilityZone, Type = BucketType.Directory },
         Location = new LocationInfo { Name = "usw2-az1", Type =
 LocationType.AvailabilityZone }
       }
     }).ConfigureAwait(false);
}
```

SDK for PHP

이 예시는 AWS SDK for PHP를 사용하여 디렉터리 버킷을 생성하는 방법을 보여줍니다.

Example

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

$s3Client = new S3Client([

    'region'      => 'us-east-1',
]);


$result = $s3Client->createBucket([
    'Bucket' => 'doc-example-bucket--use1-az4--x-s3',
    'CreateBucketConfiguration' => [
        'Location' => ['Name'=> 'use1-az4', 'Type'=> 'AvailabilityZone'],
        'Bucket' => ["DataRedundancy" => "SingleAvailabilityZone" ,"Type" =>
 "Directory"]   ],
]);
```

SDK for Python

이 예시는 AWS SDK for Python (Boto3)을 사용하여 디렉터리 버킷을 생성하는 방법을 보여줍니다.

Example

```
import logging
import boto3
from botocore.exceptions import ClientError

def create_bucket(s3_client, bucket_name, availability_zone):
    '''
    Create a directory bucket in a specified Availability Zone

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket to create; for example, 'doc-example-bucket--usw2-
az1--x-s3'
    :param availability_zone: String; Availability Zone ID to create the bucket in,
 for example, 'usw2-az1'
    :return: True if bucket is created, else False
    '''

    try:
        bucket_config = {
                'Location': {
                    'Type': 'AvailabilityZone',
                    'Name': availability_zone
                },
                'Bucket': {
                    'Type': 'Directory',
                    'DataRedundancy': 'SingleAvailabilityZone'
                }
            }
        s3_client.create_bucket(
            Bucket = bucket_name,
            CreateBucketConfiguration = bucket_config
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True


if __name__ == '__main__':
    bucket_name = 'BUCKET_NAME'
    region = 'us-west-2'
    availability_zone = 'usw2-az1'
    s3_client = boto3.client('s3', region_name = region)
```

```
    create_bucket(s3_client, bucket_name, availability_zone)
```

SDK for Ruby

이 예시는 AWS SDK for Ruby를 사용하여 디렉터리 버킷을 생성하는 방법을 보여줍니다.

Example

```
s3 = Aws::S3::Client.new(region:'us-west-2')
s3.create_bucket(
  bucket: "bucket_base_name--az_id--x-s3",
  create_bucket_configuration: {
    location: { name: 'usw2-az1', type: 'AvailabilityZone' },
    bucket: { data_redundancy: 'SingleAvailabilityZone', type: 'Directory' }
  }
)
```

AWS CLI 사용

이 예시는 AWS CLI을 사용하여 디렉터리 버킷을 생성하는 방법을 보여줍니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

디렉터리 버킷을 생성할 때는 구성 세부 정보를 제공하고 다음과 같은 명명 규칙을 사용해야 합니다. *bucket-base-name*--*zone-id*--x-s3

```
aws s3api create-bucket
--bucket bucket-base-name--zone-id--x-s3
--create-bucket-configuration 'Location={Type=AvailabilityZone,Name=usw2-
az1},Bucket={DataRedundancy=SingleAvailabilityZone,Type=Directory}'
--region us-west-2
```

자세한 내용은 AWS Command Line Interface의 create-bucket을 참조하세요.

가용 영역의 디렉터리 버킷에 대한 리전 및 영역 엔드포인트

게이트웨이 VPC 엔드포인트를 사용하여 S3 Express One Zone에 저장된 객체와 디렉터리 버킷에 액세스할 수 있습니다. 디렉터리 버킷은 리전 및 영역 API 엔드포인트를 사용합니다. 사용하는 Amazon S3 API 작업에 따라 리전 엔드포인트 또는 영역 엔드포인트가 필요합니다. 게이트웨이 엔드포인트 사용에 따르는 추가 요금은 없습니다.

버킷 수준(즉 컨트롤 플레인) API 작업은 리전 엔드포인트를 통해 사용할 수 있으며 이를 리전 엔드포인트 API 작업이라고 합니다. 리전 엔드포인트 API 작업의 예로는 `CreateBucket` 및 `DeleteBucket`이 있습니다.

S3 Express One Zone에 저장되는 디렉터리 버킷을 생성할 때 버킷이 위치할 가용 영역을 선택합니다. 영역 엔드포인트 API 작업을 사용하여 디렉터리 버킷에 객체를 업로드하고 관리할 수 있습니다.

객체 수준(즉 데이터 영역) API 작업은 영역 엔드포인트를 통해 사용할 수 있으며 이를 영역 엔드포인트 API 작업이라고 합니다. 영역 엔드포인트 API 작업의 예로는 `CreateSession` 및 `PutObject`가 있습니다.

S3 Express One Zone 성능 최적화

S3 Express One Zone은 지연 시간에 가장 민감한 애플리케이션에 대해 일관되게 10밀리초 미만의 데이터 액세스를 제공하도록 특별히 설계된 고성능 단일 가용 영역(AZ) S3 스토리지 클래스입니다. S3 Express One Zone은 Amazon Elastic Compute Cloud, Amazon Elastic Kubernetes Service, Amazon Elastic Container Service와 같은 고성능 객체 스토리지와 AWS 컴퓨팅 리소스를 단일 가용 영역 내에 함께 배치할 수 있는 옵션을 제공하는 최초의 S3 스토리지 클래스입니다. 스토리지와 컴퓨팅 리소스를 함께 배치하면 컴퓨팅 성능과 비용이 최적화되고 데이터 처리 속도가 향상됩니다.

S3 Express One Zone은 다른 S3 스토리지 클래스와 유사한 성능 탄력성을 제공하면서도 10밀리초 미만의 첫 번째 바이트 읽기 및 쓰기 요청 지연 시간을 일관되게 제공하여 S3 Standard보다 최대 10배 더 빠릅니다. S3 Express One Zone은 처음부터 매우 높은 집계 수준까지 버스트 처리량을 지원하도록 설계되었습니다. S3 Express One Zone 스토리지 클래스는 맞춤형 아키텍처를 사용하여 고성능 하드웨어에 데이터를 저장함으로써 성능을 최적화하고 지속적으로 짧은 요청 지연 시간을 제공합니다. S3 Express One Zone의 객체 프로토콜은 인증 및 메타데이터 오버헤드를 간소화하도록 향상되었습니다.

지연 시간을 더욱 줄이고 초당 최대 2백만 건의 읽기와 최대 200,000건의 쓰기를 지원하기 위해 S3 Express One Zone은 Amazon S3 디렉터리 버킷에 데이터를 저장합니다. 기본적으로 각 디렉터리 버킷은 초당 최대 200,000건의 읽기와 최대 100,000건의 쓰기를 지원합니다. 워크로드에 기본 TPS 제한보다 높은 값이 필요한 경우 AWS Support를 통해 증가를 요청할 수 있습니다.

10밀리초 미만의 데이터 액세스 속도를 제공하는 고성능 하드웨어 및 소프트웨어와 초당 대량의 트랜잭션까지 확장할 수 있는 디렉터리 버킷 조합된 덕분에 S3 Express One Zone은 요청 집약적인 작업이나 성능이 매우 중요한 애플리케이션에 가장 적합한 Amazon S3 스토리지 클래스입니다.

다음 주제에서는 S3 Express One Zone 스토리지 클래스를 사용하는 애플리케이션의 성능을 최적화하기 위한 모범 사례 지침과 설계 패턴에 대해 설명합니다.

주제

- S3 Express One Zone 성능을 최적화하는 모범 사례

S3 Express One Zone 성능을 최적화하는 모범 사례

Amazon S3 Express One Zone에서 객체를 업로드 및 검색하는 애플리케이션을 빌드할 때 모범 사례 지침에 따라 성능을 최적화합니다. S3 Express One Zone 스토리지 클래스를 사용하려면 S3 디렉터리 버킷을 생성해야 합니다. S3 Express One Zone 스토리지 클래스는 S3 범용 버킷과 함께 사용할 수 없습니다.

기타 모든 Amazon S3 스토리지 클래스 및 S3 범용 버킷에 대한 성능 지침은 모범 사례 설계 패턴: Amazon S3 성능 최적화 섹션을 참조하세요.

대규모 워크로드에서 S3 Express One Zone 스토리지 클래스 및 디렉터리 버킷의 성능과 확장성을 최적화하려면 디렉터리 버킷이 범용 버킷과 어떻게 다른지 이해하는 것이 중요합니다. 그런 다음 디렉터리 버킷의 작동 방식에 맞게 애플리케이션을 조정하는 모범 사례가 제시되어 있습니다.

디렉터리 버킷의 작동 방식

Amazon S3 Express One Zone 스토리지 클래스는 디렉터리 버킷당 초당 최대 2,000,000개의 GET 및 최대 200,000개의 PUT 트랜잭션(TPS)으로 워크로드를 지원할 수 있습니다. S3 Express One Zone을 사용하면 데이터가 가용 영역의 S3 디렉터리 버킷에 저장됩니다. 디렉터리 버킷의 객체는 파일 시스템과 유사한 계층적 네임스페이스 내에서 액세스할 수 있으며, 이는 플랫 네임스페이스를 사용하는 S3 범용 버킷과는 대조적입니다. 범용 버킷과 달리 디렉터리 버킷은 키를 접두사 대신 디렉터리에 계층적으로 구성합니다. 접두사는 객체 키 이름의 시작 부분에 있는 문자열입니다. 범용 버킷에서는 접두사를 사용하여 데이터를 구성하고 플랫 객체 스토리지 아키텍처를 관리할 수 있습니다. 자세한 내용은 접두어를 사용한 객체 구성 섹션을 참조하세요.

디렉터리 버킷에서 객체는 유일하게 지원되는 구분 기호인 슬래시(/)를 사용하여 계층적 네임스페이스에 구성됩니다. `dir1/dir2/file1.txt`와 같은 키를 사용하여 객체를 업로드하면 Amazon S3에서 `dir1/` 및 `dir2/` 디렉터리가 자동으로 생성되고 관리됩니다. 디렉터리는 `PutObject` 또는 `CreateMultiPartUpload` 작업 중에 생성되며 `DeleteObject` 또는 `AbortMultiPartUpload` 작업 후 비어 있게 되면 자동으로 제거됩니다. 디렉터리의 객체 및 하위 디렉터리 수에는 상한이 없습니다.

객체가 디렉터리 버킷에 업로드될 때 생성되는 디렉터리는 HTTP `503 (Slow Down)` 오류 발생 가능성을 줄이도록 즉시 규모가 조정될 수 있습니다. 이 자동 크기 조정을 통해 애플리케이션은 필요에 따라 디렉터리 내부 및 디렉터리 간 읽기 및 쓰기 요청을 병렬화할 수 있습니다. S3 Express One Zone의 경우 개별 디렉터리는 디렉터리 버킷의 최대 요청 속도를 지원하도록 설계되었습니다. 시스템이 균등한 로드 분배를 위해 객체를 자동으로 분산하므로 최적의 성능을 달성하기 위해 키 접두사를 무작위화

할 필요는 없습니다. 하지만 그 결과로 키가 디렉터리 버킷에 사전순으로 저장되지 않습니다. 이는 사전순으로 더 가까운 키가 동일한 서버에 함께 위치할 가능성이 더 높은 S3 범용 버킷과는 대조적입니다.

디렉터리 버킷 작업 및 디렉터리 상호 작용의 예에 대한 자세한 내용은 디렉터리 버킷 작업 및 디렉터리 상호 작용 예시 섹션을 참조하세요.

모범 사례

디렉터리 버킷 성능을 최적화하고 시간이 지남에 따라 워크로드 규모를 조정할 수 있도록 하려면 모범 사례를 따르세요.

많은 항목(객체 또는 하위 디렉터리)을 포함하는 디렉터리 사용

디렉터리 버킷은 기본적으로 모든 워크로드에 대해 높은 성능을 제공합니다. 특정 작업의 성능을 더욱 최적화하기 위해 더 많은 항목(객체 또는 하위 디렉터리)을 디렉터리에 통합하면 지연 시간이 단축되고 요청 속도가 빨라집니다.

- `PutObject`, `DeleteObject`, `CreateMultiPartUpload`, `AbortMultiPartUpload`와 같은 변형 API 작업은 수천 개의 항목을 포함하는 적은 수의 고밀도 디렉터리를 사용하여 구현할 때 최적의 성능을 발휘하며, 많은 수의 작은 디렉터리를 사용하는 것보다 더 효과적입니다.
- `ListObjectsV2` 작업은 결과 페이지를 채우기 위해 탐색해야 하는 디렉터리 수가 적을수록 더 나은 성능을 발휘합니다.

접두사에 엔트로피를 사용하지 않음

Amazon S3 작업에서 엔트로피는 접두사 이름 지정의 무작위성을 의미하며, 이는 스토리지 파티션 간에 워크로드를 균등하게 분산하는 데 도움이 됩니다. 그러나 디렉터리 버킷은 로드 분산을 내부적으로 관리하므로 최상의 성능을 위해 접두사에 엔트로피를 사용하지 않는 것이 좋습니다. 디렉터리 버킷의 경우 엔트로피로 인해 이미 생성된 디렉터리가 재사용되지 않아 요청이 느려질 수 있기 때문입니다.

`$HASH/directory/object`와 같은 키 패턴은 많은 중간 디렉터리를 생성할 수 있습니다. 다음 예시에서 모든 `job-1`은 상위 디렉터리가 다르기 때문에 서로 다른 디렉터리입니다. 디렉터리의 밀도가 낮아지고 변형 및 목록 요청 속도가 느려집니다. 이 예시에는 모두 단일 항목을 가진 중간 디렉터리가 12개 있습니다.

```
s3://my-bucket/0cc175b9c0f1b6a831c399e269772661/job-1/file1

s3://my-bucket/92eb5ffee6ae2fec3ad71c777531578f/job-1/file2
```

```
s3://my-bucket/4a8a08f09d37b73795649038408b5f33/job-1/file3

s3://my-bucket/8277e0910d750195b448797616e091ad/job-1/file4

s3://my-bucket/e1671797c52e15f763380b45e841ec32/job-1/file5

s3://my-bucket/8fa14cdd754f91cc6554c9e71929cce7/job-1/file6
```

대신 더 나은 성능을 위해 $HASH 구성 요소를 제거하고 job-1을 단일 디렉터리로 만들어 디렉터리 밀도를 높일 수 있습니다. 다음 예시에서 6개의 항목을 가진 단일 중간 디렉터리가 이전 예시에 비해 더 나은 성능을 낼 수 있습니다.

```
s3://my-bucket/job-1/file1

s3://my-bucket/job-1/file2

s3://my-bucket/job-1/file3

s3://my-bucket/job-1/file4

s3://my-bucket/job-1/file5

s3://my-bucket/job-1/file6
```

이러한 성능 이점은 객체 키가 처음 생성되고 키 이름에 디렉터리가 포함되면 해당 객체에 대한 디렉터리가 자동으로 생성되기 때문에 발생합니다. 이후에 동일한 디렉터리에 객체를 업로드할 때는 디렉터리를 생성할 필요가 없으므로 기존 디렉터리로 객체를 업로드하는 데 걸리는 지연 시간이 줄어듭니다.

ListObjectsV2 직접 호출 시 객체를 논리적으로 그룹화하는 기능이 필요 없다면 구분 기호 / 이외의 다른 구분 기호를 사용하여 키의 각 부분을 구분

디렉터리 버킷의 경우 / 구분 기호가 특수하게 처리되므로 분명한 목적을 갖고 사용해야 합니다. 디렉터리 버킷은 객체를 사전순으로 정렬하지 않지만 디렉터리 내의 객체는 여전히 ListObjectsV2 출력에서 함께 그룹화됩니다. 이 기능이 필요하지 않다면 구분 기호 /를 다른 문자로 바꾸어 중간 디렉터리가 생성되지 않도록 할 수 있습니다.

예를 들어 다음 키가 YYYY/MM/DD/HH/ 접두사 패턴을 갖는다고 가정해 보겠습니다.

```
s3://my-bucket/2024/04/00/01/file1

s3://my-bucket/2024/04/00/02/file2
```

```
s3://my-bucket/2024/04/00/03/file3

s3://my-bucket/2024/04/01/01/file4

s3://my-bucket/2024/04/01/02/file5

s3://my-bucket/2024/04/01/03/file6
```

ListObjectsV2 결과에서 시간 또는 날짜별로 객체를 그룹화할 필요가 없지만 월별로 객체를 그룹화해야 하는 경우 다음과 같은 YYYY/MM/DD-HH- 키 패턴을 사용하면 ListObjectsV2 작업의 디렉터리가 크게 줄어들고 성능이 향상됩니다.

```
s3://my-bucket/2024/04/00-01-file1

s3://my-bucket/2024/04/00-01-file2

s3://my-bucket/2024/04/00-01-file3

s3://my-bucket/2024/04/01-02-file4

s3://my-bucket/2024/04/01-02-file5

s3://my-bucket/2024/04/01-02-file6
```

가능하면 구분 기호로 구분된 목록 작업 사용

delimiter가 없는 ListObjectsV2 요청은 모든 디렉터리에 대해 깊이 우선 재귀 탐색을 수행합니다. delimiter가 있는 ListObjectsV2 요청은 prefix 파라미터로 지정된 디렉터리의 항목만 검색하므로 요청 지연 시간이 줄고 초당 집계 키 수가 늘어납니다. 디렉터리 버킷의 경우 가능하면 구분 기호로 구분된 목록 작업을 사용하도록 합니다. 구분 기호로 목록을 구분하면 디렉터리 방문 횟수가 줄어들어 초당 키 수가 늘어나고 요청 지연 시간이 단축됩니다.

예를 들어 디렉터리 버킷에 다음 디렉터리 및 객체가 있다고 해보겠습니다.

```
s3://my-bucket/2024/04/12-01-file1

s3://my-bucket/2024/04/12-01-file2

...

s3://my-bucket/2024/05/12-01-file1
```

```
s3://my-bucket/2024/05/12-01-file2

...

s3://my-bucket/2024/06/12-01-file1

s3://my-bucket/2024/06/12-01-file2

...

s3://my-bucket/2024/07/12-01-file1

s3://my-bucket/2024/07/12-01-file2

...
```

ListObjectsV2 성능 향상을 위해 애플리케이션의 로직에서 허용하는 경우 구분 기호로 구분된 목록을 사용하여 하위 디렉터리와 객체를 나열합니다. 예를 들어 구분 기호로 구분된 목록 작업을 위해 다음 명령을 실행할 수 있습니다.

```
aws s3api list-objects-v2 --bucket my-bucket --prefix '2024/' --delimiter '/'
```

출력은 하위 디렉터리 목록입니다.

```
{
    "CommonPrefixes": [
        {
            "Prefix": "2024/04/"
        },
        {
            "Prefix": "2024/05/"
        },
        {
            "Prefix": "2024/06/"
        },
        {
            "Prefix": "2024/07/"
        }
    ]
}
```

더 나은 성능으로 각 하위 디렉터리를 나열하려면 다음 예시와 같은 명령을 실행할 수 있습니다.

명령:

```
aws s3api list-objects-v2 --bucket my-bucket --prefix '2024/04' --delimiter '/'
```

출력:

```
{
    "Contents": [
        {
            "Key": "2024/04/12-01-file1"
        },
        {
            "Key": "2024/04/12-01-file2"
        }
    ]
}
```

S3 Express One Zone 스토리지를 컴퓨팅 리소스와 같은 위치에 배치

S3 Express One Zone을 사용하면 각 디렉터리 버킷이 버킷을 생성할 때 선택한 단일 가용 영역에 위치하게 됩니다. 컴퓨팅 워크로드 또는 리소스와 같은 위치의 로컬 가용 영역에 새 디렉터리 버킷을 생성하여 시작할 수 있습니다. 그러면 지연 시간이 매우 짧은 읽기 및 쓰기를 즉시 시작할 수 있습니다. 디렉터리 버킷은 컴퓨팅과 스토리지 간의 지연 시간을 줄이기 위해 AWS 리전 내에서 가용 영역을 선택할 수 있는 S3 버킷 유형입니다.

서로 다른 가용 영역에서 디렉터리 버킷에 액세스하는 경우 지연 시간이 약간 늘어날 수 있습니다. 성능을 최적화하려면 가능하면 동일한 가용 영역에 있는 Amazon Elastic Container Service, Amazon Elastic Kubernetes Service 및 Amazon Elastic Compute Cloud 인스턴스에서 디렉터리 버킷에 액세스하는 것이 좋습니다.

1MB가 넘는 객체의 경우 높은 처리량을 달성하기 위해 동시 연결 사용

디렉터리 버킷으로 여러 건의 동시 요청을 보내 요청을 별도의 연결로 분산하여 액세스 가능한 대역폭을 극대화함으로써 최상의 성능을 달성할 수 있습니다. 범용 버킷과 마찬가지로 S3 Express One Zone은 디렉터리 버킷에 대한 연결 수 제한이 없습니다. 개별 디렉터리는 동일한 디렉터리에 많은 수의 동시 쓰기가 발생하는 경우 성능을 수평적으로 자동 확장할 수 있습니다.

디렉터리 버킷에 대한 개별 TCP 연결에는 초당 업로드하거나 다운로드할 수 있는 바이트 수의 상한이 고정되어 있습니다. 객체가 커지면 요청 시간이 트랜잭션 처리가 아닌 바이트 스트리밍에 좌우됩니다. 여러 연결을 사용하여 더 큰 객체의 업로드 또는 다운로드를 병렬화하려면 엔드투엔드 지연 시간을 줄일 수 있습니다. `Java 2.x` SDK를 사용하는 경우 멀티파트 업로드 API 작업 및 바이트 범위 가져오기

와 같은 성능 향상 기능을 활용하여 데이터에 병렬로 액세스하는 S3 Transfer Manager를 사용하는 것이 좋습니다.

게이트웨이 VPC 엔드포인트 사용

게이트웨이 엔드포인트를 사용하면 VPC에 대한 인터넷 게이트웨이 또는 NAT 디바이스 없이 VPC에서 디렉터리 버킷으로 직접 연결할 수 있습니다. 패킷이 네트워크에서 소비하는 시간을 줄이기 위해 디렉터리 버킷에 게이트웨이 VPC 엔드포인트를 사용하여 VPC를 구성해야 합니다. 자세한 내용은 디렉터리 버킷에 대한 네트워킹 섹션을 참조하세요.

세션 인증 사용 및 유효한 기간 동안 세션 토큰 재사용

디렉터리 버킷은 성능에 민감한 API 작업의 지연 시간을 줄이기 위해 세션 토큰 인증 메커니즘을 제공합니다. `CreateSession`을 한 번 직접적으로 호출하여 세션 토큰을 얻을 수 있으며, 이 토큰은 이후 5분 동안 모든 요청에 유효합니다. API 직접 호출에서 지연 시간을 최소화하려면 세션 토큰을 획득하여 해당 토큰의 전체 수명 동안 재사용한 후 새로 고쳐야 합니다.

AWS SDK를 사용하면 SDK가 세션 토큰 새로 고침을 자동으로 처리하여 세션이 만료될 때 서비스가 중단되지 않도록 합니다. AWS SDK를 사용하여 `CreateSession` API 작업에 대한 요청을 시작하고 관리하는 것이 좋습니다.

`CreateSession`에 대한 자세한 정보는 `CreateSession`을 사용하여 영역 엔드포인트 API 작업 권한 부여 섹션을 참조하세요.

CRT 기반 클라이언트 사용

AWS 공통 런타임(CRT)은 C 언어로 작성된 모듈식의 고성능 고효율 라이브러리 세트로, AWS SDK의 기반 역할을 합니다. CRT를 사용하면 처리량이 증가하고, 연결 관리가 향상되며, 시작 시간이 단축됩니다. CRT는 Go를 제외한 모든 AWS SDK에서 사용할 수 있습니다.

사용하는 SDK에 CRT를 구성하는 방법에 대한 자세한 내용은 AWS 공통 런타임(CRT) 라이브러리, Accelerate Amazon S3 throughput with the AWS Common Runtime, Introducing CRT-based S3 client and the S3 Transfer Manager in the AWS SDK for Java 2.x, Amazon S3 작업에 S3CrtClient 사용, AWS CRT 기반 HTTP 클라이언트 구성을 참조하세요.

최신 AWS SDK 버전 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 SDK는 HTTP `503` 오류 후 요청을 자동으로 재시도하고 느린 연결 응답을 처리합니다.

Java 2.x SDK를 사용하는 경우 S3 Transfer Manager를 사용하는 것이 좋습니다. S3 Transfer Manager는 필요에 따라 바이트 범위 요청을 사용하여 초당 수천 개의 요청을 처리할 수 있도록 연결 규모를 수평적으로 자동 조정합니다. 바이트 범위 요청은 S3에 대한 동시 연결을 사용하여 동일한 객체 내에서 다양한 바이트 범위를 가져올 수 있으므로 성능을 향상시킬 수 있습니다. 그러면 전체 객체 요청 한 건에 비해 집계 처리량을 높일 수 있습니다. 최신 성능 최적화 기능을 활용하려면 최신 버전의 AWS SDK를 사용하는 것이 중요합니다.

성능 문제 해결

지연 시간에 민감한 애플리케이션에 재시도 요청을 설정하고 있나요?

S3 Express One Zone은 추가 조정 없이 일관된 수준의 고성능을 제공하도록 특별히 설계되었습니다. 하지만 제한 시간 값과 재시도를 적극적으로 설정하면 지연 시간과 성능을 일관되게 유지하는 데 도움이 될 수 있습니다. AWS SDK에는 특정 애플리케이션의 허용 오차에 따라 튜닝할 수 있는 구성 가능한 제한 시간 및 재시도 값이 있습니다.

AWS 공통 런타임(CRT) 라이브러리와 최적의 Amazon EC2 인스턴스 유형을 사용하고 있나요?

다수의 읽기 및 쓰기 작업을 수행하는 애플리케이션은 그렇지 않은 애플리케이션보다 메모리 또는 컴퓨팅 용량이 훨씬 더 많이 필요합니다. 성능이 요구되는 워크로드를 위해 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 시작할 때는 애플리케이션에 필요한 만큼의 리소스를 포함하는 인스턴스 유형을 선택하세요. S3 Express One Zone 고성능 스토리지는 고성능 스토리지의 이점을 활용할 수 있는 더 큰 용량의 시스템 메모리와 더 강력한 CPU 및 GPU를 갖춘 더 큰 신규 인스턴스 유형과 함께 사용하는 것이 이상적입니다. 또한 읽기 및 쓰기 요청을 병렬로 더 빠르게 처리할 수 있는 최신 버전의 CRT 지원 AWS SDK를 사용하는 것이 좋습니다.

세션 기반 인증에 AWS SDK를 사용하고 있나요?

Amazon S3를 사용하면 AWS SDK의 일부 모범 사례와 동일한 모범 사례를 따라 HTTP REST API 요청을 사용할 때 성능을 최적화할 수 있습니다. 하지만 S3 Express One Zone에서 사용하는 세션 기반 권한 부여 및 인증 메커니즘을 사용하면 AWS SDK를 사용하여 CreateSession 및 관리형 세션 토큰을 관리하는 것이 좋습니다. AWS SDK는 CreateSession API 작업을 사용하여 사용자를 대신하여 자동으로 토큰을 생성하고 새로 고칩니다. CreateSession을 사용하면 각 요청을 승인하는 데 필요한 AWS Identity and Access Management(IAM)로의 요청당 왕복 지연 시간을 줄일 수 있습니다.

디렉터리 버킷 작업 및 디렉터리 상호 작용 예시

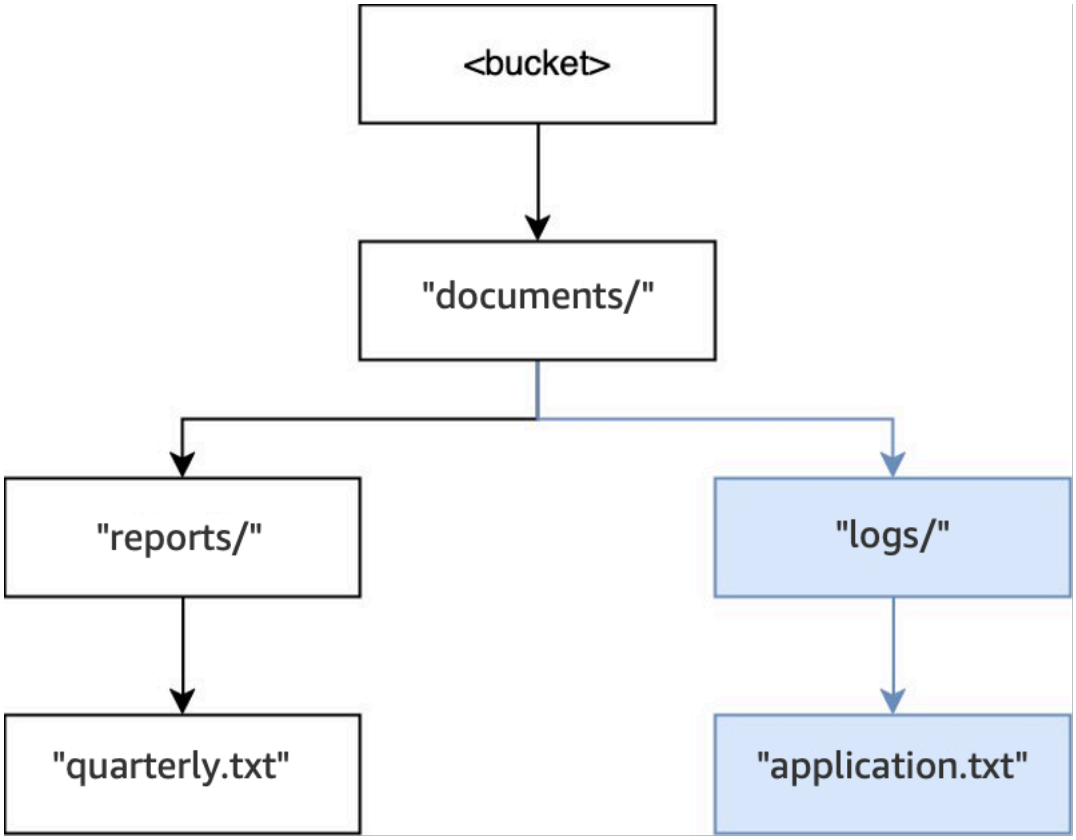
다음은 디렉터리 버킷의 작동 방식에 대한 세 가지 예시입니다.

예시 1: 디렉터리 버킷에 대한 S3 **PutObject** 요청이 디렉터리와 상호 작용하는 방식

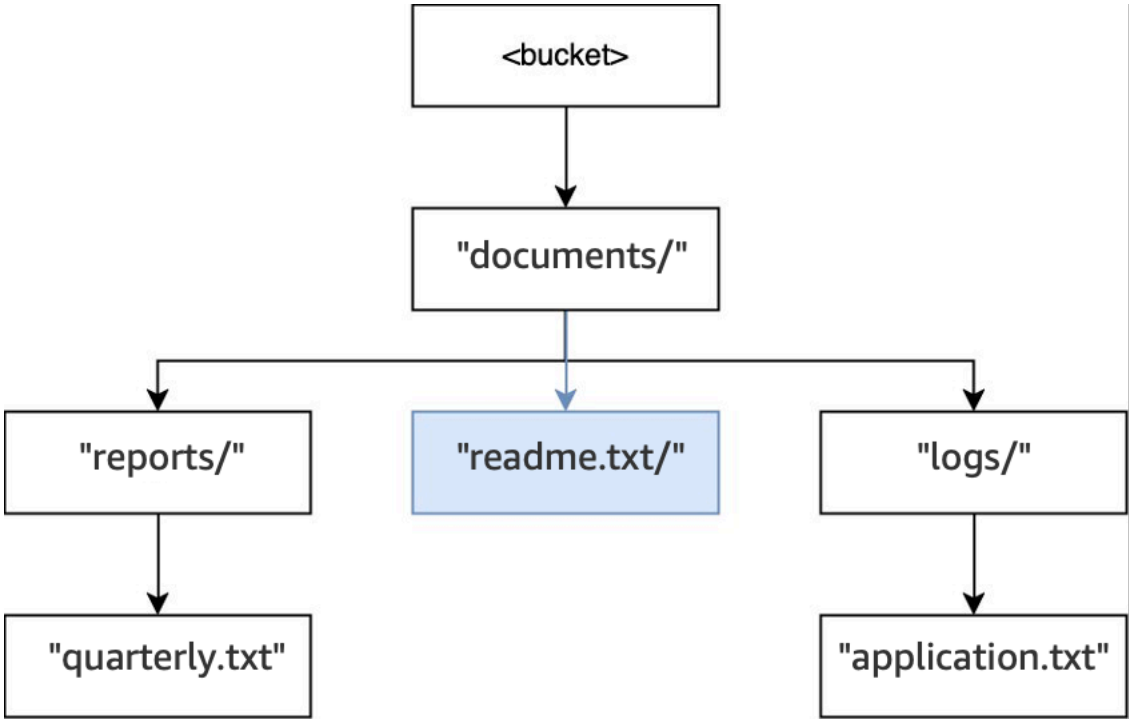
1. 빈 버킷에서 PUT(<bucket>, "documents/reports/quarterly.txt") 작업을 실행하면 버킷 루트 내에 documents/ 디렉터리가 생성되고, documents/ 내에 reports/ 디렉터리가 생성되며, reports/ 내에 quarterly.txt 객체가 생성됩니다. 이 작업의 경우 객체 외에 두 개의 디렉터리가 생성되었습니다.



2. 그런 다음 또 다른 작업 PUT(<bucket>, "documents/logs/application.txt")가 실행되면 documents/ 디렉터리는 이미 존재하고, documents/ 내의 logs/ 디렉터리는 존재하지 않고 새로 생성되며, logs/ 디렉터리 내에 application.txt 객체가 생성됩니다. 이 작업의 경우 객체 외에 하나의 디렉터리만 생성되었습니다.



3. 마지막으로, PUT(<bucket>, "documents/readme.txt") 작업이 실행되면 루트 내의 documents/ 디렉터리가 이미 존재하고 readme.txt 객체가 생성됩니다. 이 작업의 경우 디렉터리가 생성되지 않습니다.



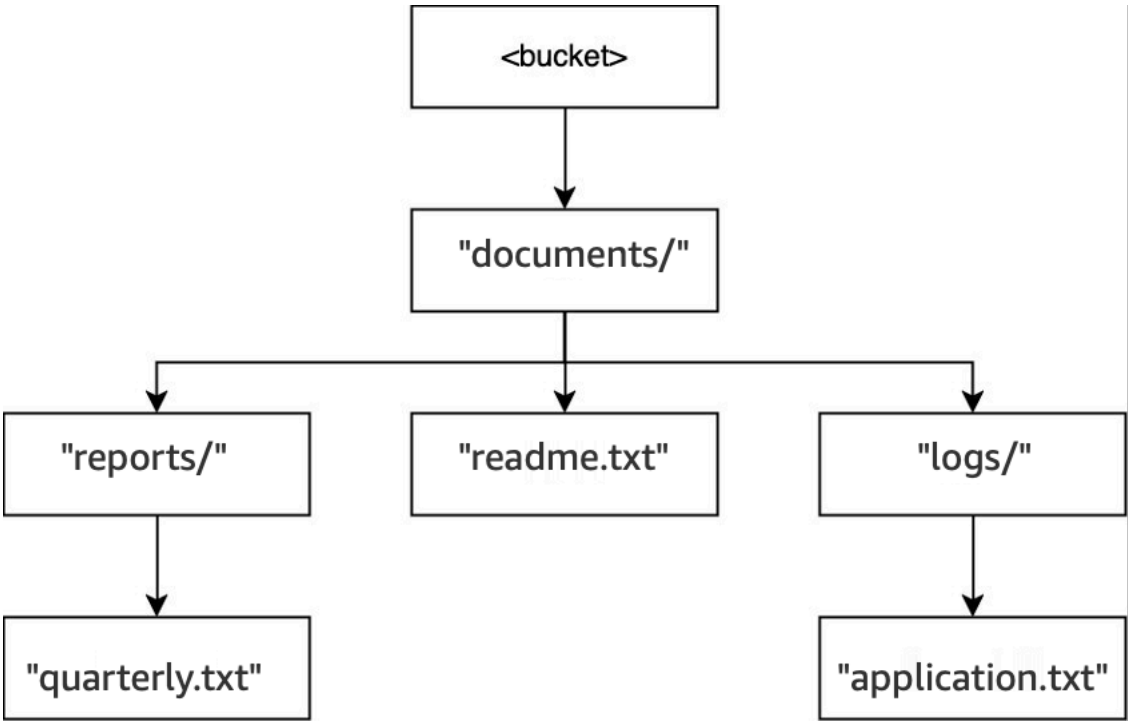
예시 2: 디렉터리 버킷에 대한 S3 **ListObjectsV2** 요청이 디렉터리와 상호 작용하는 방식

구분 기호를 지정하지 않은 S3 ListObjectsV2 요청의 경우 버킷은 깊이 우선 방식으로 탐색됩니다. 출력은 일관된 순서로 반환됩니다. 그러나 이 순서는 요청 간에 동일하게 유지되지만 사전식 순서는 아닙니다. 이전 예시에서 생성된 버킷 및 디렉터리의 경우 다음과 같습니다.

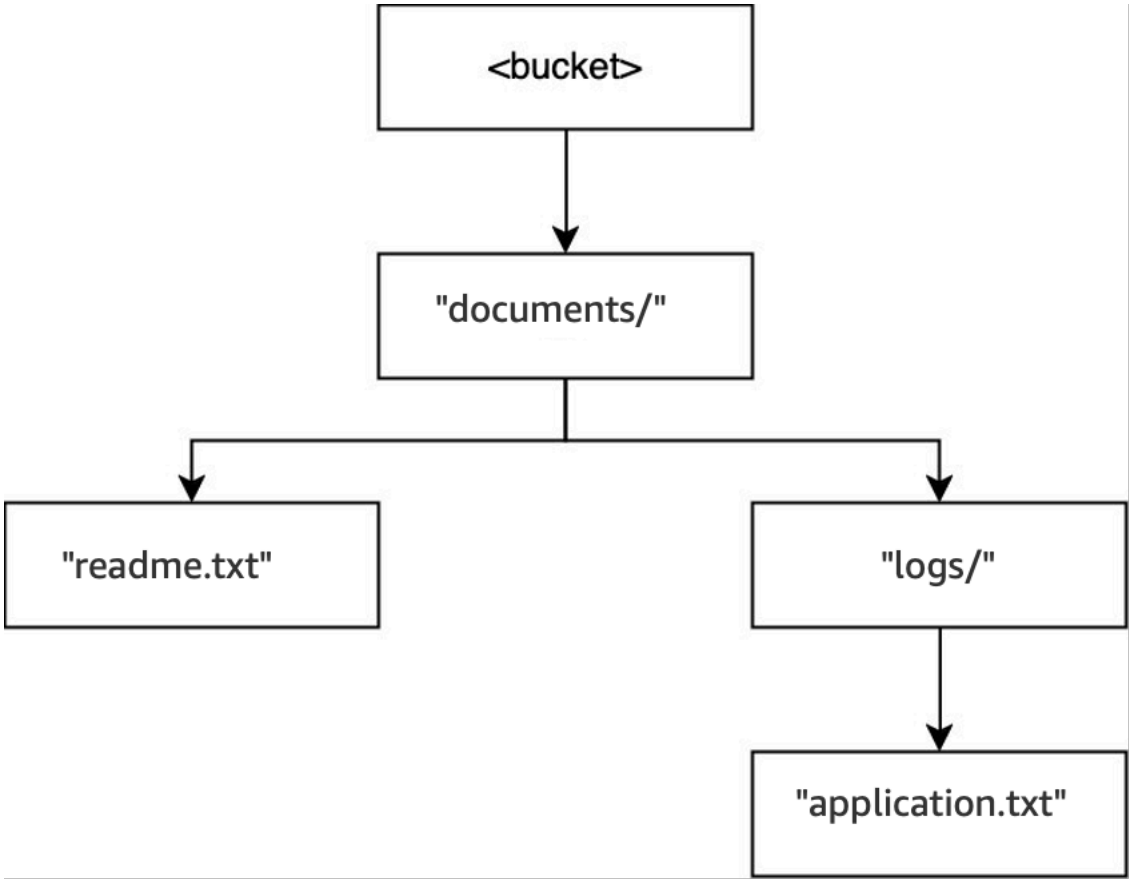
1. LIST(<bucket>)가 실행되면 documents/ 디렉터리로 이동하고 탐색이 시작됩니다.
2. 하위 디렉터리 logs/로 이동하고 탐색이 시작됩니다.
3. logs/에서 application.txt 객체가 발견됩니다.
4. logs/에 더 이상 항목이 없습니다. List 작업이 logs/에서 나와 documents/로 다시 이동합니다.
5. documents/ 디렉터리 탐색이 계속되고 readme.txt 객체가 발견됩니다.
6. documents/ 디렉터리는 계속 탐색되고 하위 디렉터리 reports/로 이동하며 탐색이 시작됩니다.
7. reports/에서 quarterly.txt 객체가 발견됩니다.
8. reports/에 더 이상 항목이 없습니다. List가 reports/에서 나와 documents/로 다시 이동합니다.
9. documents/에 더 이상 항목이 없으므로 List가 반환됩니다.

이 예시에서, logs/가 readme.txt보다 앞 순서이고, readme.txt가 reports/보다 앞 순서입니다.

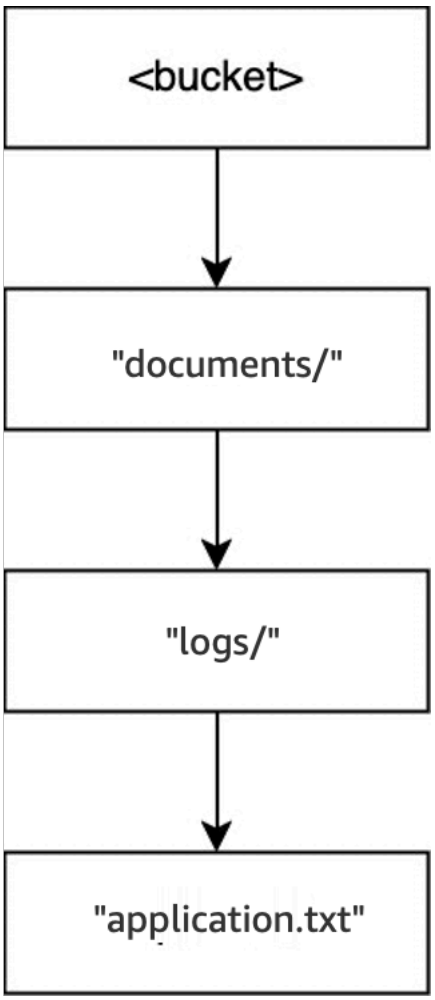
예시 3: 디렉터리 버킷에 대한 S3 **DeleteObject** 요청이 디렉터리와 상호 작용하는 방식



1. 동일한 버킷에서 `DELETE(<bucket>, "documents/reports/quarterly.txt")` 작업이 실행되면 `quarterly.txt` 객체가 삭제되어 `reports/` 디렉터리가 비어 있게 되고 즉시 삭제됩니다. `documents/` 디렉터리는 `logs/` 디렉터리와 `readme.txt` 객체를 모두 포함하며 비어 있지 않으므로 삭제되지 않습니다. 이 작업의 경우 객체 하나와 디렉터리 하나만 삭제되었습니다.



2. DELETE(<bucket>, "documents/readme.txt") 작업이 실행되면 readme.txt 객체가 삭제됩니다. documents/는 logs/ 디렉터리를 포함하며 비어 있지 않으므로 삭제되지 않습니다. 이 작업의 경우 디렉터리는 삭제되지 않으며 객체만 삭제됩니다.



3. 마지막으로, `DELETE(<bucket>, "documents/logs/application.txt")` 작업이 실행되면 `application.txt`가 삭제되어 `logs/`가 비어 있게 되고 즉시 삭제됩니다. 그러면 `documents/`도 비어 있게 되어 즉시 삭제됩니다. 이 작업의 경우 디렉터리 두 개와 객체 한 개가 삭제됩니다. 이제 버킷이 비어 있습니다.



데이터 레지던시 워크로드

AWS 전용 로컬 영역(전용 로컬 영역)은 AWS 인프라의 한 유형으로, AWS에서 완전히 관리하고, 사용자 또는 사용자의 커뮤니티가 독점적으로 사용하도록 빌드되며, 규제 요구 사항을 준수하기 위해 사용자가 지정한 위치 또는 데이터 센터에 배치됩니다. 전용 로컬 영역은 AWS 로컬 영역(로컬 영역) 제품 및 서비스의 한 유형입니다. 자세한 내용은 AWS 전용 로컬 영역을 참조하세요.

전용 로컬 영역에서는 S3 디렉터리 버킷을 생성하여 특정 데이터 경계 내에 데이터를 저장할 수 있으므로 데이터 레지던시 및 격리 사용 사례를 지원하는 데 도움이 됩니다. 전용 로컬 영역의 디렉터리 버킷은 S3 Express One Zone 및 S3 One Zone-Infrequent Access(S3 One Zone-IA, Z-IA) 스토리지 클래스를 지원할 수 있습니다. 디렉터리 버킷은 현재 다른 AWS 로컬 영역 위치에서 사용할 수 없습니다.

전용 로컬 영역에서 AWS Management Console, REST API, AWS Command Line Interface(AWS CLI) 및 AWS SDK를 사용할 수 있습니다.

로컬 영역에서의 디렉터리 버킷 작업에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- 로컬 영역의 디렉터리 버킷에 대한 개념
- 로컬 영역에 대해 계정 활성화
- VPC에서의 프라이빗 연결
- 로컬 영역에서 디렉터리 버킷 생성
- 로컬 영역의 디렉터리 버킷에 대한 인증 및 권한 부여

로컬 영역의 디렉터리 버킷에 대한 개념

로컬 영역에서 디렉터리 버킷을 생성하기 전에 버킷을 생성할 로컬 영역의 ID가 있어야 합니다. DescribeAvailabilityZones API 작업을 사용하여 모든 로컬 영역 정보를 찾을 수 있습니다. 이 API 작업은 로컬 영역 ID, 상위 리전 이름, 네트워크 경계 그룹, 옵트인 상태를 포함하여 로컬 영역에 대한 정보를 나열합니다. 로컬 영역 ID가 있고 옵트인했으면 로컬 영역에서 디렉터리 버킷을 생성할 수 있습니다. 디렉터리 버킷 이름은 사용자가 제공하는 기본 이름과 버킷 위치의 영역 ID 및 `--x-s3`가 차례로 포함된 접미사로 구성됩니다.

로컬 영역은 Amazon의 이중화되고 대역폭이 매우 높은 프라이빗 네트워크를 사용하여 상위 리전에 연결됩니다. 이로 인해 로컬 영역에서 실행되는 애플리케이션이 상위 리전의 나머지 AWS 서비스에 빠르고 안전하며 원활하게 액세스할 수 있습니다. 상위 영역 ID는 로컬 영역 컨트롤 플레인 작업을 처리하는 영역의 ID입니다. 네트워크 경계 그룹은 AWS가 퍼블릭 IP 주소를 광고하는 고유한 그룹입니다. 로컬 영역, 상위 리전 및 상위 영역 ID에 대한 자세한 내용은 AWS 로컬 영역 사용 설명서의 AWS Local Zones concepts를 참조하세요.

모든 디렉터리 버킷은 범용 버킷의 `s3` 네임스페이스와 별도의 `s3express` 네임스페이스를 사용합니다. 디렉터리 버킷의 경우 요청은 리전 엔드포인트 또는 영역 엔드포인트로 라우팅됩니다. AWS Management Console, AWS CLI 또는 AWS SDK를 사용하는 경우 라우팅이 자동으로 처리됩니다.

대부분의 버킷 수준 API 작업(예: CreateBucket 및 DeleteBucket)은 리전 엔드포인트에 라우팅되며 리전 엔드포인트 API 작업이라고 합니다. 리전 엔드포인트는 s3express-control.ParentRegionCode.amazonaws.com 형식입니다. 모든 객체 수준 API 작업(예: PutObject)과 두 개의 버킷 수준 API 작업(CreateSession 및 HeadBucket)은 영역 엔드포인트로 라우팅되며 영역 엔드포인트 API 작업이라고 합니다. 영역 엔드포인트는 s3express-LocalZoneID.ParentRegionCode.amazonaws.com형식입니다. 엔드포인트 유형별 API 작업의 전체 목록은 Directory bucket API operations을 참조하세요.

베이징 로컬 영역에서 S3 사용 가능:

로컬 영역 ID: cnn1-pkx1-az1

- 상위 리전 이름: China (Beijing)
- 상위 리전 코드: cn-north-1
- 상위 영역 ID: cnn1-az1
- 로컬 영역 이름: China (Beijing)
- 로컬 영역 코드: cnn1-pkx1-az1
- 네트워크 경계 그룹: cn-north-1-pkx-1
- 예시 버킷 ARN: arn:aws-cn:s3express:cn-north-1:*123456789012*:bucket/*amzn-s3-demo-bucket*--cnn1-pkx1-az1--x-s3
- 리전 엔드포인트: s3express-control.cn-north-1.amazonaws.com.cn
- 영역 엔드포인트: s3express-cnn1-pkx1-az1.cn-north-1.amazonaws.com.cn
- 스토리지 클래스: S3 One Zone-Infrequent Access (S3 One Zone-IA; Z-IA)

가상 프라이빗 클라우드(VPC)에서 로컬 영역의 디렉터리 버킷에 액세스하려면 게이트웨이 VPC 엔드포인트를 사용하면 됩니다. 게이트웨이 엔드포인트 사용에 따르는 추가 요금은 없습니다. 로컬 영역의 디렉터리 버킷 및 객체에 액세스하도록 게이트웨이 VPC 엔드포인트를 구성하려면 VPC에서의 프라이빗 연결 섹션을 참조하세요.

로컬 영역에 대해 계정 활성화

다음 주제에서는 전용 로컬 영역에 대해 계정을 활성화하는 방법을 설명합니다.

전용 로컬 영역에서 리소스를 생성하거나 액세스하려면 먼저 AWS 전용 로컬 영역(전용 로컬 영역)의 Amazon S3를 비롯한 모든 서비스에 대해 관리자가 AWS 계정을 활성화해야 합니다. DescribeAvailabilityZones API 작업을 사용하여 로컬 영역에 대한 계정 ID 액세스를 확인할 수 있습니다.

기본적으로, Amazon S3에서 데이터를 추가로 보호하기 위해 자신이 생성한 S3 리소스에만 액세스할 수 있습니다. 로컬 영역의 버킷에는 기본적으로 모든 S3 Block Public Access 설정이 활성화되어 있으며 S3 객체 소유권은 버킷 소유자 적용으로 설정되어 있습니다. 이 설정은 수정할 수 없습니다. 선택적으로 로컬 영역 네트워크 경계 그룹 내로만 액세스를 제한하려면 IAM 정책에서 `s3express:AllAccessRestrictedToLocalZoneGroup` 조건 키를 사용할 수 있습니다. 자세한 내용은 로컬 영역의 디렉터리 버킷에 대한 인증 및 권한 부여 섹션을 참조하세요.

VPC에서의 프라이빗 연결

게이트웨이 엔드포인트를 사용하여 가상 프라이빗 클라우드(VPC)에 인터넷 게이트웨이 또는 NAT 디바이스를 사용하지 않고도 추가 비용 없이 VPC에서 AWS 로컬 영역(로컬 영역)의 디렉터리 버킷에 액세스할 수 있습니다. 다음 주제에서는 VPC와 로컬 영역의 디렉터리 버킷 간에 게이트웨이 VPC 엔드포인트를 구성하는 방법을 설명합니다.

게이트웨이 VPC 엔드포인트를 구성하는 방법

1. Amazon VPC 콘솔을 엽니다.
2. 탐색 창에서 엔드포인트를 선택합니다.
3. Create endpoint(엔드포인트 생성)을 선택합니다.
4. 엔드포인트의 이름을 지정합니다.
5. 서비스 범주(Service category)에서 AWS 서비스를 선택합니다.
6. 서비스에서 유형=게이트웨이 필터를 추가한 다음 com.amazonaws.***region***.s3express 옆의 옵션 버튼을 선택합니다.
7. VPC에서 엔드포인트를 생성할 VPC를 선택합니다.
8. 라우팅 테이블에서 엔드포인트에서 사용할 로컬 영역의 라우팅 테이블을 선택합니다. 엔드포인트가 생성되면 이 단계에서 선택한 라우팅 테이블에 라우팅 레코드가 추가됩니다.
9. 정책에서 모든 액세스를 선택하여 VPC 엔드포인트를 통한 모든 리소스에 대한 모든 보안 주체의 모든 작업을 허용합니다. 또는 사용자 지정을 선택하여 VPC 엔드포인트를 통해 리소스에 대한 작업을 수행하는 위탁자의 권한을 제어하는 VPC 엔드포인트 정책을 연결합니다.
10. (선택 사항) 태그를 추가하려면 새 태그 추가를 선택하고 태그 키와 태그 값을 입력합니다.
11. Create endpoint(엔드포인트 생성)을 선택합니다.

게이트웨이 VPC 엔드포인트에 대한 자세한 내용은 AWS PrivateLink 안내서에서 Gateway endpoints를 참조하세요. 데이터 레지던시 사용 사례의 경우 게이트웨이 VPC 엔드포인트를 사용하여 VPC에서만 버킷에 액세스할 수 있도록 활성화하는 것이 좋습니다. VPC 또는 VPC 엔드포인트로 액세스가 제한되면 AWS Management Console, REST API, AWS CLI 및 AWS SDK를 통해 객체에 액세스할 수 있습니다.

> **Note**
>
> AWS Management Console을 사용하여 VPC 또는 VPC 엔드포인트로 액세스를 제한하려면 AWS Management Console 프라이빗 액세스를 사용해야 합니다. 자세한 내용은 AWS Management Console 안내서의 AWS Management Console Private Access를 참조하세요.

로컬 영역에서 디렉터리 버킷 생성

전용 로컬 영역에서는 디렉터리 버킷을 생성하여 특정 데이터 경계 내에서 객체를 저장하고 검색하여 데이터 레지던시 및 데이터 격리 사용 사례를 충족할 수 있습니다. S3 디렉터리 버킷은 로컬 영역에서 지원되는 유일한 버킷 유형이며 `LocalZone`이라는 버킷 위치 유형을 포함합니다. 디렉터리 버킷 이름은 사용자가 제공하는 기본 이름과 버킷 위치의 영역 ID 및 `--x-s3`가 포함된 접미사로 구성됩니다. DescribeAvailabilityZones API 작업을 사용하여 로컬 영역 ID의 목록을 가져올 수 있습니다. 자세한 내용은 디렉터리 버킷 이름 지정 규칙 섹션을 참조하세요.

> **Note**
>
> - 전용 로컬 영역에서 리소스를 생성하거나 액세스하려면 먼저 AWS 전용 로컬 영역(전용 로컬 영역)의 S3를 비롯한 모든 서비스에 대해 관리자가 AWS 계정을 활성화해야 합니다. 자세한 내용은 로컬 영역에 대해 계정 활성화 섹션을 참조하세요.
> - 데이터 레지던시 요구 사항의 경우 게이트웨이 VPC 엔드포인트에서만 버킷에 액세스할 수 있도록 활성화하는 것이 좋습니다. 자세한 내용은 VPC에서의 프라이빗 연결 섹션을 참조하세요.
> - 로컬 영역 네트워크 경계 그룹 내로만 액세스를 제한하려면 IAM 정책에서 `s3express:AllAccessRestrictedToLocalZoneGroup` 조건 키를 사용할 수 있습니다. 자세한 내용은 로컬 영역의 디렉터리 버킷에 대한 인증 및 권한 부여 섹션을 참조하세요.

다음은 AWS Management Console, AWS CLI 및 AWS SDK를 사용하여 단일 로컬 영역에서 디렉터리 버킷을 생성하는 방법을 설명합니다.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 디렉터리 버킷을 생성하려는 로컬 영역의 상위 리전을 선택합니다.

   > **Note**
   >
   > 상위 리전에 대한 자세한 내용은 로컬 영역의 디렉터리 버킷에 대한 개념 섹션을 참조하세요.

3. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
4. 버킷 만들기를 선택합니다.

   버킷 만들기 페이지가 열립니다.
5. 일반 구성에서 버킷이 생성될 AWS 리전을 확인합니다.
6. 버킷 유형에서 디렉터리를 선택합니다.

   > **Note**
   >
   > - 디렉터리 버킷을 지원하지 않는 리전을 선택한 경우 버킷 유형이 범용 버킷으로 기본 설정됩니다. 디렉터리 버킷을 생성하려면 지원되는 리전을 선택해야 합니다. 디렉터리 버킷을 지원하는 리전 목록은 the section called “디렉터리 버킷에 대한 리전 및 영역 엔드포인트” 섹션을 참조하세요.
   > - 버킷을 생성한 후에는 버킷 유형을 변경할 수 없습니다.

7. 버킷 위치에서 사용할 로컬 영역을 선택합니다.

   > **Note**
   >
   > 버킷 생성 후에는 로컬 영역을 변경할 수 없습니다.

8. 버킷 위치에서 확인란을 선택하여 로컬 영역 중단 시 데이터를 사용할 수 없거나 데이터가 손실될 수 있음을 확인합니다.

> **Important**
>
> 디렉터리 버킷은 단일 로컬 영역 내의 여러 디바이스에 저장되지만 로컬 영역 간에 데이터를 중복으로 저장하지는 않습니다.

9. 버킷 이름에서 디렉터리 버킷 이름을 입력합니다.

   디렉터리 버킷 이름 지정 규칙에 대한 자세한 내용은 범용 버킷 이름 지정 규칙 섹션을 참조하세요. 콘솔을 사용하여 디렉터리 버킷을 생성하면 제공하는 기본 이름에 접미사가 자동으로 추가됩니다. 이 접미사에는 선택한 로컬 영역의 영역 ID가 포함됩니다.

   버킷을 생성한 후에는 해당 이름을 변경할 수 없습니다.

   > **Important**
   >
   > 버킷 이름에 계정 번호와 같은 중요한 정보를 포함하지 마세요. 버킷 이름은 버킷의 객체를 가리키는 URL에 표시됩니다.

10. 객체 소유권에서 버킷 소유자 적용 설정이 자동으로 활성화되고 모든 액세스 제어 목록(ACL)이 비활성화됩니다. 디렉터리 버킷의 경우 ACL은 비활성화되어 있으며 활성화할 수 없습니다.

    버킷 소유자 적용 설정이 활성화되어 있으면 버킷 소유자는 버킷의 모든 객체를 자동으로 소유하고 완전히 제어합니다. ACL은 더 이상 S3 버킷의 데이터에 대한 권한에 영향을 주지 않습니다. 버킷은 정책을 독점적으로 사용하여 액세스 제어를 정의합니다. Amazon S3의 최신 사용 사례 대부분은 더 이상 ACL을 사용할 필요가 없습니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

11. 이 버킷의 퍼블릭 액세스 차단 설정에서 디렉터리 버킷의 퍼블릭 액세스 차단 설정이 자동으로 활성화됩니다. 디렉터리 버킷의 경우 이 설정을 수정할 수 없습니다. 퍼블릭 액세스 차단에 대한 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하십시오.

12. 기본 암호화에서 디렉터리 버킷은 기본적으로 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)를 통해 데이터를 암호화합니다. 또한 AWS Key Management Service 키를 사용한 서버 측 암호화(SSE-KMS)를 사용하여 디렉터리 버킷의 데이터를 암호화하는 옵션도 있습니다.

13. 버킷 생성을 선택합니다.

    버킷을 생성한 후 버킷에 파일 및 폴더를 추가할 수 있습니다. 자세한 내용은 the section called “디렉터리 버킷의 객체 작업” 섹션을 참조하세요.

AWS CLI 사용

이 예시는 AWS CLI를 사용하여 로컬 영역에서 디렉터리 버킷을 생성하는 방법을 보여줍니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

디렉터리 버킷을 생성할 때는 구성 세부 정보를 제공하고 *bucket-base-name*--*zone-id*--x-s3와 같은 명명 규칙을 사용해야 합니다.

```
aws s3api create-bucket
--bucket bucket-base-name--zone-id--x-s3
--create-bucket-configuration 'Location={Type=LocalZone,Name=local-zone-
id},Bucket={DataRedundancy=SingleLocalZone,Type=Directory}'
--region parent-region-code
```

로컬 영역 ID 및 상위 리전 코드에 대한 자세한 내용은 로컬 영역의 디렉터리 버킷에 대한 개념 섹션을 참조하세요. AWS CLI 명령에 대한 자세한 내용은 AWS CLI Command Reference의 create-bucket을 참조하세요.

AWS SDK 사용

SDK for Go

이 예시는 AWS SDK for Go를 사용하여 로컬 영역에서 디렉터리 버킷을 생성하는 방법을 보여줍니다.

Example

```
var bucket = "bucket-base-name--zone-id--x-s3" // The full directory bucket name

func runCreateBucket(c *s3.Client) {
    resp, err := c.CreateBucket(context.Background(), &s3.CreateBucketInput{
        Bucket: &bucket,
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            Location: &types.LocationInfo{
                Name: aws.String("local-zone-id"),
                Type: types.LocationTypeLocalZone,
            },
            Bucket: &types.BucketInfo{
                DataRedundancy: types.DataRedundancySingleLocalZone,
                Type:           types.BucketTypeDirectory,
            },
        },
    })
```

```
    var terr *types.BucketAlreadyOwnedByYou
    if errors.As(err, &terr) {
        fmt.Printf("BucketAlreadyOwnedByYou: %s\n", aws.ToString(terr.Message))
        fmt.Printf("noop...\n") // No operation performed, just printing a message
        return
    }
    if err != nil {
        log.Fatal(err)
    }

    fmt.Printf("bucket created at %s\n", aws.ToString(resp.Location))
}
```

SDK for Java 2.x

이 예시는 AWS SDK for Java 2.x를 사용하여 로컬 영역에서 디렉터리 버킷을 생성하는 방법을 보여줍니다.

Example

```
public static void createBucket(S3Client s3Client, String bucketName) {

    //Bucket name format is {base-bucket-name}--{local-zone-id}--x-s3
    //example: doc-example-bucket--local-zone-id--x-s3 is a valid name for a
 directory bucket created in a Local Zone.

    CreateBucketConfiguration bucketConfiguration =
 CreateBucketConfiguration.builder()
             .location(LocationInfo.builder()
                     .type(LocationType.LOCAL_ZONE)
                     .name("local-zone-id").build()) //this must match the Local
 Zone ID in your bucket name
             .bucket(BucketInfo.builder()
                    .type(BucketType.DIRECTORY)
                    .dataRedundancy(DataRedundancy.SINGLE_LOCAL_ZONE)
                    .build()).build();
    try {

             CreateBucketRequest bucketRequest =
 CreateBucketRequest.builder().bucket(bucketName).createBucketConfiguration(bucketConfigurat
             CreateBucketResponse response = s3Client.createBucket(bucketRequest);
             System.out.println(response);
    }
```

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

AWS SDK for JavaScript

이 예시는 AWS SDK for JavaScript를 사용하여 로컬 영역에서 디렉터리 버킷을 생성하는 방법을 보여줍니다.

Example

```
// file.mjs, run with Node.js v16 or higher
// To use with the preview build, place this in a folder
// inside the preview build directory, such as /aws-sdk-js-v3/workspace/

import { S3 } from "@aws-sdk/client-s3";

const region = "parent-region-code";
const zone = "local-zone-id";
const suffix = `${zone}--x-s3`;

const s3 = new S3({ region });

const bucketName = `bucket-base-name--${suffix}`; // Full directory bucket name

const createResponse = await s3.createBucket(
    { Bucket: bucketName,
      CreateBucketConfiguration: {Location: {Type: "LocalZone", Name: "local-zone-
id"},
      Bucket: { Type: "Directory", DataRedundancy: "SingleLocalZone" }}
    }
   );
```

SDK for .NET

이 예시는 SDK for .NET을 사용하여 로컬 영역에서 디렉터리 버킷을 생성하는 방법을 보여줍니다.

Example

```
using (var amazonS3Client = new AmazonS3Client())
```

```
{
    var putBucketResponse = await amazonS3Client.PutBucketAsync(new PutBucketRequest
    {

       BucketName = "bucket-base-name--local-zone-id--x-s3",
       PutBucketConfiguration = new PutBucketConfiguration
       {
         BucketInfo = new BucketInfo { DataRedundancy =
 DataRedundancy.SingleLocalZone, Type = BucketType.Directory },
         Location = new LocationInfo { Name = "local-zone-id", Type =
 LocationType.LocalZone }
       }
     }).ConfigureAwait(false);
}
```

SDK for PHP

이 예시는 AWS SDK for PHP를 사용하여 로컬 영역에서 디렉터리 버킷을 생성하는 방법을 보여줍니다.

Example

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

$s3Client = new S3Client([

    'region'      => 'parent-region-code',
]);


$result = $s3Client->createBucket([
    'Bucket' => 'bucket-base-name--local-zone-id--x-s3',
    'CreateBucketConfiguration' => [
        'Location' => ['Name'=> 'local-zone-id', 'Type'=> 'LocalZone'],
        'Bucket' => ["DataRedundancy" => "SingleLocalZone" ,"Type" => "Directory"]
  ],
]);
```

SDK for Python

이 예시는 AWS SDK for Python (Boto3)을 사용하여 로컬 영역에서 디렉터리 버킷을 생성하는 방법을 보여줍니다.

Example

```
import logging
import boto3
from botocore.exceptions import ClientError

def create_bucket(s3_client, bucket_name, local_zone):
    '''
    Create a directory bucket in a specified Local Zone

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket to create; for example, 'bucket-base-name--local-
zone-id--x-s3'
    :param local_zone: String; Local Zone ID to create the bucket in
    :return: True if bucket is created, else False
    '''

    try:
        bucket_config = {
                'Location': {
                    'Type': 'LocalZone',
                    'Name': local_zone
                },
                'Bucket': {
                    'Type': 'Directory',
                    'DataRedundancy': 'SingleLocalZone'
                }
            }
        s3_client.create_bucket(
            Bucket = bucket_name,
            CreateBucketConfiguration = bucket_config
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True


if __name__ == '__main__':
    bucket_name = 'BUCKET_NAME'
    region = 'parent-region-code'
    local_zone = 'local-zone-id'
    s3_client = boto3.client('s3', region_name = region)
```

```
    create_bucket(s3_client, bucket_name, local_zone)
```

SDK for Ruby

이 예시는 AWS SDK for Ruby를 사용하여 로컬 영역에서 디렉터리 버킷을 생성하는 방법을 보여줍니다.

Example

```
s3 = Aws::S3::Client.new(region:'parent-region-code')
s3.create_bucket(
  bucket: "bucket-base-name--local-zone-id--x-s3",
  create_bucket_configuration: {
    location: { name: 'local-zone-id', type: 'LocalZone' },
    bucket: { data_redundancy: 'SingleLocalZone', type: 'Directory' }
  }
)
```

로컬 영역의 디렉터리 버킷에 대한 인증 및 권한 부여

로컬 영역의 디렉터리 버킷은 AWS Identity and Access Management(IAM) 권한 부여와 세션 기반 권한 부여를 모두 지원합니다. 디렉터리 버킷의 인증 및 권한 부여에 대한 자세한 내용은 요청 인증 및 권한 부여 섹션을 참조하세요.

리소스

디렉터리 버킷의 Amazon 리소스 이름(ARN)에는 `s3express` 네임스페이스, AWS 상위 리전, AWS 계정 ID 및 디렉터리 버킷 이름(영역 ID 포함)이 포함됩니다. 디렉터리 버킷에 액세스하고 작업을 수행하려면 다음 ARN 형식을 사용해야 합니다.

```
arn:aws:s3express:region-code:account-id:bucket/bucket-base-name--ZoneID--x-s3
```

로컬 영역의 디렉터리 버킷의 경우 영역 ID는 로컬 영역의 ID입니다. 로컬 영역의 디렉터리 버킷에 대한 자세한 내용은 로컬 영역의 디렉터리 버킷에 대한 개념 섹션을 참조하세요. ARN에 대한 자세한 내용은 IAM 사용 설명서의 Amazon 리소스 이름(ARN)을 참조하세요. 리소스에 대한 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소: Resource를 참조하세요.

로컬 영역의 디렉터리 버킷에 대한 조건 키

로컬 영역에서는 IAM 정책에서 이 모든 조건 키를 사용할 수 있습니다. 또한 로컬 영역 네트워크 경계 그룹 주위에 데이터 경계를 생성하려면 `s3express:AllAccessRestrictedToLocalZoneGroup` 조건 키를 사용하여 그룹 외부의 모든 요청을 거부하면 됩니다.

IAM 정책 문이 적용되는 조건을 더 세분화하는 데 다음 조건 키를 사용할 수 있습니다. 디렉터리 버킷에서 지원하는 API 작업, 정책 작업 및 조건 키의 전체 목록은 Policy actions for directory buckets을 참조하세요.

> **Note**
>
> 다음 조건 키는 로컬 영역에만 적용되며 가용 영역 및 AWS 리전에서는 지원되지 않습니다.

API 작업	정책 작업	설명	조건 키	설명	형식
영역 엔드포인트 API 작업	`s3express:CreateSession`	모든 영역 엔드포인트 API 작업(예: `CreateSession` , `HeadBucket` , `CopyObject` , `PutObject` , `GetObject`)에 대한 액세스 권한을 부여하는 데 사용되는 세션 토큰을 생성할 권한을 부여합니다.	`s3express:AllAccessRestrictedToLocalZoneGroup`	요청이 이 조건 키에 제공된 AWS 로컬 영역 네트워크 경계 그룹에서 시작되지 않는 한 버킷에 대한 모든 액세스를 필터링합니다. 값: 로컬 영역 네트워크 경계 그룹 값	`String`

정책 예제

정의한 데이터 레지던시 경계(구체적으로, 동일한 AWS 리전의 상위 로컬 영역 집합인 로컬 영역 그룹) 내의 요청에 대한 객체 액세스를 제한하려면 다음 정책 중 하나를 설정하면 됩니다.

- 서비스 제어 정책(SCP). SCP에 대한 자세한 내용은 AWS Organizations 사용 설명서에서 Service control policies (SCPs)를 참조하세요.
- IAM 역할에 대한 IAM ID 기반 정책

- VPC 엔드포인트 정책. VPC 엔드포인트 정책에 대한 자세한 내용은 AWS PrivateLink 안내서의 Control access to VPC endpoints using endpoint policies를 참조하세요.
- S3 버킷 정책

> **Note**
>
> `s3express:AllAccessRestrictedToLocalZoneGroup` 조건 키는 온프레미스 환경에서 시작되는 액세스를 지원하지 않습니다. 온프레미스 환경에서 시작되는 액세스를 지원하려면 정책에 소스 IP를 추가해야 합니다. 자세한 내용은 IAM 사용 설명서의 aws:SourceIp를 참조하세요.

Example – SCP 정책

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Access-to-specific-LocalZones-only",
            "Effect": "Deny",
            "Action": [
                "s3express:*",
            ],
            "Resource": "*",
            "Condition": {
                "StringNotEqualsIfExists": {
                    "s3express:AllAccessRestrictedToLocalZoneGroup": [
                        "local-zone-network-border-group-value"
                    ]
                }
            }
        }
    ]
}
```

Example – IAM ID 기반 정책(IAM 역할에 연결됨)

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Deny",
        "Action": "s3express:CreateSession",
        "Resource": "*",
        "Condition": {
            "StringNotEqualsIfExists": {
                "s3express:AllAccessRestrictedToLocalZoneGroup": [
                    "local-zone-network-border-group-value"
                ]
            }
        }
    }
}
```

Example - VPC 엔드포인트 정책

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Access-to-specific-LocalZones-only",
            "Principal": "*",
            "Action": "s3express:CreateSession",
            "Effect": "Deny",
            "Resource": "*",
            "Condition": {
                 "StringNotEqualsIfExists": {
                     "s3express:AllAccessRestrictedToLocalZoneGroup": [
                         "local-zone-network-border-group-value"
                     ]
                 }
            }
        }
    ]
}
```

Example - 버킷 정책

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Access-to-specific-LocalZones-only",
            "Principal": "*",
            "Action": "s3express:CreateSession",
            "Effect": "Deny",
            "Resource": "*",
            "Condition": {
                 "StringNotEqualsIfExists": {
                     "s3express:AllAccessRestrictedToLocalZoneGroup": [
                         "local-zone-network-border-group-value"
                     ]
                 }
            }
        }
    ]
}
```

디렉터리 버킷의 차이점

Amazon S3를 사용할 때 애플리케이션 및 성능 요구 사항에 가장 적합한 버킷 유형을 선택할 수 있습니다. 디렉터리 버킷은 지연 시간이 짧은 사용 사례 또는 데이터 레지던시 사용 사례에 가장 적합한 버킷 유형입니다. 디렉터리 버킷에 대한 자세한 내용은 디렉터리 버킷 작업 섹션을 참조하세요.

디렉터리 버킷의 차이점에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- 디렉터리 버킷의 차이점
- 디렉터리 버킷에 지원되는 API 작업
- 디렉터리 버킷에서 지원되지 않는 Amazon S3 기능

디렉터리 버킷의 차이점

- 디렉터리 버킷 이름

  - 디렉터리 버킷 이름은 사용자가 제공하는 기본 이름과 버킷이 위치한 AWS 영역(가용 영역 또는 로컬 영역)의 ID, 마지막으로 `--x-s`로 구성됩니다. 디렉터리 버킷 이름의 규칙 및 예제 목록은 디렉터리 버킷 이름 지정 규칙을 참조하세요.
- **`ListObjectsV2`** 동작
  - 디렉터리 버킷의 경우 `ListObjectsV2`는 객체를 사전순(알파벳순)으로 반환하지 않습니다. 또한 접두사는 구분 기호로 끝나야 하며 '/'만 구분 기호로 지정할 수 있습니다.
  - 디렉터리 버킷의 경우 `ListObjectsV2` 응답에는 진행 중인 멀티파트 업로드에만 관련된 접두사가 포함됩니다.
- 삭제 동작 - 디렉터리 버킷에서 객체를 삭제하면 Amazon S3는 객체 경로에 있는 빈 디렉터리를 재귀적으로 삭제합니다. 예를 들어 `dir1/dir2/file1.txt` 객체 키를 삭제할 경우 Amazon S3는 `file1.txt`를 삭제합니다. `dir1/` 및 `dir2/` 디렉터리가 비어 있고 다른 객체가 없는 경우 Amazon S3는 해당 디렉터리도 삭제합니다.
- ETag와 체크섬 - 디렉터리 버킷의 엔터티 태그(ETag)는 객체에 고유한 임의의 영숫자 문자열이며 MD5 체크섬이 아닙니다. 디렉터리 버킷에서 추가 체크섬을 사용하는 방법에 대한 자세한 내용은 S3 추가 체크섬 모범 사례 섹션을 참조하세요.
- **`DeleteObjects`** 요청의 객체 키
  - `DeleteObjects` 요청의 객체 키는 공백이 아닌 문자를 하나 이상 포함해야 합니다. 공백 문자로만 구성된 문자열은 `DeleteObjects` 요청에서 지원되지 않습니다.
  - `DeleteObjects` 요청의 객체 키에는 유니코드 제어 문자를 포함할 수 없습니다. 단, 줄바꿈(`\n`), 탭(`\t`) 및 캐리지 리턴(`\r`) 문자는 예외입니다.
- 리전 및 영역 엔드포인트 – 디렉터리 버킷의 버킷 관리 API 작업은 리전 엔드포인트를 통해 사용할 수 있으며 이를 리전 엔드포인트 API 작업이라고 합니다. 리전 엔드포인트 API 작업의 예로는 CreateBucket 및 DeleteBucket이 있습니다. 디렉터리 버킷을 생성한 후 영역 엔드포인트 API 작업을 사용하여 디렉터리 버킷에 객체를 업로드하고 관리할 수 있습니다. 영역 엔드포인트 API 작업은 영역 엔드포인트를 통해 사용할 수 있습니다. 영역 엔드포인트 API 작업의 예로는 `PutObject` 및 `CopyObject`가 있습니다. 디렉터리 버킷을 사용할 때 모든 요청에서 리전을 지정해야 합니다. 리전 엔드포인트의 경우 리전을 지정합니다(예: `s3express-control.us-west-2.amazonaws.com`). 영역 엔드포인트의 경우 리전과 가용 영역을 모두 지정합니다(예: `s3express-usw2-az1.us-west-2.amazonaws.com`). 자세한 내용은 디렉터리 버킷에 대한 리전 및 영역 엔드포인트 섹션을 참조하세요.
- 멀티파트 업로드 - 멀티파트 업로드 프로세스를 사용하여 디렉터리 버킷에 저장된 큰 객체를 업로드하고 복사할 수 있습니다. 그러나 디렉터리 버킷에 저장된 객체에 멀티파트 업로드 프로세스를 사용할 경우 다음과 같은 몇 가지 차이점이 있습니다. 자세한 내용은 the section called “디렉터리 버킷에 멀티파트 업로드 사용” 섹션을 참조하세요.

  - 객체 생성 날짜는 멀티파트 업로드 완료 날짜입니다.
  - 멀티파트의 파트 번호에는 연속된 번호를 사용해야 합니다. 연속되지 않는 파트 번호로 멀티파트 업로드 요청을 완료하려고 하면 Amazon S3에서 HTTP `400 (Bad Request)` 오류가 생성됩니다.
  - 멀티파트 업로드 작업을 시작한 사용자는 `s3express:CreateSession` 권한을 통해 `AbortMultipartUpload`에 대한 명시적 허용 액세스 권한을 부여받은 경우에만 멀티파트 업로드 요청을 중단할 수 있습니다. 자세한 내용은 IAM을 사용한 리전 엔드포인트 API 권한 부여 섹션을 참조하세요.
- 디렉터리 버킷 비우기 - AWS Command Line Interface(CLI)를 통한 `s3 rm` 명령, Mountpoint를 통한 `delete` 작업 그리고 AWS Management Console을 통한 빈 버킷 옵션 버튼으로는 디렉터리 버킷에서 진행 중인 멀티파트 업로드를 삭제할 수 없습니다. 진행 중인 멀티파트 업로드를 삭제하려면 `ListMultipartUploads` 작업을 사용하여 버킷에서 진행 중인 멀티파트 업로드를 나열하고 `AbortMultipartUpload` 작업을 사용하여 진행 중인 멀티파트 업로드를 모두 중단하세요.
- AWS 로컬 영역 - 로컬 영역은 범용 버킷이 아닌 디렉터리 버킷에만 지원됩니다.
  - 로컬 영역에 있는 디렉터리 버킷에는 기존 객체에 데이터 추가가 지원되지 않습니다. 가용 영역에 있는 디렉터리 버킷의 기존 객체에만 데이터를 추가할 수 있습니다.
  - 로컬 영역의 디렉터리 버킷에는 S3 수명 주기가 지원되지 않습니다.

디렉터리 버킷에 지원되는 API 작업

디렉터리 버킷은 리전(버킷 수준, 즉 컨트롤 플레인) 및 영역(객체 수준, 즉 데이터 플레인) 엔드포인트 API 작업을 모두 지원합니다. 자세한 내용은 디렉터리 버킷에 대한 네트워킹 및 엔드포인트 및 게이트웨이 VPC 엔드포인트(을)를 참조하세요. 지원되는 API 작업 목록은 디렉터리 버킷 API 작업 섹션을 참조하세요.

디렉터리 버킷에서 지원되지 않는 Amazon S3 기능

다음은 디렉터리 버킷에서 지원되지 않는 Amazon S3 기능입니다.

- AWS 관리형 정책
- AWS PrivateLink for S3
- MD5 체크섬
- 멀티 팩터 인증(MFA) 삭제
- S3 객체 잠금
- 요청자 지불

- S3 Access Grants
- Amazon CloudWatch 요청 지표
- S3 이벤트 알림
- S3 수명 주기 전환 작업
- S3 다중 리전 액세스 포인트
- S3 객체 Lambda 액세스 포인트
- S3 버전 관리
- S3 인벤토리
- S3 복제
- 객체 태그
- S3 Select
- 서버 액세스 로그
- 정적 웹 사이트 호스팅
- S3 Storage Lens
- S3 Storage Lens 그룹
- S3 Transfer Acceleration
- AWS Key Management Service(AWS KMS) 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)
- 고객 제공 키를 사용한 서버 측 암호화(SSE-C)
- Amazon S3 콘솔에서 새 버킷을 생성할 때 기존 버킷 설정을 복사하는 옵션입니다.
- 향상된 액세스 거부(HTTP `403 Forbidden`) 오류 메시지

디렉터리 버킷에 대한 네트워킹

디렉터리 버킷 및 객체 내부에 액세스하려면 표준 Amazon S3 엔드포인트와 다른 리전 및 영역 API 엔드포인트를 사용합니다. 사용하는 S3 API 작업에 따라 영역 엔드포인트 또는 리전 엔드포인트가 필요합니다. 엔드포인트 유형별 API 작업의 전체 목록은 디렉터리 버킷의 차이점 섹션을 참조하세요.

게이트웨이 가상 프라이빗 클라우드(VPC) 엔드포인트를 통해 영역 및 리전 API 작업에 모두 액세스할 수 있습니다.

다음 주제에서는 게이트웨이 VPC 엔드포인트를 사용하여 S3 Express One Zone에 액세스하기 위한 네트워킹 요구 사항을 설명합니다.

주제

- 엔드포인트
- VPC 게이트웨이 엔드포인트 구성

엔드포인트

게이트웨이 VPC 엔드포인트를 사용하여 VPC에서 디렉터리 버킷과 객체 내부에 액세스할 수 있습니다. 디렉터리 버킷은 리전 및 영역 API 엔드포인트를 사용합니다. 사용하는 Amazon S3 API 작업에 따라 리전 엔드포인트 또는 영역 엔드포인트가 필요합니다. 게이트웨이 엔드포인트 사용에 따르는 추가 요금은 없습니다.

버킷 수준(즉 컨트롤 플레인) API 작업은 리전 엔드포인트를 통해 사용할 수 있으며 이를 리전 엔드포인트 API 작업이라고 합니다. 리전 엔드포인트 API 작업의 예로는 `CreateBucket` 및 `DeleteBucket`이 있습니다. 디렉터리 버킷을 생성할 때는 디렉터리 버킷을 생성할 단일 영역(가용 영역 또는 로컬 영역)을 선택합니다. 디렉터리 버킷을 생성한 후 영역 엔드포인트 API 작업을 사용하여 디렉터리 버킷에 객체를 업로드하고 관리할 수 있습니다.

객체 수준(즉 데이터 영역) API 작업은 영역 엔드포인트를 통해 사용할 수 있으며 이를 영역 엔드포인트 API 작업이라고 합니다. 영역 엔드포인트 API 작업의 예로는 `CreateSession` 및 `PutObject`가 있습니다.

가용 영역의 디렉터리 버킷을 지원하는 엔드포인트 및 위치에 대한 자세한 내용은 가용 영역의 디렉터리 버킷에 대한 엔드포인트 섹션을 참조하세요.

로컬 영역의 디렉터리 버킷을 지원하는 엔드포인트 및 위치에 대한 자세한 내용은 로컬 영역에 대해 계정 활성화 섹션을 참조하세요.

VPC 게이트웨이 엔드포인트 구성

가용 영역의 디렉터리 버킷 액세스를 위한 게이트웨이 VPC 엔드포인트를 구성하려면 the section called “VPC 게이트웨이 엔드포인트 구성” 섹션을 참조하세요.

로컬 영역의 디렉터리 버킷 액세스를 위한 게이트웨이 VPC 엔드포인트를 구성하려면 VPC에서의 프라이빗 연결 섹션을 참조하세요.

디렉터리 버킷 이름 지정 규칙

Amazon S3에서 디렉터리 버킷을 생성할 때 다음 버킷 이름 지정 규칙이 적용됩니다. 범용 버킷의 이름 지정 규칙은 범용 버킷 이름 지정 규칙 섹션을 참조하세요.

디렉터리 버킷 이름은 사용자가 제공하는 기본 이름과 버킷이 위치한 AWS 영역(가용 영역 또는 로컬 영역)의 ID 및 `--x-s3`이 포함된 접미사로 구성됩니다. *`zone-id`*는 가용 영역 또는 로컬 영역의 ID일 수 있습니다.

```
base-name--zoneid--x-s3
```

예를 들어, 다음 디렉터리 버킷 이름에는 가용 영역 ID인 `usw2-az1`이 포함되어 있습니다.

```
bucket-base-name--usw2-az1--x-s3
```

> **Note**
>
> 콘솔을 사용하여 디렉터리 버킷을 생성하면 제공하는 기본 이름에 접미사가 자동으로 추가됩니다. 이 접미사에는 선택한 영역(가용 영역 또는 로컬 영역)의 영역 ID가 포함됩니다.
> API를 사용하여 디렉터리 버킷을 생성할 때는 요청에 영역 ID를 포함한 전체 접미사를 제공해야 합니다. 영역 ID 목록은 엔드포인트 섹션을 참조하세요.

디렉터리 버킷에는 다음 이름 지정 규칙이 적용됩니다.

- 선택한 영역(AWS 가용 영역 또는 AWS 로컬 영역) 내에서 고유해야 합니다.
- 이름은 최소 3자에서 최대 63자 사이여야 합니다(접미사 포함).
- 소문자, 숫자, 하이픈(-)으로만 구성해야 합니다.
- 문자나 숫자로 시작하고 끝나야 합니다.
- *`--zone-id`*`--x-s3`을 접미사로 포함해야 합니다.
- 버킷 이름은 접두사 `xn--`로 시작해서는 안됩니다.
- 버킷 이름은 접두사 `sthree-`로 시작해서는 안됩니다.
- 버킷 이름은 접두사 `sthree-configurator`로 시작해서는 안됩니다.
- 버킷 이름은 접두사 `amzn-s3-demo-`로 시작해서는 안됩니다.
- 버킷 이름은 접미사 `-s3alias`로 끝나서는 안됩니다. 이 접미사는 액세스 포인트 별칭 이름 용도로 예약되어 있습니다. 자세한 내용은 액세스 포인트 별칭 섹션을 참조하세요.
- 버킷 이름은 접미사 `--ol-s3`로 끝나서는 안됩니다. 이 접미사는 객체 Lambda 액세스 포인트 별칭 이름 용도로 예약되어 있습니다. 자세한 내용은 S3 버킷 객체 Lambda 액세스 포인트에 버킷 스타일 별칭을 사용하는 방법 섹션을 참조하세요.

- 버킷 이름은 접미사 `.mrap`로 끝나서는 안됩니다. 이 접미사는 다중 리전 액세스 포인트 이름 용도로 예약되어 있습니다. 자세한 내용은 Amazon S3 다중 리전 액세스 포인트 이름 지정 규칙 섹션을 참조하세요.

디렉터리 버킷 속성 보기

Amazon S3 콘솔을 사용하여 Amazon S3 디렉터리 버킷의 속성을 보고 구성할 수 있습니다. 자세한 내용은 디렉터리 버킷 작업 섹션을 참조하세요.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 디렉터리 버킷을 선택합니다.
3. 디렉터리 버킷 목록에서 속성을 볼 버킷의 이름을 선택합니다.
4. 속성(Properties) 탭을 선택합니다.
5. 속성 탭에서 다음과 같은 버킷 속성을 확인할 수 있습니다.

   - 디렉터리 버킷 개요 - AWS 리전, 영역(가용 영역 또는 로컬 영역), Amazon 리소스 이름(ARN), 버킷 생성 날짜를 확인할 수 있습니다.
   - 서버 측 암호화 설정 - Amazon S3가 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)를 모든 S3 버킷 암호화의 기본 수준으로 적용합니다. Amazon S3에서는 객체를 디스크에 저장하기 전에 암호화하고 객체를 다운로드할 때 이를 해독합니다. 자세한 내용은 디렉터리 버킷의 기본 암호화 설정 및 모니터링 섹션을 참조하세요.

     디렉터리 버킷에 지원되는 기능에 대한 자세한 내용은 디렉터리 버킷 생성 및 사용 섹션을 참조하세요.

디렉터리 버킷 정책 관리

Amazon S3 콘솔, AWS SDK, AWS CLI를 사용하여 Amazon S3 디렉터리 버킷에 대한 버킷 정책을 추가, 삭제, 업데이트 및 조회할 수 있습니다. 자세한 내용은 다음 항목을 참조하세요. 지원되는 AWS Identity and Access Management(IAM) 작업에 대한 자세한 내용은 IAM을 사용한 리전 엔드포인트 API 권한 부여 섹션을 참조하세요. 디렉터리 버킷에 대한 버킷 정책의 예는 디렉터리 버킷에 대한 버킷 정책 예제 섹션을 참조하세요.

주제

- 버킷 정책 추가
- 버킷 정책 보기
- 버킷 정책 삭제

버킷 정책 추가

Amazon S3 콘솔, AWS SDK 또는 AWS CLI를 사용하여 디렉터리 버킷에 버킷 정책을 추가할 수 있습니다.

S3 콘솔 사용

버킷 정책 생성 또는 편집

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 디렉터리 버킷을 선택합니다.
3. 디렉터리 버킷 목록에서 정책을 추가할 버킷의 이름을 선택합니다.
4. 권한 탭을 선택합니다.
5. 버킷 정책에서 편집을 선택합니다. 버킷 정책 편집 페이지가 나타납니다.
6. 정책을 자동으로 생성하려면 정책 생성기를 선택합니다.

   정책 생성기를 선택하면 AWS 정책 생성기가 새 창에서 열립니다.

   AWS 정책 생성기를 사용하지 않으려면 정책 섹션에서 JSON 문을 추가하거나 편집할 수 있습니다.

   a. AWS 정책 생성기 페이지의 Select Type of Policy(정책 유형 선택)에 S3 Bucket Policy(S3 버킷 정책)를 선택합니다.

   b. 제공된 필드에 정보를 입력하여 명령문을 추가한 다음 명령문 추가(Add Statement)를 선택합니다. 문을 추가하려는 만큼 이 단계를 반복합니다. 이러한 필드에 대한 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하십시오.

> **Note**
>
> 편의를 위해 버킷 정책 편집 페이지의 정책 텍스트 필드 위에 현재 버킷의 버킷 ARN(Amazon 리소스 이름)이 표시됩니다. AWS 정책 생성기 페이지의 명령문에 사용하기 위해 이 ARN을 복사할 수 있습니다.

c. 명령문 추가를 마친 후 정책 생성(Generate Policy)을 선택합니다.

d. 생성된 정책 텍스트를 복사하고 닫기(Close)를 선택하고 Amazon S3 콘솔의 버킷 정책 편집(Edit bucket policy) 페이지로 돌아갑니다.

7. 정책 상자에서 기존 정책을 편집하거나 AWS 정책 생성기에서 버킷 정책을 붙여 넣습니다. 정책을 저장하기 전에 보안 경고, 오류, 일반 경고 및 제안 사항을 해결해야 합니다.

> **Note**
>
> 버킷 정책은 크기가 20KB로 제한됩니다.

8. 변경 사항 저장을 선택하면 권한 탭으로 돌아갑니다.

AWS SDK 사용

SDK for Java 2.x

Example

`PutBucketPolicy` AWS SDK for Java 2.x

```
public static void setBucketPolicy(S3Client s3Client, String bucketName, String
 policyText) {

      //sample policy text
      /**
       * policy_statement = {
        *         'Version': '2012-10-17',
        *         'Statement': [
        *             {
        *                 'Sid': 'AdminPolicy',
        *                 'Effect': 'Allow',
        *                 'Principal': {
        *                     "AWS": "111122223333"
```

```
         *                 },
         *                 'Action': 's3express:*',
         *                 'Resource':
 'arn:aws:s3express:region:111122223333:bucket/bucket-base-name--zone-id--x-s3'
         *             }
         *         ]
         *     }
         */
        System.out.println("Setting policy:");
        System.out.println("----");
        System.out.println(policyText);
        System.out.println("----");
        System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .policy(policyText)
                    .build();
            s3Client.putBucketPolicy(policyReq);
            System.out.println("Done!");
        }

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

AWS CLI 사용

이 예시는 AWS CLI를 사용하여 디렉터리 버킷에 버킷 정책을 추가하는 방법을 보여 줍니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

```
aws s3api put-bucket-policy --bucket bucket-base-name--zone-id--x-s3 --policy file://
bucket_policy.json
```

bucket_policy.json:

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

```
    "Statement": [
        {
            "Sid": "AdminPolicy",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": "s3express*",
            "Resource": "arn:aws:s3express:us-west-2:111122223333:bucket/amzn-s3-demo-
bucket--usw2-az1--x-s3"
        }
    ]
}
```

자세한 내용은 AWS Command Line Interface의 put-bucket-policy를 참조하세요.

버킷 정책 보기

다음 예시를 사용하여 디렉터리 버킷의 버킷 정책을 확인합니다.

AWS CLI 사용

이 예시는 AWS CLI를 사용하여 디렉터리 버킷에 연결된 버킷 정책을 확인하는 방법을 보여 줍니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

```
aws s3api get-bucket-policy --bucket bucket-base-name--zone-id--x-s3
```

자세한 내용은 AWS Command Line Interface의 get-bucket-policy를 참조하세요.

버킷 정책 삭제

다음 예시를 사용하여 디렉터리 버킷의 버킷 정책을 삭제합니다.

AWS SDK 사용

SDK for Java 2.x

Example

`DeleteBucketPolicy` AWS SDK for Java 2.x

```
public static void deleteBucketPolicy(S3Client s3Client, String bucketName) {
```

```
        try {
            DeleteBucketPolicyRequest deleteBucketPolicyRequest =
DeleteBucketPolicyRequest
                    .builder()
                    .bucket(bucketName)
                    .build()
            s3Client.deleteBucketPolicy(deleteBucketPolicyRequest);
            System.out.println("Successfully deleted bucket policy");
        }

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

AWS CLI 사용

이 예시는 AWS CLI를 사용하여 디렉터리 버킷의 버킷 정책을 삭제하는 방법을 보여 줍니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

```
aws s3api delete-bucket-policy --bucket bucket-base-name--zone-id--x-s3
```

자세한 내용은 AWS Command Line Interface의 delete-bucket-policy를 참조하세요.

디렉터리 버킷 비우기

Amazon S3 콘솔을 사용하여 Amazon S3 디렉터리 버킷을 비울 수 있습니다. 디렉터리 버킷에 대한 자세한 내용은 디렉터리 버킷 작업 섹션을 참조하세요.

디렉터리 버킷을 비우기 전에 다음 사항에 유의하세요.

- 디렉터리 버킷을 비우면 모든 객체가 삭제되지만 디렉터리 버킷은 유지됩니다.
- 디렉터리 버킷을 비운 후에는 비우기 작업을 실행 취소할 수 없습니다.
- 버킷 비우기 작업이 진행되는 동안 디렉터리 버킷에 추가된 객체가 삭제될 수 있습니다.

버킷도 삭제하려는 경우 다음에 유의하세요.

- 버킷 자체를 삭제하려면 먼저 디렉터리 버킷의 모든 객체를 삭제해야 합니다.
- 버킷 자체를 삭제하려면 먼저 디렉터리 버킷에서 진행 중인 멀티파트 업로드를 중단해야 합니다.

> **Note**
>
> AWS Command Line Interface(CLI)를 통한 `s3 rm` 명령, Mountpoint를 통한 `delete` 작업 그리고 AWS Management Console을 통한 빈 버킷 옵션 버튼으로는 디렉터리 버킷에서 진행 중인 멀티파트 업로드를 삭제할 수 없습니다. 진행 중인 멀티파트 업로드를 삭제하려면 `ListMultipartUploads` 작업을 사용하여 버킷에서 진행 중인 멀티파트 업로드를 나열하고 `AbortMultipartUpload` 작업을 사용하여 진행 중인 멀티파트 업로드를 모두 중단하세요.

디렉터리 버킷을 삭제하려면 디렉터리 버킷 삭제 섹션을 참조하세요. 진행 중인 멀티파트 업로드를 중단하려면 the section called “멀티파트 업로드 중단” 섹션을 참조하세요.

범용 버킷을 비우려면 범용 버킷 비우기 섹션을 참조하세요.

S3 콘솔 사용

디렉터리 버킷을 비우려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 디렉터리 버킷을 선택합니다.
3. 비우려는 버킷의 이름 옆에 있는 옵션 버튼을 선택한 후 비우기를 선택합니다.
4. 버킷 비우기 페이지에서 텍스트 필드에 **`permanently delete`**를 입력하여 해당 버킷 비우기를 확인한 후 비우기를 선택합니다.
5. 버킷 비우기: 상태 페이지에서 버킷 비우기 프로세스의 진행 상황을 모니터링합니다.

디렉터리 버킷 삭제

빈 Amazon S3 디렉터리 버킷만 삭제할 수 있습니다. 디렉터리 버킷을 삭제하기 전에 버킷 내의 모든 객체를 삭제하고 진행 중인 모든 멀티파트 업로드를 중단해야 합니다.

디렉터리 버킷이 액세스 포인트에 연결된 경우 먼저 액세스 포인트를 삭제해야 합니다. 자세한 내용은 디렉터리 버킷의 액세스 포인트 삭제 섹션을 참조하세요.

디렉터리 버킷을 비우려면 디렉터리 버킷 비우기 섹션을 참조하세요. 진행 중인 멀티파트 업로드를 중단하려면 the section called “멀티파트 업로드 중단” 섹션을 참조하세요.

범용 버킷을 삭제하려면 범용 버킷 삭제 섹션을 참조하세요.

S3 콘솔 사용

디렉터리 버킷을 비우고 진행 중인 모든 멀티파트 업로드를 중단한 후 버킷을 삭제할 수 있습니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 디렉터리 버킷을 선택합니다.
3. 디렉터리 버킷 목록에서, 삭제할 버킷의 이름 옆에 있는 옵션 버튼을 선택합니다.
4. 삭제를 선택합니다.
5. 버킷 삭제 페이지에서 텍스트 필드에 버킷 이름을 입력하여 버킷 삭제를 확인합니다.

> **Important**
>
> 디렉터리 버킷 삭제는 실행 취소할 수 없습니다.

6. 디렉터리 버킷을 삭제하려면 버킷 삭제를 선택합니다.

AWS SDK 사용

다음 예시에서는 AWS SDK for Java 2.x 및 AWS SDK for Python (Boto3)을 사용하여 디렉터리 버킷을 삭제합니다.

SDK for Java 2.x

Example

```
public static void deleteBucket(S3Client s3Client, String bucketName) {

    try {
        DeleteBucketRequest del = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();
        s3Client.deleteBucket(del);
        System.out.println("Bucket " + bucketName + " has been deleted");
    }
    catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
```

```
    }
}
```

SDK for Python

Example

```
import logging
import boto3
from botocore.exceptions import ClientError

def delete_bucket(s3_client, bucket_name):
    '''
    Delete a directory bucket in a specified Region

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket to delete; for example, 'doc-example-bucket--usw2-
az1--x-s3'
    :return: True if bucket is deleted, else False
    '''

    try:
        s3_client.delete_bucket(Bucket = bucket_name)
    except ClientError as e:
        logging.error(e)
        return False
    return True

if __name__ == '__main__':
    bucket_name = 'BUCKET_NAME'
    region = 'us-west-2'
    s3_client = boto3.client('s3', region_name = region)
```

AWS CLI 사용

이 예시는 AWS CLI를 사용하여 디렉터리 버킷을 삭제하는 방법을 보여 줍니다. 명령을 사용하려면 *### # ## ## ###*를 사용자의 정보로 대체합니다.

```
aws s3api delete-bucket --bucket bucket-base-name--zone-id--x-s3 --region us-west-2
```

자세한 내용은 AWS Command Line Interface의 delete-bucket을 참조하세요.

디렉터리 버킷 나열

다음 예시에서는 AWS Management Console, AWS SDK 및 AWS CLI를 사용하여 디렉터리 버킷을 나열하는 방법을 보여줍니다.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 디렉터리 버킷 목록을 보려는 리전을 선택합니다.
3. 왼쪽 탐색 창에서 디렉터리 버킷을 선택합니다. 디렉터리 버킷 목록이 나타납니다. 버킷의 객체, 버킷 속성, 버킷 권한, 지표, 버킷과 연결된 액세스 포인트를 보거나 버킷을 관리하려면 버킷 이름을 선택합니다.

AWS SDK 사용

SDK for Java 2.x

Example

다음 예제에서는 AWS SDK for Java 2.x을 사용하여 디렉터리 버킷을 나열합니다.

```
public static void listBuckets(S3Client s3Client) {
       try {
           ListDirectoryBucketsRequest listDirectoryBucketsRequest =
ListDirectoryBucketsRequest.builder().build();
           ListDirectoryBucketsResponse response =
s3Client.listDirectoryBuckets(listDirectoryBucketsRequest);
           if (response.hasBuckets()) {
               for (Bucket bucket: response.buckets()) {
                   System.out.println(bucket.name());
                   System.out.println(bucket.creationDate());
               }
           }
       }

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

```
            System.exit(1);
        }
    }
```

SDK for Python

Example

다음 예제에서는 AWS SDK for Python (Boto3)을 사용하여 디렉터리 버킷을 나열합니다.

```
import logging
import boto3
from botocore.exceptions import ClientError

def list_directory_buckets(s3_client):
'''
Prints a list of all directory buckets in a Region

:param s3_client: boto3 S3 client
:return: True if there are buckets in the Region, else False
'''
try:
    response = s3_client.list_directory_buckets()
    for bucket in response['Buckets']:
        print (bucket['Name'])
except ClientError as e:
    logging.error(e)
    return False
return True


if __name__ == '__main__':
    region = 'us-east-1'
    s3_client = boto3.client('s3', region_name = region)
    list_directory_buckets(s3_client)
```

SDK for .NET

Example

다음 예제에서는 AWS SDK for .NET을 사용하여 디렉터리 버킷을 나열합니다.

```
var listDirectoryBuckets = await amazonS3Client.ListDirectoryBucketsAsync(new
 ListDirectoryBucketsRequest
{
  MaxDirectoryBuckets = 10
  }).ConfigureAwait(false);
```

SDK for PHP

Example

다음 예제에서는 AWS SDK for PHP를 사용하여 디렉터리 버킷을 나열합니다.

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

$s3Client = new S3Client([
    'region'      => 'us-east-1',
]);
$result = $s3Client->listDirectoryBuckets();
```

SDK for Ruby

Example

다음 예제에서는 AWS SDK for Ruby를 사용하여 디렉터리 버킷을 나열합니다.

```
s3 = Aws::S3::Client.new(region:'us-west-1')
s3.list_directory_buckets
```

AWS CLI 사용

다음 `list-directory-buckets` 예시 명령은 AWS CLI를 사용하여 *us-east-1* 리전의 디렉터리 버킷을 나열하는 방법을 보여 줍니다. 이 명령을 실행하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws s3api list-directory-buckets --region us-east-1
```

자세한 정보는 AWS CLI 명령 참조의 list-directory-buckets 섹션을 참조하세요.

디렉터리 버킷 액세스 가능 여부 결정

다음 AWS SDK 예제에서는 HeadBucket API 작업을 사용하여 Amazon S3 디렉터리 버킷이 존재하는지, 그리고 액세스 권한이 있는지 확인하는 방법을 보여줍니다.

AWS SDK 사용

다음 AWS SDK for Java 2.x 예제에서는 버킷이 존재하고 버킷에 액세스할 수 있는 권한이 있는지 확인하는 방법을 보여줍니다.

SDK for Java 2.x

Example

AWS SDK for Java 2.x

```
public static void headBucket(S3Client s3Client, String bucketName) {
   try {
        HeadBucketRequest headBucketRequest = HeadBucketRequest
                .builder()
                .bucket(bucketName)
                .build();
        s3Client.headBucket(headBucketRequest);
        System.out.format("Amazon S3 bucket: \"%s\" found.", bucketName);
   }

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

AWS CLI 사용

다음 head-bucket 예시 명령에서는 AWS CLI를 사용하여 디렉터리 버킷이 존재하고 해당 버킷에 액세스하기 위한 권한이 있는지 확인하는 방법을 보여 줍니다. 이 명령을 실행하려면 user input placeholders를 사용자의 정보로 대체합니다.

```
aws s3api head-bucket --bucket bucket-base-name--zone-id--x-s3
```

자세한 정보는 AWS CLI 명령 참조의 head-bucket 섹션을 참조하세요.

디렉터리 버킷의 객체 작업

Amazon S3 디렉터리 버킷을 생성한 후에는 Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), AWS SDK를 사용하여 객체 작업을 할 수 있습니다.

디렉터리 버킷에서 대량 작업 수행, 객체 가져오기, 업로드, 복사, 삭제, 다운로드에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- 디렉터리 버킷으로 객체 가져오기
- 디렉터리 버킷의 S3 수명 주기 작업
- 디렉터리 버킷에 배치 작업 사용
- 디렉터리 버킷의 객체에 데이터 추가
- 디렉터리 버킷의 객체 이름 바꾸기
- 디렉터리 버킷에 객체 업로드
- 디렉터리 버킷에서 또는 디렉터리 버킷에 객체 복사
- 디렉터리 버킷에서 객체 삭제
- 디렉터리 버킷에서 객체 다운로드
- 객체 디렉터리 버킷을 공유하기 위해 미리 서명된 URL 생성
- 디렉터리 버킷에서 객체 메타데이터 검색
- 디렉터리 버킷에서 객체 나열

디렉터리 버킷으로 객체 가져오기

Amazon S3에서 디렉터리 버킷을 생성한 후 가져오기 작업을 사용하여 새 버킷에 데이터를 채울 수 있습니다. 가져오기는 범용 버킷에서 디렉터리 버킷으로 객체를 복사하는 S3 배치 작업 건을 생성하는 간소화된 방법입니다.

> **Note**
>
> 가져오기 작업에는 다음과 같은 제한 사항이 적용됩니다.

- 소스 버킷과 대상 버킷은 같은 AWS 리전 및 계정에 있어야 합니다.
- 소스 버킷은 디렉터리 버킷이 될 수 없습니다.
- 5GB보다 큰 객체는 지원되지 않으며 복사 작업에서 생략됩니다.
- Glacier Flexible Retrival, Glacier Deep Archive, Intelligent-Tiering Archive Access 계층 및 Intelligent-Tiering Deep Archive 계층 스토리지 클래스의 객체는 먼저 복원해야 가져올 수 있습니다.
- MD5 체크섬 알고리즘을 사용하여 가져온 객체는 CRC32 체크섬을 사용하도록 변환됩니다.
- 가져온 객체는 범용 버킷에서 사용하는 스토리지 클래스와는 요금 구조가 다른 Express One Zone 스토리지 클래스를 사용합니다. 다수의 객체를 가져올 때는 이러한 비용 차이를 고려하세요.

가져오기 작업을 구성할 때는 기존 객체를 복사할 소스 버킷 또는 접두사를 지정합니다. 또한 소스 객체에 액세스할 수 있는 권한이 있는 AWS Identity and Access Management(IAM) 역할을 제공합니다. 그러면 Amazon S3가 객체를 복사하고 적절한 스토리지 클래스와 체크섬 설정을 자동으로 적용하는 배치 작업 건을 시작합니다.

가져오기 작업을 구성하려면 Amazon S3 콘솔을 사용합니다.

Amazon S3 콘솔 사용

디렉터리 버킷으로 객체를 가져오는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷을 선택하고 디렉터리 버킷 탭을 선택합니다. 객체를 가져올 대상 디렉터리 버킷 옆에 있는 옵션 버튼을 선택합니다.
3. 가져오기를 선택합니다.
4. 소스에는 가져오려는 객체가 들어 있는 범용 버킷(또는 접두사를 포함한 버킷 경로)을 입력합니다. 목록에서 기존의 범용 버킷을 선택하려면 S3 찾아보기를 선택합니다.
5. 소스 객체 액세스 및 복사 권한의 경우 다음 중 하나를 수행하여 소스 객체를 가져오는 데 필요한 권한을 가진 IAM 역할을 지정합니다.
   - Amazon S3가 대신 새 IAM 역할을 생성하도록 허용하려면 새 IAM 역할 생성을 선택합니다.
   - 목록에서 기존 IAM 역할을 선택하려면 기존 IAM 역할에서 선택을 선택합니다.

- Amazon 리소스 이름(ARN)을 입력하여 기존 IAM 역할을 지정하려면 IAM 역할 ARN 입력을 선택한 다음 해당 필드에 ARN을 입력합니다.

6. 대상 및 복사된 객체 설정 섹션에 표시된 정보를 검토합니다. 대상 섹션의 정보가 정확하면 가져오기를 선택하여 복사 작업을 시작합니다.

   Amazon S3 콘솔은 배치 작업 페이지에 새 작업의 상태를 표시합니다. 작업에 대한 자세한 내용을 보려면 작업 이름 옆의 옵션 버튼을 선택한 다음 작업 메뉴에서 세부 정보 보기를 선택합니다. 객체를 가져올 대상 디렉터리 버킷을 열려면 가져오기 대상 보기를 선택합니다.

디렉터리 버킷의 S3 수명 주기 작업

S3 수명 주기는 사용자 대신 만료된 객체를 삭제하여 디렉터리 버킷의 S3 Express One Zone에 객체를 비용 효율적으로 저장하는 데 도움이 됩니다. 객체의 수명 주기를 관리하려면 디렉터리 버킷에 대한 S3 수명 주기 구성을 만듭니다. S3 수명 주기 구성은 Amazon S3이 객체 그룹에 적용하는 작업을 정의하는 일련의 규칙입니다. AWS Command Line Interface(AWS CLI), AWS SDK, Amazon S3 REST API 및 AWS CloudFormation을 사용하여 디렉터리 버킷에서 Amazon S3 수명 주기 구성을 설정할 수 있습니다.

수명 주기 구성에서 규칙을 사용하여 객체에 대해 Amazon S3가 수행할 작업을 정의합니다. 디렉터리 버킷에 저장된 객체의 경우 수명 주기 규칙을 생성하여 객체가 오래되면 만료할 수 있습니다. 또한 수명 주기 규칙을 생성하여 디렉터리 버킷에서 매일 불완전한 멀티파트 업로드를 삭제할 수 있습니다.

버킷에 수명 주기 구성을 추가할 경우 구성 규칙이 기존 객체는 물론 나중에 추가하는 객체에도 적용됩니다. 예를 들어 특정 접두사가 있는 객체를 생성한 후 30일이 경과하면 만료시키는 만료 작업을 포함하는 수명 주기 구성 규칙을 오늘 추가할 경우, S3는 30일 이상 경과하고 해당 접두사가 있는 모든 기존 객체를 삭제 대기열에 넣습니다.

디렉터리 버킷에 대한 S3 수명 주기의 차이점

디렉터리 버킷의 객체의 경우 객체를 만료시키고 불완전한 멀티파트 업로드를 삭제하는 수명 주기 규칙을 생성할 수 있습니다. 그러나 디렉터리 버킷의 S3 수명 주기는 스토리지 클래스 간 전환 작업을 지원하지 않습니다.

CreateSession

수명 주기는 퍼블릭 `DeleteObject` 및 `DeleteObjects` API 작업을 사용하여 디렉터리 버킷의 객체를 만료시킵니다. 해당 API 작업을 사용하기 위해 S3 수명 주기는 `CreateSession` API를 사용하여 임시 보안 자격 증명을 설정하고 디렉터리 버킷의 객체에 액세스합니다. 자세한 내용은 Amazon S3 API 참조의 `CreateSession` 섹션을 참조하세요.

수명 주기 보안 주체에 대한 삭제 권한을 거부하는 활성 정책이 있을 경우 S3 수명 주기가 사용자를 대신하여 객체 삭제를 차단합니다.

버킷 정책을 사용하여 S3 수명 주기 서비스 보안 주체에 권한 부여

다음 버킷 정책은 기본 `ReadWrite` 세션으로 CreateSession 직접 호출을 허용하고 수명 주기 서비스 보안 주체를 허용할 수 있는 권한을 부여합니다.

Example - 기본 **`ReadWrite`** 세션에서 **`CreateSession`** 직접 호출을 허용하는 버킷 정책

```
{
  "Version":"2008-10-17",
  "Statement":[
     {
        "Effect":"Allow",
        "Principal": {
           "Service":"lifecycle.s3.amazonaws.com"
         },
         "Action":"s3express:CreateSession",
         "Condition": {
            "StringEquals": {
               "s3express:SessionMode": "ReadWrite"
             }
          },
          "Resource":"arn:aws:s3express:us-east-2:412345678921:bucket/amzn-s3-demo-
bucket--use2-az2--x-s3"
       }
   ]
}
```

수명 주기 규칙 모니터링

디렉터리 버킷에 저장된 객체의 경우 S3 수명 주기는 AWS CloudTrail 관리 및 데이터 이벤트 로그를 생성합니다. 자세한 내용은 S3 Express One Zone에 대한 CloudTrail 로그 파일 예시를 참조하세요.

수명 주기 구성을 생성하고 S3 수명 주기 관련 문제를 해결하는 방법에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- 디렉터리 버킷에 대한 수명 주기 구성 생성 및 관리

- 디렉터리 버킷의 S3 수명 주기 문제 해결

디렉터리 버킷에 대한 수명 주기 구성 생성 및 관리

AWS Command Line Interface(AWS CLI), AWS SDK, REST API를 사용하여 디렉터리 버킷에 대한 수명 주기 구성을 만들 수 있습니다.

AWS CLI 사용

다음 AWS CLI 명령을 사용하여 S3 수명 주기 구성을 관리할 수 있습니다.

- `put-bucket-lifecycle-configuration`
- `get-bucket-lifecycle-configuration`
- `delete-bucket-lifecycle`

AWS CLI 설정에 대한 지침은 Amazon S3 API 참조의 AWS CLI를 사용하여 Amazon S3에서 개발을 참조하세요.

Amazon S3 수명 주기 구성은 XML 파일입니다. 그러나 AWS CLI를 사용할 때는 XML 형식을 지정할 수 없으며 대신 JSON 형식을 지정해야 합니다. 다음은 XML 수명 주기 구성과 AWS CLI 명령에서 지정할 수 있는 그에 상응하는 JSON 구성의 예시입니다.

다음 AWS CLI 예시에서는 수명 주기 구성 정책을 디렉터리 버킷에 적용합니다. 이 정책은 플래그가 지정된 접두사(myprefix)가 있고 정의된 개체 크기인 모든 개체가 7일 후에 만료되도록 지정합니다. 이 예제를 사용하려면 각 *`user input placeholder`*를 사용자의 정보로 대체합니다.

수명 주기 구성 정책을 JSON 파일로 저장합니다. 이 예에서는 lifecycle1.json으로 파일 이름을 지정합니다.

Example

JSON

```
{
    "Rules": [
        {
        "Expiration": {
            "Days": 7
        },
```

```
        "ID": "Lifecycle expiration rule",
        "Filter": {
            "And": {
                "Prefix": "myprefix/",
                "ObjectSizeGreaterThan": 500,
                "ObjectSizeLessThan": 64000
            }
        },
        "Status": "Enabled"
    }
    ]
}
```

put-bucket-lifecycle-configuration CLI 명령의 일부로 JSON 파일을 제출합니다. 이 명령을 사용하려면 각 *### ## ## ###*를 사용자의 정보로 대체합니다.

```
aws s3api put-bucket-lifecycle-configuration --region us-west-2 --profile default
 --bucket amzn-s3-demo-bucket--usw2-az1--x-s3 --lifecycle-configuration file://lc-
policy.json --checksum-algorithm crc32c
```

Example

XML

```
<LifecycleConfiguration>
    <Rule>
        <ID>Lifecycle expiration rule</ID>
        <Filter>
            <And>
                <Prefix>myprefix/</Prefix>
                <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
                <ObjectSizeLessThan>64000</ObjectSizeLessThan>
            </And>
        </Filter>
        <Status>Enabled</Status>
        <Expiration>
             <Days>7</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

AWS SDK 사용

SDK for Java

Example

```
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
import software.amazon.awssdk.services.s3.model.LifecycleRuleAndOperator;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleResponse;
import software.amazon.awssdk.services.s3.model.AbortIncompleteMultipartUpload;

// PUT a Lifecycle policy
LifecycleRuleFilter objectExpirationFilter =
 LifecycleRuleFilter.builder().and(LifecycleRuleAndOperator.builder().prefix("dir1/").object
LifecycleRuleFilter mpuExpirationFilter =
 LifecycleRuleFilter.builder().prefix("dir2/").build();

LifecycleRule objectExpirationRule =
 LifecycleRule.builder().id("lc").filter(objectExpirationFilter).status("Enabled").expiratio
                    .days(10)
                    .build())
                .build();
LifecycleRule mpuExpirationRule = LifecycleRule.builder().id("lc-
mpu").filter(mpuExpirationFilter).status("Enabled").abortIncompleteMultipartUpload(AbortInco
                        .daysAfterInitiation(10)
                        .build())
                .build();

PutBucketLifecycleConfigurationRequest putLifecycleRequest =
 PutBucketLifecycleConfigurationRequest.builder()
                .bucket("amzn-s3-demo-bucket--usw2-az1--x-s3")
```

```
                .checksumAlgorithm(ChecksumAlgorithm.CRC32)
                .lifecycleConfiguration(
                        BucketLifecycleConfiguration.builder()
                                .rules(objectExpirationRule, mpuExpirationRule)
                                .build()
                ).build();

PutBucketLifecycleConfigurationResponse resp =
 client.putBucketLifecycleConfiguration(putLifecycleRequest);

// GET the Lifecycle policy
GetBucketLifecycleConfigurationResponse getResp =
 client.getBucketLifecycleConfiguration(GetBucketLifecycleConfigurationRequest.builder().buc
s3-demo-bucket--usw2-az1--x-s3").build());

// DELETE the Lifecycle policy
DeleteBucketLifecycleResponse delResp =
 client.deleteBucketLifecycle(DeleteBucketLifecycleRequest.builder().bucket("amzn-
s3-demo-bucket--usw2-az1--x-s3").build());
```

SDK for Go

Example

```
package main

import (
    "context"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
)
// PUT a Lifecycle policy
func putBucketLifecycleConfiguration(client *s3.Client, bucketName string ) error {
    lifecycleConfig := &s3.PutBucketLifecycleConfigurationInput{
        Bucket: aws.String(bucketName),
        LifecycleConfiguration: &types.BucketLifecycleConfiguration{
            Rules: []types.LifecycleRule{
                {
                    ID:     aws.String("lc"),
                    Filter: &types.LifecycleRuleFilter{
```

```
                        And: &types.LifecycleRuleAndOperator{
                            Prefix: aws.String("foo/"),
                            ObjectSizeGreaterThan: aws.Int64(1000000),
                            ObjectSizeLessThan:    aws.Int64(100000000),
                            },
                        },

                    Status: types.ExpirationStatusEnabled,
                    Expiration: &types.LifecycleExpiration{
                        Days: aws.Int32(int32(1)),
                    },
                },
                {
                    ID:     aws.String("abortmpu"),
                    Filter: &types.LifecycleRuleFilter{
                        Prefix: aws.String("bar/"),
                    },
                    Status: types.ExpirationStatusEnabled,
                    AbortIncompleteMultipartUpload:
 &types.AbortIncompleteMultipartUpload{
                        DaysAfterInitiation: aws.Int32(int32(5)),
                    },
                },
            },
        },
    }
    _, err := client.PutBucketLifecycleConfiguration(context.Background(),
 lifecycleConfig)
    return err
}
// Get the Lifecycle policy
func getBucketLifecycleConfiguration(client *s3.Client, bucketName string) error {
    getLifecycleConfig := &s3.GetBucketLifecycleConfigurationInput{
        Bucket: aws.String(bucketName),
    }

    resp, err := client.GetBucketLifecycleConfiguration(context.Background(),
 getLifecycleConfig)
    if err != nil {
        return err
    }
    return nil
}
// Delete the Lifecycle policy
```

```
func deleteBucketLifecycleConfiguration(client *s3.Client, bucketName string) error
 {
    deleteLifecycleConfig := &s3.DeleteBucketLifecycleInput{
        Bucket: aws.String(bucketName),
    }
    _, err := client.DeleteBucketLifecycle(context.Background(),
 deleteLifecycleConfig)
    return err
}
func main() {
    cfg, err := config.LoadDefaultConfig(context.Background(),
 config.WithRegion("us-west-2")) // Specify your region here
    if err != nil {
        log.Fatalf("unable to load SDK config, %v", err)
    }
    s3Client := s3.NewFromConfig(cfg)
    bucketName := "amzn-s3-demo-bucket--usw2-az1--x-s3"
    putBucketLifecycleConfiguration(s3Client, bucketName)
    getBucketLifecycleConfiguration(s3Client, bucketName)
    deleteBucketLifecycleConfiguration(s3Client, bucketName)
    getBucketLifecycleConfiguration(s3Client, bucketName)
}
```

SDK for .NET

Example

```
using Amazon;
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 = "amzn-s3-demo-bucket--usw2-az1--x-s3";
        // 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 = "delete rule",
                                  Filter = new LifecycleFilter()
                                 {
                                     LifecycleFilterPredicate = new
LifecyclePrefixPredicate()
                                     {
                                         Prefix = "projectdocs/"
                                     }
                                 },
                                 Status = LifecycleRuleStatus.Enabled,
                                 Expiration = new LifecycleRuleExpiration()
                                 {
                                       Days = 10
                                 }
                            }
                        }
                };

                // 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 = "mpu abort rule",
```

```
                    Filter = new LifecycleFilter()
                    {
                        LifecycleFilterPredicate = new LifecyclePrefixPredicate()
                        {
                            Prefix = "YearlyDocuments/"
                        }
                    },
                    Expiration = new LifecycleRuleExpiration()
                    {
                        Days = 10
                    },
                    AbortIncompleteMultipartUpload = new
LifecycleRuleAbortIncompleteMultipartUpload()
                    {
                        DaysAfterInitiation = 10
                    }
                });

                // Add the configuration to the bucket.
                await AddExampleLifecycleConfigAsync(client,
lifeCycleConfiguration);

                // Verify that there are now two rules.
                lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);
                Console.WriteLine("Expected # of rulest=2; 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);
        }
    }
}
```

SDK for Python

Example

```
import boto3

client = boto3.client("s3", region_name="us-west-2")
bucket_name = 'amzn-s3-demo-bucket--usw2-az1--x-s3'

client.put_bucket_lifecycle_configuration(
    Bucket=bucket_name,
    ChecksumAlgorithm='CRC32',
    LifecycleConfiguration={
        'Rules': [
            {
                'ID': 'lc',
                'Filter': {
                    'And': {
                        'Prefix': 'foo/',
                        'ObjectSizeGreaterThan': 1000000,
                        'ObjectSizeLessThan': 100000000,
                    }
                },
                'Status': 'Enabled',
                'Expiration': {
                    'Days': 1
                }
            },
            {
                'ID': 'abortmpu',
                'Filter': {
                    'Prefix': 'bar/'
                },
                'Status': 'Enabled',
                'AbortIncompleteMultipartUpload': {
                    'DaysAfterInitiation': 5
                }
            }
        ]
    }
)

result = client.get_bucket_lifecycle_configuration(
    Bucket=bucket_name
```

```
)

client.delete_bucket_lifecycle(
    Bucket=bucket_name
)
```

디렉터리 버킷의 S3 수명 주기 문제 해결

주제

- 수명 주기 구성을 설정했지만 디렉터리 버킷의 객체가 만료되지 않음
- 수명 주기 규칙에 따라 취해진 조치를 모니터링하려면 어떻게 해야 하나요?

수명 주기 구성을 설정했지만 디렉터리 버킷의 객체가 만료되지 않음

디렉터리 버킷의 S3 수명 주기는 퍼블릭 API를 사용하여 S3 Express One Zone에서 객체를 삭제합니다. 객체 수준 퍼블릭 API를 사용하려면 `CreateSession`에 권한을 부여하고 객체를 삭제하는 S3 수명 주기 권한을 허용해야 합니다. 삭제를 거부하는 활성 정책이 있을 경우 S3 수명 주기가 사용자를 대신하여 객체 삭제를 차단합니다.

버킷 정책을 올바르게 구성하여 삭제하려는 객체가 만료될 수 있도록 해야 합니다. CloudTrail에서 `CreateSession` API 간접 호출을 위한 `AccessDenied` 추적용 AWS CloudTrail 로그를 확인하여 액세스가 거부되었는지 여부를 확인할 수 있습니다. CloudTrail 로그를 확인하면 액세스 문제를 해결하고 액세스 거부 오류의 근본 원인을 식별하는 데 도움이 될 수 있습니다. 그런 다음, 관련 정책을 업데이트하여 잘못된 액세스 제어를 수정할 수 있습니다.

버킷 정책이 올바르게 설정되어 있지만 문제가 지속될 경우 수명 주기 규칙을 검토하여 적절한 객체 하위 집합에 적용되는지 확인하는 것이 좋습니다.

수명 주기 규칙에 따라 취해진 조치를 모니터링하려면 어떻게 해야 하나요?

AWS CloudTrail 데이터 이벤트 로그를 사용하여 디렉터리 버킷에서 S3 수명 주기가 수행한 작업을 모니터링할 수 있습니다. 자세한 내용은 CloudTrail 로그 파일 예제를 참조하세요.

디렉터리 버킷에 배치 작업 사용

Amazon S3 배치 작업을 사용하여 S3 버킷에 저장된 객체에 대해 작업을 수행할 수 있습니다. S3 배치 작업에 대한 자세한 내용은 Amazon S3 객체에 대한 대규모 배치 작업 수행을 참조하세요.

다음 항목에서는 디렉터리 버킷의 S3 Express One Zone 스토리지 클래스에 대한 배치 작업 수행에 관해 설명합니다.

주제

- 디렉터리 버킷에 배치 작업 사용
- 주요 차이점

디렉터리 버킷에 배치 작업 사용

디렉터리 버킷에 저장된 객체에 대해 복사 작업과 AWS Lambda 함수 호출 작업을 수행할 수 있습니다. 복사를 사용하면 같은 유형의 버킷 간에 객체를 복사할 수 있습니다(예: 디렉터리 버킷에서 디렉터리 버킷으로). 디렉터리 버킷과 범용 버킷 간에 객체를 복사할 수도 있습니다. AWS Lambda 함수 호출을 사용하면 Lambda 함수를 사용하여 직접 정의한 코드로 디렉터리 버킷의 객체에 대한 작업을 수행할 수 있습니다.

객체 복사

동일한 버킷 유형 간에 또는 디렉터리 버킷과 범용 버킷 간에 객체를 복사할 수 있습니다. 디렉터리 버킷으로 복사할 때는 이 버킷 유형에 맞는 Amazon 리소스 이름(ARN) 형식을 사용해야 합니다. 디렉터리 버킷의 ARN 형식은 `arn:aws:s3express:`*`region`*`:`*`account-id`*`:bucket/`*`bucket-base-name`*`--x-s3`입니다.

> **Note**
>
> 소스 또는 대상 버킷이 AWS 로컬 영역에 있는 경우 다른 AWS 리전에 객체를 복사할 수 없습니다. 소스 버킷과 대상 버킷의 상위 AWS 리전이 동일해야 합니다. 소스 버킷과 대상 버킷은 서로 다른 버킷 위치 유형(가용 영역 또는 로컬 영역)일 수 있습니다.

S3 콘솔에서 가져오기 작업을 사용하여 디렉터리 버킷에 데이터를 채울 수도 있습니다. 가져오기는 범용 버킷에서 디렉터리 버킷으로 객체를 복사하는 배치 작업 건을 생성하는 간소화된 방법입니다. 범용 버킷에서 디렉터리 버킷으로의 가져오기 복사 작업의 경우 S3는 매니페스트를 자동으로 생성합니다. 자세한 내용은 디렉터리 버킷으로 객체 가져오기 및 매니페스트 지정을 참조하세요.

Lambda 함수 호출(**`LambdaInvoke`**)

배치 작업을 사용하여 디렉터리 버킷에서 작동하는 Lambda 함수를 호출하려면 특별한 요구 사항이 있습니다. 예를 들어, v2 JSON 호출 체계를 사용하여 Lambda 요청을 구조화하고 작업 생성 시

InvocationSchemaVersion 2.0을 지정해야 합니다. 자세한 내용은 AWS Lambda 함수 호출을 참조하세요.

주요 차이점

다음은 대량 작업을 사용하여 디렉터리 버킷에 저장된 오브젝트에 대한 대량 작업을 수행할 때 S3 Express One Zone 스토리지 클래스가 있는 경우의 주요 차이점 목록입니다.

- 디렉터리 버킷의 경우 SSE-S3 및 AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 사용한 서버 측 암호화가 지원됩니다. 디렉터리 버킷(소스 또는 대상)에서 고객 제공 키(SSE-C)를 사용한 서버 측 암호화를 사용하도록 지정하는 `CopyObject` 요청을 하면 응답에 HTTP `400 (Bad Request)` 오류가 반환됩니다.

  버킷의 기본 암호화는 원하는 암호화 구성을 사용하고 `CreateSession` 요청 또는 PUT 객체 요청에서 버킷 기본 암호화를 재정의하지 않는 것이 좋습니다. 그러면 새 객체가 원하는 암호화 설정으로 자동 암호화됩니다. 디렉터리 버킷의 암호화 재정의 동작 및 SSE-KMS를 사용하여 디렉터리 버킷의 새 객체 사본을 암호화하는 방법에 대한 자세한 내용은 Specifying server-side encryption with AWS KMS for new object uploads를 참조하세요.

  배치 운영의 Copy 작업을 통해 SSE-KMS로 암호화된 객체를 범용 버킷에서 디렉터리 버킷으로, 디렉터리 버킷에서 범용 버킷으로 또는 디렉터리 버킷 간에 복사하는 경우 S3 버킷 키는 지원되지 않습니다. 이 경우 Amazon S3는 KMS 암호화 개체에 대한 사본 요청이 있을 때마다 AWS KMS를 직접적으로 호출합니다. 디렉터리 버킷에서 SSE-KMS를 사용하는 방법에 대한 자세한 내용은 디렉터리 버킷의 기본 암호화 설정 및 모니터링 및 디렉터리 버킷에서 AWS KMS 키(SSE-KMS)를 사용한 서버 측 암호화 사용 섹션을 참조하세요.
- 디렉터리 버킷의 객체에는 태그를 지정할 수 없습니다. 빈 태그 세트만 지정할 수 있습니다. 기본적으로 배치 작업은 태그를 복사합니다. 태그가 있는 객체를 범용 버킷과 디렉터리 버킷 간에 복사하면 `501 (Not Implemented)` 응답이 표시됩니다.
- S3 Express One Zone은 업로드 또는 다운로드 중에 데이터를 검증하는 데 사용하는 체크섬 알고리즘을 선택할 수 있는 옵션을 제공합니다. SHA(보안 해시 알고리즘) 또는 CRC(순환 중복 검사) 데이터 무결성 확인 알고리즘(CRC32, SHA-1, SHA-256) 중 하나를 선택할 수 있습니다. MD5 기반 체크섬은 S3 Express One Zone 스토리지 클래스에서 지원되지 않습니다.
- 기본적으로 모든 Amazon S3 버킷은 객체 소유권 설정을 S3 버킷 소유자 적용으로 설정하며 액세스 제어 목록(ACL)이 비활성화되어 있습니다. 디렉터리 버킷의 경우 이 설정을 수정할 수 없습니다. 범용 버킷의 객체를 디렉터리 버킷으로 복사할 수 있습니다. 하지만 디렉터리 버킷으로 복사하거나 디렉터리 버킷에서 복사할 때는 기본 ACL을 덮어쓸 수 없습니다.

- 매니페스트를 지정하는 방법과 관계없이 목록 자체는 범용 버킷에 저장해야 합니다. 배치 작업은 디렉터리 버킷에서 기존 매니페스트를 가져오거나 생성된 매니페스트를 디렉터리 버킷에 저장할 수 없습니다. 하지만 매니페스트에 설명된 객체는 디렉터리 버킷에 저장할 수 있습니다.
- 배치 작업은 S3 인벤토리 보고서에서 디렉터리 버킷을 위치로 지정할 수 없습니다. 인벤토리 보고서는 디렉터리 버킷을 지원하지 않습니다. `ListObjectsV2` API 작업을 사용하여 객체를 나열하여 디렉터리 버킷 내의 객체에 대한 매니페스트 파일을 만들 수 있습니다. 그런 다음 CSV 파일로 목록을 삽입할 수 있습니다.

액세스 권한 부여

복사 작업을 수행하려면 다음 권한이 있어야 합니다.

- 한 디렉터리 버킷에서 다른 디렉터리 버킷으로 객체를 복사하려면 `s3express:CreateSession` 권한이 있어야 합니다.
- 디렉터리 버킷에서 범용 버킷으로 객체를 복사하려면 대상 버킷에 객체 사본을 쓸 수 있도록 `s3express:CreateSession` 권한과 `s3:PutObject` 권한이 있어야 합니다.
- 범용 버킷에서 디렉터리 버킷으로 객체를 복사하려면 복사되는 소스 객체를 읽을 수 있도록 `s3express:CreateSession` 권한과 `s3:GetObject` 권한이 있어야 합니다.

  자세한 내용은 Amazon Simple Storage Service API 참조에서 CopyObject를 참조하십시오.
- Lambda 함수를 호출하려면 Lambda 함수를 기반으로 리소스에 권한을 부여해야 합니다. 필요한 권한을 확인하려면 해당 API 권한을 확인하세요.

디렉터리 버킷의 객체에 데이터 추가

디렉터리 버킷의 S3 Express One Zone 스토리지 클래스에 저장된 기존 객체의 끝에 데이터를 추가할 수 있습니다. 일정 기간 동안 지속적으로 데이터를 작성하거나 객체에 작성하는 동안 객체를 읽어야 할 경우 객체에 데이터를 추가하는 기능을 사용하는 것이 좋습니다. 객체에 데이터를 추가하는 작업은 로그 파일에 새 로그 항목을 추가하거나 트랜스코딩 후 스트리밍될 때 비디오 파일에 새 비디오 세그먼트를 추가하는 등의 사용 사례에서 흔히 볼 수 있습니다. 객체에 데이터를 추가하면 기존에는 로컬 스토리지에 데이터를 결합한 후 최종 객체를 Amazon S3에 복사했던 애플리케이션을 간소화할 수 있습니다.

객체에 추가할 수 있는 데이터에 대한 최소 크기 요구 사항은 없습니다. 그러나 단일 요청에서 객체에 추가할 수 있는 데이터의 최대 크기는 5GB입니다. 이는 Amazon S3 API를 사용하여 데이터를 업로드할 경우의 최대 요청 크기와 한도가 같습니다.

각 추가 작업이 성공하면 객체의 일부를 생성하며 각 객체에는 최대 10,000개의 파트가 있을 수 있습니다. 다시 말해 최대 10,000번까지 객체에 데이터를 추가할 수 있습니다. S3 멀티파트 업로드를 사용하여 객체를 생성할 경우 업로드된 각 파트는 총 최대 10,000개의 파트에 포함됩니다. 예를 들어 파트 1,000개로 구성된 멀티파트 업로드로 생성된 객체에 최대 9,000번 추가할 수 있습니다.

> **Note**
>
> 파트 한도에 도달하면 TooManyParts 오류가 발생합니다. `CopyObject` API를 사용하여 횟수를 재설정할 수 있습니다.

파트를 객체에 병렬로 업로드하고 파트가 업로드되는 동안 파트를 읽을 필요가 없을 경우 Amazon S3 멀티파트 업로드를 사용하는 것이 좋습니다. 자세한 내용은 멀티파트 업로드 사용을 참조하세요.

객체에 대한 데이터 추가는 S3 Express One Zone 스토리지 클래스에 저장된 디렉터리 버킷의 객체에 대해서만 지원됩니다. S3 Express One Zone에 대한 자세한 내용은 S3 Express One Zone 시작하기를 참조하세요.

AWS SDK, AWS CLI 및 `PutObject` API를 사용하여 디렉터리 버킷의 객체에 데이터를 추가할 수 있습니다. `PutObject` 요청을 할 때 `x-amz-write-offset-bytes` 헤더를 추가할 객체의 크기로 설정합니다. `PutObject` API 작업을 사용하려면 `CreateSession` API를 사용하여 디렉터리 버킷의 객체에 액세스하기 위한 임시 보안 자격 증명을 설정해야 합니다. 자세한 내용은 Amazon S3 API Reference의 `PutObject` 및 `CreateSession` 섹션을 참조하세요.

성공적인 각 추가 작업은 `PutObject` 요청으로 청구됩니다. 자세한 가격 책정 정보는 `Amazon S3 pricing` 섹션을 참조하세요.

> **Note**
>
> 1.12 릴리스부터 Mountpoint for Amazon S3는 S3 Express One Zone에 저장된 객체에 데이터 추가를 지원합니다. 시작하려면 `--incremental-upload` 플래그를 설정하여 옵트인해야 합니다. Mountpoint에 대한 자세한 내용은 Mountpoint 작업을 참조하세요.

추가된 데이터를 업로드하는 동안 CRC(순환 중복 검사) 알고리즘을 사용할 경우 `HeadObject` 또는 `GetObject` 요청을 사용하여 전체 객체 CRC 기반 체크섬을 검색할 수 있습니다. 추가된 데이터를 업로드하는 동안 SHA-1 또는 SHA-256 알고리즘을 사용할 경우 추가된 파트의 체크섬을 검색하고 이전 PutObject 응답에서 반환된 SHA 체크섬을 사용하여 무결성을 확인할 수 있습니다. 자세한 내용은 데이터 보호 및 암호화를 참조하세요.

AWS CLI, AWS SDK 및 REST API를 사용하여 객체에 데이터 추가

AWS Command Line Interface(AWS CLI), AWS SDK 및 REST API를 사용하여 객체에 데이터를 추가할 수 있습니다.

AWS CLI 사용

다음 `put-object` 예시 명령은 AWS CLI를 사용하여 객체에서 데이터를 추가할 수 있는 방법을 보여줍니다. 이 명령을 실행하려면 *`user input placeholders`*를 사용자의 정보로 대체합니다.

```
aws s3api put-object --bucket amzn-s3-demo-bucket--azid--x-s3 --key sampleinput/
file001.bin --body bucket-seed/file001.bin --write-offset-bytes size-of-sampleinput/
file001.bin
```

AWS SDK 사용

SDK for Java

AWS SDK for Java를 사용하여 객체에 데이터를 추가할 수 있습니다.

```
var putObjectRequestBuilder = PutObjectRequest.builder()
                                              .key(key)
                                              .bucket(bucketName)
                                              .writeOffsetBytes(9);
var response = s3Client.putObject(putObjectRequestBuilder.build());
```

SDK for Python

```
s3.put_object(Bucket='amzn-s3-demo-bucket--use2-az2--x-s3', Key='2024-11-05-sdk-
test', Body=b'123456789', WriteOffsetBytes=9)
```

REST API 사용

객체에 데이터를 추가하기 위해 REST 요청을 보낼 수 있습니다. 자세한 내용은 PutObject 섹션을 참조하세요.

디렉터리 버킷의 객체 이름 바꾸기

`RenameObject` 작업을 사용하면 데이터 이동 없이 S3 Express One Zone 스토리지 클래스를 사용하는 디렉터리 버킷의 기존 객체 이름을 원자적으로 바꿀 수 있습니다. 기존 객체의 이름을 소스로 지정

하고 객체의 새 이름을 동일한 디렉터리 버킷 내의 대상으로 지정하여 객체의 이름을 바꿀 수 있습니다. 슬래시(/) 구분 기호 문자로 끝나는 객체에서는 RenameObject API 작업이 성공하지 못합니다. 자세한 내용은 Amazon S3 객체 이름 지정을 참조하세요.

RenameObject 작업은 일반적으로 객체 크기와 관계없이 몇 밀리초 내에 완료됩니다. 이 기능은 로그 파일 관리, 미디어 처리 및 데이터 분석과 같은 애플리케이션을 가속화합니다. 또한 RenameObject는 스토리지 클래스, 암호화 유형, 생성 날짜, 마지막 수정 날짜 및 체크섬 속성을 포함한 모든 객체 메타데이터 속성을 보존합니다.

> **Note**
>
> RenameObject는 S3 Express One Zone 스토리지 클래스에 저장된 객체에 대해서만 지원됩니다.

RenameObject 작업에 대한 액세스 권한을 부여하려면 세션 기반 권한 부여를 위해 CreateSession 작업을 사용하는 것이 좋습니다. 특히 버킷 정책 또는 ID 기반 정책에서 디렉터리 버킷에 s3express:CreateSession 권한을 부여합니다. 그런 다음, 디렉터리 버킷에서 CreateSession API를 직접 호출하여 세션 토큰을 가져옵니다. 요청 헤더에 세션 토큰을 포함하여 이 작업에 API 요청을 할 수 있습니다. 세션 토큰이 만료되면 다른 CreateSession API 직접 호출을 수행하여 사용할 새 세션 토큰을 생성합니다. AWS CLI 및 AWS SDK는 세션이 만료될 때 서비스 중단을 방지하기 위해 세션 토큰을 자동으로 새로 고치는 등 세션을 생성하고 관리합니다. 권한 부여에 대한 자세한 내용은 Amazon S3 API 참조의 CreateSession 섹션을 참조하세요. 영역 엔드포인트 API 작업에 대한 자세한 내용은 CreateSession을 사용하여 영역 엔드포인트 API 작업 권한 부여를 참조하세요.

기존 객체를 덮어쓰지 않으려면 RenameObject 요청에 '*' 값과 함께 If-None-Match 조건부 헤더를 추가하면 됩니다. 객체 이름이 이미 있는 경우 Amazon S3에서 412 Precondition Failed 오류를 반환합니다. 자세한 내용은 Amazon S3 API 참조의 RenameObject 섹션을 참조하세요.

RenameObject는 AWS CloudTrail에 로깅되는 영역 엔드포인트 API 작업(객체 수준 또는 데이터 플레인 작업)입니다. CloudTrail을 사용하여 디렉터리 버킷의 객체에 대해 수행된 RenameObject 작업에 대한 정보를 수집할 수 있습니다. 자세한 내용은 AWS CloudTrail로 디렉터리 버킷에 로그인 및 디렉터리 버킷에 대한 CloudTrail 로그 파일 예제를 참조하세요.

S3 Express One Zone은 RenameObject를 지원하는 유일한 스토리지 클래스이며, S3 Express One Zone에서의 PUT, COPY, POST 및 LIST 요청과 가격이 동일합니다(요청 1,000개당). 자세한 내용은 Amazon S3 요금을 참조하십시오.

객체 이름 바꾸기

디렉터리 버킷의 객체 이름을 바꾸려면 Amazon S3 콘솔, AWS CLI, AWS SDK, REST API 또는 Mountpoint for Amazon S3(버전 1.19.0 이상)를 사용하면 됩니다.

S3 콘솔 사용

디렉터리 버킷의 객체 이름을 바꾸는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 탐색 창에서 버킷을 선택하고 디렉터리 버킷 탭을 선택합니다. 이름을 바꿀 객체가 포함된 Amazon S3 디렉터리 버킷으로 이동합니다.
3. 이름을 바꾸려는 객체의 확인란을 선택합니다.
4. 작업 메뉴에서 객체 이름 바꾸기를 선택합니다.
5. 새 객체 이름 상자에 객체의 새 이름을 입력합니다.

> **Note**
>
> 기존 객체와 동일한 객체 이름을 지정하면 작업이 실패하고 Amazon S3가 412 `Precondition Failed` 오류를 반환합니다. 객체 키 이름 길이는 1,024바이트를 초과할 수 없습니다. 객체 이름에 포함된 접두사는 총 길이에 포함됩니다.

6. 객체 이름 바꾸기를 선택합니다. Amazon S3가 객체의 이름을 바꿉니다.

AWS CLI 사용

`rename-object` 예시에서는 AWS CLI를 사용하여 객체의 이름을 바꾸는 방법을 보여줍니다. 이러한 명령을 실행하려면 *### ## ## ###*를 실제 정보로 대체합니다.

다음 예시에서는 소스 객체의 ETag에 대한 조건부 검사를 사용하여 객체의 이름을 바꾸는 방법을 보여줍니다.

```
aws s3api rename-object \
    --bucket amzn-s3-demo-bucket--usw2-az1--x-s3 \
    --key new-file.txt \
    --rename-source amzn-s3-demo-bucket--usw2-az1--x-s3/original-file.txt \
    --source-if-match "\"a1b7c3d2e5f6\""
```

이 명령은 다음 작업을 수행합니다.

- *amzn-s3-demo-bucket--usw2-az1--x-s3* 디렉터리 버킷의 객체 이름을 *original-file.txt*에서 *new-file.txt*로 바꿉니다.
- 소스 객체의 ETag가 '*a1b7c3d4e5f6*'과 일치하는 경우에만 이름 변경을 수행합니다.

ETag가 일치하지 않으면 412 Precondition Failed 오류와 함께 작업이 실패합니다.

다음 예시에서는 새로 지정된 객체 이름에 대한 조건부 검사를 통해 객체의 이름을 바꾸는 방법을 보여줍니다.

```
aws s3api rename-object \
    --bucket amzn-s3-demo-bucket--usw2-az1--x-s3 \
    --key new-file.txt \
    --rename-source amzn-s3-demo-bucket--usw2-az1--x-s3/original-file.txt \
    --destination-if-none-match "\"e5f3g7h8i9j0\""
```

이 명령은 다음 작업을 수행합니다.

- *amzn-s3-demo-bucket--usw2-az1--x-s3* 디렉터리 버킷의 객체 이름을 *original-file.txt*에서 *new-file.txt*로 바꿉니다.
- 객체가 존재하고 객체의 ETag가 '*e5f3g7h8i9j0*'과 일치하지 않는 경우에만 이름 바꾸기 작업을 수행합니다.

새 이름이 지정되어 있고 ETag가 일치하는 객체가 이미 있는 경우 412 Precondition Failed 오류와 함께 작업이 실패합니다.

AWS SDK 사용

SDK for Java

AWS SDK for Java를 사용하여 객체의 이름을 바꿀 수 있습니다. 이러한 예시를 사용하려면 *### # # ## ###*를 실제 정보로 대체하세요.

다음 예시는 AWS SDK for Java를 사용하여 RenameObjectRequest를 생성하는 방법을 보여줍니다.

```
String key = "key";
```

```
String newKey = "new-key";
String expectedETag = "e5f3g7h8i9j0";
RenameObjectRequest renameRequest = RenameObjectRequest.builder()
    .bucket(amzn-s3-demo-bucket--usw2-az1--x-s3)
    .key(newKey)
    .renameSource(key)
    .destinationIfMatch(e5f3g7h8i9j0)
    .build();
```

이 코드는 다음을 수행합니다.

- *amzn-s3-demo-bucket--usw2-az1--x-s3* 디렉터리 버킷에서 객체 이름을 '*key*'에서 '*new-key*'로 바꾸는 요청을 생성합니다.
- 객체의 ETag가 '*e5f3g7h8i9j0*'과 일치하는 경우에만 이름 바꾸기가 발생한다는 조건을 포함합니다.
- ETag가 일치하지 않거나 객체가 존재하지 않으면 작업이 실패합니다.

다음 예시에서는 AWS SDK for Java를 사용하여 일치하지 않는 조건으로 `RenameObjectRequest`를 생성하는 방법을 보여줍니다.

```
String key = "key";
String newKey = "new-key";
String noneMatchETag = "e5f3g7h8i9j0";
RenameObjectRequest renameRequest = RenameObjectRequest.builder()
    .bucket(amzn-s3-demo-bucket--usw2-az1--x-s3)
    .key(newKey)
    .renameSource(key)
    .destinationIfNoneMatch(noneMatchETag)
    .build();
```

이 코드는 다음을 수행합니다.

- *amzn-s3-demo-bucket--usw2-az1--x-s3* 디렉터리 버킷에서 객체 이름을 '*key*'에서 '*new-key*'로 바꾸는 요청을 생성합니다.
- 대상 객체의 ETag가 '*e5f3g7h8i9j0*'과 일치하지 않는 경우에만 이름 바꾸기가 발생하도록 하는 `.destinationIfNoneMatch(noneMatchETag)`를 사용하는 조건을 포함합니다.

새 이름이 지정되어 있고 ETag가 지정된 객체가 이미 있는 경우 `412 Precondition Failed` 오류와 함께 작업이 실패합니다.

SDK for Python

SDK for Python을 사용하여 객체의 이름을 바꿀 수 있습니다. 이러한 예시를 사용하려면 *### ### ### ###*를 실제 정보로 대체하세요.

다음 예시에서는 AWS SDK for Python(Boto3)을 사용하여 객체의 이름을 바꾸는 방법을 보여줍니다.

```
def basic_rename(bucket, source_key, destination_key):
    try:
        s3.rename_object(
            Bucket=amzn-s3-demo-bucket--usw2-az1--x-s3,
            Key=destination_key,
            RenameSource=f"{amzn-s3-demo-bucket--usw2-az1--x-s3}/{source_key}"
        )
        print(f"Successfully renamed {source_key} to {destination_key}")
    except ClientError as e:
        print(f"Error renaming object: {e}")
```

이 코드는 다음을 수행합니다.

- *amzn-s3-demo-bucket--usw2-az1--x-s3* 디렉터리 버킷의 *source_key*에서 *destination_key*로 객체의 이름을 바꿉니다.
- 객체 이름 바꾸기에 성공하면 성공 메시지를 인쇄하고 실패하면 오류 메시지를 인쇄합니다.

다음 예시에서는 AWS SDK for Python(Boto3)을 사용하여 `SourceIfMatch` 및 `DestinationIfNoneMatch` 조건을 사용해 객체의 이름을 바꾸는 방법을 보여줍니다.

```
def rename_with_conditions(bucket, source_key, destination_key, source_etag,
 dest_etag):
    try:
        s3.rename_object(
            Bucket=amzn-s3-demo-bucket--usw2-az1--x-s3,
            Key=destination_key,
            RenameSource=f"{amzn-s3-demo-bucket--usw2-az1--x-s3}/{source_key}",
            SourceIfMatch=source_ETag,
            DestinationIfNoneMatch=dest_ETag
        )
        print(f"Successfully renamed {source_key} to {destination_key} with
 conditions")
    except ClientError as e:
```

```
        print(f"Error renaming object: {e}")
```

이 코드는 다음을 수행합니다.

- 조건부 이름 바꾸기 작업을 수행하고 SourceIfMatch 및 DestinationIfNoneMatch라는 두 가지 조건을 적용합니다. 이러한 조건을 조합하면 객체가 수정되지 않고 객체가 지정된 새 이름으로 아직 존재하지 않게 됩니다.
- *amzn-s3-demo-bucket--usw2-az1--x-s3* 디렉터리 버킷의 *source_key*에서 *destination_key*로 객체의 이름을 바꿉니다.
- 객체 이름 바꾸기에 성공하면 성공 메시지를 인쇄하고 실패하거나 조건이 충족되지 않으면 오류 메시지를 인쇄합니다.

SDK for Rust

SDK for Rust를 사용하여 객체의 이름을 바꿀 수 있습니다. 이러한 예시를 사용하려면 *### ## ## ###*를 실제 정보로 대체하세요.

다음 예시에서는 SDK for Rust를 사용하여 *amzn-s3-demo-bucket--usw2-az1--x-s3* 디렉터리 버킷의 객체 이름을 바꾸는 방법을 보여줍니다.

```
async fn basic_rename_example(client: &Client) -> Result<(), Box<dyn Error>> {
    let response = client
        .rename_object()
        .bucket(" amzn-s3-demo-bucket--usw2-az1--x-s3")
        .key("new-name.txt")  // New name/path for the object
        .rename_source("old-name.txt")  // Original object name/path
        .send()
        .await?;
    Ok(())
}
```

이 코드는 다음을 수행합니다.

- *amzn-s3-demo-bucket--usw2-az1--x-s3* 디렉터리 버킷에서 객체 이름을 '*old-name.tx*'에서 '*new-name.txt*'로 바꾸는 요청을 생성합니다.
- 잠재적 오류를 처리할 Result 유형을 반환합니다.

REST API 사용

객체 이름을 바꾸기 위해 REST 요청을 보낼 수 있습니다. 자세한 내용은 Amazon S3 API 참조의 `RenameObject` 섹션을 참조하세요.

Mountpoint for Amazon S3 사용

1.19.0 버전부터 Mountpoint for Amazon S3는 S3 Express One Zone에서 객체 이름 바꾸기를 지원합니다. Mountpoint에 대한 자세한 내용은 Mountpoint 작업을 참조하세요.

디렉터리 버킷에 객체 업로드

Amazon S3 디렉터리 버킷을 생성하면 여기에 객체를 업로드할 수 있습니다. 다음 예제에서는 S3 콘솔 및 AWS SDK를 사용하여 객체를 디렉터리 버킷에 업로드하는 방법을 보여줍니다. S3 Express One Zone을 사용한 대량 객체 작업에 대한 자세한 내용은 객체 관리 섹션을 참조하세요.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 디렉터리 버킷을 선택합니다.
3. 파일 및 폴더를 업로드하려는 버킷의 이름을 선택합니다.
4. 객체 탭에서 업로드를 선택합니다.
5. 업로드 페이지에서 다음 중 하나를 수행합니다.
   - 파일 및 폴더를 점선으로 표시된 업로드 영역으로 끌어다 놓습니다.
   - 파일 추가 또는 폴더 추가를 선택하고 업로드할 파일 또는 폴더를 선택한 후 열기 또는 업로드를 선택합니다.
6. 체크섬에서 사용하려는 체크섬 함수를 선택합니다.

   (선택 사항) 크기가 16MB 미만인 단일 객체를 업로드하는 경우 미리 계산된 체크섬 값을 지정할 수도 있습니다. 미리 계산된 값을 제공하면 Amazon S3는 선택한 체크섬 함수를 사용하여 계산한 값과 비교합니다. 값이 일치하지 않으면 업로드가 시작되지 않습니다.
7. 권한 및 속성 섹션의 옵션은 자동으로 기본 설정으로 설정되며 수정할 수 없습니다. 퍼블릭 액세스 차단은 자동으로 활성화되며 디렉터리 버킷에 대해서는 S3 버전 관리 및 S3 객체 잠금을 활성화할 수 없습니다.

   (선택 사항) 키-값 쌍의 메타데이터를 객체에 추가하려면 속성 섹션을 확장한 다음 메타데이터 섹션에서 메타데이터 추가를 선택합니다.

8. 나열된 파일을 업로드하려면 업로드를 선택합니다.

   Amazon S3가 객체와 폴더를 업로드합니다. 업로드가 완료되면 업로드: 상태 페이지에서 성공 메시지를 볼 수 있습니다.

AWS SDK 사용

SDK for Java 2.x

Example

```
public static void putObject(S3Client s3Client, String bucketName, String objectKey,
 Path filePath) {
       //Using File Path to avoid loading the whole file into memory
       try {
           PutObjectRequest putObj = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(objectKey)
                   //.metadata(metadata)
                   .build();
           s3Client.putObject(putObj, filePath);
           System.out.println("Successfully placed " + objectKey +" into bucket
 "+bucketName);

       }

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

SDK for Python

Example

```
import boto3
import botocore
from botocore.exceptions import ClientError
```

```
def put_object(s3_client, bucket_name, key_name, object_bytes):
    """
    Upload data to a directory bucket.
    :param s3_client: The boto3 S3 client
    :param bucket_name: The bucket that will contain the object
    :param key_name: The key of the object to be uploaded
    :param object_bytes: The data to upload
    """
    try:
        response = s3_client.put_object(Bucket=bucket_name, Key=key_name,
                                        Body=object_bytes)
        print(f"Upload object '{key_name}' to bucket '{bucket_name}'.")
        return response
    except ClientError:
        print(f"Couldn't upload object '{key_name}' to bucket '{bucket_name}'.")
        raise

def main():
    # Share the client session with functions and objects to benefit from S3 Express
 One Zone auth key
    s3_client = boto3.client('s3')
    # Directory bucket name must end with --zone-id--x-s3
    resp = put_object(s3_client, 'doc-bucket-example--use1-az5--x-s3', 'sample.txt',
 b'Hello, World!')
    print(resp)

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

AWS CLI 사용

다음 `put-object` 예시 명령은 AWS CLI를 사용하여 Amazon S3의 객체를 업로드하는 방법을 보여 줍니다. 이 명령을 실행하려면 *`user input placeholders`*를 사용자의 정보로 대체합니다.

```
aws s3api put-object --bucket bucket-base-name--zone-id--x-s3 --key sampleinut/
file001.bin --body bucket-seed/file001.bin
```

자세한 정보는 AWS CLI 명령 참조의 put-object 섹션을 참조하세요.

주제

- 디렉터리 버킷에 멀티파트 업로드 사용

디렉터리 버킷에 멀티파트 업로드 사용

멀티파트 업로드 프로세스를 사용하면 단일 객체를 여러 부분의 집합으로 업로드할 수 있습니다. 각 부분은 객체 데이터의 연속적인 부분입니다. 이러한 객체 부분은 독립적으로 그리고 임의의 순서로 업로드할 수 있습니다. 부분의 전송이 실패할 경우 다른 부분에 영향을 주지 않고도 해당 부분을 재전송할 수 있습니다. 객체의 모든 부분이 업로드되면 Amazon S3가 이들 부분을 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 개선 - 부분을 병렬적으로 업로드하여 처리량을 개선할 수 있습니다.
- 네트워크 문제로부터 빠른 복구 - 더 작아진 부분 크기는 네트워크 오류로 인해 실패한 업로드 재시작의 영향을 최소화합니다.
- 객체 업로드 일시 중지 및 재개 – 객체 부분을 장시간에 걸쳐 업로드할 수 있습니다. 멀티파트 업로드가 시작되면 만료 날짜가 없습니다. 멀티파트 업로드를 명시적으로 완료하거나 중단해야 합니다.
- 최종 객체 크기를 알기 전에 업로드를 시작 – 객체를 생성하는 동안 업로드할 수 있습니다.

다음 방법으로 멀티파트 업로드를 사용하는 것이 좋습니다.

- 안정적인 높은 대역폭 네트워크를 통해 큰 객체를 업로드하는 경우, 멀티파트 업로드를 사용하여 멀티 스레드 성능을 위해 여러 객체 부분을 동시에 업로드함으로써 대역폭 사용을 극대화합니다.
- 불규칙한 네트워크를 통해 업로드하는 경우, 멀티파트 업로드를 사용하여 업로드가 다시 시작되는 것을 방지하여 네트워크 오류에 대한 복원력을 높입니다. 멀티파트 업로드를 사용하는 경우, 업로드 중에 중단된 부분만 다시 업로드해야 합니다. 객체 업로드를 처음부터 다시 시작하지 않아도 됩니다.

멀티파트 업로드를 사용하여 디렉터리 버킷의 Amazon S3 Express One Zone 스토리지 클래스에 객체를 업로드하는 경우, 멀티파트 업로드 프로세스는 멀티파트 업로드를 사용하여 범용 버킷에 객체를 업로드하는 프로세스와 유사합니다. 하지만 몇 가지 큰 차이가 있습니다.

멀티파트 업로드를 사용하여 S3 Express One Zone에 객체를 업로드하는 방법에 대한 자세한 정보는 다음 주제를 참조하세요.

주제

- 멀티파트 업로드 프로세스
- 멀티파트 업로드 작업을 사용한 체크섬
- 동시 멀티파트 업로드 작업

- 멀티파트 업로드 및 요금
- 멀티파트 업로드 API 작업 및 권한
- 예시

멀티파트 업로드 프로세스

멀티파트 업로드는 다음과 같은 3단계 프로세스입니다.

- 업로드를 시작합니다.
- 객체 부분을 업로드합니다.
- 부분을 모두 업로드한 후 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.

멀티파트 업로드 시작

멀티파트 업로드 시작 요청을 전송하면 Amazon S3는 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.

파트 업로드

부분을 업로드할 때 업로드 ID와 함께 부분 번호를 지정해야 합니다. S3 Express One Zone에서 멀티파트 업로드를 사용하는 경우 멀티파트의 부분 번호는 연속된 번호여야 합니다. 연속되지 않는 부분 번호로 멀티파트 업로드 요청을 완료하려고 하면 `HTTP 400 Bad Request`(유효하지 않은 부분 순서) 오류가 생성됩니다.

부분 번호를 사용하여 업로드하는 객체에서 각 부분과 그 위치를 고유하게 식별합니다. 이전에 업로드한 부분과 동일한 부분 번호로 새 부분을 업로드할 경우 이전에 업로드한 부분을 덮어쓰게 됩니다.

부분을 업로드할 때마다 Amazon S3는 그 응답으로 엔터티 태그(ETag) 헤더를 반환합니다. 각 부분 업로드에 대해 부분 번호와 ETag 값을 기록해야 합니다. 모든 객체 파트 업로드의 ETag 값은 동일하게 유지되지만 각 파트에는 다른 파트 번호가 할당됩니다. 이후 멀티파트 업로드를 완료하기 위한 요청에 이러한 값을 포함해야 하기 때문입니다.

Amazon S3는 S3 버킷에 업로드되는 모든 새 객체를 자동으로 암호화합니다. 멀티파트 업로드를 수행할 때 요청에서 암호화 정보를 지정하지 않은 경우 업로드된 파트의 암호화 설정이 대상 버킷의 기본 암호화 구성으로 설정됩니다. Amazon S3 버킷의 기본 암호화 구성은 항상 활성화되어 있으며, 최소

한 Amazon S3 관리형 키를 통한 서버 측 암호화(SSE-S3)로 설정되어 있습니다. 디렉터리 버킷의 경우 SSE-S3 및 AWS KMS 키를 통한 서버 측 암호화(SSE-KMS)가 지원됩니다. 자세한 내용은 데이터 보호 및 암호화 섹션을 참조하세요.

멀티파트 업로드 완료

멀티파트 업로드를 완료하면 Amazon S3는 부분 번호를 바탕으로 오름차순으로 각 부분을 결합하여 객체를 완성합니다. 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 ETag는 객체 데이터의 MD5 해시가 아닙니다.

멀티파트 업로드 나열

특정 멀티파트 업로드 또는 진행 중인 모든 멀티파트 업로드에 대해 부분 목록을 확인할 수 있습니다. 부분 목록 조회 작업은 특정 멀티파트 업로드에 대해 업로드한 부분의 정보를 반환합니다. 각 부분 목록 조회 요청에 대해 Amazon S3는 특정 멀티파트 업로드에서 최대 1,000개의 부분에 대해 부분 정보를 반환합니다. 멀티파트 업로드에서 파트가 1,000개 이상일 경우 모든 파트를 검색하려면 페이지 매김을 사용해야 합니다.

반환된 부분 목록에는 업로드가 완료되지 않은 부분이 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 사용하면 안 됩니다. 그 대신, 부분을 업로드할 때 지정한 부분 번호 및 Amazon S3가 반환한 해당 ETag 값의 목록을 보관해야 합니다.

멀티파트 업로드 목록에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 ListParts를 참조하세요.

멀티파트 업로드 작업을 사용한 체크섬

객체를 업로드할 때 객체 무결성을 검사하기 위해 체크섬 알고리즘을 지정할 수 있습니다. 디렉터리 버킷에는 MD5가 지원되지 않습니다. 다음 SHA(보안 해시 알고리즘) 또는 CRC(순환 중복 검사) 데이터 무결성 확인 알고리즘 중 하나를 지정할 수 있습니다.

- 2013년 8월 9일
- CRC32C

- SHA-1
- SHA-256

Amazon S3 REST API 또는 AWS를 사용하여 `GetObject` 또는 `HeadObject`를 통해 개별 부분의 체크섬 값을 검색할 수 있습니다. 아직 진행 중인 멀티파트 업로드의 개별 부분에 대한 체크섬 값을 검색하려면 `ListParts`를 사용하면 됩니다.

> **Important**
>
> 이전의 체크섬 알고리즘을 사용하는 경우 멀티파트 번호에 연속된 파트 번호를 사용해야 합니다. 연속되지 않는 부분 번호로 멀티파트 업로드 요청을 완료하려고 하면 Amazon S3에서 `HTTP 400 Bad Request`(유효하지 않은 부분 순서) 오류가 생성됩니다.

멀티파트 업로드 객체를 통한 체크섬 작업에 대한 자세한 내용은 Amazon S3에서 객체 무결성 확인 섹션을 참조하세요.

동시 멀티파트 업로드 작업

분산 개발 환경에서는 애플리케이션에서 한 객체에 대해 동시에 여러 업데이트를 시작할 수 있습니다. 예를 들어 애플리케이션은 동일한 객체 키를 사용하여 여러 멀티파트 업로드를 시작할 수 있습니다. 이러한 각 업로드에 대해 애플리케이션은 부분을 업로드한 후 Amazon S3가 객체를 생성하도록 업로드 완료 요청을 전송할 수 있습니다. S3 Express One Zone의 경우 객체 생성 시간은 멀티파트 업로드 완료 날짜입니다.

> **Important**
>
> 디렉터리 버킷에 저장된 객체에는 버전 관리가 지원되지 않습니다.

멀티파트 업로드 및 요금

멀티파트 업로드가 시작되면 Amazon S3는 업로드가 완료되거나 중단될 때까지 모든 파트를 계속 유지합니다. 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 파트를 삭제하므로 더 이상 비용이 청구되지 않습니다. 지정한 스토리지 클래스에 관계없이 불완전한 멀티파트 업로드를 삭제하는 경우 조기 삭제 요금이 부과되지 않습니다. 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

> **Important**
>
> 전체 멀티파트 업로드 요청이 성공적으로 전송되지 않으면 객체 파트가 어셈블되지 않고 객체가 생성되지 않습니다. 업로드된 부분에 연결된 모든 스토리지에 대해 요금이 청구됩니다. 멀티파트 업로드를 완료하여 객체를 생성하거나 멀티파트 업로드를 중지하여 업로드된 파트를 제거하는 것이 중요합니다.
> 디렉터리 버킷을 삭제하기 전에 진행 중인 모든 멀티파트 업로드를 완료하거나 중단해야 합니다. 디렉터리 버킷은 S3 수명 주기 구성을 지원하지 않습니다. 필요한 경우 활성 멀티파트 업로드를 나열한 다음 업로드를 중단한 후 버킷을 삭제할 수 있습니다.

멀티파트 업로드 API 작업 및 권한

디렉터리 버킷의 객체 관리 API 작업에 대한 액세스를 허용하려면 버킷 정책 또는 AWS Identity and Access Management(IAM) ID 기반 정책에서 디렉터리 버킷에 `s3express:CreateSession` 권한을 부여합니다.

멀티파트 업로드 작업을 수행하려면 필수 권한이 있어야 합니다. 버킷 정책 또는 IAM ID 기반 정책을 사용하여 IAM 보안 주체에 이러한 작업을 수행할 권한을 부여할 수 있습니다. 다음 테이블에는 여러 멀티파트 업로드 작업에 대한 필수 권한이 나와 있습니다.

`Initiator` 요소를 통해 멀티파트 업로드의 이니시에이터를 식별할 수 있습니다. 이니시에이터가 AWS 계정일 경우 이 요소는 `Owner` 요소와 동일한 정보를 제공합니다. 시작한 사용자가 IAM 사용자일 경우 이 요소는 사용자 ARN 및 표시 이름을 제공합니다.

작업	필수 권한
멀티파트 업로드 생성	멀티파트 업로드를 생성하려면 디렉터리 버킷에 대해 `s3express:CreateSession` 작업을 수행할 수 있어야 합니다.
멀티파트 업로드 시작	멀티파트 업로드를 시작하려면 디렉터리 버킷에 대해 `s3express:CreateSession` 작업을 수행할 수 있어야 합니다.
부분 업로드	부분을 업로드하려면 디렉터리 버킷에 대해 `s3express:CreateSession` 작업을 수행할 수 있어야 합니다. 이니시에이터가 부분을 업로드하려면 버킷 소유자는 이니시에이터가 디렉터리 버킷에 대한 `s3express:CreateSession` 작업을 수행하도록 허용해야 합니다.

작업	필수 권한
부분 업로드 (복사)	부분을 업로드하려면 디렉터리 버킷에 대해 `s3express:CreateSession` 작업을 수행할 수 있어야 합니다. 버킷 소유자는 객체의 멀티파트 업로드를 시작하는 사용자에게 `s3express:CreateSession` 작업을 수행할 수 있는 권한을 부여해야 합니다.
멀티파트 업로드 완료	멀티파트 업로드를 완료하려면 디렉터리 버킷에 대해 `s3express:CreateSession` 작업을 수행할 수 있어야 합니다. 버킷 소유자는 이니시에이터가 멀티파트 업로드를 완료할 수 있도록 객체에 대해 `s3express:CreateSession` 작업을 수행할 수 있는 권한을 부여해야 합니다.
멀티파트 업로드 중단	멀티파트 업로드를 중단하려면 객체에 대해 `s3express:CreateSession` 작업을 수행할 수 있어야 합니다. 멀티파트 업로드를 중단하려면 이니시에이터에게 `s3express:CreateSession` 작업을 수행할 수 있는 명시적 허용 액세스 권한을 부여해야 합니다.
부분 나열	멀티파트 업로드의 부분을 나열하려면 디렉터리 버킷에 대해 `s3express:CreateSession` 작업을 수행할 수 있어야 합니다.
진행 중인 멀티파트 업로드 나열	버킷에 대해 진행 중인 멀티파트 업로드를 나열하려면 버킷에 대해 `s3:ListBucketMultipartUploads` 작업을 수행할 수 있어야 합니다.

멀티파트 업로드를 위한 API 작업 지원

Amazon Simple Storage Service API 참조의 다음 섹션에서는 멀티파트 업로드를 위한 Amazon S3 REST API 작업에 대해 설명합니다.

- CreateMultipartUpload
- UploadPart
- UploadPartCopy
- CompleteMultipartUpload
- AbortMultipartUpload

- ListParts
- ListMultipartUploads

예시

멀티파트 업로드를 사용하여 디렉터리 버킷의 S3 Express One Zone에 객체를 업로드하려면 다음 예제를 참조하세요.

주제

- 멀티파트 업로드 생성
- 멀티파트 업로드의 부분 업로드
- 멀티파트 업로드 완료
- 멀티파트 업로드 중단
- 멀티파트 업로드 복사 작업 생성
- 진행 중인 멀티파트 업로드 나열
- 멀티파트 업로드의 부분 나열

멀티파트 업로드 생성

> **Note**
>
> 디렉터리 버킷의 경우 `CreateMultipartUpload` 작업 및 `UploadPartCopy` 작업을 수행할 때 버킷의 기본 암호화는 원하는 암호화 구성을 사용해야 하며, `CreateMultipartUpload` 요청에 제공하는 요청 헤더는 대상 버킷의 기본 암호화 구성과 일치해야 합니다.

다음 예시는 멀티파트 업로드를 만드는 방법을 보여 줍니다.

AWS SDK 사용

SDK for Java 2.x

Example

```
/**
 * This method creates a multipart upload request that generates a unique upload ID
 that is used to track
```

```
 * all the upload parts
 *
 * @param s3
 * @param bucketName - for example, 'doc-example-bucket--use1-az4--x-s3'
 * @param key
 * @return
 */
private static String createMultipartUpload(S3Client s3, String bucketName, String
key) {

    CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

    String uploadId = null;

    try {
        CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
        uploadId = response.uploadId();
    }
    catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return uploadId;
```

SDK for Python

Example

```
def create_multipart_upload(s3_client, bucket_name, key_name):
    '''
   Create a multipart upload to a directory bucket

   :param s3_client: boto3 S3 client
   :param bucket_name: The destination bucket for the multipart upload
   :param key_name: The key name for the object to be uploaded
   :return: The UploadId for the multipart upload if created successfully, else None
   '''
```

```
  try:
       mpu = s3_client.create_multipart_upload(Bucket = bucket_name, Key =
key_name)
       return mpu['UploadId']
   except ClientError as e:
       logging.error(e)
       return None
```

AWS CLI 사용

이 예시는 AWS CLI를 사용하여 디렉터리 버킷에 대한 멀티파트 업로드를 생성하는 방법을 보여 줍니다. 이 명령은 *KEY_NAME* 객체에 대해 *bucket-base-name*--*zone-id*--x-s3 디렉터리 버킷에 멀티파트 업로드를 시작합니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

```
aws s3api create-multipart-upload --bucket bucket-base-name--zone-id--x-s3 --
key KEY_NAME
```

자세한 내용은 AWS Command Line Interface의 create-multipart-upload를 참조하세요.

멀티파트 업로드의 부분 업로드

다음 코드 예시는 멀티파트 업로드의 일부를 업로드하는 방법을 보여 줍니다.

AWS SDK 사용

SDK for Java 2.x

다음 예시는 SDK for Java 2.x를 사용하여 단일 객체를 여러 부분으로 나눈 다음 디렉터리 버킷에 업로드하는 방법을 보여 줍니다.

Example

```
/**
 * This method creates part requests and uploads individual parts to S3 and then
 returns all the completed parts
 *
 * @param s3
 * @param bucketName
 * @param key
 * @param uploadId
 * @throws IOException
 */
```

```
private static ListCompletedPartmultipartUpload(S3Client s3, String bucketName,
String key, String uploadId, String filePath) throws IOException {

        int partNumber = 1;
        ListCompletedPart completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        // read the local file, breakdown into chunks and process
        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            int position = 0;
            while (position < fileSize) {
                file.seek(position);
                int read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .partNumber(partNumber)
                        .build();

                UploadPartResponse partResponse = s3.uploadPart(
                        uploadPartRequest,
                        RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNumber)
                        .eTag(partResponse.eTag())
                        .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        }

        catch (IOException e) {
            throw e;
        }
        return completedParts;
```

```
    }
```

SDK for Python

다음 예시는 SDK for Python을 사용하여 단일 객체를 여러 부분으로 나눈 다음 디렉터리 버킷에 업로드하는 방법을 보여 줍니다.

Example

```
def multipart_upload(s3_client, bucket_name, key_name, mpu_id, part_size):
    '''
    Break up a file into multiple parts and upload those parts to a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Destination bucket for the multipart upload
    :param key_name: Key name for object to be uploaded and for the local file
 that's being uploaded
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_size: The size parts that the object will be broken into, in bytes.
                      Minimum 5 MiB, Maximum 5 GiB. There is no minimum size for the
 last part of your multipart upload.
    :return: part_list for the multipart upload if all parts are uploaded
 successfully, else None
    '''

    part_list = []
    try:
        with open(key_name, 'rb') as file:
            part_counter = 1
            while True:
                file_part = file.read(part_size)
                if not len(file_part):
                    break
                upload_part = s3_client.upload_part(
                    Bucket = bucket_name,
                    Key = key_name,
                    UploadId = mpu_id,
                    Body = file_part,
                    PartNumber = part_counter
                )
                part_list.append({'PartNumber': part_counter, 'ETag':
 upload_part['ETag']})
                part_counter += 1
    except ClientError as e:
```

```
        logging.error(e)
        return None
    return part_list
```

AWS CLI 사용

이 예시는 AWS CLI를 사용하여 단일 객체를 여러 부분으로 나눈 다음 디렉터리 버킷에 업로드하는 방법을 보여 줍니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

```
aws s3api upload-part --bucket bucket-base-name--zone-id--x-s3 --
key KEY_NAME --part-number 1 --body LOCAL_FILE_NAME --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
```

자세한 내용은 AWS Command Line Interface의 upload-part를 참조하세요.

멀티파트 업로드 완료

다음 예시는 멀티파트 업로드를 완료하는 방법을 보여 줍니다.

AWS SDK 사용

SDK for Java 2.x

다음 예시는 SDK for Java 2.x를 사용하여 멀티파트 업로드를 완료하는 방법을 보여 줍니다.

Example

```
/**
 * This method completes the multipart upload request by collating all the upload
 parts
 * @param s3
 * @param bucketName - for example, 'doc-example-bucket--usw2-az1--x-s3'
 * @param key
 * @param uploadId
 * @param uploadParts
 */
 private static void completeMultipartUpload(S3Client s3, String bucketName, String
 key, String uploadId, ListCompletedPart uploadParts) {
        CompletedMultipartUpload completedMultipartUpload =
 CompletedMultipartUpload.builder()
                .parts(uploadParts)
                .build();
```

```
        CompleteMultipartUploadRequest completeMultipartUploadRequest =
                CompleteMultipartUploadRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .multipartUpload(completedMultipartUpload)
                        .build();

        s3.completeMultipartUpload(completeMultipartUploadRequest);
    }

    public static void multipartUploadTest(S3Client s3, String bucketName, String
 key, String localFilePath)  {
        System.out.println("Starting multipart upload for: " + key);
        try {
            String uploadId = createMultipartUpload(s3, bucketName, key);
            System.out.println(uploadId);
            ListCompletedPart parts = multipartUpload(s3, bucketName, key, uploadId,
 localFilePath);
            completeMultipartUpload(s3, bucketName, key, uploadId, parts);
            System.out.println("Multipart upload completed for: " + key);
        }

        catch (Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

SDK for Python

다음 예시는 SDK for Python을 사용하여 멀티파트 업로드를 완료하는 방법을 보여 줍니다.

Example

```
def complete_multipart_upload(s3_client, bucket_name, key_name, mpu_id, part_list):
    '''
    Completes a multipart upload to a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: The destination bucket for the multipart upload
    :param key_name: The key name for the object to be uploaded
    :param mpu_id: The UploadId returned from the create_multipart_upload call
```

```
    :param part_list: The list of uploaded part numbers with their associated ETags
    :return: True if the multipart upload was completed successfully, else False
    '''

    try:
        s3_client.complete_multipart_upload(
            Bucket = bucket_name,
            Key = key_name,
            UploadId = mpu_id,
            MultipartUpload = {
                'Parts': part_list
            }
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True

if __name__ == '__main__':
    MB = 1024 ** 2
    region = 'us-west-2'
    bucket_name = 'BUCKET_NAME'
    key_name = 'OBJECT_NAME'
    part_size = 10 * MB
    s3_client = boto3.client('s3', region_name = region)
    mpu_id = create_multipart_upload(s3_client, bucket_name, key_name)
    if mpu_id is not None:
        part_list = multipart_upload(s3_client, bucket_name, key_name, mpu_id,
 part_size)
        if part_list is not None:
            if complete_multipart_upload(s3_client, bucket_name, key_name, mpu_id,
 part_list):
                print (f'{key_name} successfully uploaded through a ultipart upload
 to {bucket_name}')
            else:
                print (f'Could not upload {key_name} hrough a multipart upload to
 {bucket_name}')
```

AWS CLI 사용

이 예시는 AWS CLI을 사용하여 디렉터리 버킷에 대한 멀티파트 업로드를 완료하는 방법을 보여 줍니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

```
aws s3api complete-multipart-upload --bucket bucket-base-name--zone-id--x-s3 --
key KEY_NAME --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
 --multipart-upload file://parts.json
```

이 예시는 전체 파일로 리어셈블해야 하는 멀티파트 업로드의 부분을 설명하는 JSON 구조를 사용합니다. 이 예시에서는 `file://` 접두사를 사용하여 `parts`라는 로컬 폴더에 있는 파일에서 JSON 구조를 로드합니다.

parts.json:

```
parts.json
{
  "Parts": [
    {
      "ETag": "6b78c4a64dd641a58dac8d9258b88147",
      "PartNumber": 1
    }
  ]
}
```

자세한 내용은 AWS Command Line Interface의 complete-multipart-upload를 참조하세요.

멀티파트 업로드 중단

다음 예시는 멀티파트 업로드를 중단하는 방법을 보여 줍니다.

AWS SDK 사용

SDK for Java 2.x

다음 예시는 SDK for Java 2.x를 사용하여 멀티파트 업로드를 중단하는 방법을 보여 줍니다.

Example

```
public static void abortMultiPartUploads( S3Client s3, String bucketName ) {

        try {
            ListMultipartUploadsRequest listMultipartUploadsRequest =
 ListMultipartUploadsRequest.builder()
                    .bucket(bucketName)
```

```
                    .build();

            ListMultipartUploadsResponse response =
 s3.listMultipartUploads(listMultipartUploadsRequest);
            ListMultipartUpload uploads = response.uploads();

            AbortMultipartUploadRequest abortMultipartUploadRequest;
            for (MultipartUpload upload: uploads) {
                abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                        .bucket(bucketName)
                        .key(upload.key())
                        .uploadId(upload.uploadId())
                        .build();

                s3.abortMultipartUpload(abortMultipartUploadRequest);
            }

        }

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

SDK for Python

다음 예시는 SDK for Python을 사용하여 멀티파트 업로드를 중단하는 방법을 보여 줍니다.

Example

```
import logging
import boto3
from botocore.exceptions import ClientError


def abort_multipart_upload(s3_client, bucket_name, key_name, upload_id):
    '''
    Aborts a partial multipart upload in a directory bucket.

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket where the multipart upload was initiated - for
 example, 'doc-example-bucket--usw2-az1--x-s3'
```

```
    :param key_name: Name of the object for which the multipart upload needs to be
 aborted
    :param upload_id: Multipart upload ID for the multipart upload to be aborted
    :return: True if the multipart upload was successfully aborted, False if not
    '''
    try:
        s3_client.abort_multipart_upload(
            Bucket = bucket_name,
            Key = key_name,
            UploadId = upload_id
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True


if __name__ == '__main__':
    region = 'us-west-2'
    bucket_name = 'BUCKET_NAME'
    key_name = 'KEY_NAME'
        upload_id = 'UPLOAD_ID'
    s3_client = boto3.client('s3', region_name = region)
    if abort_multipart_upload(s3_client, bucket_name, key_name, upload_id):
        print (f'Multipart upload for object {key_name} in {bucket_name} bucket has
 been aborted')
    else:
        print (f'Unable to abort multipart upload for object {key_name} in
 {bucket_name} bucket')
```

AWS CLI 사용

다음 예시는 AWS CLI를 사용하여 멀티파트 업로드를 중단하는 방법을 보여 줍니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

```
aws s3api abort-multipart-upload --bucket bucket-base-name--zone-id--x-s3 --
key KEY_NAME --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
MAQAAAAB0OxUFeA7LTbWWFS8WYwhrxDxTIDN-pdEEq_agIHqsbg"
```

자세한 내용은 AWS Command Line Interface의 abort-multipart-upload를 참조하세요.

멀티파트 업로드 복사 작업 생성

> **Note**
>
> - UploadPartCopy가 디렉터리 버킷의 새 객체 파트 복사본을 SSE-KMS로 암호화하려면 KMS 키(특히 고객 관리형 키)를 사용하여 디렉터리 버킷의 기본 암호화 구성으로 SSE-KMS를 지정해야 합니다. AWS 관리형 키(aws/s3)는 지원되지 않습니다. SSE-KMS 구성은 버킷 수명 기간 동안 디렉터리 버킷당 고객 관리형 키 1개만 지원할 수 있습니다. SSE-KMS에 고객 관리형 키를 지정한 후에는 버킷의 SSE-KMS 구성을 위한 고객 관리형 키를 재정의할 수 없습니다. UploadPartCopy 요청 헤더에서 SSE-KMS를 사용하여 새 객체 부분 사본에 대한 서버 측 암호화 설정을 지정할 수 없습니다. 또한, `CreateMultipartUpload` 요청에서 제공하는 요청 헤더는 대상 버킷의 기본 암호화 구성과 일치해야 합니다.
> - 범용 버킷에서 디렉터리 버킷으로, 디렉터리 버킷에서 범용 버킷으로 또는 디렉터리 버킷 간에 UploadPartCopy를 통해 SSE-KMS로 암호화된 객체를 복사하는 경우 S3 버킷 키는 지원되지 않습니다. 이 경우 Amazon S3는 KMS 암호화 개체에 대한 사본 요청이 있을 때마다 AWS KMS를 직접적으로 호출합니다.

다음 예제는 멀티파트 업로드를 사용하여 버킷 간 객체를 복사하는 방법을 보여줍니다.

AWS SDK 사용

SDK for Java 2.x

다음 예시는 SDK for Java 2.x를 통해 멀티파트 업로드를 사용하여 프로그래밍 방식으로 버킷의 객체를 다른 버킷으로 복사하는 방법을 보여 줍니다.

Example

```
/**
 * This method creates a multipart upload request that generates a unique upload ID
 that is used to track
 * all the upload parts.
 *
 * @param s3
 * @param bucketName
 * @param key
 * @return
 */
```

```
private static String createMultipartUpload(S3Client s3, String bucketName, String
key) {
       CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
               .bucket(bucketName)
               .key(key)
               .build();
       String uploadId = null;
       try {
           CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
           uploadId = response.uploadId();
       } catch (S3Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return uploadId;
 }

 /**
  * Creates copy parts based on source object size and copies over individual parts
  *
  * @param s3
  * @param sourceBucket
  * @param sourceKey
  * @param destnBucket
  * @param destnKey
  * @param uploadId
  * @return
  * @throws IOException
  */
   public static ListCompletedPart multipartUploadCopy(S3Client s3, String
sourceBucket, String sourceKey, String destnBucket, String destnKey, String
uploadId) throws IOException {

       // Get the object size to track the end of the copy operation.
       HeadObjectRequest headObjectRequest = HeadObjectRequest
               .builder()
               .bucket(sourceBucket)
               .key(sourceKey)
               .build();
       HeadObjectResponse response = s3.headObject(headObjectRequest);
       Long objectSize = response.contentLength();
```

```
        System.out.println("Source Object size: " + objectSize);

        // Copy the object using 20 MB parts.
        long partSize = 20 * 1024 * 1024;
        long bytePosition = 0;
        int partNum = 1;
        ListCompletedPart completedParts = new ArrayList<>();
        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);

            System.out.println("part no: " + partNum + ", bytePosition: " +
bytePosition + ", lastByte: " + lastByte);

            // Copy this part.
            UploadPartCopyRequest req = UploadPartCopyRequest.builder()
                    .uploadId(uploadId)
                    .sourceBucket(sourceBucket)
                    .sourceKey(sourceKey)
                    .destinationBucket(destnBucket)
                    .destinationKey(destnKey)
                    .copySourceRange("bytes="+bytePosition+"-"+lastByte)
                    .partNumber(partNum)
                    .build();
            UploadPartCopyResponse res = s3.uploadPartCopy(req);
            CompletedPart part = CompletedPart.builder()
                    .partNumber(partNum)
                    .eTag(res.copyPartResult().eTag())
                    .build();
            completedParts.add(part);
            partNum++;
            bytePosition += partSize;
        }
        return completedParts;
    }


    public static void multipartCopyUploadTest(S3Client s3, String srcBucket, String
srcKey, String destnBucket, String destnKey)  {
        System.out.println("Starting multipart copy for: " + srcKey);
        try {
            String uploadId = createMultipartUpload(s3, destnBucket, destnKey);
            System.out.println(uploadId);
```

```
            ListCompletedPart parts = multipartUploadCopy(s3, srcBucket,
 srcKey,destnBucket,  destnKey, uploadId);
            completeMultipartUpload(s3, destnBucket, destnKey, uploadId, parts);
            System.out.println("Multipart copy completed for: " + srcKey);
        } catch (Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

SDK for Python

다음 예시는 SDK for Python을 통해 멀티파트 업로드를 사용하여 프로그래밍 방식으로 버킷의 객체를 다른 버킷으로 복사하는 방법을 보여 줍니다.

Example

```
import logging
import boto3
from botocore.exceptions import ClientError

def head_object(s3_client, bucket_name, key_name):
    '''
    Returns metadata for an object in a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket that contains the object to query for metadata
    :param key_name: Key name to query for metadata
    :return: Metadata for the specified object if successful, else None
    '''

    try:
        response = s3_client.head_object(
            Bucket = bucket_name,
            Key = key_name
        )
        return response
    except ClientError as e:
        logging.error(e)
        return None

def create_multipart_upload(s3_client, bucket_name, key_name):
    '''
    Create a multipart upload to a directory bucket
```

```
    :param s3_client: boto3 S3 client
    :param bucket_name: Destination bucket for the multipart upload
    :param key_name: Key name of the object to be uploaded
    :return: UploadId for the multipart upload if created successfully, else None
    '''

    try:
        mpu = s3_client.create_multipart_upload(Bucket = bucket_name, Key =
 key_name)
        return mpu['UploadId']
    except ClientError as e:
        logging.error(e)
        return None

def multipart_copy_upload(s3_client, source_bucket_name, key_name,
 target_bucket_name, mpu_id, part_size):
    '''
    Copy an object in a directory bucket to another bucket in multiple parts of a
 specified size

    :param s3_client: boto3 S3 client
    :param source_bucket_name: Bucket where the source object exists
    :param key_name: Key name of the object to be copied
    :param target_bucket_name: Destination bucket for copied object
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_size: The size parts that the object will be broken into, in bytes.
                      Minimum 5 MiB, Maximum 5 GiB. There is no minimum size for the
 last part of your multipart upload.
    :return: part_list for the multipart copy if all parts are copied successfully,
 else None
    '''

    part_list = []
    copy_source = {
        'Bucket': source_bucket_name,
        'Key': key_name
    }
    try:
        part_counter = 1
        object_size = head_object(s3_client, source_bucket_name, key_name)
        if object_size is not None:
            object_size = object_size['ContentLength']
        while (part_counter - 1) * part_size <object_size:
```

```
            bytes_start = (part_counter - 1) * part_size
            bytes_end = (part_counter * part_size) - 1
            upload_copy_part = s3_client.upload_part_copy (
                Bucket = target_bucket_name,
                CopySource = copy_source,
                CopySourceRange = f'bytes={bytes_start}-{bytes_end}',
                Key = key_name,
                PartNumber = part_counter,
                UploadId = mpu_id
            )
            part_list.append({'PartNumber': part_counter, 'ETag':
 upload_copy_part['CopyPartResult']['ETag']})
            part_counter += 1
    except ClientError as e:
        logging.error(e)
        return None
    return part_list

def complete_multipart_upload(s3_client, bucket_name, key_name, mpu_id, part_list):
    '''
    Completes a multipart upload to a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Destination bucket for the multipart upload
    :param key_name: Key name of the object to be uploaded
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_list: List of uploaded part numbers with associated ETags
    :return: True if the multipart upload was completed successfully, else False
    '''

    try:
        s3_client.complete_multipart_upload(
            Bucket = bucket_name,
            Key = key_name,
            UploadId = mpu_id,
            MultipartUpload = {
                'Parts': part_list
            }
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True
```

```
if __name__ == '__main__':
    MB = 1024 ** 2
    region = 'us-west-2'
    source_bucket_name = 'SOURCE_BUCKET_NAME'
    target_bucket_name = 'TARGET_BUCKET_NAME'
    key_name = 'KEY_NAME'
    part_size = 10 * MB
    s3_client = boto3.client('s3', region_name = region)
    mpu_id = create_multipart_upload(s3_client, target_bucket_name, key_name)
    if mpu_id is not None:
        part_list = multipart_copy_upload(s3_client, source_bucket_name, key_name,
 target_bucket_name, mpu_id, part_size)
        if part_list is not None:
            if complete_multipart_upload(s3_client, target_bucket_name, key_name,
 mpu_id, part_list):
                print (f'{key_name} successfully copied through multipart copy from
 {source_bucket_name} to {target_bucket_name}')
            else:
                print (f'Could not copy {key_name} through multipart copy from
 {source_bucket_name} to {target_bucket_name}')
```

AWS CLI 사용

다음 예시는 AWS CLI를 통해 멀티파트 업로드를 사용하여 프로그래밍 방식으로 버킷의 객체를 디렉터리 버킷으로 복사하는 방법을 보여 줍니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

```
aws s3api upload-part-copy --bucket bucket-base-name--zone-id--x-s3 --
key TARGET_KEY_NAME --copy-source SOURCE_BUCKET_NAME/SOURCE_KEY_NAME --part-number 1 --
upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
```

자세한 내용은 AWS Command Line Interface의 upload-part-copy를 참조하세요.

진행 중인 멀티파트 업로드 나열

AWS SDK 또는 AWS CLI를 사용하여 디렉터리 버킷에 대해 진행 중인 멀티파트 업로드를 나열할 수 있습니다.

AWS SDK 사용

SDK for Java 2.x

다음 예시는 SDK for Java 2.x를 사용하여 진행 중인(완료되지 않은) 멀티파트 업로드를 나열하는 방법을 보여 줍니다.

Example

```
public static void listMultiPartUploads( S3Client s3, String bucketName) {
       try {
           ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
                .bucket(bucketName)
                .build();

           ListMultipartUploadsResponse response =
s3.listMultipartUploads(listMultipartUploadsRequest);
           List MultipartUpload uploads = response.uploads();
           for (MultipartUpload upload: uploads) {
               System.out.println("Upload in progress: Key = \"" + upload.key() +
"\", id = " + upload.uploadId());
           }
     }
     catch (S3Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
     }
 }
```

SDK for Python

다음 예시는 SDK for Python을 사용하여 진행 중인(완료되지 않은) 멀티파트 업로드를 나열하는 방법을 보여 줍니다.

Example

```
import logging
import boto3
from botocore.exceptions import ClientError

def list_multipart_uploads(s3_client, bucket_name):
```

```
    '''
    List any incomplete multipart uploads in a directory bucket in e specified gion

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket to check for incomplete multipart uploads
    :return: List of incomplete multipart uploads if there are any, None if not
    '''

    try:
        response = s3_client.list_multipart_uploads(Bucket = bucket_name)
        if 'Uploads' in response.keys():
            return response['Uploads']
        else:
            return None
    except ClientError as e:
        logging.error(e)

if __name__ == '__main__':
    bucket_name = 'BUCKET_NAME'
    region = 'us-west-2'
    s3_client = boto3.client('s3', region_name = region)
    multipart_uploads = list_multipart_uploads(s3_client, bucket_name)
    if multipart_uploads is not None:
        print (f'There are {len(multipart_uploads)} ncomplete multipart uploads for
 {bucket_name}')
    else:
        print (f'There are no incomplete multipart uploads for {bucket_name}')
```

AWS CLI 사용

다음 예시는 AWS CLI를 사용하여 진행 중인(완료되지 않은) 멀티파트 업로드를 나열하는 방법을 보여 줍니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

```
aws s3api list-multipart-uploads --bucket bucket-base-name--zone-id--x-s3
```

자세한 내용은 AWS Command Line Interface의 list-multipart-uploads를 참조하세요.

멀티파트 업로드의 부분 나열

다음 예시는 디렉터리 버킷에 대한 멀티파트 업로드의 부분을 나열하는 방법을 보여 줍니다.

AWS SDK 사용

SDK for Java 2.x

다음 예시는 SDK for Java 2.x를 사용하여 디렉터리 버킷에 대한 멀티파트 업로드의 부분을 나열하는 방법을 보여 줍니다.

```
public static void listMultiPartUploadsParts( S3Client s3, String bucketName, String
 objKey, String uploadID) {

        try {
            ListPartsRequest listPartsRequest = ListPartsRequest.builder()
                .bucket(bucketName)
                .uploadId(uploadID)
                .key(objKey)
                .build();

            ListPartsResponse response = s3.listParts(listPartsRequest);
            ListPart parts = response.parts();
            for (Part part: parts) {
                System.out.println("Upload in progress: Part number = \"" +
 part.partNumber() + "\", etag = " + part.eTag());
            }

        }

        catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }


    }
```

SDK for Python

다음 예시는 SDK for Python을 사용하여 디렉터리 버킷에 대한 멀티파트 업로드의 부분을 나열하는 방법을 보여 줍니다.

```
import logging
import boto3
from botocore.exceptions import ClientError
```

```
def list_parts(s3_client, bucket_name, key_name, upload_id):
    '''
    Lists the parts that have been uploaded for a specific multipart upload to a
 directory bucket.

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket that multipart uploads parts have been uploaded to
    :param key_name: Name of the object that has parts uploaded
    :param upload_id: Multipart upload ID that the parts are associated with
    :return: List of parts associated with the specified multipart upload, None if
 there are no parts
    '''
    parts_list = []
    next_part_marker = ''
    continuation_flag = True
    try:
        while continuation_flag:
            if next_part_marker == '':
                response = s3_client.list_parts(
                    Bucket = bucket_name,
                    Key = key_name,
                    UploadId = upload_id
                )
            else:
                response = s3_client.list_parts(
                    Bucket = bucket_name,
                    Key = key_name,
                    UploadId = upload_id,
                    NextPartMarker = next_part_marker
                )
            if 'Parts' in response:
                for part in response['Parts']:
                    parts_list.append(part)
                if response['IsTruncated']:
                    next_part_marker = response['NextPartNumberMarker']
                else:
                    continuation_flag = False
            else:
                continuation_flag = False
        return parts_list
    except ClientError as e:
        logging.error(e)
        return None
```

```
if __name__ == '__main__':
    region = 'us-west-2'
    bucket_name = 'BUCKET_NAME'
    key_name = 'KEY_NAME'
    upload_id = 'UPLOAD_ID'
    s3_client = boto3.client('s3', region_name = region)
    parts_list = list_parts(s3_client, bucket_name, key_name, upload_id)
    if parts_list is not None:
        print (f'{key_name} has {len(parts_list)} parts uploaded to {bucket_name}')
    else:
        print (f'There are no multipart uploads with that upload ID for
 {bucket_name} bucket')
```

AWS CLI 사용

다음 예시는 AWS CLI를 사용하여 디렉터리 버킷에 대한 멀티파트 업로드의 부분을 나열하는 방법을 보여 줍니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

```
aws s3api list-parts --bucket bucket-base-name--zone-id--x-s3 --key KEY_NAME --upload-
id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
```

자세한 내용은 AWS Command Line Interface의 list-parts를 참조하세요.

디렉터리 버킷에서 또는 디렉터리 버킷에 객체 복사

복사 작업은 Amazon S3에 이미 저장되어 있는 객체의 복사본을 만듭니다. 디렉터리 버킷과 범용 버킷 간에 객체를 복사할 수 있습니다. 또한 버킷 내 및 동일한 유형의 버킷 간(예: 디렉터리 버킷에서 디렉터리 버킷으로)에 객체를 복사할 수 있습니다.

> **Note**
>
> 소스 또는 대상 버킷이 AWS 로컬 영역에 있는 경우 다른 AWS 리전에 객체를 복사할 수 없습니다. 소스 버킷과 대상 버킷의 상위 AWS 리전이 동일해야 합니다. 소스 버킷과 대상 버킷은 서로 다른 버킷 위치 유형(가용 영역 또는 로컬 영역)일 수 있습니다.

단일 원자성 작업으로 최대 5GB의 객체 복사본을 만들 수 있습니다. 그러나 5GB보다 큰 객체를 복사하려면 멀티파트 업로드 API 작업을 사용해야 합니다. 자세한 내용은 디렉터리 버킷에 멀티파트 업로드 사용 섹션을 참조하세요.

권한

객체를 복사하려면 다음 권한이 있어야 합니다.

- 한 디렉터리 버킷에서 다른 디렉터리 버킷으로 객체를 복사하려면 `s3express:CreateSession` 권한이 있어야 합니다.
- 디렉터리 버킷에서 범용 버킷으로 객체를 복사하려면 대상 버킷에 객체 사본을 쓸 수 있도록 `s3express:CreateSession` 권한과 `s3:PutObject` 권한이 있어야 합니다.
- 범용 버킷에서 디렉터리 버킷으로 객체를 복사하려면 복사되는 소스 객체를 읽을 수 있도록 `s3express:CreateSession` 권한과 `s3:GetObject` 권한이 있어야 합니다.

  자세한 내용은 Amazon Simple Storage Service API 참조에서 CopyObject를 참조하십시오.

암호화

Amazon S3는 S3 버킷에 업로드되는 모든 새 객체를 자동으로 암호화합니다. S3 버킷의 기본 암호화 구성은 항상 활성화되어 있으며, 최소한 Amazon S3 관리형 키를 통한 서버 측 암호화(SSE-S3)로 설정되어 있습니다.

디렉터리 버킷의 경우 SSE-S3 및 AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 사용한 서버 측 암호화가 지원됩니다. 대상 버킷이 디렉터리 버킷인 경우 대상 버킷의 기본 암호화는 원하는 암호화 구성을 사용하고 버킷 기본 암호화는 재정의하지 않는 것이 좋습니다. 그러면 새 객체가 원하는 암호화 설정으로 자동 암호화됩니다. 또한, 범용 버킷에서 디렉터리 버킷으로, 디렉터리 버킷에서 범용 버킷으로 또는 디렉터리 버킷 간에 CopyObject를 통해 SSE-KMS로 암호화된 객체를 복사하는 경우 S3 버킷 키는 지원되지 않습니다. 이 경우 Amazon S3는 KMS 암호화 개체에 대한 사본 요청이 있을 때마다 AWS KMS를 직접적으로 호출합니다. 디렉터리 버킷의 암호화 재정의 동작에 대한 자세한 내용은 Specifying server-side encryption with AWS KMS for new object uploads를 참조하세요.

범용 버킷의 경우 SSE-S3(기본값), AWS Key Management Service(AWS KMS) 키를 사용한 서버 측 암호화(SSE-KMS), AWS KMS 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS) 또는 고객 제공 키를 사용한 서버 측 암호화(SSE-C)를 사용할 수 있습니다.

디렉터리 버킷(소스 또는 대상 버킷)에 대해 DSSE-KMS 또는 SSE-C를 사용하도록 지정하는 복사 요청을 하면 응답에 오류가 반환됩니다.

Tags

디렉터리 버킷은 태그를 지원하지 않습니다. 범용 버킷의 태그가 있는 객체를 디렉터리 버킷으로 복사하면 HTTP `501 (Not Implemented)` 응답이 표시됩니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 CopyObject를 참조하십시오.

ETag

S3 Express One Zone의 엔터티 태그(ETag)는 임의의 영숫자 문자열이며 MD5 체크섬이 아닙니다. 객체 무결성을 보장하려면 추가 체크섬을 사용하세요.

추가 체크섬

S3 Express One Zone은 업로드 또는 다운로드 중에 데이터를 검증하는 데 사용하는 체크섬 알고리즘을 선택할 수 있는 옵션을 제공합니다. SHA(보안 해시 알고리즘) 또는 CRC(순환 중복 검사) 데이터 무결성 확인 알고리즘(CRC32, CRC32C, SHA-1, SHA-256) 중 하나를 선택할 수 있습니다. MD5 기반 체크섬은 S3 Express One Zone 스토리지 클래스에서 지원되지 않습니다.

자세한 내용은 S3 추가 체크섬 모범 사례 섹션을 참조하세요.

지원 기능

S3 Express One Zone에 지원되는 Amazon S3 기능에 대한 자세한 내용은 디렉터리 버킷의 차이점 섹션을 참조하세요.

S3 콘솔 사용(디렉터리 버킷에 복사)

> **Note**
>
> 콘솔을 사용하여 디렉터리 버킷에 객체를 복사할 때의 규제 및 제한은 다음과 같습니다.
>
> - Copy 작업은 지정된 폴더 내의 모든 객체(접두사)에 적용됩니다. 작업이 진행되는 동안 이 폴더에 추가된 객체가 영향을 받을 수 있습니다.
> - 고객 제공 암호화 키(SSE-C)로 암호화된 객체는 S3 콘솔을 사용하여 복사할 수 없습니다. SSE-C로 암호화된 객체를 복사하려면 AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용합니다.
> - 복사된 객체에는 원본 객체의 Object Lock 설정이 유지되지 않습니다.
> - 객체를 복사하는 버킷이 S3 객체 소유권에 대해 버킷 소유자 적용 설정을 사용하는 경우 객체 ACL은 지정된 대상에 복사되지 않습니다.
> - S3 객체 소유권에 대한 버킷 소유자 적용 설정을 사용하는 버킷에 객체를 복사하려면 소스 버킷도 버킷 소유자 적용 설정을 사용하거나 다른 AWS 계정 및 그룹에 부여된 객체 ACL을 제거해야 합니다.
> - 범용 버킷에서 디렉터리 버킷으로 복사된 객체는 객체 태그, ACL 또는 Etag 값을 유지하지 않습니다. 체크섬 값은 복사할 수 있지만, Etag와 동일하지는 않습니다. 체크섬 값은 추가되었을 때와 비교하여 변경될 수 있습니다.

- 디렉터리 버킷으로 복사되는 모든 객체가 S3 객체 소유권에 대해 버킷 소유자 적용 설정을 사용합니다.

범용 버킷 또는 디렉토리 버킷에서 디렉토리 버킷으로 개체를 복사하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 객체를 복사할 소스 버킷 유형은 다음과 같습니다.
   - 범용 버킷에서 복사하려면 범용 버킷 탭을 선택합니다.
   - 범용 버킷에서 복사하려면 디렉터리 버킷 탭을 선택합니다.
3. 복사할 객체가 포함된 범용 버킷 또는 디렉터리 버킷을 선택합니다.
4. 객체(Objects) 탭을 선택합니다. 객체 페이지에서 복사할 객체 이름 왼쪽에 있는 확인란을 선택합니다.
5. 작업 메뉴에서 복사를 선택합니다.

   복사 페이지가 나타납니다.
6. 대상에서 대상 유형에 맞는 디렉터리 버킷을 선택합니다. 대상 경로를 지정하려면 S3 찾아보기를 선택하고 대상으로 이동한 후 대상 왼쪽에 있는 옵션 버튼을 선택합니다. 오른쪽 하단 모서리에서 대상 선택(Choose destination)을 선택합니다.

   또는 대상 경로를 입력합니다.
7. 추가 복사 설정에서 소스 설정 복사, 설정 지정 안 함 또는 설정 지정 중 원하는 것을 선택합니다. 소스 설정 복사가 기본 옵션입니다. 소스 설정 속성 없이 객체만 복사하려면 설정 지정 안 함을 선택합니다. 서버 측 암호화, 체크섬 및 메타데이터에 대한 설정을 지정하려면 설정 지정을 선택합니다.
8. 오른쪽 하단 모서리에서 복사(Copy)를 선택합니다. Amazon S3가 객체를 대상에 복사합니다.

S3 콘솔 사용(범용 버킷에 복사)

> **Note**
>
> 콘솔을 사용하여 범용 버킷에 객체를 복사할 때의 규제 및 제한은 다음과 같습니다.
>
> - Copy 작업은 지정된 폴더 내의 모든 객체(접두사)에 적용됩니다. 작업이 진행되는 동안 이 폴더에 추가된 객체가 영향을 받을 수 있습니다.

- 고객 제공 암호화 키(SSE-C)로 암호화된 객체는 S3 콘솔을 사용하여 복사할 수 없습니다. SSE-C로 암호화된 객체를 복사하려면 AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용합니다.
- 복사된 객체에는 원본 객체의 Object Lock 설정이 유지되지 않습니다.
- 객체를 복사하는 버킷이 S3 객체 소유권에 대해 버킷 소유자 적용 설정을 사용하는 경우 객체 ACL은 지정된 대상에 복사되지 않습니다.
- S3 객체 소유권에 대한 버킷 소유자 적용 설정을 사용하는 버킷에 객체를 복사하려면 소스 버킷도 버킷 소유자 적용 설정을 사용하거나 다른 AWS 계정 및 그룹에 부여된 객체 ACL을 제거해야 합니다.

디렉터리 버킷의 객체를 범용 버킷으로 복사하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 디렉터리 버킷 탭을 선택합니다.
4. 삭제할 객체가 포함된 디렉터리 버킷을 선택합니다.
5. 객체(Objects) 탭을 선택합니다. 객체 페이지에서 복사할 객체 이름 왼쪽에 있는 확인란을 선택합니다.
6. 작업 메뉴에서 복사를 선택합니다.
7. 대상에서 대상 유형에 맞는 범용 버킷을 선택합니다. 대상 경로를 지정하려면 S3 찾아보기를 선택하고 대상으로 이동한 후 대상 왼쪽에 있는 옵션 버튼을 선택합니다. 오른쪽 하단 모서리에서 대상 선택(Choose destination)을 선택합니다.

   또는 대상 경로를 입력합니다.
8. 추가 복사 설정에서 소스 설정 복사, 설정 지정 안 함 또는 설정 지정 중 원하는 것을 선택합니다. 소스 설정 복사가 기본 옵션입니다. 소스 설정 속성 없이 객체만 복사하려면 설정 지정 안 함을 선택합니다. 스토리지 클래스, ACL, 객체 태그, 메타데이터, 서버 측 암호화 및 추가 체크섬에 대한 설정을 지정하려면 설정 지정을 선택합니다.
9. 오른쪽 하단 모서리에서 복사(Copy)를 선택합니다. Amazon S3가 객체를 대상에 복사합니다.

AWS SDK 사용

SDK for Java 2.x

Example

```
public static void copyBucketObject (S3Client s3, String sourceBucket, String
objectKey, String targetBucket) {
     CopyObjectRequest copyReq = CopyObjectRequest.builder()
         .sourceBucket(sourceBucket)
         .sourceKey(objectKey)
         .destinationBucket(targetBucket)
         .destinationKey(objectKey)
         .build();
      String temp = "";

      try {
          CopyObjectResponse copyRes = s3.copyObject(copyReq);
          System.out.println("Successfully copied " + objectKey +" from bucket " +
sourceBucket +" into bucket "+targetBucket);
      }

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

AWS CLI 사용

다음 `copy-object` 예시 명령은 AWS CLI를 사용하여 버킷의 객체를 다른 버킷으로 복사하는 방법을 보여 줍니다. 버킷 유형 간에 객체를 복사할 수 있습니다. 이 명령을 실행하려면 user input placeholders를 사용자의 정보로 대체합니다.

```
aws s3api copy-object --copy-source SOURCE_BUCKET/SOURCE_KEY_NAME --key TARGET_KEY_NAME
 --bucket TARGET_BUCKET_NAME
```

자세한 정보는 AWS CLI 명령 참조의 copy-object 섹션을 참조하세요.

디렉터리 버킷에서 객체 삭제

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 또는 AWS SDK를 사용하여 Amazon S3 디렉터리 버킷에서 객체를 삭제할 수 있습니다. 자세한 내용은 디렉터리 버킷 작업 및 S3 Express One Zone(을)를 참조하세요.

Warning

- 객체 삭제는 실행 취소할 수 없습니다.
- 이 작업은 지정된 모든 객체를 삭제합니다. 폴더를 삭제할 때 폴더에 새 객체를 추가하기 전에 삭제 작업이 완료될 때까지 기다립니다. 그러지 않으면 새 객체도 삭제될 수 있습니다.

Note

디렉터리 버킷에서 프로그래밍 방식으로 여러 객체를 삭제할 경우 다음에 유의하세요.

- `DeleteObjects` 요청의 객체 키는 공백이 아닌 문자를 하나 이상 포함해야 합니다. 공백 문자로만 구성된 문자열은 지원되지 않습니다.
- `DeleteObjects` 요청의 객체 키에는 유니코드 제어 문자를 포함할 수 없습니다. 단, 줄바꿈(`\n`), 탭(`\t`) 및 캐리지 리턴(`\r`)은 예외입니다.

S3 콘솔 사용

객체 삭제

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 디렉터리 버킷을 선택합니다.
3. 삭제할 객체가 포함된 디렉터리 버킷을 선택합니다.
4. 객체(Objects) 탭을 선택합니다. 객체 목록에서 삭제할 하나 또는 여러 객체 옆의 확인란을 선택합니다.
5. 삭제를 선택합니다.
6. 객체 삭제 페이지에서 텍스트 상자에 **`permanently delete`**를 입력합니다.
7. 객체 삭제를 선택합니다.

AWS SDK 사용

SDK for Java 2.x

Example

다음 예시에서는 AWS SDK for Java 2.x를 사용하여 디렉터리 버킷의 객체를 삭제합니다.

```
static void deleteObject(S3Client s3Client, String bucketName, String objectKey) {



        try {

            DeleteObjectRequest del = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            s3Client.deleteObject(del);

            System.out.println("Object " + objectKey + " has been deleted");


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

    }
```

SDK for Python

Example

다음 예시에서는 AWS SDK for Python (Boto3)을 사용하여 디렉터리 버킷의 객체를 삭제합니다.

```


import logging
import boto3
from botocore.exceptions import ClientError

def delete_objects(s3_client, bucket_name, objects):
    '''
```

```
    Delete a list of objects in a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket that contains objects to be deleted; for example,
 'doc-example-bucket--usw2-az1--x-s3'
    :param objects: List of dictionaries that specify the key names to delete
    :return: Response output, else False
    '''

    try:
        response = s3_client.delete_objects(
            Bucket = bucket_name,
            Delete = {
                'Objects': objects
            }
        )
        return response
    except ClientError as e:
        logging.error(e)
        return False


if __name__ == '__main__':
    region = 'us-west-2'
    bucket_name = 'BUCKET_NAME'
    objects = [
        {
            'Key': '0.txt'
        },
        {
            'Key': '1.txt'
        },
        {
            'Key': '2.txt'
        },
        {
            'Key': '3.txt'
        },
        {
            'Key': '4.txt'
        }
    ]

    s3_client = boto3.client('s3', region_name = region)
```

```
    results = delete_objects(s3_client, bucket_name, objects)
    if results is not None:
        if 'Deleted' in results:
            print (f'Deleted {len(results["Deleted"])} objects from {bucket_name}')
        if 'Errors' in results:
            print (f'Failed to delete {len(results["Errors"])} objects from
{bucket_name}')
```

AWS CLI 사용

다음 delete-object 예시 명령에서는 AWS CLI를 사용하여 디렉터리 버킷에서 객체를 삭제하는 방법을 보여 줍니다. 이 명령을 실행하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws s3api delete-object --bucket bucket-base-name--zone-id--x-s3 --key KEY_NAME
```

자세한 정보는 AWS CLI 명령 참조의 delete-object 섹션을 참조하세요.

다음 delete-objects 예시 명령에서는 AWS CLI를 사용하여 디렉터리 버킷에서 객체를 삭제하는 방법을 보여줍니다. 이 명령을 실행하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

delete.json 파일은 다음과 같습니다.

```
{
    "Objects": [
        {
            "Key": "0.txt"
        },
        {
            "Key": "1.txt"
        },
        {
            "Key": "2.txt"
        },
        {
            "Key": "3.txt"
        }
    ]
}
```

delete-objects 예시 명령은 다음과 같습니다.

```
aws s3api delete-objects --bucket bucket-base-name--zone-id--x-s3 --delete
 file://delete.json
```

자세한 정보는 AWS CLI 명령 참조의 delete-objects 섹션을 참조하세요.

디렉터리 버킷에서 객체 다운로드

다음 코드 예제는 GetObject API 작업을 사용하여 Amazon S3 디렉터리 버킷의 객체에서 데이터를 읽는(다운로드하는) 방법을 보여줍니다.

AWS SDK 사용

SDK for Java 2.x

Example

다음 코드 예제는 AWS SDK for Java 2.x를 사용하여 디렉터리 버킷의 객체에서 데이터를 읽는 방법을 보여줍니다.

```
public static void getObject(S3Client s3Client, String bucketName, String objectKey)
 {
     try {
         GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(objectKey)
            .bucket(bucketName)
            .build();

         ResponseBytes GetObjectResponse objectBytes =
 s3Client.getObjectAsBytes(objectRequest);
         byte[] data = objectBytes.asByteArray();

         //Print object contents to console
         String s = new String(data, StandardCharsets.UTF_8);
         System.out.println(s);
    }

    catch (S3Exception e) {
```

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

SDK for Python

Example

다음 코드 예제는 AWS SDK for Python (Boto3)을 사용하여 디렉터리 버킷의 객체에서 데이터를 읽는 방법을 보여줍니다.

```
import boto3
from botocore.exceptions import ClientError
from botocore.response import StreamingBody

def get_object(s3_client: boto3.client, bucket_name: str, key_name: str) ->
 StreamingBody:
    """
    Gets the object.
    :param s3_client:
    :param bucket_name: The bucket that contains the object.
    :param key_name: The key of the object to be downloaded.
    :return: The object data in bytes.
    """
    try:
        response = s3_client.get_object(Bucket=bucket_name, Key=key_name)
        body = response['Body'].read()
        print(f"Got object '{key_name}' from bucket '{bucket_name}'.")
    except ClientError:
        print(f"Couldn't get object '{key_name}' from bucket '{bucket_name}'.")
        raise
    else:
        return body

def main():
    s3_client = boto3.client('s3')
    resp = get_object(s3_client, 'doc-example-bucket--use1-az4--x-s3', 'sample.txt')
    print(resp)

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

AWS CLI 사용

다음 `get-object` 예제 명령은 AWS CLI를 사용하여 Amazon S3에서 객체를 다운로드하는 방법을 보여줍니다. 이 명령은 디렉터리 버킷 *bucket-base-name*`--`*zone-id*`--x-s3`에서 *KEY_NAME* 객체를 가져옵니다. 객체는 *LOCAL_FILE_NAME*라는 파일로 다운로드됩니다. 이 명령을 실행하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws s3api get-object --bucket bucket-base-name--zone-id--x-s3 --
key KEY_NAME LOCAL_FILE_NAME
```

자세한 정보는 AWS CLI 명령 참조의 get-object 섹션을 참조하세요.

객체 디렉터리 버킷을 공유하기 위해 미리 서명된 URL 생성

다음 코드 예시에서는 Amazon S3 디렉터리 버킷에서 객체를 공유하기 위해 미리 서명된 URLs을 생성하는 방법을 보여줍니다.

AWS CLI 사용

다음 예시 명령은 AWS CLI를 사용하여 Amazon S3의 객체에 대해 미리 서명된 URL을 생성하는 방법을 보여줍니다. 이 명령은 *bucket-base-name*`--`*zone-id*`--x-s3` 디렉터리 버킷에서 *KEY_NAME* 객체에 대해 미리 서명된 URL을 생성합니다. 이 명령을 실행하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws s3 presign s3://bucket-base-name--zone-id--x-s3/KEY_NAME --expires-in 7200
```

자세한 정보는 AWS CLI 명령 참조의 presign 섹션을 참조하세요.

디렉터리 버킷에서 객체 메타데이터 검색

다음 AWS SDK 및 AWS CLI 예시는 `HeadObject` 및 `GetObjectAttributes` API 작업을 사용하여 객체 자체를 반환하지 않고 Amazon S3 디렉터리 버킷의 객체에서 메타데이터를 검색하는 방법을 보여줍니다.

AWS SDK 사용

SDK for Java 2.x

Example

```
public static void headObject(S3Client s3Client, String bucketName, String
 objectKey) {
    try {
        HeadObjectRequest headObjectRequest = HeadObjectRequest
                .builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();
        HeadObjectResponse response = s3Client.headObject(headObjectRequest);
        System.out.format("Amazon S3 object: \"%s\" found in bucket: \"%s\" with
 ETag: \"%s\"", objectKey, bucketName, response.eTag());
    }
    catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
```

AWS CLI 사용

다음 head-object 예시 명령은 AWS CLI를 사용하여 객체에서 메타데이터를 검색하는 방법을 보여 줍니다. 이 명령을 실행하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws s3api head-object --bucket bucket-base-name--zone-id--x-s3 --key KEY_NAME
```

자세한 정보는 AWS CLI 명령 참조의 head-object 섹션을 참조하세요.

다음 get-object-attributes 예시 명령은 AWS CLI를 사용하여 객체에서 메타데이터를 검색하는 방법을 보여 줍니다. 이 명령을 실행하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws s3api get-object-attributes --bucket bucket-base-name--zone-id--x-s3 --key KEY_NAME
 --object-attributes "StorageClass" "ETag" "ObjectSize"
```

자세한 정보는 AWS CLI 명령 참조의 get-object-attributes 섹션을 참조하세요.

디렉터리 버킷에서 객체 나열

다음 코드 예시는 ListObjectsV2 API 작업을 사용하여 Amazon S3 디렉터리 버킷의 객체를 나열하는 방법을 보여줍니다.

AWS CLI 사용

다음 `list-objects-v2` 예시 명령은 AWS CLI를 사용하여 Amazon S3에서 객체를 나열하는 방법을 보여줍니다. 이 명령은 *`bucket-base-name`*`--`*`zone-id`*`--x-s3` 디렉터리 버 에서 객체를 나열합니다. 이 명령을 실행하려면 *`user input placeholders`*를 사용자의 정보로 대체합니다.

```
aws s3api list-objects-v2 --bucket bucket-base-name--zone-id--x-s3
```

자세한 정보는 AWS CLI 명령 참조의 list-objects-v2 섹션을 참조하세요.

디렉터리 버킷의 보안

AWS에서 클라우드 보안은 가장 중요합니다. AWS 고객은 보안에 가장 보안에 민감한 조직의 요구 사항에 부합하도록 빌드된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다. 보안은 AWS와 사용자의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드 내 클라우드 및 보안의 보안으로 설명합니다.

- 클라우드의 보안 - AWS는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호합니다. AWS는 또한 안전하게 사용할 수 있는 서비스를 제공합니다. 서드 파티 감사원은 AWS Compliance Programs의 일환으로 정기적으로 보안 효과를 테스트하고 검증합니다.

  규정 준수 프로그램에 대한 자세한 내용은 AWS 서비스 in Scope by Compliance Program 섹션을 참조하세요.
- 클라우드 내 보안 – 사용자의 책임은 사용자가 사용하는 AWS 서비스에 의해 결정됩니다. 또한 귀하는 데이터의 민감도, 회사 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 디렉터리 버킷 사용 시 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목표를 충족하도록 디렉터리 버킷을 구성하는 방법을 보여줍니다. 또한 디렉터리 버킷에서 객체를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법도 알아봅니다.

데이터 보호 및 암호화

디렉터리 버킷에 대한 암호화를 구성하는 방법은 다음 주제를 참조하세요.

주제

- 서버 측 암호화
- 디렉터리 버킷의 기본 암호화 설정 및 모니터링
- 디렉터리 버킷에서 AWS KMS 키(SSE-KMS)를 사용한 서버 측 암호화 사용
- 전송 중 암호화
- 데이터 삭제

서버 측 암호화

모든 디렉터리 버킷에는 기본적으로 암호화가 구성되어 있으며 디렉터리 버킷에 업로드되는 신규 객체는 모두 저장 시 자동으로 암호화됩니다. Amazon S3 관리형 키(SSE-S3)를 사용한 서버 측 암호화가 모든 디렉터리 버킷의 기본 암호화 구성입니다. 다른 암호화 유형을 지정하려는 경우 버킷의 기본 암호화 구성을 설정하여 AWS Key Management Service(AWS KMS) 키 (SSE-KMS)를 사용한 서버 측 암호화를 사용할 수 있습니다. 디렉터리 버킷의 SSE-KMS에 대한 자세한 내용은 디렉터리 버킷에서 AWS KMS 키(SSE-KMS)를 사용한 서버 측 암호화 사용 섹션을 참조하세요.

버킷의 기본 암호화는 원하는 암호화 구성을 사용하고 `CreateSession` 요청 또는 PUT 객체 요청에서 버킷 기본 암호화를 재정의하지 않는 것이 좋습니다. 그러면 새 객체가 원하는 암호화 설정으로 자동 암호화됩니다. 디렉터리 버킷의 암호화 재정의 동작에 대한 자세한 내용은 Specifying server-side encryption with AWS KMS for new object uploads를 참조하세요.

디렉터리 버킷이 포함된 SSE-KMS는 다음과 같은 측면에서 범용 버킷의 SSE-KMS와 다릅니다.

- SSE-KMS 구성은 버킷 수명 기간 동안 디렉터리 버킷당 고객 관리형 키 1개만 지원할 수 있습니다. AWS 관리형 키(`aws/s3`)는 지원되지 않습니다. 또한, SSE-KMS에 고객 관리형 키를 지정한 후에는 버킷의 SSE-KMS 구성을 위한 고객 관리형 키를 재정의할 수 없습니다.

  다음과 같은 방법으로 버킷의 SSE-KMS 구성에 대해 지정한 고객 관리형 키를 식별할 수 있습니다.

  - 응답에서 `x-amz-server-side-encryption-aws-kms-key-id`의 값을 찾기 위해 `HeadObject` API 작업을 요청합니다.

  데이터에 새 고객 관리형 키를 사용하려면 최신 고객 관리형 키가 있는 새 디렉터리 버킷에 기존 객체를 복사하는 것이 좋습니다.
- CopyObject 및 UploadPartCopy를 제외한 영역 엔드포인트(객체 수준) API 작업의 경우, 지연 시간이 짧도록 CreateSession을 통해 요청을 인증하고 권한을 부여합니다. 버킷의 기본 암호화는 원하는 암호화 구성을 사용하고 `CreateSession` 요청 또는 PUT 객체 요청에서 버킷 기본 암호화를 재정의하지 않는 것이 좋습니다. 그러면 새 객체가 원하는 암호화 설정으로 자동 암호화됩니다. SSE-KMS로 디렉터리 버킷의 새 객체를 암호화하려면 KMS 키(특히 고객 관리형 키)를 사용하여 디렉터

리 버킷의 기본 암호화 구성으로 SSE-KMS를 지정해야 합니다. 그런 다음 영역 엔드포인트 API 작업을 위한 세션이 생성되면 세션 중에 SSE-KMS 및 S3 버킷 키를 사용하여 새 객체가 자동으로 암호화되고 해독됩니다. 디렉터리 버킷의 암호화 재정의 동작에 대한 자세한 내용은 Specifying server-side encryption with AWS KMS for new object uploads를 참조하세요.

영역 엔드포인트 API 직접 호출(CopyObject 및 UploadPartCopy 제외)에서는 `CreateSession` 요청에서 암호화 설정(`x-amz-server-side-encryption`, `x-amz-server-side-encryption-aws-kms-key-id`, `x-amz-server-side-encryption-context`, `x-amz-server-side-encryption-bucket-key-enabled`)의 값을 재정의할 수 없습니다. 영역 엔드포인트 API 직접 호출에서 이러한 암호화 설정 값을 명시적으로 지정할 필요는 없으며, Amazon S3는 `CreateSession` 요청의 암호화 설정 값을 사용하여 디렉터리 버킷의 새 객체를 보호합니다.

> **Note**
>
> `CreateSession`의 경우 AWS CLI 또는 AWS SDK를 사용하면 세션 토큰이 자동으로 새로 고쳐져 세션이 만료될 때 서비스가 중단되지 않도록 합니다. AWS CLI 또는 AWS SDK는 `CreateSession` 요청에 버킷의 기본 암호화 구성을 사용합니다. `CreateSession` 요청의 암호화 설정 값을 재정의하는 것은 지원되지 않습니다. 또한, 영역 엔드포인트 API 직접 호출(CopyObject 및 UploadPartCopy 제외)에서는 `CreateSession` 요청의 암호화 설정 값을 재정의할 수 없습니다.

- CopyObject의 경우 SSE-KMS로 디렉터리 버킷의 새 객체 사본을 암호화하려면 KMS 키(특히 고객 관리형 키)를 사용하여 디렉터리 버킷의 기본 암호화 구성으로 SSE-KMS를 지정해야 합니다. 그런 다음 SSE-KMS를 사용하여 새 객체 사본의 서버 측 암호화 설정을 지정할 때는 암호화 키가 디렉터리 버킷의 기본 암호화 구성에 지정한 것과 동일한 고객 관리형 키인지 확인해야 합니다. UploadPartCopy의 경우 SSE-KMS로 디렉터리 버킷의 새 객체 부분 사본을 암호화하려면 KMS 키(특히 고객 관리형 키)를 사용하여 디렉터리 버킷의 기본 암호화 구성으로 SSE-KMS를 지정해야 합니다. UploadPartCopy 요청 헤더에서 SSE-KMS를 사용하여 새 객체 부분 사본에 대한 서버 측 암호화 설정을 지정할 수 없습니다. 또한, CreateMultipartUpload 요청에서 제공하는 암호화 설정은 대상 버킷의 기본 암호화 구성과 일치해야 합니다.
- S3 버킷 키는 디렉터리 버킷의 `GET` 및 `PUT` 작업에 대해 항상 활성화되며 비활성화할 수 없습니다. CopyObject, UploadPartCopy, 배치 운영의 Copy 작업 또는 import 작업을 통해 SSE-KMS로 암호화된 객체를 범용 버킷에서 디렉터리 버킷으로, 디렉터리 버킷에서 범용 버킷으로 또는 디렉터리 버킷 간에 복사하는 경우 S3 버킷 키는 지원되지 않습니다. 이 경우 Amazon S3는 KMS 암호화 개체에 대한 사본 요청이 있을 때마다 AWS KMS를 직접적으로 호출합니다.
- 디렉터리 버킷에서 암호화를 위한 AWS KMS 고객 관리형 키를 지정할 때는 키 ID 또는 키 ARN만 사용하세요. KMS 키의 키 별칭 형식은 지원되지 않습니다.

디렉터리 버킷은 AWS Key Management Service(AWS KMS) 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS) 또는 고객 제공 암호화 키를 사용한 서버 측 암호화(SSE-C)를 지원하지 않습니다.

디렉터리 버킷의 기본 암호화 설정 및 모니터링

Amazon S3 버킷에는 기본적으로 버킷 암호화가 활성화되어 있으며, 새로운 객체는 Amazon S3 관리형 키(SSE-S3)를 통한 서버 측 암호화를 사용하여 자동으로 암호화됩니다. 이 암호화는 Amazon S3 버킷의 모든 새 객체에 적용되며 무료로 제공됩니다.

키 교체 및 액세스 정책 권한 부여 관리와 같이 암호화 키를 더 세밀하게 제어해야 하는 경우 AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 통한 서버 측 암호화를 사용하는 방법도 있습니다.

> **Note**
>
> - 버킷의 기본 암호화는 원하는 암호화 구성을 사용하고 `CreateSession` 요청 또는 `PUT` 객체 요청에서 버킷 기본 암호화를 재정의하지 않는 것이 좋습니다. 그러면 새 객체가 원하는 암호화 설정으로 자동 암호화됩니다. 디렉터리 버킷의 암호화 재정의 동작에 대한 자세한 내용은 Specifying server-side encryption with AWS KMS for new object uploads를 참조하세요.
> - SSE-KMS로 디렉터리 버킷의 새 객체를 암호화하려면 KMS 키(특히 고객 관리형 키)를 사용하여 디렉터리 버킷의 기본 암호화 구성으로 SSE-KMS를 지정해야 합니다. 그런 다음 영역 엔드포인트 API 작업을 위한 세션이 생성되면 세션 중에 SSE-KMS 및 S3 버킷 키를 사용하여 새 객체가 자동으로 암호화되고 해독됩니다.
> - 기본 버킷 암호화를 SSE-KMS로 설정하면 S3 버킷 키는 디렉터리 버킷의 `GET` 및 `PUT` 작업에 대해 항상 활성화되며 비활성화할 수 없습니다. CopyObject, UploadPartCopy, 배치 운영의 Copy 작업 또는 import 작업을 통해 SSE-KMS로 암호화된 객체를 범용 버킷에서 디렉터리 버킷으로, 디렉터리 버킷에서 범용 버킷으로 또는 디렉터리 버킷 간에 복사하는 경우 S3 버킷 키는 지원되지 않습니다. 이 경우 Amazon S3는 KMS 암호화 개체에 대한 사본 요청이 있을 때마다 AWS KMS를 직접적으로 호출합니다. S3 버킷 키로 AWS KMS 요청 비용을 절감하는 방법에 대한 자세한 내용은 Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감 섹션을 참조하세요.
> - 디렉터리 버킷에서 암호화를 위한 AWS KMS 고객 관리형 키를 지정할 때는 키 ID 또는 키 ARN만 사용하세요. KMS 키의 키 별칭 형식은 지원되지 않습니다.
> - AWS KMS 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS) 및 고객 제공 키를 사용한 서버 측 암호화(SSE-C)는 디렉터리 버킷의 기본 암호화에서 지원되지 않습니다.

기본 암호화 구성 방법에 대한 자세한 내용은 기본 암호화 구성 섹션을 참조하세요.

기본 암호화에 필요한 권한에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 PutBucketEncryption 섹션을 참조하세요.

Amazon S3 콘솔, AWS SDK, Amazon S3 REST API 및 AWS Command Line Interface(AWS CLI)를 사용하여 S3 버킷에 대한 Amazon S3 기본 암호화를 구성할 수 있습니다.

S3 콘솔 사용

Amazon S3 버킷에 대해 기본 암호화를 구성하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 목록에서 원하는 버킷의 이름을 선택합니다.
4. 속성(Properties) 탭을 선택합니다.
5. 서버 측 암호화 설정에서 디렉터리 버킷이 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)를 통해 데이터를 암호화합니다.
6. 변경 사항 저장을 선택합니다.

AWS CLI 사용

다음 예시에서는 SSE-S3를 사용하거나 S3 버킷 키와 함께 SSE-KMS를 사용하여 기본 암호화를 구성하는 방법을 보여줍니다.

기본 암호화에 대한 자세한 내용은 Amazon S3 버킷에 대한 기본 서버 측 암호화 동작 설정 섹션을 참조하십시오. AWS CLI를 사용하여 기본 암호화를 구성하는 방법에 대한 자세한 내용은 put-bucket-encryption을 참조하세요.

Example – SSE-S3를 사용한 기본 암호화

이 예시에서는 Amazon S3 관리형 키를 사용하여 기본 버킷 암호화를 구성합니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

```
aws s3api put-bucket-encryption --bucket bucket-base-name--zone-id--x-s3 --server-side-encryption-configuration '{
    "Rules": [
```

```
        {
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }
    ]
}'
```

Example - S3 버킷 키와 SSE-KMS를 사용한 기본 암호화

이 예제에서는 S3 버킷 키와 SSE-KMS를 사용하여 기본 버킷 암호화를 구성합니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

```
aws s3api put-bucket-encryption --bucket bucket-base-name--zone-id--x-s3 --server-side-
encryption-configuration '{
    "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "aws:kms",
                    "KMSMasterKeyID": "KMS-Key-ARN"
                },
                "BucketKeyEnabled": true
            }
        ]
    }'
```

REST API 사용

REST API PutBucketEncryption 작업을 사용하여 사용할 서버 측 암호화 유형(SSE-S3 또는 SSE-KMS)으로 기본 암호화를 설정합니다.

자세한 내용은 Amazon Simple Storage Service API 참조에서 PutBucketEncryption를 참조하십시오.

AWS SDK 사용

AWS SDK를 사용하는 경우 서버 측 암호화에 AWS KMS keys를 사용하도록 Amazon S3에 요청할 수 있습니다. 다음 AWS SDKs for Java 및 .NET 예제에서는 SSE-KMS 및 S3 버킷 키로 디렉터리 버킷의 기본 암호화 구성을 구성합니다. 다른 SDK 사용에 대한 자세한 내용은 AWS 개발자 센터의 샘플 코드 및 라이브러리를 참조하세요.

> **Important**
>
> Amazon S3에서 서버 측 암호화에 AWS KMS key을 사용하는 경우 대칭 암호화 KMS 키를 선택해야 합니다. Amazon S3는 대칭 암호화 KMS 키만 지원합니다. 이들 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 대칭 암호화 KMS 키를 참조하십시오.

Java

AWS SDK for Java 2.x 사용 시 `applyServerSideEncryptionByDefault` 메서드를 통해 SSE-KMS로 데이터 암호화를 위한 디렉터리 버킷의 기본 암호화 구성을 지정하여 Amazon S3에 AWS KMS key 사용을 요청할 수 있습니다. 대칭 암호화 KMS 키를 만들고 요청에서 지정할 수 있습니다.

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.ServerSideEncryption;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionByDefault;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionConfiguration;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

public class Main {
    public static void main(String[] args) {
        S3Client s3 = S3Client.create();
        String bucketName = "bucket-base-name--zoneid--x-s3";
        String kmsKeyId = "your-kms-customer-managed-key-id";

        // AWS managed KMS keys aren't supported. Only customer-managed keys are
 supported.
        ServerSideEncryptionByDefault serverSideEncryptionByDefault =
 ServerSideEncryptionByDefault.builder()
                .sseAlgorithm(ServerSideEncryption.AWS_KMS)
                .kmsMasterKeyID(kmsKeyId)
                .build();

        // The bucketKeyEnabled field is enforced to be true.
        ServerSideEncryptionRule rule = ServerSideEncryptionRule.builder()
                .bucketKeyEnabled(true)
                .applyServerSideEncryptionByDefault(serverSideEncryptionByDefault)
                .build();
```

```
        ServerSideEncryptionConfiguration serverSideEncryptionConfiguration =
 ServerSideEncryptionConfiguration.builder()
                .rules(rule)
                .build();

        PutBucketEncryptionRequest putRequest = PutBucketEncryptionRequest.builder()
                .bucket(bucketName)

 .serverSideEncryptionConfiguration(serverSideEncryptionConfiguration)
                .build();

        s3.putBucketEncryption(putRequest);

    }
}
```

고객 관리형 키 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 AWS KMS API 프로그래밍을 참조하십시오.

객체를 업로드하는 사용 가능한 코드 예제는 다음 주제를 참조하십시오. 이 예시를 사용하려면 코드 예시를 업데이트하고 이전 코드 조각에서 표시된 바와 같이 암호화 정보를 제공해야 합니다.

- 단일 작업으로 객체를 업로드하려면 디렉터리 버킷에 객체 업로드 단원을 참조하십시오.
- 멀티파트 업로드 API 작업에 대해서는 디렉터리 버킷에 멀티파트 업로드 사용 섹션을 참조하세요.

.NET

AWS SDK for .NET 사용 시 `ServerSideEncryptionByDefault` 속성을 통해 SSE-KMS로 데이터 암호화를 위한 디렉터리 버킷의 기본 암호화 구성을 지정하여 Amazon S3에 AWS KMS key 사용을 요청할 수 있습니다. 대칭 암호화 고객 관리형 키를 만들고 요청에서 지정할 수 있습니다.

```
    // Set the bucket server side encryption to use AWSKMS with a customer-managed
 key id.
    // bucketName: Name of the directory bucket. "bucket-base-name--zonsid--x-s3"
    // kmsKeyId: The Id of the customer managed KMS Key. "your-kms-customer-managed-
key-id"
    // Returns True if successful.
```

```
   public static async Task<bool> SetBucketServerSideEncryption(string bucketName,
string kmsKeyId)
   {
       var serverSideEncryptionByDefault = new ServerSideEncryptionConfiguration
       {
           ServerSideEncryptionRules = new List<ServerSideEncryptionRule>
           {
               new ServerSideEncryptionRule
               {
                   ServerSideEncryptionByDefault = new
ServerSideEncryptionByDefault
                   {
                       ServerSideEncryptionAlgorithm =
ServerSideEncryptionMethod.AWSKMS,
                       ServerSideEncryptionKeyManagementServiceKeyId = kmsKeyId
                   }
               }
           }
       };
       try
       {
           var encryptionResponse =await _s3Client.PutBucketEncryptionAsync(new
PutBucketEncryptionRequest
           {
               BucketName = bucketName,
               ServerSideEncryptionConfiguration = serverSideEncryptionByDefault,
           });

           return encryptionResponse.HttpStatusCode == HttpStatusCode.OK;
       }
       catch (AmazonS3Exception ex)
       {
           Console.WriteLine(ex.ErrorCode == "AccessDenied"
               ? $"This account does not have permission to set encryption on
{bucketName}, please try again."
               : $"Unable to set bucket encryption for bucket {bucketName},
{ex.Message}");
       }
       return false;
   }
```

고객 관리형 키 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 AWS KMS API 프로그래밍을 참조하십시오.

객체를 업로드하는 사용 가능한 코드 예제는 다음 주제를 참조하십시오. 이 예시를 사용하려면 코드 예시를 업데이트하고 이전 코드 조각에서 표시된 바와 같이 암호화 정보를 제공해야 합니다.

- 단일 작업으로 객체를 업로드하려면 디렉터리 버킷에 객체 업로드 단원을 참조하십시오.
- 멀티파트 업로드 API 작업에 대해서는 디렉터리 버킷에 멀티파트 업로드 사용 섹션을 참조하세요.

AWS CloudTrail을 사용하여 디렉터리 버킷의 기본 암호화 모니터링

AWS CloudTrail 이벤트를 사용하여 Amazon S3 디렉터리 버킷에 대한 기본 암호화 구성 요청을 추적할 수 있습니다. CloudTrail 로그에 사용되는 API 이벤트 이름은 다음과 같습니다.

- `PutBucketEncryption`
- `GetBucketEncryption`
- `DeleteBucketEncryption`

> **Note**
>
> - EventBridge는 디렉터리 버킷에서 지원되지 않습니다.
> - AWS Key Management Service(AWS KMS) 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS) 또는 고객 제공 암호화 키를 사용한 서버 측 암호화(SSE-C)는 디렉터리 버킷에서 지원되지 않습니다.

AWS CloudTrail과 함께 기본 암호화를 모니터링하는 방법에 대한 자세한 내용은 AWS CloudTrail 및 Amazon EventBridge를 사용한 기본 암호화 모니터링 섹션을 참조하세요.

디렉터리 버킷에서 AWS KMS 키(SSE-KMS)를 사용한 서버 측 암호화 사용

AWS KMS 내 보안 관리가 암호화 관련 규정 준수 요구 사항을 충족하는 데 도움이 될 수 있습니다. AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 사용한 서버 측 암호화를 사용하도록 디렉터리 버킷을 구성하고, 이러한 KMS 키를 사용하여 Amazon S3 디렉터리 버킷의 데이터를 보호하도록 선택할 수 있습니다. SSE-KMS에 대한 자세한 내용은 AWS KMS 키를 사용한 서버 측 암호화(SSE-KMS) 사용 단원을 참조하십시오.

권한

AWS KMS key로 암호화된 객체를 Amazon S3에 업로드하거나 다운로드하려면 키에 대한 `kms:GenerateDataKey` 권한과 `kms:Decrypt` 권한이 있어야 합니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 키 사용자가 암호화 작업에 KMS를 사용하도록 허용을 참조하세요. –멀티파트 업로드에 필요한 AWS KMS 권한에 대한 자세한 내용은 멀티파트 업로드 API 및 권한 섹션을 참조하십시오.

SSE-KMS용 KMS 키에 대한 자세한 내용은 AWS KMS(SSE-KMS)를 사용한 서버 측 암호화 지정 섹션을 참조하세요.

주제

- AWS KMS keys
- 교차 계정 작업에 SSE-KMS 사용
- Amazon S3 버킷 키
- SSE-KMS 필요
- 암호화 컨텍스트
- AWS KMS 암호화된 객체에 대한 요청 전송
- 디렉터리 버킷의 SSE-KMS 암호화 감사
- 디렉터리 버킷의 새 객체 업로드에 AWS KMS(SSE-KMS)를 사용한 서버 측 암호화 지정

AWS KMS keys

SSE-KMS 구성은 버킷 수명 기간 동안 디렉터리 버킷당 고객 관리형 키 1개만 지원할 수 있습니다. AWS 관리형 키(`aws/s3`)는 지원되지 않습니다. 또한, SSE-KMS에 고객 관리형 키를 지정한 후에는 버킷의 SSE-KMS 구성을 위한 고객 관리형 키를 재정의할 수 없습니다.

다음과 같은 방법으로 버킷의 SSE-KMS 구성에 대해 지정한 고객 관리형 키를 식별할 수 있습니다.

- 응답에서 `x-amz-server-side-encryption-aws-kms-key-id`의 값을 찾기 위해 `HeadObject` API 작업을 요청합니다.

데이터에 새 고객 관리형 키를 사용하려면 최신 고객 관리형 키가 있는 새 디렉터리 버킷에 기존 객체를 복사하는 것이 좋습니다.

디렉터리 버킷에서 암호화를 위한 AWS KMS 고객 관리형 키를 지정할 때는 키 ID 또는 키 ARN만 사용하세요. KMS 키의 키 별칭 형식은 지원되지 않습니다.

SSE-KMS용 KMS 키에 대한 자세한 내용은 AWS KMS keys 섹션을 참조하세요.

교차 계정 작업에 SSE-KMS 사용

디렉터리 버킷에서 교차 계정 작업에 암호화를 사용하는 경우 다음 사항에 유의하세요.

- S3 객체에 대한 교차 계정 액세스 권한을 부여하는 경우 다른 계정에서 액세스할 수 있도록 고객 관리형 키 정책을 구성합니다.
- 고객 관리형 키를 지정하려면 정규화된 KMS 키 ARN을 사용해야 합니다.

Amazon S3 버킷 키

S3 버킷 키는 디렉터리 버킷의 GET 및 PUT 작업에 대해 항상 활성화되며 비활성화할 수 없습니다. CopyObject, UploadPartCopy, 배치 운영의 Copy 작업 또는 import 작업을 통해 SSE-KMS로 암호화된 객체를 범용 버킷에서 디렉터리 버킷으로, 디렉터리 버킷에서 범용 버킷으로 또는 디렉터리 버킷 간에 복사하는 경우 S3 버킷 키는 지원되지 않습니다. 이 경우 Amazon S3는 KMS 암호화 개체에 대한 사본 요청이 있을 때마다 AWS KMS를 직접적으로 호출합니다.

CopyObject 및 UploadPartCopy를 제외한 영역 엔드포인트(객체 수준) API 작업의 경우 지연 시간이 짧도록 CreateSession을 통해 요청을 인증하고 권한을 부여합니다. 버킷의 기본 암호화는 원하는 암호화 구성을 사용하고 CreateSession 요청 또는 PUT 객체 요청에서 버킷 기본 암호화를 재정의하지 않는 것이 좋습니다. 그러면 새 객체가 원하는 암호화 설정으로 자동 암호화됩니다. SSE-KMS로 디렉터리 버킷의 새 객체를 암호화하려면 KMS 키(특히 고객 관리형 키)를 사용하여 디렉터리 버킷의 기본 암호화 구성으로 SSE-KMS를 지정해야 합니다. 그런 다음 영역 엔드포인트 API 작업을 위한 세션이 생성되면 세션 중에 SSE-KMS 및 S3 버킷 키를 사용하여 새 객체가 자동으로 암호화되고 해독됩니다. 디렉터리 버킷의 암호화 재정의 동작에 대한 자세한 내용은 Specifying server-side encryption with AWS KMS for new object uploads를 참조하세요.

S3 버킷 키는 Amazon S3 내에서 제한된 기간 동안 사용되므로, Amazon S3가 암호화 작업을 완료하기 위해 AWS KMS에 요청할 필요성이 추가적으로 줄어듭니다. S3 버킷 키 사용에 대한 자세한 내용은 Amazon S3 버킷 키 및 Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감 섹션을 참조하세요.

SSE-KMS 필요

특정 디렉터리 버킷의 모든 객체에 대해 SSE-KMS를 요구하려면 버킷 정책을 사용하면 됩니다. 예를 들어, CreateSession API 작업을 사용하여 새 객체(PutObject, CopyObject, CreateMultipartUpload)를 업로드할 수 있는 권한을 부여하는 경우, CreateSession 요청에 SSE-KMS를 요청하는 x-amz-server-side-encryption-aws-kms-key-id 헤더가 포함되지 않

는다면 다음 버킷 정책은 모든 사람에 대한 업로드 객체 권한(s3express:CreateSession)을 거부합니다.

```
{
   "Version":"2012-10-17",
   "Id":"UploadObjectPolicy",
   "Statement":[{
         "Sid":"DenyObjectsThatAreNotSSEKMS",
         "Effect":"Deny",
         "Principal":"*",
         "Action":"s3express:CreateSession",
         "Resource":"arn:aws:s3express:region:account-id:bucket/bucket-base-name--zone-id--x-s3/*",
         "Condition":{
            "Null":{
               "s3express:x-amz-server-side-encryption-aws-kms-key-id":"true"
            }
         }
      }
   ]
}
```

버킷의 객체를 암호화하는 데 특정 AWS KMS key를 사용하도록 하려면 s3express:x-amz-server-side-encryption-aws-kms-key-id 조건 키를 사용합니다. arn:aws:kms:*region*:*acct-id*:key/*key-id* 형식의 Amazon 리소스 이름(ARN) 키를 사용해야 합니다. AWS Identity and Access Management은 문자열이 존재하는지 유효성을 검사하지 않습니다. Amazon S3가 객체 암호화에 사용하는 AWS KMS 키 ID는 정책의 AWS KMS 키 ID와 일치해야 합니다. 그렇지 않으면 Amazon S3는 요청을 거부합니다.

새 객체 업로드에 SSE-KMS를 사용하는 방법에 대한 자세한 내용은 디렉터리 버킷의 새 객체 업로드에 AWS KMS(SSE-KMS)를 사용한 서버 측 암호화 지정 섹션을 참조하세요.

디렉터리 버킷에 대한 특정 조건 키의 전체 목록을 보려면 IAM을 사용한 리전 엔드포인트 API 권한 부여 섹션을 참조하세요.

암호화 컨텍스트

디렉터리 버킷의 경우 암호화 컨텍스트는 데이터에 대한 컨텍스트 정보를 포함하는 키-값 페어 집합입니다. 추가 암호화 컨텍스트 값은 지원되지 않습니다. 암호화 컨텍스트에 대한 자세한 정보는 암호화 컨텍스트 섹션을 참조하십시오.

기본적으로 디렉터리 버킷에서 SSE-KMS를 사용하는 경우 Amazon S3는 버킷의 Amazon 리소스 이름(ARN)을 암호화 컨텍스트 페어로 사용합니다.

```
arn:aws:s3express:region:account-id:bucket/bucket-base-name--zone-id--x-s3
```

IAM 정책 또는 AWS KMS 키 정책에서 버킷 ARN을 암호화 컨텍스트로 사용하는지 확인하세요.

필요한 경우 CreateSession과 같은 영역 엔드포인트 API 요청의 `x-amz-server-side-encryption-context` 헤더를 사용하여 명시적 암호화 컨텍스트 페어를 제공할 수 있습니다. 이 헤더의 값은 암호화 컨텍스트를 키-값 페어로 포함하는 UTF-8 인코딩 JSON의 Base64로 인코딩된 문자열입니다. 디렉터리 버킷의 경우 암호화 컨텍스트는 기본 암호화 컨텍스트인 버킷의 Amazon 리소스 이름(ARN)과 일치해야 합니다. 또한, 암호화 컨텍스트는 암호화되지 않으므로 민감한 정보를 포함하지 않도록 해야 합니다.

암호화 컨텍스트를 사용하여 암호화 작업을 식별하고 분류할 수 있습니다. 또한, 기본 암호화 컨텍스트 ARN 값을 사용하면 어떤 디렉터리 버킷 ARN이 어떤 암호화 키와 함께 사용되었는지 확인하여 AWS CloudTrail에서 관련 요청을 추적할 수 있습니다.

CloudTrail 로그 파일 `requestParameters` 필드에서, 디렉터리 버킷에서 SSE-KMS를 사용하는 경우 암호화 컨텍스트 값은 버킷의 ARN입니다.

```
"encryptionContext": {
    "aws:s3express:arn": "arn:aws:s3:::arn:aws:s3express:region:account-
id:bucket/bucket-base-name--zone-id--x-s3"
}
```

또한, 디렉터리 버킷에서 SSE-KMS를 사용하는 객체 암호화의 경우 AWS KMS CloudTrail 이벤트가 객체 ARN 대신 버킷 ARN을 로깅합니다.

AWS KMS 암호화된 객체에 대한 요청 전송

디렉터리 버킷은 HTTPS(TLS)를 통해서만 액세스할 수 있습니다. 나아가 디렉터리 버킷은 AWS Signature Version 4(SigV4)를 사용하여 요청에 서명합니다. AWS KMS 암호화 객체의 요청 전송에 대한 자세한 내용은 AWS KMS 암호화된 객체에 대한 요청 전송 섹션을 참조하세요.

객체가 SSE-KMS를 사용하는 경우 `GET` 요청 및 `HEAD` 요청에 암호화 요청 헤더를 전송하면 안 됩니다. 그러면 HTTP 400 Bad Request 오류가 발생합니다.

디렉터리 버킷의 SSE-KMS 암호화 감사

AWS CloudTrail 로그를 사용하여 SSE-KMS 암호화 데이터의 AWS KMS 키 사용을 감사할 수 있습니다. GenerateDataKey 및 Decrypt와 같은 암호화 작업에 대한 인사이트를 얻을 수 있습니다. CloudTrail은 이벤트 이름, 사용자 이름, 이벤트 소스 등 검색 필터링을 위한 다양한 속성 값을 지원합니다.

주제

- 디렉터리 버킷의 새 객체 업로드에 AWS KMS(SSE-KMS)를 사용한 서버 측 암호화 지정

디렉터리 버킷의 새 객체 업로드에 AWS KMS(SSE-KMS)를 사용한 서버 측 암호화 지정

디렉터리 버킷의 경우 서버 측 암호화로 데이터를 암호화하려면 Amazon S3 관리형 키(SSE-S3)(기본값)를 사용하는 서버 측 암호화 또는 AWS Key Management Service(AWS KMS) 키를 사용하는 서버 측 암호화(SSE-KMS)를 사용할 수 있습니다. 버킷의 기본 암호화는 원하는 암호화 구성을 사용하고 `CreateSession` 요청 또는 PUT 객체 요청에서 버킷 기본 암호화를 재정의하지 않는 것이 좋습니다. 그러면 새 객체가 원하는 암호화 설정으로 자동 암호화됩니다. 디렉터리 버킷의 암호화 재정의 동작에 대한 자세한 내용은 Specifying server-side encryption with AWS KMS for new object uploads를 참조하세요.

모든 Amazon S3 버킷에는 기본적으로 암호화가 구성되어 있으며 S3 버킷에 업로드되는 신규 객체는 모두 저장 시 자동으로 암호화됩니다. Amazon S3 관리형 키(SSE-S3)를 사용한 서버 측 암호화가 Amazon S3 내 모든 버킷의 기본 암호화 구성입니다. 디렉터리 버킷에 다른 암호화 유형을 지정하려는 경우 AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 사용한 서버 측 암호화를 사용할 수 있습니다. SSE-KMS로 디렉터리 버킷의 새 객체를 암호화하려면 KMS 키(특히 고객 관리형 키)를 사용하여 디렉터리 버킷의 기본 암호화 구성으로 SSE-KMS를 지정해야 합니다. AWS 관리형 키(`aws/s3`)는 지원되지 않습니다. SSE-KMS 구성은 버킷 수명 기간 동안 디렉터리 버킷당 고객 관리형 키 1개만 지원할 수 있습니다. SSE-KMS에 고객 관리형 키를 지정한 후에는 버킷의 SSE-KMS 구성을 위한 고객 관리형 키를 재정의할 수 없습니다. 그런 다음 SSE-KMS를 사용하여 새 객체의 서버 측 암호화 설정을 지정할 때는 암호화 키가 디렉터리 버킷의 기본 암호화 구성에 지정한 것과 동일한 고객 관리형 키인지 확인해야 합니다. 데이터에 새 고객 관리형 키를 사용하려면 최신 고객 관리형 키가 있는 새 디렉터리 버킷에 기존 객체를 복사하는 것이 좋습니다.

새 객체를 업로드하거나 기존 객체를 복사할 때 암호화를 적용할 수 있습니다. 객체 암호화를 변경하면 새 객체가 생성되어 이전 객체를 대체합니다.

REST API 작업, AWS SDK 및 AWS Command Line Interface(AWS CLI)를 사용하여 SSE-KMS를 지정할 수 있습니다.

> **Note**
>
> - 디렉터리 버킷의 암호화 재지정 동작은 다음과 같습니다.
>   - REST API와 함께 CreateSession을 사용하여 CopyObject 및 UploadPartCopy를 제외한 영역 엔드포인트 API 요청을 인증하고 권한을 부여하는 경우, 이전에 버킷의 기본 암호화를 SSE-KMS로 지정한 경우에만 암호화 설정을 SSE-S3 또는 SSE-KMS로 재정의할 수 있습니다.
>   - AWS CLI 또는 AWS SDK와 함께 CreateSession을 사용하여 CopyObject 및 UploadPartCopy를 제외한 영역 엔드포인트 API 요청을 인증하고 권한을 부여하는 경우, 암호화 설정을 아예 재정의할 수 없습니다.
>   - CopyObject 요청을 할 때 이전에 SSE-KMS를 사용하여 버킷의 기본 암호화를 지정한 경우에만 암호화 설정을 SSE-S3 또는 SSE-KMS로 재정의할 수 있습니다. UploadPartCopy 요청을 할 때는 암호화 설정을 재정의할 수 없습니다.
> - Amazon S3에서 다중 리전 AWS KMS keys를 사용할 수 있습니다. 그러나 Amazon S3는 현재 다중 리전 키를 단일 리전 키인 것처럼 취급하며, 키의 다중 리전 기능을 사용하지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 다중 리전 키 사용을 참조하십시오.
> - 다른 계정에서 소유한 KMS 키를 사용하려면 먼저 해당 키에 대한 사용 권한이 있어야 합니다. KMS 키의 크로스 계정 권한에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서에서 다른 계정에서 사용할 수 있는 KMS 키 만들기를 참조하십시오.

REST API 사용

> **Note**
>
> 버킷의 수명 기간 동안 디렉터리 버킷당 1개의 고객 관리형 키만 지원됩니다. AWS 관리형 키(`aws/s3`)는 지원되지 않습니다. 고객 관리형 키를 사용하는 버킷의 기본 암호화 구성으로 SSE-KMS를 지정한 후에는 버킷의 SSE-KMS 구성에 대한 고객 관리형 키를 변경할 수 없습니다.

CopyObject 및 UploadPartCopy를 제외한 영역 엔드포인트(객체 수준) API 작업의 경우, 지연 시간이 짧도록 CreateSession을 통해 요청을 인증하고 권한을 부여합니다. 버킷의 기본 암호화는 원하는 암호화 구성을 사용하고 `CreateSession` 요청 또는 PUT 객체 요청에서 버킷 기본 암호화를 재정의하지 않는 것이 좋습니다. 그러면 새 객체가 원하는 암호화 설정으로 자동 암호화됩니다. SSE-KMS로 디렉

터리 버킷의 새 객체를 암호화하려면 KMS 키(특히 고객 관리형 키)를 사용하여 디렉터리 버킷의 기본 암호화 구성으로 SSE-KMS를 지정해야 합니다. 그런 다음 영역 엔드포인트 API 작업을 위한 세션이 생성되면 세션 중에 SSE-KMS 및 S3 버킷 키를 사용하여 새 객체가 자동으로 암호화되고 해독됩니다. 디렉터리 버킷의 암호화 재정의 동작에 대한 자세한 내용은 Specifying server-side encryption with AWS KMS for new object uploads를 참조하세요.

REST API를 사용하는 영역 엔드포인트 API 직접 호출(CopyObject 및 UploadPartCopy 제외)에서는 `CreateSession` 요청에서 암호화 설정(`x-amz-server-side-encryption`, `x-amz-server-side-encryption-aws-kms-key-id`, `x-amz-server-side-encryption-context`, `x-amz-server-side-encryption-bucket-key-enabled`)의 값을 재정의할 수 없습니다. 영역 엔드포인트 API 직접 호출에서 이러한 암호화 설정 값을 명시적으로 지정할 필요는 없으며, Amazon S3는 `CreateSession` 요청의 암호화 설정 값을 사용하여 디렉터리 버킷의 새 객체를 보호합니다.

> **Note**
>
> `CreateSession`의 경우 AWS CLI 또는 AWS SDK를 사용하면 세션 토큰이 자동으로 새로 고쳐져 세션이 만료될 때 서비스가 중단되지 않도록 합니다. AWS CLI 또는 AWS SDK는 `CreateSession` 요청에 버킷의 기본 암호화 구성을 사용합니다. `CreateSession` 요청의 암호화 설정 값을 재정의하는 것은 지원되지 않습니다. 또한, 영역 엔드포인트 API 직접 호출(CopyObject 및 UploadPartCopy 제외)에서는 `CreateSession` 요청의 암호화 설정 값을 재정의할 수 없습니다.

CopyObject의 경우 SSE-KMS로 디렉터리 버킷의 새 객체 사본을 암호화하려면 KMS 키(특히 고객 관리형 키)를 사용하여 디렉터리 버킷의 기본 암호화 구성으로 SSE-KMS를 지정해야 합니다. 그런 다음 SSE-KMS를 사용하여 새 객체 사본의 서버 측 암호화 설정을 지정할 때는 암호화 키가 디렉터리 버킷의 기본 암호화 구성에 지정한 것과 동일한 고객 관리형 키인지 확인해야 합니다. UploadPartCopy의 경우 SSE-KMS로 디렉터리 버킷의 새 객체 부분 사본을 암호화하려면 KMS 키(특히 고객 관리형 키)를 사용하여 디렉터리 버킷의 기본 암호화 구성으로 SSE-KMS를 지정해야 합니다. UploadPartCopy 요청 헤더에서 SSE-KMS를 사용하여 새 객체 부분 사본에 대한 서버 측 암호화 설정을 지정할 수 없습니다. 또한, CreateMultipartUpload 요청에서 제공하는 암호화 설정은 대상 버킷의 기본 암호화 구성과 일치해야 합니다.

주제

- SSE-KMS를 지원하는 Amazon S3 REST API 작업
- 암호화 컨텍스트(x-amz-server-side-encryption-context)
- AWS KMS 키 ID(x-amz-server-side-encryption-aws-kms-key-id)

- S3 버킷 키(x-amz-server-side-encryption-aws-bucket-key-enabled)

SSE-KMS를 지원하는 Amazon S3 REST API 작업

디렉터리 버킷의 다음 객체 수준 REST API 작업은 `x-amz-server-side-encryption`, `x-amz-server-side-encryption-aws-kms-key-id` 및 `x-amz-server-side-encryption-context` 요청 헤더를 수락합니다.

- CreateSession - 영역 엔드포인트(객체 수준) API 작업(CopyObject 및 UploadPartCopy 제외)을 사용하는 경우 이러한 요청 헤더를 지정할 수 있습니다.
- PutObject - PUT API 작업을 사용하여 데이터를 업로드할 때 이러한 요청 헤더를 지정할 수 있습니다.
- CopyObject - 객체를 복사할 때 소스 객체 및 대상 객체가 모두 있어야 합니다. `CopyObject` 작업을 사용하여 SSE-KMS 헤더를 전달할 경우 헤더가 대상 객체에만 적용됩니다.
- CreateMultipartUpload - 멀티파트 업로드 API 작업을 사용하여 대형 객체를 업로드할 때 이러한 헤더를 지정할 수 있습니다. `CreateMultipartUpload` 요청에서 이 헤더를 지정합니다.

객체가 서버 측 암호화를 사용하여 저장될 경우 다음 REST API 작업의 응답 헤더는 `x-amz-server-side-encryption` 헤더를 반환합니다.

- CreateSession
- PutObject
- CopyObject
- POST Object
- CreateMultipartUpload
- UploadPart
- UploadPartCopy
- CompleteMultipartUpload
- GetObject
- HeadObject

> **Important**
>
> - Transport Layer Security(TLS) 또는 Signature Version 4를 사용하여 요청을 수행하지 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:kms를 지정하면 Amazon S3 API는 명시적인 암호화 컨텍스트를 필요에 따라 x-amz-server-side-encryption-context 헤더에 제공할 수 있도록 지원합니다. 디렉터리 버킷의 경우 암호화 컨텍스트는 데이터에 대한 컨텍스트 정보를 포함하는 키-값 페어 집합입니다. 값은 기본 암호화 컨텍스트인 버킷 Amazon 리소스 이름(ARN)과 일치해야 합니다. 추가 암호화 컨텍스트 값은 지원되지 않습니다.

디렉터리 버킷의 암호화 컨텍스트에 대한 자세한 내용은 암호화 컨텍스트 섹션을 참조하세요. 암호화 컨텍스트에 대한 일반 내용은 AWS Key Management Service 개발자 안내서의 AWS Key Management Service 개념 - 암호화 컨텍스트를 참조하십시오.

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 관리형 키의 ID를 지정할 수 있습니다.

SSE-KMS 구성은 버킷 수명 기간 동안 디렉터리 버킷당 고객 관리형 키 1개만 지원할 수 있습니다. AWS 관리형 키(aws/s3)는 지원되지 않습니다. 또한, SSE-KMS에 고객 관리형 키를 지정한 후에는 버킷의 SSE-KMS 구성을 위한 고객 관리형 키를 재정의할 수 없습니다.

다음과 같은 방법으로 버킷의 SSE-KMS 구성에 대해 지정한 고객 관리형 키를 식별할 수 있습니다.

- 응답에서 x-amz-server-side-encryption-aws-kms-key-id의 값을 찾기 위해 HeadObject API 작업을 요청합니다.

데이터에 새 고객 관리형 키를 사용하려면 최신 고객 관리형 키가 있는 새 디렉터리 버킷에 기존 객체를 복사하는 것이 좋습니다.

For information about the encryption context in directory buckets, see AWS KMS keys.

S3 버킷 키(**x-amz-server-side-encryption-aws-bucket-key-enabled**)

S3 버킷 키는 디렉터리 버킷의 GET 및 PUT 작업에 대해 항상 활성화되며 비활성화할 수 없습니다. CopyObject, UploadPartCopy, 배치 운영의 Copy 작업 또는 import 작업을 통해 SSE-KMS로 암호화된 객체를 범용 버킷에서 디렉터리 버킷으로, 디렉터리 버킷에서 범용 버킷으로 또는 디렉터리 버킷 간에 복사하는 경우 S3 버킷 키는 지원되지 않습니다. 이 경우 Amazon S3는 KMS 암호화 개체에 대한 사본 요청이 있을 때마다 AWS KMS를 직접적으로 호출합니다. 디렉터리 버킷의 S3 버킷 키에 대한 자세한 내용은 암호화 컨텍스트 섹션을 참조하세요.

AWS CLI 사용

> **Note**
>
> `CreateSession`의 경우 AWS CLI를 사용하면 세션 토큰이 자동으로 새로 고쳐져 세션이 만료될 때 서비스가 중단되지 않도록 합니다. `CreateSession` 요청의 암호화 설정 값을 재정의하는 것은 지원되지 않습니다. 또한, 영역 엔드포인트 API 직접 호출(CopyObject 및 UploadPartCopy 제외)에서는 `CreateSession` 요청의 암호화 설정 값을 재정의할 수 없습니다.
>
> SSE-KMS로 디렉터리 버킷의 새 객체를 암호화하려면 KMS 키(특히 고객 관리형 키)를 사용하여 디렉터리 버킷의 기본 암호화 구성으로 SSE-KMS를 지정해야 합니다. 그런 다음 영역 엔드포인트 API 작업을 위한 세션이 생성되면 세션 중에 SSE-KMS 및 S3 버킷 키를 사용하여 새 객체가 자동으로 암호화되고 해독됩니다.

다음 예시 AWS CLI 명령을 사용하려면 *`user input placeholders`*를 실제 정보로 대체하세요.

새 객체를 업로드하거나 기존 객체를 복사할 때 AWS KMS 키를 사용하여 서버 측 암호화를 통해 데이터를 암호화하도록 지정할 수 있습니다. 이렇게 하려면 `put-bucket-encryption` 명령을 사용하여 디렉터리 버킷의 기본 암호화 구성을 SSE-KMS(`aws:kms`)로 설정하면 됩니다. 구체적으로 요청에 `--server-side-encryption aws:kms` 헤더를 추가합니다. `--ssekms-key-id` *`example-key-id`*를 사용하여, 생성해둔 고객 관리형 AWS KMS 키를 추가합니다. `--server-side-encryption aws:kms`를 지정하는 경우 고객 관리형 키의 AWS KMS 키 ID를 제공해야 합니다. 디렉터리 버킷은 AWS 관리형 키를 사용하지 않습니다. 예시 명령은 AWS CLI 사용 섹션을 참조하세요.

그 후 다음 명령으로 새 객체를 업로드하면 Amazon S3는 기본 암호화를 위한 버킷 설정을 사용하여 기본적으로 객체를 암호화합니다.

```
aws s3api put-object --bucket bucket-base-name--zone-id--x-s3 --key example-object-key
 --body filepath
```

영역 엔드포인트 API 작업 명령에 명시적으로 `-\-bucket-key-enabled`를 추가할 필요는 없습니다. S3 버킷 키는 디렉터리 버킷의 GET 및 PUT 작업에 대해 항상 활성화되며 비활성화할 수 없습니다. CopyObject, UploadPartCopy, 배치 운영의 Copy 작업 또는 import 작업을 통해 SSE-KMS로 암호화된 객체를 범용 버킷에서 디렉터리 버킷으로, 디렉터리 버킷에서 범용 버킷으로 또는 디렉터리 버킷 간에 복사하는 경우 S3 버킷 키는 지원되지 않습니다. 이 경우 Amazon S3는 KMS 암호화 개체에 대한 사본 요청이 있을 때마다 AWS KMS를 직접적으로 호출합니다.

소스 버킷(예: 범용 버킷)에서 새 버킷(예: 디렉터리 버킷)으로 객체를 복사하고 대상 객체에 SSE-KMS 암호화를 사용할 수 있습니다. 이렇게 하려면 `put-bucket-encryption` 명령을 사용하여 대상 버킷(예: 디렉터리 버킷)의 기본 암호화 구성을 SSE-KMS(`aws:kms`)로 설정합니다. 예시 명령은 AWS CLI 사용 섹션을 참조하세요. 그 후 다음 명령으로 객체를 복사하면 Amazon S3는 기본 암호화를 위한 버킷 설정을 사용하여 기본적으로 객체를 암호화합니다.

```
aws s3api copy-object --copy-source amzn-s3-demo-bucket/example-object-key --
bucket bucket-base-name--zone-id--x-s3 --key example-object-key
```

AWS SDK 사용

AWS SDK를 사용하는 경우 서버 측 암호화에 AWS KMS keys를 사용하도록 Amazon S3에 요청할 수 있습니다. 다음 예시에서는 Java 및 .NET용 AWS SDK와 함께 SSE-KMS를 사용하는 방법을 보여줍니다. 다른 SDK 사용에 대한 자세한 내용은 AWS 개발자 센터의 샘플 코드 및 라이브러리를 참조하세요.

> **Note**
>
> `CreateSession`의 경우 AWS SDK를 사용하면 세션 토큰이 자동으로 새로 고쳐져 세션이 만료될 때 서비스가 중단되지 않도록 합니다. `CreateSession` 요청의 암호화 설정 값을 재정의하는 것은 지원되지 않습니다. 또한, 영역 엔드포인트 API 직접 호출(CopyObject 및 UploadPartCopy 제외)에서는 `CreateSession` 요청의 암호화 설정 값을 재정의할 수 없습니다.
>
> SSE-KMS로 디렉터리 버킷의 새 객체를 암호화하려면 KMS 키(특히 고객 관리형 키)를 사용하여 디렉터리 버킷의 기본 암호화 구성으로 SSE-KMS를 지정해야 합니다. 그런 다음 영역 엔드포인트 API 작업을 위한 세션이 생성되면 세션 중에 SSE-KMS 및 S3 버킷 키를 사용하여 새 객체가 자동으로 암호화되고 해독됩니다.
>
> AWS SDK를 사용하여 디렉터리 버킷의 기본 암호화 구성을 SSE-KMS로 설정하는 방법에 대한 자세한 내용은 AWS SDK 사용 섹션을 참조하세요.

> **Important**
>
> Amazon S3에서 서버 측 암호화에 AWS KMS key을 사용하는 경우 대칭 암호화 KMS 키를 선택해야 합니다. Amazon S3는 대칭 암호화 KMS 키만 지원합니다. 이들 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 대칭 암호화 KMS 키를 참조하십시오.

고객 관리형 키 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 AWS KMS API 프로그래밍을 참조하십시오.

전송 중 암호화

디렉터리 버킷은 리전 및 영역 API 엔드포인트를 사용합니다. 사용하는 Amazon S3 API 작업에 따라 리전 엔드포인트 또는 영역 엔드포인트가 필요합니다. 게이트웨이 가상 프라이빗 클라우드(VPC) 엔드포인트를 통해 영역 및 리전 엔드포인트에 액세스할 수 있습니다. 게이트웨이 엔드포인트 사용에 따르는 추가 요금은 없습니다. 리전 및 영역 API 엔드포인트에 대한 자세한 내용은 디렉터리 버킷에 대한 네트워킹 섹션을 참조하세요.

데이터 삭제

Amazon S3 콘솔, AWS SDK, AWS Command Line Interface(AWS CLI) 또는 Amazon S3 REST API를 사용하여 디렉터리 버킷에서 직접 하나 이상의 객체를 삭제할 수 있습니다. 디렉터리 버킷에 있는 모든 객체로 인해 스토리지 비용이 발생하기 때문에 더 이상 필요하지 않은 객체를 삭제하는 것이 좋습니다.

디렉터리 버킷에 저장된 객체를 삭제하면 상위 디렉터리에 삭제되는 객체 외에 다른 객체가 없는 경우 상위 디렉터리도 재귀적으로 삭제됩니다.

> **Note**
>
> 다중 인증(MFA) 삭제 및 S3 버전 관리는 S3 Express One Zone에서 지원되지 않습니다.

요청 인증 및 권한 부여

기본적으로 디렉터리 버킷은 프라이빗이며 액세스 권한이 명시적으로 부여된 사용자만 액세스할 수 있습니다. 디렉터리 버킷의 액세스 제어 경계는 버킷 수준에서만 설정됩니다. 반면 범용 버킷의 액세스 제어 경계는 버킷, 접두사 또는 객체 태그 수준에서 설정할 수 있습니다. 이러한 차이는 디렉터리 버킷이 S3 Express One Zone 액세스에 대한 버킷 정책 또는 IAM ID 정책에 포함할 수 있는 유일한 리소스라는 것을 의미합니다.

Amazon S3 Express One Zone은 AWS Identity and Access Management(AWSIAM) 권한 부여와 세션 기반 권한 부여를 모두 지원합니다.

- S3 Express One Zone에서 리전 엔드포인트 API 작업(버킷 수준 작업, 즉 컨트롤 플레인 작업)을 사용하려면 세션 관리가 필요하지 않은 IAM 권한 부여 모델을 사용해야 합니다. 작업에 대한 권한은 개별적으로 부여됩니다. 자세한 내용은 IAM을 사용한 리전 엔드포인트 API 권한 부여 섹션을 참조하세요.
- `CopyObject` 및 `HeadBucket`을 제외한 영역 엔드포인트 API 작업(객체 수준 또는 데이터 플레인, 작업)을 사용하려면 `CreateSession` API 작업을 사용하여 지연 시간이 짧은 데이터 요청 권한 부여에 최적화된 세션을 만들고 관리합니다. 세션 토큰을 검색하고 사용하려면 ID 기반 정책 또는 버킷 정책에서 디렉터리 버킷에 대한 `s3express:CreateSession` 작업을 허용해야 합니다. 자세한 내용은 IAM을 사용한 리전 엔드포인트 API 권한 부여 섹션을 참조하세요. Amazon S3 콘솔에서, AWS Command Line Interface(AWS CLI)를 통해 또는 AWS SDK를 사용하여 S3 Express One Zone에 액세스하는 경우, S3 Express One Zone은 사용자를 대신하여 세션을 생성합니다.

`CreateSession` API 작업으로 새로운 세션 기반 메커니즘을 통해 요청을 인증하고 권한을 부여할 수 있습니다. `CreateSession`을 사용하여 지연 시간이 짧은 버킷 액세스를 제공하는 임시 보안 인증 정보를 요청할 수 있습니다. 이러한 임시 보안 인증 정보의 범위는 특정 디렉터리 버킷으로 지정됩니다.

`CreateSession` 작업을 수행하려면 최신 버전의 AWS SDK를 사용하거나 AWS Command Line Interface(AWS CLI)를 사용하는 것이 좋습니다. 지원되는 AWS SDK와 AWS CLI가 세션 설정, 새로 고침 및 종료를 사용자 대신 처리합니다.

영역(객체 수준) 작업에만 세션 토큰을 사용하여(`CopyObject` 및 `HeadBucket` 제외) 권한 부여와 관련된 지연 시간을 세션의 여러 요청에 분산합니다. 리전 엔드포인트 API 작업(버킷 수준 작업)의 경우 세션 관리가 필요하지 않은 IAM 인증을 사용합니다. 자세한 내용은 IAM을 사용한 리전 엔드포인트 API 권한 부여 및 `CreateSession`을 사용하여 영역 엔드포인트 API 작업 권한 부여(을)를 참조하세요.

API 작업의 권한 부여 및 인증 방법

다음 테이블에는 디렉터리 버킷 API 작업에 대한 인증 및 권한 부여 정보가 나열되어 있습니다. 테이블에는 각 API 작업에 대해 API 작업 이름, IAM 정책 작업, 엔드포인트 유형(리전 또는 영역), 권한 부여 메커니즘(IAM 또는 세션 기반)이 나와 있습니다. 또한, 교차 계정 액세스가 지원되는 곳도 나와 있습니다. 버킷 수준 작업에 대한 액세스는 IAM ID 기반 정책(사용자 또는 역할)에서만 부여할 수 있으며 버킷 정책으로는 부여할 수 없습니다.

API	엔드포인트 유형	IAM 작업	크로스 계정 액세스
CreateBucket	리전	s3express:CreateBucket	아니요
DeleteBucket	리전	s3express:DeleteBucket	아니요
ListDirectoryBuckets	리전	s3express:ListAllMyDirectoryBuckets	아니요
PutBucketPolicy	리전	s3express:PutBucketPolicy	아니요
GetBucketPolicy	리전	s3express:GetBucketPolicy	아니요
DeleteBucketPolicy	리전	s3express:DeleteBucketPolicy	아니요
CreateSession	영역	s3express:CreateSession	예
CopyObject	영역	s3express:CreateSession	예
DeleteObject	영역	s3express:CreateSession	예
DeleteObjects	영역	s3express:CreateSession	예
HeadObject	영역	s3express:CreateSession	예
PutObject	영역	s3express:CreateSession	예
RenameObject	영역	s3express:CreateSession	아니요
GetObjectAttributes	영역	s3express:CreateSession	예
ListObjectsV2	영역	s3express:CreateSession	예
HeadBucket	영역	s3express:CreateSession	예

API	엔드포인트 유형	IAM 작업	크로스 계정 액세스
CreateMultipartUpload	영역	s3express:CreateSession	예
UploadPart	영역	s3express:CreateSession	예
UploadPartCopy	영역	s3express:CreateSession	예
CompleteMultipartUpload	영역	s3express:CreateSession	예
AbortMultipartUpload	영역	s3express:CreateSession	예
ListParts	영역	s3express:CreateSession	예
ListMultipartUploads	영역	s3express:CreateSession	예
ListAccessPointsForDirectoryBuckets	영역	s3express:ListAccessPointsForDirectoryBuckets	예
GetAccessPointScope	영역	s3express:GetAccessPointScope	예
PutAccessPointScope	영역	s3express:PutAccessPointScope	예
DeleteAccessPointScope	영역	s3express:DeleteAccessPointScope	예

주제

- IAM을 사용한 리전 엔드포인트 API 권한 부여
- CreateSession을 사용하여 영역 엔드포인트 API 작업 권한 부여

IAM을 사용한 리전 엔드포인트 API 권한 부여

AWS Identity and Access Management(IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 있도록 지원하는 AWS 서비스입니다. IAM 관리자는 어떤 사용자가 디렉터리 버킷의 Amazon S3 리소스와 S3 Express One Zone 작업을 사용할 수 있도록 인증(로그인)되고 권한이 부여(권한 보유)될 수 있는지 제어합니다. IAM은 추가 요금 없이 사용할 수 있습니다.

기본적으로 사용자는 디렉터리 버킷에 대한 권한이 없습니다. 디렉터리 버킷에 대한 액세스 권한을 부여하려면 IAM을 사용하여 사용자, 그룹 또는 역할을 생성하고 해당 ID에 권한을 연결하면 됩니다. IAM에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

액세스 권한을 제공하려면 다음 방법을 사용하여 사용자, 그룹 또는 역할에 권한을 추가하면 됩니다.

- AWS IAM Identity Center 내의 사용자 및 그룹 - 권한 세트를 생성합니다. AWS IAM Identity Center 사용 설명서의 권한 세트 생성의 지침을 따릅니다.
- ID 제공업체를 통해 IAM에서 관리되는 사용자 - ID 페더레이션을 위한 역할을 생성합니다. IAM 사용 설명서의 서드 파티 자격 증명 공급자의 역할 만들기(페더레이션)의 지침을 따릅니다.
- IAM 역할 및 사용자 - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용 설명서에서 역할을 생성하여 IAM 사용자에게 권한 위임의 지침을 따르세요.

IAM for S3 Express One Zone에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- 보안 주체
- 리소스
- 디렉터리 버킷에 대한 작업
- 디렉터리 버킷에 대한 IAM ID 기반 정책
- 디렉터리 버킷에 대한 버킷 정책 예제

보안 주체

버킷에 대한 액세스 권한을 부여하는 리소스 기반 정책을 생성할 때, `Principal` 요소를 사용하여 해당 리소스에 대한 작업이나 작업을 요청할 수 있는 사람 또는 애플리케이션을 지정해야 합니다. 디렉터리 버킷 정책의 경우, 다음 보안 주체를 사용할 수 있습니다.

- AWS 계정
- IAM 사용자
- IAM 역할
- 페더레이션 사용자

자세한 내용은 IAM 사용 설명서에서 Principal 섹션을 참조하십시오.

리소스

디렉터리 버킷의 Amazon 리소스 이름(ARN)에는 `s3express` 네임스페이스, AWS 리전, AWS 계정 ID, AWS 영역 ID(가용 영역 또는 로컬 영역 ID)가 포함된 디렉터리 버킷 이름이 포함됩니다.

디렉터리 버킷에 액세스하고 작업을 수행하려면 다음 ARN 형식을 사용해야 합니다.

```
arn:aws:s3express:region:account-id:bucket/base-bucket-name--zone-id--x-s3
```

디렉터리 버킷의 액세스 포인트에 액세스하고 작업을 수행하려면 다음 ARN 형식을 사용해야 합니다.

```
arn:aws::s3express:region:account-id:accesspoint/accesspoint-basename--zone-id--xa-s3
```

ARN에 대한 자세한 내용은 IAM 사용 설명서의 Amazon Resource Names (ARNs) 섹션을 참조하세요. 리소스에 대한 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소: Resource를 참조하세요.

디렉터리 버킷에 대한 작업

IAM ID 기반 정책 또는 리소스 기반 정책에서 어떤 S3 작업을 허용 또는 거부할지 정의합니다. 작업은 특정 API 작업에 해당합니다. 디렉터리 버킷을 사용할 경우 S3 Express One Zone 네임스페이스를 사용하여 `s3express`라는 권한을 부여해야 합니다.

`s3express:CreateSession` 권한을 허용하면 `CreateSession` API 작업이 모든 영역 엔드포인트 API (객체 수준) 작업에 대한 임시 세션 토큰을 검색합니다. 이 세션 토큰은 다른 모든 영역 엔드포인트 API 작업에 사용되는 자격 증명을 반환합니다. 따라서 IAM 정책을 사용하여 영역 API 작업에 액세스 권한을 부여하지 않습니다. 대신 `CreateSession`은 모든 객체 수준 작업에 대한 액세스를 활성화합니다. 영역 API 작업 및 권한 목록은 요청 인증 및 권한 부여를 참조하세요.

`CreateSession` API 작업에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 CreateSession 섹션을 참조하세요.

IAM 정책 설명의 `Action` 요소에서는 다음 작업을 지정할 수 있습니다. 정책을 사용하여 AWS에서 작업할 수 있는 권한을 부여합니다. 정책에서 작업을 사용하면 일반적으로 이름이 같은 API 작업에 대한 액세스를 허용하거나 거부합니다. 그러나 경우에 따라 하나의 API 작업으로 둘 이상의 작업에 대한 액세스가 제어됩니다. 버킷 수준 작업에 대한 액세스는 IAM ID 기반 정책(사용자 또는 역할)에서만 부여할 수 있으며 버킷 정책으로는 부여할 수 없습니다.

액세스 포인트 정책을 구성하는 방법에 대한 자세한 내용은 디렉터리 버킷의 액세스 포인트를 사용하도록 IAM 정책 구성 섹션을 참조하세요.

자세한 내용은 Actions, resources, and condition keys for Amazon S3 Express를 참조하세요.

디렉터리 버킷에 대한 IAM ID 기반 정책

디렉터리 버킷을 생성하려면 먼저 AWS Identity and Access Management(IAM) 역할 또는 사용자에게 필요한 권한을 부여해야 합니다. 이 예시 정책은 `CreateSession` API 작업(영역 엔드포인트(객체 수준) API 작업에 사용) 및 모든 리전 엔드포인트(버킷 수준) API 작업에 대한 액세스를 허용합니다. 이 정책은 모든 디렉터리 버킷에서 `CreateSession` API 작업을 사용할 수 있도록 허용하지만, 리전 엔드포인트 API 작업은 지정된 디렉터리 버킷에만 사용할 수 있습니다. 이 정책 예를 사용하려면 *`user input placeholders`*를 실제 정보로 바꾸세요.

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Sid": "AllowAccessRegionalEndpointAPIs",
             "Effect": "Allow",
             "Action": [
                 "s3express:DeleteBucket",
                 "s3express:DeleteBucketPolicy",
                 "s3express:CreateBucket",
                 "s3express:PutBucketPolicy",
                 "s3express:GetBucketPolicy",
                 "s3express:ListAllMyDirectoryBuckets"
             ],

             "Resource": "arn:aws:s3express:region:account_id:bucket/bucket-base-name--zone-id--x-s3/*"
```

```
        },
        {
            "Sid": "AllowCreateSession",
            "Effect": "Allow",
            "Action": "s3express:CreateSession",
            "Resource": "*"
        }
    ]
}
```

디렉터리 버킷에 대한 버킷 정책 예제

이 섹션에서는 디렉터리 버킷 정책의 예제를 제공합니다. 이러한 정책을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

다음 예시 버킷 정책은 AWS 계정 ID인 *111122223333*이 지정된 디렉터리 버킷의 기본 ReadWrite 세션에서 CreateSession API 작업을 사용할 수 있도록 허용합니다. 이 정책은 영역 엔드포인트(객체 수준) API 작업에 대한 액세스 권한을 부여합니다.

Example - 기본 **ReadWrite** 세션에서 **CreateSession** 직접 호출을 허용하는 버킷 정책

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadWriteAccess",
            "Effect": "Allow",
            "Resource": "arn:aws:s3express:us-west-2:account-id:bucket/bucket-base-name--zone-id--x-s3",
            "Principal": {
                "AWS": [
                    "111122223333"
                ]
            },
            "Action": [
                "s3express:CreateSession"
            ]
        }
    ]
}
```

Example - **ReadOnly** 세션에서 **CreateSession** 직접 호출을 허용하는 버킷 정책

다음 예시 버킷 정책은 AWS 계정 ID인 *111122223333*이 CreateSession API 작업을 사용할 수 있도록 허용합니다. 이 정책은 ReadOnly 값과 함께 s3express:SessionMode 조건 키를 사용하여 읽기 전용 세션을 설정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadOnlyAccess",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": "s3express:CreateSession",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3express:SessionMode": "ReadOnly"
                }
            }
        }
    ]
}
```

Example - **CreateSession** 직접 호출에 대한 크로스 계정 액세스를 허용하는 버킷 정책

다음 예시 버킷 정책은 AWS 계정 ID인 *111122223333*이 AWS 계정 ID *444455556666*이 소유한 지정된 디렉터리 버킷에 CreateSession API 작업을 사용할 수 있도록 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CrossAccount",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
```

```
            "Action": [
                "s3express:CreateSession"
            ],
            "Resource": "arn:aws:s3express:us-west-2:444455556666:bucket/bucket-base-name--zone-id--x-s3"

        }
    ]
}
```

`CreateSession`을 사용하여 영역 엔드포인트 API 작업 권한 부여

`CopyObject` 및 `HeadBucket`을 제외한 영역 엔드포인트 API 작업(객체 수준 또는 데이터 플레인, 작업)을 사용하려면 `CreateSession` API 작업을 사용하여 지연 시간이 짧은 데이터 요청 권한 부여에 최적화된 세션을 만들고 관리합니다. 세션 토큰을 검색하고 사용하려면 ID 기반 정책 또는 버킷 정책에서 디렉터리 버킷에 대한 `s3express:CreateSession` 작업을 허용해야 합니다. 자세한 내용은 IAM을 사용한 리전 엔드포인트 API 권한 부여 섹션을 참조하세요. Amazon S3 콘솔에서, AWS Command Line Interface(AWS CLI)를 통해 또는 AWS SDK를 사용하여 S3 Express One Zone에 액세스하는 경우, S3 Express One Zone은 사용자를 대신하여 세션을 생성합니다.

Amazon S3 REST API를 사용하는 경우 `CreateSession` API 작업을 사용하여 액세스 키 ID, 비밀 액세스 키, 세션 토큰 및 만료 시간이 포함된 임시 보안 인증 정보를 얻을 수 있습니다. 임시 보안 인증 정보는 IAM 사용자 보안 인증 정보와 같은 장기 보안 인증 정보와 동일한 권한을 제공하지만 임시 보안 인증 정보에 세션 토큰을 포함해야 합니다.

세션 모드

세션 모드는 세션의 범위를 정의합니다. 버킷 정책에서 `s3express:SessionMode` 조건 키를 지정하여 `ReadWrite` 또는 `ReadOnly` 세션을 생성할 수 있는 사용자를 제어할 수 있습니다. `ReadWrite` 또는 `ReadOnly` 세션에 대한 자세한 내용은 Amazon S3 API 참조의 CreateSession에 대한 `x-amz-create-session-mode` 파라미터를 참조하세요. 생성할 버킷 정책에 대한 자세한 내용은 디렉터리 버킷에 대한 버킷 정책 예제 섹션을 참조하세요.

세션 토큰

임시 보안 인증 정보를 사용하여 직접 호출할 때 호출에 세션 토큰을 포함해야 합니다. 세션 토큰은 임시 보안 인증 정보와 함께 반환됩니다. 세션 토큰은 디렉터리 버킷으로 범위가 지정되며 보안 인증 정보가 유효하고 만료되지 않았다는 것을 확인하는 데 사용됩니다. 세션을 보호하기 위해 임시 보안 인증 정보는 5분 후에 만료됩니다.

`CopyObject` 및 `HeadBucket`

임시 보안 인증 정보는 특정 디렉터리 버킷으로 범위가 지정되며 지정된 디렉터리 버킷에 대한 모든 영역(객체 수준) 작업 API 직접 호출에 대해 자동으로 활성화됩니다. 다른 영역 엔드포인트 API 작업과 달리 `CopyObject` 및 `HeadBucket`은 `CreateSession` 인증을 사용하지 않습니다. 모든 `CopyObject` 및 `HeadBucket` 요청은 IAM 보안 인증 정보를 사용하여 인증 및 서명되어야 합니다. 하지만 다른 영역 엔드포인트 API 작업과 마찬가지로 `CopyObject` 및 `HeadBucket`은 여전히 `s3express:CreateSession`을 사용하여 권한이 부여됩니다.

자세한 내용은 Amazon Simple Storage Service API 참조에서 CreateSession를 참조하십시오.

디렉터리 버킷의 보안 모범 사례

디렉터리 버킷을 사용할 때 여러 가지 보안 기능을 고려해야 합니다. 다음 모범 사례는 일반적인 지침이며 완벽한 보안 솔루션을 나타내지는 않습니다. 이러한 모범 사례는 환경에 적절하지 않거나 충분하지 않을 수 있으므로 지침이라기보다는 권장 사항으로만 사용해 주십시오.

기본 퍼블릭 액세스 차단 및 객체 소유권 설정

디렉터리 버킷은 S3 퍼블릭 액세스 차단과 S3 객체 소유권을 지원합니다. 이러한 S3 기능은 버킷 및 객체에 대한 액세스 감사 및 관리에 사용됩니다.

기본적으로 디렉터리 버킷에 모든 퍼블릭 액세스 차단 설정이 활성화되어 있습니다. 또한 S3 객체 소유권은 버킷 소유자 적용으로 설정되므로 액세스 제어 목록(ACL)이 비활성화되어 있습니다. 이 설정은 수정할 수 없습니다. 이러한 기능에 대한 자세한 내용은 the section called “퍼블릭 액세스 차단” 및 the section called “객체 소유권 제어” 섹션을 참조하세요.

> **Note**
>
> 디렉터리 버킷에 저장된 객체에는 액세스 권한을 부여할 수 없습니다. 디렉터리 버킷에만 액세스 권한을 부여할 수 있습니다. S3 Express One Zone의 권한 부여 모델은 Amazon S3의 권한 부여 모델과 다릅니다. 자세한 내용은 `CreateSession`을 사용하여 영역 엔드포인트 API 작업 권한 부여 섹션을 참조하세요.

인증 및 권한 부여

디렉터리 버킷의 인증 및 권한 부여 메커니즘은 영역 엔드포인트 API 작업에 요청하는지 아니면 리전 엔드포인트 API 작업에 요청하는지에 따라 다릅니다. 영역 API 작업은 객체 수준(데이터 영역) 작업입니다. 리전 API 작업은 버킷 수준(컨트롤 플레인) 작업입니다.

지연 시간을 최소화하도록 최적화된 새로운 세션 기반 메커니즘을 통해 영역 엔드포인트 API 작업에 대한 요청을 인증하고 권한을 부여하게 됩니다. 세션 기반 인증에서 AWS SDK는 `CreateSession` API 작업을 사용하여 디렉터리 버킷에 대한 지연 시간이 짧은 액세스를 제공하는 임시 자격 증명을 요청합니다. 이러한 임시 보안 인증 정보의 범위는 특정 디렉터리 버킷으로 지정되며 5분 후 만료됩니다. 이러한 임시 자격 증명을 사용하여 영역(객체 수준) API 직접 호출에 서명할 수 있습니다. 자세한 내용은 `CreateSession`을 사용하여 영역 엔드포인트 API 작업 권한 부여 섹션을 참조하세요.

디렉터리 버킷 관리를 위한 자격 증명으로 서명 요청

자격 증명을 사용하여 AWS Signature Version 4를 사용하여 영역 엔드포인트(객체 수준) API 요청에 서명합니다. 서비스 이름은 `s3express`입니다. 요청에 서명할 때는 `CreateSession`에서 반환된 비밀 키를 사용하고 세션 토큰도 `x-amzn-s3session-token header`와 함께 제공하세요. 자세한 내용은 CreateSession 섹션을 참조하세요.

지원되는 AWS SDK는 사용자를 대신하여 자격 증명 및 서명 작업을 관리합니다. AWS SDK를 사용하여 자격 증명을 새로 고치고 요청에 서명하도록 하는 것이 좋습니다.

IAM 자격 증명으로 요청에 서명

모든 리전(버킷 수준) API 직접 호출은 임시 세션 자격 증명이 아닌 AWS Identity and Access Management(IAM) 자격 증명으로 인증 및 서명되어야 합니다. IAM 자격 증명은 IAM ID의 액세스 키 ID 및 보안 액세스 키로 구성됩니다. 또한 모든 `CopyObject` 및 `HeadBucket` 요청은 IAM 자격 증명을 사용하여 인증 및 서명되어야 합니다.

영역(객체 수준) 작업 직접 호출의 지연 시간을 최소화하려면 `CreateSession`을 직접 호출하여 얻은 자격 증명을 사용하여 요청에 서명하는 것이 좋습니다. 단, `CopyObject` 및 `HeadBucket`에 대한 요청은 예외입니다.

AWS CloudTrail 사용

AWS CloudTrail은 Amazon S3에서 사용자, 역할 또는 AWS 서비스가 수행한 작업에 대한 레코드를 제공합니다. CloudTrail에서 수집한 정보를 사용하여 다음을 확인할 수 있습니다.

- Amazon S3에 대한 요청
- 요청을 보낸 IP 주소
- 요청한 사람
- 요청을 한 시간
- 요청에 대한 추가 세부 정보

AWS 계정을 설정하면 CloudTrail 관리 이벤트가 기본적으로 활성화됩니다. 다음과 같은 리전 엔드포인트 API 작업(버킷 수준 또는 컨트롤 플레인, API 작업)이 CloudTrail에 로깅됩니다.

- CreateBucket
- DeleteBucket
- DeleteBucketPolicy
- PutBucketPolicy
- GetBucketPolicy
- ListDirectoryBuckets
- ListMultipartUploads
- PutBucketEncryption
- GetBucketEncryption
- DeleteBucketEncryption

> **Note**
>
> `ListMultipartUploads`는 영역 엔드포인트 API 작업입니다. 하지만 관리 이벤트로 CloudTrail에 로깅됩니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 ListMultipartUploads를 참조하십시오.

기본적으로 CloudTrail 추적은 데이터 이벤트를 로깅하지 않지만 지정한 디렉터리 버킷에 대한 데이터 이벤트를 로깅하거나 AWS 계정의 모든 디렉터리 버킷에 대한 데이터 이벤트를 로깅하도록 추적을 구성할 수 있습니다. 다음과 같은 영역 엔드포인트 API 작업(객체 수준 또는 데이터 영역, API 작업)이 CloudTrail에 로깅됩니다.

- AbortMultipartUpload
- CompleteMultipartUpload
- CreateSession
- CopyObject
- CreateMultipartUpload
- DeleteObject
- DeleteObjects

- GetObject
- GetObjectAttributes
- HeadBucket
- HeadObject
- ListObjectsV2
- ListParts
- PutObject
- UploadPart
- UploadPartCopy

디렉터리 버킷에서 AWS CloudTrail을 사용하는 방법에 대한 자세한 내용은 AWS CloudTrail로 디렉터리 버킷에 로그인을 참조하세요.

AWS 모니터링 도구를 사용하여 모니터링 구현

모니터링은 Amazon S3 및 AWS 솔루션의 안정성, 보안, 가용성 및 성능을 유지하는 데 중요한 부분입니다. AWS에서는 Amazon S3 및 다른 AWS 서비스를 모니터링할 수 있는 여러 가지 도구와 서비스를 제공합니다. 예를 들어 Amazon S3에 대한 Amazon CloudWatch 지표(특히 `BucketSizeBytes` 및 `NumberOfObjects` 스토리지 지표)를 모니터링할 수 있습니다.

디렉터리 버킷에 저장된 객체는 Amazon S3의 `BucketSizeBytes` 및 `NumberOfObjects` 스토리지 지표에 반영되지 않습니다. 그러나 디렉터리 버킷에 대해서는 `BucketSizeBytes` 및 `NumberOfObjects` 스토리지 지표가 지원됩니다. 선택한 지표를 보려면 `StorageType` 차원을 지정하여 Amazon S3 스토리지 클래스를 구분할 수 있습니다. 자세한 내용은 Amazon CloudWatch를 사용한 지표 모니터링 섹션을 참조하세요.

자세한 내용은 Amazon CloudWatch를 사용한 지표 모니터링 및 Amazon S3의 로깅 및 모니터링(을)를 참조하세요.

액세스 포인트가 있는 디렉터리 버킷에서 공유 데이터세트 액세스 관리

Amazon S3 Access Points는 Amazon S3의 공유 데이터세트에 대한 대규모 데이터 액세스 관리를 간소화합니다. 액세스 포인트는 액세스 포인트를 통해 이루어진 모든 요청에 대해 고유한 권한 및 네트워

크 제어를 적용하기 위해 만드는 고유한 호스트 이름입니다. 버킷당 수백 개의 액세스 포인트를 생성할 수 있으며, 각 액세스 포인트에는 각 애플리케이션에 대해 사용자 지정된 고유한 이름과 권한이 있습니다. 각 액세스 포인트는 기본 버킷에 연결된 버킷 정책과 함께 작동합니다.

디렉터리 버킷에서 액세스 포인트 이름은 사용자가 제공하는 기본 이름, 디렉터리 버킷 위치의 영역 ID(AWS 가용 영역 또는 로컬 영역), 그리고 `--xa-s3`으로 구성됩니다. 예를 들어 *`accesspointname`*`--`*`zoneID`*`--xa-s3`입니다. 액세스 포인트를 만든 후에는 이름이나 영역 ID를 변경할 수 없습니다.

디렉터리 버킷에 대한 액세스 포인트를 사용하면 액세스 포인트 범위를 사용하여 특정 접두사 또는 API 작업에 대한 액세스를 제한할 수 있습니다. 원하는 양의 접두사를 지정할 수 있지만 모든 접두사의 총 문자 길이는 256바이트 미만이어야 합니다.

가상 프라이빗 클라우드(VPC)의 요청만 수락하도록 액세스 포인트를 구성할 수 있습니다. 이렇게 하면 프라이빗 네트워크에 대한 Amazon S3 데이터 액세스가 제한됩니다.

이 섹션에서는 디렉터리 버킷에 액세스 포인트를 사용하는 방법을 설명합니다. 디렉터리 버킷에 대한 자세한 내용은 디렉터리 버킷 작업 섹션을 참조하세요.

주제

- 디렉터리 버킷의 액세스 포인트 이름 지정 규칙, 제한 및 한계
- 디렉터리 버킷의 액세스 포인트 참조
- 디렉터리 버킷의 액세스 포인트에 대한 객체 작업
- 디렉터리 버킷의 액세스 포인트를 사용하도록 IAM 정책 구성
- 디렉터리 버킷의 액세스 포인트 모니터링 및 로깅
- 디렉터리 버킷의 액세스 포인트 만들기
- 디렉터리 버킷의 액세스 포인트 관리

디렉터리 버킷의 액세스 포인트 이름 지정 규칙, 제한 및 한계

액세스 포인트는 Amazon S3의 공유 데이터 세트에 대한 대규모 데이터 액세스 관리를 간소화합니다. 다음 주제에서는 액세스 포인트 이름 지정 규칙, 제한 및 한계에 대한 정보를 제공합니다.

주제

- 디렉터리 버킷의 액세스 포인트 이름 지정 규칙
- 디렉터리 버킷의 액세스 포인트에 대한 제한 및 한계

디렉터리 버킷의 액세스 포인트 이름 지정 규칙

액세스 포인트는 버킷이 있는 곳과 동일한 영역에 만들어야 합니다. 액세스 포인트 이름은 영역 내에서 고유해야 합니다.

액세스 포인트 이름은 DNS를 준수해야 하며 다음과 같은 조건을 만족해야 합니다.

- 숫자 또는 소문자로 시작해야 함
- 제공하는 기본 이름은 3~50자여야 합니다.
- 하이픈(-)으로 시작하거나 끝날 수 없음
- 밑줄(_), 대문자, 공백 또는 마침표(.)를 포함할 수 없음
- 접미사 *zoneid*`--xa--s3`으로 끝나야 합니다.

디렉터리 버킷의 액세스 포인트에 대한 제한 및 한계

디렉터리 버킷의 액세스 포인트에는 다음과 같은 제한 및 한계가 있습니다.

- 각 액세스 포인트는 하나의 디렉터리 버킷에 연결됩니다. 액세스 포인트를 만든 후에는 다른 버킷과 연결할 수 없습니다. 그러나 액세스 포인트를 삭제한 다음 동일한 이름의 새 액세스 포인트를 만들어 다른 버킷에 연결할 수 있습니다.
- 액세스 포인트를 생성한 후에는 해당 가상 프라이빗 클라우드(VPC) 구성을 변경할 수 없습니다.
- 액세스 포인트 정책은 크기가 20KB로 제한됩니다.
- 액세스 포인트 범위 접두사는 총크기가 256바이트로 제한됩니다.
- AWS 리전마다 AWS 계정당 최대 1만 개의 액세스 포인트를 생성할 수 있습니다. 단일 리전에서 단일 계정에 대해 10,000개 이상의 액세스 포인트가 필요한 경우 서비스 할당량 증가를 요청할 수 있습니다. 서비스 할당량 및 증가 요청에 대한 자세한 내용은 AWS 일반 참조의 AWS 서비스 할당량을 참조하십시오.
- 액세스 포인트를 사용하면 객체에 대한 작업만 수행할 수 있습니다. 액세스 포인트를 사용하여 버킷 수정 또는 삭제와 같은 Amazon S3 버킷 작업을 수행할 수 없습니다. 지원되는 작업의 전체 목록은 디렉터리 버킷의 액세스 포인트에 대한 객체 작업 섹션을 참조하세요.
- 이름, 액세스 포인트 별칭 또는 가상 호스팅 방식 URI를 사용하여 액세스 포인트를 참조할 수 있습니다. ARN을 사용하여 액세스 포인트를 지정할 수는 없습니다. 자세한 내용은 디렉터리 버킷의 액세스 포인트 참조 섹션을 참조하세요.
- 액세스 포인트 기능(예: `PutAccessPointPolicy` 및 `GetAccessPointPolicy`)을 제어하는 API 작업에는 액세스 포인트를 소유한 AWS 계정을 지정해야 합니다.

- REST API를 사용하여 액세스 포인트에 요청할 때는 AWS 서명 버전 4를 사용해야 합니다. 요청 인증에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 요청 인증(AWS 서명 버전 4)을 참조하십시오.
- 액세스 포인트는 HTTPS를 통한 액세스만 지원합니다. Amazon S3는 HTTP를 통해 이루어진 모든 요청에 대해 자동으로 HTTP 리디렉션으로 응답하여 요청을 HTTPS로 업그레이드합니다.
- 액세스 포인트는 익명 액세스를 지원하지 않습니다.
- 다른 계정(교차 계정 액세스 포인트)이 소유한 버킷에 대한 액세스 포인트를 만드는 경우 버킷 소유자가 버킷에 액세스할 수 있는 권한을 부여할 때까지 교차 계정 액세스 포인트는 데이터에 대한 액세스 권한을 부여하지 않습니다. 버킷 소유자는 항상 데이터 액세스에 대한 최종 제어 권한을 보유하며, 크로스 계정 액세스 포인트로부터의 요청을 승인하도록 버킷 정책을 업데이트해야 합니다. 버킷 정책 예시를 보려면 디렉터리 버킷의 액세스 포인트를 사용하도록 IAM 정책 구성 섹션을 참조하십시오.

디렉터리 버킷의 액세스 포인트 참조

액세스 포인트를 만든 후 엔드포인트로 사용하여 객체 작업을 수행할 수 있습니다. 디렉터리 버킷의 액세스 포인트에서 액세스 포인트 별칭은 액세스 포인트 이름과 동일합니다. 모든 데이터 작업에 버킷 이름 대신 액세스 포인트 이름을 사용할 수 있습니다. 지원되는 작업 목록은 디렉터리 버킷의 액세스 포인트에 대한 객체 작업 섹션을 참조하세요.

가상 호스팅 방식 URI를 통한 액세스 포인트 참조

액세스 포인트는 가상 호스트 스타일의 주소 지정만 지원합니다. 액세스 포인트는 디렉터리 버킷 엔드포인트와 동일한 형식을 사용합니다. 자세한 내용은 디렉터리 버킷에 대한 리전 및 영역 엔드포인트 섹션을 참조하세요.

S3 액세스 포인트는 HTTP를 통한 액세스를 지원하지 않습니다. 액세스 포인트는 HTTPS를 통한 보안 액세스만 지원합니다.

디렉터리 버킷의 액세스 포인트에 대한 객체 작업

액세스 포인트를 사용하여 다음 S3 데이터 작업을 통해 객체에 액세스할 수 있습니다.

- AbortMultipartUpload
- CompleteMultipartUpload
- CopyObject

- CreateMultipartUpload
- CreateSession
- DeleteObject
- DeleteObjects
- GetObject
- GetObjectAttributes
- HeadBucket
- HeadObject
- ListMultipartUploads
- ListObjectsV2
- ListParts
- PutObject
- UploadPart
- UploadPartCopy

디렉터리 버킷의 액세스 포인트를 사용하도록 IAM 정책 구성

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 정책 모두에서 요청을 허용해야 합니다.

> **Important**
>
> 디렉터리 버킷에 액세스 포인트를 추가해도 버킷 이름을 통해 버킷에 직접 액세스할 때 버킷의 동작이 변경되지 않습니다. 버킷에 대한 모든 기존 작업은 이전과 같이 계속 작동합니다. 액세스 포인트 정책 또는 액세스 포인트 범위에 포함하는 제한은 해당 액세스 포인트를 통해 이루어진 요청에만 적용됩니다.

IAM 리소스 정책을 사용할 때는 AWS Identity and Access Management Access Analyzer의 보안 경고, 오류, 일반 경고 및 제안 사항을 해결한 후 정책을 저장해야 합니다. IAM Access Analyzer는 정책 확인은 실행하여 IAM 정책 문법 및 모범 사례에 대해 정책을 검증합니다. 이러한 확인은 결과를 생성하고 보안 모범 사례를 준수하고 작동하는 정책을 작성하는 데 도움이 되는 권장 사항을 제공합니다.

IAM Access Analyzer를 사용한 정책 검증에 대한 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer 정책 검증을 참조하십시오. IAM Access Analyzer에서 반환된 경고, 오류 및 제안 사항 목록을 보려면 IAM Access Analyzer 정책 확인 참조를 참조하십시오.

디렉터리 버킷의 액세스 포인트 정책 예제

다음 액세스 포인트 정책은 디렉터리 버킷에 대한 요청을 제어하는 방법을 보여줍니다. 액세스 포인트 정책에는 ARN 또는 액세스 포인트 ARN이 필요합니다. 액세스 포인트 별칭은 정책에서 지원되지 않습니다. 다음은 액세스 포인트 ARN의 예입니다.

```
arn:aws:s3express:region:account-id:accesspoint/myaccesspoint--zoneID--xa-s3
```

액세스 포인트의 세부 정보에서 액세스 포인트 ARN을 볼 수 있습니다. 자세한 내용은 디렉터리 버킷의 액세스 포인트에 대한 세부 정보 보기 섹션을 참조하세요.

> **Note**
>
> 액세스 포인트 정책에 부여된 권한은 기본 버킷이 동일한 액세스를 허용하는 경우에만 유효합니다. 다음 두 가지 방법으로 이 작업을 수행할 수 있습니다.
>
> 1. (권장) 액세스 포인트에 액세스 제어 위임에 설명된 대로 버킷의 액세스 제어를 액세스 포인트에 위임합니다.
> 2. 액세스 포인트 정책에 포함된 동일한 권한을 기본 버킷의 정책에 추가합니다.

Example 1 - 액세스 포인트를 VPC 네트워크 오리진으로 제한하는 서비스 제어 정책

다음 서비스 제어 정책에서는 가상 프라이빗 클라우드(VPC) 네트워크 오리진을 사용하여 모든 새 액세스 포인트를 만들어야 합니다. 이 정책을 적용하면 조직의 사용자가 인터넷에서 액세스할 수 있는 액세스 포인트를 만들 수 없습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Deny",
        "Action": "s3express:CreateAccessPoint",
```

```
        "Resource": "*",
        "Condition": {
            "StringNotEquals": {
                "s3express:AccessPointNetworkOrigin": "VPC"
            }
        }
    }
  ]
}
```

Example 2 - 버킷 액세스를 VPC 네트워크 오리진이 있는 액세스 포인트로 제한하는 액세스 포인트 정책

다음 액세스 포인트 정책은 버킷 *amzn-s3-demo-bucket--zoneID--x-s3*에 대한 모든 액세스를 VPC 네트워킹 오리진이 있는 액세스 포인트로 제한합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Principal": "*",
            "Action": "s3express:CreateSession",
            "Effect": "Deny",
            "Resource": "arn:aws:s3express:region:111122223333:bucket/amzn-s3-demo-
bucket--zoneID--x-s3",
            "Condition": {
                "StringNotEqualsIfExists": {
                    "s3express:AccessPointNetworkOrigin": "VPC"
                }
            }
        }
    ]
}
```

조건 키

디렉터리 버킷의 액세스 포인트에는 리소스에 대한 액세스를 제어하기 위해 IAM 정책에서 사용할 수 있는 조건 키가 있습니다. 다음 조건 키는 IAM 정책의 일부만 나타냅니다. 정책 예제를 모두 확인하려면 디렉터리 버킷의 액세스 포인트 정책 예제, 액세스 포인트에 액세스 제어 위임, 크로스 계정 액세스 포인트에 대한 권한 부여 페이지를 참조하십시오.

s3express:DataAccessPointArn

이 예제에서는 액세스 포인트의 Amazon 리소스 이름(ARN)을 기준으로 액세스를 필터링하고 *##* 리전에 있는 AWS 계정 *111122223333*의 모든 액세스 포인트와 일치시키는 방법을 보여줍니다.

```
"Condition" : {
    "StringLike": {
        "s3express:DataAccessPointArn":
 "arn:aws:s3express:region:111122223333:accesspoint/*"
    }
}
```

s3express:DataAccessPointAccount

이 예제는 액세스 포인트 소유자의 계정 ID와 일치시키는 데 사용할 수 있는 문자열 연산자를 보여줍니다. 다음 예제는 AWS 계정 *111122223333*가 소유한 모든 액세스 포인트와 일치합니다.

```
"Condition" : {
    "StringEquals": {
        "s3express:DataAccessPointAccount": "111122223333"
    }
}
```

s3express:AccessPointNetworkOrigin

이 예제는 네트워크 오리진 Internet 또는 VPC와 일치시키는 데 사용할 수 있는 문자열 연산자를 보여줍니다. 다음 예제에서는 오리진이 VPC인 액세스 포인트와만 일치합니다.

```
"Condition" : {
    "StringEquals": {
        "s3express:AccessPointNetworkOrigin": "VPC"
    }
}
```

s3express:Permissions

s3express:Permissions를 사용하여 액세스 포인트 범위 내에서 특정 API 작업에 대한 액세스를 제한할 수 있습니다. 지원되는 API 작업은 다음과 같습니다.

- PutObject
- GetObject
- DeleteObject

- ListBucket(ListObjectsV2의 경우 필수)
- GetObjectAttributes
- AbortMultipartUpload
- ListBucketMultipartUploads
- ListMultipartUploadParts

> **Note**
>
> 다중 값 조건 키를 사용하는 경우 Allow 문과 함께 ForAllValues를 사용하고 Deny 문과 함께 ForAnyValue를 사용하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의Multivalued context keys를 참조하세요.

Amazon S3에서 조건 키를 사용하는 방법에 대한 자세한 내용은 서비스 승인 참조에서 Amazon S3에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

S3 리소스 유형별로 S3 API 작업에 필요한 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

액세스 포인트에 액세스 제어 위임

버킷 정책의 액세스 제어를 액세스 포인트 정책에 위임할 수 있습니다. 다음 예제 버킷 정책은 버킷 소유자의 계정이 소유한 모든 액세스 포인트에 대한 전체 액세스를 허용합니다. 정책을 적용한 후에는 이 버킷에 대한 모든 액세스는 액세스 포인트 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",
        "Condition": {
            "StringEquals" : { "s3express:DataAccessPointAccount" : "Bucket owner's
 account ID" }
```

```
        }
    }]
}
```

크로스 계정 액세스 포인트에 대한 권한 부여

다른 계정이 소유한 버킷에 액세스 포인트를 생성하려면 먼저 버킷 이름과 계정 소유자 ID를 지정하여 액세스 포인트를 만들어야 합니다. 그런 다음, 액세스 포인트로부터의 요청을 승인하도록 버킷 정책을 버킷 소유자가 업데이트해야 합니다. 액세스 포인트를 생성하는 것은 액세스 포인트가 버킷 콘텐츠에 대한 액세스를 제공하지 않는다는 점에서 DNS CNAME을 생성하는 것과 유사합니다. 모든 버킷 액세스는 버킷 정책에 의해 제어됩니다. 다음 예제 버킷 정책은 신뢰할 수 있는 AWS 계정이 소유한 액세스 포인트에서 버킷의 `GET` 및 `LIST` 요청을 허용합니다.

*`Bucket ARN`*을 버킷의 ARN으로 바꿉니다.

Example 다른 AWS 계정에 권한을 위임하는 버킷 정책

```
{
    "Version": "2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS": "*" },
        "Action" : "s3express:CreateSession",
        "Resource" : [ "Bucket ARN" ],
        "Condition": {
            "StringEquals" : {
                "s3express:DataAccessPointAccount": "Access point owner's account ID"
            },
            "ForAllValues:StringEquals": {
                "s3express:Permissions": [
                    "GetObject",
                    "ListBucket"
                ]
            }
        }
    }]
}
```

디렉터리 버킷의 액세스 포인트 모니터링 및 로깅

AWS CloudTrail을 사용하여 액세스 포인트를 통해 이루어진 요청과 액세스 포인트를 관리하는 API에 대한 요청(예: `CreateAccessPoint` 및 `GetAccessPointPolicy`,)을 로깅할 수 있습니다. 액세스 포인트를 통해 이루어진 요청에 대한 CloudTrail 로그 항목에는 로그의 `resources` 섹션에 액세스 포인트 ARN(액세스 포인트 이름 포함)이 포함됩니다.

예를 들어, 다음과 같은 구성이 있다고 가정합니다.

- `my-image.jpg`라는 객체를 포함하는 `region` 리전의 *`amzn-s3-demo-bucket--zone-id`*`--x-s3`이라는 버킷
- *`amzn-s3-demo-bucket--zone-id`*`--x-s3`과 연결된 이름이 `my-bucket-ap--zoneID--xa-s3`인 액세스 포인트
- `123456789012`의 AWS 계정 ID

다음 예제에서는 이전 구성에 대한 CloudTrail 로그 항목의 `resources` 섹션을 보여줍니다.

```
"resources": [
        {"type": "AWS::S3Express::Object",

            "ARN": "arn:aws:s3express-region:123456789012:bucket/amzn-s3-demo-
bucket--zone-id--x-s3/my-image.jpg"
        },
        {"accountId": "c",
            "type": "AWS::S3Express::DirectoryBucket",
            "ARN": "arn:aws::s3express:region:123456789012:bucket/amzn-s3-demo-
bucket--zone-id--x-s3"
        },
        {"accountId": "123456789012",
            "type": "AWS::S3::AccessPoint",
            "ARN": "arn:aws:s3express:region:123456789012:accesspoint/my-bucket-ap--
zoneID--xa-s3"
        }
    ]
```

AWS CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 AWS CloudTrail이란 무엇인가요?를 참조하세요.

디렉터리 버킷의 액세스 포인트 만들기

디렉터리 버킷과 마찬가지로 가용 영역 또는 전용 로컬 영역에서 액세스 포인트를 생성할 수 있습니다. 액세스 포인트는 연결된 디렉터리 버킷과 동일한 영역에서 생성해야 합니다.

액세스 포인트는 정확히 하나의 Amazon S3 디렉터리 버킷과 연결됩니다. AWS 계정 내에서 디렉터리 버킷을 사용하려면 먼저 디렉터리 버킷을 생성해야 합니다. 디렉터리 버킷 생성에 대한 자세한 내용은 가용 영역에서 디렉터리 버킷 생성 또는 로컬 영역에서 디렉터리 버킷 생성 섹션을 참조하세요.

또한, 버킷 이름과 버킷 소유자의 계정 ID를 알고 있으면 다른 AWS 계정 내에 있는 버킷과 연결된 크로스 계정 액세스 포인트를 생성할 수 있습니다. 하지만 크로스 계정 액세스 포인트를 생성해도 버킷 소유자로부터 권한을 부여받기 전까지는 버킷의 데이터에 대한 액세스 권한이 부여되지 않습니다. 버킷 소유자가 버킷 정책을 통해 액세스 포인트 소유자의 계정(귀하의 계정)에게 버킷에 대한 액세스 권한을 부여해야 합니다. 자세한 내용은 크로스 계정 액세스 포인트에 대한 권한 부여 섹션을 참조하세요.

AWS Management Console, AWS CLI, REST API 또는 AWS SDK를 사용하여 모든 디렉터리 버킷에 액세스 포인트를 생성할 수 있습니다. 각 액세스 포인트는 단일 디렉터리 버킷과 연결되며 버킷당 수백 개의 액세스 포인트를 만들 수 있습니다. 액세스 포인트를 만들 때 액세스 포인트의 이름과 이를 연결할 디렉터리 버킷을 선택합니다. 액세스 포인트 이름은 사용자가 제공하는 기본 이름과 버킷 위치의 영역ID와 `--xa-s3`를 차례로 포함하는 접미사로 구성됩니다. 예를 들어 *`myaccesspoint-zoneID`*`--xa-s3`과 같이 생성할 수 있습니다. 가상 프라이빗 클라우드(VPC)를 통해 액세스 포인트에 대한 액세스를 제한할 수도 있습니다. 그런 다음 디렉터리 버킷 이름을 사용할 때와 같이 이름을 사용하여 액세스 포인트를 통해 데이터 읽기 및 쓰기를 즉시 시작할 수 있습니다.

액세스 포인트 범위를 사용하면 액세스 포인트를 통한 디렉터리 버킷 액세스를 특정 접두사 또는 API 작업으로 제한할 수 있습니다. 액세스 포인트에 범위를 추가하지 않으면 액세스 포인트를 통해 액세스할 때 디렉터리 버킷 내 모든 접두사 및 모든 API 작업을 버킷의 객체에 대해 수행할 수 있습니다. 액세스 포인트를 생성한 후에는 AWS CLI, AWS SDK 또는 REST API를 사용하여 범위를 추가, 수정 또는 삭제할 수 있습니다. 자세한 내용은 디렉터리 버킷의 액세스 포인트 범위 관리 섹션을 참조하세요.

액세스 포인트를 생성한 후 액세스 포인트 IAM 리소스 정책을 구성할 수 있습니다. 자세한 내용은 액세스 포인트 정책 보기, 편집 또는 삭제 섹션을 참조하세요.

S3 콘솔 사용

> **Note**
>
> 디렉터리 버킷 화면에서 디렉터리 버킷의 액세스 포인트를 생성할 수도 있습니다. 이렇게 하면 디렉터리 버킷 이름이 제공되므로 액세스 포인트를 생성할 때 버킷을 선택할 필요가 없습니다. 자세한 내용은 디렉터리 버킷 나열 섹션을 참조하세요.

디렉터리 버킷의 액세스 포인트를 생성하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 액세스 포인트를 생성하려는 리전을 선택합니다. 액세스 포인트는 연결된 버킷과 동일한 리전에서 생성해야 합니다.
3. 왼쪽 탐색 창에서 디렉토리 버킷용 액세스 포인트를 선택합니다.
4. 액세스 포인트(Access Points) 페이지에서 액세스 포인트 생성(Create access point)을 선택합니다.
5. 사용 중인 계정 또는 다른 계정의 디렉터리 버킷용 액세스 포인트를 생성할 수 있습니다. 다른 계정의 디렉터리 버킷용 액세스 포인트를 생성하려면 다음과 같이 합니다.

   > **Note**
   >
   > 다른 AWS 계정의 버킷을 사용하는 경우, 액세스 포인트의 요청을 승인하도록 버킷 소유자가 버킷 정책을 업데이트해야 합니다. 버킷 정책 예제는 크로스 계정 액세스 포인트에 대한 권한 부여를 참조하십시오.

   a. 디렉터리 버킷 필드에서 다른 계정의 버킷 지정을 선택합니다.

   b. 버킷 소유자 계정 ID 필드에 버킷을 소유한 AWS 계정 ID를 입력합니다.

   c. 버킷 이름 필드에 기본 이름과 영역 ID를 포함한 버킷의 이름을 입력합니다. 예를 들어 ***bucket-base-name*--*zone-id*--x-s3**입니다.

6. 사용 중인 계정의 디렉터리 버킷용 액세스 포인트를 생성하려면 다음과 같이 합니다.

   a. 디렉터리 버킷 필드에서 이 계정에서 버킷 선택을 선택합니다.

b. 버킷 이름 필드에 기본 이름과 영역 ID를 포함한 버킷의 이름을 입력합니다. 예를 들어 ***bucket-base-name*--*zone-id*--x-s3**입니다. 목록에서 버킷을 선택하려면 S3 찾아보기를 선택한 다음 해당 디렉터리 버킷을 선택합니다.

7. 액세스 포인트 이름의 기본 이름 필드에 액세스 포인트의 기본 이름을 입력합니다. 영역 ID와 전체 액세스 포인트 이름이 나타납니다. 액세스 포인트 명명에 대한 자세한 내용은 디렉터리 버킷의 액세스 포인트 이름 지정 규칙 섹션을 참조하십시오.
8. 네트워크 오리진에서 가상 프라이빗 클라우드(VPC) 또는 인터넷을 선택합니다. 가상 프라이빗 클라우드(VPC)를 선택한 경우 VPC ID 필드에 액세스 포인트에 사용할 VPC의 ID를 입력합니다.
9. (선택 사항) 이 액세스 포인트에 범위를 적용하려면 액세스 포인트 범위에서 접두사 또는 권한을 사용하여 이 액세스 포인트의 범위 제한을 선택합니다.

   a. 디렉터리 버킷에서 접두사에 대한 액세스를 제한하려면 접두사에서 하나 이상의 접두사를 입력합니다. 다른 접두사를 추가하려면 접두사 추가를 선택합니다. 접두사를 제거하려면 제거를 선택합니다.

   > **Note**
   >
   > 액세스 포인트 범위에는 모든 접두사에 대한 총 512자의 문자 제한이 있습니다. 접두사 추가 아래에서 남아 있는 문자 수를 확인할 수 있습니다.

   b. 권한에서 액세스 포인트가 허용할 API 작업을 하나 이상 선택합니다. 데이터 작업을 제거하려면 데이터 작업 이름 옆에 있는 X를 선택합니다.
10. 액세스 포인트에 범위를 적용하지 않고 액세스 포인트를 통한 디렉터리 버킷의 모든 접두사와 모든 API 작업에 대한 액세스를 허용하려면 액세스 포인트 범위에서 전체 버킷에 액세스 적용을 선택합니다.
11. 디렉터리 버킷용 액세스 포인트 생성을 선택합니다. 액세스 포인트의 이름 및 기타 정보는 디렉토리 버킷용 액세스 포인트 목록에 표시됩니다.

AWS CLI 사용

다음 예제 명령은 계정 *111122223333* 내의 버킷 *amzn-s3-demo-bucket--zone-id--x-s3*에 대해 이름이 *example-ap*인 액세스 포인트를 생성합니다.

```
aws s3control create-access-point --name example-ap--zoneID--xa-s3 --account-
id 111122223333 --bucket amzn-s3-demo-bucket--zone-id--x-s3
```

VPC를 통해 액세스 포인트에 대한 액세스를 제한하려면 `--vpc` 파라미터와 VPC ID를 포함합니다.

```
aws s3control create-access-point --name example-ap--zoneID--xa-s3 --account-
id 111122223333 --bucket amzn-s3-demo-bucket--zone-id--x-s3 --vpc vpc-id
```

교차 계정 버킷에 대한 액세스 포인트를 만들 때 `--bucket-account-id` 파라미터를 포함합니다. 다음 예제 명령은 AWS 계정 *444455556666*이 소유한 버킷 *amzn-s3-demo-bucket--zone-id--x-s3*에 대해 AWS 계정 *111122223333*에 액세스 포인트를 생성합니다.

```
aws s3control create-access-point --name example-ap--zoneID--xa-s3 --account-
id 111122223333 --bucket amzn-s3-demo-bucket--zone-id--x-s3 --bucket-account-
id 444455556666
```

자세한 내용과 예제는 AWS CLI 명령 참조의 create-access-point를 참조하세요.

REST API 사용

다음 예제 명령은 계정 *111122223333*의 버킷 *amzn-s3-demo-bucket--zone-id--x-s3*에 대해 *example-ap*이라는 액세스 포인트를 생성하고 VPC *vpc-id*(선택 사항)를 통해 액세스를 제한합니다.

```
PUT /v20180820/accesspoint/example-ap--zoneID--xa-s3 HTTP/1.1
Host: s3express-control.region.amazonaws.com
x-amz-account-id: 111122223333
<?xml version="1.0" encoding="UTF-8"?>
<CreateAccessPointRequest>
   <Bucket>amzn-s3-demo-bucket--zone-id--x-s3s</Bucket>
   <BucketAccountId>111122223333</BucketAccountId>
   <VpcConfiguration>
       <VpcId>vpc-id</VpcId>
   </VpcConfiguration>
</CreateAccessPointRequest>
```

응답:

```
HTTP/1.1 200
<?xml version="1.0" encoding="UTF-8"?>
<CreateAccessPointResult>
   <AccessPointArn>
       "arn:aws:s3express:region:111122223333:accesspoint/example-ap--zoneID--xa-s3"
```

```
   </AccessPointArn>
   <Alias>example-ap--zoneID--xa-s3</Alias>
</CreateAccessPointResult>
```

AWS SDK 사용

AWS SDK를 사용하여 액세스 포인트를 만들 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조의 지원되는 SDK 목록을 참조하세요.

디렉터리 버킷의 액세스 포인트 관리

이 섹션에서는 AWS Command Line Interface, Amazon S3 REST API 또는 AWS SDK를 사용하여 디렉터리 버킷의 액세스 포인트를 관리하는 방법에 대해 설명합니다.

주제

- 디렉터리 버킷의 액세스 포인트 나열
- 디렉터리 버킷의 액세스 포인트에 대한 세부 정보 보기
- 액세스 포인트 정책 보기, 편집 또는 삭제
- 디렉터리 버킷의 액세스 포인트 범위 관리
- 디렉터리 버킷의 액세스 포인트 삭제

디렉터리 버킷의 액세스 포인트 나열

이 섹션에서는 AWS Management Console, AWS Command Line Interface(AWS CLI), REST API 또는 AWS SDK를 사용하여 디렉터리 버킷의 액세스 포인트를 관리하는 방법을 설명합니다.

S3 콘솔 사용

AWS 계정의 액세스 포인트를 나열하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 액세스 포인트를 나열하려는 리전을 선택합니다.
3. 콘솔 왼쪽의 탐색 창에서 디렉토리 버킷용 액세스 포인트를 선택합니다.
4. (선택 사항) 이름으로 액세스 포인트를 검색합니다. 선택한 AWS 리전의 액세스 포인트만 여기에 표시됩니다.

5. 관리할 액세스 포인트의 이름을 선택합니다.

AWS CLI 사용

다음 `list-access-points-for-directory-buckets` 예시 명령은 AWS CLI를 사용하여 AWS 계정이 소유하고 디렉터리 버킷과 연결된 액세스 포인트를 나열하는 방법을 보여줍니다.

다음 명령은 버킷 *amzn-s3-demo-bucket--zone-id--x-s3*에 연결된 AWS 계정 *111122223333*의 액세스 포인트를 나열합니다.

```
aws s3control list-access-points-for-directory-buckets --account-id 111122223333 --
directory-bucket amzn-s3-demo-bucket--zone-id--x-s3
```

자세한 내용과 예제는 AWS CLI 명령 참조의 list-access-points-for-directory-buckets를 참조하세요.

REST API 사용

다음 예시 명령은 REST API를 사용하여 액세스 포인트를 나열하는 방법을 보여줍니다.

```
GET /v20180820/directoryaccesspoint?directoryBucket=amzn-s3-demo-bucket--zone-id--x-s3
&maxResults=maxResults HTTP/1.1
Host: s3express-control.region.amazonaws.com
x-amz-account-id: 111122223333
```

Example **ListAccessPointsForDirectoryBuckets** 응답

```
HTTP/1.1 200
<?xml version="1.0" encoding="UTF-8"?>
<ListDirectoryAccessPointsResult>
    <AccessPointList>
        <AccessPoint>
            <AccessPointArn>arn:aws:s3express:region:111122223333:accesspoint/example-
access-point--zoneID--xa-s3</AccessPointArn>
            <Alias>example-access-point--zoneID--xa-s3</Alias>
            <Bucket>amzn-s3-demo-bucket--zone-id--x-s3</Bucket>
            <BucketAccountId>111122223333</BucketAccountId>
            <Name>example-access-point--zoneID--xa-s3</Name>
            <NetworkOrigin>VPC</NetworkOrigin>
```

```
            <VpcConfiguration>
                <VpcId>VPC-1</VpcId>
            </VpcConfiguration>
        </AccessPoint>
    </AccessPointList>
</ListDirectoryAccessPointsResult>
```

AWS SDK 사용

AWS SDK를 사용하여 액세스 포인트를 나열할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조의 지원되는 SDK 목록을 참조하세요.

디렉터리 버킷의 액세스 포인트에 대한 세부 정보 보기

이 섹션에서는 AWS Management Console, AWS CLI, AWS SDK 또는 REST API를 사용하여 디렉터리 버킷의 액세스 포인트에 대한 세부 정보를 보는 방법을 설명합니다.

S3 콘솔 사용

디렉터리 버킷의 액세스 포인트에 대한 세부 정보를 보면 액세스 포인트 및 연결된 디렉터리 버킷에 대한 다음 정보를 확인할 수 있습니다.

- 속성:
  - 디렉터리 버킷 이름
  - 디렉터리 버킷 소유자 계정 ID
  - AWS 리전
  - 디렉터리 버킷 위치 유형
  - 디렉터리 버킷 위치 이름
  - 액세스 포인트 생성 날짜
  - 네트워크 오리진
  - VPC ID
  - S3 URI
  - 액세스 포인트 ARN
  - 액세스 포인트 별칭
- 권한:
  - IAM 외부 액세스 분석기 조사 결과

- 액세스 포인트 범위
- 액세스 포인트 정책

AWS 계정의 액세스 포인트에 대한 세부 정보를 보는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 액세스 포인트를 나열하려는 리전을 선택합니다.
3. 콘솔 왼쪽의 탐색 창에서 디렉토리 버킷용 액세스 포인트를 선택합니다.
4. (선택 사항) 이름으로 액세스 포인트를 검색합니다. 선택한 AWS 리전의 액세스 포인트만 여기에 표시됩니다.
5. 관리할 액세스 포인트의 이름을 선택합니다.
6. 속성 탭 또는 권한 탭을 선택합니다.

AWS CLI 사용

다음 `get-access-point` 예시 명령은 AWS CLI를 사용하여 액세스 포인트의 세부 정보를 보는 방법을 보여줍니다.

다음 명령은 AWS 계정 *111122223333*에 대한 액세스 포인트 *my-access-point--zoneID--xa-s3*의 세부 정보를 나열합니다.

```
aws s3control get-access-point --name my-access-point--zoneID--xa-s3 --account-
id 111122223333
```

Example / **`get-access-point`** 명령의 출력

```
{
    "Name": "example-access-point--zoneID--xa-s3",
    "Bucket": "amzn-s3-demo-bucket--zone-id--x-s3",
    "NetworkOrigin": "Internet",
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
```

```
    },
    "CreationDate": "2025-04-23T18:26:22.146000+00:00",
    "Alias": "example-access-point--zoneID--xa-s3",
    "AccessPointArn": "arn:aws:s3express:region:111122223333:accesspoint/example-
access-point--zoneID--xa-s3",
    "BucketAccountId": "296805379465"
}
```

자세한 내용과 예제는 AWS CLI 명령 참조에서 get-access-point를 참조하십시오.

REST API 사용

REST API를 사용하여 액세스 포인트의 세부 정보를 볼 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 GetAccessPoint를 참조하십시오.

AWS SDK 사용

AWS SDK를 사용하여 액세스 포인트의 세부 정보를 볼 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조의 지원되는 SDK 목록을 참조하세요.

액세스 포인트 정책 보기, 편집 또는 삭제

AWS Identity and Access Management(IAM) 액세스 포인트 정책을 사용하여 액세스 포인트에 액세스할 수 있는 위탁자와 리소스를 제어할 수 있습니다. 액세스 포인트 범위는 액세스 포인트에 대한 접두사 및 API 권한을 관리합니다. AWS Command Line Interface, REST API 또는 AWS SDK를 사용하여 액세스 포인트 정책을 생성, 편집 및 삭제할 수 있습니다. 액세스 포인트 범위에 대한 자세한 내용은 디렉터리 버킷의 액세스 포인트 범위 관리 섹션을 참조하세요.

> **Note**
>
> 디렉터리 버킷은 세션 기반 권한 부여를 사용하므로 정책에는 항상 `s3express:CreateSession` 작업이 포함되어야 합니다.

S3 콘솔 사용

액세스 포인트 정책 보기, 편집 또는 삭제 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.

2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 액세스 포인트를 나열하려는 리전을 선택합니다.
3. 콘솔 왼쪽의 탐색 창에서 디렉토리 버킷용 액세스 포인트를 선택합니다.
4. (선택 사항) 이름으로 액세스 포인트를 검색합니다. 선택한 AWS 리전의 액세스 포인트만 여기에 표시됩니다.
5. 관리할 액세스 포인트의 이름을 선택합니다.
6. 권한 탭을 선택합니다.
7. 액세스 포인트 정책을 생성하거나 편집하려면 액세스 포인트 정책에서 편집을 선택합니다. 정책을 편집합니다. 저장을 선택합니다.
8. 액세스 포인트 정책을 삭제하려면 액세스 포인트 정책에서 삭제를 선택합니다. 액세스 포인트 정책 삭제 창에서 **`confirm`**을 입력하고 삭제를 선택합니다.

AWS CLI 사용

`get-acccess-point-policy`, `put-access-point-policy` 및 `delete-access-point-policy` 명령을 사용하여 액세스 포인트 정책을 확인, 편집 또는 삭제할 수 있습니다. 자세한 내용은 AWS CLI 명령 참조의 get-access-point-policy, put-access-point-policy 또는 delete-access-point-policy 섹션을 참조하세요.

REST API 사용

REST API `GetAccessPointPolicy`, `DeleteAccessPointPolicy` 및 `PutAccessPointPolicy` 작업을 사용하여 액세스 포인트 정책을 확인, 삭제 또는 편집할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조의 PutAccessPointPolicy, GetAccessPointPolicy 또는 DeleteAccessPointPolicy 섹션을 참조하세요.

AWS SDK 사용

AWS SDK를 사용하여 액세스 포인트 정책을 확인, 삭제 또는 편집할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조의 GetAccessControlPolicy, DeleteAccessControlPolicy 및 PutAccessControlPolicy에 지원되는 SDK 목록을 참조하세요.

디렉터리 버킷의 액세스 포인트 범위 관리

이 섹션에서는 AWS Command Line Interface, REST API 또는 AWS SDK를 사용하여 디렉터리 버킷의 액세스 포인트 범위를 보고 수정하는 방법을 설명합니다. 액세스 포인트 범위를 사용하여 특정 접두사 또는 API 작업으로 액세스를 제한할 수 있습니다.

주제

- 디렉터리 버킷의 액세스 포인트 범위 보기
- 디렉터리 버킷의 액세스 포인트 범위 수정
- 디렉터리 버킷의 액세스 포인트 범위 삭제

디렉터리 버킷의 액세스 포인트 범위 보기

AWS Management Console, AWS Command Line Interface, REST API 또는 AWS SDK를 사용하여 디렉터리 버킷의 액세스 포인트 범위를 볼 수 있습니다.

S3 콘솔 사용

AWS CLI 사용

다음 `get-access-point-scope` 예시 명령은 AWS CLI를 사용하여 액세스 포인트의 범위를 보는 방법을 보여줍니다.

다음 명령은 AWS 계정 *111122223333*에 대한 액세스 포인트 *my-access-point--zoneID*--xa-s3의 범위를 표시합니다.

```
aws s3control get-access-point-scope --name my-access-point--zoneID--xa-s3 --account-
id 111122223333
```

자세한 내용과 예제는 AWS CLI 명령 참조의 get-access-point-scope를 참조하세요.

Example **`get-access-point-scope`**의 결과

```
{
    "Scope": {
        "Permissions": [
            "ListBucket",
            "PutObject"
        ]
  "Prefixes": [
            "Prefix": "MyPrefix1*",
            "Prefix": "MyObjectName.csv"
        ]
```

```
    }
}
```

REST API 사용

다음 GetAccessPointScope 예제 요청은 REST API를 사용하여 액세스 포인트의 범위를 보는 방법을 보여줍니다.

다음 요청은 AWS 계정 *111122223333*에 대한 액세스 포인트 *my-access-point*--*region-zoneID*--xa-s3의 범위를 표시합니다.

```
GET /v20180820/accesspoint/my-access-point--zoneID--xa-s3/scope HTTP/1.1
Host: s3express-control.region.amazonaws.com
x-amz-account-id: 111122223333
```

Example **GetAccessPointScope**의 결과

```
HTTP/1.1 200
<?xml version="1.0" encoding="UTF-8"?>
<GetAccessPointScopeResult>
    <Scope>
        <Prefixes>
            <Prefix>MyPrefix1*</Prefix>
            <Prefix>MyObjectName.csv</Prefix>
        </Prefixes>
        <Permissions>
            <Permission>ListBucket</Permission>
            <Permission>PutObject</Permission>
        </Permissions>
        <Scope>
</GetAccessPointScopeResult>
```

AWS SDK 사용

AWS SDK를 사용하여 액세스 포인트의 범위를 볼 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조의 지원되는 SDK 목록을 참조하세요.

디렉터리 버킷의 액세스 포인트 범위 수정

AWS Command Line Interface, REST API 또는 AWS SDK를 사용하여 디렉터리 버킷의 액세스 포인트 범위를 수정할 수 있습니다. 액세스 포인트 범위는 특정 접두사, API 작업 또는 이 둘의 조합으로 액세스를 제한하는 데 사용됩니다.

다음 API 작업 중 하나 이상을 권한으로 포함할 수 있습니다.

- `PutObject`
- `GetObject`
- `DeleteObject`
- `ListBucket`(`ListObjectsV2`의 경우 필수)
- `GetObjectAttributes`
- `AbortMultipartUploads`
- `ListBucketMultipartUploads`
- `ListMultipartUploadParts`

> **Note**
>
> - 원하는 양의 접두사를 지정할 수 있지만 모든 접두사의 총 문자 길이는 크기가 256바이트 미만이어야 합니다.
> - 액세스 포인트의 범위를 수정할 때 기존 범위를 바꿉니다.

AWS CLI 사용

다음 `put-access-point-scope` 예시 명령은 AWS CLI를 사용하여 액세스 포인트의 범위를 수정하는 방법을 보여줍니다.

다음 명령은 AWS 계정 *111122223333*에 대한 *my-access-point*--*zoneID*--xa-s3의 액세스 포인트 범위를 수정합니다.

> **Note**
>
> 별표(*) 문자를 사용하여 접두사에 와일드카드를 사용할 수 있습니다. 별표 문자를 리터럴로 사용하려면 앞에 백슬래시 문자(\)를 추가하여 이스케이프합니다.

또한 모든 접두사는 암시적 '*'로 끝납니다. 즉, 접두사 내의 모든 경로가 포함됩니다.

```
aws s3control put-access-point-scope --name my-access-point--zoneID--xa-s3 --account-
id 111122223333 --scope Prefixes=string,Permissions=string
```

자세한 내용과 예제는 AWS CLI 명령 참조의 put-access-point-scope를 참조하세요.

REST API 사용

다음 PutAccessPointScope 예제 요청은 REST API를 사용하여 액세스 포인트의 범위를 수정하는 방법을 보여줍니다.

다음 요청은 AWS 계정 *111122223333*에 대한 *my-access-point*--*zoneID*--xa-s3의 액세스 포인트 범위를 수정합니다.

> **Note**
>
> 별표(*) 문자를 사용하여 접두사에 와일드카드를 사용할 수 있습니다. 별표 문자를 리터럴로 사용하려면 앞에 백슬래시 문자(\)를 추가하여 이스케이프합니다.
> 또한 모든 접두사는 암시적 '*'로 끝납니다. 즉, 접두사 내의 모든 경로가 포함됩니다.

```
PUT /v20180820/accesspoint/my-access-point--zoneID--xa-s3/scope HTTP/1.1
Host: s3express-control.region.amazonaws.com
x-amz-account-id: 111122223333
<?xml version="1.0" encoding="UTF-8"?>
<PutAccessPointScopeRequest>
        <Scope>
            <Prefixes>
                <Prefix>Jane/*</Prefix>
            </Prefixes>
            <Permissions>
                <Permission>PutObject</Permission>
                <Permission>GetObject</Permission>
            </Permissions>
            <Scope>
    </PutAccessPointScopeRequest>
```

AWS SDK 사용

AWS CLI, AWS SDK 또는 REST API를 사용하여 액세스 포인트의 범위를 수정할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조의 지원되는 SDK 목록을 참조하세요.

디렉터리 버킷의 액세스 포인트 범위 삭제

AWS Command Line Interface, REST API 또는 AWS SDK를 사용하여 디렉터리 버킷의 액세스 포인트 범위를 삭제할 수 있습니다.

> **Note**
>
> 액세스 포인트의 범위를 삭제하면 모든 접두사와 권한이 삭제됩니다.

AWS CLI 사용

다음 `delete-access-point-scope` 예시 명령은 AWS CLI를 사용하여 액세스 포인트 범위를 삭제하는 방법을 보여줍니다.

다음 명령은 AWS 계정 *111122223333*에 대한 액세스 포인트 *my-access-point*--*zoneID*--xa-s3의 범위를 삭제합니다.

```
aws s3control delete-access-point-scope --name my-access-point--region-zoneID--xa-s3 --
account-id 111122223333
```

자세한 내용과 예제는 AWS CLI 명령 참조의 delete-access-point-scope를 참조하세요.

REST API 사용

다음 요청은 AWS 계정 *111122223333*에 대한 액세스 포인트 *my-access-point*--*zoneID*--xa-s3의 범위를 삭제합니다.

```
DELETE /v20180820/accesspoint/my-access-point--zoneID--xa-s3/scope HTTP/1.1
Host: s3express-control.region.amazonaws.com
x-amz-account-id: 111122223333
```

AWS SDK 사용

AWS SDK를 사용하여 액세스 포인트의 범위를 삭제할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조의 지원되는 SDK 목록을 참조하세요.

디렉터리 버킷의 액세스 포인트 삭제

이 섹션에서는 AWS Management Console, AWS Command Line Interface, REST API 또는 AWS SDK를 사용하여 액세스 포인트를 삭제하는 방법을 설명합니다.

> **Note**
>
> 액세스 포인트에 연결된 디렉터리 버킷을 삭제하려면 먼저 액세스 포인트를 삭제해야 합니다.

S3 콘솔 사용

AWS 계정의 디렉터리 버킷용 액세스 포인트를 삭제하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 액세스 포인트를 나열하려는 리전을 선택합니다.
3. 콘솔 왼쪽의 탐색 창에서 디렉토리 버킷용 액세스 포인트를 선택합니다.
4. (선택 사항) 이름으로 액세스 포인트를 검색합니다. 선택한 AWS 리전의 액세스 포인트만 여기에 표시됩니다.
5. 삭제할 액세스 포인트의 이름을 선택합니다.
6. 삭제를 선택합니다.
7. **`confirm`**을 입력한 후 삭제를 선택하여 삭제를 확인합니다.

AWS CLI 사용

다음 `delete-access-point` 예시 명령은 AWS CLI를 사용하여 액세스 포인트를 삭제하는 방법을 보여줍니다.

다음 명령은 AWS 계정 *111122223333*에 대한 액세스 포인트 *my-access-point*--*zoneID*--xa-s3을 삭제합니다.

```
aws s3control delete-access-point --name my-access-point--zoneID--xa-s3 --account-
id 111122223333
```

자세한 내용과 예제는 AWS CLI 명령 참조에서 delete-access-point를 참조하십시오.

REST API 사용

REST API를 사용하여 액세스 포인트를 삭제할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 DeleteAccessPoint를 참조하십시오.

AWS SDK 사용

AWS SDK를 사용하여 액세스 포인트를 삭제할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조의 지원되는 SDK 목록을 참조하세요.

AWS CloudTrail로 디렉터리 버킷에 로그인

Amazon S3는 사용자, 역할, 또는 AWS 서비스가 수행한 작업에 대한 레코드를 제공하는 서비스인 AWS CloudTrail과 통합됩니다. CloudTrail은 Amazon S3에 대한 모든 API 직접 호출을 이벤트로 캡처합니다. CloudTrail에서 수집한 정보를 사용하여 Amazon S3에 전송된 요청, 요청이 이루어진 IP 주소, 요청이 이루어진 시간, 추가 세부 정보를 확인할 수 있습니다. 지원되는 이벤트 활동이 Amazon S3에서 발생하면 해당 활동이 CloudTrail 이벤트에 기록됩니다. AWS CloudTrail 추적을 사용하여 디렉터리 버킷에 대한 관리 이벤트 및 데이터 이벤트를 로깅할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서의 Amazon S3 CloudTrail 이벤트 및 AWS CloudTrail이란 무엇입니까?를 참조하세요.

디렉터리 버킷에 대한 CloudTrail 관리 이벤트

기본적으로 CloudTrail은 디렉터리 버킷에 대한 버킷 수준 작업을 관리 이벤트로 로깅합니다. 디렉터리 버킷의 CloudTrail 관리 이벤트용 `eventsource`는 `s3express.amazonaws.com`입니다. AWS 계정을 설정하면 CloudTrail 관리 이벤트가 기본적으로 활성화됩니다. 다음과 같은 리전 엔드포인트 API 작업(버킷 수준 또는 컨트롤 플레인, API 작업)이 CloudTrail에 로깅됩니다.

- `CreateBucket`
- `DeleteBucket`
- `DeleteBucketPolicy`
- `PutBucketPolicy`
- `GetBucketPolicy`
- `ListDirectoryBuckets`
- `ListMultipartUploads`
- `GetBucketEncryption`
- `PutBucketEncryption`
- `DeleteBucketEncryption`

> **Note**
>
> `ListMultipartUploads`는 영역 엔드포인트 API 작업입니다. 하지만 이 API 작업은 관리 이벤트로 CloudTrail에 로깅됩니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 `ListMultipartUploads`를 참조하십시오.

CloudTrail 관리 이벤트에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 관리 이벤트 로깅을 참조하세요.

디렉터리 버킷에 대한 CloudTrail 데이터 이벤트

데이터 이벤트는 리소스 기반 또는 리소스에서 수행된 리소스 작업에 대한 정보를 제공합니다(예: Amazon S3 객체 읽기 또는 쓰기). 이를 데이터 영역 작업이라고도 합니다. 데이터 이벤트가 대량 활동인 경우도 있습니다. 기본적으로 CloudTrail 추적은 데이터 이벤트를 로깅하지 않지만 범용 버킷과 디렉터리 버킷에 저장된 객체에 대한 데이터 이벤트를 로깅하도록 추적을 구성할 수 있습니다. 자세한 내용은 콘솔을 사용하여 버킷의 객체에 대한 로깅 사용 설정을 참조하세요.

CloudTrail에서 추적을 위해 데이터 이벤트를 로깅할 때 고급 이벤트 선택기 또는 기본 이벤트 선택기를 사용하도록 선택할 수 있습니다. 디렉터리 버킷에 저장된 객체의 데이터 이벤트를 로깅하려면 고급 이벤트 선택기를 사용해야 합니다. 고급 리소스 선택기를 구성할 때 리소스 유형(`AWS::S3Express::Object`)을 선택하거나 지정합니다.

다음과 같은 영역 엔드포인트 API 작업(객체 수준 또는 데이터 영역, API 작업)이 CloudTrail에 로깅됩니다.

- `AbortMultipartUpload`
- `CompleteMultipartUpload`
- `CreateSession`
- `CopyObject`
- `CreateMultipartUpload`
- `DeleteObject`
- `DeleteObjects`
- `GetObject`
- `GetObjectAttributes`
- `HeadBucket`

- HeadObject
- ListObjectsV2
- ListParts
- PutObject
- RenameObject
- UploadPart
- UploadPartCopy

CloudTrail 데이터 이벤트에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 데이터 이벤트 로깅을 참조하세요.

디렉터리 버킷의 CloudTrail 이벤트에 대한 자세한 내용은 다음 주제를 참조하세요.

주제
- 디렉터리 버킷에 대한 CloudTrail 로그 파일 예제

디렉터리 버킷에 대한 CloudTrail 로그 파일 예제

CloudTrail 로그 파일에는 요청된 API 작업에 대한 정보, 작업 날짜 및 시간, 요청 파라미터 등이 포함됩니다. 이 주제에서는 디렉터리 버킷의 CloudTrail 데이터 이벤트 및 관리 이벤트 예시를 제공합니다.

주제
- 디렉터리 버킷에 대한 CloudTrail 데이터 이벤트 로그 파일 예제

디렉터리 버킷에 대한 CloudTrail 데이터 이벤트 로그 파일 예제

다음은 CreateSession을 시연하는 CloudTrail 로그 파일 예시입니다.

```
{
    "eventVersion": "1.09",
    "userIdentity": {
      "type": "AssumedRole",
      "principalId": "AROAIDPPEZS35WEXAMPLE:AssumedRoleSessionName",
      "arn": "arn:aws:sts::111122223333assumed-role/RoleToBeAssumed/MySessionName",
      "accountId": "111122223333",
      "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
      "sessionContext": {
```

```
            "sessionIssuer": {
              "type": "Role",
              "principalId": "AROAIDPPEZS35WEXAMPLE",
              "arn": "arn:aws:iam::111122223333:role/RoleToBeAssumed",
              "accountId": "111122223333",
              "userName":"RoleToBeAssumed
            },

            "attributes": {
              "creationDate": "2024-07-02T00:21:16Z",
            "mfaAuthenticated": "false"
            }
          }
        },
        "eventTime": "2024-07-02T00:22:11Z",
        "eventSource": "s3express.amazonaws.com",
        "eventName": "CreateSession",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "72.21.198.68",
        "userAgent": "aws-sdk-java/2.20.160-SNAPSHOT
 Linux/5.10.216-225.855.amzn2.x86_64 OpenJDK_64-Bit_Server_VM/11.0.23+9-LTS
 Java/11.0.23 vendor/Amazon.com_Inc. md/internal exec-env/AWS_Lambda_java11 io/sync
 http/Apache cfg/retry-mode/standard",
        "requestParameters": {
          "bucketName": "bucket-base-name--usw2-az1--x-s3".
            "host": "bucket-base-name--usw2-az1--x-s3.s3express-usw2-az1.us-
west-2.amazonaws.com",
            "x-amz-create-session-mode": "ReadWrite"
        },
        "responseElements": {
            "credentials": {
                "accessKeyId": "AKIAI44QH8DHBEXAMPLE"
                "expiration": ""Mar 20, 2024, 11:16:09 PM",
                "sessionToken": "<session token string>"
           },
        },
        "additionalEventData": {
            "SignatureVersion": "SigV4",
            "cipherSuite": "TLS_AES_128_GCM_SHA256",
            "bytesTransferredIn": 0,
            "AuthenticationMethod": "AuthHeader",
            "xAmzId2": "q6xhNJYmhg",
            "bytesTransferredOut": 1815,
            "availabilityZone": "usw2-az1"
```

```
          },
          "requestID": "28d2faaf-3319-4649-998d-EXAMPLE72818",
          "eventID": "694d604a-d190-4470-8dd1-EXAMPLEe20c1",
          "readOnly": true,
          "resources": [
            {
              "type": "AWS::S3Express::Object",
              "ARNPrefix": "arn:aws:s3express:us-west-2:111122223333:bucket-base-name--
usw2-az1--x-s3"
            },
            {
              "accountId": "111122223333"
              "type": "AWS::S3Express::DirectoryBucket",
              "ARN": "arn:aws:s3express:us-west-2:111122223333:bucket-base-name--usw2-
az1--x-s3"
             }
           ],
           "eventType": "AwsApiCall",
           "managementEvent": false,
           "recipientAccountId": "111122223333",
           "eventCategory": "Data",
           "tlsDetails": {
             "tlsVersion": "TLSv1.3",
             "cipherSuite": "TLS_AES_128_GCM_SHA256",
             "clientProvidedHostHeader": "bucket-base-name--usw2-az1--x-s3.s3express-
usw2-az1.us-west-2.amazonaws.com"
            }
          }
```

영역 엔드포인트 API 작업(객체 수준 작업, 즉 데이터 영역 작업)을 사용하려는 경우 CreateSession API 작업을 사용하여 지연 시간이 짧은 데이터 요청 권한 부여에 최적화된 세션을 생성하고 관리할 수 있습니다. CreateSession을 사용하여 로깅 양을 줄일 수도 있습니다. 세션 중에 수행된 영역 API 작업을 식별하기 위해 CreateSession 로그 파일의 responseElements 아래 있는 accessKeyId를 다른 영역 API 작업의 로그 파일에 있는 accessKeyId와 일치시킬 수 있습니다. 자세한 내용은 CreateSession 권한 부여를 참조하세요.

다음은 CreateSession에서 인증한 GetObject API 작업을 시연하는 CloudTrail 로그 파일 예시를 보여줍니다.

```
    {
        "eventVersion": "1.09",
        "userIdentity": {
```

```
          "type": "AssumedRole",
          "principalId": "AROAIDPPEZS35WEXAMPLE:AssumedRoleSessionName",
          "arn": "arn:aws:sts::111122223333assumed-role/RoleToBeAssumed/MySessionName",
          "accountId": "111122223333",
          "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
          "sessionContext": {
            "attributes": {
              "creationDate": "2024-07-02T00:21:49Z"
            }
          }
        },
        "eventTime": "2024-07-02T00:22:01Z",
        "eventSource": "s3express.amazonaws.com",
        "eventName": "GetObject",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "72.21.198.68",
        "userAgent": "aws-sdk-java/2.25.66 Linux/5.10.216-225.855.amzn2.x86_64
 OpenJDK_64-Bit_Server_VM/17.0.11+9-LTS Java/17.0.11 vendor/Amazon.com_Inc. md/internal
 exec-env/AWS_Lambda_java17 io/sync http/Apache cfg/retry-mode/legacy",
        "requestParameters": {
          "bucketName": "bucket-base-name--usw2-az1--x-s3",
          "x-amz-checksum-mode": "ENABLED",
          "Host": "bucket-base-name--usw2-az1--x-s3.s3express-usw2-az1.us-
west-2.amazonaws.com",
          "key": "test-get-obj-with-checksum"
        },
        "responseElements": null,
        "additionalEventData": {
          "SignatureVersion": "Sigv4",
          "CipherSuite": "TLS_AES_128_GCM_SHA256",
          "bytesTransferredIn": 0,
          "AuthenticationMethod": "AuthHeader",
          "x-amz-id-2": "oOy6w8K7LFsyFN",
          "bytesTransferredOut": 9,
          "availabilityZone": "usw2-az1",
          "sessionModeApplied": "ReadWrite"
         },
          "requestID": "28d2faaf-3319-4649-998d-EXAMPLE72818",
          "eventID": "694d604a-d190-4470-8dd1-EXAMPLEe20c1",
          "readOnly": true,
          "resources": [
            {
              "type": "AWS::S3Express::Object",
```

```
              "ARNPrefix": "arn:aws:s3express:us-west-2:111122223333:bucket-base-name--
usw2-az1--x-s3"
            },
            {
              "accountId": "111122223333",
              "type": "AWS::S3Express::DirectoryBucket",
              "ARN": "arn:aws:s3express:us-west-2:111122223333:bucket-base-name--usw2-
az1--x-s3"
             }
           ],
           "eventType": "AwsApiCall",
           "managementEvent": false,
           "recipientAccountId": "111122223333",
           "eventCategory": "Data",
           "tlsDetails": {
             "tlsVersion": "TLSv1.3",
             "cipherSuite": "TLS_AES_128_GCM_SHA256",
             "clientProvidedHostHeader": "bucket-base-name--usw2-az1--x-s3.s3express-
usw2-az1.us-west-2.amazonaws.com"
            }
          }
```

위의 `GetObject` 로그 파일 예시에서 `accessKeyId`(AKIAI44QH8DHBEXAMPLE)는 CreateSession 로그 파일 예시의 `responseElements` 아래에 있는 `accessKeyId`와 일치합니다. 일치하는 `accessKeyId`는 `GetObject` 작업이 수행된 세션을 나타냅니다.

다음 예제에서는 S3 수명 주기에서 간접적으로 호출한 디렉터리 버킷의 `DeleteObjects` 작업을 설명하는 CloudTrail 로그 항목을 보여줍니다. 자세한 내용은 Working with S3 Lifecycle for directory buckets 섹션을 참조하세요.

```
eventVersion:"1.09",
  userIdentity:{

    type:"AWSService",
    invokedBy:"lifecycle.s3.amazonaws.com"
  },
  eventTime:"2024-09-11T00:55:54Z",
  eventSource:"s3express.amazonaws.com",
  eventName:"DeleteObjects",
  awsRegion:"us-east-2",
  sourceIPAddress:"lifecycle.s3.amazonaws.com",
  userAgent:"gamma.lifecycle.s3.amazonaws.com",
  requestParameters:{
```

```
    bucketName:"amzn-s3-demo-bucket--use2-az2--x-s3",
    'x-amz-expected-bucket-owner':"637423581905",
    Host:"amzn-s3-demo-bucket--use2-az2--x-s3.gamma.use2-az2.express.s3.aws.dev",
    delete:"",
    'x-amz-sdk-checksum-algorithm':"CRC32C"
  },
  responseElements:null,
  additionalEventData:{

    SignatureVersion:"Sigv4",
    CipherSuite:"TLS_AES_128_GCM_SHA256",
    bytesTransferredIn:41903,
    AuthenticationMethod:"AuthHeader",
    'x-amz-id-2':"9H5YWZY0",
    bytesTransferredOut:35316,
    availabilityZone:"use2-az2",
    sessionModeApplied:"ReadWrite"
  },
  requestID:"011eeadd04000191",
  eventID:"d3d8b116-219d-4ee6-a072-5f9950733c74",
  readOnly:false,
  resources:[

    {

      type:"AWS::S3Express::Object",
      ARNPrefix:"arn:aws:s3express:us-east-2:637423581905:bucket/amzn-s3-demo-bucket--
use2-az2--x-s3/"
    },
    {

      accountId:"637423581905",
      type:"AWS::S3Express::DirectoryBucket",
      ARN:"arn:aws:s3express:us-east-2:637423581905:bucket/amzn-s3-demo-bucket--use2-
az2--x-s3"
    }
  ],
  eventType:"AwsApiCall",
  managementEvent:false,
  recipientAccountId:"637423581905",
  sharedEventID:"59f877ac-1dd9-415d-b315-9bb8133289ce",
  eventCategory:"Data"
```

```
}
```

다음 예제에서는 S3 수명 주기에서 간접적으로 호출한 `CreateSession` 작업의 `Access Denied` 요청을 설명하는 CloudTrail 로그 항목을 보여줍니다. 자세한 내용은 CreateSession 섹션을 참조하세요.

```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "gamma.lifecycle.s3.amazonaws.com"
    },
    "eventTime": "2024-09-11T18:13:08Z",
    "eventSource": "s3express.amazonaws.com",
    "eventName": "CreateSession",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "gamma.lifecycle.s3.amazonaws.com",
    "userAgent": "gamma.lifecycle.s3.amazonaws.com",
    "errorCode": "AccessDenied",
    "errorMessage": "Access Denied",
    "requestParameters": {
        "bucketName": "amzn-s3-demo-bucket--use2-az2--x-s3",
        "Host": "amzn-s3-demo-bucket--use2-az2--x-s3.gamma.use2-
az2.express.s3.aws.dev",
        "x-amz-create-session-mode": "ReadWrite",
        "x-amz-server-side-encryption": "AES256"
    },
    "responseElements": null,
    "additionalEventData": {
        "SignatureVersion": "Sigv4",
        "CipherSuite": "TLS_AES_128_GCM_SHA256",
        "bytesTransferredIn": 0,
        "AuthenticationMethod": "AuthHeader",
        "x-amz-id-2": "zuDDC1VNbC4LoNwUIc5",
        "bytesTransferredOut": 210,
        "availabilityZone": "use2-az2"
    },
    "requestID": "010932f174000191e24a0",
    "eventID": "dce7cc46-4cd3-46c0-9a47-d1b8b70e301c",
    "readOnly": true,
    "resources": [{
            "type": "AWS::S3Express::Object",
            "ARNPrefix": "arn:aws:s3express:us-east-2:637423581905:bucket/amzn-s3-demo-
bucket--use2-az2--x-s3/"
```

```
        },
        {
            "accountId": "637423581905",
            "type": "AWS::S3Express::DirectoryBucket",
            "ARN": "arn:aws:s3express:us-east-2:637423581905:bucket/amzn-s3-demo-
bucket--use2-az2--x-s3"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "637423581905",
    "sharedEventID": "da96b5bd-6066-4a8d-ad8d-f7f427ca7d58",
    "eventCategory": "Data"
}
```

디렉터리 버킷 성능 최적화

디렉터리 버킷을 사용할 때 최상의 성능을 얻으려면 다음 지침을 따르는 것이 좋습니다.

S3 Express One Zone의 모범 사례에 대한 자세한 내용은 S3 Express One Zone 성능을 최적화하는 모범 사례 섹션을 참조하세요.

세션 기반 인증 사용

디렉터리 버킷은 디렉터리 버킷에 대한 요청을 인증하고 권한을 부여하기 위해 새로운 세션 기반 권한 부여 메커니즘을 지원합니다. 세션 기반 인증을 통해 AWS SDK는 `CreateSession` API 작업을 자동으로 사용하여 짧은 지연 시간으로 디렉터리 버킷에 대한 데이터 요청에 권한을 부여하는 데 사용할 수 있는 임시 세션 토큰을 생성합니다.

AWSSDK는 `CreateSession` API 작업을 사용하여 임시 자격 증명을 요청한 다음 5분마다 사용자를 대신하여 자동으로 토큰을 생성하고 새로 고칩니다. 디렉터리 버킷의 성능적 이점을 활용하려면 AWS SDK를 사용하여 `CreateSession` API 요청을 시작하고 관리하는 것이 좋습니다. 이 세션 기반 모델에 대한 자세한 내용은 `CreateSession`을 사용하여 영역 엔드포인트 API 작업 권한 부여 섹션을 참조하세요.

S3 추가 체크섬 모범 사례

디렉터리 버킷은 업로드 또는 다운로드 중에 데이터를 검증하는 데 사용하는 체크섬 알고리즘을 선택할 수 있는 옵션을 제공합니다. SHA(보안 해시 알고리즘) 또는 CRC(순환 중복 검사) 데이터 무결성 확

인 알고리즘(CRC32, CRC32C, SHA-1, SHA-256) 중 하나를 선택할 수 있습니다. MD5 기반 체크섬은 S3 Express One Zone 스토리지 클래스에서 지원되지 않습니다.

CRC32는 디렉터리 버킷과 데이터를 주고받을 때 AWS SDK에서 사용하는 기본 체크섬입니다. 디렉터리 버킷의 성능을 극대화하려면 CRC32 및 CRC32C를 사용하는 것이 좋습니다.

최신 버전의 AWS SDK 및 공용 런타임 라이브러리를 사용하세요.

또한 일부 AWS SDK는 S3 클라이언트에서 성능을 더욱 가속화하기 위해 AWS Common Runtime(CRT) 라이브러리를 제공합니다. 이러한 SDK에는 AWS SDK for Java 2.x, AWS SDK for C++, AWS SDK for Python (Boto3) 등이 있습니다. CRT 기반 S3 클라이언트는 멀티파트 업로드 API 작업과 바이트 범위 가져오기를 자동으로 사용하여 연결을 수평적으로 스케일링하는 작업을 자동화함으로써 향상된 성능과 신뢰성을 제공하면서 디렉터리 버킷과 객체를 주고 받습니다.

디렉터리 버킷으로 최고의 성능을 달성하려면 CRT 라이브러리가 포함된 최신 버전의 AWS SDK를 사용하거나 AWS Command Line Interface(AWS CLI)를 사용하는 것이 좋습니다.

디렉터리 버킷으로 배포

디렉터리 버킷을 만든 후에는 지연 시간이 매우 짧은 읽기 및 쓰기를 즉시 시작할 수 있습니다. 가상 프라이빗 클라우드(VPC)를 통한 엔드포인트 연결을 사용하여 디렉터리 버킷과 통신하거나 영역 및 리전 API 작업을 사용하여 객체 및 디렉터리 버킷을 관리할 수 있습니다. AWS SDK, Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), Amazon S3 REST API를 통해 디렉터리 버킷을 사용할 수 있습니다.

주제

- 디렉터리 버킷에 대한 리전 및 영역 엔드포인트
- S3 콘솔, AWS CLI, AWS SDK를 통한 디렉터리 버킷 작업
- 디렉터리 버킷 API 작업

디렉터리 버킷에 대한 리전 및 영역 엔드포인트

가상 프라이빗 클라우드(VPC)에서 디렉터리 버킷에 대한 리전 및 영역 엔드포인트에 액세스하려면 게이트웨이 VPC 엔드포인트를 사용하면 됩니다. 게이트웨이 엔드포인트를 생성한 후 VPC에서 버킷으로 전송되는 트래픽에 대해 해당 엔드포인트를 라우팅 테이블의 대상으로 추가할 수 있습니다. 게이트웨이 엔드포인트 사용에 따르는 추가 요금은 없습니다. 게이트웨이 VPC 엔드포인트를 구성하는 방법에 대한 자세한 내용은 디렉터리 버킷에 대한 네트워킹 섹션을 참조하세요.

버킷 수준(컨트롤 플레인) API 작업은 리전 엔드포인트를 통해 사용할 수 있으며 이를 리전 엔드포인트 API 작업이라고 합니다. 리전 엔드포인트 API 작업의 예로는 `CreateBucket` 및 `DeleteBucket`이 있습니다.

영역(객체 수준 또는 데이터 플레인 엔드포인트 API 작업)을 사용하여 객체를 업로드하고 관리합니다. 영역 엔드포인트 API 작업은 영역 엔드포인트를 통해 사용할 수 있습니다. 영역 API 작업의 예로는 `PutObject` 및 `CopyObject`가 있습니다.

가용 영역의 디렉터리 버킷에 대한 리전 엔드포인트 및 영역 엔드포인트 관련 자세한 내용은 가용 영역의 디렉터리 버킷에 대한 리전 및 영역 엔드포인트 섹션을 참조하세요.

로컬 영역의 디렉터리 버킷에 대한 리전 엔드포인트 및 영역 엔드포인트 관련 자세한 내용은 로컬 영역의 디렉터리 버킷에 대한 개념 섹션을 참조하세요.

S3 콘솔, AWS CLI, AWS SDK를 통한 디렉터리 버킷 작업

AWS SDK, Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), Amazon S3 REST API를 사용하여 S3 Express One Zone 스토리지 클래스 및 디렉터리 버킷을 사용할 수 있습니다.

S3 콘솔

S3 콘솔 사용을 시작하려면 다음 단계를 따르세요.

- 가용 영역에서 디렉터리 버킷 생성
- 디렉터리 버킷 비우기
- 디렉터리 버킷 삭제

전체 튜토리얼은 튜토리얼: S3 Express One Zone 시작하기를 참조하세요.

AWS SDK

S3 Express One Zone은 다음 AWS SDK를 지원합니다.

- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java 2.x
- AWS SDK for JavaScript v3

- AWS SDK for .NET
- AWS SDK for PHP
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby
- AWS SDK for Kotlin
- AWS SDK for Rust

S3 Express One Zone을 사용할 때는 최신 버전의 AWS SDK를 사용하는 것이 좋습니다. S3 Express One Zone용으로 지원되는 AWS SDK는 사용자를 대신하여 세션 설정, 새로 고침 및 종료를 처리합니다. 즉, AWS SDK를 다운로드 및 설치하고 필요한 IAM 권한을 구성한 후에 즉시 API 작업 사용을 시작할 수 있습니다. 자세한 내용은 IAM을 사용한 리전 엔드포인트 API 권한 부여 섹션을 참조하세요.

다운로드 및 설치 방법을 비롯하여 AWS SDK에 대한 자세한 내용은 AWS에서의 빌드를 위한 도구를 참조하세요.

AWS SDK 예시는 다음을 참조하세요.

- 가용 영역에서 디렉터리 버킷 생성
- 디렉터리 버킷 비우기
- 디렉터리 버킷 삭제

AWS Command Line Interface (AWS CLI)

AWS Command Line Interface(AWS CLI)를 사용하여 디렉터리 버킷을 생성하고 S3 Express One Zone에 지원되는 리전 및 영역 엔드포인트 API 작업을 사용할 수 있습니다.

AWS CLI를 시작하려면 AWS CLI 명령 참조의 AWS CLI 시작을 참조하세요.

> **Note**
>
> 고급 `aws s3` 명령과 함께 디렉터리 버킷을 사용하려면 AWS CLI를 최신 버전으로 업데이트하세요. AWS CLI 설치 및 구성 방법에 대한 자세한 내용은 AWS CLI 명령 참조의 최신 버전의 AWS CLI 설치 또는 업데이트를 참조하세요.

AWS CLI 예시는 다음을 참조하세요.

- 가용 영역에서 디렉터리 버킷 생성
- 디렉터리 버킷 비우기
- 디렉터리 버킷 삭제

디렉터리 버킷 API 작업

리전(버킷 수준 또는 컨트롤 플레인) 엔드포인트 API 작업을 사용하여 디렉터리 버킷을 관리할 수 있습니다. 지역(객체 수준 또는 데이터 플레인) 엔드포인트 API 작업을 사용하여 디렉터리 버킷에서 객체를 관리할 수 있습니다. 자세한 내용은 디렉터리 버킷에 대한 네트워킹 및 엔드포인트 및 게이트웨이 VPC 엔드포인트(을)를 참조하세요.

리전 엔드포인트 API 작업

다음은 디렉터리 버킷에서 사용할 수 있는 리전 엔드포인트 API 작업입니다.

- CreateAccessPoint
- CreateBucket
- DeleteAccessPoint
- DeleteAccessPointPolicy
- DeleteAccessPointScope
- DeleteBucket
- DeleteBucketLifecycle
- DeleteBucketPolicy
- GetAccessPoint
- GetAccessPointPolicy
- GetAccessPointScope
- GetBucketLifecycleConfiguration
- GetBucketPolicy
- ListAccessPointsForDirectoryBuckets
- ListDirectoryBuckets
- ListTagsForResource
- PutAccessPointPolicy

- PutAccessPointScope
- PutBucketLifecycleConfiguration
- PutBucketPolicy
- DeleteBucketEncryption
- GetBucketEncryption
- PutBucketEncryption
- TagResource
- UntagResource

영역 엔드포인트 API 작업

다음은 디렉터리 버킷에서 지원되는 영역 엔드포인트 API 작업입니다.

- CreateSession
- CopyObject
- DeleteObject
- DeleteObjects
- GetObject
- GetObjectAttributes
- HeadBucket
- HeadObject
- ListObjectsV2
- PutObject
- RenameObject
- AbortMultipartUpload
- CompleteMultiPartUpload
- CreateMultipartUpload
- ListMultipartUploads
- ListParts
- UploadPart
- UploadPartCopy

S3 디렉터리 버킷에서 태그 사용

AWS 태그는 리소스, 이 경우 Amazon S3 디렉터리 버킷에 대한 메타데이터를 포함하는 키-값 페어입니다. S3 디렉터리 버킷을 생성할 때 태그를 지정하거나 기존 디렉터리 버킷에서 태그를 관리할 수 있습니다. 디렉터리 버킷에서 태그를 사용하는 데 표준 S3 API 요청 속도 외에는 추가 요금이 부과되지 않습니다. 자세한 내용은 Amazon S3 요금을 참조하십시오.

태그 작동 방식

Amazon S3 디렉터리 버킷은 두 가지 유형의 태그를 지원합니다.

- AWS가 생성한 태그: AWS는 이러한 태그를 자동으로 적용하며 사용자가 수정하거나 제거할 수 없습니다. AWS가 생성한 태그에 대한 자세한 내용은 AWS-generated tags를 참조하세요.
- 사용자 정의 태그: 이러한 태그는 사용자가 S3 디렉터리 버킷 또는 기타 리소스에 적용하고 관리합니다.

사용자 정의 태그

사용자 정의 태그는 리소스에 레이블을 지정하는 데 사용하는 태그 키-값 페어입니다. 사용자 정의 태그는 필수 키와 선택적 값으로 구성됩니다. 다음은 사용자 정의 태그의 주요 구성 요소입니다.

태그 키

태그 키는 태그의 필수 이름입니다. 예를 들어 `project`는 다음 태그 키-값 페어의 태그 키입니다.

키	값
`project`	`Trinity`

태그 키는 대/소문자 구분 문자열로, 유니코드 문자 1~128자를 포함해야 합니다.

태그 값

태그 값은 선택적 문자열입니다. 예를 들어 `Trinity`는 이 태그 키-값 페어의 태그 값입니다.

키	값
`project`	`Trinity`

태그 값은 대/소문자 구분 문자열로, 유니코드 문자 0~256자를 포함할 수 있습니다.

사용자 정의 태그에서 허용되는 문자와 기타 제한에 대한 세부 정보는 AWS 결제 및 비용 관리 사용 설명서의 User-Defined Tag Restrictions을 참조하세요.

태그 세트

각 S3 디렉터리 버킷에는 해당 버킷에 할당된 모든 태그 키와 값 페어를 포함하는 하나의 태그 세트가 있습니다. 태그 세트는 최대 50개의 사용자 정의 태그를 포함하거나 비어 있을 수 있습니다.

각 키는 태그 세트 내에서 고유해야 하지만 동일한 값을 여러 번 사용할 수 있습니다. 예를 들어 다음 두 태그 키의 값이 `Trinity`로 동일할 수 있습니다.

키	값
`project`	`Trinity`
`cost-center`	`Trinity`

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

AWS은(는) 태그에 의미론적 의미를 적용하지 않습니다. 태그는 엄격히 문자열로 해석됩니다.

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 또는 Amazon S3 API를 사용하여 태그를 추가, 나열, 수정 또는 삭제할 수 있습니다.

태그를 사용하는 일반적인 방법

다음을 위해 S3 디렉터리 버킷에 태그를 사용합니다.

1. 비용 할당 - AWS 결제 및 비용 관리에서 버킷 태그별로 스토리지 비용을 추적합니다.
2. 속성 기반 액세스 제어(ABAC) - 액세스 권한을 확장하고 태그를 기반으로 S3 디렉터리 버킷에 대한 액세스 권한을 부여합니다.

> **Note**
>
> 비용 할당과 액세스 제어에 동일한 태그를 사용할 수 있습니다.

비용 할당에 태그 사용

S3 디렉터리 버킷에 태그를 적용하고 이러한 태그를 비용 할당에 활성화하여 Amazon S3 스토리지 비용을 추적합니다.

비용 추적을 시작하는 방법:

1. S3 디렉터리 버킷에 태그를 추가하거나 기존 태그를 사용합니다. 디렉터리 버킷에 사용자 정의 태그를 추가하는 방법에 대한 자세한 내용은 태그 작업 섹션을 참조하세요. 예를 들어 프로젝트 또는 프로젝트 그룹을 식별하는 태그로 버킷에 레이블을 지정할 수 있습니다.
2. AWS 결제 및 비용 관리 콘솔에서 비용 할당을 위해 태그를 활성화합니다. AWS 결제 및 비용 관리 사용 설명서에서 Activating user-defined cost allocation tags를 참조하세요. 사용자 정의 태그 또는 AWS가 생성한 태그를 활성화할 수 있습니다. 자세한 내용은 Organizing and tracking costs using AWS cost allocation tags를 참조하세요.

AWS는 활성화된 태그를 사용하여 다음과 같은 다양한 결제 및 비용 관리 도구에서 리소스 비용을 정리합니다.

- 비용 할당 보고서

  활성화된 태그별로 정리된 비용에 대한 개괄적인 보기를 제공합니다. 자세한 내용은 AWS 결제 사용 설명서에서 Using the monthly cost allocation report를 참조하세요.
- 비용 및 사용량 보고서(CUR)

  태그 기반 비용 분석을 포함하여 가장 상세한 AWS 비용 및 사용량 데이터 세트를 제공합니다. 자세한 내용은 AWS Data Exports 사용 설명서의 What are AWS Cost and Usage Reports?를 참조하세요.

속성 기반 액세스 제어(ABAC)에 태그 사용

속성 기반 액세스 제어(ABAC)는 속성, 즉 태그를 기반으로 권한을 정의하는 권한 부여 전략입니다. AWS Identity and Access Management(IAM) 엔터티(사용자 또는 역할) 및 Amazon S3 디렉터리 버킷과 같은 AWS 리소스에 태그를 연결할 수 있습니다. 그런 다음 액세스 제어 정책의 태그 기반 조건을 사용해 이러한 리소스에 대한 권한을 제어하여 이러한 조건이 충족될 때 작업을 허용하거나 거부합니다.

S3 디렉터리 버킷의 ABAC

Amazon S3 디렉터리 버킷은 태그를 사용한 속성 기반 액세스 제어(ABAC)를 지원합니다. AWS 조직, IAM 및 S3 디렉터리 버킷 정책에서 태그 기반 조건 키를 사용합니다. 엔터프라이즈의 경우 Amazon S3의 ABAC는 여러 AWS 계정에 대한 권한 부여를 지원합니다.

IAM 정책에서 다음 전역 조건 키를 사용하여 버킷의 태그를 기반으로 S3 디렉터리 버킷에 대한 액세스를 제어할 수 있습니다.

- `aws:ResourceTag/key-name`
  - 이 키를 사용하여 정책에서 지정한 태그 키-값 쌍을 리소스에 연결된 키-값 쌍과 비교합니다. 예를 들어 리소스에 값이 `Marketing`인 태그 키 `Dept`와 연결된 경우에만 리소스에 대한 액세스가 필요할 수 있습니다. 자세한 내용은 AWS 리소스에 대한 액세스 제어를 참조하십시오.
- `aws:RequestTag/key-name`
  - 이 키를 사용하여 요청에서 전달된 태그 키 값 페어를 정책에서 지정한 태그 페어와 비교합니다. 예를 들어, 요청에 태그 키 `Dept`가 포함되어 있으며 값이 `Accounting`인지 확인할 수 있습니다. 자세한 내용은 AWS 요청 중 액세스 제어를 참조하세요. 이 조건 키를 사용하여 `TagResource` 및 `CreateBucket` API 작업 중에 전달할 수 있는 태그 키-값 페어를 제한할 수 있습니다.
- `aws:TagKeys`
  - 이 키를 사용하여 요청의 태그 키를 정책에서 지정한 키와 비교합니다. 정책을 사용하여 태그를 통해 액세스를 제어할 때 `aws:TagKeys` 조건 키를 사용하여 어떤 태그 키가 허용되는지 정의하는 것이 좋습니다. 예시 정책과 자세한 내용은 태그 키를 기반으로 액세스 제어를 참조하세요. 태그가 있는 S3 디렉터리 버킷을 생성할 수 있습니다. `CreateBucket` API 작업 중에 태그 지정을 허용하려면 `s3express:TagResource` 및 `s3express:CreateBucket` 작업을 모두 포함하는 정책을 생성해야 합니다. 그런 다음 `aws:TagKeys` 조건 키를 사용하여 `CreateBucket` 요청 내 특정 태그 사용을 강제 적용할 수 있습니다.
- `s3express:BucketTag/tag-key`
  - 이 조건 키를 사용하여 태그를 사용하는 디렉터리 버킷의 특정 데이터에 권한을 부여합니다. 액세스 포인트를 사용하여 디렉터리 버킷에 액세스할 때 이 조건 키는 액세스 포인트 및 디렉터리 버킷에 대해 권한을 부여할 때 디렉터리 버킷의 태그를 참조하는 반면, `aws:ResourceTag/tag-key`는 권한이 부여된 리소스의 태그만 참조합니다.

정책 예제

Amazon S3 디렉터리 버킷에 대한 다음 예시 ABAC 정책을 참조하세요.

1.1 - 특정 태그가 있는 버킷을 생성하거나 수정하는 IAM 정책

이 IAM 정책에서 이 정책을 사용하는 사용자 또는 역할은 버킷 생성 요청의 태그 키 `project` 및 태그 값 `Trinity`로 버킷에 태그를 지정하는 경우에만 S3 디렉터리 버킷을 생성할 수 있습니다. `TagResource` 요청에 태그 키-값 페어 `project:Trinity`가 포함되어 있는 한 기존 S3 디렉터리 버킷에 태그를 추가하거나 수정할 수도 있습니다. 이 정책은 버킷 또는 해당 객체에 대한 읽기, 쓰기 또는 삭제 권한을 부여하지 않습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateBucketWithTags",
      "Effect": "Allow",
      "Action": [
        "s3express:CreateBucket",
        "s3express:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/project": [
            "Trinity"
          ]
        }
      }
    }
  ]
}
```

1.2 - 태그를 사용하여 버킷에 대한 작업을 제한하는 버킷 정책

이 버킷 정책에서 IAM 위탁자(사용자 및 역할)는 버킷의 `project` 태그의 값이 위탁자의 `project` 태그의 값과 일치하는 경우에만 버킷에 대한 `CreateSession` 작업을 사용하여 작업을 수행할 수 있습니다.

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

```
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": "s3express:CreateSession",
      "Resource": "arn:aws:s3express:us-west-2:111122223333:bucket/amzn-s3-demo-
bucket--usw2-az1--x-s3",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/project": "${aws:PrincipalTag/project}"
        }
      }
    }
  ]
}
```

1.3 - 태그 지정 거버넌스를 유지 관리하는 기존 리소스의 태그를 수정하는 IAM 정책

이 IAM 정책에서 IAM 위탁자(사용자 또는 역할)는 버킷의 `project` 태그 값이 위탁자의 `project` 태그 값과 일치하는 경우에만 버킷의 태그를 수정할 수 있습니다. 이러한 디렉터리 버킷에는 `aws:TagKeys` 조건 키에 지정된 태그 네 개, 즉 `project`, `environment`, `owner` 및 `cost-center`만 허용됩니다. 이렇게 하면 태그 거버넌스를 강제 적용하고, 무단 태그 수정을 방지하고, 버킷 간에 태그 지정 스키마를 일관되게 유지할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EnforceTaggingRulesOnModification",
      "Effect": "Allow",
      "Action": [
        "s3express:TagResource"
      ],
      "Resource": "arn:aws:s3express:*:*:bucket/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/project": "${aws:PrincipalTag/project}"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": [
            "project",
            "environment",
            "owner",
            "cost-center"
```

```
          ]
        }
      }
    }
  ]
}
```

1.4 - s3express:BucketTag 조건 키 사용

이 IAM 정책에서 조건문은 버킷에 태그 키 `Environment` 및 태그 값 `Production`이 있는 경우에만 버킷의 데이터에 대한 액세스를 허용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessToSpecificAccessPoint",
      "Effect": "Allow",
      "Action": "*",
      "Resource": "arn:aws:s3express:*:*:accesspoint/*",
      "Condition": {
        "StringEquals": {
          "s3express:BucketTag/Environment": "Production"
        }
      }
    }
  ]
}
```

태그 작업

Amazon S3 콘솔, AWS Command Line Interface(CLI), AWS SDK를 사용하거나 S3 API TagResource, UntagResource 및 ListTagsForResource를 사용하여 S3 디렉터리 버킷의 태그를 추가하거나 관리할 수 있습니다. 자세한 내용은 다음을 참조하세요.

주제

- 태그가 있는 디렉터리 버킷 생성
- 디렉터리 버킷에 태그 추가
- 디렉터리 버킷 태그 보기
- 디렉터리 버킷에서 객체 삭제

태그가 있는 디렉터리 버킷 생성

Amazon S3 디렉터리 버킷을 생성할 때 태그를 지정할 수 있습니다. 디렉터리 버킷에서 태그를 사용하는 데 표준 S3 API 요청 속도 외에는 추가 요금이 부과되지 않습니다. 자세한 내용은 Amazon S3 요금을 참조하십시오. 디렉터리 버킷 태그 지정에 대한 자세한 내용은 S3 디렉터리 버킷에서 태그 사용 섹션을 참조하세요.

권한

태그가 있는 디렉터리 버킷을 생성하려면 다음 권한이 있어야 합니다.

- `s3express:CreateBucket`
- `s3express:TagResource`

오류 해결

태그가 있는 디렉터리 버킷을 생성하려고 할 때 오류가 발생하면 다음을 수행할 수 있습니다.

- 디렉터리 버킷을 생성하고 여기에 태그를 추가하는 데 필요한 권한이 있는지 확인합니다.
- IAM 사용자 정책에서 속성 기반 액세스 제어(ABAC) 조건을 확인합니다. 디렉터리 버킷에 특정 태그 키 및 값으로만 레이블을 지정해야 할 수 있습니다. 자세한 내용은 속성 기반 액세스 제어(ABAC)에 태그 사용 섹션을 참조하세요.

단계

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), Amazon S3 REST API, AWS SDK를 사용하여 태그가 적용된 디렉터리 버킷을 생성할 수 있습니다.

S3 콘솔 사용

Amazon S3 콘솔을 사용하여 태그가 있는 디렉터리 버킷을 만드는 방법:

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔에 로그인합니다.
2. 왼쪽 탐색 창에서 디렉터리 버킷을 선택합니다.
3. 새 디렉터리 버킷을 만들려면 버킷 만들기를 선택합니다.
4. 두 가지 유형의 디렉터리 버킷을 생성할 수 있습니다.

   고성능 워크로드를 위해 가용 영역에 디렉터리 버킷을 생성합니다. 자세한 내용은 고성능 워크로드 섹션을 참조하세요.

로컬 영역에서 데이터 레지던시 워크로드를 위한 디렉터리 버킷을 생성합니다. 자세한 내용은 데이터 레지던시 워크로드 섹션을 참조하세요.

5. 두 디렉터리 버킷 유형 모두 버킷 만들기 페이지에서 새 디렉터리 버킷을 생성할 때 태그가 옵션입니다.

6. 버킷의 이름을 입력합니다. 자세한 내용은

   Amazon S3에서 디렉터리 버킷을 생성할 때 다음 버킷 이름 지정 규칙이 적용됩니다. 범용 버킷의 이름 지정 규칙은 범용 버킷 이름 지정 규칙 섹션을 참조하세요.

   디렉터리 버킷 이름은 사용자가 제공하는 기본 이름과 버킷이 위치한 AWS 영역(가용 영역 또는 로컬 영역)의 ID 및 `--x-s3`이 포함된 접미사로 구성됩니다. *`zone-id`*는 가용 영역 또는 로컬 영역의 ID일 수 있습니다.

   ```
   base-name--zoneid--x-s3
   ```

   예를 들어, 다음 디렉터리 버킷 이름에는 가용 영역 ID인 `usw2-az1`이 포함되어 있습니다.

   ```
   bucket-base-name--usw2-az1--x-s3
   ```

   > **Note**
   >
   > 콘솔을 사용하여 디렉터리 버킷을 생성하면 제공하는 기본 이름에 접미사가 자동으로 추가됩니다. 이 접미사에는 선택한 영역(가용 영역 또는 로컬 영역)의 영역 ID가 포함됩니다.

   > API를 사용하여 디렉터리 버킷을 생성할 때는 요청에 영역 ID를 포함한 전체 접미사를 제공해야 합니다. 영역 ID 목록은 엔드포인트 섹션을 참조하세요.

   디렉터리 버킷에는 다음 이름 지정 규칙이 적용됩니다.

   - 선택한 영역(AWS 가용 영역 또는 AWS 로컬 영역) 내에서 고유해야 합니다.
   - 이름은 최소 3자에서 최대 63자 사이여야 합니다(접미사 포함).
   - 소문자, 숫자, 하이픈(-)으로만 구성해야 합니다.
   - 문자나 숫자로 시작하고 끝나야 합니다.
   - *`--zone-id`*`--x-s3`을 접미사로 포함해야 합니다.

- 버킷 이름은 접두사 `xn--`로 시작해서는 안됩니다.
- 버킷 이름은 접두사 `sthree-`로 시작해서는 안됩니다.
- 버킷 이름은 접두사 `sthree-configurator`로 시작해서는 안됩니다.
- 버킷 이름은 접두사 `amzn-s3-demo-`로 시작해서는 안됩니다.
- 버킷 이름은 접미사 `-s3alias`로 끝나서는 안됩니다. 이 접미사는 액세스 포인트 별칭 이름 용도로 예약되어 있습니다. 자세한 내용은 액세스 포인트 별칭 섹션을 참조하세요.
- 버킷 이름은 접미사 `--ol-s3`로 끝나서는 안됩니다. 이 접미사는 객체 Lambda 액세스 포인트 별칭 이름 용도로 예약되어 있습니다. 자세한 내용은 S3 버킷 객체 Lambda 액세스 포인트에 버킷 스타일 별칭을 사용하는 방법 섹션을 참조하세요.
- 버킷 이름은 접미사 `.mrap`로 끝나서는 안됩니다. 이 접미사는 다중 리전 액세스 포인트 이름 용도로 예약되어 있습니다. 자세한 내용은 Amazon S3 다중 리전 액세스 포인트 이름 지정 규칙 섹션을 참조하세요.

   섹션을 참조하세요.

7. 새 태그 추가를 선택하여 태그 편집기를 열고 태그 키-값 페어를 입력합니다. 태그 키는 필수지만 값은 선택 사항입니다.
8. 다른 태그를 추가하려면 새 태그 추가를 다시 선택합니다. 최대 50개의 태그 키-값 페어를 입력할 수 있습니다.
9. 새 디렉터리 버킷에 대한 옵션 지정을 완료한 후 버킷 만들기를 선택합니다.

AWS SDK 사용

SDK for Java 2.x

이 예시는 AWS SDK for Java 2.x를 사용하여 태그가 있는 디렉터리 버킷을 생성하는 방법을 보여 줍니다. 명령을 사용하려면 `### ## ## ###`를 사용자의 정보로 대체합니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.LocationInfo;
```

```
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.Tag;

public class CreateBucketWithTagsExample {
    public static void createBucketWithTagsExample() {
        S3Client s3 = S3Client.builder().region(Region.US_WEST_2).build();

        CreateBucketConfiguration bucketConfiguration =
 CreateBucketConfiguration.builder()
                .location(LocationInfo.builder()
                        .type(LocationType.AVAILABILITY_ZONE)
                        .name("usw2-az1").build())
                .bucket(BucketInfo.builder()
                        .type(BucketType.DIRECTORY)
                        .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                        .build())
                .tags(Tag.builder().key("MyTagKey").value("MyTagValue").build())
                .build();

        CreateBucketRequest createBucketRequest = CreateBucketRequest.builder()
                .bucket("amzn-s3-demo-bucket--usw2-az1--x-s3--usw2-az1--x-s3")
                .createBucketConfiguration(bucketConfiguration)
                .build();

        CreateBucketResponse response = s3.createBucket(createBucketRequest);
        System.out.println("Status code (should be 200):");
        System.out.println(response.sdkHttpResponse().statusCode());
    }
}
```

REST API 사용

태그가 있는 디렉터리 버킷 생성을 위한 Amazon S3 REST API 지원에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 다음 섹션을 참조하세요.

- CreateBucket

AWS CLI 사용

AWS CLI를 설치하려면 AWS Command Line Interface 사용 설명서에서 AWS 설치를 참조하세요.

다음 CLI 예시는 AWS CLI를 사용하여 태그가 있는 디렉터리 버킷을 생성하는 방법을 보여줍니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

디렉터리 버킷을 생성할 때는 구성 세부 정보를 제공하고 다음과 같은 명명 규칙을 사용해야 합니다. *bucket-base-name*--*zone-id*--x-s3

요청:

```
aws s3api create-bucket \
--bucket bucket-base-name--zone-id--x-s3 \
--create-bucket-configuration "Location={Type=AvailabilityZone,Name=zone-
id},Bucket={DataRedundancy=SingleAvailabilityZone,Type=Directory},Tags=[{Key=mykey1,Value=myval
 {Key=mykey2,Value=myvalue2}]"
```

응답:

```
{
  "Location": "http://bucket--use1-az4--x-s3.s3express-use1-az4.us-
east-1.amazonaws.com/"
}
```

디렉터리 버킷에 태그 추가

Amazon S3 디렉터리 버킷에 태그를 추가하고 이러한 태그를 수정할 수 있습니다. 디렉터리 버킷에서 태그를 사용하는 데 표준 S3 API 요청 속도 외에는 추가 요금이 부과되지 않습니다. 자세한 내용은 Amazon S3 요금을 참조하십시오. 디렉터리 버킷 태그 지정에 대한 자세한 내용은 S3 디렉터리 버킷에서 태그 사용 섹션을 참조하세요.

권한

디렉터리 버킷에 태그를 추가하려면 다음 권한이 있어야 합니다.

- `s3express:TagResource`

오류 해결

디렉터리 버킷에 태그를 추가하려고 할 때 오류가 발생하면 다음을 수행할 수 있습니다.

- 디렉터리 버킷에 태그를 추가하는 데 필요한 권한이 있는지 확인합니다.

- AWS 예약된 접두사 `aws:`로 시작하는 태그 키를 추가하려고 했다면 태그 키를 변경하고 다시 시도합니다.

단계

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), Amazon S3 REST API, AWS SDK를 사용해 디렉터리 버킷에 태그를 추가할 수 있습니다.

S3 콘솔 사용

Amazon S3 콘솔을 사용하여 디렉터리 버킷에 태그를 추가하는 방법:

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔에 로그인합니다.
2. 왼쪽 탐색 창에서 디렉터리 버킷을 선택합니다.
3. 버킷 이름을 선택합니다.
4. 속성(Properties) 탭을 선택합니다.
5. 태그 섹션으로 스크롤하고 새 태그 추가를 선택합니다.
6. 태그 추가 페이지가 열립니다. 최대 50개의 태그 키-값 페어를 입력할 수 있습니다.
7. 키 이름이 기존 태그와 동일한 새 태그를 추가하는 경우 새 태그의 값이 기존 태그의 값을 덮어씁니다.
8. 이 페이지에서 기존 태그의 값을 편집할 수도 있습니다.
9. 태그를 추가한 후 변경 사항 저장을 선택합니다.

AWS SDK 사용

SDK for Java 2.x

이 예시는 AWS SDK for Java 2.x를 사용하여 디렉터리 버킷에 태그를 추가하는 방법을 보여줍니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.Tag;
import software.amazon.awssdk.services.s3control.model.TagResourceRequest;
import software.amazon.awssdk.services.s3control.model.TagResourceResponse;

public class TagResourceExample {
    public static void tagResourceExample() {
```

```
        S3ControlClient s3Control =
 S3ControlClient.builder().region(Region.US_WEST_2).build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                .resourceArn("arn:aws:s3express:us-west-2:111122223333:bucket/my-
directory-bucket--usw2-az1--x-s3")
                .accountId("111122223333")
                .tags(Tag.builder().key("MyTagKey").value("MyTagValue").build())
                .build();

        TagResourceResponse response = s3Control.tagResource(tagResourceRequest);
        System.out.println("Status code (should be 204):");
        System.out.println(response.sdkHttpResponse().statusCode());
    }
}
```

REST API 사용

디렉터리 버킷에 태그 추가를 위한 Amazon S3 REST API 지원에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 다음 섹션을 참조하세요.

- TagResource

AWS CLI 사용

AWS CLI를 설치하려면 AWS Command Line Interface 사용 설명서에서 AWS 설치를 참조하세요.

다음 CLI 예시는 AWS CLI를 사용하여 디렉터리 버킷에 태그를 추가하는 방법을 보여줍니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

요청:

```
aws s3control tag-resource \
--account-id 111122223333 \
--resource-arn arn:aws:s3express:us-east-1:444455556666:bucket/prefix--use1-az4--x-s3 \
--tags "Key=mykey,Value=myvalue"
```

응답:

```
{
  "ResponseMetadata": {
```

```
        "RequestId": "EXAMPLE123456789",
        "HTTPStatusCode": 200,
        "HTTPHeaders": {
            "date": "Wed, 19 Jun 2025 10:30:00 GMT",
            "content-length": "0"
        },
        "RetryAttempts": 0
    }
}
```

디렉터리 버킷 태그 보기

S3 디렉터리 버킷에 적용된 태그를 보거나 나열할 수 있습니다. 태그에 대한 자세한 내용은 S3 디렉터리 버킷에서 태그 사용 섹션을 참조하세요.

권한

디렉터리 버킷에 적용된 태그를 보려면 다음 권한이 있어야 합니다.

- `s3express:ListTagsForResource`

오류 해결

디렉터리 버킷의 태그를 나열하거나 보려고 할 때 오류가 발생하면 다음을 수행할 수 있습니다.

- 디렉터리 버킷의 태그를 보거나 나열하는 데 필요한 권한이 있는지 확인합니다.

단계

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), Amazon S3 REST API 및 AWS SDK를 사용하여 디렉터리 버킷에 적용된 태그를 볼 수 있습니다.

S3 콘솔 사용

Amazon S3 콘솔을 사용하여 디렉터리 버킷에 적용된 태그를 보는 방법:

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔에 로그인합니다.
2. 왼쪽 탐색 창에서 디렉터리 버킷을 선택합니다.
3. 버킷 이름을 선택합니다.
4. 속성(Properties) 탭을 선택합니다.

5. 태그 섹션으로 스크롤하여 디렉터리 버킷에 적용된 모든 태그를 봅니다.
6. 태그 섹션에는 기본적으로 사용자 정의 태그가 표시됩니다. AWS가 생성한 태그 탭을 선택하여 AWS 서비스가 디렉터리 버킷에 적용한 태그를 볼 수 있습니다.

AWS SDK 사용

이 섹션에서는 AWS SDK를 사용하여 디렉터리 버킷에 적용된 태그를 보는 방법의 예를 제공합니다.

SDK for Java 2.x

이 예시는 AWS SDK for Java 2.x를 사용하여 디렉터리 버킷에 적용된 태그를 보는 방법을 보여줍니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.ListTagsForResourceRequest;
import software.amazon.awssdk.services.s3control.model.ListTagsForResourceResponse;

public class ListTagsForResourceExample {
    public static void listTagsForResourceExample() {
        S3ControlClient s3Control =
 S3ControlClient.builder().region(Region.US_WEST_2).build();

        ListTagsForResourceRequest listTagsForResourceRequest =
 ListTagsForResourceRequest.builder()
                .resourceArn("arn:aws:s3express:us-west-2:111122223333:bucket/my-
directory-bucket--usw2-az1--x-s3")
                .accountId("111122223333")
                .build();

        ListTagsForResourceResponse response =
 s3Control.listTagsForResource(listTagsForResourceRequest);
        System.out.println("Tags on my resource:");
        System.out.println(response.toString());
    }
}
```

REST API 사용

디렉터리 버킷에 적용된 태그 보기를 위한 Amazon S3 REST API 지원에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 다음 섹션을 참조하세요.

- ListTagsforResource

AWS CLI 사용

AWS CLI를 설치하려면 AWS Command Line Interface 사용 설명서에서 AWS 설치를 참조하세요.

다음 CLI 예시에서는 디렉터리 버킷에 적용된 태그를 보는 방법을 보여줍니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

요청:

```
aws s3control list-tags-for-resource \
--account-id 111122223333 \
--resource-arn arn:aws:s3express:us-east-1:444455556666:bucket/prefix--use1-az4--x-s3 \
```

응답 - 태그 있음:

```
{
  "Tags": [
      {
          "Key": "MyKey1",
          "Value": "MyValue1"
      },
      {
          "Key": "MyKey2",
          "Value": "MyValue2"
      },
      {
          "Key": "MyKey3",
          "Value": "MyValue3"
      }
  ]
}
```

응답 - 태그 없음:

```
{
  "Tags": []
}
```

디렉터리 버킷에서 객체 삭제

S3 디렉터리 버킷에서 태그를 제거할 수 있습니다. AWS 태그는 리소스, 이 경우 Amazon S3 디렉터리 버킷에 대한 메타데이터를 포함하는 키-값 페어입니다. 태그에 대한 자세한 내용은 S3 디렉터리 버킷에서 태그 사용 섹션을 참조하세요.

> **Note**
>
> 태그를 삭제하고 나중에 비용 추적 또는 액세스 제어에 사용 중이었다는 것을 알게 되면 디렉터리 버킷에 태그를 다시 추가할 수 있습니다.

권한

디렉터리 버킷에서 태그를 삭제하려면 다음 권한이 있어야 합니다.

- `s3express:UntagResource`

단계

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), Amazon S3 REST API 및 AWS SDK를 사용하여 디렉터리 버킷에서 태그를 삭제할 수 있습니다.

S3 콘솔 사용

Amazon S3 콘솔을 사용하여 디렉터리 버킷에서 태그를 삭제하는 방법:

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔에 로그인합니다.
2. 왼쪽 탐색 창에서 디렉터리 버킷을 선택합니다.
3. 버킷 이름을 선택합니다.
4. 속성(Properties) 탭을 선택합니다.
5. 태그 섹션으로 스크롤하여 삭제하려는 태그 옆의 확인란을 선택합니다.
6. 삭제를 선택합니다.
7. 사용자 정의 태그 삭제 팝업이 나타나고 선택한 태그를 확인하라는 메시지가 표시됩니다.
8. [삭제]를 선택하여 확인합니다.

AWS SDK 사용

SDK for Java 2.x

이 예시는 AWS SDK for Java 2.x를 사용하여 디렉터리 버킷에서 태그를 삭제하는 방법을 보여줍니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.UntagResourceRequest;
import software.amazon.awssdk.services.s3control.model.UntagResourceResponse;

public class UntagResourceExample {
    public static void untagResourceExample() {
        S3ControlClient s3Control =
 S3ControlClient.builder().region(Region.US_WEST_2).build();

        UntagResourceRequest untagResourceRequest = UntagResourceRequest.builder()
                .resourceArn("arn:aws:s3express:us-west-2:111122223333:bucket/my-
directory-bucket--usw2-az1--x-s3")
                .accountId("111122223333")
                .tagKeys("myTagKey")
                .build();

        UntagResourceResponse response =
 s3Control.untagResource(untagResourceRequest);
        System.out.println("Status code (should be 204):");
        System.out.println(response.sdkHttpResponse().statusCode());
    }
}
```

REST API 사용

디렉터리 버킷에서 태그 삭제를 위한 Amazon S3 REST API 지원에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 다음 섹션을 참조하세요.

- UnTagResource

AWS CLI 사용

AWS CLI를 설치하려면 AWS Command Line Interface 사용 설명서에서 AWS 설치를 참조하세요.

다음 CLI 예시는 AWS CLI를 사용하여 디렉터리 버킷에서 태그를 삭제하는 방법을 보여줍니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

요청:

```
aws s3control untag-resource \
--account-id 111122223333 \
--resource-arn arn:aws:s3express:us-east-1:444455556666:bucket/prefix--use1-az4--x-s3 \
--tag-keys "tagkey1" "tagkey2"
```

응답:

```
{
  "ResponseMetadata": {
    "RequestId": "EXAMPLE123456789",
    "HTTPStatusCode": 204,
    "HTTPHeaders": {
        "date": "Wed, 19 Jun 2025 10:30:00 GMT",
        "content-length": "0"
    },
    "RetryAttempts": 0
  }
}
```

Amazon S3 Tables 및 테이블 버킷 작업

Amazon S3 Tables은 쿼리 성능을 지속적으로 개선하고 테이블의 스토리지 비용을 절감하도록 설계된 기능이 내장되어 있으며 분석 워크로드에 최적화된 S3 스토리지를 제공합니다. S3 Tables은 일별 구매 트랜잭션, 스트리밍 센서 데이터 또는 광고 노출과 같은 테이블 형식 데이터를 저장하기 위해 특별히 빌드되었습니다. 테이블 형식 데이터는 데이터베이스 테이블과 같이 열과 행의 데이터를 나타냅니다.

S3 Tables의 데이터는 테이블을 하위 리소스로 저장하는 테이블 버킷이라는 새 버킷 유형에 저장됩니다. 테이블 버킷은 Apache Iceberg 형식의 테이블 저장을 지원합니다. 표준 SQL 문을 사용하여 Amazon Athena, Amazon Redshift 및 Apache Spark와 같이 Iceberg를 지원하는 쿼리 엔진으로 테이블을 쿼리할 수 있습니다.

주제

- S3 Tables의 특징
- 관련 서비스
- 자습서: S3 Tables 시작하기
- 테이블 버킷
- S3 Tables 유지 관리
- 테이블 네임스페이스
- S3 테이블 버킷의 테이블
- 테이블 데이터에 액세스
- S3 Tables AWS 리전, 엔드포인트 및 서비스 할당량
- S3 Tables의 보안
- S3 Tables에서 AWS CloudTrail을 사용하여 로깅

S3 Tables의 특징

테이블을 위해 특별히 빌드된 스토리지

> S3 테이블 버킷은 테이블용으로 특별히 설계되었습니다. 테이블 버킷은 S3 범용 버킷의 자체 관리형 테이블에 비해 초당 트랜잭션(TPS)이 더 높고 쿼리 처리량이 더 우수합니다. 테이블 버킷은 다른 Amazon S3 버킷 유형과 동일한 내구성, 가용성 및 확장성을 제공합니다.

Apache Iceberg에 대한 기본 지원

테이블 버킷의 테이블은 Apache Iceberg 형식으로 저장됩니다. Iceberg를 지원하는 쿼리 엔진에서 표준 SQL을 사용하여 이러한 테이블을 쿼리할 수 있습니다. Iceberg에는 스키마 진화 및 파티션 진화를 포함하여 쿼리 성능을 최적화하는 다양한 기능이 있습니다.

Iceberg를 사용하면 쿼리를 다시 작성하거나 데이터 구조를 다시 빌드할 필요 없이 시간이 지남에 따라 데이터가 진화할 수 있도록 데이터 구성 방식을 변경할 수 있습니다. Iceberg는 트랜잭션 지원을 통해 데이터 일관성과 신뢰성을 보장하도록 설계되었습니다. 문제를 수정하거나 시간 이동 쿼리를 수행하는 데 도움이 되도록 시간이 지남에 따라 데이터가 어떻게 변경되는지 추적하고 과거 버전으로 롤백할 수 있습니다.

자동화된 테이블 최적화

쿼리를 위해 테이블을 최적화하기 위해 S3는 압축, 스냅샷 관리 및 참조되지 않은 파일 제거와 같은 자동 유지 관리 작업을 지속적으로 수행합니다. 이러한 작업은 작은 객체를 개수가 더 적고 크기가 더 큰 파일로 압축하여 테이블 성능을 높입니다. 또한 유지 관리 작업은 미사용 객체를 정리하여 스토리지 비용을 절감합니다. 이 자동 유지 관리는 수동 테이블 유지 관리의 필요성을 줄여 대규모 데이터 레이크의 운영을 간소화합니다. 각 테이블 및 테이블 버킷에 대해 유지 관리 구성을 사용자 지정할 수 있습니다.

액세스 관리 및 보안

AWS Identity and Access Management(IAM) 및 AWS Organizations의 서비스 제어 정책을 사용하여 테이블 버킷과 개별 테이블 모두에 대한 액세스를 관리할 수 있습니다. S3 Tables은 Amazon S3와 다른 서비스 네임스페이스인 s3tables 네임스페이스를 사용합니다. 따라서 S3 Tables 서비스 및 해당 리소스에 대한 정책을 특별히 설계할 수 있습니다. 개별 테이블, 테이블 네임스페이스 내의 모든 테이블 또는 전체 테이블 버킷에 대한 액세스 권한을 부여하는 정책을 설계할 수 있습니다. 모든 Amazon S3 Block Public Access 설정은 테이블 버킷에 대해 항상 활성화되며 비활성화할 수 없습니다.

AWS 분석 서비스와의 통합

S3 콘솔을 통해 Amazon S3 Tables 버킷을 Amazon SageMaker Lakehouse와 자동으로 통합할 수 있습니다. 이 통합을 통해 AWS 분석 서비스는 AWS Glue Data Catalog를 통해 테이블 데이터를 자동으로 검색하고 액세스할 수 있습니다. 통합 후 Amazon Athena, Amazon Redshift, QuickSight 등과 같은 분석 서비스를 사용하여 테이블에 대한 작업을 수행할 수 있습니다. 통합 작동 방식에 대한 자세한 내용은 AWS 분석 서비스에서 Amazon S3 Tables 사용 섹션을 참조하세요.

관련 서비스

S3 Tables에서 다음 AWS 서비스를 사용하여 특정 분석 애플리케이션을 지원할 수 있습니다.

- Amazon Athena – Athena는 표준 SQL을 사용해 Amazon S3에서 직접 데이터를 분석하는 데 사용할 수 있는 대화식 쿼리 서비스입니다. 또한 Athena를 사용하면 리소스를 계획, 구성 또는 관리할 필요 없이 Apache Spark를 사용하여 데이터 분석을 대화식으로 실행할 수 있습니다. Athena에서 Apache Spark 애플리케이션을 실행하는 경우 처리를 위해 Spark 코드를 제출하고 결과를 직접 수신합니다.
- AWS Glue - AWS Glue는 여러 소스의 데이터를 검색, 준비, 이동, 통합할 수 있도록 하는 서버리스 데이터 통합 서비스입니다. 분석, 기계 학습(ML) 및 애플리케이션 개발에 AWS Glue를 사용할 수 있습니다. AWS Glue에도 작성, 작업 실행, 비즈니스 워크플로 구현을 위한 추가 생산성 및 데이터 운영 도구가 있습니다.
- Amazon EMR – Amazon EMR은 AWS에서 Apache Hadoop 및 Apache Spark와 같은 빅 데이터 프레임워크 실행을 단순화하여 방대한 양의 데이터를 처리하고 분석하는 관리형 클러스터 플랫폼입니다.
- Amazon Redshift - Amazon Redshift는 페타바이트 규모의 클라우드 내 데이터 웨어하우스 서비스입니다. Amazon Redshift Serverless를 사용하면 프로비저닝된 데이터 웨어하우스를 구성하지 않아도 데이터를 액세스하고 분석할 수 있습니다. 리소스가 자동으로 프로비저닝하고 데이터 웨어하우스 용량이 지능적으로 크기 조정되어 가장 까다롭고 예측할 수 없는 워크로드에도 빠른 성능을 제공합니다. 데이터 웨어하우스가 유휴 상태일 때는 요금이 발생하지 않으므로 사용량에 대한 요금만 지불합니다. Amazon Redshift 쿼리 에디터 v2 또는 자주 사용하는 비즈니스 인텔리전스(BI) 도구에서 바로 데이터를 로드하고 쿼리를 시작할 수 있습니다.
- QuickSight - QuickSight는 데이터를 사용하여 시각적 객체를 구축하고, 애드혹 분석을 수행하고, 사업과 관련된 인사이트를 빠르게 얻을 수 있는 비즈니스 분석 서비스입니다. QuickSight는 QuickSight Super-fast, Parallel, In-Memory, Calculation Engine(SPICE)을 사용하여 AWS 데이터 소스를 원활하게 검색하고 빠르고 응답성이 뛰어난 쿼리 성능을 제공합니다.
- AWS Lake Formation - Lake Formation은 데이터 레이크를 설정, 보안 및 관리하기 위한 프로세스를 간소화하는 관리형 서비스입니다. Lake Formation을 사용하면 데이터 소스를 검색한 다음 데이터를 분류, 정리 및 변환할 수 있습니다. Lake Formation을 사용하면 Amazon S3의 데이터 레이크 데이터와 AWS Glue Data Catalog의 해당 메타데이터에 대한 세분화된 액세스 제어를 관리할 수 있습니다.

자습서: S3 Tables 시작하기

이 자습서에서는 테이블 버킷을 생성하고 리전의 테이블 버킷을 AWS 분석 서비스와 통합합니다. 다음으로 AWS CLI를 사용하여 테이블 버킷에 첫 번째 네임스페이스와 테이블을 생성합니다. 그런 다음

AWS Lake Formation을 사용하여 테이블에 대한 권한을 부여하므로 Athena로 테이블 쿼리를 시작할 수 있습니다.

> **Tip**
>
> 범용 버킷에서 테이블 버킷으로 테이블 형식 데이터를 마이그레이션하는 경우 AWS Solutions Library에 도움이 되는 안내 솔루션이 있습니다. 이 솔루션은 AWS Glue Data Catalog에 등록되고 범용 버킷에 저장된 Apache Iceberg 및 Apache Hive 테이블을 Apache Spark와 함께 AWS Step Functions 및 Amazon EMR을 사용하여 이동하는 작업을 자동화합니다. 자세한 내용은 AWS Solutions Library의 테이블 형식 데이터를 Amazon S3에서 S3 Tables로 마이그레이션하기 위한 지침을 참조하세요.

주제

- 1단계: 테이블 버킷 생성 및 AWS 분석 서비스와 통합
- 2단계: 테이블 네임스페이스 및 테이블 생성
- (옵션) 3단계: 테이블에 대한 Lake Formation 권한 부여
- 4단계: Athena에서 SQL을 사용하여 데이터 쿼리

1단계: 테이블 버킷 생성 및 AWS 분석 서비스와 통합

이 단계에서는 Amazon S3 콘솔을 사용하여 첫 번째 테이블 버킷을 생성합니다. 테이블 버킷을 생성하는 다른 방법은 Creating a table bucket을 참조하세요.

> **Note**
>
> 기본적으로 Amazon S3 콘솔은 테이블 버킷을 Amazon SageMaker Lakehouse와 자동으로 통합하므로 AWS 분석 서비스가 S3 Tables 데이터를 자동으로 검색하고 액세스할 수 있습니다. AWS Command Line Interface(AWS CLI), AWS SDK 또는 REST API를 사용하여 프로그래밍 방식으로 첫 번째 테이블 버킷을 생성하는 경우 AWS 분석 서비스 통합을 수동으로 완료해야 합니다. 자세한 내용은 AWS 분석 서비스에서 Amazon S3 Tables 사용 섹션을 참조하세요.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.

2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 테이블 버킷을 생성하려는 리전을 선택합니다.
3. 왼쪽 탐색 창에서 테이블 버킷을 선택합니다.
4. 테이블 버킷 생성을 선택합니다.
5. 일반 구성에서 테이블 버킷의 이름을 입력합니다.

   테이블 버킷 이름은 다음과 같아야 합니다.

   - 현재 리전의 AWS 계정 내에서 고유해야 합니다.
   - 3~63자 이내여야 합니다.
   - 소문자, 숫자 및 하이픈(-)만 포함해야 합니다.
   - 문자나 숫자로 시작하고 끝나야 합니다.

   테이블 버킷을 생성한 후에는 해당 이름을 변경할 수 없습니다. 테이블 버킷을 생성하는 AWS 계정이 해당 테이블 버킷을 소유합니다. 테이블 버킷 이름 지정에 대한 자세한 내용은 테이블 버킷 이름 지정 규칙 섹션을 참조하세요.
6. AWS 분석 서비스와의 통합 섹션에서 통합 활성화 확인란이 선택되어 있는지 확인합니다.

   콘솔을 사용하여 첫 번째 테이블 버킷을 생성할 때 통합 활성화를 선택한 경우 Amazon S3는 테이블 버킷을 AWS 분석 서비스와 통합하려고 시도합니다. 이 통합을 통해 AWS 분석 서비스를 사용하여 현재 리전의 모든 테이블에 액세스할 수 있습니다. 자세한 내용은 AWS 분석 서비스에서 Amazon S3 Tables 사용 섹션을 참조하세요.
7. 버킷 생성을 선택합니다.

2단계: 테이블 네임스페이스 및 테이블 생성

이 단계에서는 테이블 버킷에 네임스페이스를 생성한 다음 해당 네임스페이스 아래에 스키마가 있는 새 테이블을 생성합니다. 콘솔 또는 AWS CLI를 사용하여 테이블 네임스페이스와 테이블을 생성할 수 있습니다.

> **Important**
>
> 테이블을 만들 때 테이블 이름 및 테이블 정의에 모두 소문자를 사용해야 합니다. 예를 들어 열 이름이 모두 소문자인지 확인합니다. 테이블 이름 또는 테이블 정의에 대문자가 포함된 경우 테이블은 AWS Lake Formation 또는 AWS Glue Data Catalog에서 지원되지 않습니다. 이 경우

테이블 버킷이 AWS 분석 서비스와 통합되어 있더라도 Amazon Athena와 같은 AWS 분석 서비스에는 테이블이 표시되지 않습니다.
테이블 정의에 대문자가 포함된 경우 Athena에서 SELECT 쿼리를 실행할 때 "GENERIC_INTERNAL_ERROR: Get table request failed: com.amazonaws.services.glue.model.ValidationException: Unsupported Federation Resource - Invalid table or column names."라는 오류 메시지가 표시됩니다.

S3 콘솔 및 Amazon Athena 사용

다음 절차에서는 Amazon S3 콘솔을 사용하여 Amazon Athena를 사용하여 네임스페이스와 테이블을 생성합니다.

테이블 네임스페이스와 테이블을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 테이블 버킷을 선택합니다.
3. 테이블 버킷 페이지에서 테이블을 생성할 테이블 버킷을 선택합니다.
4. 테이블 버킷 세부 정보 페이지에서 Athena로 테이블 생성을 선택합니다.
5. Athena로 테이블 생성 대화 상자에서 네임스페이스 생성을 선택한 다음 네임스페이스 이름 필드에 이름을 입력합니다. 네임스페이스 이름은 1~255자여야 하며 테이블 버킷 내에서 고유해야 합니다. 유효한 문자는 a-z, 0-9 및 밑줄(_)입니다. 네임스페이스 이름 시작에는 밑줄이 허용되지 않습니다.
6. Create namespace(네임스페이스 생성)를 선택합니다.
7. Athena로 테이블 생성을 선택합니다.
8. Amazon Athena 콘솔이 열리고 Athena 쿼리 편집기가 나타납니다. 쿼리 편집기는 테이블을 생성하는 데 사용할 수 있는 샘플 쿼리로 채워집니다. 쿼리를 수정하여 테이블에 포함할 테이블 이름과 열을 지정합니다.
9. 쿼리 수정을 마치면 실행을 선택하여 테이블을 생성합니다.

테이블 생성에 성공하면 Athena의 테이블 목록에 새 테이블 이름이 나타납니다. Amazon S3 콘솔로 돌아가면 목록을 새로 고친 후 테이블 버킷의 세부 정보 페이지에 있는 테이블 목록에 새 테이블이 나타납니다.

AWS CLI 사용

다음 AWS CLI 예제 명령을 사용하여 테이블 버킷에 네임스페이스를 생성한 다음 해당 네임스페이스 아래에 스키마가 있는 새 테이블을 생성하려면 *user input placeholder* 값을 사용자 고유의 값으로 바꿉니다.

사전 조건

- IAM ID에 AmazonS3TablesFullAccess 정책을 연결합니다.
- AWS CLI 버전 2.23.10 이상을 설치합니다. 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS CLI 최신 버전의 설치 또는 업데이트를 참조하세요.

1. 다음 명령을 실행하여 테이블 버킷에 새 네임스페이스를 생성합니다.

```
aws s3tables create-namespace \
--table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket \
--namespace my_namespace
```

   - 다음 명령을 실행하여 네임스페이스가 성공적으로 생성되었는지 확인합니다.

```
aws s3tables list-namespaces \
--table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket
```

2. 다음 명령을 실행하여 테이블 스키마를 사용하여 새 테이블을 생성합니다.

```
aws s3tables create-table --cli-input-json file://mytabledefinition.json
```

   mytabledefinition.json 파일의 경우 다음 예제 테이블 정의를 사용합니다.

```
{
    "tableBucketARN": "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket",
    "namespace": "my_namespace",
    "name": "my_table",
    "format": "ICEBERG",
    "metadata": {
        "iceberg": {
            "schema": {
```

```
                "fields": [
                    {"name": "id", "type": "int","required": true},
                    {"name": "name", "type": "string"},
                    {"name": "value", "type": "int"}
                ]
            }
        }
    }
}
```

(옵션) 3단계: 테이블에 대한 Lake Formation 권한 부여

이 단계에서는 다른 IAM 위탁자에게 새 테이블에 대한 Lake Formation 권한을 부여합니다. 이러한 권한을 통해 다른 위탁자는 Athena 및 기타 AWS 분석 서비스를 사용하여 테이블 버킷 리소스에 액세스할 수 있습니다. 자세한 내용은 테이블 또는 데이터베이스에 대한 권한 부여 섹션을 참조하세요. 테이블에 액세스할 유일한 사용자인 경우 이 단계를 건너뛸 수 있습니다.

1. https://console.aws.amazon.com/lakeformation/에서 AWS Lake Formation 콘솔을 열고 데이터 레이크 관리자로 로그인합니다. 데이터 레이크 관리자 생성 방법에 대한 자세한 내용은 데이터 레이크 관리자 생성을 참조하세요.
2. 탐색 창에서 데이터 권한을 선택한 다음 권한 부여를 선택합니다.
3. 권한 부여 페이지의 위탁자에서 IAM 사용자 및 역할을 선택하고 테이블에 대한 쿼리를 실행할 IAM 사용자 또는 역할을 선택합니다.
4. LF 태그 또는 카탈로그 리소스에서 명명된 데이터 카탈로그 리소스를 선택합니다.
5. 계정의 모든 테이블에 대한 액세스 권한을 부여할지 또는 생성한 테이블 버킷 내의 리소스에만 액세스 권한을 부여할지에 따라 다음 중 하나를 수행합니다.
   - 카탈로그에서 테이블 버킷을 통합할 때 생성한 계정 수준 카탈로그를 선택합니다. 예를 들어 *111122223333*:s3tablescatalog입니다.
   - 카탈로그에서 테이블 버킷의 하위 카탈로그를 선택합니다. 예를 들어 *111122223333*:s3tablescatalog/*amzn-s3-demo-table-bucket*입니다.
6. (옵션) 테이블 버킷에 대한 하위 카탈로그를 선택한 경우 다음 중 하나 또는 둘 다를 수행합니다.
   - 데이터베이스에서 생성한 테이블 버킷 네임스페이스를 선택합니다.
   - 테이블에서 테이블 버킷에 생성한 테이블을 선택하거나 모든 테이블을 선택합니다.
7. 카탈로그 또는 하위 카탈로그를 선택했는지 여부와 데이터베이스 또는 테이블을 선택했는지 여부에 따라 카탈로그, 데이터베이스 또는 테이블 수준에서 권한을 설정할 수 있습니다. Lake

Formation 권한에 대한 자세한 내용은 AWS Lake Formation 개발자 안내서의 Managing Lake Formation permissions를 참조하세요.

다음 중 하나를 수행합니다.

- 카탈로그 권한에서 슈퍼를 선택하여 다른 위탁자에게 카탈로그에 대한 모든 권한을 부여하거나 설명과 같은 세분화된 권한을 선택합니다.
- 데이터베이스 권한의 경우 슈퍼를 선택하여 다른 위탁자에게 데이터베이스에 대한 모든 권한을 부여할 수 없습니다. 대신 설명과 같이 더 세분화된 권한을 선택합니다.
- 테이블 권한에서 슈퍼를 선택하여 다른 위탁자에게 테이블에 대한 모든 권한을 부여하거나 선택 또는 설명과 같은 더 세분화된 권한을 선택합니다.

> **Note**
>
> 데이터 카탈로그 리소스에 대한 Lake Formation 권한을 외부 계정에 부여하거나 다른 계정의 IAM 보안 주체에 직접 부여하면 Lake Formation은 AWS Resource Access Manager(AWS RAM) 서비스를 사용하여 리소스를 공유합니다. 피부여자 계정이 부여자 계정과 동일한 조직에 속해 있는 경우, 피부여자는 공유 리소스를 즉시 사용할 수 있습니다. 피부여자 계정이 같은 조직에 속해 있지 않은 경우, AWS RAM은 피부여자 계정에 리소스 부여를 수락하거나 거부하라는 초대를 보냅니다. 그런 다음 공유 리소스를 사용할 수 있게 하려면 피부여자 계정의 데이터 레이크 관리자가 AWS RAM 콘솔 또는 AWS CLI를 사용하여 초대를 수락해야 합니다. 계정 간 데이터 공유에 대한 자세한 내용은 AWS Lake Formation개발자 안내서의 Lake Formation에서의 계정 간 데이터 공유를 참조하세요.

8. 권한 부여를 선택합니다.

4단계: Athena에서 SQL을 사용하여 데이터 쿼리

Athena에서 SQL을 사용하여 테이블을 쿼리할 수 있습니다. Athena는 S3 Tables에 대해 데이터 정의 언어(DDL), 데이터 조작 언어(DML) 및 데이터 쿼리 언어(DQL) 쿼리를 지원합니다.

Amazon S3 콘솔 또는 Amazon Athena 콘솔을 통해 Amazon Athena 쿼리에 액세스할 수 있습니다.

S3 콘솔 및 Amazon Athena 사용

다음 절차에서는 Amazon S3 콘솔을 사용하여 Athena 쿼리 편집기에 액세스하여 Amazon Athena로 테이블을 쿼리할 수 있습니다.

테이블을 쿼리하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 테이블 버킷을 선택합니다.
3. 테이블 버킷 페이지에서 쿼리할 테이블이 포함된 테이블 버킷을 선택합니다.
4. 테이블 버킷 세부 정보 페이지에서 쿼리하려는 테이블 이름 옆에 있는 옵션 버튼을 선택합니다.
5. Athena를 사용하여 테이블 쿼리를 선택합니다.
6. Amazon Athena 콘솔이 열리고 Athena 쿼리 편집기가 샘플 SELECT 쿼리가 로드된 상태로 나타납니다. 사용 사례에 맞게 이 쿼리를 수정합니다.
7. 그런 다음 실행(Run)을 선택하여 쿼리를 실행합니다.

Amazon Athena 콘솔 사용

테이블을 쿼리하려면

1. https://console.aws.amazon.com/athena/에서 Athena 콘솔을 엽니다.
2. 테이블을 쿼리합니다. 다음은 수정할 수 있는 샘플 쿼리입니다. *user input placeholders*를 사용자의 정보로 대체합니다.

```
SELECT * FROM "s3tablescatalog/amzn-s3-demo-table-bucket"."my_namespace"."my_table"
 LIMIT 10
```

3. 그런 다음 실행(Run)을 선택하여 쿼리를 실행합니다.

테이블 버킷

Amazon S3 테이블 버킷은 테이블을 생성하고 S3 리소스로 저장하는 데 사용할 수 있는 S3 버킷 유형입니다. 테이블 버킷은 테이블 형식 데이터 및 메타데이터를 분석 워크로드에 사용할 객체로 저장하는 데 사용됩니다. S3는 테이블 버킷에서 유지 관리를 자동으로 수행하여 테이블 스토리지 비용을 절감에 도움을 줍니다. 자세한 내용은 테이블 버킷 유지 관리 섹션을 참조하세요.

테이블 버킷 내에 저장된 테이블과 상호 작용하기 위해 Apache Iceberg를 지원하는 분석 애플리케이션과 테이블 버킷을 통합할 수 있습니다. 테이블 버킷은 AWS Glue Data Catalog를 통해 AWS 분석 서비스와 통합됩니다. 자세한 내용은 AWS 분석 서비스에서 Amazon S3 Tables 사용 섹션을 참조하세요. Amazon S3 Tables Catalog for Apache Iceberg를 사용해 오픈 소스 쿼리 엔진을 사용하여 테이블

과 상호 작용할 수도 있습니다. 자세한 내용은 Amazon S3 Tables Iceberg REST 엔드포인트를 사용하여 테이블에 액세스 섹션을 참조하세요.

각 테이블 버킷에는 고유한 Amazon 리소스 이름(ARN)과 리소스 정책이 연결되어 있습니다. 테이블 버킷 ARN은 다음 형식을 따릅니다.

```
arn:aws:s3tables:Region:OwnerAccountID:bucket/bucket-name
```

모든 테이블 버킷과 테이블은 비공개이며 공개될 수 없습니다. 이러한 리소스는 명시적으로 액세스 권한이 부여된 사용자만 액세스할 수 있습니다. 액세스 권한을 부여하려면 테이블 버킷 및 테이블에 대한 IAM 리소스 기반 정책과 사용자 및 역할에 대한 IAM ID 기반 정책을 사용할 수 있습니다.

기본적으로 AWS 계정마다 AWS 리전당 최대 10개의 테이블 버킷을 생성할 수 있습니다. 테이블 버킷 또는 테이블에 대한 할당량 증가를 요청하려면 지원에 문의하세요.

Amazon S3 버킷에는 여러 유형이 있습니다. 버킷을 만들기 전에 애플리케이션 및 성능 요구 사항에 가장 적합한 버킷 유형을 선택하세요. 다양한 버킷 유형과 각 버킷의 적절한 사용 사례에 대한 자세한 내용은 버킷 섹션을 참조하세요.

주제

- Amazon S3 테이블 버킷, 테이블 및 네임스페이스 이름 지정 규칙
- 테이블 버킷 생성
- 테이블 버킷 삭제
- Amazon S3 테이블 버킷에 대한 세부 정보 보기
- 테이블 버킷 정책 관리

Amazon S3 테이블 버킷, 테이블 및 네임스페이스 이름 지정 규칙

테이블 버킷을 생성할 때 버킷 이름과 AWS 리전을 선택합니다. 이름은 선택한 리전의 계정에서 고유해야 합니다. 테이블 버킷을 생성한 후에는 버킷 이름 또는 리전을 변경할 수 없습니다. 테이블 버킷 이름은 특정 이름 지정 규칙을 따라야 합니다. 테이블 버킷과 그 안에 있는 테이블 및 네임스페이스의 이름 지정 규칙에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- 테이블 버킷 이름 지정 규칙
- 테이블 및 네임스페이스의 이름 지정 규칙

테이블 버킷 이름 지정 규칙

Amazon S3 테이블 버킷을 생성할 때 테이블 버킷 이름을 지정합니다. 다른 버킷 유형과 마찬가지로 테이블 버킷의 이름을 바꿀 수 없습니다. 다른 버킷 유형과 달리 테이블 버킷은 글로벌 네임스페이스에 존재하지 않으므로 계정의 각 버킷 이름은 현재 AWS 리전에서만 고유하면 됩니다.

범용 버킷의 이름 지정 규칙은 범용 버킷 이름 지정 규칙 섹션을 참조하세요. 디렉터리 버킷 이름 지정 규칙은 디렉터리 버킷 이름 지정 규칙 섹션을 참조하세요.

테이블 버킷에는 다음 이름 지정 규칙이 적용됩니다.

- 버킷 이름은 3자에서 63자 사이여야 합니다.
- 버킷 이름은 소문자, 숫자 및 하이픈(-)으로만 구성될 수 있습니다.
- 버킷 이름은 문자 또는 숫자로 시작하고 끝나야 합니다.
- 버킷 이름에는 밑줄(_) 또는 마침표(.)가 포함되어서는 안 됩니다.
- 버킷 이름은 다음과 같은 예약된 접두사로 시작해서는 안됩니다.
  - `xn--`
  - `sthree-`
  - `amzn-s3-demo-`
- 버킷 이름은 다음과 같은 예약된 접미사로 끝나서는 안됩니다.
  - `-s3alias`
  - `--ol-s3`
  - `--x-s3`
  - `--table-s3`

테이블 및 네임스페이스의 이름 지정 규칙

테이블 버킷 내의 테이블 및 네임스페이스에는 다음 이름 지정 규칙이 적용됩니다.

- 이름은 1~255자 이내여야 합니다.
- 이름은 소문자, 숫자 및 밑줄(_)로만 구성될 수 있습니다. 네임스페이스 이름의 시작과 끝에는 밑줄이 허용되지 않습니다.
- 이름은 글자 또는 숫자로 시작하고 끝나야 합니다.
- 이름에 하이픈(-) 또는 마침표(.)가 포함되어서는 안 됩니다.

- 테이블 이름은 네임스페이스 내에서 고유해야 합니다.
- 네임스페이스는 테이블 버킷 내에서 고유해야 합니다.
- 네임스페이스 이름은 예약된 접두사 aws로 시작되어서는 안됩니다. 예를 들어 aws_s3_metadata를 네임스페이스로 사용할 수 없습니다. aws_s3_metadata는 메타데이터 테이블용으로 예약된 네임스페이스입니다. 자세한 내용은 S3 메타데이터를 사용하여 데이터 검색 가속화 섹션을 참조하세요.

테이블 버킷 생성

Amazon S3 테이블 버킷은 테이블을 생성하고 S3 리소스로 저장하는 데 사용할 수 있는 S3 버킷 유형입니다. S3 Tables 사용을 시작하려면 테이블을 저장하고 관리할 테이블 버킷을 생성합니다. 버킷을 생성할 때 버킷 이름과 AWS 리전을 선택합니다. 테이블 버킷 이름은 선택한 리전의 계정에서 고유해야 합니다. 테이블 버킷을 생성한 후에는 버킷 이름 또는 리전을 변경할 수 없습니다. 테이블 버킷 이름 지정에 대한 자세한 내용은 Amazon S3 테이블 버킷, 테이블 및 네임스페이스 이름 지정 규칙 섹션을 참조하세요.

테이블 버킷의 Amazon 리소스 이름(ARN) 형식은 다음과 같습니다.

```
arn:aws:s3tables:region:owner-account-id:bucket/bucket-name
```

기본적으로 AWS 계정의 리전마다 최대 10개의 테이블 버킷을 생성할 수 있습니다. 테이블 버킷 또는 테이블에 대한 할당량 증가를 요청하려면 지원에 문의하세요.

테이블 버킷을 생성할 때 해당 버킷에서 생성한 테이블을 암호화하는 데 사용할 암호화 유형을 지정할 수 있습니다. 버킷 암호화 옵션에 대한 자세한 내용은 the section called “암호화” 섹션을 참조하세요.

테이블 버킷 생성을 위한 사전 조건

테이블 버킷을 생성하려면 먼저 다음을 수행해야 합니다.

- s3tables:CreateTableBucket에 대한 AWS Identity and Access Management(IAM) 권한이 있는지 확인합니다.

> **Note**
>
> SSE-KMS를 기본 암호화 유형으로 선택하는 경우 s3tables:PutTableBucketEncryption 권한과 선택한 AWS KMS 키에 대한 DescribeKey 권한이 있어야 합니다. 또한 사용하는 AWS KMS 키가 S3 Tables에 자동 테이

블 유지 관리를 수행할 수 있는 권한을 부여해야 합니다. 자세한 내용은 S3 Tables SSE-KMS 암호화에 대한 권한 요구 사항 섹션을 참조하세요.

테이블 버킷을 생성하려면 Amazon S3 콘솔, Amazon S3 REST API, AWS Command Line Interface(AWS CLI) 또는 AWS SDK를 사용할 수 있습니다.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 버킷을 생성하려는 리전을 선택합니다.
3. 왼쪽 탐색 창에서 테이블 버킷을 선택합니다.
4. 테이블 버킷 생성을 선택하여 테이블 버킷 생성 페이지를 엽니다.
5. 속성에서 테이블 버킷의 이름을 입력합니다.

   테이블 버킷 이름은 다음과 같아야 합니다.

   - 현재 리전의 계정 내에서 고유해야 합니다.
   - 3~63자 이내여야 합니다.
   - 소문자, 숫자 및 하이픈(-)만 포함해야 합니다.
   - 문자나 숫자로 시작하고 끝나야 합니다.

   버킷을 생성한 후에는 해당 이름을 변경할 수 없습니다. 버킷을 생성하는 AWS 계정이 해당 버킷을 소유합니다. 테이블 버킷 이름 지정에 대한 자세한 내용은 Amazon S3 테이블 버킷, 테이블 및 네임스페이스 이름 지정 규칙 섹션을 참조하세요.
6. 테이블 버킷을 AWS 분석 서비스와 통합하려면 AWS 분석 서비스와 통합에서 통합 활성화를 선택해야 합니다.

> **Note**
>
> 통합 활성화 옵션이 선택된 콘솔을 사용하여 테이블 버킷을 처음 생성하면 Amazon S3는 테이블 버킷을 AWS 분석 서비스와 자동으로 통합하려고 시도합니다. 이 통합을 통해 AWS 분석 서비스를 사용하여 현재 리전의 모든 테이블을 쿼리할 수 있습니다. 자세한 내용은 AWS 분석 서비스에서 Amazon S3 Tables 사용 섹션을 참조하세요.

7. 기본 암호화를 구성하려면 암호화 유형 아래에서 다음 중 하나를 선택합니다.

   - Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)
   - AWS Key Management Service 키를 사용한 서버 측 암호화(SSE-KMS)

   테이블 데이터의 암호화 옵션에 대한 자세한 내용은 암호화로 S3 테이블 데이터 보호 섹션을 참조하세요.
8. 버킷 생성을 선택합니다.

AWS CLI 사용

이 예시는 AWS CLI를 사용하여 테이블 버킷을 생성하는 방법을 보여줍니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws s3tables create-table-bucket \
    --region us-east-2 \
    --name amzn-s3-demo-bucket1
```

기본적으로 S3 테이블 버킷은 SSE-S3를 기본 암호화 설정으로 사용하지만 선택적 `--encryption-configuration` 파라미터를 사용하여 다른 암호화 유형을 지정할 수 있습니다. 다음 예제에서는 SSE-KMS 암호화를 사용하는 버킷을 생성하는 방법을 보여줍니다. 테이블 버킷의 암호화 설정에 대한 자세한 내용은 암호화로 S3 테이블 데이터 보호 섹션을 참조하세요.

```
aws s3tables create-table-bucket \
    --region us-east-2 \
    --name amzn-s3-demo-bucket1
    --encryption-configuration '{
                    "sseAlgorithm": "aws:kms",
                    "kmsKeyArn":
 "arn:aws:kms:Region:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab" }'
```

테이블 버킷 삭제

Amazon S3 API, AWS Command Line Interface 또는 AWS SDK를 사용하여 테이블 버킷을 삭제할 수 있습니다. 테이블 버킷을 삭제하기 전에 버킷 내의 모든 네임스페이스와 테이블을 삭제해야 합니다.

AWS CLI 사용

이 예시는 AWS CLI를 사용하여 테이블 버킷을 삭제하는 방법을 보여줍니다. 이 예시를 사용하려면 *### ## ## ###*를 실제 정보로 대체하세요.

```
aws s3tables delete-table-bucket \
    --region us-east-2 \
    --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-bucket1
```

Amazon S3 테이블 버킷에 대한 세부 정보 보기

S3 Tables REST API, AWS CLI 또는 AWS SDK를 사용하여 Amazon S3 테이블 버킷의 세부 정보를 프로그래밍 방식으로 볼 수 있습니다.

AWS CLI 사용

이 예시는 AWS CLI를 사용하여 테이블 버킷에 대한 세부 정보를 가져오는 방법을 보여줍니다. 이 예시를 사용하려면 *### ## ## ###*를 실제 정보로 대체하세요.

```
aws s3tables get-table-bucket --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-bucket1
```

테이블 버킷 정책 관리

Amazon S3 REST API, AWS SDK, AWS Command Line Interface(AWS CLI)를 사용하여 Amazon S3 테이블 버킷에 대한 버킷 정책을 추가, 삭제, 업데이트 및 확인할 수 있습니다. 자세한 내용은 다음 항목을 참조하세요.

자세한 내용은 다음 항목을 참조하세요. Amazon S3 Tables에 지원되는 AWS Identity and Access Management(IAM) 작업 및 조건 키에 대한 자세한 내용은 S3 Tables의 액세스 관리 섹션을 참조하세요. 테이블 버킷에 대한 버킷 정책의 예는 S3 Tables의 리소스 기반 정책 섹션을 참조하세요.

테이블 버킷 정책 추가

테이블 버킷에 버킷 정책을 추가하려면 다음 AWS CLI 예시를 사용합니다.

AWS CLI 사용

이 예시는 AWS CLI를 사용하여 테이블 버킷 정책을 생성하는 방법을 보여줍니다. 이 명령을 사용하려면 *user input placeholders*를 실제 정보로 바꾸세요.

```
aws s3tables put-table-bucket-policy \
    --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
bucket1  \
    --resource-policy your-policy-JSON
```

테이블 버킷 정책 보기

테이블 버킷에 연결된 버킷 정책을 보려면 다음 AWS CLI 예시를 사용합니다.

AWS CLI 사용

이 예시는 AWS CLI를 사용하여 테이블 버킷에 연결된 정책을 확인하는 방법을 보여줍니다. 이 명령을 사용하려면 *user input placeholders*를 실제 정보로 바꾸세요.

```
aws s3tables get-table-bucket-policy --table-bucket-arn arn:aws:s3tables:us-
east-1:111122223333:bucket/amzn-s3-demo-bucket1
```

테이블 버킷 정책 삭제

테이블 버킷에 연결된 버킷 정책을 삭제하려면 다음 AWS CLI 예시를 사용합니다.

AWS CLI 사용

이 예시는 AWS CLI를 사용하여 테이블 버킷 정책을 삭제하는 방법을 보여줍니다. 이 명령을 사용하려면 *user input placeholders*를 실제 정보로 바꾸세요.

```
aws s3tables delete-table-bucket-policy --table-bucket-arn arn:aws:s3tables:us-
east-1:111122223333:bucket/amzn-s3-demo-bucket1
```

S3 Tables 유지 관리

Amazon S3는 S3 테이블 버킷의 테이블 성능을 개선하기 위해 유지 관리를 자동으로 수행합니다. 유지 관리는 테이블 버킷 및 개별 테이블 수준에서 수행되며 다음을 포함합니다.

테이블 버킷 수준 유지 관리:

- 참조되지 않은 파일 제거 - 분리된 파일을 정리하여 스토리지 사용량 및 비용을 최적화합니다.

테이블 수준 유지 관리:

- 파일 압축 - 작은 파일을 통합하여 쿼리 성능을 개선하고 스토리지 비용을 절감합니다.
- 스냅샷 관리 - 테이블 버전 기록을 제어하고 과도한 메타데이터 증가를 방지합니다.

이 옵션은 기본적으로 활성화되어 있습니다. 유지 관리 구성 파일을 통해 이러한 작업을 편집하거나 비활성화할 수 있습니다.

주제

- S3 Tables 유지 관리 작업 상태
- 테이블 버킷 유지 관리
- 테이블 유지 관리
- 유지 관리 작업에 대한 고려 사항 및 제한 사항

S3 Tables 유지 관리 작업 상태

S3 Tables 유지 관리 작업은 S3 테이블 또는 테이블 버킷에 대해 주기적으로 실행됩니다. `GetTableMaintenanceJobStatus` API를 사용하여 이러한 작업의 상태를 쿼리할 수 있습니다.

AWS CLI를 사용하여 유지 관리 작업의 상태를 확인하는 방법

다음 예시에서는 `GetTableMaintenanceJobStatus` API를 사용하여 유지 관리 작업의 상태를 가져옵니다.

```
aws s3tables get-table-maintenance-job-status \
   --table-bucket-arn="arn:aws:s3tables:arn:aws::111122223333:bucket/amzn-s3-demo-bucket1" \
   --namespace="mynamespace" \
   --name="testtable"
```

자세한 정보는 AWS CLI 명령 참조의 get-table-maintenance-job-status 섹션을 참조하세요.

S3 Tables 유지 관리 작업은 다음 네 가지 상태 간에 전환할 수 있습니다.

- `Successful`
- `Failed`
- `Disabled`
- `Not_Yet_Run`

실패 상태의 작업에는 실패 메시지가 포함됩니다. 다음 목록에는 나타날 수 있는 실패 메시지가 설명되어 있습니다.

- 테이블을 읽으려고 할 때 Iceberg 검증 예외가 발생했습니다. 테이블이 읽을 수 있고 Iceberg 사양을 준수하며 S3 테이블 별칭으로 시작하는 S3 경로만 포함하는지 확인합니다.
- Iceberg 스냅샷 관리는 현재 사용자 정의 태그 또는 참조를 지원하지 않습니다.
- Iceberg 테이블 유지 관리 구성은 'history.expire.max-snapshot-age-ms' 및 'history.expire.min-snapshots-to-keep' 테이블 속성과 호환되지 않습니다.
- 'gc.enabled' 테이블 속성이 false인 경우 Iceberg 스냅샷 관리 및 참조되지 않는 파일 제거가 지원되지 않습니다. 이 속성이 설정되지 않았거나 명시적으로 true로 설정되어 있는지 확인합니다.
- 오래된 메타데이터로 인해 커밋하지 못했습니다. 유지 관리는 사용 가능한 다음 기회에 다시 시도됩니다.
- 테이블 유지 관리를 수행할 수 있는 권한이 부족합니다. 테이블을 암호화하는 데 사용되는 키가 활성 상태이고 존재하며 S3 서비스 위탁자 `maintenance.s3tables.amazonaws.com`에게 액세스 권한을 부여하는 리소스 정책이 있는지 확인합니다.

> **Note**
>
> S3 Tables의 AWS KMS 권한에 대한 자세한 내용은 S3 Tables SSE-KMS 암호화에 대한 권한 요구 사항 섹션을 참조하세요.

- 내부 오류

테이블 버킷 유지 관리

Amazon S3는 테이블 버킷의 관리 및 성능을 개선하기 위한 유지 관리 작업을 제공합니다. 다음 옵션은 기본적으로 모든 테이블 버킷에 대해 활성화됩니다. 테이블 버킷에 대한 유지 관리 구성 파일을 지정하여 이 옵션을 편집하거나 비활성화할 수 있습니다.

이 구성을 편집하려면 `s3tables:PutTableBucketMaintenanceConfiguration` 권한이 필요합니다.

주제

- 참조되지 않은 파일 제거
- 고려 사항 및 제한

참조되지 않은 파일 제거

참조되지 않은 파일 제거는 어떤 테이블 스냅샷에서도 참조되지 않은 모든 객체를 식별하고 삭제합니다. 참조되지 않은 파일 제거 정책의 일부로 unreferencedDays(기본값 3일) 및 nonCurrentDays(기본값 10일)의 두 가지 속성을 구성할 수 있습니다.

테이블에서 참조하지 않고 unreferencedDays 속성보다 오래된 객체의 경우 S3는 객체를 비최신으로 표시합니다. S3는 nonCurrentDays 속성에 지정된 일수 이후에 비최신 객체를 삭제합니다.

> **Note**
>
> 비최신 객체의 삭제는 영구적이며 이러한 객체를 복구할 방법이 없습니다.

비최신으로 표시된 객체를 보거나 복구하려면 AWS Support에 문의해야 합니다. AWS Support에 문의하는 방법에 대한 자세한 내용은 AWS 문의 또는 AWS Support 설명서를 참조하세요.

참조되지 않은 파일 제거는 해당 테이블만 참조하여 테이블에서 삭제할 객체를 결정합니다. 테이블 외부에서 이러한 객체를 참조해도 참조되지 않은 파일 제거가 객체를 삭제하는 것을 막을 수 없습니다.

참조되지 않은 파일 제거를 비활성화해도 진행 중인 작업은 영향을 받지 않습니다. 새 구성은 구성 변경 후 다음 작업에 적용됩니다. 자세한 내용은 Amazon S3 요금의 요금 정보를 참조하세요.

테이블 버킷 수준에서만 참조되지 않은 파일 제거를 구성할 수 있습니다. 이 구성은 버킷의 모든 테이블에 적용됩니다.

AWS CLI를 사용하여 참조되지 않은 파일 제거를 구성하는 방법

다음 예시에서는 PutTableBucketMaintenanceConfiguration API를 사용하여 unreferencedDays를 4일로 설정하고, nonCurrentDays를 10일로 설정합니다.

```
aws s3tables put-table-bucket-maintenance-configuration \
   --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
table-bucket \
   --type icebergUnreferencedFileRemoval \
   --value '{"status":"enabled","settings":{"icebergUnreferencedFileRemoval":
{"unreferencedDays":4,"nonCurrentDays":10}}}'
```

자세한 정보는 AWS CLI 명령 참조의 put-table-bucket-maintenance-configuration 섹션을 참조하세요.

고려 사항 및 제한

참조되지 않은 파일 제거에 대한 추가 고려 사항 및 제한 사항에 대한 자세한 내용은 the section called "고려 사항 및 제한 사항" 섹션을 참조하세요.

테이블 유지 관리

S3 Tables은 유지 관리 작업을 제공하여 개별 테이블의 관리 및 성능을 향상합니다. 다음 옵션은 기본적으로 테이블 버킷의 모든 테이블에 대해 활성화됩니다. S3 테이블에 대한 유지 관리 구성 파일을 지정하여 이러한 옵션을 편집하거나 비활성화할 수 있습니다.

이 구성을 편집하려면 `s3tables:GetTableMaintenanceConfiguration` 및 `s3tables:PutTableMaintenanceConfiguration` 권한이 필요합니다.

주제

- 압축
- 스냅샷 관리
- 고려 사항 및 제한

압축

압축은 테이블 수준에서 구성되며 여러 개의 작은 객체를 더 적고 큰 객체로 결합하여 Apache Iceberg 쿼리 성능을 개선합니다. 객체를 결합할 때 압축은 테이블에서 행 수준 삭제의 영향도 적용합니다.

모든 테이블에 대해 기본적으로 압축이 활성화되며, 기본 대상 파일 크기는 512MB이고, 64MB에서 512MB까지 직접 값을 지정할 수 있습니다. 압축된 파일은 테이블의 최신 스냅샷으로 기록됩니다.

압축 전략

여러 압축 전략 중에서 선택할 수 있으며, 압축 전략을 선택하면 쿼리 패턴 및 테이블 정렬 순서에 따라 쿼리 성능을 더욱 높일 수 있습니다.

S3 Tables은 테이블에 대해 다음과 같은 압축 전략을 지원합니다.

- Auto(기본값)
  - Amazon S3는 테이블 정렬 순서에 따라 최적의 압축 전략을 선택합니다. 이는 모든 테이블에 대한 기본 압축 전략입니다.
  - 메타데이터에 정렬 순서가 정의된 테이블의 경우 `auto`는 `sort` 압축을 자동으로 적용합니다.

  - 정렬 순서가 없는 테이블의 경우 auto는 기본적으로 binpack 압축을 사용합니다.
- Binpack
  - 작은 파일을 더 큰 파일로 결합합니다. 일반적으로 100MB 이상의 크기를 대상으로 하며 보류 중인 삭제를 적용합니다. 이는 정렬되지 않은 테이블의 기본 압축 전략입니다.
- 정렬
  - 압축 중에 계층 구조별로 자동으로 정렬되는 지정된 열을 기반으로 데이터를 정리하여 필터링된 작업에 대한 쿼리 성능을 개선합니다. 이 전략은 쿼리가 특정 열을 자주 필터링할 때 권장됩니다. 이 전략을 사용하면 sort_order가 테이블 속성에 정의되어 있을 때 S3 Tables이 열에 계층적 정렬을 자동으로 적용합니다.
- Z-order
  - 여러 속성을 정렬에 사용할 수 있는 단일 스칼라 값으로 블렌딩하여 데이터 정리를 최적화하므로 여러 차원에 걸쳐 효율적인 쿼리가 가능합니다. 이 전략은 여러 차원에서 동시에 데이터를 쿼리해야 하는 경우에 권장됩니다. 이 전략을 사용하려면 sort_order 테이블 속성을 사용하여 Iceberg 테이블 속성에서 정렬 순서를 정의해야 합니다.

압축에는 추가 비용이 발생합니다. 자세한 내용은 Amazon S3 요금의 요금 정보를 참조하세요.

압축 예시

다음 예시는 테이블 압축을 위한 구성을 보여줍니다.

AWS CLI를 사용하여 압축 대상 파일 크기를 구성하는 방법

최소 대상 압축 파일 크기는 64MB이고 최대 크기는 512MB입니다.

다음 예시에서는 PutTableMaintenanceConfiguration API를 사용하여 대상 파일 크기를 256MB로 변경합니다.

```
aws s3tables put-table-maintenance-configuration \
   --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
bucket1 \
   --type icebergCompaction \
   --namespace mynamespace \
   --name testtable \
   --value='{"status":"enabled","settings":{"icebergCompaction":
{"targetFileSizeMB":256}}}'
```

자세한 정보는 AWS CLI 명령 참조의 put-table-maintenance-configuration 섹션을 참조하세요.

AWS CLI를 사용하여 압축 전략을 구성하는 방법

다음 예시에서는 PutTableMaintenanceConfiguration API를 사용하여 압축 전략을 sort로 변경합니다. 압축을 설정할 때 auto, binpack, sort 또는 z-order 압축 전략 중에서 선택할 수 있습니다.

> **Note**
>
> 압축 전략을 sort 또는 z-order로 설정하려면 다음 사전 조건을 충족해야 합니다.
>
> - Iceberg 테이블 속성에 정렬 순서가 정의되어 있어야 합니다.
> - s3tables:GetTableData 권한

```
aws s3tables put-table-maintenance-configuration \
   --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
table-bucket \
   --type icebergCompaction \
   --namespace mynamespace \
   --name testtable \
   --value='{"status":"enabled","settings":{"icebergCompaction":
{"strategy":"sort"}}}'
```

자세한 정보는 AWS CLI 명령 참조의 put-table-maintenance-configuration 섹션을 참조하세요.

AWS CLI를 사용하여 압축을 비활성화하는 방법

다음 예시에서는 PutTableMaintenanceConfiguration API를 사용하여 압축을 비활성화합니다.

```
aws s3tables put-table-maintenance-configuration \
   --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
table-bucket \
   --type icebergCompaction \
   --namespace mynamespace \
   --name testtable \
   --value='{"status":"disabled","settings":{"icebergCompaction":
{"targetFileSizeMB":256}}}'
```

자세한 정보는 AWS CLI 명령 참조의 put-table-maintenance-configuration 섹션을 참조하세요.

스냅샷 관리

스냅샷 관리는 테이블의 활성 스냅샷 수를 결정합니다. 이는 `MinimumSnapshots`(기본값 1) 및 `MaximumSnapshotAge`(기본값 120시간)을 기반으로 합니다. 스냅샷 관리는 이러한 구성을 기반으로 테이블 스냅샷을 만료하고 제거합니다.

스냅샷이 만료되면 Amazon S3는 해당 스냅샷에서만 참조하는 객체를 비최신으로 표시합니다. 이러한 비최신 객체는 참조되지 않은 파일 제거 정책의 `NoncurrentDays` 속성에 지정된 일수 후에 삭제됩니다.

> **Note**
>
> 비최신 객체의 삭제는 영구적이며 이러한 객체를 복구할 방법이 없습니다.

비최신으로 표시된 객체를 보거나 복구하려면 AWS Support에 문의해야 합니다. AWS Support에 문의하는 방법에 대한 자세한 내용은 AWS 문의 또는 AWS Support 설명서를 참조하세요.

스냅샷 관리는 해당 테이블만 참조하여 테이블에서 삭제할 객체를 결정합니다. 테이블 외부에서 이러한 객체를 참조해도 스냅샷 관리가 객체를 삭제하는 것을 막을 수 없습니다.

> **Note**
>
> 스냅샷 관리는 `metadata.json` 파일에서 또는 브랜치 또는 태그 기반 보존을 포함한 `ALTER TABLE SET TBLPROPERTIES` SQL 명령을 통해 Iceberg 테이블 속성으로 구성한 보존 값을 지원하지 않습니다. 스냅샷 관리는 브랜치 또는 태그 기반 보존 정책을 구성하거나 `metadata.json` 파일에 `PutTableMaintenanceConfiguration` API를 통해 구성된 값보다 긴 보존 정책을 구성할 때 비활성화됩니다. 이러한 경우 S3는 스냅샷을 만료하거나 제거하지 않으므로 스토리지 요금을 피하려면 스냅샷을 수동으로 삭제하거나 Iceberg 테이블에서 속성을 제거해야 합니다.

스냅샷 관리는 테이블 수준에서만 구성할 수 있습니다. 자세한 내용은 Amazon S3 요금의 요금 정보를 참조하세요.

스냅샷 관리 예시

다음 예시는 테이블 스냅샷 관리를 위한 구성을 보여줍니다.

AWS CLI를 사용하여 스냅샷 관리를 구성하는 방법

다음 예시에서는 PutTableMaintenanceConfiguration API를 사용하여 MinimumSnapshots을 10개로 설정하고, MaximumSnapshotAge를 2,500시간으로 설정합니다.

```
aws s3tables put-table-maintenance-configuration \
--table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
table-bucket \
--namespace my_namespace \
--name my_table \
--type icebergSnapshotManagement \
--value '{"status":"enabled","settings":{"icebergSnapshotManagement":
{"minSnapshotsToKeep":10,"maxSnapshotAgeHours":2500}}}'
```

AWS CLI를 사용하여 스냅샷 관리를 비활성화하는 방법

다음 예시에서는 PutTableMaintenanceConfiguration API를 사용하여 스냅샷 관리를 비활성화합니다.

```
aws s3tables put-table-maintenance-configuration \
--table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
table-bucket \
--namespace my_namespace \
--name my_table \
--type icebergSnapshotManagement \
--value '{"status":"disabled","settings":{"icebergSnapshotManagement":
{"minSnapshotsToKeep":1,"maxSnapshotAgeHours":120}}}'
```

자세한 정보는 AWS CLI 명령 참조의 put-table-maintenance-configuration 섹션을 참조하세요.

고려 사항 및 제한

압축 및 스냅샷 관리에 대한 추가 고려 사항 및 한계에 대한 자세한 내용은 the section called “고려 사항 및 제한 사항” 섹션을 참조하세요.

> **Note**
>
> S3 Tables는 128MB의 parquets row-group-default 크기를 적용합니다.

유지 관리 작업에 대한 고려 사항 및 제한 사항

Amazon S3는 S3 테이블 또는 테이블 버킷의 성능을 개선하기 위한 유지 관리 작업을 제공합니다. 이러한 옵션은 파일 압축, 스냅샷 관리 및 참조되지 않은 파일 제거입니다. 다음은 이러한 관리 옵션에 대한 제한 사항 및 고려 사항입니다.

주제

- 압축 고려 사항
- 스냅샷 관리에 대한 고려 사항
- 참조되지 않은 파일 제거 시 고려 사항
- S3 테이블 및 테이블 버킷 유지 관리에 대한 제한 사항

압축 고려 사항

압축에는 다음 고려 사항이 적용됩니다. 압축에 대한 자세한 내용은 the section called “테이블 유지 관리” 섹션을 참조하세요.

- 압축은 Apache Parquet 파일 유형에서만 지원됩니다.
- 압축은 Fixed 데이터 유형을 지원하지 않습니다.
- 압축은 `brotli`, `lz4` 압축 유형을 지원하지 않습니다.

스냅샷 관리에 대한 고려 사항

다음은 스냅샷 관리에 적용되는 고려 사항입니다. 스냅샷 관리에 대한 자세한 내용은 the section called “테이블 유지 관리” 섹션을 참조하세요.

- 스냅샷은 두 기준(보존할 최소 스냅샷 수 및 지정된 보존 기간)이 모두 충족되는 경우에만 보존됩니다.
- 스냅샷 관리는 Apache Iceberg에서 만료된 스냅샷 메타데이터를 삭제하여 만료된 스냅샷에 대한 시간 이동 쿼리를 방지하고 선택적으로 연결된 데이터 파일을 삭제합니다.
- 스냅샷 관리는 `metadata.json` 파일에서 또는 브랜치 또는 태그 기반 보존을 포함한 `ALTER TABLE SET TBLPROPERTIES` SQL 명령을 통해 Iceberg 테이블 속성으로 구성한 보존 값을 지원하지 않습니다. 스냅샷 관리는 브랜치 또는 태그 기반 보존 정책을 구성하거나 `metadata.json` 파일에 `PutTableMaintenanceConfiguration` API를 통해 구성된 값보다 긴 보존 정책을 구성할 때 비활성화됩니다. 이러한 경우 S3는 스냅샷을 만료하거나 제거하지 않으므로 스토리지 요금을 피하려면 스냅샷을 수동으로 삭제하거나 Iceberg 테이블에서 속성을 제거해야 합니다.

참조되지 않은 파일 제거 시 고려 사항

참조되지 않은 파일 제거에는 다음 고려 사항이 적용됩니다. 참조되지 않은 파일 제거에 대한 자세한 내용은 the section called “테이블 버킷 유지 관리” 섹션을 참조하세요.

- 참조되지 않은 파일 제거는 생성 시간이 보존 기간 이전인 경우 Iceberg 메타데이터에서 더 이상 참조하지 않는 데이터 및 메타데이터 파일을 삭제합니다.

S3 테이블 및 테이블 버킷 유지 관리에 대한 제한 사항

유지 관리 작업	속성	테이블 버킷 수준에서 구성 가능 여부	테이블 수준에서 구성 가능 여부	기본값	최소값
압축	targetFileSizeMB	아니요	예	512MB	64MB
스냅샷 관리	minimumSnapshots	아니요	예	1	1
스냅샷 관리	maximumSnapshotAge	아니요	예	120시간	1시간
참조되지 않은 파일 제거	unreferencedDays	예	아니요	3일	1일
참조되지 않은 파일 제거	nonCurrentDays	예	아니요	10일	1일

> **Note**
>
> S3 Tables는 128MB의 parquets row-group-default 크기를 적용합니다.

테이블 네임스페이스

Amazon S3 Tables 버킷 내에 테이블을 생성할 때 테이블을 네임스페이스라는 논리적 그룹으로 구성합니다. S3 테이블 및 테이블 버킷과 달리 네임스페이스는 리소스가 아닙니다. 네임스페이스는 확장 가능한 방식으로 테이블을 구성하고 관리하는 데 도움이 되는 구문입니다. 예를 들어 회사의 HR 부서에 속한 모든 테이블을 공통 네임스페이스 값 `hr`로 그룹화할 수 있습니다.

테이블 버킷 리소스 정책을 사용하여 특정 네임스페이스로 액세스를 제어할 수 있습니다. 자세한 내용은 the section called “리소스 기반 정책” 섹션을 참조하세요.

다음 규칙은 테이블 네임스페이스에 적용됩니다.

- 모든 네임스페이스는 테이블 버킷 내에서 고유해야 합니다.
- 테이블 버킷당 최대 1만 개의 네임스페이스를 생성할 수 있습니다.
- 모든 테이블 이름은 네임스페이스 내에서 고유해야 합니다.
- 각 테이블에는 네임스페이스 수준이 하나만 있을 수 있습니다. 네임스페이스는 중첩할 수 없습니다.
- 각 테이블은 단일 네임스페이스에 속합니다.
- 네임스페이스 간에 테이블을 이동할 수 있습니다.

주제

- 네임스페이스 생성
- 네임스페이스 삭제

네임스페이스 생성

테이블 네임스페이스는 Amazon S3 Tables 버킷으로 테이블을 그룹화하는 논리적 구문입니다. 각 테이블은 단일 네임스페이스에 속합니다. 테이블 버킷에 테이블을 생성하기 전에 테이블을 그룹화할 네임스페이스를 생성해야 합니다. Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), Amazon S3 REST API, AWS SDK 또는 통합 쿼리 엔진을 사용하여 네임스페이스를 생성할 수 있습니다.

네임스페이스 이름

네임스페이스에는 다음 이름 지정 규칙이 적용됩니다.

- 이름은 1~255자 이내여야 합니다.

- 이름은 소문자, 숫자 및 밑줄(_)로만 구성될 수 있습니다. 네임스페이스 이름의 시작과 끝에는 밑줄이 허용되지 않습니다.
- 이름은 글자 또는 숫자로 시작하고 끝나야 합니다.
- 이름에 하이픈(-) 또는 마침표(.)가 포함되어서는 안 됩니다.
- 네임스페이스는 테이블 버킷 내에서 고유해야 합니다.
- 네임스페이스 이름은 예약된 접두사 aws로 시작되어서는 안됩니다. 예를 들어 aws_s3_metadata를 네임스페이스로 사용할 수 없습니다. aws_s3_metadata는 메타데이터 테이블용으로 예약된 네임스페이스입니다. 자세한 내용은 S3 메타데이터를 사용하여 데이터 검색 가속화 섹션을 참조하세요.

유효한 네임스페이스 이름에 대한 자세한 내용은 테이블 및 네임스페이스의 이름 지정 규칙 섹션을 참조하세요.

S3 콘솔 및 Amazon Athena 사용

다음 절차에서는 Athena로 테이블 생성 워크플로를 사용하여 Amazon S3 콘솔에서 네임스페이스를 생성합니다. Amazon Athena를 사용하여 네임스페이스에 테이블을 생성하지 않으려면 네임스페이스를 생성한 후 워크플로를 취소할 수 있습니다.

네임스페이스를 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 테이블 버킷을 선택합니다.
3. 테이블 버킷 페이지에서 네임스페이스를 생성할 버킷을 선택합니다.
4. 버킷 세부 정보 페이지에서 Athena로 테이블 생성을 선택합니다.
5. Athena로 테이블 생성 대화 상자에서 네임스페이스 생성을 선택한 다음 네임스페이스 생성을 선택합니다.
6. 네임스페이스 이름 필드에 이름을 입력합니다. 네임스페이스 이름은 1~255자여야 하며 테이블 버킷 내에서 고유해야 합니다. 유효한 문자는 a-z, 0-9 및 밑줄(_)입니다. 네임스페이스 이름의 시작과 끝에는 밑줄이 허용되지 않습니다.
7. Create namespace(네임스페이스 생성)를 선택합니다.
8. 테이블도 생성하려면 Athena로 테이블 생성을 선택합니다. Athena를 사용하여 테이블을 생성하는 방법에 대한 자세한 내용은 the section called “S3 콘솔 및 Amazon Athena 사용” 섹션을 참조하세요. 지금 테이블을 생성하지 않으려면 취소를 선택합니다.

AWS CLI 사용

이 예시에서는 AWS CLI를 사용하여 테이블 네임스페이스를 생성하는 방법을 보여줍니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws s3tables create-namespace \
    --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-bucket1 \
    --namespace example_namespace
```

쿼리 엔진 사용

Amazon S3 테이블 버킷에 연결된 Apache Spark 세션에서 네임스페이스를 생성할 수 있습니다.

이 예시에서는 S3 Tables과 통합된 쿼리 엔진에서 CREATE 문을 사용하여 테이블을 생성하는 방법을 보여줍니다. 이 예시를 사용하려면 *### ## ## ###*를 실제 정보로 대체하세요.

```
spark.sql("CREATE NAMESPACE IF NOT EXISTS s3tablesbucket.my_namespace")
```

네임스페이스 삭제

Amazon S3 Tables 버킷에서 테이블 네임스페이스를 삭제하기 전에 네임스페이스 내의 모든 테이블을 삭제하거나 다른 네임스페이스로 이동해야 합니다. Amazon S3 REST API, AWS SDK, AWS Command Line Interface(AWS CLI) 또는 통합 쿼리 엔진을 사용하여 네임스페이스를 삭제할 수 있습니다.

네임스페이스 삭제에 필요한 권한에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 DeleteNamespace 섹션을 참조하세요.

AWS CLI 사용

이 예시에서는 AWS CLI를 사용하여 테이블 네임스페이스를 삭제하는 방법을 보여줍니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws s3tables delete-namespace \
    --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-bucket1 \
    --namespace example_namespace
```

S3 테이블 버킷의 테이블

S3 테이블은 기본 테이블 데이터 및 관련 메타데이터로 구성된 구조화된 데이터세트를 나타냅니다. 이 데이터는 테이블 버킷 내에 하위 리소스로 저장됩니다. 테이블 버킷의 모든 테이블은 Apache Iceberg 테이블 형식으로 저장됩니다. Amazon S3는 자동 파일 압축 및 스냅샷 관리를 통해 테이블 유지 관리를 관리합니다. 자세한 내용은 테이블 유지 관리 섹션을 참조하세요.

AWS 분석 서비스에서 계정의 테이블에 액세스할 수 있도록 하려면 Amazon S3 Tables 버킷을 Amazon SageMaker Lakehouse와 통합하세요. 이 통합을 통해 Amazon Athena 및 Amazon Redshift와 같은 AWS 분석 서비스가 테이블 데이터를 자동으로 검색하고 액세스할 수 있습니다.

테이블을 생성하면 Amazon S3가 테이블의 웨어하우스 위치를 자동으로 생성합니다. 테이블과 연결된 객체를 저장하는 고유한 S3 위치입니다. 다음 예시에서는 웨어하우스 위치의 형식을 보여줍니다.

```
s3://63a8e430-6e0b-46f5-k833abtwr6s8tmtsycedn8s4yc3xhuse1b--table-s3
```

테이블 버킷 내에서 테이블을 네임스페이스라는 논리적 그룹으로 구성할 수 있습니다. 자세한 내용은 테이블 네임스페이스 섹션을 참조하세요.

테이블의 이름을 바꿀 수 있지만 각 테이블에는 고유한 Amazon 리소스 이름(ARN)과 고유한 테이블 ID가 있습니다. 각 테이블에는 리소스 정책도 연결되어 있습니다. 이 정책을 사용하여 테이블에 대한 액세스를 관리할 수 있습니다.

테이블 ARN은 다음 형식을 사용합니다.

```
arn:aws:s3tables:region:owner-account-id:bucket/bucket-name/table/table-id
```

기본적으로 테이블 버킷에 최대 1만 개의 테이블을 생성할 수 있습니다. 테이블 버킷 또는 테이블에 대한 할당량 증가를 요청하려면 지원에 문의하세요.

Amazon S3는 테이블 버킷에서 다음 유형의 테이블을 지원합니다.

고객 테이블

고객 테이블은 사용자가 읽고 쓸 수 있는 테이블입니다. 통합 쿼리 엔진을 사용하여 이러한 테이블에서 데이터를 검색할 수 있습니다. S3 API 작업 또는 통합 쿼리 엔진을 사용하여 테이블 내에 데이터를 삽입, 업데이트 또는 삭제할 수 있습니다.

AWS 테이블

AWS 테이블은 사용자를 대신하여 AWS 서비스에서 생성되는 읽기 전용 테이블입니다. 이러한 테이블은 Amazon S3에서 관리하며 Amazon S3 외부의 IAM 위탁자는 수정할 수 없습니다. 이러한 테이블에서 정보를 검색할 수 있지만 해당 테이블의 데이터는 수정할 수 없습니다. AWS 테이블에는 S3 범용 버킷 내의 객체에서 캡처된 메타데이터가 들어 있는 S3 메타데이터 테이블이 포함됩니다. 자세한 내용은 S3 메타데이터를 사용하여 데이터 검색 가속화 섹션을 참조하세요.

주제

- Amazon S3 테이블 생성
- Amazon S3 테이블 삭제
- 테이블 정책 관리

Amazon S3 테이블 생성

Amazon S3 테이블은 테이블 버킷의 하위 리소스입니다. 테이블은 Apache Iceberg 형식으로 저장되므로 쿼리 엔진 및 Apache Iceberg를 지원하는 기타 애플리케이션을 사용하여 테이블을 사용할 수 있습니다. Amazon S3는 테이블을 지속적으로 최적화하여 스토리지 비용을 줄이고 분석 쿼리 성능을 개선합니다.

테이블을 생성하면 Amazon S3가 테이블의 웨어하우스 위치를 자동으로 생성합니다. 웨어하우스 위치는 테이블과 연결된 객체를 읽고 쓸 수 있는 고유한 S3 위치입니다. 다음 예시에서는 웨어하우스 위치의 형식을 보여줍니다.

```
s3://63a8e430-6e0b-46f5-k833abtwr6s8tmtsycedn8s4yc3xhuse1b--table-s3
```

테이블의 Amazon 리소스 이름(ARN) 형식은 다음과 같습니다.

```
arn:aws:s3tables:region:owner-account-id:bucket/bucket-name/table/table-id
```

기본적으로 테이블 버킷에 최대 1만 개의 테이블을 생성할 수 있습니다. 테이블 버킷 또는 테이블에 대한 할당량 증가를 요청하려면 지원에 문의하세요.

Amazon S3 콘솔, Amazon S3 REST API, AWS SDK, AWS Command Line Interface(AWS CLI) 또는 테이블 버킷에 연결된 쿼리 엔진을 사용하여 테이블을 생성할 수 있습니다.

Athena를 사용하여 테이블을 생성하지 않는 한, 테이블을 생성할 때 해당 테이블에 대한 암호화 설정을 지정할 수 있습니다. 암호화 설정을 지정하지 않으면 테이블이 테이블 버킷의 기본 설정으로 암호화됩니다. 자세한 내용은 테이블에 대한 암호화 지정 섹션을 참조하세요.

테이블 생성을 위한 사전 조건

테이블을 생성하려면 먼저 다음을 수행해야 합니다.

- 테이블 버킷 생성.
- 테이블 버킷에 네임스페이스 생성합니다.
- `s3tables:CreateTable` 및 `s3tables:PutTableData`에 대한 AWS Identity and Access Management(IAM) 권한이 있는지 확인합니다.
-

> **Note**
>
> 테이블에 SSE-KMS 암호화를 사용하는 경우 `s3tables:PutTableEncryption` 권한과 선택한 AWS KMS 키에 대한 `DescribeKey` 권한이 필요합니다. 또한 사용하는 AWS KMS 키가 S3 Tables에 자동 테이블 유지 관리를 수행할 수 있는 권한을 부여해야 합니다. 자세한 내용은 S3 Tables SSE-KMS 암호화에 대한 권한 요구 사항 섹션을 참조하세요.

유효한 테이블 이름에 대한 자세한 내용은 테이블 및 네임스페이스의 이름 지정 규칙 섹션을 참조하세요.

> **Important**
>
> 테이블을 만들 때 테이블 이름 및 테이블 정의에 모두 소문자를 사용해야 합니다. 예를 들어 열 이름이 모두 소문자인지 확인합니다. 테이블 이름 또는 테이블 정의에 대문자가 포함된 경우 테이블은 AWS Lake Formation 또는 AWS Glue Data Catalog에서 지원되지 않습니다. 이 경우 테이블 버킷이 AWS 분석 서비스와 통합되어 있더라도 Amazon Athena와 같은 AWS 분석 서비스에는 테이블이 표시되지 않습니다.
>
> 테이블 정의에 대문자가 포함된 경우 Athena에서 `SELECT` 쿼리를 실행할 때 "GENERIC_INTERNAL_ERROR: Get table request failed: com.amazonaws.services.glue.model.ValidationException: Unsupported Federation Resource - Invalid table or column names."라는 오류 메시지가 표시됩니다.

S3 콘솔 및 Amazon Athena 사용

다음 절차에서는 Amazon S3 콘솔을 사용하여 Amazon Athena로 테이블을 생성합니다. 테이블 버킷에 네임스페이스를 아직 생성하지 않은 경우 이 프로세스 중에 생성할 수 있습니다. 다음 단계를 수행하기 전에 테이블 버킷을 이 리전의 AWS 분석 서비스와 통합했는지 확인합니다. 자세한 내용은 the section called “AWS 분석 서비스에서 S3 Tables 사용” 섹션을 참조하세요.

> **Note**
>
> Athena를 사용하여 테이블을 생성하면 해당 테이블은 테이블 버킷에서 기본 암호화 설정을 상속합니다. 다른 암호화 유형을 사용하려면 다른 방법을 사용하여 테이블을 생성해야 합니다.

테이블을 만들려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 테이블 버킷을 선택합니다.
3. 테이블 버킷 페이지에서 테이블을 생성할 버킷을 선택합니다.
4. 버킷 세부 정보 페이지에서 Athena로 테이블 생성을 선택합니다.
5. Athena로 테이블 생성 대화 상자에서 다음 중 하나를 수행합니다.
   - 네임스페이스를 새로 생성합니다. 네임스페이스 생성을 선택한 다음 네임스페이스 이름 필드에 이름을 입력합니다. 네임스페이스 이름은 1~255자여야 하며 테이블 버킷 내에서 고유해야 합니다. 유효한 문자는 a-z, 0-9 및 밑줄(_)입니다. 네임스페이스 이름 시작에는 밑줄이 허용되지 않습니다.
   - Create namespace(네임스페이스 생성)를 선택합니다.
   - 기존 네임스페이스를 지정합니다. 이 테이블 버킷 내의 기존 네임스페이스 지정을 선택합니다. 그런 다음 기존 네임스페이스에서 선택 또는 기존 네임스페이스 이름 입력을 선택합니다. 버킷에 네임스페이스가 1,000개 이상인 경우 목록에 표시되지 않는 네임스페이스 이름을 입력해야 합니다.
6. Athena로 테이블 생성을 선택합니다.
7. Amazon Athena 콘솔이 열리고 Athena 쿼리 편집기가 나타납니다. 카탈로그 필드는 s3tablescatalog/와 같은 테이블 버킷 이름으로 채워지고, 테이블 버킷의 이름, 예를 들어, s3tablescatalog/***amzn-s3-demo-bucket***로 채워야 합니다. 데이터베이스 필드는 이전에 생성하거나 선택한 네임스페이스로 채워져야 합니다.

> **Note**
>
> 카탈로그 및 데이터베이스 필드에 이러한 값이 표시되지 않는 경우 테이블 버킷을 이 리전의 AWS 분석 서비스와 통합했는지 확인합니다. 자세한 내용은 the section called “AWS 분석 서비스에서 S3 Tables 사용” 섹션을 참조하세요.

8. 쿼리 편집기는 테이블을 생성하는 데 사용할 수 있는 샘플 쿼리로 채워집니다. 쿼리를 수정하여 테이블에 포함할 테이블 이름과 열을 지정합니다.
9. 쿼리 수정을 마치면 실행을 선택하여 테이블을 생성합니다.

> **Note**
>
> - Athena에서 쿼리를 실행하려고 할 때 “쿼리를 실행할 권한이 충분하지 않습니다. 위탁자는 지정된 리소스에 대한 권한이 없습니다." 오류가 표시되면 테이블에서 필요한 Lake Formation 권한이 부여되어야 합니다. 자세한 내용은 the section called “테이블 또는 데이터베이스에 대한 권한 부여” 섹션을 참조하세요.
> - Athena에서 쿼리를 실행하려고 할 때 "Iceberg가 요청된 리소스에 액세스할 수 없음" 오류가 표시되면 AWS Lake Formation 콘솔로 이동하여 생성한 테이블 버킷 카탈로그 및 데이터베이스(네임스페이스)에 대한 권한을 자신에게 부여했는지 확인합니다. 이러한 권한을 부여할 때 테이블을 지정하지 마십시오. 자세한 내용은 the section called “테이블 또는 데이터베이스에 대한 권한 부여” 섹션을 참조하세요.
> - Athena에서 SELECT 쿼리를 실행할 때 다음 오류 메시지가 표시되는 경우 테이블 이름이나 테이블 정의의 열 이름에 대문자가 있기 때문입니다. "GENERIC_INTERNAL_ERROR: Get table request failed: com.amazonaws.services.glue.model.ValidationException: Unsupported Federation Resource - Invalid table or column names." 테이블 및 열 이름이 모두 소문자인지 확인합니다.

테이블 생성에 성공하면 Athena의 테이블 목록에 새 테이블 이름이 나타납니다. Amazon S3 콘솔로 돌아가면 목록을 새로 고친 후 테이블 버킷의 버킷 세부 정보 페이지에 있는 테이블 목록에 새 테이블이 나타납니다.

AWS CLI 사용

이 예제에서는 AWS CLI를 사용하고 JSON으로 테이블 메타데이터를 지정하여 스키마가 있는 테이블을 생성하는 방법을 보여줍니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws s3tables create-table --cli-input-json file://mytabledefinition.json
```

mytabledefinition.json 파일의 경우 다음 예제 테이블 정의를 사용합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
{
    "tableBucketARN": "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket",
    "namespace": "your_namespace",
    "name": "example_table",
    "format": "ICEBERG",
    "metadata": {
        "iceberg": {
            "schema": {
                "fields": [
                     {"name": "id", "type": "int","required": true},
                     {"name": "name", "type": "string"},
                     {"name": "value", "type": "int"}
                ]
            }
        }
    }
}
```

쿼리 엔진 사용

Amazon EMR의 Apache Spark 세션에서와 같이 테이블 버킷에 연결된 지원되는 쿼리 엔진에서 테이블을 생성할 수 있습니다.

이 예시에서는 Spark에서 CREATE 문을 사용하여 테이블을 생성하고 INSERT 문을 사용하여 테이블 데이터를 추가하거나 기존 파일에서 데이터를 읽는 방법을 보여줍니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
spark.sql(
```

```
" CREATE TABLE IF NOT EXISTS s3tablesbucket.example_namespace.`example_table` (
    id INT,
    name STRING,
    value INT
)
USING iceberg "
)
```

테이블을 생성한 후 테이블에 데이터를 로드할 수 있습니다. 다음 방법 중에서 선택하세요.

- INSERT 문을 사용하여 테이블에 데이터를 추가합니다.

```
spark.sql(
"""
    INSERT INTO s3tablesbucket.my_namespace.my_table
    VALUES
        (1, 'ABC', 100),
        (2, 'XYZ', 200)
""")
```

- 기존 데이터 파일을 로드합니다.
  1. Spark로 데이터를 읽습니다.

```
val data_file_location = "Path such as S3 URI to data file"
val data_file = spark.read.parquet(data_file_location)
```

  2. Iceberg 테이블에 데이터를 기록합니다.

```
data_file.writeTo("s3tablesbucket.my_namespace.my_table").using("Iceberg").tableProperty
 ("format-version", "2").createOrReplace()
```

Amazon S3 테이블 삭제

Amazon S3 REST API, AWS SDK AWS CLI 또는 통합 쿼리 엔진을 사용하여 테이블을 삭제할 수 있습니다.

> **Note**
>
> S3 Tables은 purge=false를 사용한 DROP TABLE 작업을 지원하지 않습니다. Spark의 일부 버전은 DROP TABLE PURGE 명령을 실행할 때도 항상 이 플래그를 false로 설정합니다.

purge=true를 사용하여 DROP TABLE로 재시도하거나 S3 Tables DeleteTable REST API를 사용하여 테이블을 삭제할 수 있습니다.
테이블을 삭제하면 해당 테이블과 연결된 객체가 비최신이 되며 제거되는 데 최대 1일이 걸릴 수 있습니다.

AWS CLI 사용

이 예시에서는 AWS CLI를 사용하여 테이블을 삭제하는 방법을 보여줍니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

```
aws s3tables delete-table \
    --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
table-bucket \
    --namespace example_namespace --name example_table
```

쿼리 엔진 사용

Amazon S3 테이블 버킷에 연결된 Apache Spark 세션에서 테이블을 삭제할 수 있습니다.

이 예시에서는 DROP TABLE PURGE 명령을 사용하여 테이블을 삭제하는 방법을 보여줍니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

```
spark.sql(
" DROP TABLE [IF EXISTS] s3tablesbucket.example_namespace.example_table PURGE")
```

테이블 정책 관리

Amazon S3 REST API, AWS SDK 및 AWS CLI를 사용하여 테이블에 대한 테이블 정책을 추가, 삭제, 업데이트 및 확인할 수 있습니다. 자세한 내용은 다음 항목을 참조하세요. Amazon S3 Tables에 지원되는 AWS Identity and Access Management(IAM) 작업 및 조건 키에 대한 자세한 내용은 S3 Tables의 액세스 관리 섹션을 참조하세요. 테이블 정책 예시는 S3 Tables의 리소스 기반 정책 섹션을 참조하세요.

테이블 정책 추가

테이블에 테이블 정책을 추가하려면 Amazon S3 REST API, AWS SDK 및 AWS CLI를 사용할 수 있습니다.

AWS CLI 사용

이 예시에서는 AWS CLI를 사용하여 테이블 정책을 생성하는 방법을 보여줍니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

```
aws s3tables put-table-policy \
    --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
bucket1 \
    --namespace my-namespace \
    --name my-table \
    --resource-policy your-policy-JSON
```

테이블 정책 보기

테이블에 연결된 버킷 정책을 보려면 Amazon S3 REST API, AWS SDK 및 AWS CLI를 사용할 수 있습니다.

AWS CLI 사용

이 예시는 AWS CLI를 사용하여 테이블에 연결된 정책을 확인하는 방법을 보여줍니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

```
aws s3tables get-table-policy \
    --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
bucket1/table/tableID \
    --namespace my-namespace \
    --name my-table
```

테이블 정책 삭제

테이블에 연결된 정책을 삭제하려면 Amazon S3 REST API, AWS SDK 및 AWS CLI를 사용할 수 있습니다.

AWS CLI 사용

이 예시에서는 AWS CLI를 사용하여 테이블 정책을 삭제하는 방법을 보여줍니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

```
aws s3tables delete-table-policy \
    --table-ARN arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-bucket1/
table/tableID \
    --namespace your-namespace \
```

```
    --name your-table
```

테이블 데이터에 액세스

Amazon S3 테이블 버킷의 테이블에 액세스하는 방법에는 여러 가지가 있습니다. Amazon SageMaker Lakehouse를 사용하여 테이블을 AWS 분석 서비스와 통합하거나 Amazon S3 Tables Iceberg REST 엔드포인트 또는 Amazon S3 Tables Catalog for Apache Iceberg를 사용하여 직접 테이블에 액세스할 수 있습니다. 사용하는 액세스 방법은 카탈로그 설정, 거버넌스 모델 및 액세스 제어 요구 사항에 따라 달라집니다. 다음은 이러한 액세스 방법에 대한 개요입니다.

Amazon SageMaker Lakehouse 통합

이는 S3 테이블 버킷의 테이블 작업을 위해 권장되는 액세스 방법입니다. 통합을 통해 통합 테이블 관리, 중앙 집중식 거버넌스, 여러 AWS 분석 서비스에서 세분화된 액세스 제어를 제공합니다.

직접 액세스

AWS Partner Network(APN) 카탈로그 구현, 사용자 지정 카탈로그 구현을 사용해야 하거나 단일 테이블 버킷 내의 테이블에 대한 기본 읽기/쓰기 작업만 수행해야 하는 경우 이 방법을 사용합니다.

> **Note**
>
> 테이블에 액세스하려면 사용하는 IAM ID에 테이블 리소스 및 S3 Tables 작업에 대한 액세스 권한이 있어야 합니다. 자세한 내용은 S3 Tables의 액세스 관리 섹션을 참조하세요.

Amazon SageMaker Lakehouse 통합을 통해 테이블에 액세스

S3 테이블 버킷을 Amazon SageMaker Lakehouse와 통합하여 Amazon Athena, Amazon Redshift, QuickSight와 같은 AWS 분석 서비스의 테이블에 액세스할 수 있습니다. Amazon SageMaker Lakehouse는 Amazon S3 데이터 레이크와 Amazon Redshift 데이터 웨어하우스의 모든 데이터를 통합하여 단일 데이터 사본에 기반하는 분석, 머신 러닝(ML), 생성 AI 애플리케이션을 구축할 수 있도록 지원합니다. 통합은 AWS Glue Data Catalog를 테이블 리소스로 채우고 이러한 리소스에 대한 액세스를 AWS Lake Formation과 페더레이션합니다. 통합에 대한 자세한 내용은 AWS 분석 서비스에서 Amazon S3 Tables 사용 섹션을 참조하세요.

통합을 사용하면 AWS Lake Formation을 통해 세분화된 액세스 제어를 사용해서 추가 보안을 제공할 수 있습니다. Lake Formation은 자체 권한 모델과 IAM 권한 모델의 조합을 사용하여 테이블 리소스 및 기본 데이터에 대한 액세스를 제어합니다. 즉, 테이블에 대한 액세스 요청은 IAM과 Lake Formation

모두의 권한 확인을 통과해야 합니다. 자세한 내용은 AWS Lake Formation 개발자 안내서의 Lake Formation 권한 개요를 참조하세요.

다음 AWS 분석 서비스는 이 통합을 통해 테이블에 액세스할 수 있습니다.

- Amazon Athena
- Amazon Redshift
- Amazon EMR
- QuickSight
- Amazon Data Firehose

AWS GlueIceberg REST 엔드포인트를 사용하여 테이블에 액세스

S3 테이블 버킷이 Amazon SageMaker Lakehouse와 통합되면 AWS Glue Iceberg REST 엔드포인트를 사용하여 Iceberg를 지원하는 타사 쿼리 엔진의 S3 테이블에 연결할 수도 있습니다. 자세한 내용은 AWS GlueIceberg REST 엔드포인트를 사용하여 Amazon S3 Tables에 액세스 섹션을 참조하세요.

Spark, PyIceberg 또는 기타 Iceberg 호환 클라이언트에서 테이블에 액세스하려는 경우 AWS Glue Iceberg REST 엔드포인트를 사용하는 것이 좋습니다.

다음 클라이언트는 AWS Glue Iceberg REST 엔드포인트를 통해 테이블에 직접 액세스할 수 있습니다.

- Spark, PyIceberg 등을 포함한 모든 Iceberg 클라이언트

테이블에 직접 액세스

S3 Tables 관리 작업을 Apache Iceberg 분석 애플리케이션에 연결하는 방법을 통해 오픈 소스 쿼리 엔진에서 직접 테이블에 액세스할 수 있습니다. Amazon S3 Tables Iceberg REST 엔드포인트 또는 Amazon S3 Tables Catalog for Apache Iceberg라는 두 가지 직접 액세스 방법이 있습니다. REST 엔드포인트를 사용하는 것이 좋습니다.

자체 관리형 카탈로그 구현의 테이블에 액세스하거나 단일 테이블 버킷의 테이블에 대한 기본 읽기/쓰기 작업만 수행하면 되는 경우 직접 액세스하는 것이 좋습니다. 다른 액세스 시나리오의 경우 Amazon SageMaker Lakehouse 통합을 권장합니다.

테이블에 대한 직접 액세스는 IAM 자격 증명 기반 정책 또는 테이블 및 테이블 버킷에 연결된 리소스 기반 정책을 통해 관리됩니다. 테이블에 직접 액세스할 때 테이블에 대한 Lake Formation 권한을 관리할 필요가 없습니다.

Amazon S3 Tables Iceberg REST 엔드포인트를 통해 테이블에 액세스

Amazon S3 TablesIceberg REST 엔드포인트를 사용하여 HTTP 엔드포인트를 통해 Iceberg REST 호환 클라이언트에서 직접 테이블에 액세스할 수 있습니다. 자세한 내용은 Amazon S3 Tables Iceberg REST 엔드포인트를 사용하여 테이블에 액세스 섹션을 참조하세요.

다음 AWS 분석 서비스 및 쿼리 엔진은 Amazon S3 Tables Iceberg REST 엔드포인트를 사용하여 테이블에 직접 액세스할 수 있습니다.

지원되는 쿼리 엔진

- Spark, PyIceberg 등을 포함한 모든 Iceberg 클라이언트
- Amazon EMR
- AWS Glue ETL

Amazon S3 Tables Catalog for Apache Iceberg를 통해 직접 테이블에 액세스

S3 Tables 클라이언트 카탈로그를 사용하여 Apache Spark와 같은 쿼리 엔진에서 직접 테이블에 액세스할 수도 있습니다. 자세한 내용은 Amazon S3 Tables Catalog for Apache Iceberg를 사용하여 Amazon S3 Tables에 액세스 섹션을 참조하세요. 그러나 S3는 언어 또는 엔진별 코드 없이 더 많은 애플리케이션을 지원하므로 직접 액세스에 Amazon S3 TablesIceberg REST 엔드포인트를 사용할 것을 권장합니다.

다음 쿼리 엔진은 클라이언트 카탈로그를 사용하여 테이블에 직접 액세스할 수 있습니다.

- Apache Spark

AWS 분석 서비스에서 Amazon S3 Tables 사용

AWS 분석 서비스에서 계정의 테이블에 액세스할 수 있도록 하려면 Amazon S3 Tables 버킷을 Amazon SageMaker Lakehouse와 통합하세요. 이 통합을 통해 AWS 분석 서비스는 테이블 데이터를 자동으로 검색하고 액세스할 수 있습니다. 이 통합을 사용하여 아래 서비스에서 테이블을 사용할 수 있습니다.

- Amazon Athena
- Amazon Redshift
- Amazon EMR

- QuickSight
- Amazon Data Firehose

> **Note**
>
> 이 통합은 AWS Glue 및 AWS Lake Formation 서비스를 사용하며 AWS Glue 요청 및 스토리지 비용이 발생할 수 있습니다. 자세한 내용은 AWS Glue 요금을 참조하세요.
> S3 테이블에서 쿼리를 실행하는 경우 추가 요금이 적용됩니다. 자세한 내용은 사용 중인 쿼리 엔진의 요금 정보를 참조하세요.

통합의 작동 방식

콘솔에서 테이블 버킷을 생성하면 Amazon S3는 다음 작업을 시작하여 선택한 리전의 테이블 버킷을 AWS 분석 서비스와 통합합니다.

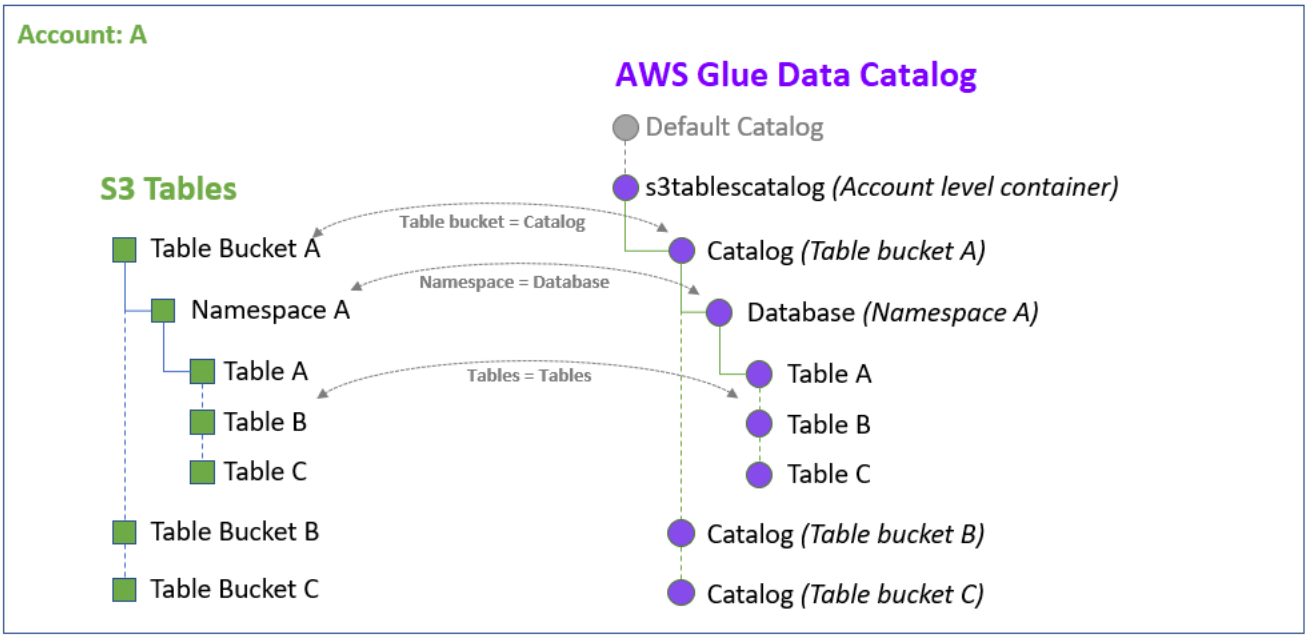
1. Lake Formation에 모든 테이블 버킷에 대한 액세스 권한을 부여하는 새 AWS Identity and Access Management(IAM) 서비스 역할을 생성합니다.
2. Lake Formation은 이 서비스 역할을 사용하여 현재 리전에 테이블 버킷을 등록합니다. 이를 통해 Lake Formation은 해당 리전의 모든 현재 및 미래 테이블 버킷에 대한 액세스, 권한 및 거버넌스를 관리할 수 있습니다.
3. 현재 리전의 AWS Glue Data Catalog에 `s3tablescatalog` 카탈로그를 추가합니다. `s3tablescatalog` 카탈로그를 추가하면 모든 테이블 버킷, 네임스페이스 및 테이블이 Data Catalog에 채워집니다.

> **Note**
>
> 이러한 작업은 Amazon S3 콘솔을 통해 자동화됩니다. 프로그래밍 방식으로 이 통합을 수행하는 경우 이러한 모든 작업을 수동으로 수행해야 합니다.

테이블 버킷을 AWS 리전당 한 번 통합합니다. 통합이 완료되면 모든 현재 및 미래의 테이블 버킷, 네임스페이스 및 테이블이 해당 리전의 AWS Glue Data Catalog에 추가됩니다.

다음 그림은 `s3tablescatalog` 카탈로그가 현재 리전의 테이블 버킷, 네임스페이스 및 테이블을 Data Catalog에서 대응하는 객체로 자동으로 채우는 방법을 보여줍니다. 테이블 버킷은 하위 카탈로그

로 채워집니다. 테이블 버킷 내의 네임스페이스는 해당 하위 카탈로그 내의 데이터베이스로 채워집니다. 테이블은 해당하는 데이터베이스의 테이블로 채워집니다.



권한이 작동하는 방법

AWS Glue Data Catalog를 메타데이터 스토어로 사용하는 서비스 전체에서 테이블 데이터를 사용할 수 있도록 테이블 버킷을 AWS 분석 서비스와 통합하는 것이 좋습니다. 통합을 사용하면 AWS Lake Formation을 통해 세분화된 액세스 제어를 사용할 수 있습니다. 이 보안 접근 방식은 AWS Identity and Access Management(IAM) 권한 외에도 IAM 위탁자에게 테이블에 대한 Lake Formation 권한을 부여해야 작업할 수 있습니다.

AWS Lake Formation에는 다음과 같은 두 가지 주요 권한 유형이 있습니다.

- 메타데이터 액세스 권한은 Data Catalog에서 메타데이터 데이터베이스 및 테이블을 생성하고, 읽고, 업데이트하고, 삭제할 수 있는 기능을 제어합니다.
- 기본 데이터 액세스 권한은 Data Catalog 리소스가 가리키는 기본 Amazon S3 위치에 데이터를 읽고 쓰는 기능을 제어합니다.

Lake Formation은 자체 권한 모델과 IAM 권한 모델의 조합을 사용하여 Data Catalog 리소스 및 기본 데이터에 대한 액세스를 제어합니다.

- Data Catalog 리소스 또는 기본 데이터에 대한 액세스 요청이 성공하려면 요청이 IAM과 Lake Formation의 권한 검사를 모두 통과해야 합니다.

- IAM 권한은 Lake Formation 및 AWS Glue API 및 리소스에 대한 액세스를 제어하는 반면, Lake Formation 권한은 Data Catalog 리소스, Amazon S3 위치 및 기본 데이터에 대한 액세스를 제어합니다.

Lake Formation 권한은 권한이 부여된 리전에서만 적용되며, 위탁자는 Lake Formation 권한을 부여받으려면 필요한 권한이 있는 데이터 레이크 관리자 또는 다른 위탁자의 승인을 받아야 합니다.

자세한 내용은 AWS Lake Formation 개발자 안내서의 Lake Formation 권한 개요를 참조하십시오.

the section called “통합 조건” 및 the section called “테이블 버킷을 AWS 분석 서비스와 통합”의 단계를 수행하여 AWS Glue Data Catalog 및 테이블 리소스에 액세스하고 AWS 분석 서비스를 사용할 수 있는 적절한 권한이 있는지 확인합니다.

> **Important**
>
> 계정의 AWS 분석 서비스와 테이블 버킷 통합을 수행한 사용자가 아닌 경우 테이블에 필요한 Lake Formation 권한을 부여받아야 합니다. 자세한 내용은 the section called “테이블 또는 데이터베이스에 대한 권한 부여” 섹션을 참조하세요.

통합 조건

테이블 버킷을 AWS 분석 서비스와 통합하려면 다음 사전 조건을 충족해야 합니다.

- 테이블 버킷을 생성합니다.
- AWSLakeFormationDataAdmin AWS 관리형 정책을 AWS Identity and Access Management(IAM) 위탁자에 연결하여 해당 사용자를 데이터 레이크 관리자로 생성합니다. 데이터 레이크 관리자 생성 방법에 대한 자세한 내용은 AWS Lake Formation 개발자 안내서의 데이터 레이크 관리자 생성을 참조하세요.
- IAM 위탁자에 `glue:PassConnection` 작업에 대한 권한을 추가합니다.
- IAM 위탁자에 `lakeformation:RegisterResource` 및 `lakeformation:RegisterResourceWithPrivilegedAccess` 작업에 대한 권한을 추가합니다.
- AWS Command Line Interface(AWS CLI)의 최신 버전으로 업데이트합니다.

> **Important**
>
> 테이블을 만들 때 테이블 이름 및 테이블 정의에 모두 소문자를 사용해야 합니다. 예를 들어 열 이름이 모두 소문자인지 확인합니다. 테이블 이름 또는 테이블 정의에 대문자가 포함된 경우 테이블은 AWS Lake Formation 또는 AWS Glue Data Catalog에서 지원되지 않습니다. 이 경우 테이블 버킷이 AWS 분석 서비스와 통합되어 있더라도 Amazon Athena와 같은 AWS 분석 서비스에는 테이블이 표시되지 않습니다.
> 테이블 정의에 대문자가 포함된 경우 Athena에서 `SELECT` 쿼리를 실행할 때 "GENERIC_INTERNAL_ERROR: Get table request failed: com.amazonaws.services.glue.model.ValidationException: Unsupported Federation Resource - Invalid table or column names."라는 오류 메시지가 표시됩니다.

테이블 버킷을 AWS 분석 서비스와 통합

이 통합은 AWS 리전당 한 번 수행해야 합니다.

> **Important**
>
> 이제 AWS 분석 서비스 통합은 `registerResource` Lake Formation API 작업의 `WithPrivilegedAccess` 옵션을 사용하여 S3 테이블 버킷을 등록합니다. 통합은 이제 `CreateCatalog` AWS Glue API 작업의 `AllowFullTableExternalDataAccess` 옵션을 사용하여 AWS Glue Data Catalog에 `s3tablescatalog` 카탈로그도 생성합니다.
> 미리 보기 릴리스와의 통합을 설정한 경우 현재 통합을 계속 사용할 수 있습니다. 그러나 업데이트된 통합 프로세스는 성능 개선을 제공하므로 마이그레이션하는 것이 좋습니다. 업데이트된 통합으로 마이그레이션하려면 the section called “업데이트된 통합 프로세스로 마이그레이션” 섹션을 참조하세요.

S3 콘솔 사용

1. https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 테이블 버킷을 선택합니다.
3. 테이블 버킷 생성을 선택합니다.

   테이블 버킷 생성 페이지가 열립니다.
4. 테이블 버킷 이름을 입력하고 통합 활성화 확인란이 선택되어 있는지 확인합니다.

5. 테이블 버킷 생성을 선택합니다. Amazon S3는 해당 리전에서 테이블 버킷을 자동으로 통합하려고 시도합니다.

어떤 리전에서든 테이블 버킷을 처음 통합하면 Amazon S3가 사용자를 대신하여 새 IAM 서비스 역할을 생성합니다. 이 역할을 통해 Lake Formation은 계정의 모든 테이블 버킷에 액세스하고 AWS Glue Data Catalog의 테이블에 대한 액세스를 페더레이션할 수 있습니다.

AWS CLI 사용

AWS CLI를 사용하여 테이블 버킷을 통합하는 방법

다음 단계에서는 AWS CLI를 사용하여 테이블 버킷을 통합하는 방법을 보여줍니다. 이 단계를 사용하려면 *user input placeholders*를 실제 정보로 대체합니다.

1. 테이블 버킷을 생성합니다.

```
aws s3tables create-table-bucket \
--region us-east-1 \
--name amzn-s3-demo-table-bucket
```

2. Lake Formation이 테이블 리소스에 액세스할 수 있도록 허용하는 IAM 서비스 역할을 생성합니다.

   a. 다음 신뢰 정책을 포함하는 `Role-Trust-Policy.json`이라는 파일을 생성합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "LakeFormationDataAccessPolicy",
        "Effect": "Allow",
        "Principal": {
          "Service": "lakeformation.amazonaws.com"
        },
        "Action": [
            "sts:AssumeRole",
            "sts:SetContext",
            "sts:SetSourceIdentity"
        ],
        "Condition": {
          "StringEquals": {
            "aws:SourceAccount": "111122223333"
          }
```

```
        }
      }
    ]
}
```

다음 명령을 사용하여 IAM 서비스 역할을 생성합니다.

```
aws iam create-role \
--role-name S3TablesRoleForLakeFormation \
--assume-role-policy-document file://Role-Trust-Policy.json
```

b. 다음 신뢰 정책을 포함하는 LF-GluePolicy.json이라는 파일을 생성합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LakeFormationPermissionsForS3ListTableBucket",
            "Effect": "Allow",
            "Action": [
                "s3tables:ListTableBuckets"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "LakeFormationDataAccessPermissionsForS3TableBucket",
            "Effect": "Allow",
            "Action": [
                "s3tables:CreateTableBucket",
                "s3tables:GetTableBucket",
                "s3tables:CreateNamespace",
                "s3tables:GetNamespace",
                "s3tables:ListNamespaces",
                "s3tables:DeleteNamespace",
                "s3tables:DeleteTableBucket",
                "s3tables:CreateTable",
                "s3tables:DeleteTable",
                "s3tables:GetTable",
                "s3tables:ListTables",
                "s3tables:RenameTable",
                "s3tables:UpdateTableMetadataLocation",
```

```
                "s3tables:GetTableMetadataLocation",
                "s3tables:GetTableData",
                "s3tables:PutTableData"
            ],
            "Resource": [
                "arn:aws:s3tables:us-east-1:111122223333:bucket/*"
            ]
        }
    ]
}
```

다음 명령을 사용하여 정책을 역할에 연결합니다.

```
aws iam put-role-policy \
--role-name S3TablesRoleForLakeFormation  \
--policy-name LakeFormationDataAccessPermissionsForS3TableBucket \
--policy-document file://LF-GluePolicy.json
```

3. 다음을 포함하는 input.json이라는 파일을 생성합니다.

```
{
    "ResourceArn": "arn:aws:s3tables:us-east-1:111122223333:bucket/*",

    "WithFederation": true,
    "RoleArn": "arn:aws:iam::111122223333:role/S3TablesRoleForLakeFormation"
}
```

다음 명령을 사용하여 Lake Formation에 테이블 버킷을 등록합니다.

```
aws lakeformation register-resource \
--region us-east-1 \
--with-privileged-access \
--cli-input-json file://input.json
```

4. 다음 카탈로그를 포함하는 catalog.json이라는 파일을 생성합니다.

```
{
   "Name": "s3tablescatalog",
   "CatalogInput": {
      "FederatedCatalog": {
          "Identifier": "arn:aws:s3tables:us-east-1:111122223333:bucket/*",
          "ConnectionName": "aws:s3tables"
```

```
        },
        "CreateDatabaseDefaultPermissions":[],
        "CreateTableDefaultPermissions":[],
        "AllowFullTableExternalDataAccess": "True"
    }
}
```

다음 명령을 사용하여 s3tablescatalog 카탈로그를 생성합니다. 이 카탈로그를 생성하면 테이블 버킷, 네임스페이스 및 테이블에 해당하는 객체로 AWS Glue Data Catalog를 채웁니다.

```
aws glue create-catalog \
--region us-east-1 \
--cli-input-json file://catalog.json
```

5. 다음 명령을 사용하여 s3tablescatalog 카탈로그가 AWS Glue에 추가되었는지 확인합니다.

```
aws glue get-catalog --catalog-id s3tablescatalog
```

업데이트된 통합 프로세스로 마이그레이션

AWS 분석 서비스 통합 프로세스가 업데이트되었습니다. 미리 보기 릴리스와의 통합을 설정한 경우 현재 통합을 계속 사용할 수 있습니다. 그러나 업데이트된 통합 프로세스는 성능을 개선하므로 다음 단계를 사용하여 마이그레이션하는 것이 좋습니다. 마이그레이션 또는 통합 프로세스에 대한 자세한 내용은 AWS Lake Formation 개발자 안내서의 AWS Glue Data Catalog에서 Amazon S3 Tables 카탈로그 생성을 참조하세요.

1. https://console.aws.amazon.com/lakeformation/에서 AWS Lake Formation 콘솔을 열고 데이터 레이크 관리자로 로그인합니다. 데이터 레이크 관리자 생성 방법에 대한 자세한 내용은 AWS Lake Formation 개발자 안내서의 데이터 레이크 관리자 생성을 참조하세요.
2. 다음을 수행하여 s3tablescatalog 카탈로그를 삭제합니다.
   - 왼쪽 탐색 창에서 카탈로그를 선택합니다.
   - 카탈로그 목록의 s3tablescatalog 카탈로그 옆에 있는 옵션 버튼을 선택합니다. [Actions] 메뉴에서 [Delete]를 선택합니다.
3. 다음을 수행하여 s3tablescatalog 카탈로그의 데이터 위치를 등록 취소합니다.
   - 왼쪽 탐색 창에서 관리 섹션으로 이동하고 데이터 레이크 위치를 선택합니다.

- s3tablescatalog 데이터 레이크 위치 옆의 옵션 버튼을 선택합니다. 예를 들어, s3://tables:*region*:*account-id*:bucket/*입니다.
- 작업 메뉴에서 제거를 선택합니다.
- 나타나는 대화 상자에서 제거를 선택합니다.

4. 이제 s3tablescatalog 카탈로그 및 데이터 레이크 위치를 삭제했으므로 업데이트된 통합 프로세스를 사용하여 테이블 버킷을 AWS 분석 서비스와 통합하는 단계를 수행할 수 있습니다.

> **Note**
>
> 통합 AWS 분석 서비스에서 SSE-KMS로 암호화된 테이블에 대한 작업을 수행하려면 사용하는 역할이 암호화 작업에 AWS KMS 키를 사용할 수 있는 권한을 가져야 합니다. 자세한 내용은 IAM 위탁자에게 통합 AWS 분석 서비스에서 암호화된 테이블에 대한 작업을 수행할 수 있는 권한 부여 섹션을 참조하세요.

다음 단계

- 네임스페이스 생성.
- 테이블 생성.

테이블 리소스에 대한 Lake Formation 권한 부여

테이블 버킷이 AWS 분석 서비스와 통합되면 Lake Formation은 테이블 리소스에 대한 액세스를 관리합니다. Lake Formation은 Data Catalog 리소스에 대한 세분화된 액세스 제어를 활성화하는 자체 권한 모델(Lake Formation 권한)을 사용합니다. Lake Formation에서는 Lake Formation에서 관리하는 리소스에 대해 작업을 수행할 수 있는 권한을 각 IAM 위탁자(사용자 또는 역할)에 부여해야 합니다. 자세한 내용은 AWS Lake Formation 개발자 안내서의 Lake Formation 권한 개요를 참조하십시오. 계정 간 데이터 공유에 대한 자세한 내용은 AWS Lake Formation개발자 안내서의 Lake Formation에서의 계정 간 데이터 공유를 참조하세요.

IAM 위탁자가 AWS 분석 서비스의 테이블에 액세스하려면 먼저 해당 리소스에 대한 Lake Formation 권한을 부여해야 합니다.

> **Note**
>
> 테이블 버킷 통합을 수행한 사용자인 경우 테이블에 대한 Lake Formation 권한이 이미 있습니다. 테이블에 액세스할 유일한 위탁자인 경우 이 단계를 건너뛸 수 있습니다. 테이블에 대한

Lake Formation 권한을 다른 IAM 위탁자에게 부여하기만 하면 됩니다. 이렇게 하면 다른 위탁자가 쿼리를 실행할 때 테이블에 액세스할 수 있습니다. 자세한 내용은 테이블 또는 데이터베이스에 대한 권한 부여 섹션을 참조하세요.

다음 서비스에서 테이블 리소스를 사용하려면 테이블 리소스에 대한 Lake Formation 권한을 다른 IAM 위탁자에게 부여해야 합니다.

- Amazon Redshift
- Amazon Data Firehose
- Amazon QuickSight
- Amazon Athena

테이블 또는 데이터베이스에 대한 권한 부여

Lake Formation 콘솔 또는 AWS CLI를 통해 위탁자에게 테이블 버킷의 테이블 또는 데이터베이스에 대한 Lake Formation 권한을 부여할 수 있습니다.

Note

데이터 카탈로그 리소스에 대한 Lake Formation 권한을 외부 계정에 부여하거나 다른 계정의 IAM 보안 주체에 직접 부여하면 Lake Formation은 AWS Resource Access Manager(AWS RAM) 서비스를 사용하여 리소스를 공유합니다. 피부여자 계정이 부여자 계정과 동일한 조직에 속해 있는 경우, 피부여자는 공유 리소스를 즉시 사용할 수 있습니다. 피부여자 계정이 같은 조직에 속해 있지 않은 경우, AWS RAM은 피부여자 계정에 리소스 부여를 수락하거나 거부하라는 초대를 보냅니다. 그런 다음 공유 리소스를 사용할 수 있게 하려면 피부여자 계정의 데이터 레이크 관리자가 AWS RAM 콘솔 또는 AWS CLI를 사용하여 초대를 수락해야 합니다. 계정 간 데이터 공유에 대한 자세한 내용은 AWS Lake Formation개발자 안내서의 Lake Formation에서의 계정 간 데이터 공유를 참조하세요.

Console

1. https://console.aws.amazon.com/lakeformation/에서 AWS Lake Formation 콘솔을 열고 데이터 레이크 관리자로 로그인합니다. 데이터 레이크 관리자 생성 방법에 대한 자세한 내용은 AWS Lake Formation 개발자 안내서의 데이터 레이크 관리자 생성을 참조하세요.
2. 탐색 창에서 데이터 권한을 선택한 다음 권한 부여를 선택합니다.

3. 권한 부여 페이지의 주체에서 다음 중 하나를 수행합니다.
   - Amazon Athena 또는 Amazon Redshift에서 IAM 사용자 및 역할을 선택하고 쿼리에 사용하는 IAM 위탁자를 선택합니다.
   - Amazon Data Firehose에서 IAM 사용자 및 역할을 선택하고 테이블로 스트리밍하기 위해 생성한 서비스 역할을 선택합니다.
   - QuickSight에서 SAML 사용자 및 그룹을 선택하고 QuickSight 관리자 사용자의 Amazon 리소스 이름(ARN)을 입력합니다.
4. LF 태그 또는 카탈로그 리소스에서 명명된 데이터 카탈로그 리소스를 선택합니다.
5. 카탈로그에서 테이블 버킷을 통합할 때 생성한 하위 카탈로그를 선택합니다. 예를 들어, *account-id*:s3tablescatalog/*amzn-s3-demo-bucket*입니다.
6. 데이터베이스에서 생성한 S3 테이블 버킷 네임스페이스를 선택합니다.
7. (옵션) 테이블에서 테이블 버킷에서 생성한 S3 테이블을 선택합니다.

   > **Note**
   >
   > Athena 쿼리 편집기에서 새 테이블을 생성하는 경우 테이블을 선택하지 마세요.

8. 다음 중 하나를 수행합니다.
   - 이전 단계에서 테이블을 지정한 경우 테이블 권한에서 슈퍼를 선택합니다.
   - 이전 단계에서 테이블을 지정하지 않은 경우 데이터베이스 권한으로 이동합니다. 교차 계정 데이터 공유의 경우 슈퍼를 선택하여 다른 위탁자에게 데이터베이스에 대한 모든 권한을 부여할 수 없습니다. 대신 설명과 같이 더 세분화된 권한을 선택합니다.
9. 권한 부여를 선택합니다.

CLI

1. 데이터 레이크 관리자로서 다음 AWS CLI 명령을 실행 중인지 확인합니다. 자세한 내용은 AWS Lake Formation 개발자 안내서의 데이터 레이크 관리자 생성을 참조하세요.
2. 다음 명령을 실행하여 S3 테이블 버킷의 테이블에 대한 Lake Formation 권한을 테이블에 액세스할 IAM 위탁자에 부여합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws lakeformation grant-permissions \
--region us-east-1 \
--cli-input-json \
```

```
'{
    "Principal": {
        "DataLakePrincipalIdentifier": "user or role ARN, for example,
 arn:aws:iam::account-id:role/example-role"
    },
    "Resource": {
        "Table": {
            "CatalogId": "account-id:s3tablescatalog/amzn-s3-demo-bucket",
            "DatabaseName": "S3 table bucket namespace, for example,
 test_namespace",
            "Name": "S3 table bucket table name, for example test_table"
        }
    },
    "Permissions": [
        "ALL"
    ]
}'
```

AWS GlueIceberg REST 엔드포인트를 사용하여 Amazon S3 Tables에 액세스

S3 Tables 버킷이 AWS Glue Data Catalog와 통합되면 AWS Glue Iceberg REST 엔드포인트를 사용하여 PyIceberg 또는 Spark와 같은 Apache Iceberg 호환 클라이언트에서 S3 Tables에 연결할 수 있습니다. AWS Glue Iceberg REST 엔드포인트는 Iceberg 테이블과 상호 작용하기 위한 표준화된 인터페이스를 제공하는 Iceberg REST Catalog Open API 사양을 구현합니다. 엔드포인트를 사용하여 S3 Tables에 액세스하려면 IAM 정책 및 AWS Lake Formation 권한 부여의 조합을 통해 권한을 구성해야 합니다. 다음 섹션에서는 필요한 IAM 역할 생성, 필요한 정책 정의, 데이터베이스 및 테이블 수준 액세스 모두에 대한 Lake Formation 권한 설정 등 액세스를 설정하는 방법을 설명합니다.

PyIceberg를 사용한 엔드 투 엔드 연습은 AWS GlueIceberg REST 엔드포인트를 통해 PyIceberg를 사용하여 Amazon S3 Tables의 데이터 액세스를 참조하세요.

사전 조건

- 테이블 버킷을 AWS 분석 서비스와 통합
- 테이블 네임스페이스 생성
- 데이터 레이크 관리자 계정에 액세스

클라이언트에 대한 IAM 역할 생성

AWS Glue 엔드포인트를 통해 테이블에 액세스하려면 AWS Glue 및 Lake Formation 작업에 대한 권한이 있는 IAM 역할을 생성해야 합니다. 이 절차에서는 이 역할을 생성하고 해당 권한을 구성하는 방법을 설명합니다.

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 정책을 선택합니다.
3. 정책 생성을 선택하고 정책 편집기에서 JSON을 선택합니다.
4. AWS Glue 및 Lake Formation 작업에 액세스할 수 있는 권한을 부여하는 다음 인라인 정책을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "glue:GetCatalog",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:GetTable",
                "glue:GetTables",
                "glue:CreateTable",
                "glue:UpdateTable"
            ],
            "Resource": [
                "arn:aws:glue:<region>:<account-id>:catalog",
                "arn:aws:glue:<region>:<account-id>:catalog/s3tablescatalog",
                "arn:aws:glue:<region>:<account-id>:catalog/
s3tablescatalog/<s3_table_bucket_name>",
                "arn:aws:glue:<region>:<account-id>:table/
s3tablescatalog/<s3_table_bucket_name>/<namespace>/*",
                "arn:aws:glue:<region>:<account-id>:database/
s3tablescatalog/<s3_table_bucket_name>/<namespace>"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "lakeformation:GetDataAccess"
            ],
            "Resource": "*"
        }
    ]
}
```

5. 정책을 생성한 후 IAM 역할을 생성하고 사용자 지정 신뢰 정책을 신뢰할 수 있는 엔터티 유형으로 선택합니다.
6. 사용자 지정 신뢰 정책에 다음을 입력합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::<accountid>:role/<Admin_role>"
            },
            "Action": "sts:AssumeRole",
            "Condition": {}
        }
    ]
}
```

Lake Formation에서 액세스 정의

Lake Formation은 데이터 레이크 테이블의 데이터를 위한 세분화된 액세스 제어를 제공합니다. S3 버킷을 AWS Glue Data Catalog과 통합한 경우 테이블이 Lake Formation에 리소스로 자동으로 등록되어 있습니다. 이러한 테이블에 액세스하려면 IAM 정책 권한 외에도 IAM 자격 증명에 특정 Lake Formation 권한을 부여해야 합니다.

다음 단계에서는 Iceberg 클라이언트가 테이블에 연결할 수 있도록 Lake Formation 액세스 제어를 적용하는 방법을 설명합니다. 이러한 권한을 적용하려면 데이터 레이크 관리자로 로그인해야 합니다.

외부 엔진이 테이블 데이터에 액세스하도록 허용

Lake Formation에서는 외부 엔진이 데이터에 액세스할 수 있도록 전체 테이블 액세스를 활성화해야 합니다. 이렇게 하면 요청된 테이블에 대한 전체 권한이 있는 IAM 역할을 사용할 때 타사 애플리케이션이 Lake Formation에서 임시 자격 증명을 가져올 수 있습니다.

Lake Formation 콘솔(https://console.aws.amazon.com/lakeformation/)을 엽니다.

1. Lake Formation 콘솔(https://console.aws.amazon.com/lakeformation/)을 열고 데이터 레이크 관리자로 로그인합니다.
2. 탐색 창의 관리에서 애플리케이션 통합 설정을 선택합니다.
3. 외부 엔진이 전체 테이블 액세스 권한으로 Amazon S3 위치의 데이터에 액세스하도록 허용을 선택합니다. 그런 다음 저장을 선택합니다.

테이블 리소스에 대한 Lake Formation 권한 부여

다음으로 Iceberg 호환 클라이언트에 대해 생성한 IAM 역할에 Lake Formation 권한을 부여합니다. 이러한 권한을 통해 역할은 네임스페이스에서 테이블을 생성하고 관리할 수 있습니다. 데이터베이스 및 테이블 수준 권한을 모두 제공해야 합니다.

데이터베이스 권한을 부여하려면

1. https://console.aws.amazon.com/lakeformation/에서 AWS Lake Formation 콘솔을 열고 데이터 레이크 관리자로 로그인합니다.
2. 탐색 창에서 데이터 권한을 선택한 다음 권한 부여를 선택합니다.
3. 권한 부여 페이지의 위탁자에서 IAM 사용자 및 역할을 선택하고 AWS Glue Iceberg REST endpoint 액세스를 위해 생성한 IAM 역할을 선택합니다.
4. LF 태그 또는 카탈로그 리소스에서 명명된 데이터 카탈로그 리소스를 선택합니다.
5. 카탈로그에서 테이블 버킷에 대해 생성된 AWS Glue 데이터 카탈로그를 선택합니다. 예를 들어 `<accoundID>:s3tablescatalog/<table-bucket-name>`입니다.
6. 데이터베이스에서 `mynamespace`을 선택합니다.
7. 테이블 권한에서 테이블 생성 및 설명을 선택합니다.
8. 권한 부여를 선택합니다.

테이블 권한을 부여하려면

1. https://console.aws.amazon.com/lakeformation/에서 AWS Lake Formation 콘솔을 열고 데이터 레이크 관리자로 로그인합니다.
2. 탐색 창에서 데이터 권한을 선택한 다음 권한 부여를 선택합니다.
3. 권한 부여 페이지의 위탁자에서 IAM 사용자 및 역할을 선택하고 AWS Glue Iceberg REST endpoint 액세스를 위해 생성한 IAM 역할을 선택합니다.

4. LF 태그 또는 카탈로그 리소스에서 명명된 데이터 카탈로그 리소스를 선택합니다.
5. 카탈로그에서 테이블 버킷에 대해 생성된 AWS Glue 데이터 카탈로그를 선택합니다. 예를 들어 *<accoundID>*:s3tablescatalog/*<table-bucket-name>*입니다.
6. 데이터베이스에서 생성한 S3 테이블 버킷 네임스페이스를 선택합니다.
7. 테이블에서 ALL_TABLES를 선택합니다.
8. 테이블 권한에서 Super를 선택합니다.
9. 권한 부여를 선택합니다.

엔드포인트를 사용하도록 환경 설정

테이블 액세스에 필요한 권한으로 IAM 역할을 설정한 후, 다음 명령을 사용하여 해당 역할로 AWS CLI를 구성하여 로컬 시스템에서 Iceberg 클라이언트를 실행하는 데 사용할 수 있습니다.

```
aws sts assume-role --role-arn "arn:aws:iam::<accountid>:role/<glue-irc-role>" --role-
session-name <glue-irc-role>
```

AWS Glue REST 엔드포인트를 통해 테이블에 액세스하려면 Iceberg 호환 클라이언트에서 카탈로그를 초기화해야 합니다. 이 초기화를 수행하려면 sigv4 속성, 엔드포인트 URI 및 웨어하우스 위치를 포함한 사용자 지정 속성을 지정해야 합니다. 다음과 같이 이러한 속성을 지정합니다.

- Sigv4 속성 - Sigv4를 활성화해야 하며 서명 이름은 glue입니다.
- 웨어하우스 위치 - 테이블 버킷으로, *<accountid>*:s3tablescatalog/*<table-bucket-name>* 형식으로 지정됩니다.
- 엔드포인트 URI - 리전별 엔드포인트는 AWS Glue 서비스 엔드포인트 참조 가이드를 참조하세요.

다음 예제에서는 pyIceberg 카탈로그를 초기화하는 방법을 보여줍니다.

```
rest_catalog = load_catalog(
        s3tablescatalog,
**{
"type": "rest",
"warehouse": "<accountid>:s3tablescatalog/<table-bucket-name>",
"uri": "https://glue.<region>.amazonaws.com/iceberg",
"rest.sigv4-enabled": "true",
"rest.signing-name": "glue",
"rest.signing-region": region
        }
```

```
)
```

AWS Glue Iceberg REST 엔드포인트 구현에 대한 자세한 내용은 AWS Glue 사용 설명서의 AWS GlueIceberg REST 엔드포인트를 사용하여 Data Catalog에 연결을 참조하세요.

Amazon S3 Tables Iceberg REST 엔드포인트를 사용하여 테이블에 액세스

Iceberg REST 클라이언트를 Amazon S3 Tables Iceberg REST 엔드포인트에 연결하고 S3 테이블 버킷에서 테이블을 생성, 업데이트 또는 쿼리하는 REST API 직접 호출을 할 수 있습니다. 엔드포인트는 Apache Iceberg RESTCatalog Open API 사양에 지정된 표준화된 Iceberg REST API 세트를 구현합니다. 엔드포인트는 Iceberg REST API 작업을 해당 S3 Tables 작업으로 변환하여 작동합니다.

> **Note**
>
> Amazon S3 Tables Iceberg REST 엔드포인트를 사용하여 AWS Partner Network(APN) 카탈로그 구현 또는 사용자 지정 카탈로그 구현의 테이블에 액세스할 수 있습니다. 또한 단일 테이블 버킷에 대한 기본 읽기/쓰기 액세스 권한만 필요한 경우에도 사용할 수 있습니다. 다른 액세스 시나리오의 경우 AWS Glue Iceberg REST 엔드포인트를 사용하여 통합 테이블 관리, 중앙 집중식 거버넌스 및 세분화된 액세스 제어를 제공하는 테이블에 연결하는 것이 좋습니다. 자세한 내용은 AWS GlueIceberg REST 엔드포인트를 사용하여 Amazon S3 Tables에 액세스 단원을 참조하세요.

엔드포인트 구성

서비스 엔드포인트를 사용하여 Amazon S3 Tables Iceberg REST 엔드포인트에 연결합니다. S3 Tables Iceberg REST 엔드포인트의 형식은 다음과 같습니다.

```
https://s3tables.<REGION>.amazonaws.com/iceberg
```

리전별 엔드포인트는 S3 Tables AWS 리전 및 엔드포인트 섹션을 참조하세요.

카탈로그 구성 속성

Iceberg 클라이언트를 사용하여 분석 엔진을 서비스 엔드포인트에 연결하는 경우 카탈로그를 초기화할 때 다음 구성 속성을 지정해야 합니다. ## ### #을 리전 및 테이블 버킷의 실제 정보로 바꿉니다.

- 리전별 엔드포인트를 엔드포인트 URI로 사용: `https://s3tables.<REGION>.amazonaws.com/iceberg`

- 테이블 버킷 ARN을 웨어하우스 위치로 사용: `arn:aws:s3tables:<region>:<accountID>:bucket/<bucketname>`
- 인증을 위한 Sigv4 속성. 서비스 엔드포인트 요청의 SigV4 서명 이름은 `s3tables`입니다.

다음 예제에서는 Amazon S3 Tables Iceberg REST 엔드포인트를 사용하도록 여러 클라이언트를 구성하는 방법을 보여줍니다.

PyIceberg

PyIceberg 에서 Amazon S3 TablesIceberg REST 엔드포인트를 사용하려면 다음 애플리케이션 구성 속성을 지정합니다.

```
rest_catalog = load_catalog(
  catalog_name,
  **{
    "type": "rest",
    "warehouse":"arn:aws:s3tables:<Region>:<accountID>:bucket/<bucketname>",
    "uri": "https://s3tables.<Region>.amazonaws.com/iceberg",
    "rest.sigv4-enabled": "true",
    "rest.signing-name": "s3tables",
    "rest.signing-region": "<Region>"
  }
)
```

Apache Spark

Spark에서 Amazon S3 Tables Iceberg REST 엔드포인트를 사용하려면 ## ### #을 리전 및 테이블 버킷의 정보로 바꾸어 다음 애플리케이션 구성 속성을 지정합니다.

```
spark-shell \
  --packages "org.apache.iceberg:iceberg-spark-
runtime-3.5_2.12:1.4.1,software.amazon.awssdk:bundle:2.20.160,software.amazon.awssdk:url-
connection-client:2.20.160" \
  --master "local[*]" \
  --conf
 "spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions"
 \
  --conf "spark.sql.defaultCatalog=spark_catalog" \
   --conf "spark.sql.catalog.spark_catalog=org.apache.iceberg.spark.SparkCatalog" \
  --conf "spark.sql.catalog.spark_catalog.type=rest" \
```

```
  --conf "spark.sql.catalog.spark_catalog.uri=https://
s3tables.<Region>.amazonaws.com/iceberg" \
  --conf
 "spark.sql.catalog.spark_catalog.warehouse=arn:aws:s3tables:<Region>:<accountID>:bucket/
<bucketname>" \
  --conf "spark.sql.catalog.spark_catalog.rest.sigv4-enabled=true" \
  --conf "spark.sql.catalog.spark_catalog.rest.signing-name=s3tables" \
  --conf "spark.sql.catalog.spark_catalog.rest.signing-region=<Region>" \
  --conf "spark.sql.catalog.spark_catalog.io-
impl=org.apache.iceberg.aws.s3.S3FileIO" \
  --conf
 "spark.hadoop.fs.s3a.aws.credentials.provider=org.apache.hadoop.fs.s3a.SimpleAWSCredentialP
 \
  --conf "spark.sql.catalog.spark_catalog.rest-metrics-reporting-enabled=false"
```

엔드포인트에 대한 액세스 인증 및 권한 부여

S3 Tables 서비스 엔드포인트에 대한 API 요청은 AWS Signature Version 4(SigV4)를 사용하여 인증됩니다. AWS SigV4에 대한 자세한 내용은 API 요청용 AWS Signature Version 4 섹션을 참조하세요.

Amazon S3 Tables Iceberg REST 엔드포인트 요청의 SigV4 서명 이름은 `s3tables`입니다.

Amazon S3 Tables Iceberg REST 엔드포인트에 대한 요청은 REST API 작업에 해당하는 `s3tables` IAM 작업을 사용하여 승인됩니다. 이러한 권한은 IAM 자격 증명 기반 정책 또는 테이블 및 테이블 버킷에 연결된 리소스 기반 정책에서 정의할 수 있습니다. 자세한 내용은 S3 Tables의 액세스 관리 단원을 참조하십시오.

AWS CloudTrail을 사용하여 REST 엔드포인트를 통해 테이블에 대한 요청을 추적할 수 있습니다. 요청은 해당 S3 IAM 작업으로 기록됩니다. 예를 들어 `LoadTable` API는 `GetTableMetadataLocation` 작업에 대한 관리 이벤트와 `GetTableData` 작업에 대한 데이터 이벤트를 생성합니다. 자세한 내용은 S3 Tables에서 AWS CloudTrail을 사용하여 로깅 단원을 참조하십시오.

접두사 및 경로 파라미터

Iceberg REST 카탈로그 API는 요청 URL에 자유 형식 접두사를 사용합니다. 예를 들어 `ListNamespaces` API 직접 호출은 `GET/v1/{prefix}/namespaces` URL 형식을 사용합니다. S3 테이블의 경우 REST 경로 `{prefix}`는 항상 URL로 인코딩된 테이블 버킷 ARN입니다.

예를 들어 다음 테이블 버킷 ARN `arn:aws:s3tables:us-east-1:111122223333:bucket/bucketname`의 경우 접두사는 `arn%3Aaws%3As3tables%3Aus-east-1%3A111122223333%3Abucket%2Fbucketname`입니다.

네임스페이스 경로 파라미터

Iceberg REST 카탈로그 API 경로의 네임스페이스는 여러 수준을 가질 수 있습니다. 그러나 S3 Tables는 단일 수준 네임스페이스만 지원합니다. 다중 수준 카탈로그 계층 구조의 네임스페이스에 액세스하려면 네임스페이스 위의 다중 수준 카탈로그에 연결하여 네임스페이스를 참조하면 됩니다. 이렇게 하면 다중 수준 네임스페이스를 사용하는 것에 비해 호환성 문제 없이 `catalog.namespace.table`의 3부 표기법을 지원하는 모든 쿼리 엔진이 S3 Tables의 카탈로그 계층 구조의 객체에 액세스할 수 있습니다.

지원되는 Iceberg REST API 작업

다음 표에는 지원되는 Iceberg REST API와 이러한 API가 S3 Tables 작업에 대응하는 방식이 나와 있습니다.

Iceberg REST 작업	REST 경로	S3 Tables IAM 작업	CloudTrail EventName
`getConfig`	`GET /v1/config`	`s3tables:GetTableBucket`	`s3tables:GetTableBucket`
`listNamespaces`	`GET /v1/{prefix}/namespaces`	`s3tables:ListNamespaces`	`s3tables:ListNamespaces`
`createNamespace`	`POST /v1/{prefix}/namespaces`	`s3tables:CreateNamespace`	`s3tables:CreateNamespace`
`loadNamespaceMetadata`	`GET /v1/{prefix}/namespaces/{namespace}`	`s3tables:GetNamespace`	`s3tables:GetNamespace`
`dropNamespace`	`DELETE /v1/{prefix}/names`	`s3tables:DeleteNamespace`	`s3tables:DeleteNamespace`

Iceberg REST 작업	REST 경로	S3 Tables IAM 작업	CloudTrail EventName
	paces/{na mespace}		
listTables	GET /v1/{pref ix}/names paces/{na mespace}/ tables	s3tables: ListTables	s3tables: ListTables
createTable	POST /v1/{pref ix}/names paces/{na mespace}/ tables	s3tables: CreateTab le , s3tables: PutTableData	s3tables: CreateTab le , s3tables: PutObject
loadTable	GET /v1/{pref ix}/names paces/{na mespace}/ tables/{table}	s3tables: GetTableM etadataLo cation , s3tables: GetTableData	s3tables: GetTableM etadataLo cation , s3tables: GetObject
updateTable	POST /v1/{pref ix}/names paces/{na mespace}/ tables/{table}	s3tables: UpdateTab leMetadat aLocation , s3tables: PutTableD ata , s3tables: GetTableData	s3tables: UpdateTab leMetadat aLocation , s3tables: PutObject , s3tables: GetObject

Iceberg REST 작업	REST 경로	S3 Tables IAM 작업	CloudTrail EventName
dropTable	DELETE /v1/{prefix}/namespaces/{namespace}/tables/{table}	s3tables:DeleteTable	s3tables:DeleteTable
renameTable	POST /v1/{prefix}/tables/rename	s3tables:RenameTable	s3tables:RenameTable
tableExists	HEAD /v1/{prefix}/namespaces/{namespace}/tables/{table}	s3tables:GetTable	s3tables:GetTable
namespaceExists	HEAD /v1/{prefix}/namespaces/{namespace}	s3tables:GetNamespace	s3tables:GetNamespace

고려 사항 및 제한 사항

다음은 Amazon S3 Tables Iceberg REST 엔드포인트 사용 시 고려 사항 및 제한 사항입니다.

고려 사항

- CreateTable API 동작 - 이 작업에서는 `stage-create` 옵션이 지원되지 않으며 `400 Bad Request` 오류가 발생합니다. 즉, `CREATE TABLE AS SELECT`(CTAS)를 사용하여 쿼리 결과에서 테이블을 생성할 수 없습니다.
- DeleteTable API 동작 - 제거가 활성화된 테이블만 삭제할 수 있습니다. `purge=false`가 포함된 테이블을 삭제하는 것은 지원되지 않으며 `400 Bad Request` 오류가 발생합니다. Spark의 일부 버전은 `DROP TABLE PURGE` 명령을 실행할 때도 항상 이 플래그를 false로 설정합니다. `DROP TABLE PURGE`를 사용하거나 S3 Tables DeleteTable 작업을 사용하여 테이블을 삭제할 수 있습니다.

- 엔드포인트는 표준 테이블 메타데이터 작업만 지원합니다. 스냅샷 관리 및 압축과 같은 테이블 유지 관리의 경우 S3 Tables 유지 관리 API 작업을 사용합니다. 자세한 내용은 S3 Tables 유지 관리 단원을 참조하십시오.

제한 사항

- 다중 수준 네임스페이스는 지원되지 않습니다.
- OAuth 기반의 인증은 지원되지 않습니다.
- 네임스페이스에는 `owner` 속성만 지원됩니다.
- Apache Iceberg REST Open API 사양에 정의된 보기 관련 API는 지원되지 않습니다.
- `metadata.json` 파일이 5MB를 초과하는 테이블에서 작업을 실행하는 것은 지원되지 않으며 `400 Bad Request` 오류가 반환됩니다. `metadata.json` 파일 크기를 제어하려면 테이블 유지 관리 작업을 사용합니다. 자세한 내용은 S3 Tables 유지 관리 단원을 참조하십시오.

Amazon S3 Tables Catalog for Apache Iceberg를 사용하여 Amazon S3 Tables에 액세스

Amazon S3 Tables Catalog for Apache Iceberg 클라이언트 카탈로그를 사용하여 Apache Spark 같은 오픈 소스 쿼리 엔진에서 S3 Tables에 액세스할 수 있습니다. Amazon S3 Tables Catalog for Apache Iceberg는 AWS Labs에서 호스팅하는 오픈 소스 라이브러리입니다. 쿼리 엔진의 Apache Iceberg 작업(예: 테이블 검색, 메타데이터 업데이트, 테이블 추가 또는 제거)을 S3 Tables API 작업으로 변환하여 작동합니다.

Amazon S3 Tables Catalog for Apache Iceberg는 `s3-tables-catalog-for-iceberg.jar`이라는 Maven JAR로 배포됩니다. AWS Labs GitHub 리포지토리에서 클라이언트 카탈로그 JAR을 빌드하거나 Maven에서 다운로드할 수 있습니다. 테이블에 연결할 경우 Apache Iceberg에 대한 Spark 세션을 초기화할 때 클라이언트 카탈로그 JAR이 종속성으로 사용됩니다.

Apache Spark와 함께 Amazon S3 Tables Catalog for Apache Iceberg 사용

Spark 세션을 초기화할 때 Amazon S3 Tables Catalog for Apache Iceberg 클라이언트 카탈로그를 사용하여 오픈 소스 애플리케이션의 테이블에 연결할 수 있습니다. 세션 구성에서 Iceberg 및 Amazon S3 종속성을 지정하고 테이블 버킷을 메타데이터 웨어하우스로 사용하는 사용자 지정 카탈로그를 생성합니다.

사전 조건

- 테이블 버킷 및 S3 Tables 작업에 대한 액세스 권한이 있는 IAM ID. 자세한 내용은 S3 Tables의 액세스 관리 섹션을 참조하세요.

Amazon S3 Tables Catalog for Apache Iceberg를 사용하여 Spark 세션을 초기화하려면

- 다음 명령을 사용하여 Spark를 초기화합니다. 명령을 사용하려면 Amazon S3 Tables Catalog for Apache Iceberg *## ##*를 AWS Labs GitHub 리포지토리의 최신 버전으로 바꾸고 *### ## ARN*을 실제 테이블 버킷 ARN으로 바꿉니다.

```
spark-shell \
--packages org.apache.iceberg:iceberg-spark-
runtime-3.5_2.12:1.6.1,software.amazon.s3tables:s3-tables-catalog-for-iceberg-
runtime:0.1.4 \
--conf spark.sql.catalog.s3tablesbucket=org.apache.iceberg.spark.SparkCatalog \
--conf spark.sql.catalog.s3tablesbucket.catalog-
impl=software.amazon.s3tables.iceberg.S3TablesCatalog \
--conf spark.sql.catalog.s3tablesbucket.warehouse=arn:aws:s3tables:us-
east-1:111122223333:bucket/amzn-s3-demo-table-bucket \
--conf
 spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
```

Spark SQL을 사용하여 S3 테이블 쿼리

Spark를 사용하면 S3 테이블에서 DQL, DML 및 DDL 작업을 실행할 수 있습니다. 테이블을 쿼리할 때 이 패턴에 따르는 세션 카탈로그 이름을 포함하여 정규화된 테이블 이름을 사용합니다.

CatalogName.*NamespaceName*.*TableName*

다음 예시 쿼리는 S3 테이블과 상호 작용할 수 있는 몇 가지 방법을 보여줍니다. 쿼리 엔진에서 이러한 예시 쿼리를 사용하려면 *### ## ## ###* 값을 실제 값으로 바꾸세요.

Spark를 사용하여 테이블을 쿼리하는 방법

- 네임스페이스 생성

```
spark.sql(" CREATE NAMESPACE IF NOT EXISTS s3tablesbucket.my_namespace")
```

- 테이블 생성

```
spark.sql(" CREATE TABLE IF NOT EXISTS s3tablesbucket.my_namespace.`my_table`
( id INT, name STRING, value INT ) USING iceberg ")
```

- 테이블 쿼리

```
spark.sql(" SELECT * FROM s3tablesbucket.my_namespace.`my_table` ").show()
```

- 테이블에 데이터 삽입

```
spark.sql(
"""
    INSERT INTO s3tablesbucket.my_namespace.my_table
    VALUES
        (1, 'ABC', 100),
        (2, 'XYZ', 200)
""")
```

- 테이블에 기존 데이터 파일 로드

1. Spark로 데이터를 읽습니다.

```
val data_file_location = "Path such as S3 URI to data file"
val data_file = spark.read.parquet(data_file_location)
```

2. Iceberg 테이블에 데이터를 기록합니다.

```
data_file.writeTo("s3tablesbucket.my_namespace.my_table").using("Iceberg").tableProperty
 ("format-version", "2").createOrReplace()
```

Athena를 사용하여 Amazon S3 테이블 쿼리

Amazon Athena는 표준 SQL을 사용해 Amazon S3에서 직접 데이터를 분석하는 데 사용할 수 있는 대화식 쿼리 서비스입니다. 자세한 내용은 Amazon Athena 사용 설명서에서 Amazon Athena란 무엇입니까?를 참조하세요.

테이블 버킷을 AWS 분석 서비스와 통합한 후 Athena를 사용하여 S3 Tables에서 데이터 정의 언어(DDL), 데이터 조작 언어(DML) 및 데이터 쿼리 언어(DQL) 쿼리를 실행할 수 있습니다. 테이블 버킷의 테이블을 쿼리하는 방법에 대한 자세한 내용은 Amazon Athena 사용 설명서의 S3 Tables 버킷 카탈로그 등록을 참조하세요.

Amazon S3 콘솔을 통해 Athena에서 쿼리를 실행할 수도 있습니다.

S3 콘솔 및 Amazon Athena 사용

다음 절차에서는 Amazon S3 콘솔을 사용하여 Athena 쿼리 편집기에 액세스하여 Amazon Athena로 테이블을 쿼리할 수 있습니다.

> **Note**
>
> 다음 단계를 수행하기 전에 테이블 버킷을 이 리전의 AWS 분석 서비스와 통합했는지 확인합니다. 자세한 내용은 the section called “AWS 분석 서비스에서 S3 Tables 사용” 섹션을 참조하세요.

테이블을 쿼리하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 테이블 버킷을 선택합니다.
3. 테이블 버킷 페이지에서 쿼리할 테이블이 포함된 버킷을 선택합니다.
4. 버킷 세부 정보 페이지에서 쿼리하려는 테이블 이름 옆에 있는 옵션 버튼을 선택합니다.
5. Athena를 사용하여 테이블 쿼리를 선택합니다.
6. Amazon Athena 콘솔이 열리고 Athena 쿼리 편집기가 샘플 `SELECT` 쿼리가 로드된 상태로 나타납니다. 사용 사례에 맞게 이 쿼리를 수정합니다.

   쿼리 편집기에서 카탈로그 필드는 s3tablescatalog/로 채워지고 테이블 버킷의 이름, 예를 들어 s3tablescatalog/***amzn-s3-demo-bucket***로 채워야 합니다. 데이터베이스 필드는 테이블이 저장된 네임스페이스로 채워져야 합니다.

   > **Note**
   >
   > 카탈로그 및 데이터베이스 필드에 이러한 값이 표시되지 않는 경우 테이블 버킷을 이 리전의 AWS 분석 서비스와 통합했는지 확인합니다. 자세한 내용은 the section called “AWS 분석 서비스에서 S3 Tables 사용” 섹션을 참조하세요.

7. 그런 다음 실행(Run)을 선택하여 쿼리를 실행합니다.

> **Note**
>
> - Athena에서 쿼리를 실행하려고 할 때 “쿼리를 실행할 권한이 충분하지 않습니다. 위탁자는 지정된 리소스에 대한 권한이 없습니다." 오류가 표시되면 테이블에서 필요한 Lake Formation 권한이 부여되어야 합니다. 자세한 내용은 the section called “테이블 또는 데이터베이스에 대한 권한 부여” 섹션을 참조하세요.
> - 쿼리를 실행하려고 할 때 "Iceberg가 요청된 리소스에 액세스할 수 없음" 오류가 표시되면 AWS Lake Formation 콘솔로 이동하여 생성한 테이블 버킷 카탈로그 및 데이터베이스(네임스페이스)에 대한 권한을 자신에게 부여했는지 확인합니다. 이러한 권한을 부여할 때 테이블을 지정하지 마십시오. 자세한 내용은 the section called “테이블 또는 데이터베이스에 대한 권한 부여” 섹션을 참조하세요.
> - Athena에서 SELECT 쿼리를 실행할 때 다음 오류 메시지가 표시되는 경우 테이블 이름이나 테이블 정의의 열 이름에 대문자가 있기 때문입니다. "GENERIC_INTERNAL_ERROR: Get table request failed: com.amazonaws.services.glue.model.ValidationException: Unsupported Federation Resource - Invalid table or column names." 테이블 및 열 이름이 모두 소문자인지 확인합니다.

Amazon Redshift를 사용하여 Amazon S3 테이블에 액세스

Amazon Redshift는 속도가 빠른 페타바이트 규모의 완전 관리형 데이터 웨어하우스 서비스로, 간편하고 비용 효율적으로 모든 데이터를 기존 비즈니스 인텔리전스 도구를 사용하여 효율적으로 분석할 수 있게 해줍니다. Amazon Redshift Serverless를 사용하면 프로비저닝된 데이터 웨어하우스를 구성하지 않아도 데이터를 액세스하고 분석할 수 있습니다. 자세한 내용은 Amazon Redshift 시작 안내서에서 서버리스 데이터 웨어하우스 시작하기를 참조하세요.

Amazon Redshift를 사용하여 Amazon S3 테이블 쿼리

사전 조건

- 테이블 버킷을 AWS 분석 서비스와 통합합니다.
  - 네임스페이스 생성.
  - 테이블 생성.
- 테이블 리소스에 대한 Lake Formation 권한 부여.

사전 조건을 충족한 후 Amazon Redshift를 사용하여 다음 방법 중 하나로 테이블을 쿼리할 수 있습니다.

- Amazon Redshift Query Editor V2 사용
- SQL 클라이언트 도구를 사용하여 Amazon Redshift 데이터 웨어하우스에 연결
- Amazon Redshift Data API 사용

Amazon EMR을 사용하여 Amazon S3 테이블에 액세스

Amazon EMR(이전의 Amazon Elastic MapReduce)은 AWS에서 Apache Hadoop 및 Apache Spark와 같은 빅 데이터 프레임워크 실행을 간소화하여 방대한 양의 데이터를 처리하고 분석하는 관리형 클러스터 플랫폼입니다. 이러한 프레임워크와 함께 관련 오픈 소스 프로젝트를 사용하여 분석용 데이터와 비즈니스 인텔리전스 워크로드를 처리할 수 있습니다. 또한 Amazon EMR을 사용하면 대량의 데이터를 다른 AWS 데이터 저장소 및 데이터베이스 간에 변환하고 이동할 수 있습니다.

Amazon EMR의 Apache Iceberg 클러스터를 사용하여 Spark 세션의 테이블 버킷에 연결하여 S3 테이블로 작업할 수 있습니다. Amazon EMR의 테이블 버킷에 연결하려면 AWS Glue Data Catalog를 통해 AWS 분석 서비스 통합을 사용하거나 오픈 소스 Amazon S3 Tables Catalog for Apache Iceberg 클라이언트 카탈로그를 사용할 수 있습니다.

> **Note**
>
> S3 Tables은 Amazon EMR 버전 7.5 이상에서 지원됩니다.

Amazon EMR Iceberg 클러스터에서 Spark를 사용하여 S3 테이블 버킷에 연결

이 절차에서는 Apache Iceberg에 대해 구성된 Amazon EMR 클러스터를 설정한 다음 테이블 버킷에 연결하는 Spark 세션을 시작합니다. AWS Glue를 통한 AWS 분석 서비스 통합을 사용하여 이를 설정하거나 오픈 소스 Amazon S3 Tables Catalog for Apache Iceberg 클라이언트 카탈로그를 사용할 수 있습니다. 클라이언트 카탈로그에 대한 자세한 내용은 Amazon S3 Tables Iceberg REST 엔드포인트를 사용하여 테이블에 액세스 섹션을 참조하세요.

다음 옵션 중에서 Amazon EMR에서 테이블을 사용하는 방법을 선택합니다.

Amazon S3 Tables Catalog for Apache Iceberg

Amazon S3 Tables Catalog for Apache Iceberg를 사용하여 Amazon EMR에서 Spark로 테이블을 쿼리하려면 다음 사전 조건을 충족해야 합니다.

사전 조건

- Amazon EMR에 사용하는 IAM 역할에 AmazonS3TablesFullAccess 정책을 연결합니다.

Spark를 사용하여 테이블을 쿼리하도록 Amazon EMR 클러스터를 설정하는 방법

1. 다음 구성을 사용하여 클러스터를 생성합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws emr create-cluster --release-label emr-7.5.0 \
--applications Name=Spark \
--configurations file://configurations.json \
--region us-east-1 \
--name My_Spark_Iceberg_Cluster \
--log-uri s3://amzn-s3-demo-bucket/ \
--instance-type m5.xlarge \
--instance-count 2 \
--service-role EMR_DefaultRole \
--ec2-attributes \

InstanceProfile=EMR_EC2_DefaultRole,SubnetId=subnet-1234567890abcdef0,KeyName=my-key-pair
```

configurations.json:

```
[{
"Classification":"iceberg-defaults",
"Properties":{"iceberg.enabled":"true"}
}]
```

2. SSH를 사용하여 Spark 프라이머리 노드에 연결합니다.
3. 테이블 버킷에 연결하는 Iceberg에 대한 Spark 세션을 초기화하려면 다음 명령을 입력합니다. *user input placeholders*를 테이블 버킷 ARN으로 바꿉니다.

```
spark-shell \
--packages software.amazon.s3tables:s3-tables-catalog-for-iceberg-runtime:0.1.3
 \
--conf spark.sql.catalog.s3tablesbucket=org.apache.iceberg.spark.SparkCatalog \
--conf spark.sql.catalog.s3tablesbucket.catalog-
impl=software.amazon.s3tables.iceberg.S3TablesCatalog \
```

```
--conf spark.sql.catalog.s3tablesbucket.warehouse=arn:aws:s3tables:us-
east-1:111122223333:bucket/amzn-s3-demo-bucket1 \
--conf spark.sql.defaultCatalog=s3tablesbucket \
--conf
 spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
```

4. Spark SQL을 사용하여 테이블을 쿼리합니다. 예시 쿼리는 the section called “Spark SQL을 사용하여 S3 테이블 쿼리” 섹션을 참조하세요.

AWS analytics services integration

AWS 분석 서비스 통합을 사용하여 Amazon EMR에서 Spark를 사용하여 테이블을 쿼리하려면 다음 사전 조건을 충족해야 합니다.

사전 조건

- 테이블 버킷을 AWS 분석 서비스와 통합합니다.
- Amazon EMR에 대한 기본 서비스 역할(EMR_DefaultRole_V2)을 생성합니다. 자세한 내용은 Service role for Amazon EMR (EMR role)을 참조하세요.
- Amazon EMR에 대한 Amazon EC2 인스턴스 프로파일(EMR_EC2_DefaultRole)을 생성합니다. 자세한 내용은 Service role for cluster EC2 instances (EC2 instance profile)를 참조하세요.
  - AmazonS3TablesFullAccess 정책을 EMR_EC2_DefaultRole에 연결합니다.

Spark를 사용하여 테이블을 쿼리하도록 Amazon EMR 클러스터를 설정하는 방법

1. 다음 구성을 사용하여 클러스터를 생성합니다. 이 예제를 사용하려면 *user input placeholder* 값을 사용자의 정보로 대체합니다.

```
aws emr create-cluster --release-label emr-7.5.0 \
--applications Name=Spark \
--configurations file://configurations.json \
--region us-east-1 \
--name My_Spark_Iceberg_Cluster \
--log-uri s3://amzn-s3-demo-bucket/ \
--instance-type m5.xlarge \
--instance-count 2 \
--service-role EMR_DefaultRole \
--ec2-attributes \
```

```
InstanceProfile=EMR_EC2_DefaultRole,SubnetId=subnet-1234567890abcdef0,KeyName=my-key-pair
```

configurations.json:

```
[{
"Classification":"iceberg-defaults",
"Properties":{"iceberg.enabled":"true"}
}]
```

2. SSH를 사용하여 Spark 프라이머리 노드에 연결합니다.
3. 다음 명령을 입력하여 테이블에 연결하는 Iceberg에 대한 Spark 세션을 초기화합니다. *user input placeholders*를 사용자의 정보로 대체합니다.

```
spark-shell \
--conf
 spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
 \
--conf spark.sql.defaultCatalog=s3tables \
--conf spark.sql.catalog.s3tables=org.apache.iceberg.spark.SparkCatalog \
--conf spark.sql.catalog.s3tables.catalog-
impl=org.apache.iceberg.aws.glue.GlueCatalog \
--conf spark.sql.catalog.s3tables.client.region=us-east-1 \
--conf spark.sql.catalog.s3tables.glue.id=111122223333
```

4. Spark SQL을 사용하여 테이블을 쿼리합니다. 예시 쿼리는 the section called “Spark SQL을 사용하여 S3 테이블 쿼리” 섹션을 참조하세요.

> **Note**
>
> Amazon EMR에서 `DROP TABLE PURGE` 명령을 사용하는 경우:
>
> - Amazon EMR 버전 7.5
>
> Spark 구성 `spark.sql.catalog.`*your-catalog-name*`.cache-enabled`를 `false`로 설정합니다. 이 구성이 `true`로 설정된 경우 테이블 캐시가 활성화되지 않도록 새 세션 또는 애플리케이션에서 명령을 실행합니다.
> - 7.5보다 높은 Amazon EMR 버전

DROP TABLE는 지원되지 않습니다. S3 Tables DeleteTable REST API를 사용하여 테이블을 삭제할 수 있습니다.

QuickSight를 사용하여 테이블 데이터 시각화

QuickSight는 데이터를 사용하여 시각적 객체를 구축하고, 애드혹 분석을 수행하고, 사업과 관련된 인사이트를 빠르게 얻을 수 있는 신속한 비즈니스 분석 서비스입니다. QuickSight는 QuickSight Super-fast, Parallel, In-Memory, Calculation Engine(SPICE)을 사용하여 AWS 데이터 소스를 원활하게 검색하고, 조직이 수십만 명의 사용자로 규모를 조정할 수 있도록 하고, 빠르고 응답성이 뛰어난 쿼리 성능을 제공합니다. 자세한 내용은 QuickSight 사용 설명서의 QuickSight란 무엇인가요?를 참조하세요.

테이블 버킷을 AWS 분석 서비스와 통합한 후 SPICE 또는 쿼리 엔진의 직접 SQL 쿼리를 사용하여 테이블에서 데이터세트를 생성하고 QuickSight에서 작업할 수 있습니다. QuickSight는 Athena를 S3 테이블의 데이터 소스로 지원합니다.

QuickSight가 테이블에 액세스할 수 있도록 권한 구성

QuickSight에서 S3 테이블 데이터를 사용하기 전에 QuickSight 서비스 역할과 Amazon QuickSight 관리자 사용자에게 권한을 부여하고 액세스하려는 테이블에 대해 Lake Formation 권한을 부여해야 합니다.

QuickSight 서비스 역할에 권한 부여

계정에서 QuickSight를 처음 설정하면 AWS는 QuickSight가 Athena 또는 Amazon Redshift를 비롯한 다른 AWS 서비스의 데이터 소스에 액세스할 수 있도록 허용하는 서비스 역할을 생성합니다. 기본 역할 이름은 aws-quicksight-service-role-v0입니다.

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 역할을 선택하고 QuickSight 서비스 역할을 선택합니다. 기본 이름은 aws-quicksight-service-role-v0입니다.
3. 권한 추가를 선택한 후 인라인 정책 만들기를 선택합니다.
4. JSON을 선택하여 JSON 정책 편집기를 연 다음, 다음 정책을 추가합니다.

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

```
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "glue:GetCatalog",
      "Resource": "*"
    }
  ]
}
```

5. 다음을 선택하고 정책 이름을 입력한 후 정책 생성을 선택합니다.

QuickSight 관리자 사용자에 대한 권한을 구성하는 방법

1. 다음 AWS CLI 명령을 실행하여 QuickSight 관리자 사용자의 ARN을 찾습니다.

```
aws quicksight list-users --aws-account-id 111122223333 --namespace default --
region region
```

2. Lake Formation에 이 ARN에 대한 권한을 부여합니다. 자세한 내용은 테이블 리소스에 대한 Lake Formation 권한 부여을 참조하세요.

QuickSight에서 테이블 데이터 사용

Athena를 데이터 소스로 사용하여 테이블 데이터에 연결할 수 있습니다.

사전 조건

- 테이블 버킷을 AWS 분석 서비스와 통합합니다.
  - 네임스페이스 생성.
  - 테이블 생성.
  - QuickSight가 테이블에 액세스할 수 있도록 권한 구성.
- QuickSight에 가입합니다.

1. https://quicksight.aws.amazon.com/에서 QuickSight 계정에 로그인합니다.
2. 대시보드에서 새 분석을 선택합니다.
3. 새 데이터 세트를 선택합니다.
4. Athena를 선택합니다.
5. 데이터 소스의 이름을 입력한 후 데이터 소스 생성을 선택합니다.

6. 사용자 지정 SQL 사용을 선택합니다. 테이블 선택 창에서 테이블을 선택할 수 없습니다.
7. 시각화하려는 열을 캡처하는 Athena SQL 쿼리를 입력한 다음 쿼리 확인을 선택합니다. 예를 들어 모든 열을 선택하려면 다음 쿼리를 사용합니다.

```
SELECT * FROM "s3tablescatalog/table-bucket-name".namespace.table-name
```

8. 시각화를 선택하여 데이터를 분석하고 대시보드 구축을 시작합니다. 자세한 내용은 QuickSight에서 데이터 시각화 및 QuickSight의 대화형 대시보드 탐색을 참조하세요.

Amazon Data Firehose를 사용하여 테이블에 데이터 스트리밍

Amazon Data Firehose는 Amazon S3, Amazon Redshift, Amazon OpenSearch Service, Splunk, Apache Iceberg Tables 그리고 모든 사용자 지정 HTTP 엔드포인트 또는 서드 파티 서비스 제공업체가 소유 및 지원하는 HTTP 엔드포인트 등의 대상으로 실시간 스트리밍 데이터를 전송하는 완전관리형 서비스입니다. Amazon Data Firehose를 사용하면 애플리케이션을 쓰거나 리소스를 관리할 필요가 없습니다. 데이터 생산자가 데이터를 Firehose로 보내도록 구성하면 지정한 대상으로 데이터를 자동 전송합니다. 전송 전에 데이터를 변환하도록 Firehose를 구성할 수도 있습니다. Amazon Data Firehose에 대해 자세히 알아보려면 What is Amazon Data Firehose?를 참조하세요.

다음 단계를 완료하여 S3 테이블 버킷의 테이블로 Firehose 스트리밍을 설정합니다.

1. 테이블 버킷을 AWS 분석 서비스와 통합합니다.
2. S3 Tables로 데이터를 전송하도록 Firehose를 구성합니다. 이렇게 하려면 Firehose가 테이블에 액세스할 수 있도록 허용하는 AWS Identity and Access Management(IAM) 서비스 역할을 생성합니다.
3. Firehose 서비스 역할에 테이블 또는 테이블의 네임스페이스에 대한 명시적 권한을 부여합니다. 자세한 내용은 테이블 리소스에 대한 Lake Formation 권한 부여를 참조하세요.
4. 데이터를 테이블로 라우팅하는 Firehose 스트림을 생성합니다.

Firehose가 S3 테이블을 대상으로 사용하도록 역할 생성

Firehose에는 AWS Glue 테이블에 액세스하고 S3 테이블에 데이터를 쓸 수 있는 특정 권한이 있는 IAM 서비스 역할이 필요합니다. Firehose 스트림을 생성할 때 이 IAM 역할을 제공해야 합니다.

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 정책을 선택합니다.

3. 정책 생성을 선택하고 정책 편집기에서 JSON을 선택합니다.
4. 데이터 카탈로그의 모든 데이터베이스 및 테이블에 권한을 부여하는 다음 인라인 정책을 추가합니다. 원하는 경우 특정 테이블 및 데이터베이스에 대해서만 권한을 부여할 수 있습니다. 이 정책을 사용하려면 *user input placeholders*를 실제 정보로 대체합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3TableAccessViaGlueFederation",
      "Effect": "Allow",
      "Action": [
        "glue:GetTable",
        "glue:GetDatabase",
        "glue:UpdateTable"
      ],
      "Resource": [
        "arn:aws:glue:region:account-id:catalog/s3tablescatalog/*",
        "arn:aws:glue:region:account-id:catalog/s3tablescatalog",
        "arn:aws:glue:region:account-id:catalog",
        "arn:aws:glue:region:account-id:database/*",
        "arn:aws:glue:region:account-id:table/*/*"
      ]
    },
    {
      "Sid": "S3DeliveryErrorBucketPermission",
      "Effect": "Allow",
      "Action": [
        "s3:AbortMultipartUpload",
        "s3:GetBucketLocation",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::error delivery bucket",
        "arn:aws:s3:::error delivery bucket/*"
      ]
    },
    {
      "Sid": "RequiredWhenUsingKinesisDataStreamsAsSource",
      "Effect": "Allow",
```

```
      "Action": [
        "kinesis:DescribeStream",
        "kinesis:GetShardIterator",
        "kinesis:GetRecords",
        "kinesis:ListShards"
      ],
      "Resource": "arn:aws:kinesis:region:account-id:stream/stream-name"
    },
    {
      "Sid":
 "RequiredWhenDoingMetadataReadsANDDataAndMetadataWriteViaLakeformation",
      "Effect": "Allow",
      "Action": [
        "lakeformation:GetDataAccess"
      ],
      "Resource": "*"
    },
    {
      "Sid": "RequiredWhenUsingKMSEncryptionForS3ErrorBucketDelivery",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "arn:aws:kms:region:account-id:key/KMS-key-id"
      ],
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "s3.region.amazonaws.com"
        },
        "StringLike": {
          "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::error delivery bucket/
prefix*"
        }
      }
    },
    {
      "Sid": "LoggingInCloudWatch",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],
      "Resource": [
```

```
        "arn:aws:logs:region:account-id:log-group:log-group-name:log-stream:log-
stream-name"
      ]
    },
    {
      "Sid": "RequiredWhenAttachingLambdaToFirehose",
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction",
        "lambda:GetFunctionConfiguration"
      ],
      "Resource": [
        "arn:aws:lambda:region:account-id:function:function-name:function-version"
      ]
    }
  ]
}
```

이 정책에는 Kinesis Data Streams에 대한 액세스, Lambda 함수 간접 호출, AWS KMS 키에 대한 액세스를 허용하는 문이 있습니다. 이러한 리소스를 사용하지 않는 경우, 해당 문을 제거할 수 있습니다.

오류 로깅이 활성화되면 Firehose는 데이터 전송 오류도 CloudWatch 로그 그룹 및 스트림으로 보냅니다. 이를 위해서는 로그 그룹 및 로그 스트림 이름을 구성해야 합니다. 로그 그룹 및 로그 스트림 이름은 Monitor Amazon Data Firehose Using CloudWatch Logs를 참조하세요.

5. 정책을 생성한 후 AWS 서비스를 신뢰할 수 있는 엔터티 유형으로 설정하여 IAM 역할을 생성합니다.
6. 서비스 또는 사용 사례에서 Kinesis를 선택합니다. 사용 사례로 Kinesis Firehose를 선택합니다.
7. 다음을 선택하고 앞에서 만든 정책을 선택합니다.
8. 역할에 이름을 지정합니다. 역할 세부 정보를 검토하고 역할 생성을 선택합니다. 역할은 다음과 같은 신뢰 정책을 갖게 됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
```

```
            "Principal": {
                "Service": [
                    "firehose.amazonaws.com"
                ]
            }
        }
    ]
}
```

S3 테이블로 Firehose 스트림 생성

다음 절차에서는 콘솔을 사용하여 S3 테이블에 데이터를 전송하도록 Firehose 스트림을 생성하는 방법을 보여줍니다. S3로 Firehose 스트림을 설정하려면 다음 사전 조건을 충족해야 합니다.

사전 조건

- 테이블 버킷을 AWS 분석 서비스와 통합합니다.
  - 네임스페이스 생성.
  - 테이블 생성.
- Firehose가 S3 Tables에 액세스하기 위한 역할을 생성합니다.
- 테이블에 액세스하기 위해 생성한 Firehose 서비스 역할에 Lake Formation 권한을 부여합니다.

스트림을 구성할 때 Firehose에 라우팅 정보를 제공하려면 네임스페이스를 데이터베이스 이름으로 사용하고 해당 네임스페이스에 있는 테이블의 이름을 사용합니다. Firehose 스트림 구성의 고유 키 섹션에서 이러한 값을 사용하여 데이터를 단일 테이블로 라우팅할 수 있습니다. 이 값을 사용하여 JSON 쿼리 표현식을 사용하여 테이블로 라우팅할 수도 있습니다. 자세한 정보는 Route incoming records to a single Iceberg table을 참조하세요.

S3 테이블로 Firehose 스트림을 설정하는 방법(콘솔)

1. https://console.aws.amazon.com/firehose/에서 Firehose 콘솔을 엽니다.
2. Firehose 스트림 생성을 선택합니다.
3. 소스에서 다음 소스 중 하나를 선택합니다.
   - Amazon Kinesis Data Streams
   - Amazon MSK
   - Direct PUT

4. 대상에서 Apache Iceberg Tables를 선택합니다.
5. Firehose 스트림 이름을 입력합니다.
6. 소스 설정을 구성합니다.
7. 대상 설정에서 현재 계정을 선택하여 계정의 테이블로 스트리밍하거나 교차 계정을 선택하여 다른 계정의 테이블로 스트리밍합니다.
   - 현재 계정의 테이블의 경우 카탈로그 드롭다운에서 S3 Tables 카탈로그를 선택합니다.
   - 교차 계정의 테이블의 경우 다른 계정에서 스트리밍하려는 카탈로그의 카탈로그 ARN을 입력합니다.
8. 고유 키 구성, JSONQuery 표현식을 사용하거나 Lambda 함수에서 데이터베이스 및 테이블 이름을 구성합니다. 자세한 내용은 Amazon Data Firehose 개발자 안내서의 Route incoming records to a single Iceberg table 및 Route incoming records to different Iceberg tables을 참조하세요.
9. 백업 설정에서 S3 백업 버킷을 지정합니다.
10. 기존 IAM 역할의 경우 고급 설정에서 Firehose에 대해 생성한 IAM 역할을 선택합니다.
11. Firehose 스트림 생성을 선택합니다.

스트림에 대해 구성할 수 있는 다른 설정에 대한 자세한 내용은 Amazon Data Firehose 개발자 안내서의 Set up the Firehose stream을 참조하세요.

AWS Glue를 사용하여 Amazon S3 테이블에서 ETL 작업 실행

AWS Glue는 분석 사용자가 여러 소스의 데이터를 쉽게 검색, 준비, 이동, 통합할 수 있도록 하는 서버리스 데이터 통합 서비스입니다. AWS Glue 작업을 사용하여 데이터 레이크에 데이터를 로드하도록 추출, 전환, 적재(ETL) 파이프라인을 실행할 수 있습니다. AWS Glue에 대한 자세한 내용은 AWS Glue 개발자 안내서의 AWS Glue란 무엇입니까?를 참조하세요.

AWS Glue 작업은 소스 데이터에 연결하여 처리한 다음 데이터 대상에 작성하는 스크립트를 캡슐화합니다. 일반적으로 작업은 추출, 변환 및 로드(ETL) 스크립트를 실행합니다. 작업은 Apache Spark 런타임 환경용으로 설계된 스크립트를 실행할 수 있습니다. 작업 실행을 모니터링하여 완료 상태, 지속 시간, 시작 시간 같은 실행 시간 지표를 이해할 수 있습니다.

AWS Glue 작업을 사용하는 경우 AWS 분석 서비스와의 통합을 통해 테이블에 연결하여 S3 테이블의 데이터를 처리하거나 Amazon S3 Tables Iceberg REST 엔드포인트 또는 Amazon S3 Tables Catalog for Apache Iceberg를 사용하여 직접 연결할 수 있습니다. 이 안내서에서는 다음을 포함하여 S3 Tables에서 AWS Glue의 사용을 시작하기 위한 기본 단계를 다룹니다.

주제

- 사전 조건
- 테이블 버킷에 연결할 스크립트 생성
- 테이블을 쿼리하는 AWS Glue 작업 생성

> **Note**
>
> S3 Tables은 AWS Glue 버전 5.0 이상에서 지원됩니다.

사전 조건

AWS Glue 작업에서 테이블을 쿼리하려면 먼저 AWS Glue가 작업을 실행하는 데 사용할 수 있는 IAM 역할을 구성하고 작업을 실행할 때 AWS Glue가 액세스할 수 있는 S3 버킷에 Amazon S3 Tables Catalog for Apache Iceberg JAR을 업로드해야 합니다.

- 테이블 버킷을 AWS 분석 서비스와 통합합니다.
- AWS Glue용 IAM 역할을 생성합니다.
  - `AmazonS3TablesFullAccess` 관리형 정책을 역할에 연결합니다.
  - `AmazonS3FullAccess` 관리형 정책을 역할에 연결합니다.
- (선택 사항) Amazon S3 Tables Catalog for Apache Iceberg를 사용하는 경우 클라이언트 카탈로그 JAR을 다운로드하여 S3 버킷에 업로드해야 합니다.

  카탈로그 JAR 다운로드

  1. Maven Central에서 최신 버전을 확인하세요. 브라우저를 사용하거나 다음 명령을 사용하여 Maven Central에서 JAR을 다운로드할 수 있습니다. *## ##*를 최신 버전으로 바꿔야 합니다.

     ```
     wget https://repo1.maven.org/maven2/software/amazon/s3tables/s3-tables-catalog-for-iceberg-runtime/0.1.5/s3-tables-catalog-for-iceberg-runtime-0.1.5.jar
     ```

  2. AWS Glue IAM 역할이 액세스할 수 있는 S3 버킷에 다운로드한 JAR을 업로드합니다. 다음 AWS CLI 명령을 사용하여 JAR을 업로드할 수 있습니다. *## ##*를 최신 버전으로 바꾸고 *## ##*과 *##*를 실제 정보로 바꿔야 합니다.

     ```
     aws s3 cp s3-tables-catalog-for-iceberg-runtime-0.1.5.jar s3://amzn-s3-demo-bucket/jars/
     ```

테이블 버킷에 연결할 스크립트 생성

AWS Glue ETL 작업을 실행할 때 테이블 데이터에 액세스하려면 S3 테이블 버킷에 연결되는 Apache Iceberg에 대한 Spark 세션을 구성합니다. 기존 스크립트를 수정하여 테이블 버킷에 연결하거나 새 스크립트를 생성할 수 있습니다. AWS Glue 스크립트 생성에 대한 자세한 내용은 AWS Glue 개발자 안내서의 자습서: AWS Glue for Spark 스크립트 작성을 참조하세요.

다음 S3 Tables 액세스 방법 중 하나를 통해 테이블 버킷에 연결하도록 세션을 구성할 수 있습니다.

- AWS 분석 서비스와 S3 Tables 통합
- Amazon S3 Tables Iceberg REST 엔드포인트
- Amazon S3 Tables Catalog for Apache Iceberg

다음 액세스 방법 중에서 선택하여 설정 지침과 구성 예제를 살펴보세요.

AWS analytics services integration

AWS 분석 서비스 통합을 사용하여 AWS Glue에서 Spark로 테이블을 쿼리하려면 먼저 테이블 버킷을 AWS 분석 서비스와 통합해야 합니다.

작업의 Spark 세션을 통해 또는 대화형 세션의 AWS Glue Studio 매직을 사용하여 테이블 버킷에 대한 연결을 구성할 수 있습니다. 다음 예제를 사용하려면 *## ### #*을 실제 테이블 버킷에 대한 정보로 바꿉니다.

PySpark 스크립트 사용

PySpark 스크립트에서 다음 코드 스니펫을 사용하여 통합을 통해 테이블 버킷에 연결하도록 AWS Glue 작업을 구성합니다.

```
spark = SparkSession.builder.appName("SparkIcebergSQL") \
    .config("spark.sql.extensions",
 "org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions") \
    .config("spark.sql.defaultCatalog","s3tables") \
    .config("spark.sql.catalog.s3tables",
 "org.apache.iceberg.spark.SparkCatalog") \
    .config("spark.sql.catalog.s3tables.catalog-impl",
 "org.apache.iceberg.aws.glue.GlueCatalog") \
    .config("spark.sql.catalog.s3tables.glue.id",
 "111122223333:s3tablescatalog/amzn-s3-demo-table-bucket") \
    .config("spark.sql.catalog.s3tables.warehouse", "s3://amzn-s3-demo-table-
bucket/warehouse/") \
```

```
    .getOrCreate()
```

대화형 AWS Glue 세션 사용

AWS Glue 5.0으로 대화형 노트북 세션을 사용하는 경우 코드 실행 전에 셀에서 `%%configure` 매직을 사용하여 동일한 구성을 지정합니다.

```
%%configure
{"conf":
 "spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
 --conf spark.sql.defaultCatalog=s3tables --conf
 spark.sql.catalog.s3tables=org.apache.iceberg.spark.SparkCatalog --conf
 spark.sql.catalog.s3tables.catalog-impl=org.apache.iceberg.aws.glue.GlueCatalog
 --conf spark.sql.catalog.s3tables.glue.id=111122223333:s3tablescatalog/amzn-s3-
demo-table-bucket --conf spark.sql.catalog.s3tables.warehouse=s3://amzn-s3-demo-
table-bucket/warehouse/"}
```

Amazon S3 Tables Iceberg REST endpoint

작업의 Spark 세션을 통해 또는 대화형 세션의 AWS Glue Studio 매직을 사용하여 테이블 버킷에 대한 연결을 구성할 수 있습니다. 다음 예제를 사용하려면 *## ### #*을 실제 테이블 버킷에 대한 정보로 바꿉니다.

PySpark 스크립트 사용

PySpark 스크립트에서 다음 코드 스니펫을 사용하여 엔드포인트를 통해 테이블 버킷에 연결하도록 AWS Glue 작업을 구성합니다.

```
spark = SparkSession.builder.appName("glue-s3-tables-rest") \
    .config("spark.sql.extensions",
 "org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions") \
    .config("spark.sql.defaultCatalog", "s3_rest_catalog") \
    .config("spark.sql.catalog.s3_rest_catalog",
 "org.apache.iceberg.spark.SparkCatalog") \
    .config("spark.sql.catalog.s3_rest_catalog.type", "rest") \
    .config("spark.sql.catalog.s3_rest_catalog.uri", "https://
s3tables.Region.amazonaws.com/iceberg") \
    .config("spark.sql.catalog.s3_rest_catalog.warehouse",
 "arn:aws:s3tables:Region:111122223333:s3tablescatalog/amzn-s3-demo-table-
bucket") \
    .config("spark.sql.catalog.s3_rest_catalog.rest.sigv4-enabled", "true") \
```

```
    .config("spark.sql.catalog.s3_rest_catalog.rest.signing-name", "s3tables") \
    .config("spark.sql.catalog.s3_rest_catalog.rest.signing-region", "Region") \
    .config('spark.sql.catalog.s3_rest_catalog.io-
impl','org.apache.iceberg.aws.s3.S3FileIO') \
    .config('spark.sql.catalog.s3_rest_catalog.rest-metrics-reporting-
enabled','false') \
    .getOrCreate()
```

대화형 AWS Glue 세션 사용

AWS Glue 5.0으로 대화형 노트북 세션을 사용하는 경우 코드 실행 전에 셀에서 %%configure 매직을 사용하여 동일한 구성을 지정합니다. 자리 표시자 값을 테이블 버킷의 실제 정보로 바꿉니다.

```
%%configure
{"conf":
 "spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
 --conf spark.sql.defaultCatalog=s3_rest_catalog --conf
 spark.sql.catalog.s3_rest_catalog=org.apache.iceberg.spark.SparkCatalog
 --conf spark.sql.catalog.s3_rest_catalog.type=rest --conf
 spark.sql.catalog.s3_rest_catalog.uri=https://s3tables.Region.amazonaws.com/
iceberg --conf
 spark.sql.catalog.s3_rest_catalog.warehouse=arn:aws:s3tables:Region:111122223333:s3table
amzn-s3-demo-table-bucket --conf spark.sql.catalog.s3_rest_catalog.rest.sigv4-
enabled=true --conf spark.sql.catalog.s3_rest_catalog.rest.signing-name=s3tables
 --conf spark.sql.catalog.s3_rest_catalog.rest.signing-region=Region --conf
 spark.sql.catalog.s3_rest_catalog.io-impl=org.apache.iceberg.aws.s3.S3FileIO --
conf spark.sql.catalog.s3_rest_catalog.rest-metrics-reporting-enabled=false"}
```

Amazon S3 Tables Catalog for Apache Iceberg

Amazon S3 Tables Catalog for Apache Iceberg를 사용하여 테이블에 연결하려면 먼저 최신 카탈로그 jar을 다운로드하여 S3 버킷에 업로드해야 합니다. 그런 다음 작업을 생성할 때 클라이언트 카탈로그 JAR의 경로를 특수 파라미터로 추가합니다. AWS Glue의 작업 파라미터에 대한 자세한 내용은 AWS Glue 개발자 안내서의 AWS Glue 작업에 사용되는 특수 파라미터를 참조하세요.

작업의 Spark 세션을 통해 또는 대화형 세션의 AWS Glue Studio 매직을 사용하여 테이블 버킷에 대한 연결을 구성할 수 있습니다. 다음 예제를 사용하려면 ## ### #을 실제 테이블 버킷에 대한 정보로 바꿉니다.

PySpark 스크립트 사용

PySpark 스크립트에서 다음 코드 스니펫을 사용하여 JAR을 통해 테이블 버킷에 연결하도록 AWS Glue 작업을 구성합니다. 자리 표시자 값을 테이블 버킷의 실제 정보로 바꿉니다.

```
spark = SparkSession.builder.appName("glue-s3-tables") \
    .config("spark.sql.extensions",
 "org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions") \
    .config("spark.sql.defaultCatalog", "s3tablesbucket") \
    .config("spark.sql.catalog.s3tablesbucket",
 "org.apache.iceberg.spark.SparkCatalog") \
    .config("spark.sql.catalog.s3tablesbucket.catalog-impl",
 "software.amazon.s3tables.iceberg.S3TablesCatalog") \
    .config("spark.sql.catalog.s3tablesbucket.warehouse",
 "arn:aws:s3tables:Region:111122223333:bucket/amzn-s3-demo-table-bucket") \
    .getOrCreate()
```

대화형 AWS Glue 세션 사용

AWS Glue 5.0으로 대화형 노트북 세션을 사용하는 경우 코드 실행 전에 셀에서 `%%configure` 매직을 사용하여 동일한 구성을 지정합니다. 자리 표시자 값을 테이블 버킷의 실제 정보로 바꿉니다.

```
%%configure
{"conf":
 "spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
 --conf spark.sql.defaultCatalog=s3tablesbucket --conf
 spark.sql.catalog.s3tablesbucket=org.apache.iceberg.spark.SparkCatalog
 --conf spark.sql.catalog.s3tablesbucket.catalog-
impl=software.amazon.s3tables.iceberg.S3TablesCatalog --conf
 spark.sql.catalog.s3tablesbucket.warehouse=arn:aws:s3tables:Region:111122223333:bucket/
amzn-s3-demo-table-bucket", "extra-jars": "s3://amzn-s3-demo-bucket/jars/s3-
tables-catalog-for-iceberg-runtime-0.1.5.jar"}
```

샘플 스크립트

다음 예제 AWS Glue 스크립트는 PySpark 작업으로 S3 테이블 쿼리를 테스트하는 데 사용할 수 있습니다. 이러한 스크립트는 테이블 버킷에 연결한 다음 쿼리를 실행하여 새 네임스페이스를 생성하고, 샘플 테이블을 생성하고, 테이블에 데이터를 삽입하고, 테이블 데이터를 반환합니다. 스크립트를 사용하려면 *## ### #*을 실제 테이블 버킷에 대한 정보로 바꿉니다.

S3 Tables 액세스 방법에 따라 다음 스크립트 중에서 선택합니다.

S3 Tables integration with AWS analytics services

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("SparkIcebergSQL") \
    .config("spark.sql.extensions",
 "org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions") \
    .config("spark.sql.defaultCatalog","s3tables")
    .config("spark.sql.catalog.s3tables", "org.apache.iceberg.spark.SparkCatalog") \
    .config("spark.sql.catalog.s3tables.catalog-impl",
 "org.apache.iceberg.aws.glue.GlueCatalog") \
    .config("spark.sql.catalog.s3tables.glue.id",
 "111122223333:s3tablescatalog/amzn-s3-demo-table-bucket") \
    .config("spark.sql.catalog.s3tables.warehouse", "s3://amzn-s3-demo-table-bucket/
bucket/amzn-s3-demo-table-bucket") \
    .getOrCreate()

namespace = "new_namespace"
table = "new_table"

spark.sql("SHOW DATABASES").show()

spark.sql(f"DESCRIBE NAMESPACE {namespace}").show()

spark.sql(f"""
    CREATE TABLE IF NOT EXISTS {namespace}.{table} (
       id INT,
       name STRING,
       value INT
    )
""")

spark.sql(f"""
    INSERT INTO {namespace}.{table}
    VALUES
       (1, 'ABC', 100),
       (2, 'XYZ', 200)
""")

spark.sql(f"SELECT * FROM {namespace}.{table} LIMIT 10").show()
```

Amazon S3 Tables Iceberg REST endpoint

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("glue-s3-tables-rest") \
    .config("spark.sql.extensions",
 "org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions") \
    .config("spark.sql.defaultCatalog", "s3_rest_catalog") \
    .config("spark.sql.catalog.s3_rest_catalog",
 "org.apache.iceberg.spark.SparkCatalog") \
    .config("spark.sql.catalog.s3_rest_catalog.type", "rest") \
    .config("spark.sql.catalog.s3_rest_catalog.uri", "https://
s3tables.Region.amazonaws.com/iceberg") \
    .config("spark.sql.catalog.s3_rest_catalog.warehouse",
 "arn:aws:s3tables:Region:111122223333:bucket/amzn-s3-demo-table-bucket") \
    .config("spark.sql.catalog.s3_rest_catalog.rest.sigv4-enabled", "true") \
    .config("spark.sql.catalog.s3_rest_catalog.rest.signing-name", "s3tables") \
    .config("spark.sql.catalog.s3_rest_catalog.rest.signing-region", "Region") \
    .config('spark.sql.catalog.s3_rest_catalog.io-
impl','org.apache.iceberg.aws.s3.S3FileIO') \
    .config('spark.sql.catalog.s3_rest_catalog.rest-metrics-reporting-
enabled','false') \
    .getOrCreate()

namespace = "s3_tables_rest_namespace"
table = "new_table_s3_rest"

spark.sql("SHOW DATABASES").show()

spark.sql(f"DESCRIBE NAMESPACE {namespace}").show()

spark.sql(f"""
    CREATE TABLE IF NOT EXISTS {namespace}.{table} (
       id INT,
       name STRING,
       value INT
    )
""")

spark.sql(f"""
    INSERT INTO {namespace}.{table}
    VALUES
       (1, 'ABC', 100),
       (2, 'XYZ', 200)
```

```
""")

spark.sql(f"SELECT * FROM {namespace}.{table} LIMIT 10").show()
```

Amazon S3 Tables Catalog for Apache Iceberg

```
from pyspark.sql import SparkSession

#Spark session configurations
spark = SparkSession.builder.appName("glue-s3-tables") \
    .config("spark.sql.extensions",
 "org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions") \
    .config("spark.sql.defaultCatalog", "s3tablesbucket") \
    .config("spark.sql.catalog.s3tablesbucket",
 "org.apache.iceberg.spark.SparkCatalog") \
    .config("spark.sql.catalog.s3tablesbucket.catalog-impl",
 "software.amazon.s3tables.iceberg.S3TablesCatalog") \
    .config("spark.sql.catalog.s3tablesbucket.warehouse",
 "arn:aws:s3tables:Region:111122223333:bucket/amzn-s3-demo-table-bucket") \
    .getOrCreate()

#Script
namespace = "new_namespace"
table = "new_table"

spark.sql(f"CREATE NAMESPACE IF NOT EXISTS s3tablesbucket.{namespace}")
spark.sql(f"DESCRIBE NAMESPACE {namespace}").show()

spark.sql(f"""
    CREATE TABLE IF NOT EXISTS {namespace}.{table} (
       id INT,
       name STRING,
       value INT
    )
""")

spark.sql(f"""
    INSERT INTO {namespace}.{table}
    VALUES
       (1, 'ABC', 100),
       (2, 'XYZ', 200)
```

```
""")

spark.sql(f"SELECT * FROM {namespace}.{table} LIMIT 10").show()
```

테이블을 쿼리하는 AWS Glue 작업 생성

다음 절차에서는 S3 테이블 버킷에 연결하는 AWS Glue 작업을 설정하는 방법을 보여줍니다. AWS CLI를 사용하거나 콘솔의 AWS Glue Studio 스크립트 편집기를 사용하여 이 작업을 수행할 수 있습니다. 자세한 내용은 AWS Glue 사용 설명서에서 Authoring jobs in AWS Glue를 참조하세요.

AWS Glue Studio 스크립트 편집기 사용

다음 절차에서는 AWS Glue Studio 스크립트 편집기를 사용하여 S3 테이블을 쿼리하는 ETL 작업을 생성하는 방법을 보여줍니다.

사전 조건

- 사전 조건
- 테이블 버킷에 연결할 스크립트 생성

1. https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창에서 ETL 작업을 선택합니다.
3. 스크립트 편집기를 선택한 다음 스크립트 업로드를 선택하고 생성한 PySpark 스크립트를 업로드하여 S3 테이블을 쿼리합니다.
4. 작업 세부 정보 탭을 선택하고 기본 속성에 다음을 입력합니다.
   - 이름에 작업의 이름을 입력합니다.
   - IAM 역할에서 AWS Glue에 대해 생성한 IAM 역할을 선택합니다.
5. (선택 사항) Amazon S3 Tables Catalog for Apache Iceberg 액세스 방법을 사용하는 경우 고급 속성을 확장하고 종속 JAR 경로에 사전 조건으로 S3 버킷에 업로드한 클라이언트 카탈로그 jar의 S3 URI를 입력합니다. 예: s3://*amzn-s3-demo-bucket1*/*jars*/s3-tables-catalog-for-iceberg-runtime-*0.1.5*.jar
6. 저장을 선택하여 작업을 생성합니다.
7. 실행을 시작하여 작업을 시작하고 실행 탭에서 작업 상태를 검토합니다.

AWS CLI 사용

다음 절차에서는 AWS CLI를 사용하여 S3 테이블을 쿼리하는 ETL 작업을 생성하는 방법을 보여줍니다. 명령을 사용하려면 *## ### #*을 실제 값으로 바꿉니다.

사전 조건

- 사전 조건
- 테이블 버킷에 연결할 스크립트 생성한 다음 S3 버킷에 업로드합니다.

1. AWS Glue 작업을 생성합니다.

```
aws glue create-job \
--name etl-tables-job \
--role arn:aws:iam::111122223333:role/AWSGlueServiceRole \
--command '{
    "Name": "glueetl",
    "ScriptLocation": "s3://amzn-s3-demo-bucket1/scripts/glue-etl-query.py",
    "PythonVersion": "3"
}' \
--default-arguments '{
    "--job-language": "python",
    "--class": "GlueApp"
}' \
--glue-version "5.0"
```

> **Note**
>
> (선택 사항) Amazon S3 Tables Catalog for Apache Iceberg 액세스 방법을 사용하는 경우 `--extra-jars` 파라미터를 사용하여 `--default-arguments`에 클라이언트 카탈로그 JAR을 추가합니다. 파라미터를 추가할 때 *## ## ###*를 실제 정보로 바꿉니다.
>
> ```
> "--extra-jars": "s3://amzn-s3-demo-bucket/jar-path/s3-tables-catalog-for-iceberg-runtime-0.1.5.jar"
> ```

2. 작업을 시작합니다.

```
aws glue start-job-run \
```

```
--job-name etl-tables-job
```

3. 작업 상태를 검토하려면 이전 명령에서 실행 ID를 복사하여 다음 명령에 입력합니다.

```
aws glue get-job-run --job-name etl-tables-job \
--run-id jr_ec9a8a302e71f8483060f87b6c309601ea9ee9c1ffc2db56706dfcceb3d0e1ad
```

S3 Tables AWS 리전, 엔드포인트 및 서비스 할당량

다음 섹션에는 S3 Tables에 대해 지원되는 AWS 리전 및 서비스 할당량이 포함되어 있습니다.

주제

- S3 Tables AWS 리전 및 엔드포인트
- S3 Tables 할당량

S3 Tables AWS 리전 및 엔드포인트

현재 S3 Tables을 사용할 수 있는 리전 목록은 Amazon S3 endpoints를 참조하세요. AWS 서비스에 프로그래밍 방식으로 연결하려면 엔드포인트를 사용해야 합니다. 자세한 내용은 AWS 서비스 엔드포인트를 참조하세요. Amazon S3 엔드포인트에 대한 자세한 내용은 Amazon S3 endpoints를 참조하세요.

리전 이름	지역	엔드포인트	프로토콜	서명 버전 지원
아프리카(케이프타운)	af-south-1	s3tables.af-south-1.amazonaws.com	HTTPS	4
아시아 태평양(홍콩)	ap-east-1	s3tables.ap-east-1.amazonaws.com	HTTPS	4
아시아 태평양(도쿄)	ap-northeast-1	s3tables.ap-northeast-1.amazonaws.com	HTTPS	4

리전 이름	지역	엔드포인트	프로토콜	서명 버전 지원
아시아 태평양(서울)	ap-northeast-2	s3tables.ap-northeast-2.amazonaws.com	HTTPS	4
아시아 태평양(오사카)	ap-northeast-3	s3tables.ap-northeast-3.amazonaws.com	HTTPS	4
아시아 태평양(뭄바이)	ap-south-1	s3tables.ap-south-1.amazonaws.com	HTTPS	4
아시아 태평양(하이데라바드)	ap-south-2	s3tables.ap-south-2.amazonaws.com	HTTPS	4
아시아 태평양(싱가포르)	ap-southeast-1	s3tables.ap-southeast-1.amazonaws.com	HTTPS	4
아시아 태평양(시드니)	ap-southeast-2	s3tables.ap-southeast-2.amazonaws.com	HTTPS	4
아시아 태평양(자카르타)	ap-southeast-3	s3tables.ap-southeast-3.amazonaws.com	HTTPS	4

리전 이름	지역	엔드포인트	프로토콜	서명 버전 지원
아시아 태평양(멜버른)	ap-southeast-4	s3tables.ap-southeast-4.amazonaws.com	HTTPS	4
아시아 태평양(말레이시아)	ap-southeast-5	s3tables.ap-southeast-5.amazonaws.com	HTTPS	4
아시아 태평양(태국)	ap-southeast-7	s3tables.ap-southeast-7.amazonaws.com	HTTPS	4
캐나다(중부)	ca-central-1	s3tables.ca-central-1.amazonaws.com	HTTPS	4
캐나다 서부(캘거리)	ca-west-1	s3tables.ca-west-1.amazonaws.com	HTTPS	4
유럽(프랑크푸르트)	eu-central-1	s3tables.eu-central-1.amazonaws.com	HTTPS	4
유럽(취리히)	eu-central-2	s3tables.eu-central-2.amazonaws.com	HTTPS	4

리전 이름	지역	엔드포인트	프로토콜	서명 버전 지원
유럽(스톡홀름)	eu-north-1	s3tables.eu-north-1.amazonaws.com	HTTPS	4
유럽(밀라노)	eu-south-1	s3tables.eu-south-1.amazonaws.com	HTTPS	4
유럽(스페인)	eu-south-2	s3tables.eu-south-2.amazonaws.com	HTTPS	4
유럽(아일랜드)	eu-west-1	s3tables.eu-west-1.amazonaws.com	HTTPS	4
유럽(런던)	eu-west-2	s3tables.eu-west-2.amazonaws.com	HTTPS	4
유럽(파리)	eu-west-3	s3tables.eu-west-3.amazonaws.com	HTTPS	4
이스라엘(텔아비브)	il-central-1	s3tables.il-central-1.amazonaws.com	HTTPS	4

리전 이름	지역	엔드포인트	프로토콜	서명 버전 지원
중동(UAE)	me-central-1	s3tables.me-central-1.amazonaws.com	HTTPS	4
중동(바레인)	me-south-1	s3tables.me-south-1.amazonaws.com	HTTPS	4
멕시코(중부)	mx-central-1	s3tables.mx-central-1.amazonaws.com	HTTPS	4
남아메리카(상파울루)	sa-east-1	s3tables.sa-east-1.amazonaws.com	HTTPS	4
미국 동부(버지니아 북부)	us-east-1	s3tables.us-east-1.amazonaws.com	HTTPS	4
미국 동부(오하이오)	us-east-2	s3tables.us-east-2.amazonaws.com	HTTPS	4
미국 서부(캘리포니아 북부)	us-west-1	s3tables.us-west-1.amazonaws.com	HTTPS	4

리전 이름	지역	엔드포인트	프로토콜	서명 버전 지원
미국 서부(오레곤)	us-west-2	`s3tables.us-west-2.amazonaws.com`	HTTPS	4

S3 Tables 할당량

한도라고도 하는 할당량은 AWS 계정의 최대 서비스 리소스 수 또는 작업 수입니다. S3 Tables 리소스에 대한 할당량이 아래에 나와 있습니다. Amazon S3 할당량에 대한 자세한 내용은 Amazon S3 quotas를 참조하세요.

명칭	기본값	조정 가능	설명
테이블 버킷	10	할당량 증가를 요청하려면 지원에 문의하십시오.	계정에서 AWS 리전당 생성할 수 있는 Amazon S3 테이블 버킷의 수입니다.
네임스페이스	10,000개	할당량 증가를 요청하려면 지원에 문의하십시오.	테이블 버킷당 생성할 수 있는 Amazon S3 테이블 네임스페이스 수입니다.
표	10,000개	할당량 증가를 요청하려면 지원에 문의하십시오.	테이블 버킷당 생성할 수 있는 Amazon S3 테이블 수입니다.

S3 Tables의 보안

Amazon S3은 다양한 보안 기능 및 도구를 제공합니다. 다음은 S3 Tables에서 지원하는 이러한 기능 및 도구의 목록입니다. 이러한 도구를 올바르게 적용하면 리소스를 보호하고 의도한 사용자만 리소스에 액세스하도록 할 수 있습니다.

ID 기반 정책

자격 증명 기반 정책은 IAM 사용자, 그룹 또는 역할에 연결됩니다. ID 기반 정책을 사용하여 테이블 버킷이나 테이블에 대한 IAM ID 액세스 권한을 부여할 수 있습니다. 기본적으로 사용자 및 역할에는 테이블 및 테이블 버킷을 생성하고 수정할 수 있는 권한이 없습니다. 이들은 S3 콘솔, AWS CLI 또는 Amazon S3 REST API를 사용해서도 작업을 수행할 수 없습니다. 내 계정으로 IAM 사용자, 그룹, 역할을 만들고 액세스 정책을 연결할 수 있습니다. 그런 다음 리소스에 대한 액세스 권한을 부여할 수 있습니다. 테이블 버킷 및 테이블을 생성하고 액세스하려면 IAM 관리자가 AWS Identity and Access Management(IAM) 역할 또는 사용자에게 필요한 권한을 부여해야 합니다. 자세한 내용은 Access management for S3 Tables을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결됩니다. 테이블 버킷 및 테이블에 대한 리소스 기반 정책을 생성할 수 있습니다. 테이블 버킷 정책을 사용하여 테이블 버킷 및 네임스페이스 수준 API 액세스 권한을 제어할 수 있습니다. 또한 테이블 버킷 정책을 사용하여 버킷의 여러 테이블에 대한 테이블 수준 API 권한을 제어할 수 있습니다. 정책 정의에 따라 버킷에 연결된 권한이 버킷의 모든 테이블 또는 특정 테이블에 적용될 수 있습니다. 또한 테이블 정책을 사용하여 버킷의 개별 테이블에 테이블 수준 API 액세스 권한을 부여할 수 있습니다.

S3 Tables이 테이블 버킷 작업 또는 테이블 작업을 수행하기 위한 요청을 수신하면 먼저 요청자에게 필요한 권한이 있는지 확인합니다. 요청을 승인할지 결정할 때 모든 관련 액세스 정책, 사용자 정책 및 리소스 기반 정책(IAM 사용자 정책, IAM 역할 정책, 테이블 버킷 정책 및 테이블 정책)을 평가합니다. 테이블 버킷 정책 및 테이블 정책을 사용하면 리소스에 대한 액세스를 개인 맞춤화하여 승인한 ID만 리소스에 액세스하고 해당 리소스에 대한 작업을 수행할 수 있도록 할 수 있습니다. 자세한 내용은 Access management for S3 Tables을 참조하세요.

S3 Tables에 대한 AWS Organizations 서비스 제어 정책(SCP)

서비스 제어 정책(SCP)에서 Amazon S3 Tables을 사용하여 조직의 사용자에 대한 권한을 관리할 수 있습니다. IAM 및 리소스 정책과 마찬가지로 모든 테이블 및 버킷 수준 작업은 정책에서 `s3tables` 네임스페이스의 일부로 참조됩니다. 자세한 내용은 AWS Organizations 사용 설명서의 서비스 제어 정책(SCP)을 참조하세요.

주제

- 암호화로 S3 테이블 데이터 보호
- S3 Tables의 액세스 관리
- S3 Tables의 VPC 연결
- S3 Tables의 보안 고려 사항 및 제한 사항

암호화로 S3 테이블 데이터 보호

테이블 버킷에서 AWS KMS 키(SSE-KMS)를 사용한 서버 측 암호화 사용

주제

- 테이블 및 테이블 버킷에서 SSE-KMS의 작동 방식
- 테이블 및 테이블 버킷에 대한 SSE-KMS 사용 적용 및 범위 지정
- 테이블 및 테이블 버킷에 대한 SSE-KMS 암호화 모니터링 및 감사
- S3 Tables SSE-KMS 암호화에 대한 권한 요구 사항
- 테이블 버킷에서 AWS KMS 키(SSE-KMS)를 사용한 서버 측 암호화 지정

테이블 버킷에는 Amazon S3 관리형 키(SSE-S3)를 통한 서버 측 암호화를 사용하여 테이블을 자동으로 암호화하는 기본 암호화 구성이 있습니다. 이 암호화는 S3 테이블 버킷의 모든 객체에 적용되며 무료로 제공됩니다.

키 교체 및 액세스 정책 권한 부여 관리와 같이 암호화 키를 더 세밀하게 제어해야 하는 경우 AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 통한 서버 측 암호화를 사용하도록 테이블 버킷을 구성할 수 있습니다. AWS KMS 내 보안 관리가 암호화 관련 규정 준수 요구 사항을 충족하는 데 도움이 될 수 있습니다. SSE-KMS에 대한 자세한 내용은 AWS KMS 키를 사용한 서버 측 암호화(SSE-KMS) 사용 단원을 참조하십시오.

테이블 및 테이블 버킷에서 SSE-KMS의 작동 방식

테이블 버킷의 SSE-KMS는 다음과 같은 측면에서 범용 버킷의 SSE-KMS와 다릅니다.

- 테이블 버킷 및 개별 테이블에 대해 암호화 설정을 지정할 수 있습니다.
- SSE-KMS에서는 고객 관리형 키만 사용할 수 있습니다. AWS 관리형 키는 지원되지 않습니다.
- 특정 역할 및 AWS 서비스 위탁자에게 AWS KMS 키에 액세스할 수 있는 권한을 부여해야 합니다. 자세한 내용은 S3 Tables SSE-KMS 암호화에 대한 권한 요구 사항 섹션을 참조하세요. 여기에는 다음과 같은 권한 부여가 포함됩니다.
  - S3 유지 관리 위탁자 - 암호화된 테이블 유지 관리
  - S3 Tables 통합 역할 - AWS 분석 서비스에서 암호화된 테이블에 대한 작업 수행
  - 클라이언트 액세스 역할 - Apache Iceberg 클라이언트에서 암호화된 테이블에 직접 액세스
  - S3 메타데이터 위탁자 - 암호화된 S3 메타데이터 테이블 업데이트
- 암호화된 테이블은 SSE-KMS로 암호화된 테이블을 더 비용 효율적으로 사용하기 위해 AWS KMS에 대한 요청 수를 최소화하는 테이블 수준 키를 사용합니다.

테이블 버킷에 대한 SSE-KMS 암호화

테이블 버킷을 생성할 때 SSE-KMS를 기본 암호화 유형으로 선택하고 암호화에 사용할 특정 KMS 키를 선택할 수 있습니다. 해당 버킷 내에 생성된 모든 테이블은 테이블 버킷에서 이러한 암호화 설정을 자동으로 상속합니다. AWS CLI, S3 API 또는 AWS SDK를 사용하여 언제든지 테이블 버킷의 기본 암호화 설정을 수정하거나 제거할 수 있습니다. 테이블 버킷의 암호화 설정을 수정할 때 그러한 설정은 해당 버킷에서 생성된 새 테이블에만 적용됩니다. 기존 테이블에 대한 암호화 설정은 변경되지 않습니다. 자세한 내용은 테이블 버킷에 대한 암호화 지정 섹션을 참조하세요.

테이블에 대한 SSE-KMS 암호화

버킷의 기본 암호화 구성에 관계없이 다른 KMS 키로 개별 테이블을 암호화하는 옵션도 있습니다. 개별 테이블에 대한 암호화를 설정하려면 테이블 생성 시 원하는 암호화 키를 지정해야 합니다. 기존 테이블의 암호화를 변경하려면 원하는 키로 테이블을 생성하고 이전 테이블에서 새 테이블로 데이터를 복사해야 합니다. 자세한 내용은 테이블에 대한 암호화 지정 섹션을 참조하세요.

AWS KMS 암호화를 사용하는 경우 S3 Tables는 각 테이블과 연결된 새 객체를 암호화하는 고유한 테이블 수준 데이터 키를 자동으로 생성합니다. 이러한 키는 제한된 기간 동안 사용되므로 암호화 작업 중 추가 AWS KMS 요청의 필요성이 최소화되고 암호화 비용이 절감됩니다. 이는 SSE-KMS용 S3 버킷 키와 비슷합니다.

테이블 및 테이블 버킷에 대한 SSE-KMS 사용 적용 및 범위 지정

S3 Tables 리소스 기반 정책, KMS 키 정책, IAM ID 기반 정책 또는 이들의 조합을 사용하여 S3 테이블 및 테이블 버킷에 대한 SSE-KMS 사용을 적용할 수 있습니다. 테이블의 ID 및 리소스 정책에 대한 자세한 내용은 S3 Tables의 액세스 관리 섹션을 참조하세요. 키 정책 작성에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 키 정책을 참조하세요. 다음 예제에서는 정책을 사용하여 SSE-KMS를 적용하는 방법을 보여줍니다.

테이블 버킷 정책을 통해 모든 테이블에 SSE-KMS 사용 적용

이는 사용자가 특정 AWS KMS 키로 테이블을 암호화하지 않는 한 특정 테이블 버킷에 테이블을 생성하지 못하도록 하는 테이블 버킷 정책의 예입니다. 이 정책을 사용하려면 *### ## ## ###*를 실제 정보로 바꿉니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EnforceKMSEncryption",
```

```
      "Effect": "Deny",
      "Principal": "*",
      "Action": [
        "s3tables:CreateTable"
      ],
      "Resource": [
        "<table-bucket-arn>/*"
      ],
      "Condition": {
        "StringNotEquals": {
          "s3tables:sseAlgorithm": "aws:kms",
          "s3tables:kmsKeyArn": "<kms-key-arn>"
        }
      }
    }
  ]
}
```

IAM 정책을 사용하여 사용자에게 SSE-KMS 암호화 사용 요구

이 IAM ID 정책은 사용자가 S3 Tables 리소스를 생성하거나 구성할 때 암호화에 특정 AWS KMS 키를 사용하도록 요구합니다. 이 정책을 사용하려면 *### ## ## ###*를 실제 정보로 바꿉니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RequireKMSKeyOnTables",
      "Action": [
          "s3tables:CreateTableBucket",
          "s3tables:PutTableBucketEncryption",
          "s3tables:CreateTable"
      ]
      "Effect": "Deny",
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
            "s3tables:sseAlgorithm": "aws:kms",
            "s3tables:kmsKeyArn": "<key_arn>"
        }
      }
    }
  ]
```

```
}
```

KMS 키 정책을 사용하여 키 사용을 특정 테이블 버킷으로 제한

이 예제 KMS 키 정책은 특정 사용자가 특정 테이블 버킷의 암호화 작업에만 키를 사용할 수 있도록 허용합니다. 이 유형의 정책은 교차 계정 시나리오에서 키에 대한 액세스를 제한하는 데 유용합니다. 이 정책을 사용하려면 *### ## ## ###*를 실제 정보로 바꿉니다.

```
{
  "Version": "2012-10-17",
  "Id": "Id",
  "Statement": [
    {
      "Sid": "AllowPermissionsToKMS",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "kms:EncryptionContext:aws:s3:arn": "<table-bucket-arn>/*"
        }
      }
    }
  ]
}
```

테이블 및 테이블 버킷에 대한 SSE-KMS 암호화 모니터링 및 감사

AWS CloudTrail 로그를 사용하여 SSE-KMS 암호화 데이터의 AWS KMS 키 사용을 감사할 수 있습니다. `GenerateDataKey` 및 `Decrypt`와 같은 암호화 작업에 대한 인사이트를 얻을 수 있습니다. CloudTrail은 이벤트 이름, 사용자 이름, 이벤트 소스 등 검색 필터링을 위한 다양한 속성 값을 지원합니다.

CloudTrail 이벤트를 사용하여 Amazon S3 테이블 및 테이블 버킷에 대한 기본 암호화 구성 요청을 추적할 수 있습니다. CloudTrail 로그에 사용되는 API 이벤트 이름은 다음과 같습니다.

- `s3tables:PutTableBucketEncryption`
- `s3tables:GetTableBucketEncryption`
- `s3tables:DeleteTableBucketEncryption`
- `s3tables:GetTableEncryption`
- `s3tables:CreateTable`
- `s3tables:CreateTableBucket`

> **Note**
>
> 테이블 버킷에는 EventBridge가 지원되지 않습니다.

S3 Tables SSE-KMS 암호화에 대한 권한 요구 사항

S3 테이블 버킷의 테이블에 대해 AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 통한 서버 측 암호화를 사용하는 경우 계정의 다양한 ID에 권한을 부여해야 합니다. 최소한 액세스 ID와 S3 Tables 유지 관리 위탁자가 키에 대한 액세스 권한을 가져야 하며 사용 사례에 따라 다른 권한도 필요합니다.

필수 권한

KMS 키로 암호화된 테이블에 액세스하려면 해당 키에 대한 다음 권한이 필요합니다.

- `kms:GenerateDataKey`
- `kms:Decrypt`

> **Important**
>
> 테이블에서 SSE-KMS를 사용하려면 Amazon S3 Tables 유지 관리 서비스 위탁자(`maintenance.s3tables.amazonaws.com`)에게 키에 대한 `kms:GenerateDataKey` 및 `kms:Decrypt` 권한이 필요합니다.

추가 권한

사용 사례에 따라 다음과 같은 추가 권한이 필요합니다.

- AWS 분석 서비스 통합 권한 - AWS 분석 서비스에서 SSE-KMS로 암호화된 테이블에 대한 작업을 수행하는 경우 통합 역할에 KMS 키를 사용할 수 있는 권한이 필요합니다.

- 직접 액세스 권한 - Amazon S3 Tables Iceberg REST 엔드포인트 또는 Amazon S3 Tables Catalog for Apache Iceberg와 같은 방법을 통해 SSE-KMS로 암호화된 테이블로 직접 작업하는 경우 클라이언트가 사용하는 IAM 역할에 키에 대한 액세스 권한을 부여해야 합니다.
- S3 Metadata 테이블에 대한 권한 - S3 Metadata 테이블에 SSE-KMS 암호화를 사용하는 경우 S3 Metadata 서비스 위탁자(`metadata.s3.amazonaws.com`)에게 KMS 키에 대한 액세스 권한을 제공해야 합니다. 이렇게 하면 S3 Metadata가 암호화된 테이블을 업데이트하여 최신 데이터 변경 사항을 반영할 수 있습니다.

> **Note**
>
> 교차 계정 KMS 키의 경우 IAM 역할에 키 액세스 권한과 키 정책 내 명시적 권한 부여가 모두 필요합니다. KMS 키의 교차 계정 권한에 대한 자세한 내용은 AWS Key Management Service 서비스 개발자 안내서의 외부 AWS 계정에서 KMS 키를 사용하도록 허용을 참조하세요.

주제

- S3 Tables 유지 관리 서비스 위탁자에게 KMS 키 권한 부여
- IAM 위탁자에게 통합 AWS 분석 서비스에서 암호화된 테이블에 대한 작업을 수행할 수 있는 권한 부여
- IAM 위탁자에게 암호화된 테이블로 직접 작업할 수 있는 권한 부여
- S3 Metadata 서비스 위탁자에게 KMS 키를 사용할 수 있는 권한 부여

S3 Tables 유지 관리 서비스 위탁자에게 KMS 키 권한 부여

이 권한은 SSE-KMS로 암호화된 테이블을 생성하고 암호화된 테이블에서 압축, 스냅샷 관리 및 참조되지 않은 파일 제거와 같은 자동 테이블 유지 관리를 수행할 수 있도록 하는 데 필요합니다.

> **Note**
>
> SSE-KMS로 암호화된 테이블을 생성하도록 요청할 때마다 S3 Tables는 `maintenance.s3tables.amazonaws.com` 위탁자에게 KMS 키에 대한 액세스 권한이 있는지 확인합니다. 이 확인을 수행하기 위해 테이블 버킷에 0바이트 객체가 일시적으로 생성되며, 이 객체는 참조되지 않은 파일 제거 유지 관리 작업에 의해 자동으로 제거됩니다. 암호화를 위해 지정한 KMS 키에 유지 관리 액세스 권한이 없는 경우 createTable 작업이 실패합니다.

SSE-KMS로 암호화된 테이블에 대한 유지 관리 액세스 권한을 부여하려면 다음 예제 키 정책을 사용할 수 있습니다. 이 정책에서는 `maintenance.s3tables.amazonaws.com` 서비스 위탁자에게 특정 테이블 버킷의 테이블 암호화 및 복호화에 특정 KMS 키를 사용할 수 있는 권한이 부여되어 있습니다. 이 정책을 사용하려면 *### ## ## ###*를 실제 정보로 바꿉니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnableKeyUsage",
            "Effect": "Allow",
            "Principal": {
                "Service": "maintenance.s3tables.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "<kms-key-arn>",
            "Condition": {
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn":"<table-or-table-bucket-arn>/*"
                }
            }
        }
    ]
}
```

IAM 위탁자에게 통합 AWS 분석 서비스에서 암호화된 테이블에 대한 작업을 수행할 수 있는 권한 부여

AWS 분석 서비스에서 S3 테이블에 대한 작업을 수행하려면 테이블 버킷을 Amazon SageMaker Lakehouse와 통합하세요. 이 통합을 통해 AWS 분석 서비스가 테이블 데이터를 자동으로 검색하고 액세스할 수 있습니다. 통합에 대한 자세한 내용은 AWS 분석 서비스에서 Amazon S3 Tables 사용 섹션을 참조하세요.

이러한 서비스에서 SSE-KMS로 암호화된 테이블에 대한 작업을 수행하려면 사용하는 역할이 암호화 작업에 AWS KMS 키를 사용할 수 있는 권한을 가져야 합니다. 이러한 권한을 통합 중에 생성된 `S3TablesRoleForLakeFormation` 역할 또는 자체 IAM 역할에 적용할 수 있습니다.

다음 인라인 IAM 정책 예제는 계정에서 암호화 작업에 특정 KMS 키를 사용할 수 있는 권한을 `S3TablesRoleForLakeFormation` 서비스 역할에 부여하는 데 사용할 수 있습니다. 이 정책을 사용하려면 *## ## ### #*을 실제 값으로 바꿉니다.

```
{
  "Sid": "AllowTableRoleAccess",
  "Effect": "Allow",
  "Principal": {
 "AWS": "arn:aws:iam::111122223333:role/service-role/S3TablesRoleForLakeFormation"
  },
  "Action": [
      "kms:GenerateDataKey",
      "kms:Decrypt"
  ],
  "Resource": "<kms-key-arn>"
}
```

IAM 위탁자에게 암호화된 테이블로 직접 작업할 수 있는 권한 부여

서드파티 또는 직접 액세스 방법을 사용하여 암호화된 테이블로 작업하는 경우 사용하는 역할에 KMS 키에 대한 액세스 권한을 부여해야 합니다. 다음 예제에서는 IAM 정책 또는 KMS 키 정책을 통해 액세스 권한을 부여하는 방법을 보여줍니다.

IAM policy

이 인라인 정책을 IAM 역할에 연결하여 KMS 키 액세스를 허용합니다. 이 정책을 사용하려면 *## # # ### #*을 실제 KMS 키 ARN으로 바꿉니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Effect": "Allow",
        "Action": [
            "kms:Decrypt",
            "kms:GenerateDataKey"
         ],
        "Resource": "<kms-key-arn>"
        }
    ]
}
```

KMS key policy

지정된 AWS KMS 역할이 키를 사용할 수 있도록 이 인라인 정책을 KMS 키에 연결합니다. 이 정책을 사용하려면 *## ## ### #*을 실제 IAM 역할로 바꿉니다.

```
{
    "Sid": "Allow use of the key",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::<catalog-account-id>:role/<role-name>"
        ]
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey",
    ],
    "Resource": "*"
}
```

S3 Metadata 서비스 위탁자에게 KMS 키를 사용할 수 있는 권한 부여

Amazon S3가 SSE-KMS로 암호화된 메타데이터 테이블을 업데이트하고 해당 메타데이터 테이블에 대한 유지 관리를 수행하도록 허용하려면 다음 예제 키 정책을 사용할 수 있습니다. 이 정책에서는 `metadata.s3.amazonaws.com` 및 `maintenance.s3tables.amazonaws.com` 서비스 위탁자가 특정 키를 사용하여 특정 테이블 버킷의 테이블을 암호화하고 복호화하도록 허용합니다. 이 정책을 사용하려면 *### ## ## ###*를 실제 정보로 바꿉니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnableKeyUsage",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "maintenance.s3tables.amazonaws.com",
                    "metadata.s3.amazonaws.com"
                ]
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "<kms-key-arn>",
            "Condition": {
```

```
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn":"<table-or-table-bucket-arn>/*"
                }
            }
        }
    ]
}
```

테이블 버킷에서 AWS KMS 키(SSE-KMS)를 사용한 서버 측 암호화 지정

모든 Amazon S3 테이블 버킷에는 기본적으로 암호화가 구성되어 있으며, 테이블 버킷에 생성된 새 테이블은 모두 저장 시 자동으로 암호화됩니다. Amazon S3 관리형 키(SSE-S3)를 사용한 서버 측 암호화가 모든 테이블 버킷의 기본 암호화 구성입니다. 다른 암호화 유형을 지정하려는 경우 AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 통한 서버 측 암호화를 사용할 수 있습니다.

`CreateTableBucket` 또는 `CreateTable` 요청에서 SSE-KMS 암호화를 지정하거나 `PutTableBucketEncryption` 요청의 테이블 버킷에서 기본 암호화 구성을 설정할 수 있습니다.

> **Important**
>
> SSE-KMS로 암호화된 테이블 및 테이블 버킷에 대한 자동 유지 관리를 허용하려면 maintenance.s3tables.amazonaws.com 서비스 위탁자에게 KMS 키를 사용할 수 있는 권한을 부여해야 합니다. 자세한 내용은 S3 Tables SSE-KMS 암호화에 대한 권한 요구 사항 섹션을 참조하세요.

테이블 버킷에 대한 암호화 지정

새 테이블 버킷을 생성할 때 SSE-KMS를 기본 암호화 유형으로 지정할 수 있습니다. 관련 예제는 테이블 버킷 생성 섹션을 참조하세요. 테이블 버킷을 생성한 후 REST API 작업, AWS SDK 및 AWS Command Line Interface(AWS CLI)를 사용하여 SSE-KMS 사용을 기본 암호화 설정으로 지정할 수 있습니다.

> **Note**
>
> SSE-KMS를 기본 암호화 유형으로 지정하는 경우 암호화에 사용하는 키가 S3 Tables 유지 관리 서비스 위탁자에게 액세스를 허용해야 합니다. 유지 관리 서비스 위탁자에게 액세스 권한이 없는 경우 해당 테이블 버킷에 테이블을 생성할 수 없습니다. 자세한 내용은 S3 Tables 유지 관리 서비스 위탁자에게 KMS 키 권한 부여 섹션을 참조하세요.

AWS CLI 사용

다음 예제 AWS CLI 명령을 사용하려면 *### ## ## ###*를 실제 정보로 바꿉니다.

```
aws s3tables put-table-bucket-encryption \
    --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket; \
    --encryption-configuration '{
        "sseAlgorithm": "aws:kms",
        "kmsKeyArn": "arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }' \
    --region us-east-1
```

DeleteTableBucketEncryption API 작업을 사용하여 테이블 버킷의 기본 암호화 설정을 제거할 수 있습니다. 암호화 설정을 제거하면 테이블 버킷에 생성된 새 테이블이 기본 SSE-S3 암호화를 사용합니다.

테이블에 대한 암호화 지정

쿼리 엔진, REST API 작업, AWS SDK 및 AWS Command Line Interface(AWS CLI)를 사용하여 새 테이블을 생성할 때 새 테이블에 SSE-KMS 암호화를 적용할 수 있습니다. 테이블을 생성할 때 지정하는 암호화 설정이 테이블 버킷의 기본 암호화 설정보다 우선합니다.

> **Note**
>
> 테이블에 SSE-KMS 암호화를 사용하는 경우 암호화에 사용하는 키가 S3 Tables 유지 관리 서비스 위탁자에게 키에 대한 액세스를 허용해야 합니다. 유지 관리 서비스 위탁자에게 액세스 권한이 없으면 테이블을 생성할 수 없습니다. 자세한 내용은 S3 Tables 유지 관리 서비스 위탁자에게 KMS 키 권한 부여 섹션을 참조하세요.

필요한 권한

암호화된 테이블을 생성하려면 다음 권한이 필요합니다.

- `s3tables:CreateTable`
- `s3tables:PutTableEncryption`

AWS CLI 사용

다음 AWS CLI 예제에서는 기본 스키마를 사용하여 새 테이블을 생성하고 고객 관리형 AWS KMS 키로 암호화합니다. 명령을 사용하려면 *### ## ## ###*를 사용자의 정보로 대체합니다.

```
aws s3tables create-table \
  --table-bucket-arn "arn:aws:s3tables:Region:ownerAccountId:bucket/amzn-s3-demo-table-bucket" \
  --namespace "mydataset" \
  --name "orders" \
  --format "ICEBERG" \
  --encryption-configuration '{
    "sseAlgorithm": "aws:kms",
    "kmsKeyArn":
 "arn:aws:kms:Region:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  }' \
  --metadata '{
    "iceberg": {
      "schema": {
        "fields": [
          {
            "name": "order_id",
            "type": "string",
            "required": true
          },
          {
            "name": "order_date",
            "type": "timestamp",
            "required": true
          },
          {
            "name": "total_amount",
            "type": "decimal(10,2)",
            "required": true
          }
        ]
      }
    }
  }'
```

데이터 보호란 전송 중(Amazon S3 안팎으로 데이터가 이동 중)과 저장 시(Amazon S3 데이터 센터의 디스크에 데이터가 저장된 동안) 데이터를 보호하는 것을 말합니다. S3 Tables는 항상 HTTPS를 통해

전송 계층 보안(1.2 이상)을 사용하여 전송 중 데이터를 보호합니다. S3 테이블 버킷에서 저장 데이터를 보호하는 데는 다음과 같은 옵션이 있습니다.

Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)

모든 Amazon S3 테이블 버킷에는 기본적으로 암호화가 구성되어 있습니다. 서버 측 암호화의 기본 옵션은 Amazon S3 관리형 키(SSE-S3)를 사용하는 것입니다. 이 암호화는 무료로 제공되며 다른 형태의 암호화를 지정하지 않는 한 S3 테이블 버킷의 모든 테이블에 적용됩니다. 각 객체가 고유한 키로 암호화됩니다. 또한 추가 보안 조치로 SSE-S3는 주기적으로 교체되는 루트 키를 사용하여 키 자체를 암호화합니다. SSE-S3는 가장 강력한 블록 암호 중 하나인 256비트 Advanced Encryption Standard(AES-256)를 사용하여 데이터를 암호화합니다.

AWS KMS 키를 사용한 서버 측 암호화(SSE-KMS)

AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 통한 서버 측 암호화를 사용하도록 테이블 버킷 또는 테이블을 구성할 수도 있습니다. AWS KMS 내 보안 관리가 암호화 관련 규정 준수 요구 사항을 충족하는 데 도움이 될 수 있습니다. SSE-KMS를 사용하면 다음 작업을 수행할 수 있어 암호화 키를 더 효과적으로 제어할 수 있습니다.

- KMS 키를 생성, 조회, 편집, 모니터링, 활성화/비활성화, 교체, 예약 삭제합니다.
- KMS 키를 사용할 수 있는 방법 및 주체를 제어하는 정책을 정의합니다.
- AWS CloudTrail에서 키 사용을 추적하여 KMS 키가 올바르게 사용되고 있는지 확인합니다.

S3 Tables는 SSE-KMS에서 고객 관리형 키를 사용하여 테이블을 암호화하도록 지원합니다. AWS 관리형 키는 지원되지 않습니다. S3 테이블 및 테이블 버킷에 SSE-KMS를 사용하는 방법에 대한 자세한 내용은 테이블 버킷에서 AWS KMS 키(SSE-KMS)를 사용한 서버 측 암호화 사용 섹션을 참조하세요.

S3 Tables의 액세스 관리

S3 Tables 리소스에는 테이블 버킷과 여기에 들어 있는 테이블이 포함됩니다. 리소스를 생성한 AWS 계정의 루트 사용자(리소스 소유자)와 해당 계정 내에서 필요한 권한을 가진 AWS Identity and Access Management IAM 사용자는 자신이 생성한 리소스에 액세스할 수 있습니다. 리소스 소유자는 리소스에 액세스할 수 있는 다른 사용자와 리소스에서 이들이 수행할 수 있는 작업을 지정합니다. Amazon S3에는 다른 사람에게 내 S3 리소스에 대한 액세스 권한을 부여하는 데 사용할 수 있는 다양한 액세스 관리 도구가 있습니다. 다음 주제에서는 S3 Tables의 리소스, IAM 작업 및 조건 키에 대한 개요를 제공합니다. 또한 S3 Tables용 리소스 기반 정책 및 ID 기반 정책 모두에 대한 예시도 제공합니다.

주제

- 리소스
- S3 Tables에 대한 작업
- S3 Tables의 조건 키
- S3 Tables의 IAM ID 기반 정책
- S3 Tables의 리소스 기반 정책
- S3 Tables에 대한 AWS 관리형 정책
- SQL 의미 체계를 사용하여 액세스 권한 부여

리소스

S3 Tables 리소스에는 테이블 버킷과 여기에 들어 있는 테이블이 포함됩니다.

- 테이블 버킷 - 테이블 버킷은 테이블을 위해 특별히 설계되었으며 범용 S3 버킷의 자체 관리형 테이블에 비해 초당 트랜잭션 수(TPS)가 높고 쿼리 처리량이 더 우수합니다. 테이블 버킷은 Amazon S3 범용 버킷과 동일한 내구성, 가용성, 확장성 및 성능 특성을 제공합니다.
- 테이블 - 테이블 버킷의 테이블은 Apache Iceberg 형식으로 저장됩니다. Iceberg를 지원하는 쿼리 엔진에서 표준 SQL을 사용하여 이러한 테이블을 쿼리할 수 있습니다.

테이블 및 테이블 버킷의 Amazon 리소스 이름(ARN)에는 `s3tables` 네임스페이스, AWS 리전, AWS 계정 ID 및 버킷 이름이 포함됩니다. 테이블 및 테이블 버킷에 액세스하고 작업을 수행하려면 다음 ARN 형식을 사용해야 합니다.

- 표 ARN 형식:

```
arn:aws:s3tables:us-west-2:111122223333:bucket/amzn-s3-demo-bucket/
table/demo-tableID
```

S3 Tables에 대한 작업

ID 기반 정책 또는 리소스 기반 정책에서 특정 IAM 위탁자에 대해 어떤 S3 Tables 작업을 허용 또는 거부할지 정의합니다. 테이블 작업은 버킷 및 테이블 수준 API 작업에 해당합니다. 모든 작업은 고유한 IAM 네임스페이스인 `s3tables`의 일부입니다.

정책에서 작업을 사용하면 일반적으로 이름이 같은 API 작업에 대한 액세스를 허용하거나 거부합니다. 그러나 경우에 따라 하나의 API 작업으로 둘 이상의 작업에 대한 액세스가 제어됩니다. 예를 들어

s3tables:GetTableData 작업에 GetObject, ListParts 및 ListMultiparts API 작업에 대한 권한이 포함됩니다.

다음은 테이블 버킷에 지원되는 작업입니다. IAM 정책 또는 리소스 정책의 Action 요소에서 다음 작업을 지정할 수 있습니다.

작업	설명	액세스 레벨	크로스 계정 액세스
s3tables:CreateTableBucket	테이블 버킷을 생성할 수 있는 권한을 부여합니다.	Write	아니요
s3tables:GetTableBucket	테이블 버킷 ARN, 테이블 버킷 이름 및 생성 날짜를 검색할 수 있는 권한을 부여합니다.	Write	예
s3tables:ListTableBuckets	이 계정의 모든 테이블 버킷을 나열할 수 있는 권한을 부여합니다.	Read	아니요
s3tables:CreateNamespace	테이블 버킷에 네임스페이스를 생성할 수 있는 권한을 부여합니다.	Read	예
s3tables:GetNamespace	네임스페이스 세부 정보를 검색할 수 있	Read	예

작업	설명	액세스 레벨	크로스 계정 액세스
	는 권한을 부여합니다.		
s3tables:ListNamespaces	테이블 버킷의 모든 네임스페이스를 나열할 수 있는 권한을 부여합니다.	Read	예
s3tables:DeleteNamespace	테이블 버킷에서 네임스페이스를 삭제할 수 있는 권한을 부여합니다.	Write	예
s3tables:DeleteTableBucket	버킷을 삭제할 수 있는 권한을 부여합니다.	Write	예
s3tables:PutTableBucketPolicy	버킷 정책을 추가하거나 교체할 수 있는 권한을 부여합니다.	Permissions Management	아니요
s3tables:GetTableBucketPolicy	버킷 정책을 검색할 수 있는 권한을 부여합니다.	Read	아니요

작업	설명	액세스 레벨	크로스 계정 액세스
s3tables:DeleteTableBucketPolicy	버킷 정책을 삭제할 수 있는 권한을 부여합니다.	Permissions Management	아니요
s3tables:GetTableBucketMaintenanceConfiguration	테이블 버킷에 대한 유지 관리 구성을 검색할 수 있는 권한을 부여합니다.	Read	예
s3tables:PutTableBucketMaintenanceConfiguration	테이블 버킷에 대한 유지 관리 구성을 추가하거나 교체할 수 있는 권한을 부여합니다.	Write	예
s3tables:PutTableBucketEncryption	테이블 버킷에 대한 암호화 구성을 추가하거나 교체할 수 있는 권한을 부여합니다.	Write	아니요
s3tables:GetTableBucketEncryption	테이블 버킷에 대한 암호화 구성을 검색할 수 있는 권한을 부여합니다.	Read	아니요

작업	설명	액세스 레벨	크로스 계정 액세스
s3tables:DeleteTableBucketEncryption	테이블 버킷에 대한 암호화 구성을 삭제할 수 있는 권한을 부여합니다.	Write	아니요

테이블에 다음 작업이 지원됩니다.

작업	설명	액세스 레벨	크로스 계정 액세스
s3tables:GetTableMaintenanceConfiguration	테이블에 대한 유지 관리 구성을 검색할 수 있는 권한을 부여합니다.	Read	예
s3tables:PutTableMaintenanceConfiguration	테이블에 대한 유지 관리 구성을 추가하거나 교체할 수 있는 권한을 부여합니다.	Write	예
s3tables:PutTablePolicy	테이블 정책을 추가하거나 교체할 수 있는 권한을 부여합니다.	Permissions Management	아니요
s3tables:GetTablePolicy	테이블 정책을 검색할 수	Read	아니요

작업	설명	액세스 레벨	크로스 계정 액세스
	있는 권한을 부여합니다.		
s3tables:DeleteTablePolicy	테이블 정책을 삭제할 수 있는 권한을 부여합니다.	Permissions management	아니요
s3tables:CreateTable	테이블 버킷에 테이블을 생성할 수 있는 권한을 부여합니다.	Write	예
s3tables:GetTable	테이블 정보를 검색할 수 있는 권한을 부여합니다.	Read	예
s3tables:GetTableMetadataLocation	테이블 루트 포인터(메타데이터 파일)를 검색할 수 있는 권한을 부여합니다.	Read	예
s3tables:ListTables	테이블 버킷의 모든 테이블을 나열할 수 있는 권한을 부여합니다.	Read	예

작업	설명	액세스 레벨	크로스 계정 액세스
s3tables:RenameTable	테이블 이름을 변경할 수 있는 권한을 부여합니다.	Write	예
s3tables:UpdateTableMetadataLocation	테이블 루트 포인터(메타데이터 파일)를 업데이트할 수 있는 권한을 부여합니다.	Write	예
s3tables:GetTableData	테이블 버킷에 저장된 테이블 메타데이터 및 데이터 객체를 읽을 수 있는 권한을 부여합니다.	Read	예
s3tables:PutTableData	테이블 버킷에 저장된 테이블 메타데이터 및 데이터 객체를 쓸 수 있는 권한을 부여합니다.	Write	예

작업	설명	액세스 레벨	크로스 계정 액세스
s3tables:GetTableEncryption	테이블에 대한 암호화 설정을 검색할 수 있는 권한을 부여합니다.	Write	아니요
s3tables:PutTableEncryption	테이블에 암호화를 추가할 수 있는 권한을 부여합니다.	Write	아니요

테이블 수준 읽기 및 쓰기 작업을 수행하기 위해 S3 Tables은 GetObject 및 PutObject와 같은 Amazon S3 API 작업을 지원합니다. 다음 표에는 객체 수준 작업 목록이 나와 있습니다. 테이블에 읽기 및 쓰기 권한을 부여할 때 다음 작업을 사용합니다.

작업	S3 객체 API		
s3tables:GetTableData	GetObject, ListParts, HeadObject		
s3tables:PutTableData	PutObject, CreateMultipartUpload, CompleteMultipartUpload,		

작업	S3 객체 API		
	`UploadPart`, `AbortMultipartUpload`		

예를 들어 사용자에게 `GetTableData` 권한이 있는 경우 메타데이터 파일, 매니페스트, 매니페스트 목록 파일, 파켓 데이터 파일 등 테이블과 연결된 모든 파일을 읽을 수 있습니다.

S3 Tables의 조건 키

S3 Tables은 AWS 전역 조건 컨텍스트 키를 지원합니다.

또한 S3 Tables은 액세스 정책에 사용할 수 있는 다음과 같은 조건 키를 정의합니다.

조건 키	설명	형식	
`s3tables:tableName`	테이블 버킷의 테이블 이름을 기준으로 액세스를 필터링합니다. `s3tables:tableName` 조건 키를 사용하여 이 이름 조건을 충족하는 테이블로만 사용자 또는 애플리케이션 액세스를 제한하는 IAM 또는 테이블	`String`	

조건 키	설명	형식	
	버킷 정책을 작성할 수 있습니다. `s3tables:tableName` 조건 키를 사용하여 액세스를 제어하는 경우 테이블 이름을 변경하면 이러한 정책에 영향을 미칠 수 있습니다. 예시 값: `"s3tables:tableName":"department*"`		

조건 키	설명	형식	
s3tables:namespace	테이블 버킷에 생성된 네임스페이스를 기준으로 액세스를 필터링합니다. s3tables:namespace 조건 키를 사용하여 특정 네임스페이스의 일부인 테이블로 사용자 또는 애플리케이션 액세스를 제한하는 IAM, 테이블 또는 테이블 버킷 정책을 작성할 수 있습니다. 예시 값: "s3tables:namespace":"hr" s3tables:namespace 조건 키를 사용하여 액세스를 제어하는 경우 네임스페이스를 변경하면 이	String	

조건 키	설명	형식	
	러한 정책에 영향을 미칠 수 있습니다.		

조건 키	설명	형식	
s3tables:SSEAlgorithm	테이블을 암호화하는 데 사용되는 서버 측 암호화 알고리즘을 기준으로 액세스를 필터링합니다. s3tables:SSEAlgorithm 조건 키를 사용하여 특정 암호화 유형으로 암호화된 테이블로 사용자 또는 애플리케이션 액세스를 제한하는 IAM, 테이블 또는 테이블 버킷 정책을 작성할 수 있습니다. 예시 값: "s3tables:SSEAlgorithm":"aws:kms" s3tables:SSEAlgorithm 조건 키를 사용하여	String	

조건 키	설명	형식	
	액세스를 제어하는 경우 암호화를 변경하면 이러한 정책에 영향을 미칠 수 있습니다.		

조건 키	설명	형식	
s3tables:KMSKeyArn	테이블을 암호화하는 데 사용되는 키의 AWS KMS 키 ARN을 기준으로 액세스를 필터링합니다. s3tables:KMSKeyArn 조건 키를 사용하여 특정 KSM 키를 통해 암호화된 테이블로 사용자 또는 애플리케이션 액세스를 제한하는 IAM, 테이블 또는 테이블 버킷 정책을 작성할 수 있습니다. s3tables:KMSKeyArn 조건 키를 사용하여 액세스를 제어하는 경우 KSM 키를 변경하면 이러한 정책에 영	ARN	

조건 키	설명	형식	
	향을 미칠 수 있습니다.		

S3 Tables의 IAM ID 기반 정책

기본적으로 사용자 및 역할에는 테이블 및 테이블 버킷을 생성하거나 수정할 수 있는 권한이 없습니다. 이들은 S3 콘솔, AWS Command Line Interface(AWS CLI) 또는 Amazon S3 REST API를 사용해서도 작업을 수행할 수 없습니다. 테이블 버킷 및 테이블을 생성하고 액세스하려면 AWS Identity and Access Management(IAM) 관리자가 IAM 역할 또는 사용자에게 필요한 권한을 부여해야 합니다. 이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하십시오.

다음 주제에는 IAM ID 기반 정책의 예가 포함되어 있습니다. 다음 예시 정책을 사용하려면 *### ## # # ###*를 실제 정보로 바꾸세요.

주제

- 예시 1: 테이블 버킷 생성 및 사용에 대한 액세스 허용
- 예시 2: 테이블 버킷에서 테이블 생성 및 사용에 대한 액세스 허용

예시 1: 테이블 버킷 생성 및 사용에 대한 액세스 허용

.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "AllowBucketActions",
        "Effect": "Allow",
        "Action": [
            "s3tables:CreateTableBucket",
            "s3tables:PutTableBucketPolicy",
            "s3tables:GetTableBucketPolicy",
            "s3tables:ListTableBuckets",
            "s3tables:GetTableBucket"
        ],

        "Resource": "arn:aws:s3tables:region:account_id:bucket/*"
```

```
    }]
}
```

예시 2: 테이블 버킷에서 테이블 생성 및 사용에 대한 액세스 허용

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Sid": "AllowBucketActions",
             "Effect": "Allow",
             "Action": [
                 "s3tables:CreateTable",
                 "s3tables:PutTableData",
                 "s3tables:GetTableData",
                 "s3tables:GetTableMetadataLocation",
                 "s3tables:UpdateTableMetadataLocation",
                 "s3tables:GetNamespace",
                 "s3tables:CreateNamespace"
             ],

             "Resource": [
               "arn:aws:s3tables:region:account_id:bucket/amzn-s3-demo-bucket",
               "arn:aws:s3tables:region:account_id:bucket/amzn-s3-demo-bucket/table/*"
            ]
         }
     ]
 }
```

S3 Tables의 리소스 기반 정책

S3 Tables은 테이블 버킷 및 테이블에 대한 액세스를 관리하기 위한 리소스 기반 정책, 즉 테이블 버킷 정책 및 테이블 정책을 제공합니다. 테이블 버킷 정책을 사용하여 테이블 버킷, 네임스페이스 또는 테이블 수준에서 API 액세스 권한을 부여할 수 있습니다. 테이블 버킷에 연결된 권한은 정책 정의에 따라 버킷의 모든 테이블 또는 버킷의 특정 테이블에 적용될 수 있습니다. 테이블 정책을 사용하여 테이블 수준에서 권한을 부여할 수 있습니다.

S3 Tables이 요청을 수신하면 먼저 요청자에게 필요한 권한이 있는지 확인합니다. 요청을 승인할지 결정할 때 모든 관련 액세스 정책, 사용자 정책 및 리소스 기반 정책(IAM 사용자 정책, IAM 역할 정책, 테이블 버킷 정책 및 테이블 정책)을 평가합니다. 예를 들어 테이블 버킷 정책이 버킷의 테이블(`DeleteTable` 포함)에서 모든 작업을 수행할 수 있는 권한을 사용자에게 부여하지만 개별 테이블에

모든 사용자에 대해 DeleteTable 작업을 거부하는 테이블 정책이 있는 경우 사용자는 테이블을 삭제할 수 없습니다.

다음 주제에는 테이블 및 테이블 버킷 정책의 예가 포함되어 있습니다. 이러한 정책을 사용하려면 ### ## ## ###를 실제 정보로 바꾸세요.

> **Note**
>
> - 테이블을 수정할 수 있는 권한을 부여하는 모든 정책에는 GetTableMetadataLocation이 테이블 루트 파일에 액세스할 수 있는 권한이 포함되어야 합니다. 자세한 내용은 GetTableMetadataLocation 섹션을 참조하세요.
> - 테이블에서 쓰기 또는 삭제 활동을 수행할 때마다 액세스 정책에 UpdateTableMetadataLocation에 대한 권한을 포함합니다.
> - 버킷 수준 작업에 대한 액세스를 관리하기 위해서는 테이블 버킷 정책을 사용하고 테이블 수준 작업에 대한 액세스를 관리하기 위해서는 테이블 정책을 사용하는 것이 좋습니다. 여러 테이블에서 동일한 권한 세트를 정의하려는 경우 테이블 버킷 정책을 사용하는 것이 좋습니다.

주제

- 예 1: 계정의 버킷에서 PutBucketMaintenanceConfiguration에 대한 액세스를 허용하는 테이블 버킷 정책
- 예 2: hr 네임스페이스에 저장된 테이블에 대한 읽기(SELECT) 액세스를 허용하는 테이블 버킷 정책
- 예 3: 사용자가 테이블을 삭제하도록 허용하는 테이블 정책

예 1: 계정의 버킷에서 **PutBucketMaintenanceConfiguration**에 대한 액세스를 허용하는 테이블 버킷 정책

다음 테이블 버킷 정책 예시는 PutBucketMaintenanceConfiguration에 대한 액세스를 허용하여 IAM data steward가 계정의 모든 버킷에서 참조되지 않은 객체를 삭제할 수 있도록 허용합니다.

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

```
                "AWS": "arn:aws:iam::account_id:role/datasteward"
             },
             "Action": ["s3tables:PutTableBucketMaintenanceConfiguration"],
             "Resource":"arn:aws:s3tables:region:account_id:bucket/*"
     }]
 }
```

예 2: **hr** 네임스페이스에 저장된 테이블에 대한 읽기(SELECT) 액세스를 허용하는 테이블 버킷 정책

다음 테이블 버킷 정책 예시에서는 AWS 계정 ID 123456789012의 사용자인 Jane이 테이블 버킷의 hr 네임스페이스에 저장된 테이블에 액세스할 수 있도록 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Jane"
            },
            "Action": [
                "s3tables:GetTableData",
                "s3tables:GetTableMetadataLocation"
            ],
            "Resource": "arn:aws:s3tables:region:account_id:bucket/amzn-s3-demo-table-
bucket/table/*",
            "Condition": {
                "StringLike": {"s3tables:namespace": "hr"}
            }
        }
    ]
}
```

예 3: 사용자가 테이블을 삭제하도록 허용하는 테이블 정책

다음 테이블 정책 예시는 IAM 역할 data steward가 테이블을 삭제하도록 허용합니다.

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

```
       "Principal": {
         "AWS": "arn:aws:iam::account_id:role/datasteward"
        },
        "Action": [
           "s3tables:DeleteTable",
           "s3tables:UpdateTableMetadataLocation",
           "s3tables:PutTableData",
           "s3tables:GetTableMetadataLocation"

        ],
        "Resource":"arn:aws:s3tables:region:account_id:bucket/amzn-s3-demo-bucket1/
table/tableUUID"
     }]
}
```

S3 Tables에 대한 AWS 관리형 정책

AWS 관리형 정책은 AWS에 의해 생성되고 관리되는 독립 실행형 정책입니다. AWS 관리형 정책은 사용자, 그룹 및 역할에 권한 할당을 시작할 수 있도록 많은 일반 사용 사례에 대한 권한을 제공하도록 설계되었습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있기 때문에 특정 사용 사례에 대해 최소 권한을 부여하지 않을 수 있습니다. 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다.

AWS 관리형 정책에서 정의한 권한은 변경할 수 없습니다. 만약 AWS가 AWS 관리형 정책에 정의된 권한을 업데이트할 경우 정책이 연결되어 있는 모든 위탁자 ID(사용자, 그룹 및 역할)에도 업데이트가 적용됩니다. 새 AWS 서비스을(를) 시작하거나 새 API 작업을 기존 서비스에 이용하는 경우, AWS가 AWS 관리형 정책을 업데이트할 가능성이 높습니다.

자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하세요.

AWS 관리형 정책: AmazonS3TablesFullAccess

`AmazonTablesS3FullAccess` 정책을 IAM 보안 인증에 연결할 수 있습니다. 이 정책은 Amazon S3 Tables에 대한 전체 액세스를 허용하는 권한을 부여합니다.

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

```
                "s3tables:*"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS 관리형 정책: AmazonS3TablesReadOnlyAccess

AmazonS3TablesReadOnlyAccess 정책을 IAM 보안 인증에 연결할 수 있습니다. 이 정책은 Amazon S3 Tables에 대한 읽기 전용 액세스를 허용하는 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3tables:Get*",
                "s3tables:List*"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS 관리형 정책: AmazonS3TablesLakeFormationServiceRole

AmazonS3TablesLakeFormationServiceRole 정책을 IAM 보안 인증에 연결할 수 있습니다. 이 정책은 AWS Lake Formation 서비스 역할이 S3 Tables에 액세스할 수 있는 권한을 부여합니다. AWS KMS 권한은 Lake Formation이 암호화된 테이블에 액세스할 수 있도록 하는 데 사용됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PermissionsForS3ListTableBuckets",
            "Effect": "Allow",
            "Action": [
                "s3tables:ListTableBuckets"
            ],
            "Resource": [
                "*"
```

```
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "111122223333"
                }
            }
        },
        {
            "Sid": "DataAccessPermissionsForS3TablesResources",
            "Effect": "Allow",
            "Action": [
                "s3tables:CreateTableBucket",
                "s3tables:GetTableBucket",
                "s3tables:CreateNamespace",
                "s3tables:GetNamespace",
                "s3tables:ListNamespaces",
                "s3tables:DeleteNamespace",
                "s3tables:DeleteTableBucket",
                "s3tables:CreateTable",
                "s3tables:DeleteTable",
                "s3tables:GetTable",
                "s3tables:ListTables",
                "s3tables:RenameTable",
                "s3tables:UpdateTableMetadataLocation",
                "s3tables:GetTableMetadataLocation",
                "s3tables:GetTableData",
                "s3tables:PutTableData"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "111122223333"
                }
            }
        },
        {
            "Sid": "KMSDataAccessPermissionsForS3TablesResources",
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
```

```
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                        "s3.*.amazonaws.com"
                    ],
                    "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3tables:*:*:bucket/*/
table/*"
                },
                "StringEquals": {
                    "aws:ResourceAccount": "111122223333"
                }
            }
        },
        {
            "Sid": "KMSDescribeKeyAccessPermissionsForS3TablesResources",
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                        "s3tables.*.amazonaws.com"
                    ]
                },
                "StringEquals": {
                    "aws:ResourceAccount": "111122223333"
                }
            }
        }
    ]
}
```

AWS 관리형 정책에 대한 Amazon S3 Tables 업데이트

S3 Tables에서 이러한 변경 내용을 추적하기 시작한 이후부터 Amazon S3 Tables의 AWS 관리형 정책 업데이트에 대한 세부 정보를 봅니다.

변경 사항	설명	날짜
Amazon S3 Tables에 `AmazonS3TablesLakeFormationServiceRole` 를 추가했습니다.	S3 Tables에 `AmazonS3TablesLakeFormationServiceRole` 라는 새로운 AWS 관리형 정책을 추가했습니다. 이 정책은 Lake Formation 서비스 역할이 S3 Tables에 액세스할 수 있는 권한을 부여합니다.	2025년 5월 19일
Amazon S3 Tables에 `AmazonTablesS3FullAccess` 를 추가했습니다.	S3 Tables에 `AmazonTablesS3FullAccess` 라는 새로운 AWS 관리형 정책을 추가했습니다. 이 정책은 Amazon S3 Tables에 대한 전체 액세스를 허용하는 권한을 부여합니다.	2024년 12월 3일
Amazon S3 Tables에 `AmazonS3TablesReadOnlyAccess` 를 추가했습니다.	S3 Tables에 `AmazonS3TablesReadOnlyAccess` 라는 새로운 AWS 관리형 정책을 추가했습니다. 이 정책은 Amazon S3 Tables에 대한 읽기 전용 액세스를 허용하는 권한을 부여합니다.	2024년 12월 3일
Amazon S3 Tables에서 변경 사항 추적을 시작했습니다.	Amazon S3 Tables에서 AWS 관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2024년 12월 3일

SQL 의미 체계를 사용하여 액세스 권한 부여

테이블 및 테이블 버킷 정책에서 SQL 의미 체계를 사용하여 테이블에 권한을 부여할 수 있습니다. 사용할 수 있는 SQL 의미 체계의 예로는 `CREATE`, `INSERT`, `DELETE`, `UPDATE` 및 `ALTER`가 있습니다. 다음 표에는 사용자에게 권한을 부여하는 데 사용할 수 있는 SQL 의미 체계와 관련된 API 작업 목록이 나와 있습니다.

S3 Tables은 SQL 의미 체계를 사용하는 권한을 부분적으로 지원합니다. 예를 들어 `CreateTable` API는 테이블 버킷에 빈 테이블만 생성합니다. 테이블 스키마를 설정하려면 `UpdateTableMetadata`, `PutTableData` 및 `GetTableMetadataLocation`과 같은 추가 권한이 필요합니다. 또한 이러한 추가 권한을 제공한다는 것은 사용자에게 테이블에 행을 삽입할 수 있는 액세스 권한도 부여한다는 뜻입니다. SQL 의미 체계만을 기반으로 액세스를 제어하려면 S3 Tables과 통합된 AWS Lake Formation 또는 서드 파티 솔루션을 사용하는 것이 좋습니다.

테이블 수준 활동	IAM 작업		
SELECT	s3tables:GetTableData , s3tables:GetTableMetadataLocation		
CREATE	s3tables:CreateTable , s3tables:UpdateTableMetadataLocation , s3tables:PutTableData , s3tables:GetTableMetadataLocation ,		
INSERT	s3tables:UpdateTableMetadataLocation		

테이블 수준 활동	IAM 작업		
	, s3tables:PutTableData , s3tables:GetTableMetadataLocation		
UPDATE	s3tables:UpdateTableMetadataLocation , s3tables:PutTableData , s3tables:GetTableMetadataLocation		

테이블 수준 활동	IAM 작업		
ALTER,RENAME	s3tables:UpdateTableMetadataLocation , s3tables:PutTableData , s3tables:GetTableMetadataLocation , s3tables:RenameTable		
DELETE,DROP	s3tables:DeleteTable , s3tables:UpdateTableMetadataLocation , s3tables:PutTableData , s3tables:GetTableMetadataLocation		

S3 Tables의 VPC 연결

S3 Tables의 모든 테이블은 Apache Iceberg 형식이며 두 가지 유형의 S3 객체로 구성됩니다. 이러한 두 가지 유형의 객체는 데이터를 저장하는 데이터 파일, 그리고 서로 다른 시점의 데이터 파일에 대한 정보를 추적하는 메타데이터 파일입니다. 모든 테이블 버킷, 네임스페이스 및 테이블 작업(예: `CreateNamespace`, `CreateTable` 등)은 S3 Tables 엔드포인트(`s3tables.region.amazonaws.com`)를 통해 라우팅되며 데이터 및 메타데이터 파일을 읽거나 쓰는 모든 객체 수준 작업은 S3 서비스 엔드포인트(`s3.region.amazonaws.com`)를 통해 계속 라우팅됩니다.

S3 Tables에 액세스하기 위해 Amazon S3는 AWS PrivateLink를 사용하여 두 가지 유형의 VPC 엔드포인트, 즉 게이트웨이 엔드포인트와 인터페이스 엔드포인트를 지원합니다. 게이트웨이 엔드포인트는 AWS 네트워크를 통해 VPC에서 S3에 액세스하기 위해 라우팅 테이블에 지정하는 게이트웨이입니다. 인터페이스 엔드포인트는 프라이빗 IP 주소를 통해 VPC 내에서, 온프레미스에서 또는 VPC 피어링이나 AWS Transit Gateway를 사용하여 다른 AWS 리전의 VPC에서 Amazon S3로 요청을 라우팅함으로써 게이트웨이 엔드포인트의 기능을 확장합니다.

VPC에서 S3 Tables에 액세스하려면 두 개의 VPC 엔드포인트(S3용 엔드포인트와 S3 Tables용 엔드포인트)를 생성하는 것이 좋습니다. 게이트웨이 또는 인터페이스 엔드포인트를 생성하여 파일(객체) 수준 작업을 S3로 라우팅하고 인터페이스 엔드포인트를 생성하여 버킷 및 테이블 수준 작업을 S3 Tables로 라우팅할 수 있습니다. S3를 사용하여 파일 수준 요청에 VPC 엔드포인트를 생성하고 사용할 수 있습니다. 자세한 내용은 AWS PrivateLink 사용 설명서의 Gateway endpoints를 참조하세요.

AWS PrivateLink를 사용하여 S3 Tables용 엔드포인트를 생성하고 작업하는 방법에 대한 자세한 내용은 다음 주제를 참조하세요. VPC 인터페이스 엔드포인트를 생성하려면 AWS PrivateLink 설명서의 VPC 엔드포인트 생성을 참조하십시오.

주제

- S3 Tables용 VPC 엔드포인트 생성
- AWS CLI를 사용하여 엔드포인트를 통해 테이블 버킷 및 테이블에 액세스
- 쿼리 엔진을 사용할 때 VPC 네트워크 구성
- VPC 네트워크 내에서 S3 Tables에 대한 액세스 제한

S3 Tables용 VPC 엔드포인트 생성

VPC 엔드포인트를 생성할 때 S3 Tables에서는 두 가지 유형의 엔드포인트별 DNS 이름, 즉 리전용과 영역용 DNS 이름을 생성합니다.

- 리전 DNS 이름은 `VPCendpointID.s3tables.AWSregion.vpce.amazonaws.com` 형식입니다. 예를 들어, VPC 엔드포인트 ID `vpce-1a2b3c4d`의 경우, 생성된 DNS 이름은 `vpce-1a2b3c4d-5e6f.s3tables.us-east-1.vpce.amazonaws.com`과 비슷합니다.
- 영역 DNS 이름은 `VPCendpointID-AvailabilityZone.s3tables.AWSregion.vpce.amazonaws.com` 형식입니다. 예를 들어, VPC 엔드포인트 ID `vpce-1a2b3c4d-5e6f.`의 경우, 생성된 DNS 이름은 `vpce-1a2b3c4d-5e6f-us-east-1a.s3tables.us-east-1.vpce.amazonaws.com`과 비슷합니다.

  영역 DNS 이름에는 가용 영역이 포함됩니다. 아키텍처가 가용 영역을 분리하는 경우 영역 DNS 이름을 사용할 수 있습니다. 엔드포인트별 S3 DNS 이름은 S3 퍼블릭 DNS 도메인에서 확인할 수 있습니다.

프라이빗 DNS 옵션을 사용하여 VPC 엔드포인트를 통한 S3 트래픽 라우팅을 단순화하고 애플리케이션에서 사용할 수 있는 가장 저렴한 네트워크 경로를 활용할 수도 있습니다. 프라이빗 DNS는 S3 Tables의 퍼블릭 엔드포인트, 예를 들어 `s3tables.region.amazonaws.com`을 VPC의 프라이빗 IP에 매핑합니다. 프라이빗 DNS 옵션을 사용하면 인터페이스 엔드포인트의 엔드포인트별 DNS 이름을 사용하도록 S3 클라이언트를 업데이트하지 않고도 리전 S3 트래픽을 라우팅할 수 있습니다.

> **Note**
>
> AWS PrivateLink for Amazon S3는 Amazon S3 듀얼 스택 엔드포인트 사용을 지원하지 않습니다. 자세한 내용은 Amazon S3 API 참조의 Amazon S3 듀얼 스택 엔드포인트 사용을 참조하세요.

AWS CLI를 사용하여 엔드포인트를 통해 테이블 버킷 및 테이블에 액세스

AWS Command Line Interface(AWS CLI)를 사용하여 인터페이스 엔드포인트를 통해 테이블 버킷 및 테이블에 액세스할 수 있습니다. AWS CLI를 사용하면 `aws s3` 명령이 Amazon S3 엔드포인트를 통해 트래픽을 라우팅합니다. `aws s3tables` AWS CLI 명령은 Amazon S3 Tables 엔드포인트를 사용합니다.

`s3tables` VPC 엔드포인트의 예는 `vpce-0123456afghjipljw-nmopsqea.s3tables.region.vpce.amazonaws.com`입니다.

`s3tables` VPC 엔드포인트에는 버킷 이름이 포함되지 않습니다. `aws s3tables` AWS CLI 명령을 사용하여 `s3tables` VPC 엔드포인트에 액세스할 수 있습니다.

s3 VPC 엔드포인트의 예는 amzn-s3-demo-bucket.vpce-0123456afghjipljw-nmopsqea.s3.region.vpce.amazonaws.com입니다.

aws s3 AWS CLI 명령을 사용하여 s3 VPC 엔드포인트에 액세스할 수 있습니다.

AWS CLI 사용

AWS CLI를 사용하여 인터페이스 엔드포인트를 통해 테이블 버킷 및 테이블에 액세스하려면 --region 및 --endpoint-url 파라미터를 사용합니다. 테이블 버킷 및 테이블 수준 작업을 수행하려면 S3 Tables 엔드포인트 URL을 사용합니다. 객체 수준 작업을 수행하려면 Amazon S3 엔드포인트 URL을 사용합니다.

다음 예시에서 *### ## ## ###*를 실제 정보로 바꾸세요.

예 1: 엔드포인트 URL을 사용하여 계정의 테이블 버킷 나열

```
aws s3tables list-table-buckets --endpoint https://vpce-0123456afghjipljb-aac.s3tables.us-east-1.vpce.amazonaws.com –region us-east-1
```

예 2: 엔드포인트 URL을 사용하여 버킷의 테이블 나열

```
aws s3tables list-tables --table-bucket-arn arn:aws:s3tables:us-east-1:123456789301:bucket/amzn-s3-demo-bucket --endpoint
 https://vpce-0123456afghjipljb-aac.s3tables.us-east-1.vpce.amazonaws.com --region us-east-1
```

쿼리 엔진을 사용할 때 VPC 네트워크 구성

쿼리 엔진을 사용할 때 다음 단계에 따라 VPC 네트워크를 구성합니다.

1. 시작하려면 VPC를 생성하거나 업데이트하면 됩니다. 자세한 내용은 VPC 생성을 참조하세요.
2. S3 Tables로 라우팅되는 테이블 및 테이블 버킷 수준 작업의 경우 새 인터페이스 엔드포인트를 생성합니다. 자세한 내용은 인터페이스 VPC 엔드포인트를 사용하여 AWS 서비스에 액세스를 참조하세요.
3. Amazon S3로 라우팅되는 모든 객체 수준 작업의 경우 게이트웨이 엔드포인트 또는 인터페이스 엔드포인트를 생성합니다. 자세한 내용은 Create a gateway endpoint를 참조하세요.
4. 다음으로, 데이터 리소스를 구성하고 Amazon EMR 클러스터를 시작합니다. 자세한 내용은 Getting started with Amazon EMR을 참조하세요.
5. 그런 다음 VPC 엔드포인트에서 DNS 이름을 선택하여 추가 구성으로 Spark 애플리케이션을 제출할 수 있습니다. 예를 들어 spark.sql.catalog.ice_catalog.s3tables.endpoint 및

`https://interface-endpoint.s3tables.us-east-1.vpce.amazonaws.com`입니다. 자세한 내용은 Submit work to your Amazon EMR cluster를 참조하세요.

VPC 네트워크 내에서 S3 Tables에 대한 액세스 제한

리소스 기반 정책과 마찬가지로 테이블 및 테이블 버킷에 대한 액세스를 제어하는 엔드포인트 정책을 VPC 엔드포인트에 연결할 수 있습니다. 다음 예시에서 인터페이스 엔드포인트 정책은 특정 테이블 버킷으로만 액세스를 제한합니다.

```
{
    "Version": "2012-10-17",
    "Id": "Policy141511512309",
    "Statement": [{
        "Sid": "Access-to-specific-bucket-only",
        "Principal": "*",
        "Action": "s3tables:*",
        "Effect": "Allow",
        "Resource": [
            "arn:aws:s3tables:region:account_id:bucket/amzn-s3-demo-bucket",
            "arn:aws:s3tables:region:account_id:bucket/amzn-s3-demo-bucket/*"
        ]
    }]
}
```

S3 Tables의 보안 고려 사항 및 제한 사항

다음 목록에서는 S3 Tables에서 지원되지 않거나 제한되는 보안 및 액세스 제어 기능을 설명합니다.

- 퍼블릭 액세스 정책은 지원되지 않습니다. 사용자는 퍼블릭 액세스를 허용하도록 버킷 또는 테이블 정책을 수정할 수 없습니다.
- 테이블과 연결된 객체에 액세스하기 위한 미리 서명된 URL은 지원되지 않습니다.
- 테이블 버킷 및 테이블에 태그가 지원되지 않습니다. 따라서 속성 기반 액세스 제어 및 태그 기반 할당도 사용할 수 없습니다.
- HTTP를 통한 요청은 지원되지 않습니다. Amazon S3는 HTTP를 통해 이루어진 모든 요청에 대해 자동으로 HTTP 리디렉션으로 응답하여 요청을 HTTPS로 업그레이드합니다.
- REST API를 사용하여 액세스 포인트에 요청할 때는 AWS 서명 버전 4를 사용해야 합니다.
- 인터넷 프로토콜 버전 6(IPv6)을 통한 요청은 테이블 스토리지 엔드포인트를 통한 객체 수준 작업에만 지원되며 테이블 및 버킷 수준 작업에는 지원되지 않습니다.

- 테이블 버킷 및 테이블 액세스 정책은 크기가 20KB로 제한됩니다.

S3 Tables에서 AWS CloudTrail을 사용하여 로깅

Amazon S3는 사용자, 역할, 또는 AWS 서비스가 수행한 작업에 대한 레코드를 제공하는 서비스인 AWS CloudTrail과 통합됩니다. CloudTrail은 Amazon S3에 대한 모든 API 직접 호출을 이벤트로 캡처합니다. CloudTrail에서 수집한 정보를 사용하여 Amazon S3에 전송된 요청, 요청이 이루어진 IP 주소, 요청이 이루어진 시간, 추가 세부 정보를 확인할 수 있습니다. 지원되는 이벤트 활동이 Amazon S3에서 발생하면 해당 활동이 CloudTrail 이벤트에 기록됩니다. AWS CloudTrail 추적을 사용하여 S3 Tables에 대한 관리 이벤트 및 데이터 이벤트를 로깅할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서의 Amazon S3 CloudTrail 이벤트 및 What is AWS CloudTrail?을 참조하세요.

S3 Tables에 대한 CloudTrail 관리 이벤트

관리 이벤트는 AWS 계정의 리소스에 대해 수행되는 관리 작업에 대한 정보를 제공합니다.

기본적으로 CloudTrail은 S3 Tables의 관리 이벤트를 로깅합니다. S3 Tables에 대한 CloudTrail 관리 이벤트의 `eventsource`는 `s3tables.amazonaws.com`입니다. AWS 계정을 설정하면 CloudTrail 관리 이벤트가 기본적으로 활성화됩니다. 다음 관리 이벤트가 CloudTrail에 로깅됩니다.

- `CreateNamespace`
- `CreateTable`
- `CreateTableBucket`
- `DeleteNamespace`
- `DeleteTable`
- `DeleteTableBucket`
- `DeleteTableBucketPolicy`
- `DeleteTablePolicy`
- `GetNamespace`
- `GetTable`
- `GetTableBucket`
- `GetTableBucketMaintenanceConfiguration`
- `GetTableBucketPolicy`
- `GetTableMaintenanceConfiguration`
- `GetTableMaintenanceJobStatus`

- GetTableMetadataLocation
- GetTablePolicy
- ListNamespaces
- ListTableBuckets
- ListTables
- PutTableBucketMaintenanceConfiguration
- PutTableMaintenanceConfiguration
- PutBucketPolicy
- PutTablePolicy
- RenameTable
- UpdateTableMetadataLocation

CloudTrail 관리 이벤트에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 Logging management events를 참조하세요.

S3 Tables에 대한 CloudTrail 데이터 이벤트

데이터 이벤트 리소스에 대해 또는 리소스 내에서 수행된 리소스 작업에 대한 정보를 제공합니다. 기본적으로 CloudTrail 추적은 데이터 이벤트를 로깅하지 않지만 데이터 이벤트를 로깅하도록 추적을 구성할 수 있습니다.

CloudTrail에서 추적에 데이터 이벤트를 로깅할 때 리소스 유형을 선택하거나 지정합니다. S3 Tables에는 `AWS::S3Tables::Table` 및 `AWS::S3Tables::TableBucket`이라는 두 가지 리소스 유형이 있습니다.

다음 데이터 이벤트가 CloudTrail에 로깅됩니다.

- AbortMultipartUpload
- CompleteMultipartUpload
- CreateMultipartUpload
- GetObject
- HeadObject
- ListParts
- PutObject

- `UploadPart`

CloudTrail 데이터 이벤트에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 Logging data events를 참조하세요.

S3 Tables의 CloudTrail 이벤트에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- S3 Tables의 AWS CloudTrail 데이터 이벤트 로그 파일 예시

S3 Tables의 AWS CloudTrail 데이터 이벤트 로그 파일 예시

AWS CloudTrail 로그 파일에는 요청된 API 작업에 대한 정보, 작업 날짜 및 시간, 요청 파라미터 등이 포함됩니다. 이 주제에서는 S3 Tables의 CloudTrail 데이터 이벤트에 대한 로그 파일의 예를 제공합니다.

주제

- 예 - GetObject 데이터 이벤트에 대한 CloudTrail 로그 파일
- 예 - PutObject 데이터 이벤트에 대한 CloudTrail 로그 파일

예 - **`GetObject`** 데이터 이벤트에 대한 CloudTrail 로그 파일

다음은 `GetObject` API 작업을 보여주는 CloudTrail 로그 파일 예시입니다.

```
    {
        "eventVersion": "1.11",
        "userIdentity": {
          "type": "IAMUser",
          "principalId": "123456789012",
          "arn": "arn:aws:iam::111122223333:user/"myUserName",
          "accountId": "111122223333",
          "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
          "userName":"myUserName"
        },
        "eventTime": "2024-11-22T17:12:25Z",
        "eventSource": "s3tables.amazonaws.com",
        "eventName": "GetObject",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.0.2.0",
```

```
        "userAgent": "[aws-cli/2.18.5]",
        "requestParameters": {
            "Host": "tableWarehouseLocation.s3.us-east-1.amazonaws.com",
            "key": "product-info.json"
        },
        "responseElements":  null,
        "additionalEventData": {
            "SignatureVersion": "SigV4",
            "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
            "bytesTransferredIn": 0,
            "AuthenticationMethod": "AuthHeader",
            "xAmzId2": "q6xhNJYmhg",
            "bytesTransferredOut": 28441

          },
          "requestID": "07D681123BD12AED",
          "eventID": "f2b287f3-0df1-1234-a2f4-c4bdfed47657",
          "readOnly": true,
          "resources": [{
              "accountId": "111122223333",
              "type": "AWS::S3Tables::TableBucket",
               "ARN": "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
bucket1"
           }, {
              "accountId": "111122223333",
              "type": "AWS::S3Tables::Table",
              "ARN": "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
bucket/table/111aa1111-22bb-33cc-44dd-5555eee66ffff"

           }],
           "eventType": "AwsApiCall",
           "managementEvent": false,
           "recipientAccountId": "444455556666",
           "eventCategory": "Data",
           "tlsDetails": {
             "tlsVersion": "TLSv1.2",
             "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
             "clientProvidedHostHeader": "tableWarehouseLocation.s3.us-
east-1.amazonaws.com"
          }
    }
```

예 - **PutObject** 데이터 이벤트에 대한 CloudTrail 로그 파일

다음은 PutObject API 작업을 보여주는 CloudTrail 로그 파일 예시입니다.


```
{
        "eventVersion": "1.11",
        "userIdentity": {
          "type": "IAMUser",
          "principalId": "123456789012",
          "arn":  "arn:aws:iam::444455556666:user/"myUserName",
          "accountId": "444455556666",
          "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
          "userName":"myUserName"
        },
        "eventTime": "2024-11-22T17:12:25Z",
        "eventSource": "s3tables.amazonaws.com",
        "eventName": "PutObject",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "[aws-cli/2.18.5]",
        "requestParameters": {
            "Host": "tableWarehouseLocation.s3.us-east-1.amazonaws.com",
            "key": "product-info.json"
        },
        "responseElements":  {
            "x-amz-server-side-encryption": "AES256",
            "x-amz-version-id": "13zAFMdccAjt3MWd6ehxgCCCDRdkAKDw"
        },
        "additionalEventData": {
            "SignatureVersion": "SigV4",
            "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
            "bytesTransferredIn": 28441,
            "AuthenticationMethod": "AuthHeader",
            "xAmzId2": "q6xhCJYmhg",
            "bytesTransferredOut": 0

          },
          "requestID": "28d2faaf-1234-4649-997d-EXAMPLE72818",
          "eventID": "694d604a-d190-1234-0dd1-EXAMPLEe20c1",
          "readOnly": false,
          "resources": [{
              "accountId": "444455556666",
              "type": "AWS::S3Tables::TableBucket",
```

```
                "ARN": "arn:aws:s3tables:us-east-1:444455556666:bucket/amzn-s3-demo-
bucket1"
            }, {
               "accountId": "444455556666",
               "type": "AWS::S3Tables::Table",
               "ARN": "arn:aws:s3tables:us-east-1:444455556666:bucket/amzn-s3-demo-
bucket1/table/b89ec883-b1d9-4b37-9cd7-b86f590123f4"
            }],
            "eventType": "AwsApiCall",
            "managementEvent": false,
            "recipientAccountId": "111122223333",
            "eventCategory": "Data",
            "tlsDetails": {
              "tlsVersion": "TLSv1.2",
              "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
              "clientProvidedHostHeader": "tableWarehouseLocation.s3.us-
east-1.amazonaws.com"
             }
          }
```

Amazon S3의 액세스 제어

AWS에서 리소스는 작업할 수 있는 엔터티입니다. Amazon Simple Storage Service(S3)에서 버킷과 객체는 원래 Amazon S3 리소스입니다. 모든 S3 고객은 객체가 들어 있는 버킷을 가지고 있을 것입니다. S3에 새 기능이 추가되면서 리소스도 추가되었지만 모든 고객이 이러한 기능별 리소스를 사용하는 것은 아닙니다. Amazon S3 리소스에 대한 자세한 내용은 S3 리소스 섹션을 참조하세요.

모든 Amazon S3 리소스는 기본적으로 비공개입니다. 또한, 기본적으로 리소스를 만든 AWS 계정의 루트 사용자(리소스 소유자)와 해당 계정 내에서 필요한 권한을 가진 IAM 사용자는 직접 만든 리소스에 액세스할 수 있습니다. 리소스 소유자는 리소스에 액세스할 수 있는 다른 사용자와 리소스에서 다른 사용자가 수행할 수 있는 작업을 결정합니다. S3에는 다른 사람에게 내 S3 리소스에 대한 액세스 권한을 부여하는 데 사용할 수 있는 다양한 액세스 관리 도구가 있습니다.

다음 섹션에서는 S3 리소스에 대한 개요, 사용 가능한 S3 액세스 관리 도구, 각 액세스 관리 도구의 가장 좋은 사용 사례를 안내합니다. 이 섹션의 목록은 포괄적으로 다루는 것을 목표로 하며 모든 S3 리소스, 액세스 관리 도구 및 일반적인 액세스 관리 사용 사례를 포함합니다. 동시에 이 섹션들은 원하는 기술 세부 정보로 이어지는 디렉터리 역할을 하도록 설계되었습니다. 주제의 일부 내용을 잘 알고 있다면 필요한 섹션으로 건너뛰어도 됩니다.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

주제

- S3 리소스
- ID
- 액세스 관리 도구
- 작업
- 액세스 관리 사용 사례
- 액세스 관리 문제 해결

S3 리소스

원래 Amazon S3 리소스는 버킷과 버킷에 포함된 객체입니다. S3에 새 기능이 추가되면 새 리소스도 추가됩니다. 다음은 S3 리소스 및 해당 기능의 전체 목록입니다.

리소스 유형	Amazon S3 기능	설명
bucket	핵심 기능	버킷은 객체에 대한 컨테이너입니다. S3에 객체를 저장하려면 버킷을 생성한 다음 버킷에 객체를 하나 이상 업로드합니다. 자세한 내용은 Amazon S3 범용 버킷 생성, 구성 및 작업 섹션을 참조하세요.
object		객체는 파일과 해당 파일을 설명하는 모든 메타데이터일 수 있습니다. 객체가 버킷에 있으면 객체를 열고 다운로드하고 이동할 수 있습니다. 자세한 내용은 Amazon S3의 객체 작업 섹션을 참조하세요.
accesspoint	액세스 포인트	액세스 포인트는 GetObject 및 PutObject 같은 Amazon S3 객체 작업을 수행하는 데 사용할 수 있는 버킷에 연결된 네트워크 엔드포인트입니다. 각 액세스 포인트에는 기본 버킷에 연결된 버킷 정책과 함께 작동하는 구별된 권한, 네트워크 제어, 사용자 지정 액세스 포인트 정책이 있습니다. 가상 프라이빗 클라우드(VPC)의 요청만 수락하도록 액세스 포인트를 구성하거나 각 액세스 포인트에 대한 사용자 지정 퍼블릭 액세스 차단 설정을 구성할 수 있습니다. 자세한 내용은 액세스 포인트로 공유 데이터세트에 대한 액세스 관리 섹션을 참조하세요.
objectlambdaaccesspoint		객체 Lambda 액세스 포인트는 Lambda 함수와도 연결된 버킷의 액세스 포인트입니다. 객체 Lambda 액세스 포인트를 통해 자체 코드를 Amazon S3 GET, LIST, HEAD 요청에 추가하여 애플리케이션으로 데이터가 반환될 때 데이터를 수정 및 처리할 수 있습니다. 자세한 내용은 객체 Lambda 액세스 포인트 생성 섹션을 참조하세요.
multiregionaccesspoint		다중 리전 액세스 포인트는 애플리케이션이 여러 AWS 리전에 있는 Amazon S3 버킷의 요청을 이행하는 데 사용할 수 있는 글로벌 엔드포인트를 제공합니다. 다중 리전 액세스 포인트를 사용하여 단일 리전에서 사용되는 것과 동일한 아키텍처로 다중 리전 애플리케이션을 빌드하면 전 세계 어디에서나 해당 애플리케이션을 실행할 수 있습니다. 다중 리전 액세

리소스 유형	Amazon S3 기능	설명
		스 포인트 글로벌 엔드포인트에 대한 애플리케이션 요청은 혼잡이 심한 퍼블릭 인터넷을 통해 요청을 보내는 대신 AWS 글로벌 네트워크를 통해 가장 가까운 Amazon S3 버킷으로 자동으로 라우팅됩니다. 자세한 내용은 다중 리전 액세스 포인트로 다중 리전 트래픽 관리 섹션을 참조하세요.
`job`	S3 배치 작업	작업은 S3 배치 작업 기능의 리소스입니다. 지정한 Amazon S3 객체 목록에 대해 S3 배치 작업을 사용하여 대규모 배치 작업을 수행할 수 있습니다. Amazon S3는 배치 작업의 진행 상황을 추적하고 알림을 보내며 모든 작업에 대한 자세한 완료 보고서를 저장하여 감사 가능한 완전관리형 서버리스 경험을 제공합니다. 자세한 내용은 Batch Operations를 사용하여 대량으로 객체 작업 수행 섹션을 참조하세요.
`storagelensconfiguration`	S3 Storage Lens	S3 Storage Lens 구성은 조직 전체의 스토리지 지표 및 계정 전체의 사용자 데이터를 수집합니다. S3 Storage Lens는 관리자에게 조직 내 수백 또는 수천 개의 계정 전반에 걸친 객체 스토리지 사용 및 활동을 한 눈에 볼 수 있는 단일 보기와 다양한 집계 수준에서 인사이트를 생성하는 데 필요한 세부 정보를 제공합니다. 자세한 내용은 Amazon S3 스토리지 렌즈를 사용하여 스토리지 활동 및 사용량 평가 섹션을 참조하세요.
`storagelensgroup`		S3 Storage Lens 그룹은 객체 메타데이터를 기반으로 사용자 지정 필터를 사용하여 지표를 집계합니다. S3 Storage Lens 그룹을 사용하면 연령별 객체 분포, 가장 일반적인 파일 유형 등과 같은 데이터의 특성을 조사할 수 있습니다. 자세한 내용은 S3 Storage Lens 그룹을 사용하여 지표 필터링 및 집계 섹션을 참조하세요.

리소스 유형	Amazon S3 기능	설명
`accessgrantsinstance`	S3 Access Grants	S3 Access Grants 인스턴스는 사용자가 생성한 S3 권한 부여를 위한 컨테이너입니다. S3 Access Grants를 사용하면 계정 내 IAM ID, 다른 계정의 IAM ID(크로스 계정) 및 회사 디렉터리에서 AWS IAM Identity Center에 추가된 디렉터리 ID에 대한 권한 부여를 Amazon S3 데이터에 생성할 수 있습니다. S3 Access Grants에 대한 자세한 내용은 S3 Access Grants를 통한 액세스 관리 섹션을 참조하세요.
`accessgrantslocation`		Access Grants 위치는 버킷, 버킷 내 접두사 또는 S3 Access Grants 인스턴스에 등록한 객체입니다. S3 Access Grants 인스턴스 내에 위치를 등록해야 해당 위치에 대한 권한 부여를 생성할 수 있습니다. 그런 다음, S3 Access Grants를 사용하면 계정 내 IAM ID, 다른 계정의 IAM ID(크로스 계정) 및 회사 디렉터리에서 AWS IAM Identity Center에 추가된 디렉터리 ID에 대한 권한 부여를 버킷, 접두사 또는 객체에 생성할 수 있습니다. S3 Access Grants에 대한 자세한 내용은 S3 Access Grants를 통한 액세스 관리 섹션을 참조하세요.
`accessgrant`		Access Grant는 Amazon S3 데이터에 대한 개별 권한 부여입니다. S3 Access Grants를 사용하면 계정 내 IAM ID, 다른 계정의 IAM ID(크로스 계정) 및 회사 디렉터리에서 AWS IAM Identity Center에 추가된 디렉터리 ID에 대한 권한 부여를 Amazon S3 데이터에 생성할 수 있습니다. S3 Access Grants에 대한 자세한 내용은 S3 Access Grants를 통한 액세스 관리 섹션을 참조하세요.

버킷

Amazon S3 버킷에는 범용 버킷과 디렉터리 버킷이라는 두 가지 유형이 있습니다.

- 범용 버킷은 S3 버킷 본래의 유형이며 대부분의 사용 사례 및 액세스 패턴에 권장됩니다. 또한 범용 버킷을 사용하면 S3 Express One Zone을 제외한 모든 스토리지 클래스에 객체를 저장할 수 있습니다. S3 스토리지 클래스에 대한 자세한 정보는 Amazon S3 스토리지 클래스 이해 및 관리 섹션을 참조하세요.

- 디렉터리 버킷은 S3 Express One Zone 스토리지 클래스를 사용합니다. 이 스토리지 클래스는 애플리케이션이 성능에 민감하고 10밀리초 미만의 PUT 및 GET 지연 시간이 필요한 경우에 권장됩니다. 자세한 내용은 디렉터리 버킷 작업, S3 Express One Zone, IAM을 사용한 리전 엔드포인트 API 권한 부여 섹션을 참조하세요.

S3 리소스 분류

Amazon S3는 S3 리소스를 분류하고 정리하는 기능을 제공합니다. 리소스를 분류하면 리소스를 정리하는 데 유용할 뿐만 아니라 리소스 범주에 따라 액세스 관리 규칙을 설정할 수도 있습니다. 특히 접두사와 태그 지정은 액세스 관리 권한을 설정할 때 사용할 수 있는 두 가지 스토리지 정리 기능입니다.

> **Note**
>
> 다음 정보는 범용 버킷에 적용됩니다. 디렉터리 버킷은 태그 지정을 지원하지 않으며 접두사 제한이 있습니다. 자세한 내용은 IAM을 사용한 리전 엔드포인트 API 권한 부여 섹션을 참조하세요.

- 접두사 - Amazon S3의 접두사는 객체 키 이름의 앞에 있는 문자열로, S3 버킷에 저장된 객체를 정리하는 데 사용됩니다. 슬래시(/)와 같은 구분자를 사용하여 객체 키 이름 내의 접두사 끝을 표시할 수 있습니다. 예를 들어 `engineering/` 접두사로 시작하는 객체 키 이름이나 `marketing/campaigns/` 접두사로 시작하는 객체 키 이름이 있을 수 있습니다. 접두사 끝에 구분자(예: 슬래시 문자(/))를 사용하면 폴더 및 파일 이름 지정 규칙을 모방할 수 있습니다. 그러나 S3에서 접두사는 객체 키 이름의 일부입니다. 범용 S3 버킷에는 실제 폴더 계층 구조가 없습니다.

  Amazon S3는 접두사를 사용하여 객체 정리와 그룹화를 지원합니다. 접두사로 객체에 대한 액세스를 관리할 수도 있습니다. 예를 들어, 이름이 특정 접두사로 시작하는 객체만으로 액세스를 제한할 수 있습니다.

  자세한 내용은 접두어를 사용한 객체 구성 섹션을 참조하세요. S3 콘솔은 폴더라는 개념을 사용합니다. 범용 버킷에서 폴더는 객체 키 이름 앞에 붙는 접두사입니다. 자세한 내용은 Amazon S3 콘솔에서 폴더를 사용하여 객체 구성 섹션을 참조하세요.
- 태그 - 각 태그는 리소스에 할당하는 키 값 페어입니다. 예를 들어, `topicCategory=engineering` 태그를 사용하여 일부 리소스에 태그를 지정할 수 있습니다. 태그 지정을 통해 비용 할당, 분류 및 정리, 액세스 제어에 도움이 될 수 있습니다. 버킷 태그 지정은 비용 할당에만 사용됩니다. 정리 또는 액세스 제어를 위해 객체, S3 Storage Lens, 작업 및 S3 Access Grants에 태그를 지정할 수 있습니다. S3 Access Grants에서는 비용 할당을 위해서도 태그를 지정할 수 있습니다. 태그를 사용하여 리소스에 대한 액세스를 제어하는 예로, 특정 태그 또는 태그 조합이 있는 객체만 공유할 수 있습니다.

자세한 내용은 IAM 사용 설명서의 리소스 태그를 사용하여 AWS에 대한 액세스 제어를 참조하세요.

ID

Amazon S3에서 리소스 소유자는 리소스를 생성한 ID(예: 버킷 또는 객체)입니다. 기본적으로 리소스를 생성한 계정의 루트 사용자와 필요한 권한이 있는 계정 내의 IAM ID만 S3 리소스에 액세스할 수 있습니다. 리소스 소유자는 다른 ID에 자신의 S3 리소스에 대한 액세스 권한을 부여할 수 있습니다.

리소스를 소유하지 않은 ID는 해당 리소스에 대한 액세스를 요청할 수 있습니다. 리소스에 대한 요청은 인증되거나 인증되지 않습니다. 인증된 요청에는 요청 발신자를 인증하는 서명 값을 포함해야 하지만 인증되지 않은 요청에는 서명이 필요하지 않습니다. 인증된 사용자에게만 액세스 권한을 부여하는 것이 좋습니다. 요청 인증에 대한 자세한 내용은 Amazon S3 API 참조에서 요청을 참조하세요.

> **Important**
>
> AWS 계정의 루트 사용자 자격 증명을 사용하여 인증된 요청을 하지 않는 것이 좋습니다. 대신, IAM 역할을 생성하고 해당 역할에 모든 액세스 권한을 부여합니다. 이러한 역할이 부여된 사용자를 관리자 사용자라고 합니다. AWS 계정 루트 사용자 자격 증명 대신, 관리자 역할에 할당된 자격 증명을 사용하여 AWS와 상호 작용하고 버킷 생성, 사용자 생성, 권한 부여 등의 태스크를 수행할 수 있습니다. 자세한 내용은 AWS 일반 참조의 AWS 계정 root user credentials and IAM user credentials 및 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

Amazon S3의 데이터에 액세스하는 ID는 다음 중 하나일 수 있습니다.

AWS 계정 소유자

리소스를 만든 AWS 계정. 버킷을 생성한 계정을 예로 들 수 있습니다. 이 계정이 리소스를 소유합니다. 자세한 내용은 AWS 계정 루트 사용자를 참조하세요.

AWS 계정 소유자와 동일한 계정에 있는 IAM ID

S3 액세스가 필요한 새 팀원을 위한 계정을 설정할 때 AWS 계정 소유자는 AWS Identity and Access Management(IAM)를 사용하여 사용자, 그룹 및 역할을 생성할 수 있습니다. 그런 다음 AWS 계정 소유자가 이러한 IAM ID와 리소스를 공유할 수 있습니다. 또한 계정 소유자는 IAM ID에 부여할 권한을 지정하여 공유 리소스에서 수행할 수 있는 작업을 허용하거나 거부할 수 있습니다.

IAM ID를 통해 사용자가 공유 리소스에 액세스하기 전에 로그인 자격 증명을 입력하도록 요구하는 등 더욱 강력한 기능을 구현할 수 있습니다. IAM ID를 사용하면 IAM 다중 인증(MFA)의 한 형태를 구현하

여 강력한 ID 기반을 지원할 수 있습니다. IAM 모범 사례는 개별 사용자에게 권한을 부여하는 대신 액세스 관리를 위한 역할을 생성하는 것입니다. 개별 사용자에게 적절한 역할을 할당합니다. 자세한 내용은 IAM의 보안 모범 사례를 참조하세요.

기타 AWS 계정 소유자 및 해당 IAM ID(크로스 계정 액세스)

AWS 계정 소유자는 다른 AWS 계정 소유자 또는 다른 AWS 계정에 속한 IAM ID에게 리소스에 대한 액세스 권한을 부여할 수도 있습니다.

> **Note**
>
> 권한 위임 - AWS 계정이 리소스를 소유하는 경우 다른 AWS 계정에 해당 권한을 부여할 수 있습니다. 그런 다음 해당 계정은 권한 또는 권한의 하위 세트를 동일한 계정에 속한 사용자에게 위임할 수 있습니다. 이를 권한 위임이라고 합니다. 그러나 다른 계정으로부터 권한을 받는 계정은 다른 AWS 계정에 '크로스 계정'으로 해당 권한을 위임할 수 없습니다.

익명 사용자(퍼블릭 액세스)

AWS 계정 소유자는 리소스를 공개할 수 있습니다. 리소스를 공개하면 익명 사용자와 리소스를 공유하는 것입니다. 2023년 4월 이후 생성된 버킷은 이 설정을 변경하지 않는 한 기본적으로 모든 퍼블릭 액세스를 차단합니다. 퍼블릭 액세스를 차단하도록 버킷을 설정하고 인증된 사용자에게만 액세스 권한을 부여하는 것이 좋습니다. 퍼블릭 액세스 차단에 대한 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하십시오.

AWS 서비스

리소스 소유자는 다른 AWS 서비스에 Amazon S3 리소스에 대한 액세스 권한을 부여할 수 있습니다. 예를 들어, AWS CloudTrail 서비스에 로그 파일을 버킷에 쓸 수 있는 `s3:PutObject` 권한을 부여할 수 있습니다. 자세한 내용은 서비스에 대한 AWS 서비스에 액세스 권한 제공을 참조하세요.

기업 디렉터리 ID

리소스 소유자는 S3 Access Grants를 사용하여 회사 디렉터리의 사용자 또는 역할에 S3 리소스에 대한 액세스 권한을 부여할 수 있습니다. AWS IAM Identity Center에 회사 디렉터리를 추가하는 방법에 대한 자세한 정보는 What is IAM Identity Center?를 참고하세요.

버킷 또는 리소스 소유자

버킷 생성 및 객체 업로드에 사용하는 AWS 계정은 해당 리소스를 소유합니다. 버킷 소유자는 다른 AWS 계정(또는 다른 계정의 사용자)에 객체를 업로드할 수 있는 교차 계정 권한을 부여할 수 있습니다.

다른 계정이 버킷에 객체를 업로드하도록 버킷 소유자가 허용하면 기본적으로 버킷 소유자가 자신의 버킷에 업로드된 모든 객체를 소유합니다. 그러나 버킷 소유자 적용과 버킷 소유자 기본 설정 버킷 설정이 모두 비활성화된 경우 객체를 업로드하는 AWS 계정이 해당 객체를 소유하며 버킷 소유자는 다른 계정이 소유한 객체에 대한 권한을 갖지 않습니다. 단, 다음과 같은 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 여부와 관계 없이 모든 객체를 아카이빙하거나 아카이빙된 객체를 복원할 수 있습니다. 아카이빙이란 객체를 저장하는 데 사용하는 스토리지 클래스를 말합니다. 자세한 내용은 객체 수명 주기 관리 섹션을 참조하세요.

액세스 관리 도구

Amazon S3은 다양한 보안 기능 및 도구를 제공합니다. 다음은 이러한 기능 및 도구의 전체 목록입니다. 이러한 액세스 관리 도구가 모두 필요한 것은 아니지만 Amazon S3 리소스에 대한 액세스 권한을 부여하려면 하나 이상의 도구를 사용해야 합니다. 이러한 도구를 올바르게 적용하면 의도한 사용자만 리소스에 액세스하도록 할 수 있습니다.

가장 일반적으로 사용되는 액세스 관리 도구는 액세스 정책입니다. 액세스 정책은 AWS 리소스에 연결된 리소스 기반 정책일 수 있습니다(예: 버킷의 버킷 정책). 액세스 정책은 IAM 사용자, 그룹 또는 역할과 같은 AWS Identity and Access Management(IAM) ID에 연결된 ID 기반 정책일 수도 있습니다. AWS 계정, IAM 사용자, 그룹, 역할에 리소스에서 작업을 수행할 수 있는 권한을 부여하는 액세스 정책을 작성하세요. 예를 들어, 다른 AWS 계정에서 내 버킷에 객체를 업로드할 수 있도록 다른 계정에 PUT `Object` 권한을 부여할 수 있습니다.

액세스 정책은 누가 무엇에 액세스 할 수 있는지를 나타냅니다. 요청을 받으면 Amazon S3가 모든 액세스 정책을 평가해 권한 부여나 요청 거부를 결정합니다. Amazon S3의 해당 정책 평가에 대한 자세한 정보는 Amazon S3에서 요청에 권한을 부여하는 방법 단원을 참조하십시오.

Amazon S3에서 제공되는 액세스 관리 도구는 다음과 같습니다.

버킷 정책

Amazon S3 버킷 정책은 특정 버킷에 연결된 JSON 형식 AWS Identity and Access Management(IAM) 리소스 기반 정책입니다. 버킷 정책을 사용하여 다른 AWS 계정 또는 IAM ID에 버킷과 버킷에 포함된 객체에 대한 권한을 부여합니다. 버킷 정책을 사용하면 많은 S3 액세스 관리 사용 사례를 충족할 수 있습니다. 버킷 정책을 사용하면 버킷 액세스를 개인화하여 승인된 ID만 리소스에 액세스하고 리소스 내에서 작업을 수행하도록 할 수 있습니다. 자세한 내용은 Amazon S3의 버킷 정책 섹션을 참조하세요.

다음은 버킷 정책의 예입니다. 버킷 정책은 JSON 파일로 나타냅니다. 이 예시 정책은 버킷의 모든 객체에 IAM 역할 읽기 권한을 부여합니다. 여기에는 `amzn-s3-demo-bucket1` 버킷의 객체에 대해 `s3:GetObject` 작업(읽기 권한)을 허용하는 `BucketLevelReadPermissions` 문이 하나 있습니다. 이 정책은 IAM 역할을 `Principal`로 지정함으로써 이 역할을 가진 모든 IAM 사용자에게 액세스 권한을 부여합니다. 이 정책 예를 사용하려면 *`user input placeholders`*를 실제 정보로 바꾸세요.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid":"BucketLevelReadPermissions",
      "Effect":"Allow",
      "Principal": {
     "AWS": "arn:aws:iam::123456789101:role/s3-role"
      },
      "Action":["s3:GetObject"],
      "Resource":["arn:aws:s3:::amzn-s3-demo-bucket1/*"]
    }]
  }
```

> **Note**
>
> 정책을 생성할 때 `Principal` 요소에 와일드카드 문자(*)를 사용하면 모든 사용자가 Amazon S3 리소스에 액세스할 수 있게 되므로 와일드카드 문자는 사용하지 않도록 합니다. 대신, 버킷에 액세스할 수 있는 사용자 또는 그룹을 명시적으로 지정하거나 정책의 조건 절을 사용하여 충족해야 하는 조건을 나열하세요. 또한 사용자 또는 그룹의 작업에 와일드카드 문자를 포함하는 대신 해당되는 경우 사용자 또는 그룹에 특정 권한을 부여합니다.

자격 증명 기반 정책

ID 기반 또는 IAM 사용자 정책은 일종의 AWS Identity and Access Management(IAM) 정책입니다. ID 기반 정책은 AWS 계정의 IAM 사용자, 그룹 또는 역할에 연결된 JSON 형식의 정책입니다. ID 기반 정책을 사용하여 버킷이나 객체에 대한 IAM ID 액세스 권한을 부여할 수 있습니다. 내 계정으로 IAM 사용자, 그룹, 역할을 만들고 액세스 정책을 연결할 수 있습니다. 그런 다음 Amazon S3 리소스를 포함한 AWS 리소스에 대한 액세스 권한을 부여할 수 있습니다. 자세한 내용은 Amazon S3의 ID 기반 정책 섹션을 참조하세요.

다음은 ID 기반 정책의 예시입니다. 예시로 든 정책은 연결된 IAM 역할이 버킷과 버킷에 포함된 객체에 대해 6개의 서로 다른 Amazon S3 작업(권한)을 수행할 수 있도록 허용합니다. 이 정책을 계정의 IAM 역할에 연결하고 일부 IAM 사용자에게 역할을 할당하면 이 역할을 가진 사용자는 정책에 지정된 리소스(버킷)에서 이러한 작업을 수행할 수 있습니다. 이 정책 예를 사용하려면 *user input placeholders*를 실제 정보로 바꾸세요.

```
{
  "Version": "2012-10-17",
  "Statement": [
  {
    "Sid": "AssignARoleActions",
    "Effect": "Allow",
    "Action": [
   "s3:PutObject",
   "s3:GetObject",
   "s3:ListBucket",
   "s3:DeleteObject",
   "s3:GetBucketLocation"
    ],
    "Resource": [
   "arn:aws:s3:::amzn-s3-demo-bucket1/*",
   "arn:aws:s3:::amzn-s3-demo-bucket1"
    ]
    },
  {
    "Sid": "AssignARoleActions2",
    "Effect": "Allow",
    "Action": "s3:ListAllMyBuckets",
    "Resource": "*"
  }
  ]
}
```

S3 Access Grants

S3 Access Grants를 사용하여 회사 ID 디렉터리의 ID(예: Active Directory) 및 AWS Identity and Access Management(IAM) ID 모두에 대해 Amazon S3 데이터에 대한 액세스 권한을 생성할 수 있습니다. S3 Access Grants는 대규모로 데이터 권한을 관리하는 데 도움이 됩니다. 또한 S3 Access Grants는 AWS CloudTrail의 S3 데이터에 액세스하는 데 사용된 최종 사용자 ID와 애플리케이션을 로깅합니다. 이를 통해 S3 버킷의 데이터에 대한 모든 액세스에 대해 최종 사용자 ID까지 상세한 감사 기록이 제공됩니다. 자세한 내용은 S3 Access Grants를 통한 액세스 관리 섹션을 참조하세요.

액세스 포인트

Amazon S3 액세스 포인트는 S3의 공유 데이터세트를 사용하는 애플리케이션에 대한 대규모 데이터 액세스 관리를 간소화합니다. 액세스 포인트는 버킷에 연결된 명명된 네트워크 엔드포인트입니다. 액세스 포인트를 사용하여 객체 업로드 및 검색과 같은 S3 객체 작업을 수행할 수 있습니다. 버킷에는 최대 1만 개의 액세스 포인트가 연결될 수 있으며, 각 액세스 포인트에 대해 고유한 권한과 네트워크 제어를 적용하여 S3 객체에 대한 액세스를 세부적으로 제어할 수 있습니다. S3 액세스 포인트는 동일한 계정 또는 다른 신뢰할 수 있는 계정의 버킷과 연결할 수 있습니다. 액세스 포인트 정책은 기본 버킷 정책과 함께 평가되는 리소스 기반 정책입니다. 자세한 내용은 액세스 포인트로 공유 데이터세트에 대한 액세스 관리 섹션을 참조하세요.

ACL(액세스 제어 목록)

ACL은 피부여자와 그에 부여된 권한을 식별하는 권한 부여 목록입니다. ACL은 다른 AWS 계정에 기본적인 읽기 또는 쓰기 권한을 부여합니다. ACL은 Amazon S3별 XML 스키마를 사용합니다. ACL은 일종의 AWS Identity and Access Management(IAM) 정책입니다. 객체 ACL은 객체에 대한 액세스를 관리하는 데 사용되고, 버킷 ACL은 버킷에 대한 액세스를 관리하는 데 사용됩니다. 버킷 정책의 경우 전체 버킷에 적용되는 하나의 버킷 정책이 있지만 객체 ACL은 객체별로 지정됩니다. 각 객체에 대해 액세스를 개별적으로 제어해야 하는 드문 상황을 제외하고는 ACL을 비활성화한 채로 두는 것이 좋습니다. ACL 사용에 대한 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

> **Warning**
>
> Amazon S3의 최신 사용 사례 대부분은 ACL을 사용할 필요가 없습니다.

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>
```

객체 소유권

객체에 대한 액세스를 관리하려면 객체 소유자여야 합니다. 객체 소유권 버킷 수준 설정을 사용하여 버킷에 업로드되는 객체의 소유권을 제어할 수 있습니다. 또한 ACL을 활성화하는 데도 객체 소유권을 사용합니다. 기본적으로 객체 소유권은 버킷 소유자 적용 설정으로 설정되며 모든 ACL이 비활성화되어 있습니다. ACL이 비활성화되면 버킷 소유자는 버킷의 모든 객체를 소유하고 데이터에 대한 액세스를 독점적으로 관리합니다. 액세스를 관리하기 위해 버킷 소유자는 정책이나 다른 액세스 관리 도구(ACL 제외)를 사용합니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

객체 소유권에는 버킷에 업로드된 객체의 소유권을 제어하고 ACL을 활성화하는 데 사용할 수 있는 다음과 같은 세 가지 설정이 있습니다.

ACL 비활성화됨

- 버킷 소유자 적용(기본값) – ACL이 비활성화되고 버킷 소유자는 버킷의 모든 객체를 자동으로 소유하고 완전히 제어합니다. ACL은 S3 버킷의 데이터에 대한 권한에 영향을 주지 않습니다. 버킷은 정책을 독점적으로 사용하여 액세스 제어를 정의합니다.

ACL 활성화됨

- 버킷 소유자 기본(Bucket owner preferred) – 버킷 소유자가 `bucket-owner-full-control` 미리 제공 ACL을 사용하여 다른 계정이 버킷에 작성하는 새 객체를 소유하고 완전히 제어합니다.

- 객체 작성자(Object writer) – 객체를 업로드하는 AWS 계정은 객체를 소유하고 완전히 제어하며 ACL을 통해 다른 사용자에게 이에 대한 액세스 권한을 부여할 수 있습니다.

추가 모범 사례

다음 버킷 설정과 도구를 사용하여 전송 중 데이터와 저장된 데이터를 보호하는 것을 고려하세요. 이 두 가지 데이터 보호는 데이터의 무결성과 액세스 가능성을 유지하는 데 매우 중요합니다.

- 퍼블릭 액세스 차단 - 기본 버킷 수준 설정인 퍼블릭 액세스 차단을 비활성화하지 마세요. 이 설정은 기본적으로 데이터에 대한 퍼블릭 액세스를 차단합니다. 퍼블릭 액세스 차단에 대한 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하십시오.
- S3 버전 관리 - 데이터 무결성을 위해 S3 버전 관리 버킷 설정을 구현하여 업데이트 시 객체를 덮어쓰지 않고 객체의 버전을 만들 수 있습니다. 필요한 경우 S3 버전 관리를 사용하여 이전 버전을 보존, 검색 및 복원할 수 있습니다. S3 버전 관리에 대한 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하세요.
- S3 객체 잠금 - S3 객체 잠금은 데이터 무결성을 확보하기 위해 구현할 수 있는 또 다른 설정입니다. 이 기능은 WORM(Write Once Read Many) 모델을 구현하여 객체를 변경할 수 없게 저장할 수 있습니다. 객체 잠금에 대한 자세한 내용은 Object Lock으로 객체 잠금 섹션을 참조하세요.
- 객체 암호화 - Amazon S3는 전송 중 데이터와 저장된 데이터를 보호하는 몇 가지 객체 암호화 옵션을 제공합니다. 서버 측 암호화 - 서버 측 암호화는 데이터 센터의 디스크에 저장하기 전에 객체를 암호화하고 객체를 다운로드할 때 암호를 해독합니다. 요청을 인증하면 액세스 권한을 갖게 되며, 객체의 암호화 여부와 관계없이 액세스 방식에는 차이가 없습니다. 자세한 내용은 서버 측 암호화를 사용하여 데이터 보호 섹션을 참조하세요. S3는 새로 업로드한 객체를 기본적으로 암호화합니다. 자세한 내용은 Amazon S3 버킷에 대한 기본 서버 측 암호화 동작 설정 섹션을 참조하세요. 클라이언트 측 암호화는 Amazon S3로 보내기 전에 데이터를 암호화하는 것을 가리킵니다. 자세한 내용은 클라이언트측 암호화를 사용하여 데이터 보호 섹션을 참조하세요.
- 서명 방법 - 서명 버전 4는 HTTP로 전송된 AWS 요청에 인증 정보를 추가하는 프로세스입니다. 보안을 위해 대부분의 AWS 요청은 액세스 키 ID와 비밀 액세스 키로 구성된 액세스 키로 서명해야 합니다. 이 두 키는 일반적으로 보안 자격 증명이라고 합니다. 자세한 내용은 요청 인증(AWS 서명 버전 4) 및 서명 버전 4 서명 프로세스를 참조하십시오.

작업

S3 권한 및 조건 키의 전체 목록은 서비스 승인 참조에서 Actions, resources, and condition keys for Amazon S3를 참조하세요.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

작업

Amazon S3의 AWS Identity and Access Management(IAM) 작업은 S3 버킷 또는 객체에 수행할 수 있는 작업입니다. 이러한 작업을 ID에 부여하면 ID가 S3 리소스에서 작업을 수행할 수 있습니다. S3 작업의 예로는 버킷의 객체를 읽기 위한 `s3:GetObject`, 버킷에 객체를 쓰기 위한 `s3:PutObject` 등이 있습니다.

조건 키

작업 외에도 IAM 조건 키는 조건이 충족될 때만 액세스 권한을 부여하도록 제한됩니다. 조건 키는 선택 사항입니다.

> **Note**
>
> 리소스 기반 액세스 정책(예: 버킷 정책) 또는 ID 기반 정책에서 다음을 지정할 수 있습니다.
>
> - 정책 문의 `Action` 요소에 있는 작업 또는 일련의 작업입니다.
> - 정책 문의 `Effect` 요소에서 나열된 작업을 허용하도록 `Allow`를 지정하거나 나열된 작업을 차단하도록 `Deny`를 지정할 수 있습니다. 최소 권한의 원칙을 더 엄격히 지키기 위해 액세스 정책에서 `Effect` 요소의 `Deny` 문은 최대한 광범위해야 하며 `Allow` 명령문은 최대한 제한적이어야 합니다. `s3:*` 작업과 페어링된 `Deny` 효과 또한 정책 조건 명령문에 포함된 ID에게 옵트인 모범 사례를 적용하는 좋은 방법입니다.
> - 정책 문의 `Condition` 요소에 있는 조건 키입니다.

액세스 관리 사용 사례

Amazon S3는 리소스 소유자에게 액세스 권한 부여를 위한 다양한 도구를 제공합니다. 사용하는 S3 액세스 관리 도구는 공유하려는 S3 리소스, 액세스 권한을 부여받는 ID, 허용하거나 거부하려는 작업에 따라 달라집니다. S3 리소스에 대한 액세스를 관리하는 데 S3 액세스 관리 도구 중 하나 또는 도구의 조합을 사용할 수 있습니다.

대부분의 경우 액세스 정책을 사용하여 권한을 관리할 수 있습니다. 액세스 정책은 버킷과 같은 리소스 또는 다른 Amazon S3 리소스(S3 리소스)에 연결된 리소스 기반 정책일 수 있습니다. 액세스 정책은 계정 내 AWS Identity and Access Management(IAM) 사용자, 그룹 또는 역할에 연결된 ID 기반 정책일

수도 있습니다. 사용 사례에 버킷 정책이 더 적합할 수도 있습니다. 자세한 내용은 Amazon S3의 버킷 정책 섹션을 참조하세요. 또는 AWS Identity and Access Management(IAM)를 사용하여 AWS 계정 내에서 IAM 사용자, 그룹, 역할을 만들고 ID 기반 정책을 통해 이들의 버킷과 객체에 대한 액세스를 관리할 수 있습니다. 자세한 내용은 Amazon S3의 ID 기반 정책 섹션을 참조하세요.

이러한 액세스 관리 옵션을 탐색하는 데 도움이 되도록 다음은 각 S3 액세스 관리 도구에 대한 일반적인 Amazon S3 고객 사용 사례 및 권장 사항입니다.

AWS 계정 소유자가 동일한 계정 내의 사용자와만 버킷을 공유하려는 경우

모든 액세스 관리 도구가 이 기본 사용 사례를 충족할 수 있습니다. 이 사용 사례에는 다음과 같은 액세스 관리 도구를 사용하는 것이 좋습니다.

- 버킷 정책 - 버킷 하나 또는 소수의 버킷에 대한 액세스 권한을 부여하거나 버킷 액세스 권한이 버킷마다 비슷하다면 버킷 정책을 사용하세요. 버킷 정책을 사용하면 각 버킷에 대해 하나의 정책을 관리합니다. 자세한 내용은 Amazon S3의 버킷 정책 섹션을 참조하세요.
- ID 기반 정책 - 버킷마다 액세스 권한이 다른 버킷 수가 매우 많고 관리할 사용자 역할이 적은 경우 사용자, 그룹 또는 역할에 대해 IAM 정책을 사용할 수 있습니다. Amazon S3 리소스뿐 아니라 다른 AWS 리소스에 대한 사용자 액세스를 관리하는 경우에도 IAM 정책을 사용하는 것이 좋습니다. 자세한 내용은 예제 1: 버킷 소유자가 자신의 사용자에게 버킷 권한 부여 섹션을 참조하세요.
- S3 Access Grants - S3 Access Grants를 사용하여 S3 버킷, 접두사 또는 객체에 대한 액세스 권한을 부여할 수 있습니다. S3 Access Grants를 사용하면 대규모로 다양한 객체 수준 권한을 지정할 수 있지만, 버킷 정책의 크기는 20KB로 제한됩니다. 자세한 내용은 S3 Access Grants 시작하기 섹션을 참조하세요.
- 액세스 포인트 - 버킷에 연결된 명명된 네트워크 엔드포인트인 액세스 포인트를 사용할 수 있습니다. 버킷에는 최대 1만 개의 액세스 포인트가 연결될 수 있으며, 각 액세스 포인트에 대해 고유한 권한과 네트워크 제어를 적용하여 S3 객체에 대한 액세스를 세부적으로 제어할 수 있습니다. 자세한 내용은 액세스 포인트로 공유 데이터세트에 대한 액세스 관리 섹션을 참조하세요.

AWS 계정 소유자가 다른 AWS 계정의 사용자와 버킷 또는 객체를 공유하려는 경우(크로스 계정)

다른 AWS 계정에 권한을 부여하려면 버킷 정책 또는 다음과 같은 권장 액세스 관리 도구 중 하나를 사용해야 합니다. 이 사용 사례에는 ID 기반 액세스 정책을 사용할 수 없습니다. 크로스 계정 액세스 권한 부여에 대한 자세한 내용은 Amazon S3 버킷에 있는 객체에 크로스 계정 액세스를 제공하려면 어떻게 해야 하나요?를 참조하세요.

이 사용 사례에는 다음과 같은 액세스 관리 도구를 사용하는 것이 좋습니다.

- 버킷 정책 - 버킷 정책을 사용하면 각 버킷에 대해 하나의 정책을 관리합니다. 자세한 내용은 Amazon S3의 버킷 정책 섹션을 참조하세요.
- S3 Access Grants - S3 Access Grants를 사용하여 S3 버킷, 접두사 또는 객체에 대한 크로스 계정 권한을 부여할 수 있습니다. S3 Access Grants를 사용하면 대규모로 다양한 객체 수준 권한을 지정할 수 있지만, 버킷 정책의 크기는 20KB로 제한됩니다. 자세한 내용은 S3 Access Grants 시작하기 섹션을 참조하세요.
- 액세스 포인트 - 버킷에 연결된 명명된 네트워크 엔드포인트인 액세스 포인트를 사용할 수 있습니다. 버킷에는 최대 1만 개의 액세스 포인트가 연결될 수 있으며, 각 액세스 포인트에 대해 고유한 권한과 네트워크 제어를 적용하여 S3 객체에 대한 액세스를 세부적으로 제어할 수 있습니다. 자세한 내용은 액세스 포인트로 공유 데이터세트에 대한 액세스 관리 섹션을 참조하세요.

AWS 계정 소유자 또는 버킷 소유자가 객체 수준 또는 접두사 수준에서 권한을 부여해야 하며, 이러한 권한이 객체마다 또는 접두사마다 다른 경우

예를 들어 버킷 정책에서는 특정 키 이름 접두사를 공유하거나 특정 태그가 있는 버킷 내 객체에 대한 액세스 권한을 부여할 수 있습니다. 키 이름 접두사 `logs/`로 시작하는 객체에 대해 읽기 권한을 부여할 수 있습니다. 그러나 액세스 권한이 객체별로 다른 경우 버킷 정책을 사용하여 개별 객체에 권한을 부여하는 것은 실용적이지 않을 수 있습니다. 특히 버킷 정책의 크기가 20KB로 제한되어 있기 때문에 더욱 그렇습니다.

이 사용 사례에는 다음과 같은 액세스 관리 도구를 사용하는 것이 좋습니다.

- S3 Access Grants - S3 Access Grants를 사용하여 객체 수준 또는 접두사 수준의 권한을 관리할 수 있습니다. 버킷 정책과 달리 S3 Access Grants를 사용하면 대규모로 다양한 객체 수준 권한을 지정할 수 있습니다. 버킷 정책은 크기가 20KB로 제한됩니다. 자세한 내용은 S3 Access Grants 시작하기 섹션을 참조하세요.
- 액세스 포인트 - 액세스 포인트를 사용하여 객체 수준 또는 접두사 수준의 권한을 관리할 수 있습니다. 액세스 포인트는 버킷에 연결된 명명된 네트워크 엔드포인트입니다. 버킷에는 최대 1만 개의 액세스 포인트가 연결될 수 있으며, 각 액세스 포인트에 대해 고유한 권한과 네트워크 제어를 적용하여 S3 객체에 대한 액세스를 세부적으로 제어할 수 있습니다. 자세한 내용은 액세스 포인트로 공유 데이터세트에 대한 액세스 관리 섹션을 참조하세요.
- ACL - 액세스 제어 목록(ACL)은 사용하지 않는 것이 좋습니다. ACL은 객체당 권한 부여의 수가 100개로 제한되기 때문입니다. 하지만 ACL을 활성화하는 경우 버킷 설정에서 객체 소유권을 버킷 소유자 기본 설정과 ACL 활성화됨으로 설정하세요. 이 설정을 사용하면 `bucket-owner-full-control` 미리 제공 ACL로 작성된 새 객체는 객체 작성자가 아닌 버킷 소유자가 자동으로 소유합니

다. 그런 다음 XML 형식의 액세스 정책인 객체 ACL을 사용하여 다른 사용자에게 객체에 대한 액세스 권한을 부여할 수 있습니다. 자세한 내용은 ACL(액세스 제어 목록) 개요 섹션을 참조하세요.

AWS 계정 소유자 또는 버킷 소유자가 특정 계정 ID로만 버킷 액세스를 제한하려는 경우

이 사용 사례에는 다음과 같은 액세스 관리 도구를 사용하는 것이 좋습니다.

- 버킷 정책 - 버킷 정책을 사용하면 각 버킷에 대해 하나의 정책을 관리합니다. 자세한 내용은 Amazon S3의 버킷 정책 섹션을 참조하세요.
- 액세스 포인트 - 액세스 포인트는 버킷에 연결된 명명된 네트워크 엔드포인트입니다. 버킷에는 최대 1만 개의 액세스 포인트가 연결될 수 있으며, 각 액세스 포인트에 대해 고유한 권한과 네트워크 제어를 적용하여 S3 객체에 대한 액세스를 세부적으로 제어할 수 있습니다. 자세한 내용은 액세스 포인트로 공유 데이터세트에 대한 액세스 관리 섹션을 참조하세요.

AWS 계정 소유자 또는 버킷 소유자가 데이터에 액세스하는 각 사용자 또는 애플리케이션에 대해 서로 다른 엔드포인트를 원하는 경우

이 사용 사례에는 다음과 같은 액세스 관리 도구를 사용하는 것이 좋습니다.

- 액세스 포인트 - 액세스 포인트는 버킷에 연결된 명명된 네트워크 엔드포인트입니다. 버킷에는 최대 1만 개의 액세스 포인트가 연결될 수 있으며, 각 액세스 포인트에 대해 고유한 권한과 네트워크 제어를 적용하여 S3 객체에 대한 액세스를 세부적으로 제어할 수 있습니다. 각 액세스 포인트는 기본 버킷에 연결된 버킷 정책과 함께 작동하는 사용자 지정 액세스 포인트 정책을 적용합니다. 자세한 내용은 액세스 포인트로 공유 데이터세트에 대한 액세스 관리 섹션을 참조하세요.

AWS 계정 소유자 또는 버킷 소유자가 S3의 가상 프라이빗 클라우드(VPC) 엔드포인트에서 액세스를 관리해야 하는 경우

Amazon S3용 가상 프라이빗 클라우드(VPC) 엔드포인트는 VPC 내의 논리적 엔터티로서, S3에만 연결을 허용합니다. 이 사용 사례에는 다음과 같은 액세스 관리 도구를 사용하는 것이 좋습니다.

- VPC 환경의 버킷 - 버킷 정책을 사용하여 버킷에 액세스할 수 있는 사람과 이들이 액세스할 수 있는 VPC 엔드포인트를 제어할 수 있습니다. 자세한 내용은 버킷 정책을 사용하여 VPC 엔드포인트에서 액세스 제어 섹션을 참조하세요.

- 액세스 포인트 - 액세스 포인트를 설정하는 경우 액세스 포인트 정책을 사용할 수 있습니다. 가상 프라이빗 클라우드(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. 자세한 내용은 액세스 포인트로 공유 데이터세트에 대한 액세스 관리 섹션을 참조하세요.

AWS 계정 소유자 또는 버킷 소유자가 정적 웹 사이트를 공개적으로 사용 가능하도록 설정해야 하는 경우

S3를 사용하면 정적 웹 사이트를 호스팅하여 누구나 S3 버킷에서 호스팅되는 웹 사이트의 콘텐츠를 볼 수 있도록 허용할 수 있습니다.

이 사용 사례에는 다음과 같은 액세스 관리 도구를 사용하는 것이 좋습니다.

- Amazon CloudFront - 이 솔루션을 사용하면 Amazon S3 정적 웹 사이트를 공개적으로 호스팅하는 동시에 버킷 콘텐츠에 대한 모든 퍼블릭 액세스를 계속 차단할 수 있습니다. 네 개의 S3 퍼블릭 액세스 차단 설정을 모두 활성화하여 S3 정적 웹 사이트를 호스팅하려는 경우 Amazon CloudFront 원본 액세스 제어(OAC)를 사용할 수 있습니다. Amazon CloudFront는 안전한 정적 웹 사이트를 설정하는 데 필요한 기능을 제공합니다. 또한 이 솔루션을 사용하지 않는 Amazon S3 정적 웹 사이트는 HTTP 엔드포인트만 지원할 수 있습니다. CloudFront는 Amazon S3의 내구성 있는 스토리지를 사용하면서 HTTPS와 같은 추가 보안 헤더를 제공합니다. HTTPS는 일반적인 HTTP 요청을 암호화하고 일반적인 사이버 공격으로부터 보호함으로써 보안을 강화합니다.

  자세한 내용은 Amazon CloudFront 개발자 안내서의 안전한 정적 웹 사이트 시작하기를 참조하십시오.
- Amazon S3 버킷을 공개적으로 액세스할 수 있도록 설정 - 공개적으로 액세스되는 정적 웹 사이트로 사용할 버킷을 구성할 수 있습니다.

  > **Warning**
  >
  > 이 방법은 권장하지 않습니다. 대신 Amazon S3 정적 웹 사이트를 Amazon CloudFront의 일부로 사용하는 것이 좋습니다. 자세한 내용은 이전 옵션을 참조하거나 안전한 정적 웹 사이트 시작하기를 참조하세요.

  Amazon CloudFront를 사용하지 않고 Amazon S3 정적 웹 사이트를 만들려면 먼저 퍼블릭 액세스 차단 설정을 모두 비활성화해야 합니다. 정적 웹 사이트에 대해 버킷 정책을 작성할 때는 `s3:GetObject` 작업만 허용하고 `ListObject` 또는 `PutObject` 권한은 허용하지 않도록 합니다.

이렇게 하면 사용자가 버킷의 모든 객체를 보거나 자체 콘텐츠를 추가할 수 없습니다. 자세한 내용은 웹 사이트 액세스에 대한 권한 설정 섹션을 참조하세요.

AWS 계정 소유자 또는 버킷 소유자가 버킷의 콘텐츠를 공개하려는 경우

새 Amazon S3 버킷을 생성할 때 퍼블릭 액세스 차단 설정이 기본적으로 활성화됩니다. 퍼블릭 액세스 차단에 대한 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하십시오.

버킷에 대해 퍼블릭 액세스를 허용하지 않는 것이 좋습니다. 하지만 특정 사용 사례에서 허용해야만 한다면 이 사용 사례에 맞는 다음 액세스 관리 도구를 사용하는 것이 좋습니다.

- 퍼블릭 액세스 차단 설정 비활성화 - 버킷 소유자가 버킷에 대한 인증되지 않은 요청을 허용할 수 있습니다. 예를 들어 인증되지 않은 PUT Object 요청은 버킷에 퍼블릭 버킷 정책이 있거나 버킷 ACL이 퍼블릭 액세스 권한을 부여할 때 허용됩니다. 인증되지 않은 모든 요청은 다른 임의 AWS 사용자 또는 인증되지 않은 익명 사용자가 수행합니다. 이 사용자는 ACL에서 특정 정식 사용자 ID `65a011a29cdf8ec533ec3d1ccaae921c`로 표시됩니다. `WRITE` 또는 `FULL_CONTROL`에 객체를 업로드하면 모든 사용자 그룹 또는 익명 사용자에게 액세스 권한이 명확히 부여됩니다. 퍼블릭 버킷 정책 및 퍼블릭 ACL(액세스 제어 목록)에 대한 자세한 내용은 "퍼블릭"의 의미 섹션을 참조하십시오.

AWS 계정 소유자 또는 버킷 소유자가 액세스 정책 크기 제한을 초과한 경우

버킷 정책과 ID 기반 정책 모두 20KB의 크기 제한이 있습니다. 액세스 권한 요구 사항이 복잡한 경우 이 크기 제한을 초과할 수 있습니다.

이 사용 사례에는 다음과 같은 액세스 관리 도구를 사용하는 것이 좋습니다.

- 액세스 포인트 - 사용 사례에 맞는 경우 액세스 포인트를 사용하세요. 액세스 포인트의 경우 각 버킷에는 이름이 지정된 네트워크 엔드포인트가 여러 개 있으며, 각 엔드포인트에는 기본 버킷 정책에 맞는 고유한 액세스 포인트 정책이 있습니다. 하지만 액세스 포인트는 객체에서만 작동할 수 있고 버킷에서는 작동할 수 없으며 교차 리전 복제는 지원하지 않습니다. 자세한 내용은 액세스 포인트로 공유 데이터세트에 대한 액세스 관리 섹션을 참조하세요.
- S3 Access Grants - 버킷, 접두사 또는 객체에 대한 액세스 권한을 부여하는 매우 많은 권한 부여를 지원하는 S3 Access Grants를 사용하세요. 자세한 내용은 S3 Access Grants 시작하기 섹션을 참조하세요.

AWS 계정 소유자 또는 관리자 역할이 회사 디렉터리의 사용자 또는 그룹에 직접 버킷, 접두사 또는 객체 액세스 권한을 부여하려는 경우

AWS Identity and Access Management(IAM)를 통해 사용자, 그룹 및 역할을 관리하는 대신 회사 디렉터리를 AWS IAM Identity Center에 추가할 수 있습니다. 자세한 정보는 What is IAM Identity Center?를 참조하세요.

AWS IAM Identity Center에 회사 디렉터리를 추가한 후에는 다음 액세스 관리 도구를 사용하여 회사 디렉터리 ID에 S3 리소스에 대한 액세스 권한을 부여하는 것이 좋습니다.

- S3 Access Grants - 회사 디렉터리 내 사용자 또는 역할에 대한 액세스 권한 부여를 지원하는 S3 Access Grants를 사용하세요. 자세한 내용은 S3 Access Grants 시작하기 섹션을 참조하세요.

AWS 계정 소유자 또는 버킷 소유자가 CloudFront 로그를 S3 버킷에 쓸 수 있는 액세스 권한을 AWS CloudFront 서비스에 부여하고자 하는 경우

이 사용 사례에는 다음과 같은 액세스 관리 도구를 사용하는 것이 좋습니다.

- 버킷 ACL - 버킷 ACL이 권장되는 유일한 사용 사례는 Amazon CloudFront `awslogsdelivery` 계정과 같은 특정 AWS 서비스에 권한을 부여하는 것입니다. 배포를 생성 또는 업데이트하고 CloudFront 로깅을 활성화하면 CloudFront는 버킷 ACL을 업데이트하여 `awslogsdelivery` 계정에 버킷에 로그를 쓸 수 있는 `FULL_CONTROL` 권한을 부여합니다. 자세한 내용은 Amazon CloudFront 개발자 안내서의 표준 로깅 구성 및 로그 파일 액세스에 필요한 권한을 참조하십시오. 로그를 저장하는 버킷이 S3 객체 소유권에 대해 버킷 소유자 적용 설정을 사용하여 ACL을 비활성화하면 CloudFront에서 버킷에 로그를 쓸 수 없습니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

버킷 소유자가 다른 사용자가 버킷에 추가한 객체에 대해 완전한 제어를 유지하려는 경우

버킷 정책, 액세스 포인트 또는 S3 Access Grants를 사용하여 버킷에 객체를 업로드할 수 있는 액세스 권한을 다른 계정에 부여할 수 있습니다. 버킷에 대한 크로스 계정 액세스 권한을 부여한 경우 버킷에 업로드된 모든 객체의 완전한 소유권을 버킷 소유자가 갖도록 할 수 있습니다.

이 사용 사례에는 다음과 같은 액세스 관리 도구를 사용하는 것이 좋습니다.

- 객체 소유권 - 버킷 수준 설정인 객체 소유권을 기본값인 버킷 소유자 적용 설정으로 유지합니다.

액세스 관리 문제 해결

다음 리소스는 S3 액세스 관리 문제를 해결하는 데 도움이 될 수 있습니다.

액세스 거부(403 금지) 오류 문제 해결

액세스 거부 문제가 발생하는 경우 계정 수준 및 버킷 수준 설정을 확인하세요. 또한 액세스 권한을 부여하는 데 사용하는 액세스 관리 기능을 확인하여 정책, 설정 또는 구성이 올바른지 확인하세요. Amazon S3의 액세스 거부(403 금지) 오류의 일반적인 원인에 대한 자세한 내용은 Amazon S3의 액세스 거부(403 Forbidden) 오류 문제 해결 섹션을 참조하십시오.

IAM Access Analyzer for S3

리소스를 공개하고 싶지 않거나 리소스에 대한 퍼블릭 액세스를 제한하려는 경우 IAM Access Analyzer for S3를 사용할 수 있습니다. Amazon S3 콘솔에서 IAM Access Analyzer for S3를 사용하여 퍼블릭 또는 공유 액세스 권한을 부여하는 ACL(버킷 액세스 제어 목록), 버킷 정책 또는 액세스 포인트 정책이 있는 모든 버킷을 검토하세요. IAM Access Analyzer for S3는 인터넷상의 모든 사용자 또는 조직 외부의 AWS 계정을 포함한 다른 AWS 계정에 대한 액세스를 허용하도록 구성된 S3 버킷에 대한 알림을 제공합니다. 각 퍼블릭 버킷 또는 공유 버킷에 대해 퍼블릭 액세스 또는 공유 액세스의 수준과 소스를 보고하는 결과가 수신됩니다.

IAM Access Analyzer for S3에서는 하나의 작업으로 버킷에 대한 모든 퍼블릭 액세스를 차단할 수 있습니다. 특정 사용 사례를 지원하기 위해 퍼블릭 액세스가 필요하지 않은 경우 버킷에 대한 모든 퍼블릭 액세스를 차단하는 것이 좋습니다. 모든 퍼블릭 액세스를 차단하기 전에 애플리케이션이 퍼블릭 액세스 없이 계속 올바르게 작동하도록 하세요. 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하세요.

또한 버킷 수준 권한 설정을 검토하여 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.

Amazon S3 콘솔에서는 IAM Access Analyzer for S3를 추가 비용 없이 사용할 수 있습니다. IAM Access Analyzer for S3는 AWS Identity and Access Management(IAM) IAM Access Analyzer를 통해 제공됩니다. Amazon S3 콘솔에서 IAM Access Analyzer for S3를 사용하려면 IAM 콘솔을 방문하여 리전별로 IAM Access Analyzer에서 계정 수준 분석기를 생성해야 합니다.

IAM Access Analyzer for S3에 대한 자세한 내용은 IAM Access Analyzer for S3를 사용하여 버킷 액세스 검토 섹션을 참조하십시오.

로깅 및 모니터링

모니터링은 Amazon S3 솔루션의 신뢰성, 가용성 및 성능을 유지하는 데 중요합니다. 이를 통해 액세스 장애가 발생할 경우 보다 쉽게 디버깅할 수 있습니다. 로깅을 통해 사용자가 수신하는 오류와 요청이 언제 어떤 것인지 파악할 수 있습니다. AWS에서는 Amazon S3 리소스를 모니터링하기 위해 다음과 같이 몇 가지 도구를 제공합니다.

- AWS CloudTrail
- Amazon S3 액세스 로그
- AWS Trusted Advisor
- Amazon CloudWatch

자세한 내용은 Amazon S3의 로깅 및 모니터링 섹션을 참조하세요.

Amazon S3의 ID 및 액세스 관리

AWS Identity and Access Management(IAM)은 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 있도록 지원하는 AWS 서비스입니다. IAM 관리자는 어떤 사용자가 Amazon S3 리소스를 사용할 수 있도록 인증(로그인)되고 권한이 부여(권한 있음)될 수 있는지 제어합니다. IAM은 추가 비용 없이 사용할 수 있는 AWS 서비스입니다.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

> **Note**
>
> Amazon S3 Express One Zone 스토리지 클래스를 디렉터리 버킷과 함께 사용하는 방법에 대한 자세한 내용은 S3 Express One Zone 및 디렉터리 버킷 작업 섹션을 참조하세요.

주제

- 대상
- ID를 통한 인증
- 정책을 사용하여 액세스 관리
- Amazon S3에서 IAM을 사용하는 방법
- Amazon S3에서 요청에 권한을 부여하는 방법

- Amazon S3 API 작업에 필요한 권한
- Amazon S3의 정책 및 권한
- Amazon S3의 버킷 정책
- Amazon S3의 ID 기반 정책
- 정책을 사용하여 Amazon S3 리소스에 대한 액세스를 관리하는 연습
- Amazon S3 스토리지 렌즈에 대한 서비스 연결 역할 사용
- Amazon S3 자격 증명 및 액세스 문제 해결
- Amazon S3용 AWS 관리형 정책

대상

AWS Identity and Access Management(IAM)를 사용하는 방법은 Amazon S3에서 수행하는 작업에 따라 달라집니다.

서비스 사용자 – Amazon S3 서비스를 사용하여 작업을 수행하는 경우 필요한 자격 증명과 권한을 관리자가 제공합니다. 더 많은 Amazon S3 기능을 사용하여 작업을 수행한다면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방법을 이해하면 관리자에게 올바른 권한을 요청하는 데 도움이 됩니다. Amazon S3의 기능에 액세스할 수 없다면 Amazon S3 자격 증명 및 액세스 문제 해결 섹션을 참조하세요.

서비스 관리자 – 회사에서 Amazon S3 리소스를 책임지고 있다면 Amazon S3에 대한 모든 액세스 권한이 있을 것입니다. 서비스 관리자는 서비스 사용자가 액세스해야 하는 Amazon S3 기능과 리소스를 결정합니다. 그런 다음 IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하세요. 회사가 Amazon S3에서 IAM을 사용하는 방법에 대한 자세한 내용은 Amazon S3에서 IAM을 사용하는 방법 섹션을 참조하세요.

IAM 관리자 - IAM 관리자라면 Amazon S3에 대한 액세스 권한을 관리하는 정책을 작성하는 방법을 자세히 알고 싶을 것입니다. IAM에서 사용할 수 있는 Amazon S3 자격 증명 기반 정책 예시를 보려면 Amazon S3의 ID 기반 정책 섹션을 참조하세요.

ID를 통한 인증

인증은 ID 자격 증명을 사용하여 AWS에 로그인하는 방식입니다. AWS 계정 루트 사용자이나 IAM 사용자로, 또는 IAM 역할을 수임하여 인증(AWS에 로그인)받아야 합니다.

ID 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 ID로 AWS에 로그인할 수 있습니다. AWS IAM Identity Center (IAM Identity Center) 사용자, 회사의 Single Sign-On 인증, Google 또는 Facebook

보안 인증이 페더레이션 ID의 예제입니다. 페더레이션형 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 연동을 사용하여 AWS에 액세스하면 간접적으로 역할을 수임합니다.

사용자 유형에 따라AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. AWS에 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS 계정에 로그인하는 방법을 참조하세요.

AWS에 프로그래밍 방식으로 액세스하는 경우, AWS에서는 자격 증명을 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트(SDK) 및 명령줄 인터페이스(CLI)를 제공합니다. AWS 도구를 사용하지 않는 경우, 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 요청에 직접 서명하는 자세한 방법은 IAM 사용 설명서에서 API 요청용 AWS Signature Version 4를 참조하세요.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, AWS은(는) 다중 인증(MFA)을 사용하여 계정의 보안을 강화하는 것을 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 다중 인증 및 IAM 사용 설명서에서 IAM의 AWS 다중 인증을 참조하세요.

AWS 계정 루트 사용자

AWS 계정을(를) 생성할 때는 해당 계정의 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 단일 로그인 ID로 시작합니다. 이 ID는 AWS 계정루트 사용자라고 하며, 계정을 생성할 때 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명을 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 전체 작업 목록은 IAM 사용 설명서의 루트 사용자 보안 인증이 필요한 작업을 참조하세요.

페더레이션 자격 증명

가장 좋은 방법은 관리자 액세스가 필요한 사용자를 포함한 사용자가 ID 공급자와의 페더레이션을 사용하여 임시 자격 증명을 사용하여 AWS 서비스에 액세스하도록 요구합니다.

페더레이션 ID는 엔터프라이즈 사용자 디렉터리, 웹 ID 공급자, AWS Directory Service, Identity Center 디렉터리의 사용자 또는 자격 증명 소스를 통해 제공된 자격 증명을 사용하여 AWS 서비스에 액세스하는 모든 사용자입니다. 페더레이션 ID는 AWS 계정에 액세스할 때 역할을 수임하고 역할은 임시 자격 증명을 제공합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center을(를) 사용하는 것이 좋습니다. IAM Identity Center에서 사용자 및 그룹을 생성하거나 모든 AWS 계정 및 애플리케이션에서 사용하기 위해 고유한

ID 소스의 사용자 및 그룹 집합에 연결하고 동기화할 수 있습니다. IAM Identity Center에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 IAM Identity Center란 무엇인가요?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가지고 있는 AWS 계정 내 ID입니다. 가능하면 암호 및 액세스 키와 같은 장기 자격 증명이 있는 IAM 사용자를 생성하는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명이 필요한 특정 사용 사례가 있는 경우, 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 장기 보안 인증이 필요한 사용 사례의 경우, 정기적으로 액세스 키 교체를 참조하세요.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 자격 증명을 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세한 내용은 IAM 사용 설명서에서 IAM 사용자 사용 사례를 참조하세요.

IAM 역할

IAM 역할은 특정 권한을 가지고 있는 AWS 계정 계정 내 ID입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. AWS Management Console에서 일시적으로 IAM 역할을 수임하려면 사용자에서 IAM 역할로 전환(콘솔)하면 됩니다. AWS CLI 또는 AWS API 작업을 직접적으로 호출하거나 사용자 지정 URL을 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 페더레이션 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 ID가 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 관련 역할에 대한 자세한 내용은 IAM 사용 설명서의 Create a role for a third-party identity provider (federation)를 참조하세요. IAM Identity Center를 사용하는 경우, 권한 집합을 구성합니다. 인증 후 ID가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 집합을 IAM의 역할과 연관짓습니다. 권한 집합에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 집합을 참조하세요.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 작업에 대한 다양한 권한을 임시로 받을 수 있습니다.

- 교차 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 하지만 일부 AWS 서비스을(를) 사용하면 리소스에 정책을 직접 연결할 수 있습니다(역할을 프록시로서 사용하는 대신) . 교차 계정 액세스에 대한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 교차 계정 리소스 액세스를 참조하세요.
- 교차 서비스 액세스 - 일부 AWS 서비스는 다른 AWS 서비스의 특성을 사용합니다. 예를 들어, 서비스에서 호출하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 직접적으로 호출하는 위탁자의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
  - 전달 액세스 세션(FAS) - IAM 사용자 또는 역할을 사용하여 AWS에서 작업을 수행하는 사람은 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우, 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 AWS 서비스를 직접 호출하는 위탁자의 권한과 요청하는 AWS 서비스를 함께 사용하여 다운스트림 서비스에 대한 요청을 수행합니다. FAS 요청은 서비스에서 완료를 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 받은 경우에만 이루어집니다. 이 경우, 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.
  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 Create a role to delegate permissions to an AWS 서비스를 참조하세요.
  - 서비스 연결 역할 – 서비스 연결 역할은 AWS 서비스에 연결된 서비스 역할의 한 유형입니다. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 링크 역할은 AWS 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집할 수는 없습니다.
- Amazon EC2에서 실행 중인 애플리케이션 – IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS역할을 할당하고 해당 역할을 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로필을 생성합니다. 인스턴스 프로필에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

정책을 사용하여 액세스 관리

정책을 생성하고 AWS ID 또는 리소스에 연결하여 AWS에서 내 액세스를 제어합니다. 정책은 ID 또는 리소스와 연결될 때 해당 권한을 정의하는 AWS의 객체입니다. AWS는 보안 주체(사용자, 루트 사용자

또는 역할 세션)가 요청을 보낼 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은 AWS에 JSON 문서로 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용 설명서의 JSON 정책 개요를 참조하세요.

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

기본적으로, 사용자 및 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole` 작업을 허용하는 정책이 있다고 가정합니다. 해당 정책이 있는 사용자는 AWS Management Console, AWS CLI 또는 AWS API에서 역할 정보를 가져올 수 있습니다.

ID 기반 정책

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

ID 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은 AWS 계정에 속한 다수의 사용자, 그룹 및 역할에 독립적으로 추가할 수 있는 정책입니다. 관리형 정책에는 AWS 관리형 정책과 고객 관리형 정책이 포함되어 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책 및 인라인 정책 중에서 선택을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 위탁자가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 위탁자를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는 AWS 서비스가 포함될 수 있습니다.

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

액세스 제어 목록(ACL)

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

Amazon S3, AWS WAF 및 VPC는 ACL을 지원하는 대표적인 서비스입니다. ACL에 관한 자세한 내용은 Amazon Simple Storage Service 개발자 가이드의 액세스 제어 목록(ACL) 개요를 참조하세요.

기타 정책 타입

AWS는 비교적 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 ID 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 객체의 ID 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용 설명서의 IAM 엔티티에 대한 권한 경계를 참조하세요.
- 서비스 제어 정책(SCP) – SCP는 AWS Organizations에서 조직 또는 조직 단위(OU)에 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations는 기업이 소유하는 여러 개의 AWS 계정을 그룹화하고 중앙에서 관리하기 위한 서비스입니다. 조직에서 모든 특성을 활성화할 경우, 서비스 제어 정책(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 각 AWS 계정 루트 사용자을 비롯하여 멤버 계정의 엔티티에 대한 권한을 제한합니다. 조직 및 SCP에 대한 자세한 내용은 AWS Organizations 사용 설명서에서 Service control policies을 참조하세요.
- 리소스 제어 정책(RCP) - RCP는 소유한 각 리소스에 연결된 IAM 정책을 업데이트하지 않고 계정의 리소스에 대해 사용 가능한 최대 권한을 설정하는 데 사용할 수 있는 JSON 정책입니다. RCP는 멤버 계정의 리소스에 대한 권한을 제한하며 조직에 속하는지 여부에 관계없이 AWS 계정 루트 사용자를 포함한 ID에 대한 유효 권한에 영향을 줄 수 있습니다. RCP를 지원하는 AWS 서비스 목록을 포함하여 Organizations 및 RCP에 대한 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCP)을 참조하세요.
- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 ID 기반 정책의 교차와 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

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

Amazon S3에서 IAM을 사용하는 방법

IAM을 사용하여 Amazon S3에 대한 액세스를 관리하기 전에 Amazon S3에서 사용할 수 있는 IAM 기능에 대해 알아봅니다.

Amazon S3에서 사용할 수 있는 IAM 기능

IAM 기능	Amazon S3 지원
ID 기반 정책	예
리소스 기반 정책	예
정책 작업	예
정책 리소스	예
정책 조건 키(서비스별)	예
ACL	예
ABAC(정책 내 태그)	부분
임시 자격 증명	예
전달 액세스 세션(FAS)	예
서비스 역할	예
서비스 연결 역할	부분

Amazon S3 및 기타 AWS 서비스가 대부분의 IAM 기능과 작동하는 방법을 개괄적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

Amazon S3의 ID 기반 정책

ID 기반 정책 지원: 예

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

IAM ID 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. ID 기반 정책에서는 위탁자가 연결된 사용자 또는 역할에 적용되므로 위탁자를 지정할 수 없습니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

Amazon S3의 ID 기반 정책 예시

Amazon S3 ID 기반 정책 예시를 보려면 Amazon S3의 ID 기반 정책 섹션을 참조하세요.

Amazon S3 내의 리소스 기반 정책

리소스 기반 정책 지원: 예

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 위탁자가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 위탁자를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는 AWS 서비스가 포함될 수 있습니다.

교차 계정 액세스를 활성화하려는 경우, 전체 계정이나 다른 계정의 IAM 개체를 리소스 기반 정책의 위탁자로 지정할 수 있습니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 트러스트 관계 설정의 절반밖에 되지 않는다는 것을 유념하세요. 보안 주체와 리소스가 서로 다른 AWS 계정에 있는 경우, 신뢰할 수 있는 계정의 IAM 관리자는 보안 주체 엔터티(사용자 또는 역할)에도 리소스 액세스 권한을 부여해야 합니다. 엔터티에 ID 기반 정책을 연결하여 권한을 부여합니다. 하지만 리소스 기반 정책이 동일 계정의 위탁자에 액세스를 부여하는 경우, 추가 자격 증명 기반 정책이 필요하지 않습니다. 자세한 내용은 IAM 사용 설명서의 교차 계정 리소스 액세스를 참조하세요.

Amazon S3 서비스는 버킷 정책, 액세스 포인트 정책 및 액세스 권한을 지원합니다.

- 버킷 정책은 Amazon S3 버킷에 연결되는 리소스 기반 정책입니다. 버킷 정책은 버킷에서 작업을 수행할 수 있는 보안 주체를 정의합니다.
- 액세스 포인트 정책은 기본 버킷 정책과 함께 평가되는 리소스 기반 정책입니다.
- Access Grants는 Amazon S3의 데이터에 대한 액세스 권한을 접두사, 버킷 또는 객체별로 정의하는 간소화된 모델입니다. S3 Access Grants에 대한 자세한 내용은 S3 Access Grants를 통한 액세스 관리 섹션을 참조하세요.

버킷 정책의 보안 주체

`Principal` 요소는 리소스에 대한 액세스가 허용 또는 거부된 사용자, 계정, 서비스 또는 기타 엔터티를 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::AccountIDWithoutHyphens:root"}
```

```
"Principal":{"AWS":
["arn:aws:iam::AccountID1WithoutHyphens:root","arn:aws:iam::AccountID2WithoutHyphens:root"]}
```

> **Note**
>
> 위의 예시는 루트 사용자에게 권한을 부여하며, 루트 사용자는 계정 수준에 권한을 위임합니다. 그러나 IAM 정책은 계정의 특정 역할 및 사용자에게 여전히 필요합니다.

IAM 사용자에게 권한 부여

계정 내의 IAM 사용자에게 권한을 부여하려면 `"AWS":"user-ARN"` 이름-값 페어를 제공해야 합니다.

```
"Principal":{"AWS":"arn:aws:iam::account-number-without-hyphens:user/username"}
```

단계별 지침을 제공하는 자세한 예는 예제 1: 버킷 소유자가 자신의 사용자에게 버킷 권한 부여 및 예제 3: 버킷 소유자가 자신이 소유하지 않는 객체에 대한 권한 부여 단원을 참조하십시오.

> **Note**
>
> 버킷 정책을 업데이트한 후 IAM 자격 증명이 삭제되면 버킷 정책은 ARN 대신 principal 요소에 고유 식별자를 표시합니다. 이러한 고유 ID는 절대 재사용되지 않으므로 모든 정책 설명에서 고유 식별자가 있는 보안 주체를 안전하게 제거할 수 있습니다. 고유 식별자에 대한 자세한 내용은 IAM 사용 설명서의 IAM 식별자를 참조하십시오.

익명 권한 부여

> **Warning**
>
> Amazon 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.

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.

```
"Principal":"*"
```

```
"Principal":{"AWS":"*"}
```

리소스 기반 정책에서 `"Principal": "*"`을(를) `Allow` 효과와 함께 사용하면 사용자 리소스에 액세스하기 위해 AWS에 로그인하지 않아도 누구나 허용할 수 있습니다.

리소스 기반 정책에서 `"Principal" : { "AWS" : "*" }`를 `Allow` 효과와 함께 사용하면 동일한 파티션의 모든 계정에 있는 모든 루트 사용자, IAM 사용자, 수임된 역할 세션 또는 페더레이션 사용자가 리소스에 액세스할 수 있습니다.

익명 사용자의 경우 이들 두 메서드는 동일합니다. 자세한 내용은 IAM 사용 설명서의 모든 보안 주체를 참조하십시오.

보안 주체 이름이나 ARN의 일부를 나타내기 위해 와일드카드를 사용할 수 없습니다.

> **Important**
>
> AWS 액세스 제어 정책에서 위탁자 "*" 및 {"AWS": "*"}는 동일하게 작동합니다.

리소스 권한 제한

또한 리소스 정책을 사용하여 IAM 보안 주체가 사용할 수 있을 리소스에 대한 액세스를 제한할 수 있습니다. 액세스를 방지하려면 Deny 명령문을 사용합니다.

다음은 보안 전송 프로토콜을 사용하지 않는 경우 액세스를 차단하는 예시입니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyBucketAccessIfSTPNotUsed",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "false"
        }
      }
    }
  ]
}
```

이 방법을 사용하여 특정 계정이나 보안 주체에 대한 액세스를 거부하는 대신 `"Principal": "*"`을 사용하여 이 제한이 모든 사람에게 적용되도록 하는 것이 이 정책의 모범 사례입니다.

CloudFront URL을 통한 액세스 요구

사용자가 Amazon S3 URL 대신 CloudFront URL만 사용하여 Amazon S3 콘텐츠에 액세스하도록 요구할 수 있습니다. 이렇게 하려면 CloudFront 원본 액세스 제어(OAC)를 만듭니다. 그런 다음 S3 데이터에 대한 권한을 변경하세요. 버킷 정책에서 다음과 같이 CloudFront를 보안 주체로 설정할 수 있습니다.

```
"Principal":{"Service":"cloudfront.amazonaws.com"}
```

정책의 Condition 요소를 사용하여 S3 오리진이 포함된 CloudFront 배포를 위한 요청인 경우에만 CloudFront에서 버킷에 액세스하도록 허용합니다.

```
        "Condition": {
           "StringEquals": {
              "AWS:SourceArn":
"arn:aws:cloudfront::111122223333:distribution/CloudFront-distribution-ID"
           }
        }
```

CloudFront URL을 통한 S3 액세스 요구에 대한 자세한 내용은 Amazon CloudFront 개발자 안내서의 Amazon Simple Storage Service 오리진에 대한 액세스 제한을 참조하세요. Amazon CloudFront 사용의 보안 및 개인 정보 보호 이점에 대한 자세한 내용은 콘텐츠에 대한 보안 액세스 구성 및 액세스 제한을 참조하세요.

Amazon S3용 리소스 기반 정책 예시

- Amazon S3 버킷의 정책 예시를 보려면 Amazon S3의 버킷 정책 섹션을 참조하세요.
- 액세스 포인트의 정책 예시를 보려면 액세스 포인트를 사용하도록 IAM 정책 구성 섹션을 참조하세요.

Amazon S3의 정책 작업

정책 작업 지원: 예

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

JSON 정책의 Action 요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 일반적으로 정책 작업의 이름은 연결된 AWSAPI 작업의 이름과 동일합니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하세요.

다음은 S3 API 작업과 필수 정책 작업 간의 다양한 유형 매핑 관계 유형을 보여줍니다.

- 이름이 동일한 일대일 매핑. 예를 들어, PutBucketPolicy API 작업을 사용하려면 s3:PutBucketPolicy 정책 작업이 필요합니다.

- 이름이 다른 일대일 매핑. 예를 들어, `ListObjectsV2` API 작업을 사용하려면 `s3:ListBucket` 정책 작업이 필요합니다.
- 일대다 매핑. 예를 들어, `HeadObject` API 작업을 사용하려면 `s3:GetObject`가 필요합니다. 또한 S3 객체 잠금을 사용하고 객체의 법적 보존 상태 또는 보존 설정을 가져오려는 경우 `HeadObject` API 작업을 사용하기 전에 해당 `s3:GetObjectLegalHold` 또는 `s3:GetObjectRetention` 정책 작업도 필요합니다.
- 다대일 매핑. 예를 들어, `ListObjectsV2` 또는 `HeadBucket` API 작업을 사용하려면 `s3:ListBucket` 정책 작업이 필요합니다.

정책에 사용하기 위한 Amazon S3 작업의 목록을 보려면 서비스 승인 참조의 Amazon S3에서 정의한 작업을 참조하세요. Amazon S3 API 작업의 전체 목록은 Amazon Simple Storage Service API 참조에서 Amazon S3 API Actions를 참조하세요.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

Amazon S3의 정책 작업에서는 작업 앞에 다음과 같은 접두사를 사용합니다.

```
s3
```

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "s3:action1",
      "s3:action2"
         ]
```

버킷 작업

버킷 작업은 버킷 리소스 유형에서 작동하는 S3 API 작업입니다. 예를 들면 `CreateBucket`, `ListObjectsV2` 및 `PutBucketPolicy`입니다. 버킷 작업에 대한 S3 정책 작업을 위해서는 버킷 정책 또는 IAM 자격 증명 기반 정책의 `Resource` 요소가 다음 예시 형식의 S3 버킷 유형 Amazon 리소스 이름(ARN) 식별자여야 합니다.

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
```

다음 버킷 정책은 *12345678901* 계정이 있는 *Akua* 사용자에게 ListObjectsV2 API 작업을 수행하고 S3 버킷에 객체를 나열할 수 있는 `s3:ListBucket` 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to list objects in the bucket",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
            "Action": [
            "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
        }
    ]
}
```

범용 버킷의 액세스 포인트에 대한 정책의 버킷 작업

범용 버킷의 액세스 포인트 정책에서 부여된 권한은 기본 버킷이 동일한 권한을 허용하는 경우에만 유효합니다. S3 액세스 포인트를 사용하는 경우, 버킷의 액세스 제어를 액세스 포인트에 위임하거나 액세스 포인트 정책의 동일한 권한을 기본 버킷 정책에 추가해야 합니다. 자세한 내용은 액세스 포인트를 사용하도록 IAM 정책 구성 섹션을 참조하세요. 액세스 포인트 정책에서 버킷 작업을 위한 S3 정책 작업에서는 다음 형식의 `Resource` 요소에 액세스 포인트 ARN을 사용해야 합니다.

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/example-access-point"
```

다음 액세스 포인트 정책은 *12345678901* 계정을 가진 *Akua* 사용자에게 이름이 *example-access-point*인 S3 액세스 포인트를 통해 ListObjectsV2 API 작업을 수행할 수 있는 `s3:ListBucket` 권한을 부여합니다. 이 *Akua* 권한을 사용하면 *example-access-point*에 연결된 버킷에서 객체를 나열할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to list objects in the bucket through access point",
            "Effect": "Allow",
```

```
            "Principal": {
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
            "Action": [
            "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/example-access-
point"
        }
    ]
}
```

> **Note**
>
> 범용 버킷의 액세스 포인트에서 모든 버킷 작업을 지원하는 것은 아닙니다. 자세한 내용은 S3 작업과의 액세스 포인트 호환성 섹션을 참조하세요.

디렉터리 버킷의 액세스 포인트에 대한 정책의 버킷 작업

디렉터리 버킷의 액세스 포인트 정책에서 부여된 권한은 기본 버킷이 동일한 권한을 허용하는 경우에만 유효합니다. S3 액세스 포인트를 사용하는 경우, 버킷의 액세스 제어를 액세스 포인트에 위임하거나 액세스 포인트 정책의 동일한 권한을 기본 버킷 정책에 추가해야 합니다. 자세한 내용은 디렉터리 버킷의 액세스 포인트를 사용하도록 IAM 정책 구성 섹션을 참조하세요. 액세스 포인트 정책에서 버킷 작업을 위한 S3 정책 작업에서는 다음 형식의 `Resource` 요소에 액세스 포인트 ARN을 사용해야 합니다.

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/example-access-point--usw2-
az1--xa-s3"
```

다음 액세스 포인트 정책은 *12345678901* 계정을 가진 *Akua* 사용자에게 이름이 *example-access-point--usw2-az1--xa-s3*인 액세스 포인트를 통해 ListObjectsV2 API 작업을 수행할 수 있는 `s3:ListBucket` 권한을 부여합니다. 이 *Akua* 권한을 사용하면 *example-access-point--usw2-az1--xa-s3*에 연결된 버킷에서 객체를 나열할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to list objects in the bucket through access point",
```

```
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
            "Action": [
            "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3express:us-west-2:123456789012:accesspoint/example-access-point--usw2-az1--xa-s3"
        }
    ]
}
```

> **Note**
>
> 디렉터리 버킷의 액세스 포인트에서 모든 버킷 작업을 지원하는 것은 아닙니다. 자세한 내용은 디렉터리 버킷의 액세스 포인트에 대한 객체 작업 섹션을 참조하세요.

객체 작업

객체 작업은 객체 리소스 유형에 따라 작동하는 S3 API 작업입니다. 예를 들면 `GetObject`, `PutObject` 및 `DeleteObject`입니다. 객체 작업에 대한 S3 정책 작업을 위해서는 정책의 `Resource` 요소가 다음 예시 형식의 S3 객체 ARN이어야 합니다.

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
```

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/prefix/*"
```

> **Note**
>
> 이전 예시에서 볼 수 있듯이 객체 ARN에는 버킷 이름 뒤에 슬래시가 있어야 합니다.

다음 버킷 정책은 *12345678901* 계정을 가진 *Akua* 사용자에게 `s3:PutObject` 권한을 부여합니다. 이 권한을 사용하면 *Akua*가 PutObject API 작업을 사용하여 이름이 *amzn-s3-demo-bucket*인 S3 버킷에 객체를 업로드할 수 있습니다.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to upload objects",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
            "Action": [
            "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

액세스 포인트 정책의 버킷 작업

S3 액세스 포인트를 사용하여 객체 작업에 대한 액세스를 제어하는 경우 액세스 포인트 정책을 사용할 수 있습니다. 액세스 포인트 정책을 사용할 때 객체 작업에 대한 S3 정책 작업을 수행하려면 `Resource` 요소에 액세스 포인트 ARN을 `arn:aws:s3:region:account-id:accesspoint/access-point-name/object/resource` 형식으로 사용해야 합니다. 액세스 포인트를 사용하는 객체 작업의 경우 `Resource` 요소에서 전체 액세스 포인트 ARN 뒤에 `/object/` 값을 포함해야 합니다. 여기 몇 가지 예가 있습니다.

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/example-access-point/object/
*"
```

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/example-access-point/
object/prefix/*"
```

다음 액세스 포인트 정책은 *12345678901* 계정을 가진 *Akua* 사용자에게 `s3:GetObject` 권한을 부여합니다. 이 권한을 통해 *Akua*는 액세스 포인트와 연결된 버킷의 모든 객체에 대해 이름이 *example-access-point*인 액세스 포인트를 통해 GetObject API 작업을 수행할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to get objects through access point",
            "Effect": "Allow",
```

```
            "Principal": {
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
            "Action": [
            "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/example-access-point/object/*"
        }
    ]
}
```

> **Note**
>
> 액세스 포인트에서 모든 객체 작업을 지원하는 것은 아닙니다. 자세한 내용은 S3 작업과의 액세스 포인트 호환성 섹션을 참조하세요.

디렉터리 버킷의 액세스 포인트에 대한 정책의 객체 작업

디렉터리 버킷의 액세스 포인트를 사용하여 객체 작업에 대한 액세스를 제어하는 경우 액세스 포인트 정책을 사용할 수 있습니다. 액세스 포인트 정책을 사용할 때 객체 작업에 대한 S3 정책 작업을 수행하려면 Resource 요소에 액세스 포인트 ARN을 `arn:aws:s3:region:account-id:accesspoint/access-point-name/object/resource` 형식으로 사용해야 합니다. 액세스 포인트를 사용하는 객체 작업의 경우 Resource 요소에서 전체 액세스 포인트 ARN 뒤에 `/object/` 값을 포함해야 합니다. 여기 몇 가지 예가 있습니다.

```
"Resource": "arn:aws:s3express:us-west-2:123456789012:accesspoint/example-access-point--usw2-az1--xa-s3/object/*"
```

```
"Resource": "arn:aws:s3express:us-west-2:123456789012:accesspoint/example-access-point--usw2-az1--xa-s3/object/prefix/*"
```

다음 액세스 포인트 정책은 *12345678901* 계정을 가진 *Akua* 사용자에게 `s3:GetObject` 권한을 부여합니다. 이 권한을 통해 *Akua*는 액세스 포인트와 연결된 버킷의 모든 객체에 대해 이름이 *example-access-point--usw2-az1--xa-s3*인 액세스 포인트를 통해 GetObject API 작업을 수행할 수 있습니다.

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

```
    "Statement": [
        {
            "Sid": "Allow Akua to get objects through access point",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
            "Action": "s3express:CreateSession","s3:GetObject"
            "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/example-access-
point--usw2-az1--xa-s3/object/*"
        }
    ]
}
```

> **Note**
>
> 디렉터리 버킷의 액세스 포인트에서 모든 객체 작업을 지원하는 것은 아닙니다. 자세한 내용은 디렉터리 버킷의 액세스 포인트에 대한 객체 작업 섹션을 참조하세요.

범용 버킷의 액세스 포인트 작업

액세스 포인트 작업은 `accesspoint` 리소스 유형에서 작동하는 S3 API 작업입니다. 예를 들면 `CreateAccessPoint`, `DeleteAccessPoint` 및 `GetAccessPointPolicy`입니다. 액세스 포인트 작업에 대한 S3 정책 작업은 IAM 자격 증명 기반 정책에서만 사용할 수 있으며 버킷 정책이나 액세스 포인트 정책에서는 사용할 수 없습니다. 액세스 포인트 작업을 수행하려면 다음 예제 형식의 `Resource` 요소가 액세스 포인트 ARN이어야 합니다.

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/example-access-point"
```

다음 IAM ID 기반 정책은 이름이 *`example-access-point`*인 S3 액세스 포인트에서 GetAccessPointPolicy API 작업을 수행할 수 있는 `s3:GetAccessPointPolicy` 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Grant permission to retrieve the access point policy of access
 point example-access-point",
            "Effect": "Allow",
            "Action": [
```

```
            "s3:GetAccessPointPolicy"
            ],
            "Resource": "arn:aws:s3:*:123456789012:accesspoint/example-access-point"
        }
    ]
}
```

액세스 포인트를 사용하는 경우 버킷 작업에 대한 액세스를 제어하려면 범용 버킷의 액세스 포인트에 대한 정책의 버킷 작업 섹션을, 객체 작업에 대한 액세스를 제어하려면 액세스 포인트 정책의 버킷 작업 섹션을 참조하세요. 액세스 포인트 정책을 구성하는 방법에 대한 자세한 내용은 액세스 포인트를 사용하도록 IAM 정책 구성 섹션을 참조하세요.

디렉터리 버킷의 액세스 포인트 작업

디렉터리 버킷의 액세스 포인트 작업은 accesspoint 리소스 유형에서 작동하는 S3 API 작업입니다. 예를 들면 CreateAccessPoint, DeleteAccessPoint 및 GetAccessPointPolicy입니다. 액세스 포인트 작업에 대한 S3 정책 작업은 IAM 자격 증명 기반 정책에서만 사용할 수 있으며 버킷 정책이나 액세스 포인트 정책에서는 사용할 수 없습니다. 디렉터리 버킷의 액세스 포인트 작업을 수행하려면 다음 예제 형식의 Resource 요소가 액세스 포인트 ARN이어야 합니다.

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/example-access-point--usw2-az1--xa-s3"
```

다음 IAM ID 기반 정책은 이름이 *example-access-point--usw2-az1--xa-s3*인 액세스 포인트에서 GetAccessPointPolicy API 작업을 수행할 수 있는 s3express:GetAccessPointPolicy 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Grant permission to retrieve the access point policy of access
 point example-access-point--usw2-az1--xa-s3",
            "Effect": "Allow",
            "Action": [
            "s3express:CreateSession","s3express:GetAccessPointPolicy"
            ],
            "Resource": "arn:aws:s3:*:123456789012:accesspoint/example-access-point--
usw2-az1--xa-s3"
        }
    ]
```

```
}
```

다음 IAM ID 기반 정책은 디렉터리 버킷의 액세스 포인트를 만들 수 있는 `s3express:CreateAccessPoint` 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Grant CreateAccessPoint.",
            "Principal": "*",
            "Action": "s3express:CreateSession",
            "s3express:CreateAccessPoint""Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

다음 IAM ID 기반 정책은 디렉터리 버킷의 액세스 포인트에 대한 액세스 포인트 범위를 만들 수 있는 `s3express:PutAccessPointScope` 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Grant PutAccessPointScope",
            "Principal": "*",
            "Action": "s3express:CreateSession",
            "s3express:CreateAccessPoint",
            "S3Express:PutAccessPointScope""Effect": "Allow",
            "Resource": "*",
        }
    ]
}
```

디렉터리 버킷의 액세스 포인트를 사용하는 경우 버킷 작업에 대한 액세스를 제어하려면 디렉터리 버킷의 액세스 포인트에 대한 정책의 버킷 작업 섹션을, 객체 작업에 대한 액세스를 제어하려면 디렉터리 버킷의 액세스 포인트에 대한 정책의 객체 작업 섹션을 참조하세요. 디렉터리 버킷의 액세스 포인트 정

책을 구성하는 방법에 대한 자세한 내용은 디렉터리 버킷의 액세스 포인트를 사용하도록 IAM 정책 구성 섹션을 참조하세요.

객체 Lambda 액세스 포인트 작업

Amazon S3 객체 Lambda를 통해 자체 코드를 Amazon S3 GET, LIST, HEAD 요청에 추가하여 애플리케이션으로 데이터가 반환될 때 데이터를 수정 및 처리할 수 있습니다. 객체 Lambda 액세스 포인트를 통해 요청할 수 있으며, 이는 다른 액세스 포인트를 통해 요청하는 것과 동일하게 작동합니다. 자세한 내용은 S3 객체 Lambda를 사용하여 객체 변환 섹션을 참조하세요.

객체 Lambda 액세스 포인트 작업용 정책을 구성하는 방법에 대한 자세한 내용은 객체 Lambda 액세스 포인트에 대한 IAM 정책 구성 섹션을 참조하세요.

다중 리전 액세스 포인트 작업

다중 리전 액세스 포인트는 애플리케이션이 여러 AWS 리전에 있는 S3 버킷의 요청을 이행하는 데 사용할 수 있는 글로벌 엔드포인트를 제공합니다. 다중 리전 액세스 포인트를 사용하여 단일 리전에서 사용되는 것과 동일한 아키텍처로 다중 리전 애플리케이션을 빌드하면 전 세계 어디에서나 해당 애플리케이션을 실행할 수 있습니다. 자세한 내용은 다중 리전 액세스 포인트로 다중 리전 트래픽 관리 섹션을 참조하세요.

다중 리전 액세스 포인트 작업용 정책을 구성하는 방법에 대한 자세한 내용은 다중 리전 액세스 포인트 정책 예시 섹션을 참조하세요.

배치 작업

(배치 작업) 작업은 작업 리소스 유형에서 작동하는 S3 API 작업입니다. 예: `DescribeJob` 및 `CreateJob`. 작업에 대한 S3 정책 작업은 IAM 자격 증명 기반 정책에서만 사용할 수 있으며 버킷 정책에서는 사용할 수 없습니다. 또한 작업을 위해서는 IAM 자격 증명 기반 정책의 `Resource` 요소가 다음 예시 형식의 `job` ARN이어야 합니다.

```
"Resource": "arn:aws:s3:*:123456789012:job/*"
```

다음 IAM ID 기반 정책은 이름이 *`example-job`*인 S3 배치 운영 작업에 대해 DescribeJob API 작업을 수행할 수 있는 `s3:DescribeJob` 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow describing the Batch operation job example-job",
```

```
            "Effect": "Allow",
            "Action": [
            "s3:DescribeJob"
            ],
            "Resource": "arn:aws:s3:*:123456789012:job/example-job"
        }
    ]
}
```

S3 Storage Lens 구성 작업

S3 Storage Lens 구성 작업을 구성하는 방법에 대한 자세한 내용은 Amazon S3 Storage Lens 권한 설정 섹션을 참조하세요.

계정 작업

계정 작업은 계정 수준에서 작동하는 S3 API 작업입니다. 예: `GetPublicAccessBlock`(계정의 경우). 계정은 Amazon S3에서 정의한 리소스 유형이 아닙니다. 계정 작업에 대한 S3 정책 작업은 IAM 자격 증명 기반 정책에서만 사용할 수 있으며 버킷 정책에서는 사용할 수 없습니다. 또한 계정 작업을 위해서는 IAM 자격 증명 기반 정책의 `Resource` 요소가 `"*"`여야 합니다.

다음 IAM ID 기반 정책은 계정 수준의 GetPublicAccessBlock API 작업을 수행하고 계정 수준의 퍼블릭 액세스 차단 설정을 검색할 수 있는 `s3:GetAccountPublicAccessBlock` 권한을 부여합니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"Allow retrieving the account-level Public Access Block settings",
         "Effect":"Allow",
         "Action":[
            "s3:GetAccountPublicAccessBlock"
         ],
         "Resource":[
            "*"
         ]
       }
    ]
}
```

Amazon S3용 정책 예시

- Amazon S3 ID 기반 정책 예시를 보려면 Amazon S3의 ID 기반 정책 섹션을 참조하세요.

- Amazon S3 리소스 기반 정책의 예시를 보려면 Amazon S3의 버킷 정책 및 액세스 포인트를 사용하도록 IAM 정책 구성 섹션을 참조하세요.

Amazon S3의 정책 리소스

정책 리소스 지원: 예

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

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 문에는 `Resource`또는 `NotResource`요소가 반드시 추가되어야 합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 지원하는 작업에 대해 이를 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

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

일부 Amazon S3 API 작업에서는 여러 리소스가 지원됩니다. 예를 들어, `s3:GetObject`는 *example-resource-1*과 *example-resource-2*에 액세스하므로 보안 주체는 두 리소스에 모두 액세스할 수 있는 권한을 반드시 가져야 합니다. 단일 문에서 여러 리소스를 지정하려면 다음 예제와 같이 각 ARN을 쉼표로 구분합니다.

```
"Resource": [
      "example-resource-1",
      "example-resource-2"
```

Amazon S3의 리소스는 버킷, 객체, 액세스 포인트 또는 작업입니다. 정책에서는 버킷, 객체, 액세스 포인트 또는 작업의 Amazon 리소스 이름(ARN)을 사용하여 리소스를 식별합니다.

Amazon S3 리소스 유형 및 해당 ARN의 목록을 보려면 서비스 권한 부여 참조의 Amazon S3에서 정의한 리소스를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 Amazon S3에서 정의한 작업을 참조하세요.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

리소스 ARN의 와일드카드 문자

와일드카드 문자는 리소스 ARN에도 사용할 수 있습니다. ARN 세그먼트(콜론으로 구분된 부분) 내에서 와일드카드 문자(* 및 ?)를 사용할 수 있습니다. 별표(*)는 0개 이상의 문자 조합을 나타내고 물음표(?)는 단일 문자를 나타냅니다. 각 세그먼트에 여러 개의 * 또는 ? 문자를 사용할 수 있습니다. 하지만 와일드카드 문자는 여러 세그먼트에 걸쳐 사용할 수 없습니다.

- 다음 ARN은 ARN의 `relative-ID` 부분에 * 와일드카드 문자를 사용하여 *amzn-s3-demo-bucket* 버킷의 모든 객체를 식별합니다.

```
arn:aws:s3:::amzn-s3-demo-bucket/*
```

- 다음 ARN에서는 * 기호를 사용하여 모든 S3 버킷 및 객체를 나타냅니다.

```
arn:aws:s3:::*
```

- 다음 ARN에서는 `relative-ID` 부분에 와일드카드 문자(* 및 ?)를 모두 사용합니다. 이 ARN은 *amzn-s3-demo-example1bucket*, *amzn-s3-demo-example2bucket*, *amzn-s3-demo-example3bucket* 등 버킷의 모든 객체를 나타냅니다.

```
arn:aws:s3:::amzn-s3-demo-example?bucket/*
```

리소스 ARN의 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.

Amazon S3용 정책 예시

- Amazon S3 ID 기반 정책 예시를 보려면 Amazon S3의 ID 기반 정책 섹션을 참조하세요.

- Amazon S3 리소스 기반 정책의 예시를 보려면 Amazon S3의 버킷 정책 및 액세스 포인트를 사용하도록 IAM 정책 구성 섹션을 참조하세요.

Amazon S3의 정책 조건 키

서비스별 정책 조건 키 지원: 예

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

`Condition` 요소(또는 `Condition` 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. `Condition` 요소는 옵션입니다. 같거나 작음과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 `Condition` 요소를 지정하거나 단일 `Condition` 요소에서 여러 키를 지정하는 경우, AWS는 논리적 `AND` 작업을 사용하여 평가합니다. 단일 조건 키의 여러 값을 지정하는 경우, AWS는 논리적 `OR` 작업을 사용하여 조건을 평가합니다. 문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하세요.

AWS는 전역 조건 키와 서비스별 조건 키를 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

각 Amazon S3 조건 키는 API에서 조건이 설정될 수 있도록 허용된 같은 이름의 요청 헤더에 매핑됩니다. Amazon S3 전용 조건 키는 이름이 같은 요청 헤더의 동작을 명령합니다. 예를 들어 `s3:GetObjectVersion` 권한에 대한 조건부 권한을 부여하는 데 사용할 수 있는 조건 키 `s3:VersionId`는 GET Object 요청에서 설정한 `versionId` 쿼리 파라미터의 동작을 정의합니다.

Amazon S3 조건 키 목록을 보려면 서비스 승인 참조의 Amazon S3에 사용되는 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 Amazon S3에서 정의한 작업을 참조하세요.

예: 특정 스토리지 클래스를 이용한 객체 업로드 제한

계정 ID *123456789012*로 표시되는 Account A가 버킷을 소유하고 있다고 가정합니다. Account A 관리자는 Account A에 속한 사용자인 *Dave*가 `STANDARD_IA` 스토리지 클래스에 객체가 저장된 경우

에만 버킷에 객체를 업로드할 수 있도록 *Dave*의 권한을 제한하려고 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::123456789012:user/Dave"
                    },
                    "Action": "s3:PutObject",
                    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
                    "Condition": {
                      "StringEquals": {
                        "s3:x-amz-storage-class": [
                          "STANDARD_IA"
                        ]
                      }
                    }
                  }
                ]
            }
```

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 `PutObject` 요청에 `public-read` 값을 가진 `x-amz-acl` 헤더를 포함하도록 요구합니다.

Amazon S3용 정책 예시

- Amazon S3 ID 기반 정책 예시를 보려면 Amazon S3의 ID 기반 정책 섹션을 참조하세요.
- Amazon S3 리소스 기반 정책의 예시를 보려면 Amazon S3의 버킷 정책 및 액세스 포인트를 사용하도록 IAM 정책 구성 섹션을 참조하세요.

Amazon S3의 ACL

ACL 지원: 예

Amazon S3에서 액세스 제어 목록(ACL)은 리소스에 액세스할 수 있는 AWS 계정을 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

> **Important**
>
> Amazon S3의 최신 사용 사례 대부분은 더 이상 ACL을 사용할 필요가 없습니다.

Amazon S3의 ACL을 사용한 액세스 제어에 대한 자세한 내용은 ACL을 사용한 액세스 관리 섹션을 참조하세요.

Amazon S3의 ABAC

ABAC 지원(정책의 태그): 부분적

속성 기반 액세스 제어(ABAC)는 속성에 근거하여 권한을 정의하는 권한 부여 전략입니다. AWS에서는 이러한 속성을 태그라고 합니다. IAM 엔티티(사용자 또는 역할) 및 많은 AWS리소스에 태그를 연결할 수 있습니다. ABAC의 첫 번째 단계로 개체 및 리소스에 태그를 지정합니다. 그런 다음 위탁자의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하도록 ABAC 정책을 설계합니다.

ABAC는 빠르게 성장하는 환경에서 유용하며 정책 관리가 번거로운 상황에 도움이 됩니다.

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

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

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

태그를 기반으로 S3 배치 작업에 대한 액세스를 제한하는 ID 기반 정책의 예시를 보려면 작업 태그를 사용하여 Batch Operations에 대한 권한 제어 섹션을 참조하세요.

ABAC 및 객체 태그

ABAC 정책에서 객체는 `aws:` 태그 대신 `s3:` 태그를 사용합니다. 객체 태그를 기반으로 객체에 대한 액세스를 제어하려면 다음 태그를 사용하여 정책의 Condition 요소에 태그 정보를 제공하세요.

- `s3:ExistingObjectTag/tag-key`

- `s3:RequestObjectTagKeys`
- `s3:RequestObjectTag/tag-key`

권한 정책 예시를 포함하여 객체 태그를 사용한 액세스 제어에 대한 자세한 내용은 태그 지정 및 액세스 제어 정책 섹션을 참조하세요.

Amazon S3에서 임시 자격 증명 사용

임시 자격 증명 지원: 예

일부 AWS 서비스는 임시 자격 증명을 사용하여 로그인할 때 작동하지 않습니다. 임시 보안 인증으로 작동하는 AWS 서비스를 비롯한 추가 정보는 IAM 사용 설명서의 IAM으로 작업하는 AWS 서비스를 참조하세요.

사용자 이름과 암호를 제외한 다른 방법을 사용하여 AWS Management Console에 로그인하면 임시 자격 증명을 사용하는 것입니다. 예를 들어, 회사의 Single Sign-On(SSO) 링크를 사용하여 AWS에 액세스하면 해당 프로세스에서 자동으로 임시 자격 증명을 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 자격 증명을 자동으로 생성합니다. 역할 전환에 대한 자세한 내용은 IAM 사용 설명서의 사용자에서 IAM 역할로 전환(콘솔)을 참조하세요.

AWS CLI 또는 AWS API를 사용하여 임시 자격 증명을 수동으로 만들 수 있습니다 그런 다음 이러한 임시 자격 증명을 사용하여 AWS에 액세스할 수 있습니다. AWS에서는 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성할 것을 권장합니다. 자세한 정보는 IAM의 임시 보안 자격 증명 섹션을 참조하세요.

Amazon S3에 대한 액세스 세션 전달

전달 액세스 세션(FAS) 지원: 예

IAM 사용자 또는 역할을 사용하여 AWS에서 작업을 수행하는 사람은 위탁자로 간주됩니다. 일부 서비스를 사용하는 경우, 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 AWS 서비스를 직접 호출하는 위탁자의 권한과 요청하는 AWS 서비스를 함께 사용하여 다운스트림 서비스에 대한 요청을 수행합니다. FAS 요청은 서비스에서 완료를 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 받은 경우에만 이루어집니다. 이 경우, 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

- SSE-KMS를 사용하여 암호화되었을 때 Amazon S3에서 객체 암호 해독을 위해 AWS KMS를 직접적으로 호출할 때 FAS가 사용됩니다. 자세한 내용은 AWS KMS 키를 사용한 서버 측 암호화(SSE-KMS) 사용 섹션을 참조하세요.

- S3 Access Grants 또한 FAS를 사용합니다. 특정 ID와 관련해 S3 데이터에 대한 액세스 권한 부여를 생성한 후, 피부여자는 S3 Access Grants에 임시 자격 증명을 요청합니다. S3 Access Grants는 AWS STS로부터 요청자를 위한 임시 자격 증명을 받아 요청자에게 자격 증명을 벤딩합니다. 자세한 내용은 S3 Access Grants를 통해 Amazon S3 데이터에 대한 액세스 요청 섹션을 참조하세요.

Amazon S3의 서비스 역할

서비스 역할 지원: 예

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

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 Amazon S3 기능이 중단될 수 있습니다. Amazon S3에서 관련 지침을 제공하는 경우에만 서비스 역할을 편집합니다.

Amazon S3의 서비스 연결 역할

서비스 링크 역할 지원: 부분적

서비스 연결 역할은 AWS 서비스에 연결된 서비스 역할의 한 유형입니다. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 링크 역할은 AWS 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

Amazon S3는 Amazon S3 Storage Lens 대한 서비스 연결 역할을 지원합니다. Amazon S3 서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 Amazon S3 스토리지 렌즈에 대한 서비스 연결 역할 사용 섹션을 참조하세요.

보안 주체로서의 Amazon S3 서비스

정책의 서비스 이름	S3 기능	추가 정보
`s3.amazonaws.com`	S3 복제	라이브 복제 설정 개요
`s3.amazonaws.com`	S3 이벤트 알림	Amazon S3 이벤트 알림

정책의 서비스 이름	S3 기능	추가 정보
s3.amazonaws.com	S3 인벤토리	S3 Inventory를 사용한 데이터 카탈로그화 및 분석
access-grants.s3.amazonaws.com	S3 Access Grants	위치 등록
batchoperations.s3.amazonaws.com	S3 배치 작업	배치 작업에 대한 권한 부여
logging.s3.amazonaws.com	S3 서버 액세스 로깅	Amazon S3 서버 액세스 로깅 사용 설정
storage-lens.s3.amazonaws.com	S3 Storage Lens	데이터 내보내기를 사용하여 Amazon S3 스토리지 렌즈 지표 보기

Amazon S3에서 요청에 권한을 부여하는 방법

Amazon S3가 요청을 수신하면(예: 버킷 또는 객체 작업) 먼저 요청자가 필요한 권한을 보유하고 있는지 확인합니다. Amazon S3는 요청을 승인할지 여부를 결정할 때 모든 관련 액세스 정책, 사용자 정책 및 리소스 기반 정책(버킷 정책, 버킷 액세스 제어 목록(ACL), 객체 ACL)을 평가합니다.

> **Note**
>
> Amazon S3 권한 검사에서 유효한 권한을 찾지 못한 경우, 액세스 거부(403 금지) 권한 거부 오류가 반환됩니다. 자세한 내용은 Amazon S3의 액세스 거부(403 금지) 오류 문제 해결을 참조하세요.

요청자가 특정 작업을 수행할 권한을 보유하고 있는지 확인하기 위해 Amazon S3는 요청을 수신할 경우 다음을 순서대로 수행합니다.

1. 런타임 시 모든 관련 액세스 정책(사용자 정책, 버킷 정책, ACL)을 평가를 위한 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 요청에 포함된 리소스(버킷, 객체)를 소유하는 경우에도, Amazon S3는 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 사용자 자격 증명을 사용하여 요청이 이루어진 경우, Amazon S3는 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.

다음은 Amazon S3가 요청을 승인하는 방법을 보여 주는 몇 가지 예시 시나리오입니다.

Example – 요청자가 IAM 보안 주체인 경우

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. 리소스에서 특정 작업을 수행하려면 IAM 보안 주체는 소속된 상위 AWS 계정과 리소스를 소유한 AWS 계정에서 모두 권한이 필요합니다.

Example – 요청자가 IAM 보안 주체인 경우 - 버킷 소유자가 소유하지 않은 객체에 대한 작업 요청인 경우

버킷 소유자가 소유하지 않은 객체에 대한 작업 요청인 경우, Amazon S3는 요청자가 객체 소유자로부터 권한을 받았는지 확인해야 할 뿐만 아니라 버킷 정책을 통해 버킷 소유자가 객체에 대한 명시적인 거부를 설정하지 않았는지 확인해야 합니다. 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.

기본적으로 다른 AWS 계정이 S3 범용 버킷에 객체를 업로드하면 해당 계정(객체 작성자)이 객체를 소유하고 객체에 액세스할 수 있으며 액세스 제어 목록(ACL)을 통해 다른 사용자에게 객체에 대한 액세스 권한을 부여할 수 있습니다. 객체 소유권을 사용하여 ACL이 사용 중지되고 버킷 소유자로서 범용 버킷의 모든 객체를 자동으로 소유하도록 이 기본 동작을 변경할 수 있습니다. 결과적으로 데이터에 대한 액세스 제어는 IAM 사용자 정책, S3 버킷 정책, 가상 프라이빗 클라우드(VPC) 엔드포인트 정책 및 AWS Organizations 서비스 제어 정책(SCP)과 같은 정책을 기반으로 합니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

Amazon S3가 액세스 정책을 평가하여 버킷 작업 및 객체 작업에 대한 요청을 승인하거나 거부하는 방법에 대한 자세한 내용은 다음 주제를 참조하십시오.

주제

- Amazon S3에서 버킷 작업 요청에 권한을 부여하는 방법
- Amazon S3에서 객체 작업에 대한 요청에 권한을 부여하는 방법

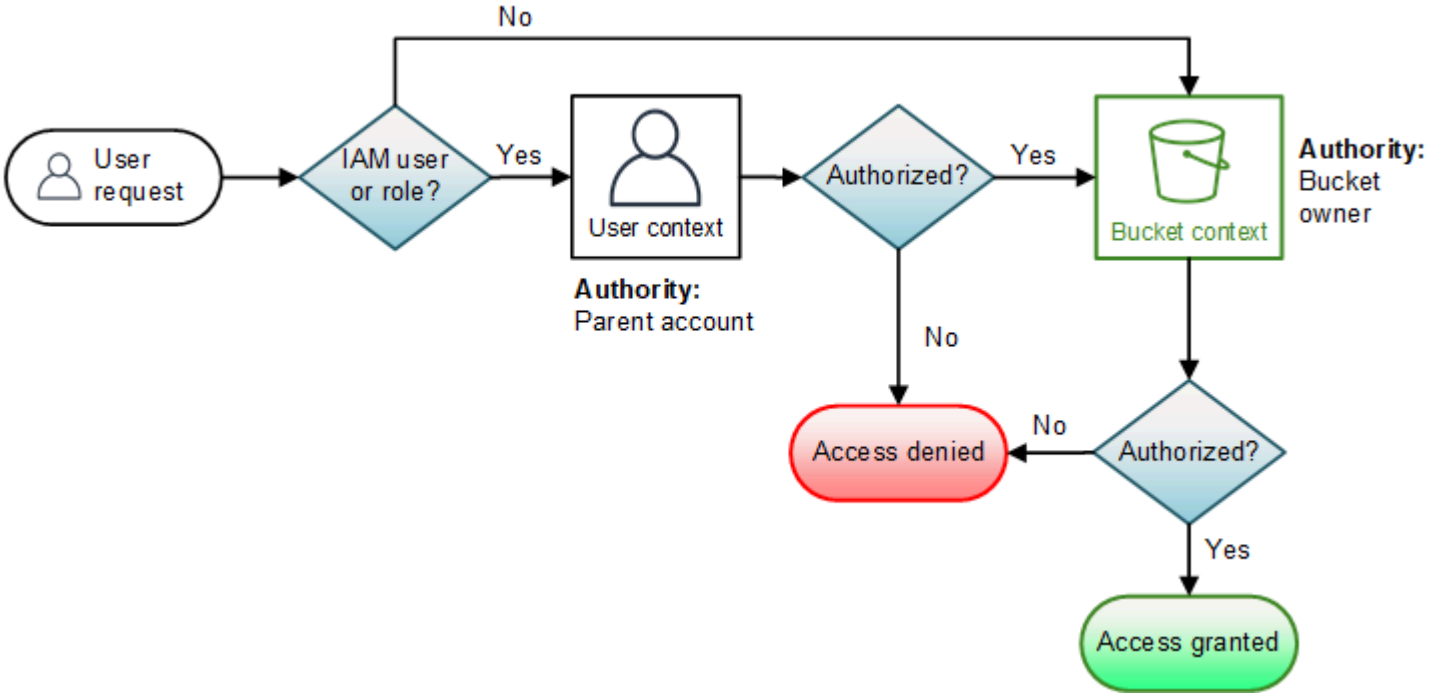
Amazon S3에서 버킷 작업 요청에 권한을 부여하는 방법

Amazon S3에서 버킷 작업에 대한 요청을 수신하면, Amazon S3는 모든 관련 권한을 정책 세트로 변환하여 런타임 시 평가합니다. 관련 권한에는 리소스 기반 권한(예: 버킷 정책 및 버킷 액세스 제어 목록)과 사용자 정책(요청이 IAM 보안 주체인 경우)이 포함됩니다. 그러면 Amazon S3가 특정 컨텍스트(사용자 컨텍스트 또는 버킷 컨텍스트)에 따라 일련의 단계를 통해 변환된 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 서버 액세스 로깅을 사용한 요청 로깅 섹션을 참조하세요.
2. 버킷 컨텍스트 – 요청자는 특정 버킷 작업을 수행할 수 있는 버킷 소유자의 권한을 보유해야 합니다. 이 단계에서 Amazon S3는 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.

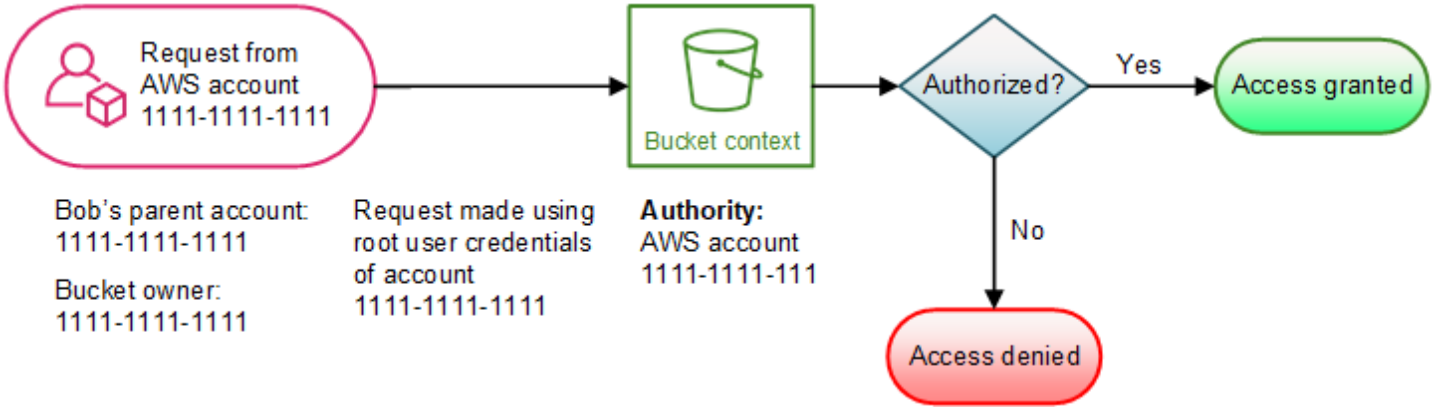
다음은 버킷 작업의 컨텍스트 기반 평가에 대한 그래픽을 이용한 그림입니다.



다음 예는 평가 논리를 설명합니다.

예제 1: 버킷 소유자가 요청한 버킷 작업

이 예제에서 버킷 소유자는 AWS 계정의 루트 자격 증명을 사용하여 버킷 작업을 위한 요청을 보냅니다.

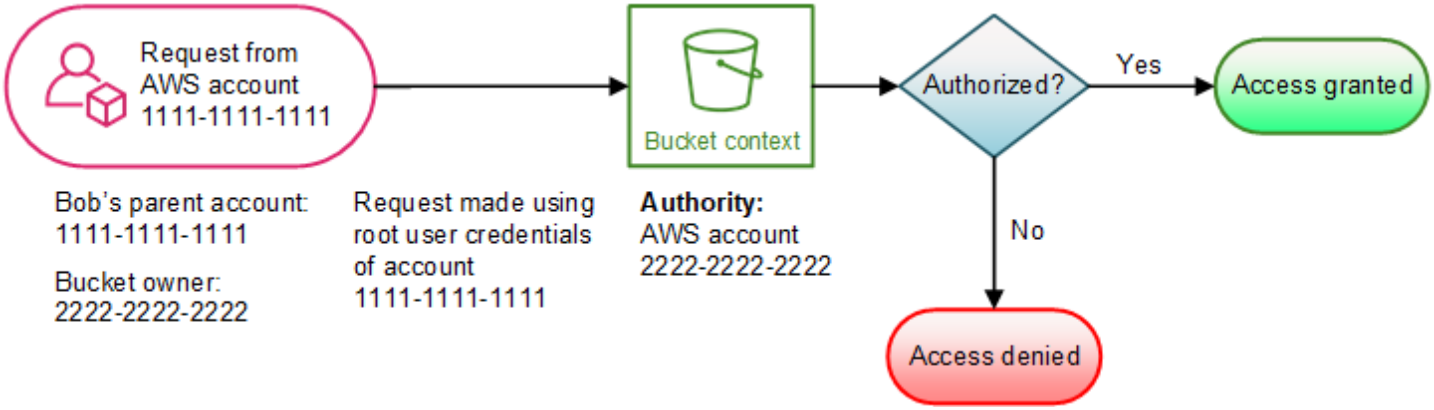


Amazon S3은 다음과 같이 컨텍스트 평가를 수행합니다.

1. AWS 계정의 루트 사용자 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.

예제 2: 버킷 소유자가 아닌 AWS 계정에서 요청한 버킷 작업

이 예제에서는 AWS 계정 2222-2222-2222가 소유한 버킷 작업을 위해 AWS 계정 1111-1111-1111의 루트 사용자 자격 증명을 사용하여 요청이 이루어졌습니다. 이 요청과 관련된 어떤 IAM 사용자도 없습니다.

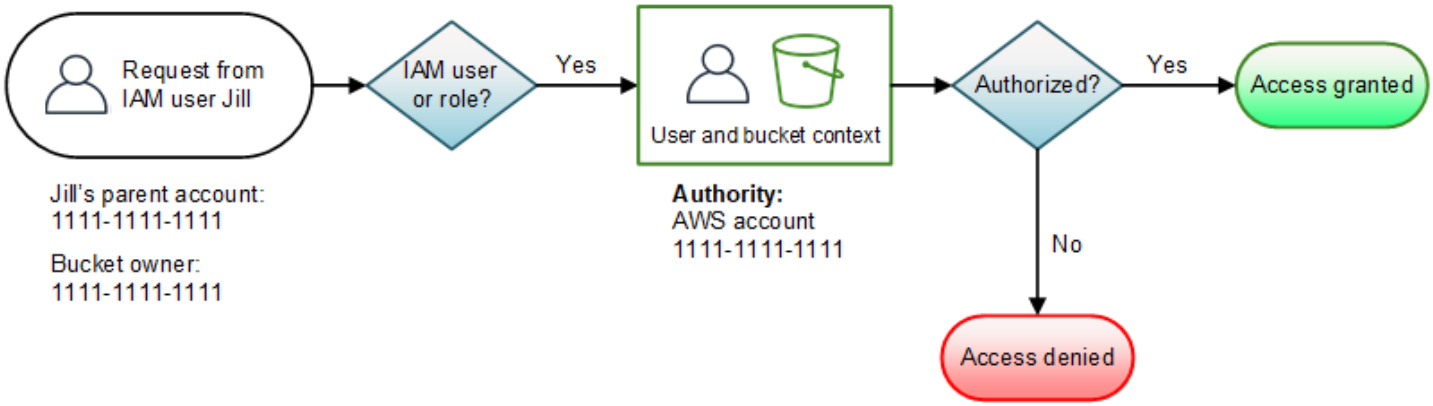


이 경우, Amazon S3는 다음과 같이 컨텍스트를 평가합니다.

1. AWS 계정의 루트 사용자 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 S3가 요청을 거부합니다. 그렇지 않으면 Amazon S3은 요청에 권한을 부여하고 해당 작업을 수행합니다.

예제 3: 상위 AWS 계정이 버킷 소유자인 IAM 보안 주체가 요청한 버킷 작업

이 예제에서는 또한 버킷을 소유하고 있는 AWS 계정 1111-1111-1111의 IAM 사용자인 Jill이 요청을 보냅니다.



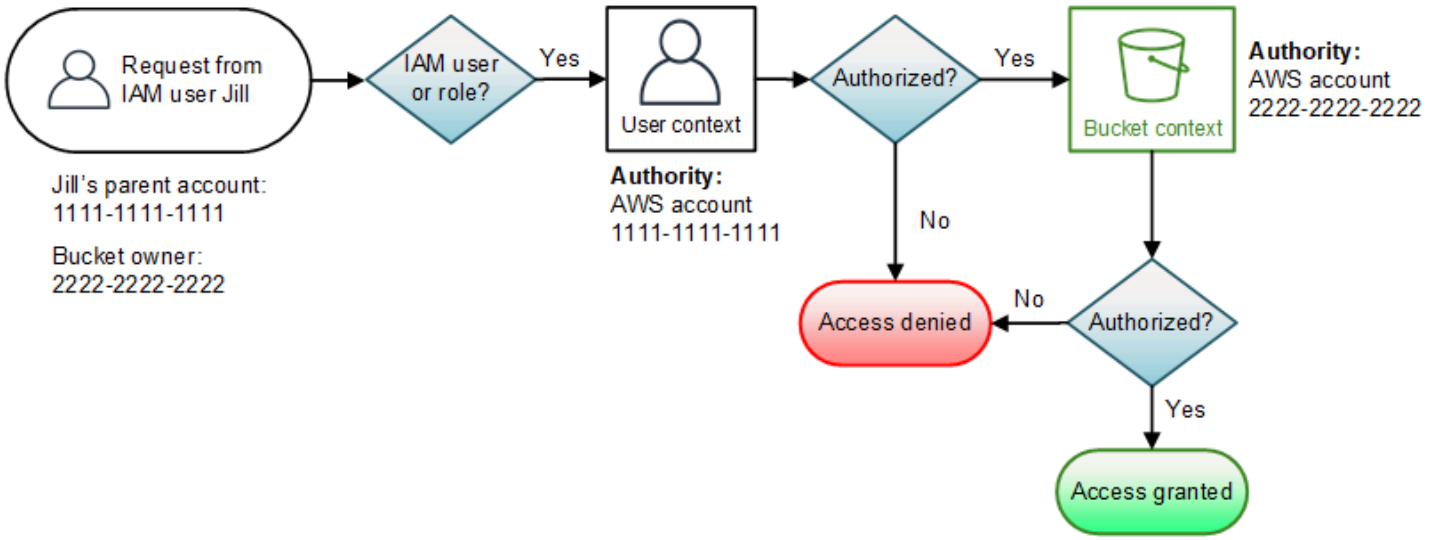
Amazon S3은 다음의 컨텍스트 평가를 수행합니다.

1. 사용자 컨텍스트에서 IAM 보안 주체가 요청을 하기 때문에 Amazon S3는 상위 AWS 계정에 속한 모든 정책을 평가하여 Jill이 작업을 수행할 권한을 보유하고 있는지 확인합니다.

   이 예제에서는 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.

예제 4: 상위 AWS 계정이 버킷 소유자가 아닌 IAM 보안 주체가 요청한 버킷 작업

이 예제에서는 상위 AWS 계정이 1111-1111-1111인 IAM 사용자인 Jill이 요청을 전송하지만 버킷은 다른 AWS 계정, 2222-2222-2222가 소유합니다.



Jill은 상위 AWS 계정 및 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 S3가 계속하여 버킷 컨텍스트를 평가합니다. Jill에게 권한이 없는 경우 Amazon S3는 요청을 거부합니다.
2. 버킷 컨텍스트에서 Amazon S3는 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 S3가 요청을 승인하고 작업을 수행합니다. 그렇지 않으면 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 정책을 런타임에 평가할 정책 집합으로 변환합니다. 그런 다음 일련의 단계에서 발생한 정책 세트를 평가합니다. 각 단계에서 Amazon S3가 세 개의 특정 컨텍스트(사용자 컨텍스트, 버킷 컨텍스트 및 객체 컨텍스트)에서 정책의 하위 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 요청에 포함된 리소스(버킷, 객체)를 소유하는 경우에도, Amazon S3는 동시에 해당 리소스 정책(버킷 정책, 버킷 ACL, 객체 ACL)도 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.

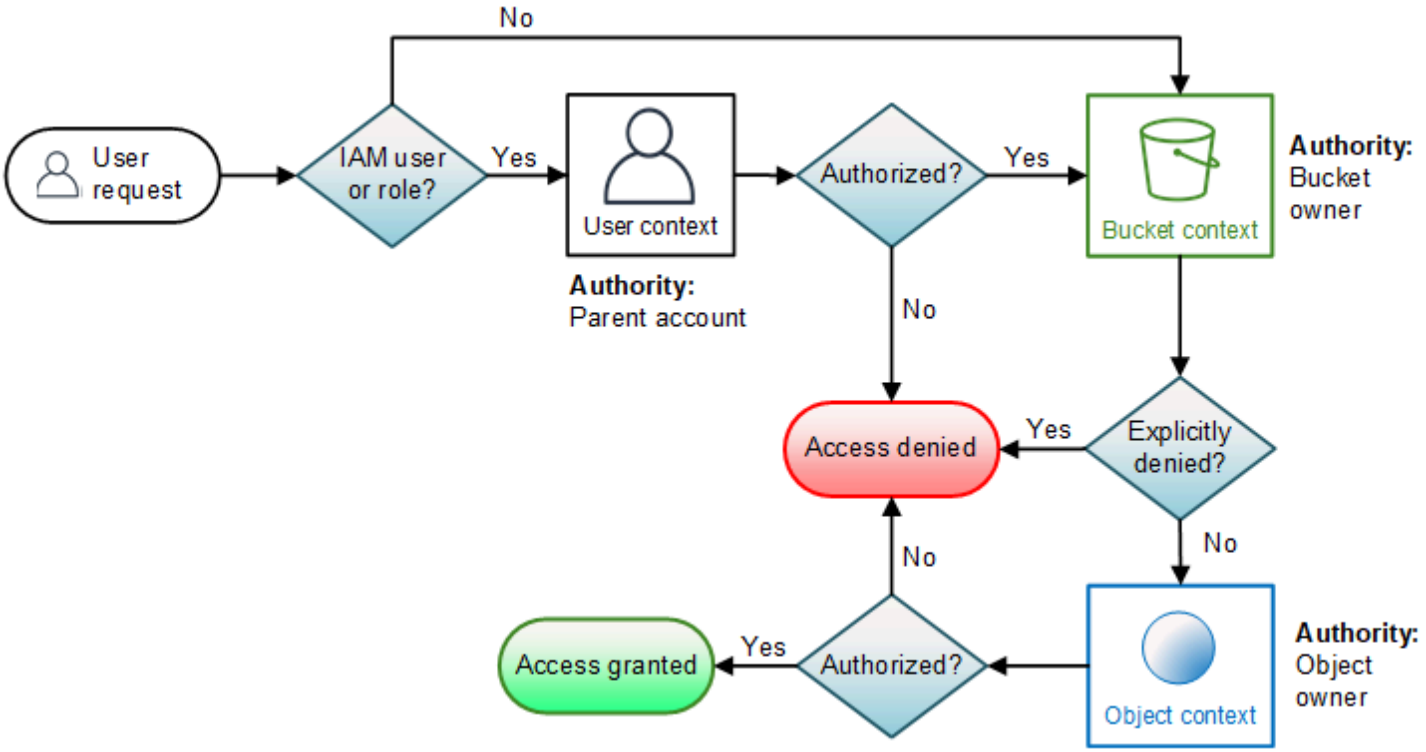
2. 버킷 컨텍스트 – 이 컨텍스트에서 Amazon S3은 버킷을 소유한 AWS 계정이 소유한 정책을 평가합니다.

   요청에 포함된 객체를 소유한 AWS 계정이 버킷 소유자와 동일하지 않으면, Amazon S3는 버킷 소유자가 객체에 대한 액세스를 명시적으로 거부했는지 정책을 통해 평가합니다. 객체에 명시적으로 거부된 세트가 있는 경우 Amazon S3은 해당 요청에 권한을 부여하지 않습니다.

3. 객체 컨텍스트 – 요청자는 특정 객체 작업을 수행할 수 있는 객체 소유자의 권한을 보유해야 합니다. 이 단계에서 Amazon S3은 ACL을 평가합니다.

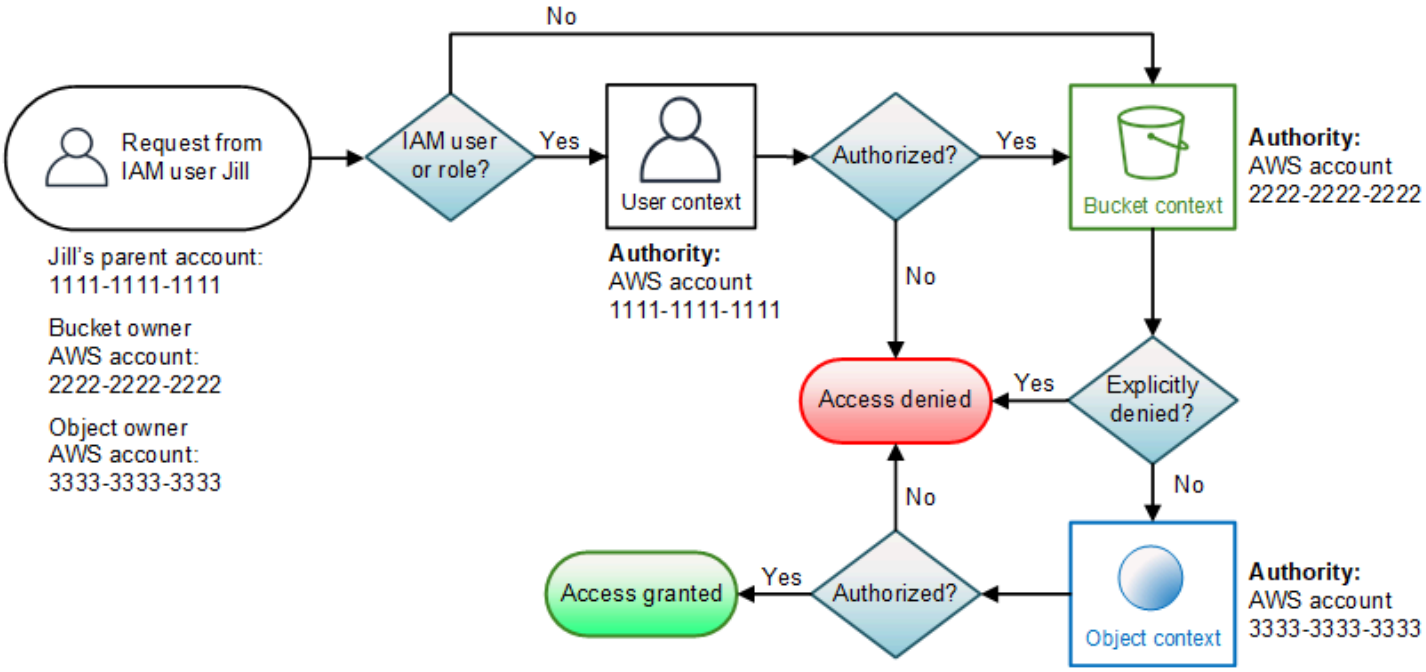
> **Note**
>
> 버킷과 객체 소유자가 같지 않은 경우 버킷 컨텍스트에서 평가되는 버킷 정책에서 객체에 대한 액세스 권한을 부여할 수 있습니다. 소유자가 다르면 객체 소유자는 객체 ACL을 사용하여 권한을 부여해야 합니다. 객체를 소유한 AWS 계정이 IAM 보안 주체가 속한 상위 계정이기도 한 경우, 사용자 컨텍스트에서 평가되는 사용자 정책에서 객체 권한을 구성할 수 있습니다. 이러한 액세스 정책 대안을 사용하는 것에 대한 자세한 내용은 정책을 사용하여 Amazon S3 리소스에 대한 액세스를 관리하는 연습을 참조하십시오.
>
> 버킷 소유자가 버킷의 모든 객체를 소유하고 버킷 정책 또는 IAM 기반 정책을 사용하여 이러한 객체에 대한 액세스를 관리하려는 경우 객체 소유권에 대해 버킷 소유자 시행 설정을 적용할 수 있습니다. 이 설정을 사용하면 버킷 소유자가 버킷의 모든 객체를 자동으로 소유하고 완전히 제어할 수 있습니다. 버킷 및 객체 ACL은 편집할 수 없으며 더 이상 액세스 대상으로 간주되지 않습니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

다음은 객체 작업의 컨텍스트 기반 평가에 대한 그림입니다.



객체 작업 요청의 예시

이 예시에서는 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: `GetObject`)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 S3는 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.

Amazon S3 API 작업에 필요한 권한

> **Note**
>
> 이 페이지에서는 범용 버킷에 대한 Amazon S3 정책 작업에 대해 설명합니다. 디렉터리 버킷에 대한 Amazon S3 정책 작업 관련 자세한 내용은 디렉터리 버킷에 대한 작업 섹션을 참조하세요.

S3 API 작업을 수행하려면 적절한 권한이 있어야 합니다. 이 페이지는 S3 API 작업을 필요한 권한에 연결합니다. S3 API 작업을 수행할 권한을 부여하려면 유효한 정책(예: S3 버킷 정책 또는 IAM 자격 증명 기반 정책)을 작성하고 정책의 `Action` 요소에 해당 작업을 지정해야 합니다. 이러한 작업을 정책 작업이라고 합니다. 버킷 작업에 권한을 부여하는 액세스 포인트 정책에서 요소는 ARN이어야 합니다.

정책을 작성할 때는 해당 Amazon S3 정책 작업에 필요한 올바른 리소스 유형을 기반으로 `Resource` 요소를 지정해야 합니다. 이 페이지는 리소스 유형별로 S3 API 작업에 대한 권한을 분류합니다. 리소스 유형에 대한 자세한 내용은 서비스 승인 참조의 Amazon S3에서 정의한 리소스 유형을 참조하세요. Amazon S3 정책 작업, 리소스 및 조건 키의 전체 목록은 서비스 승인 참조에서 Amazon S3에 사용되는 작업, 리소스 및 조건 키를 참조하세요. Amazon S3 API 작업의 전체 목록은 Amazon Simple Storage Service API 참조에서 Amazon S3 API Actions를 참조하세요.

S3의 HTTP `403 Forbidden` 오류를 해결하는 방법에 대한 자세한 내용은 Amazon S3의 액세스 거부(403 Forbidden) 오류 문제 해결 섹션을 참조하세요. S3와 함께 사용할 IAM 기능에 대한 자세한 내용

은 Amazon S3에서 IAM을 사용하는 방법 섹션을 참조하세요. S3 보안 모범 사례에 대한 자세한 내용은 Amazon S3의 보안 모범 사례 섹션을 참조하세요.

주제

- 버킷 작업 및 권한
- 객체 작업 및 권한
- 범용 버킷 작업 및 권한에 대한 액세스 포인트
- 객체 Lambda 액세스 포인트 작업 및 권한
- 다중 리전 액세스 포인트 작업 및 권한
- 배치 운영 작업 및 권한
- S3 Storage Lens 구성 작업 및 권한
- S3 Storage Lens 그룹 작업 및 권한
- S3 Access Grants 인스턴스 작업 및 권한
- S3 Access Grants 위치 작업 및 권한
- S3 Access Grants 권한 부여 작업 및 권한
- 계정 작업 및 권한

버킷 작업 및 권한

버킷 작업은 버킷 리소스 유형에서 작동하는 S3 API 작업입니다. 버킷 정책 또는 IAM ID 기반 정책에서 버킷 작업에 대한 S3 정책 작업을 지정해야 합니다.

정책에서 `Resource` 요소는 버킷 Amazon 리소스 이름(ARN)이어야 합니다. `Resource` 요소 형식 및 정책 예제에 대한 자세한 내용은 버킷 작업 섹션을 참조하세요.

> **Note**
>
> 액세스 포인트 정책에서 버킷 작업에 권한을 부여하려면 다음을 참고하세요.
>
> - 액세스 포인트 정책에서 버킷 작업에 부여된 권한은 기본 버킷이 동일한 권한을 허용하는 경우에만 유효합니다. 액세스 포인트를 사용하는 경우, 버킷의 액세스 제어를 액세스 포인트에 위임하거나 액세스 포인트 정책의 동일한 권한을 기본 버킷 정책에 추가해야 합니다.
> - 버킷 작업에 권한을 부여하는 액세스 포인트 정책에서 `Resource` 요소는 `accesspoint` ARN이어야 합니다. `Resource` 요소 형식 및 정책 예제에 대한 자세한 내용은 범용 버킷의

액세스 포인트에 대한 정책의 버킷 작업 섹션을 참조하세요. 액세스 포인트 정책에 대한 자세한 내용은 액세스 포인트를 사용하도록 IAM 정책 구성 섹션을 참조하세요.

- 액세스 포인트에서 모든 버킷 작업을 지원하는 것은 아닙니다. 자세한 내용은 S3 작업과의 액세스 포인트 호환성 섹션을 참조하세요.

다음은 버킷 작업과 필수 정책 작업을 연결한 것입니다.

API 작업	정책 작업	정책 작업 설명
CreateBucket	(필수) `s3:CreateBucket`	새 S3 버킷을 만드는 데 필요합니다.
	(조건부 필수) `s3:PutBucketAcl`	`CreateBucket` 요청 시 액세스 제어 목록(ACL)을 사용하여 버킷에 대한 권한을 지정하려는 경우 필요합니다.
	(조건부 필수) `s3:PutBucketObjectLockConfiguration` , `s3:PutBucketVersioning`	버킷을 만들 때 Object Lock을 활성화하려는 경우 필요합니다.
	(조건부 필수) `s3:PutBucketOwnershipControls`	버킷을 만들 때 S3 객체 소유권을 지정하려는 경우 필요합니다.
CreateBucketMetadataTableConfiguration	(필수) `s3:CreateBucketMetadataTableConfiguration` , `s3tables:CreateNamespace` , `s3tables:CreateTable` , `s3tables:GetTable` , `s3tables:PutTablePolicy`	범용 버킷에서 메타데이터 테이블 구성을 생성하는 데 필요합니다. 메타데이터 테이블 구성에 지정된 테이블 버킷에 메타데이터 테이블을 생성하려면 지정된 `s3tables` 권한이 있어야 합니다.

API 작업	정책 작업	정책 작업 설명
		또한 메타데이터 테이블을 쿼리할 수 있도록 테이블 버킷을 AWS 분석 서비스와 통합하려면 추가 권한이 필요합니다. 자세한 내용은 Integrating Amazon S3 Tables with AWS analytics services를 참조하세요.
DeleteBucket	(필수) `s3:DeleteBucket`	S3 버킷을 삭제하는 데 필요합니다.
DeleteBucketAnalyticsConfiguration	(필수) `s3:PutAnalyticsConfiguration`	S3 버킷에서 S3 분석 구성을 삭제하는 데 필요합니다.
DeleteBucketCors	(필수) `s3:PutBucketCORS`	버킷에 대한 Cross-Origin Resource Sharing(CORS) 구성을 삭제하는 데 필요합니다.
DeleteBucketEncryption	(필수) `s3:PutEncryptionConfiguration`	Amazon S3 관리형 키(SSE-S3)를 사용하여 S3 버킷의 기본 암호화 구성을 서버 측 암호화로 재설정하는 데 필요합니다.
DeleteBucketIntelligentTieringConfiguration	(필수) `s3:PutIntelligentTieringConfiguration`	S3 버킷에서 기존 S3 Intelligent-Tiering 구성을 삭제하는 데 필요합니다.
DeleteBucketInventoryConfiguration	(필수) `s3:PutInventoryConfiguration`	S3 버킷에서 S3 인벤토리 구성을 삭제하는 데 필요합니다.
DeleteBucketLifecycle	(필수) `s3:PutLifecycleConfiguration`	S3 버킷에 대한 S3 수명 주기 구성을 삭제하는 데 필요합니다.

API 작업	정책 작업	정책 작업 설명
DeleteBucketMetadataTableConfiguration	(필수) s3:DeleteBucketMetadataTableConfiguration	범용 버킷에서 메타데이터 테이블 구성을 삭제하는 데 필요합니다.
DeleteBucketMetricsConfiguration	(필수) s3:PutMetricsConfiguration	S3 버킷에서 Amazon CloudWatch 요청 지표에 대한 지표 구성을 삭제하는 데 필요합니다.
DeleteBucketOwnershipControls	(필수) s3:PutBucketOwnershipControls	S3 버킷에 대한 객체 소유권 설정을 제거하는 데 필요합니다. 제거 후에는 객체 소유권 설정이 Object writer가 됩니다.
DeleteBucketPolicy	(필수) s3:DeleteBucketPolicy	S3 버킷의 정책을 삭제하는 데 필요합니다.
DeleteBucketReplication	(필수) s3:PutReplicationConfiguration	S3 버킷의 복제 구성을 삭제하는 데 필요합니다.
DeleteBucketTagging	(필수) s3:PutBucketTagging	S3 버킷에서 태그를 삭제하는 데 필요합니다.
DeleteBucketWebsite	(필수) s3:DeleteBucketWebsite	S3 버킷에 대한 웹 사이트 구성을 제거하는 데 필요합니다.
DeletePublicAccessBlock(버킷 수준)	(필수) s3:PutBucketPublicAccessBlock	S3 버킷의 퍼블릭 액세스 차단 구성을 제거하는 데 필요합니다.
GetBucketAccelerateConfiguration	(필수) s3:GetAccelerateConfiguration	가속화 하위 리소스를 사용하여 버킷의 Amazon S3 Transfer Acceleration 상태(활성화 또는 일시 중단)를 반환하는 데 필요합니다.

API 작업	정책 작업	정책 작업 설명
GetBucketAcl	(필수) s3:GetBucketAcl	S3 버킷의 액세스 제어 목록(ACL)을 반환하는 데 필요합니다.
GetBucketAnalyticsConfiguration	(필수) s3:GetAnalyticsConfiguration	S3 버킷에서 분석 구성 ID로 식별되는 분석 구성을 반환하는 데 필요합니다.
GetBucketCors	(필수) s3:GetBucketCORS	S3 버킷에 대한 Cross-Origin Resource Sharing(CORS) 구성을 반환하는 데 필요합니다.
GetBucketEncryption	(필수) s3:GetEncryptionConfiguration	S3 버킷에 대한 기본 암호화 구성을 반환하는 데 필요합니다.
GetBucketIntelligentTieringConfiguration	(필수) s3:GetIntelligentTieringConfiguration	S3 버킷의 S3 Intelligent-Tiering 구성을 가져오는 데 필요합니다.
GetBucketInventoryConfiguration	(필수) s3:GetInventoryConfiguration	버킷에서 인벤토리 구성 ID로 식별된 인벤토리 구성을 반환하는 데 필요합니다.
GetBucketLifecycle	(필수) s3:GetLifecycleConfiguration	버킷의 S3 수명 주기 구성을 반환하는 데 필요합니다.
GetBucketLocation	(필수) s3:GetBucketLocation	S3 버킷이 상주하는 AWS 리전을 반환하는 데 필요합니다.
GetBucketLogging	(필수) s3:GetBucketLogging	S3 버킷의 로깅 상태와 사용자가 해당 상태를 보고 수정해야 하는 권한을 반환하는 데 필요합니다.

API 작업	정책 작업	정책 작업 설명
GetBucketMetadataTableConfiguration	(필수) s3:GetBucketMetadataTableConfiguration	범용 버킷에 대한 메타데이터 테이블 구성을 검색하는 데 필요합니다.
GetBucketMetricsConfiguration	(필수) s3:GetMetricsConfiguration	버킷에서 지표 구성 ID로 지정된 지표 구성을 가져오는 데 필요합니다.
GetBucketNotificationConfiguration	(필수) s3:GetBucketNotification	S3 버킷의 알림 구성을 반환하는 데 필요합니다.
GetBucketOwnershipControls	(필수) s3:GetBucketOwnershipControls	S3 버킷에 대한 객체 소유권 설정을 검색하는 데 필요합니다.
GetBucketPolicy	(필수) s3:GetBucketPolicy	S3 버킷의 정책을 반환하는 데 필요합니다.
GetBucketPolicyStatus	(필수) s3:GetBucketPolicyStatus	버킷의 공개 여부를 나타내는 S3 버킷의 정책 상태를 검색하는 데 필요합니다.
GetBucketReplication	(필수) s3:GetReplicationConfiguration	S3 버킷의 복제 구성을 반환하는 데 필요합니다.
GetBucketRequestPayment	(필수) s3:GetBucketRequestPayment	S3 버킷의 지불 요청 구성을 반환하는 데 필요합니다.
GetBucketVersioning	(필수) s3:GetBucketVersioning	S3 버킷의 버전 관리 상태를 반환하는 데 필요합니다.
GetBucketTagging	(필수) s3:GetBucketTagging	S3 버킷과 연결된 태그 집합을 반환하는 데 필요합니다.
GetBucketWebsite	(필수) s3:GetBucketWebsite	S3 버킷에 대한 웹 사이트 구성을 반환하는 데 필요합니다.

API 작업	정책 작업	정책 작업 설명
GetObjectLockConfiguration	(필수) s3:GetBucketObjectLockConfiguration	S3 버킷에 대한 Object Lock 구성을 가져오는 데 필요합니다.
GetPublicAccessBlock(버킷 수준)	(필수) s3:GetBucketPublicAccessBlock	S3 버킷의 퍼블릭 액세스 차단 구성을 검색하는 데 필요합니다.
HeadBucket	(필수) s3:ListBucket	버킷의 존재 여부와 버킷에 액세스할 수 있는 권한이 있는지 확인하는 데 필요합니다.
ListBucketAnalyticsConfigurations	(필수) s3:GetAnalyticsConfiguration	S3 버킷에 대한 분석 구성을 나열하는 데 필요합니다.
ListBucketIntelligentTieringConfigurations	(필수) s3:GetIntelligentTieringConfiguration	S3 버킷의 S3 Intelligent-Tiering 구성을 나열하는 데 필요합니다.
ListBucketInventoryConfigurations	(필수) s3:GetInventoryConfiguration	S3 버킷의 인벤토리 구성 목록을 반환하는 데 필요합니다.
ListBucketMetricsConfigurations	(필수) s3:GetMetricsConfiguration	S3 버킷의 지표 구성을 나열하는 데 필요합니다.
ListObjects	(필수) s3:ListBucket	S3 버킷에서 객체를 일부 또는 전부(최대 1,000개) 나열하는 데 필요합니다.
	(조건부 필수) s3:GetObjectAcl	객체 소유자 정보를 표시하려는 경우 필요합니다.
ListObjectsV2	(필수) s3:ListBucket	S3 버킷에서 객체를 일부 또는 전부(최대 1,000개) 나열하는 데 필요합니다.

API 작업	정책 작업	정책 작업 설명
	(조건부 필수) s3:GetObjectAcl	객체 소유자 정보를 표시하려는 경우 필요합니다.
ListObjectVersions	(필수) s3:ListBucketVersions	S3 버킷의 모든 객체 버전에 대한 메타데이터를 가져오는 데 필요합니다.
PutBucketAccelerateConfiguration	(필수) s3:PutAccelerateConfiguration	기존 버킷의 가속화 구성을 설정하는 데 필요합니다.
PutBucketAcl	(필수) s3:PutBucketAcl	액세스 제어 목록(ACL)을 사용하여 기존 버킷에 대한 권한을 설정하는 데 필요합니다.
PutBucketAnalyticsConfiguration	(필수) s3:PutAnalyticsConfiguration	S3 버킷에 대한 분석 구성을 설정하는 데 필요합니다.
PutBucketCors	(필수) s3:PutBucketCORS	S3 버킷에 대한 Cross-Origin Resource Sharing(CORS) 구성을 설정하는 데 필요합니다.
PutBucketEncryption	(필수) s3:PutEncryptionConfiguration	S3 버킷의 기본 암호화를 구성하는 데 필요합니다.
PutBucketIntelligentTieringConfiguration	(필수) s3:PutIntelligentTieringConfiguration	S3 Intelligent-Tiering 구성을 S3 버킷에 적용하는 데 필요합니다.
PutBucketInventoryConfiguration	(필수) s3:PutInventoryConfiguration	S3 버킷에 인벤토리 구성을 추가하는 데 필요합니다.
PutBucketLifecycle	(필수) s3:PutLifecycleConfiguration	새 S3 수명 주기 구성을 만들거나 S3 버킷의 기존 수명 주기 구성을 교체하는 데 필요합니다.

API 작업	정책 작업	정책 작업 설명
PutBucketLogging	(필수) s3:PutBucketLogging	S3 버킷의 로깅 파라미터를 설정하고 로깅 파라미터를 보고 수정할 수 있는 사람에 대한 권한을 지정하는 데 필요합니다.
PutBucketMetricsConfiguration	(필수) s3:PutMetricsConfiguration	S3 버킷의 Amazon CloudWatch 요청 지표에 대한 지표 구성을 설정하거나 업데이트하는 데 필요합니다.
PutBucketNotificationConfiguration	(필수) s3:PutBucketNotification	S3 버킷에 지정된 이벤트 알림을 활성화하는 데 필요합니다.
PutBucketOwnershipControls	(필수) s3:PutBucketOwnershipControls	S3 버킷에 대한 객체 소유권 설정을 만들거나 수정하는 데 필요합니다.
PutBucketPolicy	(필수) s3:PutBucketPolicy	버킷에 S3 버킷 정책을 적용하는 데 필요합니다.
PutBucketReplication	(필수) s3:PutReplicationConfiguration	새 복제 구성을 만들거나 S3 버킷의 기존 복제 구성을 교체하는 데 필요합니다.
PutBucketRequestPayment	(필수) s3:PutBucketRequestPayment	버킷의 지불 요청 구성을 설정하는 데 필요합니다.
PutBucketTagging	(필수) s3:PutBucketTagging	S3 버킷에 태그 집합을 추가하는 데 필요합니다.
PutBucketVersioning	(필수) s3:PutBucketVersioning	S3 버킷의 버전 관리 상태를 설정하는 데 필요합니다.
PutBucketWebsite	(필수) s3:PutBucketWebsite	버킷을 웹 사이트로 구성하고 웹 사이트의 구성을 설정하는 데 필요합니다.

API 작업	정책 작업	정책 작업 설명
PutObjectLockConfiguration	(필수) s3:PutBucketObjectLockConfiguration	S3 버킷에 Object Lock 구성을 적용하는 데 필요합니다.
PutPublicAccessBlock(버킷 수준)	(필수) s3:PutBucketPublicAccessBlock	S3 버킷에 대한 퍼블릭 액세스 차단 구성을 만들거나 수정하는 데 필요합니다.

객체 작업 및 권한

객체 작업은 객체 리소스 유형에서 작동하는 S3 API 작업입니다. 리소스 기반 정책(예: 버킷 정책, 액세스 포인트 정책, 다중 리전 액세스 포인트 정책, VPC 엔드포인트 정책) 또는 IAM ID 기반 정책에서 객체 작업에 대한 S3 정책 작업을 지정해야 합니다.

정책에서 Resource 요소는 객체 ARN이어야 합니다. Resource 요소 형식 및 정책 예제에 대한 자세한 내용은 객체 작업 섹션을 참조하세요.

> **Note**
>
> - AWS KMS 정책 작업(kms:GenerateDataKey 및 kms:Decrypt)은 AWS KMS 리소스 유형에만 적용되며 IAM ID 기반 정책 및 AWS KMS 리소스 기반 정책(AWS KMS 키 정책)에서 지정해야 합니다. S3 버킷 정책과 같은 S3 리소스 기반 정책에서는 AWS KMS 정책 작업을 지정할 수 없습니다.
> - 액세스 포인트를 사용하여 객체 작업에 대한 액세스를 제어하는 경우 액세스 포인트 정책을 사용할 수 있습니다. 액세스 포인트 정책에서 객체 작업에 권한을 부여하려면 다음을 참고하세요.
>   - 객체 작업에 권한을 부여하는 액세스 포인트 정책에서 Resource 요소는 액세스 포인트를 통해 액세스하는 객체의 ARN이어야 합니다. Resource 요소 형식 및 정책 예제에 대한 자세한 내용은 액세스 포인트 정책의 버킷 작업 섹션을 참조하세요.
>   - 액세스 포인트에서 모든 객체 작업을 지원하는 것은 아닙니다. 자세한 내용은 S3 작업과의 액세스 포인트 호환성 섹션을 참조하세요.
> - 다중 리전 액세스 포인트에서 모든 객체 작업을 지원하는 것은 아닙니다. 자세한 내용은 S3 작업과 다중 리전 액세스 포인트의 호환성 섹션을 참조하세요.

다음은 객체 작업과 필수 정책 작업을 연결한 것입니다.

API 작업	정책 작업	정책 작업 설명
AbortMultipartUpload	(필수) s3:AbortMultipartUpload	멀티파트 업로드를 중단하는 데 필요합니다.
CompleteMultipartUpload	(필수) s3:PutObject	멀티파트 업로드를 완료하는 데 필요합니다.
	(조건부 필수) kms:Decrypt	AWS KMS 고객 관리형 키로 암호화된 객체에 대한 멀티파트 업로드를 완료하려는 경우에 필요합니다.
CopyObject	소스 객체의 경우:	소스 객체의 경우:
	(필수) s3:GetObject 또는 s3:GetObjectVersion	• s3:GetObject - 요청에 versionId 를 지정하지 않고 소스 버킷의 객체를 복사하려는 경우 필요합니다. • s3:GetObjectVersion - 요청에 versionId 를 지정하여 소스 버킷의 특정 객체 버전을 복사하려는 경우 필요합니다.
	(조건부 필수) kms:Decrypt	소스 버킷에서 AWS KMS 고객 관리형 키로 암호화된 객체를 복사하려는 경우 필요합니다.
	대상 객체의 경우:	대상 객체의 경우:
	(필수) s3:PutObject	복사한 객체를 대상 버킷에 넣는 데 필요합니다.
	(조건부 필수) s3:PutObjectAcl	CopyObject 요청 시 복사한 객체를 객체 액세스 제어 목록

API 작업	정책 작업	정책 작업 설명
		(ACL)과 함께 대상 버킷에 넣으려는 경우 필요합니다.
	(조건부 필수) s3:PutObjectTagging	CopyObject 요청 시 객체 태그가 지정된 복사된 객체를 대상 버킷에 넣으려는 경우 필요합니다.
	(조건부 필수) kms:GenerateDataKey	AWS KMS 고객 관리형 키로 복사한 객체를 암호화하여 대상 버킷에 넣으려는 경우 필요합니다.
	(조건부 필수) s3:PutObjectRetention	새 객체에 대해 Object Lock 보존 구성을 설정하려는 경우 필요합니다.
	(조건부 필수) s3:PutObjectLegalHold	새 객체에 Object Lock 법적 보존을 적용하려는 경우 필요합니다.
CreateMultipartUpload	(필수) s3:PutObject	멀티파트 업로드를 만드는 데 필요합니다.
	(조건부 필수) s3:PutObjectAcl	업로드된 객체에 대한 객체 액세스 제어 목록(ACL) 권한을 설정하려는 경우 필요합니다.
	(조건부 필수) s3:PutObjectTagging	업로드된 객체에 객체 태그 지정을 추가하려는 경우 필요합니다.
	(조건부 필수) kms:GenerateDataKey	멀티파트 업로드를 시작할 때 AWS KMS 고객 관리형 키를 사용하여 객체를 암호화하려는 경우 필요합니다.

API 작업	정책 작업	정책 작업 설명
	(조건부 필수) s3:PutObjectRetention	업로드된 객체에 대해 Object Lock 보존 구성을 설정하려는 경우 필요합니다.
	(조건부 필수) s3:PutObjectLegalHold	업로드된 객체에 Object Lock 법적 보존을 적용하려는 경우 필요합니다.
DeleteObject	(필수) s3:DeleteObject 또는 s3:DeleteObjectVersion	• s3:DeleteObject - 요청에 versionId 를 지정하지 않고 객체를 제거하려는 경우 필요합니다. • s3:DeleteObjectVersion - 요청에 versionId 를 지정하여 객체의 특정 버전을 제거하려는 경우 필요합니다.
	(조건부 필수) s3:BypassGovernanceRetention	Object Lock 보존을 위한 거버넌스 모드로 보호되는 객체를 삭제하려는 경우 필요합니다.
DeleteObjects	(필수) s3:DeleteObject 또는 s3:DeleteObjectVersion	• s3:DeleteObject - 요청에 versionId 를 지정하지 않고 객체를 제거하려는 경우 필요합니다. • s3:DeleteObjectVersion - 요청에 versionId 를 지정하여 객체의 특정 버전을 제거하려는 경우 필요합니다.
	(조건부 필수) s3:BypassGovernanceRetention	Object Lock 보존을 위한 거버넌스 모드로 보호되는 객체를 삭제하려는 경우 필요합니다.

API 작업	정책 작업	정책 작업 설명
DeleteObjectTagging	(필수) s3:DeleteObjectTagging 또는 s3:DeleteObjectVersionTagging	• s3:DeleteObjectTagging - 요청에 versionId 를 지정하지 않고 객체의 전체 태그 집합을 제거하려는 경우 필요합니다. • s3:DeleteObjectVersionTagging - 요청에 versionId 를 지정하여 특정 객체 버전의 태그를 삭제하려는 경우 필요합니다.
GetObject	(필수) s3:GetObject 또는 s3:GetObjectVersion	• s3:GetObject - 요청에 versionId 를 지정하지 않고 객체를 가져오려는 경우 필요합니다. • s3:GetObjectVersion - 요청에 versionId 를 지정하여 객체의 특정 버전을 가져오려는 경우 필요합니다.
	(조건부 필수) kms:Decrypt	AWS KMS 고객 관리형 키로 암호화된 객체를 가져와 해독하려는 경우 필요합니다.
	(조건부 필수) s3:GetObjectTagging	GetObject 요청 시 객체의 태그 집합을 가져오려는 경우 필요합니다.
	(조건부 필수) s3:GetObjectLegalHold	객체의 현재 Object Lock 법적 보존 상태를 가져오려는 경우 필요합니다.

API 작업	정책 작업	정책 작업 설명
	(조건부 필수) s3:GetObjectRetention	객체에 대한 Object Lock 보존 설정을 검색하려는 경우 필요합니다.
GetObjectAcl	(필수) s3:GetObjectAcl 또는 s3:GetObjectVersionAcl	• s3:GetObjectAcl - 요청에 versionId 를 지정하지 않고 객체의 액세스 제어 목록(ACL)을 가져오려는 경우 필요합니다. • s3:GetObjectVersionAcl - 요청에 versionId 를 지정하여 객체의 액세스 제어 목록(ACL)을 가져오려는 경우 필요합니다.
GetObjectAttributes	(필수) s3:GetObject 또는 s3:GetObjectVersion	• s3:GetObject - 요청에 versionId 를 지정하지 않고 객체와 관련된 속성을 검색하려는 경우 필요합니다. • s3:GetObjectVersion - 요청에 versionId 를 지정하여 특정 객체 버전과 관련된 속성을 검색하려는 경우 필요합니다.
	(조건부 필수) kms:Decrypt	AWS KMS 고객 관리형 키로 암호화된 객체와 관련된 속성을 검색하려는 경우 필요합니다.
GetObjectLegalHold	(필수) s3:GetObjectLegalHold	객체의 현재 Object Lock 법적 보존 상태를 가져오는 데 필요합니다.

API 작업	정책 작업	정책 작업 설명
GetObjectRetention	(필수) s3:GetObjectRetention	객체에 대한 Object Lock 보존 설정을 검색하는 데 필요합니다.
GetObjectTagging	(필수) s3:GetObjectTagging 또는 s3:GetObjectVersionTagging	• s3:GetObjectTagging - 요청에 versionId 를 지정하지 않고 객체의 태그 집합을 가져오려는 경우 필요합니다. • s3:GetObjectVersionTagging - 요청에 versionId 를 지정하여 특정 객체 버전의 태그를 가져오려는 경우 필요합니다.
GetObjectTorrent	(필수) s3:GetObject	객체의 토렌트 파일을 반환하는 데 필요합니다.
HeadObject	(필수) s3:GetObject	객체 자체를 반환하지 않고 객체에서 메타데이터를 검색하는 데 필요합니다.
	(조건부 필수) s3:GetObjectLegalHold	객체의 현재 Object Lock 법적 보존 상태를 가져오려는 경우 필요합니다.
	(조건부 필수) s3:GetObjectRetention	객체에 대한 Object Lock 보존 설정을 검색하려는 경우 필요합니다.
ListMultipartUploads	(필수) s3:ListBucketMultipartUploads	버킷에서 진행 중인 멀티파트 업로드를 나열하는 데 필요합니다.

API 작업	정책 작업	정책 작업 설명
ListParts	(필수) s3:ListMultipartUploadParts	특정 멀티파트 업로드에 대해 업로드된 파트를 나열하는 데 필요합니다.
	(조건부 필수) kms:Decrypt	AWS KMS 고객 관리형 키로 암호화된 멀티파트 업로드의 파트를 나열하려는 경우 필요합니다.
PutObject	(필수) s3:PutObject	객체를 넣는 데 필요합니다.
	(조건부 필수) s3:PutObjectAcl	PutObject 요청 시 객체 액세스 제어 목록(ACL)을 적용하려는 경우 필요합니다.
	(조건부 필수) s3:PutObjectTagging	PutObject 요청 시 객체 태그 지정을 적용하려는 경우 필요합니다.
	(조건부 필수) kms:GenerateDataKey	AWS KMS 고객 관리형 키로 객체를 암호화하려는 경우 필요합니다.
	(조건부 필수) s3:PutObjectRetention	객체에 Object Lock 보존 구성을 설정하려는 경우 필요합니다.
	(조건부 필수) s3:PutObjectLegalHold	지정된 객체에 Object Lock 법적 보존 구성을 적용하려는 경우 필요합니다.

API 작업	정책 작업	정책 작업 설명
PutObjectAcl	(필수) s3:PutObjectAcl 또는 s3:PutObjectVersionAcl	• s3:PutObjectAcl - 요청에 versionId 를 지정하지 않고 새 객체 또는 기존 객체에 대한 액세스 제어 목록(ACL) 권한을 설정하려는 경우에 필요합니다. • s3:PutObjectVersionAcl - 요청에 versionId 를 지정하여 새 객체 또는 기존 객체에 대한 액세스 제어 목록(ACL) 권한을 설정하려는 경우에 필요합니다.
PutObjectLegalHold	(필수) s3:PutObjectLegalHold	Object Lock 법적 보존 구성을 객체에 적용하는 데 필요합니다.
PutObjectRetention	(필수) s3:PutObjectRetention	Object Lock 보존 구성을 객체에 적용하는 데 필요합니다.
	(조건부 필수) s3:BypassGovernanceRetention	Object Lock 보존 구성의 거버넌스 모드를 우회하려는 경우 필요합니다.

API 작업	정책 작업	정책 작업 설명
PutObjectTagging	(필수) s3:PutObjectTagging 또는 s3:PutObjectVersionTagging	• s3:PutObjectTagging - 요청에 versionId 를 지정하지 않고 제공된 태그 집합을 버킷에 이미 존재하는 객체로 설정하려는 경우에 필요합니다. • s3:PutObjectVersionTagging - 요청에 versionId 를 지정하여 제공된 태그 집합을 버킷에 이미 존재하는 객체로 설정하려는 경우에 필요합니다.
RestoreObject	(필수) s3:RestoreObject	아카이브된 객체의 사본을 복원하는 데 필요합니다.
SelectObjectContent	(필수) s3:GetObject	간단한 구조화된 쿼리 언어(SQL) 문을 기반으로 S3 객체의 내용을 필터링하는 데 필요합니다.
	(조건부 필수) kms:Decrypt	AWS KMS 고객 관리형 키로 암호화된 S3 객체의 콘텐츠를 필터링하려는 경우 필요합니다.
UploadPart	(필수) s3:PutObject	멀티파트 업로드에서 파트를 업로드하는 데 필요합니다.
	(조건부 필수) kms:GenerateDataKey	업로드 파트를 넣고 AWS KMS 고객 관리형 키로 암호화하려는 경우 필요합니다.
UploadPartCopy	소스 객체의 경우:	소스 객체의 경우:

API 작업	정책 작업	정책 작업 설명
	(필수) `s3:GetObject` 또는 `s3:GetObjectVersion`	• `s3:GetObject` - 요청에 `versionId` 를 지정하지 않고 소스 버킷의 객체를 복사하려는 경우 필요합니다. • `s3:GetObjectVersion` - 요청에 `versionId` 를 지정하여 소스 버킷의 특정 객체 버전을 복사하려는 경우 필요합니다.
	(조건부 필수) `kms:Decrypt`	소스 버킷에서 AWS KMS 고객 관리형 키로 암호화된 객체를 복사하려는 경우 필요합니다.
	대상 파트의 경우:	대상 파트의 경우:
	(필수) `s3:PutObject`	대상 버킷에 멀티파트 업로드 파트를 업로드하는 데 필요합니다.
	(조건부 필수) `kms:GenerateDataKey`	대상 버킷에 파트를 업로드할 때 AWS KMS 고객 관리형 키로 파트를 암호화하려는 경우 필요합니다.

범용 버킷 작업 및 권한에 대한 액세스 포인트

액세스 포인트 작업은 `accesspoint` 리소스 유형에서 작동하는 S3 API 작업입니다. 버킷 정책 또는 액세스 포인트 정책이 아닌 IAM ID 기반 정책에서 액세스 포인트 작업에 대한 S3 정책 작업을 지정해야 합니다.

정책에서 `Resource` 요소는 `accesspoint` ARN이어야 합니다. `Resource` 요소 형식 및 정책 예제에 대한 자세한 내용은 범용 버킷의 액세스 포인트 작업 섹션을 참조하세요.

> **Note**
>
> 액세스 포인트를 사용하여 버킷 또는 객체 작업에 대한 액세스를 제어하려면 다음 사항을 참고하세요.
>
> - 액세스 포인트를 사용하여 버킷 작업에 대한 액세스를 제어하는 방법은 범용 버킷의 액세스 포인트에 대한 정책의 버킷 작업 섹션을 참조하세요.
> - 액세스 포인트를 사용하여 객체 작업에 대한 액세스를 제어하는 방법은 액세스 포인트 정책의 버킷 작업 섹션을 참조하세요.
> - 액세스 포인트 정책을 구성하는 방법에 대한 자세한 내용은 액세스 포인트를 사용하도록 IAM 정책 구성 섹션을 참조하세요.

다음은 액세스 포인트 작업과 필수 정책 작업을 연결한 것입니다.

API 작업	정책 작업	정책 작업 설명
CreateAccessPoint	(필수) `s3:CreateAccessPoint`	S3 버킷과 연결된 액세스 포인트를 만드는 데 필요합니다.
DeleteAccessPoint	(필수) `s3:DeleteAccessPoint`	액세스 포인트를 삭제하는 데 필요합니다.
DeleteAccessPointPolicy	(필수) `s3:DeleteAccessPointPolicy`	액세스 포인트 정책을 삭제하는 데 필요합니다.
GetAccessPointPolicy	(필수) `s3:GetAccessPointPolicy`	액세스 포인트 정책을 검색하는 데 필요합니다.
GetAccessPointPolicyStatus	(필수) `s3:GetAccessPointPolicyStatus`	지정된 액세스 포인트에 현재 퍼블릭 액세스를 허용하는 정책이 있는지 여부에 대한 정보를 검색하는 데 필요합니다.
PutAccessPointPolicy	(필수) `s3:PutAccessPointPolicy`	액세스 포인트 정책을 적용하는 데 필요합니다.

객체 Lambda 액세스 포인트 작업 및 권한

객체 Lambda 액세스 포인트 작업은 `objectlambdaaccesspoint` 리소스 유형에서 작동하는 S3 API 작업입니다. 객체 Lambda 액세스 포인트 작업용 정책을 구성하는 방법에 대한 자세한 내용은 객체 Lambda 액세스 포인트에 대한 IAM 정책 구성 섹션을 참조하세요.

다음은 객체 Lambda 액세스 포인트 작업과 필수 정책 작업을 연결한 것입니다.

API 작업	정책 작업	정책 작업 설명
CreateAccessPointForObjectLambda	(필수) `s3:CreateAccessPointForObjectLambda`	객체 Lambda 액세스 포인트를 만드는 데 필요합니다.
DeleteAccessPointForObjectLambda	(필수) `s3:DeleteAccessPointForObjectLambda`	지정된 객체 Lambda 액세스 포인트를 삭제하는 데 필요합니다.
DeleteAccessPointPolicyForObjectLambda	(필수) `s3:DeleteAccessPointPolicyForObjectLambda`	지정된 객체 Lambda 액세스 포인트에서 정책을 삭제하는 데 필요합니다.
GetAccessPointConfigurationForObjectLambda	(필수) `s3:GetAccessPointConfigurationForObjectLambda`	객체 Lambda 액세스 포인트의 구성을 검색하는 데 필요합니다.
GetAccessPointForObjectLambda	(필수) `s3:GetAccessPointForObjectLambda`	객체 Lambda 액세스 포인트에 대한 정보를 검색하는 데 필요합니다.
GetAccessPointPolicyForObjectLambda	(필수) `s3:GetAccessPointPolicyForObjectLambda`	지정된 객체 Lambda 액세스 포인트와 연결된 액세스 포인트 정책을 반환하는 데 필요합니다.
GetAccessPointPolicyStatusForObjectLambda	(필수) `s3:GetAccessPointPolicyStatusForObjectLambda`	특정 객체 Lambda 액세스 포인트 정책의 정책 상태를 반환하는 데 필요합니다.

API 작업	정책 작업	정책 작업 설명
PutAccessPointConfigurationForObjectLambda	(필수) s3:PutAccessPointConfigurationForObjectLambda	객체 Lambda 액세스 포인트의 구성을 설정하는 데 필요합니다.
PutAccessPointPolicyForObjectLambda	(필수) s3:PutAccessPointPolicyForObjectLambda	액세스 정책을 지정된 객체 Lambda 액세스 포인트에 연결하는 데 필요합니다.

다중 리전 액세스 포인트 작업 및 권한

다중 리전 액세스 포인트 작업은 multiregionaccesspoint 리소스 유형에서 작동하는 S3 API 작업입니다. 다중 리전 액세스 포인트 작업용 정책을 구성하는 방법에 대한 자세한 내용은 다중 리전 액세스 포인트 정책 예시 섹션을 참조하세요.

다음은 다중 리전 액세스 포인트 작업과 필수 정책 작업을 연결한 것입니다.

API 작업	정책 작업	정책 작업 설명
CreateMultiRegionAccessPoint	(필수) s3:CreateMultiRegionAccessPoint	다중 리전 액세스 포인트를 만들고 S3 버킷에 연결하는 데 필요합니다.
DeleteMultiRegionAccessPoint	(필수) s3:DeleteMultiRegionAccessPoint	다중 리전 액세스 포인트를 삭제하는 데 필요합니다.
DescribeMultiRegionAccessPointOperation	(필수) s3:DescribeMultiRegionAccessPointOperation	다중 리전 액세스 포인트를 관리하기 위한 비동기식 요청 상태를 검색하는 데 필요합니다.
GetMultiRegionAccessPoint	(필수) s3:GetMultiRegionAccessPoint	지정된 다중 리전 액세스 포인트에 대한 구성 정보를 반환하는 데 필요합니다.

API 작업	정책 작업	정책 작업 설명
GetMultiRegionAccessPointPolicy	(필수) s3:GetMultiRegionAccessPointPolicy	지정된 다중 리전 액세스 포인트의 액세스 제어 정책을 반환하는 데 필요합니다.
GetMultiRegionAccessPointPolicyStatus	(필수) s3:GetMultiRegionAccessPointPolicyStatus	지정된 다중 리전 액세스 포인트에 퍼블릭 액세스를 허용하는 액세스 제어 정책이 있는지 여부에 대한 특정 다중 리전 액세스 포인트의 정책 상태를 반환하는 데 필요합니다.
GetMultiRegionAccessPointRoutes	(필수) s3:GetMultiRegionAccessPointRoutes	다중 리전 액세스 포인트에 대한 경로 구성을 반환하는 데 필요합니다.
PutMultiRegionAccessPointPolicy	(필수) s3:PutMultiRegionAccessPointPolicy	지정된 다중 리전 액세스 포인트의 액세스 제어 정책을 업데이트하는 데 필요합니다.
SubmitMultiRegionAccessPointRoutes	(필수) s3:SubmitMultiRegionAccessPointRoutes	다중 리전 액세스 포인트에 대한 업데이트된 경로 구성을 제출하는 데 필요합니다.

배치 운영 작업 및 권한

(배치 작업) 운영 작업은 `job` 리소스 유형에서 작동하는 S3 API 작업입니다. 버킷 정책이 아닌 IAM ID 기반 정책에서 작업에 대한 S3 정책 작업을 지정해야 합니다.

정책에서 `Resource` 요소는 `job` ARN이어야 합니다. `Resource` 요소 형식 및 정책 예제에 대한 자세한 내용은 배치 작업 섹션을 참조하세요.

다음은 배치 운영 작업과 필수 정책 작업을 연결한 것입니다.

API 작업	정책 작업	정책 작업 설명
DeleteJobTagging	(필수) s3:DeleteJobTagging	기존 S3 배치 운영 작업에서 태그를 제거하는 데 필요합니다.
DescribeJob	(필수) s3:DescribeJob	배치 운영 작업의 구성 파라미터 및 상태를 검색하는 데 필요합니다.
GetJobTagging	(필수) s3:GetJobTagging	기존 S3 배치 운영 작업의 태그 집합을 반환하는 데 필요합니다.
PutJobTagging	(필수) s3:PutJobTagging	기존 S3 배치 운영 작업에 태그를 넣거나 교체하는 데 필요합니다.
UpdateJobPriority	(필수) s3:UpdateJobPriority	기존 작업의 우선순위를 업데이트하는 데 필요합니다.
UpdateJobStatus	(필수) s3:UpdateJobStatus	지정된 작업의 상태를 업데이트하는 데 필요합니다.

S3 Storage Lens 구성 작업 및 권한

S3 Storage Lens 구성 작업은 `storagelensconfiguration` 리소스 유형에서 작동하는 S3 API 작업입니다. S3 Storage Lens 구성 작업을 구성하는 방법에 대한 자세한 내용은 Amazon S3 Storage Lens 권한 설정 섹션을 참조하세요.

다음은 S3 Storage Lens 구성 작업 및 필수 정책 작업을 연결한 것입니다.

API 작업	정책 작업	정책 작업 설명
DeleteStorageLensConfiguration	(필수) s3:DeleteStorageLensConfiguration	S3 Storage Lens 구성을 삭제하는 데 필요합니다.

API 작업	정책 작업	정책 작업 설명
DeleteStorageLensConfigurationTagging	(필수) s3:DeleteStorageLensConfigurationTagging	S3 Storage Lens 구성 태그를 삭제하는 데 필요합니다.
GetStorageLensConfiguration	(필수) s3:GetStorageLensConfiguration	S3 Storage Lens 구성을 가져오는 데 필요합니다.
GetStorageLensConfigurationTagging	(필수) s3:GetStorageLensConfigurationTagging	S3 Storage Lens 구성의 태그를 가져오는 데 필요합니다.
PutStorageLensConfigurationTagging	(필수) s3:PutStorageLensConfigurationTagging	기존 S3 Storage Lens 구성에 태그를 넣거나 교체하는 데 필요합니다.

S3 Storage Lens 그룹 작업 및 권한

S3 Storage Lens 그룹 작업은 `storagelensgroup` 리소스 유형에서 작동하는 S3 API 작업입니다. S3 Storage Lens 그룹 권한을 구성하는 방법에 대한 자세한 내용은 스토리지 렌즈 그룹 권한 섹션을 참조하세요.

다음은 S3 Storage Lens 그룹 작업 및 필수 정책 작업을 연결한 것입니다.

API 작업	정책 작업	정책 작업 설명
DeleteStorageLensGroup	(필수) s3:DeleteStorageLensGroup	기존 S3 Storage Lens 그룹을 삭제하는 데 필요합니다.
GetStorageLensGroup	(필수) s3:GetStorageLensGroup	S3 Storage Lens 그룹 구성 세부 정보를 검색하는 데 필요합니다.
UpdateStorageLensGroup	(필수) s3:UpdateStorageLensGroup	기존 S3 Storage Lens 그룹을 업데이트하는 데 필요합니다.

API 작업	정책 작업	정책 작업 설명
CreateStorageLensGroup	(필수) s3:CreateStorageLensGroup	새 Storage Lens 그룹을 만드는 데 필요합니다.
CreateStorageLensGroup, TagResource	(필수) s3:CreateStorageLensGroup , s3:TagResource	태그를 포함해 새 Storage Lens 그룹을 만드는 데 필요합니다.
ListStorageLensGroups	(필수) s3:ListStorageLensGroups	홈 리전의 모든 Storage Lens 그룹을 나열하는 데 필요합니다.
ListTagsForResource	(필수) s3:ListTagsForResource	Storage Lens 그룹에 추가된 태그를 나열하는 데 필요합니다.
TagResource	(필수) s3:TagResource	기존 Storage Lens 그룹의 Storage Lens 그룹 태그를 추가 또는 업데이트하는 데 필요합니다.
UntagResource	(필수) s3:UntagResource	Storage Lens 그룹에서 태그를 삭제하는 데 필요합니다.

S3 Access Grants 인스턴스 작업 및 권한

S3 Access Grants 인스턴스 작업은 accessgrantsinstance 리소스 유형에서 작동하는 S3 API 작업입니다. S3 Access Grants 인스턴스는 액세스 권한 부여의 논리적 컨테이너입니다. S3 Access Grants 인스턴스 작업에 대한 자세한 내용은 S3 Access Grants 인스턴스 사용 섹션을 참조하세요.

다음은 S3 Access Grants 인스턴스 구성 작업 및 필수 정책 작업을 연결한 것입니다.

API 작업	정책 작업	정책 작업 설명
AssociateAccessGrantsIdentityCenter	(필수) s3:AssociateAccessGrantsIdentityCenter	AWS IAM Identity Center 인스턴스를 S3 Access Grants 인스턴스와 연결하는 데 필요하므

API 작업	정책 작업	정책 작업 설명
		로 회사 자격 증명 디렉터리에서 사용자 및 그룹에 대한 액세스 권한을 생성할 수 있습니다. 이 경우 다음 권한이 있어야 합니다. `sso:CreateApplication` , `sso:PutApplicationGrant` 및 `sso:PutApplicationAuthenticationMethod` .
CreateAccessGrantsInstance	(필수) `s3:CreateAccessGrantsInstance`	개별 액세스 권한 부여를 위한 컨테이너인 S3 Access Grants 인스턴스(`accessgrantsinstance` 리소스)를 생성하는 데 필요합니다. AWS IAM Identity Center 인스턴스를 S3 Access Grants 인스턴스와 연결하려면, `sso:DescribeInstance` , `sso:CreateApplication` , `sso:PutApplicationGrant` 및 `sso:PutApplicationAuthenticationMethod` 권한도 있어야 합니다.
DeleteAccessGrantsInstance	(필수) `s3:DeleteAccessGrantsInstance`	계정의 AWS 리전에서 S3 Access Grants 인스턴스(`accessgrantsinstance` 리소스)를 삭제하는 데 필요합니다.

API 작업	정책 작업	정책 작업 설명
DeleteAccessGrantsInstanceResourcePolicy	(필수) s3:DeleteAccessGrantsInstanceResourcePolicy	S3 Access Grants 인스턴스의 리소스 정책을 삭제하는 데 필요합니다.
DissociateAccessGrantsIdentityCenter	(필수) s3:DissociateAccessGrantsIdentityCenter	S3 Access Grants 인스턴스에서 AWS IAM Identity Center 인스턴스의 연결을 해제하는 데 필요합니다. 이 경우 다음 권한이 있어야 합니다. sso:DeleteApplication
GetAccessGrantsInstance	(필수) s3:GetAccessGrantsInstance	계정의 AWS 리전에 대한 S3 Access Grants 인스턴스를 검색하는 데 필요합니다.
GetAccessGrantsInstanceForPrefix	(필수) s3:GetAccessGrantsInstanceForPrefix	특정 접두사가 포함된 S3 Access Grants 인스턴스를 검색하는 데 필요합니다.
GetAccessGrantsInstanceResourcePolicy	(필수) s3:GetAccessGrantsInstanceResourcePolicy	S3 Access Grants 인스턴스의 리소스 정책을 반환하는 데 필요합니다.
ListAccessGrantsInstances	(필수) s3:ListAccessGrantsInstances	계정의 S3 Access Grants 인스턴스 목록을 반환하는 데 필요합니다.
PutAccessGrantsInstanceResourcePolicy	(필수) s3:PutAccessGrantsInstanceResourcePolicy	S3 Access Grants 인스턴스 리소스 정책을 업데이트하는 데 필요합니다.

S3 Access Grants 위치 작업 및 권한

S3 Access Grants 위치 작업은 `accessgrantslocation` 리소스 유형에서 작동하는 S3 API 작업입니다. S3 Access Grants 위치 작업에 대한 자세한 내용은 S3 Access Grants 위치 사용 섹션을 참조하세요.

다음은 S3 Access Grants 위치 구성 작업 및 필수 정책 작업을 연결한 것입니다.

API 작업	정책 작업	정책 작업 설명
CreateAccessGrantsLocation	(필수) `s3:CreateAccessGrantsLocation`	S3 Access Grants 인스턴스에 위치를 등록하는 데 필요합니다(`accessgrantslocation` 리소스 생성). 지정된 IAM 역할에 대한 다음 권한도 있어야 합니다. `iam:PassRole`
DeleteAccessGrantsLocation	(필수) `s3:DeleteAccessGrantsLocation`	Amazon S3 Access Grants 인스턴스에서 등록된 위치를 제거하는 데 필요합니다.
GetAccessGrantsLocation	(필수) `s3:GetAccessGrantsLocation`	S3 Access Grants 인스턴스에 등록된 특정 위치의 세부 정보를 검색하는 데 필요합니다.
ListAccessGrantsLocations	(필수) `s3:ListAccessGrantsLocations`	S3 Access Grants 인스턴스에 등록된 위치 목록을 반환하는 데 필요합니다.
UpdateAccessGrantsLocation	(필수) `s3:UpdateAccessGrantsLocation`	S3 Access Grants 인스턴스에 등록된 위치의 IAM 역할을 업데이트하는 데 필요합니다.

S3 Access Grants 권한 부여 작업 및 권한

S3 Access Grants권한 부여 작업은 `accessgrant` 리소스 유형에서 작동하는 S3 API 작업입니다. S3 Access Grants를 사용한 개별 권한 부여 사용에 대한 자세한 내용은 S3 Access Grants에서 권한 부여 사용 섹션을 참조하세요.

다음은 S3 Access Grants 권한 부여 구성 작업 및 필수 정책 작업을 연결한 것입니다.

API 작업	정책 작업	정책 작업 설명
CreateAccessGrant	(필수) `s3:CreateAccessGrant`	S3 Access Grants 인스턴스에서 사용자 또는 그룹에 대한 개별 권한 부여(`accessgrant` 리소스)를 생성하는 데 필요합니다. 이 경우 다음 권한이 있어야 합니다. 디렉터리 자격 증명의 경우 - `sso:DescribeInstance` 및 `sso:DescribeApplication` 디렉터리 사용자의 경우 - `identitystore:DescribeUser`
DeleteAccessGrant	(필수) `s3:DeleteAccessGrant`	S3 Access Grants 인스턴스에서 개별 액세스 권한(`accessgrant` 리소스)을 삭제하는 데 필요합니다.
GetAccessGrant	(필수) `s3:GetAccessGrant`	S3 Access Grants 인스턴스에서 개별 액세스 권한 부여에 대한 세부 정보를 가져오는 데 필요합니다.
ListAccessGrants	(필수) `s3:ListAccessGrants`	S3 Access Grants 인스턴스의 개별 액세스 권한 목록을 반환하는 데 필요합니다.

API 작업	정책 작업	정책 작업 설명
ListCallerAccessGrants	(필수) s3:ListCallerAccessGrants	호출자에게 S3 Access Grants를 통해 Amazon S3 데이터에 대한 액세스 권한을 부여하는 액세스 권한 부여를 나열하는 데 필요합니다.

계정 작업 및 권한

계정 작업은 계정 수준에서 작동하는 S3 API 작업입니다. 계정은 Amazon S3에서 정의한 리소스 유형이 아닙니다. 버킷 정책이 아닌 IAM ID 기반 정책에서 계정 작업에 대한 S3 정책 작업을 지정해야 합니다.

정책에서 Resource 요소는 "*" 기호와 같아야 합니다. 정책 예제에 대한 자세한 내용은 계정 작업 섹션을 참조하세요.

다음은 계정 작업과 필수 정책 작업을 연결한 것입니다.

API 작업	정책 작업	정책 작업 설명
CreateJob	(필수) s3:CreateJob	새 S3 배치 운영 작업을 만드는 데 필요합니다.
CreateStorageLensGroup	(필수) s3:CreateStorageLensGroup	새 S3 Storage Lens 그룹을 만들어 지정된 AWS 계정 ID와 연결하는 데 필요합니다.
	(조건부 필수) s3:TagResource	AWS 리소스 태그가 있는 S3 Storage Lens 그룹을 만들려는 경우 필요합니다.
DeletePublicAccessBlock(계정 수준)	(필수) s3:PutAccountPublicAccessBlock	AWS 계정에서 퍼블릭 액세스 차단 구성을 제거하는 데 필요합니다.

API 작업	정책 작업	정책 작업 설명
GetAccessPoint	(필수) s3:GetAccessPoint	지정한 액세스 포인트에 대한 구성 정보를 검색하는 데 필요합니다.
GetAccessPointPolicy(계정 수준)	(필수) s3:GetAccountPublicAccessBlock	AWS 계정에 대한 퍼블릭 액세스 차단 구성을 검색하는 데 필요합니다.
ListAccessPoints	(필수) s3:ListAccessPoints	AWS 계정에서 소유한 S3 버킷의 액세스 포인트를 나열하는 데 필요합니다.
ListAccessPointsForObjectLambda	(필수) s3:ListAccessPointsForObjectLambda	객체 Lambda 액세스 포인트를 나열하는 데 필요합니다.
ListBuckets	(필수) s3:ListAllMyBuckets	요청의 인증된 발신자가 소유한 모든 버킷의 목록을 반환하는 데 필요합니다.
ListJobs	(필수) s3:ListJobs	현재 작업 및 최근 종료된 작업을 나열하는 데 필요합니다.
ListMultiRegionAccessPoints	(필수) s3:ListMultiRegionAccessPoints	현재 지정된 AWS 계정과 연결된 다중 리전 액세스 포인트 목록을 반환하는 데 필요합니다.
ListStorageLensConfigurations	(필수) s3:ListStorageLensConfigurations	AWS 계정에 대한 S3 Storage Lens 구성 목록을 가져오는 데 필요합니다.
ListStorageLensGroups	(필수) s3:ListStorageLensGroups	지정된 홈 AWS 리전에 있는 모든 S3 Storage Lens 그룹을 나열하는 데 필요합니다.

API 작업	정책 작업	정책 작업 설명
PutPublicAccessBlock(계정 수준)	(필수) s3:PutAccountPublicAccessBlock	AWS 계정에 대한 퍼블릭 액세스 차단 구성을 만들거나 수정하는 데 필요합니다.
PutStorageLensConfiguration	(필수) s3:PutStorageLensConfiguration	S3 Storage Lens 구성을 넣는 데 필요합니다.

Amazon S3의 정책 및 권한

이 페이지에서는 Amazon S3의 버킷 및 사용자 정책에 대한 개요를 제공하고 AWS Identity and Access Management(IAM) 정책의 기본 요소에 대해 설명합니다. 나열된 각 요소는 해당 요소에 대한 자세한 내용과 사용 방법의 예제로 연결됩니다.

Amazon S3 작업, 리소스, 조건의 전체 목록은 서비스 승인 참조에서 Amazon S3에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

가장 기본적인 경우 정책에는 다음 요소가 포함되어 있습니다.

- Resource - 정책이 적용되는 Amazon S3 버킷, 객체, 액세스 포인트 또는 작업입니다. 버킷, 객체, 액세스 포인트 또는 작업의 Amazon 리소스 이름(ARN)을 사용하여 리소스를 식별합니다.

  버킷 수준 작업의 예:

  `"Resource": "arn:aws:s3:::bucket_name"`

  객체 수준 작업의 예:

  - `"Resource": "arn:aws:s3:::bucket_name/*"` 버킷에 있는 모든 객체에 대한 것입니다.
  - `"Resource": "arn:aws:s3:::bucket_name/prefix/*"` 버킷에서 특정 접두사를 갖는 객체에 대한 것입니다.

  자세한 내용은 Amazon S3의 정책 리소스 섹션을 참조하세요.
- Actions – 각 리소스에 대해 Amazon S3에서는 작업의 집합을 지원합니다. 작업 키워드를 사용하여 허용(또는 거부)할 리소스 작업을 식별합니다.

예를 들어 `s3:ListBucket` 권한은 사용자가 Amazon S3 ListObjectsV2 작업을 사용할 수 있도록 허용합니다. (`s3:ListBucket` 권한은 작업(action) 이름이 작업(operation) 이름에 직접 매핑되지 않는 경우입니다.) Amazon S3 작업 사용에 대한 자세한 내용은 Amazon S3의 정책 작업 단원을 참조하십시오. Amazon S3 작업의 전체 목록은 Amazon Simple Storage Service API Reference에서 작업을 참조하십시오.

- Effect - 사용자가 특정 작업을 요청하는 경우의 결과입니다. 이는 `Allow` 또는 `Deny` 중에 하나가 될 수 있습니다.

  명시적으로 리소스에 대한 액세스 권한을 부여(허용)하지 않는 경우, 액세스는 묵시적으로 거부됩니다. 리소스에 대한 액세스를 명시적으로 거부할 수도 있습니다. 다른 정책에서 액세스 권한을 부여하더라도 사용자가 해당 리소스에 액세스할 수 없도록 하려고 할 때 이러한 작업을 수행할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소: 효과를 참조하세요.
- Principal - 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 버킷 정책의 보안 주체 섹션을 참조하세요.
- Condition – 정책이 적용되기 위한 조건입니다. AWS 전역 키와 Amazon S3 전용 키를 사용하여 Amazon S3 액세스 정책에서 조건을 지정할 수 있습니다. 자세한 내용은 조건 키를 사용한 버킷 정책 예시 섹션을 참조하세요.

다음 버킷 정책 예시는 `Effect`, `Principal`, `Action`, `Resource` 요소를 나타냅니다. 이 정책에서는 *123456789012* 계정의 사용자인 *Akua*에게 *amzn-s3-demo-bucket1* 버킷에 대한 `s3:GetObject`, `s3:GetBucketLocation`, `s3:ListBucket` Amazon S3 권한을 허용합니다.

```
{
    "Version": "2012-10-17",
    "Id": "ExamplePolicy01",
    "Statement": [
        {
            "Sid": "ExampleStatement01",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Akua"
            },
            "Action": [
                "s3:GetObject",
                "s3:GetBucketLocation",
                "s3:ListBucket"
            ],
```

```
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1/*",
                "arn:aws:s3:::amzn-s3-demo-bucket1"
            ]
        }
    ]
}
```

전체 정책 언어 정보는 IAM 사용 설명서의 IAM 정책 및 권한과 IAM JSON 정책 참조를 참조하세요.

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 크로스 계정 권한을 위임할 수 없습니다.

Amazon S3 버킷 및 객체 소유권

버킷과 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 계정에서 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 섹션을 참조하세요.

소유권 및 인증 요청

버킷에 대한 모든 요청은 인증되었거나 인증되지 않았습니다. 인증된 요청에는 요청 발신자를 인증하는 서명 값이 포함되어야 하며 인증되지 않은 요청에는 포함되지 않아야 합니다. 요청 인증에 대한 자세한 내용은 Amazon S3 API 참조에서 요청을 참조하세요.

버킷 소유자는 인증되지 않은 요청을 허용할 수 있습니다. 예를 들어 인증되지 않은 PutObject 요청은 버킷에 퍼블릭 버킷 정책이 있거나 버킷 ACL이 `All Users` 그룹 또는 익명 사용자에게 `WRITE` 또는 `FULL_CONTROL` 액세스 권한을 부여한 경우 허용됩니다. 퍼블릭 버킷 정책 및 퍼블릭 ACL(액세스 제어 목록)에 대한 자세한 내용은 "퍼블릭"의 의미 섹션을 참조하십시오.

모든 인증되지 않은 요청은 익명 사용자가 수행합니다. 이 사용자는 ACL에서 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 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하십시오. ACL에 대한 자세한 내용은 ACL(액세스 제어 목록) 개요 단원을 참조하십시오.

> **Important**
>
> AWS 계정의 루트 사용자 자격 증명을 사용하여 인증된 요청을 하지 않는 것이 좋습니다. 대신, IAM 역할을 생성하고 해당 역할에 모든 액세스 권한을 부여합니다. 이러한 역할이 부여된 사용자를 관리자 사용자라고 합니다. AWS 계정 루트 사용자 자격 증명 대신, 관리자 역할에 할당된 자격 증명을 사용하여 AWS와 상호 작용하고 버킷 생성, 사용자 생성, 권한 부여 등의 태스크를 수행할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 보안 자격 증명과 IAM 보안 모범 사례를 참조하세요.

Amazon S3의 버킷 정책

버킷 정책은 Amazon S3 버킷과 그 내부의 객체에 대한 액세스 권한을 부여할 수 있는 리소스 기반 정책입니다. 버킷 소유자만 정책을 버킷에 연결할 수 있습니다. 버킷에 연결된 권한은 버킷 소유자가 소유한 모든 버킷의 객체에 적용됩니다. 이러한 권한은 다른 AWS 계정이 소유한 객체에는 적용되지 않습니다.

S3 객체 소유권은 버킷에 업로드되는 객체의 소유권을 제어하고 액세스 제어 목록(ACL)을 비활성화 또는 활성화하는 데 사용할 수 있는 Amazon S3 버킷 수준 설정입니다. 기본적으로 객체 소유권은 버

킷 소유자 적용 설정으로 설정되며 모든 ACL이 비활성화되어 있습니다. 버킷 소유자는 버킷의 모든 객체를 소유하고 정책을 사용하여 객체에 대한 액세스를 독점적으로 관리할 수 있습니다.

버킷 정책에는 JSON 기반 AWS Identity and Access Management(IAM) 정책 언어가 사용됩니다. 버킷 정책을 사용하여 버킷의 객체에 대한 권한을 추가하거나 거부할 수 있습니다. 버킷 정책은 정책의 요소를 기반으로 요청을 허용 또는 거부할 수 있습니다. 이러한 요소에는 요청자, S3 작업, 리소스 및 요청의 측면 또는 (요청을 수행하는 데 사용된 IP 주소 등의) 조건이 포함됩니다.

예를 들어, 다음을 수행하는 버킷 정책을 만들 수 있습니다.

- S3 버킷에 객체를 업로드할 수 있는 크로스 계정 권한을 다른 계정에 부여
- 업로드된 객체를 버킷 소유자인 귀하가 완전히 제어할 수 있음

자세한 내용은 Amazon S3 버킷 정책 예시 섹션을 참조하세요.

> **Important**
>
> 버킷 정책을 사용하여 S3 수명 주기 규칙에 따른 삭제 또는 이전을 방지할 수는 없습니다. 예를 들어, 버킷 정책이 모든 보안 주체에 대한 모든 작업을 거부하더라도 S3 수명 주기 구성은 여전히 정상적으로 작동합니다.

이 섹션의 주제에서는 예제를 제공하고 S3 콘솔에서 버킷 정책을 추가하는 방법을 보여줍니다. ID 기반 정책에 대한 자세한 내용은 Amazon S3의 ID 기반 정책 섹션을 참조하세요. 버킷 정책 언어에 대한 자세한 내용은 Amazon S3의 정책 및 권한 페이지를 참조하십시오.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

주제

- Amazon S3 콘솔을 사용하여 버킷 정책 추가
- 버킷 정책을 사용하여 VPC 엔드포인트에서 액세스 제어
- Amazon S3 버킷 정책 예시
- 조건 키를 사용한 버킷 정책 예시

Amazon S3 콘솔을 사용하여 버킷 정책 추가

AWS 정책 생성기 및 Amazon S3 콘솔을 사용하여 새 버킷 정책을 추가하거나 기존 버킷 정책을 편집할 수 있습니다. 버킷 정책은 리소스 기반 AWS Identity and Access Management(IAM) 정책입니다. 다른 AWS 계정 또는 IAM 사용자에게 버킷과 버킷에 포함된 객체에 대한 액세스 권한을 부여하는 버킷 정책을 버킷에 추가할 수 있습니다. 객체 권한은 해당 버킷 소유자가 생성하는 객체에만 적용됩니다. 버킷 정책에 대한 자세한 내용은 Amazon S3의 ID 및 액세스 관리를 참조하십시오.

정책을 저장하기 전에 AWS Identity and Access Management Access Analyzer의 보안 경고, 오류, 일반 경고 및 제안 사항을 해결해야 합니다. IAM Access Analyzer는 정책 확인은 실행하여 IAM 정책 문법 및 모범 사례에 대해 정책을 검증합니다. 이러한 확인은 결과를 생성하고 보안 모범 사례를 준수하고 작동하는 정책을 작성하는 데 도움이 되는 실행 가능한 권장 사항을 제공합니다. IAM Access Analyzer를 사용한 정책 검증에 대한 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer 정책 검증을 참조하십시오. IAM Access Analyzer에서 반환된 경고, 오류 및 제안 사항 목록을 보려면 IAM Access Analyzer 정책 확인 참조를 참조하십시오.

정책으로 오류를 해결하는 방법에 대한 지침은 Amazon S3의 액세스 거부(403 Forbidden) 오류 문제 해결을 참조하십시오.

버킷 정책 생성 또는 편집

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷 또는 디렉터리 버킷을 선택합니다.
3. 버킷 목록에서 버킷 정책을 만들 버킷 이름 또는 버킷 정책을 편집할 버킷 이름을 선택합니다.
4. 권한 탭을 선택합니다.
5. 버킷 정책에서 편집을 선택합니다. 버킷 정책 편집 페이지가 나타납니다.
6. 버킷 정책 편집 페이지에서 다음 중 하나를 수행합니다.

   - 버킷 정책의 예를 보려면 정책 예제를 선택합니다. 또는 Amazon S3 사용 설명서의 Amazon S3 버킷 정책 예시 섹션을 참조하세요.
   - 정책을 자동으로 생성하거나 정책 섹션에서 JSON을 편집하려면 정책 생성기를 선택합니다.

   정책 생성기를 선택하면 AWS 정책 생성기가 새 창에서 열립니다.

   a. AWS 정책 생성기 페이지의 Select Type of Policy(정책 유형 선택)에 S3 Bucket Policy(S3 버킷 정책)를 선택합니다.

b. 제공된 필드에 정보를 입력하여 명령문을 추가한 다음 명령문 추가(Add Statement)를 선택합니다. 문을 추가하려는 만큼 이 단계를 반복합니다. 이러한 필드에 대한 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하십시오.

> **Note**
>
> 편의를 위해 버킷 정책 편집 페이지의 정책 텍스트 필드 위에 현재 버킷의 버킷 ARN(Amazon 리소스 이름)이 표시됩니다. AWS 정책 생성기 페이지의 명령문에 사용하기 위해 이 ARN을 복사할 수 있습니다.

c. 명령문 추가를 마친 후 정책 생성(Generate Policy)을 선택합니다.

d. 생성된 정책 텍스트를 복사하고 닫기(Close)를 선택하고 Amazon S3 콘솔의 버킷 정책 편집(Edit bucket policy) 페이지로 돌아갑니다.

7. 정책 상자에서 기존 정책을 편집하거나 AWS 정책 생성기에서 버킷 정책을 붙여 넣습니다. 정책을 저장하기 전에 보안 경고, 오류, 일반 경고 및 제안 사항을 해결해야 합니다.

> **Note**
>
> 버킷 정책은 크기가 20KB로 제한됩니다.

8. (선택 사항) 오른쪽 하단 외부 액세스 미리 보기를 선택하면 새 정책이 리소스의 퍼블릭 및 크로스 계정 액세스에 미치는 영향을 미리 확인할 수 있습니다. 정책을 저장하기 전에 새로운 IAM Access Analyzer 검색 결과가 나오는지, 기존 결과가 해결되는지 여부를 확인할 수 있습니다. 활성 분석기가 표시되지 않으면 Access Analyzer로 이동을 선택하여 IAM Access Analyzer에서 계정 분석기를 생성합니다. 자세한 내용은 IAM 사용 설명서에서 평가판 액세스를 참조하십시오.

9. 변경 사항 저장을 선택하면 권한 탭으로 돌아갑니다.

버킷 정책을 사용하여 VPC 엔드포인트에서 액세스 제어

Amazon S3 버킷 정책을 사용하여 특정 가상 프라이빗 클라우드(VPC) 엔드포인트 또는 특정 VPC의 버킷에 대한 액세스를 제어할 수 있습니다. 이 섹션에는 VPC 엔드포인트에서 Amazon S3 버킷 액세스를 제어하는 데 사용할 수 있는 예시 버킷 정책이 포함되어 있습니다. VPC 엔드포인트의 설정 방법을 알아보려면 VPC 사용 설명서의 VPC 엔드포인트를 참조하십시오.

VPC에서는 AWS 리소스를 사용자가 정의한 가상 네트워크로 시작할 수 있습니다. VPC 엔드포인트를 사용하면 VPC와 다른 AWS 서비스 사이에 프라이빗 연결을 생성할 수 있습니다. 이 프라이빗 연결에

는 인터넷, 가상 프라이빗 네트워크(VPN) 연결, NAT 인스턴스 또는 AWS Direct Connect를 통해 액세스하지 않아도 됩니다.

Amazon S3용 VPC 엔드포인트는 Amazon S3로의 연결만 허용하는 VPC 내 논리적 엔터티입니다. VPC 엔드포인트는 Amazon S3로 요청을 라우팅하고 응답을 다시 VPC로 라우팅합니다. VPC 종단점은 요청이 라우팅되는 방식만을 변경합니다. Amazon S3 퍼블릭 엔드포인트와 DNS 이름은 계속해서 VPC 엔드포인트로 작업합니다. Amazon S3와 함께 VPC 엔드포인트를 사용하는 것에 대한 중요 정보는 VPC 사용 설명서의 게이트웨이 엔드포인트 및 Amazon S3의 게이트웨이 엔드포인트를 참조하세요.

Amazon S3용 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 VPC 사용 설명서의 엔드포인트 정책을 사용하여 VPC 엔드포인트에 대한 액세스 제어를 참조하세요.
- Amazon S3 버킷 정책을 사용하여 버킷에 액세스할 수 있는 VPC 또는 VPC 엔드포인트를 제어할 수 있습니다. 이러한 유형의 버킷 정책 제어에 대한 예는 액세스 제한에 대한 다음 주제를 참조하십시오.

주제

- 특정 VPC 엔드포인트에 대한 액세스 제한
- 특정 VPC에 대한 액세스 제한

> **Important**
>
> 이 섹션에서 설명하는 VPC 엔드포인트에 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 AWS Support 지식 센터의 VPC 또는 VPC 엔드포인트 ID가 올바르지 않은 버킷 정책을 수정하는 방법을 참조하세요.

특정 VPC 엔드포인트에 대한 액세스 제한

다음은 ID가 `vpce-1a2b3c4d`인 VPC 엔드포인트에서만 특정 버킷 `awsexamplebucket1`에 대한 액세스를 제한하는 Amazon S3 버킷 정책의 예제입니다. 지정된 엔드포인트가 사용되지 않으면 이 정

책은 버킷에 대한 모든 액세스를 거부합니다. aws:SourceVpce 조건은 엔드포인트를 지정합니다. aws:SourceVpce 조건은 VPC 엔드포인트 리소스의 Amazon 리소스 이름(ARN)을 요구하지 않고 VPC 엔드포인트 ID만 요구합니다. 정책에서 조건을 사용하는 것에 대한 자세한 내용은 조건 키를 사용한 버킷 정책 예시을 참조하십시오.

> **Important**
>
> - 다음 예제 정책을 사용하기 전에 VPC 엔드포인트 ID를 사용 사례에 적합한 값으로 바꾸십시오. 그렇지 않으면 버킷에 액세스할 수 없습니다.
> - 콘솔 요청은 지정된 VPC 엔드포인트에서 발생하지 않으므로 이 정책은 지정된 버킷에 대한 콘솔 액세스를 비활성화합니다.

```
{
   "Version": "2012-10-17",
   "Id": "Policy1415115909152",
   "Statement": [
     {
       "Sid": "Access-to-specific-VPCE-only",
       "Principal": "*",
       "Action": "s3:*",
       "Effect": "Deny",
       "Resource": ["arn:aws:s3:::awsexamplebucket1",
                    "arn:aws:s3:::awsexamplebucket1/*"],
       "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 엔드포인트가 구성되어 있으며, 모든 엔드포인트에서 Amazon S3 버킷에 대한 액세스를 관리하고자 하는 경우 유용합니다. 다음은 VPC awsexamplebucket1 외부에 있는 자가 vpc-111bbb22 및 그 객체에 액세스할 수 없도록 거부하는 정책의 예제입니다. 지정된 VPC가 사용되지 않으면 이 정책은 버킷에 대한 모든 액세스를 거부합니다. 이 문은 버킷에 대한

액세스 권한을 부여하지 않습니다. 액세스 권한을 부여하려면 별도의 `Allow` 문을 추가해야 합니다. `vpc-111bbb22` 조건 키는 VPC 리소스의 ARN을 요구하지 않고 VPC ID만 요구합니다.

> **Important**
>
> - 다음 예제 정책을 사용하기 전에 VPC ID를 사용 사례에 적합한 값으로 바꾸십시오. 그렇지 않으면 버킷에 액세스할 수 없습니다.
> - 콘솔 요청은 지정된 VPC에서 발생하지 않으므로 이 정책은 지정된 버킷에 대한 콘솔 액세스를 비활성화합니다.

```
{
   "Version": "2012-10-17",
   "Id": "Policy1415115909153",
   "Statement": [
     {
       "Sid": "Access-to-specific-VPC-only",
       "Principal": "*",
       "Action": "s3:*",
       "Effect": "Deny",
       "Resource": ["arn:aws:s3:::awsexamplebucket1",
                    "arn:aws:s3:::awsexamplebucket1/*"],
       "Condition": {
         "StringNotEquals": {
           "aws:SourceVpc": "vpc-111bbb22"
         }
       }
     }
   ]
}
```

특정 IP 또는 VPC를 기반으로 버킷 액세스를 제한하는 방법에 대한 자세한 내용은 AWS re:Post 지식 센터의 특정 VPC 엔드포인트 또는 IP 주소만 Amazon S3 버킷에 액세스하도록 하려면 어떻게 해야 합니까?를 참조하세요.

Amazon S3 버킷 정책 예시

Amazon S3 버킷 정책을 사용하면 적절한 권한을 가진 사용자만 객체에 액세스할 수 있도록 하여 버킷의 객체 액세스를 보호할 수 있습니다. 인증받은 사용자라도 적절한 권한이 없다면 Amazon S3 리소스에 액세스하지 못하게 할 수도 있습니다.

이 섹션에서는 버킷 정책의 일반적인 사용 사례에 대한 예제를 제시합니다. 이러한 샘플 정책은 리소스 값으로 *amzn-s3-demo-bucket*를 사용합니다. 이러한 정책을 테스트하려면 *user input placeholders*를 (귀하의 버킷 이름 등) 자체 정보로 대체합니다.

객체 집합으로의 권한을 부여 또는 거부하려면 Amazon 리소스 이름(ARN) 및 기타 값에 와일드카드 문자(*)를 사용하면 됩니다. 예를 들어, 공통 접두사로 시작하거나 .html과 같은 특정 확장자로 끝나는 객체 그룹에 대한 액세스를 제어할 수 있습니다.

AWS Identity and Access Management(IAM) 정책 언어에 대한 자세한 내용은 Amazon S3의 정책 및 권한를 참조하십시오.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

> **Note**
>
> Amazon S3 콘솔을 사용하여 권한을 테스트할 경우 콘솔이 요구하는 추가 권한 즉, `s3:ListAllMyBuckets`, `s3:GetBucketLocation` 및 `s3:ListBucket`을 부여해야 합니다. 콘솔을 사용하여 사용자에게 권한을 부여하고 해당 권한을 테스트하는 방법에 대한 예제는 사용자 정책을 사용하여 버킷에 대한 액세스 제어을 참조하십시오.

버킷 정책 생성을 위한 추가 리소스에는 다음이 포함됩니다.

- 버킷 정책을 생성할 때 사용할 수 있는 IAM 정책 작업, 리소스 및 조건 키의 전체 목록은 서비스 승인 참조의 Amazon S3에 사용되는 작업, 리소스 및 조건 키를 참조하세요.
- S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.
- S3 정책 생성에 대한 지침은 Amazon S3 콘솔을 사용하여 버킷 정책 추가를 참조하십시오.
- 정책 관련 오류 문제를 해결하려면 Amazon S3의 액세스 거부(403 Forbidden) 오류 문제 해결을 참조하십시오.

정책을 추가하거나 업데이트하는 데 문제가 있는 경우 AWS re:Post 지식 센터의 Amazon S3 버킷 정책을 업데이트하려고 할 때 "Invalid principal in policy"라는 오류가 표시되는 이유는 무엇입니까?를 참조하세요.

주제

- 일반 익명 사용자에게 읽기 전용 권한 부여

- 암호화 필요
- 준비된 ACL을 사용한 버킷 관리
- 객체 태깅을 통한 객체 액세스 관리
- 전역 조건 키를 사용한 객체 액세스 관리
- HTTP 또는 HTTPS 요청 기반 액세스 관리
- 특정 폴더에 대한 사용자 액세스 관리
- 액세스 로그에 액세스 관리
- Amazon CloudFront OAI에 대한 액세스 관리
- Amazon S3 스토리지 렌즈 액세스 관리
- S3 인벤토리, S3 분석 및 S3 인벤토리 보고서 권한 관리
- MFA 필요
- 사용자가 객체를 삭제하지 못하도록 방지

일반 익명 사용자에게 읽기 전용 권한 부여

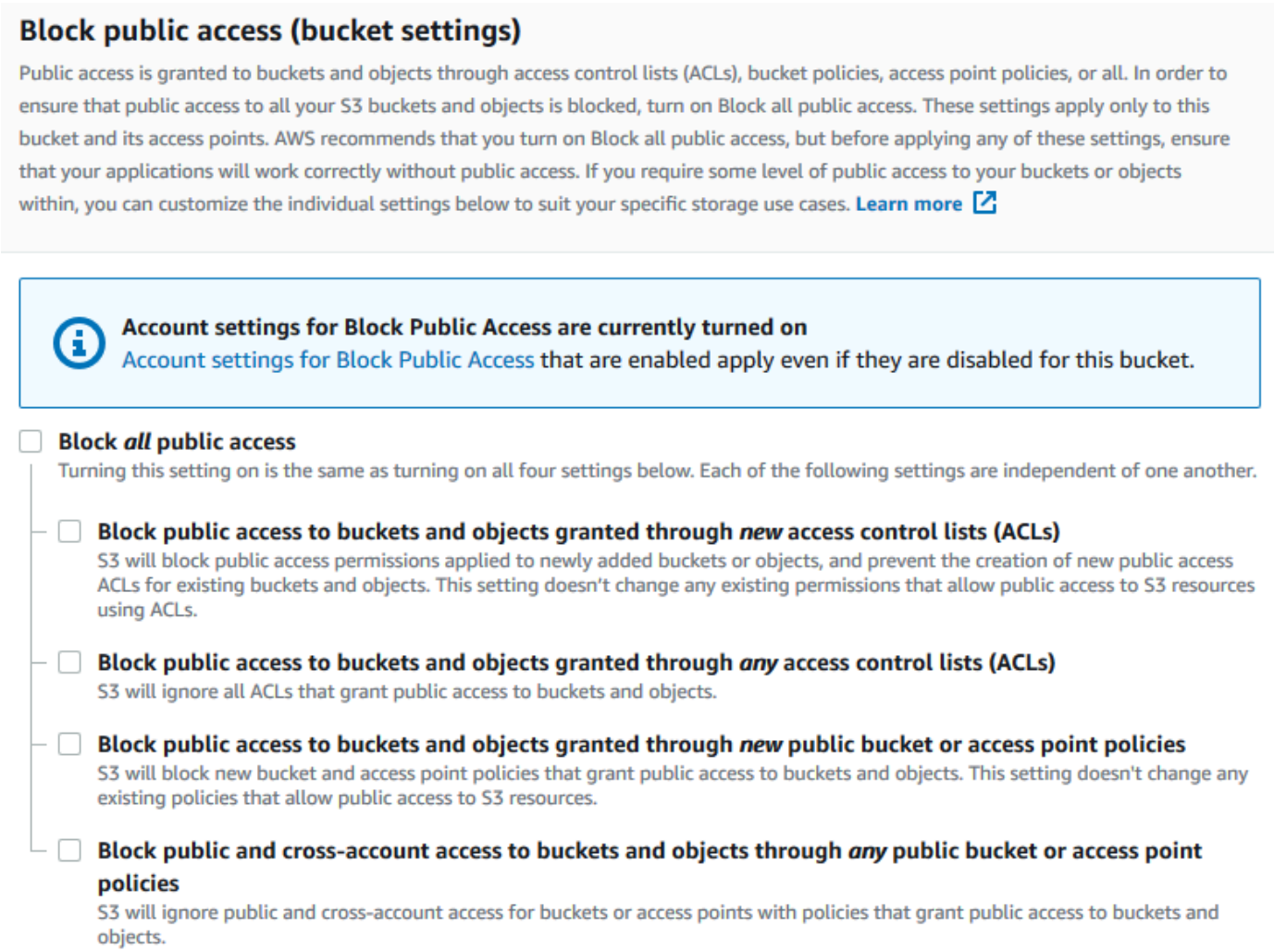
정책 설정을 사용하여 일반 익명 사용자에게 액세스 권한을 부여할 수 있으며, 이는 버킷을 정적 웹 사이트로 구성하는 경우에 유용합니다. 익명 퍼블릭 사용자에게 액세스 권한을 부여하려면 버킷에 대한 퍼블릭 액세스 차단 설정을 비활성화해야 합니다. 이 작업을 수행하는 방법과 필요한 정책에 대한 자세한 내용은 웹 사이트 액세스에 대한 권한 설정 섹션을 참조하세요. 동일한 목적으로 더 제한적인 정책을 설정하는 방법을 알아보려면 AWS 지식 센터의 Amazon S3 버킷의 일부 객체에 대해 공개 읽기 액세스 권한을 부여하려면 어떻게 해야 하나요?를 참조하세요.

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**
>
> 이러한 단계를 완료하기 전에 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단을 검토하여 퍼블릭 액세스 허용과 관련된 위험을 이해하고 이에 동의하는지 확인합니다. 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. 퍼블릭 액세스 차단(버킷 설정)(Block public access (bucket settings))에서 편집(Edit)을 선택합니다.
5. 모든 퍼블릭 액세스 차단(Block all public access)을 선택 취소하고 변경 사항 저장(Save changes)을 선택합니다.



Amazon S3은 버킷에 대한 퍼블릭 액세스 차단 설정을 끕니다. 정적 퍼블릭 웹 사이트를 생성하려면 버킷 정책을 추가하기 전에 계정에 대한 퍼블릭 액세스 차단 설정을 편집해야 할 수도 있습니다. 계정에 대한 퍼블릭 액세스 차단 설정이 현재 켜져 있는 경우 퍼블릭 액세스 차단(버킷 설정) 아래에 메모가 표시됩니다.

암호화 필요

다음 예와 같이 AWS Key Management Service(AWS KMS) 키를 사용한 서버 측 암호화(SSE-KMS)를 요구할 수 있습니다.

버킷에 쓰는 모든 객체에 SSE-KMS 필요

다음 예제 정책에서는 버킷에 쓰는 모든 객체가 AWS Key Management Service(AWS KMS) 키를 사용한 서버 측 암호화(SSE-KMS)로 암호화되어야 합니다. 객체가 SSE-KMS로 암호화되지 않은 경우, 요청이 거부됩니다.

```
{
"Version": "2012-10-17",
"Id": "PutObjPolicy",
"Statement": [{
  "Sid": "DenyObjectsThatAreNotSSEKMS",
  "Principal": "*",
  "Effect": "Deny",
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
  "Condition": {
    "Null": {
      "s3:x-amz-server-side-encryption-aws-kms-key-id": "true"
    }
  }
}]
}
```

버킷에 쓰는 모든 객체에 특정 AWS KMS key를 사용하는 SSE-KMS 필요

다음 예제 정책에서는 객체가 특정 KMS 키 ID를 사용한 SSE-KMS로 암호화되지 않았다면 버킷에 써지지 않도록 거부합니다. 요청별 헤더 또는 버킷 기본 암호화를 사용하여 객체가 SSE-KMS로 암호화되었더라도 지정된 KMS 키로 암호화되지 않았다면 객체를 버킷에 쓸 수 없습니다. 이 예제에 사용된 KMS 키 ARN은 자체 KMS 키 ARN으로 대체하십시오.

```
{
"Version": "2012-10-17",
"Id": "PutObjPolicy",
"Statement": [{
  "Sid": "DenyObjectsThatAreNotSSEKMSWithSpecificKey",
  "Principal": "*",
```

```
  "Effect": "Deny",
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
  "Condition": {
    "ArnNotEqualsIfExists": {
      "s3:x-amz-server-side-encryption-aws-kms-key-id": "arn:aws:kms:us-
east-2:111122223333:key/01234567-89ab-cdef-0123-456789abcdef"
    }
  }
}]
}
```

준비된 ACL을 사용한 버킷 관리

여러 계정에 객체를 업로드하거나 객체 ACL을 공개 액세스 가능으로 설정할 수 있는 권한 부여

다음 예시 정책에서는 여러 AWS 계정에 `s3:PutObject` 및 `s3:PutObjectAcl` 권한을 부여합니다. 또한 정책 예시는 이러한 작업에 대한 요청에 `public-read` 표준 액세스 제어 목록(ACL)을 반드시 포함하도록 요구합니다. 자세한 내용은 Amazon S3의 정책 작업 및 Amazon S3의 정책 조건 키(을)를 참조하세요.

> **Warning**
>
> `public-read` 준비된 ACL을 사용하면 전 세계 누구나 버킷 내의 객체를 볼 수 있습니다. Amazon S3 버킷에 대한 익명 액세스 권한을 부여하거나 퍼블릭 액세스 차단 설정을 사용 중지할 때 주의하십시오. 익명 액세스 권한을 부여하면 전 세계 누구나 버킷에 액세스할 수 있습니다. 정적 웹 사이트 호스팅용과 같이 특별히 필요한 경우가 아니면 Amazon S3 버킷에 익명 액세스 권한을 부여하지 않는 것이 좋습니다. 정적 웹 사이트 호스팅에 대한 퍼블릭 액세스 차단 설정을 활성화하려면 자습서: Amazon S3에서 정적 웹 사이트 구성을 참조하십시오.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AddPublicReadCannedAcl",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root",
                    "arn:aws:iam::444455556666:root"
```

```
                ]
            },
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": [
                        "public-read"
                    ]
                }
            }
        }
    ]
}
```

버킷 소유자가 완벽하게 제어할 수 있도록 보증하면서 객체에 업로드하는 크로스 계정 권한 부여

다음 예에서는 업로드된 객체를 완전히 제어하면서 다른 AWS 계정이 버킷에 객체를 업로드하도록 허용하는 방법을 보여줍니다. 이 정책은 특정 AWS 계정(*111122223333*)이 업로드 시 `bucket-owner-full-control`이라는 표준 ACL을 포함하고 있는 경우에만 객체를 업로드할 수 있는 권한을 부여합니다. 정책의 `StringEquals` 조건은 준비된 ACL 요구 사항을 표현하도록 `s3:x-amz-acl` 조건 키를 지정합니다. 자세한 내용은 Amazon S3의 정책 조건 키 섹션을 참조하세요.

```
{
   "Version":"2012-10-17",
   "Statement":[
     {
       "Sid":"PolicyForAllowUploadWithACL",
       "Effect":"Allow",
       "Principal":{"AWS":"111122223333"},
       "Action":"s3:PutObject",
       "Resource":"arn:aws:s3:::amzn-s3-demo-bucket/*",
       "Condition": {
         "StringEquals": {"s3:x-amz-acl":"bucket-owner-full-control"}
       }
     }
   ]
}
```

객체 태깅을 통한 객체 액세스 관리

사용자가 특정 태그 키 및 값이 있는 객체만 읽도록 허용

다음 권한 정책은 environment: production 태그 키 및 값이 있는 객체만 사용자가 읽을 수 있도록 제한합니다. 이 정책은 s3:ExistingObjectTag 조건 키를 사용하여 태그 키 및 값을 지정합니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Principal":{
            "AWS":"arn:aws:iam::111122223333:role/JohnDoe"
         },
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket/*",
         "Condition":{
            "StringEquals":{
               "s3:ExistingObjectTag/environment":"production"
            }
         }
      }
   ]
}
```

사용자가 추가할 수 있는 객체 태그 키 제한

다음 권한 정책은 s3:PutObjectTagging 작업을 수행할 수 있는 권한을 사용자에게 부여하여, 사용자가 기존 객체에 태그를 추가할 수 있게 합니다. 조건이 s3:RequestObjectTagKeys 조건 키를 사용하여 Owner, CreationDate 등의 허용 태그 키 세트를 지정합니다. 자세한 내용은 IAM 사용 설명서의 다수의 키 또는 값을 사용하는 조건 생성을 참조하십시오.

요청에 지정된 모든 태그 키가 인증된 태그 키임을 정책이 보장합니다. 지정된 값 중 최소 1개가 요청에 존재함을 조건 내의 ForAnyValue 한정자가 보장합니다.

```
{
   "Version": "2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
```

```
            "arn:aws:iam::111122223333:role/JohnDoe"
          ]
        },
 "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ],
      "Condition": {"ForAnyValue:StringEquals": {"s3:RequestObjectTagKeys": [
            "Owner",
            "CreationDate"
          ]
        }
      }
    }
  ]
}
```

사용자가 객체 태그를 추가할 수 있게 하려면 특정 태그 키 및 값 필요

다음 권한 정책은 s3:PutObjectTagging 작업을 수행할 수 있는 권한을 사용자에게 부여하여, 사용자가 기존 객체에 태그를 추가할 수 있게 합니다. 이 조건은 값이 *X*로 설정된 (*Project* 등의) 특정 태그 키를 포함하도록 사용자에게 요구합니다.

```
{
   "Version": "2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
       "arn:aws:iam::111122223333:user/JohnDoe"
        ]
      },
      "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ],
      "Condition": {"StringEquals": {"s3:RequestObjectTag/Project": "X"
        }
      }
```

```
    }
  ]
}
```

사용자가 특정 객체 태그 키 및 값이 있는 객체만 추가하도록 허용

다음 예제 정책은 `s3:PutObject` 작업을 수행할 권한을 사용자에게 부여하여, 객체를 버킷에 추가할 수 있게 합니다. 하지만 `Condition` 문은 업로드된 객체에 허용된 태그 키와 값만 사용할 수 있도록 제한합니다. 이 예제에서는 값이 *`Finance`*로 설정된 특정 태그 키(*`Department`*)가 있는 객체만 사용자가 버킷에 추가할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Principal":{
            "AWS":[
                 "arn:aws:iam::111122223333:user/JohnDoe"
         ]
        },
        "Effect": "Allow",
        "Action": [
            "s3:PutObject"
        ],
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket/*"
        ],
        "Condition": {
            "StringEquals": {
                "s3:RequestObjectTag/Department": "Finance"
            }
        }
    }]
}
```

전역 조건 키를 사용한 객체 액세스 관리

전역 조건 키는 `aws` 접두사가 있는 조건 키입니다. AWS 서비스는 전역 조건 키나, 서비스 접두사가 포함된 서비스별 키를 지원할 수 있습니다. JSON 정책의 `Condition` 요소를 사용하면 요청 컨텍스트의 키를 정책에서 지정한 키 값과 비교할 수 있습니다.

Amazon S3 서버 액세스 로그 전달로 액세스 제한

다음 예제 버킷 정책에서는 aws:SourceArn 전역 조건 키를 사용하여, 서비스 대 서비스 요청을 하는 리소스의 Amazon 리소스 이름(ARN)을 정책 내에 지정된 ARN과 비교합니다. `aws:SourceArn` 전역 조건 키는 서비스 간 트랜잭션 중에 Amazon S3 서비스가 혼동된 대리자로 사용되는 것을 방지하는 용도로 사용됩니다. Amazon S3 서비스만 Amazon S3 버킷에 객체를 추가할 수 있습니다.

이 예제 버킷 정책은 로깅 서비스 보안 주체(`logging.s3.amazonaws.com`)에만 `s3:PutObject` 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutObjectS3ServerAccessLogsPolicy",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket-logs/*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111111111111"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::EXAMPLE-SOURCE-BUCKET"
                }
            }
        },
        {
            "Sid": "RestrictToS3ServerAccessLogs",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket-logs/*",
            "Condition": {
                "ForAllValues:StringNotEquals": {
                    "aws:PrincipalServiceNamesList": "logging.s3.amazonaws.com"
                }
            }
        }
    ]
```

```
}
```

내가 속한 조직에만 액세스 허용

리소스에 액세스하는 모든 IAM 보안 주체가 귀하가 속한 조직 내 AWS 계정(AWS Organizations 관리 계정 포함)에 속하도록 요구하려면 `aws:PrincipalOrgID` 전역 조건 키를 사용하면 됩니다.

이 유형의 액세스를 부여하거나 제한하려면 `aws:PrincipalOrgID` 조건을 정의하고 버킷 정책에서 해당 값을 귀하의 조직 ID로 설정합니다. 조직 ID는 버킷에 대한 액세스를 제어하는 데 사용됩니다. `aws:PrincipalOrgID` 조건을 사용하면, 조직에 새로 추가된 모든 계정에도 버킷 정책에서 설정된 권한이 적용됩니다.

다음은 귀하의 조직 내 특정 IAM 보안 주체에게 버킷의 직접 액세스를 부여하는 데 사용할 수 있는 리소스 기반 버킷 정책의 예제입니다. 버킷 정책에 `aws:PrincipalOrgID` 글로벌 조건 키를 추가하면, 보안 주체 계정이 귀하의 조직에 속해야만 리소스에 액세스할 수 있습니다. 액세스 권한을 부여할 때 실수로 잘못된 계정을 지정하더라도 aws:PrincipalOrgID 글로벌 조건 키가 추가 보호 장치 역할을 합니다. 이 글로벌 키가 정책 내에 사용되면 지정된 조직 외부의 보안 주체는 누구든 Amazon S3 버킷에 액세스할 수 없습니다. 목록에 있는 조직에 속한 계정의 보안 주체만 리소스에 액세스할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "AllowGetObject",
        "Principal": {
            "AWS": "*"
        },
        "Effect": "Allow",
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "Condition": {
            "StringEquals": {
                "aws:PrincipalOrgID": ["o-aa111bb222"]
            }
        }
    }]
}
```

HTTP 또는 HTTPS 요청 기반 액세스 관리

HTTPS 요청으로만 액세스 제한

잠재적인 공격자가 네트워크 트래픽을 조작하지 못하도록 하려면, HTTPS(TLS)를 사용하여 암호화된 연결만 허용하고 HTTP 요청은 버킷에 액세스하지 못하도록 제한하면 됩니다. 요청이 HTTP인지 HTTPS인지 확인하려면 S3 버킷 정책에서 aws:SecureTransport 글로벌 조건 키를 사용합니다. `aws:SecureTransport` 조건 키는 요청이 HTTP를 사용하여 전송되었는지 여부를 확인합니다.

요청이 `true`로 반환되면 HTTPS를 통해 전송된 요청입니다. 요청이 `false`로 반환되면 HTTP를 통해 전송된 요청입니다. 그러면 원하는 요청 체계에 기반하여 버킷에 액세스를 허용하거나 거부하면 됩니다.

다음 예제에서는 버킷 정책이 HTTP 요청을 명시적으로 거부합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "RestrictToTLSRequestsOnly",
        "Action": "s3:*",
        "Effect": "Deny",
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket",
            "arn:aws:s3:::amzn-s3-demo-bucket/*"
        ],
        "Condition": {
            "Bool": {
                "aws:SecureTransport": "false"
            }
        },
        "Principal": "*"
    }]
}
```

특정 HTTP 리퍼러로 액세스 제한

도메인 이름이 *www.example.com* 또는 *example.com*인 웹 사이트에, 이름이 *amzn-s3-demo-bucket*인 버킷에 저장된 사진과 동영상 링크가 포함되어 있다고 가정합니다. 기본적으로 모든 Amazon S3 리소스는 비공개이므로 리소스를 만든 AWS 계정만 이 리소스에 액세스할 수 있습니다.

웹 사이트에서 이들 객체에 대한 읽기 권한을 허용하려면, GET 요청이 특정 웹 페이지에서 출발한 경우에 한해 s3:GetObject 권한을 허용하는 버킷 정책을 추가하면 됩니다. 다음 정책은 StringLike 조건과 aws:Referer 조건 키를 사용하여 요청을 제한합니다.

```
{
  "Version":"2012-10-17",
  "Id":"HTTP referer policy example",
  "Statement":[
    {
      "Sid":"Allow only GET requests originating from www.example.com and
 example.com.",
      "Effect":"Allow",
      "Principal":"*",
      "Action":["s3:GetObject","s3:GetObjectVersion"],
      "Resource":"arn:aws:s3:::amzn-s3-demo-bucket/*",
      "Condition":{
        "StringLike":{"aws:Referer":["http://www.example.com/*","http://example.com/
*"]}
      }
    }
  ]
}
```

사용하는 브라우저에는 요청의 HTTP referer 헤더가 포함되어야 합니다.

> **Warning**
>
> aws:Referer 조건 키를 사용할 때 주의하는 것이 좋습니다. 공개적으로 알려진 HTTP 참조자 헤더 값을 포함하는 것은 위험합니다. 권한이 없는 사용자가 수정된 브라우저나 사용자 지정 브라우저를 사용하여 원하는 aws:Referer 값을 제공할 수 있습니다. 따라서 승인되지 않은 당사자가 직접 AWS 요청을 하지 못하도록 aws:Referer를 사용하지 마십시오.
> aws:Referer 조건 키는 고객이 Amazon S3에 저장된 콘텐츠 등의 디지털 콘텐츠를 권한이 없는 서드 파티 사이트에서 참조하지 못하도록 보호하기 위해서만 제공됩니다. 자세한 내용은 IAM 사용 설명서에서 aws:Referer 섹션을 참조하십시오.

특정 폴더에 대한 사용자 액세스 관리

사용자에게 특정 폴더 액세스 부여

사용자에게 특정 폴더에 대한 액세스 권한을 부여하려고 한다고 가정합니다. IAM 사용자와 S3 버킷이 동일한 AWS 계정에 속한 경우, IAM 정책을 사용하여 사용자에게 특정 버킷 폴더에 대한 액세스 권한을 부여할 수 있습니다. 이 접근법을 사용하면 액세스 권한을 부여하기 위해 버킷 정책을 업데이트하지 않아도 됩니다. 여러 사용자가 전환할 수 있는 IAM 역할에 IAM 정책을 추가할 수 있습니다.

IAM ID와 S3 버킷이 다른 AWS 계정에 속해 있다면 IAM 정책과 버킷 정책 모두에 크로스 계정 액세스를 부여해야 합니다. 크로스 계정 액세스 부여에 대한 자세한 내용은 버킷 소유자가 크로스 계정 버킷 권한 부여를 참조하십시오.

다음 예제 버킷 정책은 *JohnDoe* 콘솔 전체 액세스 권한을 자신의 폴더(home/*JohnDoe*/)에만 부여합니다. home 폴더를 만들고 사용자에게 적절한 권한을 부여하면, 여러 명의 사용자가 하나의 버킷을 공유하도록 할 수 있습니다. 이 정책은 다음 세 가지 `Allow` 문으로 구성됩니다.

- *AllowRootAndHomeListingOfCompanyBucket*: 사용자(*JohnDoe*)가 *amzn-s3-demo-bucket* 버킷의 루트 수준 및 home 폴더에 있는 객체를 나열할 수 있도록 허용합니다. 이 문은 사용자가 콘솔을 사용하여 접두사 home/를 검색할 수 있게 하기도 합니다.
- *AllowListingOfUserFolder*: 사용자(*JohnDoe*)가 home/*JohnDoe*/ 폴더와 그 모든 하위 폴더에 있는 모든 객체를 나열할 수 있도록 허용합니다.
- *AllowAllS3ActionsInUserFolder*: `Read`, `Write`, `Delete` 권한을 부여하여 사용자가 모든 Amazon S3 작업을 수행할 수 있도록 허용합니다. 권한은 버킷 소유자의 홈 폴더로 제한됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRootAndHomeListingOfCompanyBucket",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/JohnDoe"
                ]
            },
            "Effect": "Allow",
            "Action": ["s3:ListBucket"],
            "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket"],
            "Condition": {
```

```
                "StringEquals": {
                    "s3:prefix": ["", "home/", "home/JohnDoe"],
                    "s3:delimiter": ["/"]
                }
            }
        },
        {
            "Sid": "AllowListingOfUserFolder",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/JohnDoe"
                ]
            },
            "Action": ["s3:ListBucket"],
            "Effect": "Allow",
            "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket"],
            "Condition": {
                "StringLike": {
                    "s3:prefix": ["home/JohnDoe/*"]
                }
            }
        },
        {
            "Sid": "AllowAllS3ActionsInUserFolder",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/JohnDoe"
                ]
            },
            "Action": ["s3:*"],
            "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket/home/JohnDoe/*"]
        }
    ]
}
```

액세스 로그에 액세스 관리

액세스 로그를 활성화할 수 있도록 Application Load Balancer 액세스 권한 부여

Application Load Balancer에 액세스 로그를 활성화할 때는 로드 밸런서가 로그를 저장할 S3 버킷의 이름을 지정해야 합니다. 버킷에 액세스 로그를 쓸 수 있으려면 Elastic Load Balancing 권한을 부여하는 연결된 정책이 버킷에 있어야 합니다.

다음 예제에서는 버킷 정책이 Elastic Load Balancing(ELB) 권한을 부여하여 액세스 로그를 버킷에 쓸 수 있도록 합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": {
        "AWS": "arn:aws:iam::elb-account-id:root"
      },
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/prefix/AWSLogs/111122223333/*"
    }
  ]
}
```

> **Note**
>
> *elb-account-id*는 귀하의 AWS 리전에 해당하는 Elastic Load Balancing용 AWS 계정 ID로 대체합니다. Elastic Load Balancing 리전 목록은 Elastic Load Balancing 사용 설명서의 Attach a policy to your Amazon S3 bucket(정책을 Amazon S3 버킷에 연결)을 참조하십시오.

귀하의 AWS 리전이 지원되는 Elastic Load Balancing 리전 목록에 보이지 않으면, 지정된 로그 전달 서비스에 권한을 부여하는 다음 정책을 사용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
       "Principal": {
         "Service": "logdelivery.elasticloadbalancing.amazonaws.com"
          },
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/prefix/AWSLogs/111122223333/*"
    }
  ]
}
```

그런 다음, Elastic Load Balancing 액세스 로그를 활성화하여 구성합니다. 테스트 파일을 생성하면 버킷 권한을 확인할 수 있습니다.

Amazon CloudFront OAI에 대한 액세스 관리

Amazon CloudFront OAI에 대한 권한 부여

다음 예제 버킷 정책은 CloudFront 오리진 액세스 ID(OAI) 권한을 부여하여 S3 버킷에서 모든 객체를 가져옵니다(읽습니다). CloudFront OAI를 사용하면 사용자가 Amazon S3이 아니라 CloudFront를 통해 버킷의 객체에 액세스하도록 허용할 수 있습니다. 자세한 내용은 Amazon CloudFront 개발자 안내서의 Amazon S3 오리진에 대한 액세스 제한을 참조하십시오.

다음 정책은 OAI의 ID를 정책의 `Principal`로 사용합니다. S3 버킷 정책을 사용하여 CloudFront OAI에 대한 액세스 권한을 부여하는 방법에 대한 자세한 내용은 Amazon CloudFront 개발자 안내서의 오리진 액세스 ID(OAI)에서 오리진 액세스 제어(OAC)로 마이그레이션을 참조하십시오.

이 예제 사용

- *EH1HDMB1FH2TC*를 OAI의 ID로 대체합니다. OAI의 ID를 찾으려면 CloudFront 콘솔에서 Origin Access Identity page(오리진 액세스 ID 페이지)를 확인하거나 CloudFront API에서 ListCloudFrontOriginAccessIdentities를 사용합니다.
- *amzn-s3-demo-bucket*을 사용 중인 버킷의 이름으로 바꿉니다.

```
{
    "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:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

Amazon S3 스토리지 렌즈 액세스 관리

Amazon S3 스토리지 렌즈 권한 부여

S3 스토리지 렌즈는 지표를 집계하고 Amazon S3 콘솔 버킷 페이지의 계정 스냅샷 섹션에 이 정보를 표시합니다. S3 스토리지 렌즈는 또한 인사이트와 추세를 시각화하고, 이상치에 플래그를 지정하고, 스토리지 비용 최적화와 데이터 보호 모범 사례 적용을 위한 권장 사항을 수신하는 데 사용할 수 있는 대화형 대시보드를 제공합니다. 대시보드에 있는 드릴다운 옵션을 통해 조직, 계정, AWS 리전, 스토리지 클래스, 버킷, 접두사 또는 스토리지 렌즈 그룹 수준에서 인사이트를 생성하거나 시각화할 수 있습니다. 또한 CSV 또는 Parquet 형식의 일일 지표 내보내기를 S3 버킷으로 전송할 수 있습니다.

S3 스토리지 렌즈에서 집계된 스토리지 사용량 지표를 Amazon S3 버킷으로 내보내면 추가로 분석할 수 있습니다. S3 스토리지 렌즈가 지표 내보내기를 배치하는 버킷을 대상 버킷이라고 합니다. S3 스토리지 렌즈 지표 내보내기를 설정할 때 대상 버킷에 대한 버킷 정책이 있어야 합니다. 자세한 내용은 Amazon S3 스토리지 렌즈를 사용하여 스토리지 활동 및 사용량 평가 섹션을 참조하세요.

다음 예시 버킷 정책은 대상 버킷에 객체(PUT 요청)를 쓸 수 있는 권한을 Amazon S3에 부여합니다. S3 스토리지 렌즈 지표 내보내기를 설정할 때는 대상 버킷에 이와 같은 버킷 정책을 사용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3StorageLensExamplePolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "storage-lens.s3.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket/destination-prefix/
StorageLens/111122223333/*"
            ],
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control",
                    "aws:SourceAccount": "111122223333",
                    "aws:SourceArn": "arn:aws:s3:region-code:111122223333:storage-
lens/storage-lens-dashboard-configuration-id"
                }
            }
        }
```

```
    ]
}
```

S3 스토리지 렌즈 조직 수준 지표 내보내기를 설정할 때, 이전 버킷 정책의 Resource 문을 다음과 같이 수정합니다.

```
"Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/destination-prefix/
StorageLens/your-organization-id/*",
```

S3 인벤토리, S3 분석 및 S3 인벤토리 보고서 권한 관리

S3 인벤토리 및 S3 분석 권한 부여

S3 인벤토리는 버킷 내에 객체 목록을 생성하고, S3 분석 스토리지 클래스 분석 내보내기는 분석에서 사용되는 데이터의 출력 파일을 생성합니다. 인벤토리가 객체를 열거하는 버킷을 원본 버킷이라고 합니다. 인벤토리 파일 또는 분석 내보내기 파일이 작성되는 버킷을 대상 버킷이라고 합니다. 인벤토리 또는 분석 내보내기를 설정할 때는 대상 버킷의 버킷 정책을 생성해야 합니다. 자세한 내용은 S3 Inventory를 사용한 데이터 카탈로그화 및 분석 및 Amazon S3 분석 - 스토리지 클래스 분석 섹션을 참조하세요.

다음 버킷 정책 예는 원본 버킷의 계정에서 대상 버킷에 객체를 쓸 수 있는 Amazon S3 권한(PUT 요청)을 부여합니다. S3 인벤토리와 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:::DOC-EXAMPLE-DESTINATION-BUCKET/*"
            ],
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET"
```

```
                },
                "StringEquals": {
                    "aws:SourceAccount": "111122223333",
                    "s3:x-amz-acl": "bucket-owner-full-control"
                }
            }
        }
    ]
}
```

S3 인벤토리 보고서 구성 생성 제어

S3 Inventory를 사용한 데이터 카탈로그화 및 분석는 S3 버킷 내 객체들의 목록과 각 객체의 메타데이터를 생성합니다. `s3:PutInventoryConfiguration` 권한을 통해 사용자는 기본적으로 사용 가능한 모든 객체 메타데이터 필드를 포함하는 인벤토리 구성을 생성하고 인벤토리를 저장할 대상 버킷을 지정할 수 있습니다. 대상 버킷의 객체에 대한 읽기 권한이 있는 사용자는 인벤토리 보고서에서 사용할 수 있는 모든 객체 메타데이터 필드에 액세스할 수 있습니다. S3 인벤토리에서 사용 가능한 메타데이터 필드에 대한 자세한 내용은 Amazon S3 인벤토리 목록 섹션을 참조하십시오.

사용자가 S3 인벤토리 보고서를 구성하지 못하도록 제한하려면 해당 사용자의 `s3:PutInventoryConfiguration` 권한을 제거하세요.

S3 인벤토리 보고서 구성의 일부 객체 메타데이터 필드는 선택 사항입니다. 즉, 기본적으로 사용할 수 있지만 사용자에게 `s3:PutInventoryConfiguration` 권한을 부여하면 제한될 수 있습니다. `s3:InventoryAccessibleOptionalFields` 조건 키를 사용하여 사용자가 보고서에 이러한 선택적 메타데이터 필드를 포함할 수 있는지를 제어할 수 있습니다. S3 인벤토리에서 사용할 수 있는 선택적 메타데이터 필드 목록은 Amazon Simple Storage Service API 참조의 OptionalFields 섹션을 참조하세요.

특정 선택적 메타데이터 필드가 포함된 인벤토리 구성을 생성할 권한을 사용자에게 부여하려면 `s3:InventoryAccessibleOptionalFields` 조건 키를 사용하여 버킷 정책의 조건을 구체화하세요.

다음 예시 정책은 사용자(*Ana*)에게 조건부로 인벤토리 구성을 생성할 수 있는 권한을 부여합니다. 정책의 `ForAllValues:StringEquals` 조건은 `s3:InventoryAccessibleOptionalFields` 조건 키를 사용하여 허용되는 두 개의 선택적 메타데이터 필드(`Size`, `StorageClass`)를 지정합니다. 따라서 *Ana*가 인벤토리 구성을 생성할 때 포함할 수 있는 유일한 선택적 메타데이터 필드는 `Size` 및 `StorageClass`입니다.

```
{
```

```
 "Id": "InventoryConfigPolicy",
 "Version": "2012-10-17",
 "Statement": [{
   "Sid": "AllowInventoryCreationConditionally",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/Ana"
   },
   "Action":
    "s3:PutInventoryConfiguration",
   "Resource":
    "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET",
   "Condition": {
    "ForAllValues:StringEquals": {
     "s3:InventoryAccessibleOptionalFields": [
        "Size",
        "StorageClass"
        ]
      }
    }
   }
 ]
}
```

사용자가 특정 선택적 메타데이터 필드를 포함하는 S3 인벤토리 보고서를 구성하지 못하도록 제한하려면 소스 버킷의 버킷 정책에 명시적 Deny 문을 추가하세요. 다음 예시 버킷 정책은 사용자 *Ana*가 소스 버킷 *DOC-EXAMPLE-SOURCE-BUCKET*에서 선택적 ObjectAccessControlList 또는 ObjectOwner 메타데이터 필드를 포함하는 인벤토리 구성을 생성하는 것을 거부합니다. 사용자 *Ana*는 여전히 다른 선택적 메타데이터 필드를 사용하여 인벤토리 구성을 생성할 수 있습니다.

```
{
 "Id": "InventoryConfigSomeFields",
 "Version": "2012-10-17",
 "Statement": [{
   "Sid": "AllowInventoryCreation",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/Ana"
   },
   "Action": "s3:PutInventoryConfiguration",
   "Resource":
    "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET",
```

```
  },
  {
   "Sid": "DenyCertainInventoryFieldCreation",
   "Effect": "Deny",
   "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/Ana"
   },
   "Action": "s3:PutInventoryConfiguration",
   "Resource":
     "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET",
   "Condition": {
    "ForAnyValue:StringEquals": {
     "s3:InventoryAccessibleOptionalFields": [
        "ObjectOwner",
        "ObjectAccessControlList"
        ]
      }
    }
   }
 ]
}
```

Note

버킷 정책에서 `s3:InventoryAccessibleOptionalFields` 조건 키를 사용해도 기존 인벤토리 구성을 기반으로 하는 인벤토리 보고서 전송에는 영향을 미치지 않습니다.

Important

이전 예시에서 볼 수 있듯이 `Allow` 효과를 적용하여 `ForAllValues`를 사용하거나 `Deny` 효과를 적용하여 `ForAnyValue`를 사용하는 것이 좋습니다.

`Deny` 효과와 함께 `ForAllValues`를 사용하거나 `Allow` 효과와 함께 `ForAnyValue`를 사용하지 마세요. 이러한 조합은 지나치게 제한적일 수 있고 인벤토리 구성 삭제를 차단할 수 있기 때문입니다.

`ForAllValues` 및 `ForAnyValue` 조건 집합 연산자에 대해 자세히 알아보려면 IAM 사용 설명서의 다중 값 컨텍스트 키를 참조하세요.

MFA 필요

Amazon S3은 Amazon S3 리소스에 액세스하기 위해 멀티 팩터 인증(MFA)을 적용할 수 있는 기능인 MFA 보호 API 액세스를 지원합니다. 멀티 팩터 인증은 AWS 환경에 적용할 수 있는 추가 보안 레벨을 제공합니다. MFA는 사용자가 유효한 MFA 코드를 제공하여 MFA 디바이스를 물리적으로 가지고 있음을 증명하게 하는 보안 기능입니다. 자세한 내용은 AWS 멀티 팩터 인증을 참조하십시오. Amazon S3 리소스에 액세스하는 모든 요청에 대해 MFA가 필요할 수 있습니다.

MFA가 필요하도록 강제하려면 버킷 정책 내에 `aws:MultiFactorAuthAge` 조건 키를 사용합니다. IAM 사용자는 AWS Security Token Service(AWS STS)에서 발급한 임시 보안 인증을 사용하여 Amazon S3 리소스에 액세스할 수 있습니다. AWS STS 요청 시 MFA 코드를 제공합니다.

멀티 팩터 인증을 사용한 요청을 Amazon S3가 수신하면 얼마나 오래 전(초 단위)에 임시 보안 인증이 생성되었는지를 `aws:MultiFactorAuthAge` 조건 키가 숫자 값으로 제공합니다. 요청 시 제공된 임시 자격 증명이 MFA 디바이스를 사용하여 만들어진 경우 이 키 값은 null(없음)입니다. 버킷 정책에서는 다음 예시에서 나온 것처럼 이 값을 확인할 수 있는 조건을 추가할 수 있습니다.

이 예에서 MFA를 사용하여 요청이 인증되지 않으면 정책은 *`/taxdocuments`* 버킷의 *`amzn-s3-demo-bucket`* 폴더에서 모든 Amazon S3 작업을 거부합니다. MFA에 대한 자세한 내용은 IAM 사용 설명서의 AWS에서 멀티 팩터 인증(MFA) 사용을 참조하십시오.

```
{
    "Version": "2012-10-17",
    "Id": "123",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/taxdocuments/*",
        "Condition": { "Null": { "aws:MultiFactorAuthAge": true }}
      }
    ]
 }
```

`Condition` 블록의 `Null` 조건이 `true`로 평가되려면 `aws:MultiFactorAuthAge` 조건 키 값이 null이 되어 요청 시 임시 보안 인증이 MFA 디바이스 없이 만들어졌음을 나타내야 합니다.

다음 버킷 정책은 이전 버킷 정책을 확장한 것입니다. 다음 정책에는 두 가지 정책 문이 포함됩니다. 하나의 문은 모든 사용자에게 버킷(*`amzn-s3-demo-bucket`*)에 대한 `s3:GetObject` 권한을 허용합니

다. 다른 문은 버킷의 *amzn-s3-demo-bucket/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:::amzn-s3-demo-bucket/taxdocuments/*",
        "Condition": { "Null": { "aws:MultiFactorAuthAge": true } }
      },
      {
        "Sid": "",
        "Effect": "Allow",
        "Principal": "*",
        "Action": ["s3:GetObject"],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
      }
    ]
 }
```

aws:MultiFactorAuthAge 키가 유효한 기간을 제한하는 숫자 조건을 사용해도 됩니다. aws:MultiFactorAuthAge 키로 지정된 기간은 요청을 인증하는 데 사용되는 임시 보안 인증의 수명 주기와 무관합니다.

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:::amzn-s3-demo-bucket/taxdocuments/*",
        "Condition": {"Null": {"aws:MultiFactorAuthAge": true }}
      },
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/taxdocuments/*",
        "Condition": {"NumericGreaterThan": {"aws:MultiFactorAuthAge": 3600 }}
       },
       {
         "Sid": "",
         "Effect": "Allow",
         "Principal": "*",
         "Action": ["s3:GetObject"],
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
       }
    ]
}
```

사용자가 객체를 삭제하지 못하도록 방지

기본적으로 사용자에게는 아무 권한이 없습니다. 그러나 정책을 생성할 때 사용자에게 부여하고 싶지 않은 권한을 부여할 수 있습니다. 그러한 권한 허점을 피하기 위해 명시적 거부를 추가하여 더 엄격한 액세스 정책을 작성할 수 있습니다.

객체를 삭제하지 못하도록 명시적으로 사용자 또는 계정을 차단하려면 버킷 정책에 `s3:DeleteObject`, `s3:DeleteObjectVersion` 및 `s3:PutLifecycleConfiguration` 권한 작업을 추가해야 합니다. Amazon S3에서 객체의 수명이 만료되었을 때 이를 제거할 수 있도록 명시적으로 `DELETE Object` API 작업을 직접적으로 호출하거나 해당 수명 주기(객체 수명 주기 관리 참조)를 구성하여 객체를 삭제할 수 있으므로 세 개의 작업 모두 필요합니다.

다음 정책 예시에서는 사용자 *MaryMajor*의 `DELETE Object` 권한을 명시적으로 거부합니다. 명시적 `Deny` 문은 이미 부여된 다른 모든 권한에 항상 우선합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "statement1",
      "Effect": "Allow",
```

```
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/MaryMajor"
      },
      "Action": [
        "s3:GetObjectVersion",
        "s3:GetBucketAcl"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket1",
   "arn:aws:s3:::amzn-s3-demo-bucket1/*"
      ]
    },
    {
      "Sid": "statement2",
      "Effect": "Deny",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/MaryMajor"
      },
      "Action": [
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:PutLifecycleConfiguration"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket1",
     "arn:aws:s3:::amzn-s3-demo-bucket1/*"
      ]
    }
  ]
}
```

조건 키를 사용한 버킷 정책 예시

권한 부여 시 액세스 정책 언어를 사용하여 조건을 지정할 수 있습니다. 선택적 `Condition` 요소 또는 `Condition` 블록을 사용하여 정책이 적용되는 시기에 대한 조건을 지정할 수 있습니다.

객체 및 버킷 작업에 Amazon S3 조건 키를 사용하는 정책은 다음 예제를 참조하십시오. 조건 키에 대한 자세한 내용은 Amazon S3의 정책 조건 키 섹션을 참조하십시오. 정책에 지정할 수 있는 Amazon S3 작업, 조건 키, 리소스의 전체 목록은 서비스 승인 참조에서 Amazon S3에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

예제: 객체 작업을 위한 Amazon S3 조건 키

다음 예제는 객체 작업에 Amazon S3별 조건 키를 사용하는 방법을 보여줍니다. 정책에 지정할 수 있는 Amazon S3 작업, 조건 키, 리소스의 전체 목록은 서비스 승인 참조에서 Amazon S3에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

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 조건 키를 통해 사용자가 객체를 업로드할 때 특정 액세스 권한이 필요하도록 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 제어 목록) 개요 단원을 참조하십시오.

주제

- 예제 1: 서버 측 암호화를 사용하여 객체를 저장하도록 요구하는 s3:PutObject 권한 부여
- 예제 2: 복사 소스에 대한 제한이 있는 객체를 복사하기 위한 s3:PutObject 권한 부여
- 예시 3: 특정 버전의 객체에 대한 액세스 권한 부여
- 예시 4: 객체 태그 기반 권한 부여
- 예시 5: 버킷 소유자의 AWS 계정 ID로 액세스 제한
- 예시 6: 최소 TLS 버전 요구
- 예제 7: Deny 문에서 특정 보안 주체 제외
- 예시 8: 클라이언트가 객체 키 이름 또는 ETags 기반으로 객체를 조건부로 업로드하도록 강제 적용

예제 1: 서버 측 암호화를 사용하여 객체를 저장하도록 요구하는 **`s3:PutObject`** 권한 부여

Account A에서 버킷을 소유하고 있다고 가정합니다. 계정 관리자는 Account A의 사용자인 Jane에게 항상 Amazon S3 관리형 키(SSE-S3)로 서버 측 암호화를 요청한다는 조건으로 객체를 업로드할 수 있는 권한을 부여하려고 합니다. Account A 관리자는 다음과 같은 `s3:x-amz-server-side-encryption` 조건 키를 사용하여 이 요구 사항을 지정할 수 있습니다. 다음 `Condition` 블록의 키-값 페어는 `s3:x-amz-server-side-encryption` 조건 키와 SSE-S3(`AES256`)를 암호화 유형으로 지정합니다.

```
"Condition": {
```

```
    "StringNotEquals": {
        "s3:x-amz-server-side-encryption": "AES256"
    }}
```

AWS CLI를 사용하여 이 권한을 테스트할 때는 다음 예제와 같이 `--server-side-encryption` 파라미터를 사용하여 필요한 암호화를 추가해야 합니다. 이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key HappyFace.jpg --body c:
\HappyFace.jpg --server-side-encryption "AES256" --profile AccountAadmin
```

예제 2: 복사 소스에 대한 제한이 있는 객체를 복사하기 위한 **s3:PutObject** 권한 부여

PUT 객체 요청에서 소스 객체를 지정하면 요청은 복사 작업에 해당합니다(CopyObject 참조). 따라서 버킷 소유자는 원본에 대한 제한이 있는 객체 복사 권한을 사용자에게 부여할 수 있습니다. 예를 들면 다음과 같습니다.

- 지정한 소스 버킷에서만 객체 복사를 허용합니다(예: *amzn-s3-demo-source-bucket*).
- 지정한 소스 버킷에서 객체를 복사할 수 있으며, *public/*처럼 키 이름 접두사가 특정 접두사로 시작하는 객체(예: *amzn-s3-demo-source-bucket/public/**)만 복사할 수 있습니다.
- 소스 버킷에서 특정 객체만 복사할 수 있습니다(예: *amzn-s3-demo-source-bucket/example.jpg*).

다음 버킷 정책은 사용자(*Dave*)에게 `s3:PutObject` 권한을 부여합니다. 이 정책은 요청에 `s3:x-amz-copy-source` 헤더를 포함하고 헤더 값으로 */amzn-s3-demo-source-bucket/public/** 키 이름 접두사를 지정하는 조건을 충족하는 객체만 복사하도록 허용합니다. 이 정책 예를 사용하려면 *user input placeholders*를 실제 정보로 바꾸세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
       {
            "Sid": "cross-account permission to user in your own account",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
```

```
        },
        {
            "Sid": "Deny your user permission to upload object if copy source is not /
bucket/prefix",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-source-bucket/*",
            "Condition": {
                "StringNotLike": {
                    "s3:x-amz-copy-source": "amzn-s3-demo-source-bucket/public/*"
                }
            }
        }
    ]
}
```

AWS CLI에서 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 Amazon S3 API 참조의 AWS CLI를 사용하여 Amazon S3에서 개발을 참조하세요.

```
aws s3api copy-object --bucket amzn-s3-demo-source-bucket --key HappyFace.jpg
--copy-source amzn-s3-demo-source-bucket/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 `StringNotLike`에서 `StringNotEquals`로 조건을 변경한 뒤 다음 예제와 같이 정확한 객체 키를 지정합니다. 이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
"Condition": {
       "StringNotEquals": {
           "s3:x-amz-copy-source": "amzn-s3-demo-source-bucket/public/
PublicHappyFace1.jpg"
       }
```

```
}
```

예시 3: 특정 버전의 객체에 대한 액세스 권한 부여

Account A에서 버전 관리가 활성화된 버킷을 소유하고 있다고 가정합니다. 버킷에는 몇 가지 버전의 *HappyFace.jpg* 객체가 있습니다. 이제 Account A 관리자는 사용자 *Dave*에게 특정 버전의 객체만 가져올 수 있는 권한을 부여하려고 합니다. 계정 관리자는 다음 예제와 같이 조건부로 사용자 *Dave*에게 `s3:GetObjectVersion` 권한을 부여하여 이 작업을 수행할 수 있습니다. `Condition` 블록의 키-값 페어는 `s3:VersionId` 조건 키를 지정합니다. 이 경우 지정된 버전 관리를 사용하는 버킷에서 객체를 가져오려면 *Dave*에서 정확한 객체 버전 ID를 알고 있어야 합니다. 이 정책 예를 사용하려면 *user input placeholders*를 실제 정보로 바꾸세요.

자세한 내용은 Amazon Simple Storage Service API 참조에서 GetObject를 참조하십시오.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:GetObjectVersion",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/HappyFace.jpg"
        },
        {
            "Sid": "statement2",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:GetObjectVersion",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/HappyFace.jpg",
            "Condition": {
                "StringNotEquals": {
                    "s3:VersionId": "AaaHbAQitwiL_h47_44lRO2DDfLlBO5e"
                }
            }
        }
    ]
}
```

AWS CLI에서 정책 테스트

`--version-id` 파라미터와 함께 AWS CLI `get-object` 명령을 통해 가져올 특정 객체 버전을 식별하여 이 정책의 권한을 테스트할 수 있습니다. 이 명령은 지정된 버전의 객체를 가져와 *`OutputFile.jpg`* 파일에 저장합니다.

```
aws s3api get-object --bucket amzn-s3-demo-bucket --key HappyFace.jpg OutputFile.jpg --
version-id AaaHbAQitwiL_h47_44lRO2DDfLlBO5e --profile AccountADave
```

예시 4: 객체 태그 기반 권한 부여

Amazon S3 작업에 객체 태그 지정 조건 키를 사용하는 방법에 대한 예제는 태그 지정 및 액세스 제어 정책 섹션을 참조하세요.

예시 5: 버킷 소유자의 AWS 계정 ID로 액세스 제한

`aws:ResourceAccount` 또는 `s3:ResourceAccount` 조건 키를 사용하여 특정 AWS 계정 ID가 소유한 Amazon S3 버킷에 대한 사용자, 역할 또는 애플리케이션 액세스를 제한하는 IAM 또는 가상 프라이빗 클라우드(VPC) 엔드포인트 정책을 작성할 수 있습니다. 이 조건 키를 사용하면 소유하지 않은 버킷에 VPC 내의 클라이언트가 액세스하는 것을 제한할 수 있습니다.

그러나 일부 AWS 서비스는 AWS 관리되는 버킷에 대한 액세스에 의존합니다. 따라서 IAM 정책에서 `aws:ResourceAccount` 또는 `s3:ResourceAccount` 키를 사용하면 이러한 리소스에 대한 액세스에도 영향을 미칠 수 있습니다. 자세한 정보는 다음 자료를 참조하세요.

- AWS PrivateLink 가이드의 지정된 AWS 계정에서 버킷에 대한 액세스 제한
- Amazon ECR 안내서에서 Amazon ECR이 사용하는 버킷에 대한 액세스 제한
- AWS Systems Manager 가이드의 AWS 관리되는 Amazon S3 버킷에 대해 Systems Manager에 대한 필수 액세스 권한 제공

`aws:ResourceAccount` 및 `s3:ResourceAccount` 조건 키에 대한 자세한 내용과 사용 방법을 보여주는 예제는 AWS 스토리지 블로그에서 특정 AWS 계정에서 소유한 Amazon S3 버킷에 대한 액세스 제한을 참조하세요.

예시 6: 최소 TLS 버전 요구

`s3:TlsVersion` 조건 키를 사용하면 IAM, 가상 프라이빗 클라우드 엔드포인트(VPCE) 또는 버킷 정책을 작성하여 클라이언트가 사용한 TLS 버전에 따라 Amazon S3 버킷에 대한 사용자 또는 애플리케

이션의 액세스를 제한할 수 있습니다. 이 조건 키를 사용하여 최소 TLS 버전을 요구하는 정책을 작성할 수 있습니다.

> **Note**
>
> AWS 서비스가 사용자를 대신하여 다른 AWS 서비스를 직접적으로 호출하면(서비스 간 호출), `s3:TlsVersion`, `aws:SecureTransport`, `aws:SourceIp`, `aws:VpcSourceIp`를 포함한 특정 네트워크별 권한 부여 컨텍스트가 삭제됩니다. 정책에서 `Deny` 문과 함께 이러한 조건 키를 사용하는 경우 AWS 서비스 위탁자가 의도치 않게 차단될 수 있습니다. 보안 요구 사항을 유지하면서 AWS 서비스가 제대로 작동하도록 하려면 값이 `false`인 `aws:PrincipalIsAWSService` 조건 키를 추가하여 `Deny` 문에서 서비스 위탁자를 제외합니다. 예시:
>
> ```
> {
>   "Effect": "Deny",
>   "Action": "s3:*",
>   "Resource": "*",
>   "Condition": {
>     "Bool": {
>       "aws:SecureTransport": "false",
>       "aws:PrincipalIsAWSService": "false"
>     }
>   }
> }
> ```
>
> 이 정책은 HTTPS가 사용되지 않을 때(`aws:SecureTransport`가 false) S3 작업에 대한 액세스를 거부하지만 AWS 서비스 위탁자가 아닌 사용자에게만 적용됩니다. 이렇게 하면 AWS 서비스 위탁자를 제외한 모든 위탁자에 조건부 제한이 적용됩니다.

Example

다음 예제 버킷 정책은 TLS 버전이 1.2보다 낮은 클라이언트(예: 1.1 또는 1.0)의 `PutObject` 요청을 거부합니다. 이 정책 예를 사용하려면 *`user input placeholders`*를 실제 정보로 바꾸세요.

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

```
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1",
                "arn:aws:s3:::amzn-s3-demo-bucket1/*"
            ],
            "Condition": {
                "NumericLessThan": {
                    "s3:TlsVersion": 1.2
                }
            }
        }
    ]
}
```

Example

다음 버킷 정책 예제는 TLS 버전이 1.1보다 높은 클라이언트(예: 1.2, 1.3 이상)의 `PutObject` 요청을 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1",
                "arn:aws:s3:::amzn-s3-demo-bucket1/*"
            ],
            "Condition": {
                "NumericGreaterThan": {
                    "s3:TlsVersion": 1.1
                }
            }
        }
    ]
}
```

예제 7: **Deny** 문에서 특정 보안 주체 제외

다음 버킷 정책은 *123456789012* 계정 번호가 있는 보안 주체를 제외하고 *amzn-s3-demo-bucket*에 대한 s3:GetObject 액세스를 거부합니다. 이 정책 예를 사용하려면 *user input placeholders*를 실제 정보로 바꾸세요.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAccessFromPrincipalNotInSpecificAccount",
      "Principal": {
        "AWS": "*"
      },
      "Action": "s3:GetObject",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:PrincipalAccount": [
            "123456789012"
          ]
        }
      }
    }
  ]
}
```

예시 8: 클라이언트가 객체 키 이름 또는 ETags 기반으로 객체를 조건부로 업로드하도록 강제 적용

조건부 쓰기를 사용하면 S3 작업에 사전 조건을 지정하기 위해 WRITE 요청에 헤더를 더 추가할 수 있습니다. 이 헤더는 충족되지 않으면 S3 작업이 실패할 수 있는 조건을 지정합니다. 예를 들어, 객체 업로드 시 버킷에 이미 같은 키 이름을 가진 객체가 없는지 확인하여 기존 데이터를 덮어쓰지 않도록 방지할 수 있습니다. 또는 객체를 쓰기 전에 Amazon S3에서 객체의 엔터티 태그(ETag)를 확인할 수 있습니다.

버킷 정책의 조건을 사용하여 조건부 쓰기를 강제 적용하는 버킷 정책 예시는 the section called “조건부 쓰기 강제 적용” 섹션을 참조하세요.

예제: 버킷 작업을 위한 Amazon S3 조건 키

다음 정책 예제는 버킷 작업에 Amazon S3별 조건 키를 사용하는 방법을 보여줍니다.

주제

- 예제 1: IP 주소를 조건으로 s3:GetObject 권한 부여
- 예제 2: 특정 접두사가 있는 버킷의 객체 목록 가져오기
- 예제 3: 최대 키 수 설정

예제 1: IP 주소를 조건으로 **`s3:GetObject`** 권한 부여

요청이 특정 범위의 IP 주소(예:*192.0.2.**)에서 비롯된 경우 인증된 사용자가 `s3:GetObject` 작업을 사용하도록 권한을 부여합니다. 단, 제외하려는 IP 주소(예: *192.0.2.188*)는 예외입니다. `Condition` 블록에서 `IpAddress` 및 `NotIpAddress`가 조건에 해당하며, 각 조건에는 평가를 위한 키-값 페어가 제공됩니다. 이 예제에서의 키-값 페어는 모두 `aws:SourceIp` AWS 차원의 키를 사용합니다. 이 정책 예를 사용하려면 *`user input placeholders`*를 실제 정보로 바꾸세요.

> **Note**
>
> `Condition` 블록에 지정된 `IPAddress` 및 `NotIpAddress` 키 값은 RFC 4632에 설명된 대로 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:::amzn-s3-demo-bucket/*",
            "Condition" : {
                "IpAddress" : {
                    "aws:SourceIp": "192.0.2.0/24"
                },
                "NotIpAddress" : {
```

```
                    "aws:SourceIp": "192.0.2.188/32"
                }
            }
        }
    ]
}
```

Amazon S3 정책에서 다른 AWS 전역 조건 키를 사용할 수도 있습니다. 예를 들어 VPC 엔드포인트에 대한 버킷 정책에서 `aws:SourceVpce` 및 `aws:SourceVpc` 조건 키를 지정할 수 있습니다. 구체적인 예는 버킷 정책을 사용하여 VPC 엔드포인트에서 액세스 제어 섹션을 참조하십시오.

> **Note**
>
> 일부 AWS 전역 조건 키의 경우, 특정 리소스 유형만 지원됩니다. 따라서 Amazon S3가 사용하려는 전역 조건 키 및 리소스 유형을 지원하는지 아니면 Amazon S3 관련 조건 키를 대신 사용해야 하는지 확인하세요. Amazon S3에 지원되는 리소스 유형과 조건 키의 전체 목록은 서비스 승인 참조에서 Amazon S3에 사용되는 작업, 리소스 및 조건 키를 참조하세요.
> S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

예제 2: 특정 접두사가 있는 버킷의 객체 목록 가져오기

`s3:prefix` 조건 키를 사용하여 ListObjectsV2 API 작업의 응답을 특정 접두사가 있는 키 이름으로 제한할 수 있습니다. 버킷 소유자인 경우 이 조건 키로 사용자가 버킷에 있는 특정 접두사의 내용을 나열하도록 제한할 수 있습니다. `s3:prefix` 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. 예를 들어, 키 이름이 *public/object1.jpg* 및 *public/object2.jpg*인 2개의 객체가 있는 경우 콘솔에서는 *public* 폴더 아래에 객체가 표시됩니다. Amazon S3 API에서 이러한 객체는 폴더의 객체가 아니라 접두사가 있는 객체입니다. 접두사 및 구분 기호를 사용하여 액세스 권한을 필터링하는 방법에 대한 자세한 내용은 사용자 정책을 사용하여 버킷에 대한 액세스 제어 단원을 참조하십시오.

다음 시나리오에서 버킷 소유자와 사용자가 속한 상위 계정은 동일합니다. 따라서 버킷 소유자는 버킷 정책 또는 사용자 정책을 사용하여 액세스 권한을 부여할 수 있습니다. `ListObjectsV2` API 작업에 사용할 수 있는 다른 조건 키에 대한 자세한 내용은 ListObjectsV2 섹션을 참조하세요.

> **Note**
>
> 버킷에서 버전 관리가 활성화된 경우 버킷의 객체를 나열하려면 다음 정책에서 `s3:ListBucket` 권한 대신 `s3:ListBucketVersions` 권한을 부여해야 합니다. 또한, `s3:ListBucketVersions` 권한은 `s3:prefix` 조건 키를 지원합니다.

사용자 정책

다음 사용자 정책은 사용자에게 요청에서 값이 *`projects`*인 접두사를 지정하도록 요구하는 `Condition` 문을 통해 `s3:ListBucket` 권한을 부여합니다(ListObjectsV2 참조). 이 정책 예를 사용하려면 *`user input placeholders`*를 실제 정보로 바꾸세요.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action": "s3:ListBucket",
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket",
         "Condition" : {
             "StringEquals" : {
                 "s3:prefix": "projects"
             }
          }
       },
      {
         "Sid":"statement2",
         "Effect":"Deny",
         "Action": "s3:ListBucket",
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
         "Condition" : {
             "StringNotEquals" : {
                 "s3:prefix": "projects"
             }
          }
       }
    ]
}
```

Condition 문은 사용자가 *projects* 접두사가 있는 객체 키만 나열하도록 제한합니다. 추가된 명시적 Deny 문은 사용자가 다른 권한을 가지고 있든 없든 다른 접두사가 있는 키를 나열하는 것을 거부합니다. 예를 들어, 이전 사용자 정책 업데이트 또는 버킷 정책을 통해 사용자가 제한 없이 객체 키를 나열할 수 있는 권한을 얻을 수 있습니다. 명시적 Deny 문은 항상 Allow 문보다 우선하므로, 사용자가 *projects* 접두사가 있는 키 이외의 키를 나열하려고 하면 요청이 거부됩니다.

버킷 정책

사용자를 식별하는 Principal 요소를 위의 사용자 정책에 추가하는 경우, 다음 예제에서처럼 버킷 정책을 갖게 됩니다. 이 정책 예를 사용하려면 *user input placeholders*를 실제 정보로 바꾸세요.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/bucket-owner"
         },
         "Action":  "s3:ListBucket",
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
         "Condition" : {
             "StringEquals" : {
                 "s3:prefix": "projects"
             }
          }
       },
      {
         "Sid":"statement2",
         "Effect":"Deny",
         "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/bucket-owner"
         },
         "Action": "s3:ListBucket",
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
         "Condition" : {
             "StringNotEquals" : {
                 "s3:prefix": "projects"
             }
          }
       }
```

```
    ]
}
```

AWS CLI에서 정책 테스트

다음 `list-object` AWS CLI 명령을 사용하는 정책을 테스트할 수 있습니다. 이 명령에서 `--profile` 파라미터를 사용하여 사용자 자격 증명을 제공합니다. AWS CLI 사용 및 설정에 대한 자세한 내용은 Amazon S3 API 참조의 AWS CLI를 사용하여 Amazon S3에서 개발을 참조하세요.

```
aws s3api list-objects --bucket amzn-s3-demo-bucket --prefix projects --
profile AccountA
```

예제 3: 최대 키 수 설정

`s3:max-keys` 조건 키를 사용하여 요청자가 ListObjectsV2 또는 ListObjectVersions 요청에서 반환할 수 있는 최대 키 수를 설정할 수 있습니다. 기본적으로 이러한 API 작업은 최대 1,000개의 키를 반환합니다. `s3:max-keys`와 함께 사용할 수 있는 숫자 조건 연산자 목록은 IAM 사용 설명서의 숫자 조건 연산자를 참조하십시오.

Amazon S3의 ID 기반 정책

기본적으로 사용자 및 역할은 Amazon S3 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console, AWS Command Line Interface(AWS CLI) 또는 AWS API를 사용해 태스크를 수행할 수 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 맡을 수 있습니다.

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

각 리소스 유형에 대한 ARN 형식을 포함하여 Amazon S3에서 정의한 작업 및 리소스 유형에 대한 자세한 내용은 서비스 권한 부여 참조의 Amazon S3의 작업, 리소스 및 조건 키를 참조하세요.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

주제

- 정책 모범 사례
- 사용자 정책을 사용하여 버킷에 대한 액세스 제어
- Amazon S3의 ID 기반 정책 예시

정책 모범 사례

ID 기반 정책에 따라 해당 계정에서 누가 Amazon S3 리소스를 생성, 액세스 또는 삭제할 수 있는지 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따릅니다.

- AWS 관리형 정책으로 시작하고 최소 권한을 향해 나아가기 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. AWS 계정에서 사용할 수 있습니다. 사용 사례에 고유한 AWS 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS직무에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. AWS CloudFormation와 같이, 특정 AWS 서비스를 통해 사용되는 경우에만 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을 확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer에서 정책 검증을 참조하세요.
- 다중 인증(MFA) 필요 – AWS 계정에 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우, 추가 보안을 위해 MFA를 설정합니다. API 작업을 직접 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 MFA를 통한 보안 API 액세스를 참조하세요.

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

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
- 연습 요약
- 연습 준비
- 1단계: 버킷 만들기
- 2단계: IAM 사용자 및 그룹 만들기
- 3단계: IAM 사용자에게 권한이 없는지 확인
- 4단계: 그룹 수준 권한 부여
- 5단계: IAM 사용자인 Alice에게 특정 권한 부여
- 6단계: IAM 사용자인 Bob에게 특정 권한 부여
- 7단계: Private 폴더에 보안 지정
- 8단계: 정리
- 관련 리소스

버킷 및 폴더 기본 사항

Amazon S3 데이터 모델은 단순한 구조를 가지고 있습니다. 사용자가 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 버킷에 있는 이러한 논리적 폴더와 하위 폴더의 보기를 표시할 수 있습니다.

콘솔에는 `companybucket`이라는 버킷에 세 개의 폴더 `Private`, `Development` 및 `Finance`와 하나의 객체 `s3-dg.pdf`가 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.xls`라는 객체를 `Development` 폴더에 업로드하면 콘솔에서 객체가 업로드되고 객체에 키 `Development/Projects1.xls`가 제공됩니다.

  키에서 `Development`는 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 Development 버킷의 companybucket 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.

콘솔은 객체 키를 사용하여 논리적 계층 구조를 추론하고 있습니다. Amazon S3에는 물리적 계층 구조가 없습니다. Amazon S3에는 객체가 포함된 버킷만 플랫 파일 구조로 존재합니다. 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 Finance와 루트 수준 객체인 s3-dg.pdf가 포함된 논리적 계층 구조가 생성됩니다. Amazon S3 콘솔에서 버킷 이름을 선택하면 루트 수준 항목이 나타납니다. 콘솔에는 최상위 수준 접두사(Private/, Development/ 및 Finance/)가 루트 수준 폴더로 표시됩니다. 객체 키 s3-dg.pdf에는 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 예시에서는 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 생성합니다. 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 계정 자격 증명으로 AWS Management Console에 연결하고 다른 브라우저를 사용하여 IAM 사용자 자격 증명과 연결합니다.

AWS 계정 자격 증명으로 AWS Management Console에 로그인하려면 https://console.aws.amazon.com/으로 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 Console 로그인 페이지를 참조하십시오.

IAM 사용자를 위한 로그인 링크 제공

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색(Navigation) 창에서 IAM 대시보드(IAM Dashboard)를 선택합니다.
3. IAM 사용자 로그인 링크:(IAM users sign in link:) 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, 폴더를 버킷에 추가하고, 한 개 또는 두 개의 샘플 문서를 각 폴더에 업로드합니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 버킷을 만듭니다.

   단계별 지침은 범용 버킷 생성섹션을 참조하세요.
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.

   폴더 생성에 대한 단계별 지침은 Amazon Simple Storage Service 사용 설명서의 Amazon S3 콘솔에서 폴더를 사용하여 객체 구성>을(를) 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 객체 업로드섹션을 참조하세요.

2단계: IAM 사용자 및 그룹 만들기

이제 IAM 콘솔을 사용하여 AWS 계정에 Alice와 Bob이라는 두 IAM 사용자를 추가합니다. 단계별 지침은 IAM 사용 설명서의 AWS 계정에서 IAM 사용자 생성을 참조하세요.

이름이 Consultants인 관리 그룹도 생성합니다. 그런 다음 두 사용자를 그룹에 추가합니다. 단계별 지침은 IAM 사용자 그룹 생성을 참조하세요.

> **Warning**
>
> 사용자와 그룹을 추가할 때 이러한 사용자에게 권한을 부여하는 정책을 연결하지 마십시오. 처음에는 이러한 사용자가 어떠한 권한도 가지고 있지 않습니다. 다음 섹션에서는 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 참조하세요. 이 시연에서 사용자를 생성할 때는 AWS Management Console 액세스를 선택하고 프로그래밍 방식 액세스 선택을 취소하십시오.

관리자 그룹 만들기에 대한 단계별 지침은 IAM 사용 설명서의 IAM 사용자와 관리자 그룹 처음 만들기를 참조하십시오.

3단계: IAM 사용자에게 권한이 없는지 확인

브라우저를 두 개 사용할 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 링크 제공 참조)를 사용하여 IAM 사용자 자격 증명 중 하나로 AWS Management Console에 로그인합니다.
2. https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.

   액세스가 거부되었다는 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 Console에 로그인하여 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 자격 증명 권한 추가 및 제거를 참조하세요.

   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 링크 제공 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 companybucket을 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 흐름을 약간 더 자세하게 살펴보세요.

버킷 이름을 선택하면 콘솔에서 GET Bucket (List Objects) 요청을 Amazon S3로 전송합니다. 이 요청에는 다음 파라미터가 포함되어 있습니다.

- 빈 문자열이 값으로 포함된 prefix 파라미터입니다.
- delimiter가 값으로 포함된 / 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/> 요소를 포함하는 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. 그러나 이 버킷 예제의 다른 모든 키에는 / 구분 기호가 포함되어 있습니다. Amazon S3에서는 이러한 키를 그룹화하여 고유한 각 접두사 값인 <CommonPrefixes>, Development/ 및 Finance/에 대해 Private/ 요소를 반환합니다. 이는 해당 키의 시작 부분부터 지정된 / 구분 기호의 첫 번째 발생 지점까지의 범위에서 가져온 하위 문자열입니다.

콘솔에서 이 결과를 해석하여 루트 수준 항목을 폴더 3개와 객체 키 하나로 표시합니다.

Bob 또는 Alice가 Development 폴더를 열면 콘솔에서 prefix 및 delimiter 파라미터가 다음 값으로 설정된 GET Bucket(List Objects) 요청이 Amazon S3로 전송됩니다.

- 값 prefix가 있는 Development/ 파라미터입니다.
- "delimiter" 값이 있는 / 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 Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 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. *companybucket* 정책에서 Resource을 버킷의 이름으로 대체해야 합니다.

   단계별 지침은 IAM 사용 설명서의 IAM 정책 편집을 참조하세요. 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 링크 제공 참조)를 사용하여 AWS Management Console에 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 "`delimiter`"가 값으로 있는 `/` 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 참조하세요.

1. AWS Management Console에 로그인하여 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 붙여넣습니다.

```
{
    "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 링크 제공 참조)를 사용하여 AWS Management Console에 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 ListObjects가 있는 /Development 요청이 Amazon S3로 전송됩니다. 접두사 Development 및 구분 기호 /로 사용자에게 객체 목록 확인 권한이 부여되기 때문에 Amazon S3에서 키 접두사 Development/가 지정된 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 Console에 로그인한 후 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 링크 제공 참조)를 사용하여 AWS Management Console에 로그인합니다.

   b. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.

   c. Amazon S3 콘솔에서 Alice가 이제 Development 폴더에서 객체를 추가하고 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 섹션을 참조하세요.
- StringNotLike 조건식은 prefix 파라미터의 값이 지정되고 Development/*가 아닌 경우 요청이 실패하도록 합니다.

이전 섹션의 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 섹션을 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. Private 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 Console에 로그인한 후 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 편집 섹션을 참조하세요. 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 참조하세요.

스토리지에 대한 추가 요금이 부과되지 않도록 하려면 이 연습을 위해 만든 객체와 버킷도 삭제해야 합니다.

관련 리소스

- IAM 사용 설명서의 IAM 정책 관리

Amazon S3의 ID 기반 정책 예시

이 섹션에서는 Amazon S3에 대한 액세스를 제어하는 몇 가지 예시 AWS Identity and Access Management(IAM) ID 기반 정책을 보여 줍니다. 예를 들어 버킷 정책(리소스 기반 정책)은 Amazon S3의 버킷 정책 섹션을 참조하세요. IAM 정책 언어에 대한 정보는 Amazon S3의 정책 및 권한 섹션을 참조하십시오.

다음 정책 예제는 프로그래밍 방식으로 사용하는 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 사용자 정책을 사용하여 버킷에 대한 액세스 제어 단원을 참조하십시오.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

주제

- 버킷 중 하나에 대한 IAM 사용자 액세스 허용
- 버킷의 폴더에 대한 각 IAM 사용자의 액세스 허용
- Amazon S3에 공유 폴더를 보유할 수 있도록 그룹을 허용
- 모든 사용자가 버킷의 일부로 객체를 읽을 수 있도록 허용
- 파트너가 버킷의 특정 부분에 파일을 둘 수 있도록 허용
- 특정 AWS 계정의 Amazon S3 버킷에 대한 액세스 제한
- 조직 단위 내의 Amazon S3 버킷에 대한 액세스 제한
- 조직 내의 Amazon S3 버킷에 대한 액세스 제한
- AWS 계정에 대한 PublicAccessBlock 구성을 검색할 수 있는 권한 부여
- 버킷 생성을 한 리전으로 제한

버킷 중 하나에 대한 IAM 사용자 액세스 허용

이 예제에서는 AWS 계정의 IAM 사용자에게 버킷 중 하나인 *amzn-s3-demo-bucket1*에 대한 액세스 권한을 부여하고 이 사용자에게 객체를 추가, 업데이트, 삭제하도록 허용하려 합니다.

이 정책에서는 `s3:PutObject`, `s3:GetObject` 및 `s3:DeleteObject` 권한을 사용자에게 부여할 뿐만 아니라 `s3:ListAllMyBuckets`, `s3:GetBucketLocation` 및 `s3:ListBucket` 권한도 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 사용하

여 사용자에게 권한을 부여하고 테스트하는 방법에 대한 예는 사용자 정책을 사용하여 버킷에 대한 액세스 제어을 참조하십시오.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action": "s3:ListAllMyBuckets",
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":["s3:ListBucket","s3:GetBucketLocation"],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:GetObject",
            "s3:GetObjectAcl",
            "s3:DeleteObject"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/*"
      }
   ]
}
```

버킷의 폴더에 대한 각 IAM 사용자의 액세스 허용

이 예제에서는 2명의 IAM 사용자인 Mary와 Carlos에게 *amzn-s3-demo-bucket1* 버킷에 대한 액세스 권한을 부여하여 이들이 객체를 추가, 업데이트, 삭제할 수 있게 하려고 합니다. 그러나 각 사용자의 액세스 권한을 버킷의 단일 접두사(폴더)로 제한하려 합니다. 사용자 이름과 일치하는 이름으로 폴더를 만들 수 있습니다.

```
amzn-s3-demo-bucket1
   Mary/
   Carlos/
```

각 사용자에게 각자의 폴더에 대해서만 액세스 권한을 부여하려면 각 사용자에 대한 정책을 작성하고 이를 개별적으로 연결하면 됩니다. 예를 들어, 다음 정책을 사용자 중 Mary에게 연결하여 *amzn-s3-demo-bucket1*/*Mary* 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:::amzn-s3-demo-bucket1/Mary/*"
      }
   ]
}
```

그런 다음 사용자 Carlos에게 유사한 정책을 연결하여 Resource 값의 *Carlos* 폴더를 지정할 수 있습니다.

정책을 개별 사용자에게 연결하는 대신 정책 변수를 사용하는 단일 정책을 작성하여 이 정책을 그룹에 연결할 수도 있습니다. 우선 그룹을 하나 만들어 Mary와 Carlos을 모두 이 그룹에 추가합니다. 다음 정책 예제에서는 *amzn-s3-demo-bucket1*/${aws:username} 폴더의 Amazon S3 권한 집합을 허용합니다. 정책이 평가될 때 정책 변수 ${aws:username}은 요청자의 사용자 이름으로 대체됩니다. 예를 들어, Mary가 요청을 보내 객체를 배치하는 경우 이 작업은 Mary가 객체를 *amzn-s3-demo-bucket1*/Mary 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:::amzn-s3-demo-bucket1/${aws:username}/*"
      }
   ]
}
```

> **Note**
>
> 정책 변수를 사용하는 경우 정책에 명시적으로 버전 2012-10-17을 지정해야 합니다. IAM 정책 언어의 기본 버전인 2008-10-17은 정책 변수를 지원하지 않습니다.

앞서 다룬 정책을 Amazon S3 콘솔에서 테스트하려는 경우 콘솔에서는 다음 정책에서와 같이 추가 권한을 요구합니다. 콘솔에서 이와 같은 정책을 사용하는 방법에 대한 자세한 내용은 사용자 정책을 사용하여 버킷에 대한 액세스 제어 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:::amzn-s3-demo-bucket1",
      "Condition":{
            "StringEquals":{
                    "s3:prefix":[""], "s3:delimiter":["/"]
                          }
                }
    },
    {
      "Sid": "AllowListBucketOfASpecificUserPrefix",
      "Action": "s3:ListBucket",
```

```
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1",
      "Condition":{  "StringLike":{"s3:prefix":["${aws:username}/*"] }
       }
    },
      {
     "Sid": "AllowUserSpecificActionsOnlyInTheSpecificUserPrefix",
        "Effect":"Allow",
        "Action":[
           "s3:PutObject",
           "s3:GetObject",
           "s3:GetObjectVersion",
           "s3:DeleteObject",
           "s3:DeleteObjectVersion"
        ],
        "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/${aws:username}/*"
      }
  ]
}
```

> **Note**
>
> 2012-10-17 버전의 정책에서, 정책 변수는 $로 시작합니다. 이러한 구문상의 변경은 객체 키(객체 이름)에 $가 포함된 경우 잠재적으로 충돌을 야기할 수 있습니다.
> 이 충돌을 방지하려면 ${$} 기호를 사용하여 $ 문자를 지정하십시오. 예를 들어, 객체 키 my$file을 정책에 포함하려면 my${$}file로 지정하면 됩니다.

IAM 사용자 이름은 익숙하고 읽기 쉬운 식별자여야 하되, 고유한 전역 이름일 필요는 없습니다. 예를 들어, Carlos라는 사용자가 조직을 떠난 후 또 다른 Carlos가 조직에 합류하는 경우 새로 온 Carlos는 이전 Carlos의 정보에 액세스할 수도 있습니다.

사용자 이름을 이용하는 대신 IAM 사용자 ID를 기반으로 한 폴더를 만들 수 있습니다. 각 IAM 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:::amzn-s3-demo-bucket1/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 액세스 권한을 각자의 폴더로 제한하려고 하는데, 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 있도록 허용하고 사용자별 폴더를 만들고 데이터를 업로드하는 등의 작업을 수행하는 IAM 정책을 만들 수 있습니다. 웹 아이덴티티 페더레이션에 대한 자세한 내용은 IAM 사용 설명서의 웹 아이덴티티 페더레이션 정보를 참조하십시오.

Amazon S3에 공유 폴더를 보유할 수 있도록 그룹을 허용

다음 정책을 그룹에 연결하면 그룹 내 모두에게 Amazon S3의 *amzn-s3-demo-bucket1*/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:::amzn-s3-demo-bucket1/share/marketing/*"
      }
   ]
}
```

모든 사용자가 버킷의 일부로 객체를 읽을 수 있도록 허용

이 예제에서는 AWS 계정에서 소유한 전체 IAM 사용자를 포함하는 *AllUsers*라는 그룹을 만듭니다. 그런 뒤 그룹에 GetObject 및 GetObjectVersion에 대해서만 *amzn-s3-demo-bucket1*/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:::amzn-s3-demo-bucket1/readonly/*"
      }
   ]
}
```

파트너가 버킷의 특정 부분에 파일을 둘 수 있도록 허용

이 예제에서는 파트너사를 나타내는 *AnyCompany*라는 그룹을 만듭니다. 파트너사에서 액세스 권한을 필요로 하는 사람이나 애플리케이션을 위해 IAM 사용자를 만든 다음 그 사용자를 그룹에 넣습니다.

그런 뒤 그룹에 버킷의 다음 폴더에 대한 PutObject 액세스 권한을 부여하는 정책을 연결합니다.

amzn-s3-demo-bucket1/uploads/anycompany

AnyCompany 그룹이 버킷을 사용하여 다른 작업을 수행하지 못하도록 하려면 AWS 계정의 Amazon S3 리소스에 대한 PutObject를 제외하고 모든 Amazon S3 작업에 대한 권한을 명시적으로 거부하는 문을 추가하면 됩니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"s3:PutObject",
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/uploads/anycompany/*"
      },
      {
         "Effect":"Deny",
         "Action":"s3:*",
         "NotResource":"arn:aws:s3:::amzn-s3-demo-bucket1/uploads/anycompany/*"
      }
   ]
}
```

특정 AWS 계정의 Amazon S3 버킷에 대한 액세스 제한

Amazon S3 보안 주체가 신뢰할 수 있는 AWS 계정 내에 있는 리소스에만 액세스하고 있는지 확인하려면 액세스를 제한하면 됩니다. 예를 들어, 이 자격 증명 기반 IAM 정책에서는 액세스 중인 Amazon S3 리소스가 *222222222222* 계정에 있지 않으면 Deny 효과를 사용하여 Amazon S3 작업에 대한 액세스를 차단합니다. AWS 계정의 IAM 보안 주체가 계정 외부의 Amazon S3 객체에 액세스하지 못하게 하려면 다음 IAM 정책을 연결하십시오.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyS3AccessOutsideMyBoundary",
      "Effect": "Deny",
      "Action": [
        "s3:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "222222222222"
          ]
        }
      }
    }
```

```
  ]
}
```

> **Note**
>
> 이 정책은 액세스 권한을 부여하지 않으므로 기존 IAM 액세스 제어를 대체하지 않습니다. 대신 이 정책은 다른 IAM 정책을 통해 부여된 권한에 관계없이 다른 IAM 권한에 대한 추가 가드레일 역할을 합니다.

정책의 *222222222222* 계정 ID를 본인의 AWS 계정로 대체해야 합니다. 이 제한을 유지하면서 여러 계정에 정책을 적용하려면 계정 ID를 `aws:PrincipalAccount` 조건 키로 바꾸면 됩니다. 이 조건에서는 보안 주체와 리소스가 동일한 계정에 있어야 합니다.

조직 단위 내의 Amazon S3 버킷에 대한 액세스 제한

조직 단위(OU)를 AWS Organizations에 설정한 경우 조직의 특정 부분으로 Amazon S3 버킷 액세스를 제한할 수 있습니다. 이 예제에서는 `aws:ResourceOrgPaths` 키를 사용하여 조직의 OU로 Amazon S3 버킷 액세스를 제한합니다. 이 예제에서 OU ID는 *ou-acroot-exampleou*입니다. 자체 정책에서 이 값을 자체 OU ID로 바꿔야 합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Sid": "AllowS3AccessOutsideMyBoundary",
       "Effect": "Allow",
       "Action": [
         "s3:*"
       ],
       "Resource": "*",
       "Condition": {
         "ForAllValues:StringNotLike": {
           "aws:ResourceOrgPaths": [
             "o-acorg/r-acroot/ou-acroot-exampleou/"
           ]
         }
       }
     }
   ]
 }
```

> **Note**
>
> 이 정책은 어떤 액세스 권한도 부여하지 않습니다. 대신 이 정책은 다른 IAM 권한에 대한 백스톱 역할을 하므로, 보안 주체가 OU 정의 경계 외부의 Amazon S3 객체에 액세스할 수 없습니다.

액세스 중인 Amazon S3 객체가 조직의 *ou-acroot-exampleou* OU에 없는 한 정책은 Amazon S3 작업에 대한 액세스를 거부합니다. IAM 정책 조건은 나열된 OU 경로를 포함하는 데 다중 값 조건 키인 `aws:ResourceOrgPaths`를 필요로 합니다. 이 정책은 `ForAllValues:StringNotLike` 연산자를 사용하여 대소문자를 구분하지 않고 `aws:ResourceOrgPaths`의 값을 나열된 OU와 비교합니다.

조직 내의 Amazon S3 버킷에 대한 액세스 제한

조직 내의 Amazon S3 객체에 대한 액세스를 제한하려면 IAM 정책을 조직의 루트에 연결하여 조직의 모든 계정에 적용합니다. IAM 보안 주체가 이 규칙을 따르도록 요구하려면 서비스 제어 정책(SCP)을 사용하면 됩니다. SCP를 사용하기로 선택한 경우 조직 루트에 정책을 연결하기 전에 SCP를 철저히 테스트해야 합니다.

다음 예제 정책에서 액세스 중인 Amazon S3 객체가 액세스 중인 IAM 보안 주체와 동일한 조직에 있지 않으면 Amazon S3 작업에 대한 액세스가 거부됩니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Sid": "DenyS3AccessOutsideMyBoundary",
       "Effect": "Deny",
       "Action": [
         "s3:*"
       ],
       "Resource": "arn:aws:s3:::*/*",
       "Condition": {
         "StringNotEquals": {
           "aws:ResourceOrgID": "${aws:PrincipalOrgID}"
         }
       }
     }
   ]
 }
```

> **Note**
>
> 이 정책은 어떤 액세스 권한도 부여하지 않습니다. 대신 이 정책은 다른 IAM 권한에 대한 백스톱 역할을 하므로, 보안 주체가 조직 외부의 모든 Amazon S3 객체에 액세스할 수 없습니다. 이 정책은 정책이 적용된 후에 생성된 Amazon S3 리소스에도 적용됩니다.

이 예제의 IAM 정책 조건에서는 서로 동일해지려면 `aws:ResourceOrgID` 및 `aws:PrincipalOrgID`를 필요로 합니다. 이 요구 사항에서는 요청을 수행하는 보안 주체와 액세스 중인 리소스가 동일한 조직에 있어야 합니다.

AWS 계정에 대한 PublicAccessBlock 구성을 검색할 수 있는 권한 부여

다음 ID 기반 정책 예시에서는 사용자에게 `s3:GetAccountPublicAccessBlock` 권한을 부여합니다. 이러한 권한에 대해 `Resource` 값을 `"*"`로 설정합니다. 리소스 ARN에 대한 자세한 내용은 Amazon S3의 정책 리소스 섹션을 참조하세요.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action":[
            "s3:GetAccountPublicAccessBlock"
         ],
         "Resource":[
            "*"
         ]
       }
    ]
}
```

버킷 생성을 한 리전으로 제한

AWS 계정 관리자가 해당 계정 사용자(Dave)에게 남아메리카(상파울루) 리전에서만 버킷을 생성하는 권한을 부여하려 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 리전 목록은 AWS 일반 참조의 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-1`을 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-1`이 아닌 경우 거부됩니다. 명시적 거부는 사용자가 다른 어떤 권한을 갖고 있더라도 사용자가 다른 리전에서 버킷을 생성하는 것을 허용하지 않습니다. 다음 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 CLI에서 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 확인하세요. 버킷 이름과 경로를 적절히 업데이트해야 합니다. `--profile` 파라미터를 사용하여 사용자 자격 증명을 제공해야 합니다. AWS CLI 사용 및 설정에 대한 자세한 내용은 Amazon S3 API 참조의 AWS CLI를 사용하여 Amazon S3에서 개발을 참조하세요.

```
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"}
```

정책을 사용하여 Amazon S3 리소스에 대한 액세스를 관리하는 연습

이 항목에서는 Amazon S3 리소스에 대한 액세스 권한을 부여하는 예제 안내를 소개합니다. 이들 예제는 AWS Management Console을 사용하여 리소스(버킷, 객체, 사용자)를 만들고 그러한 리소스에 권한을 부여합니다. 그런 다음 명령줄 도구를 사용하여 권한을 확인하는 방법을 보여 주므로 코드를 작성할 필요가 없습니다. AWS Command Line Interface(AWS CLI) 및 AWS Tools for Windows PowerShell을 모두 사용한 명령이 제공됩니다.

- 예제 1: 버킷 소유자가 자신의 사용자에게 버킷 권한 부여

  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

  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 액세스 권한을 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 예제에 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 자신이 소유하지 않는 객체에 대한 권한 부여

    이 연습에서 먼저 버킷 소유자는 객체 소유자로부터 권한을 받습니다. 그런 다음 버킷 소유자는 이러한 권한을 자신이 소유한 계정의 사용자에게 위임합니다.

  - 예시 4: 버킷 소유자가 자신의 소유가 아닌 객체에 크로스 계정 권한 부여

    크로스 계정 위임은 지원되지 않기 때문에 버킷 소유자는 객체 소유자로부터 권한을 받은 후 다른 AWS 계정에 권한을 위임할 수 없습니다(권한 위임 참조). 대신에 버킷 소유자는 특정 작업(예: 객

체 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 사용 설정하는 방법을 보여 줍니다.

예제 안내를 실습하기 전에 알아 두어야 할 사항

이들 예제는 AWS Management Console을 사용하여 리소스를 만들고 권한을 부여합니다. 예시에서는 권한을 테스트하기 위해 명령줄 도구인 AWS CLI 및 AWS Tools for Windows PowerShell을 사용하므로 직접 코드를 작성할 필요가 없습니다. 권한을 테스트하려면 이러한 도구 중 하나를 설정해야 합니다. 자세한 내용은 연습을 위한 도구 설정 섹션을 참조하세요.

또한, 리소스를 만들 경우, 이 예시는 AWS 계정의 루트 사용자 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 사용자 자격 증명을 사용하여 요청하지 않을 것을 권장합니다. 대신 IAM 사용자 또는 역할을 만들고 모든 권한을 부여한 후 자격 증명을 사용하여 요청을 수행합니다. 이를 관리 사용자 또는 역할이라고 합니다. 자세한 내용은 AWS 일반 참조의 AWS 계정 루트 사용자 자격 증명 및 IAM 인증 정보 및 IAM 사용 설명서의 IAM 모범 사례를 참조하십시오.

이 섹션의 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 Console에 로그인하려면 IAM 사용자 로그인 URL을 사용해야 합니다. IAM 콘솔은 AWS 계정에 이 URL을 제공합니다. 이 항목에서는 URL을 받는 방법을 보여 줍니다.

연습을 위한 도구 설정

이 소개 예제는 (정책을 사용하여 Amazon S3 리소스에 대한 액세스를 관리하는 연습 참조) AWS Management Console을 사용하여 리소스를 만들고 권한을 부여합니다. 예시에서는 권한을 테스트하기 위해 명령줄 도구인 AWS Command Line Interface(AWS CLI) 및 AWS Tools for Windows PowerShell을 사용하므로 직접 코드를 작성할 필요가 없습니다. 권한을 테스트하려면 이러한 도구 중 하나를 설정해야 합니다.

AWS CLI를 설정하려면

1. AWS CLI를 다운로드하고 구성합니다. 관련 지침은 AWS Command Line Interface 사용자 가이드에서 다음 주제를 참조하십시오.

   최신 AWS Command Line Interface 버전의 설치 또는 업데이트

   AWS Command Line Interface 시작

2. 기본 프로필을 설정합니다.

   AWS CLI 구성 파일에 사용자 자격 증명을 저장합니다. AWS 계정 자격 증명을 사용하여 구성 파일에 기본 프로파일을 만듭니다. AWS CLI 구성 파일을 찾고 편집하는 방법에 대한 지침은 구성 및 자격 증명 파일 설정을 참조하세요.

   ```
   [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 명령을 사용해 봅니다.

     ```
     aws help
     ```

   - aws s3 ls 명령을 사용하여 구성된 계정의 버킷 목록을 가져옵니다.

     ```
     aws s3 ls
     ```

연습을 진행하면서 다음 예시와 같이 사용자를 만들고, 프로파일을 생성하여 구성 파일에 사용자 자격 증명을 저장합니다. 이러한 프로파일의 이름은 AccountAadmin 및 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` 프로파일을 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 CLI에서는 환경 변수에 설정된 프로파일을 기본 프로파일로 사용합니다.

```
$ export AWS_DEFAULT_PROFILE=AccountAadmin
```

AWS Tools for Windows PowerShell 설정

1. AWS Tools for Windows PowerShell를 다운로드하고 구성합니다. 지침은 AWS Tools for PowerShell 사용 설명서의 AWS Tools for Windows PowerShell 설치를 참조하세요.

   > **Note**
   >
   > AWS Tools for Windows PowerShell 모듈을 로드하려면 PowerShell 스크립트 실행을 활성화해야 합니다. 자세한 내용은 AWS Tools for PowerShell 사용 설명서의 스크립트 실행 활성화를 참조하세요.

2. 이 연습에서는 `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` 명령을 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 AWS Tools for PowerShell Cmdlet 참조를 참조하세요.

이제 연습 준비가 되었습니다. 각 섹션 시작 부분에 있는 링크를 따라 가세요.

예제 1: 버킷 소유자가 자신의 사용자에게 버킷 권한 부여

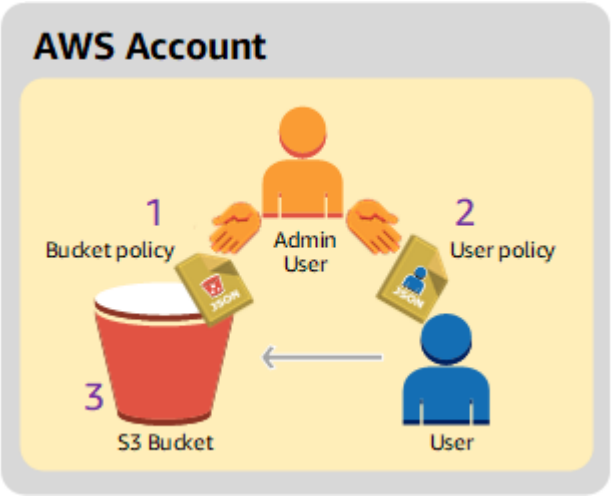
> **Important**
>
> IAM 역할에 권한을 부여하는 것이 개별 사용자에게 권한을 부여하는 것보다 더 좋습니다. IAM 역할에 권한을 부여하는 방법에 대한 자세한 내용은 크로스 계정 권한과 IAM 역할 사용에 대한 이해 섹션을 참조하세요.

주제

- 연습 준비
- 1단계: 계정 A에서 리소스를 생성하고 권한 부여
- 3단계: 권한 테스트

이 연습에서는 AWS 계정이 버킷을 소유하고 있으며 이 계정에는 IAM 사용자가 포함되어 있습니다. 기본적으로 사용자에게는 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 사용자 자격 증명을 사용하는 대신 관리자 사용자를 만듭니다(관리자 사용자를 사용하여 리소스를 만들고 권한을 부여하는 것에 대한 소개 참조). AWS 계정 및 관리자 사용자는 다음 테이블과 같이 지칭합니다.

계정 ID	계정 이름	계정의 관리자 사용자
1111-1111-1111	계정 A	AccountAadmin

> **Note**
>
> 이 예제의 관리자 사용자는 AccountAdmin이 아니라 계정 A라는 AccountAadmin입니다.

사용자를 만들고 권한을 부여하는 모든 작업은 AWS Management Console에서 수행됩니다. 권한을 확인하기 위해 본 시연에서는 명령줄 도구, AWS Command Line Interface(AWS CLI) 및 AWS Tools for Windows PowerShell을 사용합니다. 따라서 사용자가 코드를 작성할 필요가 없습니다.

연습 준비

1. AWS 계정이 있으며 이 계정에 관리자 권한이 있는 사용자가 있는지 확인합니다.

   a. 필요한 경우 AWS 계정에 가입합니다. 이 계정을 계정 A라고 합니다.

      i. https://aws.amazon.com/s3로 이동하여 AWS 계정 생성을 선택합니다.

      ii. 화면에 표시되는 지시 사항을 따릅니다.

      계정이 활성화되고 사용 가능한 상태가 되면 AWS에서 사용자에게 이메일로 알립니다.

   b. 계정 A에서 관리자 사용자인 **AccountAadmin**을 생성합니다. 계정 A의 자격 증명을 사용하여 IAM 콘솔에 로그인하고 다음을 수행합니다.

      i. **AccountAadmin** 사용자를 생성하고 사용자 보안 자격 증명을 적어 둡니다.

      지침은 IAM 사용 설명서의 AWS 계정에서 IAM 사용자 생성을 참조하세요.

      ii. 모든 권한을 제공하는 사용자 정책을 연결하여 AccountAadmin에 관리자 권한을 부여합니다.

      지침은 IAM 사용 설명서에서 IAM 정책 관리를 참조하십시오.

iii. AccountAadmin의 IAM 사용자 로그인 URL을 적어 둡니다. AWS Management Console에 로그인할 때 이 URL을 사용해야 합니다. 로그인 URL을 찾을 수 있는 곳에 대한 자세한 내용은 IAM 사용 설명서의 AWS Management Console에 IAM 사용자로 로그인을 참조하세요. 각 계정의 URL을 적어 둡니다.

2. AWS CLI 또는 AWS Tools for Windows PowerShell을 설정합니다. 관리자 사용자 자격 증명은 다음과 같이 저장합니다.

   - AWS CLI를 사용하는 경우 구성 파일에 프로파일 `AccountAadmin`를 생성합니다.
   - AWS Tools for Windows PowerShell을 사용하는 경우 세션의 자격 증명을 `AccountAadmin`으로 저장해야 합니다.

   지침은 연습을 위한 도구 설정 섹션을 참조하세요.

1단계: 계정 A에서 리소스를 생성하고 권한 부여

계정 A의 사용자 `AccountAadmin`에 대한 자격 증명과 특별 IAM 사용자 로그인 URL을 사용하여 AWS Management Console에 로그인하고 다음을 수행합니다.

1. 버킷의 리소스 및 IAM 사용자를 생성합니다.

   a. Amazon S3 콘솔에서 버킷을 생성합니다. 버킷을 생성한 AWS 리전을 적어 둡니다. 지침은 범용 버킷 생성 섹션을 참조하세요.

   b. IAM 콘솔에서 다음을 수행합니다.

      i. Dave라는 사용자를 생성합니다.

         단계별 지침은 IAM 사용 설명서의 IAM 사용자 생성(콘솔)을 참조하세요.

      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. 권한을 부여합니다.

   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 *awsexamplebucket1*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 단원을 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:::awsexamplebucket1"
         ]
      },
      {
         "Sid": "statement2",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
         },
         "Action": [
             "s3:GetObject"
         ],
         "Resource": [
            "arn:aws:s3:::awsexamplebucket1/*"
         ]
```

```
      }
   ]
}
```

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:::awsexamplebucket1/*"
         ]
      }
   ]
}
```

지침은 IAM 사용 설명서의 IAM 정책 관리를 참조하세요. 계정 A의 자격 증명을 사용하여 콘솔에 로그인해야 합니다.

3단계: 권한 테스트

Dave의 자격 증명을 사용하여 권한이 작동하는지 확인하십시오. 다음 두 가지 절차 중 하나를 사용할 수 있습니다.

AWS CLI를 사용하여 권한을 테스트합니다.

1. 다음 AWS CLI 프로파일을 추가하여 `UserDaveAccountA` 구성 파일을 업데이트합니다. 자세한 내용은 연습을 위한 도구 설정 섹션을 참조하세요.

```
[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 awsexamplebucket1 --key HappyFace.jpg --
   body HappyFace.jpg --profile UserDaveAccountA
   ```

   다음 AWS CLI 명령을 실행하여 객체를 가져옵니다.

   ```
   aws s3api get-object --bucket awsexamplebucket1 --key HappyFace.jpg OutputFile.jpg
    --profile UserDaveAccountA
   ```

AWS Tools for Windows PowerShell을 사용하여 권한을 테스트합니다.

1. Dave의 자격 증명을 `AccountADave`로 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 AWS Tools for Windows PowerShell `Write-S3Object` 명령을 통해 샘플 객체를 업로드합니다.

   ```
   Write-S3Object -bucketname awsexamplebucket1 -key HappyFace.jpg -file HappyFace.jpg
    -StoredCredentials AccountADave
   ```

   이전에 업로드한 객체를 다운로드합니다.

   ```
   Read-S3Object -bucketname awsexamplebucket1 -key HappyFace.jpg -file Output.jpg -
   StoredCredentials AccountADave
   ```

예제 2: 버킷 소유자가 교차 계정 버킷 권한 부여

> **Important**
>
> IAM 역할에 권한을 부여하는 것이 개별 사용자에게 권한을 부여하는 것보다 더 좋습니다. 이 작업을 수행하는 방법은 크로스 계정 권한과 IAM 역할 사용에 대한 이해 단원을 참조하십시오.

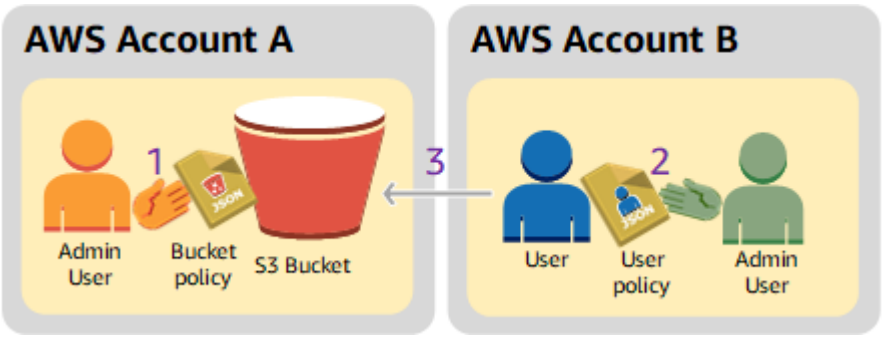
주제

- 연습 준비
- 1단계: 계정 A 태스크 수행
- 2단계: 계정 B 태스크 수행
- 3단계: (선택 사항) 명시적 거부 시도
- 4단계: 정리

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 권한을 받아야 합니다. 이는 계정 B가 계정 A로부터 권한을 받지 못했더라도 마찬가지입니다. 사용자가 두 리소스 소유자 모두와 상위 계정으로부터 권한을 받은 이상 해당 사용자는 리소스에 액세스할 수 있습니다.

다음은 연습 단계 의 요약입니다.



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 참조)에 따라 본 연습에서는 루트 사용자 자격 증명을 사용하지 않습니다. 대신 각 계정에서 관리자 사용자를 만들고 리소스를 만들고 권한을 부여할 때 해당 자격 증명을 사용합니다.

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 Console에서 수행됩니다. 권한을 확인하기 위해 본 시연에서는 명령줄 도구, AWS Command Line Interface(CLI) 및 AWS Tools for Windows PowerShell을 사용합니다. 따라서 사용자가 코드를 작성할 필요가 없습니다.

연습 준비

1. AWS 계정이 두 개 있으며 각 계정에 이전 섹션의 표에 나와 있는 것과 같이 관리자 사용자가 한 명 있는지 확인합니다.

a. 필요한 경우 AWS 계정에 가입합니다.

b. 계정 A의 자격 증명을 사용하여 IAM 콘솔에 로그인하여 관리자 사용자를 만듭니다.

   i. **AccountAadmin** 사용자를 생성하고 보안 자격 증명을 적어 둡니다. 지침은 IAM 사용 설명서의 AWS 계정에서 IAM 사용자 생성을 참조하십시오.

   ii. 모든 권한을 제공하는 사용자 정책을 연결하여 AccountAadmin에 관리자 권한을 부여합니다. 지침은 IAM 사용 설명서의 정책 작업 섹션을 참조하십시오.

c. IAM 콘솔의 대시보드에서 IAM 사용자 로그인 URL을 적어 둡니다. 계정에 속한 모든 사용자가 AWS Management Console에 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 **AccountBadmin**을 만듭니다.

2. AWS Command Line Interface(AWS CLI) 또는 AWS Tools for Windows PowerShell을 설정합니다. 관리자 사용자 자격 증명은 다음과 같이 저장합니다.

   - AWS CLI를 사용하는 경우 구성 파일에 `AccountAadmin` 및 `AccountBadmin`의 프로파일 2개를 생성합니다.
   - AWS Tools for Windows PowerShell을 사용하는 경우 세션의 자격 증명을 `AccountAadmin` 및 `AccountBadmin`으로 저장해야 합니다.

   지침은 연습을 위한 도구 설정 섹션을 참조하세요.

3. 프로파일이라고도 하는 관리자 사용자 자격 증명을 저장합니다. 입력하는 각 명령에 대해 자격 증명을 지정하는 대신 프로파일 이름을 사용할 수 있습니다. 자세한 내용은 연습을 위한 도구 설정 섹션을 참조하세요.

   a. 두 계정의 AWS CLI 자격 증명 파일에서 각 관리자 사용자, `AccountAadmin` 및 `AccountBadmin`에 대해 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. AWS Tools for Windows PowerShell을 사용 중이라면 다음 명령을 실행합니다.

```
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 태스크 수행

1.1단계: AWS Management Console에 로그인

우선 계정 A의 IAM 사용자 로그인 URL을 사용해 AccountAadmin 사용자로 AWS Management Console에 로그인합니다. 이 사용자가 버킷을 만들고 버킷에 정책을 연결합니다.

1.2단계: 버킷 만들기

1. Amazon S3 콘솔에서 버킷을 생성합니다. 이 연습에서는 버킷이 미국 동부(버지니아 북부) AWS 리전에서 생성되었으며 이름이 *amzn-s3-demo-bucket*이라고 가정합니다.

   지침은 범용 버킷 생성 섹션을 참조하세요.
2. 버킷에 샘플 객체를 업로드합니다.

   지침은 2단계: 버킷에 객체 업로드을 참조하십시오.

1.3단계: 계정 B에 교차 계정 권한을 부여하기 위해 버킷 정책 연결

버킷 정책을 사용하면 계정 B에 `s3:GetLifecycleConfiguration` 및 `s3:ListBucket` 권한이 부여됩니다. AccountAadmin 사용자 자격 증명을 사용하여 여전히 콘솔에 로그인된 상태라고 가정합니다.

1. *amzn-s3-demo-bucket*에 다음 버킷 정책을 연결합니다. 이 정책은 계정 B에게 `s3:GetLifecycleConfiguration` 및 `s3:ListBucket` 작업에 대한 권한을 부여합니다.

   지침은 Amazon S3 콘솔을 사용하여 버킷 정책 추가 섹션을 참조하세요.

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

```
      {
         "Sid": "Example permissions",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountB-ID:root"
         },
         "Action": [
            "s3:GetLifecycleConfiguration",
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket"
         ]
      }
   ]
}
```

2. 계정 B와 해당 관리자 사용자가 작업을 수행할 수 있는지 확인합니다.

   - AWS CLI를 사용하여 확인

     ```
     aws s3 ls s3://amzn-s3-demo-bucket --profile AccountBadmin
     aws s3api get-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket --
     profile AccountBadmin
     ```

   - AWS Tools for Windows PowerShell을 사용하여 확인

     ```
     get-s3object -BucketName amzn-s3-demo-bucket -StoredCredentials AccountBadmin
     get-s3bucketlifecycleconfiguration -BucketName amzn-s3-demo-bucket -
     StoredCredentials AccountBadmin
     ```

2단계: 계정 B 태스크 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 Console에 로그인합니다.

2.2단계: 계정 B에서 사용자 Dave 만들기

IAM 콘솔에서 사용자 **Dave**를 만듭니다.

지침은 IAM 사용 설명서에서 IAM 사용자 생성(콘솔)을 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 가정합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Example",
         "Effect": "Allow",
         "Action": [
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket"
         ]
      }
   ]
}
```

지침은 IAM 사용 설명서에서 IAM 정책 관리를 참조하십시오.

2.4단계: 권한 테스트

이제 계정 B에 속한 Dave는 계정 A가 소유하는 *amzn-s3-demo-bucket*의 콘텐츠를 나열할 수 있습니다. 다음 절차 중 하나를 사용하여 권한을 확인할 수 있습니다.

AWS CLI를 사용하여 권한 테스트

1. AWS CLI 구성 파일에 UserDave 프로파일을 추가합니다. 구성 파일에 대한 자세한 내용은 연습을 위한 도구 설정을 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 *amzn-s3-demo-bucket*에서 객체 목록을 가져올 수 있는지 확인합니다. 명령에 UserDave 프로파일이 지정된 것을 확인합니다.

```
aws s3 ls s3://amzn-s3-demo-bucket --profile UserDave
```

Dave에게는 다른 권한이 없습니다. 따라서 다른 작업(예: 다음 get-bucket-lifecycle 구성)을 시도하면 Amazon S3는 권한 거부를 반환합니다.

```
aws s3api get-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket --profile
 UserDave
```

AWS Tools for Windows PowerShell을 사용하여 권한 테스트

1. Dave의 자격 증명을 AccountBDave로 저장합니다.

```
set-awscredentials -AccessKey AccessKeyID -SecretKey SecretAccessKey -storeas
 AccountBDave
```

2. List Bucket 명령을 시도합니다.

```
get-s3object -BucketName amzn-s3-demo-bucket -StoredCredentials AccountBDave
```

Dave에게는 다른 권한이 없습니다. 따라서 다른 작업(예: 다음 get-s3bucketlifecycleconfiguration)을 시도할 경우 Amazon S3는 권한 거부를 반환합니다.

```
get-s3bucketlifecycleconfiguration -BucketName amzn-s3-demo-bucket -
StoredCredentials AccountBDave
```

3단계: (선택 사항) 명시적 거부 시도

액세스 제어 목록 (ACL), 버킷 정책 또는 사용자 정책을 사용하여 권한을 부여할 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 거부합니다. 또한 이 정책은 s3:ListBucket 권한을 부여합니다. 그러나 명시적 거부가 우선 적용되어 계정 B 또는 계정 B에 속한 사용자가 *amzn-s3-demo-bucket*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:GetLifecycleConfiguration",
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket"
         ]
      },
      {
         "Sid": "Deny permission",
         "Effect": "Deny",
         "Principal": {
            "AWS": "arn:aws:iam::AccountB-ID:root"
         },
         "Action": [
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket"
         ]
      }
   ]
}
```

2. 이제 AccountBadmin 자격 증명을 사용하여 버킷 목록을 가져오려고 하면 액세스가 거부됩니다.

   - AWS CLI를 사용하여 다음 명령을 실행합니다.

     ```
     aws s3 ls s3://amzn-s3-demo-bucket --profile AccountBadmin
     ```

   - AWS Tools for Windows PowerShell을 사용하여 다음 명령을 실행합니다.

```
get-s3object -BucketName amzn-s3-demo-bucket -StoredCredentials AccountBDave
```

4단계: 정리

1. 테스트를 완료한 후, 다음과 같이 수행해 정리합니다.
   - 계정 A의 자격 증명을 사용하여 AWS Management Console(AWS Management Console)에 로그인하고 다음을 수행합니다.
     - Amazon S3 콘솔에서 *amzn-s3-demo-bucket*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 제거합니다.
2. 계정 B의 자격 증명을 사용하여 IAM 콘솔에 로그인합니다. `AccountBadmin` 사용자를 삭제합니다. 단계별 지침은 IAM 사용 설명서의 IAM 사용자 삭제를 참조하세요.

예제 3: 버킷 소유자가 자신이 소유하지 않는 객체에 대한 권한 부여

> **Important**
>
> IAM 역할에 권한을 부여하는 것이 개별 사용자에게 권한을 부여하는 것보다 더 좋습니다. 이 작업을 수행하는 방법은 크로스 계정 권한과 IAM 역할 사용에 대한 이해 단원을 참조하십시오.

주제

- 0단계: 시연 준비
- 1단계: 계정 A 태스크 수행
- 2단계: 계정 B 태스크 수행
- 3단계: 권한 테스트
- 4단계: 정리

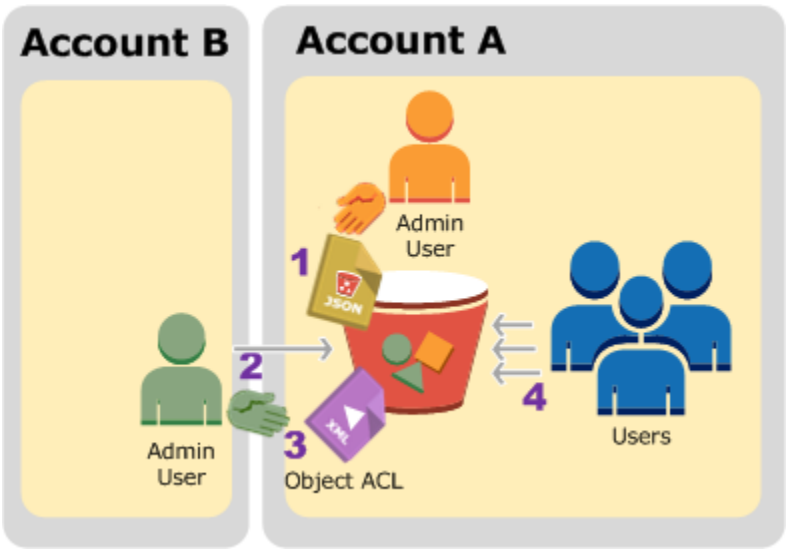
이 예시의 시나리오에서는 버킷 소유자가 객체 액세스를 위해 권한을 부여하고자 하지만 버킷 소유자가 버킷의 일부 객체를 소유하지 않습니다. 이 예제에서 버킷 소유자는 자신의 계정에 속한 사용자에게 권한을 부여하려고 합니다.

버킷 소유자는 객체를 업로드하기 위해 다른 AWS 계정을 사용할 수 있습니다. 기본적으로 버킷 소유자는 다른 AWS 계정이 버킷에 작성된 객체를 소유하지 않습니다. 객체를 S3 버킷에 작성하는 계정이 객체를 소유합니다. 버킷 소유자가 버킷의 객체를 소유하지 않은 경우 객체 소유자가 먼저 객체 액세스 제어 목록(ACL)을 사용하여 버킷 소유자에게 권한을 부여해야 합니다. 그러면 버킷 소유자는 자신이 소유하지 않은 객체에 대한 권한을 부여할 수 있습니다. 자세한 내용은 Amazon S3 버킷 및 객체 소유권 섹션을 참조하세요.

버킷 소유자가 버킷의 S3 객체 소유권에 대해 버킷 소유자 시행 설정을 적용하는 경우 버킷 소유자는 다른 AWS 계정이 작성한 객체를 포함하여 버킷의 모든 객체를 소유하게 됩니다. 이렇게 하면 버킷 소유자가 객체를 소유하지 않은 문제가 해결됩니다. 그런 다음 자신의 계정에 있는 사용자나 다른 AWS 계정에 권한을 위임할 수 있습니다.

> **Note**
>
> S3 객체 소유권은 버킷에 업로드되는 객체의 소유권을 제어하고 ACL을 비활성화 또는 활성화하는 데 사용할 수 있는 Amazon S3 버킷 수준 설정입니다. 기본적으로 객체 소유권은 버킷 소유자 적용 설정으로 설정되며 모든 ACL이 비활성화되어 있습니다. ACL이 비활성화되면 버킷 소유자는 버킷의 모든 객체를 소유하고 액세스 관리 정책을 사용하여 객체에 대한 액세스를 독점적으로 관리합니다.
>
> Amazon S3의 최신 사용 사례 대부분은 더 이상 ACL을 사용할 필요가 없습니다. 각 객체에 대해 액세스를 개별적으로 제어할 필요가 있는 드문 상황을 제외하고는 ACL을 비활성화한 채로 두는 것이 좋습니다. ACL을 비활성화하면 누가 객체를 버킷에 업로드했는지에 관계없이 정책을 사용하여 버킷의 모든 객체에 대한 액세스를 제어할 수 있습니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

이 예에서는 버킷 소유자가 객체 소유권에 대해 버킷 소유자 시행 설정을 적용하지 않았다고 가정합니다. 버킷 소유자가 자신의 계정에 속한 사용자에게 권한을 위임합니다. 다음은 연습 단계 의 요약입니다.



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 권장 지침에 따라 이 연습에서는 계정 루트 사용자 자격 증명을 사용하지 않습니다. 자세한 내용은 관리자 사용자를 사용하여 리소스를 만들고 권한을 부여하는 것에 대한 소개 섹션을 참조하세요. 대신 사용자가 각 계정에서 관리자를 생성한 후 해당 자격 증명을 사용하여 리소스를 생성하고 해당 리소스에 대한 권한을 부여합니다.

AWS 계정 ID	계정 이름	계정의 관리자
1111-1111-1111	계정 A	AccountAadmin
2222-2222-2222	계정 B	AccountBadmin

사용자를 만들고 권한을 부여하는 모든 작업은 AWS Management Console에서 수행됩니다. 권한을 확인하기 위해 본 시연에서는 명령줄 도구, AWS Command Line Interface(AWS CLI) 및 AWS Tools for Windows PowerShell을 사용합니다. 따라서 사용자가 코드를 작성할 필요가 없습니다.

0단계: 시연 준비

1. AWS 계정이 두 개 있으며 각 계정에 이전 섹션의 표에 나와 있는 것과 같이 관리자가 한 명 있는지 확인합니다.

   a. 필요한 경우 AWS 계정에 가입합니다.

   b. 계정 A 자격 증명을 사용하여 IAM 콘솔에 로그인하고 다음을 수행하여 관리자 사용자를 만듭니다.

      - **AccountAadmin** 사용자를 생성하고 사용자 보안 자격 증명을 적어 둡니다. 사용자 추가에 대한 자세한 내용은 IAM 사용 설명서의 AWS 계정에서 IAM 사용자 생성을 참조하십시오.
      - 모든 액세스 권한을 제공하는 사용자 정책을 연결하여 AccountAadmin에 관리자 권한을 부여합니다. 지침은 IAM 사용 설명서에서 IAM 정책 관리를 참조하십시오.
      - IAM 콘솔 대시보드에서 IAM 사용자 로그인 URL을 적어 둡니다. 해당 계정 사용자는 AWS Management Console에 로그인할 때 이 URL을 사용해야 합니다. 자세한 내용은 IAM 사용 설명서에서 사용자의 계정 로그인 방법을 참조하십시오.

   c. 계정 B의 자격 증명을 사용하여 이전 단계를 반복하고 관리자 사용자인 **AccountBadmin**을 만듭니다.

2. AWS CLI 또는 Tools for Windows PowerShell을 설정합니다. 관리자 자격 증명을 다음과 같이 저장해야 합니다.

   - AWS CLI를 사용하는 경우 구성 파일에 `AccountAadmin` 및 `AccountBadmin`의 프로파일 2개를 생성합니다.
   - Tools for Windows PowerShell을 사용하는 경우 세션의 자격 증명을 `AccountAadmin` 및 `AccountBadmin`으로 저장해야 합니다.

   지침은 연습을 위한 도구 설정 섹션을 참조하세요.

1단계: 계정 A 태스크 수행

계정 A에 대해 다음 단계를 수행합니다.

1.1단계: 콘솔에 로그인

계정 A의 IAM 사용자 로그인 URL을 사용하여 **AccountAadmin** 사용자로 AWS Management Console에 로그인합니다. 이 사용자가 버킷을 만들고 버킷에 정책을 연결합니다.

1.2단계: 버킷 및 사용자를 생성하고 사용자 권한을 부여하는 버킷 정책 추가

1. Amazon S3 콘솔에서 버킷을 생성합니다. 이 연습에서는 버킷이 미국 동부(버지니아 북부) AWS 리전에서 생성되었으며 이름이 *amzn-s3-demo-bucket1*이라고 가정합니다.

   지침은 범용 버킷 생성 섹션을 참조하세요.
2. IAM 콘솔에서 사용자 **Dave**를 만듭니다.

   단계별 지침은 IAM 사용 설명서의 IAM 사용자 생성(콘솔)을 참조하세요.
3. 사용자 Dave의 자격 증명을 적어 둡니다.
4. Amazon S3 콘솔에서 *amzn-s3-demo-bucket1* 버킷에 다음 버킷 정책을 연결합니다. 지침은 Amazon S3 콘솔을 사용하여 버킷 정책 추가 섹션을 참조하세요. 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:::amzn-s3-demo-bucket1/*",
            "arn:aws:s3:::amzn-s3-demo-bucket1"
         ]
      },
      {
```

```
            "Sid": "Statement3",
            "Effect": "Allow",
            "Principal": {
               "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
            },
            "Action": [
               "s3:GetObject"
            ],
            "Resource": [
               "arn:aws:s3:::amzn-s3-demo-bucket1/*"
            ]
         }
      ]
}
```

2단계: 계정 B 태스크 수행

이제 계정 B가 계정 A의 버킷에 대한 작업을 수행할 수 있는 권한을 가지므로 계정 B의 관리자가 다음을 수행합니다.

- 계정 A의 버킷에 객체 업로드
- 객체 ACL에서 버킷 소유자인 계정 A에 전체 제어 권한을 부여하는 권한 부여를 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 amzn-s3-demo-bucket1 --key HappyFace.jpg --body
    HappyFace.jpg --profile AccountBadmin
   ```

2. 객체 ACL에서 버킷 소유자에게 객체에 대한 모든 권한을 부여하는 허가를 추가합니다. 표준 사용자 ID를 찾는 방법에 대한 자세한 내용은 AWS 계정 관리 참조 안내서의 AWS 계정의 표준 사용자 ID 찾기를 참조하세요.

   ```
   aws s3api put-object-acl --bucket amzn-s3-demo-bucket1 --key HappyFace.jpg --grant-
   full-control id="AccountA-CanonicalUserID" --profile AccountBadmin
   ```

Tools for Windows PowerShell 사용

1. `Write-S3Object` 명령을 사용하여 객체를 업로드합니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket1 -key HappyFace.jpg -file
 HappyFace.jpg -StoredCredentials AccountBadmin
```

2. 객체 ACL에서 버킷 소유자에게 객체에 대한 모든 권한을 부여하는 허가를 추가합니다.

```
Set-S3ACL -BucketName amzn-s3-demo-bucket1 -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 확인합니다.

AWS CLI 사용

1. AWS CLI 구성 파일에 사용자 Dave의 자격 증명을 추가하고 새로운 프로파일인 `UserDaveAccountA`를 만듭니다. 자세한 내용은 연습을 위한 도구 설정 섹션을 참조하세요.

```
[profile UserDaveAccountA]
aws_access_key_id = access-key
aws_secret_access_key = secret-access-key
region = us-east-1
```

2. `get-object` CLI 명령을 실행하여 `HappyFace.jpg`를 다운로드하고 로컬에 저장합니다. 사용자 Dave의 자격 증명은 `--profile` 파라미터를 추가하여 제공합니다.

```
aws s3api get-object --bucket amzn-s3-demo-bucket1 --key
 HappyFace.jpg Outputfile.jpg --profile UserDaveAccountA
```

Tools for Windows PowerShell 사용

1. 사용자 Dave의 AWS 자격 증명을 `UserDaveAccountA`로 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 amzn-s3-demo-bucket1 -Key HappyFace.jpg -file
 HappyFace.jpg  -StoredCredentials UserDaveAccountA
```

4단계: 정리

1. 테스트를 완료한 후, 다음과 같이 수행해 정리합니다.

   - 계정 A의 자격 증명을 사용하여 AWS Management Console에 로그인하고 다음을 수행합니다.

     - Amazon S3 콘솔에서 *amzn-s3-demo-bucket1*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 제거합니다. 단계별 지침은 IAM 사용 설명서의 IAM 사용자 삭제를 참조하세요.

2. 계정 B의 자격 증명을 사용하여 AWS Management Console에 로그인합니다. IAM 콘솔에서 AccountBadmin 사용자를 삭제합니다.

예시 4: 버킷 소유자가 자신의 소유가 아닌 객체에 크로스 계정 권한 부여

주제

- 크로스 계정 권한과 IAM 역할 사용에 대한 이해
- 0단계: 시연 준비
- 1단계: 계정 A 작업 수행
- 2단계: 계정 B 태스크 수행
- 3단계: 계정 C 작업 수행
- 4단계: 정리
- 관련 리소스

이 예제 시나리오에서 사용자가 한 버킷을 소유하고 있고 다른 AWS 계정에 객체 업로드를 허용합니다. 버킷의 S3 객체 소유권에 대해 버킷 소유자 시행 설정을 적용하면 다른 AWS 계정이 작성한 객체를 포함하여 버킷의 모든 객체를 소유하게 됩니다. 이렇게 하면 버킷 소유자가 객체를 소유하지 않은 문제가 해결됩니다. 그런 다음 자신의 계정에 있는 사용자나 다른 AWS 계정에 권한을 위임할 수 있습니다. S3 객체 소유권에 대해 버킷 소유자 시행 설정이 사용되지 않는다고 가정합니다. 즉, 소유자의 버킷에 다른 AWS 계정이 소유한 객체를 둘 수 있습니다.

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. 버킷 소유자가 자신의 소유가 아닌 객체에 대한 권한을 부여하려면, 우선 객체 소유자가 버킷 소유자에게 권한을 부여해야 합니다. 객체를 생성한 AWS 계정이 객체 소유자입니다. 그러면 버킷 소유자가 해당 권한을 위임할 수 있습니다.
- 버킷 소유자 계정은 자신의 계정에 속한 사용자에게 권한을 위임할 수 있습니다(예제 3: 버킷 소유자가 자신이 소유하지 않는 객체에 대한 권한 부여 참조). 하지만 크로스 계정 위임이 지원되지 않기 때문에 버킷 소유자 계정은 다른 AWS 계정에게 권한을 위임할 수 없습니다.

이 시나리오에서 버킷 소유자는 객체 액세스 권한을 가진 AWS Identity and Access Management(IAM) 역할을 생성합니다. 그러면 버킷 소유자는 다른 AWS 계정에 이 역할을 맡을 권한을 부여하여 버킷의 객체에 일시적으로 액세스 하도록 허용할 수 있습니다.

> **Note**
>
> S3 객체 소유권은 버킷에 업로드되는 객체의 소유권을 제어하고 ACL을 비활성화 또는 활성화하는 데 사용할 수 있는 Amazon S3 버킷 수준 설정입니다. 기본적으로 객체 소유권은 버킷 소유자 적용 설정으로 설정되며 모든 ACL이 비활성화되어 있습니다. ACL이 비활성화되면 버킷 소유자는 버킷의 모든 객체를 소유하고 액세스 관리 정책을 사용하여 객체에 대한 액세스를 독점적으로 관리합니다.
>
> Amazon S3의 최신 사용 사례 대부분은 더 이상 ACL을 사용할 필요가 없습니다. 각 객체에 대해 액세스를 개별적으로 제어할 필요가 있는 드문 상황을 제외하고는 ACL을 비활성화한 채로 두는 것이 좋습니다. ACL을 비활성화하면 누가 객체를 버킷에 업로드했는지에 관계없이 정책을 사용하여 버킷의 모든 객체에 대한 액세스를 제어할 수 있습니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

크로스 계정 권한과 IAM 역할 사용에 대한 이해

IAM 역할은 사용자 리소스 액세스를 위임하는 여러 시나리오를 사용 설정하는 데 있으며, 교차 계정 액세스는 키 시나리오 중 한 가지에 해당합니다. 이 예시에서는 버킷 소유자인 계정 A가 IAM 역할을 사용하여 다른 AWS 계정의 사용자인 계정 C에 크로스 계정 객체 액세스 권한을 일시적으로 위임합니다. 생성하는 각 IAM 역할에 다음 두 개의 정책이 연결됩니다.

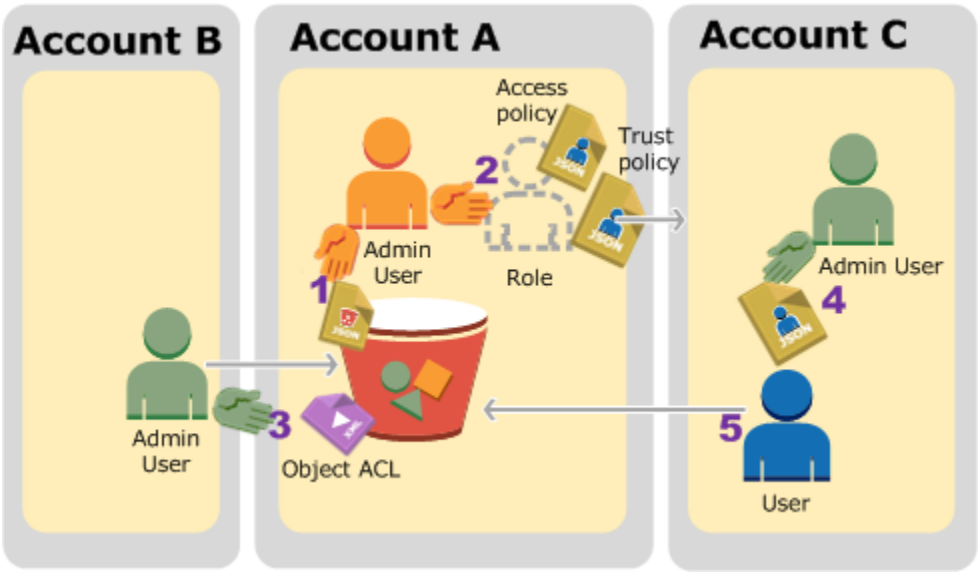
- 역할을 맡을 다른 AWS 계정을 식별하는 신뢰 정책.
- 사용자가 역할을 수임할 때 `s3:GetObject` 등 권한의 내용을 정의하는 액세스 정책. 정책에서 지정할 수 있는 권한 목록에 대해서는 Amazon S3의 정책 작업을 참조하십시오.

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 섹션을 참조하세요.

다음은 연습 단계 의 요약입니다.



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 참조)에 따라 본 연습에서는 AWS 계정 루트 사용자 자격 증명을 사용하지 않습니다. 대신 각 계정에서 관리자 사용자를 만들고 리소스를 만들고 권한을 부여할 때 해당 자격 증명을 사용합니다.

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: 시연 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. 이전 섹션의 테이블에서와 같이 AWS 계정이 세 개이며 각 계정에 관리자 사용자가 한 명인지 확인합니다.

   a. 필요한 경우 AWS 계정에 가입합니다. 이 계정을 계정 A, 계정 B, 계정 C라고 칭합니다.
   b. 계정 A의 자격 증명을 사용하여 IAM 콘솔에 로그인하고 다음을 수행하여 관리자 사용자를 만듭니다.

- **AccountAadmin** 사용자를 생성하고 보안 자격 증명을 적어 둡니다. 사용자 추가에 대한 자세한 내용은 IAM 사용 설명서의 AWS 계정에서 IAM 사용자 생성을 참조하십시오.
- 모든 권한을 제공하는 사용자 정책을 연결하여 AccountAadmin에 관리자 권한을 부여합니다. 지침은 IAM 사용 설명서에서 IAM 정책 관리를 참조하십시오.
- IAM 콘솔 대시보드에서 IAM 사용자 로그인 URL을 적어 둡니다. 해당 계정 사용자는 AWS Management Console에 로그인할 때 이 URL을 사용해야 합니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자로 AWS Management Console에 로그인을 참조하세요.

c. 이전 단계를 반복하며 계정 B와 계정 C에 속한 관리자 사용자를 만듭니다.

2. 계정 C의 경우 표준 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 ID 또는 계정 별칭, IAM 사용자 이름, 암호를 사용하여 Amazon S3 콘솔에 로그인합니다.

   b. Amazon S3 버킷의 이름을 선택하여 해당 버킷에 대한 세부 정보를 확인합니다.

   c. 권한 탭을 선택한 다음 액세스 제어 목록을 선택합니다.

   d. AWS 계정 계정에 액세스 섹션의 계정 열에 긴 식별자(예: `c1daexampleaaf850ea79cf0430f33d72579fd1611c97f7ded193374c0b163b6`)가 있습니다. 이것이 정식 사용자 ID입니다.

3. 버킷 정책을 만들려면 다음과 같은 정보가 필요합니다. 다음 값을 적어 둡니다.

   - 계정 A의 정식 사용자 ID – 계정 A 관리자가 계정 B 관리자에게 조건부 객체 업로드 권한을 부여할 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 콘솔에서 사용자 ARN을 찾을 수 있습니다. 사용자를 선택하고 요약 탭에서 사용자의 ARN을 찾아야 합니다.

     버킷 정책에서 `AccountBadmin`에 객체 업로드 권한을 위임하고 ARN을 이용해 사용자를 지정합니다. 다음은 ARN 값의 예입니다.

```
arn:aws:iam::AccountB-ID:user/AccountBadmin
```

4. AWS Command Line Interface(CLI) 또는 AWS Tools for Windows PowerShell을 설정합니다. 관리자 사용자 자격 증명은 다음과 같이 저장합니다.

   - AWS CLI를 사용하는 경우 구성 파일에 `AccountAadmin` 및 `AccountBadmin`의 프로파일을 생성합니다.
   - AWS Tools for Windows PowerShell을 사용하는 경우 세션의 자격 증명을 `AccountAadmin` 및 `AccountBadmin`으로 저장해야 합니다.

   지침은 연습을 위한 도구 설정 섹션을 참조하세요.

1단계: 계정 A 작업 수행

계정 A를 버킷 소유자로 예를 들어봅니다. 따라서 계정 A의 AccountAadmin은 다음 작업을 수행합니다.

- 버킷을 만듭니다.
- 계정 B 관리자에게 객체 업로드 권한을 부여하는 버킷 정책을 연결합니다.
- 버킷 내 객체에 액세스할 수 있도록 계정 C에 역할을 맡을 권한을 부여하는 IAM 역할을 생성합니다.

1.1단계: AWS Management Console에 로그인

계정 A의 IAM 사용자 로그인 URL을 사용하여 우선 **`AccountAadmin`** 사용자로 AWS Management Console에 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 *`amzn-s3-demo-bucket1`*(으)로 가정합니다.

   지침은 범용 버킷 생성 섹션을 참조하세요.
2. 다음 버킷 정책을 연결합니다. 해당 정책은 계정 B 관리자에게 객체를 업로드할 조건부 권한을 부여합니다.

   `amzn-s3-demo-bucket1`, *`AccountB-ID`* 및 *`CanonicalUserId-of-AWSaccountA-BucketOwner`*에 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:::amzn-s3-demo-bucket1/*"
        },
        {
            "Sid": "112",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-grant-full-control": "id=CanonicalUserId-of-
AWSaccountA-BucketOwner"
                }
            }
        }
    ]
}
```

1.3단계: 계정 A에서 계정 C의 크로스 계정 액세스를 허용하는 IAM 역할 생성

IAM 콘솔에서 계정 C에 역할을 맡을 권한을 부여하는 IAM 역할(**examplerole**)을 생성합니다. 반드시 계정 A에서 역할을 만들어야 하므로 사용자는 계정 A 관리자로 로그인된 상태를 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 선택한 다음 정책 생성을 선택합니다.

   b. 자체 정책 생성 옆의 선택을 선택하세요.

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 해 줍니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*"
    }
  ]
}
```

e. 정책 생성을 선택하세요.

새로운 정책이 관리형 정책 목록에 나타납니다.

2. 왼쪽 탐색 창에서 역할을 선택한 후 새 역할 생성을 선택합니다.

3. 역할 유형 선택에서 계정 간 액세스를 위한 역할을 선택한 뒤, 소유한 AWS 계정 계정 간 액세스 제공 옆의 선택 버튼을 클릭합니다.

4. 계정 C 계정 ID를 입력합니다.

이 연습에서는 사용자가 역할을 맡기 위해 다중 인증(MFA)을 설정하도록 요구할 필요가 없으므로 해당 옵션을 선택하지 않은 상태로 둡니다.

5. 다음 단계를 선택해 역할과 연결되는 권한을 설정합니다.

6. 생성한 access-accountA-bucket 정책 옆의 확인란을 선택한 후 다음 단계를 선택합니다.

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 사용하면 계정 ID와 역할 이름 필드가 이미 채워진 역할 전환 페이지로 바로 연결됩니다. 이 링크는 나중에 크로스 계정 역할에 대한 역할 요약 페이지에서도 확인할 수 있습니다.

7. 역할 이름에 `examplerole`을 입력한 뒤, 다음 단계를 선택합니다.

8. 역할을 검토한 후 역할 생성을 선택합니다.

역할 목록에 examplerole 역할이 표시됩니다.

9. 역할 이름 examplerole을 선택합니다.
10. 신뢰 관계 탭을 선택합니다.
11. 정책 문서 표시를 클릭해 나타나는 신뢰 정책이 다음 정책과 일치하는지 확인합니다.

    다음 신뢰 정책으로 계정 C에 sts:AssumeRole 작업을 허용해 계정 C와의 신뢰를 설정합니다. 자세한 내용을 알아보려면 AWS Security Token Service API 참조의 AssumeRole 섹션을 참조하세요.

```
{
  "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 태스크 수행

계정 A가 소유한 예시 버킷은 다른 계정 소유의 객체가 필요합니다. 이 단계에서 계정 B 관리자는 명령 행 도구를 이용해 객체를 업로드합니다.

- put-object AWS CLI 명령을 사용하여 객체를 *amzn-s3-demo-bucket1*에 업로드합니다.

```
aws s3api put-object --bucket amzn-s3-demo-bucket1 --key HappyFace.jpg --
body HappyFace.jpg --grant-full-control id="canonicalUserId-ofTheBucketOwner" --
profile AccountBadmin
```

다음 사항에 유의하세요.

- `--Profile` 파라미터가 `AccountBadmin` 프로파일을 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

전 단계에서 계정 A가 이미 역할 `examplerole`을 만들어 계정 C와의 신뢰를 설정했기 때문에 계정 C 사용자는 해당 역할을 사용하여 계정 A에 액세스할 수 있습니다. 이번 단계에서는 계정 C 관리자가 사용자(Dave)를 생성해 계정 A에서 받은 `sts:AssumeRole` 권한을 Dave에게 위임합니다. 이러한 접근 방식을 통해 Dave는 `examplerole`을 맡아 계정 A에 일시적으로 액세스할 수 있습니다. 계정 A가 이 역할에 연결한 액세스 정책은 Dave가 계정 A에 액세스해 할 수 있는 활동, 구체적으로는 *`amzn-s3-demo-bucket1`*에 있는 객체를 가져오는 활동을 제한합니다.

3.1단계: 계정 C에 사용자를 생성하고 examplerole을 맡을 권한 위임

1. 계정 C의 IAM 사용자 로그인 URL을 사용하여 우선 **`AccountCadmin`** 사용자로 AWS Management Console에 로그인합니다.

2. IAM 콘솔에서 사용자 Dave를 만듭니다.

   단계별 지침은 IAM 사용 설명서의 IAM 사용자 생성(AWS Management Console)을 참조하세요.

3. Dave의 자격 증명을 적어 둡니다. Dave가 `examplerole` 역할을 수임하려면 이 자격 증명이 필요합니다.

4. Dave IAM 사용자에 대한 인라인 정책을 만들어 계정 A의 `examplerole` 역할에 대한 `sts:AssumeRole` 권한을 Dave에게 위임합니다.

   a. 왼쪽에 있는 탐색 창에서 Users(사용자)를 선택합니다.

   b. 사용자 이름 Dave를 선택합니다.

   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 선택한 후 선택을 선택합니다.

f. 정책 이름 필드에 정책 이름을 입력합니다.

g. 다음 정책을 정책 문서 필드에 복사해 넣습니다.

*AccountA-ID*를 제공하여 정책을 업데이트해야 합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["sts:AssumeRole"],
      "Resource": "arn:aws:iam::AccountA-ID:role/examplerole"
    }
  ]
}
```

h. 정책 적용을 선택합니다.

5. 또 다른 프로파일 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)을 맡아 객체에 액세스

이제 Dave는 다음과 같이 계정 A가 소유한 버킷 내 객체에 액세스할 수 있습니다.

- Dave는 먼저 자신의 자격 증명으로 examplerole을 수임합니다. 이 때 임시 자격 증명이 반환됩니다.
- 임시 자격 증명으로 Dave는 계정 A의 버킷 내 객체에 액세스합니다.

1. 명령 프롬프트에서 AccountCDave 프로파일로 다음 AWS CLI assume-role 명령을 실행합니다.

examplerole가 정의된 *AccountA-ID*를 입력해 명령의 ARN 값을 업데이트해야 합니다.

```
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(AWS STS)가 임시 보안 자격 증명(액세스 키 ID, 비밀 액세스 키 및 세션 토큰)을 반환합니다.

2. 임시 보안 자격 증명을 TempCred 프로파일의 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 amzn-s3-demo-bucket1 --
key HappyFace.jpg SaveFileAs.jpg --profile TempCred
```

examplerole에 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 권한이 거부됩니다.

```
aws s3api get-object-acl --bucket amzn-s3-demo-bucket1 --key HappyFace.jpg --
profile TempCred
```

사용자 Dave로 그 역할을 수임해 임시 자격 증명으로 객체에 액세스합니다. *amzn-s3-demo-bucket1*의 객체에 액세스하는 계정 C의 애플리케이션으로도 엑세스가 가능합니다. 애플리케이션으로 임시 보안 자격 증명을 얻을 수 있습니다. 또한, 계정 C는 애플리케이션에 examplerole을 맡을 권한을 위임할 수 있습니다.

4단계: 정리

1. 테스트를 완료한 후, 다음과 같이 수행해 정리합니다.

- 계정 A의 자격 증명을 사용하여 AWS Management Console에 로그인하고 다음을 수행합니다.
  - Amazon S3 콘솔에서 *amzn-s3-demo-bucket1*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 Console에서 계정 A에 만든 `examplerole`을 제거합니다. 단계별 지침은 IAM 사용 설명서의 IAM 사용자 삭제를 참조하세요.
  - IAM 콘솔에서 AccountAadmin 사용자를 제거합니다.

2. 계정 B의 자격 증명을 사용하여 IAM 콘솔에 로그인합니다. 사용자 AccountBadmin을 삭제합니다.
3. 계정 C의 자격 증명을 사용하여 IAM 콘솔에 로그인합니다. AccountCadmin과 사용자 Dave를 삭제합니다.

관련 리소스

이 연습과 관련된 자세한 내용은 IAM 사용 설명서의 다음 리소스를 참조하세요.

- 역할을 생성하여 IAM 사용자에게 권한 위임
- 자습서: IAM 역할을 사용한 AWS 계정 간 액세스 권한 위임
- IAM 정책 관리

Amazon S3 스토리지 렌즈에 대한 서비스 연결 역할 사용

Amazon S3 스토리지 렌즈를 사용하여 AWS Organizations의 모든 계정에 대한 지표를 수집하고 집계하려면 먼저 S3 스토리지 렌즈에 조직의 관리 계정에 의해 사용되는 신뢰할 수 있는 액세스 권한이 있는지 확인해야 합니다. S3 Storage Lens는 조직에 속한 AWS 계정 목록을 가져올 수 있는 서비스 연결 역할을 생성합니다. S3 스토리지 렌즈는 이 계정 목록을 사용하여 S3 스토리지 렌즈 대시보드 또는 구성이 생성되거나 업데이트될 때 모든 구성원 계정의 S3 리소스에 대한 지표를 수집합니다.

Amazon S3 스토리지 렌즈는 AWS Identity and Access Management(IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 S3 스토리지 렌즈에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 S3 Storage Lens에서 사전 정의하며 서비스에서 사용자를 대신하여 다른 AWS 서비스를 직접적으로 호출하기 위해 요구하는 모든 권한을 포함합니다.

필요한 권한을 수동으로 추가할 필요가 없으므로 서비스 연결 역할은 S3 스토리지 렌즈를 더 쉽게 설정할 수 있습니다. S3 스토리지 렌즈에서 서비스 연결 역할 권한을 정의하므로, 달리 정의되지 않은 한 S3 스토리지 렌즈만 해당 역할을 맡을 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며, 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

먼저 관련 리소스를 삭제한 다음에만 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 S3 스토리지 렌즈 리소스에 대한 액세스 권한을 부주의로 삭제할 수 없기 때문에 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는 AWS 서비스를 참조하고 서비스 연결 역할(Service-Linked Role) 열에 예(Yes)가 표시된 서비스를 찾으십시오. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

Amazon S3 스토리지 렌즈에 대한 서비스 연결 역할 권한

S3 스토리지 렌즈는 AWSServiceRoleForS3StorageLens라는 서비스 연결 역할을 사용합니다. 이 역할이 있으면 S3 스토리지 렌즈에서 사용하거나 관리하는 AWS 서비스 및 리소스에 액세스할 수 있습니다. S3 스토리지 렌즈는 이 역할을 사용하여 사용자 대신 AWS Organizations 리소스에 액세스할 수 있습니다.

S3 스토리지 렌즈 서비스 연결 역할은 조직의 스토리지에서 다음 서비스를 신뢰합니다.

- `storage-lens.s3.amazonaws.com`

역할 권한 정책은 S3 스토리지 렌즈가 다음 작업을 완료하도록 허용합니다.

- `organizations:DescribeOrganization`

  `organizations:ListAccounts`

  `organizations:ListAWSServiceAccessForOrganization`

  `organizations:ListDelegatedAdministrators`

IAM 개체(사용자, 그룹, 역할 등)가 서비스 연결 역할을 작성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하십시오.

S3 스토리지 렌즈에 대한 서비스 연결 역할 생성

서비스 연결 역할은 수동으로 생성할 필요가 없습니다. AWS Organizations 관리 또는 위임 관리자 계정에 로그인한 상태에서 다음 태스크 중 하나를 완료하면 S3 스토리지 렌즈가 서비스 연결 역할을 생성합니다.

- Amazon S3 콘솔에서 조직에 대한 S3 스토리지 렌즈 대시보드 구성을 생성합니다.
- REST API, AWS CLI 및 SDK를 사용하여 조직에 대한 S3 스토리지 렌즈 구성에 대한 PUT 작업을 수행합니다.

> **Note**
>
> S3 Storage Lens는 조직당 최대 5명의 위임된 관리자를 지원합니다.

이 서비스 연결 역할을 삭제하는 경우 이전 작업을 수행하면 필요에 따라 서비스 연결 역할이 다시 생성됩니다.

S3 스토리지 렌즈 서비스 연결 역할에 대한 정책 예제

Example S3 스토리지 렌즈 서비스 연결 역할에 대한 권한 정책

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AwsOrgsAccess",
            "Effect": "Allow",
            "Action": [
                "organizations:DescribeOrganization",
                "organizations:ListAccounts",
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:ListDelegatedAdministrators"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Amazon S3 스토리지 렌즈의 서비스 연결 역할 편집

S3 Storage Lens에서는 AWSServiceRoleForS3StorageLens 서비스 연결 역할을 편집하도록 허용하지 않습니다. 서비스 연결 역할을 생성한 후에는 다양한 엔터티가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하십시오.

Amazon S3 스토리지 렌즈의 서비스 연결 역할 삭제

서비스 연결 역할을 더 이상 사용하지 않을 경우에는 해당 역할을 삭제하는 것이 좋습니다. 이렇게 하면 적극적으로 모니터링하거나 유지 관리하지 않은 미사용 엔터티가 없습니다. 단, 서비스 연결 역할에 대한 리소스를 먼저 정리해야 수동으로 삭제할 수 있습니다.

> **Note**
>
> 리소스를 삭제하려고 할 때 Amazon S3 스토리지 렌즈 서비스가 역할을 사용 중이면 삭제에 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하세요.

AWSServiceRoleForS3StorageLens를 삭제하려면 AWS Organizations 관리 또는 위임 관리자 계정을 사용하여 모든 AWS 리전에 있는 모든 조직 수준 S3 Storage Lens 구성을 삭제해야 합니다.

리소스는 조직 수준의 S3 스토리지 렌즈 구성입니다. S3 Storage Lens를 사용하여 리소스를 정리한 다음 IAM 콘솔, CLI, REST API 또는 AWS SDK를 사용하여 역할을 삭제합니다.

REST API, AWS CLI 및 SDK에서 조직이 S3 Storage Lens 구성을 생성한 모든 리전에서 `ListStorageLensConfigurations`를 사용하여 S3 Storage Lens 구성을 검색할 수 있습니다. `DeleteStorageLensConfiguration` 작업을 사용하여 이러한 구성을 삭제한 다음 역할을 삭제할 수 있습니다.

> **Note**
>
> 서비스 연결 역할을 삭제하려면 해당 역할이 있는 모든 리전에서 조직 수준 S3 스토리지 렌즈 구성을 모두 삭제해야 합니다.

AWSServiceRoleForS3StorageLens SLR에서 사용하는 Amazon S3 Storage Lens 리소스를 삭제하려면

1. 조직 수준 구성에 대한 목록을 가져오려면 S3 Storage Lens 구성이 있는 모든 리전에서 `ListStorageLensConfigurations`를 사용해야 합니다. 이 목록은 Amazon S3 콘솔에서도 가져올 수 있습니다.
2. `DeleteStorageLensConfiguration` API 직접 호출을 간접적으로 호출하거나 Amazon S3 콘솔을 사용하여 해당 리전의 엔드포인트에서 이러한 구성을 삭제합니다.

IAM을 사용하여 수동으로 서비스 연결 역할을 삭제하려면 다음을 수행하세요.

구성을 삭제한 후에는 IAM 콘솔을 사용하거나 IAM API `DeleteServiceLinkedRole`을 간접적으로 호출하거나 AWS CLI 또는 AWS SDK를 사용하여 AWSServiceRoleForS3StorageLens SLR을 삭제합니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 삭제를 참조하십시오.

S3 스토리지 렌즈 서비스 연결 역할의 지원 리전

S3 스토리지 렌즈는 서비스가 제공되는 모든 AWS 리전에서 서비스 연결 역할 사용을 지원합니다. 자세한 내용은 Amazon S3 리전 및 엔드포인트를 참조하십시오.

Amazon S3 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 Amazon S3 및 IAM으로 작업할 때 발생할 수 있는 일반적인 문제를 진단하고 수정할 수 있습니다.

주제

- 액세스 거부 오류 메시지를 받음
- Amazon S3에서 작업을 수행할 권한이 없음
- iam:PassRole을 수행하도록 인증되지 않음
- 내 AWS 계정 외부의 사람에게 내 Amazon S3 리소스 액세스를 허용하고 싶음
- Amazon S3의 액세스 거부(403 Forbidden) 오류 문제 해결

액세스 거부 오류 메시지를 받음

버킷 정책 또는 ID 기반 정책에 권한을 부여하려는 요청자에 대한 명시적인 `Deny` 문이 없어야 합니다.

액세스 거부 오류 문제 해결에 대한 자세한 내용은 Amazon S3의 액세스 거부(403 Forbidden) 오류 문제 해결 섹션을 참조하세요.

Amazon S3에서 작업을 수행할 권한이 없음

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

다음의 예제 오류는 mateojackson IAM 사용자가 콘솔을 사용하여 가상 *my-example-widget* 리소스에 대한 세부 정보를 보려고 하지만 가상 s3:*GetWidget* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 s3:GetWidget on resource: my-example-widget
```

이 경우, s3:*GetWidget* 작업을 사용하여 *my-example-widget* 리소스에 액세스할 수 있도록 mateojackson 사용자 정책을 업데이트해야 합니다.

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

iam:PassRole을 수행하도록 인증되지 않음

iam:PassRole 작업을 수행할 권한이 없다는 오류가 수신되면 Amazon S3에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스에서는 새로운 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 해당 서비스에 기존 역할을 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예시 오류는 이름이 marymajor인 IAM 사용자가 콘솔을 사용하여 Amazon S3에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우, Mary가 iam:PassRole 작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

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

내 AWS 계정 외부의 사람에게 내 Amazon S3 리소스 액세스를 허용하고 싶음

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

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

자세히 알아보려면 다음을 참조하세요.

- Amazon S3에서 이러한 기능을 지원하는지 여부를 알아보려면 Amazon S3에서 IAM을 사용하는 방법 섹션을 참조하세요.
- 소유하고 있는 AWS 계정의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 자신이 소유한 다른 AWS 계정의 IAM 사용자에 대한 액세스 권한 제공을 참조하세요.
- 리소스에 대한 액세스 권한을 서드 파티 AWS 계정에게 제공하는 방법을 알아보려면 IAM 사용 설명서의 서드 파티가 소유한 AWS 계정에 대한 액세스 제공을 참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.

Amazon S3의 액세스 거부(403 Forbidden) 오류 문제 해결

액세스 거부(HTTP `403 Forbidden`) 오류는 AWS가 권한 부여 요청을 명시적 또는 묵시적으로 거부할 때 나타납니다.

- 명시적 거부는 정책에 특정 AWS 작업에 대한 `Deny` 문이 포함되어 있을 때 발생합니다.
- 적용 가능한 `Deny` 문이 없고 적용 가능한 `Allow` 문도 없다면 암시적 거부가 발생합니다.

기본적으로 AWS Identity and Access Management(IAM) 정책은 IAM 보안 주체를 묵시적으로 거부하므로 정책은 보안 주체가 작업을 수행하도록 명시적으로 허용해야 합니다. 그렇지 않으면 정책이 암시적으로 액세스를 거부합니다. 자세한 내용은 IAM 사용 설명서의 명시적 거부와 묵시적 거부의 차이를 참조하세요. 액세스 요청의 허용 또는 거부 여부를 결정하는 정책 평가 로직에 대한 자세한 내용은 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

다음 주제에서는 Amazon S3의 액세스 거부 오류의 가장 일반적인 원인을 다룹니다.

> **Note**
>
> 액세스 거부(HTTP `403 Forbidden`) 오류의 경우 Amazon S3는 요청이 버킷 소유자의 개인 AWS 계정이나 버킷 소유자의 AWS 조직 외부에서 시작된 경우 버킷 소유자에게 요금을 청구하지 않습니다.

주제

- 액세스 거부 메시지의 예와 문제 해결 방법
- 버킷 정책 및 IAM 정책
- Amazon S3 ACL 설정
- S3 퍼블릭 액세스 차단 설정
- Amazon S3 암호화 설정
- S3 객체 잠금 설정
- VPC 엔드포인트 정책
- AWS Organizations 정책
- CloudFront 배포 액세스
- 액세스 포인트 설정
- 추가 리소스

> **Note**
>
> 권한 문제를 해결하려는 경우 the section called “액세스 거부 메시지의 예와 문제 해결 방법” 섹션부터 시작하여 ??? 섹션으로 이동하세요. 또한 the section called “권한 확인을 위한 팁”의 지침을 반드시 따르세요.

액세스 거부 메시지의 예와 문제 해결 방법

> **Important**
>
> 2025년 6월 16일부터 Amazon S3는 AWS Organizations에서 동일한 조직 내 계정의 리소스에 대한 요청에 대해 향상된 액세스 거부 오류 메시지를 제공하기 시작할 예정입니다. 이러한 메

시지는 AWS GovCloud (US) Regions 및 중국 리전을 포함한 모든 AWS 리전에서 사용할 수 있습니다.

이제 Amazon S3는 동일한 AWS 계정 내 또는 AWS Organizations의 동일한 조직 내의 리소스에 대한 요청의 액세스 거부(HTTP `403 Forbidden`) 오류에 추가 컨텍스트를 포함합니다. 이 새로운 컨텍스트에는 액세스를 거부한 정책 유형, 거부 이유, 리소스에 대한 액세스를 요청한 IAM 사용자 또는 역할에 대한 정보가 포함됩니다.

이 추가 컨텍스트는 액세스 문제를 해결하고, 액세스 거부 오류의 근본 원인을 파악하고, 관련 정책을 업데이트하여 잘못된 액세스 제어를 수정하는 데 도움이 됩니다. 이 추가 컨텍스트는 AWS CloudTrail 로그에서도 사용할 수 있습니다. 동일 계정 또는 동일 조직 요청에 대한 향상된 액세스 거부 오류 메시지는 이제 AWS GovCloud (US) Regions 및 중국 리전을 포함한 모든 AWS 리전에서 사용할 수 있습니다.

대부분의 액세스 거부 오류 메시지는 `User` *`user-arn`* `is not authorized to perform` *`action`* `on "`*`resource-arn`*`" because` *`context`* 형식으로 나타납니다. 이 예제에서 *`user-arn`*은 액세스를 수신하지 않는 Amazon 리소스 이름(ARN)이고, *`action`*은 정책이 거부하는 서비스 작업이고, *`resource-arn`*은 정책이 작용하는 리소스의 ARN입니다. *`context`* 필드는 정책이 액세스를 거부한 이유를 설명하는 정책 유형에 대한 추가 컨텍스트를 나타냅니다.

정책에 `Deny` 문이 포함되어 있어 정책이 액세스를 명시적으로 거부하는 경우 액세스 거부 오류 메시지에 `with an explicit deny in a` *`type`* `policy`라는 구절이 포함됩니다. 정책이 묵시적으로 액세스를 거부하는 경우 액세스 거부 오류 메시지에 `because no` *`type`* `policy allows the` *`action`* `action`이라는 구절이 포함됩니다.

Important

- 향상된 액세스 거부 메시지는 동일 계정 내 요청 또는 AWS Organizations의 동일 조직 내 요청인 경우에만 반환됩니다. 동일 조직 외부의 교차 계정 요청은 일반 `Access Denied` 메시지를 반환합니다.

  크로스 계정 액세스 요청의 허용 또는 거부 여부를 결정하는 정책 평가 로직에 대한 자세한 내용은 IAM 사용 설명서의 크로스 계정 정책 평가 로직을 참조하세요. 크로스 계정 액세스 권한을 부여하는 방법을 보여주는 연습은 the section called “교차 계정 권한 부여” 섹션을 참조하세요.

- AWS Organizations의 동일한 조직 내 요청인 경우:

  - 가상 프라이빗 클라우드(VPC) 엔드포인트 정책으로 인해 거부가 발생하면 향상된 액세스 거부 메시지가 반환되지 않습니다.
  - 향상된 액세스 거부 메시지는 버킷 소유자와 호출자 계정이 모두 AWS Organizations의 동일한 조직에 속할 때마다 제공됩니다. S3 객체 소유권 버킷 소유자 기본 설정 또는 객체 라이터 설정으로 구성된 버킷에 서로 다른 계정이 소유한 객체가 포함될 수 있지만 객체 소유권은 향상된 액세스 거부 메시지에 영향을 주지 않습니다. 버킷 소유자와 호출자가 특정 객체를 소유한 사람과 관계없이 동일한 조직에 있는 한 모든 객체 요청에 대해 향상된 액세스 거부 메시지가 반환됩니다. 객체 소유권 설정 및 구성에 대한 자세한 내용은 the section called “객체 소유권 제어” 섹션을 참조하세요.
- 디렉터리 버킷에 대한 요청의 경우 향상된 액세스 거부 오류 메시지가 반환되지 않습니다. 디렉터리 버킷 요청은 일반 `Access Denied` 메시지를 반환합니다.
- 동일한 정책 유형의 여러 정책이 권한 부여 요청을 거부하는 경우, 액세스 거부 오류 메시지에 정책 수가 지정되지 않습니다.
- 여러 정책 유형이 권한 부여 요청을 거부하는 경우 오류 메시지에 이러한 정책 유형 중 하나만 포함됩니다.
- 여러 이유로 인해 액세스 요청이 거부되는 경우, 오류 메시지에는 거부 이유 중 하나만 포함됩니다.

다음 예제는 다양한 유형의 액세스 거부 오류 메시지와 각 메시지 유형의 문제를 해결하는 방법을 보여줍니다.

서비스 제어 정책으로 인한 액세스 거부 - 명시적 거부

1. 리소스 제어 정책(RCP)에서 작업에 대한 `Deny` 문이 있는지 확인합니다. 다음 예제에서 작업은 `s3:GetObject`입니다.
2. `Deny` 문을 제거하여 RCP를 업데이트합니다. 자세한 내용은 AWS Organizations 사용 설명서의 Update a resource control policy (RCP)를 참조하세요.

```
An error occurred (AccessDenied) when calling the GetObject operation:
User: arn:aws:iam::777788889999:user/MaryMajor is not authorized to perform:
s3:GetObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name"
with an explicit deny in a resource control policy
```

서비스 제어 정책으로 인한 액세스 거부 - 암묵적 거부

1. 서비스 제어 정책(SCP)에서 작업에 대한 누락된 `Allow` 문이 있는지 확인합니다. 다음 예제에서 작업은 `s3:GetObject`입니다.
2. `Allow` 문을 추가하여 SCP를 업데이트합니다. 자세한 내용은 AWS Organizations 사용 설명서의 SCP 업데이트를 참조하세요.

```
User: arn:aws:iam::777788889999:user/MaryMajor is not authorized to perform:
s3:GetObject because no service control policy allows the s3:GetObject action
```

서비스 제어 정책으로 인한 액세스 거부 - 명시적 거부

1. 서비스 제어 정책(SCP)에서 작업에 대한 `Deny` 문이 있는지 확인합니다. 다음 예제에서 작업은 `s3:GetObject`입니다.
2. 사용자에게 필요한 액세스를 허용하도록 `Deny` 문을 변경하여 SCP를 업데이트합니다. 작업 수행 방법에 대한 예제는 AWS Organizations 사용 설명서의 IAM 사용자 및 역할이 지정된 변경을 수행하지 못하도록 방지(지정된 관리자 역할은 제외)를 참조하세요. SCP 업데이트에 대한 자세한 내용은 AWS Organizations 사용 설명서의 SCP 업데이트를 참조하세요.

```
User: arn:aws:iam::777788889999:user/MaryMajor is not authorized to perform:
s3:GetObject with an explicit deny in a service control policy
```

VPC 엔드포인트 정책으로 인한 액세스 거부 - 암시적 거부

1. 가상 프라이빗 클라우드(VPC) 엔드포인트 정책에서 작업에 대한 누락된 `Allow` 문을 확인합니다. 다음 예제에서 작업은 `s3:GetObject`입니다.
2. `Allow` 문을 추가하여 VPC 엔드포인트 정책을 업데이트합니다. 자세한 정보는 AWS PrivateLink 안내서의 VPC 엔드포인트 정책 업데이트를 참조하세요.

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject because no VPC endpoint policy allows the s3:GetObject action
```

VPC 엔드포인트 정책으로 인한 액세스 거부 - 명시적 거부

1. 가상 프라이빗 클라우드(VPC) 엔드포인트 정책에서 작업에 대한 명시적 `Deny` 문을 확인합니다. 다음 예제에서 작업은 `s3:GetObject`입니다.

2. 사용자에게 필요한 액세스를 허용하도록 Deny 문을 변경하여 VPC 엔드포인트 정책을 업데이트합니다. 예를 들어, the section called “예제 7: Deny 문에서 보안 주체 제외”에서와 같이 StringNotEquals 조건 연산자와 함께 aws:PrincipalAccount 조건 키를 사용하여 특정 보안 주체 액세스를 허용하도록 Deny 문을 업데이트할 수 있습니다. VPC 엔드포인트 정책 업데이트에 대한 자세한 내용은 AWS PrivateLink 안내서의 VPC 엔드포인트 정책 업데이트를 참조하세요.

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name" with
an explicit deny in a VPC endpoint policy
```

권한 경계로 인한 액세스 거부 - 암시적 거부

1. 권한 경계에서 작업에 대한 누락된 Allow 문을 확인합니다. 다음 예제에서 작업은 s3:GetObject입니다.
2. IAM 정책에 Allow 문을 추가하여 권한 경계를 업데이트합니다. 자세한 내용은 IAM 사용 설명서의 IAM 엔터티에 대한 권한 경계 및 IAM 정책 편집을 참조하세요.

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name"
because no permissions boundary allows the s3:GetObject action
```

권한 경계로 인한 액세스 거부 - 명시적 거부

1. 권한 경계에서 작업에 대한 명시적 Deny 문을 확인합니다. 다음 예제에서 작업은 s3:GetObject입니다.
2. 사용자에게 필요한 액세스를 허용하도록 IAM 정책의 Deny 문을 변경하여 권한 경계를 업데이트합니다. 예를 들어, IAM 사용 설명서의 aws:PrincipalAccount에 나와 있는 것처럼 StringNotEquals 조건 연산자와 함께 aws:PrincipalAccount 조건 키를 사용하여 특정 보안 주체 액세스를 허용하도록 Deny 문을 업데이트할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 엔터티에 대한 권한 경계 및 IAM 정책 편집을 참조하세요.

```
User: arn:aws:iam::777788889999:user/MaryMajor is not authorized to perform:
s3:GetObject with an explicit deny in a permissions boundary
```

세션 정책으로 인한 액세스 거부 - 암시적 거부

1. 세션 정책에서 작업에 대한 누락된 `Allow` 문을 확인합니다. 다음 예제에서 작업은 `s3:GetObject`입니다.
2. `Allow` 문을 추가하여 세션 정책을 업데이트합니다. 자세한 내용은 IAM 사용 설명서의 세션 정책 및 IAM 정책 편집을 참조하세요.

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject because no session policy allows the s3:GetObject action
```

세션 정책으로 인한 액세스 거부 - 명시적 거부

1. 세션 정책에서 작업에 대한 명시적 `Deny` 문을 확인합니다. 다음 예제에서 작업은 `s3:GetObject`입니다.
2. 사용자에게 필요한 액세스를 허용하도록 `Deny` 문을 변경하여 세션 정책을 업데이트합니다. 예를 들어, the section called “예제 7: Deny 문에서 보안 주체 제외”에서와 같이 `StringNotEquals` 조건 연산자와 함께 `aws:PrincipalAccount` 조건 키를 사용하여 특정 보안 주체 액세스를 허용하도록 `Deny` 문을 업데이트할 수 있습니다. 세션 정책 업데이트에 대한 자세한 내용은 IAM 사용 설명서의 세션 정책 및 IAM 정책 편집을 참조하세요.

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name" with
an explicit deny in a session policy
```

리소스 기반 정책으로 인한 액세스 거부 - 암시적 거부

> **Note**
>
> 리소스 기반 정책은 버킷 정책 및 액세스 포인트 정책과 같은 정책을 의미합니다.

1. 리소스 기반 정책에서 작업에 대한 누락된 `Allow` 문을 확인합니다. 또한 `IgnorePublicAcls` S3 퍼블릭 액세스 차단 설정이 버킷, 액세스 포인트 또는 계정 수준에 적용되는지 확인하세요. 다음 예제에서 작업은 `s3:GetObject`입니다.
2. `Allow` 문을 추가하여 정책을 업데이트합니다. 자세한 내용은 IAM 사용 설명서의 리소스 기반 정책 및 IAM 정책 편집을 참조하세요.

버킷, 액세스 포인트 또는 계정에 대한 IgnorePublicAcls 퍼블릭 액세스 차단 설정을 조정해야 할 수도 있습니다. 자세한 내용은 the section called “퍼블릭 액세스 차단 설정으로 인한 액세스 거부” 및 the section called “버킷 및 액세스 포인트 설정 구성”(을)를 참조하세요.

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject because no resource-based policy allows the s3:GetObject action
```

리소스 기반 정책으로 인한 액세스 거부 - 명시적 거부

> **Note**
>
> 리소스 기반 정책은 버킷 정책 및 액세스 포인트 정책과 같은 정책을 의미합니다.

1. 리소스 기반 정책에서 작업에 대한 명시적 Deny 문을 확인합니다. 또한 RestrictPublicBuckets S3 퍼블릭 액세스 차단 설정이 버킷, 액세스 포인트 또는 계정 수준에 적용되는지 확인하세요. 다음 예제에서 작업은 s3:GetObject입니다.
2. 사용자에게 필요한 액세스를 허용하도록 Deny 문을 변경하여 정책을 업데이트합니다. 예를 들어, the section called “예제 7: Deny 문에서 보안 주체 제외”에서와 같이 StringNotEquals 조건 연산자와 함께 aws:PrincipalAccount 조건 키를 사용하여 특정 보안 주체 액세스를 허용하도록 Deny 문을 업데이트할 수 있습니다. 리소스 기반 정책 업데이트에 대한 자세한 내용은 IAM 사용 설명서의 리소스 기반 정책 및 IAM 정책 편집을 참조하세요.

   버킷, 액세스 포인트 또는 계정에 대한 RestrictPublicBuckets 퍼블릭 액세스 차단 설정을 조정해야 할 수도 있습니다. 자세한 내용은 the section called “퍼블릭 액세스 차단 설정으로 인한 액세스 거부” 및 the section called “버킷 및 액세스 포인트 설정 구성”(을)를 참조하세요.

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name" with
an explicit deny in a resource-based policy
```

자격 증명 기반 정책으로 인한 액세스 거부 - 암시적 거부

1. 자격 증명에 연결된 자격 증명 기반 정책에서 작업에 대한 누락된 Allow 문을 확인합니다. 다음 예제에서 작업은 사용자 MaryMajor에게 연결된 s3:GetObject입니다.

2. Allow 문을 추가하여 정책을 업데이트합니다. 자세한 내용은 IAM 사용 설명서의 자격 증명 기반 정책 및 IAM 정책 편집을 참조하세요.

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject because no identity-based policy allows the s3:GetObject action
```

자격 증명 기반 정책으로 인한 액세스 거부 - 명시적 거부

1. 자격 증명에 연결된 자격 증명 기반 정책에서 작업에 대한 명시적 Deny 문을 확인합니다. 다음 예제에서 작업은 사용자 MaryMajor에게 연결된 s3:GetObject입니다.
2. 사용자에게 필요한 액세스를 허용하도록 Deny 문을 변경하여 정책을 업데이트합니다. 예를 들어, IAM 사용 설명서의 aws:PrincipalAccount에 나와 있는 것처럼 StringNotEquals 조건 연산자와 함께 aws:PrincipalAccount 조건 키를 사용하여 특정 보안 주체 액세스를 허용하도록 Deny 문을 업데이트할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 자격 증명 기반 정책 및 IAM 정책 편집을 참조하세요.

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name" with
an explicit deny in an identity-based policy
```

퍼블릭 액세스 차단 설정으로 인한 액세스 거부

Amazon S3 퍼블릭 액세스 차단 기능은 액세스 포인트, 버킷 및 계정에 대한 설정을 제공하여 Amazon S3 리소스에 대한 퍼블릭 액세스를 관리하는 데 도움을 줍니다. Amazon S3가 '퍼블릭'을 어떻게 정의하는지에 대한 자세한 내용은 "퍼블릭"의 의미 섹션을 참조하세요.

기본적으로 새 버킷, 액세스 포인트 및 객체는 퍼블릭 액세스를 허용하지 않습니다. 그러나 사용자는 퍼블릭 액세스를 허용하도록 버킷 정책, 액세스 포인트 정책, IAM 사용자 정책, 객체 권한 또는 액세스 제어 목록(ACL)을 수정할 수 있습니다. S3 퍼블릭 액세스 차단 설정은 이러한 정책, 권한 및 ACL보다 우선합니다. 2023년 4월부터 모든 퍼블릭 액세스 차단 설정이 새 버킷에 기본값으로 활성화되어 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.

Amazon S3 퍼블릭 액세스 차단에서 제공하는 네 가지 설정은 다음과 같습니다.

- `BlockPublicAcls` - 이 설정은 `PutBucketAcl`, `PutObjectAcl`, `PutObject`, `CreateBucket`, `CopyObject` 및 `POST Object` 요청에 적용됩니다. `BlockPublicAcls` 설정을 사용하면 다음과 같은 동작이 발생합니다.
  - 지정된 액세스 제어 목록(ACL)이 퍼블릭이면 `PutBucketAcl` 및 `PutObjectAcl` 호출이 실패합니다.
  - 요청에 퍼블릭 ACL이 포함되어 있으면 `PutObject` 호출이 실패합니다.
  - 이 설정이 계정에 적용되면, 요청에 퍼블릭 ACL이 포함된 경우 `CreateBucket` 호출이 실패하고 HTTP `400`(`Bad Request`) 응답이 반환됩니다.

  예를 들어, `BlockPublicAcls` 설정 때문에 `CopyObject` 요청에 대한 액세스가 거부되면 다음 메시지가 표시됩니다.

  ```
  An error occurred (AccessDenied) when calling the CopyObject operation:
  User: arn:aws:sts::123456789012:user/MaryMajor is not authorized to
  perform: s3:CopyObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name"
  because public access control lists (ACLs) are blocked by the BlockPublicAcls block
  public access setting.
  ```

- `IgnorePublicAcls` - `IgnorePublicAcls` 설정을 사용하면 Amazon S3가 버킷의 모든 퍼블릭 ACL 및 거기에 포함된 모든 객체를 무시합니다. 퍼블릭 ACL에서만 요청 권한을 부여한 경우 `IgnorePublicAcls` 설정이 요청을 거부합니다.

  `IgnorePublicAcls` 설정으로 인한 모든 거부는 묵시적입니다. 예를 들어 퍼블릭 ACL로 인해 `IgnorePublicAcls`가 `GetObject` 요청을 거부하는 경우 다음 메시지가 표시됩니다.

  ```
  User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
  s3:GetObject because no resource-based policy allows the s3:GetObject action
  ```

- `BlockPublicPolicy` - 이 설정은 `PutBucketPolicy` 및 `PutAccessPointPolicy` 요청에 적용됩니다.

버킷에 BlockPublicPolicy를 설정하면 지정된 버킷 정책이 퍼블릭 액세스를 허용하는 경우 PutBucketPolicy에 대한 직접 호출을 Amazon S3가 거부합니다. 또한 이 설정을 사용하면 지정된 버킷 정책이 퍼블릭 액세스를 허용하는 경우, 버킷의 동일한 계정 액세스 포인트 모두에 PutAccessPointPolicy에 대한 직접 호출을 Amazon S3가 거부합니다.

액세스 포인트에 BlockPublicPolicy를 설정하면 지정된 정책(액세스 포인트 또는 기본 버킷)이 퍼블릭 액세스를 허용하는 경우, 액세스 포인트를 통해 이루어지는 PutAccessPointPolicy 및 PutBucketPolicy에 대한 직접 호출을 Amazon S3가 거부합니다.

예를 들어, BlockPublicPolicy 설정 때문에 PutBucketPolicy 요청에 대한 액세스가 거부되면 다음 메시지가 표시됩니다.

```
An error occurred (AccessDenied) when calling the PutBucketPolicy operation:
User: arn:aws:sts::123456789012:user/MaryMajor is not authorized to
perform: s3:PutBucketPolicy on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-
name"
because public policies are blocked by the BlockPublicPolicy block public
access setting.
```

- RestrictPublicBuckets - RestrictPublicBuckets 설정을 사용하면 퍼블릭 정책이 있는 액세스 포인트 또는 버킷에 대한 액세스가 버킷 소유자 계정 및 액세스 포인트 소유자 계정 내에서 권한 있는 사용자와 AWS 서비스 보안 주체로만 제한됩니다. 이 설정은 AWS 서비스 보안 주체에서 제외한 액세스 포인트 또는 버킷에 대한 모든 크로스 계정 액세스를 차단하지만 계정 내 사용자는 계속 액세스 포인트 또는 버킷을 관리할 수 있습니다. 또한 이 설정은 모든 익명(또는 서명되지 않은) 호출을 거부합니다.

  RestrictPublicBuckets 설정으로 인한 모든 거부는 명시적입니다. 예를 들어 퍼블릭 버킷 또는 액세스 포인트 정책으로 인해 RestrictPublicBuckets가 GetObject 요청을 거부하는 경우 다음 메시지가 표시됩니다.

  ```
  User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
  s3:GetObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name" with
  an explicit deny in a resource-based policy
  ```

이러한 설정에 대한 자세한 내용은 the section called “퍼블릭 액세스 차단 설정” 섹션을 참조하세요. 이러한 설정을 검토하고 업데이트하려면 the section called “퍼블릭 액세스 차단 구성” 섹션을 참조하세요.

버킷 정책 및 IAM 정책

버킷 수준 작업

버킷 정책이 없는 경우 버킷은 버킷 소유 계정의 모든 AWS Identity and Access Management(IAM) 자격 증명에서 오는 요청을 묵시적으로 허용합니다. 또한 버킷은 다른 계정의 다른 IAM 아이덴티티의 요청과 익명(서명되지 않은) 요청을 암시적으로 거부합니다. 그러나 IAM 사용자 정책이 없는 경우 요청자(AWS 계정 루트 사용자가 아닌 경우)의 요청이 묵시적으로 거부됩니다. 이 평가 논리에 대한 자세한 내용은 IAM 사용 설명서에서 계정 내에서 요청 허용 여부 결정을 참조하세요.

객체 수준 작업

버킷 소유 계정에서 객체를 소유한 경우 버킷 정책 및 IAM 사용자 정책은 객체 수준 작업에 대해 버킷 수준 작업과 동일한 방식으로 작동합니다. 예를 들어, 버킷 정책이 없는 경우 버킷은 버킷 소유 계정의 모든 IAM 아이덴티티에서 오는 객체 요청을 묵시적으로 허용합니다. 또한 버킷은 다른 계정의 다른 IAM 아이덴티티의 객체 요청과 익명(서명되지 않은) 요청을 암시적으로 거부합니다. 그러나 IAM 사용자 정책이 없는 경우 요청자(AWS 계정 루트 사용자가 아닌 경우)의 객체 요청이 묵시적으로 거부됩니다.

외부 계정에서 객체를 소유한 경우 객체 액세스 제어 목록(ACL)을 통해서만 객체에 대한 액세스 권한을 부여할 수 있습니다. 여전히 버킷 정책과 IAM 사용자 정책을 사용하여 객체 요청을 거부할 수 있습니다.

따라서 버킷 정책 또는 IAM 사용자 정책으로 인해 액세스 거부(403 금지) 오류가 발생하지 않게 하려면 다음 요구 사항을 충족해야 합니다.

- 동일 계정 액세스의 경우 버킷 정책 또는 IAM 사용자 정책 내에 권한을 부여하려는 요청자에 대한 명시적인 `Deny` 명령문이 없어야 합니다. 버킷 정책과 IAM 사용자 정책만 사용하여 권한을 부여하려면 이러한 정책 중 하나에 명시적인 `Allow` 명령문이 하나 이상 있어야 합니다.
- 크로스 계정 액세스의 경우 버킷 정책 또는 IAM 사용자 정책 내에 권한을 부여하려는 요청자에 대한 명시적인 `Deny` 명령문이 없어야 합니다. 버킷 정책과 IAM 사용자 정책만 사용하여 크로스 계정 권한을 부여하려면 요청자의 버킷 정책과 IAM 사용자 정책 모두에 명시적인 `Allow` 명령문을 포함해야 합니다.

> **Note**
>
> 버킷 정책의 `Allow` 명령문은 동일한 버킷 소유 계정이 소유한 객체에만 적용됩니다. 하지만 버킷 정책의 `Deny` 명령문은 객체 소유권과 관계없이 모든 객체에 적용됩니다.

버킷 정책을 검토 또는 편집하는 방법

> **Note**
>
> 버킷 정책을 보거나 편집하려면 `s3:GetBucketPolicy` 권한이 있어야 합니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 목록에서 버킷 정책을 보거나 편집할 버킷의 이름을 선택합니다.
4. 권한 탭을 선택합니다.
5. 버킷 정책에서 편집을 선택합니다. 버킷 정책 편집 페이지가 나타납니다.

AWS Command Line Interface(AWS CLI)를 사용하여 버킷 정책을 검토하거나 편집하려면 get-bucket-policy 명령을 사용하세요.

> **Note**
>
> 잘못된 버킷 정책으로 인해 버킷이 잠긴 경우 AWS 계정 루트 사용자 자격 증명을 사용하여 AWS Management Console에 로그인하세요. 버킷에 다시 액세스하려면 AWS 계정 루트 사용자 자격 증명을 사용하여 잘못된 버킷 정책을 삭제해야 합니다.

권한 확인을 위한 팁

요청자에게 Amazon S3 작업을 수행할 수 있는 적절한 권한이 있는지 확인하려면 다음을 시도해 보세요.

- 요청자를 식별합니다. 서명되지 않은 요청인 경우 IAM 사용자 정책이 없는 익명 요청입니다. 미리 서명된 URL을 사용하는 요청인 경우 사용자 정책은 요청에 서명한 IAM 사용자 또는 역할에 대한 정책과 동일합니다.
- 올바른 IAM 사용자 또는 역할을 사용하고 있는지 확인합니다. AWS Management Console 오른쪽 상단을 확인하거나 aws sts get-caller-identity 명령을 사용하여 IAM 사용자 또는 역할을 확인할 수 있습니다.
- 요청을 수행하는 IAM 사용자 또는 역할과 관련된 IAM 정책을 확인합니다. 다음 방법 중 하나를 사용할 수 있습니다.

  - IAM 정책 시뮬레이터로 IAM 정책을 테스트합니다.
  - 다양한 IAM 정책 유형을 검토합니다.
- 필요한 경우 IAM 사용자 정책을 편집합니다.
- 액세스를 명시적으로 거부하거나 허용하는 다음 정책의 예시를 검토합니다.
  - 명시적 허용 IAM 정책: IAM: 프로그래밍 방식과 콘솔에서 여러 서비스에 대한 액세스 허용 및 거부
  - 명시적 허용 버킷 정책: 여러 계정에 객체를 업로드하거나 퍼블릭 액세스에 객체 ACL을 퍼블릭 설정할 수 있는 권한 부여
  - 명시적 거부 IAM 사용자 정책: AWS: 요청된 AWS 리전에 따라 AWS에 대한 액세스 거부
  - 명시적 거부 버킷 정책: 버킷에 작성되는 모든 객체에 SSE-KMS 요구

Amazon S3 ACL 설정

ACL 설정을 확인할 때는 먼저 객체 소유권 설정을 검토하여 버킷에 ACL이 활성화되어 있는지 확인하세요. ACL 권한은 권한을 부여하는 용도로만 사용할 수 있으며 요청을 거부하는 데는 사용할 수 없다는 점을 유의하시기 바랍니다. 또한 ACL은 버킷 정책 또는 IAM 사용자 정책의 명시적 거부에 의해 거부된 요청자에게 액세스 권한을 부여하는 데 사용할 수 없습니다.

객체 소유권 설정이 버킷 소유자 적용으로 설정됨

버킷 소유자 적용 설정을 활성화하면 ACL 설정으로 인해 액세스 거부(403 금지) 오류가 발생할 가능성은 거의 없습니다. 이 설정은 버킷과 객체에 적용되는 모든 ACL을 비활성화하기 때문입니다. 버킷 소유자 적용은 Amazon S3 버킷의 기본 및 권장 설정입니다.

객체 소유권 설정이 버킷 소유자 기본 설정 또는 객체 라이터로 설정됨

ACL 권한은 버킷 소유자 기본 설정 또는 객체 라이터 설정에서도 여전히 유효합니다. ACL에는 버킷 ACL과 객체 ACL이라는 두 가지 종류가 있습니다. 두 ACL 유형의 차이점에 대해서는 ACL 권한 및 액세스 정책 권한 매핑을 참조하세요.

거부된 요청의 작업에 따라 버킷 또는 객체에 대한 ACL 권한을 확인하세요.

- Amazon S3에서 `LIST`, `PUT` 객체, `GetBucketAcl` 또는 `PutBucketAcl` 요청을 거부한 경우 버킷에 대한 ACL 권한을 검토하세요.

> **Note**
>
> 버킷 ACL 설정으로는 `GET` 객체 권한을 부여할 수 없습니다.

- Amazon S3가 S3 객체에 대한 GET 요청 또는 PutObjectAcl 요청을 거부한 경우 해당 객체에 대한 ACL 권한을 검토하세요.

> **Important**
>
> 객체를 소유한 계정이 버킷을 소유한 계정과 다른 경우 객체에 대한 액세스는 버킷 정책으로 제어되지 않습니다.

크로스 계정 객체 소유권인 경우 **GET** 객체 요청으로 인한 액세스 거부(403 금지) 오류 문제 해결

버킷의 객체 소유권 설정을 검토하여 객체 소유자를 파악합니다. 객체 ACL에 액세스할 수 있는 경우 객체 소유자의 계정도 확인할 수 있습니다. (객체 소유자의 계정을 보려면 Amazon S3 콘솔에서 객체 ACL 설정을 검토하세요.) 또는 객체 소유자의 정식 ID를 찾도록 GetObjectAcl 요청을 수행하여 객체 소유자 계정을 확인할 수도 있습니다. 기본적으로 ACL은 GET 요청에 대한 명시적 허용 권한을 객체 소유자의 계정에 부여합니다.

객체 소유자가 버킷 소유자와 다르다는 것을 확인한 후 사용 사례 및 액세스 수준에 따라 다음 방법 중 하나를 선택하여 액세스 거부(403 금지) 오류를 해결하는 데 도움을 받으세요.

- ACL 비활성화(권장) - 이 방법은 모든 객체에 적용되며 버킷 소유자가 수행할 수 있습니다. 이 방법은 버킷 소유자에게 버킷의 모든 객체에 대한 소유권과 완전한 제어 권한을 자동으로 부여합니다. 이 방법을 구현하기 전에 ACL 사용 중지를 위한 사전 조건을 확인하세요. 버킷을 버킷 소유자 적용(권장) 모드로 설정하는 방법에 대한 자세한 내용은 기존 버킷에 대한 객체 소유권 설정을 참조하세요.

> **Important**
>
> 액세스 거부(403 금지) 오류를 방지하려면 ACL을 비활성화하기 전에 반드시 ACL 권한을 버킷 정책으로 마이그레이션해야 합니다. 자세한 내용은 ACL 권한에서 마이그레이션하기 위한 버킷 정책 예시를 참조하세요.

- 객체 소유자를 버킷 소유자로 변경 - 이 방법은 개별 객체에 적용할 수 있지만 객체 소유자(또는 적절한 권한이 있는 사용자)만 객체 소유권을 변경할 수 있습니다. 추가 PUT 요금이 적용될 수 있습니다. (자세한 내용은 Amazon S3 요금을 참조하세요.) 이 방법은 버킷 소유자에게 객체의 전체 소유권을 부여하여 버킷 소유자가 버킷 정책을 통해 객체에 대한 액세스를 제어할 수 있도록 합니다.

  객체 소유권을 변경하려면 다음 중 하나를 수행합니다.

  - 사용자(버킷 소유자)는 객체를 다시 버킷에 복사할 수 있습니다.

- 사용자는 버킷의 객체 소유권 설정을 버킷 소유자 기본 설정으로 변경할 수 있습니다. 버전 관리가 비활성화된 경우 버킷의 객체를 덮어씁니다. 버전 관리가 활성화된 경우 동일한 객체의 중복 버전이 버킷에 표시되며, 버킷 소유자는 만료되도록 수명 주기 규칙을 설정할 수 있습니다. 객체 소유권 설정을 변경하는 자세한 방법은 기존 버킷에 대한 객체 소유권 설정 섹션을 참조하세요.

  > **Note**
  >
  > 객체 소유권 설정을 버킷 소유자 기본 설정으로 업데이트하면 해당 설정은 버킷에 업로드되는 신규 객체에만 적용됩니다.

- 객체 소유자가 `bucket-owner-full-control` 미리 제공된 객체 ACL을 사용하여 객체를 다시 업로드하도록 할 수 있습니다.

  > **Note**
  >
  > 크로스 계정 업로드의 경우 버킷 정책에 `bucket-owner-full-control` 미리 제공된 객체 ACL을 요구할 수도 있습니다. 예시 버킷 정책은 버킷 소유자가 완벽하게 제어할 수 있도록 보증하면서 객체에 업로드하는 크로스 계정 권한 부여를 참조하세요.

- 객체 라이터를 객체 소유자로 유지 - 이 방법을 사용하면 객체 소유자를 변경하지 않지만 객체에 개별적으로 액세스 권한을 부여할 수 있습니다. 객체에 대한 액세스 권한을 부여하려면 객체에 대한 `PutObjectAcl` 권한이 있어야 합니다. 그런 다음 액세스 거부(403 금지) 오류를 수정하려면 요청자를 객체 ACL에 있는 객체에 액세스할 수 있는 피부여자로 추가하세요. 자세한 내용은 ACL 구성 섹션을 참조하세요.

S3 퍼블릭 액세스 차단 설정

실패한 요청에 퍼블릭 액세스 또는 퍼블릭 정책이 포함된 경우 계정, 버킷 또는 액세스 포인트에서 S3 퍼블릭 액세스 차단 설정을 확인하세요. S3 퍼블릭 액세스 차단 설정과 관련된 액세스 거부 오류 문제 해결에 대한 자세한 내용은 the section called “퍼블릭 액세스 차단 설정으로 인한 액세스 거부” 섹션을 참조하세요.

Amazon S3 암호화 설정

Amazon S3는 버킷에 대한 서버 측 암호화를 지원합니다. 서버 측 암호화는 데이터를 받는 애플리케이션 또는 서비스에 의해 해당 대상에서 데이터를 암호화하는 것입니다. Amazon S3에서 AWS 데이터 센터의 디스크에 데이터를 쓰면서 객체 수준에서 데이터를 암호화하고 사용자가 해당 데이터에 액세스할 때 자동으로 암호를 해독합니다.

이제 기본적으로 Amazon S3가 Amazon S3 관리형 키(SSE-S3)를 사용한 서버 측 암호화를 Amazon S3 내 모든 버킷 암호화의 기본 수준으로 적용합니다. 또한 Amazon S3에서는 객체를 업로드할 때 서버 측 암호화 방법을 지정할 수 있습니다.

버킷의 서버측 암호화 상태 및 암호화 설정을 검토하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 목록에서 암호화 설정을 확인할 버킷을 선택합니다.
4. 속성(Properties) 탭을 선택합니다.
5. 기본 암호화 섹션까지 아래로 스크롤하여 암호화 유형 설정을 확인합니다.

AWS CLI를 사용하여 암호화 설정을 확인하려면 get-bucket-encryption 명령을 사용합니다.

객체의 암호화 상태를 확인하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 목록에서 객체가 포함된 버킷의 이름을 선택합니다.
4. 객체 목록에서 암호화를 추가하거나 변경할 객체의 이름을 선택합니다.

   객체 세부 정보 페이지가 나타납니다.
5. 서버 측 암호화 설정 섹션까지 아래로 스크롤하여 객체의 서버 측 암호화 설정을 확인합니다.

AWS CLI를 사용하여 객체 암호화 상태를 확인하려면 head-object 명령을 사용합니다.

암호화 및 권한 요구 사항

Amazon S3에서는 3가지 유형의 서버 측 암호화를 지원합니다.

- Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)
- AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 사용한 서버 측 암호화
- 고객 제공 키를 사용한 서버 측 암호화(SSE-C)

암호화 설정에 따라 다음 권한 요구 사항을 충족하는지 확인하세요.

- SSE-S3 - 추가 권한이 필요하지 않습니다.
- SSE-KMS(고객 관리형 키 사용) - 객체를 업로드하려면 AWS KMS key에 대한 `kms:GenerateDataKey` 권한이 필요합니다. 객체를 다운로드하고 객체의 멀티파트 업로드를 수행하려면 KMS 키에 대한 `kms:Decrypt` 권한이 필요합니다.
- SSE-KMS(AWS 관리형 키 사용) - 요청자는 `aws/s3` KMS 키를 소유한 계정과 동일한 계정을 사용해야 합니다. 또한 요청자는 객체에 액세스할 수 있는 올바른 Amazon S3 권한을 가지고 있어야 합니다.
- SSE-C(고객 제공 키 사용) - 추가 권한이 필요하지 않습니다. 버킷의 객체에 고객 제공 암호화 키를 사용하여 서버 측 암호화를 요구하고 제한하도록 버킷 정책을 구성할 수 있습니다.

객체가 고객 관리형 키로 암호화된 경우 KMS 키 정책에서 `kms:GenerateDataKey` 또는 `kms:Decrypt` 작업을 수행하도록 허용해야 합니다. KMS 키 정책을 확인하는 방법에 대한 지침은 AWS Key Management Service 개발자 안내서의 키 정책 보기를 참조하세요.

S3 객체 잠금 설정

버킷에 S3 객체 잠금이 활성화되어 있고 객체가 보존 기간 또는 법적 보존으로 보호되는 경우 객체를 삭제하려고 하면 Amazon S3에서 액세스 거부(403 금지) 오류를 반환합니다.

버킷에 객체 잠금이 활성화되어 있는지 확인하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 목록에서 검토하려는 버킷의 이름을 선택합니다.
4. 속성(Properties) 탭을 선택합니다.
5. 객체 잠금 섹션까지 아래로 스크롤합니다. 객체 잠금 설정이 활성화됨인지, 비활성화됨인지 확인합니다.

객체가 보존 기간 또는 법적 보존으로 보호되는지 확인하려면 해당 객체의 잠금 정보를 확인하세요.

객체가 보존 기간 또는 법적 보존으로 보호되는 경우 다음을 확인하세요.

- 객체 버전이 규정 준수 보존 모드로 보호되는 경우 영구적으로 삭제할 방법이 없습니다. AWS 계정 루트 사용자를 포함하여 요청자가 영구적 `DELETE` 요청을 수행하면 액세스 거부(403 금지) 오류가

발생합니다. 또한 규정 준수 보존 모드로 보호되는 객체에 대해 DELETE 요청을 제출하면 Amazon S3는 해당 객체에 삭제 마커를 생성한다는 점을 유의하시기 바랍니다.

- 객체 버전이 거버넌스 보존 모드로 보호되고 사용자에게 s3:BypassGovernanceRetention 권한이 있는 경우 보호를 우회하고 버전을 영구적으로 삭제할 수 있습니다. 자세한 내용은 거버넌스 모드 우회를 참조하세요.
- 객체 버전이 법적 보존으로 보호되는 경우 영구 DELETE 요청으로 인해 액세스 거부(403 금지) 오류가 발생할 수 있습니다. 객체 버전을 영구적으로 삭제하려면 객체 버전에 대한 법적 보존을 제거해야 합니다. 법적 보존을 제거하려면 s3:PutObjectLegalHold 권한이 있어야 합니다. 보존을 제거하는 방법에 대한 자세한 내용은 S3 객체 잠금 구성 섹션을 참조하세요.

VPC 엔드포인트 정책

가상 프라이빗 클라우드(VPC) 엔드포인트를 사용하여 Amazon S3에 액세스하는 경우 VPC 엔드포인트가 Amazon S3 리소스에 액세스하는 것을 차단하지 않는지 확인하세요. 기본적으로 VPC 엔드포인트 정책은 Amazon S3에 대한 모든 요청을 허용합니다. 특정 요청을 제한하도록 VPC 엔드포인트 정책을 구성할 수도 있습니다. VPC 엔드포인트 정책을 확인하는 방법에 대한 자세한 내용은 다음 리소스를 참조하세요.

- the section called “VPC 엔드포인트 정책으로 인한 액세스 거부 - 암시적 거부”
- the section called “VPC 엔드포인트 정책으로 인한 액세스 거부 - 명시적 거부”
- AWS PrivateLink 안내서의 엔드포인트 정책을 사용하여 VPC 엔드포인트로 액세스 제어

AWS Organizations 정책

AWS 계정이 조직에 속한 경우 AWS Organizations 정책에 따라 Amazon S3 리소스에 액세스하지 못할 수 있습니다. 기본적으로 AWS Organizations 정책은 Amazon S3에 대한 요청을 차단하지 않습니다. 하지만 AWS Organizations 정책이 S3 버킷에 대한 액세스를 차단하도록 구성되지 않았는지 확인하세요. AWS Organizations 정책을 확인하는 방법에 대한 지침은 다음 리소스를 참조하세요.

- the section called “서비스 제어 정책으로 인한 액세스 거부 - 암묵적 거부”
- the section called “서비스 제어 정책으로 인한 액세스 거부 - 명시적 거부”
- the section called “서비스 제어 정책으로 인한 액세스 거부 - 명시적 거부”
- AWS Organizations 사용 설명서의 모든 정책 나열

또한 S3 버킷에 대한 모든 사용자의 액세스를 거부하도록 멤버 계정의 버킷 정책을 잘못 구성할 경우 IAM에서 멤버 계정에 대한 권한 있는 세션을 시작하여 버킷을 잠금 해제할 수 있습니다. 권한 있는 세

션을 시작한 후에 잘못 구성된 버킷 정책을 삭제하여 버킷에 대한 액세스 권한을 다시 얻을 수 있습니다. 자세한 내용은 AWS Identity and Access Management 사용 설명서의 AWS Organizations 멤버 계정에서 권한 있는 작업 수행을 참조하세요.

CloudFront 배포 액세스

CloudFront를 통해 S3 정적 웹 사이트에 액세스하려고 할 때 Access Denied(403 Forbidden) 오류가 발생할 경우 다음과 같은 일반적인 문제를 확인합니다.

- 올바른 오리진 도메인 이름 형식을 사용하고 있나요?
  - REST API 엔드포인트 대신 S3 웹 사이트 엔드포인트 형식(bucket-name.s3-website-region.amazonaws.com)을 사용하고 있는지 확인합니다.
  - 버킷에서 정적 웹 사이트 호스팅이 활성화되어 있는지 확인합니다.
- 버킷 정책에서 CloudFront 액세스를 허용하나요?
  - 버킷 정책에 CloudFront 배포의 오리진 액세스 ID(OAI) 또는 오리진 액세스 제어(OAC)에 대한 권한이 포함되어 있는지 확인합니다.
  - 정책에 필요한 s3:GetObject 권한이 포함되어 있는지 확인합니다.

오류 페이지 설정과 프로토콜 설정을 포함한 추가 문제 해결 단계 및 구성은 AWS re:Post 지식 센터의 Amazon S3 웹사이트 엔드포인트를 CloudFront 배포의 오리진으로 사용할 때 “403 Access Denied” 오류가 발생하는 이유는 무엇입니까?를 참조하세요.

> **Note**
>
> 이 오류는 S3에 직접 액세스할 때 발생할 수 있는 403 오류와 다릅니다. CloudFront 관련 문제의 경우 CloudFront 배포 설정과 S3 구성을 모두 확인해야 합니다.

액세스 포인트 설정

Amazon S3 액세스 포인트를 통해 요청하는 동안 액세스 거부(403 금지) 오류가 발생하는 경우 다음을 확인해야 할 수 있습니다.

- 액세스 포인트의 구성
- 액세스 포인트에 사용되는 IAM 사용자 정책
- 크로스 계정 액세스 포인트를 관리하거나 구성하는 데 사용되는 버킷 정책

액세스 포인트 구성 및 정책

- 액세스 포인트를 만들 때 인터넷 또는 VPC를 네트워크 오리진으로 지정할 수 있습니다. 네트워크 오리진이 VPC로만 설정된 경우 Amazon S3는 지정된 VPC에서 시작되지 않은 액세스 포인트에 대한 모든 요청을 거부합니다. 액세스 포인트의 네트워크 오리진을 확인하려면 가상 프라이빗 클라우드(VPC)로 제한된 액세스 포인트 생성 섹션을 참조하세요.
- 액세스 포인트를 사용하여 사용자 지정 퍼블릭 액세스 차단 설정을 구성할 수도 있습니다. 이 설정은 버킷 또는 계정 수준의 퍼블릭 액세스 차단 설정과 유사하게 작동합니다. 사용자 지정 퍼블릭 액세스 차단 설정을 확인하려면 범용 버킷의 액세스 포인트에 대한 퍼블릭 액세스 관리 섹션을 참조하세요.
- 액세스 포인트를 사용하여 Amazon S3에 성공적으로 요청하려면 요청자에게 필요한 IAM 권한이 있는지 확인하세요. 자세한 내용은 액세스 포인트를 사용하도록 IAM 정책 구성 섹션을 참조하세요.
- 요청에 크로스 계정 액세스 포인트가 포함된 경우, 버킷 소유자가 액세스 포인트에서 온 요청을 승인하도록 버킷 정책을 업데이트했는지 확인하세요. 자세한 내용은 크로스 계정 액세스 포인트에 대한 권한 부여 섹션을 참조하세요.

이 주제의 모든 항목을 확인한 후에도 액세스 거부(403 금지) 오류가 계속 발생하는 경우 Amazon S3 요청 ID를 검색하고 지원에 추가 지침을 요청하세요.

추가 리소스

Access Denied(403 Forbidden) 오류에 대한 자세한 지침은 다음 리소스를 참조합니다.

- AWS re:Post 지식 센터의 Amazon S3의 403 액세스 거부 오류를 해결하려면 어떻게 해야 합니까?
- AWS re:Post 지식 센터의 Amazon S3에 파일을 업로드하려고 할 때 “403 Forbidden” 오류가 발생하는 이유는 무엇인가요?
- AWS re:Post 지식 센터의 리소스 모두가 동일한 AWS 계정에 있는 Amazon S3 버킷에서 발생하는 403 액세스 거부 오류를 해결하려면 어떻게 해야 하나요?
- AWS re:Post 지식 센터의 퍼블릭 읽기 액세스 권한이 있는 Amazon S3 버킷에서 발생하는 403 Access Denied 오류를 해결하려면 어떻게 해야 하나요?
- AWS re:Post 지식 센터의 Amazon S3에서 미리 서명된 URL 요청에 대한 "SignatureDoesNotMatch" 또는 "403 Forbidden" 오류를 해결하려면 어떻게 해야 합니까?
- AWS re:Post 지식 센터의 Amazon S3 버킷에서 동기화 명령을 실행할 때 ListObjectsV2에 대한 액세스 거부됨 오류가 발생하는 이유는 무엇인가요?
- AWS re:Post 지식 센터의 Amazon S3 웹사이트 엔드포인트를 CloudFront 배포의 오리진으로 사용할 때 “403 Access Denied” 오류가 발생하는 이유는 무엇입니까?

Amazon S3용 AWS 관리형 정책

AWS 관리형 정책은 AWS에 의해 생성되고 관리되는 독립 실행형 정책입니다. AWS 관리형 정책은 사용자, 그룹 및 역할에 권한 할당을 시작할 수 있도록 많은 일반 사용 사례에 대한 권한을 제공하도록 설계되었습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있기 때문에 특정 사용 사례에 대해 최소 권한을 부여하지 않을 수 있습니다. 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다.

AWS 관리형 정책에서 정의한 권한은 변경할 수 없습니다. 만약 AWS가 AWS 관리형 정책에 정의된 권한을 업데이트할 경우 정책이 연결되어 있는 모든 위탁자 ID(사용자, 그룹 및 역할)에도 업데이트가 적용됩니다. 새 AWS 서비스을(를) 시작하거나 새 API 작업을 기존 서비스에 이용하는 경우, AWS가 AWS 관리형 정책을 업데이트할 가능성이 높습니다.

자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하세요.

AWS 관리형 정책: AmazonS3FullAccess

`AmazonS3FullAccess` 정책을 IAM 보안 인증에 연결할 수 있습니다. 이 정책은 Amazon S3에 대한 전체 액세스를 허용하는 권한을 부여합니다.

이 정책의 권한을 보려면 AWS Management Console에서 AmazonS3FullAccess를 참조하세요.

AWS 관리형 정책: AmazonS3ReadOnlyAccess

`AmazonS3ReadOnlyAccess` 정책을 IAM 보안 인증에 연결할 수 있습니다. 이 정책은 Amazon S3에 대한 읽기 전용 액세스를 허용하는 권한을 부여합니다.

이 정책의 권한을 보려면 AWS Management Console에서 AmazonS3ReadOnlyAccess를 참조하세요.

AWS 관리형 정책: AmazonS3ObjectLambdaExecutionRolePolicy

S3 객체 Lambda 액세스 포인트에 요청이 있을 때 S3 객체 Lambda에 데이터를 보내는 데 필요한 권한을 AWS Lambda 함수에 제공합니다. 또한 Amazon CloudWatch Logs에 쓸 수 있는 Lambda 권한을 부여합니다.

이 정책의 권한을 보려면 AWS Management Console에서 AmazonS3ObjectLambdaExecutionRolePolicy를 참조하세요.

AWS 관리형 정책: S3UnlockBucketPolicy

S3 버킷에 대한 모든 사용자의 액세스를 거부하도록 멤버 계정의 버킷 정책을 잘못 구성할 경우 이 AWS 관리형 정책(S3UnlockBucketPolicy)을 사용하여 버킷을 잠금 해제할 수 있습니다. 모든 보안 주체의 Amazon S3 버킷에 대한 액세스를 거부하도록 잘못 구성된 버킷 정책을 제거하는 방법을 자세히 확인하려면 AWS Identity and Access Management 사용 설명서의 AWS Organizations 멤버 계정에서 권한 있는 작업 수행을 참조하세요.

AWS 관리형 정책에 대한 Amazon S3 업데이트

이 서비스가 이러한 변경 내용을 추적하기 시작한 이후부터 Amazon S3의 AWS 관리형 정책 업데이트에 대한 세부 정보를 봅니다.

변경 사항	설명	날짜
Amazon S3에 S3UnlockBucketPolicy 추가됨	Amazon S3에 버킷을 잠금 해제하고 모든 보안 주체의 Amazon S3 버킷에 대한 액세스를 거부하도록 잘못 구성된 버킷 정책을 제거하는 새로운 AWS 관리형 정책(S3UnlockBucketPolicy)이 추가되었습니다.	2024년 11월 1일
Amazon S3에서 AmazonS3ReadOnlyAccess 에 Describe 권한이 추가됨	Amazon S3에서 AmazonS3ReadOnlyAccess 에 s3:Describe* 권한이 추가되었습니다.	2023년 8월 11일
Amazon S3에서는 S3 객체 Lambda 권한을 AmazonS3FullAccess 및 AmazonS3ReadOnlyAccess 에 추가했습니다.	Amazon S3에서는 S3 객체 Lambda에 대한 권한을 포함하는 AmazonS3FullAccess 및 AmazonS3ReadOnlyAccess 정책을 업데이트했습니다.	2021년 9월 27일
Amazon S3에 AmazonS3ObjectLambdaExecutionRolePolicy 추가됨	Amazon S3에 S3 객체 Lambda와 상호 작용하고 CloudWatch Logs에 쓸 수 있	2021년 8월 18일

변경 사항	설명	날짜
	는 Lambda 함수 권한을 제공하는 `AmazonS3ObjectLambdaExecutionRolePolicy` 라는 새로운 AWS 관리형 정책이 추가되었습니다.	
Amazon S3 변경 사항 추적 시작	Amazon S3가 AWS 관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2021년 8월 18일

액세스 포인트로 공유 데이터세트에 대한 액세스 관리

Amazon S3 액세스 포인트는 모든 AWS서비스 또는 S3에 데이터를 저장하는 고객 애플리케이션에 대한 데이터 액세스를 간소화합니다. 액세스 포인트는 버킷 또는 Amazon FSx for OpenZFS 볼륨과 같은 데이터 소스에 연결된 명명된 네트워크 엔드포인트입니다. 버킷 작업에 대한 자세한 내용은 범용 버킷 개요 단원을 참조하십시오. FSx for OpenZFS 작업에 대한 자세한 내용은 OpenZFS 사용 설명서의 What is Amazon FSx for OpenZFS를 참조하세요.

액세스 포인트를 사용하여 `GetObject` 및 `PutObject` 같은 S3 객체 작업을 수행할 수 있습니다. 각 액세스 포인트에는 해당 액세스 포인트를 통해 이루어진 모든 요청에 대해 S3가 적용하는 고유한 권한 및 네트워크 제어가 있습니다. 각 엔드포인트는 리소스, 사용자 또는 기타 조건에서 사용을 제어할 수 있는 사용자 지정 액세스 포인트 정책을 강제 적용합니다. 액세스 포인트가 버킷에 연결되어 있으면 액세스 포인트 정책은 기본 버킷 정책과 함께 작동합니다. 가상 프라이빗 클라우드(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 삭제 또는 S3 복제 구성 생성과 같은 다른 Amazon S3 작업을 수행할 수 없습니다. 액세스 포인트를 지원하는 전체 S3 작업 목록은 액세스 포인트 호환성 단원을 참조하십시오.

이 섹션의 주제에서는 Amazon S3 액세스 포인트로 작업하는 방법에 대해 설명합니다. 디렉터리 버킷에서 액세스 포인트를 사용하는 방법에 대한 주제는 액세스 포인트가 있는 디렉터리 버킷에서 공유 데이터세트 액세스 관리 섹션을 참조하세요.

주제

- 액세스 포인트 이름 지정 규칙, 제한 및 한계
- ARN, 액세스 포인트 별칭 또는 가상 호스팅 방식 URI를 사용하여 액세스 포인트 참조
- 액세스 포인트 호환성
- 액세스 포인트를 사용하도록 IAM 정책 구성
- 액세스 포인트 모니터링 및 로깅
- 액세스 포인트 생성
- 범용 버킷의 Amazon S3 Access Points 관리
- 범용 버킷의 Amazon S3 Access Points 사용

액세스 포인트 이름 지정 규칙, 제한 및 한계

액세스 포인트는 데이터 관리를 간소화하는 Amazon FSx 파일 시스템의 버킷 또는 볼륨에 연결된 명명된 네트워크 엔드포인트입니다. 액세스 포인트를 만들 때 이름과 액세스 포인트를 만들 AWS 리전을 선택합니다. 이어지는 주제에서는 액세스 포인트 이름 지정 규칙, 제한 및 한계에 대한 정보를 제공합니다.

주제

- 액세스 포인트 이름 지정 규칙
- 액세스 포인트 제한 및 한계
- Amazon FSx 파일 시스템의 볼륨에 연결된 액세스 포인트에 대한 제한 및 한계

액세스 포인트 이름 지정 규칙

액세스 포인트를 만들 때 이름과 액세스 포인트가 만들어질 AWS 리전을 선택합니다. 범용 버킷과 달리 액세스 포인트 이름은 AWS 계정 또는 AWS 리전에서 고유할 필요가 없습니다. 동일한 AWS 계정에서 서로 다른 AWS 리전에 동일한 이름의 액세스 포인트를 생성할 수도 있고, 서로 다른 두 에AWS 계정서 동일한 액세스 포인트 이름을 사용할 수도 있습니다. 그러나 단일 AWS 리전 내에서는 AWS 계정에 이름이 동일한 두 개의 액세스 포인트가 있을 수 없습니다.

> **Note**
>
> 액세스 포인트 이름을 공개하기로 선택한 경우 액세스 포인트 이름에 민감한 정보를 포함하지 마세요. 도메인 이름 시스템(DNS)이라고 하는 공개적으로 액세스할 수 있는 데이터베이스에 액세스 포인트 이름이 게시됩니다.

액세스 포인트 이름은 DNS를 준수해야 하며 다음과 같은 조건을 만족해야 합니다.

- 단일 AWS 계정 및 AWS 리전 내에서 고유해야 함
- 숫자 또는 소문자로 시작해야 함
- 3~50자 이내여야 함
- 하이픈(-)으로 시작하거나 끝날 수 없음
- 밑줄(_), 대문자, 공백 또는 마침표(.)를 포함할 수 없음
- 접미사 `-s3alias` 또는 `-ext-s3alias`로 끝날 수 없습니다. 이러한 접미사는 액세스 포인트 별칭 이름 용도로 예약되어 있습니다. 자세한 내용은 액세스 포인트 별칭 섹션을 참조하세요.

액세스 포인트 제한 및 한계

Amazon S3 액세스 포인트에는 다음과 같은 규제 및 제한이 있습니다.

- 각 액세스 포인트는 정확히 하나의 버킷 또는 FSx for OpenZFS 볼륨과 연결됩니다. 액세스 포인트를 생성할 때 이를 지정해야 합니다. 액세스 포인트를 생성한 후에는 다른 버킷 또는 FSx for OpenZFS 볼륨과 연결할 수 없습니다. 그러나 액세스 포인트를 삭제한 다음, 이름이 동일한 다른 액세스 포인트를 생성할 수 있습니다.
- 액세스 포인트를 생성한 후에는 해당 가상 프라이빗 클라우드(VPC) 구성을 변경할 수 없습니다.
- 액세스 포인트 정책은 크기가 20KB로 제한됩니다.
- AWS 리전마다 AWS 계정당 최대 1만 개의 액세스 포인트를 생성할 수 있습니다. 단일 리전에서 단일 계정에 대해 10,000개 이상의 액세스 포인트가 필요한 경우 서비스 할당량 증가를 요청할 수 있습니다. 서비스 할당량 및 증가 요청에 대한 자세한 내용은 AWS 일반 참조의 AWS 서비스 할당량을 참조하십시오.
- 액세스 포인트는 S3 복제의 대상으로 사용할 수 없습니다. 복제에 대한 자세한 내용은 리전 내 및 리전 간 객체 복제 섹션을 참조하십시오.
- Amazon S3 콘솔에서는 S3 액세스 포인트 별칭을 이동 작업의 원본 또는 대상으로 사용할 수 없습니다.

- 가상 호스트 스타일의 URL을 사용해야만 액세스 포인트의 주소를 지정할 수 있습니다. 가상 호스트 방식 주소 지정에 대한 자세한 내용은 Amazon S3 범용 버킷 액세스 단원을 참조하십시오.
- 액세스 포인트 기능을 제어하는 API 작업(예: `PutAccessPoint` 및 `GetAccessPointPolicy`)은 크로스 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 참조의 요청 인증(AWS 서명 버전 4)을 참조하십시오.
- 액세스 포인트는 HTTPS를 통한 액세스만 지원합니다. Amazon S3는 HTTP를 통해 이루어진 모든 요청에 대해 자동으로 HTTP 리디렉션으로 응답하여 요청을 HTTPS로 업그레이드합니다.
- 액세스 포인트는 익명 액세스를 지원하지 않습니다.
- 액세스 포인트를 생성한 후에는 퍼블릭 액세스 차단 설정을 변경할 수 없습니다.
- 버킷 소유자로부터 권한을 부여받기 전까지는 크로스 계정 액세스 포인트가 데이터에 대한 액세스 권한을 부여하지 않습니다. 버킷 소유자는 항상 데이터 액세스에 대한 최종 제어 권한을 보유하며, 크로스 계정 액세스 포인트로부터의 요청을 승인하도록 버킷 정책을 업데이트해야 합니다. 버킷 정책 예시를 보려면 액세스 포인트를 사용하도록 IAM 정책 구성 섹션을 참조하십시오.
- 1,000개 이상의 액세스 포인트가 있는 AWS 리전에서는 Amazon S3 콘솔에서 이름으로 액세스 포인트를 검색할 수 없습니다.
- Amazon S3 콘솔에서 크로스스 계정 액세스 포인트를 볼 때 액세스 열에 알 수 없음이 표시됩니다. Amazon S3 콘솔은 연결된 버킷 및 객체에 퍼블릭 액세스가 허용되는지 여부를 판단할 수 없습니다. 특정 사용 사례에 대해 퍼블릭 구성이 필요한 경우가 아니라면 사용자와 버킷 소유자가 액세스 포인트 및 버킷에 대한 모든 퍼블릭 액세스를 차단하는 것이 좋습니다. 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하세요.

Amazon FSx 파일 시스템의 볼륨에 연결된 액세스 포인트에 대한 제한 및 한계

다음은 Amazon FSx 파일 시스템의 볼륨에 연결된 액세스 포인트를 사용할 때 적용되는 구체적인 제한 사항입니다.

- 액세스 포인트를 생성할 때는 소유한 Amazon FSx 파일 시스템의 볼륨에만 액세스 포인트를 연결할 수 있습니다. 다른 AWS 계정이 소유한 볼륨에는 연결할 수 없습니다.
- 액세스 포인트를 생성하여 Amazon FSx 파일 시스템의 볼륨에 연결할 때는 `CreateAccessPoint` API를 사용할 수 없습니다. `CreateAndAttachS3AccessPoint` API를 사용해야 합니다.
- Amazon FSx 파일 시스템의 볼륨에 연결된 액세스 포인트를 생성하거나 사용할 때는 퍼블릭 액세스 차단 설정을 끌 수 없습니다.

- Amazon FSx 파일 시스템의 볼륨에 연결된 액세스 포인트를 사용하여 S3 콘솔에서 객체를 나열하거나 복사 또는 이동 작업을 사용할 수 없습니다.
- 소스와 대상이 동일한 액세스 포인트인 경우에만 FSx for OpenZFS 볼륨에 연결된 액세스 포인트에 대해 `CopyObject`가 지원됩니다. 액세스 포인트 호환성에 대한 자세한 내용은 액세스 포인트 호환성 섹션을 참조하세요.
- 멀티파트 업로드는 5GB로 제한됩니다.
- FSx for OpenZFS 배포 유형 및 스토리지 클래스 지원은 AWS 리전에 따라 다릅니다. 자세한 내용은 OpenZFS 사용 설명서의 Availability by AWS 리전를 참조하세요.

ARN, 액세스 포인트 별칭 또는 가상 호스팅 방식 URI를 사용하여 액세스 포인트 참조

액세스 포인트를 생성한 후 이러한 엔드포인트를 사용하여 여러 작업을 미리 구성할 수 있습니다. 액세스 포인트를 참조할 때 Amazon 리소스 이름(ARN), 액세스 포인트 별칭 또는 가상 호스팅 방식 URI를 사용할 수 있습니다.

주제

- 액세스 포인트 ARN
- 액세스 포인트 별칭
- 가상 호스팅 방식 URI

액세스 포인트 ARN

액세스 포인트에 Amazon 리소스 이름(ARN)이 있습니다. 액세스 포인트 ARN은 버킷 ARN과 비슷하지만 명시적으로 입력되고 액세스 포인트의 AWS 리전 및 액세스 포인트 소유자의 AWS 계정 ID를 인코딩합니다. ARN에 대한 자세한 내용은 IAM 사용 설명서의 Amazon 리소스 이름(ARN)으로 AWS 리소스를 식별합니다를 참조하세요.

액세스 포인트 ARN은 다음 형식을 사용합니다.

```
arn:aws:s3:region:account-id:accesspoint/resource
```

- `arn:aws:s3:us-west-2:123456789012:accesspoint/test`는 *us-west-2* 리전의 계정 *123456789012*에서 소유한 *test*라는 이름의 액세스 포인트를 나타냅니다.

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

액세스 포인트를 통해 액세스하는 객체의 ARN은 다음 형식을 사용합니다.

```
arn:aws:s3:region:account-id:accesspoint/access-point-name/object/resource
```

- arn:aws:s3:*us-west-2*:*123456789012*:accesspoint/*test*/object/*unit-01*은 *us-west-2* 리전의 계정 *123456789012*에서 소유한 *test*라는 이름의 액세스 포인트를 통해 액세스되는 *unit-01* 객체를 나타냅니다.
- arn:aws:s3:*us-west-2*:*123456789012*:accesspoint/*test*/object/*는 *us-west-2* 리전의 계정 *123456789012*에서 *test*라는 이름의 액세스 포인트에 대한 모든 객체를 나타냅니다.
- arn:aws:s3:*us-west-2*:*123456789012*:accesspoint/*test*/object/*unit-01*/*finance*/*는 *us-west-2* 리전의 계정 *123456789012*에서 *test*라는 이름의 액세스 포인트에 대한 접두사 *unit-01/finance/*에 해당하는 모든 객체를 나타냅니다.

액세스 포인트 별칭

액세스 포인트를 생성하면 Amazon S3가 자동으로 데이터 액세스에 버킷 이름 대신 사용할 수 있는 별칭을 생성합니다. 액세스 지점 데이터 영역 작업에 Amazon 리소스 이름(ARN) 대신 이 액세스 지점 별칭을 사용할 수 있습니다. 해당 작업의 목록은 액세스 포인트 호환성을(를) 참조하십시오.

액세스 포인트 별칭 이름은 Amazon S3 버킷과 동일한 네임스페이스 내에서 생성됩니다. 이 별칭 이름은 자동으로 생성되며 변경할 수 없습니다. 액세스 포인트 별칭 이름은 유효한 Amazon S3 버킷 이름의 모든 요구 사항을 충족하며 다음의 파트로 구성됩니다.

ACCESS POINT NAME-METADATA-s3alias(Amazon S3 버킷에 연결된 액세스 포인트의 경우)

ACCESS POINT NAME-METADATA-ext-s3alias(S3 버킷이 아닌 데이터 소스에 연결된 액세스 포인트의 경우)

> **Note**
>
> -s3alias 및 -ext-s3alias 접미사는 액세스 포인트 별칭 이름용으로 예약되어 있으며 버킷 또는 액세스 포인트 이름에는 사용할 수 없습니다. Amazon S3 버킷 이름 지정 규칙에 대한 자세한 내용은 범용 버킷 이름 지정 규칙 섹션을 참조하십시오.

액세스 포인트 별칭 사용 사례 및 제한

액세스 포인트를 채택할 때 광범위한 코드 변경을 하지 않고 액세스 포인트 별칭 이름을 사용할 수 있습니다.

액세스 포인트를 생성하면 Amazon S3는 다음 예와 같이 액세스 포인트 별칭 이름을 자동으로 생성합니다. 이 명령을 실행하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws s3control create-access-point --bucket amzn-s3-demo-bucket1 --name my-access-point
 --account-id 111122223333
{
    "AccessPointArn": "arn:aws:s3:region:111122223333:accesspoint/my-access-point",
    "Alias": "my-access-point-aqfqprnstn7aefdfbarligizwgyfouse1a-s3alias"
}
```

모든 데이터 영역 작업에 Amazon S3 버킷 이름 대신 이 액세스 포인트 별칭을 사용할 수 있습니다. 해당 작업의 목록은 액세스 포인트 호환성을(를) 참조하세요.

`get-object` 명령에 대한 다음 AWS CLI 예시에서는 버킷의 액세스 지점 별칭을 사용하여 지정된 객체에 대한 정보를 반환합니다. 이 명령을 실행하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws s3api get-object --bucket my-access-point-aqfqprnstn7aefdfbarligizwgyfouse1a-
s3alias --key dir/my_data.rtf my_data.rtf

{
    "AcceptRanges": "bytes",
    "LastModified": "2020-01-08T22:16:28+00:00",
    "ContentLength": 910,
    "ETag": "\"00751974dc146b76404bb7290f8f51bb\"",
    "VersionId": "null",
    "ContentType": "text/rtf",
    "Metadata": {}
}
```

액세스 포인트 별칭 제한 사항

- 별칭은 고객이 구성할 수 없습니다.
- 액세스 지점에서 별칭을 삭제, 수정 또는 비활성화할 수 없습니다.
- 일부 데이터 영역 작업에 Amazon S3 버킷 이름 대신 이 액세스 지점 별칭을 사용할 수 있습니다. 해당 작업의 목록은 S3 작업과의 액세스 포인트 호환성을(를) 참조하십시오.

- Amazon S3 제어 영역 작업에는 액세스 포인트 별칭 이름을 사용할 수 없습니다. Amazon S3 제어 영역 작업 목록은 Amazon Simple Storage Service API 참조에서 Amazon S3 제어를 참조하십시오.
- Amazon S3 콘솔에서는 S3 액세스 포인트 별칭을 이동 작업의 원본 또는 대상으로 사용할 수 없습니다.
- 별칭은 AWS Identity and Access Management(IAM) 정책에서 사용할 수 없습니다.
- 별칭은 S3 서버 액세스 로그의 로깅 대상으로 사용할 수 없습니다.
- 별칭은 AWS CloudTrail 로그의 로깅 대상으로 사용할 수 없습니다.
- Amazon SageMaker AI GroundTruth는 액세스 포인트 별칭을 지원하지 않습니다.

가상 호스팅 방식 URI

액세스 포인트는 가상 호스트 스타일의 주소 지정만 지원합니다. 가상 호스팅 방식 URI에서 액세스 포인트 이름, AWS 계정, AWS 리전은 URL에서 도메인 이름의 일부입니다. 가상 호스팅에 대한 자세한 내용은 범용 버킷의 가상 호스팅 섹션을 참조하세요.

액세스 포인트에 대한 가상 호스팅 방식 URI는 다음 형식을 사용합니다.

```
https://access-point-name-account-id.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 계정 *123456789012*에서 소유한 *finance-docs*라는 이름의 액세스 포인트를 사용하려면 적절한 URL은 `https://finance-docs-123456789012.s3-accesspoint.us-west-2.amazonaws.com`입니다.
> - S3 액세스 포인트는 HTTP를 통한 액세스를 지원하지 않습니다. 액세스 포인트는 HTTPS를 통한 보안 액세스만 지원합니다.

액세스 포인트 호환성

액세스 포인트를 사용하여 다음 Amazon S3 API 하위 집합을 사용하는 객체에 액세스할 수 있습니다. 아래 나열된 모든 작업은 액세스 포인트 ARN 또는 액세스 포인트 별칭을 수락할 수 있습니다.

액세스 포인트를 사용하여 객체에 대한 작업을 수행하는 예시는 범용 버킷의 Amazon S3 Access Points 사용 섹션을 참조하세요.

S3 작업과의 액세스 포인트 호환성

다음 표는 Amazon S3 작업 중 일부와 액세스 포인트와의 호환 여부를 나열한 목록입니다. 아래의 모든 작업은 S3 버킷을 데이터 소스로 사용하는 액세스 포인트에서 지원되지만 FSx for OpenZFS 볼륨을 데이터 소스로 사용하는 액세스 포인트에서는 일부 작업만 지원됩니다.

자세한 내용은 FSx for OpenZFS 사용 설명서의 액세스 포인트 호환성을 참조하세요.

S3 작업	S3 버킷에 연결된 액세스 포인트	FSx for OpenZFS 볼륨에 연결된 액세스 포인트
AbortMultipartUpload	지원	지원
CompleteMultipartUpload	지원	지원
CopyObject (동일한 리전 사본만 해당)	지원	소스와 대상이 동일한 액세스 포인트인 경우 지원됨
CreateMultipartUpload	지원	지원
DeleteObject	지원	지원
DeleteObjects	지원	지원
DeleteObjectTagging	지원	지원
GetBucketAcl	지원	지원되지 않음
GetBucketCors	지원	지원되지 않음
GetBucketLocation	지원	지원
GetBucketNotificationConfiguration	지원	지원되지 않음
GetBucketPolicy	지원	지원되지 않음

S3 작업	S3 버킷에 연결된 액세스 포인트	FSx for OpenZFS 볼륨에 연결된 액세스 포인트
GetObject	지원	지원
GetObjectAcl	지원	지원되지 않음
GetObjectAttributes	지원	지원
GetObjectLegalHold	지원	지원되지 않음
GetObjectRetention	지원	지원되지 않음
GetObjectTagging	지원	지원
HeadBucket	지원	지원
HeadObject	지원	지원
ListMultipartUploads	지원	지원
ListObjects	지원	지원
ListObjectsV2	지원	지원
ListObjectVersions	지원	지원되지 않음
ListParts	지원	지원
Presign	지원	지원
PutObject	지원	지원
PutObjectAcl	지원	지원되지 않음
PutObjectLegalHold	지원	지원되지 않음
PutObjectRetention	지원	지원되지 않음
PutObjectTagging	지원	지원
RestoreObject	지원	지원되지 않음

S3 작업	S3 버킷에 연결된 액세스 포인트	FSx for OpenZFS 볼륨에 연결된 액세스 포인트
UploadPart	지원	지원
UploadPartCopy (동일한 리전 사본만 해당)	지원	소스와 대상이 동일한 액세스 포인트인 경우 지원됨

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 또는 Amazon FSx 파일 시스템에서 요청을 허용해야 합니다.

> **Important**
>
> 액세스 포인트 정책에 포함시키는 제한은 해당 액세스 포인트를 통해 이루어진 요청에만 적용됩니다. 버킷에 액세스 포인트를 연결해도 기본 리소스의 동작은 변경되지 않습니다. 액세스 포인트를 통해 수행되지 않은 버킷에 대한 모든 기존 작업은 이전과 같이 계속 작동합니다.

IAM 리소스 정책을 사용할 때는 AWS Identity and Access Management Access Analyzer의 보안 경고, 오류, 일반 경고 및 제안 사항을 해결한 후 정책을 저장해야 합니다. IAM Access Analyzer는 정책 확인은 실행하여 IAM 정책 문법 및 모범 사례에 대해 정책을 검증합니다. 이러한 확인은 결과를 생성하고 보안 모범 사례를 준수하고 작동하는 정책을 작성하는 데 도움이 되는 권장 사항을 제공합니다.

IAM Access Analyzer를 사용한 정책 검증에 대한 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer 정책 검증을 참조하십시오. IAM Access Analyzer에서 반환된 경고, 오류 및 제안 사항 목록을 보려면 IAM Access Analyzer 정책 확인 참조를 참조하십시오.

액세스 포인트의 정책 예시

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

> **Note**
>
> 액세스 포인트 정책에 부여된 권한은 기본 버킷이 동일한 액세스를 허용하는 경우에만 유효합니다. 다음 두 가지 방법으로 이 작업을 수행할 수 있습니다.
>
> 1. (권장) 액세스 포인트에 액세스 제어 위임에 설명된 대로 버킷의 액세스 제어를 액세스 포인트에 위임합니다.
> 2. 액세스 포인트 정책에 포함된 동일한 권한을 기본 버킷의 정책에 추가합니다. 예제 1 액세스 포인트 정책 예제는 기본 버킷 정책을 수정하여 필요한 액세스를 허용하는 방법을 보여줍니다.

Example 1 - 액세스 포인트 정책 권한 부여

다음 액세스 포인트 정책은 계정 *Jane* 내의 IAM 사용자 *123456789012*에게 계정 GET 내의 액세스 포인트 PUT를 통해 접두사 *Jane/*를 포함하는 *my-access-point* 및 *123456789012* 객체에 대한 권한을 부여합니다.

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

> **Note**
>
> 액세스 포인트 정책이 *Jane*에 대한 액세스 권한을 효과적으로 부여하려면 기본 버킷에서도 *Jane*에 대해 동일한 액세스를 허용해야 합니다. 액세스 포인트에 액세스 제어 위임에 설명된 대로 버킷의 액세스 제어를 액세스 포인트에 위임할 수 있습니다. 또는 다음 정책을 기본 버킷에 추가하여 Jane에게 필요한 권한을 부여할 수 있습니다. 액세스 포인트와 버킷 정책 간에 `Resource` 항목이 다릅니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Jane"
        },
        "Action": ["s3:GetObject", "s3:PutObject"],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/Jane/*"
    }]
}
```

Example 2 - 태그 조건이 있는 액세스 포인트 정책

다음 액세스 포인트 정책은 계정 *123456789012* 내의 IAM 사용자 *Mateo*에게 태그 키 *data*가 *finance* 값으로 설정된 계정 *123456789012* 내의 액세스 포인트 *my-access-point*를 통해 GET 객체에 대한 권한을 부여합니다.

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

Example 3 - 버킷 목록 조회를 허용하는 액세스 포인트 정책

다음 액세스 포인트 정책은 계정 *123456789012* 내의 IAM 사용자 Arnav가 계정 *123456789012* 내의 버킷 기본 액세스 포인트 *my-access-point*에 포함된 객체를 볼 수 있는 권한을 허용합니다.

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

Example 4 - 서비스 제어 정책

다음 서비스 제어 정책에서는 가상 프라이빗 클라우드(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 5 - S3 작업을 VPC 네트워크 오리진으로 제한하는 버킷 정책

다음 버킷 정책은 버킷 *amzn-s3-demo-bucket*에 대한 모든 S3 객체 작업에 대한 액세스를 네트워크 오리진이 VPC인 액세스 포인트로 제한합니다.

> **Important**
>
> 이 예제에서 보이는 것과 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:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:AccessPointNetworkOrigin": "VPC"
                }
            }
        }
    ]
}
```

조건 키

S3 액세스 포인트는 리소스에 대한 액세스를 제어하기 위해 IAM 정책에서 사용할 수 있는 조건 키를 사용합니다. 다음 조건 키는 IAM 정책의 일부만 나타냅니다. 정책 예제를 모두 확인하려면 액세스 포인트의 정책 예시, the section called “액세스 포인트에 액세스 제어 위임”, the section called “크로스 계정 액세스 포인트에 대한 권한 부여” 페이지를 참조하십시오.

s3:DataAccessPointArn

이 예제는 액세스 포인트 ARN과 일치시키는 데 사용할 수 있는 문자열을 보여줍니다. 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

이 예제는 액세스 포인트 소유자의 계정 ID와 일치시키는 데 사용할 수 있는 문자열 연산자를 보여줍니다. 다음 예제는 AWS 계정 *123456789012*가 소유한 모든 액세스 포인트와 일치합니다.

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

s3:AccessPointNetworkOrigin

이 예제는 네트워크 오리진 Internet 또는 VPC와 일치시키는 데 사용할 수 있는 문자열 연산자를 보여줍니다. 다음 예제에서는 오리진이 VPC인 액세스 포인트와만 일치합니다.

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

Amazon S3에서 조건 키를 사용하는 방법에 대한 자세한 내용은 서비스 승인 참조에서 Amazon S3에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

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 6 - 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" }
        }
    }]
}
```

크로스 계정 액세스 포인트에 대한 권한 부여

다른 계정이 소유한 버킷에 액세스 포인트를 생성하려면 먼저 버킷 이름과 계정 소유자 ID를 지정하여 액세스 포인트를 만들어야 합니다. 그런 다음, 액세스 포인트로부터의 요청을 승인하도록 버킷 정책을 버킷 소유자가 업데이트해야 합니다. 액세스 포인트를 생성하는 것은 액세스 포인트가 버킷 콘텐츠에 대한 액세스를 제공하지 않는다는 점에서 DNS CNAME을 생성하는 것과 유사합니다. 모든 버킷 액세스는 버킷 정책에 의해 제어됩니다. 다음 예제 버킷 정책은 신뢰할 수 있는 AWS 계정이 소유한 액세스 포인트에서 버킷의 `GET` 및 `LIST` 요청을 허용합니다.

*`Bucket ARN`*을 버킷의 ARN으로 바꿉니다.

Example 7 - 다른 AWS 계정에 권한을 위임하는 버킷 정책

```
{
    "Version": "2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS": "*" },
        "Action" : ["s3:GetObject","s3:ListBucket"],
        "Resource" : [ "Bucket ARN", "Bucket ARN/*"],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointAccount" : "Access point owner's
 account ID" }
        }
    }]
}
```

> **Note**
>
> 교차 계정 액세스 포인트는 S3 버킷에 연결된 액세스 포인트에만 사용할 수 있습니다. 다른 AWS 계정이 소유한 Amazon FSx 파일 시스템의 볼륨에는 액세스 포인트를 연결할 수 없습니다.

액세스 포인트 모니터링 및 로깅

Amazon S3는 액세스 포인트를 통해 이루어진 요청과 액세스 포인트를 관리하는 API 작업에 대한 요청(예: `CreateAccessPoint` 및 `GetAccessPointPolicy`)을 로그로 기록합니다. 사용 패턴을 모니

터링하고 관리하기 위해 액세스 포인트에 대한 Amazon CloudWatch Logs 요청 지표를 구성할 수도 있습니다.

주제

- CloudWatch 요청 지표
- 액세스 포인트를 통해 이루어진 요청에 대한 AWS CloudTrail 로그

CloudWatch 요청 지표

액세스 포인트를 사용하는 애플리케이션의 성능을 이해하고 개선하기 위해 Amazon S3 용 CloudWatch 요청 지표를 사용할 수 있습니다. 요청 지표를 사용하면 운영 문제를 신속하게 확인하여 조치하기 위해 Amazon S3 요청을 모니터링합니다.

기본값으로 요청 지표가 버킷 수준에서 제공됩니다. 그러나 공유 접두사, 개체 태그 또는 액세스 포인트를 사용하여 요청 지표에 대한 필터를 정의할 수 있습니다. 액세스 포인트 필터를 생성할 때 요청 지표 구성에는 지정한 액세스 포인트에 대한 요청이 포함됩니다. 지표를 수신하고 경보를 설정하며 대시보드에 액세스하여 이 액세스 포인트를 통해 수행된 실시간 작업을 확인할 수 있습니다.

콘솔에서 또는 Amazon S3 API를 사용해 요청 지표를 구성하여 이 지표를 선택해야 합니다. 요청 지표는 약간의 처리 지연 시간 후에 1분 간격으로 제공됩니다. 요청 지표는 CloudWatch 사용자 지정 지표와 동일한 요금으로 청구됩니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

액세스 포인트를 기준으로 필터링하는 요청 지표 구성을 생성하려면 접두사, 객체 태그 또는 액세스 포인트로 필터링하는 지표 구성 생성 섹션을 참조하십시오.

액세스 포인트를 통해 이루어진 요청에 대한 AWS CloudTrail 로그

서버 액세스 로깅 및 AWS CloudTrail을 사용하여 액세스 포인트를 통해 이루어진 요청과 액세스 포인트를 관리하는 API에 대한 요청(예: `CreateAccessPoint` 및 `GetAccessPointPolicy`,)을 로그로 기록할 수 있습니다.

액세스 포인트를 통해 수행된 요청에 대한 CloudTrail 로그 항목에는 로그의 `resources` 섹션에 액세스 포인트 ARN이 포함됩니다.

예를 들어, 다음과 같은 구성이 있다고 가정합니다.

- *`my-image.jpg`*라는 객체를 포함하는 *`us-west-2`* 리전의 *`amzn-s3-demo-bucket1`*이라는 버킷
- *`amzn-s3-demo-bucket1`*과 연결된 이름이 *`my-bucket-ap`*인 액세스 포인트
- *`123456789012`*의 AWS 계정 ID

다음 예제에서는 이전 구성에 대한 CloudTrail 로그 항목의 `resources` 섹션을 보여줍니다.

```
"resources": [
        {"type": "AWS::S3::Object",
            "ARN": "arn:aws:s3:::amzn-s3-demo-bucket1/my-image.jpg"
        },
        {"accountId": "123456789012",
            "type": "AWS::S3::Bucket",
            "ARN": "arn:aws:s3:::amzn-s3-demo-bucket1"
        },
        {"accountId": "123456789012",
            "type": "AWS::S3::AccessPoint",
            "ARN": "arn:aws:s3:us-west-2:123456789012:accesspoint/my-bucket-ap"
        }
    ]
```

Amazon FSx 파일 시스템의 볼륨에 연결된 액세스 포인트를 사용하는 경우 CloudTrail 로그 항목의 `resources` 섹션이 다르게 표시됩니다. 예시:

```
"resources": [
        {
            "accountId": "123456789012",
            "type": "AWS::FSx::Volume",
            "ARN": "arn:aws:fsx:us-east-1:123456789012:volume/fs-0123456789abcdef9/
fsvol-01234567891112223"
        }
    ]
```

S3 서버 액세스 로그에 대한 자세한 내용은 서버 액세스 로깅을 사용한 요청 로깅 단원을 참조하십시오. AWS CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 AWS CloudTrail이란 무엇입니까?를 참조하십시오.

액세스 포인트 생성

AWS Management Console, AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하여 S3 액세스 포인트를 생성할 수 있습니다. 액세스 포인트는 버킷 또는 Amazon FSx for OpenZFS 볼륨과 같은 데이터 소스에 연결된 명명된 네트워크 엔드포인트입니다.

기본적으로 각 AWS 계정에 대해 리전당 최대 10,000개의 액세스 포인트를 생성할 수 있습니다. 단일 리전에서 단일 계정에 대해 10,000개 이상의 액세스 포인트가 필요한 경우 서비스 할당량 증가를 요청

할 수 있습니다. 서비스 할당량 및 증가 요청에 대한 자세한 내용은 AWS 일반 참조의 AWS 서비스 할당량을 참조하십시오.

주제

- S3 버킷이 있는 액세스 포인트 생성
- Amazon FSx를 사용하여 액세스 포인트 생성
- 가상 프라이빗 클라우드(VPC)로 제한된 액세스 포인트 생성
- 범용 버킷의 액세스 포인트에 대한 퍼블릭 액세스 관리

S3 버킷이 있는 액세스 포인트 생성

액세스 포인트는 범용 버킷의 Amazon S3 버킷 정확히 하나와 연결됩니다. AWS 계정 내에서 버킷을 사용하려면 먼저 버킷을 생성해야 합니다. 버킷 생성에 대한 자세한 내용은 Amazon S3 범용 버킷 생성, 구성 및 작업 섹션을 참조하십시오.

또한, 버킷 이름과 버킷 소유자의 계정 ID를 알고 있으면 다른 AWS 계정 내에 있는 버킷과 연결된 크로스 계정 액세스 포인트를 생성할 수 있습니다. 하지만 크로스 계정 액세스 포인트를 생성해도 버킷 소유자로부터 권한을 부여받기 전까지는 버킷의 데이터에 대한 액세스 권한이 부여되지 않습니다. 버킷 소유자가 버킷 정책을 통해 액세스 포인트 소유자의 계정(귀하의 계정)에게 버킷에 대한 액세스 권한을 부여해야 합니다. 자세한 내용은 크로스 계정 액세스 포인트에 대한 권한 부여 섹션을 참조하세요.

S3 콘솔 사용

액세스 포인트 생성

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 액세스 포인트를 생성하려는 리전을 선택합니다. 액세스 포인트는 연결된 버킷과 동일한 리전에서 생성해야 합니다.
3. 탐색 창에서 액세스 포인트(Access Points)를 선택합니다.
4. 액세스 포인트(Access Points) 페이지에서 액세스 포인트 생성(Create access point)을 선택합니다.
5. 액세스 포인트 이름 필드에 액세스 포인트 이름을 입력합니다. 액세스 포인트 명명에 대한 자세한 내용은 액세스 포인트 이름 지정 규칙 섹션을 참조하십시오.
6. 데이터 소스에 액세스 포인트에 사용할 S3 버킷을 지정합니다.

계정의 버킷을 사용하려면 이 계정의 버킷 선택을 선택하고 대상 버킷 이름을 입력하거나 찾아봅니다.

다른 AWS 계정의 버킷을 사용하려면 다른 계정의 버킷 지정을 선택하고 버킷의 AWS 계정 ID와 이름을 입력합니다. 다른 AWS 계정의 버킷을 사용하는 경우, 액세스 포인트의 요청을 승인하도록 버킷 소유자가 버킷 정책을 업데이트해야 합니다. 버킷 정책 예제는 크로스 계정 액세스 포인트에 대한 권한 부여를 참조하십시오.

> **Note**
>
> FSx for OpenZFS 볼륨을 데이터 소스로 사용하는 방법에 대한 자세한 내용은 Amazon FSx를 사용하여 액세스 포인트 생성 섹션을 참조하세요.

7. 인터넷 또는 가상 프라이빗 클라우드(VPC) 중에서 네트워크 오리진을 선택합니다. 가상 프라이빗 클라우드(VPC)를 선택한 경우 액세스 포인트에 사용할 VPC ID를 입력합니다.

   액세스 포인트의 네트워크 오리진에 대한 자세한 내용은 가상 프라이빗 클라우드(VPC)로 제한된 액세스 포인트 생성 섹션을 참조하십시오.

8. 이 액세스 포인트에 대한 퍼블릭 액세스 차단 설정에서 액세스 포인트에 적용할 퍼블릭 액세스 차단 설정을 선택합니다. 신규 액세스 포인트에는 기본적으로 모든 퍼블릭 액세스 차단 설정이 활성화되어 있습니다. 특정 설정을 사용 중지해야 하는 경우가 아니면 모든 설정을 그대로 유지하는 것이 좋습니다.

   > **Note**
   >
   > 액세스 포인트를 생성한 후에는 퍼블릭 액세스 차단 설정을 변경할 수 없습니다.

   액세스 포인트에 Amazon S3 퍼블릭 액세스 차단을 사용하는 방법에 대한 자세한 내용은 범용 버킷의 액세스 포인트에 대한 퍼블릭 액세스 관리 섹션을 참조하십시오.

9. (선택 사항) 액세스 포인트 정책 - 선택 사항에서 액세스 포인트 정책을 지정합니다. 정책을 저장하기 전에 보안 경고, 오류, 일반 경고 및 제안 사항을 해결해야 합니다. 액세스 포인트 정책 지정에 대한 자세한 내용은 액세스 포인트의 정책 예시 섹션을 참조하십시오.

10. 액세스 포인트 생성을 선택합니다.

AWS CLI 사용

다음 예제 명령에서는 계정 *111122223333* 내의 버킷 *amzn-s3-demo-bucket*에 대해 이름이 *example-ap*인 액세스 포인트를 생성합니다. 액세스 포인트를 생성하려면 다음을 지정하는 요청을 Amazon S3에 보냅니다.

- 액세스 포인트 이름. 이름 지정 규칙에 대한 자세한 내용은 the section called "액세스 포인트 이름 지정 규칙" 단원을 참조하십시오.
- 액세스 포인트와 연결하려는 버킷의 이름.
- 액세스 포인트를 소유한 AWS 계정의 계정 ID.

```
aws s3control create-access-point --name example-ap --account-id 111122223333 --
bucket amzn-s3-demo-bucket
```

다른 AWS 계정 내의 버킷을 사용하여 액세스 포인트를 생성하려면 `--bucket-account-id` 파라미터를 포함합니다. 다음 예제 명령에서는 AWS 계정 *444455556666* 내의 버킷 *amzn-s3-demo-bucket2*를 사용하여 AWS 계정 *111122223333* 내에 액세스 포인트를 생성합니다.

```
aws s3control create-access-point --name example-ap --account-id 111122223333 --
bucket amzn-s3-demo-bucket --bucket-account-id 444455556666
```

REST API 사용

REST API를 사용하여 액세스 포인트를 만들 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 CreateAccessPoint를 참조하십시오.

Amazon FSx를 사용하여 액세스 포인트 생성

Amazon FSx 콘솔, AWS CLI 또는 API를 사용하여 액세스 포인트를 생성하여 FSx for OpenZFS 볼륨에 연결할 수 있습니다. 연결되면 S3 객체 API를 사용하여 파일 데이터에 액세스할 수 있습니다. 데이터는 Amazon FSx 파일 시스템에 계속 상주하며 기존 워크로드에서 직접 액세스할 수 있습니다. 백업, 스냅샷, 사용자 및 그룹 할당량, 압축 등 모든 FSx for OpenZFS 스토리지 관리 기능을 사용하여 스토리지를 계속 관리합니다.

액세스 포인트를 생성하고 FSx for OpenZFS 볼륨에 연결하는 방법에 대한 지침은 FSx for OpenZFS 사용 설명서의 Creating an access point를 참조하세요.

가상 프라이빗 클라우드(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 가상 프라이빗 클라우드(VPC)에서 시작되도록 지정할 수 있습니다. 인터넷에서 액세스할 수 있는 액세스 포인트의 네트워크 오리진은 `Internet`입니다. 액세스 포인트, 기본 데이터 소스 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 VPC이고, Amazon S3은 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 포인트 사용을 허용할 VPC ID를 지정합니다. 액세스 포인트를 통해 요청이 이루어진 경우 요청은 VPC에서 시작되어야 하며 그렇지 않으면 Amazon S3에서 이를 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 VPC입니다. 그렇지 않으면 액세스 포인트의 네트워크 오리진은 `Internet`입니다.

예: 액세스 포인트 생성 및 VPC ID로만 제한

다음 예제에서는 `vpc-1a2b3c` VPC에서만 액세스하도록 허용하는 계정 `123456789012`의 버킷 *`amzn-s3-demo-bucket`*에 이름이 `example-vpc-ap`인 액세스 포인트를 생성합니다. 그런 다음 새 액세스 포인트의 네트워크 오리진이 VPC인지 확인합니다.

AWS CLI

```
aws s3control create-access-point --name example-vpc-ap --account-id 123456789012 --
bucket amzn-s3-demo-bucket --vpc-configuration VpcId=vpc-1a2b3c
```

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

{
```

```
    "Name": "example-vpc-ap",
    "Bucket": "amzn-s3-demo-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 S3으로 트래픽이 흐를 수 있습니다. VPC 내의 리소스가 Amazon S3와 상호 작용할 수 있는 방법을 제어하는 액세스 제어 정책이 있습니다. VPC에서 Amazon S3로의 요청은 VPC 엔드포인트 정책이 액세스 포인트와 기본 버킷 모두에 대한 액세스 권한을 부여하는 경우에만 액세스 포인트를 통해 성공합니다.

> **Note**
>
> VPC 내에서만 리소스에 액세스할 수 있도록 하려면 VPC 엔드포인트에 대한 프라이빗 호스팅 영역을 생성해야 합니다. 프라이빗 호스팅 영역을 사용하려면 VPC 네트워크 속성 및 `enableDnsHostnames`가 `enableDnsSupport`로 설정되도록 VPC 설정을 수정합니다.

다음 예제 정책 문은 이름이 `awsexamplebucket1`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:::awsexamplebucket1/*",
            "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 ARN, 액세스 포인트 별칭 또는 가상 호스팅 방식 URI를 사용하여 액세스 포인트 참조 단원을 참조하십시오.

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

VPC 엔드포인트를 사용하여 액세스 포인트를 생성하는 방법에 대한 자습서는 Managing Amazon S3 access with VPC endpoints and access points를 참조하세요.

예: FSx for OpenZFS 볼륨에 연결된 액세스 포인트 생성 및 VPC ID로만 제한

Amazon FSx 콘솔, AWS CLI 또는 API를 사용하여 FSx for OpenZFS 볼륨에 연결된 액세스 포인트를 생성할 수 있습니다. 연결되면 S3 객체 API를 사용하여 지정된 VPC에서 파일 데이터에 액세스할 수 있습니다.

FSx for OpenZFS 볼륨에 연결된 액세스 포인트를 생성하고 제한하는 방법에 대한 지침은 FSx for OpenZFS 사용 설명서의 가상 프라이빗 클라우드(VPC)로 제한된 범용 버킷의 액세스 포인트 만들기를 참조하세요.

범용 버킷의 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 S3은 해당 액세스 포인트, 기본 버킷 및 버킷 소유자 계정에 대한 퍼블릭 액세스 차단 설정을 평가합니다. 이러한 설정 중 하나라도 요청이 차단되어야 함을 나타내는 경우, Amazon S3은 요청을 거부합니다.

S3 퍼블릭 액세스 차단 기능에 대한 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 단원을 참조하십시오.

Important

- 액세스 포인트에는 기본적으로 모든 퍼블릭 액세스 차단 설정이 사용 설정되어 있습니다. 액세스 포인트 생성 중 원치 않는 설정을 명시적으로 비활성화해야 합니다.
- Amazon FSx 파일 시스템에 연결된 액세스 포인트를 생성하거나 사용할 때는 퍼블릭 액세스 차단 설정을 끌 수 없습니다.
- 액세스 포인트를 생성한 후에는 퍼블릭 액세스 차단 설정을 변경할 수 없습니다.

Example

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

이 예제에서는 기본값이 아닌 퍼블릭 액세스 차단 설정을 사용하여 계정 123456789012의 버킷 *amzn-s3-demo-bucket*에 이름이 example-ap인 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 amzn-s3-demo-bucket--public-access-block-configuration
 BlockPublicAcls=false,IgnorePublicAcls=false,BlockPublicPolicy=true,RestrictPublicBuckets=t
```

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

{
    "Name": "example-ap",
    "Bucket": "amzn-s3-demo-bucket",
    "NetworkOrigin": "Internet",
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": false,
        "IgnorePublicAcls": false,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2019-11-27T00:00:00Z"
}
```

범용 버킷의 Amazon S3 Access Points 관리

이 섹션에서는 AWS Management Console, AWS Command Line Interface 또는 REST API를 사용하여 범용 버킷의 Amazon S3 Access Points를 관리하는 방법에 대해 설명합니다. FSx for OpenZFS 볼륨에 연결된 액세스 포인트를 관리하는 방법에 대한 자세한 내용은 FSx for OpenZFS 사용 설명서의 Managing your Amazon S3 access points를 참조하세요.

> **Note**
>
> 액세스 포인트를 사용하면 객체에 대한 작업만 수행할 수 있습니다. 액세스 포인트를 사용하여 버킷 삭제 또는 S3 복제 구성 생성과 같은 다른 Amazon S3 작업을 수행할 수 없습니다. 액세스 포인트를 지원하는 전체 S3 작업 목록은 액세스 포인트 호환성 단원을 참조하십시오.

주제

- 범용 버킷의 액세스 포인트 나열
- 범용 버킷의 액세스 포인트에 대한 세부 정보 보기
- 범용 버킷의 액세스 포인트 삭제

범용 버킷의 액세스 포인트 나열

이 섹션에서는 AWS Management Console, AWS Command Line Interface 또는 REST API를 사용하여 범용 버킷의 액세스 포인트를 나열하는 방법에 대해 설명합니다.

S3 콘솔 사용

AWS 계정의 액세스 포인트를 나열하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 액세스 포인트를 나열하려는 리전을 선택합니다.
3. 콘솔 왼쪽의 탐색 창에서 액세스 포인트를 선택합니다.
4. (선택 사항) 이름으로 액세스 포인트를 검색합니다. 선택한 AWS 리전의 액세스 포인트만 여기에 표시됩니다.
5. 관리하거나 사용할 액세스 포인트의 이름을 선택합니다.

AWS CLI 사용

다음 `list-access-points` 예시 명령은 AWS CLI를 사용하여 액세스 포인트를 나열하는 방법을 보여줍니다.

다음 명령은 AWS 계정 *111122223333*의 액세스 포인트를 나열합니다.

```
aws s3control list-access-points --account-id 111122223333
```

다음 명령은 버킷 *amzn-s3-demo-bucket*에 연결된 AWS 계정 *111122223333*의 액세스 포인트를 나열합니다.

```
aws s3control list-access-points --account-id 111122223333 --bucket amzn-s3-demo-bucket
```

자세한 내용과 예제는 AWS CLI 명령 참조에서 list-access-points를 참조하십시오.

REST API 사용

REST API를 사용하여 액세스 포인트를 나열할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 ListAccessPoints를 참조하십시오.

범용 버킷의 액세스 포인트에 대한 세부 정보 보기

이 섹션에서는 AWS Management Console, AWS Command Line Interface 또는 REST API를 사용하여 범용 버킷의 액세스 포인트에 대한 세부 정보를 보는 방법을 설명합니다.

S3 콘솔 사용

AWS 계정의 액세스 포인트에 대한 세부 정보를 보는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 액세스 포인트를 나열하려는 리전을 선택합니다.
3. 콘솔 왼쪽의 탐색 창에서 액세스 포인트를 선택합니다.
4. (선택 사항) 이름으로 액세스 포인트를 검색합니다. 선택한 AWS 리전의 액세스 포인트만 여기에 표시됩니다.
5. 관리하거나 사용할 액세스 포인트의 이름을 선택합니다.

6. 속성 탭을 선택하여 선택한 액세스 포인트의 액세스 포인트 데이터 소스, 계정 ID, AWS 리전, 생성 날짜, 네트워크 오리진, S3 URI, ARN 및 액세스 포인트 별칭을 확인합니다.
7. 권한 탭을 선택하여 선택한 액세스 포인트에 대한 퍼블릭 액세스 차단 설정 및 액세스 포인트 정책을 봅니다.

> **Note**
>
> 액세스 포인트를 만든 후에는 액세스 포인트의 퍼블릭 액세스 차단 설정을 변경할 수 없습니다.

AWS CLI 사용

다음 `get-access-point` 예시 명령은 AWS CLI를 사용하여 액세스 포인트의 세부 정보를 보는 방법을 보여줍니다.

다음 명령은 S3 버킷 *amzn-s3-demo-bucket*에 연결된 AWS 계정 *111122223333*의 액세스 포인트 *my-access-point*에 대한 세부 정보를 나열합니다.

```
aws s3control get-access-point --name my-access-point --account-id 111122223333
```

출력 예시:

```
{
    "Name": "my-access-point",
    "Bucket": "amzn-s3-demo-bucket",
    "NetworkOrigin": "Internet",
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2016-08-29T22:57:52Z",
    "Alias": "my-access-point-u1ny6bhm7moymqx8cuon8o1g4mwikuse2a-s3alias",
    "AccessPointArn": "arn:aws:s3:AWS ##:111122223333:accesspoint/my-access-point",
    "Endpoints": {
        "ipv4": "s3-accesspoint.AWS ##.amazonaws.com",
        "fips": "s3-accesspoint-fips.AWS ##.amazonaws.com",
        "fips_dualstack": "s3-accesspoint-fips.dualstack.AWS ##.amazonaws.com",
```

```
        "dualstack": "s3-accesspoint.dualstack.AWS ##.amazonaws.com"
    },
    "BucketAccountId": "111122223333"
}
```

다음 명령은 AWS 계정 *444455556666*의 액세스 포인트 *example-fsx-ap*에 대한 세부 정보를 나열합니다. 이 액세스 포인트는 Amazon FSx 파일 시스템에 연결되어 있습니다.

```
aws s3control get-access-point --name example-fsx-ap --account-id 444455556666
```

출력 예시:

```
{
    "Name": "example-fsx-ap",
    "Bucket": "",
    "NetworkOrigin": "Internet",
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2025-01-19T14:16:12Z",
    "Alias": "example-fsx-ap-qrqbyebjtsxorhhaa5exx6r3q7-ext-s3alias",
    "AccessPointArn": "arn:aws:s3:AWS ##:444455556666:accesspoint/example-fsx-ap",
    "Endpoints": {
        "ipv4": "s3-accesspoint.AWS ##.amazonaws.com",
        "fips": "s3-accesspoint-fips.AWS ##.amazonaws.com",
        "fips_dualstack": "s3-accesspoint-fips.dualstack.AWS ##.amazonaws.com",
        "dualstack": "s3-accesspoint.dualstack.AWS ##.amazonaws.com"
    },
    "DataSourceId": "arn:aws::fsx:AWS ##:444455556666:file-system/fs-5432106789abcdef0/
volume/vol-0123456789abcdef0",
    "DataSourceType": "FSX_OPENZFS"
}
```

자세한 내용과 예제는 AWS CLI 명령 참조에서 get-access-point를 참조하십시오.

REST API 사용

REST API를 사용하여 액세스 포인트의 세부 정보를 볼 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 GetAccessPoint를 참조하십시오.

범용 버킷의 액세스 포인트 삭제

이 섹션에서는 AWS Management Console, AWS Command Line Interface 또는 REST API를 사용하여 범용 버킷의 액세스 포인트를 삭제하는 방법을 설명합니다.

S3 콘솔 사용

AWS 계정의 액세스 포인트를 삭제하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 액세스 포인트를 나열하려는 리전을 선택합니다.
3. 콘솔 왼쪽의 탐색 창에서 액세스 포인트를 선택합니다.
4. (선택 사항) 이름으로 액세스 포인트를 검색합니다. 선택한 AWS 리전의 액세스 포인트만 여기에 표시됩니다.
5. 관리하거나 사용할 액세스 포인트의 이름을 선택합니다.
6. 액세스 포인트 페이지에서 삭제를 선택하여 선택한 액세스 포인트를 삭제합니다.
7. 삭제를 확정하려면 액세스 포인트의 이름을 입력하고 삭제를 선택합니다.

AWS CLI 사용

다음 `delete-access-point` 예시 명령은 AWS CLI를 사용하여 액세스 포인트를 삭제하는 방법을 보여줍니다.

다음 명령은 AWS 계정 *111122223333*의 액세스 포인트 *my-access-point*를 삭제합니다.

```
aws s3control delete-access-point --name my-access-point --account-id 111122223333
```

자세한 내용과 예제는 AWS CLI 명령 참조에서 delete-access-point를 참조하십시오.

REST API 사용

REST API를 사용하여 액세스 포인트의 세부 정보를 볼 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 DeleteAccessPoint를 참조하십시오.

범용 버킷의 Amazon S3 Access Points 사용

다음 예제에서는 Amazon S3에서 호환되는 작업에 범용 버킷의 액세스 포인트를 사용하는 방법을 보여 줍니다.

> **Note**
>
> S3는 모든 액세스 포인트에 대한 액세스 포인트 별칭을 자동으로 생성하며, 버킷 이름이 객체 수준 작업을 수행하는 데 사용되는 모든 곳에서 이러한 별칭을 사용할 수 있습니다. 자세한 내용은 액세스 포인트 별칭 섹션을 참조하세요.

범용 버킷의 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 액세스 포인트 호환성 단원을 참조하십시오.

주제

- 범용 버킷의 액세스 포인트를 통해 객체 나열
- 범용 버킷의 액세스 포인트를 통해 객체 다운로드
- 범용 버킷의 액세스 포인트를 통해 액세스 제어 목록(ACL) 구성
- 범용 버킷의 액세스 포인트를 통해 객체 업로드
- 범용 버킷의 액세스 포인트를 통해 태그 세트 추가
- 범용 버킷의 액세스 포인트를 통해 객체 삭제

범용 버킷의 액세스 포인트를 통해 객체 나열

이 섹션에서는 AWS Management Console, AWS Command Line Interface 또는 REST API를 사용하여 범용 버킷의 액세스 포인트를 통해 객체를 나열하는 방법에 대해 설명합니다.

S3 콘솔 사용

AWS 계정의 액세스 포인트를 통해 객체를 나열하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 액세스 포인트를 나열하려는 리전을 선택합니다.

3. 콘솔 왼쪽의 탐색 창에서 액세스 포인트를 선택합니다.
4. (선택 사항) 이름으로 액세스 포인트를 검색합니다. 선택한 AWS 리전의 액세스 포인트만 여기에 표시됩니다.
5. 관리하거나 사용할 액세스 포인트의 이름을 선택합니다.
6. 객체 탭에서 액세스 포인트를 통해 액세스하려는 객체의 이름을 볼 수 있습니다. 액세스 포인트를 사용하는 동안에는 액세스 포인트 권한에서 허용하는 객체 작업만 수행할 수 있습니다.

> **Note**
>
> - 콘솔 보기에는 항상 버킷의 모든 객체가 표시됩니다. 이 절차에 설명된 대로 액세스 포인트를 사용하면 해당 객체에 대해 수행할 수 있는 작업은 제한되지만 버킷에 해당 객체가 존재하는지 여부는 제한되지 않습니다.
> - AWS Management Console은 가상 프라이빗 클라우드(VPC) 액세스 포인트를 사용하여 버킷 리소스에 액세스하는 것을 지원하지 않습니다. VPC 액세스 포인트에서 버킷 리소스에 액세스하려면 AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용합니다.

AWS CLI 사용

다음 `list-objects-v2` 예시 명령은 AWS CLI를 사용하여 액세스 포인트를 통해 객체를 나열하는 방법을 보여줍니다.

다음 명령은 액세스 포인트 *my-access-point*를 사용하여 AWS 계정 *111122223333*의 객체를 나열합니다.

```
aws s3api list-objects-v2 --bucket arn:aws:s3:AWS ##:111122223333:accesspoint/my-access-point
```

> **Note**
>
> S3는 모든 액세스 포인트에 대한 액세스 포인트 별칭을 자동으로 생성하며, 버킷 이름이 객체 수준 작업을 수행하는 데 사용되는 모든 곳에서 이러한 별칭을 사용할 수 있습니다. 자세한 내용은 액세스 포인트 별칭 섹션을 참조하세요.

자세한 내용과 예제는 AWS CLI 명령 참조에서 list-access-points를 참조하십시오.

REST API 사용

REST API를 사용하여 액세스 포인트를 나열할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 ListObjectsV2를 참조하십시오.

범용 버킷의 액세스 포인트를 통해 객체 다운로드

이 섹션에서는 AWS Management Console, AWS Command Line Interface 또는 REST API를 사용하여 범용 버킷의 액세스 포인트를 통해 객체를 다운로드하는 방법에 대해 설명합니다.

S3 콘솔 사용

AWS 계정의 액세스 포인트를 통해 객체를 다운로드하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 액세스 포인트를 나열하려는 리전을 선택합니다.
3. 콘솔 왼쪽의 탐색 창에서 액세스 포인트를 선택합니다.
4. (선택 사항) 이름으로 액세스 포인트를 검색합니다. 선택한 AWS 리전의 액세스 포인트만 여기에 표시됩니다.
5. 관리하거나 사용할 액세스 포인트의 이름을 선택합니다.
6. 객체 탭에서 다운로드할 객체의 이름을 선택합니다.
7. 다운로드를 선택합니다.

AWS CLI 사용

다음 `get-object` 예시 명령은 AWS CLI를 사용하여 액세스 포인트를 통해 객체를 다운로드하는 방법을 보여줍니다.

다음 명령은 액세스 포인트 *my-access-point*를 사용하여 AWS 계정 *111122223333*의 `puppy.jpg` 객체를 다운로드합니다. 다운로드한 객체의 파일 이름인 `outfile`을 포함해야 합니다(예: *my_downloaded_image.jpg*).

```
aws s3api get-object --bucket arn:aws:s3:AWS ##:111122223333:accesspoint/my-access-point --key puppy.jpg my_downloaded_image.jpg
```

> **Note**
>
> S3는 모든 액세스 포인트에 대한 액세스 포인트 별칭을 자동으로 생성하며, 버킷 이름이 객체 수준 작업을 수행하는 데 사용되는 모든 곳에서 이러한 별칭을 사용할 수 있습니다. 자세한 내용은 액세스 포인트 별칭 섹션을 참조하세요.

자세한 내용과 예제는 AWS CLI 명령 참조에서 get-object를 참조하십시오.

REST API 사용

REST API를 사용하여 액세스 포인트를 통해 객체를 다운로드할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 GetObject를 참조하십시오.

AWS SDK 사용

AWS SDK for Python을 사용하여 액세스 포인트를 통해 객체를 다운로드할 수 있습니다.

Python

다음 예제에서는 *my-access-point*라는 액세스 포인트를 사용하여 AWS 계정 *111122223333*에 대해 이름이 *hello.txt*인 파일이 다운로드됩니다.

```
import boto3
s3 = boto3.client('s3')
s3.download_file('arn:aws:s3:us-east-1:111122223333:accesspoint/my-access-point',
 'hello.txt', '/tmp/hello.txt')
```

범용 버킷의 액세스 포인트를 통해 액세스 제어 목록(ACL) 구성

이 섹션에서는 AWS Management Console, AWS Command Line Interface 또는 REST API를 사용하여 범용 버킷의 액세스 포인트를 통해 ACL을 구성하는 방법을 설명합니다. ACL에 대한 자세한 내용은 ACL(액세스 제어 목록) 개요 단원을 참조하십시오.

S3 콘솔 사용

AWS 계정의 액세스 포인트를 통해 ACL을 구성하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.

2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 액세스 포인트를 나열하려는 리전을 선택합니다.
3. 콘솔 왼쪽의 탐색 창에서 액세스 포인트를 선택합니다.
4. (선택 사항) 이름으로 액세스 포인트를 검색합니다. 선택한 AWS 리전의 액세스 포인트만 여기에 표시됩니다.
5. 관리하거나 사용할 액세스 포인트의 이름을 선택합니다.
6. 객체 탭에서 ACL을 구성하려는 객체의 이름을 선택합니다.
7. 권한 탭에서 편집을 선택하여 객체 ACL을 구성합니다.

> **Note**
>
> Amazon S3에서는 현재 액세스 포인트가 생성된 후 액세스 포인트의 퍼블릭 액세스 차단 설정을 변경하도록 지원하지 않습니다.

AWS CLI 사용

다음 `put-object-acl` 예시 명령은 AWS CLI를 사용하여 액세스 포인트를 통해 ACL을 사용해 액세스 권한을 구성하는 방법을 보여줍니다.

다음 명령은 AWS 계정*111122223333*에서 소유한 액세스 포인트를 통해 기존 객체 `puppy.jpg`에 ACL을 적용합니다.

```
aws s3api put-object-acl --bucket arn:aws:s3:AWS ##:111122223333:accesspoint/my-access-point --key puppy.jpg --acl private
```

> **Note**
>
> S3는 모든 액세스 포인트에 대한 액세스 포인트 별칭을 자동으로 생성하며, 버킷 이름이 객체 수준 작업을 수행하는 데 사용되는 모든 곳에서 이러한 별칭을 사용할 수 있습니다. 자세한 내용은 액세스 포인트 별칭 섹션을 참조하세요.

자세한 내용과 예제는 AWS CLI 명령 참조에서 put-object-acl를 참조하십시오.

REST API 사용

REST API를 사용하여 액세스 포인트를 통해 ACL을 사용해 액세스 권한을 구성할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 PutObjectAcl를 참조하십시오.

범용 버킷의 액세스 포인트를 통해 객체 업로드

이 섹션에서는 AWS Management Console, AWS Command Line Interface 또는 REST API를 사용하여 범용 버킷의 액세스 포인트를 통해 객체를 업로드하는 방법에 대해 설명합니다.

S3 콘솔 사용

AWS 계정의 액세스 포인트를 통해 객체를 업로드하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 액세스 포인트를 나열하려는 리전을 선택합니다.
3. 콘솔 왼쪽의 탐색 창에서 액세스 포인트를 선택합니다.
4. (선택 사항) 이름으로 액세스 포인트를 검색합니다. 선택한 AWS 리전의 액세스 포인트만 여기에 표시됩니다.
5. 관리하거나 사용할 액세스 포인트의 이름을 선택합니다.
6. 객체 탭에서 업로드를 선택합니다.
7. 업로드할 파일과 폴더를 여기에 끌어다 놓거나 파일 추가 또는 폴더 추가를 선택합니다.

   > **Note**
   >
   > Amazon S3 콘솔을 사용하여 업로드할 수 있는 파일의 최대 크기는 160GB입니다. 160GB가 넘는 파일을 업로드하려면 AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하십시오.

8. 액세스 제어 목록 권한을 변경하려면 권한(Permissions)을 선택합니다.
9. 액세스 제어 목록(ACL)(Access control list (ACL))에서 권한을 편집합니다.

   객체 액세스 권한에 대한 자세한 내용은 S3 콘솔을 사용하여 객체에 대한 ACL 권한 설정를 참조하십시오. 현재 업로드 중인 모든 파일에 대해 대중(전 세계 모든 사람)에게 객체에 대한 읽기 액세스 권한을 부여할 수 있습니다. 그러나 퍼블릭 읽기 액세스의 기본 설정은 변경하지 않는 것이 좋

습니다. 퍼블릭 읽기 액세스 권한 부여는 웹 사이트에 버킷을 사용하는 경우와 같은 소수의 사용 사례에 적용될 수 있습니다. 객체를 업로드한 후에도 언제든 객체 권한을 변경할 수 있습니다.

10. 다른 추가 속성을 구성하려면 속성(Properties)을 선택합니다.
11. 스토리지 클래스에서 업로드하려는 파일의 스토리지 클래스를 선택합니다.

    스토리지 클래스에 대한 자세한 정보는 Amazon S3 스토리지 클래스 이해 및 관리를 참조하세요.
12. 객체에 대한 암호화 설정을 업데이트하려면 서버 측 암호화 설정에서 다음을 수행합니다.

    a. 암호화 키 지정(Specify an encryption key)을 선택합니다.

    b. 암호화 설정에서 버킷 기본 암호화 설정 사용 또는 기본 암호화에 버킷 설정 재정의를 선택합니다.

    c. 기본 암호화에 버킷 설정 재정의를 선택한 경우 다음과 같은 암호화 설정을 구성해야 합니다.

    - Amazon S3에서 관리하는 키를 사용하여 업로드된 파일을 암호화하려면 Amazon S3 관리형 키(SSE-S3)를 선택합니다.

      자세한 내용은 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) 사용 섹션을 참조하세요.
    - AWS Key Management Service(AWS KMS)에 저장된 키를 사용하여 업로드된 파일을 암호화하려면 AWS Key Management Service 키(SSE-KMS)를 선택합니다. 그리고 나서 AWS KMS 키에 다음 옵션 중 하나를 선택합니다.
      - 사용 가능한 KMS 키 목록에서 AWS KMS keys 중에서 선택을 선택한 다음, 사용 가능한 키 목록에서 KMS 키를 선택합니다.

        AWS 관리형 키(aws/s3)와 고객 관리형 키가 모두 목록에 표시됩니다. 고객 관리형 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 키 및 AWS 키를 참조하세요.
      - KMS 키 ARN을 입력하려면 AWS KMS key ARN 입력을 선택한 다음 나타나는 필드에 KMS 키 ARN을 입력합니다.
      - AWS KMS 콘솔에서 고객 관리형 키를 생성하려면 KMS 키 생성을 선택합니다.

        AWS KMS key 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 키 생성을 참조하십시오.

> **Important**
>
> 버킷과 동일한 AWS 리전에서 사용할 수 있는 KMS 키만 사용 가능합니다. Amazon S3 콘솔은 버킷과 동일한 리전에 있는 처음 100개의 KMS 키만 나열합니다. 목록에 없는 KMS 키를 사용하려면 KMS 키 ARN을 입력해야 합니다. 다른 계정에서 소유한 KMS 키를 사용하려면 먼저 해당 키에 대한 사용 권한이 있어야 하고, 다음 단계로 KMS 키 ARN을 입력해야 합니다.
> Amazon S3는 대칭 암호화 KMS 키만 지원하며 비대칭 KMS 키는 지원하지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 대칭 및 비대칭 KMS 키 식별을 참조하세요.

13. 추가 체크섬을 사용하려면 켜기(On)를 선택합니다. 그런 다음 체크섬 함수(Checksum function)에서 사용할 함수를 선택합니다. Amazon S3는 전체 객체를 수신한 후 체크섬 값을 계산하고 저장합니다. 미리 계산된 값(Precalculated value) 상자를 사용하여 미리 계산된 값을 제공할 수 있습니다. 이렇게 하면 Amazon S3가 제공한 값과 계산한 값을 비교합니다. 두 값이 일치하지 않으면 Amazon S3가 오류를 생성합니다.

    추가 체크섬을 사용하면 데이터를 확인하는 데 사용할 체크섬 알고리즘을 지정할 수 있습니다. 추가 체크섬에 대한 자세한 내용은 Amazon S3에서 객체 무결성 확인 단원을 참조하십시오.

14. 업로드 중인 모든 객체에 태그를 추가하려면 태그 추가를 선택합니다. 키 필드에 태그 이름을 입력합니다. 태그 값을 입력합니다.

    객체 태그 지정을 통해 스토리지를 분류할 수 있습니다. 각 태그는 키-값 페어입니다. 키와 태그 값은 대/소문자를 구분합니다. 객체마다 태그를 10개까지 포함할 수 있습니다. 태그 키는 최대 128개 길이의 유니코드 문자이며, 태그 값은 최대 255개의 유니코드 문자입니다. 객체 태그에 대한 자세한 내용은 태그를 사용하여 스토리지 분류 섹션을 참조하세요.

15. 메타데이터를 추가하려면 메타데이터 추가를 선택합니다.

    a. 유형에서 시스템 정의 또는 사용자 정의를 선택합니다.

       시스템 정의 메타데이터의 경우, 콘텐츠 유형 및 콘텐츠 처리 같은 일반적인 HTTP 헤더를 선택할 수 있습니다. 시스템 정의 메타데이터 목록과 값 추가 가능 여부를 확인하려면 시스템 정의 객체 메타데이터 단원을 참조하십시오. 접두사 `x-amz-meta-`로 시작하는 모든 메타데이터는 사용자 정의 메타데이터로 처리됩니다. 사용자 정의 메타데이터는 객체와 함께 저장되었다가 해당 객체를 다운로드할 때 반환됩니다. 키와 값 모두 US-ASCII 표준에 부합해야 합니다. 사용자 정의 메타데이터의 최대 크기는 2KB입니다. 시스템 정의 메타데이터와 사용자 정의 메타데이터에 대한 자세한 내용은 객체 메타데이터 작업 단원을 참조하십시오.

b. 키에 대해 키를 선택합니다.

c. 키 값을 입력합니다.

16. 객체를 업로드하려면 업로드를 선택합니다.

Amazon S3가 객체를 업로드합니다. 업로드가 완료되면 업로드: 상태 페이지에서 성공 메시지를 볼 수 있습니다.

AWS CLI 사용

다음 `put-object` 예시 명령은 AWS CLI를 사용하여 액세스 포인트를 통해 객체를 업로드하는 방법을 보여줍니다.

다음 명령은 액세스 포인트 *my-access-point*를 사용하여 AWS 계정 *111122223333*의 `puppy.jpg` 객체를 업로드합니다.

```
aws s3api put-object --bucket arn:aws:s3:AWS ##:111122223333:accesspoint/my-access-point --key puppy.jpg --body puppy.jpg
```

> **Note**
>
> S3는 모든 액세스 포인트에 대한 액세스 포인트 별칭을 자동으로 생성하며, 버킷 이름이 객체 수준 작업을 수행하는 데 사용되는 모든 곳에서 액세스 포인트 별칭을 사용할 수 있습니다. 자세한 내용은 액세스 포인트 별칭 섹션을 참조하세요.

자세한 내용과 예제는 AWS CLI 명령 참조에서 put-object를 참조하십시오.

REST API 사용

REST API를 사용하여 액세스 포인트를 통해 객체를 업로드할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 PutObject를 참조하십시오.

AWS SDK 사용

AWS SDK for Python을 사용하여 액세스 포인트를 통해 객체를 업로드할 수 있습니다.

Python

다음 예제에서는 *my-access-point*라는 액세스 포인트를 사용하여 AWS 계정 *111122223333*에 대해 이름이 *hello.txt*인 파일이 업로드됩니다.

```
import boto3
s3 = boto3.client('s3')
s3.upload_file('/tmp/hello.txt', 'arn:aws:s3:us-east-1:111122223333:accesspoint/my-access-point', 'hello.txt')
```

범용 버킷의 액세스 포인트를 통해 태그 세트 추가

이 섹션에서는 AWS Management Console, AWS Command Line Interface 또는 REST API를 사용하여 범용 버킷의 액세스 포인트를 통해 태그 세트를 추가하는 방법에 대해 설명합니다. 자세한 내용은 태그를 사용하여 스토리지 분류 섹션을 참조하세요.

S3 콘솔 사용

AWS 계정의 액세스 포인트를 통해 태그 세트를 추가하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 액세스 포인트를 나열하려는 리전을 선택합니다.
3. 콘솔 왼쪽의 탐색 창에서 액세스 포인트를 선택합니다.
4. (선택 사항) 이름으로 액세스 포인트를 검색합니다. 선택한 AWS 리전의 액세스 포인트만 여기에 표시됩니다.
5. 관리하거나 사용할 액세스 포인트의 이름을 선택합니다.
6. 객체 탭에서 태그 세트를 추가할 객체의 이름을 선택합니다.
7. 속성 탭에서 태그 하위 헤더를 찾아 편집을 선택합니다.
8. 나열된 객체를 검토하고 태그 추가를 선택합니다.
9. 각 객체 태그는 키-값 페어입니다. 키와 값을 입력합니다. 태그를 더 추가하려면 태그 추가(Add Tag)를 선택합니다.

   한 개체에 태그를 최대 10개까지 입력할 수 있습니다.
10. 변경 사항 저장을 선택합니다.

AWS CLI 사용

다음 `put-object-tagging` 예시 명령은 AWS CLI를 사용하여 액세스 포인트를 통해 태그 세트를 추가하는 방법을 보여줍니다.

다음 명령은 액세스 포인트 *my-access-point*를 사용하여 기존 객체 puppy.jpg에 태그 세트를 추가합니다.

```
aws s3api put-object-tagging --bucket arn:aws:s3:AWS ##:111122223333:accesspoint/my-access-point --key puppy.jpg --tagging TagSet=[{Key="animal",Value="true"}]
```

> **Note**
>
> S3는 모든 액세스 포인트에 대한 액세스 포인트 별칭을 자동으로 생성하며, 버킷 이름이 객체 수준 작업을 수행하는 데 사용되는 모든 곳에서 액세스 포인트 별칭을 사용할 수 있습니다. 자세한 내용은 액세스 포인트 별칭 섹션을 참조하세요.

자세한 내용과 예제는 AWS CLI 명령 참조에서 put-object-tagging를 참조하십시오.

REST API 사용

REST API를 사용하여 액세스 포인트를 통해 객체에 태그 세트를 추가할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 PutObjectTagging를 참조하십시오.

범용 버킷의 액세스 포인트를 통해 객체 삭제

이 섹션에서는 AWS Management Console, AWS Command Line Interface 또는 REST API를 사용하여 범용 버킷의 액세스 포인트를 통해 객체를 삭제하는 방법에 대해 설명합니다.

S3 콘솔 사용

AWS 계정의 액세스 포인트를 통해 객체를 삭제하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 액세스 포인트를 나열하려는 리전을 선택합니다.
3. 콘솔 왼쪽의 탐색 창에서 액세스 포인트를 선택합니다.
4. (선택 사항) 이름으로 액세스 포인트를 검색합니다. 선택한 AWS 리전의 액세스 포인트만 여기에 표시됩니다.
5. 관리하거나 사용할 액세스 포인트의 이름을 선택합니다.
6. 객체 탭에서 삭제하려는 객체의 이름을 선택합니다.

7. 삭제할 객체 목록을 검토하고 확인 상자에 delete를 입력합니다.
8. 객체 삭제를 선택합니다.

AWS CLI 사용

다음 `delete-object` 예시 명령은 AWS CLI를 사용하여 액세스 포인트를 통해 객체를 삭제하는 방법을 보여줍니다.

다음 명령은 액세스 포인트 *my-access-point*를 사용하여 기존 객체 `puppy.jpg`를 삭제합니다.

```
aws s3api delete-object --bucket arn:aws:s3:AWS ##:111122223333:accesspoint/my-access-point --key puppy.jpg
```

> **Note**
>
> S3는 모든 액세스 포인트에 대한 액세스 포인트 별칭을 자동으로 생성하며, 버킷 이름이 객체 수준 작업을 수행하는 데 사용되는 모든 곳에서 액세스 포인트 별칭을 사용할 수 있습니다. 자세한 내용은 액세스 포인트 별칭 섹션을 참조하세요.

자세한 내용과 예제는 AWS CLI 명령 참조에서 delete-object를 참조하십시오.

REST API 사용

REST API를 사용하여 액세스 포인트를 통해 객체를 삭제할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 DeleteObject를 참조하십시오.

S3 Access Grants를 통한 액세스 관리

최소 권한 원칙을 준수하려면 애플리케이션, 페르소나, 그룹 또는 조직 단위를 기반으로 Amazon S3 데이터에 대한 세분화된 액세스를 정의합니다. 액세스 패턴의 규모와 복잡성에 따라 다양한 접근 방식을 사용하여 Amazon S3의 데이터에 세분화된 액세스를 달성할 수 있습니다.

Amazon S3의 중소 규모 데이터 세트에 대한 액세스를 AWS Identity and Access Management(IAM) 보안 주체를 통해 관리하는 가장 간단한 방법은 IAM 권한 정책과 S3 버킷 정책을 정의하는 것입니다. 이 전략은 필요한 정책이 S3 버킷 정책 및 IAM 정책의 정책 크기 제한(각각 20KB, 5KB)과 계정당 허용되는 IAM 보안 주체 수를 준수하는 경우 효과가 있습니다.

데이터 세트와 사용 사례 수가 늘어남에 따라 더 많은 정책 공간이 필요할 수 있습니다. 정책 설명에 훨씬 더 많은 공간을 제공하는 접근 방식은 S3 액세스 포인트를 S3 버킷의 추가 엔드포인트로 사용하는 것입니다. 각 액세스 포인트는 자체 정책을 가질 수 있기 때문입니다. 계정당 AWS 리전별로 수천 개의 액세스 포인트를 보유할 수 있고 각 액세스 포인트에 대해 최대 20KB의 정책을 적용할 수 있으므로 매우 세분화된 액세스 제어 패턴을 정의할 수 있습니다. S3 액세스 포인트가 사용 가능한 정책 공간을 늘리긴 하지만, 이를 위해서는 클라이언트가 올바른 데이터 세트에 적합한 액세스 포인트를 검색할 수 있는 메커니즘이 필요합니다.

세 번째 접근 방식은 IAM 세션 브로커 패턴을 구현하는 것입니다. 이 패턴에서는 액세스 결정 로직을 구현하고 각 액세스 세션에 대해 단기 IAM 세션 자격 증명을 동적으로 생성합니다. IAM 세션 브로커 접근 방식은 임의적인 동적 권한 패턴을 지원하고 효과적으로 확장할 수 있지만 액세스 패턴 로직을 구축해야 합니다.

이러한 접근 방식을 사용하는 대신 S3 Access Grants를 사용하여 Amazon S3 데이터에 대한 액세스를 관리할 수 있습니다. S3 Access Grants는 Amazon S3의 데이터에 대한 액세스 권한을 접두사, 버킷 또는 객체별로 정의하는 간소화된 모델을 제공합니다. 또한 S3 Access Grants를 사용하여 IAM 보안 주체 모두에 액세스 권한을 부여하거나 기업 디렉터리의 사용자 또는 그룹에 직접 액세스 권한을 부여할 수 있습니다.

일반적으로 사용자와 그룹을 데이터 세트에 매핑하여 Amazon S3의 데이터에 대한 권한을 정의합니다. S3 Access Grants를 사용하여 Amazon S3 버킷 및 객체 내의 사용자 및 역할에 대한 S3 접두사의 직접 액세스 매핑을 정의할 수 있습니다. S3 Access Grants의 간소화된 액세스 체계를 사용하면 S3 접두사별로 읽기 전용, 쓰기 전용 또는 읽기-쓰기 액세스 권한을 IAM 보안 주체 모두에 부여하고 기업 디렉터리에서 사용자 또는 그룹에 직접 부여할 수 있습니다. 이러한 S3 Access Grants 기능을 사용하면 애플리케이션이 현재 인증된 애플리케이션 사용자를 대신하여 Amazon S3에 데이터를 요청할 수 있습니다.

S3 Access Grants를 AWS IAM Identity Center의 신뢰할 수 있는 ID 전파 기능과 통합하면 애플리케이션은 인증된 기업 디렉터리 사용자를 대신하여 AWS 서비스(S3 Access Grants 포함)에 직접 요청을 보낼 수 있습니다. 애플리케이션은 더 이상 사용자를 IAM 보안 주체에 먼저 매핑할 필요가 없습니다. 또한 최종 사용자 ID가 Amazon S3까지 전파되므로 어떤 사용자가 어떤 S3 객체에 액세스했는지 감사하는 작업이 간소화됩니다. 더 이상 서로 다른 사용자와 IAM 세션 간의 관계를 재구성할 필요가 없습니다. IAM Identity Center의 신뢰할 수 있는 ID 전파와 함께 S3 Access Grants를 사용하는 경우, Amazon S3의 각 AWS CloudTrail 데이터 이벤트는 대신 데이터에 액세스된 최종 사용자에 대한 직접 참조를 포함합니다.

신뢰할 수 있는 ID 전파는 연결된 AWS 서비스의 관리자가 서비스 데이터에 대한 액세스 권한을 부여하고 감사하는 데 사용할 수 있는 AWS IAM Identity Center 기능입니다. 이 데이터에 대한 액세스는 그룹 연결과 같은 사용자 속성을 기반으로 합니다. 신뢰할 수 있는 ID 전파를 설정하려면 연결된 AWS 서

비스의 관리자와 IAM Identity Center 관리자 간의 협업이 필요합니다. 자세한 내용은 사전 조건 및 고려 사항을 참조하세요.

S3 Access Grants에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- S3 Access Grants의 개념
- S3 Access Grants 및 기업 디렉터리 ID
- S3 Access Grants 시작하기
- S3 Access Grants 인스턴스 사용
- S3 Access Grants 위치 사용
- S3 Access Grants에서 권한 부여 사용
- 액세스 권한 부여를 사용하여 S3 데이터 가져오기
- S3 Access Grants 크로스 계정 액세스
- S3 Access Grants에 AWS 태그 사용
- S3 Access Grants 한도
- S3 Access Grants 통합

S3 Access Grants의 개념

S3 Access Grants 워크플로

S3 Access Grants 워크플로는 다음과 같습니다.

1. S3 Access Grants 인스턴스를 생성합니다. S3 Access Grants 인스턴스 사용을(를) 참조하세요.
2. S3 Access Grants 인스턴스 내에서 Amazon S3 데이터에 위치를 등록하고 이 위치를 AWS Identity and Access Management(IAM) 역할에 매핑합니다. 위치 등록을(를) 참조하세요.
3. 피부여자를 위한 권한 부여를 생성하여 피부여자에게 S3 리소스에 대한 액세스 권한을 부여합니다. S3 Access Grants에서 권한 부여 사용을(를) 참조하세요.
4. 피부여자는 S3 Access Grants로부터 임시 자격 증명을 요청합니다. S3 Access Grants를 통해 Amazon S3 데이터에 대한 액세스 요청을(를) 참조하세요.
5. 피부여자는 이러한 임시 자격 증명을 사용하여 S3 데이터에 액세스합니다. S3 Access Grants에서 제공하는 자격 증명을 사용하여 S3 데이터에 액세스을(를) 참조하세요.

자세한 내용은 S3 Access Grants 시작하기 섹션을 참조하세요.

S3 Access Grants 인스턴스

S3 Access Grants 인스턴스는 개별 권한 부여의 논리적 컨테이너입니다. S3 Access Grants 인스턴스를 생성할 때 AWS 리전을 지정해야 합니다. AWS 계정의 각 AWS 리전은 하나의 S3 Access Grants 인스턴스를 보유할 수 있습니다. 자세한 내용은 S3 Access Grants 인스턴스 사용 섹션을 참조하세요.

S3 Access Grants를 사용하여 회사 디렉터리의 사용자 및 그룹 ID에 대한 액세스 권한을 부여하려면 S3 Access Grants 인스턴스를 AWS IAM Identity Center 인스턴스와도 연결해야 합니다. 자세한 내용은 S3 Access Grants 및 기업 디렉터리 ID 섹션을 참조하세요.

새로 생성된 S3 Access Grants 인스턴스가 비어 있습니다. 인스턴스에 위치를 등록해야 합니다. 위치는 S3 기본 경로(`s3://`), 버킷 또는 버킷 내 접두사일 수 있습니다. 위치를 하나 이상 등록하면 등록된 이 위치의 데이터에 액세스할 수 있도록 액세스 권한 부여를 생성할 수 있습니다.

위치

S3 Access Grants 위치는 버킷 또는 접두사를 AWS Identity and Access Management(IAM) 역할에 매핑합니다. S3 Access Grants는 이 IAM 역할을 맡아 해당 특정 위치에 액세스하는 피부여자에게 임시 자격 증명을 제공합니다. 액세스 권한 부여를 생성하려면 먼저 S3 Access Grants 인스턴스에 위치를 하나 이상 등록해야 합니다.

기본 위치(`s3://`)를 등록하고 이를 IAM 역할에 매핑하는 것이 좋습니다. 기본 S3 경로(`s3://`)의 위치는 계정의 AWS 리전에 있는 모든 S3 버킷에 대한 액세스를 포함합니다. 액세스 권한 부여를 생성할 때 기본 위치 내의 버킷, 접두사 또는 객체로 권한 부여 범위를 좁힐 수 있습니다.

더 복잡한 액세스 관리 사용 사례에서는 기본 위치보다 더 많이 등록해야 할 수도 있습니다. 이러한 사용 사례의 몇 가지 예는 다음과 같습니다.

- *`amzn-s3-demo-bucket`*은 S3 Access Grants 인스턴스의 등록된 위치로서 IAM 역할이 매핑되어 있지만 이 IAM 역할은 버킷 내 특정 접두사에 대한 액세스가 거부되었다고 가정해 보겠습니다. 이 경우 IAM 역할이 액세스할 수 없는 접두사를 별도의 위치로 등록하고 필요한 액세스 권한이 있는 다른 IAM 역할에 해당 위치를 매핑할 수 있습니다.
- 가상 프라이빗 클라우드(VPC) 엔드포인트 내 사용자로만 액세스 권한을 제한하는 권한 부여를 생성한다고 가정해 보겠습니다. 이 경우, IAM 역할이 VPC 엔드포인트에 대한 액세스를 제한하는 버킷 위치를 등록할 수 있습니다. 나중에 피부여자가 S3 Access Grants에 자격 증명을 요청하면 S3 Access Grants는 해당 위치의 IAM 역할을 맡아 임시 자격 증명을 제공합니다. 이 자격 증명은 호출자가 VPC 엔드포인트 내에 있지 않는 한 특정 버킷에 대한 액세스를 거부합니다. 이 거부 권한은 권한 부여에 지정된 일반 READ, WRITE 또는 READWRITE 권한에 추가로 적용됩니다.

사용 사례에서 S3 Access Grants 인스턴스에 여러 위치를 등록해야 하는 경우 다음 중 하나를 등록할 수 있습니다.

- 기본 S3 위치(`s3://`)
- 하나의 버킷(예: *`amzn-s3-demo-bucket`*) 또는 여러 버킷
- 하나의 버킷과 접두사(예: *`amzn-s3-demo-bucket`*`/prefix*`) 또는 여러 접두사

S3 Access Grants 인스턴스에 등록할 수 있는 최대 위치 수는 S3 Access Grants 한도 섹션을 참조하세요. S3 Access Grants 위치 등록에 대한 자세한 내용은 위치 등록 섹션을 참조하세요.

S3 Access Grants 인스턴스에서 첫 번째 위치를 등록한 후에도 인스턴스에는 여전히 개별 액세스 권한이 부여되지 않습니다. 따라서 S3 데이터에 대한 액세스 권한은 아직 부여되지 않았습니다. 이제 액세스 권한 부여를 생성하여 액세스 권한을 부여할 수 있습니다. 권한 부여 생성에 대한 자세한 내용은 S3 Access Grants에서 권한 부여 사용 섹션을 참조하세요.

권한 부여

S3 Access Grants 인스턴스 내 개별 권한 부여를 통해 특정 자격 증명(IAM 보안 주체나 기업 디렉터리의 사용자 또는 그룹)이 S3 Access Grants 인스턴스에 등록된 위치 내에서 액세스 권한을 부여받을 수 있습니다.

권한 부여를 생성할 때 등록된 위치 전체에 대한 액세스 권한을 부여할 필요는 없습니다. 특정 위치 내에서 권한 부여의 액세스 범위를 좁힐 수 있습니다. 등록된 위치가 기본 S3 경로(`s3://`)인 경우 권한 부여 범위를 버킷, 버킷 내 접두사 또는 특정 객체로 좁혀야 합니다. 권한 부여의 등록 위치가 버킷 또는 접두사인 경우 전체 버킷 또는 접두사에 대한 액세스 권한을 부여하거나 선택적으로 접두사, 하위 접두사 또는 객체로 권한 부여 범위를 좁힐 수 있습니다.

또한 권한 부여에서 권한 부여의 액세스 수준을 READ, WRITE 또는 READWRITE로 설정합니다. 기업 디렉터리 그룹 `01234567-89ab-cdef-0123-456789abcdef`에 버킷 `s3://`*`amzn-s3-demo-bucket`*`/projects/items/*`에 대한 READ 액세스 권한을 부여하는 권한 부여가 있다고 가정해 보겠습니다. 이 그룹의 사용자는 이름이 지정된 버킷 *`amzn-s3-demo-bucket`*에서 접두사 `projects/items/`로 시작하는 객체 키 이름을 가진 모든 객체에 대해 READ 액세스 권한을 가질 수 있습니다.

S3 Access Grants 인스턴스에서 생성할 수 있는 최대 권한 부여 수는 S3 Access Grants 한도 섹션을 참조하세요. 권한 부여 생성에 대한 자세한 내용은 권한 부여 생성 섹션을 참조하세요.

S3 Access Grants 임시 자격 증명

권한 부여를 생성한 후, 권한 부여에 지정된 자격 증명을 사용하는 승인된 애플리케이션은 적시 액세스 자격 증명을 요청할 수 있습니다. 이를 위해 애플리케이션은 GetDataAccess S3 API 작업을

직접적으로 호출합니다. 피부여자는 이 API 작업을 사용하여 공유한 S3 데이터에 대한 액세스를 요청할 수 있습니다.

S3 Access Grants 인스턴스는 보유한 권한 부여와 비교하여 `GetDataAccess` 요청을 평가합니다. 요청자에 대한 일치하는 권한 부여가 있는 경우 S3 Access Grants는 일치하는 권한 부여의 등록된 위치와 연결된 IAM 역할을 맡습니다. S3 Access Grants는 권한 부여 범위에 지정된 S3 버킷, 접두사 또는 객체에만 액세스할 수 있도록 임시 자격 증명의 권한 범위를 지정합니다.

임시 액세스 자격 증명의 만료 시간은 기본적으로 1시간이지만 15분에서 12시간 사이의 값으로 설정할 수 있습니다. AssumeRole API 참조에서 최대 지속 시간 세션을 참조하세요.

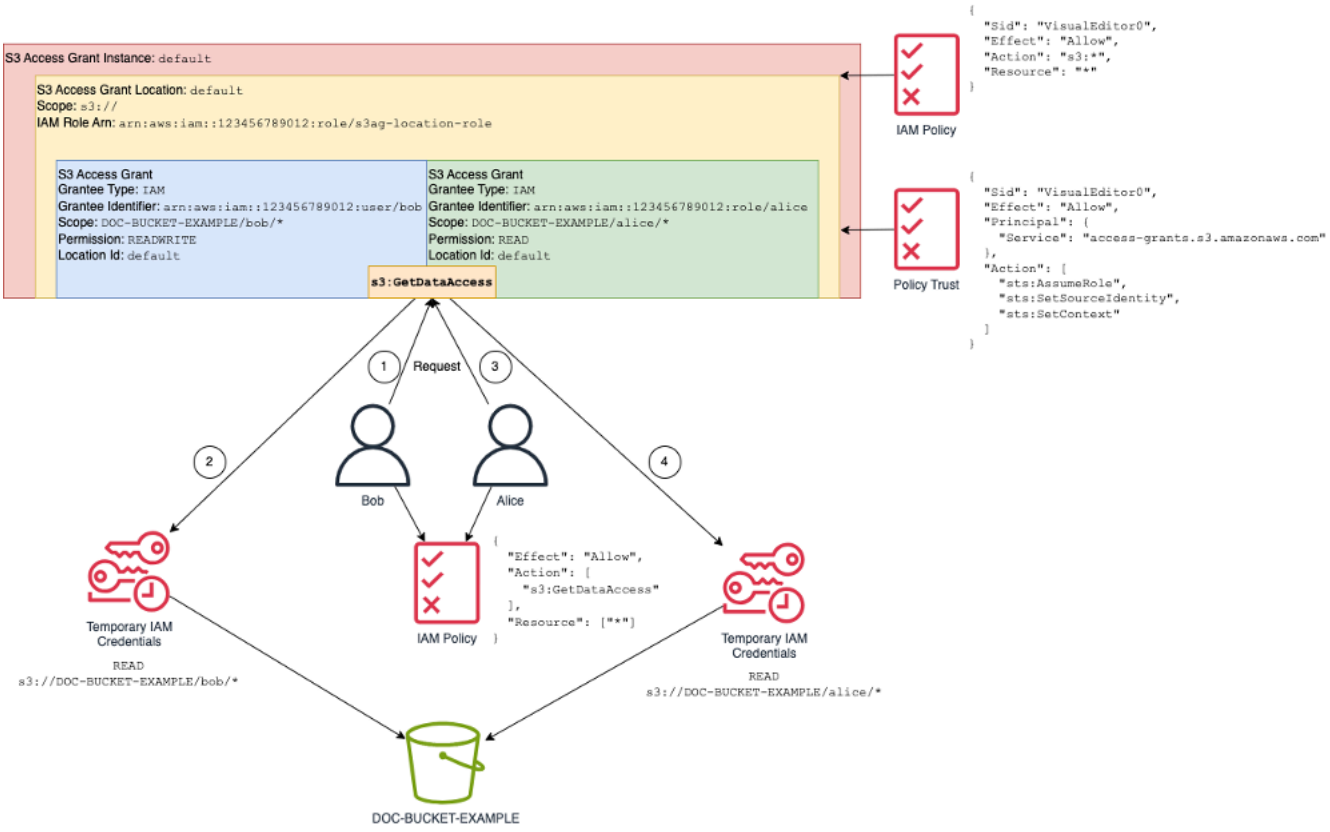
작동 방법

다음 다이어그램에서는 범위가 `s3://`인 기본 Amazon S3 위치가 `s3ag-location-role` IAM 역할과 함께 등록되어 있습니다. 이 IAM 역할에는 S3 Access Grants를 통해 자격 증명을 획득하면 계정 내에서 Amazon S3 작업을 수행할 권한이 있습니다.

이 위치 내에서 2명의 IAM 사용자에 대해 두 개의 개별 액세스 권한 부여가 생성되어 있습니다. IAM 사용자 Bob에게는 버킷의 `DOC-BUCKET-EXAMPLE` 버킷에서 `bob/` 접두사에 대한 `READ` 및 `WRITE` 액세스 권한이 모두 부여되었습니다. 또 다른 IAM 역할인 Alice에게는 `DOC-BUCKET-EXAMPLE` 버킷의 `alice/` 접두사에 대한 `READ` 액세스 권한만 부여되었습니다. Bob이 `DOC-BUCKET-EXAMPLE` 버킷의 `bob/` 접두사에 액세스할 수 있는 권한이 파란색으로 표시되어 있습니다. Alice가 `DOC-BUCKET-EXAMPLE` 버킷의 `alice/` 접두사에 액세스할 수 있는 권한은 초록색으로 표시되어 있습니다.

Bob이 데이터에 `READ` 작업을 수행할 때가 되면 Bob의 권한 부여가 있는 위치와 연결된 IAM 역할이 S3 Access Grants GetDataAccess API 작업을 호출합니다. Bob이 `s3://DOC-BUCKET-EXAMPLE/bob/*`으로 시작하는 S3 접두사 또는 객체에 `READ` 작업을 시도하면 `GetDataAccess` 요청은 `s3://DOC-BUCKET-EXAMPLE/bob/*`에 대한 권한이 있는 임시 IAM 세션 자격 증명 세트를 반환합니다. 이와 유사하게 Bob은 `s3://DOC-BUCKET-EXAMPLE/bob/*`으로 시작하는 모든 S3 접두사 또는 객체에 `WRITE` 작업을 수행할 수 있습니다. 권한 부여가 이 작업도 허용하기 때문입니다.

마찬가지로 Alice는 `s3://DOC-BUCKET-EXAMPLE/alice/`로 시작하는 모든 것에 대해 `READ` 작업을 수행할 수 있습니다. 하지만 `s3://`의 버킷, 접두사 또는 객체에 대해 `WRITE` 작업을 수행하려고 시도하면 Alice에게 데이터에 대한 `WRITE` 액세스를 제공하는 권한 부여가 없기 때문에 액세스 거부됨(403 금지됨 오류가 발생합니다. 또한 Alice가 `s3://DOC-BUCKET-EXAMPLE/alice/` 외부의 데이터에 대해 어떤 수준의 액세스(`READ` 또는 `WRITE`)를 요청해도 액세스 거부됨 오류가 다시 발생합니다.



이 패턴은 많은 사용자 및 버킷으로 확장되며 이러한 권한의 관리를 단순화합니다. 개별 사용자-접두사 액세스 관계를 추가하거나 제거할 때마다 크기가 매우 클 수도 있는 S3 버킷 정책을 편집하는 대신 개별 권한을 추가하고 제거할 수 있습니다.

S3 Access Grants 및 기업 디렉터리 ID

Amazon S3 Access Grants를 사용하여 동일한 AWS 계정 및 서로 다른 계정에서 AWS Identity and Access Management(IAM) 보안 주체(사용자 또는 역할)에 대한 액세스 권한을 부여할 수 있습니다. 하지만 대부분의 경우 데이터에 액세스하는 엔터티는 기업 디렉터리의 최종 사용자입니다. IAM 보안 주체에 액세스 권한을 부여하는 대신 S3 Access Grants를 사용하여 기업 사용자 및 그룹에 직접 액세스 권한을 부여할 수 있습니다. S3 Access Grants를 사용하면 기업 애플리케이션을 통해 S3 데이터에 액세스하기 위해 더 이상 기업 ID를 중간의 IAM 보안 주체에 매핑할 필요가 없습니다.

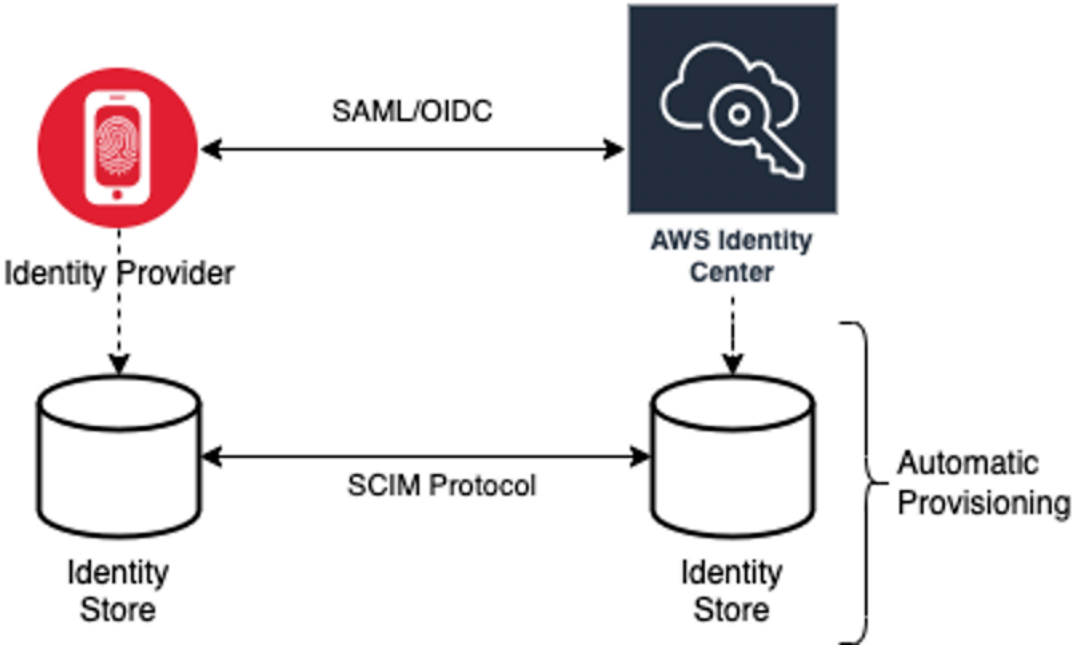
데이터에 대한 최종 사용자 ID 액세스를 지원하는 이 새로운 기능은 S3 Access Grants 인스턴스를 AWS IAM Identity Center 인스턴스와 연결하여 제공됩니다. IAM Identity Center는 표준 기반 ID 제공 업체를 지원하며 AWS 내에서 최종 사용자 ID를 지원하는 S3 Access Grants를 비롯한 모든 서비스 또는 기능의 허브 역할을 합니다. IAM Identity Center는 신뢰할 수 있는 ID 전파 기능을 통해 기업 ID에 대한 인증 지원을 제공합니다. 자세한 내용은 Trusted identity propagation across applications를 참조하세요.

S3 Access Grants의 인력 ID 지원을 시작하려면 먼저 IAM Identity Center에서 기업 ID 제공업체와 IAM Identity Center 간에 ID 프로비저닝을 구성해야 합니다. IAM Identity Center는 Okta, Microsoft Entra ID(구 Azure Active Directory)와 같은 기업 ID 제공업체 또는 System for Cross-domain Identity Management(SCIM) 프로토콜을 지원하는 기타 외부 ID 제공업체(IdP)를 지원합니다. IAM Identity Center를 IdP에 연결하고 자동 프로비저닝을 활성화하면 IdP의 사용자 및 그룹이 IAM Identity Center의 ID 저장소에 동기화됩니다. 이 단계가 끝나면 IAM Identity Center는 사용자 및 그룹을 자체적으로 파악하므로 S3 Access Grants와 같은 다른 AWS 서비스 및 기능을 사용하여 해당 사용자와 그룹을 참조할 수 있습니다. IAM Identity Center 자동 프로비저닝을 구성하는 방법에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 자동 프로비저닝을 참조하세요.

IAM Identity Center는 AWS Organizations와 통합되어 있으므로 각 계정을 수동으로 구성하지 않고도 여러 AWS 계정의 권한을 중앙에서 관리할 수 있습니다. 일반적인 조직에서는 ID 관리자가 전체 조직에 대해 하나의 IAM Identity Center 인스턴스를 단일 ID 동기화 지점으로 구성합니다. 이 IAM Identity Center 인스턴스는 일반적으로 조직의 전용 AWS 계정에서 실행됩니다. 이 일반적인 구성에서는 조직 내 모든 AWS 계정에서 S3 Access Grants의 사용자 및 그룹 ID를 참조할 수 있습니다.

하지만 AWS Organizations 관리자가 아직 중앙 IAM Identity Center 인스턴스를 구성하지 않은 경우 S3 Access Grants 인스턴스와 동일한 계정에 로컬 인스턴스를 생성할 수 있습니다. 개념 증명 또는 로컬 개발 사용 사례에서는 이러한 구성이 더 일반적입니다. 모든 경우에 IAM Identity Center 인스턴스는 해당 인스턴스가 연결될 S3 Access Grants 인스턴스와 동일한 AWS 리전에 있어야 합니다.

외부 IdP를 사용한 IAM Identity Center 구성을 보여주는 다음 다이어그램에서 IdP는 IdP의 ID 저장소를 IAM Identity Center의 ID 저장소와 동기화하도록 SCIM으로 구성되어 있습니다.



S3 Access Grants와 함께 기업 디렉터리 ID를 사용하려면 다음을 수행하세요.

- IAM Identity Center에서 자동 프로비저닝을 설정하여 IdP의 사용자 및 그룹 정보를 IAM Identity Center로 동기화합니다.
- IAM Identity Center 내에서 외부 ID 소스를 신뢰할 수 있는 토큰 발급자로 구성합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서의 Trusted identity propagation across applications를 참조하세요.
- S3 Access Grants 인스턴스를 IAM Identity Center 인스턴스와 연결합니다. S3 Access Grants 인스턴스를 생성할 때 이 작업을 수행할 수 있습니다. 이미 S3 Access Grants 인스턴스를 생성한 경우 IAM Identity Center 인스턴스 연결 또는 연결 해제 섹션을 참조하세요.

디렉터리 ID가 S3 데이터에 액세스하는 방법

외부 IdP(예: Okta)와 통합되어 사용자를 인증하는 기업 애플리케이션(예: 문서 뷰어 애플리케이션)을 통해 S3 데이터에 액세스해야 하는 회사 디렉터리 사용자가 있다고 가정해 보겠습니다. 이러한 애플리케이션의 사용자 인증은 일반적으로 사용자 웹 브라우저의 리디렉션을 통해 이루어집니다. 디렉터리의 사용자는 IAM 보안 주체가 아니므로 애플리케이션에는 사용자를 대신하여 S3 데이터에 대한 액세스 자격 증명을 가져오기 위해 S3 Access Grants `GetDataAccess` API 작업을 호출하는 데 사용할 수 있는 IAM 자격 증명이 필요합니다. 자격 증명을 직접 얻는 IAM 사용자 및 역할과 달리, 애플리케이션에는 IAM 역할에 매핑되지 않은 디렉터리 사용자를 표현할 방법이 필요합니다. 그래야 사용자가 S3 Access Grants를 통해 데이터 액세스 권한을 받을 수 있습니다.

인증된 디렉터리 사용자에서 디렉터리 사용자를 대신하여 S3 Access Grants에 요청할 수 있는 IAM 호출자로의 전환은 애플리케이션에서 IAM Identity Center의 신뢰할 수 있는 토큰 발급자 기능을 통해 이루어집니다. 디렉터리 사용자를 인증한 애플리케이션은 IdP(예: Okta)의 ID 토큰을 받게 되며, 이 토큰은 Okta에 따라 디렉터리 사용자를 나타냅니다. IAM Identity Center의 신뢰할 수 있는 토큰 발급자 구성을 통해 애플리케이션은 이 Okta 토큰(Okta 테넌트는 '신뢰할 수 있는 발급자'로 구성됨)을 IAM Identity Center의 다른 ID 토큰으로 교환할 수 있습니다. 이 ID 토큰은 AWS 서비스 내에서 디렉터리 사용자를 안전하게 나타냅니다. 그러면 데이터 애플리케이션이 IAM 역할을 맡아 IAM Identity Center의 디렉터리 사용자 토큰을 추가 컨텍스트로 제공합니다. 애플리케이션은 생성된 IAM 세션을 사용하여 S3 Access Grants를 호출할 수 있습니다. 토큰은 애플리케이션의 ID(IAM 보안 주체 자체)와 디렉터리 사용자의 ID를 모두 나타냅니다.

이 전환의 주요 단계는 토큰 교환입니다. 애플리케이션은 IAM Identity Center에서 `CreateTokenWithIAM` API 작업을 호출하여 이 토큰 교환을 수행합니다. 물론 이 역시 AWS API 직접 호출이며 서명하려면 IAM 보안 주체가 필요합니다. 이 요청을 하는 IAM 보안 주체는 일반적으로 애플리케이션과 연결된 IAM 역할입니다. 예를 들어, 애플리케이션이 Amazon EC2에서 실행되는 경우

`CreateTokenWithIAM` 요청은 일반적으로 애플리케이션이 실행되는 EC2 인스턴스에 연결된 IAM 역할에 의해 수행됩니다. 성공적인 `CreateTokenWithIAM` 호출의 결과로 새 ID 토큰이 생성되며, 이 토큰은 AWS 서비스 내에서 인식됩니다.

애플리케이션이 디렉터리 사용자를 대신하여 `GetDataAccess` 호출할 수 있으려면 먼저 애플리케이션이 디렉터리 사용자의 ID가 포함된 IAM 세션을 확보해야 합니다. 애플리케이션은 디렉터리 사용자를 위한 IAM Identity Center 토큰을 추가 ID 컨텍스트로 포함하는 AWS Security Token Service(AWS STS) `AssumeRole` 요청을 통해 이 작업을 수행합니다. 이 추가 컨텍스트를 통해 IAM Identity Center는 디렉터리 사용자의 ID를 다음 단계로 전파할 수 있습니다. 애플리케이션이 맡는 IAM 역할은 `GetDataAccess` 작업을 호출하기 위해 IAM 권한이 필요한 역할입니다.

디렉터리 사용자를 위한 IAM Identity Center 토큰을 추가 컨텍스트로 사용하여 ID 보유자 IAM 역할을 맡은 애플리케이션은 이제 인증된 디렉터리 사용자를 대신하여 `GetDataAccess`에 서명된 요청을 보내는 데 필요한 모든 것을 갖추게 되었습니다.

토큰 전파는 다음 단계를 기반으로 합니다.

IAM Identity Center 애플리케이션 생성

먼저 IAM Identity Center에서 새 애플리케이션을 생성합니다. 이 애플리케이션은 사용할 수 있는 애플리케이션 설정 유형을 IAM Identity Center에서 식별할 수 있는 템플릿을 사용합니다. 애플리케이션을 생성하기 위한 명령을 실행하려면 IAM Identity Center 인스턴스 Amazon 리소스 이름(ARN), 애플리케이션 이름, 애플리케이션 공급자 ARN을 제공해야 합니다. 애플리케이션 공급자는 애플리케이션이 IAM Identity Center에 호출하는 데 사용할 SAML 또는 OAuth 애플리케이션 공급자입니다.

다음 예시 명령을 사용하려면 *`user input placeholders`*를 실제 정보로 대체하세요.

```
aws sso-admin create-application \
 --instance-arn "arn:aws:sso:::instance/ssoins-ssoins-1234567890abcdef" \
 --application-provider-arn "arn:aws:sso::aws:applicationProvider/custom" \
 --name MyDataApplication
```

응답:

```
{
   "ApplicationArn": "arn:aws:sso::123456789012:application/ssoins-
ssoins-1234567890abcdef/apl-abcd1234a1b2c3d"
}
```

신뢰할 수 있는 토큰 발급자 생성

IAM Identity Center 애플리케이션이 준비되었으므로 다음 단계는 IdP의 `IdToken` 값을 IAM Identity Center 토큰으로 교환하는 데 사용할 신뢰할 수 있는 토큰 발급자를 구성하는 것입니다. 이 단계에서는 다음 항목을 제공해야 합니다.

- ID 제공업체 발급자 URL
- 신뢰할 수 있는 토큰 발급자 이름
- 클레임 속성 경로
- ID 저장소 속성 경로
- JSON Web Key Set(JWKS) 검색 옵션

클레임 속성 경로는 ID 저장소 속성에 매핑하는 데 사용되는 ID 제공업체 속성입니다. 일반적으로 클레임 속성 경로는 사용자의 이메일 주소이지만 다른 속성을 사용하여 매핑을 수행할 수 있습니다.

다음 정보를 사용하여 `oidc-configuration.json`이라는 파일을 생성합니다. 이 파일을 사용하려면 *`user input placeholders`*를 실제 정보로 대체합니다.

```
{
  "OidcJwtConfiguration":
     {
      "IssuerUrl": "https://login.microsoftonline.com/a1b2c3d4-abcd-1234-b7d5-
b154440ac123/v2.0",
      "ClaimAttributePath": "preferred_username",
      "IdentityStoreAttributePath": "userName",
      "JwksRetrievalOption": "OPEN_ID_DISCOVERY"
     }
}
```

신뢰할 수 있는 토큰 발급자를 생성하려면 다음 명령을 실행합니다. 이 예 명령을 사용하려면 *`user input placeholders`*를 실제 정보로 대체하세요.

```
aws sso-admin create-trusted-token-issuer \
  --instance-arn "arn:aws:sso:::instance/ssoins-1234567890abcdef" \
  --name MyEntraIDTrustedIssuer \
  --trusted-token-issuer-type OIDC_JWT \
  --trusted-token-issuer-configuration file://./oidc-configuration.json
```

응답

```
{
```

```
  "TrustedTokenIssuerArn": "arn:aws:sso::123456789012:trustedTokenIssuer/
ssoins-1234567890abcdef/tti-43b4a822-1234-1234-1234-a1b2c3d41234"
}
```

IAM Identity Center 애플리케이션을 신뢰할 수 있는 토큰 발급자와 연결

신뢰할 수 있는 토큰 발급자가 작동하려면 몇 가지 구성 설정이 더 필요합니다. 신뢰할 수 있는 토큰 발급자가 신뢰할 대상을 설정합니다. 대상은 키로 식별되는 IdToken 내부의 값이며 ID 제공업체 설정에서 찾을 수 있습니다. 예시:

```
1234973b-abcd-1234-abcd-345c5a9c1234
```

다음 콘텐츠가 포함된 grant.json이라는 파일을 생성합니다. 이 파일을 사용하려면 대상을 ID 제공업체 설정과 일치하도록 변경하고 이전 명령에서 반환된 신뢰할 수 있는 토큰 발급자 ARN을 제공합니다.

```
{
   "JwtBearer":
     {
       "AuthorizedTokenIssuers":
         [
           {
             "TrustedTokenIssuerArn": "arn:aws:sso::123456789012:trustedTokenIssuer/
ssoins-1234567890abcdef/tti-43b4a822-1234-1234-1234-a1b2c3d41234",
               "AuthorizedAudiences":
                 [
                   "1234973b-abcd-1234-abcd-345c5a9c1234"
                 ]
            }
         ]
     }
 }
```

다음 예시 명령을 실행합니다. 이 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체합니다.

```
aws sso-admin put-application-grant \
  --application-arn "arn:aws:sso::123456789012:application/ssoins-
ssoins-1234567890abcdef/apl-abcd1234a1b2c3d" \
  --grant-type "urn:ietf:params:oauth:grant-type:jwt-bearer" \
  --grant file://./grant.json \
```

이 명령은 신뢰할 수 있는 토큰 발급자가 `grant.json` 파일의 대상을 신뢰하고 `jwt-bearer` 유형의 토큰을 교환하기 위해 첫 단계에서 만든 애플리케이션과 이 대상을 연결하도록 구성을 설정합니다. `urn:ietf:params:oauth:grant-type:jwt-bearer` 문자열은 임의의 문자열이 아닙니다. 이 문자열은 OAuth JSON Web Token(JWT) 어설션 프로필에 등록된 네임스페이스입니다. 이 네임스페이스에 대한 자세한 내용은 RFC 7523에서 확인할 수 있습니다.

그런 다음, 다음 명령을 사용하여 ID 제공업체로부터 `IdToken` 값을 교환할 때 신뢰할 수 있는 토큰 발급자가 포함할 범위를 설정합니다. S3 Access Grants의 경우 `--scope` 파라미터의 값은 `s3:access_grants:read_write`입니다.

```
aws sso-admin put-application-access-scope \
  --application-arn "arn:aws:sso::111122223333:application/ssoins-
ssoins-111122223333abcdef/apl-abcd1234a1b2c3d" \
  --scope "s3:access_grants:read_write"
```

마지막 단계는 IAM Identity Center 애플리케이션에 리소스 정책을 연결하는 것입니다. 이 정책을 통해 애플리케이션 IAM 역할은 `sso-oauth:CreateTokenWithIAM` API 작업에 요청을 보내고 IAM Identity Center로부터 `IdToken` 값을 수신할 수 있습니다.

다음 콘텐츠가 포함된 `authentication-method.json`이라는 파일을 생성합니다. *123456789012*을 계정 ID로 바꿉니다.

```
{
   "Iam":
       {
         "ActorPolicy":
             {
                "Version": "2012-10-17",
                    "Statement":
                    [
                        {
                           "Effect": "Allow",
                            "Principal":
                            {
                              "AWS": "arn:aws:iam::123456789012:role/webapp"
                            },
                           "Action": "sso-oauth:CreateTokenWithIAM",
                            "Resource": "*"
                        }
                    ]
                }
            }
```

```
        }
```

IAM Identity Center 애플리케이션에 정책을 연결하려면 다음 명령을 실행합니다.

```
aws sso-admin put-application-authentication-method \
   --application-arn "arn:aws:sso::123456789012:application/ssoins-
ssoins-1234567890abcdef/apl-abcd1234a1b2c3d" \
   --authentication-method-type IAM \
   --authentication-method file://./authentication-method.json
```

웹 애플리케이션을 통해 디렉터리 사용자와 함께 S3 Access Grants를 사용하기 위한 구성 설정이 완료되었습니다. 애플리케이션에서 직접 이 설정을 테스트하거나 IAM Identity Center 애플리케이션 정책에서 허용된 IAM 역할에서 다음 명령을 사용하여 `CreateTokenWithIAM` API 작업을 호출할 수 있습니다.

```
aws sso-oidc create-token-with-iam \
   --client-id "arn:aws:sso::123456789012:application/ssoins-ssoins-1234567890abcdef/
apl-abcd1234a1b2c3d"  \
   --grant-type urn:ietf:params:oauth:grant-type:jwt-bearer \
   --assertion IdToken
```

응답은 다음과 유사합니다.

```
{
    "accessToken": "<suppressed long string to reduce space>",
    "tokenType": "Bearer",
    "expiresIn": 3600,
    "refreshToken": "<suppressed long string to reduce space>",
    "idToken": "<suppressed long string to reduce space>",
    "issuedTokenType": "urn:ietf:params:oauth:token-type:refresh_token",
    "scope": [
      "sts:identity_context",
      "s3:access_grants:read_write",
      "openid",
      "aws"
    ]
}
```

base64로 인코딩된 `IdToken` 값을 디코딩하면 키-값 쌍이 JSON 형식으로 표시됩니다. `sts:identity_context` 키에는 디렉터리 사용자의 ID 정보를 포함하기 위해 애플리케이션이 `sts:AssumeRole` 요청에 전송해야 하는 값이 들어 있습니다. 다음은 디코딩된 `IdToken`의 예입니다.

```
{
    "aws:identity_store_id": "d-996773e796",
    "sts:identity_context": "AQoJb3JpZ2luX2VjEOTtl;<SUPRESSED>",
    "sub": "83d43802-00b1-7054-db02-f1d683aacba5",
    "aws:instance_account": "123456789012",
    "iss": "https://identitycenter.amazonaws.com/ssoins-1234567890abcdef",
    "sts:audit_context": "AQoJb3JpZ2luX2VjEOT<SUPRESSED>==",
    "aws:identity_store_arn": "arn:aws:identitystore::232642235904:identitystore/
d-996773e796",
    "aud": "abcd12344U0gi7n4Yyp0-WV1LWNlbnRyYWwtMQ",
    "aws:instance_arn": "arn:aws:sso:::instance/ssoins-6987d7fb04cf7a51",
    "aws:credential_id": "EXAMPLEHI5glPh40y9TpApJn8...",
    "act": {
       "sub": "arn:aws:sso::232642235904:trustedTokenIssuer/
ssoins-6987d7fb04cf7a51/43b4a822-1020-7053-3631-cb2d3e28d10e"
    },
    "auth_time": "2023-11-01T20:24:28Z",
    "exp": 1698873868,
    "iat": 1698870268
}
```

`sts:identity_context`에서 값을 가져오고 이 정보를 `sts:AssumeRole` 호출에 전달할 수 있습니다. 다음은 이 구문의 CLI 예시입니다. 수임되는 역할은 `s3:GetDataAccess` 호출 권한이 있는 임시 역할입니다.

```
aws sts assume-role \
   --role-arn "arn:aws:iam::123456789012:role/temp-role" \
   --role-session-name "TempDirectoryUserRole" \
   --provided-contexts ProviderArn="arn:aws:iam::aws:contextProvider/
IdentityCenter",ContextAssertion="value from sts:identity_context"
```

이제 이 호출에서 받은 자격 증명을 사용하여 `s3:GetDataAccess` API 작업을 호출하고 S3 리소스 액세스 권한이 있는 최종 자격 증명을 받을 수 있습니다.

S3 Access Grants 시작하기

Amazon S3 Access Grants는 S3 데이터에 대한 확장 가능한 액세스 제어 솔루션을 제공하는 Amazon S3 기능입니다. S3 Access Grants는 S3 자격 증명의 벤더입니다. 즉, S3 Access Grants에 권한 부여 목록 및 수준을 등록하게 됩니다. 이후 사용자나 클라이언트가 S3 데이터에 액세스해야 하는 경우 먼저 S3 Access Grants에 자격 증명을 요청합니다. 액세스를 승인하는 권한 부여가 있는 경우, S3 Access Grants는 최소 권한의 임시 액세스 자격 증명을 제공합니다. 그러면 사용자 또는 클라이언트가

S3 Access Grants에서 제공한 자격 증명을 사용하여 S3 데이터에 액세스할 수 있습니다. 이를 염두에 두고 S3 데이터 요구 사항에 복잡하거나 규모가 큰 권한 구성이 필요한 경우 S3 Access Grants를 사용하여 사용자, 그룹, 역할 및 애플리케이션을 위한 S3 데이터 권한의 규모를 조정할 수 있습니다.

대부분의 사용 사례에서는 AWS Identity and Access Management(IAM) 버킷 정책 또는 IAM ID 기반 정책을 사용하여 S3 데이터에 대한 액세스 제어를 관리할 수 있습니다.

하지만 다음과 같이 복잡한 S3 액세스 제어 요구 사항이 있는 경우 S3 Access Grants를 사용하면 큰 이점을 얻을 수 있습니다.

- 20KB의 버킷 정책 크기 한도에 도달한 경우
- 분석 및 빅 데이터를 위해 예컨대 사람 ID, Microsoft Entra ID(구 Azure Active Directory), Okta 또는 Ping 사용자 및 그룹에 S3 데이터에 대한 액세스 권한을 부여하는 경우
- IAM 정책을 자주 업데이트하지 않고도 크로스 계정 액세스를 제공해야 하는 경우
- 데이터가 행 및 열 형식의 정형 데이터가 아닌 객체 수준의 비정형 데이터인 경우

S3 Access Grants 워크플로는 다음과 같습니다.

단계	설명
1	S3 Access Grants 인스턴스 생성 시작하려면 개별 액세스 권한을 포함할 S3 Access Grants 인스턴스를 시작합니다.
2	위치 등록 다음으로, S3 데이터 위치(예: 기본값 `s3://`)를 등록한 다음 S3 데이터 위치에 대한 액세스를 제공할 때 S3 Access Grants가 맡는 기본 IAM 역할을 지정합니다. 또한 특정 버킷이나 접두사에 사용자 지정 위치를 추가하고 이를 사용자 지정 IAM 역할에 매핑할 수 있습니다.
3	권한 부여 생성 개별 권한 부여를 생성합니다. 이러한 권한 부여에 등록된 S3 위치, 위치 내 데이터 액세스 범위, 피부여자의 ID, 피부여자의 액세스 수준(`READ`, `WRITE` 또는 `READWRITE`)을 지정합니다.

단계	설명
4	S3 데이터에 대한 액세스 요청 사용자, 애플리케이션 및 AWS 서비스에서 S3 데이터에 액세스하려는 경우 먼저 액세스를 요청합니다. S3 Access Grants는 요청을 승인해야 하는지를 결정합니다. 액세스를 승인하는 권한 부여가 있는 경우 S3 Access Grants는 해당 권한 부여와 연결된 등록 위치의 IAM 역할을 사용하여 요청자에게 임시 자격 증명을 다시 제공합니다.
5	S3 데이터에 액세스 애플리케이션이 S3 Access Grants에서 제공하는 임시 자격 증명을 사용하여 S3 데이터에 액세스합니다.

S3 Access Grants 인스턴스 사용

AmazonS3 Access Grants 사용을 시작하려면 먼저 S3 Access Grants 인스턴스를 생성해야 합니다. 계정당 AWS 리전별로 S3 Access Grants 인스턴스를 하나만 생성할 수 있습니다. S3 Access Grants 인스턴스는 등록된 위치 및 권한 부여를 포함하는 S3 Access Grants 리소스의 컨테이너 역할을 합니다.

S3 Access Grants를 사용하면 AWS Identity and Access Management(IAM) 사용자 및 역할에 대한 S3 데이터에 권한 부여를 생성할 수 있습니다. AWS IAM Identity Center에 기업 ID 디렉터리를 추가한 경우 기업 디렉터리의 이 IAM Identity Center 인스턴스를 S3 Access Grants 인스턴스와 연결할 수 있습니다. 이렇게 하면 기업 사용자 및 그룹에 대한 액세스 권한 부여를 생성할 수 있습니다. 아직 기업 디렉터리를 IAM Identity Center에 추가하지 않은 경우 나중에 S3 Access Grants 인스턴스를 IAM Identity Center 인스턴스와 연결할 수 있습니다.

주제

- S3 Access Grants 인스턴스 생성
- S3 Access Grants 인스턴스의 세부 정보 가져오기
- S3 Access Grants 인스턴스 나열
- IAM Identity Center 인스턴스 연결 또는 연결 해제
- S3 Access Grants 인스턴스 삭제

S3 Access Grants 인스턴스 생성

AmazonS3 Access Grants 사용을 시작하려면 먼저 S3 Access Grants 인스턴스를 생성해야 합니다. 계정당 AWS 리전별로 S3 Access Grants 인스턴스를 하나만 생성할 수 있습니다. S3 Access Grants 인스턴스는 등록된 위치 및 권한 부여를 포함하는 S3 Access Grants 리소스의 컨테이너 역할을 합니다.

S3 Access Grants를 사용하면 AWS Identity and Access Management(IAM) 사용자 및 역할에 대한 S3 데이터에 권한 부여를 생성할 수 있습니다. AWS IAM Identity Center에 기업 ID 디렉터리를 추가한 경우 기업 디렉터리의 이 IAM Identity Center 인스턴스를 S3 Access Grants 인스턴스와 연결할 수 있습니다. 이렇게 하면 기업 사용자 및 그룹에 대한 액세스 권한 부여를 생성할 수 있습니다. 아직 기업 디렉터리를 IAM Identity Center에 추가하지 않은 경우 나중에 S3 Access Grants 인스턴스를 IAM Identity Center 인스턴스와 연결할 수 있습니다.

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), Amazon S3 REST API, AWS SDK를 사용하여 S3 Access Grants 인스턴스를 생성할 수 있습니다.

S3 콘솔 사용

S3 Access Grants를 사용하여 S3 데이터에 대한 액세스 권한을 부여하려면 먼저 S3 데이터와 동일한 AWS 리전에 S3 Access Grants 인스턴스를 생성해야 합니다.

사전 조건

기업 디렉터리의 ID를 사용하여 S3 데이터에 대한 액세스 권한을 부여하려면 기업 ID 디렉터리를 AWS IAM Identity Center에 추가하세요. 아직 준비가 되지 않은 경우 나중에 S3 Access Grants 인스턴스를 IAM Identity Center 인스턴스와 연결할 수 있습니다.

S3 Access Grants 인스턴스를 생성하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 전환하려는 리전을 선택합니다.
3. 왼쪽 탐색 창에서 Access Grants를 선택합니다.
4. S3 Access Grants 페이지에서 S3 Access Grants 인스턴스 생성을 선택합니다.

    a. Access Grants 인스턴스 설정 마법사의 1단계에서 현재 AWS 리전에 인스턴스를 생성할 것인지 확인합니다. S3 데이터와 동일한 AWS 리전인지 확인하세요. 계정당 AWS 리전별로 S3 Access Grants 인스턴스를 하나 생성할 수 있습니다.

b. (선택 사항) AWS IAM Identity Center에 기업 ID 디렉터리를 추가한 경우 기업 디렉터리의 이 IAM Identity Center 인스턴스를 S3 Access Grants 인스턴스와 연결할 수 있습니다.

이렇게 하려면 **##**에 IAM Identity Center 인스턴스 추가를 선택합니다. 그런 다음 IAM Identity Center 인스턴스 Amazon 리소스 이름(ARN)을 입력합니다.

아직 기업 디렉터리를 IAM Identity Center에 추가하지 않은 경우 나중에 S3 Access Grants 인스턴스를 IAM Identity Center 인스턴스와 연결할 수 있습니다.

c. S3 Access Grants 인스턴스를 생성하려면 다음을 선택합니다. 위치를 등록하려면 2단계 - 위치 등록을 참조하세요.

5. 다음 또는 S3 Access Grants 인스턴스 생성이 비활성화된 경우:

인스턴스를 생성할 수 없음

- 동일한 AWS 리전에 이미 S3 Access Grants 인스턴스가 있을 수 있습니다. 왼쪽 탐색 창에서 Access Grants를 선택합니다. S3 Access Grants 페이지에서 계정의 S3 Access Grants 인스턴스 섹션까지 아래로 스크롤하여 인스턴스가 이미 존재하는지 확인합니다.
- S3 Access Grants 인스턴스를 생성하는 데 필요한 `s3:CreateAccessGrantsInstance` 권한이 없을 수도 있습니다. 계정 관리자에게 문의하세요. IAM Identity Center 인스턴스를 S3 Access Grants 인스턴스와 연결하는 데 필요한 추가 권한은 CreateAccessGrantsInstance 섹션을 참조하세요.

AWS CLI 사용

AWS CLI를 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치를 참조하세요.

다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

Example S3 Access Grants 인스턴스 생성

```
aws s3control create-access-grants-instance \
--account-id 111122223333 \
--region us-east-2
```

응답:

```
{
    "CreatedAt": "2023-05-31T17:54:07.893000+00:00",
    "AccessGrantsInstanceId": "default",
```

```
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default"
}
```

REST API 사용

Amazon S3 REST API를 사용하여 S3 Access Grants 인스턴스를 생성할 수 있습니다. S3 Access Grants 인스턴스 관리를 위한 REST API 지원에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 다음 섹션을 참조하세요.

- AssociateAccessGrantsIdentityCenter
- CreateAccessGrantsInstance
- DeleteAccessGrantsInstance
- DissociateAccessGrantsIdentityCenter
- GetAccessGrantsInstance
- GetAccessGrantsInstanceForPrefix
- GetAccessGrantsInstanceResourcePolicy
- ListAccessGrantsInstances
- PutAccessGrantsInstanceResourcePolicy

AWS SDK 사용

이 섹션에서는 AWS SDK를 사용하여 S3 Access Grants 인스턴스를 생성하는 방법의 예시를 보여줍니다.

Java

이 예시에서는 개별 액세스 권한 부여의 컨테이너 역할을 하는 S3 Access Grants 인스턴스를 생성합니다. 계정에서 AWS 리전당 S3 Access Grants 인스턴스를 하나 보유할 수 있습니다. 응답에는 S3 Access Grants 인스턴스에 대해 생성된 인스턴스 ID `default` 및 Amazon 리소스 이름(ARN)이 포함됩니다.

Example S3 Access Grants 인스턴스 요청 생성

```
public void createAccessGrantsInstance() {
CreateAccessGrantsInstanceRequest createRequest =
 CreateAccessGrantsInstanceRequest.builder().accountId("111122223333").build();
```

```
CreateAccessGrantsInstanceResponse createResponse =
 s3Control.createAccessGrantsInstance(createRequest);LOGGER.info("CreateAccessGrantsInstance
 " + createResponse);
}
```

응답:

```
CreateAccessGrantsInstanceResponse(
CreatedAt=2023-06-07T01:46:20.507Z,
AccessGrantsInstanceId=default,
AccessGrantsInstanceArn=arn:aws:s3:us-east-2:111122223333:access-grants/default)
```

S3 Access Grants 인스턴스의 세부 정보 가져오기

특정 AWS 리전에 있는 Amazon S3 Access Grants 인스턴스의 세부 정보를 가져올 수 있습니다. Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), Amazon S3 REST API, AWS SDK를 사용하여 S3 Access Grants 인스턴스 세부 정보를 가져올 수 있습니다.

S3 콘솔 사용

S3 Access Grants 인스턴스의 세부 정보를 가져오려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Access Grants를 선택합니다.
3. S3 Access Grants 페이지에서 사용하려는 S3 Access Grants 인스턴스가 포함된 리전을 선택합니다.
4. S3 Access Grants 페이지에는 S3 Access Grants 인스턴스와 계정과 공유된 모든 계정 간 인스턴스가 나열됩니다. 인스턴스 세부 정보를 보려면 세부 정보 보기를 선택합니다.

AWS CLI 사용

AWS CLI를 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치를 참조하세요.

다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

Example - S3 Access Grants 인스턴스의 세부 정보 가져오기

```
aws s3control get-access-grants-instance \
```

```
 --account-id 111122223333 \
 --region us-east-2
```

응답:

```
{
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2: 111122223333:access-grants/
default",
    "AccessGrantsInstanceId": "default",
    "CreatedAt": "2023-05-31T17:54:07.893000+00:00"
}
```

REST API 사용

S3 Access Grants 인스턴스 관리를 위한 Amazon S3 REST API 지원에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 다음 섹션을 참조하세요.

- GetAccessGrantsInstance
- GetAccessGrantsInstanceForPrefix

AWS SDK 사용

이 섹션에서는 AWS SDK를 사용하여 S3 Access Grants 인스턴스의 세부 정보를 가져오는 방법의 예시를 보여줍니다.

다음 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

Java

Example - S3 Access Grants 인스턴스 가져오기

```
public void getAccessGrantsInstance() {
GetAccessGrantsInstanceRequest getRequest = GetAccessGrantsInstanceRequest.builder()
.accountId("111122223333")
.build();
GetAccessGrantsInstanceResponse getResponse =
 s3Control.getAccessGrantsInstance(getRequest);
LOGGER.info("GetAccessGrantsInstanceResponse: " + getResponse);
}
```

응답:

```
GetAccessGrantsInstanceResponse(
AccessGrantsInstanceArn=arn:aws:s3:us-east-2: 111122223333:access-grants/default,
CreatedAt=2023-06-07T01:46:20.507Z)
```

S3 Access Grants 인스턴스 나열

AWS Resource Access Manager(AWS RAM)를 통해 공유된 인스턴스를 포함하여 S3 Access Grants 인스턴스를 나열할 수 있습니다.

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), Amazon S3 REST API, AWS SDK를 사용하여 S3 Access Grants 인스턴스를 나열할 수 있습니다.

S3 콘솔 사용

S3 Access Grants 인스턴스를 나열하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Access Grants를 선택합니다.
3. S3 Access Grants 페이지에서 사용하려는 S3 Access Grants 인스턴스가 포함된 리전을 선택합니다.
4. S3 Access Grants 페이지에는 S3 Access Grants 인스턴스와 계정과 공유된 모든 계정 간 인스턴스가 나열됩니다. 인스턴스 세부 정보를 보려면 세부 정보 보기를 선택합니다.

AWS CLI 사용

AWS CLI를 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치를 참조하세요.

다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

Example - 계정의 모든 S3 Access Grants 인스턴스 나열

이 작업은 계정의 S3 Access Grants 인스턴스를 나열합니다. AWS 리전당 하나의 S3 Access Grants 인스턴스만 보유할 수 있습니다. 이 작업은 계정에서 액세스할 수 있는 다른 계정 간 S3 Access Grants 인스턴스도 나열합니다.

```
aws s3control list-access-grants-instances \
 --account-id 111122223333 \
 --region us-east-2
```

응답:

```
{
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2: 111122223333:access-grants/
default",
    "AccessGrantsInstanceId": "default",
    "CreatedAt": "2023-05-31T17:54:07.893000+00:00"
}
```

REST API 사용

S3 Access Grants 인스턴스 관리를 위한 Amazon S3 REST API 지원에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 다음 섹션을 참조하세요.

- ListAccessGrantsInstances

AWS SDK 사용

이 섹션에서는 AWS SDK를 사용하여 S3 Access Grants 인스턴스의 세부 정보를 가져오는 방법의 예시를 보여줍니다.

다음 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

Java

Example - 계정의 모든 S3 Access Grants 인스턴스 나열

이 작업은 계정의 S3 Access Grants 인스턴스를 나열합니다. 리전당 하나의 S3 Access Grants 인스턴스만 보유할 수 있습니다. 이 작업은 계정에서 액세스할 수 있는 다른 계정 간 S3 Access Grants 인스턴스도 나열할 수 있습니다.

```
public void listAccessGrantsInstances() {
ListAccessGrantsInstancesRequest listRequest =
 ListAccessGrantsInstancesRequest.builder()
.accountId("111122223333")
.build();
ListAccessGrantsInstancesResponse listResponse =
 s3Control.listAccessGrantsInstances(listRequest);
LOGGER.info("ListAccessGrantsInstancesResponse: " + listResponse);
}
```

응답:

```
ListAccessGrantsInstancesResponse(
AccessGrantsInstancesList=[
ListAccessGrantsInstanceEntry(
AccessGrantsInstanceId=default,
AccessGrantsInstanceArn=arn:aws:s3:us-east-2:111122223333:access-grants/default,
CreatedAt=2023-06-07T04:28:11.728Z
)
]
)
```

IAM Identity Center 인스턴스 연결 또는 연결 해제

Amazon S3 Access Grants에서는 기업 ID 디렉터리의 AWS IAM Identity Center 인스턴스를 S3 Access Grants 인스턴스와 연결할 수 있습니다. 이렇게 하면 AWS Identity and Access Management(IAM) 사용자 및 역할 외에도 기업 디렉터리 사용자 및 그룹에 대한 액세스 권한 부여를 생성할 수 있습니다.

기업 디렉터리 사용자 및 그룹에 대한 액세스 권한을 더 이상 생성하지 않으려면 S3 Access Grants 인스턴스에서 IAM Identity Center 인스턴스의 연결을 해제하면 됩니다.

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), Amazon S3 REST API, AWS SDK를 사용하여 IAM Identity Center 인스턴스를 연결하거나 연결 해제할 수 있습니다.

S3 콘솔 사용

IAM Identity Center 인스턴스를 S3 Access Grants 인스턴스와 연결하기 전에 기업 ID 디렉터리를 IAM Identity Center에 추가해야 합니다. 자세한 내용은 the section called “S3 Access Grants 및 기업 디렉터리 ID” 섹션을 참조하세요.

IAM Identity Center 인스턴스를 S3 Access Grants 인스턴스와 연결하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Access Grants를 선택합니다.
3. S3 Access Grants 페이지에서 사용하려는 S3 Access Grants 인스턴스가 포함된 리전을 선택합니다.
4. 인스턴스의 세부 정보 보기를 선택합니다.
5. 세부 정보 페이지의 IAM Identity Center 섹션에서 IAM Identity Center 인스턴스 추가 또는 이미 연결된 IAM Identity Center 인스턴스의 등록 취소 중에서 선택합니다.

AWS CLI 사용

AWS CLI를 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치를 참조하세요.

다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

Example - IAM Identity Center 인스턴스를 S3 Access Grants 인스턴스와 연결

```
aws s3control associate-access-grants-identity-center \
 --account-id 111122223333 \
 --identity-center-arn arn:aws:sso:::instance/ssoins-1234a567bb89012c \
 --profile access-grants-profile \
 --region eu-central-1

// No response body
```

Example - S3 Access Grants 인스턴스에서 IAM Identity Center 인스턴스의 연결 해제

```
aws s3control dissociate-access-grants-identity-center \
 --account-id 111122223333 \
 --profile access-grants-profile \
 --region eu-central-1

// No response body
```

REST API 사용

IAM Identity Center 인스턴스와 S3 Access Grants 인스턴스의 연결 관리를 위한 Amazon S3 REST API 지원에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 다음 섹션을 참조하세요.

- AssociateAccessGrantsIdentityCenter
- DissociateAccessGrantsIdentityCenter

S3 Access Grants 인스턴스 삭제

계정의 AWS 리전에서 Amazon S3 Access Grants 인스턴스를 삭제할 수 있습니다. 하지만 S3 Access Grants 인스턴스를 삭제하기 전에 먼저 다음 사항을 수행해야 합니다.

- 모든 권한 부여 및 위치를 포함하여 S3 Access Grants 인스턴스 내의 모든 리소스를 삭제합니다. 자세한 내용은 권한 부여 삭제 및 Delete a location을 참조하세요.

- AWS IAM Identity Center 인스턴스를 S3 Access Grants 인스턴스와 연결한 경우 IAM Identity Center 인스턴스의 연결을 해제해야 합니다. 자세한 내용은 IAM Identity Center 인스턴스 연결 또는 연결 해제를 참조하세요.

> **Important**
>
> S3 Access Grants 인스턴스를 삭제하는 경우 삭제는 영구적이며 취소할 수 없습니다. 이 S3 Access Grants 인스턴스의 권한 부여를 통해 액세스 권한이 부여된 모든 피부여자는 S3 데이터에 대한 액세스 권한을 잃게 됩니다.

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), Amazon S3 REST API, AWS SDK를 사용하여 S3 Access Grants 인스턴스를 삭제할 수 있습니다.

S3 콘솔 사용

S3 Access Grants 인스턴스를 삭제하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Access Grants를 선택합니다.
3. S3 Access Grants 페이지에서 사용하려는 S3 Access Grants 인스턴스가 포함된 리전을 선택합니다.
4. 인스턴스의 세부 정보 보기를 선택합니다.
5. 인스턴스 세부 정보 페이지에서 오른쪽 상단 모서리의 인스턴스 삭제를 선택합니다.
6. 나타나는 대화 상자에서 삭제를 선택합니다. 이 작업은 실행 취소할 수 없습니다.

AWS CLI 사용

AWS CLI를 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치를 참조하세요.

다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

> **Note**
>
> S3 Access Grants 인스턴스를 삭제하려면 먼저 해당 S3 Access Grants 인스턴스 내에 생성된 모든 권한 부여 및 위치를 삭제해야 합니다. IAM Identity Center 인스턴스를 S3 Access Grants 인스턴스와 연결한 경우 먼저 연결을 해제해야 합니다.

Example - S3 Access Grants 인스턴스 삭제

```
aws s3control delete-access-grants-instance \
--account-id 111122223333 \
--profile access-grants-profile \
--region us-east-2 \
--endpoint-url https://s3-control.us-east-2.amazonaws.com \

 // No response body
```

REST API 사용

S3 Access Grants 인스턴스 삭제를 위한 Amazon S3 REST API 지원에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 DeleteAccessGrantsInstance 섹션을 참조하세요.

AWS SDK 사용

이 섹션에서는 AWS SDK를 사용하여 S3 Access Grants 인스턴스를 삭제하는 방법의 예시를 보여줍니다.

다음 예시를 사용하려면 *user input placeholders*를 실제 정보로 대체하십시오.

Java

> **Note**
>
> S3 Access Grants 인스턴스를 삭제하려면 먼저 해당 S3 Access Grants 인스턴스 내에 생성된 모든 권한 부여 및 위치를 삭제해야 합니다. IAM Identity Center 인스턴스를 S3 Access Grants 인스턴스와 연결한 경우 먼저 연결을 해제해야 합니다.

Example - S3 Access Grants 인스턴스 삭제

```
public void deleteAccessGrantsInstance() {
```

```
DeleteAccessGrantsInstanceRequest deleteRequest =
 DeleteAccessGrantsInstanceRequest.builder()
.accountId("111122223333")
.build();
DeleteAccessGrantsInstanceResponse deleteResponse =
 s3Control.deleteAccessGrantsInstance(deleteRequest);
LOGGER.info("DeleteAccessGrantsInstanceResponse: " + deleteResponse);
}
```

S3 Access Grants 위치 사용

계정의 AWS 리전에서 Amazon S3 Access Grants 인스턴스를 생성한 후 해당 인스턴스에 S3 위치를 등록합니다. S3 Access Grans 위치는 기본 S3 위치(`s3://`), 버킷 또는 접두사를 AWS Identity and Access Management(IAM) 역할에 매핑합니다. S3 Access Grants는 이 IAM 역할을 맡아 해당 특정 위치에 액세스하는 피부여자에게 임시 자격 증명을 제공합니다. 액세스 권한 부여를 생성하려면 먼저 S3 Access Grants 인스턴스에 위치를 하나 이상 등록해야 합니다.

위치를 등록하고, 위치의 세부 정보를 보고, 위치를 편집하고, 위치를 삭제할 수 있습니다.

> **Note**
>
> S3 Access Grants 인스턴스에서 첫 번째 위치를 등록한 후에도 인스턴스에는 여전히 개별 액세스 권한이 부여되지 않습니다. 액세스 권한 부여를 생성하려면 권한 부여 생성 섹션을 참조하세요.

주제

- 위치 등록
- 등록된 위치에 대한 세부 정보 보기
- 등록된 위치 업데이트
- 등록된 위치 삭제

위치 등록

계정의 AWS 리전에서 Amazon S3 Access Grants 인스턴스를 생성한 후 해당 인스턴스에 S3 위치를 등록합니다. S3 Access Grans 위치는 기본 S3 위치(`s3://`), 버킷 또는 접두사를 AWS Identity and Access Management(IAM) 역할에 매핑합니다. S3 Access Grants는 이 IAM 역할을 맡아 해당 특정 위

치에 액세스하는 피부여자에게 임시 자격 증명을 제공합니다. 액세스 권한 부여를 생성하려면 먼저 S3 Access Grants 인스턴스에 위치를 하나 이상 등록해야 합니다.

권장 사용 사례

기본 위치(s3://)를 등록하고 이를 IAM 역할에 매핑하는 것이 좋습니다. 기본 S3 경로(s3://)의 위치는 계정의 AWS 리전에 있는 모든 S3 버킷에 대한 액세스를 포함합니다. 액세스 권한 부여를 생성할 때 기본 위치 내의 버킷, 접두사 또는 객체로 권한 부여 범위를 좁힐 수 있습니다.

복잡한 액세스 관리 사용 사례

더 복잡한 액세스 관리 사용 사례에서는 기본 위치보다 더 많이 등록해야 할 수도 있습니다. 이러한 사용 사례의 몇 가지 예는 다음과 같습니다.

- *amzn-s3-demo-bucket*은 S3 Access Grants 인스턴스의 등록된 위치로서 IAM 역할이 매핑되어 있지만 이 IAM 역할은 버킷 내 특정 접두사에 대한 액세스가 거부되었다고 가정해 보겠습니다. 이 경우 IAM 역할이 액세스할 수 없는 접두사를 별도의 위치로 등록하고 필요한 액세스 권한이 있는 다른 IAM 역할에 해당 위치를 매핑할 수 있습니다.
- 가상 프라이빗 클라우드(VPC) 엔드포인트 내 사용자로만 액세스 권한을 제한하는 권한 부여를 생성한다고 가정해 보겠습니다. 이 경우, IAM 역할이 VPC 엔드포인트에 대한 액세스를 제한하는 버킷 위치를 등록할 수 있습니다. 나중에 피부여자가 S3 Access Grants에 자격 증명을 요청하면 S3 Access Grants는 해당 위치의 IAM 역할을 맡아 임시 자격 증명을 제공합니다. 이 자격 증명은 호출자가 VPC 엔드포인트 내에 있지 않는 한 특정 버킷에 대한 액세스를 거부합니다. 이 거부 권한은 권한 부여에 지정된 일반 READ, WRITE 또는 READWRITE 권한에 추가로 적용됩니다.

위치를 등록할 때는 S3 Access Grants가 임시 자격 증명을 제공하고 특정 권한 부여에 대한 권한 범위를 지정하기 위해 S3 Access Grants가 맡는 IAM 역할도 지정해야 합니다.

사용 사례에서 S3 Access Grants 인스턴스에 여러 위치를 등록해야 하는 경우 다음 중 하나를 등록할 수 있습니다.

S3 URI	IAM 역할	설명
s3://	*Default-IAM-role*	기본 위치인 s3://에는 AWS 리전의 모든 버킷이 포함됩니다.
s3://*amzn-s3-demo-bucket1* /	*IAM-role-For-bucket*	이 위치에는 지정된 버킷의 모든 객체가 포함됩니다.

S3 URI	IAM 역할	설명
s3://*amzn-s3-demo-bucket1* /*prefix-name*	*IAM-role-For-prefix*	이 위치에는 이 접두사로 시작하는 객체 키 이름을 가진 버킷 내 모든 객체가 포함됩니다.

특정 버킷 또는 접두사를 등록하기 전에 다음을 수행해야 합니다.

- 액세스 권한을 부여할 데이터가 포함된 버킷을 하나 이상 생성합니다. 이러한 버킷은 S3 Access Grants 인스턴스와 동일한 AWS 리전에 있어야 합니다. 자세한 내용은 버킷 생성 단원을 참조하세요.

  접두사를 추가하는 것은 선택적 단계입니다. 접두사는 객체 키 이름의 시작 부분에 있는 문자열입니다. 이를 사용하여 버킷의 객체를 구성하고 액세스 관리를 수행할 수 있습니다. 버킷에 접두사를 추가하려면 객체 키 이름 생성을 참조하세요.
- AWS 리전 내 S3 데이터에 액세스할 수 있는 권한이 있는 IAM 역할을 생성합니다. 자세한 내용은 AWS IAM Identity Center IAM 사용 설명서의 IAM 역할 생성을 참조하세요.
- IAM 역할 신뢰 정책에서 S3 Access Grants 서비스(`access-grants.s3.amazonaws.com`) 보안 주체에 생성한 IAM 역할에 대한 액세스 권한을 부여합니다. 이를 위해 다음 문이 포함된 JSON 파일을 생성할 수 있습니다. 계정에 신뢰 정책을 추가하려면 사용자 지정 신뢰 정책을 사용하여 역할 생성을 참조하세요.

  TestRolePolicy.json

```
{
  "Version": "2012-10-17",
    "Statement": [
    {
      "Sid": "TestRolePolicy",
      "Effect": "Allow",
      "Principal": {
        "Service":"access-grants.s3.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringEquals": {
```

```
          "aws:SourceAccount":"accountId",
          "aws:SourceArn":"arn:aws:s3:region:accountId:access-grants/default"
        }
      }
    }
  ]
}
```

또는 IAM Identity Center 사용 사례의 경우 두 번째 문이 포함된 다음 정책을 사용합니다.

```
{
  "Version": "2012-10-17",
    "Statement": [
    {
      "Sid": "TestRolePolicyIdentityCenter",
      "Effect": "Allow",
      "Principal": {
        "Service":"access-grants.s3.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount":"accountId",
          "aws:SourceArn":"arn:aws:s3:region:accountId:access-grants/default"
        }
      }
    },
    {
      "Sid": "Stmt1234567891012",
        "Effect": "Allow",
        "Principal": {
          "Service": "access-grants.s3.amazonaws.com"
        },
        "Action": "sts:SetContext",
        "Condition":{
          "StringEquals":{
            "aws:SourceAccount":"accountId",
            "aws:SourceArn":"arn:aws:s3:region:accountId:access-grants/default"
          },
          "ForAllValues:ArnEquals": {
```

```
            "sts:RequestContextProviders":"arn:aws:iam::aws:contextProvider/
IdentityCenter"
          }
        }
      }
  ]
}
```

- Amazon S3 권한을 생성한 IAM 역할에 연결하기 위해 IAM 정책을 생성합니다. 다음 예시 `iam-policy.json` 파일을 확인하고 *`user input placeholders`*를 실제 정보로 대체하세요.

> **Note**
>
> - AWS Key Management Service(AWS KMS) 키를 통한 서버 측 암호화를 사용하여 데이터를 암호화하는 경우 다음 예시는 IAM 역할에 필요한 AWS KMS 권한을 정책에 포함합니다. 이 기능을 사용하지 않는 경우 IAM 정책에서 이러한 권한을 제거할 수 있습니다.
> - S3 Access Grants에서 벤딩하는 자격 증명인 경우에만 IAM 역할이 S3 데이터에 액세스하도록 제한할 수 있습니다. 이 예시는 특정 S3 Access Grants 인스턴스에 대한 `Condition` 문을 추가하는 방법을 보여 줍니다. 이 `Condition`을 사용하려면 `Condition` 문의 S3 Access Grants 인스턴스 ARN을 `arn:aws:s3:`*`region`*`:`*`accountId`*`:access-grants/default` 형식의 S3 Access Grants 인스턴스 ARN으로 바꾸세요.

iam-policy.json

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
         "Sid": "ObjectLevelReadPermissions",
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:GetObjectAcl",
            "s3:GetObjectVersionAcl",
            "s3:ListMultipartUploadParts"
         ],
         "Resource":[
            "arn:aws:s3:::*"
```

```
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:region:accountId:access-
grants/default"]
            }
        }
      },
      {
         "Sid": "ObjectLevelWritePermissions",
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:PutObjectVersionAcl",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion",
            "s3:AbortMultipartUpload"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:AWS ##:accountId:access-
grants/default"]
            }
         }
      },
      {
         "Sid": "BucketLevelReadPermissions",
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
```

```
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:AWS ##:accountId:access-
grants/default"]
            }
         }
      },
   //Optionally add the following section if you use SSE-KMS encryption
      {
         "Sid": "KMSPermissions",
         "Effect":"Allow",
         "Action":[
            "kms:Decrypt",
            "kms:GenerateDataKey"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), Amazon S3 REST API, AWS SDK를 사용하여 S3 Access Grants 인스턴스에 위치를 등록할 수 있습니다.

> **Note**
>
> S3 Access Grants 인스턴스에서 첫 번째 위치를 등록한 후에도 인스턴스에는 여전히 개별 액세스 권한이 부여되지 않습니다. 액세스 권한 부여를 생성하려면 권한 부여 생성 섹션을 참조하세요.

S3 콘솔 사용

S3 Access Grants를 사용하여 S3 데이터에 액세스 권한을 부여하려면 먼저 등록된 위치가 하나 이상 있어야 합니다.

S3 Access Grants 인스턴스에 위치를 등록하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Access Grants를 선택합니다.

3. S3 Access Grants 페이지에서 사용하려는 S3 Access Grants 인스턴스가 포함된 리전을 선택합니다.

   S3 Access Grants 인스턴스를 처음 사용하는 경우 1단계 - S3 Access Grants 인스턴스 생성을 완료하고 Access Grants 인스턴스 설정 마법사의 2단계로 이동했는지 확인하세요. 이미 S3 Access Grants 인스턴스가 있는 경우 세부 정보 보기를 선택한 다음 위치 탭에서 위치 등록을 선택합니다.

   a. 위치 범위에서 S3 찾아보기를 선택하거나 등록하려는 위치의 S3 URI 경로를 입력합니다. S3 URI 형식에 대해서는 위치 형식 테이블을 참조하세요. URI를 입력한 후 보기를 선택하여 위치를 찾아볼 수 있습니다.

   b. IAM 역할의 경우, 다음 중 하나를 선택합니다.

      - 기존 IAM 역할에서 선택

        드롭다운 목록에서 IAM 역할을 선택합니다. 역할을 선택한 후 보기를 선택하여 등록하려는 위치를 관리하는 데 필요한 권한이 이 역할에 있는지 확인하세요. 특히, 이 역할이 S3 Access Grants에 `sts:AssumeRole` 및 `sts:SetSourceIdentity` 권한을 부여하는지 확인하세요.

      - IAM 역할 ARN 입력

        IAM 콘솔로 이동합니다. IAM 역할의 Amazon 리소스 이름(ARN)을 복사하여 이 상자에 붙여넣습니다.

   c. 완료하려면 다음 또는 위치 등록을 선택합니다.

4. 문제 해결:

   위치를 등록할 수 없음

   - 위치가 이미 등록되었을 수 있습니다.

     위치를 등록할 `s3:CreateAccessGrantsLocation` 권한이 없을 수도 있습니다. 계정 관리자에게 문의하세요.

AWS CLI 사용

AWS CLI를 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치를 참조하세요.

S3 Access Grants 인스턴스에 기본 위치인 `s3://` 또는 사용자 지정 위치를 등록할 수 있습니다. 먼저 해당 위치에 대한 보안 주체 액세스 권한이 있는 IAM 역할을 생성한 다음 S3 Access Grants에 이 역할을 수임할 권한을 부여해야 합니다.

다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

Example 리소스 정책 생성

S3 Access Grants가 IAM 역할을 수임하도록 허용하는 정책을 생성합니다. 이를 위해 다음 문이 포함된 JSON 파일을 생성할 수 있습니다. 계정에 리소스 정책을 추가하려면 첫 번째 고객 관리형 정책 만들기 및 연결을 참조하세요.

TestRolePolicy.json

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1234567891011",
      "Action": ["sts:AssumeRole", "sts:SetSourceIdentity"],
      "Effect": "Allow",
      "Principal": {"Service":"access-grants.s3.amazonaws.com"}
    }
  ]
}
```

Example 역할 생성

다음 IAM 명령을 실행하여 역할을 생성합니다.

```
aws iam create-role --role-name accessGrantsTestRole \
 --region us-east-2 \
 --assume-role-policy-document file://TestRolePolicy.json
```

`create-role` 명령을 실행하면 정책이 반환됩니다.

```
{
    "Role": {
        "Path": "/",
        "RoleName": "accessGrantsTestRole",
        "RoleId": "AROASRDGX4WM4GH55GIDA",
        "Arn": "arn:aws:iam::111122223333:role/accessGrantsTestRole",
        "CreateDate": "2023-05-31T18:11:06+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
```

```
                {
                    "Sid": "Stmt1685556427189",
                    "Action": [
                        "sts:AssumeRole",
                        "sts:SetSourceIdentity"
                    ],
                    "Effect": "Allow",
                    "Principal": {
                        "Service":"access-grants.s3.amazonaws.com"
                    }
                }
            ]
        }
    }
}
```

Example

Amazon S3 권한을 IAM 역할에 연결하기 위해 IAM 정책을 생성합니다. 다음 예시 `iam-policy.json` 파일을 확인하고 *`user input placeholders`*를 실제 정보로 대체하세요.

> **Note**
>
> AWS Key Management Service(AWS KMS) 키를 통한 서버 측 암호화를 사용하여 데이터를 암호화하는 경우 다음 예시는 IAM 역할에 필요한 AWS KMS 권한을 정책에 추가합니다. 이 기능을 사용하지 않는 경우 IAM 정책에서 이러한 권한을 제거할 수 있습니다.
> S3 Access Grants에서 자격 증명을 제공하는 경우에만 IAM 역할을 사용하여 S3의 데이터에 액세스할 수 있도록 하기 위해 이 예시에서는 IAM 정책에 S3 Access Grants 인스턴스(`s3:AccessGrantsInstance:` *`InstanceArn`*)를 지정하는 `Condition` 문을 추가하는 방법을 보여줍니다. 다음 예시 정책을 사용할 때 *`user input placeholders`*를 실제 정보로 대체하세요.

iam-policy.json

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
         "Sid": "ObjectLevelReadPermissions",
```

```
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:GetObjectAcl",
            "s3:GetObjectVersionAcl",
            "s3:ListMultipartUploadParts"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:region:accountId:access-
grants/default"]
            }
        }
      },
      {
         "Sid": "ObjectLevelWritePermissions",
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:PutObjectVersionAcl",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion",
            "s3:AbortMultipartUpload"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:AWS ##:accountId:access-
grants/default"]
            }
         }
      },
      {
         "Sid": "BucketLevelReadPermissions",
         "Effect":"Allow",
```

```
         "Action":[
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:AWS ##:accountId:access-
grants/default"]
            }
         }
      },
      {
         "Sid": "KMSPermissions",
         "Effect":"Allow",
         "Action":[
            "kms:Decrypt",
            "kms:GenerateDataKey"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

Example

다음 명령 실행:

```
aws iam put-role-policy \
--role-name accessGrantsTestRole \
--policy-name accessGrantsTestRole \
--policy-document file://iam-policy.json
```

Example 기본 위치 등록

```
aws s3control create-access-grants-location \
 --account-id 111122223333 \
 --location-scope s3:// \
 --iam-role-arn arn:aws:iam::111122223333:role/accessGrantsTestRole
```

응답:

```
{"CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "default",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/default",
    "LocationScope": "s3://"
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
}
```

Example 리전당 사용자 정의 위치

```
aws s3control create-access-grants-location \
 --account-id 111122223333 \
 --location-scope s3://DOC-BUCKET-EXAMPLE/ \
 --iam-role-arn arn:aws:iam::123456789012:role/accessGrantsTestRole
```

응답:

```
{"CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "635f1139-1af2-4e43-8131-a4de006eb456",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2: 111122223333:access-grants/
default/location/635f1139-1af2-4e43-8131-a4de006eb888",
    "LocationScope": "s3://DOC-BUCKET-EXAMPLE/",
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
}
```

REST API 사용

S3 Access Grants 인스턴스 관리를 위한 Amazon S3 REST API 지원에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 다음 섹션을 참조하세요.

- CreateAccessGrantsLocation
- DeleteAccessGrantsLocation
- GetAccessGrantsLocation
- ListAccessGrantsLocations
- UpdateAccessGrantsLocation

AWS SDK 사용

이 섹션에서는 AWS SDK를 사용하여 위치를 등록하는 방법의 예시를 보여줍니다.

다음 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

Java

S3 Access Grants 인스턴스에 기본 위치인 s3:// 또는 사용자 지정 위치를 등록할 수 있습니다. 먼저 해당 위치에 대한 보안 주체 액세스 권한이 있는 IAM 역할을 생성한 다음 S3 Access Grants에 이 역할을 수임할 권한을 부여해야 합니다.

다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

Example 기본 위치 등록

요청:

```
public void createAccessGrantsLocation() {
CreateAccessGrantsLocationRequest createRequest =
 CreateAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.locationScope("s3://")
.iamRoleArn("arn:aws:iam::123456789012:role/accessGrantsTestRole")
.build();
CreateAccessGrantsLocationResponse createResponse =
 s3Control.createAccessGrantsLocation(createRequest);
LOGGER.info("CreateAccessGrantsLocationResponse: " + createResponse);
}
```

응답:

```
CreateAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:11.027Z,
AccessGrantsLocationId=default,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/default,
LocationScope=s3://,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
```

Example 리전당 사용자 정의 위치

요청:

```
public void createAccessGrantsLocation() {
CreateAccessGrantsLocationRequest createRequest =
 CreateAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.locationScope("s3://DOC-BUCKET-EXAMPLE/")
.iamRoleArn("arn:aws:iam::111122223333:role/accessGrantsTestRole")
.build();
CreateAccessGrantsLocationResponse createResponse =
 s3Control.createAccessGrantsLocation(createRequest);
LOGGER.info("CreateAccessGrantsLocationResponse: " + createResponse);
}
```

응답:

```
CreateAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=18cfe6fb-eb5a-4ac5-aba9-8d79f04c2012,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/18cfe6fb-eb5a-4ac5-aba9-8d79f04c2666,
LocationScope= s3://test-bucket-access-grants-user123/,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
```

등록된 위치에 대한 세부 정보 보기

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), Amazon S3 REST API, AWS SDK를 사용하여 S3 Access Grants 인스턴스에 등록된 위치의 세부 정보를 가져올 수 있습니다.

S3 콘솔 사용

S3 Access Grants 인스턴스에 등록된 위치를 보는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Access Grants를 선택합니다.
3. S3 Access Grants 페이지에서 사용하려는 S3 Access Grants 인스턴스가 포함된 리전을 선택합니다.

4. 인스턴스의 세부 정보 보기를 선택합니다.
5. 인스턴스의 세부 정보 페이지에서 위치 탭을 선택합니다.
6. 확인하고자 하는 등록 위치를 찾습니다. 등록된 위치 목록을 필터링하려면 검색 상자를 사용합니다.

AWS CLI 사용

AWS CLI를 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치를 참조하세요.

다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

Example - 등록된 위치에 대한 세부 정보 가져오기

```
aws s3control get-access-grants-location \
--account-id 111122223333 \
--access-grants-location-id default
```

응답:

```
{
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "default",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/default",
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
}
```

Example - S3 Access Grants 인스턴스에 등록된 모든 위치 나열

결과를 S3 접두사 또는 버킷으로 제한하려면 선택적으로 `--location-scope s3://`*bucket-and-or-prefix* 파라미터를 사용할 수 있습니다.

```
aws s3control list-access-grants-locations \
--account-id 111122223333 \
--region us-east-2
```

응답:

```
{"AccessGrantsLocationsList": [
```

```
  {
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "default",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/default",
    "LocationScope": "s3://"
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
     },
  {
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "635f1139-1af2-4e43-8131-a4de006eb456",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/635f1139-1af2-4e43-8131-a4de006eb888",
    "LocationScope": "s3://amzn-s3-demo-bucket/prefixA*",
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
     }
   ]
  }
```

REST API 사용

등록된 위치의 세부 정보를 가져오거나 S3 Access Grants 인스턴스에 등록된 모든 위치를 나열하기 위한 Amazon S3 REST API 지원에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 다음 섹션을 참조하세요.

- GetAccessGrantsLocation
- ListAccessGrantsLocations

AWS SDK 사용

이 섹션에서는 AWS SDK를 사용하여 S3 Access Grants 인스턴스에 등록된 위치에 대한 세부 정보를 가져오거나 S3 Access Grants 인스턴스에 등록된 위치를 모두 나열하는 방법의 예시를 보여줍니다.

다음 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

Java

Example - 등록된 위치에 대한 세부 정보 가져오기

```
public void getAccessGrantsLocation() {
GetAccessGrantsLocationRequest getAccessGrantsLocationRequest =
 GetAccessGrantsLocationRequest.builder()
```

```
.accountId("111122223333")
.accessGrantsLocationId("default")
.build();
GetAccessGrantsLocationResponse getAccessGrantsLocationResponse =
 s3Control.getAccessGrantsLocation(getAccessGrantsLocationRequest);
LOGGER.info("GetAccessGrantsLocationResponse: " + getAccessGrantsLocationResponse);
}
```

응답:

```
GetAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=default,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/default,
LocationScope= s3://,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
```

Example - S3 Access Grants 인스턴스에 등록된 모든 위치 나열

결과를 S3 접두사 또는 버킷으로 제한하려면 선택적으로 `LocationScope` 파라미터에 `s3://bucket-and-or-prefix`와 같은 S3 URI를 전달할 수 있습니다.

```
public void listAccessGrantsLocations() {

ListAccessGrantsLocationsRequest listRequest =
 ListAccessGrantsLocationsRequest.builder()
.accountId("111122223333")
.build();

ListAccessGrantsLocationsResponse listResponse =
 s3Control.listAccessGrantsLocations(listRequest);
LOGGER.info("ListAccessGrantsLocationsResponse: " + listResponse);
}
```

응답:

```
ListAccessGrantsLocationsResponse(
AccessGrantsLocationsList=[
ListAccessGrantsLocationsEntry(
CreatedAt=2023-06-07T04:35:11.027Z,
```

```
AccessGrantsLocationId=default,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/default,
LocationScope=s3://,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
),
ListAccessGrantsLocationsEntry(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=635f1139-1af2-4e43-8131-a4de006eb456,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/635f1139-1af2-4e43-8131-a4de006eb888,
LocationScope=s3://amzn-s3-demo-bucket/prefixA*,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
]
)
```

등록된 위치 업데이트

Amazon S3 Access Grants 인스턴스에 등록된 위치의 AWS Identity and Access Management(IAM) 역할을 업데이트할 수 있습니다. S3 Access Grants에 위치를 등록하는 데 사용하는 새 IAM 역할마다 S3 Access Grants 서비스 보안 주체(`access-grants.s3.amazonaws.com`)에게 이 역할에 대한 액세스 권한을 부여해야 합니다. 이렇게 하려면 처음 위치를 등록할 때 사용한 것과 동일한 신뢰 정책 JSON 파일에 새 IAM 역할 항목을 추가하세요.

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), Amazon S3 REST API, AWS SDK를 사용하여 S3 Access Grants 인스턴스의 위치를 업데이트할 수 있습니다.

S3 콘솔 사용

S3 Access Grants 인스턴스에 등록된 위치의 IAM 역할을 업데이트하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Access Grants를 선택합니다.
3. S3 Access Grants 페이지에서 사용하려는 S3 Access Grants 인스턴스가 포함된 리전을 선택합니다.
4. 인스턴스의 세부 정보 보기를 선택합니다.
5. 인스턴스의 세부 정보 페이지에서 위치 탭을 선택합니다.

6. 업데이트하려는 위치를 찾습니다. 위치 목록을 필터링하려면 검색 상자를 사용합니다.
7. 업데이트할 등록된 위치 옆에 있는 옵션 버튼을 선택합니다.
8. IAM 역할을 업데이트한 다음 변경 사항 저장을 선택합니다.

AWS CLI 사용

AWS CLI를 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치를 참조하세요.

다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

Example - 등록된 위치의 IAM 역할 업데이트

```
aws s3control update-access-grants-location \
--account-id 111122223333 \
--access-grants-location-id 635f1139-1af2-4e43-8131-a4de006eb999 \
--iam-role-arn arn:aws:iam::777788889999:role/accessGrantsTestRole
```

응답:

```
{
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "635f1139-1af2-4e43-8131-a4de006eb999",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:777788889999:access-grants/
default/location/635f1139-1af2-4e43-8131-a4de006eb888",
    "LocationScope": "s3://amzn-s3-demo-bucket/prefixB*",
    "IAMRoleArn": "arn:aws:iam::777788889999:role/accessGrantsTestRole"
}
```

REST API 사용

S3 Access Grants 인스턴스의 위치 업데이트를 위한 Amazon S3 REST API 지원에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 UpdateAccessGrantsLocation 섹션을 참조하세요.

AWS SDK 사용

이 섹션에서는 AWS SDK를 사용하여 등록된 위치의 IAM 역할을 업데이트하는 방법의 예시를 보여줍니다.

다음 예시를 사용하려면 *user input placeholders*를 실제 정보로 대체하십시오.

Java

Example - 등록된 위치의 IAM 역할 업데이트

```
public void updateAccessGrantsLocation() {
UpdateAccessGrantsLocationRequest updateRequest =
 UpdateAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("635f1139-1af2-4e43-8131-a4de006eb999")
.iamRoleArn("arn:aws:iam::777788889999:role/accessGrantsTestRole")
.build();
UpdateAccessGrantsLocationResponse updateResponse =
 s3Control.updateAccessGrantsLocation(updateRequest);
LOGGER.info("UpdateAccessGrantsLocationResponse: " + updateResponse);
}
```

응답:

```
UpdateAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=635f1139-1af2-4e43-8131-a4de006eb999,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:777788889999:access-grants/default/
location/635f1139-1af2-4e43-8131-a4de006eb888,
LocationScope=s3://amzn-s3-demo-bucket/prefixB*,
IAMRoleArn=arn:aws:iam::777788889999:role/accessGrantsTestRole
)
```

등록된 위치 삭제

Amazon S3 Access Grants 인스턴스에서 위치 등록을 삭제할 수 있습니다. 위치를 삭제하면 S3 Access Grants 인스턴스에서 등록이 취소됩니다.

S3 Access Grants 인스턴스에서 위치 등록을 제거하려면 먼저 이 위치와 관련된 모든 권한 부여를 삭제해야 합니다. 권한 부여를 삭제하는 방법에 대한 자세한 정보는 권한 부여 삭제를 참조하세요.

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), Amazon S3 REST API, AWS SDK를 사용하여 S3 Access Grants 인스턴스의 위치를 삭제할 수 있습니다.

S3 콘솔 사용

S3 Access Grants 인스턴스에서 위치 등록을 삭제하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Access Grants를 선택합니다.
3. S3 Access Grants 페이지에서 사용하려는 S3 Access Grants 인스턴스가 포함된 리전을 선택합니다.
4. 인스턴스의 세부 정보 보기를 선택합니다.
5. 인스턴스의 세부 정보 페이지에서 위치 탭을 선택합니다.
6. 업데이트하려는 위치를 찾습니다. 위치 목록을 필터링하려면 검색 상자를 사용합니다.
7. 삭제할 등록된 위치 옆에 있는 옵션 버튼을 선택합니다.
8. 등록 취소(Deregister)를 선택합니다.
9. 이 작업을 취소할 수 없다는 경고 대화 상자가 나타납니다. 위치를 삭제하려면 등록 취소를 선택합니다.

AWS CLI 사용

AWS CLI를 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치를 참조하세요.

다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

Example - 위치 등록 삭제

```
aws s3control delete-access-grants-location \
--account-id 111122223333 \
--access-grants-location-id  a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
 // No response body
```

REST API 사용

S3 Access Grants 인스턴스에서 위치 삭제를 위한 Amazon S3 REST API 지원에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 DeleteAccessGrantsLocation 섹션을 참조하세요.

AWS SDK 사용

이 섹션에서는 AWS SDK를 사용하여 위치를 삭제하는 방법의 예시를 보여줍니다.

다음 예시를 사용하려면 *user input placeholders*를 실제 정보로 대체하십시오.

Java

Example - 위치 등록 삭제

```
public void deleteAccessGrantsLocation() {
DeleteAccessGrantsLocationRequest deleteRequest =
 DeleteAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("a1b2c3d4-5678-90ab-cdef-EXAMPLE11111")
.build();
DeleteAccessGrantsLocationResponse deleteResponse =
 s3Control.deleteAccessGrantsLocation(deleteRequest);
LOGGER.info("DeleteAccessGrantsLocationResponse: " + deleteResponse);
}
```

응답:

```
DeleteAccessGrantsLocationResponse()
```

S3 Access Grants에서 권한 부여 사용

S3 Access Grants 인스턴스 내 개별 액세스 권한 부여를 통해 특정 자격 증명(AWS Identity and Access Management(IAM) 보안 주체나 기업 디렉터리의 사용자 또는 그룹)이 S3 Access Grants 인스턴스에 등록된 위치 내에서 액세스 권한을 부여받을 수 있습니다. 위치는 버킷 또는 접두사를 IAM 역할에 매핑합니다. S3 Access Grants는 이 IAM 역할을 맡아 피부여자에게 임시 자격 증명을 제공합니다.

S3 Access Grants 인스턴스에 위치를 하나 이상 등록한 후 액세스 권한 부여를 생성할 수 있습니다.

피부여자는 IAM 사용자나 역할 또는 디렉터리 사용자나 그룹일 수 있습니다. 디렉터리 사용자는 S3 Access Grants 인스턴스와 연결한 기업 디렉터리 또는 외부 자격 증명 소스의 사용자입니다. 자세한 내용은 S3 Access Grants 및 기업 디렉터리 ID 섹션을 참조하세요. IAM Identity Center에서 특정 디렉터리 사용자 또는 그룹에 대한 권한 부여를 생성하려면 IAM Identity Center에서 해당 사용자를 식별하는 데 사용하는 GUID를 찾으세요(예: a1b2c3d4-5678-90ab-cdef-EXAMPLE11111). IAM Identity Center를 사용하여 사용자 정보를 보는 방법에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 사용자 및 그룹 할당 보기를 참조하세요.

버킷, 접두사 또는 객체에 액세스 권한을 부여할 수 있습니다. Amazon S3의 접두사는 객체 키 이름의 앞에 있는 문자열로, 버킷 내의 객체를 구성하는 데 사용됩니다. 이는 허용되는 모든 문자열(예: `engineering/` 접두사로 시작하는 버킷의 객체 키 이름)이 될 수 있습니다.

주제

- 권한 부여 생성
- 권한 부여 보기
- 권한 부여 삭제

권한 부여 생성

S3 Access Grants 인스턴스 내 개별 액세스 권한 부여를 통해 특정 자격 증명(AWS Identity and Access Management(IAM) 보안 주체나 기업 디렉터리의 사용자 또는 그룹)이 S3 Access Grants 인스턴스에 등록된 위치 내에서 액세스 권한을 부여받을 수 있습니다. 위치는 버킷 또는 접두사를 IAM 역할에 매핑합니다. S3 Access Grants는 이 IAM 역할을 맡아 피부여자에게 임시 자격 증명을 제공합니다.

S3 Access Grants 인스턴스에 위치를 하나 이상 등록한 후 액세스 권한 부여를 생성할 수 있습니다.

피부여자는 IAM 사용자나 역할 또는 디렉터리 사용자나 그룹일 수 있습니다. 디렉터리 사용자는 S3 Access Grants 인스턴스와 연결한 기업 디렉터리 또는 외부 자격 증명 소스의 사용자입니다. 자세한 내용은 S3 Access Grants 및 기업 디렉터리 ID 섹션을 참조하세요. IAM Identity Center에서 특정 디렉터리 사용자 또는 그룹에 대한 권한 부여를 생성하려면 IAM Identity Center에서 해당 사용자를 식별하는 데 사용하는 GUID를 찾으세요(예: `a1b2c3d4-5678-90ab-cdef-EXAMPLE11111`). IAM Identity Center를 사용하여 사용자 정보를 보는 방법에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 사용자 및 그룹 할당 보기를 참조하세요.

버킷, 접두사 또는 객체에 액세스 권한을 부여할 수 있습니다. Amazon S3의 접두사는 객체 키 이름의 앞에 있는 문자열로, 버킷 내의 객체를 구성하는 데 사용됩니다. 이는 허용되는 모든 문자열(예: `engineering/` 접두사로 시작하는 버킷의 객체 키 이름)이 될 수 있습니다.

하위 접두사

등록된 위치에 대한 액세스 권한을 부여할 때 `Subprefix` 필드를 사용하여 액세스 범위를 위치 범위의 하위 집합으로 좁힐 수 있습니다. 권한 부여를 위해 선택한 등록 위치가 기본 S3 경로(`s3://`)인 경우 권한 부여 범위를 좁혀야 합니다. 기본 위치(`s3://`)에 대해서는 액세스 권한 부여를 생성할 수 없습니다. 생성할 경우 피부여자에게 AWS 리전 내 모든 버킷에 대한 액세스 권한을 부여하게 됩니다. 대신 다음 중 하나를 사용하여 권한 부여 범위를 좁혀야 합니다.

- 버킷: `s3://bucket/*`
- 버킷 내의 접두사: `s3://bucket/prefix*`
- 접두사 내의 접두사: `s3://bucket/prefixA/prefixB*`
- 객체: `s3://bucket/object-key-name`

등록된 위치가 버킷인 곳에 액세스 권한 부여를 생성하는 경우 `Subprefix` 필드에 다음 중 하나를 전달하여 권한 부여 범위를 좁힐 수 있습니다.

- 버킷 내의 접두사: *prefix**
- 접두사 내의 접두사: *prefixA/prefixB**
- 객체: */object-key-name*

권한 부여를 생성한 후 Amazon S3 콘솔에 표시된 권한 범위나 API 또는 AWS Command Line Interface(AWS CLI) 응답에 반환되는 `GrantScope`은 위치 경로를 `Subprefix`와 결합한 결과입니다. 이 연결 경로가 액세스 권한을 부여하려는 S3 버킷, 접두사 또는 객체에 올바르게 매핑되는지 확인하세요.

> **Note**
>
> - 하나의 객체에만 액세스를 허용하는 액세스 권한 부여를 생성해야 하는 경우 객체에 대한 권한 부여 유형을 지정해야 합니다. API 직접 호출 또는 CLI 명령으로 이 작업을 수행하려면 `s3PrefixType` 파라미터를 `Object` 값과 함께 전달합니다. Amazon S3 콘솔에서 권한 부여를 생성할 때 위치를 선택한 후 권한 부여 범위에서 권한 부여 범위는 객체입니다 확인란을 선택합니다.
> - 버킷이 아직 존재하지 않는 경우 버킷에 대한 권한 부여를 생성할 수 없습니다. 하지만 아직 존재하지 않는 접두사에 대한 권한 부여를 생성할 수는 있습니다.
> - S3 Access Grants 인스턴스에서 생성할 수 있는 최대 권한 부여 수는 S3 Access Grants 한도 섹션을 참조하세요.

Amazon S3 콘솔, AWS CLI, Amazon S3 REST API, AWS SDK를 사용하여 액세스 권한 부여를 생성할 수 있습니다.

S3 콘솔 사용

액세스 권한 부여 생성

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Access Grants를 선택합니다.
3. S3 Access Grants 페이지에서 사용하려는 S3 Access Grants 인스턴스가 포함된 리전을 선택합니다.

   S3 Access Grants 인스턴스를 처음 사용하는 경우 2단계 - 위치 등록을 완료하고 Access Grants 인스턴스 설정 마법사의 3단계로 이동했는지 확인하세요. 이미 S3 Access Grants 인스턴스가 있는 경우 세부 정보 보기를 선택한 다음 권한 부여 탭에서 권한 부여 생성을 선택합니다.

   a. 권한 부여 범위 섹션에서 등록된 위치를 선택하거나 입력합니다.

      기본 `s3://` 위치를 선택한 경우 하위 접두사 상자를 사용하여 액세스 권한 부여 범위를 좁힐 수 있습니다. 자세한 내용은 하위 접두사를 참조하세요. 객체에만 액세스 권한을 부여하는 경우 권한 부여 범위가 객체임을 선택합니다.

   b. 권한 및 액세스에서 권한 수준(읽기, 쓰기 또는 둘 다)을 선택합니다.

      그런 다음 피부여자 유형을 선택합니다. 기업 디렉터리를 IAM Identity Center에 추가하고 이 IAM Identity Center 인스턴스를 S3 Access Grants 인스턴스와 연결한 경우, IAM Identity Center의 디렉터리 ID를 선택할 수 있습니다. 이 옵션을 선택하는 경우 IAM Identity Center에서 사용자 또는 그룹의 ID를 가져와 이 섹션에 입력하세요.

      피부여자 유형이 IAM 사용자 또는 역할인 경우 IAM 보안 주체를 선택합니다. IAM 보안 주체 유형에서 사용자 또는 역할을 선택합니다. 그런 다음, IAM 보안 주체 사용자 아래에서 자격 증명의 ID를 목록에서 선택하거나 입력합니다.

   c. S3 Access Grants 권한 부여를 생성하려면 다음 또는 권한 부여 생성을 선택합니다.
4. 다음 또는 권한 부여 생성이 비활성화된 경우:

   권한 부여를 생성할 수 없음

   - S3 Access Grants 인스턴스에 먼저 위치를 등록해야 할 수 있습니다.
   - 액세스 권한 부여를 생성하는 데 필요한 `s3:CreateAccessGrant` 권한이 없을 수도 있습니다. 계정 관리자에게 문의하세요.

AWS CLI 사용

AWS CLI를 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치를 참조하세요.

다음 예는 IAM 보안 주체에 대한 액세스 권한 부여 요청을 생성하는 방법과 기업 디렉터리 사용자 또는 그룹에 대한 액세스 권한 부여 요청을 생성하는 방법을 보여줍니다.

다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

> **Note**
>
> 하나의 객체에만 액세스 권한을 부여하는 액세스 권한 부여를 생성하는 경우 필수 파라미터인 `--s3-prefix-type Object`를 포함하세요.

Example IAM 보안 주체에 대한 액세스 권한 부여 요청 생성

```
aws s3control create-access-grant \
--account-id 111122223333 \
--access-grants-location-id a1b2c3d4-5678-90ab-cdef-EXAMPLE22222 \
--access-grants-location-configuration S3SubPrefix=prefixB* \
--permission READ \
--grantee GranteeType=IAM,GranteeIdentifier=arn:aws:iam::123456789012:user/data-consumer-3
```

Example 액세스 권한 부여 응답 생성

```
{"CreatedAt": "2023-05-31T18:41:34.663000+00:00",
    "AccessGrantId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "Grantee": {
        "GranteeType": "IAM",
        "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-3"
    },
    "AccessGrantsLocationId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "AccessGrantsLocationConfiguration": {
        "S3SubPrefix": "prefixB*"
    },
    "GrantScope": "s3://DOC-BUCKET-EXAMPLE/prefix*",
    "Permission": "READ"
}
```

디렉터리 사용자 또는 그룹에 대한 액세스 권한 부여 요청 생성

디렉터리 사용자 또는 그룹에 대한 액세스 권한 부여 요청을 생성하려면 먼저 다음 명령 중 하나를 실행하여 디렉터리 사용자 또는 그룹의 GUID를 가져와야 합니다.

Example 디렉터리 사용자 또는 그룹에 대한 GUID 가져오기

IAM Identity Center 사용자의 GUID는 IAM Identity Center 콘솔이나 AWS CLI 또는 AWS SDK를 사용하여 찾을 수 있습니다. 다음 명령은 지정된 IAM Identity Center 인스턴스의 사용자를 이름 및 식별자와 함께 나열합니다.

```
aws identitystore list-users --identity-store-id d-1a2b3c4d1234
```

이 명령은 지정된 IAM Identity Center 인스턴스의 그룹을 나열합니다.

```
aws identitystore list-groups --identity-store-id d-1a2b3c4d1234
```

Example 디렉터리 사용자 또는 그룹에 대한 액세스 권한 부여 생성

이 명령은 IAM 사용자 또는 역할에 대한 권한 부여를 생성하는 것과 비슷합니다. 단, 피부여자 유형은 `DIRECTORY_USER` 또는 `DIRECTORY_GROUP`이고 피부여자 식별자는 디렉터리 사용자 또는 그룹의 GUID입니다.

```
aws s3control create-access-grant \
--account-id 123456789012 \
--access-grants-location-id default \
--access-grants-location-configuration S3SubPrefix="amzn-s3-demo-bucket/rafael/*" \
--permission READWRITE \
--grantee GranteeType=DIRECTORY_USER,GranteeIdentifier=83d43802-00b1-7054-db02-
f1d683aacba5 \
```

REST API 사용

액세스 권한 부여 관리를 위한 Amazon S3 REST API 지원에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 다음 섹션을 참조하세요.

- CreateAccessGrant
- DeleteAccessGrant
- GetAccessGrant
- ListAccessGrants

AWS SDK 사용

이 섹션에서는 AWS SDK를 사용하여 액세스 권한 부여를 생성하는 방법의 예시를 보여줍니다.

Java

다음 예시를 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

> **Note**
>
> 하나의 객체에만 액세스 권한을 부여하는 액세스 권한 부여를 생성하는 경우 필수 파라미터인 `.s3PrefixType(S3PrefixType.Object)`를 포함하세요.

Example 액세스 권한 부여 요청 생성

```
public void createAccessGrant() {
CreateAccessGrantRequest createRequest = CreateAccessGrantRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa")
.permission("READ")
.accessGrantsLocationConfiguration(AccessGrantsLocationConfiguration.builder().s3SubPrefix("
.grantee(Grantee.builder().granteeType("IAM").granteeIdentifier("arn:aws:iam::111122223333:u
data-consumer-3").build())
.build();
CreateAccessGrantResponse createResponse =
 s3Control.createAccessGrant(createRequest);
LOGGER.info("CreateAccessGrantResponse: " + createResponse);
}
```

Example 액세스 권한 부여 응답 생성

```
CreateAccessGrantResponse(
CreatedAt=2023-06-07T05:20:26.330Z,
AccessGrantId=a1b2c3d4-5678-90ab-cdef-EXAMPLE33333,
AccessGrantArn=arn:aws:s3:us-east-2:444455556666:access-grants/default/grant/
a1b2c3d4-5678-90ab-cdef-EXAMPLE33333,
Grantee=Grantee(
GranteeType=IAM,
GranteeIdentifier=arn:aws:iam::111122223333:user/data-consumer-3
),
AccessGrantsLocationId=a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa,
```

```
AccessGrantsLocationConfiguration=AccessGrantsLocationConfiguration(
S3SubPrefix=prefixB*
),
GrantScope=s3://DOC-BUCKET-EXAMPLE/prefixB,
Permission=READ
)
```

권한 부여 보기

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), Amazon S3 REST API, AWS SDK를 사용하여 S3 Access Grants 인스턴스에 있는 액세스 권한 부여의 세부 정보를 볼 수 있습니다.

S3 콘솔 사용

액세스 권한 부여의 세부 정보를 보는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Access Grants를 선택합니다.
3. S3 Access Grants 페이지에서 사용하려는 S3 Access Grants 인스턴스가 포함된 리전을 선택합니다.
4. 인스턴스의 세부 정보 보기를 선택합니다.
5. 세부 정보 페이지에서 권한 부여 탭을 선택합니다.
6. 권한 부여 섹션에서 보려는 액세스 권한 부여를 찾습니다. 권한 부여 목록을 필터링하려면 검색 상자를 사용하세요.

AWS CLI 사용

AWS CLI를 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치를 참조하세요.

다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

Example - 액세스 권한 부여의 세부 정보 가져오기

```
aws s3control get-access-grant \
--account-id 111122223333 \
--access-grant-id a1b2c3d4-5678-90ab-cdef-EXAMPLE22222
```

응답:

```
{
    "CreatedAt": "2023-05-31T18:41:34.663000+00:00",
    "AccessGrantId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "Grantee": {
        "GranteeType": "IAM",
        "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-3"
    },
    "Permission": "READ",
    "AccessGrantsLocationId": "12a6710f-5af8-41f5-b035-0bc795bf1a2b",
    "AccessGrantsLocationConfiguration": {
        "S3SubPrefix": "prefixB*"
    },
    "GrantScope": "s3://amzn-s3-demo-bucket/"
}
```

Example - S3 Access Grants 인스턴스의 모든 액세스 권한 부여 나열

선택적으로 다음 파라미터를 사용하여 결과를 S3 접두사 또는 AWS Identity and Access Management(IAM) ID로 제한할 수 있습니다.

- 하위 접두사 - `--grant-scope s3://bucket-name/prefix*`
- IAM ID - `--grantee-type IAM` 및 `--grantee-identifier arn:aws:iam::123456789000:role/accessGrantsConsumerRole`

```
aws s3control list-access-grants \
--account-id 111122223333
```

응답:

```
{
    "AccessGrantsList": [{"CreatedAt": "2023-06-14T17:54:46.542000+00:00",
            "AccessGrantId": "dd8dd089-b224-4d82-95f6-975b4185bbaa",
            "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/dd8dd089-b224-4d82-95f6-975b4185bbaa",
            "Grantee": {
                "GranteeType": "IAM",
                "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-3"
            },
            "Permission": "READ",
```

```
            "AccessGrantsLocationId": "23514a34-ea2e-4ddf-b425-d0d4bfcarda1",
            "GrantScope": "s3://amzn-s3-demo-bucket/prefixA*"
        },
        {"CreatedAt": "2023-06-24T17:54:46.542000+00:00",
            "AccessGrantId": "ee8ee089-b224-4d72-85f6-975b4185a1b2",
            "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/ee8ee089-b224-4d72-85f6-975b4185a1b2",
            "Grantee": {
                "GranteeType": "IAM",
                "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-9"
            },
            "Permission": "READ",
            "AccessGrantsLocationId": "12414a34-ea2e-4ddf-b425-d0d4bfcacao0",
            "GrantScope": "s3://amzn-s3-demo-bucket/prefixB*"
        },

    ]
}
```

REST API 사용

Amazon S3 API 작업을 사용하여 액세스 권한 부여의 세부 정보를 보고 S3 Access Grants 인스턴스의 모든 액세스 권한 부여를 나열할 수 있습니다. 액세스 권한 부여 관리를 위한 REST API 지원에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 다음 섹션을 참조하세요.

- GetAccessGrant
- ListAccessGrants

AWS SDK 사용

이 섹션에서는 AWS SDK를 사용하여 액세스 권한 부여의 세부 정보를 가져오는 방법의 예시를 보여줍니다.

다음 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

Java

Example - 액세스 권한 부여의 세부 정보 가져오기

```
public void getAccessGrant() {
GetAccessGrantRequest getRequest = GetAccessGrantRequest.builder()
.accountId("111122223333")
```

```
.accessGrantId("a1b2c3d4-5678-90ab-cdef-EXAMPLE22222")
.build();
GetAccessGrantResponse getResponse = s3Control.getAccessGrant(getRequest);
LOGGER.info("GetAccessGrantResponse: " + getResponse);
}
```

응답:

```
GetAccessGrantResponse(
CreatedAt=2023-06-07T05:20:26.330Z,
AccessGrantId=a1b2c3d4-5678-90ab-cdef-EXAMPLE22222,
AccessGrantArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant-fd3a5086-42f7-4b34-9fad-472e2942c70e,
Grantee=Grantee(
GranteeType=IAM,
GranteeIdentifier=arn:aws:iam::111122223333:user/data-consumer-3
),
Permission=READ,
AccessGrantsLocationId=12a6710f-5af8-41f5-b035-0bc795bf1a2b,
AccessGrantsLocationConfiguration=AccessGrantsLocationConfiguration(
S3SubPrefix=prefixB*
),
GrantScope=s3://amzn-s3-demo-bucket/
)
```

Example - S3 Access Grants 인스턴스의 모든 액세스 권한 부여 나열

선택적으로 다음 파라미터를 사용하여 결과를 S3 접두사 또는 IAM ID로 제한할 수 있습니다.

- 범위 - `GrantScope=s3://bucket-name/prefix*`
- 피부여자 - `GranteeType=IAM` 및 `GranteeIdentifier=arn:aws:iam::111122223333:role/accessGrantsConsumerRole`

```
public void listAccessGrants() {
ListAccessGrantsRequest listRequest = ListAccessGrantsRequest.builder()
.accountId("111122223333")
.build();
ListAccessGrantsResponse listResponse = s3Control.listAccessGrants(listRequest);
LOGGER.info("ListAccessGrantsResponse: " + listResponse);
}
```

응답:

```
ListAccessGrantsResponse(
AccessGrantsList=[
ListAccessGrantEntry(
CreatedAt=2023-06-14T17:54:46.540z,
AccessGrantId=dd8dd089-b224-4d82-95f6-975b4185bbaa,
AccessGrantArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/dd8dd089-b224-4d82-95f6-975b4185bbaa,
Grantee=Grantee(
GranteeType=IAM, GranteeIdentifier= arn:aws:iam::111122223333:user/data-consumer-3
),
Permission=READ,
AccessGrantsLocationId=23514a34-ea2e-4ddf-b425-d0d4bfcarda1,
GrantScope=s3://amzn-s3-demo-bucket/prefixA
),
ListAccessGrantEntry(
CreatedAt=2023-06-24T17:54:46.540Z,
AccessGrantId=ee8ee089-b224-4d72-85f6-975b4185a1b2,
AccessGrantArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/ee8ee089-b224-4d72-85f6-975b4185a1b2,
Grantee=Grantee(
GranteeType=IAM, GranteeIdentifier= arn:aws:iam::111122223333:user/data-consumer-9
),
Permission=READ,
AccessGrantsLocationId=12414a34-ea2e-4ddf-b425-d0d4bfcacao0,
GrantScope=s3://amzn-s3-demo-bucket/prefixB*
)
]
)
```

권한 부여 삭제

Amazon S3 Access Grants 인스턴스에서 액세스 권한 부여를 삭제할 수 있습니다. 액세스 권한 부여 삭제는 실행 취소할 수 없습니다. 액세스 권한 부여를 삭제하면 피부여자는 더 이상 Amazon S3 데이터에 액세스할 수 없습니다.

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), Amazon S3 REST API, AWS SDK를 사용하여 액세스 권한 부여를 삭제할 수 있습니다.

S3 콘솔 사용

액세스 권한 부여를 삭제하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Access Grants를 선택합니다.
3. S3 Access Grants 페이지에서 사용하려는 S3 Access Grants 인스턴스가 포함된 리전을 선택합니다.
4. 인스턴스의 세부 정보 보기를 선택합니다.
5. 세부 정보 페이지에서 권한 부여 탭을 선택합니다.
6. 삭제할 권한 부여를 검색합니다. 권한 부여를 찾으면 옆에 있는 라디오 버튼을 선택합니다.
7. 삭제를 선택합니다. 이 작업을 취소할 수 없다는 경고 대화 상자가 나타납니다. 삭제를 다시 선택하여 권한 부여를 삭제합니다.

AWS CLI 사용

AWS CLI를 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치를 참조하세요.

다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

Example - 액세스 권한 부여 삭제

```
aws s3control delete-access-grant \
--account-id 111122223333 \
--access-grant-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111

// No response body
```

REST API 사용

액세스 권한 부여 관리를 위한 Amazon S3 REST API 지원에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 DeleteAccessGrant 섹션을 참조하세요.

AWS SDK 사용

이 섹션에서는 AWS SDK를 사용하여 액세스 권한 부여를 삭제하는 방법의 예시를 보여줍니다. 다음 예시를 사용하려면 *user input placeholders*를 실제 정보로 대체하십시오.

Java

Example - 액세스 권한 부여 삭제

```
public void deleteAccessGrant() {
DeleteAccessGrantRequest deleteRequest = DeleteAccessGrantRequest.builder()
.accountId("111122223333")
.accessGrantId("a1b2c3d4-5678-90ab-cdef-EXAMPLE11111")
.build();
DeleteAccessGrantResponse deleteResponse =
 s3Control.deleteAccessGrant(deleteRequest);
LOGGER.info("DeleteAccessGrantResponse: " + deleteResponse);
}
```

응답:

```
DeleteAccessGrantResponse()
```

액세스 권한 부여를 사용하여 S3 데이터 가져오기

S3 Access Grants를 통해 S3 데이터에 대한 액세스 권한을 받은 피부여자는 S3 데이터에 액세스하는 데 사용하는 S3 Access Grants에서 임시 자격 증명을 요청해야 합니다. 자세한 내용은 S3 Access Grants를 통해 Amazon S3 데이터에 대한 액세스 요청 섹션을 참조하세요. 그런 다음 피부여자는 임시 자격 증명을 사용하여 S3 데이터에 대해 허용 가능한 S3 작업을 수행합니다. 자세한 내용은 S3 Access Grants에서 제공하는 자격 증명을 사용하여 S3 데이터에 액세스 섹션을 참조하세요. 피부여자는 필요에 따라 자격 증명을 요청하기 전에 AWS 계정에 대한 액세스 권한 부여 목록을 요청할 수 있습니다. 자세한 내용은 호출자의 액세스 권한 부여 목록 섹션을 참조하세요.

주제

- S3 Access Grants를 통해 Amazon S3 데이터에 대한 액세스 요청
- S3 Access Grants에서 제공하는 자격 증명을 사용하여 S3 데이터에 액세스
- 호출자의 액세스 권한 부여 목록

S3 Access Grants를 통해 Amazon S3 데이터에 대한 액세스 요청

S3 Access Grants를 사용하여 액세스 권한을 생성한 후 피부여자는 액세스 권한이 부여된 S3 데이터에 액세스하기 위해 자격 증명을 요청할 수 있습니다. 피부여자는 AWS Identity and Access Management(IAM) 위탁자, 회사 디렉터리 자격 증명 또는 승인된 애플리케이션일 수 있습니다.

애플리케이션 또는 AWS 서비스가 S3 Access Grants `GetDataAccess` API 작업을 사용하여 피부여자를 대신하여 S3 데이터에 대한 액세스를 S3 Access Grants에 요청하면, `GetDataAccess`는 먼저 이 자격 증명에 해당 데이터에 대한 액세스 권한을 부여했는지 확인합니다. 그러면 S3 Access Grants가 AssumeRole API 작업을 사용하여 임시 자격 증명 토큰을 가져와 요청자에게 벤딩합니다. 이 임시 자격 증명 토큰은 AWS Security Token Service(AWS STS) 토큰입니다.

`GetDataAccess` 요청에는 임시 자격 증명이 적용되는 S3 데이터의 범위를 지정하는 `target` 파라미터가 포함되어야 합니다. 이 `target` 범위는 권한 부여 범위 또는 해당 범위의 하위 집합과 같을 수 있지만, `target` 범위는 피부여자에게 제공된 권한 부여 범위 내에 있어야 합니다. 또한 요청은 임시 자격 증명의 권한 수준(`READ`, `WRITE` 또는 `READWRITE`)을 나타내는 `permission` 파라미터를 지정해야 합니다.

권한

요청자는 자격 증명 요청에서 임시 토큰의 권한 수준을 지정할 수 있습니다. 요청자는 `privilege` 파라미터를 사용하여 권한 부여 범위 내에서 임시 자격 증명의 액세스 범위를 줄이거나 늘릴 수 있습니다. `privilege` 파라미터의 기본값은 `Default`이며, 이는 반환되는 자격 증명의 대상 범위가 원래 권한 부여 범위임을 의미합니다. `Minimal` 또한 `privilege`의 값이 될 수 있습니다. `target` 범위가 원래 권한 부여 범위보다 축소되면 `target` 범위가 권한 부여 범위 내에 있는 한 `target` 범위와 일치하도록 임시 자격 증명의 범위가 축소됩니다.

다음 테이블에는 두 권한 부여에 대한 `privilege` 파라미터의 효과가 자세히 나와 있습니다. 권한 부여의 범위는 `S3://amzn-s3-demo-bucket1/bob/*`이며, 여기에는 *amzn-s3-demo-bucket1* 버킷의 전체 `bob/` 접두사가 포함됩니다. 다른 권한 부여의 범위는 `S3://amzn-s3-demo-bucket1/bob/reports/*`이며, 여기에는 *amzn-s3-demo-bucket1* 버킷의 `bob/reports/` 접두사만 포함됩니다.

권한 부여 범위	요청된 범위	권한	반환된 범위	Effect
S3://*amzn-s3-demo-bucket1* /bob/*	*amzn-s3-demo-bucket1* /bob/*	Default	*amzn-s3-demo-bucket1* /bob/*	요청자는 *amzn-s3-demo-bucket1* 버킷에서 키 이름이 접두사 *bob/*으로 시작하는 모든 객체에 액세스할 수 있습니다.
S3://*amzn-s3-d*	*amzn-s3-demo-*	Minimal	*amzn-s3-demo-bucket1* /bob/	접두사 이름 bob/ 뒤에 와일드카드를 나타내는

권한 부여 범위	요청된 범위	권한	반환된 범위	Effect
emo-bucket1 /bob/*	*bucket1* /bob/			* 문자가 없으면 요청자는 *amzn-s3-demo-bucket1* 버킷에서 이름이 bob/인 객체에만 액세스할 수 있습니다. 이러한 객체가 있는 경우는 흔하지 않습니다. 요청자는 키 이름이 접두사 bob/으로 시작하는 객체를 비롯해 다른 객체에 액세스할 수 없습니다.
S3://*amzn-s3-demo-bucket1* /bob/*	*amzn-s3-demo-bucket1* /bob/images/*	Minimal	*amzn-s3-demo-bucket1* /bob/images/*	요청자는 *amzn-s3-demo-bucket1* 버킷에서 키 이름이 접두사 *bob/images/** 로 시작하는 모든 객체에 액세스할 수 있습니다.
S3://*amzn-s3-demo-bucket1* /bob/reports/*	*amzn-s3-demo-bucket1* /bob/reports/file.txt	Default	*amzn-s3-demo-bucket1* /bob/reports/*	요청자는 *amzn-s3-demo-bucket1* 버킷에서 키 이름이 접두사 bob/reports 로 시작하는 모든 객체에 액세스할 수 있습니다. 이는 일치하는 권한 부여의 범위에 해당합니다.

권한 부여 범위	요청된 범위	권한	반환된 범위	Effect
S3://*amzn-s3-demo-bucket1* /bob/reports/*	*amzn-s3-demo-bucket1* /bob/reports/file.txt	Minimal	*amzn-s3-demo-bucket1* /bob/reports/file.txt	요청자는 *amzn-s3-demo-bucket1* 버킷에서 키 이름이 bob/reports/file.txt 인 객체에만 액세스할 수 있습니다. 다른 객체에는 액세스할 수 없습니다.

디렉터리 자격 증명

GetDataAccess는 적절한 권한 부여와 일치할 때 요청에 관련된 모든 자격 증명을 고려합니다. 회사 디렉터리 자격 증명의 경우 GetDataAccess는 자격 증명 인식 세션에 사용되는 IAM 자격 증명의 권한 부여도 반환합니다. 자격 증명 인식 세션에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 자격 증명 인식 콘솔 세션을 사용할 수 있는 권한 부여를 참조하세요. GetDataAccess는 다음 표와 같이 가장 제한적인 권한 부여로 범위를 제한하는 자격 증명을 생성합니다.

IAM 자격 증명에 대한 권한 부여 범위	디렉터리 자격 증명에 대한 권한 부여 범위	요청된 범위	반환된 범위	권한	Effect
S3://*amzn-s3-demo-bucket1* /bob/*	*amzn-s3-demo-bucket1* /bob/images/*	S3://*amzn-s3-demo-bucket1* /bob/images/	S3://*amzn-s3-demo-bucket1* /bob/images/*	Default	요청자는 IAM 역할에 대한 권한 부여의 일부로 접두사 bob/로 시작하지만 디렉터리 자격 증명에 대한 권한 부여의 일부로 접두사 bob/images/로 제한된 키 이름을 가진 모든 객체에 액세스할 수 있습니

IAM 자격 증명에 대한 권한 부여 범위	디렉터리 자격 증명에 대한 권한 부여 범위	요청된 범위	반환된 범위	권한	Effect
		image1.jpeg			다. IAM 역할과 디렉터리 자격 증명 모두 요청된 범위인 bob/images/image1.jpeg 에 대한 액세스를 제공하지만 디렉터리 자격 증명에는 더 제한적인 권한이 있습니다. 따라서 반환된 범위는 디렉터리 자격 증명에 대한 보다 제한적인 권한 부여로 제한됩니다.
S3://*amzn-s3-demo-bucket1* / bob/*	*amzn-s3-demo-bucket1* / bob/images/*	S3://*amzn-s3-demo-bucket1* / bob/images/image1.jpeg	S3://*amzn-s3-demo-bucket1* / bob/images/image1.jpeg	Minimal	권한은 Minimal로 설정되므로 자격 증명이 더 큰 범위에 액세스할 수 있더라도 요청된 범위만 bob/images/image1.jpeg 반환됩니다.

IAM 자격 증명에 대한 권한 부여 범위	디렉터리 자격 증명에 대한 권한 부여 범위	요청된 범위	반환된 범위	권한	Effect
S3://*amzn-s3-demo-bucket1* / bob/images/*	*amzn-s3-demo-bucket1* / bob/*	S3://*amzn-s3-demo-bucket1* / bob/images/image1.jpeg	S3://*amzn-s3-demo-bucket1* / bob/images/*	Default	요청자는 디렉터리 자격 증명에 대한 권한 부여의 일부로 접두사 bob/로 시작하지만 IAM 역할에 대한 권한 부여의 일부로 접두사 bob/images/로 제한된 키 이름을 가진 모든 객체에 액세스할 수 있습니다. IAM 역할과 디렉터리 자격 증명 모두 요청된 범위인 bob/images/image1.jpeg 에 대한 액세스를 제공하지만 IAM 역할에는 더 제한적인 권한이 있습니다. 따라서 반환된 범위는 IAM 역할에 대한 보다 제한적인 권한 부여로 제한됩니다.

IAM 자격 증명에 대한 권한 부여 범위	디렉터리 자격 증명에 대한 권한 부여 범위	요청된 범위	반환된 범위	권한	Effect
S3://*amzn-s3-demo-bucket1* / bob/images/*	*amzn-s3-demo-bucket1* / bob/*	S3://*amzn-s3-demo-bucket1* / bob/images/image1.jpeg	S3://*amzn-s3-demo-bucket1* / bob/images/image1.jpeg	Minimal	권한은 Minimal로 설정되므로 자격 증명이 더 큰 범위에 액세스할 수 있더라도 요청된 범위만 bob/images/image1.jpeg 반환됩니다.

지속 시간

durationSeconds 파라미터는 임시 자격 증명의 기간을 초 단위로 설정합니다. 기본값은 3600초(1시간)이지만 요청자(피부여자)는 900초(15분)에서 최대 43200초(12시간)까지 범위를 지정할 수 있습니다. 피부여자가 이 최댓값보다 큰 값을 요청하면 요청이 실패합니다.

> **Note**
>
> 임시 토큰을 요청할 때 위치가 객체인 경우 요청의 targetType 파라미터 값을 Object로 설정하세요. 이 파라미터는 위치가 객체이고 권한 수준이 Minimal인 경우에만 필요합니다. 위치가 버킷 또는 접두사인 경우 이 파라미터를 지정할 필요가 없습니다.

예시

AWS Command Line Interface(AWS CLI), Amazon S3 REST API, AWS SDK를 사용하여 임시 자격 증명을 요청할 수 있습니다. 다음 예제를 참조하십시오.

자세한 내용은 Amazon Simple Storage Service API 참조의 GetDataAccess를 참조하세요.

AWS CLI 사용

AWS CLI를 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치를 참조하세요.

다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

Example 임시 자격 증명 요청

요청:

```
aws s3control get-data-access \
--account-id 111122223333 \
--target s3://amzn-s3-demo-bucket/prefixA* \
--permission READ \
--privilege Default \
--region us-east-2
```

응답:

```
{
"Credentials": {
"AccessKeyId": "Example-key-id",
"SecretAccessKey": "Example-access-key",
"SessionToken": "Example-session-token",
"Expiration": "2023-06-14T18:56:45+00:00"},
"MatchedGrantTarget": "s3://amzn-s3-demo-bucket/prefixA**",
"Grantee": {
    "GranteeType": "IAM",
    "GranteeIdentifier": "arn:aws:iam::111122223333:role/role-name"
 }
}
```

REST API 사용

S3 Access Grants에서 임시 자격 증명 요청을 위한 Amazon S3 REST API 지원에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 GetDataAccess를 참조하세요.

AWS SDK 사용

이 섹션에서는 피부여자가 AWS SDK를 사용하여 S3 Access Grants로부터 임시 자격 증명을 요청하는 방법의 예를 제공합니다.

Java

다음 예시 코드는 피부여자가 S3 데이터에 액세스하는 데 사용하는 임시 자격 증명을 반환합니다. 이 예시 코드를 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

Example 임시 자격 증명 가져오기

요청:

```
public void getDataAccess() {
GetDataAccessRequest getDataAccessRequest = GetDataAccessRequest.builder()
.accountId("111122223333")
.permission(Permission.READ)
.privilege(Privilege.MINIMAL)
.target("s3://amzn-s3-demo-bucket/prefixA*")
.build();
GetDataAccessResponse getDataAccessResponse =
 s3Control.getDataAccess(getDataAccessRequest);
LOGGER.info("GetDataAccessResponse: " + getDataAccessResponse);
}
```

응답:

```
GetDataAccessResponse(
Credentials=Credentials(
AccessKeyId="Example-access-key-id",
SecretAccessKey="Example-secret-access-key",
SessionToken="Example-session-token",
Expiration=2023-06-07T06:55:24Z
))
```

S3 Access Grants에서 제공하는 자격 증명을 사용하여 S3 데이터에 액세스

액세스 권한 부여를 통해 임시 자격 증명을 획득한 피부여자는 해당 임시 자격 증명을 사용하여 Amazon S3 API 작업을 호출하여 데이터에 액세스할 수 있습니다.

피부여자는 AWS Command Line Interface(AWS CLI), AWS SDK, Amazon S3 REST API를 사용하여 S3 데이터에 액세스할 수 있습니다. 또한, AWS Python 및 Java 플러그인을 사용하여 S3 Access Grants를 직접 호출할 수 있습니다.

AWS CLI 사용

S3 Access Grants로부터 임시 자격 증명을 받은 후 피부여자는 해당 자격 증명으로 프로필을 설정하여 데이터를 검색할 수 있습니다.

AWS CLI를 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치를 참조하세요.

다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

Example - 프로필 설정

```
aws configure set aws_access_key_id "$accessKey" --profile access-grants-consumer-access-profile
aws configure set aws_secret_access_key "$secretKey" --profile access-grants-consumer-access-profile
aws configure set aws_session_token "$sessionToken" --profile access-grants-consumer-access-profile
```

다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

Example – S3 데이터 가져오기

피부여자는 get-object AWS CLI 명령을 사용하여 데이터에 액세스할 수 있습니다. 피부여자는 put-object, ls 및 기타 S3 AWS CLI 명령도 사용할 수 있습니다.

```
aws s3api get-object \
--bucket amzn-s3-demo-bucket1 \
--key myprefix \
--region us-east-2 \
--profile access-grants-consumer-access-profile
```

AWS SDK 사용

이 섹션에서는 피부여자가 AWS SDK를 사용하여 S3 데이터에 액세스하는 방법의 예시를 보여줍니다.

Java

> 임시 자격 증명을 사용하여 S3 데이터를 가져오는 방법의 예는 AWS SDK를 사용하여 객체를 가져오는 방법과 Amazon S3 code examples for the AWS SDK for Java 2.x를 참조하세요.

S3 Access Grants에서 지원되는 S3 작업

피부여자는 S3 Access Grants에서 제공하는 임시 자격 증명을 사용하여 액세스 권한이 있는 S3 데이터에 대해 S3 작업을 수행할 수 있습니다. 다음은 피부여자가 수행할 수 있는 허용 가능한 S3 작업 목록입니다. 허용되는 작업은 액세스 권한 부여에 부여된 권한 수준(`READ`, `WRITE` 또는 `READWRITE`)에 따라 다릅니다.

> **Note**
>
> 아래 나열된 Amazon S3 권한 외에도, Amazon S3는 AWS Key Management Service(AWS KMS) Decrypt(`kms:decrypt`) `READ` 권한 또는 AWS KMS GenerateDataKey(`kms:generateDataKey`) `WRITE` 권한을 직접적으로 호출할 수 있습니다. 이러한 권한으로는 AWS KMS 키에 직접 액세스할 수 없습니다.

S3 IAM 작업	API 작업 및 문서	S3 Access Grants 권한	S3 리소스
`s3:GetObject`	GetObject	`READ`	객체
`s3:GetObjectVersion`	GetObject	`READ`	객체
`s3:GetObjectAcl`	GetObjectAcl	`READ`	객체
`s3:GetObjectVersionAcl`	GetObjectAcl	`READ`	객체
`s3:ListMultipartUploads`	ListParts	`READ`	객체
`s3:PutObject`	PutObject, CreateMultipartUpload, UploadPart,	`WRITE`	객체

S3 IAM 작업	API 작업 및 문서	S3 Access Grants 권한	S3 리소스			
	UploadPartCopy, CompleteMultipartUpload					
s3:PutObjectAcl	PutObjectAcl	WRITE	객체			
s3:PutObjectVersionAcl	PutObjectAcl	WRITE	객체			
s3:DeleteObject	DeleteObject	WRITE	객체			
s3:DeleteObjectVersion	DeleteObject	WRITE	객체			
s3:AbortMultipartUpload	AbortMultipartUpload	WRITE	객체			
s3:ListBucket	HeadBucket, ListObjectsV2, ListObjects	READ	버킷			
s3:ListBucketVersions	ListObjectVersions	READ	버킷			
s3:ListBucketMultipartUploads	ListMultipartUploads	READ	버킷			

호출자의 액세스 권한 부여 목록

S3 데이터 소유자는 S3 Access Grants를 사용하여 AWS Identity and Access Management(IAM) ID 또는 AWS IAM Identity Center 기업 디렉터리 ID에 대한 액세스 권한 부여를 만들 수 있습니다. IAM ID 및 IAM Identity Center 디렉터리 ID는 차례로 `ListCallerAccessGrants` API를 사용하여 S3 Access Grants에서 정의한 대로 액세스할 수 있는 모든 Amazon S3 버킷, 접두사 및 객체를 나열할 수 있습니다. 이 API를 사용하면 IAM 또는 디렉터리 ID가 S3 Access Grants를 통해 액세스할 수 있는 모든 S3 데이터를 검색할 수 있습니다.

이 기능을 통해 특정 최종 사용자가 액세스할 수 있는 데이터를 표시하는 애플리케이션을 빌드할 수 있습니다. 예를 들어, 고객이 S3 버킷에 액세스하는 데 사용하는 오픈 소스 UI 구성 요소인 S3용 AWS Storage Browser는 해당 기능을 사용하여 최종 사용자에게 S3 Access Grants를 기반으로 Amazon S3에서 액세스할 수 있는 데이터를 제공합니다. 또 다른 예로, Amazon S3에서 데이터를 탐색, 업로드 또는 다운로드하기 위한 애플리케이션을 빌드할 때 이 기능을 사용하여 최종 사용자가 탐색할 수 있는 애플리케이션의 트리 구조를 빌드할 수 있습니다.

> **Note**
>
> 회사 디렉터리 자격 증명의 경우 호출자의 액세스 권한 부여를 나열할 때 S3 Access Grants는 자격 증명 인식 세션에 사용되는 IAM 자격 증명의 권한을 반환합니다. 자격 증명 인식 세션에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 자격 증명 인식 콘솔 세션을 사용할 수 있는 권한 부여를 참조하세요.

IAM ID이든, 기업 디렉터리 ID든 상관없이 피부여자는 AWS Command Line Interface(AWS CLI), Amazon S3 REST API 및 AWS SDK를 사용하여 액세스 권한 부여 목록을 가져올 수 있습니다.

AWS CLI 사용

AWS CLI를 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치를 참조하세요.

다음 예시 명령을 사용하려면 *`user input placeholders`*를 실제 정보로 대체하세요.

Example 호출자의 액세스 권한 부여 목록

요청:

```
aws s3control list-caller-access-grants \
--account-id 111122223333 \
```

```
--region us-east-2
--max-results 5
```

응답:

```
{
 "NextToken": "6J9S...",
 "CallerAccessGrantsList": [
  {
   "Permission": "READWRITE",
   "GrantScope": "s3://amzn-s3-demo-bucket/prefix1/sub-prefix1/*",
   "ApplicationArn": "NA"
  },
  {
   "Permission": "READWRITE",
   "GrantScope": "s3://amzn-s3-demo-bucket/prefix1/sub-prefix2/*",
   "ApplicationArn": "ALL"
  },
  {
   "Permission": "READWRITE",
   "GrantScope": "s3://amzn-s3-demo-bucket/prefix1/sub-prefix3/*",
   "ApplicationArn": "arn:aws:sso::111122223333:application/ssoins-
ssoins-1234567890abcdef/apl-abcd1234a1b2c3d"
  }
 ]
}
```

Example 버킷에 대한 호출자의 액세스 권한 부여 목록

grantscope 파라미터를 사용하여 결과의 범위를 좁힐 수 있습니다.

요청:

```
aws s3control list-caller-access-grants \
--account-id 111122223333 \
--region us-east-2
--grant-scope "s3://amzn-s3-demo-bucket""
--max-results 1000
```

응답:

```
{
 "NextToken": "6J9S...",
 "CallerAccessGrantsList": [
  {
   "Permission": "READ",
   "GrantScope": "s3://amzn-s3-demo-bucket*",
   "ApplicationArn": "ALL"
  },
  {
   "Permission": "READ",
   "GrantScope": "s3://amzn-s3-demo-bucket/prefix1/*",
   "ApplicationArn": "arn:aws:sso::111122223333:application/ssoins-
ssoins-1234567890abcdef/apl-abcd1234a1b2c3d"
  }
 ]
}
```

REST API 사용

API 호출자의 액세스 권한 부여 목록을 가져오기 위한 Amazon S3 REST API 지원에 대한 내용은 Amazon Simple Storage Service API 참조의 ListCallerAccessGrants를 참조하세요.

AWS SDK 사용

이 섹션에서는 피부여자가 AWS SDK를 사용하여 S3 Access Grants로부터 임시 자격 증명을 요청하는 방법의 예를 제공합니다.

Java

다음 코드 예제는 특정 AWS 계정의 S3 데이터에 대한 API 호출자의 액세스 권한 부여를 반환합니다. 이 예시 코드를 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

Example 호출자의 액세스 권한 부여 목록

요청:

```
Public void ListCallerAccessGrants() {
 ListCallerAccessGrantsRequest listRequest = ListCallerAccessGrantsRequest.builder()
    .withMaxResults(1000)
    .withGrantScope("s3://")
    .accountId("111122223333");
 ListCallerAccessGrantsResponse listResponse =
 s3control.listCallerAccessGrants(listRequest);
```

```
 LOGGER.info("ListCallerAccessGrantsResponse: " + listResponse);
 }
```

응답:

```
ListCallerAccessGrantsResponse(
CallerAccessGrantsList=[
 ListCallerAccessGrantsEntry(
  S3Prefix=s3://amzn-s3-demo-bucket/prefix1/,
  Permission=READ,
  ApplicationArn=ALL
 )
])
```

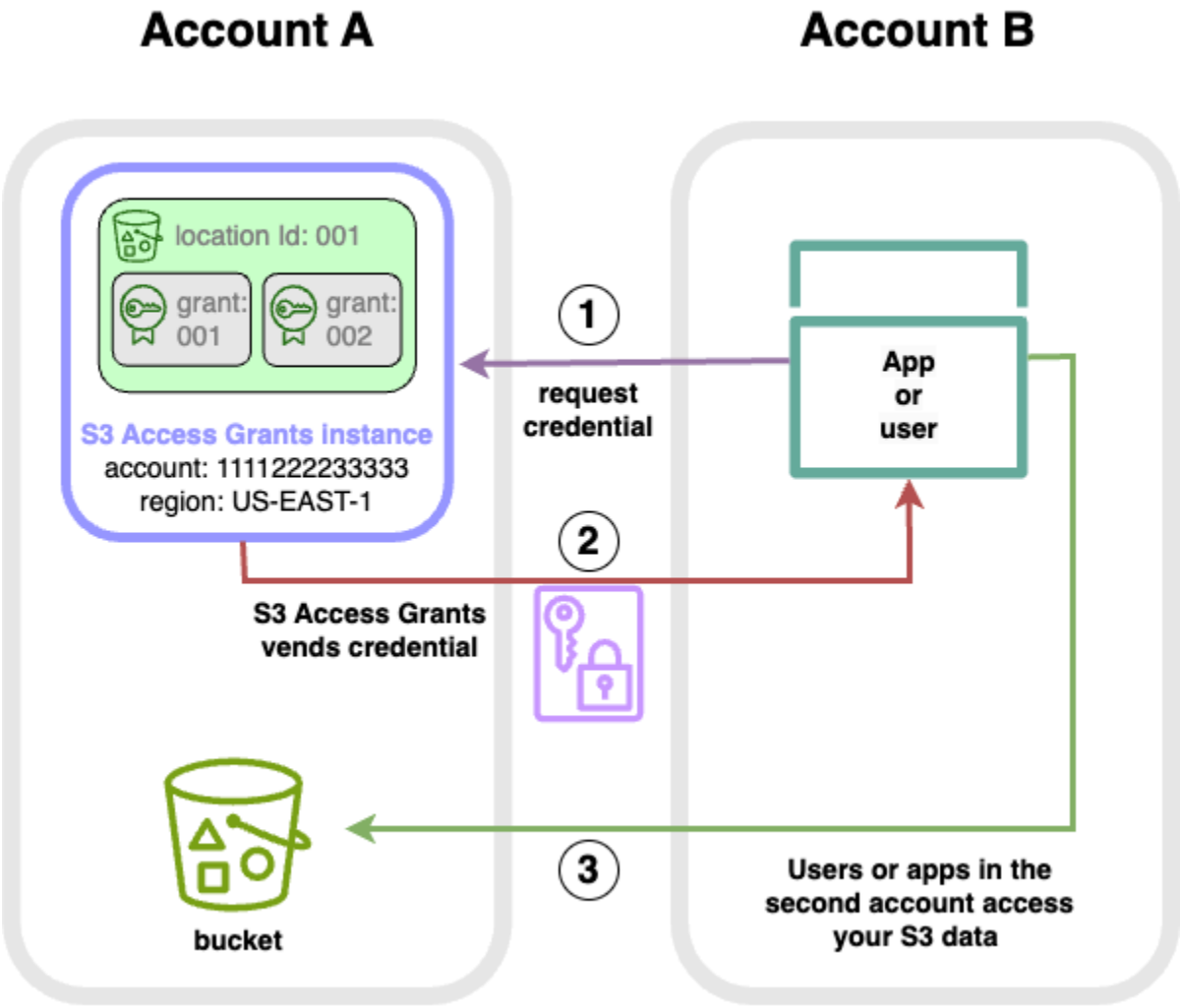
S3 Access Grants 크로스 계정 액세스

S3 Access Grants를 사용하면 다음에 Amazon S3 데이터 액세스 권한을 부여할 수 있습니다.

- 계정 내 AWS Identity and Access Management(IAM) 자격 증명
- 다른 AWS 계정의 IAM 자격 증명
- AWS IAM Identity Center 인스턴스의 디렉터리 사용자 또는 그룹

먼저 다른 계정에 크로스 계정 액세스를 구성합니다. 여기에는 리소스 정책을 사용하여 S3 Access Grants 인스턴스에 대한 액세스 권한을 부여하는 것이 포함됩니다. 그런 다음 권한 부여를 사용하여 S3 데이터(버킷, 접두사 또는 객체)에 대한 액세스 권한을 부여하세요.

크로스 계정 액세스를 구성한 후 다른 계정은 S3 Access Grants에서 Amazon S3 데이터에 대한 임시 액세스 자격 증명을 요청할 수 있습니다. 다음 이미지는 S3 Access Grants를 통한 크로스 계정 S3 액세스의 사용자 흐름을 보여 줍니다.



1. 두 번째 계정(B)의 사용자 또는 애플리케이션이 Amazon S3 데이터가 저장되는 사용자 계정(A)의 S3 Access Grants 인스턴스에서 자격 증명을 요청합니다. 자세한 내용은 S3 Access Grants를 통해 Amazon S3 데이터에 대한 액세스 요청 섹션을 참조하세요.
2. 두 번째 계정에 Amazon S3 데이터에 대한 액세스 권한을 부여하는 권한 부여가 있는 경우 사용자 계정(A)의 S3 Access Grants 인스턴스가 임시 자격 증명을 반환합니다. 액세스 권한 부여에 대한 자세한 내용은 S3 Access Grants에서 권한 부여 사용 섹션을 참조하세요.
3. 두 번째 계정(B)의 사용자 또는 애플리케이션이 S3 Access Grants에서 벤딩한 자격 증명을 사용하여 사용자 계정(A)의 S3 데이터에 액세스합니다.

S3 Access Grants 크로스 계정 액세스 구성

S3 Access Grants를 통해 크로스 계정 S3 액세스 권한을 부여하려면 다음 단계를 따르세요.

- 1단계: 계정(예: S3 데이터가 저장되는 계정 ID 111122223333)에서 S3 Access Grants 인스턴스를 구성합니다.
- 2단계: 계정 111122223333에서 S3 Access Grants 인스턴스에 대한 리소스 정책을 구성하여 두 번째 계정(예: 계정 ID 444455556666)에 액세스 권한을 부여합니다.
- 3단계: 계정 111122223333의 S3 Access Grants 인스턴스로부터 자격 증명을 요청하도록 두 번째 계정 444455556666의 IAM 보안 주체에 대한 IAM 권한을 구성합니다.
- 4단계: 계정 111122223333에 권한 부여를 생성하여 두 번째 계정 444455556666의 IAM 보안 주체에 계정 111122223333의 일부 S3 데이터에 대한 액세스 권한을 부여합니다.

1단계: 계정에서 S3 Access Grants 인스턴스 구성

먼저 Amazon S3 데이터에 대한 액세스를 관리하려면 계정 111122223333에 S3 Access Grants 인스턴스가 있어야 합니다. 공유하려는 S3 데이터가 저장된 각 AWS 리전에서 S3 Access Grants 인스턴스를 생성해야 합니다. 둘 이상의 AWS 리전에서 데이터를 공유하는 경우 각 AWS 리전에 대해 이러한 구성 단계를 반복하세요. S3 데이터가 저장된 AWS 리전에 이미 S3 Access Grants 인스턴스가 있는 경우 다음 단계로 진행합니다. S3 Access Grants 인스턴스를 구성하지 않은 경우 S3 Access Grants 인스턴스 사용 단원을 참조하여 이 단계를 완료하세요.

2단계: S3 Access Grants 인스턴스에 크로스 계정 액세스 권한을 부여하기 위한 리소스 정책 구성

111122223333 계정에서 크로스 계정 액세스를 위한 S3 Access Grants 인스턴스를 생성한 후에는 계정 111122223333의 S3 Access Grants 인스턴스에 리소스 기반 정책을 구성하여 크로스 계정 액세스 권한을 부여합니다. S3 AccessGrants 인스턴스는 리소스 기반 정책을 지원합니다. 적절한 리소스 기반 정책을 적용하면 다른 AWS 계정의 AWS Identity and Access Management(IAM) 사용자 또는 역할에 S3 Access Grants 인스턴스에 대한 액세스 권한을 부여할 수 있습니다. 크로스 계정 액세스는 다음과 같은 권한(작업)만 부여합니다.

- `s3:GetAccessGrantsInstanceForPrefix` - 사용자, 역할 또는 앱이 특정 접두사가 포함된 S3 Access Grants 인스턴스를 검색할 수 있습니다.
- `s3:ListAccessGrants`
- `s3:ListAccessLocations`
- `s3:ListCallerAccessGrants`
- `s3:GetDataAccess` - 사용자, 역할 또는 앱이 S3 Access Grants를 통해 부여받은 액세스 권한을 기반으로 임시 자격 증명을 요청할 수 있습니다. 해당 자격 증명을 사용하여 액세스 권한이 부여된 S3 데이터에 액세스합니다.

이러한 권한 중 리소스 정책에 포함할 권한을 선택할 수 있습니다. S3 Access Grants 인스턴스의 이 리소스 정책은 일반적인 리소스 기반 정책이며 IAM 정책 언어가 지원하는 모든 것을 지원합니다. 동일한 정책에서 예컨대 `aws:PrincipalArn` 조건을 사용하여 계정 `111122223333`의 특정 IAM 자격 증명에 액세스 권한을 부여할 수 있지만, S3 Access Grants를 사용하면 그렇게 할 필요가 없습니다. 대신 S3 Access Grants 인스턴스 내에서 계정에 있는 개별 IAM ID뿐만 아니라 다른 계정을 위한 권한 부여를 생성할 수 있습니다. S3 Access Grants를 통해 각 액세스 권한 부여를 관리함으로써 권한의 규모를 조정할 수 있습니다.

이미 AWS Resource Access Manager(AWS RAM)를 사용하는 경우 이를 사용하여 `s3:AccessGrants` 리소스를 다른 계정이나 조직 내에서 공유할 수 있습니다. 자세한 내용은 공유 AWS 리소스 작업을 참조하세요. AWS RAM를 사용하지 않는 경우 S3 Access Grants API 작업 또는 AWS Command Line Interface(AWS CLI)를 사용하여 리소스 정책을 추가할 수도 있습니다.

S3 콘솔 사용

AWS Resource Access Manager(AWS RAM) 콘솔을 사용하여 `s3:AccessGrants` 리소스를 다른 계정이나 조직 내에서 공유하는 것이 좋습니다. 계정 간에 S3 Access Grants를 공유하려면 다음을 수행합니다.

S3 Access Grants 인스턴스 리소스 정책을 구성하려면:

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. AWS 리전 선택기에서 AWS 리전을 선택합니다.
3. 왼쪽 탐색 창에서 Access Grants를 선택합니다.
4. Access Grants 인스턴스 페이지의 이 계정의 인스턴스 섹션에서 인스턴스 공유를 선택합니다. 그러면 AWS RAM 콘솔로 리디렉션됩니다.
5. 리소스 공유 생성을 선택합니다.
6. 리소스 공유를 생성하기 위한 AWS RAM 단계를 따릅니다. 자세한 내용은 AWS RAM에서 리소스 공유 생성을 참조하세요.

AWS CLI 사용

AWS CLI를 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치를 참조하세요.

`put-access-grants-instance-resource-policy` CLI 명령을 사용하여 리소스 정책을 추가할 수 있습니다.

사용자 계정 111122223333의 S3 Access Grants 인스턴스에 대한 크로스 계정 액세스 권한을 두 번째 계정 444455556666에 부여하려면 계정 111122223333의 S3 Access Grants 인스턴스에 대한 리소스 정책이 두 번째 계정 444455556666에 다음 작업을 수행하기 위한 권한을 부여해야 합니다.

- `s3:ListAccessGrants`
- `s3:ListAccessGrantsLocations`
- `s3:GetDataAccess`
- `s3:GetAccessGrantsInstanceForPrefix`

S3 Access Grants 인스턴스 리소스 정책에서 S3 Access Grants 인스턴스의 ARN을 `Resource`로 지정하고 두 번째 계정 444455556666은 `Principal`로 지정합니다. 다음 예시를 사용하려면 *### ### ## ###*를 실제 정보로 바꾸세요.

```
{
"Version": "2012-10-17",
"Statement": [
{
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
},
 "Action": [
  "s3:ListAccessGrants",
  "s3:ListAccessGrantsLocations",
  "s3:GetDataAccess",
  "s3:GetAccessGrantsInstanceForPrefix"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
} ]
}
```

S3 Access Grants 인스턴스 리소스 정책을 추가하거나 업데이트하려면 다음 명령을 사용할 수 있습니다. 다음 예시 명령을 사용할 때는 *user input placeholders*를 실제 정보로 바꾸세요.

Example S3 Access Grants 인스턴스 리소스 정책 추가 또는 업데이트

```
aws s3control put-access-grants-instance-resource-policy \
--account-id 111122223333 \
```

```
 --policy file://resourcePolicy.json \
 --region us-east-2
 {
  "Policy": "{\n
    \"Version\": \"2012-10-17\",\n
    \"Statement\": [{\n
   \"Effect\": \"Allow\",\n
   \"Principal\": {\n
     \"AWS\": \"444455556666\"\n
   },\n
   \"Action\": [\n
     \"s3:ListAccessGrants\",\n
     \"s3:ListAccessGrantsLocations\",\n
     \"s3:GetDataAccess\",\n
     \"s3:GetAccessGrantsInstanceForPrefix\",\n
     \"s3:ListCallerAccessGrants"\n
   ],\n
   \"Resource\": \"arn:aws:s3:us-east-2:111122223333:access-grants/default\"\n
     }\n
    ]\n
    }\n",
  "CreatedAt": "2023-06-16T00:07:47.473000+00:00"
 }
```

Example S3 Access Grants 리소스 정책 가져오기

또한 CLI를 사용하여 S3 Access Grants 인스턴스의 리소스 정책을 가져오거나 삭제할 수 있습니다.

S3 Access Grants 리소스 정책을 가져오려면 다음 예시 명령을 사용하세요. 이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3control get-access-grants-instance-resource-policy \
--account-id 111122223333 \
--region us-east-2

{
"Policy": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow
\",\"Principal\":{\"AWS\":\"arn:aws:iam::111122223333:root\"},\"Action\":
[\"s3:ListAccessGrants\",\"s3:ListAccessGrantsLocations\",\"s3:GetDataAccess\",
\"s3:GetAccessGrantsInstanceForPrefix\",\"s3:ListCallerAccessGrants\"],\"Resource\":
\"arn:aws:
s3:us-east-2:111122223333:access-grants/default\"}]}",
"CreatedAt": "2023-06-16T00:07:47.473000+00:00"
```

```
}
```

Example S3 Access Grants 리소스 정책 삭제

S3 Access Grants 리소스 정책을 삭제하려면 다음 예시 명령을 사용하세요. 이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3control delete-access-grants-instance-resource-policy \
--account-id 111122223333 \
--region us-east-2

// No response body
```

REST API 사용

PutAccessGrantsInstanceResourcePolicy API를 사용하여 리소스 정책을 추가할 수 있습니다.

사용자 계정 111122223333의 S3 Access Grants 인스턴스에 대한 크로스 계정 액세스 권한을 두 번째 계정 444455556666에 부여하려면 계정 111122223333의 S3 Access Grants 인스턴스에 대한 리소스 정책이 두 번째 계정 444455556666에 다음 작업을 수행하기 위한 권한을 부여해야 합니다.

- `s3:ListAccessGrants`
- `s3:ListAccessGrantsLocations`
- `s3:GetDataAccess`
- `s3:GetAccessGrantsInstanceForPrefix`

S3 Access Grants 인스턴스 리소스 정책에서 S3 Access Grants 인스턴스의 ARN을 `Resource`로 지정하고 두 번째 계정 444455556666은 `Principal`로 지정합니다. 다음 예시를 사용하려면 *### ### ## ###*를 실제 정보로 바꾸세요.

```
{
"Version": "2012-10-17",
"Statement": [
{
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
},
 "Action": [
```

```
  "s3:ListAccessGrants",
  "s3:ListAccessGrantsLocations",
  "s3:GetDataAccess",
  "s3:GetAccessGrantsInstanceForPrefix"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
} ]
}
```

그러고 나서 PutAccessGrantsInstanceResourcePolicy API를 사용하여 정책을 구성할 수 있습니다.

S3 Access Grants 인스턴스의 리소스 정책을 가져오거나 업데이트, 삭제하기 위한 REST API 지원에 대한 내용은 Amazon Simple Storage Service API 참조의 다음 단원을 참조하세요.

- PutAccessGrantsInstanceResourcePolicy
- GetAccessGrantsInstanceResourcePolicy
- DeleteAccessGrantsInstanceResourcePolicy

AWS SDK 사용

이 단원에서는 일부 S3 데이터에 대한 액세스 권한을 두 번째 AWS 계정에 부여하도록 S3 Access Grants 리소스 정책을 구성하는 방법에 대한 AWS SDK 예시를 제공합니다.

Java

S3 Access Grants 인스턴스에 대한 크로스 계정 액세스를 관리하기 위해 리소스 정책을 추가 또는 업데이트하거나 가져오거나 삭제합니다.

Example S3 Access Grants 인스턴스 리소스 정책 추가 또는 업데이트

사용자 계정 111122223333의 S3 Access Grants 인스턴스에 대한 크로스 계정 액세스 권한을 두 번째 계정 444455556666에 부여하려면 계정 111122223333의 S3 Access Grants 인스턴스에 대한 리소스 정책이 두 번째 계정 444455556666에 다음 작업을 수행하기 위한 권한을 부여해야 합니다.

- `s3:ListAccessGrants`
- `s3:ListAccessGrantsLocations`
- `s3:GetDataAccess`
- `s3:GetAccessGrantsInstanceForPrefix`

S3 Access Grants 인스턴스 리소스 정책에서 S3 Access Grants 인스턴스의 ARN을 `Resource`로 지정하고 두 번째 계정 444455556666은 `Principal`로 지정합니다. 다음 예시를 사용하려면 *### ## ## ###*를 실제 정보로 바꾸세요.

```
{
"Version": "2012-10-17",
"Statement": [
{
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
},
 "Action": [
  "s3:ListAccessGrants",
  "s3:ListAccessGrantsLocations",
  "s3:GetDataAccess",
  "s3:GetAccessGrantsInstanceForPrefix"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
} ]
}
```

S3 Access Grants 인스턴스 리소스 정책을 추가하거나 업데이트하려면 다음 예시 코드를 사용하세요.

```
public void putAccessGrantsInstanceResourcePolicy() {
 PutAccessGrantsInstanceResourcePolicyRequest putRequest =
 PutAccessGrantsInstanceResourcePolicyRequest.builder()
 .accountId(111122223333)
 .policy(RESOURCE_POLICY)
 .build();
 PutAccessGrantsInstanceResourcePolicyResponse putResponse =
 s3Control.putAccessGrantsInstanceResourcePolicy(putRequest);
 LOGGER.info("PutAccessGrantsInstanceResourcePolicyResponse: " + putResponse);
 }
```

응답:

```
PutAccessGrantsInstanceResourcePolicyResponse(
 Policy={
 "Version": "2012-10-17",
 "Statement": [{
```

```
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
 },
 "Action": [
 "s3:ListAccessGrants",
 "s3:ListAccessGrantsLocations",
 "s3:GetDataAccess",
 "s3:GetAccessGrantsInstanceForPrefix",
 "s3:ListCallerAccessGrants"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
 }]
 }
 )
```

Example S3 Access Grants 리소스 정책 가져오기

S3 Access Grants 리소스 정책을 가져오려면 다음 예시 코드를 사용하세요. 다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
public void getAccessGrantsInstanceResourcePolicy() {
 GetAccessGrantsInstanceResourcePolicyRequest getRequest =
 GetAccessGrantsInstanceResourcePolicyRequest.builder()
 .accountId(111122223333)
 .build();
 GetAccessGrantsInstanceResourcePolicyResponse getResponse =
 s3Control.getAccessGrantsInstanceResourcePolicy(getRequest);
 LOGGER.info("GetAccessGrantsInstanceResourcePolicyResponse: " + getResponse);
 }
```

응답:

```
GetAccessGrantsInstanceResourcePolicyResponse(
 Policy={"Version":"2012-10-17","Statement":[{"Effect":"Allow","Principal":
{"AWS":"arn:aws:iam::444455556666:root"},"Action":
["s3:ListAccessGrants","s3:ListAccessGrantsLocations","s3:GetDataAccess","s3:GetAccessGrants
east-2:111122223333:access-grants/default"}]},
 CreatedAt=2023-06-15T22:54:44.319Z
 )
```

Example S3 Access Grants 리소스 정책 삭제

S3 Access Grants 리소스 정책을 삭제하려면 다음 예시 코드를 사용하세요. 다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
public void deleteAccessGrantsInstanceResourcePolicy() {
 DeleteAccessGrantsInstanceResourcePolicyRequest deleteRequest =
 DeleteAccessGrantsInstanceResourcePolicyRequest.builder()
 .accountId(111122223333)
 .build();
 DeleteAccessGrantsInstanceResourcePolicyResponse deleteResponse =
 s3Control.putAccessGrantsInstanceResourcePolicy(deleteRequest);
 LOGGER.info("DeleteAccessGrantsInstanceResourcePolicyResponse: " + deleteResponse);
 }
```

응답:

```
DeleteAccessGrantsInstanceResourcePolicyResponse()
```

3단계: 두 번째 계정의 IAM 자격 증명에 사용자 계정의 S3 Access Grants 인스턴스를 직접 호출할 수 있는 권한을 부여합니다.

Amazon S3 데이터 소유자가 계정 111122223333의 S3 Access Grants 인스턴스에 대한 크로스 계정 정책을 구성한 후, 두 번째 계정 444455556666의 소유자는 해당 IAM 사용자 또는 역할에 대한 자격 증명 기반 정책을 생성하고 소유자는 해당 사용자 또는 역할에 S3 Access Grants 인스턴스에 대한 액세스 권한을 부여해야 합니다. S3 Access Grants 인스턴스 리소스 정책에서 부여된 내용과 부여하려는 권한에 따라 자격 증명 기반 정책에 다음 작업 중 하나 이상을 포함하세요.

- `s3:ListAccessGrants`
- `s3:ListAccessGrantsLocations`
- `s3:GetDataAccess`
- `s3:GetAccessGrantsInstanceForPrefix`
- `s3:ListCallerAccessGrants`

AWS 크로스 계정 액세스 패턴에 따라 두 번째 계정444455556666의 IAM 사용자 또는 역할은 이러한 권한 중 하나 이상을 명시적으로 보유해야 합니다. 예를 들어, IAM 사용자 또는 역할이 계정

111122223333의 S3 Access Grants 인스턴스를 직접 호출하여 자격 증명을 요청할 수 있도록 `s3:GetDataAccess` 권한을 부여합니다.

이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
  "Effect": "Allow",
  "Action": [
   "s3:GetDataAccess",
  ],
   "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
 }
 ]
}
```

IAM 자격 증명 기반 정책 편집에 대한 자세한 내용은 AWS Identity and Access Management 설명서의 IAM 정책 편집을 참조하세요.

4단계: 사용자 계정의 S3 Access Grants 인스턴스에서 권한 부여를 생성하여 두 번째 계정의 IAM 자격 증명에 일부 S3 데이터에 대한 액세스 권한을 부여합니다.

최종 구성 단계에서는 계정 111122223333의 S3 Access Grants 인스턴스에서 권한 부여를 생성하여 두 번째 계정 444455556666의 IAM 자격 증명에 사용자 계정의 일부 S3 데이터에 대한 액세스 권한을 부여할 수 있습니다. Amazon S3 콘솔, CLI, API 및 SDK를 사용하여 이 작업을 수행할 수 있습니다. 자세한 내용은 권한 부여 생성 섹션을 참조하세요.

권한 부여에서 두 번째 계정의 IAM 자격 증명의 AWS ARN을 지정하고 액세스 권한을 부여하려는 S3 데이터의 위치(버킷, 접두사 또는 객체)를 지정합니다. 이 위치는 S3 Access Grants 인스턴스에 등록되어 있어야 합니다. 자세한 내용은 위치 등록 섹션을 참조하세요. 원한다면 하위 접두사를 지정할 수 있습니다. 예를 들어, 액세스 권한을 부여하려는 위치가 버킷이고 해당 버킷의 특정 객체로 액세스를 추가적으로 제한하려면 `S3SubPrefix` 필드에 객체 키 이름을 전달합니다. 또는 키 이름이 특정 접두사(예: `2024-03-research-results/`)로 시작하는 버킷 내 객체로 액세스를 제한하려면 `S3SubPrefix=2024-03-research-results/`를 전달합니다.

다음은 두 번째 계정의 자격 증명을 위한 액세스 권한 부여를 생성하는 예시 CLI 명령입니다. 자세한 정보는 권한 부여 생성을 참조하세요. 이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3control create-access-grant \
--account-id 111122223333 \
--access-grants-location-id default \
--access-grants-location-configuration S3SubPrefix=prefixA* \
--permission READ \
--grantee GranteeType=IAM,GranteeIdentifier=arn:aws:iam::444455556666:role/data-consumer-1
```

크로스 계정 액세스를 구성한 후 두 번째 계정의 사용자 또는 역할은 다음을 수행할 수 있습니다.

- `ListAccessGrantsInstances`를 직접 호출하여 AWS RAM를 통해 공유된 S3 Access Grants 인스턴스를 나열합니다. 자세한 내용은 S3 Access Grants 인스턴스의 세부 정보 가져오기 섹션을 참조하세요.
- S3 Access Grants로부터 임시 자격 증명을 요청합니다. 요청을 만드는 방법에 대한 자세한 내용은 S3 Access Grants를 통해 Amazon S3 데이터에 대한 액세스 요청 단원을 참조하세요.

S3 Access Grants에 AWS 태그 사용

Amazon S3 Access Grants의 태그는 Amazon S3의 객체 태그와 유사한 특성을 가지고 있습니다. 각 태그는 키-값 페어입니다. 태그를 지정할 수 있는 S3 Access Grants의 리소스는 S3 Access Grants 인스턴스, 위치 및 권한 부여입니다.

> **Note**
>
> S3 Access Grants에서는 태그 지정에 객체 태그 지정과는 다른 API 작업을 사용합니다. S3 Access Grants는 TagResource, UntagResource 및 ListTagsForResource API 작업을 사용하며, 리소스는 S3 Access Grants 인스턴스, 등록된 위치 또는 액세스 권한 부여일 수 있습니다.

객체 태그와 마찬가지로 다음과 같은 제한이 적용됩니다.

- S3 Access Grants 리소스를 만들 때 새로운 리소스에 태그를 추가하거나 기존 리소스에 태그를 추가할 수 있습니다.
- 한 리소스에 태그를 최대 10개까지 연결할 수 있습니다. 여러 태그가 동일한 리소스에 연결된 경우 고유한 태그 키가 있어야 합니다.
- 태그 키의 최대 길이는 128개 유니코드 문자이며, 태그 값의 최대 길이는 256개 유니코드 문자입니다. 태그는 내부적으로 UTF-16 형식으로 표시됩니다. UTF-16에서 문자는 1 또는 2자 위치를 차지합니다.

- 키와 값은 대/소문자를 구분합니다.

태그 제한에 대한 자세한 내용은 AWS Billing 사용 설명서의 사용자 정의 태그 제한을 참조하세요.

AWS Command Line Interface(AWS CLI), Amazon S3 REST API 또는 AWS SDK를 사용하여 S3 Access Grants의 리소스에 태그를 지정할 수 있습니다.

AWS CLI 사용

AWS CLI를 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치를 참조하세요.

S3 Access Grants 리소스를 생성할 때 또는 생성한 후에 리소스에 태그를 지정할 수 있습니다. 다음 예시는 S3 Access Grants 인스턴스에 태그를 지정하거나 태그를 해제하는 방법을 보여줍니다. 등록된 위치 및 액세스 권한 부여에도 유사한 작업을 수행할 수 있습니다.

다음 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

Example - 태그를 지정하여 S3 Access Grants 인스턴스 생성

```
aws s3control create-access-grants-instance \
 --account-id 111122223333 \
 --profile access-grants-profile \
 --region us-east-2 \
 --tags Key=tagKey1,Value=tagValue1
```

응답:

```
 {
    "CreatedAt": "2023-10-25T01:09:46.719000+00:00",
    "AccessGrantsInstanceId": "default",
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default"
}
```

Example - 이미 생성된 S3 Access Grants 인스턴스에 태그 지정

```
aws s3control tag-resource \
--account-id 111122223333 \
--resource-arn "arn:aws:s3:us-east-2:111122223333:access-grants/default" \
--profile access-grants-profile \
--region us-east-2 \
--tags Key=tagKey2,Value=tagValue2
```

Example - S3 Access Grants 인스턴스의 태그 나열

```
aws s3control list-tags-for-resource \
--account-id 111122223333 \
--resource-arn "arn:aws:s3:us-east-2:111122223333:access-grants/default" \
--profile access-grants-profile \
--region us-east-2
```

응답:

```
{
    "Tags": [
        {
            "Key": "tagKey1",
            "Value": "tagValue1"
        },
        {
            "Key": "tagKey2",
            "Value": "tagValue2"
        }
    ]
}
```

Example - S3 Access Grants 인스턴스의 태그 해제

```
aws s3control untag-resource \
 --account-id 111122223333 \
 --resource-arn "arn:aws:s3:us-east-2:111122223333:access-grants/default" \
 --profile access-grants-profile \
 --region us-east-2 \
 --tag-keys "tagKey2"
```

REST API 사용

Amazon S3 API를 사용하여 S3 Access Grants 인스턴스, 등록된 위치 또는 액세스 권한 부여에 태그를 지정하거나 해제하거나 나열할 수 있습니다. S3 Access Grants 태그 관리를 위한 REST API 지원에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 다음 섹션을 참조하세요.

- TagResource
- UntagResource
- ListTagsForResource

S3 Access Grants 한도

S3 Access Grants에는 다음과 같은 한도가 있습니다.

> **Note**
>
> 사용 사례가 이러한 한도를 초과하는 경우 AWS Support에 문의하여 한도 상향 조정을 요청하세요.

S3 Access Grants 인스턴스

계정당 AWS 리전별로 S3 Access Grants 인스턴스 1개를 생성할 수 있습니다. S3 Access Grants 인스턴스 생성을 참조하세요.

S3 Access Grants 위치

S3 Access Grants 인스턴스당 1,000개의 S3 Access Grants 위치를 등록할 수 있습니다. Register an S3 Access Grants location을 참조하세요.

권한 부여

S3 Access Grants 인스턴스당 10만 개의 권한 부여를 생성할 수 있습니다. 권한 부여 생성을 참조하세요.

S3 Access Grants AWS 리전

S3 Access Grants는 현재 다음 AWS 리전에서 사용할 수 있습니다.

AWS 리전 코드	AWS 리전 이름
us-east-1	미국 동부(버지니아 북부)
us-east-2	미국 동부(오하이오)
us-west-1	미국 서부(캘리포니아 북부)
us-west-2	미국 서부(오리건)
af-south-1	아프리카(케이프타운)

AWS 리전 코드	AWS 리전 이름
ap-east-1	아시아 태평양(홍콩)
ap-northeast-1	아시아 태평양(도쿄)
ap-northeast-2	아시아 태평양(서울)
ap-northeast-3	아시아 태평양(오사카)
ap-south-1	아시아 태평양(뭄바이)
ap-south-2	아시아 태평양(하이데라바드)
ap-southeast-1	아시아 태평양(싱가포르)
ap-southeast-2	아시아 태평양(시드니)
ap-southeast-3	아시아 태평양(자카르타)
ap-southeast-4	아시아 태평양(멜버른)
ca-central-1	캐나다(중부)
ca-west-1	캐나다 서부(캘거리)
eu-central-1	유럽(프랑크푸르트)
eu-central-2	유럽(취리히)
eu-north-1	유럽(스톡홀름)
eu-south-1	유럽(밀라노)
eu-south-2	유럽(스페인)
eu-west-1	유럽(아일랜드)
eu-west-2	유럽(런던)
eu-west-3	유럽(파리)

AWS 리전 코드	AWS 리전 이름
il-central-1	이스라엘(텔아비브)
me-central-1	중동(UAE)
me-south-1	중동(바레인)
sa-east-1	남아메리카(상파울루)
us-gov-east-1	AWS GovCloud(미국 동부)
us-gov-west-1	AWS GovCloud(미국 서부)

S3 Access Grants 통합

S3 Access Grants는 다음 AWS 서비스 및 기능과 함께 사용할 수 있습니다. 이 페이지는 새로운 통합이 제공되는 대로 업데이트될 예정입니다.

> **Tip**
>
> 이 AWS workshop for S3 Access Grants는 AWS Identity and Access Management(IAM) 사용자, IAM Identity Center 사용자, Amazon EMR 및 AWS Transfer Family에서 S3 Access Grants를 사용하는 방법을 안내합니다.

Amazon Athena

- IAM Identity Center 지원 Athena 작업 그룹 사용

Amazon EMR

- S3 Access Grants를 사용하여 Amazon EMR 클러스터 시작

Amazon EMR on EKS

- S3 Access Grants를 사용하여 Amazon EMR on EKS 클러스터 시작

Amazon EMR Serverless 애플리케이션

- S3 Access Grants를 사용하여 Amazon EMR Serverless 애플리케이션 시작

Amazon Redshift

Amazon S3 Access Grants 및 Amazon Redshift 통합

Amazon SageMaker AI Studio

Amazon SageMaker AI Studio 및 SDK for Python(Boto3) 플러그인과 함께 Amazon S3 Access Grants 사용

이제 SDK for Python(Boto3) 플러그인을 사용하면 Amazon SageMaker AI Studio 노트북에서 S3 Access Grants를 더 쉽게 사용할 수 있습니다. IAM 보안 주체와 AWS IAM Identity Center 디렉터리 사용자를 위한 액세스 권한 부여를 미리 설정하세요. Amazon SageMaker AI Studio는 기본적으로 ID 제공업체 디렉터리 사용자를 지원하지 않지만, 이러한 ID가 S3 Access Grants를 통해 S3 데이터에 액세스할 수 있도록 하는 플러그인으로 사용자 지정 Python 코드를 작성할 수 있습니다. 데이터 액세스는 Amazon SageMaker AI를 통하지 않고 플러그인의 지원으로 이루어집니다.

AWS Glue

AWS Glue와 함께 Amazon S3 Access Grants 사용

AWS IAM Identity Center

애플리케이션 간 신뢰할 수 있는 ID 전파

AWS Transfer Family

AWS Transfer Family에 대해 Amazon S3 Access Grants 구성

Storage Browser for S3

Storage Browser for S3를 사용하여 대규모 데이터 액세스 관리

오픈소스 Python 프레임워크

이제 Amazon S3 Access Grants가 오픈소스 Python 프레임워크와 통합됨

ACL을 사용한 액세스 관리

액세스 제어 목록(ACL)은 리소스 기반 옵션 중 하나로, 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.

S3 객체 소유권은 버킷에 업로드되는 객체의 소유권을 제어하고 ACL을 비활성화 또는 활성화하는 데 사용할 수 있는 Amazon S3 버킷 수준 설정입니다. 기본적으로 객체 소유권은 버킷 소유자 적용 설정으로 설정되며 모든 ACL이 비활성화되어 있습니다. ACL이 비활성화되면 버킷 소유자는 버킷의 모든 객체를 소유하고 액세스 관리 정책을 사용하여 객체에 대한 액세스를 독점적으로 관리합니다.

Amazon S3의 최신 사용 사례 대부분은 더 이상 ACL을 사용할 필요가 없습니다. 각 객체에 대해 액세스를 개별적으로 제어할 필요가 있는 드문 상황을 제외하고는 ACL을 비활성화한 채로 두는 것이 좋습니다. ACL을 비활성화하면 누가 객체를 버킷에 업로드했는지에 관계없이 정책을 사용하여 버킷의 모든 객체에 대한 액세스를 제어할 수 있습니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

> **Important**
>
> 범용 버킷이 S3 객체 소유권에 대해 버킷 소유자 적용 설정을 사용하는 경우 정책을 사용하여 범용 버킷과 버킷의 객체에 대한 액세스 권한을 부여해야 합니다. 버킷 소유자 적용 설정이 활성화된 상태에서 액세스 제어 목록(ACL) 설정 또는 ACL 업데이트 요청은 실패하고 `AccessControlListNotSupported` 오류 코드를 반환합니다. ACL 읽기 요청은 계속 지원됩니다.

ACL에 대한 자세한 내용은 다음 주제를 참조하십시오.

주제

- ACL(액세스 제어 목록) 개요
- ACL 구성
- ACL에 대한 정책 예시

ACL(액세스 제어 목록) 개요

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.

S3 객체 소유권은 버킷에 업로드되는 객체의 소유권을 제어하고 ACL을 비활성화 또는 활성화하는 데 사용할 수 있는 Amazon S3 버킷 수준 설정입니다. 기본적으로 객체 소유권은 버킷 소유자 적용 설정으로 설정되며 모든 ACL이 비활성화되어 있습니다. ACL이 비활성화되면 버킷 소유자는 버킷의 모든 객체를 소유하고 액세스 관리 정책을 사용하여 객체에 대한 액세스를 독점적으로 관리합니다.

Amazon S3의 최신 사용 사례 대부분은 더 이상 ACL을 사용할 필요가 없습니다. 각 객체에 대해 액세스를 개별적으로 제어할 필요가 있는 드문 상황을 제외하고는 ACL을 비활성화한 채로 두는 것이 좋습니다. ACL을 비활성화하면 누가 객체를 버킷에 업로드했는지에 관계없이 정책을 사용하여 버킷의 모든 객체에 대한 액세스를 제어할 수 있습니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

> **Important**
>
> 범용 버킷이 S3 객체 소유권에 대해 버킷 소유자 적용 설정을 사용하는 경우 정책을 사용하여 범용 버킷과 버킷의 객체에 대한 액세스 권한을 부여해야 합니다. 버킷 소유자 적용 설정이 활성화된 상태에서 액세스 제어 목록(ACL) 설정 또는 ACL 업데이트 요청은 실패하고 `AccessControlListNotSupported` 오류 코드를 반환합니다. ACL 읽기 요청은 계속 지원됩니다.

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 계정 정식 사용자 ID로 소유자를 식별하는 `Owner` 요소가 포함되어 있습니다. 정식 사용자 ID를 찾는 방법에 대한 자세한 내용은 AWS 계정 정식 사용자 ID 찾기을 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.

주제

- 피부여자란?
- 부여할 수 있는 권한
- 일반적인 Amazon S3 요청에 대한 aclRequired 값
- 샘플 ACL
- 미리 제공된 ACL

피부여자란?

> **Important**
>
> 지원 종료 공지: 2025년 10월 1일부터 Amazon S3는 새 이메일 피부여자 액세스 제어 목록(ACL) 생성에 대한 지원을 중단할 예정입니다. 이 날짜 이전에 생성된 이메일 피부여자 ACL은 계속 작동하며 AWS Management Console, 명령줄 인터페이스(CLI), SDK 및 REST API를 통해 액세스할 수 있습니다. 그러나 더 이상 새 이메일 피부여자 ACL을 생성할 수 없습니다. 2025년 7월 15일부터 2025년 10월 1일 사이에 새 이메일 피부여자 ACL을 생성하려고 하면 Amazon S3에 대한 요청에서 `HTTP 405` 오류 발생률이 증가할 것입니다.
> 이 변경 사항은 미국 동부(버지니아 북부), 미국 서부(캘리포니아 북부), 미국 서부(오리건), 아시아 태평양(싱가포르), 아시아 태평양(시드니), 아시아 태평양(도쿄), 유럽(아일랜드), 남아메리카(상파울루) AWS 리전에만 영향을 미칩니다.

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 적은 경우, Amazon S3에서 해당 계정의 정식 사용자 ID를 찾아서 ACL에 추가합니다. 결과 ACL에는 항상 AWS 계정의 이메일 주소가 아닌 AWS 계정의 정식 사용자 ID가 포함됩니다.

액세스 권한을 부여할 때 *type*="*value*" 쌍으로 각 피부여자를 지정합니다. 여기서 *type*은 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)
> - 미국 서부(오리건)
> - 아시아 태평양(싱가포르)
> - 아시아 태평양(시드니)
> - 아시아 태평양(도쿄)
> - 유럽(아일랜드)
> - 남아메리카(상파울루)
>
> 모든 Amazon S3 지원 리전 및 엔드포인트 목록은 Amazon Web Services 일반 참조에서 리전 및 엔드포인트를 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@example.com", emailAddress="abc@example.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 설명서에서 역할을 만들어 IAM 사용자에게 권한 위임 단원을 참조하십시오.

AWS 계정 정식 사용자 ID 찾기

정식 사용자 ID는 AWS 계정과 연결되어 있습니다. 이 ID는 다음과 같은 긴 문자열입니다.

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

계정의 표준 사용자 ID를 찾는 방법에 대한 자세한 내용은 AWS 계정 관리 참조 안내서의 AWS 계정의 표준 사용자 ID 찾기를 참조하세요.

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 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 섹션을 참조하세요.

Amazon S3의 미리 정의된 그룹

Amazon S3에는 여러 개의 미리 정의된 그룹이 있습니다. 그룹에 계정 액세스를 허용하려면 정식 사용자 ID 대신 Amazon S3 URI 하나를 지정합니다. Amazon S3는 다음과 같은 미리 정의된 그룹을 제공합니다.

- 인증된 사용자 그룹(Authenticated Users group) – `http://acs.amazonaws.com/groups/global/AuthenticatedUsers`로 표시.

이 그룹은 모든 AWS 계정에 있습니다. 이 그룹에 대한 액세스 권한은 모든 AWS 계정에 리소스 액세스를 허용합니다. 단, 모든 요청에는 서명을(인증) 해야 합니다.

> **Warning**
>
> 인증된 사용자 그룹에 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` 권한은 비소유자가 기존 객체를 덮어쓰거나 삭제하는 것을 거부하지만, 여전히 `WRITE` 권한은 요금 청구를 받는 버킷에 모든 사람이 객체를 저장할 수 있도록 허용합니다. 이러한 권한에 대한 자세한 내용은 다음 부여할 수 있는 권한 단원을 참조하십시오.

- 로그 전달 그룹(Log Delivery group) – `http://acs.amazonaws.com/groups/s3/LogDelivery`로 표시.

  버킷 내 `WRITE` 권한으로 이 그룹은 버킷에 서버 액세스 로그를 쓸 수 있습니다(서버 액세스 로깅을 사용한 요청 로깅 참조).

> **Note**
>
> ACL을 사용하는 경우, 피부여자는 AWS 계정이나 미리 정의된 Amazon S3 그룹 중 하나가 됩니다. 하지만, IAM 사용자는 피부여자가 될 수 없습니다. AWS 사용자 및 IAM 내 권한에 대한 자세한 내용은 AWS Identity and Access Management 사용을 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.

Amazon S3 콘솔의 ACL 권한에 대한 자세한 내용은 ACL 구성 단원을 참조하십시오.

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`	피부여자가 버킷에서 새 객체를 생성할 수 있습니다. 기존 객체의 버킷 및 객체 소유자는 해당 객체의 삭제 및 덮어쓰기도 허용합니다.	해당 사항 없음
`READ_ACP`	피부여자에게 버킷 ACL 읽기를 허용합니다	피부여자에게 객체 ACL 읽기를 허용합니다
`WRITE_ACP`	피부여자에게 해당 버킷에 대한 ACL 쓰기를 허용합니다	피부여자에게 해당 객체에 대한 ACL 쓰기를 허용합니다
`FULL_CONTROL`	피부여자에게 버킷에 대한 `READ`, `WRITE`, `READ_ACP` 및 `WRITE_ACP` 권한 허용	피부여자에게 버킷에 대한 `READ`, `READ_ACP` 및 `WRITE_ACP` 권한 허용

> **Warning**
>
> S3 버킷과 객체에 액세스 권한을 부여할 때 주의하십시오. 예를 들어, 버킷에 `WRITE` 액세스 권한을 부여하면 피부여자가 버킷에서 객체를 생성할 수 있습니다. 권한을 부여하기 전에 이 ACL(액세스 제어 목록) 개요 섹션 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 S3의 정책 작업 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 Amazon S3의 ID 및 액세스 관리을 참조하십시오.

Amazon S3 콘솔의 ACL 권한에 대한 자세한 내용은 ACL 구성 단원을 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` 및 `s3:GetObjectVersion`
WRITE	`s3:PutObject` 버킷 소유자는 버킷의 모든 객체를 생성, 덮어쓰기 및 삭제할 수 있으며 객체 소유자는 객체에 대해 `FULL_CONTROL` 을(를) 소유합니다. 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`

ACL 권한	버킷에 대한 ACL 권한 부여 시 해당 액세스 정책 권한	객체에 대한 ACL 권한 부여 시 해당 액세스 정책 권한
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. 다음 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 ACL이 필요합니다.

ACL 관련 헤더를 포함하는 정책 예는 버킷 소유자가 전체 권한을 가져오도록 요구하는 조건으로 s3:PutObject 권한 부여을 참조하십시오. Amazon S3에 사용되는 조건 키의 전체 목록은 서비스 승인 참조에서 Amazon S3에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

일반적인 Amazon S3 요청에 대한 **`aclRequired`** 값

인증을 위해 ACL이 필요한 Amazon S3 요청을 식별하려면, Amazon S3 서버 액세스 로그 또는 `aclRequired` 내의 AWS CloudTrail 값을 사용하면 됩니다. CloudTrail 또는 Amazon S3 서버 액세스 로그에 표시되는 `aclRequired` 값은 호출된 작업과 요청자, 객체 소유자 및 버킷 소유자에 대한 특정 정보에 따라 달라집니다. ACL이 필요하지 않았거나, `bucket-owner-full-control` 미리 제공

된 ACL을 설정하고 있거나, 버킷 정책에 의해 요청이 허용되는 경우, Amazon S3 서버 액세스 로그에서 `aclRequired` 값 문자열은 "-"이며 CloudTrail에는 표시되지 않습니다.

다음 테이블에는 다양한 Amazon S3 API 작업에 대해 CloudTrail 또는 Amazon S3 서버 액세스 로그에서 예상되는 `aclRequired` 값이 나열되어 있습니다. 이 정보를 사용하여 어떤 Amazon S3 작업이 권한 부여를 위해 ACL에 의존하는지 파악할 수 있습니다. 다음 테이블에서 A, B, C는 요청자, 객체 소유자, 버킷 소유자와 연관된 서로 다른 계정을 나타냅니다. 별표(*)가 있는 항목은 계정 A, B 또는 C 중 하나를 나타냅니다.

> **Note**
>
> 다음 테이블의 `PutObject` 작업은 달리 명시되지 않는 한, ACL이 `bucket-owner-full-control` ACL이 아닌 한, ACL을 설정하지 않은 요청을 나타냅니다. `aclRequired`에 대한 null 값은 AWS CloudTrail 로그에 `aclRequired`가 없음을 나타냅니다.

다음 테이블은 CloudTrail의 `aclRequired` 값을 보여줍니다.

작업 이름	요청자	객체 소유자	버킷 소유자	버킷 정책으로 액세스 권한 부여	**aclRequired** 값	이유
`GetObject`	A	A	A	[Yes] 또는 [No]	null	동일한 계정 액세스
`GetObject`	A	B	A	[Yes] 또는 [No]	null	버킷 소유자에게 적용된 동일 계정 액세스
`GetObject`	A	A	B	예	null	버킷 정책에 의해 부여된 크로스 계정 액세스

작업 이름	요청자	객체 소유자	버킷 소유자	버킷 정책으로 액세스 권한 부여	**aclRequired** 값	이유
GetObject	A	A	B	아니요	예	ACL에 의존하는 크로스 계정 액세스
GetObject	A	A	B	예	null	버킷 정책에 의해 부여된 크로스 계정 액세스
GetObject	A	B	B	아니요	예	ACL에 의존하는 크로스 계정 액세스
GetObject	A	B	C	예	null	버킷 정책에 의해 부여된 크로스 계정 액세스
GetObject	A	B	C	아니요	예	ACL에 의존하는 크로스 계정 액세스
PutObject	A	해당 사항 없음	A	[Yes] 또는 [No]	null	동일한 계정 액세스

작업 이름	요청자	객체 소유자	버킷 소유자	버킷 정책으로 액세스 권한 부여	**aclRequired** 값	이유
PutObject	A	해당 사항 없음	B	예	null	버킷 정책에 의해 부여된 크로스 계정 액세스
PutObject	A	해당 사항 없음	B	아니요	예	ACL에 의존하는 크로스 계정 액세스
ACL 포함된 PutObject (bucket-owner-full-control 제외)	*	해당 사항 없음	*	[Yes] 또는 [No]	예	요청 권한 부여 ACL
ListObjects	A	해당 사항 없음	A	[Yes] 또는 [No]	null	동일한 계정 액세스
ListObjects	A	해당 사항 없음	B	예	null	버킷 정책에 의해 부여된 크로스 계정 액세스
ListObjects	A	해당 사항 없음	B	아니요	예	ACL에 의존하는 크로스 계정 액세스

작업 이름	요청자	객체 소유자	버킷 소유자	버킷 정책으로 액세스 권한 부여	**aclRequired** 값	이유
DeleteObject	A	해당 사항 없음	A	[Yes] 또는 [No]	null	동일한 계정 액세스
DeleteObject	A	해당 사항 없음	B	예	null	버킷 정책에 의해 부여된 크로스 계정 액세스
DeleteObject	A	해당 사항 없음	B	아니요	예	ACL에 의존하는 크로스 계정 액세스
PutObjectAcl	*	*	*	[Yes] 또는 [No]	예	요청 권한 부여 ACL
PutBucketAcl	*	해당 사항 없음	*	[Yes] 또는 [No]	예	요청 권한 부여 ACL

> **Note**
>
> 다음 테이블의 REST.PUT.OBJECT 작업은 달리 명시되지 않는 한, ACL이 bucket-owner-full-control ACL이 아닌 한, ACL을 설정하지 않은 요청을 나타냅니다. aclRequired 값 문자열이 "-"인 경우 Amazon S3 서버 액세스 로그에서 null 값을 나타냅니다.

다음 테이블은 Amazon S3 서버 액세스 로그의 aclRequired 값을 보여줍니다.

작업 이름	요청자	객체 소유자	버킷 소유자	버킷 정책으로 액세스 권한 부여	**aclRequired** 값	이유
REST.GET.OBJECT	A	A	A	[Yes] 또는 [No]	-	동일한 계정 액세스
REST.GET.OBJECT	A	B	A	[Yes] 또는 [No]	-	버킷 소유자에게 적용된 동일 계정 액세스
REST.GET.OBJECT	A	A	B	예	-	버킷 정책에 의해 부여된 크로스 계정 액세스
REST.GET.OBJECT	A	A	B	아니요	예	ACL에 의존하는 크로스 계정 액세스
REST.GET.OBJECT	A	B	B	예	-	버킷 정책에 의해 부여된 크로스 계정 액세스
REST.GET.OBJECT	A	B	B	아니요	예	ACL에 의존하는 크로스 계정 액세스
REST.GET.OBJECT	A	B	C	예	-	버킷 정책에 의해 부

작업 이름	요청자	객체 소유자	버킷 소유자	버킷 정책으로 액세스 권한 부여	**aclRequired** 값	이유
						여된 크로스 계정 액세스
REST.GET.OBJECT	A	B	C	아니요	예	ACL에 의존하는 크로스 계정 액세스
REST.PUT.OBJECT	A	해당 사항 없음	A	[Yes] 또는 [No]	-	동일한 계정 액세스
REST.PUT.OBJECT	A	해당 사항 없음	B	예	-	버킷 정책에 의해 부여된 크로스 계정 액세스
REST.PUT.OBJECT	A	해당 사항 없음	B	아니요	예	ACL에 의존하는 크로스 계정 액세스
ACL 포함된 REST.PUT.OBJECT (bucket-owner-full-control 제외)	*	해당 사항 없음	*	[Yes] 또는 [No]	예	요청 권한 부여 ACL
REST.GET.BUCKET	A	해당 사항 없음	A	[Yes] 또는 [No]	-	동일한 계정 액세스

작업 이름	요청자	객체 소유자	버킷 소유자	버킷 정책으로 액세스 권한 부여	**aclRequired** 값	이유
REST.GET.BUCKET	A	해당 사항 없음	B	예	-	버킷 정책에 의해 부여된 크로스 계정 액세스
REST.GET.BUCKET	A	해당 사항 없음	B	아니요	예	ACL에 의존하는 크로스 계정 액세스
REST.DELETE.OBJECT	A	해당 사항 없음	A	[Yes] 또는 [No]	-	동일한 계정 액세스
REST.DELETE.OBJECT	A	해당 사항 없음	B	예	-	버킷 정책에 의해 부여된 크로스 계정 액세스
REST.DELETE.OBJECT	A	해당 사항 없음	B	아니요	예	ACL에 의존하는 크로스 계정 액세스
REST.PUT.ACL	*	*	*	[Yes] 또는 [No]	예	요청 권한 부여 ACL

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_CONTROL을 보유합니다. 또한, ACL

은 이전 섹션에서 다룬 2가지 사전 정의된 Amazon S3 그룹과 정식 사용자 ID로 식별되는 2가지 AWS 계정에 리소스에 대한 권한을 부여하는 방법을 보여 줍니다.

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 사전 정의된 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? 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 가집니다. Amazon EC2는 Amazon S3에서 Amazon Machine Image(AMI) 번들을 `GET`하기 위해 `READ` 액세스 권한을 가져옵니다.
`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.

미리 제공된 ACL	적용 대상	ACL에 추가된 권한
`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 서버 액세스 로깅을 사용한 요청 로깅 단원을 참조하십시오.

> **Note**
>
> 이러한 미리 제공된 ACL은 요청에 하나만 지정할 수 있습니다.

`x-amz-acl` 요청 헤더를 사용하여 요청에 미리 제공된 ACL을 지정합니다. Amazon S3에서 미리 제공된 ACL이 포함된 요청을 수신하면, 사전 정의된 권한을 리소스 ACL에 추가합니다.

ACL 구성

이 섹션에서는 ACL(액세스 제어 목록)을 사용하여 S3 버킷에 대한 액세스 권한을 관리하는 방법을 설명합니다. AWS Management Console, AWS Command Line Interface(CLI), REST API 또는 AWS SDK를 사용하면 리소스 AC에 대한 권한 부여를 추가할 수 있습니다.

버킷 및 객체 권한은 서로 독립적입니다. 객체는 해당 버킷으로부터 권한을 상속하지 않습니다. 예를 들어, 버킷을 만들고 사용자에게 쓰기 액세스 권한을 부여하는 경우 사용자로부터 명시적으로 권한을 부여 받지 않는 한 해당 사용자의 객체에 액세스할 수 없습니다.

다른 AWS 계정 사용자나 미리 정의된 그룹에 권한을 부여할 수 있습니다. 권한을 부여하는 사용자 또는 그룹을 피부여자라고 합니다. 기본적으로 소유자, 즉 버킷을 만든 AWS 계정에는 모든 권한이 있습니다.

사용자 또는 그룹에 부여하는 각 권한에 대해 버킷과 연결된 ACL에 항목이 추가됩니다. ACL은 피부여자와 그에 부여된 권한을 식별하는 권한 부여를 나열합니다.

S3 객체 소유권은 버킷에 업로드되는 객체의 소유권을 제어하고 ACL을 비활성화 또는 활성화하는 데 사용할 수 있는 Amazon S3 버킷 수준 설정입니다. 기본적으로 객체 소유권은 버킷 소유자 적용 설정

으로 설정되며 모든 ACL이 비활성화되어 있습니다. ACL이 비활성화되면 버킷 소유자는 버킷의 모든 객체를 소유하고 액세스 관리 정책을 사용하여 객체에 대한 액세스를 독점적으로 관리합니다.

Amazon S3의 최신 사용 사례 대부분은 더 이상 ACL을 사용할 필요가 없습니다. 각 객체에 대해 액세스를 개별적으로 제어할 필요가 있는 드문 상황을 제외하고는 ACL을 비활성화한 채로 두는 것이 좋습니다. ACL을 비활성화하면 누가 객체를 버킷에 업로드했는지에 관계없이 정책을 사용하여 버킷의 모든 객체에 대한 액세스를 제어할 수 있습니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

> **Important**
>
> 범용 버킷이 S3 객체 소유권에 대해 버킷 소유자 적용 설정을 사용하는 경우 정책을 사용하여 범용 버킷과 버킷의 객체에 대한 액세스 권한을 부여해야 합니다. 버킷 소유자 적용 설정이 활성화된 상태에서 액세스 제어 목록(ACL) 설정 또는 ACL 업데이트 요청은 실패하고 `AccessControlListNotSupported` 오류 코드를 반환합니다. ACL 읽기 요청은 계속 지원됩니다.

> **Warning**
>
> 모든 사람(퍼블릭 액세스) 또는 인증된 사용자 그룹(모든 AWS 인증 사용자) 그룹에는 쓰기 액세스 권한을 부여하지 않는 것이 좋습니다. 이러한 그룹에 쓰기 액세스 권한을 부여할 때의 영향에 대한 자세한 내용은 Amazon S3의 미리 정의된 그룹 섹션을 참조하십시오.

S3 콘솔을 사용하여 버킷에 대한 ACL 권한 설정

콘솔에는 중복된 피부여자에 대해 결합된 액세스 권한이 표시됩니다. ACL의 전체 목록을 보려면 Amazon S3 REST API, AWS CLI 또는 AWS SDK를 사용합니다.

다음 표에서는 Amazon S3 콘솔에서 버킷에 대해 구성할 수 있는 ACL 권한을 보여줍니다.

버킷에 대한 Amazon S3 콘솔 ACL 권한

콘솔 권한	ACL 권한	액세스
객체(Objects) – 나열(List)	READ	피부여자가 버킷에서 객체를 나열할 수 있습니다.

콘솔 권한	ACL 권한	액세스
객체 - 쓰기	WRITE	피부여자가 버킷에서 새 객체를 생성할 수 있습니다. 기존 객체의 버킷 및 객체 소유자는 해당 객체의 삭제 및 덮어쓰기도 허용합니다.
버킷 ACL - 읽기	READ_ACP	피부여자가 버킷 ACL을 읽을 수 있습니다.
버킷 ACL - 쓰기	WRITE_ACP	피부여자가 해당 버킷에 대한 ACL을 쓸 수 있습니다.
모든 사람(퍼블릭 액세스): 객체 - 목록	READ	버킷의 객체에 대한 공용 읽기 액세스 권한을 부여합니다. 모든 사람(퍼블릭 액세스)에 대한 목록 액세스 권한을 부여하면 전 세계 누구나 버킷의 객체에 액세스할 수 있습니다.
모든 사람(퍼블릭 액세스): 버킷 ACL - 읽기	READ_ACP	버킷 ACL에 대한 공용 읽기 액세스 권한을 부여합니다. 모든 사람(퍼블릭 액세스)에 읽기 권한을 부여하면 전 세계 누구나 버킷 ACL에 액세스할 수 있습니다.

ACL 권한에 대한 자세한 내용은 ACL(액세스 제어 목록) 개요 단원을 참조하십시오.

> **Important**
>
> 범용 버킷이 S3 객체 소유권에 대해 버킷 소유자 적용 설정을 사용하는 경우 정책을 사용하여 범용 버킷과 버킷의 객체에 대한 액세스 권한을 부여해야 합니다. 버킷 소유자 적용 설정이 활성화된 상태에서 액세스 제어 목록(ACL) 설정 또는 ACL 업데이트 요청은 실패하고 `AccessControlListNotSupported` 오류 코드를 반환합니다. ACL 읽기 요청은 계속 지원됩니다.

버킷에 대한 ACL 권한 설정

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.

3. 버킷 목록에서 권한을 설정하려는 버킷의 이름을 선택합니다.
4. Permissions를 선택합니다.
5. [액세스 제어 목록(Access control list)]에서 [편집(Edit)]을 선택합니다.

   버킷에 대한 다음과 같은 ACL 권한을 편집할 수 있습니다.

   Objects

   - 나열 - 피부여자는 버킷의 객체를 나열할 수 있습니다.
   - 쓰기 – 피부여자가 버킷에서 새 객체를 생성할 수 있습니다. 기존 객체의 버킷 및 객체 소유자는 해당 객체의 삭제 및 덮어쓰기도 허용합니다.

     S3 콘솔에서는 S3 로그 전송 그룹과 버킷 소유자(AWS 계정)에게만 쓰기 액세스 권한을 부여할 수 있습니다. 다른 피부여자에게는 쓰기 액세스 권한을 부여하지 않는 것이 좋습니다. 그러나 쓰기 액세스 권한을 부여해야 하는 경우 AWS CLI, AWS SDK 또는 REST API를 사용할 수 있습니다.

   Bucket ACL

   - 읽기 – 피부여자는 버킷 ACL을 읽을 수 있습니다.
   - 쓰기 – 피부여자는 해당하는 버킷의 ACL을 쓸 수 있습니다.
6. 버킷 소유자의 권한을 변경하려면 버킷 소유자(AWS 계정) 옆에 있는 다음 ACL 권한을 선택하거나 선택 취소합니다.

   - [객체(Objects) – [나열(List)] 또는 [쓰기(Write)]
   - [버킷 ACL(Bucket ACL)] – [읽기(Read)] 또는 [쓰기(Write)]

   소유자란 AWS Identity and Access Management IAM 사용자가 아닌 AWS 계정 루트 사용자를 지칭합니다. 루트 사용자에 대한 자세한 내용은 IAM 사용 설명서의 AWS 계정 루트 사용자를 참조하세요.
7. 일반 대중(인터넷의 모든 사용자)에게 권한을 부여하거나 권한 부여를 취소하려면 [모든 사용자(Everyone)] 옆에 있는 다음 ACL 권한을 지우거나 선택합니다.

   - [객체(Objects)] – [나열(List)]
   - [버킷 ACL(Bucket ACL) – [읽기(Read)]

> **Warning**
>
> S3 버킷에 모든 사람 그룹 퍼블릭 액세스 권한을 부여할 때 주의하십시오. 이 그룹에 액세스 권한을 부여하면 전 세계 누구나 버킷에 액세스할 수 있습니다. S3 버킷에 대한 퍼블릭 쓰기 액세스 권한을 부여하지 않는 것이 좋습니다.

8. AWS 계정을 가진 모든 사용자에게 권한을 부여하거나 권한 부여를 취소하려면 인증된 사용자 그룹(AWS 계정이 있는 모든 사용자) 옆에 있는 다음 ACL 권한을 선택하거나 선택 취소합니다.

   - [객체(Objects)] – [나열(List)]
   - [버킷 ACL(Bucket ACL) – [읽기(Read)]

9. 서버 액세스 로그를 버킷에 쓸 수 있는 권한을 Amazon S3에 부여하거나 권한 부여를 취소하려면 [S3 로그 전송 그룹(S3 log delivery group)] 다음 ACL 권한을 지우거나 선택합니다.

   - [객체(Objects) – [나열(List)] 또는 [쓰기(Write)]
   - [버킷 ACL(Bucket ACL)] – [읽기(Read)] 또는 [쓰기(Write)]

   버킷이 액세스 로그를 수신할 대상 버킷으로 설정되면 버킷 권한을 통해 로그 전달 그룹에게 버킷에 대한 쓰기 액세스 권한을 허용해야 합니다. 버킷에서 서버 액세스 로깅을 사용 설정하면 로그 수신을 위해 선택한 대상 버킷에서 Amazon S3 콘솔이 쓰기 액세스 권한을 로그 전달 그룹에 부여합니다. 서버 액세스 로깅에 대한 자세한 내용은 Amazon S3 서버 액세스 로깅 사용 설정 단원을 참조하십시오.

10. 다른 AWS 계정에 대한 액세스 권한을 부여하려면 다음을 수행합니다.

    a. [피부여자 추가(Add grantee)]를 선택합니다.

    b. [피부여자(Grantee)] 상자에 다른 AWS 계정의 정식 ID를 입력합니다.

    c. 다음 ACL 권한 중에서 선택합니다.

       - [객체(Objects) – [나열(List)] 또는 [쓰기(Write)]
       - [버킷 ACL(Bucket ACL)] – [읽기(Read)] 또는 [쓰기(Write)]

> **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 설명서에서 역할을 만들어 IAM 사용자에게 권한 위임 단원을 참조하십시오.

11. 다른 AWS 계정에 대한 액세스를 제거하려면 다른 AWS 계정에 대한 액세스에서 제거를 선택합니다.
12. 변경 사항을 저장하려면 변경 사항 저장을 선택합니다.

S3 콘솔을 사용하여 객체에 대한 ACL 권한 설정

콘솔에는 중복된 피부여자에 대해 결합된 액세스 권한이 표시됩니다. ACL의 전체 목록을 보려면 Amazon S3 REST API, AWS CLI 또는 AWS SDK를 사용합니다. 다음 표에서는 Amazon S3 콘솔에서 객체에 대해 구성할 수 있는 ACL 권한을 보여줍니다.

객체에 대한 Amazon S3 콘솔 ACL 권한

콘솔 권한	ACL 권한	액세스
객체 - 읽기	`READ`	피부여자가 객체 데이터와 그 메타데이터를 읽을 수 있습니다.
객체 ACL - 읽기	`READ_ACP`	피부여자가 객체 ACL을 읽을 수 있습니다.
객체 ACL - 쓰기	`WRITE_ACP`	피부여자에게 해당 객체에 대한 ACL 쓰기를 허용합니다

ACL 권한에 대한 자세한 내용은 ACL(액세스 제어 목록) 개요 단원을 참조하십시오.

Important

범용 버킷이 S3 객체 소유권에 대해 버킷 소유자 적용 설정을 사용하는 경우 정책을 사용하여 범용 버킷과 버킷의 객체에 대한 액세스 권한을 부여해야 합니다. 버킷 소유자 적용 설정이 활성화된 상태에서 액세스 제어 목록(ACL) 설정 또는 ACL 업데이트 요청은 실패하고 `AccessControlListNotSupported` 오류 코드를 반환합니다. ACL 읽기 요청은 계속 지원됩니다.

객체에 대한 ACL 권한 설정

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷(Buckets) 목록에서 객체가 포함된 버킷의 이름을 선택합니다.
3. [객체(objects)] 목록에서 권한을 설정하려는 객체의 이름을 선택합니다.
4. Permissions를 선택합니다.
5. ACL(액세스 제어 목록)에서 [편집(Edit)]을 선택합니다.

   객체에 대한 다음과 같은 ACL 권한을 편집할 수 있습니다.

   객체

   - 읽기 – 피부여자는 객체 데이터와 해당하는 메타데이터를 읽을 수 있습니다.

   객체 ACL

   - 읽기 – 피부여자는 객체 ACL을 읽을 수 있습니다.
   - 쓰기 – 피부여자는 해당하는 객체의 ACL을 쓸 수 있습니다. S3 콘솔에서는 버킷 소유자(AWS 계정)에게만 쓰기 액세스 권한을 부여할 수 있습니다. 다른 피부여자에게는 쓰기 액세스 권한을 부여하지 않는 것이 좋습니다. 그러나 쓰기 액세스 권한을 부여해야 하는 경우 AWS CLI, AWS SDK 또는 REST API를 사용할 수 있습니다.
6. 다음에 대한 객체 액세스 권한을 관리할 수 있습니다.

   a. 객체 소유자의 액세스

      소유자란 AWS Identity and Access Management IAM 사용자가 아닌 AWS 계정 루트 사용자를 지칭합니다. 루트 사용자에 대한 자세한 내용은 IAM 사용 설명서의 AWS 계정 루트 사용자를 참조하십시오.

      소유자의 객체 액세스 권한을 변경하려면 객체 소유자 액세스에서 해당 AWS 계정(소유자)를 선택합니다.

      변경하려는 권한에 대한 확인란을 선택한 다음, 저장을 선택합니다.

   b. 다른 AWS 계정에 대한 액세스

      다른 AWS 계정의 AWS 사용자에게 권한을 부여하려면 다른 AWS 계정에 대한 액세스 아래에서 계정 추가를 선택합니다. [ID 입력(Enter an ID)] 필드에 객체 권한을 부여할 AWS 사용자

의 정식 ID를 입력합니다. 정식 ID 찾기에 대한 자세한 내용은 Amazon Web Services 일반 참조의 AWS 계정 식별자를 참조하십시오. 사용자는 최대 99명까지 추가할 수 있습니다.

사용자에게 부여할 권한의 확인란을 선택한 다음, 저장을 선택합니다. 권한에 대한 정보를 표시하려면 도움말 아이콘을 선택합니다.

c. 퍼블릭 액세스

일반 대중(전 세계 모든 사람)에게 객체 액세스 권한을 부여하려면 퍼블릭 액세스에서 모든 사람을 선택합니다. 퍼블릭 액세스 권한을 부여한다는 것은 전 세계 누구든 객체에 액세스할 수 있다는 뜻입니다.

부여하려는 권한에 대한 확인란을 선택한 다음, 저장을 선택합니다.

> **Warning**
>
> - 모든 사람 그룹에 Amazon S3 객체에 대한 익명 액세스 권한을 부여할 때는 주의하십시오. 이 그룹에 대해 액세스 권한을 부여하면 전 세계 누구나 객체에 액세스할 수 있습니다. 모두에게 액세스 권한을 부여해야 하는 경우, Read objects(객체 읽기)에 대한 권한만 부여하는 것이 좋습니다.
> - 모든 사람 그룹에 객체 쓰기 권한을 부여하지 않는 것이 좋습니다. 이렇게 하면 누구든지 객체에 대한 ACL 권한을 덮어쓸 수 있습니다.

AWS SDK 사용

이 섹션에서는 버킷 및 객체에 대한 ACL(액세스 제어 목록) 권한 부여를 구성하는 방법을 보여주는 예제를 제공합니다.

> **Important**
>
> 범용 버킷이 S3 객체 소유권에 대해 버킷 소유자 적용 설정을 사용하는 경우 정책을 사용하여 범용 버킷과 버킷의 객체에 대한 액세스 권한을 부여해야 합니다. 버킷 소유자 적용 설정이 활성화된 상태에서 액세스 제어 목록(ACL) 설정 또는 ACL 업데이트 요청은 실패하고 `AccessControlListNotSupported` 오류 코드를 반환합니다. ACL 읽기 요청은 계속 지원됩니다.

Java

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 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.

Example 버킷을 생성하고, S3 로그 전송 그룹에 권한을 부여하는 미리 준비된 ACL을 지정합니다.

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();
        }
    }
}
```

Example 기존 객체의 ACL 업데이트

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 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();
        }
    }
}
```

.NET

Example 버킷을 생성하고, S3 로그 전송 그룹에 권한을 부여하는 미리 준비된 ACL을 지정합니다.

이 C# 예제에서는 버킷을 생성합니다. 요청 시 이 코드는 로그 전달 그룹 권한을 부여하는 준비된 ACL을 지정하여 버킷에 로그를 기록합니다.

코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

```
using Amazon;
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());
            }
        }
    }
```

```
}
```

Example 기존 객체의 ACL 업데이트

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.

코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

```
using Amazon;
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 API 사용

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 Reference의 다음 단원을 참조하십시오.

- GetBucketAcl
- PutBucketAcl
- GetObjecAcl
- PutObjectAcl
- PutObject
- CreateBucket
- CopyObject
- CreateMultipartUpload

> **Important**
>
> 범용 버킷이 S3 객체 소유권에 대해 버킷 소유자 적용 설정을 사용하는 경우 정책을 사용하여 범용 버킷과 버킷의 객체에 대한 액세스 권한을 부여해야 합니다. 버킷 소유자 적용 설정이 활성화된 상태에서 액세스 제어 목록(ACL) 설정 또는 ACL 업데이트 요청은 실패하고 `AccessControlListNotSupported` 오류 코드를 반환합니다. ACL 읽기 요청은 계속 지원됩니다.

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(액세스 제어 목록) 개요 섹션을 참조하세요.

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 사전 정의된 ACL 집합을 지원합니다. 미리 제공된 각 ACL에는 사전 정의된 피부여자와 권한 집합이 있습니다. 자세한 내용은 미리 제공된 ACL 섹션을 참조하세요.
- 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 제어 목록) 개요 섹션을 참조하세요. 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

AWS CLI 사용

AWS CLI를 사용한 ACL 관리에 대한 자세한 내용은 AWS CLI 명령 참조의 put-bucket-acl을 참조하십시오.

> **Important**
>
> 범용 버킷이 S3 객체 소유권에 대해 버킷 소유자 적용 설정을 사용하는 경우 정책을 사용하여 범용 버킷과 버킷의 객체에 대한 액세스 권한을 부여해야 합니다. 버킷 소유자 적용 설정이 활성화된 상태에서 액세스 제어 목록(ACL) 설정 또는 ACL 업데이트 요청은 실패하고 `AccessControlListNotSupported` 오류 코드를 반환합니다. ACL 읽기 요청은 계속 지원됩니다.

ACL에 대한 정책 예시

버킷 정책의 조건 키를 사용하여 Amazon S3에 대한 액세스를 제어할 수 있습니다.

주제

- 버킷 소유자가 전체 권한을 가져오도록 요구하는 조건으로 s3:PutObject 권한 부여
- x-amz-acl 헤더에 조건부로 s3:PutObject 권한 부여

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:::awsexamplebucket1/*",
      "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:::awsexamplebucket1/*",
            "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:::awsexamplebucket1/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"
                }
            }
        }
    ]
}
```

AWS CLI에서 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 사용 및 설정에 대한 자세한 내용은 Amazon S3 API 참조의 AWS CLI를 사용하여 Amazon S3에서 개발을 참조하세요.

```
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": {
    "StringEquals": {
        "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
```

x-amz-acl 헤더에 조건부로 s3:PutObject 권한 부여

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:::awsexamplebucket1/*"],
            "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 S3에서는 Amazon S3 전용 조건을 포함한 이러한 유형의 의미 체계 오류를 테스트할 수 있습니다. 그러나, IAM 사용자 또는 역할에 대해 정책을 만드는 경우 의미 체계상으로 잘못된 Amazon S3 조건을 포함한다 해도, IAM에서 Amazon S3 조건을 검증할 수 없으므로 오류는 보고되지 않습니다.

Amazon S3 스토리지에 대한 퍼블릭 액세스 차단

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 관리자와 버킷 소유자가 리소스 생성 방식에 관계없이 적용되는 Amazon S3 리소스에 대한 퍼블릭 액세스를 제한하는 중앙 집중식 제어를 쉽게 설정할 수 있습니다.

퍼블릭 블록 액세스 구성에 대한 지침은 퍼블릭 액세스 차단 구성에서 확인하십시오.

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.

Important

ACL(액세스 제어 목록), 액세스 포인트 정책, 버킷 정책 또는 모두를 통해 버킷 및 객체에 퍼블릭 액세스 권한이 부여됩니다. 모든 Amazon S3 액세스 포인트, 버킷 및 객체의 퍼블릭 액세스가 차단되도록 하려면 계정에 대해 퍼블릭 액세스 차단 설정 4개를 모두 켜는 것이 좋습니다. 이러한 설정은 현재와 미래의 모든 버킷 및 액세스 포인트에 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 S3를 사용하여 정적 웹 사이트 호스팅에 설명된 대로 정적 웹 사이트를 호스팅하는 경우), 스토리지 사용 사례에 적합하도록 개별 설정을 사용자 지정할 수 있습니다.

퍼블릭 액세스 차단을 활성화하면 S3 리소스에 직접 연결된 리소스 정책 또는 액세스 제어 목록(ACL)을 통해 퍼블릭 액세스가 부여되지 않도록 하여 리소스를 보호하는 데 도움이 됩니다. 퍼블릭 액세스 차단을 활성화하는 것 외에도 다음 정책을 주의 깊게 검토하여 퍼블릭 액세스를 허용하지 않는지 확인하세요.

- 관련 AWS 보안 주체(예: IAM 역할)에 연결된 자격 증명 기반 정책
- 관련 AWS 리소스(예: AWS Key Management Service(KMS) 키)에 연결된 리소스 기반 정책

Note

- 액세스 포인트, 버킷 및 AWS 계정에 대해서만 퍼블릭 액세스 차단 설정을 사용할 수 있습니다. Amazon S3는 객체별로 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

- 퍼블릭 액세스 차단 설정

- 액세스 포인트에서 퍼블릭 액세스 차단 작업 수행
- "퍼블릭"의 의미
- IAM Access Analyzer for S3를 사용하여 퍼블릭 버킷 검토
- 권한
- 퍼블릭 액세스 차단 구성
- 계정에 대한 퍼블릭 액세스 차단 설정 구성
- S3 버킷에 대한 퍼블릭 액세스 차단 설정 구성

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 액세스 제어 목록(ACL)이 퍼블릭이면 `PutBucketAcl` 및 `PutObjectAcl` 호출이 실패합니다. • 요청에 퍼블릭 ACL이 포함되어 있으면 `PutObject` 호출이 실패합니다. • 이 설정이 계정에 적용되면, 요청에 퍼블릭 ACL이 포함된 경우 `PUT Bucket` 직접 호출이 실패합니다. 이 설정을 `TRUE`로 설정하면 지정된 작업이 실패합니다(REST API, AWS CLI 또는 AWS SDK 등 설정 경로와 상관없음). 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.

이름	설명
	Note 액세스 포인트에 연결된 ACL이 없습니다. 이 설정을 액세스 포인트에 적용하면 기본 버킷에 대한 패스스루 역할을 합니다. 액세스 포인트에 이 설정이 사용 설정된 경우, 버킷에 실제로 이 설정이 사용 설정되어 있는지 여부와 상관없이 액세스 포인트를 통해 이루어진 요청은 기본 버킷에 이 설정이 사용 설정된 것처럼 동작합니다.
`IgnorePublicAcls`	이 옵션을 TRUE로 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 `PutObject` 직접 호출을 허용할 수 있습니다(퍼블릭 ACL이 포함된 `PutObject` 직접 호출을 거부하는 `BlockPublicAcls` 와 반대). 이 설정을 사용 설정하면 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 사용 설정된 경우, 버킷에 실제로 이 설정이 사용 설정되어 있는지 여부와 상관없이 액세스 포인트를 통해 이루어진 요청은 기본 버킷에 이 설정이 사용 설정된 것처럼 동작합니다.

이름	설명
BlockPublicPolicy	버킷에 대해 이 옵션을 TRUE로 설정하면 지정된 버킷 정책이 퍼블릭 액세스를 허용하는 경우 PutBucketPolicy 에 대한 직접 호출을 Amazon S3가 거부합니다. 버킷에 대해 이 옵션을 TRUE로 설정하면 지정된 정책이 퍼블릭 액세스를 허용하는 경우, 버킷의 동일한 계정 액세스 포인트 모두에 PutAccessPointPolicy 에 대한 직접 호출을 Amazon S3가 거부합니다. 액세스 포인트에 대해 이 옵션을 TRUE로 설정하면 지정된 정책(액세스 포인트 또는 기본 버킷)이 퍼블릭 액세스를 허용하는 경우, 액세스 포인트를 통해 이루어지는 PutAccessPointPolicy 및 PutBucketPolicy 에 대한 직접 호출을 Amazon S3가 거부합니다. 이 설정을 사용하면 사용자가 버킷 또는 버킷에 포함된 객체를 공개적으로 공유하지 않고 사용자가 액세스 포인트 및 버킷 정책을 관리할 수 있습니다. 이 설정을 사용 설정해도 기존 액세스 포인트 또는 버킷 정책에는 영향을 주지 않습니다. **Important** 이 설정을 효과적으로 사용하려면 계정 레벨에서 적용하는 것이 좋습니다. 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 사용 중지하도록 허용하는 정책을 삽입할 수 있습니다. 특정 버킷이 아닌 전체 계정에 이 설정이 사용 설정되면 사용자가 이 설정을 사용 중지하기 위해 버킷 정책을 변경한 경우에도 Amazon S3가 퍼블릭 정책을 차단합니다.

이름	설명
RestrictPublicBuckets	이 옵션을 TRUE로 설정하면 퍼블릭 정책이 있는 액세스 포인트 또는 버킷에 대한 액세스가 버킷 소유자 계정 및 액세스 포인트 소유자 계정 내에서 권한 있는 사용자와 AWS 서비스 보안 주체로만 제한됩니다. 이 설정은 AWS 서비스 보안 주체에서 제외한 액세스 포인트 또는 버킷에 대한 모든 교차 계정 액세스를 차단하지만 계정 내 사용자는 계속 액세스 포인트 또는 버킷을 관리할 수 있습니다. 이 설정을 사용 설정하면 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

- GetBucketAcl 및 GetObjectAcl에 대한 직접 호출은 항상 지정된 버킷 또는 객체에 대한 효율적인 권한을 반환합니다. 예를 들어 버킷에 퍼블릭 액세스 권한을 부여하는 ACL이 있고 버킷에 IgnorePublicAcls 설정도 사용 설정되어 있다고 가정하겠습니다. 이 경우 GetBucketAcl은 버킷과 연결된 실제 ACL이 아닌 Amazon S3가 적용하는 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.

액세스 포인트에서 퍼블릭 액세스 차단 작업 수행

액세스 포인트에서 퍼블릭 액세스 차단 작업을 수행하려면 AWS CLI 서비스 s3control을 사용하십시오.

> **Important**
>
> 액세스 포인트를 만든 후에는 액세스 포인트의 퍼블릭 액세스 차단 설정을 변경할 수 없습니다. 액세스 포인트를 생성할 때만 액세스 포인트에 대한 퍼블릭 액세스 차단 설정을 지정할 수 있습니다.

"퍼블릭"의 의미

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 단원을 참조하십시오.

버킷 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 이상에 대한 고정 값(와일드카드 또는 AWS Identity and Access Management 정책 변수가 없는 값)에만 액세스 권한을 부여해야 합니다.

- AWS 보안 주체, 사용자, 역할 또는 서비스 보안 주체(예: `aws:PrincipalOrgID`)
- `aws:SourceIp`를 사용하는 CIDR(Classless Inter-Domain Routing) 블록 집합. CIDR에 대한 자세한 내용은 RFC Editor 웹 사이트에서 RFC 4632를 참조하세요.

  > **Note**
  >
  > IP 범위가 매우 넓은(예: 0.0.0.0/1) `aws:SourceIp` 조건 키를 기반으로 액세스를 부여하는 버킷 정책은 '퍼블릭'으로 평가됩니다. 여기에는 IPv4의 경우 /8, IPv6의 경우 /32보다 광범위한 값이 포함됩니다(RFC1918 프라이빗 범위 제외). 퍼블릭 액세스 차단은 이러한 '퍼블릭' 정책을 거부하고 이러한 '퍼블릭' 정책을 이미 사용하고 있는 버킷에 대한 크로스 계정 액세스를 차단합니다.

- `aws:SourceArn`
- `aws:SourceVpc`
- `aws:SourceVpce`
- `aws:SourceOwner`

- `aws:SourceAccount`
- `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 `123456789012`의 계정 `us-west-2`와 연결된 모든 액세스 포인트에 대한 액세스가 허용됩니다. 이 동작은 액세스 포인트 정책에 따라 다릅니다. 자세한 내용은 액세스 포인트 섹션을 참조하세요.

- `s3:DataAccessPointAccount`

버킷 정책에 대한 자세한 내용은 Amazon S3의 버킷 정책를 참조하십시오.

> **Note**
>
> 다중 값 컨텍스트 키를 사용하는 경우 `ForAllValues` 또는 `ForAnyValue` 집합 연산자를 사용해야 합니다.

Example : 퍼블릭 버킷 정책

이 규칙에서 다음 예제 정책은 퍼블릭으로 간주됩니다.

```
{
  "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"}}
 }
```

Amazon S3가 퍼블릭 액세스 권한과 퍼블릭이 아닌 액세스 권한을 모두 포함하는 버킷 정책을 평가하는 방법

이 예제는 Amazon S3가 퍼블릭 액세스 권한과 퍼블릭이 아닌 액세스 권한을 모두 포함하는 버킷 정책을 어떻게 평가하는지 보여줍니다.

버킷에 고정된 보안 주체 집합에 대한 액세스 권한을 부여하는 정책이 있다고 가정하겠습니다. 이전에 설명한 규칙에 따르면 이 정책은 퍼블릭이 아닙니다. 따라서 `RestrictPublicBuckets` 설정을 사용 설정하면 `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`가 사용 설정되면, Amazon S3는 CloudTrail의 액세스만 허용합니다. 설명문 2가 퍼블릭이 아니더라도 Amazon S3는 "Account-2"에 의한 액세스를 사용 중지합니다. 설명문 3이 전체 정책을 퍼블릭으로 만들기 때문으로, `RestrictPublicBuckets`이 적용됩니다. 결과

적으로 정책이 특정 계정 "Account-2"에 대한 액세스를 위임하더라도 Amazon S3는 교차 계정 액세스를 사용 중지합니다. 그러나 정책에서 설명문 3을 제거하면 정책은 퍼블릭으로 평가되지 않으며 `RestrictPublicBuckets`은 더 이상 적용되지 않습니다. 따라서 `RestrictPublicBuckets`을 사용 설정해 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, `s3:DataAccessPointArn`와 일치하는 `arn:aws:s3:us-west-2:123456789012:accesspoint/*` 값에 액세스 권한을 부여하는 버킷 정책은 공개로 간주되지 않습니다. 그러나 액세스 포인트 정책의 동일한 명령문은 액세스 포인트를 퍼블릭으로 렌더링합니다.

IAM Access Analyzer for S3를 사용하여 퍼블릭 버킷 검토

IAM Access Analyzer for S3를 사용하여 퍼블릭 액세스 권한을 부여하는 버킷 ACL, 버킷 정책 또는 액세스 포인트 정책이 있는 버킷을 검토할 수 있습니다. IAM Access Analyzer for S3는 인터넷상의 모든 사용자 또는 조직 외부의 AWS 계정을 포함한 다른 AWS 계정에 대한 액세스를 허용하도록 구성된 S3 버킷에 대한 알림을 제공합니다. 각 퍼블릭 버킷 또는 공유 버킷에 대해 퍼블릭 액세스 또는 공유 액세스의 수준과 소스를 보고하는 결과가 수신됩니다.

IAM 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 경우지만 IAM Access Analyzer for S3 및 Amazon S3 Block Public Access 평가는 버킷이 퍼블릭인지에 따라 다를 수 있습니다. 이 동작은 Amazon S3 Block Public Access가 퍼블릭 액세스 평가 외에도 작업의 존재 여부에 대한 검증을 수행하기 때문에 발생합니다. 버킷 정책에 Amazon S3에서 지원하지 않는 작업에 대한 퍼블릭 액세스를 허용하는 `Action` 문이 포함되어 있다고 가정해 보겠습니다(예: `s3:NotASupportedAction`). 이 경우 Amazon S3 Block Public Access는 버킷을 퍼블릭으로 평가

합니다. 이러한 문은 나중에 작업이 지원될 경우 버킷을 퍼블릭으로 만들 수 있기 때문입니다. Amazon S3 Block Public Access와 IAM Access Analyzer for S3의 평가가 다른 경우 버킷 정책을 검토하고 지원되지 않는 작업을 제거하는 것이 좋습니다.

IAM Access Analyzer for S3에 대한 자세한 내용은 IAM Access Analyzer for S3를 사용하여 버킷 액세스 검토 섹션을 참조하십시오.

권한

Amazon S3 퍼블릭 액세스 차단 기능을 사용하려면 다음 권한이 있어야 합니다.

작업	필수 권한
버킷 정책 상태 GET	`s3:GetBucketPolicyStatus`
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`
액세스 포인트 퍼블릭 액세스 차단 설정 PUT	`s3:CreateAccessPoint`

> **Note**
>
> DELETE 작업에는 PUT 작업과 동일한 권한이 필요합니다. DELETE 작업에 대한 별도의 권한은 없습니다.

퍼블릭 액세스 차단 구성

AWS 계정, Amazon S3 버킷에 대한 퍼블릭 액세스 차단을 구성하는 방법과 관련된 자세한 내용은 다음 주제를 참조하세요.

- 계정에 대한 퍼블릭 액세스 차단 설정 구성
- S3 버킷에 대한 퍼블릭 액세스 차단 설정 구성
- 액세스 포인트에서 퍼블릭 액세스 차단 작업 수행

계정에 대한 퍼블릭 액세스 차단 설정 구성

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.

자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하세요.

> **Note**
>
> 계정 수준 설정은 개별 객체의 설정을 덮어씁니다. 퍼블릭 액세스를 차단하도록 계정을 구성하면 계정 내 개별 객체에 대한 모든 퍼블릭 액세스 설정을 덮어씁니다.

S3 콘솔, AWS CLI, AWS SDK 및 REST API를 사용하여 계정의 모든 버킷에 대한 퍼블릭 액세스 차단 설정을 구성할 수 있습니다. 자세한 내용은 아래의 섹션을 참조하세요.

버킷에 대한 퍼블릭 액세스 차단 설정을 구성하려면 S3 버킷에 대한 퍼블릭 액세스 차단 설정 구성 단원을 참조하십시오. 액세스 포인트에 대한 자세한 내용은 액세스 포인트에서 퍼블릭 액세스 차단 작업 수행 단원을 참조하십시오.

S3 콘솔 사용

Amazon S3 퍼블릭 액세스 차단은 S3 버킷 내에서 데이터에 대한 퍼블릭 액세스를 허용하는 모든 설정의 적용을 차단합니다. 이 섹션에서는 AWS 계정의 모든 S3 버킷에 대한 퍼블릭 액세스 차단 설정을 편집하는 방법에 대해 설명합니다. 퍼블릭 액세스 차단에 대한 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하십시오.

AWS 계정의 모든 S3 버킷에 대한 퍼블릭 액세스 차단 설정 편집

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 이 계정에 대한 퍼블릭 액세스 차단 설정을 선택합니다.
3. AWS 계정의 모든 버킷에 대한 퍼블릭 액세스 차단 설정을 변경하려면 [편집(Edit)]을 선택합니다.

4. 변경할 설정을 선택한 다음 변경 사항 저장(Save changes)을 선택합니다.
5. 확인 메시지가 표시되면 **confirm**을 입력합니다. 그런 다음확인을 선택해 변경 사항을 저장합니다.

AWS CLI 사용

AWS CLI를 통해 Amazon S3 퍼블릭 액세스 차단을 사용할 수 있습니다. AWS CLI의 설정 및 사용에 대한 자세한 내용은 AWS Command Line Interface란? 단원을 참조하십시오.

Account

계정에서 퍼블릭 액세스 차단 작업을 수행하려면 AWS CLI 서비스 `s3control`을 사용하십시오. 이 서비스를 사용하는 계정 수준 작업은 다음과 같습니다.

- `PutPublicAccessBlock` (계정의 경우)
- `GetPublicAccessBlock` (계정의 경우)
- `DeletePublicAccessBlock` (계정의 경우)

추가 정보 및 예제는 AWS CLI 참조의 put-public-access-block을 참조하세요.

AWS SDK 사용

Java

다음 예제에서는 Amazon S3 퍼블릭 액세스 차단을 AWS SDK for Java와 함께 사용하여 Amazon S3 계정에 퍼블릭 액세스 차단 구성을 설정하는 방법을 보여 줍니다.

```
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.

Other SDKs

다른 AWS SDK 사용에 대한 정보는 Amazon S3 API 참조의 AWS SDK를 사용하여 Amazon S3에서 개발을 참조하세요.

REST API 사용

REST API를 통한 Amazon S3 퍼블릭 액세스 차단 사용에 대한 자세한 내용은 Amazon Simple Storage Service API Reference의 다음 주제를 참조하십시오.

- 계정 수준 작업
  - PutPublicAccessBlock
  - GetPublicAccessBlock
  - DeletePublicAccessBlock

S3 버킷에 대한 퍼블릭 액세스 차단 설정 구성

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.

자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하세요.

S3 콘솔, AWS CLI, AWS SDK 및 REST API를 사용하여 하나 이상의 버킷에 퍼블릭 액세스 권한을 부여할 수 있습니다. 이미 퍼블릭 상태인 버킷에 대한 퍼블릭 액세스를 차단할 수도 있습니다. 자세한 내용은 아래의 섹션을 참조하십시오.

계정의 모든 버킷에 대한 퍼블릭 액세스 차단 설정을 구성하려면 계정에 대한 퍼블릭 액세스 차단 설정 구성 섹션을 참조하십시오. 액세스 포인트에 대한 퍼블릭 액세스 차단 구성에 대한 자세한 내용은 액세스 포인트에서 퍼블릭 액세스 차단 작업 수행 단원을 참조하십시오.

S3 콘솔 사용

Amazon S3 퍼블릭 액세스 차단은 S3 버킷 내에서 데이터에 대한 퍼블릭 액세스를 허용하는 모든 설정의 적용을 차단합니다. 이 섹션에서는 하나 이상의 S3 버킷에 대한 퍼블릭 액세스 차단 설정을 편집하는 방법에 대해 설명합니다. AWS CLI, AWS SDK 및 Amazon S3 REST API를 사용하여 퍼블릭 액세스를 차단하는 방법에 대한 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하십시오.

버킷 목록의 IAM Access Analyzer 열에서 버킷에 공개적으로 액세스할 수 있는지 확인할 수 있습니다. 자세한 내용은 IAM Access Analyzer for S3를 사용하여 버킷 액세스 검토 섹션을 참조하세요.

버킷 및 퍼블릭 액세스 설정을 나열할 때 `Error`가 표시되면 필요한 권한이 없을 수 있습니다. 사용자 또는 역할 정책에 다음 권한이 추가되었는지 확인합니다.

```
s3:GetAccountPublicAccessBlock
s3:GetBucketPublicAccessBlock
s3:GetBucketPolicyStatus
s3:GetBucketLocation
s3:GetBucketAcl
s3:ListAccessPoints
s3:ListAllMyBuckets
```

드문 경우지만 AWS 리전 중단으로 인해 요청이 실패할 수도 있습니다.

단일 S3 버킷의 Amazon S3 퍼블릭 액세스 차단 설정 편집

단일 S3 버킷에 대한 퍼블릭 액세스 설정을 변경해야 하는 경우 다음 단계를 따르십시오.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷 이름 목록에서 원하는 버킷의 이름을 선택합니다.
3. Permissions를 선택합니다.
4. 퍼블릭 액세스 차단(버킷 설정) 옆의 편집을 선택하여 버킷의 퍼블릭 액세스 설정을 변경합니다. 네 가지 Amazon S3 퍼블릭 액세스 차단 설정에 대한 자세한 내용은 퍼블릭 액세스 차단 설정 섹션을 참조하십시오.
5. 설정 중 하나를 선택한 다음 변경 내용 저장을 선택합니다.
6. 확인 메시지가 표시되면 **`confirm`**을 입력합니다. 그런 다음확인을 선택해 변경 사항을 저장합니다.

버킷을 만들 때 Amazon S3 Block Public Access 설정을 변경할 수도 있습니다. 자세한 내용은 범용 버킷 생성 섹션을 참조하세요.

AWS CLI 사용

버킷에서 퍼블릭 액세스를 차단하거나 퍼블릭 액세스 블록을 삭제하려면 AWS CLI 서비스 `s3api`를 사용하십시오. 이 서비스를 사용하는 버킷 수준 작업은 다음과 같습니다.

- `PutPublicAccessBlock` (버킷의 경우)
- `GetPublicAccessBlock` (버킷의 경우)
- `DeletePublicAccessBlock` (버킷의 경우)
- `GetBucketPolicyStatus`

자세한 내용과 예제는 AWS CLI 참조의 put-public-access-block 섹션을 참조하십시오.

AWS SDK 사용

Java

```
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.

Other SDKs

다른 AWS SDK 사용에 대한 정보는 Amazon S3 API 참조의 AWS SDK를 사용하여 Amazon S3에서 개발을 참조하세요.

REST API 사용

REST API를 통한 Amazon S3 퍼블릭 액세스 차단 사용에 대한 자세한 내용은 Amazon Simple Storage Service API Reference의 다음 주제를 참조하십시오.

- 버킷 수준 작업
  - PutPublicAccessBlock
  - GetPublicAccessBlock
  - DeletePublicAccessBlock
  - GetBucketPolicyStatus

IAM Access Analyzer for S3를 사용하여 버킷 액세스 검토

IAM Access Analyzer for S3는 인터넷상의 모든 사용자 또는 조직 외부의 AWS 계정을 포함한 다른 AWS 계정에 대한 액세스를 허용하도록 구성된 S3 버킷에 대한 S3 알림을 제공합니다. 각 퍼블릭 버킷 또는 공유 버킷에 대해 퍼블릭 액세스 또는 공유 액세스의 수준과 원본을 보고하는 결과가 수신됩니다. 예를 들어 IAM Access Analyzer for S3는 버킷에 버킷 액세스 제어 목록(ACL), 버킷 정책, 다중 리전 액세스 포인트 정책 또는 액세스 포인트 정책을 통해 제공된 읽기 또는 쓰기 권한이 있음을 표시할 수 있습니다. 이러한 내용을 바탕으로 즉각적이고 정확한 시정 조치를 취하여 버킷 액세스를 원하는 대로 복원할 수 있습니다.

IAM Access Analyzer for S3에서 위험 버킷을 검토할 때 클릭 한 번으로 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 확인하세요. 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하세요.

또한 버킷 수준 권한 설정으로 드릴다운하여 세부적인 액세스 수준을 구성할 수 있습니다. 정적 웹 사이트 호스팅, 공개 다운로드 또는 교차 계정 공유와 같이 퍼블릭 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.

Amazon S3 콘솔에서는 IAM Access Analyzer for S3를 추가 비용 없이 사용할 수 있습니다. IAM Access Analyzer for S3는 AWS Identity and Access Management(IAM) IAM Access Analyzer를 통해 제공됩니다. Amazon S3 콘솔에서 IAM Access Analyzer for S3를 사용하려면 IAM 콘솔을 방문하여 리전별로 IAM Access Analyzer를 활성화해야 합니다.

IAM Access Analyzer에 대한 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer란 무엇입니까?를 참조하세요. IAM Access Analyzer for S3에 관한 자세한 내용은 다음 섹션들을 검토하십시오.

Important

- IAM Access Analyzer for S3에는 계정 수준 분석기가 필요합니다. IAM Access Analyzer for S3를 사용하려면 IAM Access Analyzer를 방문하여 신뢰 영역으로서의 계정이 있는 분석기를 생성해야 합니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer 활성화를 참조하세요.
- IAM Access Analyzer for S3는 크로스 계정 액세스 포인트에 연결된 액세스 포인트 정책을 분석하지 않습니다. 액세스 포인트와 그 정책이 신뢰 영역, 즉 계정을 벗어났기 때문에 이러한 동작이 발생됩니다. `RestrictPublicBuckets` 블록 퍼블릭 액세스 설정을 버킷 또는 계정에 적용하지 않으면, 크로스 계정 액세스 포인트에 대한 액세스를 위임하는 버킷이 Buckets with public access(퍼블릭 액세스가 있는 버킷)에 나열됩니다. `RestrictPublicBuckets` 블록 퍼블릭 액세스 설정을 적용하면 Buckets with access from other AWS 계정 — including third-party AWS 계정(제3자 AWS 계정 포함 다른 AWS 계정에서의 액세스가 있는 버킷)에 나열됩니다.
- 버킷 정책 또는 버킷 ACL이 추가되거나 수정되면 IAM Access Analyzer는 30분 이내에 변경 사항을 기반으로 결과를 생성하고 업데이트합니다. 계정 수준 퍼블릭 액세스 차단 설정과 관련된 결과는 설정을 변경한 후 최대 6시간 동안 생성되거나 업데이트되지 않을 수 있습니다. 다중 리전 액세스 포인트와 관련된 검색 결과는 다중 리전 액세스 포인트가 생성, 삭제되거나 해당 정책을 변경한 후 최대 6시간 동안 생성 또는 업데이트되지 않을 수 있습니다.

주제

- IAM Access Analyzer for S3는 어떤 정보를 제공합니까?
- IAM Access Analyzer for S3 활성화
- 모든 퍼블릭 액세스 차단
- 버킷 액세스 검토 및 변경
- 버킷 결과 보관

- 보관된 버킷 결과 활성화
- 결과 세부 정보 보기
- IAM Access Analyzer for S3 보고서 다운로드

IAM Access Analyzer for S3는 어떤 정보를 제공합니까?

IAM Access Analyzer for S3에서는 AWS 계정 외부에서 액세스할 수 있는 버킷에 대한 결과를 제공합니다. 퍼블릭 액세스가 가능한 버킷 아래에 나열된 버킷은 인터넷상의 모든 사용자가 액세스할 수 있습니다. IAM Access Analyzer for S3가 퍼블릭 버킷을 식별하면 페이지 상단에 리전의 퍼블릭 버킷 수를 보여주는 경고가 표시됩니다. 서드 파티 AWS 계정을 포함한 다른 AWS 계정에서 액세스할 수 있는 버킷 아래에 나열된 버킷은 조직 외부의 계정을 비롯한 다른 AWS 계정과 조건부로 공유됩니다.

각 버킷에서 IAM Access Analyzer for S3는 다음과 같은 정보를 제공합니다.

- Bucket name
- Access Analyzer에 의해 검색됨 - IAM Access Analyzer for S3가 퍼블릭 버킷 액세스 또는 공유 버킷 액세스를 검색한 경우
- 공유 방식 - 버킷 정책, 버킷 ACL, 다중 리전 액세스 포인트 정책 또는 액세스 포인트 정책을 통해 버킷이 공유되는 방식입니다. 다중 리전 액세스 포인트 및 크로스 계정 액세스 포인트는 액세스 포인트 아래에 반영됩니다. 버킷은 정책과 ACL을 통해 공유할 수 있습니다. 버킷 액세스에 대한 소스를 찾고 검토하려는 경우 이 열의 정보를 즉각적이고 정확한 시정 조치를 취하기 위한 시작점으로 사용할 수 있습니다.
- 상태 - 버킷 결과의 상태입니다. IAM Access Analyzer for S3는 모든 퍼블릭 버킷 및 공유 버킷에 대한 결과를 표시합니다.
  - 활성 - 결과가 검토되지 않았습니다.
  - 보관됨 - 결과가 의도한 대로 검토 및 확인되었습니다.
  - 모두 - 조직 외부의 AWS 계정을 포함한 다른 AWS 계정과 공유되는 버킷이나 퍼블릭 버킷에 대한 모든 결과입니다.
- 액세스 수준 - 다음과 같이 버킷에 부여된 액세스 권한입니다.
  - 목록 - 리소스를 나열합니다.
  - 읽기 - 리소스 콘텐츠 및 속성을 읽기만 하고 편집하지 않습니다.
  - 쓰기 -리소스를 생성, 삭제 또는 수정합니다.
  - 권한 - 리소스 권한을 부여하거나 수정합니다.
  - 태그 지정 - 리소스와 연결된 태그를 업데이트합니다.

IAM Access Analyzer for S3 활성화

IAM Access Analyzer for S3를 사용하려면 다음과 같은 사전 필수 단계들을 완료해야 합니다.

1. 필수 권한을 부여하십시오.

   자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer를 사용하는 데 필요한 권한을 참조하십시오.
2. IAM을 방문하여 IAM Access Analyzer를 사용할 각 리전에 대한 계정 수준 분석기를 만드십시오.

   IAM Access Analyzer for S3에는 계정 수준 분석기가 필요합니다. IAM Access Analyzer for S3를 사용하려면 신뢰 영역으로서의 계정이 있는 분석기를 생성해야 합니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer 활성화를 참조하십시오.

모든 퍼블릭 액세스 차단

클릭 한 번으로 버킷에 대한 모든 액세스를 차단하려는 경우 IAM Access Analyzer for S3에서 모든 퍼블릭 액세스 차단 버튼을 사용할 수 있습니다. 버킷에 대한 모든 퍼블릭 액세스를 차단하면 퍼블릭 액세스 권한이 부여되지 않습니다. 확인된 특정 사용 사례를 지원하기 위해 퍼블릭 액세스가 필요하지 않은 경우 버킷에 대한 모든 공개 액세스를 차단하는 것이 좋습니다. 모든 퍼블릭 액세스를 차단하기 전에 애플리케이션이 퍼블릭 액세스 없이 계속 올바르게 작동하는지 확인하십시오.

버킷에 대한 모든 퍼블릭 액세스를 차단하지 않으려면 Amazon S3 콘솔에서 퍼블릭 액세스 차단 설정을 편집하여 버킷에 대한 세분화된 액세스 수준을 구성할 수 있습니다. 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하세요.

드문 경우지만 IAM Access Analyzer for S3 및 Amazon S3 Block Public Access 평가는 버킷이 퍼블릭인지에 따라 다를 수 있습니다. 이 동작은 Amazon S3 Block Public Access가 퍼블릭 액세스 평가 외에도 작업의 존재 여부에 대한 검증을 수행하기 때문에 발생합니다. 버킷 정책에 Amazon S3에서 지원하지 않는 작업에 대한 퍼블릭 액세스를 허용하는 `Action` 문이 포함되어 있다고 가정해 보겠습니다(예: `s3:NotASupportedAction`). 이 경우 Amazon S3 Block Public Access는 버킷을 퍼블릭으로 평가합니다. 이러한 문은 나중에 작업이 지원될 경우 버킷을 퍼블릭으로 만들 수 있기 때문입니다. Amazon S3 Block Public Access와 IAM Access Analyzer for S3의 평가가 다른 경우 버킷 정책을 검토하고 지원되지 않는 작업을 제거하는 것이 좋습니다.

IAM Access Analyzer for S3를 사용하여 버킷에 대한 모든 퍼블릭 액세스 차단

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.

2. 왼쪽의 탐색 창의 대시보드, 아래에서 S3에 대한 액세스 분석기를 선택합니다.
3. IAM Access Analyzer for S3에서 버킷을 선택합니다.
4. 모든 퍼블릭 액세스 차단을 선택합니다.
5. 버킷에 대한 모든 퍼블릭 액세스를 차단할 것인지 확인하려면 모든 퍼블릭 액세스 차단(버킷 설정)에 **`confirm`**을 입력합니다.

   Amazon S3는 버킷에 대한 모든 퍼블릭 액세스를 차단합니다. 버킷 결과의 상태가 해결됨으로 업데이트되며 IAM Access Analyzer for S3 목록에서 버킷이 사라집니다. 해결된 버킷을 검토하려면 IAM 콘솔에서 IAM Access Analyzer를 엽니다.

버킷 액세스 검토 및 변경

조직 외부의 계정을 포함한 퍼블릭 또는 다른 AWS 계정 계정에 대한 액세스 권한을 부여하지 않으려는 경우 버킷 ACL, 버킷 정책, 다중 리전 액세스 포인트 정책 또는 액세스 포인트 정책을 수정하여 버킷에 대한 액세스 권한을 제거할 수 있습니다. Shared through(공유) 열에는 버킷 정책, 버킷 ACL 및/또는 액세스 포인트 정책 등 모든 버킷 액세스 소스가 표시됩니다. 다중 리전 액세스 포인트 및 크로스 계정 액세스 포인트는 액세스 포인트 아래에 반영됩니다.

버킷 정책, 버킷 ACL, 다중 리전 액세스 포인트 또는 액세스 포인트 정책 검토 및 변경

1. https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 탐색 창에서 S3에 대한 액세스 분석기를 선택합니다.
3. 버킷 정책, 버킷 ACL, 다중 리전 액세스 포인트 정책 또는 액세스 포인트 정책을 통해 퍼블릭 액세스 또는 공유 액세스가 부여되는지 확인하려면 공유 방법(Shared through) 열을 참조하십시오.
4. 버킷(Buckets)에서 변경 또는 검토하려는 버킷 정책, 버킷 ACL, 다중 리전 액세스 포인트 정책 또는 액세스 포인트 정책이 있는 버킷의 이름을 선택합니다.
5. 버킷 ACL을 변경하거나 보려면 다음과 같이 하십시오.

   a. Permissions를 선택합니다.

   b. [Access Control List]를 선택합니다.

   c. 버킷 ACL을 검토하고 필요에 따라 변경합니다.

      자세한 정보는 ACL 구성을 참조하세요.
6. 버킷 정책을 변경하거나 검토하려면 다음과 같이 하십시오.

   a. Permissions를 선택합니다.

b. [Bucket Policy]를 선택합니다.

c. 필요에 따라 버킷 정책을 검토하거나 변경합니다.

자세한 정보는 Amazon S3 콘솔을 사용하여 버킷 정책 추가을 참조하세요.

7. 다중 리전 액세스 포인트 정책을 검토하거나 변경하려면 다음을 수행하십시오.

a. 다중 리전 액세스 포인트(Multi-Region Access Point)를 선택합니다.

b. 다중 리전 액세스 포인트 이름을 선택합니다.

c. 필요에 따라 다중 리전 액세스 포인트 정책을 검토하거나 변경합니다.

자세한 정보는 권한을 참조하세요.

8. 액세스 포인트 정책을 검토하거나 변경하려면 다음과 같이 하십시오.

a. Access points(액세스 포인트)를 선택합니다.

b. 액세스 포인트 이름을 선택합니다.

c. 필요에 따라 액세스 권한을 검토하거나 변경합니다.

자세한 내용은 범용 버킷의 Amazon S3 Access Points 관리 섹션을 참조하세요.

버킷 ACL, 버킷 정책 또는 액세스 포인트 정책을 편집하거나 제거하여 퍼블릭 또는 공유 액세스를 제거하면 버킷 결과의 상태가 해결됨으로 업데이트됩니다. 해결된 버킷 결과는 IAM Access Analyzer for S3 목록에서 사라지지만 IAM Access Analyzer에서 볼 수 있습니다.

버킷 결과 보관

버킷이 특정 사용 사례(예: 정적 웹 사이트, 공개 다운로드 또는 교차 계정 공유)를 지원하기 위해 조직 외부의 계정을 포함하여 퍼블릭 계정 또는 다른 AWS 계정에 대한 액세스 권한을 부여하는 경우 버킷에 대한 결과를 보관할 수 있습니다. 버킷 결과를 보관할 때 버킷을 퍼블릭 상태로 유지할 것인지 공유 상태로 유지할 것인지 확인하고 기록할 수 있습니다. 보관된 버킷 결과는 IAM Access Analyzer for S3 목록에 남아 있으므로 어떤 버킷이 퍼블릭 버킷인지 또는 공유 버킷인지 항상 알 수 있습니다.

IAM Access Analyzer for S3에 버킷 결과 보관

1. https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 탐색 창에서 S3에 대한 액세스 분석기를 선택합니다.
3. IAM Access Analyzer for S3에서 활성 버킷을 선택합니다.

4. 조직 외부의 계정을 포함하여 퍼블릭 계정 또는 다른 AWS 계정에서 이 버킷에 액세스하도록 할 것인지 확인하려면 [아카이브(Archive)]를 선택합니다.
5. **confirm**을 입력하고 아카이브를 선택합니다.

보관된 버킷 결과 활성화

결과를 보관한 후에는 언제든지 다시 방문하여 상태를 다시 활성으로 변경할 수 있습니다. 이는 버킷에 다른 검토가 필요함을 나타냅니다.

IAM Access Analyzer for S3에서 보관된 버킷 결과 활성화

1. https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 탐색 창에서 S3에 대한 액세스 분석기를 선택합니다.
3. 보관된 버킷 결과를 선택합니다.
4. 활성으로 표시를 선택합니다.

결과 세부 정보 보기

버킷에 대한 자세한 정보가 필요한 경우 IAM 콘솔의 IAM Access Analyzer에서 버킷 결과 세부 정보를 열 수 있습니다.

IAM Access Analyzer for S3에서 결과 세부 정보 보기

1. https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 탐색 창에서 S3에 대한 액세스 분석기를 선택합니다.
3. IAM Access Analyzer for S3에서 버킷을 선택합니다.
4. View details(세부 정보 보기)를 선택합니다.

   결과 세부 정보는 IAM 콘솔의 IAM Access Analyzer에서 열립니다.

IAM Access Analyzer for S3 보고서 다운로드

버킷 결과를 감사 목적으로 사용할 수 있는 CSV 보고서로 다운로드할 수 있습니다. 보고서에는 Amazon S3 콘솔의 IAM Access Analyzer for S3에 표시되는 것과 동일한 정보가 포함됩니다.

보고서 다운로드

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽의 탐색 창에서 S3에 대한 액세스 분석기를 선택합니다.
3. 리전 필터에서 리전을 선택합니다.

   IAM Access Analyzer for S3가 업데이트되어 선택한 리전의 버킷을 표시합니다.
4. 보고서 다운로드를 선택합니다.

   CSV 보고서가 생성되어 컴퓨터에 저장됩니다.

버킷 소유자 조건을 사용하여 버킷 소유권 확인

Amazon S3 버킷 소유자 조건은 S3 작업에 사용하는 버킷이 예상한 AWS 계정에 속하도록 보장합니다.

대부분의 S3 작업은 특정 S3 버킷에서 읽거나 씁니다. 이러한 작업에는 객체 업로드, 복사 및 다운로드, 버킷 구성 검색 또는 수정, 객체 구성 검색 또는 수정 등이 포함됩니다. 이러한 작업을 수행할 때 요청에 버킷의 이름을 포함하여 사용할 버킷을 지정합니다. 예를 들어 S3에서 객체를 검색하려면 버킷의 이름과 해당 버킷에서 검색할 객체 키를 지정하는 요청을 합니다.

Amazon S3은 버킷 이름을 기반으로 버킷을 식별하기 때문에 요청에 잘못된 버킷 이름을 사용할 경우 애플리케이션이 실수로 예상과 다른 버킷에 대해 작업을 수행할 수 있습니다. 이와 같은 상황에서 의도하지 않은 버킷 상호 작용을 피하기 위해 버킷 소유자 조건을 사용할 수 있습니다. 버킷 소유자 조건을 사용하면 대상 버킷이 예상되는 AWS 계정에 속해 있는지 확인할 수 있으므로 S3 작업이 의도한 영향을 받는지 확인할 수 있습니다.

주제

- 버킷 소유자 조건을 사용해야 하는 경우
- 버킷 소유자 확인
- 예시
- 규제 및 제한

버킷 소유자 조건을 사용해야 하는 경우

예상되는 버킷 소유자의 계정 ID를 알고 있으면서 지원되는 S3 작업을 수행할 때마다 버킷 소유자 조건을 사용하는 것이 좋습니다. 버킷 소유자 조건은 모든 S3 객체 작업과 대부분의 S3 버킷 작업에 사용할 수 있습니다. 버킷 소유자 조건을 지원하지 않는 S3 작업 목록은 규제 및 제한 단원을 참조하십시오.

버킷 소유자 조건 사용의 이점을 확인하려면 AWS 고객 Bea에 대한 다음 시나리오를 고려하십시오.

1. Bea가 Amazon S3을 사용하는 애플리케이션을 개발합니다. Bea가 개발 중 테스트 전용 AWS 계정을 사용하여 `bea-data-test`라는 버킷을 생성하고 `bea-data-test`에 요청하도록 애플리케이션을 구성합니다.
2. Bea가 애플리케이션을 배포하면서 깜빡 잊고 프로덕션 AWS 계정에서 버킷을 사용하도록 애플리케이션을 재구성하지 않습니다.
3. 프로덕션 환경에서 Bea의 애플리케이션이 `bea-data-test`에 요청하고 이 요청이 성공합니다. 이로 인해 프로덕션 데이터가 Bea의 테스트 계정에 있는 버킷에 기록됩니다.

Bea는 버킷 소유자 조건을 사용하여 이와 같은 상황으로부터 보호할 수 있습니다. 버킷 소유자 조건을 사용할 경우 Bea는 요청에 예상 버킷 소유자의 AWS 계정 ID를 포함할 수 있습니다. 그러면 Amazon S3에서 각 요청을 처리하기 전에 버킷 소유자의 계정 ID를 확인합니다. 실제 버킷 소유자가 예상 버킷 소유자와 일치하지 않으면 요청이 실패합니다.

Bea가 버킷 소유자 조건을 사용할 경우 앞에서 설명한 시나리오에서 Bea의 애플리케이션이 실수로 테스트 버킷에 기록하는 것을 방지할 수 있습니다. 대신, 3단계에서 수행한 애플리케이션의 요청이 실패하며 `Access Denied` 오류 메시지가 표시됩니다. Bea는 버킷 소유자 조건을 사용하여 잘못된 AWS 계정의 버킷과 실수로 상호 작용할 위험을 없앨 수 있습니다.

버킷 소유자 확인

버킷 소유자 조건을 사용하려면 요청에 예상 버킷 소유자를 지정하는 파라미터를 포함해야 합니다. 대부분의 S3 작업에는 단일 버킷만 포함되며 버킷 소유자 조건을 사용하려면 이 단일 파라미터만 필요합니다. `CopyObject` 작업의 경우 이 첫 번째 파라미터는 대상 버킷의 예상 소유자를 지정하며, 원본 버킷의 예상 소유자를 지정하려면 두 번째 파라미터를 포함합니다.

버킷 소유자 조건 파라미터를 포함하는 요청을 하면 S3은 요청을 처리하기 전에 지정된 파라미터와 비교하여 버킷 소유자의 계정 ID를 확인합니다. 파라미터가 버킷 소유자의 계정 ID와 일치하면 S3이 요청을 처리합니다. 파라미터가 버킷 소유자의 계정 ID와 일치하지 않으면 요청이 실패하고 `Access Denied` 오류 메시지가 표시됩니다.

AWS Command Line Interface(AWS CLI), AWS SDK 및 Amazon S3 REST API에서 버킷 소유자 조건을 사용할 수 있습니다. AWS CLI 및 Amazon S3 REST API에서 버킷 소유자 조건을 사용하는 경우 다음 파라미터 이름을 사용합니다.

액세스 방법	비복사 작업에 대한 파라미터	복사 작업 원본 파라미터	복사 작업 대상 파라미터
AWS CLI	`--expected-bucket-owner`	`--expected-source-bucket-owner`	`--expected-bucket-owner`
Amazon S3 REST API	`x-amz-expected-bucket-owner` 헤더	`x-amz-source-expected-bucket-owner` 헤더	`x-amz-expected-bucket-owner` 헤더

AWS SDK에서 버킷 소유자 조건을 사용하는 데 필요한 파라미터 이름은 언어에 따라 다릅니다. 필요한 파라미터를 확인하려면 원하는 언어에 대한 SDK 설명서를 참조하십시오. SDK 설명서는 AWS에서의 빌드를 위한 도구에서 찾을 수 있습니다.

예시

다음 예제에서는 AWS CLI 또는 AWS SDK for Java 2.x를 사용하여 Amazon S3에 버킷 소유자 조건을 구현하는 방법을 보여 줍니다.

Example

예제: 객체 업로드

다음 예제에서는 버킷 소유자 조건을 사용하여 객체를 S3 버킷 *amzn-s3-demo-bucket1*에 업로드함으로써 AWS 계정 111122223333이 *amzn-s3-demo-bucket1*을 소유하는지 확인합니다.

AWS CLI

```
aws s3api put-object \
                --bucket amzn-s3-demo-bucket1 --key exampleobject --
body example_file.txt \
                --expected-bucket-owner 111122223333
```

AWS SDK for Java 2.x

```
public void putObjectExample() {
    S3Client s3Client = S3Client.create();;
    PutObjectRequest request = PutObjectRequest.builder()
            .bucket("amzn-s3-demo-bucket1")
            .key("exampleobject")
            .expectedBucketOwner("111122223333")
            .build();
    Path path = Paths.get("example_file.txt");
    s3Client.putObject(request, path);
}
```

Example

예제: 객체 복사

다음 예제에서는 S3 버킷 *amzn-s3-demo-bucket1*에서 S3 버킷 *amzn-s3-demo-bucket2*로 `object1` 객체를 복사합니다. 버킷 소유자 조건을 사용하여 버킷이 다음 표에 따라 예상 계정에 속해 있는지 확인합니다.

버킷	예상 소유자
amzn-s3-demo-bucket1	111122223333
amzn-s3-demo-bucket2	444455556666

AWS CLI

```
aws s3api copy-object --copy-source amzn-s3-demo-bucket1/object1 \
                            --bucket amzn-s3-demo-bucket2 --key object1copy \
                            --expected-source-bucket-owner 111122223333 --expected-
bucket-owner 444455556666
```

AWS SDK for Java 2.x

```
public void copyObjectExample() {
        S3Client s3Client = S3Client.create();
        CopyObjectRequest request = CopyObjectRequest.builder()
```

```
                .copySource("amzn-s3-demo-bucket1/object1")
                .destinationBucket("amzn-s3-demo-bucket2")
                .destinationKey("object1copy")
                .expectedSourceBucketOwner("111122223333")
                .expectedBucketOwner("444455556666")
                .build();
        s3Client.copyObject(request);
    }
```

Example

예제: 버킷 정책 검색

다음 예제에서는 버킷 소유자 조건을 사용하여 S3 버킷 *amzn-s3-demo-bucket1*에 대한 액세스 정책을 검색함으로써 AWS 계정 1111222333333이 *amzn-s3-demo-bucket1*을 소유하는지 확인합니다.

AWS CLI

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket1 --expected-bucket-
owner 111122223333
```

AWS SDK for Java 2.x

```
public void getBucketPolicyExample() {
    S3Client s3Client = S3Client.create();
    GetBucketPolicyRequest request = GetBucketPolicyRequest.builder()
            .bucket("amzn-s3-demo-bucket1")
            .expectedBucketOwner("111122223333")
            .build();
    try {
        GetBucketPolicyResponse response = s3Client.getBucketPolicy(request);
    }
    catch (S3Exception e) {
        // The call was transmitted successfully, but Amazon S3 couldn't process
        // it, so it returned an error response.
        e.printStackTrace();
    }
}
```

규제 및 제한

Amazon S3 버킷 소유자 조건에는 다음과 같은 규제 및 제한이 있습니다.

- 버킷 소유자 조건 파라미터의 값은 AWS 계정 ID(12자리 숫자 값)여야 합니다. 서비스 보안 주체는 지원되지 않습니다.
- 버킷 소유자 조건은 CreateBucket, ListBuckets 또는 AWS S3 제어에 포함된 모든 작업에서 사용할 수 없습니다. Amazon S3은 이러한 작업에 대한 요청에 포함된 모든 버킷 소유자 조건 파라미터를 무시합니다.
- 버킷 소유자 조건은 확인 파라미터에 지정된 계정이 버킷을 소유하고 있는지만 확인하며 버킷의 구성은 확인하지 않습니다. 또한 버킷의 구성이 특정 조건을 충족하거나 과거 상태와 일치함을 보장하지 않습니다.

객체 소유권 제어 및 버킷에 대해 ACL 사용 중지

S3 객체 소유권은 버킷에 업로드되는 객체의 소유권을 제어하고 액세스 제어 목록(ACL)을 비활성화 또는 활성화하는 데 사용할 수 있는 Amazon S3 버킷 수준 설정입니다. 기본적으로 객체 소유권은 버킷 소유자 적용 설정으로 설정되며 모든 ACL이 비활성화되어 있습니다. ACL이 비활성화되면 버킷 소유자는 버킷의 모든 객체를 소유하고 액세스 관리 정책을 사용하여 데이터에 대한 액세스를 독점적으로 관리합니다.

Amazon S3의 최신 사용 사례 대부분은 더 이상 ACL을 사용할 필요가 없으며, 각 객체에 대해 액세스를 개별적으로 제어해야 하는 비정상적인 상황을 제외하고는 ACL을 비활성화한 채로 두는 것이 좋습니다. ACL을 비활성화하면 누가 객체를 버킷에 업로드했는지에 관계없이 정책을 사용하여 버킷의 모든 객체에 대한 액세스를 보다 쉽게 제어할 수 있습니다.

객체 소유권에는 버킷에 업로드된 객체의 소유권을 제어하고 ACL을 사용 중지하거나 사용하는 다음과 같은 세 가지 설정이 있습니다.

ACL 사용 중지됨

- 버킷 소유자 적용(기본값) – ACL이 비활성화되고 버킷 소유자는 버킷의 모든 객체를 자동으로 소유하고 완전히 제어합니다. ACL은 더 이상 S3 버킷의 데이터에 대한 권한에 영향을 주지 않습니다. 버킷은 정책을 사용하여 액세스 제어를 정의합니다.

ACL 사용됨

- 버킷 소유자 기본(Bucket owner preferred) – 버킷 소유자가 `bucket-owner-full-control` 미리 제공 ACL을 사용하여 다른 계정이 버킷에 작성하는 새 객체를 소유하고 완전히 제어합니다.
- 객체 작성자(Object writer) – 객체를 업로드하는 AWS 계정은 객체를 소유하고 완전히 제어하며 ACL을 통해 다른 사용자에게 이에 대한 액세스 권한을 부여할 수 있습니다.

S3의 최신 사용 사례 대부분의 경우 버킷 소유자 적용 설정을 선택하여 ACL을 비활성화한 상태로 두고 버킷 정책을 사용하여 필요에 따라 계정 외부의 사용자와 데이터를 공유하는 것이 좋습니다. 이렇게 하면 권한 관리가 간소화됩니다. 새로 생성된 버킷과 기존 버킷 모두에서 ACL을 사용 중지할 수 있습니다. 새로 만든 버킷의 경우 기본적으로 ACL이 비활성화됩니다. 이미 객체가 있는 기존 버킷의 경우 ACL을 사용 중지하면 객체 및 버킷 ACL이 더 이상 액세스 평가의 일부가 아니며 정책에 따라 액세스가 부여되거나 거부됩니다. 기존 버킷의 경우 ACL을 사용 중지한 후 언제든지 다시 사용 설정할 수 있으며 기존 버킷과 객체 ACL이 복원됩니다.

ACL을 사용 중지하기 전에 버킷 정책을 검토하여 계정 외부에서 버킷에 대한 액세스 권한을 부여하려는 모든 방법을 포함하는지 확인하는 것이 좋습니다. ACL을 비활성화한 후 버킷은 ACL을 지정하지 않은 PUT 요청이나 버킷 소유자가 전체 제어 ACL을 사용하는 PUT 요청(예: `bucket-owner-full-control` 준비 ACL 또는 XML로 표현된 이 ACL의 동등한 형식)만 수락합니다. 버킷 소유자 전체 제어 ACL을 지원하는 기존 애플리케이션은 영향을 받지 않습니다. 다른 ACL(예: 특정 AWS 계정에 대한 사용자 정의 권한 부여)을 포함하는 PUT 요청은 실패하고 오류 코드 `AccessControlListNotSupported`와 함께 `400` 오류를 반환합니다.

반대로 버킷 소유자 선호 설정이 된 버킷은 버킷 및 객체 ACL을 계속 수락하고 준수합니다. 이 설정을 사용하면 `bucket-owner-full-control` 미리 제공 ACL로 작성된 새 객체는 객체 작성자가 아닌 버킷 소유자가 자동으로 소유합니다. 다른 모든 ACL 동작은 그대로 유지됩니다. 모든 Amazon S3 PUT 작업에 `bucket-owner-full-control` 준비된 ACL을 포함하도록 하려면 이 ACL을 사용하여 객체 업로드만 허용하는 버킷 정책을 추가하면 됩니다.

버킷에 적용되는 객체 소유권 설정을 확인하려면 Amazon S3 스토리지 렌즈 지표를 사용하면 됩니다. S3 스토리지 렌즈는 조직 전반에서 객체 스토리지 사용 및 활동에 대한 가시성을 확보하는 데 사용할 수 있는 클라우드 스토리지 분석 기능입니다. 자세한 내용은 Using S3 스토리지 렌즈 to audit Object Ownership settings(S3 스토리지 렌즈를 사용하여 객체 소유권 설정 감사)를 참조하십시오.

> **Note**
>
> Amazon S3 Express One Zone 스토리지 클래스를 디렉터리 버킷과 함께 사용하는 방법에 대한 자세한 내용은 S3 Express One Zone 및 디렉터리 버킷 작업 섹션을 참조하세요.

객체 소유권 설정

이 표는 각 객체 소유권 설정이 ACL, 객체, 객체 소유권 및 객체 업로드에 미치는 영향을 보여줍니다.

설정	적용 대상	객체 소유권에 미치는 영향	ACL에 미치는 영향	업로드 수락됨
버킷 소유자 적용(기본값)	모든 새 객체 및 기존 객체	버킷 소유자가 모든 객체를 소유합니다.	ACL이 사용 중지되어 더 이상 버킷에 대한 액세스 권한에 영향을 미치지 않습니다. ACL 설정 또는 업데이트 요청이 실패합니다. 그러나 ACL 읽기 요청은 지원됩니다. 버킷 소유자는 전체 소유권과 제어 권한을 가집니다. 객체 작성자가 더 이상 전체 소유권과 제어 권한을 갖지 않습니다.	버킷 소유자 전체 제어 ACL이 있는 업로드 또는 ACL을 지정하지 않는 업로드
버킷 소유자 기본	새 객체	객체 업로드에 `bucket-owner-full-control` 미리 제공 ACL이 포함	ACL이 업데이트 가능하며 권한을 부여할 수 있습니다.	모든 업로드

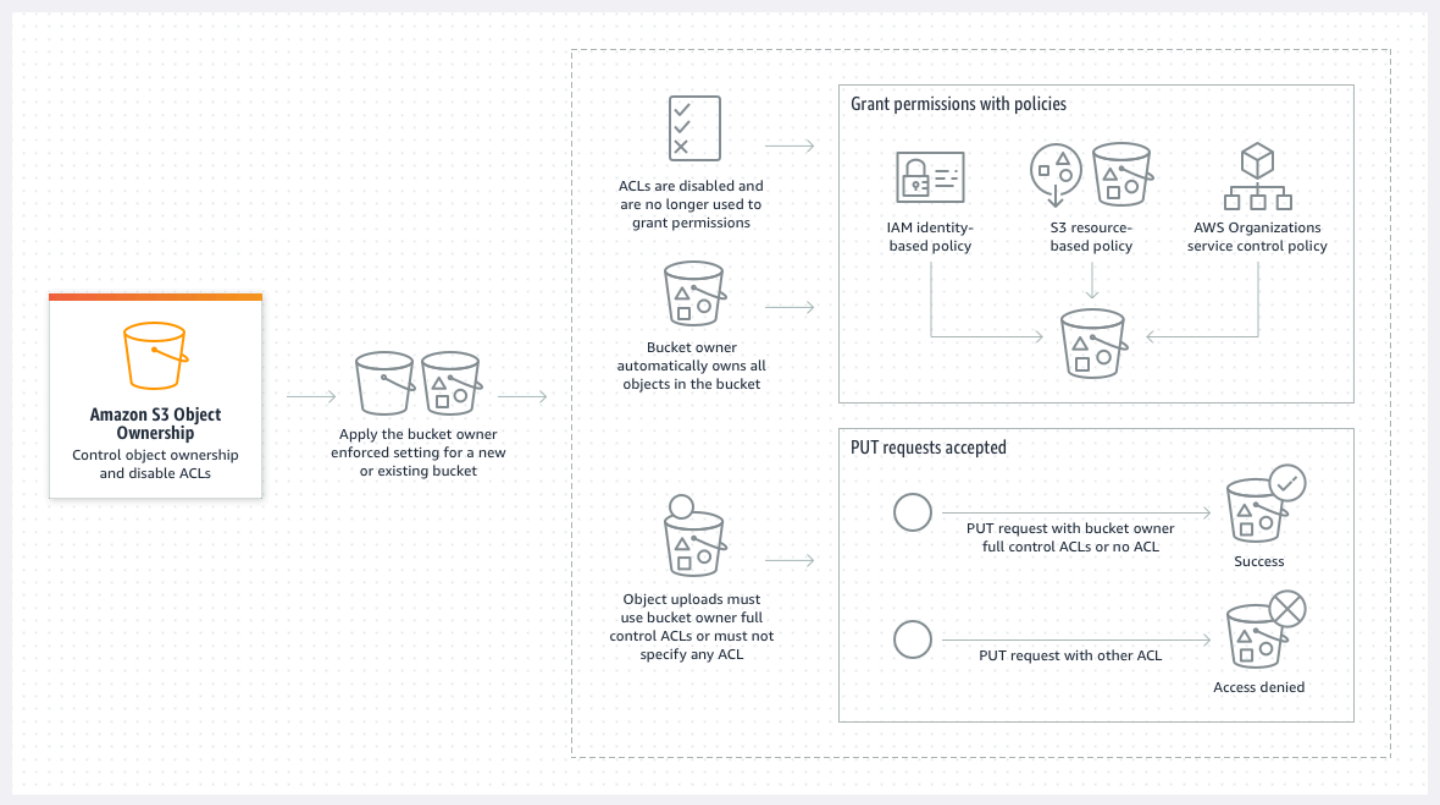
설정	적용 대상	객체 소유권에 미치는 영향	ACL에 미치는 영향	업로드 수락됨
		된 경우 버킷 소유자가 객체를 소유합니다. 다른 ACL과 함께 업로드된 객체는 쓰기 계정이 소유합니다.	객체 업로드에 `bucket-owner-full-control` 미리 제공 ACL이 포함된 경우 버킷 소유자는 전체 제어 액세스 권한을 가지며 객체 작성자는 더 이상 전체 제어 액세스 권한을 갖지 않습니다.	
객체 라이터	새 객체	객체 작성자가 객체를 소유합니다.	ACL이 업데이트 가능하며 권한을 부여할 수 있습니다. 객체 작성자가 전체 제어 액세스 권한을 가집니다.	모든 업로드

ACL 사용 중지로 인한 변경 사항

객체 소유권에 대해 버킷 소유자 적용 설정을 적용하면 ACL이 비활성화되고 추가 작업 없이 버킷의 모든 객체를 자동으로 소유하고 완전히 제어할 수 있습니다. 버킷 소유자 적용은 새로 생성된 모든 버킷의 기본 설정입니다. 버킷 소유자 적용 설정을 적용하면 세 가지 변경 사항이 표시됩니다.

- 모든 버킷 ACL 및 객체 ACL이 사용 중지되어 버킷 소유자로서 사용자에게 전체 액세스 권한이 부여됩니다. 버킷 또는 객체에 대해 읽기 ACL 요청을 수행하면 버킷 소유자에게만 전체 액세스 권한이 부여됩니다.
- 버킷 소유자가 버킷의 모든 객체를 자동으로 소유하고 완전히 제어할 수 있습니다.

- ACL이 더 이상 버킷에 대한 액세스 권한에 영향을 미치지 않습니다. 따라서 데이터에 대한 액세스 제어는 AWS Identity and Access Management(IAM) ID 기반 정책, Amazon S3 버킷 정책, VPC 엔드포인트 정책, 조직 서비스 제어 정책(SCP) 또는 리소스 제어 정책(RCP) 등 정책을 기반으로 합니다.



S3 버전 관리를 사용하는 경우 버킷 소유자는 버킷의 모든 객체 버전을 소유하고 완전히 제어합니다. 버킷 소유자 적용 설정을 적용해도 객체의 새 버전이 추가되지 않습니다.

새 객체는 버킷 소유자 전체 제어 ACL을 사용하거나 ACL을 지정하지 않는 경우에만 버킷에 업로드할 수 있습니다. 객체 업로드가 다른 ACL을 지정하면 실패합니다. 자세한 내용은 문제 해결 섹션을 참조하세요.

AWS Command Line Interface(AWS CLI)를 사용하는 다음 예제 `PutObject` 작업에는 `bucket-owner-full-control` 미리 제공 ACL이 포함되어 있으므로 객체를 ACL이 사용 중지된 버킷에 업로드할 수 있습니다.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key key-name --body path-to-file --
acl bucket-owner-full-control
```

다음 `PutObject` 작업은 ACL을 지정하지 않으므로 ACL이 사용 중지된 버킷에서도 성공합니다.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key key-name --body path-to-file
```

> **Note**
>
> 다른 AWS 계정이 업로드 후 객체에 액세스해야 하는 경우 버킷 정책을 통해 해당 계정에 추가 권한을 부여해야 합니다. 자세한 내용은 정책을 사용하여 Amazon S3 리소스에 대한 액세스를 관리하는 연습 섹션을 참조하세요.

ACL 다시 사용 설정

언제든지 버킷 소유자 적용 설정에서 다른 객체 소유권 설정으로 변경하여 ACL을 다시 활성화할 수 있습니다. 버킷 소유자 적용 설정을 적용하기 전에 권한 관리를 위해 객체 ACL을 사용하고 이러한 객체 ACL 권한을 버킷 정책으로 마이그레이션하지 않은 경우 ACL을 다시 활성화하면 이러한 권한이 복원됩니다. 또한 버킷 소유자 적용 설정이 적용된 동안 버킷에 작성된 객체는 여전히 버킷 소유자가 소유합니다.

예를 들어 버킷 소유자 적용 설정에서 다시 객체 라이터 설정으로 변경하면 버킷 소유자는 더 이상 다른 AWS 계정이 소유한 객체를 소유하지 않으며 해당 객체에 대한 전체 제어 권한을 갖게 됩니다. 대신 업로드하는 계정이 이러한 객체를 다시 소유합니다. 다른 계정이 소유한 객체는 권한에 ACL을 사용하므로 정책을 사용하여 이러한 객체에 대한 권한을 부여할 수 없습니다. 그러나 버킷 소유자는 버킷 소유자 적용 설정이 적용된 동안 버킷에 작성된 모든 객체를 계속 소유합니다. 이러한 객체는 ACL을 다시 사용 설정하더라도 객체 작성자가 소유하지 않습니다.

AWS Management Console, AWS Command Line Interface(CLI), REST API 또는 AWS SDK를 사용하여 ACL을 활성화하고 관리하는 방법은 ACL 구성 페이지를 참조하십시오.

ACL 사용 중지를 위한 사전 조건

다음은 기존 버킷에 대해 ACL을 사용 중지하기 위한 사전 조건입니다.

- 버킷 및 객체 ACL 검토와 ACL 권한 마이그레이션
- 인증을 위해 ACL이 필요한 모든 요청 식별
- ACL 관련 조건 키를 사용하는 버킷 정책 검토 및 업데이트

객체 소유권 권한

버킷에 대한 객체 소유권 설정을 적용, 업데이트 또는 삭제하려면 `s3:PutBucketOwnershipControls` 권한이 필요합니다. 버킷에 대한 객체 소유권 설정을 반환하려면 `s3:GetBucketOwnershipControls` 권한이 필요합니다. 자세한 내용은 버킷을 생성할 때 객체 소유권 설정 및 S3 버킷에 대한 객체 소유권 설정 보기 단원을 참조하세요.

모든 새 버킷에 대해 ACL 사용 중지

기본적으로 모든 새 버킷은 버킷 소유자 적용 설정이 적용된 상태로 생성되며 ACL은 비활성화됩니다. ACL을 비활성화된 상태로 유지하는 것이 좋습니다. 일반적으로 ACL 대신 액세스 제어를 위해 S3 리소스 기반 정책(버킷 정책 및 액세스 포인트 정책) 또는 IAM 정책을 사용하는 것이 좋습니다. 정책은 단순하고 더 유연한 액세스 제어 옵션입니다. 버킷 정책과 액세스 포인트 정책을 사용하면 Amazon S3 리소스에 대한 모든 요청에 광범위하게 적용되는 규칙을 정의할 수 있습니다.

복제 및 객체 소유권

S3 복제를 사용하고 서로 다른 AWS 계정이 소스 버킷과 대상 버킷을 소유하는 경우 ACL을 비활성화하여(객체 소유권에 대해 버킷 소유자 적용 설정 사용) 복제본 소유권을 대상 버킷을 소유하는 AWS 계정으로 변경할 수 있습니다. 이 설정은 `s3:ObjectOwnerOverrideToBucketOwner` 권한 없이 기존 소유자 재정의 동작을 모방합니다. 버킷 소유자 적용 설정으로 대상 버킷에 복제되는 모든 객체는 대상 버킷 소유자가 소유합니다. 복제 구성의 소유자 재정의 옵션에 대한 자세한 내용은 복제본 소유자 변경 섹션을 참조하십시오.

객체 소유권 설정

Amazon S3 콘솔, AWS CLI, AWS SDK, Amazon S3 REST API 또는 AWS CloudFormation을 사용하여 객체 소유권 설정을 적용할 수 있습니다. 다음 REST API 및 AWS CLI 명령은 객체 소유권을 지원합니다.

REST API	AWS CLI	설명
PutBucketOwnershipControls	put-bucket-ownership-controls	기존 S3 버킷에 대한 객체 소유권 설정을 생성하거나 수정합니다.
CreateBucket	create-bucket	`x-amz-object-ownership` 요청 헤더를 사용하여

REST API	AWS CLI	설명
		객체 소유권 설정을 지정하는 버킷을 생성합니다.
GetBucketOwnershipControls	get-bucket-ownership-controls	Amazon S3 버킷에 대한 객체 소유권 설정을 검색합니다.
DeleteBucketOwnershipControls	delete-bucket-ownership-controls	Amazon S3 버킷에 대한 객체 소유권 설정을 삭제합니다.

객체 소유권 설정 적용 및 작업에 대한 자세한 내용은 다음 주제를 참조하십시오.

주제

- ACL 사용 중지를 위한 사전 조건
- 버킷을 생성할 때 객체 소유권 설정
- 기존 버킷에 대한 객체 소유권 설정
- S3 버킷에 대한 객체 소유권 설정 보기
- 모든 새 버킷에 대해 ACL 사용 중지 및 객체 소유권 시행
- 문제 해결

ACL 사용 중지를 위한 사전 조건

Amazon S3의 버킷 액세스 제어 목록(ACL)은 S3 버킷 내의 개별 객체에 대한 세분화된 권한을 정의하여 해당 객체에 액세스하고 수정할 수 있는 AWS 계정 또는 그룹을 지정할 수 있는 메커니즘입니다. Amazon S3의 최신 사용 사례 대부분은 더 이상 ACL을 사용할 필요가 없습니다. 각 객체에 대해 액세스를 개별적으로 제어할 필요가 있는 드문 상황을 제외하고는 AWS Identity and Access Management(IAM) 및 버킷 정책을 사용하여 액세스를 관리하고 ACL을 비활성화한 채로 두는 것이 좋습니다.

버킷에서 ACL을 활성화한 경우 ACL을 비활성화하기 전에 다음 사전 조건을 충족해야 합니다.

주제

- 버킷 및 객체 ACL 검토와 ACL 권한 마이그레이션
- 인증을 위해 ACL이 필요한 모든 요청 식별

- ACL 관련 조건 키를 사용하는 버킷 정책 검토 및 업데이트
- 사용 사례 예제

버킷 및 객체 ACL 검토와 ACL 권한 마이그레이션

ACL을 사용 중지하면 버킷 및 객체 ACL에서 부여한 권한이 더 이상 액세스에 영향을 미치지 않습니다. ACL을 사용 중지하기 전에 버킷 및 객체 ACL을 검토합니다.

기존 버킷 및 객체 ACL마다 IAM 정책에 해당 항목이 있습니다. 다음 버킷 정책 예는 버킷 및 객체 ACL에 대한 `READ` 및 `WRITE` 권한이 IAM 권한에 매핑되는 방식을 보여줍니다. 각 ACL이 IAM 권한으로 변환되는 방법에 대한 자세한 내용은 ACL 권한과 액세스 정책 권한의 매핑 섹션을 참조하십시오.

ACL을 비활성화하기 전에:

- 버킷 ACL이 AWS 계정 외부에 액세스 권한을 부여하는 경우 먼저 버킷 ACL 권한을 버킷 정책으로 마이그레이션해야 합니다.
- 그런 다음 버킷 ACL을 기본 프라이빗 ACL로 재설정합니다.
- 또한 객체 수준 ACL 권한을 검토하고 버킷 정책으로 마이그레이션하는 것이 좋습니다.

버킷 ACL이 계정 외부의 다른 사용자에게 읽기 또는 쓰기 권한을 부여하는 경우 ACL을 비활성화하려면 먼저 이러한 권한을 버킷 정책으로 마이그레이션해야 합니다. 이러한 권한을 마이그레이션한 후 객체 소유권을 버킷 소유자 적용 설정으로 설정할 수 있습니다. 읽기 또는 쓰기 액세스 권한을 계정 외부로 부여하는 버킷 ACL을 마이그레이션하지 않으면 버킷 소유자 적용 설정을 적용하기 위한 요청이 실패하고 InvalidBucketAclWithObjectOwnership 오류 코드를 반환합니다.

버킷 ACL이 AWS 계정 외부의 액세스 권한을 부여하는 경우 ACL을 사용 중지하기 전에 버킷 ACL 권한을 버킷 정책으로 마이그레이션하고 버킷 ACL을 기본 프라이빗 ACL로 재설정해야 합니다. 이러한 버킷 ACL을 마이그레이션하고 재설정하지 않으면 ACL 비활성화를 위한 버킷 소유자 적용 설정 적용 요청이 실패하고 InvalidBucketAclWithObjectOwnership 오류 코드를 반환합니다. 또한 객체 ACL 권한을 검토하고 버킷 정책으로 마이그레이션하는 것이 좋습니다.

ACL 권한을 검토하고 버킷 정책으로 마이그레이션하려면 다음 주제를 참조하십시오.

주제

- 버킷 정책 예제
- S3 콘솔을 사용하여 ACL 권한 검토 및 마이그레이션
- AWS CLI를 사용하여 ACL 권한 검토 및 마이그레이션

버킷 정책 예제

이 예제 버킷 정책은 서드 파티 AWS 계정에 대한 READ 및 WRITE 버킷과 객체 ACL 권한을 버킷 정책으로 마이그레이션하는 방법을 보여줍니다. READ_ACP 및 WRITE_ACP ACL은 ACL 관련 권한(s3:GetBucketAcl, s3:GetObjectAcl, s3:PutBucketAcl 및 s3:PutObjectAcl)을 부여하기 때문에 정책과 관련이 적습니다.

Example - 버킷에 대한 **READ** ACL

버킷의 콘텐츠를 나열할 수 있는 AWS 계정 *111122223333* 권한을 부여하는 READ ACL이 버킷에 있는 경우, 버킷에 대
한 s3:ListBucket, s3:ListBucketVersions, s3:ListBucketMultipartUploads 권한을 부여하는 버킷 정책을 작성할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
   {
    "Sid": "Permission to list the objects in a bucket",
    "Effect": "Allow",
    "Principal": {
     "AWS": [

      "arn:aws:iam::111122223333:root"
     ]
    },
    "Action": [
     "s3:ListBucket",
     "s3:ListBucketVersions",
     "s3:ListBucketMultipartUploads"
    ],
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
   }
  ]
 }
```

Example - 버킷의 모든 객체에 대한 **READ** ACL

AWS 계정 *111122223333*에 액세스 권한을 부여하는 READ ACL이 버킷의 모든 객체에 있는 경우, 버킷의 모든 객체에 대해 이 계정에 s3:GetObject 및 s3:GetObjectVersion 권한을 부여하는 버킷 정책을 작성할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
   {
    "Sid": "Read permission for every object in a bucket",
    "Effect": "Allow",
    "Principal": {
     "AWS": [
      "arn:aws:iam::111122223333:root"
     ]
    },
    "Action": [
     "s3:GetObject",
     "s3:GetObjectVersion"
    ],
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
   }
  ]
 }
```

이 예제 리소스 요소는 특정 객체에 대한 액세스 권한을 부여합니다.

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/OBJECT-KEY"
```

Example – 버킷에 객체를 쓸 수 있는 권한을 부여하는 **WRITE** ACL

버킷에 객체를 쓸 수 있는 AWS 계정 *111122223333* 권한을 부여하는 WRITE ACL이 있는 경우, 버킷에 대한 s3:PutObject 권한을 부여하는 버킷 정책을 작성할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
   {
    "Sid": "Permission to write objects to a bucket",
    "Effect": "Allow",
    "Principal": {
     "AWS": [
      "arn:aws:iam::111122223333:root"
     ]
    },
    "Action": [
     "s3:PutObject"
```

```
    ],
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
   }
  ]
}
```

S3 콘솔을 사용하여 ACL 권한 검토 및 마이그레이션

버킷의 ACL 권한 검토

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 버킷 목록에서 버킷 이름을 선택합니다.
3. 권한 탭을 선택합니다.
4. 액세스 제어 목록(ACL)(Access control list (ACL))에서 버킷 ACL 권한을 검토합니다.

객체의 ACL 권한 검토

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. Buckets(버킷) 목록에서 객체가 포함된 버킷의 이름을 선택합니다.
3. 객체(Objects) 목록에서 사용자의 객체 이름을 선택합니다.
4. 권한 탭을 선택합니다.
5. 액세스 제어 목록(ACL)(Access control list (ACL))에서 객체 ACL 권한을 검토합니다.

ACL 권한 마이그레이션 및 버킷 ACL 업데이트

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 버킷 목록에서 버킷 이름을 선택합니다.
3. 권한(Permissions) 탭의 버킷 정책(Bucket policy)에서 편집(Edit)을 선택합니다.
4. 정책(Policy) 상자에서 버킷 정책을 추가하거나 업데이트합니다.

   예제 버킷 정책은 버킷 정책 예제 및 사용 사례 예제 섹션을 참조하십시오.
5. 변경 사항 저장을 선택합니다.
6. 버킷 ACL을 업데이트하여 다른 그룹 또는 AWS 계정에 대한 ACL 부여를 제거합니다.

7. 객체 소유권에 대한 버킷 소유자 적용 설정을 적용합니다.

AWS CLI를 사용하여 ACL 권한 검토 및 마이그레이션

1. 버킷에 대한 버킷 ACL을 반환하려면 get-bucket-acl AWS CLI 명령을 사용합니다.

```
aws s3api get-bucket-acl --bucket amzn-s3-demo-bucket
```

예를 들어, 이 버킷 ACL은 서드 파티 계정에 WRITE 및 READ 액세스 권한을 부여합니다. 이 ACL에서 서드 파티 계정은 정식 사용자 ID로 식별됩니다. 버킷 소유자 적용 설정을 적용하고 ACL을 비활성화하려면 서드 파티 계정에 대한 이러한 권한을 버킷 정책으로 마이그레이션해야 합니다.

```
{
  "Owner": {
   "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
   "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
  },
  "Grants": [
   {
    "Grantee": {
     "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
     "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
     "Type": "CanonicalUser"
    },
    "Permission": "FULL_CONTROL"
   },
   {
    "Grantee": {
     "DisplayName": "THIRD-PARTY-EXAMPLE-ACCOUNT",
     "ID":
 "72806de9d1ae8b171cca9e2494a8d1335dfced4ThirdPartyAccountCanonicalUserID",
     "Type": "CanonicalUser"
    },
    "Permission": "READ"
   },
   {
    "Grantee": {
     "DisplayName": "THIRD-PARTY-EXAMPLE-ACCOUNT",
     "ID":
 "72806de9d1ae8b171cca9e2494a8d1335dfced4ThirdPartyAccountCanonicalUserID",
     "Type": "CanonicalUser"
```

```
    },
    "Permission": "WRITE"
   }
  ]
 }
```

다른 예제 ACL은 사용 사례 예제 섹션을 참조하십시오.

2. 버킷 ACL 권한을 버킷 정책으로 마이그레이션합니다.

   이 예제 버킷 정책은 서드 파티 계정에 `s3:PutObject` 및 `s3:ListBucket` 권한을 부여합니다. 버킷 정책에서 서드 파티 계정은 AWS 계정 ID(*111122223333*)로 식별됩니다.

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://
policy.json

 policy.json:
 {
  "Version": "2012-10-17",
  "Statement": [
   {
    "Sid": "PolicyForCrossAccountAllowUpload",
    "Effect": "Allow",
    "Principal": {
     "AWS": [
      "arn:aws:iam::111122223333:root"
     ]
    },
    "Action": [
     "s3:PutObject",
     "s3:ListBucket"
    ],
    "Resource": [
     "arn:aws:s3:::amzn-s3-demo-bucket",
     "arn:aws:s3:::amzn-s3-demo-bucket/*"
   }
  ]
 }
```

   더 많은 예제 버킷 정책은 버킷 정책 예제 및 사용 사례 예제 섹션을 참조하십시오.

3. 특정 객체에 대한 ACL을 반환하려면 get-object-acl AWS CLI 명령을 사용합니다.

```
aws s3api get-object-acl --bucket amzn-s3-demo-bucket --key EXAMPLE-OBJECT-KEY
```

4. 필요한 경우 객체 ACL 권한을 버킷 정책으로 마이그레이션합니다.

   이 예제 리소스 요소는 버킷 정책의 특정 객체에 대한 액세스 권한을 부여합니다.

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/EXAMPLE-OBJECT-KEY"
```

5. 버킷의 ACL을 기본 ACL로 재설정합니다.

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-bucket --acl private
```

6. 객체 소유권에 대해 버킷 소유자 적용 설정을 적용합니다.

인증을 위해 ACL이 필요한 모든 요청 식별

인증을 위해 ACL이 필요한 Amazon S3 요청을 식별하려면, Amazon S3 서버 액세스 로그 또는 `aclRequired` 내의 AWS CloudTrail 값을 사용하면 됩니다. 요청에 권한 부여를 위해 ACL이 필요하거나 ACL을 지정하는 PUT 요청이 있는 경우, 문자열은 `Yes`입니다. ACL이 필요하지 않았거나, `bucket-owner-full-control` 미리 제공된 ACL을 설정하고 있거나, 버킷 정책에 의해 요청이 허용되는 경우, Amazon S3 서버 액세스 로그에서 `aclRequired` 값 문자열은 "-"이며 CloudTrail에는 표시되지 않습니다. 예상되는 `aclRequired` 값에 대한 자세한 내용은 일반적인 일반적인 Amazon S3 요청에 대한 `aclRequired` 값을 참조하십시오.

`bucket-owner-full-control` 미리 제공된 ACL을 제외하고 ACL 기반 권한을 부여하는 헤더가 포함된 `PutBucketAcl` 또는 `PutObjectAcl` 요청이 있는 경우, ACL을 비활성화하려면 먼저 해당 헤더를 제거해야 합니다. 그렇지 않으면 요청이 실패합니다.

권한 부여를 위해 ACL이 필요한 다른 모든 요청의 경우, 해당 ACL 권한을 버킷 정책으로 마이그레이션하십시오. 그런 다음 버킷 소유자 적용 설정을 활성화하기 전에 모든 버킷 ACL을 제거하십시오.

> **Note**
>
> 객체 ACL을 제거하지 마십시오. 그렇지 않으면 객체 ACL에 권한을 의존하는 애플리케이션이 액세스 권한을 잃게 됩니다.

권한 부여를 위해 ACL이 필요한 요청이 없는 것으로 확인되면 ACL을 비활성화할 수 있습니다. 요청 식별에 대한 자세한 내용은 Amazon S3 서버 액세스 로그를 사용하여 요청 식별 및 CloudTrail을 사용하여 Amazon S3 요청 식별 섹션을 참조하십시오.

ACL 관련 조건 키를 사용하는 버킷 정책 검토 및 업데이트

버킷 소유자 적용 설정을 적용하여 ACL을 비활성화한 후에는 요청이 버킷 소유자 전체 제어 ACL을 사용하거나 ACL을 지정하지 않는 경우에만 버킷에 새 객체를 업로드할 수 있습니다. ACL을 사용 중지하기 전에 ACL 관련 조건 키에 대한 버킷 정책을 검토합니다.

버킷 정책이 ACL 관련 조건 키를 사용하여 `bucket-owner-full-control` 미리 제공 ACL(예: `s3:x-amz-acl`)을 요구하는 경우 버킷 정책을 업데이트할 필요가 없습니다. 다음 버킷 정책은 `s3:x-amz-acl`을 사용하여 S3 `PutObject` 요청에 대해 `bucket-owner-full-control` 미리 제공 ACL을 요구합니다. 이 정책은 여전히 객체 작성자가 `bucket-owner-full-control` 미리 제공 ACL을 지정하도록 요구합니다. 그러나 ACL이 사용 중지된 버킷은 여전히 이 ACL을 수락하므로 클라이언트 측 변경 없이 요청이 계속 성공합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Only allow writes to my bucket with bucket owner full control",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/ExampleUser"
                ]
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control"
                }
            }
        }
    ]
}
```

그러나 버킷 정책에서 다른 ACL이 필요한 ACL 관련 조건 키를 사용하는 경우 이 조건 키를 제거해야 합니다. 이 예제 버킷 정책은 S3 `PutObject` 요청을 위해 `public-read` ACL을 필요로 하므로 ACL 사용 중지 전에 업데이트되어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Only allow writes to my bucket with public read access",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/ExampleUser"                ]
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "public-read"
                }
            }
        }
    ]
}
```

사용 사례 예제

다음 예에서는 특정 사용 사례에 대해 ACL 권한을 버킷 정책으로 마이그레이션하는 방법을 보여줍니다.

주제

- 서버 액세스 로깅을 위해 S3 로그 전송 그룹에 대한 액세스 권한 부여
- 버킷의 객체에 대한 퍼블릭 읽기 액세스 권한 부여
- Amazon ElastiCache(Redis OSS)에 S3 버킷에 대한 액세스 권한 부여

서버 액세스 로깅을 위해 S3 로그 전송 그룹에 대한 액세스 권한 부여

버킷 소유자 적용 설정을 적용하여 서버 액세스 로깅 대상 버킷(대상 버킷이라고도 함)에 대해 ACL을 비활성화하려는 경우 S3 로그 전송 그룹에 대한 버킷 ACL 권한을 버킷 정책의 로깅 서비스 보안 주체(`logging.s3.amazonaws.com`)로 마이그레이션해야 합니다. 로그 전달 권한에 대한 자세한 내용은 로그 전달을 위한 권한 섹션을 참조하십시오.

이 버킷 ACL은 S3 로그 전달 그룹에 `WRITE` 및 `READ_ACP` 액세스 권한을 부여합니다.

```
{
    "Owner": {
        "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "Type": "CanonicalUser",
                "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery"
            },
            "Permission": "WRITE"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery"
            },
            "Permission": "READ_ACP"
        }
    ]
}
```

버킷 정책의 로깅 서비스 보안 주체로 S3 로그 전달 그룹에 대한 버킷 ACL 권한 마이그레이션

1. 다음 버킷 정책을 대상 버킷에 추가하여 예시 값을 대체합니다.

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://
policy.json

policy.json:      {
    {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ServerAccessLogsPolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/EXAMPLE-LOGGING-PREFIX*",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::SOURCE-BUCKET-NAME"
                },
                "StringEquals": {
                    "aws:SourceAccount": "SOURCE-AWS-ACCOUNT-ID"
                }
            }
        }
    ]
}
```

2. 대상 버킷의 ACL을 기본 ACL로 재설정합니다.

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-bucket --acl private
```

3. 대상 버킷에 객체 소유권에 대해 버킷 소유자 적용 설정을 적용합니다.

버킷의 객체에 대한 퍼블릭 읽기 액세스 권한 부여

객체 ACL이 버킷의 모든 객체에 대해 퍼블릭 읽기 액세스 권한을 부여하는 경우 이러한 ACL 권한을 버킷 정책으로 마이그레이션할 수 있습니다.

이 객체 ACL은 버킷의 객체에 대한 퍼블릭 읽기 액세스 권한을 부여합니다.

```
{
    "Owner": {
        "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
            },
            "Permission": "READ"
        }
    ]
}
```

버킷 정책으로 퍼블릭 읽기 ACL 권한 마이그레이션

1. 버킷의 모든 객체에 대한 퍼블릭 읽기 액세스 권한을 부여하려면 다음 버킷 정책을 추가하여 예제 값을 바꿉니다.

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy
 file://policy.json

policy.json:
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

버킷 정책의 특정 객체에 대한 퍼블릭 액세스 권한을 부여하려면 Resource 요소에 다음 형식을 사용합니다.

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/OBJECT-KEY"
```

특정 접두사가 있는 모든 객체에 대한 퍼블릭 액세스 권한을 부여하려면 Resource 요소에 다음 형식을 사용합니다.

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/PREFIX/*"
```

2. 객체 소유권에 대해 버킷 소유자 적용 설정을 적용합니다.

Amazon ElastiCache(Redis OSS)에 S3 버킷에 대한 액세스 권한 부여

S3 버킷으로 ElastiCache(Redis OSS) 백업을 내보내면 ElastiCache 외부에서 백업에 액세스할 수 있습니다. S3 버킷으로 백업을 내보내려면 버킷에 스냅샷을 복사할 수 있는 권한을 ElastiCache에 부여해야 합니다. 버킷 ACL에서 ElastiCache에 권한을 부여한 경우 버킷 소유자 적용 설정을 적용하여 ACL을 비활성화하기 전에 이러한 권한을 버킷 정책으로 마이그레이션해야 합니다. 자세한 내용은 Amazon ElastiCache 사용 설명서의 ElastiCache에 Amazon S3 버킷에 대한 액세스 권한 부여를 참조하십시오.

다음 예에서는 ElastiCache에 권한을 부여하는 버킷 ACL 권한을 보여줍니다.

```
{
    "Owner": {
```

```
        "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "DisplayName": "aws-scs-s3-readonly",
                "ID":
 "540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353",
                "Type": "CanonicalUser"
            },
            "Permission": "READ"
        },
        {
            "Grantee": {
                "DisplayName": "aws-scs-s3-readonly",
                "ID":
 "540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353",
                "Type": "CanonicalUser"
            },
            "Permission": "WRITE"
        },
        {
            "Grantee": {
                "DisplayName": "aws-scs-s3-readonly",
                "ID":
 "540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353",
                "Type": "CanonicalUser"
            },
            "Permission": "READ_ACP"
        }
    ]
}
```

버킷 정책으로 ElastiCache(Redis OSS)에 대한 버킷 ACL 권한 마이그레이션

1. 다음 버킷 정책을 버킷에 추가하여 예제 값을 대체합니다.

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy
 file://policy.json

policy.json:
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt15399483",
            "Effect": "Allow",
            "Principal": {
                "Service": "Region.elasticache-snapshot.amazonaws.com"
            },
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketAcl",
                "s3:ListMultipartUploadParts",
                "s3:ListBucketMultipartUploads"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

2. 버킷의 ACL을 기본 ACL로 재설정합니다.

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-bucket --acl private
```

3. 객체 소유권에 대해 버킷 소유자 적용 설정을 적용합니다.

버킷을 생성할 때 객체 소유권 설정

버킷을 생성할 때 S3 객체 소유권을 구성할 수 있습니다. 기존 버킷에 대한 객체 소유권을 설정하려면 기존 버킷에 대한 객체 소유권 설정 섹션을 참조하세요.

S3 객체 소유권은 액세스 제어 목록(ACL)을 사용 중지하고 버킷에 있는 모든 객체의 소유권을 가져오는 데 사용할 수 있는 Amazon S3 버킷 수준 설정으로, Amazon S3에 저장된 데이터에 대한 액세스 관리를 간소화합니다. 기본적으로 S3 객체 소유권은 버킷 소유자 적용 설정으로 설정되며 신규 버킷에 대해 ACL이 비활성화되어 있습니다. ACL이 비활성화되면 버킷 소유자는 버킷의 모든 객체를 소유하고 액세스 관리 정책을 사용하여 데이터에 대한 액세스를 독점적으로 관리합니다. 각 객체에 대해 액세스를 개별적으로 제어해야 하는 드문 상황을 제외하고는 ACL을 비활성화한 채로 두는 것이 좋습니다.

객체 소유권에는 버킷에 업로드된 객체의 소유권을 제어하고 ACL을 사용 중지하거나 사용하는 다음과 같은 세 가지 설정이 있습니다.

ACL 사용 중지됨

- 버킷 소유자 적용(기본값) – ACL이 비활성화되고 버킷 소유자는 버킷의 모든 객체를 자동으로 소유하고 완전히 제어합니다. ACL은 더 이상 S3 버킷의 데이터에 대한 권한에 영향을 주지 않습니다. 버킷은 정책을 사용하여 액세스 제어를 정의합니다.

ACL 사용됨

- 버킷 소유자 기본(Bucket owner preferred) – 버킷 소유자가 `bucket-owner-full-control` 미리 제공 ACL을 사용하여 다른 계정이 버킷에 작성하는 새 객체를 소유하고 완전히 제어합니다.
- 객체 작성자(Object writer) – 객체를 업로드하는 AWS 계정은 객체를 소유하고 완전히 제어하며 ACL을 통해 다른 사용자에게 이에 대한 액세스 권한을 부여할 수 있습니다.

권한: 버킷 소유자 적용 설정이나 버킷 소유자 선호 설정을 적용하려면 `s3:CreateBucket` 및 `s3:PutBucketOwnershipControls` 권한이 있어야 합니다. 객체 라이터 설정이 적용된 버킷을 만들 때는 추가 권한이 필요하지 않습니다. Amazon S3 권한에 대한 자세한 내용은 서비스 승인 참조에서 Amazon S3에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

> **Important**
>
> Amazon S3의 최신 사용 사례 대부분은 더 이상 ACL을 사용할 필요가 없으며, 각 객체에 대해 액세스를 개별적으로 제어해야 하는 비정상적인 상황을 제외하고는 ACL을 사용 중지하는 것이 좋습니다. 객체 소유권으로 ACL을 사용 중지하고 액세스 제어 정책을 사용할 수 있습니다. ACL을 비활성화하면 다른 AWS 계정이 업로드한 객체로 버킷을 쉽게 유지 관리할 수 있습니다. 버킷 소유자는 버킷의 모든 객체를 소유하고 정책을 사용하여 객체에 대한 액세스를 관리할 수 있습니다.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 버킷을 생성하려는 리전을 선택합니다.

   > **Note**
   >
   > - 버킷을 만든 후에는 리전을 변경할 수 없습니다.
   > - 지연 시간과 요금을 최소화하고 규제 요건을 충족하려면 가장 가까운 리전을 선택하십시오. 특정 리전에 저장된 객체는 사용자가 명시적으로 객체를 다른 리전으로 전송하지 않는 한 해당 리전을 벗어나지 않습니다. Amazon S3 AWS 리전 목록은 Amazon Web Services 일반 참조의 AWS 서비스 엔드포인트를 참조하십시오.

3. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
4. 버킷 만들기를 선택합니다. 버킷 만들기 페이지가 열립니다.
5. [Bucket Name]에서 버킷 이름을 입력합니다.

   버킷 이름은 다음과 같아야 합니다.

   - 파티션 내에서 고유해야 합니다. 파티션은 리전의 그룹입니다. AWS에는 현재 `aws`(상용 리전), `aws-cn`(중국 리전) 및 `aws-us-gov`(AWS GovCloud (US) Regions), 이렇게 세 가지 파티션이 있습니다.
   - 3~63자 이내여야 합니다.

- 소문자, 숫자, 마침표(.) 및 하이픈(-)만 포함해야 합니다. 최상의 호환성을 위해 정적 웹 사이트 호스팅에만 사용되는 버킷을 제외하고 버킷 이름에 마침표(.)를 사용하지 않는 것이 좋습니다.
- 문자나 숫자로 시작하고 끝나야 합니다.
- 버킷 이름 지정 규칙의 전체 목록은 범용 버킷 이름 지정 규칙 섹션을 참조하세요.

> **Important**
>
> - 버킷을 생성한 후에는 해당 이름을 변경할 수 없습니다.
> - 버킷 이름에 민감한 정보가 포함되어서는 안 됩니다. 버킷 이름은 버킷의 객체를 가리키는 URL에 표시됩니다.

6. (선택 사항) 일반 구성 아래에서 기존 버킷의 설정을 새 버킷에 복사하도록 선택할 수 있습니다. 기존 버킷의 설정을 복사하지 않으려면 다음 단계로 건너뛰세요.

> **Note**
>
> 이 옵션:
>
> - AWS CLI에서는 사용할 수 없으며 Amazon S3 콘솔에서만 사용할 수 있습니다.
> - 기존 버킷의 버킷 정책을 새 버킷으로 복사하지 않습니다.

기존 버킷의 설정을 복사하려면 기존 버킷에서 설정 복사 아래에서 버킷 선택을 선택합니다. 버킷 선택 창이 열립니다. 복사하려는 설정이 있는 버킷을 찾아 버킷 선택을 선택합니다. 버킷 선택 창이 닫히고 버킷 생성 창이 다시 열립니다.

이제 기존 버킷에서 설정 복사 아래에 선택한 버킷의 이름이 표시됩니다. 이제 새 버킷의 설정이 선택한 버킷의 설정과 일치합니다. 복사된 설정을 제거하려면 기본값 복원을 선택합니다. 버킷 생성 페이지에서 나머지 버킷 설정을 검토하세요. 변경하지 않으려면 마지막 단계로 건너뛸 수 있습니다.

7. 객체 소유권(Object Ownership)에서 ACL을 사용 중지 또는 사용 설정하고 버킷에 업로드된 객체의 소유권을 제어하려면 다음 설정 중 하나를 선택합니다.

ACL 사용 중지됨

- 버킷 소유자 적용(기본값) – ACL이 비활성화되고 버킷 소유자는 범용 버킷의 모든 객체를 자동으로 소유하고 완전히 제어합니다. ACL은 더 이상 S3 범용 버킷의 데이터에 대한 권한에 영향을 주지 않습니다. 버킷은 정책을 독점적으로 사용하여 액세스 제어를 정의합니다.

  기본적으로 ACL은 비활성화되어 있습니다. Amazon S3의 최신 사용 사례 대부분은 더 이상 ACL을 사용할 필요가 없습니다. 각 객체에 대해 액세스를 개별적으로 제어해야 하는 드문 상황을 제외하고는 ACL을 비활성화한 채로 두는 것이 좋습니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

ACL 사용됨

- 버킷 소유자 기본(Bucket owner preferred) – 버킷 소유자가 `bucket-owner-full-control` 미리 제공 ACL을 사용하여 다른 계정이 버킷에 작성하는 새 객체를 소유하고 완전히 제어합니다.

  버킷 소유자 선호 설정을 적용하는 경우 모든 Amazon S3 업로드에 `bucket-owner-full-control` 미리 제공된 ACL을 포함하도록 요구하려면 이 ACL을 사용하는 객체 업로드만 허용하는 버킷 정책을 추가할 수 있습니다.
- 객체 작성자(Object writer) – 객체를 업로드하는 AWS 계정은 객체를 소유하고 완전히 제어하며 ACL을 통해 다른 사용자에게 이에 대한 액세스 권한을 부여할 수 있습니다.

> **Note**
>
> 기본 설정은 버킷 소유자 적용입니다. 기본 설정을 적용하고 ACL을 비활성화된 상태로 유지하려면 `s3:CreateBucket` 권한만 있으면 됩니다. ACL을 활성화하려면 `s3:PutBucketOwnershipControls` 권한이 있어야 합니다.

8. 퍼블릭 액세스 차단을 위한 버킷 설정에서 버킷에 적용할 퍼블릭 액세스 차단 설정을 선택합니다.

   기본적으로, 네 개의 퍼블릭 액세스 차단 설정이 모두 활성화되어 있습니다. 특정 사용 사례에 대해 하나 이상의 설정을 해제해야 하는 경우가 아니라면 모든 설정을 활성화된 상태로 유지하는 것이 좋습니다. 퍼블릭 액세스 차단에 대한 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하십시오.

> **Note**
>
> 모든 퍼블릭 액세스 차단 설정을 활성화하려면 `s3:CreateBucket` 권한만 있으면 됩니다. 퍼블릭 액세스 차단 설정을 해제하려면 `s3:PutBucketPublicAccessBlock` 권한이 있어야 합니다.

9. (선택 사항) 버킷 버전 관리는 기본으로 비활성화됩니다. 버전 관리는 동일 버킷 내에 여러 개의 객체 변형을 보유하는 것을 의미합니다. 버전 관리를 사용하여 버킷에 저장된 모든 버전의 객체를 모두 보존, 검색 및 복원할 수 있습니다. 또한, 의도치 않은 사용자 작업 및 애플리케이션 장애가 발생해도 쉽게 복구할 수 있습니다. 버전 관리에 대한 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하십시오.

   버킷에서 버전 관리를 활성화하려면 활성화를 선택합니다.

10. (선택 사항) Tags(태그)에서 버킷에 태그를 추가할 수 있습니다. AWS 비용 할당을 하면서 버킷 태그를 지정해 버킷 사용에 대한 요금 청구 주석을 달 수 있습니다. 태그는 버킷에 할당된 라벨을 나타내는 한 쌍의 키-값입니다. 자세한 내용은 the section called “비용 할당 태그 사용” 섹션을 참조하세요.

    버킷 태그를 추가하려면 Key(키) 및 원하는 경우 Value(값)를 입력하고 Add Tag(태그 추가)를 선택합니다.

11. 기본 암호화를 구성하려면 암호화 유형 아래에서 다음 중 하나를 선택합니다.

    - Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)
    - AWS Key Management Service 키를 사용한 서버 측 암호화(SSE-KMS)
    - AWS Key Management Service(AWS KMS) 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)

    > **Important**
    >
    > 기본 암호화 구성으로 SSE-KMS 또는 DSSE-KMS 옵션을 사용할 경우 AWS KMS의 초당 요청 수(RPS) 할당량이 적용됩니다. AWS KMS 할당량과 할당량 증대를 요청하는 방법에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 할당량을 참조하십시오.

버킷과 새 객체는 Amazon S3 관리형 키(SSE-S3)를 암호화 구성의 기본 수준으로 사용하는 서버 측 암호화로 암호화됩니다. 기본 암호화에 대한 자세한 내용은 Amazon S3 버킷에 대한 기본 서버 측 암호화 동작 설정 섹션을 참조하십시오. SSE-S3에 대한 자세한 내용은 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) 사용 섹션을 참조하세요.

서버 측 암호화를 사용하는 데이터 암호화에 대한 자세한 내용은 the section called “데이터 암호화” 섹션을 참조하세요.

12. Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) 또는 AWS Key Management Service(AWS KMS) 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)를 선택한 경우 다음을 수행하세요.

    a. AWS KMS 키에서 다음 방법 중 하나로 KMS 키를 지정합니다.

    - 사용 가능한 KMS 키 목록에서 AWS KMS keys 중에서 선택을 선택하고 사용 가능한 키 목록에서 KMS 키를 선택합니다.

      AWS 관리형 키(aws/s3)와 고객 관리형 키가 모두 목록에 표시됩니다. 고객 관리형 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 키 및 AWS 키를 참조하십시오.

    - KMS 키 ARN을 입력하려면 AWS KMS key ARN 입력을 선택하고 나타나는 필드에 KMS 키 ARN을 입력합니다.

    - AWS KMS 콘솔에서 고객 관리형 키를 생성하려면 KMS 키 생성을 선택합니다.

      AWS KMS key 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 키 생성을 참조하십시오.

> **Important**
>
> 버킷과 동일한 AWS 리전에서 사용할 수 있는 KMS 키만 사용 가능합니다. Amazon S3 콘솔은 버킷과 동일한 리전에 있는 처음 100개의 KMS 키만 나열합니다. 목록에 없는 KMS 키를 사용하려면 KMS 키 ARN을 입력해야 합니다. 다른 계정에서 소유한 KMS 키를 사용하려면 먼저 해당 키에 대한 사용 권한이 있어야 하고, 다음 단계로 KMS 키 ARN을 입력해야 합니다. KMS 키의 크로스 계정 권한에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 Creating KMS keys that other accounts can use를 참조하세요. SSE-KMS에 대한 자세한 내용은 AWS KMS(SSE-KMS)를 사용한 서버 측 암호화 지정 단원을 참조하십시오. DSSE-KMS에 대한 자세

한 내용은 the section called “이중 계층 서버 측 암호화(DSSE-KMS)” 섹션을 참조하세요.

Amazon S3에서 서버 측 암호화에 AWS KMS key을 사용하는 경우 대칭 암호화 KMS 키를 선택해야 합니다. Amazon S3는 대칭 암호화 KMS 키만 지원하며 비대칭 KMS 키는 지원하지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 대칭 및 비대칭 KMS 키 식별을 참조하십시오.

b. SSE-KMS와 함께 기본 암호화를 사용하도록 버킷을 구성할 때 S3 버킷 키를 사용할 수도 있습니다. S3 버킷 키를 사용하면 Amazon S3에서 AWS KMS로의 요청 트래픽이 줄어 암호화 비용이 절감됩니다. 자세한 내용은 Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감 섹션을 참조하세요. S3 버킷 키는 DSSE-KMS에서 지원되지 않습니다.

   기본적으로 S3 버킷 키는 Amazon S3 콘솔에서 활성화됩니다. 비용을 낮추려면 S3 버킷 키를 활성화된 상태로 두는 것이 좋습니다. 버킷에 대해 S3 버킷 키를 비활성화하려면 버킷 키 아래에서 비활성화를 선택합니다.

13. (선택 사항) S3 Object Lock을 사용하면 새 객체가 삭제되거나 덮어쓰이지 않도록 보호할 수 있습니다. 자세한 내용은 Object Lock으로 객체 잠금 섹션을 참조하세요. S3 Object Lock을 활성화하려면 다음을 수행합니다.

    a. 고급 설정(Advanced Settings)을 선택합니다.

    > **Important**
    >
    > Object Lock을 활성화하면 버킷의 버전 관리도 활성화됩니다. 버킷을 활성화하고 성공적으로 만든 후에는 버킷의 속성 탭에서 Object Lock 기본 보존 및 법적 보존 설정도 구성해야 합니다.

    b. 객체 잠금을 활성화하려면 Enable(활성화)을 선택하고 표시되는 경고를 읽고 확인합니다.

> **Note**
>
> Object Lock을 사용하는 버킷을 만들려면 `s3:CreateBucket`, `s3:PutBucketVersioning` 및 `s3:PutBucketObjectLockConfiguration` 권한이 있어야 합니다.

14. 버킷 생성을 선택합니다.

AWS CLI 사용

새 버킷을 생성할 때 객체 소유권을 설정하려면 `--object-ownership` 파라미터와 함께 `create-bucket` AWS CLI 명령을 사용합니다.

이 예에서는 AWS CLI를 사용하여 새 버킷에 대해 버킷 소유자 적용 설정을 적용합니다.

```
aws s3api create-bucket --bucket  amzn-s3-demo-bucket --region us-east-1 --object-
ownership BucketOwnerEnforced
```

> **Important**
>
> AWS CLI를 사용하여 버킷을 만들 때 객체 소유권을 설정하지 않으면 기본 설정은 `ObjectWriter`(ACL 활성화)가 됩니다.

Java용 AWS SDK 사용

이 예에서는 AWS SDK for Java를 사용하여 새 버킷에 대해 버킷 소유자 적용 설정을 지정합니다.

```
    // Build the ObjectOwnership for CreateBucket
    CreateBucketRequest createBucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectOwnership(ObjectOwnership.BucketOwnerEnforced)
            .build()

     // Send the request to Amazon S3
     s3client.createBucket(createBucketRequest);
```

AWS CloudFormation 사용하기

새 버킷을 생성할 때 `AWS::S3::Bucket` AWS CloudFormation 리소스를 사용하여 객체 소유권을 설정하려면 AWS CloudFormation 사용 설명서의 AWS::S3::Bucket 내 OwnershipControls 섹션을 참조하십시오.

REST API 사용

S3 객체 소유권에 대해 버킷 소유자 적용 설정을 적용하려면 `x-amz-object-ownership` 요청 헤더를 `BucketOwnerEnforced`로 설정한 상태에서 `CreateBucket` API 작업을 사용합니다. 자세한 내용 및 예제는 Amazon Simple Storage Service API 참조의 CreateBucket를 참조하십시오.

다음 단계: 객체 소유권에 대해 버킷 소유자 적용 또는 버킷 소유자 기본 설정을 적용한 후 다음 단계를 추가로 수행할 수 있습니다.

- 버킷 소유자 시행(Bucket owner enforced) – IAM 또는 조직 정책을 사용하여 사용 중지된 ACL로 모든 새 버킷을 생성하도록 요구합니다.
- 버킷 소유자 기본(Bucket owner preferred) - 버킷에 모든 객체 업로드에 대해 `bucket-owner-full-control` 미리 제공 ACL을 요구하도록 S3 버킷 정책을 추가합니다.

기존 버킷에 대한 객체 소유권 설정

기존 S3 버킷에 대한 S3 객체 소유권을 구성할 수 있습니다. 버킷을 생성할 때 객체 소유권을 적용하려면 버킷을 생성할 때 객체 소유권 설정 섹션을 참조하십시오.

S3 객체 소유권은 액세스 제어 목록(ACL)을 사용 중지하고 버킷에 있는 모든 객체의 소유권을 가져오는 데 사용할 수 있는 Amazon S3 버킷 수준 설정으로, Amazon S3에 저장된 데이터에 대한 액세스 관리를 간소화합니다. 기본적으로 S3 객체 소유권은 버킷 소유자 적용 설정으로 설정되며 신규 버킷에 대해 ACL이 비활성화되어 있습니다. ACL이 비활성화되면 버킷 소유자는 버킷의 모든 객체를 소유하고 액세스 관리 정책을 사용하여 데이터에 대한 액세스를 독점적으로 관리합니다. 각 객체에 대해 액세스를 개별적으로 제어해야 하는 드문 상황을 제외하고는 ACL을 비활성화한 채로 두는 것이 좋습니다.

객체 소유권에는 버킷에 업로드된 객체의 소유권을 제어하고 ACL을 사용 중지하거나 사용하는 다음과 같은 세 가지 설정이 있습니다.

ACL 사용 중지됨

- 버킷 소유자 적용(기본값) – ACL이 비활성화되고 버킷 소유자는 버킷의 모든 객체를 자동으로 소유하고 완전히 제어합니다. ACL은 더 이상 S3 버킷의 데이터에 대한 권한에 영향을 주지 않습니다. 버킷은 정책을 사용하여 액세스 제어를 정의합니다.

ACL 사용됨

- 버킷 소유자 기본(Bucket owner preferred) – 버킷 소유자가 `bucket-owner-full-control` 미리 제공 ACL을 사용하여 다른 계정이 버킷에 작성하는 새 객체를 소유하고 완전히 제어합니다.
- 객체 작성자(Object writer) – 객체를 업로드하는 AWS 계정은 객체를 소유하고 완전히 제어하며 ACL을 통해 다른 사용자에게 이에 대한 액세스 권한을 부여할 수 있습니다.

사전 조건: 버킷 소유자 적용 설정을 적용하여 ACL을 비활성화하기 전에 버킷 ACL 권한을 버킷 정책으로 마이그레이션하고 버킷 ACL을 기본 프라이빗 ACL로 재설정해야 합니다. 또한 객체 ACL 권한을

버킷 정책으로 마이그레이션하고 버킷 소유자 전체 제어 ACL 이외의 ACL이 필요한 버킷 정책을 편집하는 것이 좋습니다. 자세한 내용은 ACL 사용 중지를 위한 사전 조건 섹션을 참조하세요.

권한: 이 작업을 사용하려면 `s3:PutBucketOwnershipControls` 권한이 있어야 합니다. Amazon S3 권한에 대한 자세한 내용은 서비스 승인 참조에서 Amazon S3에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 버킷(Buckets) 목록에서 S3 객체 소유권 설정을 적용할 버킷의 이름을 선택합니다.
3. 권한 탭을 선택합니다.
4. 객체 소유권(Object Ownership)에서 편집(Edit)을 선택합니다.
5. 객체 소유권(Object Ownership)에서 ACL을 사용 중지 또는 사용 설정하고 버킷에 업로드된 객체의 소유권을 제어하려면 다음 설정 중 하나를 선택합니다.

   ACL 사용 중지됨

   - 버킷 소유자 시행(Bucket owner enforced) – ACL이 사용 중지되고 버킷 소유자는 버킷의 모든 객체를 자동으로 소유하고 완전히 제어합니다. ACL은 더 이상 S3 버킷의 데이터에 대한 권한에 영향을 주지 않습니다. 버킷은 정책을 사용하여 액세스 제어를 정의합니다.

     IAM 또는 AWS Organizations 정책을 사용하여 사용 중지된 ACL로 모든 새 버킷을 생성하도록 요구하려면 모든 새 버킷에 대해 ACL 사용 중지(버킷 소유자 시행) 섹션을 참조하십시오.

   ACL 사용됨

   - 버킷 소유자 기본(Bucket owner preferred) – 버킷 소유자가 `bucket-owner-full-control` 미리 제공 ACL을 사용하여 다른 계정이 버킷에 작성하는 새 객체를 소유하고 완전히 제어합니다.

     버킷 소유자 기본 설정을 적용하는 경우 모든 Amazon S3 업로드에 `bucket-owner-full-control` 미리 제공 ACL을 포함하도록 요구하려면 이 ACL을 사용하는 객체 업로드만 허용하는 버킷 정책을 추가할 수 있습니다.

- 객체 작성자(Object writer) – 객체를 업로드하는 AWS 계정은 객체를 소유하고 완전히 제어하며 ACL을 통해 다른 사용자에게 이에 대한 액세스 권한을 부여할 수 있습니다.

6. 저장을 선택합니다.

AWS CLI 사용

기존 버킷에 대한 객체 소유권 설정을 적용하려면 `--ownership-controls` 파라미터와 함께 `put-bucket-ownership-controls` 명령을 사용합니다. 유효한 소유권 값은`BucketOwnerEnforced`, `BucketOwnerPreferred`, 또는 `ObjectWriter`입니다.

이 예에서는 AWS CLI를 사용하여 기존 버킷에 대해 버킷 소유자 적용 설정을 적용합니다.

```
aws s3api put-bucket-ownership-controls --bucket amzn-s3-demo-bucket --ownership-
controls="Rules=[{ObjectOwnership=BucketOwnerEnforced}]"
```

`put-bucket-ownership-controls`에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 put-bucket-ownership-controls 섹션을 참조하십시오.

Java용 AWS SDK 사용

이 예에서는 AWS SDK for Java를 사용하여 기존 버킷에 대해 `BucketOwnerEnforced` 설정을 적용합니다.

```
        // Build the ObjectOwnership for BucketOwnerEnforced
        OwnershipControlsRule rule = OwnershipControlsRule.builder()
               .objectOwnership(ObjectOwnership.BucketOwnerEnforced)
               .build();

        OwnershipControls ownershipControls = OwnershipControls.builder()
                  .rules(rule)
                  .build()

         // Build the PutBucketOwnershipControlsRequest
         PutBucketOwnershipControlsRequest putBucketOwnershipControlsRequest =
               PutBucketOwnershipControlsRequest.builder()
                       .bucket(BUCKET_NAME)
                       .ownershipControls(ownershipControls)
                       .build();

         // Send the request to Amazon S3
```

```
s3client.putBucketOwnershipControls(putBucketOwnershipControlsRequest);
```

AWS CloudFormation 사용하기

AWS CloudFormation을 사용하여 기존 버킷에 대한 객체 소유권 설정을 적용하려면 AWS CloudFormation 사용 설명서의 AWS::S3::Bucket OwnershipControls 섹션을 참조하십시오.

REST API 사용

REST API를 사용하여 기존 S3 버킷에 객체 소유권 설정을 적용하려면 `PutBucketOwnershipControls`를 사용하십시오. 자세한 내용은 Amazon Simple Storage Service API 참조의 PutBucketOwnershipControls를 참조하십시오.

다음 단계: 객체 소유권에 대해 버킷 소유자 적용 또는 버킷 소유자 기본 설정을 적용한 후 다음 단계를 추가로 수행할 수 있습니다.

- 버킷 소유자 시행(Bucket owner enforced) – IAM 또는 조직 정책을 사용하여 사용 중지된 ACL로 모든 새 버킷을 생성하도록 요구합니다.
- 버킷 소유자 기본(Bucket owner preferred) - 버킷에 모든 객체 업로드에 대해 `bucket-owner-full-control` 미리 제공 ACL을 요구하도록 S3 버킷 정책을 추가합니다.

S3 버킷에 대한 객체 소유권 설정 보기

S3 객체 소유권은 액세스 제어 목록(ACL)을 사용 중지하고 버킷에 있는 모든 객체의 소유권을 가져오는 데 사용할 수 있는 Amazon S3 버킷 수준 설정으로, Amazon S3에 저장된 데이터에 대한 액세스 관리를 간소화합니다. 기본적으로 S3 객체 소유권은 버킷 소유자 적용 설정으로 설정되며 신규 버킷에 대해 ACL이 비활성화되어 있습니다. ACL이 비활성화되면 버킷 소유자는 버킷의 모든 객체를 소유하고 액세스 관리 정책을 사용하여 데이터에 대한 액세스를 독점적으로 관리합니다. 각 객체에 대해 액세스를 개별적으로 제어해야 하는 드문 상황을 제외하고는 ACL을 비활성화한 채로 두는 것이 좋습니다.

객체 소유권에는 버킷에 업로드된 객체의 소유권을 제어하고 ACL을 사용 중지하거나 사용하는 다음과 같은 세 가지 설정이 있습니다.

ACL 사용 중지됨

- 버킷 소유자 적용(기본값) – ACL이 비활성화되고 버킷 소유자는 버킷의 모든 객체를 자동으로 소유하고 완전히 제어합니다. ACL은 더 이상 S3 버킷의 데이터에 대한 권한에 영향을 주지 않습니다. 버킷은 정책을 사용하여 액세스 제어를 정의합니다.

ACL 사용됨

- 버킷 소유자 기본(Bucket owner preferred) – 버킷 소유자가 `bucket-owner-full-control` 미리 제공 ACL을 사용하여 다른 계정이 버킷에 작성하는 새 객체를 소유하고 완전히 제어합니다.
- 객체 작성자(Object writer) – 객체를 업로드하는 AWS 계정은 객체를 소유하고 완전히 제어하며 ACL을 통해 다른 사용자에게 이에 대한 액세스 권한을 부여할 수 있습니다.

Amazon S3 버킷에 대한 S3 객체 소유권 설정을 볼 수 있습니다. 새 버킷에 대한 객체 소유권을 설정하려면 버킷을 생성할 때 객체 소유권 설정 섹션을 참조하십시오. 기존 버킷에 대한 객체 소유권을 설정하려면 기존 버킷에 대한 객체 소유권 설정 섹션을 참조하십시오.

권한: 이 작업을 사용하려면 `s3:GetBucketOwnershipControls` 권한이 있어야 합니다. Amazon S3 권한에 대한 자세한 내용은 서비스 승인 참조에서 Amazon S3에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 버킷(Buckets) 목록에서 객체 소유권 설정을 적용할 버킷의 이름을 선택합니다.
3. 권한 탭을 선택합니다.
4. 객체 소유권(Object Ownership)에서 버킷에 대한 객체 소유권 설정을 볼 수 있습니다.

AWS CLI 사용

S3 버킷에 대한 S3 객체 소유권 설정을 검색하려면 get-bucket-ownership-controls AWS CLI 명령을 사용합니다.

```
aws s3api get-bucket-ownership-controls --bucket amzn-s3-demo-bucket
```

REST API 사용

S3 버킷에 대한 객체 소유권 설정을 검색하려면 `GetBucketOwnershipControls` API 작업을 사용합니다. 자세한 내용은 GetBucketOwnershipControls 섹션을 참조하세요.

모든 새 버킷에 대해 ACL 사용 중지 및 객체 소유권 시행

Amazon S3 버킷에서 ACL을 사용 중지하는 것이 좋습니다. S3 객체 소유권에 대해 버킷 소유자 적용 설정을 적용하여 이 작업을 수행할 수 있습니다. 이 설정을 적용하면 ACL이 사용 중지되고 버킷의 모든 객체를 자동으로 소유하고 완전히 제어할 수 있습니다. 모든 새 버킷이 ACL을 비활성화한 상태로 생성되도록 하려면 다음 섹션에 설명된 대로 AWS Identity and Access Management(IAM) 정책 또는 AWS Organizations 서비스 제어 정책(SCP)을 사용하십시오.

ACL을 사용 중지하지 않고 새 객체에 대해 객체 소유권을 시행하려면 버킷 소유자 기본 설정을 적용합니다. 이 설정을 적용할 때 버킷에 대한 모든 PUT 요청에 `bucket-owner-full-control` 미리 제공된 ACL을 요구하도록 버킷 정책을 업데이트하는 것이 좋습니다. 다른 계정에서 버킷으로 `bucket-owner-full-control` 미리 제공된 ACL을 보내도록 클라이언트도 업데이트해야 합니다.

주제

- 모든 새 버킷에 대해 ACL 사용 중지(버킷 소유자 시행)
- Amazon S3 PUT 작업에 bucket-owner-full-control 미리 제공된 ACL 요구(버킷 소유자 기본)

모든 새 버킷에 대해 ACL 사용 중지(버킷 소유자 시행)

다음 예시 IAM 정책은 객체 소유권에 대해 버킷 소유자 적용 설정이 적용되지 않는 한 특정 IAM 사용자 또는 역할에 대해 `s3:CreateBucket` 권한을 거부합니다. `Condition` 블록의 키-값 페어는 `s3:x-amz-object-ownership`을 키로 지정하고 `BucketOwnerEnforced` 설정을 값으로 지정합니다. 즉, IAM 사용자는 객체 소유권에 대해 버킷 소유자 적용 설정을 지정하고 ACL을 비활성화하는 경우에만 버킷을 생성할 수 있습니다. 이 정책을 AWS 조직의 경계 SCP로 사용할 수도 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RequireBucketOwnerFullControl",
            "Action": "s3:CreateBucket",
            "Effect": "Deny",
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-object-ownership": "BucketOwnerEnforced"
                }
            }
        }
    ]
```

```
}
```

Amazon S3 **PUT** 작업에 bucket-owner-full-control 미리 제공된 ACL 요구(버킷 소유자 기본)

객체 소유권에 대해 버킷 소유자 기본 설정을 사용하면 버킷 소유자는 다른 계정이 `bucket-owner-full-control` 미리 제공된 ACL을 사용하여 버킷에 작성하는 새 객체를 소유하고 완전히 제어할 수 있습니다. 그러나 다른 계정이 `bucket-owner-full-control` 미리 제공 ACL 없이 버킷에 객체를 작성하는 경우 객체 작성자는 모든 제어 액세스 권한을 유지합니다. 버킷 소유자는 `bucket-owner-full-control` 미리 제공 ACL을 지정하는 경우에만 쓰기를 허용하는 버킷 정책을 구현할 수 있습니다.

> **Note**
>
> 버킷 소유자 적용 설정으로 ACL을 비활성화한 경우 버킷 소유자는 버킷의 모든 객체를 자동으로 소유하고 완전히 제어할 수 있습니다. 버킷 소유자에 대해 객체 소유권을 시행하기 위해 이 섹션을 사용하여 버킷 정책을 업데이트할 필요가 없습니다.

다음 버킷 정책은 객체의 ACL이 111122223333로 설정된 경우에만 *amzn-s3-demo-bucket* 계정이 *bucket-owner-full-control*에 객체를 업로드할 수 있도록 지정합니다. *111122223333*을 계정으로 바꾸고 *amzn-s3-demo-bucket*을 버킷의 이름으로 바꿔야 합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Only allow writes to my bucket with bucket owner full control",
         "Effect": "Allow",
         "Principal": {
            "AWS": [
               "arn:aws:iam::111122223333:user/ExampleUser"
            ]
         },
         "Action": [
            "s3:PutObject"
         ],
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
         "Condition": {
            "StringEquals": {
```

```
                "s3:x-amz-acl": "bucket-owner-full-control"
            }
        }
    }
  ]
}
```

다음은 AWS Command Line Interface(AWS CLI)를 사용하여 미리 제공된 ACL `bucket-owner-full-control`을 포함하는 복사 작업을 수행하는 예입니다.

```
aws s3 cp file.txt s3://amzn-s3-demo-bucket --acl bucket-owner-full-control
```

버킷 정책이 적용된 후 클라이언트에 `bucket-owner-full-control` 미리 제공된 ACL이 포함되어 있지 않으면 작업이 실패하고 업로드하는 사용자에게 다음 오류가 표시됩니다.

An error occurred (AccessDenied) when calling the PutObject operation: Access Denied(PutObject 작업을 호출할 때 오류(AccessDenied) 발생: 액세스 거부).

> **Note**
>
> 업로드 후 클라이언트에서 객체에 액세스해야 하는 경우 업로드하는 계정에 추가 권한을 부여해야 합니다. 계정에 리소스에 대한 액세스 권한을 부여하는 방법은 정책을 사용하여 Amazon S3 리소스에 대한 액세스를 관리하는 연습 단원을 참조하십시오.

문제 해결

S3 객체 소유권에 대해 버킷 소유자 적용 설정을 적용하면 액세스 제어 목록(ACL)이 비활성화되고 버킷 소유자는 버킷의 모든 객체를 자동으로 소유합니다. ACL은 더 이상 버킷의 객체에 대한 권한에 영향을 주지 않습니다. 정책을 사용하여 권한을 부여할 수 있습니다. 모든 S3 PUT 요청은 `bucket-owner-full-control` 미리 준비된 ACL을 지정하거나 ACL을 지정하지 않아야 합니다. 이렇게 하지 않으면 요청이 실패합니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

잘못된 ACL이 지정되거나 버킷 ACL 권한이 AWS 계정 외부의 액세스 권한을 부여하면 다음과 같은 오류 응답이 표시될 수 있습니다.

AccessControlListNotSupported

객체 소유권에 대해 버킷 소유자 적용 설정을 적용하면 ACL이 비활성화됩니다. ACL 설정 또는 ACL 업데이트 요청은 400 오류와 함께 실패하고 AccessControlListNotSupported 오류 코드를 반환합니다. ACL 읽기 요청은 계속 지원됩니다. ACL 읽기 요청은 항상 버킷 소유자에 대한 전체 제어 권한을 보여주는 응답을 반환합니다. PUT 작업에서 버킷 소유자 전체 권한 ACL을 지정하거나 ACL을 지정하지 않아야 합니다. 그렇지 않으면 PUT 작업이 실패합니다.

다음 예제 put-object AWS CLI 명령에는 public-read 미리 제공된 ACL이 포함되어 있습니다.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key object-key-name --body doc-example-body --acl public-read
```

버킷이 버킷 소유자 적용 설정을 사용하여 ACL을 비활성화하면 작업이 실패하고 업로드하는 사용자에게 다음 오류 메시지가 표시됩니다.

PutObject 작업을 호출할 때 오류(AccessControlListNotSupported)가 발생했습니다. 버킷이 ACL을 허용하지 않습니다.

InvalidBucketAclWithObjectOwnership

버킷 소유자 적용 설정을 적용하여 ACL을 비활성화하려면 버킷 ACL이 버킷 소유자에게만 전체 제어 권한을 부여해야 합니다. 버킷 ACL은 외부 AWS 계정 또는 다른 그룹에 대한 액세스 권한을 부여할 수 없습니다. 예를 들어 CreateBucket 요청이 버킷 소유자 시행을 설정하고 외부 AWS 계정에 대한 액세스를 제공하는 버킷 ACL을 지정하는 경우 요청은 400 오류와 함께 실패하고 InvalidBucketAclWithObjectOwnership 오류 코드를 반환합니다. 마찬가지로 PutBucketOwnershipControls 요청이 다른 사람에게 권한을 부여하는 버킷 ACL이 있는 버킷에 적용되는 버킷 소유자를 설정하는 경우 요청이 실패합니다.

Example : 기존 버킷 ACL이 퍼블릭 읽기 액세스 권한 부여

예를 들어 기존 버킷 ACL이 퍼블릭 읽기 액세스 권한을 부여하는 경우 이러한 ACL 권한을 버킷 정책으로 마이그레이션하고 버킷 ACL을 기본 프라이빗 ACL로 재설정할 때까지 객체 소유권에 대해 버킷 소유자 적용 설정을 적용할 수 없습니다. 자세한 내용은 ACL 사용 중지를 위한 사전 조건 섹션을 참조하세요.

이 예제 버킷 ACL은 퍼블릭 읽기 액세스 권한을 부여합니다.

```
{
```

```
    "Owner": {
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
            },
            "Permission": "READ"
        }
    ]
}
```

다음 예시 `put-bucket-ownership-controls` AWS CLI 명령은 객체 소유권에 대해 버킷 소유자 적용 설정을 적용합니다.

```
aws s3api put-bucket-ownership-controls --bucket amzn-s3-demo-bucket --ownership-
controls Rules=[{ObjectOwnership=BucketOwnerEnforced}]
```

버킷 ACL이 퍼블릭 읽기 액세스 권한을 부여하기 때문에 요청이 실패하고 다음 오류 코드를 반환합니다.

PutBucketOwnershipControls 작업을 호출할 때 오류(InvalidBucketAclWithObjectOwnership)가 발생했습니다. 버킷은 ObjectOwnership의 BucketOwnerEnforced 설정으로 ACL을 설정할 수 없습니다.

Amazon S3의 보안

AWS에서는 클라우드 보안을 가장 중요하게 생각합니다. AWS 고객은 보안에 가장 보안에 민감한 조직의 요구 사항에 부합하도록 빌드된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 AWS과 사용자의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드 내 보안 및 클라우드의 보안으로 설명합니다.

클라우드의 보안

AWS는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다. AWS 또한 안전하게 사용할 수 있는 서비스를 제공합니다. 서드 파티 감사자는 정기적으로 AWS 규정 준수 프로그램의 일환으로 보안 효과를 테스트하고 검증합니다. Amazon S3에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 AWS규정 준수 프로그램 제공 범위 내 서비스를 참조하십시오.

클라우드의 보안

귀하의 책임은 귀하가 사용하는 AWS 서비스에 의해 결정됩니다. 또한 데이터의 민감도, 조직의 요건 및 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다. Amazon S3에 대한 고객의 책임에는 다음 영역이 포함됩니다.

- 객체 소유권 및 암호화를 포함한 데이터 관리.
- 자산 분류
- IAM 역할과 적절한 권한을 적용하기 위한 기타 서비스 구성을 사용하여 데이터에 대한 액세스를 관리합니다.
- AWS CloudTrail 또는 Amazon S3용 Amazon GuardDuty와과 같은 탐지 제어 사용 설정.

이 설명서는 Amazon S3 사용 시 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목적에 맞게 Amazon S3를 구성하는 방법을 보여줍니다. 또한, Amazon S3 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스의 사용 방법을 배우게 됩니다.

> **Note**
>
> Amazon S3 Express One Zone 스토리지 클래스를 디렉터리 버킷과 함께 사용하는 방법에 대한 자세한 내용은 S3 Express One Zone 및 디렉터리 버킷 작업 섹션을 참조하세요.

주제

- Amazon S3의 보안 모범 사례
- Amazon S3의 데이터 보호
- 암호화로 데이터 보호
- 인터네트워크 트래픽 개인 정보
- Amazon S3에 대한 규정 준수 확인
- Amazon S3의 복원성
- Amazon S3의 인프라 보안
- Amazon S3의 구성 및 취약성 분석
- 액세스 관리

Amazon S3의 보안 모범 사례

Amazon S3는 자체 보안 정책을 개발하고 구현할 때 고려해야 할 여러 보안 기능을 제공합니다. 다음 모범 사례는 일반적인 지침이며 완벽한 보안 솔루션을 나타내지는 않습니다. 이러한 모범 사례는 환경에 적절하지 않거나 충분하지 않을 수 있으므로 지침이라기보다는 권장 사항으로만 사용해 주십시오.

주제

- Amazon SNS 보안 모범 사례
- Amazon S3 모니터링 및 감사 모범 사례
- 관리형 AWS 보안 서비스를 사용하여 데이터 보안 모니터링

Amazon SNS 보안 모범 사례

다음과 같은 Amazon S3 모범 사례를 통해 보안 사고를 예방할 수 있습니다.

액세스 제어 목록(ACL) 비활성화

S3 객체 소유권은 버킷에 업로드되는 객체의 소유권을 제어하고 ACL을 비활성화 또는 활성화하는 데 사용할 수 있는 Amazon S3 버킷 수준 설정입니다. 기본적으로 객체 소유권은 버킷 소유자 적용 설정으로 설정되며 모든 ACL이 비활성화되어 있습니다. ACL이 비활성화되면 버킷 소유자는 버킷의 모든 객체를 소유하고 액세스 관리 정책을 사용하여 데이터에 대한 액세스를 독점적으로 관리합니다.

Amazon S3의 현대적인 사용 사례 대부분은 더 이상 액세스 제어 목록(ACL) 사용을 요구하지 않습니다. 각 객체에 대해 액세스를 개별적으로 제어해야 하는 드문 상황을 제외하고는 ACL을 비활성화하는 것이 좋습니다. ACL을 비활성화하고 버킷 내 모든 객체의 소유권을 얻으려면 S3 객체 소유권에 버킷 소유자 적용 설정을 적용합니다. ACL을 사용 중지하면 다른 AWS 계정이 업로드한 객체로 버킷을 쉽게 유지 관리할 수 있습니다.

ACL이 비활성화되면 데이터에 대한 액세스 제어가 다음과 같은 정책을 기반으로 합니다.

- AWS Identity and Access Management(IAM) 사용자 정책
- S3 버킷 정책
- 가상 프라이빗 클라우드(VPC) 엔드포인트 정책
- AWS Organizations 서비스 제어 정책(SCP)
- AWS Organizations 리소스 제어 정책(RCP)

ACL을 비활성화하면 권한 관리와 감사가 단순화됩니다. 신규 버킷에는 기본적으로 ACL이 비활성화됩니다. 기존 버킷에 대해서도 ACL을 비활성화할 수 있습니다. 이미 객체가 있는 기존 버킷이 있는 경우 ACL을 비활성화하면 객체 및 버킷 ACL이 더 이상 액세스 평가 프로세스에 포함되지 않습니다. 대신 정책에 따라 액세스가 허용되거나 거부됩니다.

ACL을 사용하려면 다음을 수행해야 합니다.

- 버킷 정책을 검토하여 계정 외부에서 버킷에 대한 액세스 권한을 부여하려는 모든 방법을 포함하는지 확인합니다.
- 버킷 ACL을 기본값으로 재설정합니다(버킷 소유자에 대한 전체 제어 권한).

ACL을 비활성화하면 다음과 같은 동작이 발생합니다.

- 버킷은 ACL을 지정하지 않은 PUT 요청이나 버킷 소유자의 전체 제어 ACL이 있는 PUT 요청만 수락합니다. 이러한 ACL에는 `bucket-owner-full-control` 미리 준비된 ACL 또는 XML로 표현되는 이와 동등한 형식의 ACL이 포함됩니다.
- 버킷 소유자 전체 제어 ACL을 지원하는 기존 애플리케이션은 영향을 받지 않습니다.
- 다른 ACL(예: 특정 AWS 계정에 대한 사용자 정의 권한 부여)이 포함된 PUT 요청은 실패하고 오류 코드 `AccessControlListNotSupported`와 함께 `400 (Bad Request)` 오류를 반환합니다.

자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

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 하십시오. 퍼블릭 액세스를 차단하기 위해 다음과 같이 몇 가지 단계를 수행할 수 있습니다.

- S3 퍼블릭 액세스 차단을 사용합니다. S3 퍼블릭 액세스 차단을 사용하면 중앙 집중식 제어를 쉽게 설정하여 Amazon S3 리소스에 대한 퍼블릭 액세스를 제한할 수 있습니다. 이러한 중앙 집중식 제어는 리소스 생성 방식과 관계없이 적용됩니다. 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하세요.
- `"Principal": "*"`(사실상 '누구나'를 의미함)과 같은 와일드카드 ID를 허용하는 Amazon S3 버킷 정책을 식별합니다. 와일드카드 작업 `"*"`(Amazon S3 버킷에서 작업을 수행하도록 사용자에게 효과적으로 허용)를 허용하는 정책을 찾습니다.
- 마찬가지로 '모든 사용자'나 '인증된 모든 AWS 사용자'에게 읽기, 쓰기 또는 모든 액세스 권한을 제공하는 Amazon S3 버킷 액세스 제어 목록(ACL)을 찾습니다.

- `ListBuckets` API 작업을 사용하여 모든 Amazon S3 버킷을 스캔합니다. 그런 다음 `GetBucketAcl`, `GetBucketWebsite` 및 `GetBucketPolicy`를 사용하여 버킷에 규정을 준수하는 액세스 제어와 규정을 준수하는 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 규칙을 사용하여 지속적인 탐지 제어를 구현하는 것을 고려해 보십시오.

자세한 내용은 Amazon S3의 ID 및 액세스 관리 섹션을 참조하세요.

최소 권한 액세스 구현

권한을 부여할 때 누가 어떤 Amazon S3 리소스에 대해 어떤 권한을 갖는지 결정합니다. 해당 리소스에서 허용할 작업을 사용 설정합니다. 따라서 작업을 수행하는 데 필요한 권한만 부여하는 것을 권장합니다. 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.

- Amazon S3의 정책 작업 및 IAM 엔터티의 권한 범위
- Amazon S3에서 IAM을 사용하는 방법
- ACL(액세스 제어 목록) 개요

앞에 나온 메커니즘 중 하나 이상을 선택할 때 고려할 사항에 관한 지침은 Amazon S3의 ID 및 액세스 관리 단원을 참조하십시오.

Amazon S3 액세스가 필요한 애플리케이션 및 AWS 서비스에 IAM 역할 사용

Amazon EC2 또는 다른 AWS 서비스에서 실행되는 애플리케이션이 Amazon S3 리소스에 액세스하기 위해서는 AWS API 요청에 유효한 AWS 자격 증명이 있어야 합니다. AWS 자격 증명을 애플리케이션이나 Amazon EC2 인스턴스에 직접 저장하지 않는 것을 권장합니다. 이러한 보안 인증은 자동으로 교체되지 않으며 손상된 경우 비즈니스에 큰 영향을 줄 수 있는 장기 보안 인증입니다.

그 대신 IAM 역할을 사용하여 Amazon S3에 액세스해야 하는 애플리케이션이나 서비스의 임시 자격 증명을 관리하십시오. 역할을 사용할 때 Amazon EC2 인스턴스나 AWS 서비스(예: AWS Lambda)에 장기 자격 증명(예: 사용자 이름과 암호 또는 액세스 키)를 배포할 필요가 없습니다. 애플리케이션에서 다른 AWS 리소스를 호출할 때 사용할 수 있는 임시 권한을 역할이 제공합니다.

자세한 내용은 IAM 사용 설명서에서 다음 주제를 참조하십시오.

- IAM 역할

- 역할에 대한 일반적인 시나리오: 사용자, 애플리케이션 및 서비스

유휴 데이터 암호화 고려

Amazon S3에서 유휴 데이터를 보호하는 다음과 같은 옵션이 있습니다.

- 서버 측 암호화 - 모든 Amazon S3 버킷에는 기본적으로 암호화가 구성되어 있으며 S3 버킷에 업로드되는 신규 객체는 모두 저장 시 자동으로 암호화됩니다. Amazon S3 관리형 키(SSE-S3)를 사용한 서버 측 암호화가 Amazon S3 내 모든 버킷의 기본 암호화 구성입니다. 다른 유형의 암호화를 사용하려면 S3 PUT 요청에 사용할 서버 측 암호화 유형을 지정하거나 대상 버킷에 기본 암호화 구성을 설정할 수 있습니다.

  또한 Amazon S3는 다음과 같은 서버 측 암호화 옵션을 제공합니다.

  - AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 사용한 서버 측 암호화
  - AWS Key Management Service(AWS KMS) 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)
  - 고객 제공 키를 사용한 서버 측 암호화(SSE-C)

  자세한 내용은 서버 측 암호화를 사용하여 데이터 보호 섹션을 참조하세요.
- 클라이언트측 암호화 – 클라이언트 측에서 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 섹션을 참조하세요.

전송 중인 데이터의 암호화 강제 시행

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 IAM 정책에 aws:SecureTransport 조건을 사용하여 HTTPS(TLS)를 통해 암호화된 연결만 허용하는 것을 권장합니다. 자세한 내용은 예시 S3 버킷 정책 HTTP 또는 HTTPS 요청 기반 액세스 관리를 참조하세요. HTTP 요청을 거부하는 것 외에도 콘텐츠에 HTTP 액세스 시도가 이루어지는 경우 알림을 보내도록 `tlsDetails.tlsVersion NOT EXISTS`에 대해 Amazon CloudWatch 경보를 설정하는 것이 좋습니다. Amazon CloudWatch 경보를 구성하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 Creating CloudWatch alarms for CloudTrail events: examples 및 CloudTrail record contents를 참조하세요.

> **Important**
>
> AWS는 공개적으로 신뢰할 수 있는 인증서의 고정을 지원하지 않으므로 애플리케이션에서 Amazon S3 TLS 인증서를 고정하지 않는 것이 좋습니다. S3는 인증서를 자동으로 갱신하므로 인증서 만료 전에는 언제든지 인증서를 갱신할 수 있습니다. 인증서를 갱신하면 새로운 퍼블릭-프라이빗 키 페어가 생성됩니다. 최근에 새 공개 키로 갱신된 S3 인증서를 고정하면 애플리케이션에서 새 인증서를 사용할 때까지 S3에 연결할 수 없습니다.

또한 s3-bucket-ssl-requests-only 관리형 AWS Config 규칙을 사용하여 지속적인 탐지 제어를 구현하는 것을 고려해 보세요.

S3 객체 잠금 고려

S3 객체 잠금을 사용하면 Write Once Read Many(WORM) 모델을 사용하여 객체를 저장할 수 있습니다. S3 객체 잠금은 부적절하거나 실수로 인한 데이터 삭제를 예방하는 데 도움이 됩니다. 예를 들어 S3 객체 잠금을 사용하여 AWS CloudTrail 로그를 보호할 수 있습니다.

자세한 내용은 Object Lock으로 객체 잠금 섹션을 참조하세요.

S3 버전 관리 활성화

S3 버전 관리는 동일 버킷 내에 여러 개의 객체 변형을 보유하는 것을 의미합니다. 버전 관리를 사용하여 버킷에 저장된 모든 버전의 객체를 모두 보존, 검색 및 복원할 수 있습니다. 또한 의도치 않은 사용자 작업 및 애플리케이션 장애로부터 쉽게 복구할 수 있습니다.

또한 s3-bucket-versioning-enabled 관리형 AWS Config 규칙을 사용하여 지속적인 탐지 제어를 구현하는 것을 고려해 보세요.

자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하세요.

S3 교차 리전 복제 사용 고려

Amazon S3는 기본적으로 여러 지역에 있는 다양한 가용 영역에 데이터를 저장하지만 규정 준수 요구 사항에 따라 훨씬 먼 거리에 데이터를 저장해야 할 수도 있습니다. 교차 리전 복제(CRR)를 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 리전 내 및 리전 간 객체 복제 섹션을 참조하세요.

> **Note**
>
> CRR을 사용할 경우 원본 및 대상 S3 버킷 둘 다 버전 관리가 활성화되어 있어야 합니다.

또한 s3-bucket-replication-enabled 관리형 AWS Config 규칙을 사용하여 지속적인 탐지 제어를 구현하는 것을 고려해 보십시오.

Amazon S3 액세스에 대해 VPC 엔드포인트 사용 고려

Amazon S3용 가상 프라이빗 클라우드(VPC) 엔드포인트는 VPC 내의 논리적 엔터티로서, Amazon S3에만 연결을 허용합니다. VPC 엔드포인트는 트래픽이 개방형 인터넷을 통과하는 것을 방지하는 데 도움이 될 수 있습니다.

Amazon S3용 VPC 엔드포인트는 Amazon S3 데이터에 대한 액세스를 제어하는 다양한 방법을 제공합니다.

- S3 버킷 정책을 사용하여 특정 VPC 엔드포인트를 통해 허용되는 요청, 사용자 또는 그룹을 제어할 수 있습니다.
- S3 버킷 정책을 사용하여 S3 버킷에 대한 액세스를 갖게 되는 VPC 또는 VPC 종단점을 제어할 수 있습니다.
- 인터넷 게이트웨이가 없는 VPC를 사용하여 데이터 유출을 방지할 수 있습니다.

자세한 내용은 버킷 정책을 사용하여 VPC 엔드포인트에서 액세스 제어 섹션을 참조하세요.

관리형 AWS 보안 서비스를 사용하여 데이터 보안 모니터링

여러 관리형 AWS 보안 서비스를 통해 Amazon S3 데이터의 보안 및 규정 준수 위험을 식별, 평가 및 모니터링할 수 있습니다. 이러한 서비스는 이러한 위험으로부터 데이터를 보호하는 데도 도움이 될 수 있습니다. 이러한 서비스에는 단일 AWS 계정의 Amazon S3 리소스에서 수천 개의 계정에 걸친 조직을 위한 리소스로 확장하도록 설계된 자동화된 탐지, 모니터링 및 보호 기능이 포함됩니다.

자세한 내용은 관리형 AWS 보안 서비스를 사용하여 데이터 보안 모니터링 섹션을 참조하세요.

Amazon S3 모니터링 및 감사 모범 사례

다음과 같은 Amazon S3 모범 사례는 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 대해 조치를 취해야 합니다. 리소스를 감사하려면 다음을 수행하는 것이 좋습니다.

- Tag Editor를 사용하여 보안이나 감사에 민감한 리소스를 식별하고 태그를 지정한 후, 이 리소스를 검색해야 할 때 태그를 이용하십시오. 자세한 내용은 AWS 리소스 태그 지정 사용 설명서의 태그를 지정할 리소스 검색을 참조하십시오.

- S3 인벤토리를 사용하여 비즈니스, 규정 준수 및 규제 요건에 대한 객체의 복제 및 암호화 상태를 감사하고 보고합니다. 자세한 내용은 S3 Inventory를 사용한 데이터 카탈로그화 및 분석 섹션을 참조하세요.
- Amazon S3 리소스에 대한 Resource Groups을 생성합니다. Resource Groups에 대한 자세한 내용은 AWS Resource Groups 사용 설명서의 Resource Groups란 무엇인가요?를 참조하십시오.

AWS 모니터링 도구를 사용하여 모니터링 구현

모니터링은 Amazon S3 및 AWS 솔루션의 안정성, 보안, 가용성 및 성능을 유지하는 데 중요한 부분입니다. AWS에서는 Amazon S3 및 다른 AWS 서비스를 모니터링할 수 있는 여러 가지 도구와 서비스를 제공합니다. 예를 들어 Amazon S3에 대한 Amazon CloudWatch 지표(특히 `PutRequests`, `GetRequests`, `4xxErrors` 및 `DeleteRequests` 지표)를 모니터링할 수 있습니다. 자세한 내용은 Amazon CloudWatch를 사용한 지표 모니터링 및 Amazon S3의 로깅 및 모니터링(을)를 참조하세요.

두 번째 예는 예제: Amazon S3 버킷 작업을 참조하십시오. 이 예에서는 버킷 정책, 버킷 수명 주기 또는 버킷 복제 구성을 PUT 또는 `DELETE`하거나 버킷 ACL을 PUT하기 위해 Amazon S3 API가 호출될 때 트리거되는 CloudWatch 경보를 생성하는 방법을 설명합니다.

Amazon S3 서버 액세스 로깅 사용

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 사용 설정에 대한 자세한 내용은 서버 액세스 로깅을 사용한 요청 로깅 섹션을 참조하십시오.

또한 s3-bucket-logging-enabled AWS Config 관리형 규칙을 사용하여 지속적인 탐지 제어를 구현하는 것을 고려해 보십시오.

AWS CloudTrail 사용

AWS CloudTrail은 Amazon S3에서 사용자, 역할 또는 AWS 서비스가 수행한 작업에 대한 레코드를 제공합니다. CloudTrail에서 수집한 정보를 사용하여 다음을 확인할 수 있습니다.

- Amazon S3에 대한 요청
- 요청을 보낸 IP 주소
- 요청한 사람
- 요청을 한 시간
- 요청에 대한 추가 세부 정보

예를 들어 데이터 액세스에 영향을 주는 PUT 작업(특히 `PutBucketAcl`, `PutObjectAcl`, `PutBucketPolicy` 및 `PutBucketWebsite`)의 CloudTrail 항목을 식별할 수 있습니다.

AWS 계정을 설정하면 CloudTrail이 기본적으로 사용됩니다. CloudTrail 콘솔에서 최근 이벤트를 볼 수 있습니다. Amazon S3 버킷에 대한 활동 및 이벤트에 대한 지속적인 레코드를 생성하려면 CloudTrail 콘솔에서 추적을 생성하면 됩니다. 자세한 내용을 알아보려면 AWS CloudTrail 사용 설명서의 데이터 이벤트 로깅을 참조하십시오.

추적을 생성할 때 데이터 이벤트를 로그로 기록하도록 CloudTrail을 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. CloudTrail은 `GetObject`, `DeleteObject` 및 `PutObject`와 같은 Amazon S3 객체 수준 API 작업의 하위 집합을 지원합니다. CloudTrail이 Amazon S3에서 작동하는 방식에 대한 자세한 내용은 AWS CloudTrail을 사용하여 Amazon S3 API 호출 로깅 단원을 참조하십시오. Amazon S3 콘솔에서 S3 버킷을 S3 버킷 및 객체에 대한 CloudTrail 이벤트 로깅 사용 설정에 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 개발자 안내서의 cloudtrail-s3-dataevents-enabled 섹션을 참조하십시오.

사용 설정AWS Config

이 주제에 나오는 여러 가지 모범 사례는 AWS Config 규칙 생성을 제안합니다. AWS Config를 사용하면 AWS 리소스의 구성을 평가, 감사 및 측정하는 데 도움이 됩니다. AWS Config는 리소스 구성을 모니터링하여 원하는 보안 구성을 기준으로 기록된 구성을 평가할 수 있게 해줍니다. AWS Config을 사용하여 다음 작업을 할 수 있습니다.

- 구성 변경 사항 및 AWS 리소스 간의 관계 검토
- 자세한 리소스 구성 기록 조사
- 내부 지침에 지정된 구성을 기준으로 전반적인 준수 여부를 확인합니다.

AWS Config를 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.

> **Important**
>
> AWS Config 관리형 규칙은 Amazon S3 리소스를 평가할 때 범용 버킷만 지원합니다. AWS Config는 디렉터리 버킷의 구성 변경을 기록하지 않습니다. 자세한 내용은 AWS Config 개발자 안내서의 AWS Config 관리형 규칙 및 AWS Config 관리형 규칙 목록을 참조하세요.

AWS Config를 사용하는 방법의 예시를 보려면 AWS 보안 블로그의 AWS Config를 사용하여 퍼블릭 액세스를 허용하는 Amazon S3 버킷을 모니터링하고 이에 응답하는 방법을 참조하십시오.

S3 스토리지 렌즈 사용

S3 스토리지 렌즈는 조직 전반에서 객체 스토리지 사용 및 활동에 대한 가시성을 확보하는 데 사용할 수 있는 클라우드 스토리지 분석 기능입니다. S3 스토리지 렌즈는 또한 지표를 분석해 스토리지 비용을 최적화하고 데이터 보호에 대한 모범 사례를 적용하는 데 사용할 수 있는 상황별 권장 사항을 제공합니다.

S3 Storage Lens의 지표를 사용하여 조직 전반의 스토리지 용량 또는 가장 빠르게 증가하는 버킷과 접두사 등의 요약 인사이트를 생성할 수 있습니다. 또한 S3 스토리지 렌즈 지표를 사용하여 비용 최적화 기회를 식별하고, 데이터 보호 및 액세스 관리 모범 사례를 구현하고, 애플리케이션 워크로드의 성능을 개선할 수 있습니다.

예를 들어 S3 수명 주기 규칙이 없는 버킷을 식별하여 7일이 넘게 경과한 미완료 멀티파트 업로드를 중단할 수 있습니다. 또한 S3 복제 또는 S3 버전 관리 사용과 같은 데이터 보호 모범 사례를 따르지 않는 버킷을 식별할 수 있습니다. 자세한 내용은 Amazon S3 스토리지 렌즈 이해를 참조하십시오.

AWS 보안 공지 모니터링

AWS 계정의 Trusted Advisor에 게시되는 보안 권고 사항을 정기적으로 확인하는 것이 좋습니다. 특히 '공개 액세스 권한'이 있는 Amazon S3 버킷에 대한 경고를 찾아보십시오. 이 작업은 describe-trusted-advisor-checks를 사용하여 프로그래밍 방식으로 수행할 수 있습니다.

뿐만 아니라, 각 AWS 계정에 등록된 기본 이메일 주소를 적극적으로 모니터링하십시오. 사용자에게 영향을 줄 수 있는 보안 문제가 생기면 AWS에서 이 이메일 주소를 사용하여 연락 드립니다.

널리 영향을 미치는 AWS 운영 문제는 AWS Health Dashboard - 서비스 상태에 게시됩니다. AWS Health Dashboard를 통해 개별 계정에도 운영 문제가 게시됩니다. 자세한 내용은 AWS Health 설명서를 참조하십시오.

관리형 AWS 보안 서비스를 사용하여 데이터 보안 모니터링

여러 관리형 AWS 보안 서비스를 통해 Amazon S3 데이터의 보안 및 규정 준수 위험을 식별, 평가 및 모니터링할 수 있습니다. 이러한 서비스는 해당 위험으로부터 데이터를 보호하는 데도 도움이 될 수 있습니다. 이러한 서비스에는 단일 AWS 계정의 Amazon S3 리소스에서 수천 개의 AWS 계정에 걸친 조직을 위한 리소스로 확장하도록 설계된 자동화된 탐지, 모니터링 및 보호 기능이 포함됩니다.

AWS 탐지 및 대응 서비스를 사용하면 잠재적인 보안 구성 오류, 위협 또는 예상치 못한 동작을 식별하여 사용자 환경에서 잠재적인 무단 활동 또는 악의적인 활동에 신속하게 대응할 수 있습니다. AWS 데이터 보호 서비스를 사용하면 데이터, 계정 및 워크로드를 모니터링하고 무단 액세스로부터 보호할 수 있습니다. 또한 Amazon S3 데이터 자산에서 개인 식별 정보(PII)와 같은 민감한 데이터를 검색하는 데 도움이 될 수 있습니다.

관리형 AWS 보안 서비스는 데이터 보안 및 규정 준수 위험을 식별하고 평가하는 데 도움이 되도록 조사 결과를 생성하여 Amazon S3 데이터와 관련된 잠재적 보안 이벤트 또는 문제를 알려줍니다. 조사 결과는 인시던트 대응 워크플로 및 정책에 따라 이러한 위험을 조사 및 평가하고 조치를 취하는 데 사용할 수 있는 관련 세부 정보를 제공합니다. 각 서비스를 사용하여 조사 결과 데이터에 직접 액세스할 수 있습니다. 또한 보안 인시던트 및 이벤트 관리 시스템(SIEM)과 같은 다른 애플리케이션, 서비스 및 시스템으로 데이터를 보낼 수도 있습니다.

Amazon S3 데이터의 보안을 모니터링하려면 이러한 관리형 AWS 보안 서비스를 사용해 보십시오.

Amazon GuardDuty

Amazon GuardDuty는 AWS 계정 및 워크로드에 악의적인 활동이 있는지 지속적으로 모니터링하고 가시성 및 해결을 위한 상세한 보안 조사 결과를 제공하는 위협 탐지 서비스입니다.

GuardDuty의 S3 보호 기능을 사용하면 GuardDuty를 구성하여 Amazon S3 리소스에 대한 AWS CloudTrail 관리 및 데이터 이벤트를 분석할 수 있습니다. 그러면 GuardDuty는 해당 이벤트를 모니터링하여 악의적이고 의심스러운 활동이 있는지 확인합니다. GuardDuty는 분석 결과를 알리고 잠재적인 보안 위험을 식별하기 위해 위협 인텔리전스 피드와 기계 학습을 사용합니다.

GuardDuty는 Amazon S3 리소스에 대한 다양한 종류의 활동을 모니터링할 수 있습니다. 예를 들어, Amazon S3용 CloudTrail 관리 이벤트에는 `ListBuckets`, `DeleteBucket` 및 `PutBucketReplication`과 같은 버킷 수준 작업이 포함됩니다. Amazon S3용 CloudTrail 데이터 이벤트에는 `GetObject`, `ListObjects` 및 `PutObject`와 같은 객체 수준 작업이 포함됩니다. GuardDuty가 비정상적이거나 잠재적으로 악의적인 활동을 탐지하면 조사 결과를 생성하여 사용자에게 알립니다.

자세한 내용은 Amazon GuardDuty 사용 설명서의 Amazon GuardDuty에서 Amazon S3 보호를 참조하십시오.

Amazon Detective

Amazon Detective는 조사 프로세스를 간소화하고 더 빠르고 효과적인 보안 조사를 수행할 수 있도록 도와줍니다. Detective는 가능한 보안 문제의 특성과 범위를 분석하고 평가하는 데 도움이 되는 사전 빌드된 데이터 집계, 요약 및 컨텍스트를 제공합니다.

Detective는 AWS CloudTrail의 API 호출 및 AWS 리소스의 Amazon VPC 흐름 로그와 같은 시간 기반 이벤트를 자동으로 추출합니다. 또한 Amazon GuardDuty에서 생성한 결과를 수집합니다. 그런 다음 Detective는 기계 학습, 통계 분석 및 그래프 이론을 사용하여 효과적인 보안 조사를 보다 신속하게 수행하는 데 도움이 되는 시각화를 생성합니다.

이러한 시각화는 리소스 동작과 시간 경과에 따른 리소스 동작 간의 상호 작용에 대한 통합된 대화형 보기를 제공합니다. 이 동작 그래프를 탐색하여 실패한 로그인 시도 또는 의심스러운 API 호출과 같은 잠재적으로 악의적인 작업을 검사할 수 있습니다. 또한 이러한 작업이 S3 버킷이나 객체와 같은 리소스에 미치는 영향을 확인할 수 있습니다.

자세한 내용은 Amazon Detective 관리 가이드를 참조하십시오.

IAM Access Analyzer

AWS Identity and Access Management Access Analyzer(IAM Access Analyzer)를 사용하면 외부 엔터티와 공유되는 조직의 리소스 및 계정을 식별할 수 있습니다. 또한 IAM Access Analyzer를 사용하여 정책 문법 및 모범 사례에 따라 IAM 정책을 검증하고 AWS CloudTrail 로그의 액세스 활동을 기반으로 IAM 정책을 생성할 수 있습니다.

IAM Access Analyzer는 논리 기반 추론을 사용하여 사용자 AWS 환경의 리소스 정책(예: 버킷 정책)을 분석합니다. IAM Access Analyzer for S3를 사용하면 인터넷의 모든 사용자 또는 조직 외부의 계정을 포함한 다른 AWS 계정이 액세스할 수 있게 S3 버킷이 구성되면 알림을 받게 됩니다. 예를 들어 IAM Access Analyzer for S3는 버킷에 버킷 액세스 제어 목록(ACL), 버킷 정책, 다중 리전 액세스 포인트 정책 또는 액세스 포인트 정책을 통해 제공된 읽기 또는 쓰기 권한이 있음을 보고할 수 있습니다. 각 퍼블릭 버킷 또는 공유 버킷에 대해 퍼블릭 액세스 또는 공유 액세스의 수준과 소스를 알려주는 결과가 수신됩니다. 이러한 내용을 바탕으로 즉각적이고 정확한 시정 조치를 취하여 버킷 액세스를 원하는 대로 복원할 수 있습니다.

자세한 내용은 IAM Access Analyzer for S3를 사용하여 버킷 액세스 검토 섹션을 참조하세요.

Amazon Macie

Amazon Macie는 기계 학습 및 패턴 일치를 사용하여 민감한 데이터를 검색하는 보안 서비스입니다. Macie는 데이터 보안 위험에 대한 가시성을 제공하고 이러한 위험에 대한 자동화된 보호를 지

원합니다. Macie를 사용하면 Amazon S3 데이터 자산에서 민감한 데이터의 검색 및 보고를 자동화하여 조직이 S3에 저장하는 데이터를 더 잘 이해할 수 있습니다.

Macie를 사용하여 민감한 데이터를 감지하려면 규모가 크고 점점 더 늘어나는 많은 국가 및 리전에 대한 민감한 데이터 유형 목록을 탐지하도록 설계된 기본 제공 기준 및 기법을 사용할 수 있습니다. 이러한 민감한 데이터 유형에는 여러 유형의 개인 식별 정보(PII), 재무 데이터 및 보안 인증 데이터가 포함됩니다. 직접 정의하는 사용자 지정 기준을 사용할 수도 있습니다. 즉, 일치시킬 텍스트 패턴을 정의하는 정규 표현식 및 원하는 경우 결과를 세분화하는 문자 시퀀시 및 근접성 규칙을 사용할 수 있습니다.

Macie가 S3 객체에서 민감한 데이터를 탐지하면 Macie는 보안 조사 결과를 생성하여 사용자에게 알립니다. 이 조사 결과는 영향을 받는 객체에 대한 정보, Macie가 발견한 민감한 데이터의 유형 및 발생 횟수, 영향을 받는 S3 버킷 및 객체를 조사하는 데 도움이 되는 추가 세부 정보를 제공합니다. 자세한 내용은 Amazon Macie 사용 설명서를 참조하십시오.

AWS Security Hub

AWS Security Hub는 보안 모범 사례 검사를 수행하고, 여러 소스의 경고 및 조사 결과를 단일 형식으로 집계하고, 자동화된 문제 해결을 지원하는 보안 태세 관리 서비스입니다.

Security Hub는 통합 AWS Partner Network 보안 솔루션과 Amazon Detective, Amazon GuardDuty, IAM Access Analyzer 및 Amazon Macie를 비롯한 AWS 서비스에서 보안 조사 결과 데이터를 수집하고 제공합니다. 또한 AWS 모범 사례 및 지원되는 업계 표준을 기반으로 자동화된 보안 검사를 지속적으로 실행하여 자체 결과를 생성합니다.

그런 다음 Security Hub는 가장 중요한 조사 결과의 우선순위를 지정하고 처리할 수 있도록 공급자 간 결과를 상호 연관시키고 통합합니다. 또한 특정 조사 결과 클래스에 대한 응답 또는 수정 조치를 호출하는 데 사용할 수 있는 사용자 지정 작업에 대한 지원도 제공합니다.

Security Hub를 사용하면 Amazon S3 리소스의 보안 및 규정 준수 상태를 평가할 수 있습니다. 이는 개별 AWS 리전 및 여러 리전에 걸친 조직의 보안 태세에 대한 광범위한 분석의 일환으로 수행할 수 있습니다. 여기에는 보안 추세를 분석하고 우선순위가 가장 높은 보안 문제를 식별하는 것이 포함됩니다. 또한 여러 AWS 리전의 조사 결과를 집계하고 집계된 결과 데이터를 단일 리전에서 모니터링 및 처리할 수 있습니다.

자세한 내용은 AWS Security Hub 사용 설명서의 Amazon Simple Storage Service 제어를 참조하십시오.

Amazon S3의 데이터 보호

Amazon S3에서는 미션 크리티컬 및 기본 데이터 스토리지에 적합하게 설계된, 내구성이 뛰어난 스토리지 인프라를 제공합니다. S3 Standard, S3 Intelligent-Tiering, S3 Standard-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive는 AWS 리전의 최소 3개 가용 영역에 걸쳐 여러 디바이스에 객체를 중복 저장합니다. 가용 영역은 AWS 리전에 중복 전원, 네트워킹 및 연결이 있는 하나 이상의 개별 데이터 센터입니다. 가용 영역은 모두 서로 100km 내에 있지만 다른 가용 영역과 의미 있는 거리(수 킬로미터)만큼 물리적으로 분리됩니다. S3 One Zone-IA 스토리지 클래스는 단일 가용 영역 내의 여러 디바이스에 걸쳐 데이터를 중복으로 저장합니다. 이러한 서비스는 손실된 중복성을 신속하게 탐지하고 복구하여 동시 장치 오류를 처리하도록 설계되었으며 또한 체크섬을 사용하여 데이터 무결성을 정기적으로 확인합니다.

Amazon S3 Standard 스토리지는 다음 기능을 제공합니다.

- Amazon S3 서비스 수준 계약으로 신뢰성을 보장합니다.
- 지정된 기간 동안 객체에 대해 99.999999999%의 내구성과 99.99%의 가용성을 제공할 수 있도록 설계되었습니다.
- S3 Standard, S3 Intelligent-Tiering, S3 Standard-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive는 모두 전체 Amazon S3 가용 영역이 손실된 경우 데이터를 유지하도록 설계되었습니다.

Amazon S3에서는 버전 관리를 사용하여 데이터에 대한 향상된 보호를 제공합니다. 버전 관리를 사용하면 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.

S3 버전 관리 외에도 Amazon S3 객체 잠금 및 S3 복제를 사용하여 데이터를 보호할 수도 있습니다. 자세한 내용은 Amazon S3의 데이터 보호 섹션을 참조하세요.

데이터 보호를 위해 AWS 계정 자격 증명을 보호하고 AWS Identity and Access Management에서 개별 사용자 계정을 설정하여 각 사용자에게 직무를 수행하는 데 필요한 권한만 부여하는 것이 좋습니다.

명령행 인터페이스 또는 API를 통해 AWS에 액세스할 때 FIPS 140-2 검증된 암호화 모듈이 필요한 경우, FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-2를 참조하십시오.

다음 보안 모범 사례에서도 Amazon S3의 데이터 보호를 다룹니다.

- Implement server-side encryption
- Enforce encryption of data in transit
- Consider using Macie with Amazon S3
- Identify and audit all your Amazon S3 buckets
- Monitor Amazon Web Services security advisories

암호화로 데이터 보호

> **Important**
>
> 이제 Amazon S3가 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)를 Amazon S3 내 모든 버킷 암호화의 기본 수준으로 적용합니다. 2023년 1월 5일부터 Amazon S3로의 모든 새 객체 업로드는 추가 비용 없이 성능에 영향을 미치지 않고 자동으로 암호화됩니다. S3 버킷 기본 암호화 구성에 및 신규 객체 업로드에 대한 자동 암호화 상태는 AWS CloudTrail 로그, S3 인벤토리, S3 스토리지 렌즈, Amazon S3 콘솔에서 사용할 수 있으며, AWS Command Line Interface 및 AWS SDK에서 추가 Amazon S3 API 응답 헤더로도 사용할 수 있습니다. 자세한 내용은 기본 암호화 관련 FAQ를 참조하십시오.

데이터 보호란 전송 중(Amazon S3 안팎으로 데이터가 이동 중)과 저장 시(Amazon S3 데이터 센터의 디스크에 데이터가 저장된 동안) 데이터를 보호하는 것을 말합니다. Secure Socket Layer/Transport Layer Security(SSL/TLS)를 사용하거나 클라이언트측 암호화를 사용하여 전송 중 데이터를 보호할 수 있습니다. Amazon S3에서 저장 데이터를 보호하는 데는 다음과 같은 옵션이 있습니다.

- 서버 측 암호화 – AWS 데이터 센터의 디스크에 저장하기 전에 Amazon S3가 객체를 암호화하고 객체를 다운로드할 때 해독합니다.

  모든 Amazon S3 버킷에는 기본적으로 암호화가 구성되어 있으며 S3 버킷에 업로드되는 신규 객체는 모두 저장 시 자동으로 암호화됩니다. Amazon S3 관리형 키(SSE-S3)를 사용한 서버 측 암호화가 Amazon S3 내 모든 버킷의 기본 암호화 구성입니다. 다른 유형의 암호화를 사용하려면 S3 PUT 요청에 사용할 서버 측 암호화 유형을 지정하거나 대상 버킷에 기본 암호화 구성을 업데이트할 수 있습니다.

  PUT 요청에 다른 암호화 유형을 지정하려는 경우 AWS Key Management Service(AWS KMS) 키를 사용한 서버 측 암호화(SSE-KMS), AWS KMS 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)

또는 고객 제공 키를 사용한 서버 측 암호화 (SSE-C)를 사용할 수 있습니다. 대상 버킷에 다른 기본 암호화 구성을 설정하려는 경우 SSE-KMS 또는 DSSE-KMS를 사용할 수 있습니다.

범용 버킷의 기본 암호화 구성을 변경하는 방법에 대한 자세한 내용은 기본 암호화 구성 섹션을 참조하세요.

버킷의 기본 암호화 구성을 SSE-KMS로 변경해도 버킷에 있는 기존 Amazon S3 객체의 암호화 유형은 변경되지 않습니다. 기본 암호화 구성을 SSE-KMS로 업데이트한 후 기존 객체의 암호화 유형을 변경하려는 경우 Amazon S3 Batch Operations를 사용할 수 있습니다. 객체 목록을 S3 Batch Operations에 제공하면 Batch Operations가 각각의 API 작업을 직접적으로 호출합니다. 객체 복사 작업을 사용하면 기존 객체를 복사하여 SSE-KMS로 암호화된 객체와 동일한 버킷에 다시 쓸 수 있습니다. 단일 배치 작업 건으로 수십억 개의 객체에서 지정된 작업을 수행할 수 있습니다. 자세한 내용은 Batch Operations를 사용하여 대량으로 객체 작업 수행 및 AWS Storage Blog 게시물 How to retroactively encrypt existing objects in Amazon S3 using S3 Inventory, Amazon Athena, and S3 Batch Operations를 참조하세요.

서버 측 암호화의 각 옵션에 대한 자세한 정보는 서버 측 암호화를 사용하여 데이터 보호 섹션을 참조하세요.

서버 측 암호화를 구성하려면 다음을 참조하십시오.

- Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) 지정
- AWS KMS(SSE-KMS)를 사용한 서버 측 암호화 지정
- the section called “DSSE-KMS 지정”
- 고객 제공 키를 사용한 서버 측 암호화 지정(SSE-C).

- 클라이언트측 암호화 – 클라이언트 측에서 데이터를 암호화하여 암호화된 데이터를 Amazon S3에 업로드합니다. 이 경우 사용자가 암호화 프로세스, 암호화 키 및 관련 도구를 관리합니다.

  클라이언트 측 암호화를 구성하려면 클라이언트측 암호화를 사용하여 데이터 보호 단원을 참조하십시오 .

스토리지 바이트의 몇 퍼센트가 암호화되었는지 확인하려면 Amazon S3 스토리지 렌즈 지표를 사용하면 됩니다. S3 스토리지 렌즈는 조직 전반에서 객체 스토리지 사용 및 활동에 대한 가시성을 확보하는 데 사용할 수 있는 클라우드 스토리지 분석 기능입니다. 자세한 내용은 S3 스토리지 렌즈를 사용한 스토리지 활동 및 사용량 평가를 참조하십시오. 지표의 전체 목록은 S3 스토리지 렌즈 지표 용어집을 참조하세요.

서버 측 암호화 및 클라이언트측 암호화에 대한 자세한 내용은 다음 주제를 검토하십시오.

주제

- 서버 측 암호화를 사용하여 데이터 보호
- 클라이언트측 암호화를 사용하여 데이터 보호

서버 측 암호화를 사용하여 데이터 보호

> **Important**
>
> 이제 Amazon S3가 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)를 Amazon S3 내 모든 버킷 암호화의 기본 수준으로 적용합니다. 2023년 1월 5일부터 Amazon S3로의 모든 새 객체 업로드는 추가 비용 없이 성능에 영향을 미치지 않고 자동으로 암호화됩니다. S3 버킷 기본 암호화 구성에 및 신규 객체 업로드에 대한 자동 암호화 상태는 AWS CloudTrail 로그, S3 인벤토리, S3 스토리지 렌즈, Amazon S3 콘솔에서 사용할 수 있으며, AWS Command Line Interface 및 AWS SDK에서 추가 Amazon S3 API 응답 헤더로도 사용할 수 있습니다. 자세한 내용은 기본 암호화 관련 FAQ를 참조하십시오.

서버 측 암호화는 데이터를 받는 애플리케이션 또는 서비스에 의해 해당 대상에서 데이터를 암호화하는 것입니다. Amazon S3에서 AWS 데이터 센터의 디스크에 데이터를 쓰면서 객체 수준에서 데이터를 암호화하고 사용자가 해당 데이터에 액세스할 때 자동으로 암호를 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 API 작업은 암호화 여부와 관계없이 전체 객체의 목록을 반환합니다.

모든 Amazon S3 버킷에는 기본적으로 암호화가 구성되어 있으며 S3 버킷에 업로드되는 신규 객체는 모두 저장 시 자동으로 암호화됩니다. Amazon S3 관리형 키(SSE-S3)를 사용한 서버 측 암호화가 Amazon S3 내 모든 버킷의 기본 암호화 구성입니다. 다른 유형의 암호화를 사용하려면 S3 PUT 요청에 사용할 서버 측 암호화 유형을 지정하거나 대상 버킷에 기본 암호화 구성을 업데이트할 수 있습니다.

PUT 요청에 다른 암호화 유형을 지정하려는 경우 AWS Key Management Service(AWS KMS) 키를 사용한 서버 측 암호화(SSE-KMS), AWS KMS 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS) 또는 고객 제공 키를 사용한 서버 측 암호화 (SSE-C)를 사용할 수 있습니다. 대상 버킷에 다른 기본 암호화 구성을 설정하려는 경우 SSE-KMS 또는 DSSE-KMS를 사용할 수 있습니다.

범용 버킷의 기본 암호화 구성을 변경하는 방법에 대한 자세한 내용은 기본 암호화 구성 섹션을 참조하세요.

버킷의 기본 암호화 구성을 SSE-KMS로 변경해도 버킷에 있는 기존 Amazon S3 객체의 암호화 유형은 변경되지 않습니다. 기본 암호화 구성을 SSE-KMS로 업데이트한 후 기존 객체의 암호화 유형을 변경하려는 경우 Amazon S3 Batch Operations를 사용할 수 있습니다. 객체 목록을 S3 Batch Operations에 제공하면 Batch Operations가 각각의 API 작업을 직접적으로 호출합니다. 객체 복사 작업을 사용하면 기존 객체를 복사하여 SSE-KMS로 암호화된 객체와 동일한 버킷에 다시 쓸 수 있습니다. 단일 배치 작업 건으로 수십억 개의 객체에서 지정된 작업을 수행할 수 있습니다. 자세한 내용은 Batch Operations를 사용하여 대량으로 객체 작업 수행 및 AWS Storage Blog 게시물 How to retroactively encrypt existing objects in Amazon S3 using S3 Inventory, Amazon Athena, and S3 Batch Operations를 참조하세요.

> **Note**
>
> 동일한 객체에 서로 다른 서버 측 암호화 유형을 동시에 적용할 수는 없습니다.

기존 객체를 암호화해야 하는 경우 S3 배치 작업 및 S3 인벤토리를 사용하십시오. 자세한 내용은 Amazon S3 배치 작업으로 객체 암호화 및 Batch Operations를 사용하여 대량으로 객체 작업 수행 섹션을 참조하십시오.

Amazon S3에 데이터를 저장할 때는 선택한 암호화 키 관리 방법과 적용하려는 암호화 계층의 수에 따라 서버 측 암호화에 대해 네 가지 옵션을 독립적으로 사용할 수 있습니다.

Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)

모든 Amazon S3 버킷에는 기본적으로 암호화가 구성되어 있습니다. 서버 측 암호화의 기본 옵션은 Amazon S3 관리형 키(SSE-S3)를 사용하는 것입니다. 각 객체가 고유한 키로 암호화됩니다. 또한 추가 보안 조치로 SSE-S3는 주기적으로 교체되는 루트 키를 사용하여 키 자체를 암호화합니다. SSE-S3는 가장 강력한 블록 암호 중 하나인 256비트 Advanced Encryption Standard(AES-256)를 사용하여 데이터를 암호화합니다. 자세한 내용은 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) 사용 섹션을 참조하세요.

AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 사용한 서버 측 암호화

AWS KMS keys(SSE-KMS)를 사용한 서버 측 암호화는 Amazon S3와 AWS KMS 서비스를 통합하여 제공됩니다. AWS KMS를 사용하면 키에 대한 제어 기능이 더 많아집니다. 예를 들어, 개별 키를 보고, 제어 정책을 편집하고, AWS CloudTrail에서 키를 팔로우할 수 있습니다. 또한 고객 관리형 키를 생성하

고 관리하거나 사용자, 서비스 및 리전에 고유한 AWS 관리형 키를 사용할 수 있습니다. 자세한 내용은 AWS KMS 키를 사용한 서버 측 암호화(SSE-KMS) 사용 섹션을 참조하세요.

AWS Key Management Service(AWS KMS) 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)

AWS KMS keys를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)는 SSE-KMS와 비슷하지만 DSSE-KMS는 한 계층이 아닌 두 개의 개별 객체 수준 암호화 계층을 적용합니다. 두 암호화 계층 모두 서버 측 객체에 적용되므로 다양한 AWS 서비스 및 도구를 사용하여 S3의 데이터를 분석하는 동시에 규정 준수 요구 사항을 충족하는 암호화 방법을 사용할 수 있습니다. 자세한 내용은 AWS KMS 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS) 사용 섹션을 참조하세요.

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 키(SSE-C)로 서버 측 암호화 사용 섹션을 참조하세요.

> **Note**
>
> S3 액세스 포인트를 사용하는 Amazon FSx 파일 시스템에 액세스 포인트를 사용하는 경우 서버 측 암호화를 위한 한 가지 옵션이 있습니다.
>
> Amazon FSx를 사용한 서버 측 암호화(SSE-FSX)
>
> 모든 Amazon FSx 파일 시스템에는 기본적으로 암호화가 구성되어 있으며 AWS Key Management Service를 사용하여 관리형 키로 저장 시 암호화됩니다. 데이터는 파일 시스템에 데이터를 쓰고 읽을 때 파일 시스템에서 자동으로 암호화 및 복호화됩니다. 이러한 프로세스는 Amazon FSx에서 투명하게 처리됩니다.

Amazon S3 버킷에 대한 기본 서버 측 암호화 동작 설정

> **Important**
>
> 이제 Amazon S3가 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)를 Amazon S3 내 모든 버킷 암호화의 기본 수준으로 적용합니다. 2023년 1월 5일부터 Amazon S3로의 모든 새 객체 업로드는 추가 비용 없이 성능에 영향을 미치지 않고 자동으로 암호화됩니다. S3 버킷 기본 암호화 구성에 및 신규 객체 업로드에 대한 자동 암호화 상태는 AWS CloudTrail 로그, S3 인벤토리, S3 스토리지 렌즈, Amazon S3 콘솔에서 사용할 수 있으며, AWS Command Line

Interface 및 AWS SDK에서 추가 Amazon S3 API 응답 헤더로도 사용할 수 있습니다. 자세한 내용은 기본 암호화 관련 FAQ를 참조하십시오.

모든 Amazon S3 버킷에는 기본적으로 암호화가 구성되어 있으며, 객체는 Amazon S3 관리형 키(SSE-S3)를 통한 서버 측 암호화를 사용하여 자동으로 암호화됩니다. 이 암호화 설정은 Amazon S3 버킷의 모든 객체에 적용됩니다.

키 교체 및 액세스 정책 권한 부여 관리와 같이 키를 더 세밀하게 제어해야 하는 경우 AWS Key Management Service(AWS KMS) 키를 사용한 서버 측 암호화(SSE-KMS) 또는 AWS KMS 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)를 사용하는 방법도 있습니다. KMS 키 편집에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 키 편집을 참조하세요.

Note

새 객체 업로드를 자동으로 암호화하도록 버킷을 변경했습니다. 이전에 기본 암호화 없이 버킷을 생성한 경우 Amazon S3는 SSE-S3를 사용하여 버킷에 대해 기본적으로 암호화를 활성화합니다. 이미 SSE-S3 또는 SSE-KMS가 구성된 기존 버킷은 기본 암호화 구성이 변경되지 않습니다. SSE-KMS로 객체를 암호화하려면 버킷 설정에서 암호화 유형을 변경해야 합니다. 자세한 내용은 AWS KMS 키를 사용한 서버 측 암호화(SSE-KMS) 사용 섹션을 참조하세요.

SSE-KMS의 기본 암호화를 사용하도록 버킷을 구성할 때 S3 버킷 키를 사용하면 Amazon S3에서 AWS KMS로의 요청 트래픽을 줄이고 암호화 비용을 절감할 수 있습니다. 자세한 내용은 Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감 섹션을 참조하세요.

SSE-KMS가 기본 암호화로 사용 설정된 버킷을 식별하려면 Amazon S3 스토리지 렌즈 지표를 사용하면 됩니다. S3 스토리지 렌즈는 조직 전반에서 객체 스토리지 사용 및 활동에 대한 가시성을 확보하는 데 사용할 수 있는 클라우드 스토리지 분석 기능입니다. 자세한 내용은 Using S3 Storage Lens to protect your data(S3 스토리지 렌즈를 사용한 데이터 보호)를 참조하세요.

서버 측 암호화를 사용하는 경우 Amazon S3에서는 객체를 디스크에 저장하기 전에 암호화하고 객체를 다운로드할 때 이를 해독합니다. 서버 측 암호화를 사용한 데이터 보호 및 암호화 키 관리에 대한 자세한 내용은 서버 측 암호화를 사용하여 데이터 보호 섹션을 참조하세요.

기본 암호화에 필요한 권한에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 PutBucketEncryption 섹션을 참조하세요.

Amazon S3 콘솔, AWS SDK, Amazon S3 REST API 및 AWS Command Line Interface(AWS CLI CLI)를 사용하여 S3 버킷에 대한 Amazon S3 기본 암호화 동작을 구성할 수 있습니다.

기존 객체 암호화

기존에 암호화되지 않은 Amazon S3 객체를 암호화하기 위해 Amazon S3 배치 작업을 사용할 수 있습니다. 작업할 객체 목록을 S3 배치 작업에 제공하면 배치 작업은 각각의 API를 호출하여 지정된 작업을 수행합니다. 배치 작업 복사 작업을 사용하여 암호화되지 않은 기존 객체를 복사하고 암호화된 새로운 객체를 동일한 버킷에 작성할 수 있습니다. 단일 배치 작업 건으로 수십억 개의 객체에서 지정된 작업을 수행할 수 있습니다. 자세한 내용은 Batch Operations를 사용하여 대량으로 객체 작업 수행 및 AWS 스토리지 블로그 게시물 Amazon S3 배치 작업에서 객체 암호화를 참조하세요.

`CopyObject` API 작업이나 `copy-object` AWS CLI 명령을 사용하여 기존 객체를 암호화할 수도 있습니다. 자세한 내용은 AWS 스토리지 블로그 게시물 AWS CLI를 사용하여 기존 Amazon S3 객체 암호화를 참조하세요.

> **Note**
>
> 기본 버킷 암호화가 SSE-KMS로 설정된 Amazon S3 버킷은 the section called “서버 액세스 로깅”의 대상 버킷으로 사용할 수 없습니다. 서버 액세스 로그 대상 버킷에는 SSE-S3 기본 암호화만 지원됩니다.

크로스 계정 작업에 SSE-KMS 암호화 사용

크로스 계정 작업에 암호화를 사용하는 경우 다음 사항에 유의하세요.

- 요청 시 또는 버킷의 기본 암호화 구성을 통해 AWS KMS key Amazon 리소스 이름(ARN) 또는 별칭이 제공되지 않은 경우 AWS 관리형 키(`aws/s3`)가 사용됩니다.
- KMS 키와 동일한 AWS 계정에 있는 AWS Identity and Access Management(IAM) 보안 주체를 사용하여 S3 객체를 업로드하거나 액세스하는 경우 AWS 관리형 키(`aws/s3`)를 사용할 수 있습니다.
- S3 객체에 크로스 계정 액세스 권한을 부여하려면 고객 관리형 키를 사용합니다. 다른 계정으로부터의 액세스를 허용하도록 고객 관리형 키의 정책을 구성할 수 있습니다.
- 고객 관리형 KMS 키를 지정하는 경우 정규화된 KMS 키 ARN을 사용하는 것이 좋습니다. KMS 키 별칭을 대신 사용하는 경우 AWS KMS가 요청자의 계정 내에서 키를 해독합니다. 이 동작으로 인해 버킷 소유자가 아닌 요청자에게 속한 KMS로 데이터가 암호화될 수 있습니다.

- 사용자(요청자)에게 `Encrypt` 권한이 부여된 키를 지정해야 합니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 키 사용자가 암호화 작업에 KMS를 사용하도록 허용을 참조하세요.

고객 관리형 키와 AWS 관리형 KMS 키를 사용해야 하는 상황에 대한 자세한 내용은 Amazon S3의 객체를 암호화하기 위해 AWS 관리형 키와 고객 관리형 KMS 키 중 무엇을 사용해야 합니까?를 참조하세요.

복제에 기본 암호화 사용

복제 대상 버킷에 대한 기본 암호화를 사용 설정하면 다음 암호화 동작이 적용됩니다.

- 소스 버킷의 객체가 암호화되지 않은 경우 대상 버킷의 복제본 객체는 대상 버킷의 기본 암호화 설정을 사용하여 암호화됩니다. 결과적으로 소스 객체의 엔터티 태그(ETag)는 복제본 객체의 ETag와 다릅니다. ETag를 사용하는 애플리케이션이 있는 경우 이 차이를 고려하도록 해당 애플리케이션을 업데이트해야 합니다.
- 소스 버킷의 객체가 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3), AWS Key Management Service(AWS KMS) 키를 사용한 서버 측 암호화(SSE-KMS) 또는 AWS KMS 키를 사용한 이중 계층 서브 측 암호화(DSSE-KMS)를 사용하여 암호화되는 경우, 대상 버킷의 복제본 객체는 소스 객체와 동일한 유형의 암호화를 사용합니다. 대상 버킷의 기본 암호화 설정은 사용되지 않습니다.

SSE-KMS와 함께 기본 암호화를 사용하는 방법에 대한 자세한 내용은 암호화된 객체 복제 섹션을 참조하세요.

Amazon S3 버킷 키와 기본 암호화 사용

새 객체에서 기본 암호화 동작으로 SSE-KMS를 사용하도록 버킷을 구성할 때 S3 버킷 키도 구성할 수 있습니다. S3 버킷 키는 Amazon S3에서 AWS KMS로의 트랜잭션 수를 줄여 SSE-KMS의 비용을 줄입니다.

새 객체에서 SSE-KMS용 S3 버킷 키를 사용하도록 버킷을 구성하면 AWS KMS는 버킷의 객체에 대해 고유한 데이터 키를 만드는 데 사용되는 버킷 수준 키를 생성합니다. 이 S3 버킷 키는 Amazon S3 내에서 제한된 기간 동안 사용되므로 Amazon S3가 암호화 작업을 완료하기 위해 AWS KMS에 요청할 필요성이 줄어듭니다.

S3 버킷 키 사용에 대한 자세한 내용은 Amazon S3 버킷 키 사용 단원을 참조하십시오.

기본 암호화 구성

> **Important**
>
> 이제 Amazon S3가 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)를 Amazon S3 내 모든 버킷 암호화의 기본 수준으로 적용합니다. 2023년 1월 5일부터 Amazon S3로의 모든 새 객체 업로드는 추가 비용 없이 성능에 영향을 미치지 않고 자동으로 암호화됩니다. S3 버킷 기본 암호화 구성에 및 신규 객체 업로드에 대한 자동 암호화 상태는 AWS CloudTrail 로그, S3 인벤토리, S3 스토리지 렌즈, Amazon S3 콘솔에서 사용할 수 있으며, AWS Command Line Interface 및 AWS SDK에서 추가 Amazon S3 API 응답 헤더로도 사용할 수 있습니다. 자세한 내용은 기본 암호화 관련 FAQ를 참조하십시오.

Amazon S3 버킷에는 기본적으로 버킷 암호화가 활성화되어 있으며, 새로운 객체는 Amazon S3 관리형 키(SSE-S3)를 통한 서버 측 암호화를 사용하여 자동으로 암호화됩니다. 이 암호화는 Amazon S3 버킷의 모든 새 객체에 적용되며 무료로 제공됩니다.

키 교체 및 액세스 정책 권한 부여 관리와 같이 암호화 키를 더 세밀하게 제어해야 하는 경우 AWS Key Management Service(AWS KMS) 키를 사용한 서버 측 암호화(SSE-KMS) 또는 AWS KMS 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)를 사용하는 방법도 있습니다. SSE-KMS에 대한 자세한 내용은 AWS KMS(SSE-KMS)를 사용한 서버 측 암호화 지정 단원을 참조하십시오. DSSE-KMS에 대한 자세한 내용은 the section called “이중 계층 서버 측 암호화(DSSE-KMS)” 섹션을 참조하세요.

다른 계정에서 소유한 KMS 키를 사용하려면 먼저 해당 키에 대한 사용 권한이 있어야 합니다. KMS 키의 크로스 계정 권한에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서에서 다른 계정에서 사용할 수 있는 KMS 키 만들기를 참조하십시오.

기본 버킷 암호화를 SSE-KMS로 설정하는 경우 S3 버킷 키도 설정하여 AWS KMS 요청 비용을 절감할 수 있습니다. 자세한 내용은 Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감 섹션을 참조하세요.

> **Note**
>
> PutBucketEncryption을 사용하여 기본 버킷 암호화를 SSE-KMS로 설정하려는 경우 KMS 키 ID가 올바른지 확인해야 합니다. Amazon S3는 PutBucketEncryption 요청에 제공된 KMS 키 ID를 검증하지 않습니다.

S3 버킷의 기본 암호화 사용에 대한 추가 비용은 없습니다. 기본 암호화 동작을 구성하도록 요청할 경우 표준 Amazon S3 요청 요금이 발생합니다. 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오. SSE-KMS 및 DSSE-KMS의 경우 AWS KMS 요금이 적용됩니다. 요금은 AWS KMS 요금에 나와 있습니다.

고객 제공 암호화 키(SSE-C)를 통한 서버 측 암호화는 기본 암호화로 지원되지 않습니다.

Amazon S3 콘솔, AWS SDK, Amazon S3 REST API 및 AWS Command Line Interface(AWS CLI)를 사용하여 S3 버킷에 대한 Amazon S3 기본 암호화를 구성할 수 있습니다.

기본 암호화를 사용 설정하기 전에 유의할 변경 사항

버킷에 대한 기본 암호화를 사용 설정한 후에는 다음 암호화 동작이 적용됩니다.

- 기본 암호화가 사용 설정되기 전에 버킷에 있었던 객체의 암호화는 변경되지 않습니다.
- 기본 암호화를 사용 설정한 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 구성으로 SSE-KMS 또는 DSSE-KMS 옵션을 사용할 경우 AWS KMS의 초당 요청 수(RPS) 할당량이 적용됩니다. AWS KMS 할당량과 할당량 증대를 요청하는 방법에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 할당량을 참조하세요.

> **Note**
>
> 기본 암호화가 활성화되기 전에 업로드된 객체는 암호화되지 않습니다. 기존 객체 암호화에 대한 자세한 내용은 the section called “기본 버킷 암호화 설정” 섹션을 참조하세요.

S3 콘솔 사용

Amazon S3 버킷에 대해 기본 암호화를 구성하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.

3. 버킷 목록에서 원하는 버킷의 이름을 선택합니다.
4. 속성(Properties) 탭을 선택합니다.
5. 기본 암호화에서 편집을 선택합니다.
6. 암호화를 구성하려면 암호화 유형에서 다음 중 하나를 선택합니다.

   - Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)
   - AWS Key Management Service 키를 사용한 서버 측 암호화(SSE-KMS)
   - AWS Key Management Service 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)

   > **Important**
   >
   > 기본 암호화 구성으로 SSE-KMS 또는 DSSE-KMS 옵션을 사용할 경우 AWS KMS의 초당 요청 수(RPS) 할당량이 적용됩니다. AWS KMS 할당량과 할당량 증대를 요청하는 방법에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 할당량을 참조하십시오.

   버킷에 다른 유형의 기본 암호화를 지정하지 않으면 버킷과 새 객체는 기본적으로 SSE-S3로 암호화됩니다. 기본 암호화에 대한 자세한 내용은 Amazon S3 버킷에 대한 기본 서버 측 암호화 동작 설정 섹션을 참조하십시오.

   Amazon S3 서버 측 암호화를 사용하는 데이터 암호화에 대한 자세한 내용은 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) 사용 섹션을 참조하십시오.
7. AWS Key Management Service키를 사용한 서버 측 암호화(SSE-KMS) 또는 AWS Key Management Service 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)를 선택한 경우 다음을 수행하세요.

   a. AWS KMS 키에서 다음 방법 중 하나로 KMS 키를 지정합니다.

      - 사용 가능한 KMS 키 목록에서 AWS KMS keys 중에서 선택을 선택하고 사용 가능한 키 목록에서 KMS 키를 선택합니다.

        AWS 관리형 키(aws/s3)와 고객 관리형 키가 모두 목록에 표시됩니다. 고객 관리형 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 키 및 AWS 키를 참조하십시오.
      - KMS 키 ARN을 입력하려면 AWS KMS key ARN 입력을 선택하고 나타나는 필드에 KMS 키 ARN을 입력합니다.

- AWS KMS 콘솔에서 고객 관리형 키를 생성하려면 KMS 키 생성을 선택합니다.

  AWS KMS key 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 키 생성을 참조하십시오.

> **Important**
>
> 버킷과 동일한 AWS 리전에서 사용되는 KMS 키만 사용할 수 있습니다. KMS 키에서 선택(Choose from your KMS keys)을 선택하면 S3 콘솔에는 KMS 키가 리전당 100개씩만 나열됩니다. 동일한 리전에 100개 이상의 KMS 키가 있는 경우, S3 콘솔에서 처음 100개의 KMS 키만 볼 수 있습니다. 콘솔에 나열되지 않은 KMS 키를 사용하려면 AWS KMS key ARN 입력을 선택하고 KMS 키 ARN을 입력합니다.
> Amazon S3에서 서버 측 암호화에 AWS KMS key을 사용하는 경우 대칭 암호화 KMS 키를 선택해야 합니다. Amazon S3는 대칭 암호화 KMS 키만 지원합니다. 이들 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 대칭 암호화 KMS 키를 참조하세요.

Amazon S3에서 SSE-KMS를 사용하는 방법에 대한 자세한 내용은 AWS KMS 키를 사용한 서버 측 암호화(SSE-KMS) 사용 섹션을 참조하세요. DSSE-KMS 사용에 대한 자세한 내용은 the section called “이중 계층 서버 측 암호화(DSSE-KMS)” 섹션을 참조하세요.

b. SSE-KMS와 함께 기본 암호화를 사용하도록 버킷을 구성할 때 S3 버킷 키도 활성화할 수 있습니다. S3 버킷 키를 사용하면 Amazon S3에서 AWS KMS로의 요청 트래픽이 줄어 암호화 비용이 절감됩니다. 자세한 내용은 Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감 섹션을 참조하세요.

   S3 버킷 키를 사용하려면 [버킷 키(Bucket Key)]에서 [사용(Enable)]을 선택합니다.

> **Note**
>
> S3 버킷 키는 DSSE-KMS에서 지원되지 않습니다.

8. 변경 사항 저장을 선택합니다.

AWS CLI 사용

다음 예시에서는 SSE-S3를 사용하거나 S3 버킷 키와 함께 SSE-KMS를 사용하여 기본 암호화를 구성하는 방법을 보여줍니다.

기본 암호화에 대한 자세한 내용은 Amazon S3 버킷에 대한 기본 서버 측 암호화 동작 설정 섹션을 참조하십시오. AWS CLI를 사용하여 기본 암호화를 구성하는 방법에 대한 자세한 내용은 put-bucket-encryption을 참조하세요.

Example – SSE-S3를 사용한 기본 암호화

이 예시에서는 Amazon S3 관리형 키를 사용하여 기본 버킷 암호화를 구성합니다.

```
aws s3api put-bucket-encryption --bucket amzn-s3-demo-bucket --server-side-encryption-
configuration '{
    "Rules": [
        {
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }
    ]
}'
```

Example - S3 버킷 키와 SSE-KMS를 사용한 기본 암호화

이 예제에서는 S3 버킷 키와 SSE-KMS를 사용하여 기본 버킷 암호화를 구성합니다.

```
aws s3api put-bucket-encryption --bucket amzn-s3-demo-bucket --server-side-encryption-
configuration '{
    "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "aws:kms",
                    "KMSMasterKeyID": "KMS-Key-ARN"
                },
                "BucketKeyEnabled": true
            }
        ]
    }'
```

REST API 사용

REST API `PutBucketEncryption` 작업을 사용하여 기본 암호화를 활성화하고 사용할 서버 측 암호화 유형(SSE-S3, SSE-KMS 또는 DSSE-KMS)을 설정합니다.

자세한 내용은 Amazon Simple Storage Service API 참조에서 PutBucketEncryption를 참조하십시오.

AWS CloudTrail 및 Amazon EventBridge를 사용한 기본 암호화 모니터링

> **Important**
>
> 이제 Amazon S3가 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)를 Amazon S3 내 모든 버킷 암호화의 기본 수준으로 적용합니다. 2023년 1월 5일부터 Amazon S3로의 모든 새 객체 업로드는 추가 비용 없이 성능에 영향을 미치지 않고 자동으로 암호화됩니다. S3 버킷 기본 암호화 구성에 및 신규 객체 업로드에 대한 자동 암호화 상태는 AWS CloudTrail 로그, S3 인벤토리, S3 스토리지 렌즈, Amazon S3 콘솔에서 사용할 수 있으며, AWS Command Line Interface 및 AWS SDK에서 추가 Amazon S3 API 응답 헤더로도 사용할 수 있습니다. 자세한 내용은 기본 암호화 관련 FAQ를 참조하십시오.

AWS CloudTrail 이벤트를 사용하여 Amazon S3 버킷에 대한 기본 암호화 구성 요청을 추적할 수 있습니다. CloudTrail 로그에 사용되는 API 이벤트 이름은 다음과 같습니다.

- `PutBucketEncryption`
- `GetBucketEncryption`
- `DeleteBucketEncryption`

또한 CloudTrail 이벤트를 이러한 API 호출에 매칭하는 EventBridge 규칙을 생성할 수 있습니다. CloudTrail 이벤트에 대한 자세한 내용은 콘솔을 사용하여 버킷의 객체에 대한 로깅 사용 설정 섹션을 참조하세요. EventBridge 이벤트에 자세한 내용은 AWS 서비스의 이벤트를 참조하세요.

객체 수준 Amazon S3 작업에 CloudTrail 로그를 사용하여 Amazon S3에 대한 `PUT` 및 `POST` 요청을 추적할 수 있습니다. 이러한 작업을 사용하면 들어오는 `PUT` 요청에 암호화 헤더가 없을 때 기본 암호화를 사용하여 개체가 암호화되는지 여부를 확인할 수 있습니다.

Amazon S3가 기본 암호화 설정을 사용하여 객체를 암호화하는 경우 로그에는 이름-값 페어로 `"SSEApplied":"Default_SSE_S3"`, `"SSEApplied":"Default_SSE_KMS"` 또는 `"SSEApplied":"Default_DSSE_KMS"` 필드 중 하나가 포함됩니다.

Amazon S3가 PUT 암호화 헤더를 사용하여 객체를 암호화하는 경우 로그에는 이름-값 페어로 `"SSEApplied":"SSE_S3"`, `"SSEApplied":"SSE_KMS"`, `"SSEApplied":"DSSE_KMS"` 또는 `"SSEApplied":"SSE_C"` 필드 중 하나가 포함됩니다.

멀티파트 업로드의 경우 이 정보가 `InitiateMultipartUpload` API 작업 요청에 포함됩니다. CloudTrail 및 CloudWatch 사용에 대한 자세한 내용은 Amazon S3의 로깅 및 모니터링 단원을 참조하십시오.

기본 암호화 관련 FAQ

이제 Amazon S3가 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)를 Amazon S3 내 모든 버킷 암호화의 기본 수준으로 적용합니다. 2023년 1월 5일부터 Amazon S3로의 모든 새 객체 업로드는 추가 비용 없이 성능에 영향을 미치지 않고 자동으로 암호화됩니다. 256비트 고급 암호화 표준(AES-256)을 사용하는 SSE-S3는 모든 새 버킷과 기본 암호화가 구성되어 있지 않은 기존 S3 버킷에 자동으로 적용됩니다. S3 버킷 기본 암호화 구성에 및 신규 객체 업로드에 대한 자동 암호화 상태는 AWS CloudTrail 로그, S3 인벤토리, S3 스토리지 렌즈, Amazon S3 콘솔에서 사용할 수 있으며, AWS Command Line Interface(AWS CLI) 및 AWS SDK에서 추가 Amazon S3 API 응답 헤더로도 사용할 수 있습니다.

다음 섹션에서는 이 업데이트에 대한 질문과 답변을 제공합니다.

이미 기본 암호화가 구성되어 있는 기존 버킷의 기본 암호화 설정도 Amazon S3가 변경하나요?

아니요. 이미 SSE-S3 또는 AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 사용한 서버 측 암호화가 구성된 기존 버킷은 기본 암호화 구성이 변경되지 않습니다. 버킷에 기본 암호화 동작을 설정하는 방법에 대한 자세한 내용은 Amazon S3 버킷에 대한 기본 서버 측 암호화 동작 설정 페이지를 참조하십시오. SSE-S3 및 SSE-KMS 암호화 설정에 대한 자세한 내용은 서버 측 암호화를 사용하여 데이터 보호 페이지를 참조하십시오.

기본 암호화가 구성되지 않은 기존 버킷에서 기본 암호화가 활성화되나요?

예. 암호화되지 않은 기존의 모든 버킷에 이제 Amazon S3가 기본 암호화를 구성하여, 이들 버킷에 새로 업로드되는 객체에 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)를 적용합니다. 기존의 암호화되지 않은 버킷에 이미 있는 객체는 자동으로 암호화되지 않습니다.

새 객체 업로드의 기본 암호화 상태를 보려면 어떻게 해야 합니까?

현재, 신규 객체 업로드의 기본 암호화 상태는 AWS CloudTrail 로그, S3 인벤토리, S3 스토리지 렌즈, Amazon S3 콘솔에서 확인할 수 있으며, AWS Command Line Interface(AWS CLI) 및 AWS SDK에서 추가 Amazon S3 API 응답 헤더로도 확인할 수 있습니다.

- CloudTrail 이벤트를 보려면 AWS CloudTrail 사용 설명서의 CloudTrail 콘솔에서 CloudTrail 이벤트 보기를 참조하십시오. CloudTrail 로그는 Amazon S3로의 PUT 및 POST 요청에 대한 API 추적을 제공합니다. 버킷 내 객체 암호화에 기본 암호화가 사용되면 PUT 및 POST API 요청에 대한 CloudTrail 로그에 `"SSEApplied":"Default_SSE_S3"` 필드가 이름-값 쌍으로 포함됩니다.
- S3 인벤토리에서 새 객체 업로드의 자동 암호화 상태를 보려면 Encryption(암호화) 메타데이터 필드를 포함하도록 S3 인벤토리 보고서를 구성한 다음 보고서에서 각 새 객체의 암호화 상태를 확인하십시오. 자세한 내용은 Amazon S3 인벤토리 설정을 참조하십시오.
- S3 스토리지 렌즈에서 새 객체 업로드에 대한 자동 암호화 상태를 보려면 S3 스토리지 렌즈 대시보드를 구성하고 대시보드의 Data protection(데이터 보호) 범주에서 Encrypted bytes(암호화된 바이트) 및 Encrypted object count(암호화된 객체 수) 지표를 확인하십시오. 자세한 내용은 S3 콘솔 사용 및 대시보드에서 S3 스토리지 렌즈 지표 보기(을)를 참조하세요.
- Amazon S3 콘솔에서 자동 버킷 수준 암호화 상태를 보려면 Amazon S3 콘솔에서 Amazon S3 버킷의 기본 암호화를 확인하십시오. 자세한 내용은 기본 암호화 구성 섹션을 참조하세요.
- 자동 암호화 상태를 AWS Command Line Interface(AWS CLI) 및 AWS SDK에서 추가 Amazon S3 API 응답 헤더로 확인하려면 객체 작업 API(예: PutObject 및 GetObject)를 사용할 때 `x-amz-server-side-encryption` 응답 헤더를 확인하십시오.

이 변경 사항을 활용하려면 어떻게 해야 합니까?

기존 애플리케이션을 변경할 필요는 없습니다. 모든 버킷에 기본 암호화가 활성화되어 있어, Amazon S3에 새로 업로드되는 모든 객체는 자동으로 암호화됩니다.

버킷에 새로 기록되는 객체의 암호화를 비활성화할 수 있습니까?

아니요. SSE-S3 암호화는 버킷에 새로 업로드되는 모든 객체에 적용되는 새로운 기본 암호화 수준입니다. 새 객체 업로드에 대한 암호화는 더 이상 비활성화할 수 없습니다.

요금이 영향을 받습니까?

아니요. SSE-S3 기본 암호화 기능은 추가 비용 없이 제공됩니다. 스토리지, 요청 등 S3 기능에 대한 요금이 기존과 똑같이 청구됩니다. 요금에 대해서는 Amazon S3 요금을 참조하십시오.

암호화되지 않은 기존 객체도 Amazon S3가 암호화합니까?

아니요. Amazon S3는 2023년 1월 5일부터 새로 업로드되는 객체만 자동으로 암호화합니다. 기존 객체를 암호화하려면 S3 배치 작업을 사용하여 객체의 암호화된 사본을 생성하면 됩니다. 이렇게 암호화된 사본은 기존 객체 데이터 및 이름을 유지하고, 지정한 암호화 키를 사용하여 암호화됩니다. 자세한 내용은 AWS Storage 블로그의 Encrypting objects with Amazon S3 Batch Operations(Amazon S3 배치 작업에서 객체 암호화)를 참조하십시오.

이번 릴리스 전에 내 버킷에 암호화를 활성화하지 않았습니다. 객체에 액세스하는 방법을 변경해야 합니까?

아니요. SSE-S3 기본 암호화 기능을 사용하면 Amazon S3에 기록되는 데이터가 자동으로 암호화되고, 해당 데이터에 액세스할 때 데이터가 자동으로 복호화됩니다. 자동으로 암호화된 객체에 액세스하는 방식에는 변화가 없습니다.

클라이언트측 암호화된 내 객체에 액세스하는 방법을 변경해야 합니까?

아니요. Amazon S3에 업로드되기 전에 클라이언트측 암호화된 모든 객체는 암호화된 사이퍼텍스트 객체로 Amazon S3에 도착합니다. 이러한 객체에 이제 SSE-S3 암호화 계층이 추가로 놓입니다. 클라이언트측 암호화된 객체를 사용하는 워크로드는 클라이언트 서비스 또는 권한 부여 설정을 변경할 필요가 없습니다.

> **Note**
>
> 업데이트된 버전의 AWS Provider를 사용하지 않는 HashiCorp Terraform 사용자는 고객 정의 암호화 구성 없이 새 S3 버킷을 생성하면 예기치 않은 드리프트를 경험할 수 있습니다. 이러한 드리프트를 피하려면 Terraform AWS Provider 버전을 모든 4.x 릴리스, 3.76.1, 2.70.4 버전 중 하나로 업데이트해야 합니다.

Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) 사용

> **Important**
>
> 이제 Amazon S3가 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)를 Amazon S3 내 모든 버킷 암호화의 기본 수준으로 적용합니다. 2023년 1월 5일부터 Amazon S3로의 모든 새 객체 업로드는 추가 비용 없이 성능에 영향을 미치지 않고 자동으로 암호화됩니다. S3 버킷 기본 암호화 구성에 및 신규 객체 업로드에 대한 자동 암호화 상태는 AWS CloudTrail 로그, S3 인벤토리, S3 스토리지 렌즈, Amazon S3 콘솔에서 사용할 수 있으며, AWS Command Line Interface 및 AWS SDK에서 추가 Amazon S3 API 응답 헤더로도 사용할 수 있습니다. 자세한 내용은 기본 암호화 관련 FAQ를 참조하십시오.

Amazon S3 버킷에 대한 모든 신규 객체 업로드는 기본적으로 Amazon S3 관리형 키(SSE-S3)를 통한 서버 측 암호화로 암호화됩니다.

서버 측 암호화를 사용하여 저장된 데이터를 보호합니다. Amazon S3는 각 객체를 고유한 키로 암호화합니다. 또한 추가 보안 조치로 주기적으로 바뀌는 키를 사용하여 키 자체를 암호화합니다. Amazon S3 서버 측 암호화는 256비트 고급 암호화 표준 갈루아/카운터 모드(AES-GCM)를 사용하여 업로드된 모든 객체를 암호화합니다.

Amazon S3 관리형 키(SSE-S3)를 통한 서버 측 암호화를 사용하는 경우 추가 요금이 부과되지 않습니다. 그러나 기본 암호화 기능을 구성하도록 요청할 경우 표준 Amazon S3 요청 요금이 발생합니다. 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

Amazon S3 관리형 키만 사용하여 데이터 업로드를 암호화하도록 요구하는 경우 다음 버킷 정책을 사용할 수 있습니다. 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": "PutObjectPolicy",
  "Statement": [
    {
      "Sid": "DenyObjectsThatAreNotSSES3",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
      "Condition": {
        "StringNotEquals": {
          "s3:x-amz-server-side-encryption": "AES256"
        }
      }
    }
  ]
}
```

> **Note**
>
> 서버 측 암호화는 객체 메타데이터가 아닌 객체 데이터만 암호화합니다.

서버 측 암호화를 위한 API 지원

모든 Amazon S3 버킷에는 기본적으로 암호화가 구성되어 있으며 S3 버킷에 업로드되는 신규 객체는 모두 저장 시 자동으로 암호화됩니다. Amazon S3 관리형 키(SSE-S3)를 사용한 서버 측 암호화가

Amazon S3 내 모든 버킷의 기본 암호화 구성입니다. 다른 유형의 암호화를 사용하려면 S3 PUT 요청에 사용할 서버 측 암호화 유형을 지정하거나 대상 버킷에 기본 암호화 구성을 업데이트할 수 있습니다.

PUT 요청에 다른 암호화 유형을 지정하려는 경우 AWS Key Management Service(AWS KMS) 키를 사용한 서버 측 암호화(SSE-KMS), AWS KMS 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS) 또는 고객 제공 키를 사용한 서버 측 암호화 (SSE-C)를 사용할 수 있습니다. 대상 버킷에 다른 기본 암호화 구성을 설정하려는 경우 SSE-KMS 또는 DSSE-KMS를 사용할 수 있습니다.

범용 버킷의 기본 암호화 구성을 변경하는 방법에 대한 자세한 내용은 기본 암호화 구성 섹션을 참조하세요.

버킷의 기본 암호화 구성을 SSE-KMS로 변경해도 버킷에 있는 기존 Amazon S3 객체의 암호화 유형은 변경되지 않습니다. 기본 암호화 구성을 SSE-KMS로 업데이트한 후 기존 객체의 암호화 유형을 변경하려는 경우 Amazon S3 Batch Operations를 사용할 수 있습니다. 객체 목록을 S3 Batch Operations에 제공하면 Batch Operations가 각각의 API 작업을 직접적으로 호출합니다. 객체 복사 작업을 사용하면 기존 객체를 복사하여 SSE-KMS로 암호화된 객체와 동일한 버킷에 다시 쓸 수 있습니다. 단일 배치 작업 건으로 수십억 개의 객체에서 지정된 작업을 수행할 수 있습니다. 자세한 내용은 Batch Operations를 사용하여 대량으로 객체 작업 수행 및 AWS Storage Blog 게시물 How to retroactively encrypt existing objects in Amazon S3 using S3 Inventory, Amazon Athena, and S3 Batch Operations를 참조하세요.

객체 생성 REST API를 사용하여 서버 측 암호화를 구성하려면 `x-amz-server-side-encryption` 요청 헤더를 제공해야 합니다. REST API에 대한 정보는 REST API 사용 단원을 참조하십시오.

다음 Amazon S3 API는 이러한 헤더를 지원합니다.

- PUT 작업 - PUT API를 사용하여 데이터를 업로드할 때 요청 헤더를 지정합니다. 자세한 내용은 PUT Object 단원을 참조하십시오.
- 멀티파트 업로드 시작 – 멀티파트 업로드 API 작업을 사용하여 대용량 객체를 업로드할 때 시작 요청에 헤더를 지정합니다. 자세한 내용은 멀티파트 업로드 시작을 참조하십시오.
- COPY 작업 - 객체를 복사할 때 소스 객체 및 대상 객체가 둘 다 있습니다. 자세한 내용은 PUT Object - Copy를 참조하십시오.

> **Note**
>
> POST 작업을 사용하여 객체를 업로드할 경우에는 요청 헤더를 제공하는 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 Console을 사용하여 객체를 업로드하고 서버 측 암호화를 요청할 수도 있습니다.

더 일반적인 내용은 AWS Key Management Service 개발자 안내서의 AWS KMS 개념을 참조하십시오.

주제

- Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) 지정

Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) 지정

모든 Amazon S3 버킷에는 기본적으로 암호화가 구성되어 있으며 S3 버킷에 업로드되는 신규 객체는 모두 저장 시 자동으로 암호화됩니다. Amazon S3 관리형 키(SSE-S3)를 사용한 서버 측 암호화가 Amazon S3 내 모든 버킷의 기본 암호화 구성입니다. 다른 유형의 암호화를 사용하려면 S3 PUT 요청에 사용할 서버 측 암호화 유형을 지정하거나 대상 버킷에 기본 암호화 구성을 업데이트할 수 있습니다.

PUT 요청에 다른 암호화 유형을 지정하려는 경우 AWS Key Management Service(AWS KMS) 키를 사용한 서버 측 암호화(SSE-KMS), AWS KMS 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS) 또는 고객 제공 키를 사용한 서버 측 암호화 (SSE-C)를 사용할 수 있습니다. 대상 버킷에 다른 기본 암호화 구성을 설정하려는 경우 SSE-KMS 또는 DSSE-KMS를 사용할 수 있습니다.

범용 버킷의 기본 암호화 구성을 변경하는 방법에 대한 자세한 내용은 기본 암호화 구성 섹션을 참조하세요.

버킷의 기본 암호화 구성을 SSE-KMS로 변경해도 버킷에 있는 기존 Amazon S3 객체의 암호화 유형은 변경되지 않습니다. 기본 암호화 구성을 SSE-KMS로 업데이트한 후 기존 객체의 암호화 유형을 변경하려는 경우 Amazon S3 Batch Operations를 사용할 수 있습니다. 객체 목록을 S3 Batch Operations에 제공하면 Batch Operations가 각각의 API 작업을 직접적으로 호출합니다. 객체 복사 작업을 사용하면 기존 객체를 복사하여 SSE-KMS로 암호화된 객체와 동일한 버킷에 다시 쓸 수 있습니다. 단일 배치 작업 건으로 수십억 개의 객체에서 지정된 작업을 수행할 수 있습니다. 자세한 내용은 Batch Operations를 사용하여 대량으로 객체 작업 수행 및 AWS Storage Blog 게시물 How to retroactively encrypt existing objects in Amazon S3 using S3 Inventory, Amazon Athena, and S3 Batch Operations를 참조하세요.

S3 콘솔, REST API, AWS SDK 및 AWS Command Line Interface(AWS CLI)를 사용하여 SSE-S3를 지정할 수 있습니다. 자세한 내용은 Amazon S3 버킷에 대한 기본 서버 측 암호화 동작 설정 섹션을 참조하세요.

S3 콘솔 사용

이 주제에서는 AWS Management Console을 사용하여 객체 암호화의 유형을 설정하거나 변경하는 방법을 설명합니다. 콘솔을 사용하여 객체를 복사할 경우 Amazon S3는 원본 그대로 객체를 복사합니다. 즉, 소스 객체가 암호화된 상태이면 대상 객체도 암호화됩니다. 콘솔을 사용하여 객체에 대한 암호화를 추가하거나 변경할 수 있습니다.

> **Note**
>
> - 객체의 용량이 5GB 미만인 경우 객체의 암호화를 변경할 수 있습니다. 객체가 5GB보다 큰 경우, 객체의 암호화를 변경하려면 AWS CLI 또는 AWS SDK를 사용해야 합니다.
> - 객체의 암호화를 변경하는 데 필요한 추가 권한 목록은 the section called “S3 API 작업에 필요한 권한” 섹션을 참조하세요. 이 권한을 부여하는 정책의 예시는 the section called “자격 증명 기반 정책 예제” 단원을 참조하세요.
> - 객체 암호화를 변경하면 새 객체가 생성되어 이전 객체를 대체합니다. S3 버전 관리가 사용 설정된 경우 객체의 새 버전이 생성되고 기존 객체는 이전 버전이 됩니다. 또한 속성을 변경하는 역할도 새 객체(또는 객체 버전)의 소유자가 됩니다.

객체에 대한 암호화를 변경하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 탐색 창에서 버킷을 선택하고 범용 버킷 탭을 선택합니다. 변경할 객체가 포함된 Amazon S3 버킷 또는 폴더로 이동합니다.
3. 변경하려는 객체의 확인란을 선택합니다.
4. 작업 메뉴에 표시되는 옵션 목록에서 서버 측 암호화 편집을 선택합니다.
5. 서버 측 암호화 섹션으로 스크롤합니다.
6. 암호화 설정에서 버킷 기본 암호화 설정 사용 또는 기본 암호화에 버킷 설정 재정의를 선택합니다.
7. 기본 암호화에 버킷 설정 재정의를 선택한 경우 다음과 같은 암호화 설정을 구성해야 합니다.
    - 암호화 유형에서 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)를 선택합니다. SSE-S3는 가장 강력한 블록 암호 중 하나인 256비트 Advanced Encryption Standard(AES-256)을 사용하여 각 객체를 암호화합니다. 자세한 내용은 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) 사용 섹션을 참조하세요.

8. 추가 복사 설정에서 소스 설정 복사, 설정 지정 안 함 또는 설정 지정 중 원하는 것을 선택합니다. 소스 설정 복사가 기본 옵션입니다. 소스 설정 속성 없이 객체만 복사하려면 설정 지정 안 함을 선택합니다. 스토리지 클래스, ACL, 객체 태그, 메타데이터, 서버 측 암호화 및 추가 체크섬에 대한 설정을 지정하려면 설정 지정을 선택합니다.
9. 변경 사항 저장을 선택합니다.

> **Note**
>
> 이 작업은 지정된 모든 객체에 암호화를 적용합니다. 폴더를 암호화할 때 폴더에 새 객체를 추가하기 전에 저장 작업이 완료될 때까지 기다립니다.

REST API 사용

새 객체를 업로드하거나 기존 객체를 복사하여 객체가 생성될 경우 요청에 `x-amz-server-side-encryption` 헤더를 추가하여 Amazon S3에서 Amazon S3 관리형 키(SSE-S3)를 사용하여 데이터를 암호화하도록 할지 지정할 수 있습니다. Amazon S3에서 지원하는 암호화 알고리즘인 `AES256`으로 헤더의 값을 설정합니다. Amazon S3는 `x-amz-server-side-encryption` 응답 헤더를 반환하여 객체가 SSE-S3를 사용하여 저장됨을 확인해 줍니다.

다음 REST 업로드 API 작업은 `x-amz-server-side-encryption` 요청 헤더를 수락합니다.

- PUT Object
- PUT Object - Copy
- POST Object
- 멀티파트 업로드 시작

멀티파트 업로드 API 작업을 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 소스 객체의 암호화 여부에 관계없이 명시적으로 서버 측 암호화를 요청하지 않는 한 대상 객체는 암호화되지 않습니다.

객체가 SSE-S3를 사용하여 저장될 경우 다음 REST API 작업의 응답 헤더는 `x-amz-server-side-encryption` 헤더를 반환합니다.

- PUT Object
- PUT Object - Copy

- POST Object
- 멀티파트 업로드 시작
- 부분 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 전송하지 마십시오. 전송할 경우 HTTP 상태 코드 400(잘못된 요청) 오류가 발생합니다.

AWS SDK 사용

AWS SDK를 사용할 때 Amazon S3가 Amazon S3 관리형 암호화 키(SSE-S3)를 통한 서버 측 암호화를 사용하도록 요청할 수 있습니다. 이 섹션에서는 여러 언어로 AWS SDK를 사용한 예제를 제공합니다. 다른 SDK에 대한 자세한 내용은 샘플 코드 및 라이브러리 섹션을 참조하세요.

Java

AWS SDK for Java를 사용하여 객체를 업로드할 때 SSE-S3를 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 S3에서는 데이터를 암호화해 저장합니다.

멀티파트 업로드 API 작업을 사용하여 객체를 업로드할 때도 SSE-S3 암호화를 요청할 수 있습니다.

- 상위 수준 멀티파트 업로드 API 작업을 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 업로드를 사용한 객체 업로드 섹션을 참조하세요.
- 하위 수준 멀티파트 업로드 API 작업을 사용하는 경우 멀티파트 업로드를 시작할 때 서버 측 암호화를 지정합니다. `ObjectMetadata` 메서드를 호출하여

`InitiateMultipartUploadRequest.setObjectMetadata()` 속성을 추가합니다. 자세한 내용은 AWS SDK 사용(하위 수준 API) 섹션을 참조하세요.

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 S3에서는 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 예제

다음 예시에서는 AWS SDK for Java를 사용하여 서버 측 암호화를 설정하는 방법을 보여줍니다. 이 예제에서는 다음 작업을 수행하는 방법을 설명합니다.

- SSE-S3를 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 API 사용 섹션을 참조하십시오. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 AWS SDK for Java 개발자 안내서의 시작하기 섹션을 참조하세요.

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

.NET

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`에서 SSE-S3를 지정하려면 다음을 추가합니다.

```
ServerSideEncryptionMethod = ServerSideEncryptionMethod.AES256
```

객체를 복사하는 방법에 대한 실제 예제는 AWS SDK 사용를 참조하십시오.

다음은 객체를 업로드하는 예제입니다. 요청 시 이 예제는 Amazon S3에 객체를 암호화하도록 지시합니다. 그런 다음 객체 메타데이터를 검색해 사용된 암호화 방법을 확인합니다. 코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

```
using Amazon;
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);
```

```
            }
        }
    }
}
```

PHP

이 주제에서는 Amazon S3에 업로드하는 객체에 SSE-S3를 추가하기 위해 AWS SDK for PHP 버전 3의 클래스를 사용하는 방법을 보여줍니다. AWS SDK for Ruby API에 대한 자세한 내용은 AWS SDK for Ruby – 버전 2를 참조하세요.

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 API 사용 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 API 작업을 사용하여 대용량 객체를 업로드할 때 업로드할 객체에 대해 다음과 같이 SSE-S3를 지정할 수 있습니다.

- 하위 수준 멀티파트 업로드 API 작업을 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 API 작업에 대한 자세한 내용은 AWS SDK 사용(하위 수준 API) 섹션을 참조하십시오.
- 상위 수준 멀티파트 업로드 API 작업을 사용하는 경우 CreateMultipartUpload API 작업의 `ServerSideEncryption` 파라미터를 사용하여 서버 측 암호화를 지정합니다. 상위 수준 멀티파트 업로드 API 작업과 함께 `setOption()` 메서드를 사용하는 예제는 멀티파트 업로드를 사용한 객체 업로드 섹션을 참조하십시오.

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 암호화 상태를 변경하려면 Aws\S3\S3Client::copyObject() 메서드를 사용하여 객체를 복사하고 소스 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',
]);
```

자세한 정보는 다음의 주제를 참조하세요.

- Amazon S3 Aws\S3\S3Client Class용 AWS SDK for PHP
- AWS SDK for PHP 설명서

Ruby

AWS SDK for Ruby를 사용하여 객체를 업로드하는 경우, 객체가 SSE-S3를 통해 암호화된 상태로 저장되도록 지정할 수 있습니다. 다시 객체를 읽으면 객체가 자동으로 복호화됩니다.

다음 AWS SDK for Ruby 버전 3 예시는 파일을 Amazon S3에 업로드한 후 암호화하여 저장하는 방법을 보여줍니다.

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

```
# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
 object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
 #{encryption}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

다음 코드 예제는 기존 객체의 암호화 상태를 확인하는 방법을 보여줍니다.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object
```

```
  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def object
    @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  obj_data = wrapper.get_object
  return unless obj_data

  encryption = obj_data.server_side_encryption.nil? ? 'no' :
 obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Amazon S3에 저장된 객체에 서버 측 암호화가 사용되지 않은 경우 메서드는 `null`을 반환합니다.

기존 객체의 암호화 상태를 변경하려면 객체를 복사한 다음 원본 객체를 삭제합니다. 기본적으로 명시적으로 서버 측 암호화를 요청하지 않는 한, 복사 메서드는 대상을 암호화하지 않습니다. 다음 Ruby 코드 예시와 같이 옵션 해시 인수에 `server_side_encryption` 값을 지정하여 대상 객체 암호화를 요청할 수 있습니다. 이 코드 예시는 객체를 복사하고 복사본을 SSE-S3로 암호화하는 방법을 보여줍니다.

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

```
# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the target
 key, and encrypt it.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key, encryption)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
 server_side_encryption: encryption)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key, target_encryption)
  return unless target_object
```

```
  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key} and "\
       "encrypted the target with #{target_object.server_side_encryption}
 encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

AWS CLI 사용

AWS CLI를 사용하여 객체를 업로드할 때 SSE-S3를 지정하려면 다음 예시를 사용합니다.

```
aws s3api put-object --bucket amzn-s3-demo-bucket1 --key object-key-name --server-side-
encryption AES256  --body file path
```

자세한 내용은 AWS CLI 참조에서 put-object를 참조하십시오. AWS CLI를 사용하여 객체를 복사할 때 SSE-S3를 지정하려면 copy-object를 참조하십시오.

AWS CloudFormation 사용하기

AWS CloudFormation을 사용하여 암호화를 설정하는 방법의 예제는 AWS CloudFormation 사용 설명서의 `AWS::S3::Bucket ServerSideEncryptionRule`에 있는 기본 암호화로 버킷 생성 및 S3 버킷 키로 AWS KMS 서버 측 암호화를 사용하여 버킷 생성을 참조하십시오.

AWS KMS 키를 사용한 서버 측 암호화(SSE-KMS) 사용

> **Important**
>
> 이제 Amazon S3가 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)를 Amazon S3 내 모든 버킷 암호화의 기본 수준으로 적용합니다. 2023년 1월 5일부터 Amazon S3로의 모든 새 객체 업로드는 추가 비용 없이 성능에 영향을 미치지 않고 자동으로 암호화됩니다. S3 버킷 기본 암호화 구성에 및 신규 객체 업로드에 대한 자동 암호화 상태는 AWS CloudTrail 로그, S3 인벤토리, S3 스토리지 렌즈, Amazon S3 콘솔에서 사용할 수 있으며, AWS Command Line Interface 및 AWS SDK에서 추가 Amazon S3 API 응답 헤더로도 사용할 수 있습니다. 자세한 내용은 기본 암호화 관련 FAQ를 참조하십시오.

서버 측 암호화는 데이터를 받는 애플리케이션 또는 서비스에 의해 해당 대상에서 데이터를 암호화하는 것입니다.

Amazon S3는 새 객체 업로드에 대해 Amazon S3 관리형 키(SSE-S3)를 통한 서버 측 암호화를 자동으로 활성화합니다.

달리 지정하지 않는 한 버킷은 기본적으로 SSE-S3를 사용하여 객체를 암호화합니다. 하지만 AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 통한 서버 측 암호화를 사용하도록 버킷을 구성할 수도 있습니다. 자세한 내용은 AWS KMS(SSE-KMS)를 사용한 서버 측 암호화 지정 섹션을 참조하세요.

AWS KMS는 클라우드에 맞게 확장된 키 관리 시스템을 제공하기 위해 안전하고 가용성이 높은 하드웨어 및 소프트웨어를 결합하는 서비스입니다. Amazon S3는 AWS KMS를 사용한 서버 측 암호화(SSE-KMS)를 사용하여 S3 객체 데이터를 암호화합니다. 또한, 객체에 SSE-KMS를 요청하면 S3 체크섬(객체 메타데이터의 일부)이 암호화된 형식으로 저장됩니다. 체크섬에 대한 자세한 내용은 Amazon S3에서 객체 무결성 확인 페이지를 참조하십시오.

KMS 키를 사용하는 경우 AWS Management Console 또는 AWS KMS API를 통해 AWS KMS를 사용하여 다음을 수행할 수 있습니다.

- KMS 키를 중앙에서 생성, 조회, 편집, 모니터링, 활성화/비활성화, 순환, 예약 삭제할 수 있습니다.
- KMS 키를 사용할 수 있는 방법 및 주체를 제어하는 정책을 정의합니다.
- 올바른 사용을 위해 KMS 키 사용을 감사합니다. 감사는 AWS KMS 콘솔이 아닌 AWS KMS API로 지원됩니다.

AWS KMS 내 보안 관리가 암호화 관련 규정 준수 요구 사항을 충족하는 데 도움이 될 수 있습니다. 이러한 KMS 키를 사용하여 Amazon S3 버킷에서 데이터를 보호할 수 있습니다. S3 버킷에 SSE-KMS 암호화를 사용할 때 AWS KMS keys는 버킷과 동일한 리전에 있어야 합니다.

AWS KMS keys 사용에 따르는 추가 요금이 있습니다. 자세한 내용은 AWS Key Management Service 개발자 가이드의 AWS KMS key 개념 및 AWS KMS 요금을 참조하십시오.

IAM 사용자가 KMS 암호화 버킷에 액세스하도록 허용하는 방법에 대한 지침은 Amazon S3 버킷에 사용자 지정 AWSAWS KMS 키를 사용하는 기본 암호화가 설정되어 있습니다. 사용자가 버킷에서 다운로드 및 업로드를 수행할 수 있도록 하려면 어떻게 해야 하나요?를 AWS re:Post 지식 센터에서 참조하세요.

권한

Amazon S3에 대한 AWS KMS 키로 객체를 암호화하도록 성공적으로 `PutObject` 요청을 수행하려면 키에 대한 `kms:GenerateDataKey` 권한이 필요합니다. AWS KMS key를 사용하여 암호화된 객체

를 다운로드하려면 키에 대한 `kms:Decrypt` 권한이 필요합니다. 멀티파트 업로드를 수행하여 AWS KMS key로 객체를 암호화하려면 키에 대한 `kms:GenerateDataKey` 및 `kms:Decrypt` 권한이 있어야 합니다.

> **Important**
>
> KMS 키 정책에 부여된 권한을 주의 깊게 검토하세요. 고객 관리형 KMS 키 정책 권한은 항상 관련 AWS KMS 키 작업에 액세스해야 하는 IAM 보안 주체 및 AWS 서비스로 제한하세요. 자세한 내용은 AWS KMS에서 키 정책을 참조하세요.

주제

- AWS KMS keys
- Amazon S3 버킷 키
- 서버 측 암호화 필요
- 암호화 컨텍스트
- AWS KMS 암호화된 객체에 대한 요청 전송
- AWS KMS(SSE-KMS)를 사용한 서버 측 암호화 지정
- Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감

AWS KMS keys

AWS KMS를 통한 서버 측 암호화(SSE-KMS)를 사용하는 경우 기본 AWS 관리형 키를 사용하거나 이미 생성한 고객 관리형 키를 지정할 수 있습니다. AWS KMS는 봉투 암호화를 사용합니다. S3는 봉투 암호화에 AWS KMS 기능을 사용하여 데이터를 추가로 보호합니다. 봉투 암호화는 데이터 키로 일반 텍스트 데이터를 암호화한 후 KMS 키로 데이터 키를 암호화하는 방법입니다. 봉투 암호화에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서에서 봉투 암호화를 참조하십시오.

고객 관리형 키를 지정하지 않으면 Amazon S3에서는 SSE-KMS로 암호화된 객체를 버킷에 처음 추가할 때 AWS 계정에 AWS 관리형 키를 자동으로 생성합니다. 기본적으로 Amazon S3에서는 SSE-KMS에 이 KMS를 사용합니다.

> **Note**
>
> AWS 관리형 키로 SSE-KMS를 사용하여 암호화된 객체는 계정 간에 공유할 수 없습니다. 계정 간 SSE-KMS 데이터를 공유해야 하는 경우 AWS KMS의 고객 관리형 키를 사용해야 합니다.

SSE-KMS에 고객 관리형 키를 사용하려는 경우 SSE-KMS를 구성하기 전에 대칭 암호화 고객 관리형 키를 생성합니다. 그런 다음 버킷에 대해 SSE-KMS를 구성할 때 기존 고객 관리형 키를 지정합니다. 대칭 암호화 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 대칭 암호화 KMS 키를 참조하십시오.

고객 관리형 키를 생성하면 보다 유연하게 제어할 수 있습니다. 예를 들어, 고객 관리형 키를 생성, 교체 및 사용 중지할 수 있습니다. 또한 액세스 제어를 정의하고 데이터를 보호하는 데 사용하는 고객 관리형 키를 감사할 수 있습니다. 고객 관리형 및 AWS 관리형 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 키 및 AWS 키를 참조하십시오.

> **Note**
>
> 외부 키 스토어에 저장된 고객 관리형 키로 서버 측 암호화를 사용하면, 표준 KMS 키를 사용할 때와는 달리, 키 구성 요소의 가용성과 내구성을 직접 담보할 책임이 있습니다. 외부 키 스토어와 이들이 공유 책임 모델을 전환하는 방법에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 External key stores(외부 키 스토어)를 참조하십시오.

크로스 계정 작업에 SSE-KMS 암호화 사용

크로스 계정 작업에 암호화를 사용하는 경우 다음 사항에 유의하세요.

- 요청 시 또는 버킷의 기본 암호화 구성을 통해 AWS KMS key Amazon 리소스 이름(ARN) 또는 별칭이 제공되지 않은 경우 AWS 관리형 키(`aws/s3`)가 사용됩니다.
- KMS 키와 동일한 AWS 계정에 있는 AWS Identity and Access Management(IAM) 보안 주체를 사용하여 S3 객체를 업로드하거나 액세스하는 경우 AWS 관리형 키(`aws/s3`)를 사용할 수 있습니다.
- S3 객체에 크로스 계정 액세스 권한을 부여하려면 고객 관리형 키를 사용합니다. 다른 계정으로부터의 액세스를 허용하도록 고객 관리형 키의 정책을 구성할 수 있습니다.
- 고객 관리형 KMS 키를 지정하는 경우 정규화된 KMS 키 ARN을 사용하는 것이 좋습니다. KMS 키 별칭을 대신 사용하는 경우 AWS KMS가 요청자의 계정 내에서 키를 해독합니다. 이 동작으로 인해 버킷 소유자가 아닌 요청자에게 속한 KMS로 데이터가 암호화될 수 있습니다.
- 사용자(요청자)에게 `Encrypt` 권한이 부여된 키를 지정해야 합니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 키 사용자가 암호화 작업에 KMS를 사용하도록 허용을 참조하세요.

고객 관리형 키와 AWS 관리형 KMS 키를 사용해야 하는 상황에 대한 자세한 내용은 Amazon S3의 객체를 암호화하기 위해 AWS 관리형 키와 고객 관리형 KMS 키 중 무엇을 사용해야 합니까?를 참조하세요.

SSE-KMS 암호화 워크플로

AWS 관리형 키 또는 고객 관리형 키를 사용하여 데이터를 암호화하도록 선택하면, AWS KMS 및 Amazon S3가 다음 봉투 암호화 작업을 수행합니다.

1. Amazon S3는 일반 텍스트 데이터 키와 지정된 KMS 키로 암호화된 키의 사본을 요청합니다.
2. AWS KMS가 데이터 키를 생성하고 KMS 키 하에서 암호화한 후, 일반 텍스트 데이터 키와 암호화된 데이터 키를 모두 Amazon S3로 보냅니다.
3. Amazon S3는 데이터 키를 사용해 데이터를 암호화하고, 사용 후 가급적 빨리 메모리에서 일반 텍스트 키를 제거합니다.
4. Amazon S3는 암호화한 데이터 키를 암호화한 데이터와 함께 메타데이터로 저장합니다.

사용자가 데이터 복호화를 요청하면 Amazon S3와 AWS KMS가 다음 작업을 수행합니다.

1. Amazon S3가 `Decrypt` 요청 내 AWS KMS로 암호화된 데이터 키를 보냅니다.
2. AWS KMS가 동일한 KMS 키를 사용하여 암호화된 데이터 키를 복호화하고 일반 텍스트 데이터 키를 Amazon S3에 반환합니다.
3. Amazon S3가 암호화된 데이터를 일반 텍스트 데이터 키를 사용해 복호화하고, 가급적 신속하게 일반 텍스트 데이터 키를 메모리에서 제거합니다.

> **Important**
>
> Amazon S3에서 서버 측 암호화에 AWS KMS key을 사용하는 경우 대칭 암호화 KMS 키를 선택해야 합니다. Amazon S3는 대칭 암호화 KMS 키만 지원합니다. 이들 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 대칭 암호화 KMS 키를 참조하십시오.

SSE-KMS 암호화 감사

SSE-KMS를 지정하는 요청을 식별하려면 Amazon S3 스토리지 렌즈 지표 내에서 All SSE-KMS requests(모든 SSE-KMS 요청) 및 % all SSE-KMS requests(모든 SSE-KMS 요청 비율) 지표를 사용하면 됩니다. S3 스토리지 렌즈는 조직 전반에서 객체 스토리지 사용 및 활동에 대한 가시성을 확보하는 데 사용할 수 있는 클라우드 스토리지 분석 기능입니다. SSE-KMS 활성화 버킷 수 및 SSE-KMS 활성

화 버킷 비율을 사용하여 기본 버킷 암호화에 대한 SSE-KMS 활성화 버킷 수를 파악할 수도 있습니다. 자세한 내용은 S3 스토리지 렌즈를 사용한 스토리지 활동 및 사용량 평가를 참조하십시오. 지표의 전체 목록은 S3 스토리지 렌즈 지표 용어집을 참조하십시오.

AWS CloudTrail 로그를 사용하여 SSE-KMS 암호화 데이터의 AWS KMS 키 사용을 감사할 수 있습니다. GenerateDataKey 및 Decrypt와 같은 암호화 작업에 대한 인사이트를 얻을 수 있습니다. CloudTrail은 이벤트 이름, 사용자 이름, 이벤트 소스 등 검색 필터링을 위한 다양한 속성 값을 지원합니다.

Amazon S3 버킷 키

AWS KMS(SSE-KMS)를 통한 서버 측 암호화를 구성할 때 SSE-KMS용 S3 버킷 키를 사용하도록 버킷을 구성할 수 있습니다. SSE-KMS용 버킷 수준 키를 사용하면 Amazon S3에서 AWS KMS로 가는 요청 트래픽을 줄여 AWS KMS 요청 비용을 최대 99%까지 줄일 수 있습니다.

새 객체에서 SSE-KMS용 S3 버킷 키를 사용하도록 버킷을 구성하면 AWS KMS는 버킷의 객체에 대해 고유한 데이터 키를 만드는 데 사용되는 버킷 수준 키를 생성합니다. 이 S3 버킷 키는 Amazon S3 내에서 제한된 기간 동안 사용되므로 Amazon S3가 암호화 작업을 완료하기 위해 AWS KMS에 요청할 필요성이 추가적으로 줄어듭니다. S3 버킷 키 사용에 대한 자세한 내용은 Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감 단원을 참조하십시오.

서버 측 암호화 필요

특정 Amazon S3 버킷의 모든 객체에 대한 서버 측 암호화를 요구하려면 버킷 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-aws-kms-key-id`) 권한을 거부합니다.

```
{
   "Version":"2012-10-17",
   "Id":"PutObjectPolicy",
   "Statement":[{
         "Sid":"DenyObjectsThatAreNotSSEKMS",
         "Effect":"Deny",
         "Principal":"*",
         "Action":"s3:PutObject",
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/*",
         "Condition":{
            "Null":{
               "s3:x-amz-server-side-encryption-aws-kms-key-id":"true"
            }
         }
      }
```

```
    ]
}
```

버킷의 객체를 암호화하는 데 특정 AWS KMS key를 사용하도록 하려면 `s3:x-amz-server-side-encryption-aws-kms-key-id` 조건 키를 사용합니다. `arn:aws:kms:region:acct-id:key/key-id` 형식의 Amazon 리소스 이름(ARN) 키를 사용해야 합니다. AWS Identity and Access Management은 문자열이 존재하는지 유효성을 검사하지 않습니다.

> **Note**
>
> 객체를 업로드할 때 `x-amz-server-side-encryption-aws-kms-key-id` 헤더를 사용하여 KMS 키를 지정하거나 기본 버킷 암호화 구성을 사용할 수 있습니다. PutObject 요청이 `x-amz-server-side-encryption` 헤더에 `aws:kms`를 지정하지만 `x-amz-server-side-encryption-aws-kms-key-id` 헤더는 지정하지 않는 경우 Amazon S3에서 사용자가 AWS 관리형 키를 사용하려는 것으로 간주합니다. 이와 관계없이 Amazon S3가 객체 암호화에 사용하는 AWS KMS 키 ID는 정책의 AWS KMS 키 ID와 일치해야 합니다. 그렇지 않으면 Amazon S3는 요청을 거부합니다.

Amazon S3에서 사용되는 조건 키의 전체 목록을 보려면 서비스 승인 참조에서 Amazon S3에 사용되는 조건 키를 참조하세요.

암호화 컨텍스트

암호화 컨텍스트는 데이터에 대한 추가 컨텍스트 정보를 포함하는 키-값 페어 집합입니다. 암호화 컨텍스트는 암호화되지 않습니다. 암호화 작업에 대해 암호화 컨텍스트가 지정되면 Amazon S3가 복호화 작업에 대해 동일한 암호화 컨텍스트를 지정해야 합니다. 그렇지 않으면 암호화 해제가 실패합니다. AWS KMS는 암호화 컨텍스트를 추가 인증 데이터(AAD)로 사용하여 인증된 암호화를 지원합니다. 암호화 컨텍스트에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 암호화 컨텍스트를 참조하십시오.

기본적으로 Amazon S3는 객체 또는 버킷의 Amazon 리소스 이름(ARN)을 암호화 컨텍스트 쌍으로 사용합니다.

- S3 버킷 키를 활성화하지 않고 SSE-KMS를 사용하는 경우 객체 ARN이 암호화 컨텍스트로 사용됩니다.

```
arn:aws:s3:::object_ARN
```

- S3 버킷 키를 활성화하고 SSE-KMS를 사용하는 경우 버킷 ARN이 암호화 컨텍스트로 사용됩니다. S3 버킷 키에 대한 자세한 내용은 Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감 단원을 참조하십시오.

```
arn:aws:s3:::bucket_ARN
```

필요한 경우 s3:PutObject 요청에서 `x-amz-server-side-encryption-context` 헤더를 사용하여 추가 암호화 컨텍스트 쌍을 제공할 수 있습니다. 그러나 암호화 컨텍스트는 암호화되지 않으므로 민감한 정보를 포함하지 않도록 해야 합니다. Amazon S3는 기본 암호화 컨텍스트와 함께 이 추가 키 페어를 저장합니다. PUT 요청을 처리할 때 Amazon S3는 의 기본 암호화 컨텍스트를 사용자가 제공한 `aws:s3:arn` 암호화 컨텍스트에 추가합니다.

암호화 컨텍스트를 사용하여 암호화 작업을 식별하고 분류할 수 있습니다. 또한 기본 암호화 컨텍스트 ARN 값을 사용하면 어떤 Amazon S3 ARN이 어떤 암호화 키와 함께 사용되었는지 확인하여 AWS CloudTrail에서 관련 요청을 추적할 수 있습니다.

CloudTrail 로그 파일의 `requestParameters` 필드에서 암호화 컨텍스트는 다음과 비슷하게 표시됩니다.

```
"encryptionContext": {
    "aws:s3:arn": "arn:aws:s3:::amzn-s3-demo-bucket1/file_name"
}
```

S3 버킷 키 기능과 함께 SSE-KMS를 사용하는 경우 암호화 컨텍스트 값은 버킷의 ARN입니다.

```
"encryptionContext": {
    "aws:s3:arn": "arn:aws:s3:::amzn-s3-demo-bucket1"
}
```

AWS KMS 암호화된 객체에 대한 요청 전송

> **Important**
>
> AWS KMS 암호화된 객체에 대한 모든 GET 및 PUT 요청은 Secure Sockets Layer(SSL) 또는 전송 계층 보안(TLS)을 사용하여 이루어져야 합니다. 또한 AWS 서명 버전 4(또는 AWS 서명 버전 2)와 같은 유효한 자격 증명을 사용하여 요청에 서명해야 합니다.

AWS Signature 버전 4는 HTTP로 전송된 AWS 요청에 인증 정보를 추가하는 프로세스입니다. 보안을 위해 대부분의 AWS 요청은 액세스 키 ID와 비밀 액세스 키로 구성된 액세스 키로 서명해야 합니다. 이 두 키는 일반적으로 보안 자격 증명이라고 합니다. 자세한 내용은 요청 인증(AWS 서명 버전 4) 및 서명 버전 4 서명 프로세스를 참조하십시오.

> **Important**
>
> 객체가 SSE-KMS를 사용하는 경우 GET 요청 및 HEAD 요청에 암호화 요청 헤더를 전송하면 안 됩니다. 그러면 HTTP 400 Bad Request 오류가 발생합니다.

주제

- AWS KMS(SSE-KMS)를 사용한 서버 측 암호화 지정
- Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감

AWS KMS(SSE-KMS)를 사용한 서버 측 암호화 지정

모든 Amazon S3 버킷에는 기본적으로 암호화가 구성되어 있으며 S3 버킷에 업로드되는 신규 객체는 모두 저장 시 자동으로 암호화됩니다. Amazon S3 관리형 키(SSE-S3)를 사용한 서버 측 암호화가 Amazon S3 내 모든 버킷의 기본 암호화 구성입니다. 다른 유형의 암호화를 사용하려면 S3 PUT 요청에 사용할 서버 측 암호화 유형을 지정하거나 대상 버킷에 기본 암호화 구성을 업데이트할 수 있습니다.

PUT 요청에 다른 암호화 유형을 지정하려는 경우 AWS Key Management Service(AWS KMS) 키를 사용한 서버 측 암호화(SSE-KMS), AWS KMS 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS) 또는 고객 제공 키를 사용한 서버 측 암호화 (SSE-C)를 사용할 수 있습니다. 대상 버킷에 다른 기본 암호화 구성을 설정하려는 경우 SSE-KMS 또는 DSSE-KMS를 사용할 수 있습니다.

범용 버킷의 기본 암호화 구성을 변경하는 방법에 대한 자세한 내용은 기본 암호화 구성 섹션을 참조하세요.

버킷의 기본 암호화 구성을 SSE-KMS로 변경해도 버킷에 있는 기존 Amazon S3 객체의 암호화 유형은 변경되지 않습니다. 기본 암호화 구성을 SSE-KMS로 업데이트한 후 기존 객체의 암호화 유형을 변경하려는 경우 Amazon S3 Batch Operations를 사용할 수 있습니다. 객체 목록을 S3 Batch Operations에 제공하면 Batch Operations가 각각의 API 작업을 직접적으로 호출합니다. 객체 복사 작업을 사용하면 기존 객체를 복사하여 SSE-KMS로 암호화된 객체와 동일한 버킷에 다시 쓸 수 있습니다. 단일 배치 작업 건으로 수십억 개의 객체에서 지정된 작업을 수행할 수 있습니다. 자세한 내용은 Batch Operations를 사용하여 대량으로 객체 작업 수행 및 AWS Storage Blog 게시물 How to retroactively encrypt

existing objects in Amazon S3 using S3 Inventory, Amazon Athena, and S3 Batch Operations를 참조하세요.

Amazon S3 콘솔, REST API 작업, AWS SDK 및 AWS Command Line Interface(AWS CLI)를 사용하여 SSE-KMS를 지정할 수 있습니다. 자세한 내용은 다음 항목을 참조하세요.

> **Note**
>
> Amazon S3에서 다중 리전 AWS KMS keys를 사용할 수 있습니다. 그러나 Amazon S3는 현재 다중 리전 키를 단일 리전 키인 것처럼 취급하며, 키의 다중 리전 기능을 사용하지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 다중 리전 키 사용을 참조하세요.

> **Note**
>
> 다른 계정에서 소유한 KMS 키를 사용하려면 먼저 해당 키에 대한 사용 권한이 있어야 합니다. KMS 키의 크로스 계정 권한에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서에서 다른 계정에서 사용할 수 있는 KMS 키 만들기를 참조하십시오.

S3 콘솔 사용

이 주제에서는 Amazon S3 콘솔을 사용하여 AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 통한 서버 측 암호화를 사용하도록 객체 암호화 유형을 설정하거나 변경하는 방법을 설명합니다.

> **Note**
>
> - 객체의 용량이 5GB 미만인 경우 객체의 암호화를 변경할 수 있습니다. 객체가 5GB보다 큰 경우, 객체의 암호화를 변경하려면 AWS CLI 또는 AWS SDK를 사용해야 합니다.
> - 객체의 암호화를 변경하는 데 필요한 추가 권한 목록은 the section called “S3 API 작업에 필요한 권한” 섹션을 참조하세요. 이 권한을 부여하는 정책의 예시는 the section called “자격 증명 기반 정책 예제” 단원을 참조하세요.
> - 객체 암호화를 변경하면 새 객체가 생성되어 이전 객체를 대체합니다. S3 버전 관리가 사용 설정된 경우 객체의 새 버전이 생성되고 기존 객체는 이전 버전이 됩니다. 또한 속성을 변경하는 역할도 새 객체(또는 객체 버전)의 소유자가 됩니다.

객체에 대한 암호화 추가 또는 변경

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 탐색 창에서 버킷을 선택하고 범용 버킷 탭을 선택합니다. 변경할 객체가 포함된 Amazon S3 버킷 또는 폴더로 이동합니다.
3. 변경하려는 객체의 확인란을 선택합니다.
4. 작업 메뉴에 표시되는 옵션 목록에서 서버 측 암호화 편집을 선택합니다.
5. 서버 측 암호화 섹션으로 스크롤합니다.
6. 암호화 설정에서 버킷 기본 암호화 설정 사용 또는 기본 암호화에 버킷 설정 재정의를 선택합니다.

> **Important**
>
> 기본 암호화 구성에 대해 SSE-KMS 옵션을 사용할 경우 AWS KMS의 초당 요청 수(RPS) 제한이 적용됩니다. AWS KMS 할당량과 할당량 증대를 요청하는 방법에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 할당량을 참조하세요.

7. 기본 암호화에 버킷 설정 재정의를 선택한 경우 다음과 같은 암호화 설정을 구성해야 합니다.

   a. 암호화 유형에서 AWS Key Management Service 키를 사용한 서버 측 암호화(SSE-KMS)를 선택합니다.

   b. AWS KMS 키에서 다음 중 하나를 수행하여 KMS 키를 선택합니다.

      - 사용 가능한 KMS 키 목록에서 AWS KMS keys 중에서 선택을 선택한 다음, 사용 가능한 키 목록에서 KMS 키를 선택합니다.

        AWS 관리형 키(aws/s3)와 고객 관리형 키가 모두 목록에 표시됩니다. 고객 관리형 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 키 및 AWS 키를 참조하세요.

      - KMS 키 ARN을 입력하려면 AWS KMS key ARN 입력을 선택한 다음 나타나는 필드에 KMS 키 ARN을 입력합니다.

      - AWS KMS 콘솔에서 고객 관리형 키를 생성하려면 KMS 키 생성을 선택합니다.

        AWS KMS key 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 키 생성을 참조하십시오.

> **Important**
>
> 버킷과 동일한 AWS 리전에서 사용할 수 있는 KMS 키만 사용 가능합니다. Amazon S3 콘솔은 버킷과 동일한 리전에 있는 처음 100개의 KMS 키만 나열합니다. 목록에 없는 KMS 키를 사용하려면 KMS 키 ARN을 입력해야 합니다. 다른 계정에서 소유한 KMS 키를 사용하려면 먼저 해당 키에 대한 사용 권한이 있어야 하고, 다음 단계로 KMS 키 ARN을 입력해야 합니다.
> Amazon S3는 대칭 암호화 KMS 키만 지원하며 비대칭 KMS 키는 지원하지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 대칭 및 비대칭 KMS 키 식별을 참조하세요.

8. 추가 복사 설정에서 소스 설정 복사, 설정 지정 안 함 또는 설정 지정 중 원하는 것을 선택합니다. 소스 설정 복사가 기본 옵션입니다. 소스 설정 속성 없이 객체만 복사하려면 설정 지정 안 함을 선택합니다. 스토리지 클래스, ACL, 객체 태그, 메타데이터, 서버 측 암호화 및 추가 체크섬에 대한 설정을 지정하려면 설정 지정을 선택합니다.
9. 변경 사항 저장을 선택합니다.

> **Note**
>
> 이 작업은 지정된 모든 객체에 암호화를 적용합니다. 폴더를 암호화할 때 폴더에 새 객체를 추가하기 전에 저장 작업이 완료될 때까지 기다립니다.

REST API 사용

객체를 만들 때, 즉 새 객체를 업로드하거나 기존 객체를 복사할 때 AWS KMS keys(SSE-KMS)를 통한 서버 측 암호화를 사용하여 데이터를 암호화하도록 지정할 수 있습니다. 이렇게 하려면 요청에 `x-amz-server-side-encryption` 헤더를 추가합니다. 암호화 알고리즘 `aws:kms`로 헤더의 값을 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`

- `x-amz-server-side-encryption-bucket-key-enabled`

주제

- SSE-KMS를 지원하는 Amazon S3 REST API 작업
- 암호화 컨텍스트(x-amz-server-side-encryption-context)
- AWS KMS 키 ID(x-amz-server-side-encryption-aws-kms-key-id)
- S3 버킷 키(x-amz-server-side-encryption-aws-bucket-key-enabled)

SSE-KMS를 지원하는 Amazon S3 REST API 작업

다음 REST API 작업은 `x-amz-server-side-encryption`, `x-amz-server-side-encryption-aws-kms-key-id` 및 `x-amz-server-side-encryption-context` 요청 헤더를 수락합니다.

- PutObject - PUT API 작업을 사용하여 데이터를 업로드할 때 이러한 요청 헤더를 지정할 수 있습니다.
- CopyObject - 객체를 복사할 때 소스 객체 및 대상 객체가 모두 있어야 합니다. `CopyObject` 작업을 사용하여 SSE-KMS 헤더를 전달할 경우 헤더가 대상 객체에만 적용됩니다. 기존 객체를 복사할 때 소스 객체의 암호화 여부와 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.
- CreateMultipartUpload - 멀티파트 업로드 API 작업을 사용하여 대형 객체를 업로드할 때 이러한 헤더를 지정할 수 있습니다. `CreateMultipartUpload` 요청에서 이 헤더를 지정합니다.

객체가 서버 측 암호화를 사용하여 저장될 경우 다음 REST API 작업의 응답 헤더는 `x-amz-server-side-encryption` 헤더를 반환합니다.

- PutObject
- CopyObject
- POST Object
- CreateMultipartUpload
- UploadPart
- UploadPartCopy

- CompleteMultipartUpload
- GetObject
- HeadObject

> **Important**
>
> - 보안 소켓 계층(SSL), 전송 계층 보안(TLS) 또는 서명 버전 4를 사용하여 요청을 수행하지 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 추가 컨텍스트 정보를 포함하는 키-값 페어 집합입니다.

Amazon S3는 자동으로 객체 또는 버킷의 Amazon 리소스 이름(ARN)을 암호화 컨텍스트 쌍으로 사용합니다. S3 버킷 키를 활성화하지 않고 SSE-KMS를 사용하는 경우 객체 ARN을 암호화 컨텍스트로 사용합니다. 예: `arn:aws:s3:::`*`object_ARN`*. 그러나 SSE-KMS를 사용하고 S3 버킷 키를 활성화하는 경우 암호화 컨텍스트에 버킷 ARN을 사용합니다. 예: `arn:aws:s3:::`*`bucket_ARN`*.

필요한 경우 `x-amz-server-side-encryption-context` 헤더를 사용하여 추가 암호화 컨텍스트 쌍을 제공할 수 있습니다. 그러나 암호화 컨텍스트는 암호화되지 않으므로 민감한 정보를 포함하지 않도록 해야 합니다. Amazon S3는 기본 암호화 컨텍스트와 함께 이 추가 키 페어를 저장합니다.

Amazon S3의 암호화 컨텍스트에 대한 자세한 내용은 암호화 컨텍스트 단원을 참조하십시오. 암호화 컨텍스트에 대한 일반 내용은 AWS Key Management Service 개발자 안내서의 AWS Key Management Service 개념 - 암호화 컨텍스트를 참조하십시오.

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 관리형 키의 ID를 지정할 수 있습니다. `x-amz-server-side-encryption:aws:kms` 헤더를 지정하지만 `x-amz-server-side-encryption-aws-kms-key-id` 헤더를 제공하지 않는 경우 Amazon S3는 AWS 관리형 키(`aws/s3`)를 사용하여 데이터를 보호합니다. 고객 관리형 키를 사

용하려면 고객 관리형 키의 `x-amz-server-side-encryption-aws-kms-key-id` 헤더를 제공해야 합니다.

> **Important**
>
> Amazon S3에서 서버 측 암호화에 AWS KMS key을 사용하는 경우 대칭 암호화 KMS 키를 선택해야 합니다. Amazon S3는 대칭 암호화 KMS 키만 지원합니다. 이들 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 대칭 암호화 KMS 키를 참조하십시오.

S3 버킷 키(**`x-amz-server-side-encryption-aws-bucket-key-enabled`**)

`x-amz-server-side-encryption-aws-bucket-key-enabled` 요청 헤더를 사용하여 객체 수준에서 S3 버킷 키를 활성화하거나 비활성화할 수 있습니다. S3 버킷 키는 Amazon S3에서 AWS KMS로 가는 요청 트래픽을 줄여 AWS KMS 요청 비용을 줄입니다. 자세한 내용은 Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감 섹션을 참조하세요.

`x-amz-server-side-encryption:aws:kms` 헤더를 지정하되 `x-amz-server-side-encryption-aws-bucket-key-enabled` 헤더를 제공하지 않은 경우 객체는 대상 버킷에 대한 S3 버킷 키 설정을 사용하여 객체를 암호화합니다. 자세한 내용은 객체 수준에서 S3 버킷 키 구성 섹션을 참조하세요.

AWS CLI 사용

다음 예시 AWS CLI 명령을 사용하려면 *`user input placeholders`*를 실제 정보로 대체하세요.

새 객체를 업로드하거나 기존 객체를 복사할 때 AWS KMS 키를 사용하여 서버 측 암호화를 통해 데이터를 암호화하도록 지정할 수 있습니다. 이렇게 하려면 요청에 `--server-side-encryption aws:kms` 헤더를 추가합니다. `--ssekms-key-id` *`example-key-id`*를 사용하여, 생성해둔 고객 관리형 AWS KMS 키를 추가합니다. AWS KMS를 지정했지만 `--server-side-encryption aws:kms` 키 ID를 제공하지 않으면, Amazon S3가 AWS 관리형 키를 사용합니다.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key example-object-key --server-side-encryption aws:kms --ssekms-key-id example-key-id --body filepath
```

`--bucket-key-enabled` 또는 `--no-bucket-key-enabled`를 추가하면 PUT 또는 COPY 작업에서 Amazon S3 버킷 키를 추가로 활성화하거나 비활성화할 수 있습니다. Amazon S3 버킷 키는 Amazon S3에서 AWS KMS로 가는 요청 트래픽을 줄여 AWS KMS 요청 비용을 줄일 수 있습니다. 자세한 내용은 Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감을 참조하십시오.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key example-object-key --server-
side-encryption aws:kms --bucket-key-enabled --body filepath
```

암호화되지 않은 객체를 암호화하면, 해당 객체를 제자리에 복사하여 SSE-KMS를 사용할 수 있습니다.

```
aws s3api copy-object --bucket amzn-s3-demo-bucket --key example-object-key --
body filepath --bucket amzn-s3-demo-bucket --key example-object-key --sse aws:kms --
sse-kms-key-id example-key-id --body filepath
```

AWS SDK 사용

AWS SDK를 사용하는 경우 서버 측 암호화에 AWS KMS keys를 사용하도록 Amazon S3에 요청할 수 있습니다. 다음 예시에서는 Java 및 .NET용 AWS SDK와 함께 SSE-KMS를 사용하는 방법을 보여줍니다. 다른 SDK 사용에 대한 자세한 내용은 AWS 개발자 센터의 샘플 코드 및 라이브러리를 참조하세요.

> **Important**
>
> Amazon S3에서 서버 측 암호화에 AWS KMS key을 사용하는 경우 대칭 암호화 KMS 키를 선택해야 합니다. Amazon S3는 대칭 암호화 KMS 키만 지원합니다. 이들 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 대칭 암호화 KMS 키를 참조하십시오.

`CopyObject` 작업

객체를 복사할 때 동일한 요청 속성(`ServerSideEncryptionMethod` 및 `ServerSideEncryptionKeyManagementServiceKeyId`)을 추가하여 Amazon S3에서 AWS KMS key를 사용하도록 요청합니다. 객체 복사에 대한 자세한 내용은 객체 복사, 이동, 이름 변경 단원을 참조하십시오.

`PUT` 작업

Java

AWS SDK for Java를 사용하여 객체를 업로드할 때 다음 요청에 표시된 대로 `SSEAwsKeyManagementParams` 속성을 추가하여 Amazon S3에서 AWS KMS key를 사용하도록 요청할 수 있습니다.

```
PutObjectRequest putRequest = new PutObjectRequest(bucketName,
```

```
    keyName, file).withSSEAwsKeyManagementParams(new SSEAwsKeyManagementParams());
```

이 경우 Amazon S3는 AWS 관리형 키(aws/s3)를 사용합니다. 자세한 내용은 AWS KMS 키를 사용한 서버 측 암호화(SSE-KMS) 사용 섹션을 참조하세요. 필요한 경우 대칭 암호화 KMS 키를 생성하고 다음 예시와 같이 요청에서 지정할 수 있습니다.

```
PutObjectRequest putRequest = new PutObjectRequest(bucketName,
    keyName, file).withSSEAwsKeyManagementParams(new
 SSEAwsKeyManagementParams(keyID));
```

고객 관리형 키 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 AWS KMS API 프로그래밍을 참조하십시오.

객체를 업로드하는 사용 가능한 코드 예제는 다음 주제를 참조하십시오. 이 예시를 사용하려면 코드 예시를 업데이트하고 이전 코드 조각에서 표시된 바와 같이 암호화 정보를 제공해야 합니다.

- 단일 작업으로 객체를 업로드하려면 객체 업로드 단원을 참조하십시오.
- 상위 수준 또는 하위 수준의 멀티파트 업로드 API 작업을 사용하는 멀티파트 업로드에 대해서는 멀티파트 업로드를 사용한 객체 업로드 섹션을 참조하세요.

.NET

AWS SDK for .NET를 사용하여 객체를 업로드할 때 다음 요청에 표시된 대로 ServerSideEncryptionMethod 속성을 추가하여 Amazon S3에서 AWS KMS key를 사용하도록 요청할 수 있습니다.

```
PutObjectRequest putRequest = new PutObjectRequest
 {
     BucketName = amzn-s3-demo-bucket,
     Key = keyName,
     // other properties
     ServerSideEncryptionMethod = ServerSideEncryptionMethod.AWSKMS
 };
```

이 경우 Amazon S3는 AWS 관리형 키를 사용합니다. 자세한 내용은 AWS KMS 키를 사용한 서버 측 암호화(SSE-KMS) 사용 섹션을 참조하세요. 필요한 경우 자체적인 대칭 암호화 고객 관리형 키를 생성하고 다음 예시와 같이 요청에서 지정할 수 있습니다.

```
PutObjectRequest putRequest1 = new PutObjectRequest
```

```
{
  BucketName = amzn-s3-demo-bucket,
  Key = keyName,
  // other properties
  ServerSideEncryptionMethod = ServerSideEncryptionMethod.AWSKMS,
  ServerSideEncryptionKeyManagementServiceKeyId = keyId
};
```

고객 관리형 키 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 AWS KMS API 프로그래밍을 참조하십시오.

객체를 업로드하는 사용 가능한 코드 예제는 다음 주제를 참조하십시오. 이 예시를 사용하려면 코드 예시를 업데이트하고 이전 코드 조각에서 표시된 바와 같이 암호화 정보를 제공해야 합니다.

- 단일 작업으로 객체를 업로드하려면 객체 업로드 단원을 참조하십시오.
- 상위 수준 또는 하위 수준의 멀티파트 업로드 API 작업을 사용하는 멀티파트 업로드에 대해서는 멀티파트 업로드를 사용한 객체 업로드 섹션을 참조하세요.

미리 서명된 URL

Java

AWS KMS key를 사용하여 암호화된 객체에 대해 미리 서명된 URL을 생성할 때 다음 예시와 같이 서명 버전 4를 명시적으로 지정해야 합니다.

```
ClientConfiguration clientConfiguration = new ClientConfiguration();
clientConfiguration.setSignerOverride("AWSS3V4SignerType");
AmazonS3Client s3client = new AmazonS3Client(
        new ProfileCredentialsProvider(), clientConfiguration);
...
```

코드에 대한 예는 미리 서명된 URL을 통해 객체 공유 단원을 참조하세요.

.NET

AWS KMS key를 사용하여 암호화된 객체에 대해 미리 서명된 URL을 생성할 때 다음 예시와 같이 서명 버전 4를 명시적으로 지정해야 합니다.

```
AWSConfigs.S3Config.UseSignatureVersion4 = true;
```

코드에 대한 예는 미리 서명된 URL을 통해 객체 공유 단원을 참조하세요.

Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감

Amazon S3 버킷 키는 AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 사용하여 Amazon S3 서버 측 암호화 비용을 절감합니다. SSE-KMS용 버킷 수준 키를 사용하면 Amazon S3에서 AWS KMS로 가는 요청 트래픽을 줄여 AWS KMS 요청 비용을 최대 99%까지 줄일 수 있습니다. AWS Management Console에서 몇 번만 클릭하면 클라이언트 애플리케이션을 변경하지 않고도 새 객체에 대한 SSE-KMS 암호화에 S3 버킷 키를 사용하도록 버킷을 구성할 수 있습니다.

> **Note**
>
> S3 버킷 키는 AWS Key Management Service(AWS KMS) 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)에 지원되지 않습니다.

SSE-KMS용 S3 버킷 키

SSE-KMS로 암호화된 수백만 또는 수십억 개의 객체에 액세스하는 워크로드는 AWS KMS에 대한 대규모 볼륨 요청을 생성할 수 있습니다. S3 버킷 키 없이 SSE-KMS를 사용하여 데이터를 보호하는 경우 Amazon S3는 모든 객체에 대해 개별 AWS KMS 데이터 키를 사용합니다. 이 경우, Amazon S3는 KMS 암호화 객체에 대해 요청이 이루어질 때마다 AWS KMS를 호출합니다. SSE-KMS 작동 방식에 대한 자세한 내용은 AWS KMS 키를 사용한 서버 측 암호화(SSE-KMS) 사용 단원을 참조하십시오.

SSE-KMS에 S3 버킷 키를 사용하도록 버킷을 구성하면, AWS가 AWS KMS에서 수명이 짧은 버킷 수준 키를 생성하여 S3 내에 임시로 보관합니다. 이 버킷 수준 키는 수명 주기 동안 새 객체의 데이터 키를 생성합니다. S3 버킷 키는 Amazon S3 내에서 제한된 기간 동안 사용되므로 S3가 암호화 작업을 완료하기 위해 AWS KMS에 요청할 필요성이 줄어듭니다. 이렇게 하면 S3에서 AWS KMS로 가는 트래픽이 줄어들어, 이전 비용의 일부만 부담하면 Amazon S3의 AWS KMS 암호화 객체에 액세스할 수 있습니다.

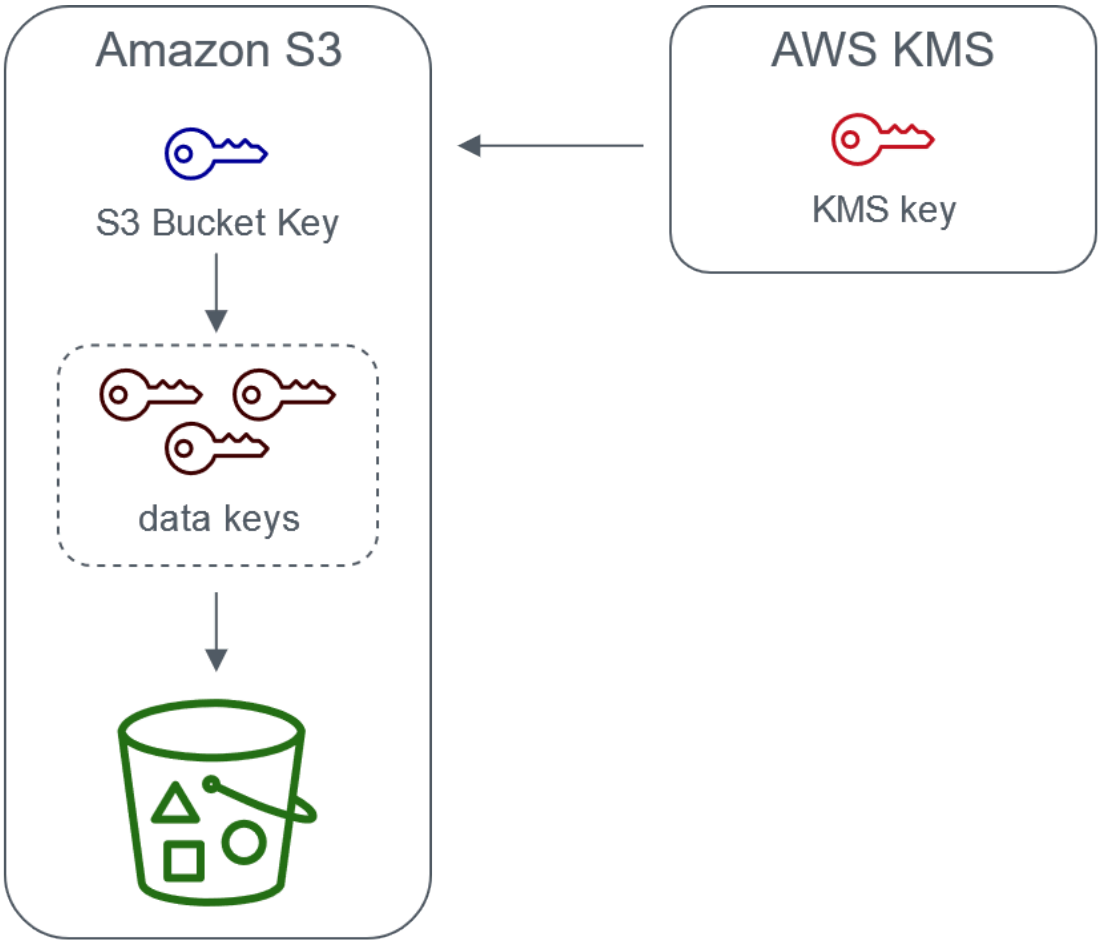
요청자의 키 액세스가 AWS KMS CloudTrail 이벤트에서 캡처되도록 요청자당 최소 한 번은 고유한 버킷 수준 키를 가져옵니다. Amazon S3는 호출자가 서로 다른 역할 또는 계정을 사용하거나 범위 지정 정책이 다른 동일한 역할을 사용하는 경우 호출자를 다른 요청자로 취급합니다. AWS KMS 요청 절감액에는 요청자 수, 요청 패턴 및 요청된 객체의 상대적 기간이 반영됩니다. 예를 들어, 요청자 수가 적고 제한된 시간 내에 여러 객체를 요청한다면, 동일한 버킷 수준 키로 암호화해도 절감 효과가 큽니다.

> **Note**
>
> S3 버킷 키를 사용하면 버킷 수준 키를 사용하여 `Encrypt`, `GenerateDataKey`, `Decrypt` 작업에 대한 AWS KMS 요청을 줄여 AWS KMS 요청 비용을 절감할 수 있습니다. 설계상 이 버킷

수준 키를 이용하는 후속 요청은 AWS KMS API 요청으로 이어지지 않으며 AWS KMS 키 정책에 대한 액세스를 검증하지 않습니다.

S3 버킷 키를 구성할 때 이미 버킷에 있는 객체에는 S3 버킷 키가 사용되지 않습니다. 기존 객체에 대해 S3 버킷 키를 구성하려면 `CopyObject` 작업을 사용하면 됩니다. 자세한 내용은 객체 수준에서 S3 버킷 키 구성 섹션을 참조하세요.

Amazon S3는 동일한 AWS KMS key로 암호화된 객체에 대해서만 S3 버킷 키를 공유합니다. S3 버킷 키는 AWS KMS에서 생성한 KMS 키, 가져온 키 구성 요소 및 사용자 지정 키 스토어가 지원하는 키 구성 요소와 호환됩니다.



Server-side encryption with AWS Key Management service using an S3 Bucket Key

S3 버킷 키 구성

Amazon S3 콘솔, AWS SDK, AWS CLI 또는 REST API를 통해 새 객체에 SSE-KMS용 S3 버킷 키를 사용하도록 버킷을 구성할 수 있습니다. 버킷에서 S3 버킷 키가 활성화되면, SSE-KMS 키가 별도로

지정된 채로 업로드된 객체는 자체 S3 버킷 키를 사용합니다. S3 버킷 키 설정과 무관하게 `true` 또는 `false` 값이 있는 `x-amz-server-side-encryption-bucket-key-enabled` 헤더를 요청에 포함하면 버킷 설정을 재정의할 수 있습니다.

S3 버킷 키를 사용하도록 버킷을 구성하기 전에 S3 버킷 키를 사용 설정하기 전에 유의할 변경 사항을(를) 검토하십시오.

Amazon S3 콘솔을 사용하여 S3 버킷 키 구성

새 버킷을 생성할 때 새 객체에 SSE-KMS용 S3 버킷 키를 사용하도록 버킷을 구성할 수 있습니다. 버킷 속성을 업데이트하여 새 객체에 SSE-KMS용 S3 버킷 키를 사용하도록 기존 버킷을 구성할 수도 있습니다.

자세한 내용은 새 객체에 SSE-KMS와 함께 S3 버킷 키를 사용하도록 버킷 구성 섹션을 참조하세요.

S3 버킷 키에 대한 REST API, AWS CLI 및 AWS SDK 지원

REST API, AWS CLI 또는 AWS SDK를 사용하여 새 객체에서 SSE-KMS용 S3 버킷 키를 사용하도록 버킷을 구성할 수 있습니다. 객체 수준에서 S3 버킷 키를 사용 설정할 수도 있습니다.

자세한 내용은 다음 자료를 참조하세요.

- 객체 수준에서 S3 버킷 키 구성
- 새 객체에 SSE-KMS와 함께 S3 버킷 키를 사용하도록 버킷 구성

다음 API 작업은 SSE-KMS에 대한 S3 버킷 키를 지원합니다.

- PutBucketEncryption
  - `ServerSideEncryptionRule`은(는) S3 버킷 키를 사용 설정 및 사용 중지하기 위한 `BucketKeyEnabled` 파라미터를 허용합니다.
- GetBucketEncryption
  - `ServerSideEncryptionRule`은(는) `BucketKeyEnabled`에 대한 설정을 반환합니다.
- PutObject, CopyObject, CreateMutlipartUpload 및 POST Object
  - `x-amz-server-side-encryption-bucket-key-enabled` 요청 헤더는 객체 수준에서 S3 버킷 키를 활성화하거나 비활성화합니다.
- HeadObject, GetObject, UploadPartCopy, UploadPart 및 CompleteMultipartUpload
  - `x-amz-server-side-encryption-bucket-key-enabled` 응답 헤더는 S3 버킷 키가 객체에 대해 활성화 또는 비활성화되었는지 여부를 나타냅니다.

AWS CloudFormation 작업

AWS CloudFormation에서 `AWS::S3::Bucket` 리소스에는 S3 버킷 키를 사용하거나 사용 중지하는 데 사용할 수 있는 `BucketKeyEnabled`라는 암호화 속성이 포함되어 있습니다.

자세한 내용은 AWS CloudFormation 사용하기 섹션을 참조하십시오.

S3 버킷 키를 사용 설정하기 전에 유의할 변경 사항

S3 버킷 키를 활성화하기 전에 다음과 같은 관련 변경 사항에 유의하십시오.

IAM 및 AWS KMS 키 정책

기존 AWS Identity and Access Management(IAM) 정책 또는 AWS KMS 키 정책이 객체 Amazon 리소스 이름(ARN)을 암호화 컨텍스트로 사용하여 KMS 키에 대한 액세스를 구체화하거나 제한하는 경우, 이러한 정책은 S3 버킷 키와 함께 작동하지 않습니다. S3 버킷 키는 버킷 ARN을 암호화 컨텍스트로 사용합니다. S3 버킷 키를 사용하기 전에 버킷 ARN을 암호화 컨텍스트로 사용하도록 IAM 정책 또는 AWS KMS 키 정책을 업데이트합니다.

암호화 컨텍스트 및 S3 버킷 키에 대한 자세한 내용은 암호화 컨텍스트 섹션을 참조하십시오.

AWS KMS에 대한 CloudTrail 이벤트

S3 버킷 키를 사용하면 AWS KMS CloudTrail 이벤트가 객체 ARN 대신 버킷 ARN을 기록합니다. 또한, 로그에 SSE-KMS 객체에 대한 KMS CloudTrail 이벤트가 줄어든 것을 확인할 수 있습니다. Amazon S3에서 키 구성 요소의 시간은 제한되어 있으므로 AWS KMS에 대한 요청 수가 줄어듭니다.

복제와 함께 S3 버킷 키 사용

S3 버킷 키를 동일 리전 복제(SRR) 및 교차 리전 복제(CRR)와 함께 사용할 수 있습니다.

Amazon S3는 암호화된 객체를 복제할 때 일반적으로 대상 버킷에 복제본 객체의 암호화 설정을 보존합니다. 그러나 원본 객체가 암호화되지 않고 대상 버킷에서 기본 암호화 또는 S3 버킷 키를 사용하는 경우, Amazon S3는 대상 버킷의 구성으로 객체를 암호화합니다.

다음 예에서는 S3 버킷 키가 복제와 함께 작동하는 방식을 보여 줍니다. 자세한 내용은 암호화(SSE-S3, SSE-KMS, DSSE-KMS, SSE-C)된 객체 복제 섹션을 참조하세요.

Example 예시 1 – 소스 객체가 S3 버킷 키를 사용하고 대상 버킷은 기본 암호화를 사용합니다.

원본 객체가 S3 버킷 키를 사용하지만 대상 버킷은 SSE-KMS와 함께 기본 암호화를 사용하는 경우, 복제본 객체는 대상 버킷에서 S3 버킷 키 암호화 설정을 유지합니다. 대상 버킷은 여전히 SSE-KMS와 함께 기본 암호화를 사용합니다.

Example 예시 2 – 소스 객체가 암호화되지 않았고 대상 버킷은 SSE-KMS와 함께 S3 버킷 키를 사용합니다.

소스 객체가 암호화되지 않았고 대상 버킷은 SSE-KMS와 함께 S3 버킷 키를 사용하는 경우, 복제본 객체는 대상 버킷에서 SSE-KMS와 함께 S3 버킷 키로 암호화됩니다. 결과적으로 원본 객체의 ETag는 복제본 객체의 ETag와 다릅니다. 이러한 차이점을 수용하도록 ETag를 사용하는 애플리케이션을 업데이트해야 합니다.

S3 버킷 키를 사용한 작업

S3 버킷 키 사용 설정 및 사용에 대한 자세한 내용은 다음 섹션을 참조하십시오.

- 새 객체에 SSE-KMS와 함께 S3 버킷 키를 사용하도록 버킷 구성
- 객체 수준에서 S3 버킷 키 구성
- S3 버킷 키에 대한 설정 보기

새 객체에 SSE-KMS와 함께 S3 버킷 키를 사용하도록 버킷 구성

AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 통해 서버 측 암호화를 구성할 때 새 객체에서 SSE-KMS용 S3 버킷 키를 사용하도록 버킷을 구성할 수 있습니다. S3 버킷 키는 Amazon S3에서 AWS KMS로 가는 요청 트래픽을 줄이고 SSE-KMS 비용을 절감합니다. 자세한 내용은 Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감 섹션을 참조하세요.

Amazon S3 콘솔, REST API, AWS SDK, AWS Command Line Interface(AWS CLI) 또는 AWS CloudFormation을 사용하여 새 객체에 SSE-KMS용 S3 버킷 키를 사용하도록 버킷을 구성할 수 있습니다. 기존 객체에 대해 S3 버킷 키를 사용하거나 사용 중지하려는 경우 `CopyObject` 작업을 사용할 수 있습니다. 자세한 내용은 객체 수준에서 S3 버킷 키 구성 및 Batch Operations를 사용하여 SSE-KMS용 S3 버킷 키 활성화 섹션을 참조하십시오.

원본 또는 대상 버킷에 대해 S3 버킷 키가 사용 설정되면 암호화 컨텍스트는 객체 ARN이 아니라 버킷 Amazon 리소스 이름(ARN)이 됩니다. 예: `arn:aws:s3:::`*`bucket_ARN`*. 암호화 컨텍스트에 버킷 ARN을 사용하려면 IAM 정책을 업데이트해야 합니다. 자세한 내용은 S3 버킷 키 및 복제 섹션을 참조하세요.

다음 예에서는 S3 버킷 키가 복제와 함께 작동하는 방식을 보여 줍니다. 자세한 내용은 암호화(SSE-S3, SSE-KMS, DSSE-KMS, SSE-C)된 객체 복제 섹션을 참조하세요.

사전 조건

S3 버킷 키를 사용하도록 버킷을 구성하기 전에 S3 버킷 키를 사용 설정하기 전에 유의할 변경 사항을(를) 검토하십시오.

S3 콘솔 사용

S3 콘솔에서 새 버킷 또는 기존 버킷에 대해 S3 버킷 키를 사용 설정하거나 사용 중지할 수 있습니다. S3 콘솔의 객체는 버킷 구성에서 S3 버킷 키 설정을 상속합니다. 버킷에 대해 S3 버킷 키를 활성화하면 버킷에 업로드하는 새 객체는 SSE-KMS용 S3 버킷 키를 사용합니다.

S3 버킷 키가 사용 설정된 버킷의 객체 업로드, 복사 또는 수정

S3 버킷 키가 활성화된 버킷의 객체를 업로드, 수정 또는 복사하는 경우 해당 객체에 대한 S3 버킷 키 설정이 버킷 구성에 맞게 업데이트될 수 있습니다.

객체에 이미 S3 버킷 키가 사용 설정되어 있으면 객체를 복사하거나 수정할 때 해당 객체에 대한 S3 버킷 키 설정이 변경되지 않습니다. 그러나 S3 버킷 키가 사용 설정되지 않은 객체를 수정하거나 복사하고 대상 버킷에 S3 버킷 키 구성이 있으면 객체는 대상 버킷의 S3 버킷 키 설정을 상속합니다. 예를 들어, 소스 객체에 S3 버킷 키가 활성화되어 있지 않지만 대상 버킷에 S3 버킷 키가 활성화되어 있으면 객체에 대해 S3 버킷 키가 활성화됩니다.

새 버킷을 생성할 때 S3 버킷 키 사용 설정

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 만들기를 선택합니다.
4. 버킷 이름을 입력하고 AWS 리전을 선택합니다.
5. 기본 암호화의 암호화 키 유형에서 AWS Key Management Service 키(SSE-KMS)를 선택합니다.
6. AWS KMS 키에서 다음 중 하나를 수행하여 KMS 키를 선택합니다.

   - 사용 가능한 KMS 키 목록에서 AWS KMS keys 중에서 선택을 선택한 다음, 사용 가능한 키 목록에서 KMS 키를 선택합니다.

     AWS 관리형 키(aws/s3)와 고객 관리형 키가 모두 목록에 표시됩니다. 고객 관리형 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 키 및 AWS 키를 참조하십시오.

   - KMS 키 ARN을 입력하려면 AWS KMS key ARN 입력을 선택하고 나타나는 필드에 KMS 키 ARN을 입력합니다.

- AWS KMS 콘솔에서 고객 관리형 키를 생성하려면 KMS 키 생성을 선택합니다.

  AWS KMS key 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 키 생성을 참조하십시오.

7. 버킷 키(Bucket Key)에서 사용(Enable)을 선택합니다.
8. 버킷 만들기를 선택합니다.

   Amazon S3는 S3 버킷 키가 사용 설정된 버킷을 생성합니다. 버킷에 업로드하는 새 객체는 S3 버킷 키를 사용합니다.

   S3 버킷 키를 사용 중지하려면 이전 단계를 따르고 사용 중지(Disable)를 선택합니다.

기존 버킷에 대해 S3 버킷 키 사용 설정

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷을 선택합니다.
3. 버킷(Buckets) 목록에서 S3 버킷 키를 사용 설정할 버킷을 선택합니다.
4. 속성(Properties) 탭을 선택합니다.
5. 기본 암호화에서 편집을 선택합니다.
6. 기본 암호화의 암호화 키 유형에서 AWS Key Management Service 키(SSE-KMS)를 선택합니다.
7. AWS KMS 키에서 다음 중 하나를 수행하여 KMS 키를 선택합니다.

   - 사용 가능한 KMS 키 목록에서 AWS KMS keys 중에서 선택을 선택한 다음, 사용 가능한 키 목록에서 KMS 키를 선택합니다.

     AWS 관리형 키(aws/s3)와 고객 관리형 키가 모두 목록에 표시됩니다. 고객 관리형 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 키 및 AWS 키를 참조하십시오.
   - KMS 키 ARN을 입력하려면 AWS KMS key ARN 입력을 선택하고 나타나는 필드에 KMS 키 ARN을 입력합니다.
   - AWS KMS 콘솔에서 고객 관리형 키를 생성하려면 KMS 키 생성을 선택합니다.

     AWS KMS key 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 키 생성을 참조하십시오.

8. 버킷 키(Bucket Key)에서 사용(Enable)을 선택합니다.
9. [변경 사항 저장(Save changes)]을 선택합니다.

Amazon S3는 버킷에 추가된 새 객체에 대해 S3 버킷 키를 사용 설정합니다. 기존 객체는 S3 버킷 키를 사용하지 않습니다. 기존 객체에 대해 S3 버킷 키를 구성하려면 `CopyObject` 작업을 사용하면 됩니다. 자세한 내용은 객체 수준에서 S3 버킷 키 구성 섹션을 참조하세요.

S3 버킷 키를 사용 중지하려면 이전 단계를 따르고 사용 중지(Disable)를 선택합니다.

REST API 사용

PutBucketEncryption을 선택하여 버킷에 대한 S3 버킷 키를 사용하거나 사용 중지할 수 있습니다. `PutBucketEncryption`에서 S3 버킷 키를 구성하려면 ServerSideEncryptionRule 데이터 유형을 사용합니다. 여기에는 SSE-KMS를 사용하는 기본 암호화가 포함되어 있습니다. 원하면 고객 관리형 키에 대한 KMS 키 ID를 지정하여 고객 관리형 키를 사용할 수도 있습니다.

자세한 내용과 예제 구문에 대해서는 PutBucketEncryption을 참조하십시오.

Java용 AWS SDK 사용

다음 예시에서는 AWS SDK for Java를 사용하여 SSE-KMS 및 S3 버킷 키로 기본 버킷 암호화를 사용합니다.

Java

```
AmazonS3 s3client = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.DEFAULT_REGION)
    .build();

ServerSideEncryptionByDefault serverSideEncryptionByDefault = new
 ServerSideEncryptionByDefault()
    .withSSEAlgorithm(SSEAlgorithm.KMS);
ServerSideEncryptionRule rule = new ServerSideEncryptionRule()
    .withApplyServerSideEncryptionByDefault(serverSideEncryptionByDefault)
    .withBucketKeyEnabled(true);
ServerSideEncryptionConfiguration serverSideEncryptionConfiguration =
    new ServerSideEncryptionConfiguration().withRules(Collections.singleton(rule));

SetBucketEncryptionRequest setBucketEncryptionRequest = new
 SetBucketEncryptionRequest()
    .withServerSideEncryptionConfiguration(serverSideEncryptionConfiguration)
    .withBucketName(bucketName);

s3client.setBucketEncryption(setBucketEncryptionRequest);
```

AWS CLI 사용

다음 예시에서는 AWS CLI를 사용하여 SSE-KMS 및 S3 버킷 키로 기본 버킷 암호화를 사용합니다. *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws s3api put-bucket-encryption --bucket amzn-s3-demo-bucket --server-side-encryption-
configuration '{
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "aws:kms",
                    "KMSMasterKeyID": "KMS-Key-ARN"
                },
                "BucketKeyEnabled": true
            }
        ]
    }'
```

AWS CloudFormation 사용하기

AWS CloudFormation을 통해 S3 버킷 키를 구성하는 방법에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS::S3::Bucket ServerSideEncryptionRule을 참조하십시오.

객체 수준에서 S3 버킷 키 구성

REST API, AWS SDK 또는 AWS CLI를 사용하여 PUT 또는 COPY 작업을 수행할 때 `true` 또는 `false` 값이 있는 `x-amz-server-side-encryption-bucket-key-enabled` 요청 헤더를 추가하면 객체 수준에서 S3 버킷 키를 활성화하거나 비활성화할 수 있습니다. S3 버킷 키는 Amazon S3에서 AWS KMS로 가는 요청 트래픽을 줄여 AWS Key Management Service(AWS KMS)(SSE-KMS)를 사용한 서버 측 암호화 비용을 절감합니다. 자세한 내용은 Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감 섹션을 참조하세요.

PUT 또는 COPY 작업을 사용하여 객체에 대한 S3 버킷 키를 구성하면 Amazon S3는 해당 객체에 대한 설정만 업데이트합니다. 대상 버킷에 대한 S3 버킷 키 설정은 변경되지 않습니다. KMS로 암호화된 객체에 대한 PUT 또는 COPY 요청을 S3 버킷 키가 활성화된 버킷에 제출하면, 요청 헤더에서 키를 비활성화하지 않은 한 객체 수준 작업에서 자동으로 S3 버킷 키를 사용합니다. 객체에 S3 버킷 키를 지정하지 않으면 Amazon S3가 대상 버킷에 대한 S3 버킷 키 설정을 객체에 적용합니다.

사전 조건:

S3 버킷 키를 사용하도록 객체를 구성하기 전에 S3 버킷 키를 사용 설정하기 전에 유의할 변경 사항을(를) 검토하십시오.

주제

- Amazon S3 배치 작업
- REST API 사용
- Java용 AWS SDK(PutObject) 사용
- AWS CLI(PutObject) 사용

Amazon S3 배치 작업

기존 Amazon S3 객체를 암호화하기 위해 Amazon S3 배치 작업을 사용할 수 있습니다. 작업할 객체 목록을 S3 배치 작업에 제공하면 배치 작업은 각각의 API를 호출하여 지정된 작업을 수행합니다.

S3 배치 작업 복사 작업을 사용하여 암호화되지 않은 기존 객체를 복사하고 암호화된 새로운 객체를 동일한 버킷에 작성할 수 있습니다. 단일 배치 작업 건으로 수십억 개의 객체에서 지정된 작업을 수행할 수 있습니다. 자세한 내용은 Batch Operations를 사용하여 대량으로 객체 작업 수행 및 Amazon S3 Batch Operations에서 객체 암호화를 참조하십시오.

REST API 사용

SSE-KMS를 사용하면 다음 API 작업을 사용하여 객체에 대해 S3 버킷 키를 활성화할 수 있습니다.

- PutObject – 객체를 업로드하면 `x-amz-server-side-encryption-bucket-key-enabled` 요청 헤더를 지정하여 객체 레벨에서 S3 버킷 키를 사용 설정하거나 사용 중지할 수 있습니다.
- CopyObject – 객체를 복사하고 SSE-KMS를 구성하면 `x-amz-server-side-encryption-bucket-key-enabled` 요청 헤더를 지정하여 객체에 대한 S3 버킷 키를 사용 설정하거나 사용 중지할 수 있습니다.
- Post Object – `POST` 작업을 사용하여 객체를 업로드하고 SSE-KMS를 구성하면 `x-amz-server-side-encryption-bucket-key-enabled` 양식 필드를 사용하여 객체에 대한 S3 버킷 키를 활성화하거나 비활성화할 수 있습니다.
- CreateMultipartUpload – `CreateMultipartUpload` API 작업을 사용하여 대용량 객체를 업로드하고 SSE-KMS를 구성하면 `x-amz-server-side-encryption-bucket-key-enabled` 요청 헤더를 사용하여 객체에 대한 S3 버킷 키를 활성화하거나 비활성화할 수 있습니다.

객체 수준에서 S3 버킷 키를 사용 설정하려면 `x-amz-server-side-encryption-bucket-key-enabled` 요청 헤더를 포함합니다. SSE-KMS 및 REST API에 대한 자세한 내용은 REST API 사용 단원을 참조하십시오.

Java용 AWS SDK(PutObject) 사용

다음 예제를 사용하여 AWS SDK for Java를 사용해 객체 수준에서 S3 버킷 키를 구성할 수 있습니다.

Java

```
AmazonS3 s3client = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.DEFAULT_REGION)
    .build();

String bucketName = "amzn-s3-demo-bucket1";
String keyName = "key name for object";
String contents = "file contents";

PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, keyName,
 contents)
    .withBucketKeyEnabled(true);

s3client.putObject(putObjectRequest);
```

AWS CLI(PutObject) 사용

다음 AWS CLI 예제를 사용하여 `PutObject` 요청의 일부로 객체 수준에서 S3 버킷 키를 구성할 수 있습니다.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key object key name --server-side-
encryption aws:kms --bucket-key-enabled --body filepath
```

S3 버킷 키에 대한 설정 보기

Amazon S3 콘솔, REST API, AWS Command Line Interface(AWS CLI) 또는 AWS SDK를 사용하여 버킷 수준 또는 객체 수준에서 S3 버킷 키에 대한 설정을 볼 수 있습니다.

S3 버킷 키는 Amazon S3에서 AWS KMS로 가는 요청 트래픽을 줄여 AWS Key Management Service(SSE-KMS)를 사용한 서버 측 암호화 비용을 절감합니다. 자세한 내용은 Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감 섹션을 참조하세요.

버킷 구성에서 S3 버킷 키 설정을 상속받은 버킷 또는 객체에 대한 S3 버킷 키 설정을 보려면 `s3:GetEncryptionConfiguration` 작업을 수행할 권한이 필요합니다. 자세한 내용은 Amazon Simple Storage Service API Reference의 GetBucketEncryption을 참조하세요.

S3 콘솔 사용

S3 콘솔에서 버킷 또는 객체에 대한 S3 버킷 키 설정을 볼 수 있습니다. S3 버킷 키 설정은 원본 객체에 S3 버킷 키가 이미 구성되어 있는 경우를 제외하고는 버킷 구성에서 상속됩니다.

동일한 버킷의 객체와 폴더는 다른 S3 버킷 키 설정을 가질 수 있습니다. 예를 들어, REST API를 사용하여 객체를 업로드하고 객체에 대해 S3 버킷 키를 사용 설정하는 경우, 대상 버킷에서 S3 버킷 키가 사용 중지되어 있더라도 객체는 대상 버킷에 S3 버킷 키 설정을 유지합니다. 또 다른 예로, 기존 버킷에 대해 S3 버킷 키를 사용 설정하면 이미 버킷에 있는 객체는 S3 버킷 키를 사용하지 않습니다. 그러나 새 객체에는 S3 버킷 키가 사용 설정되어 있습니다.

버킷에 대한 S3 버킷 키 설정 보기

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷(Buckets) 목록에서 S3 버킷 키를 사용 설정할 버킷을 선택합니다.
4. [속성(Properties)]을 선택합니다.
5. 기본 암호화(Default encryption) 섹션의 버킷 키(Bucket Key) 아래에 버킷의 S3 버킷 키 설정이 표시됩니다.

   S3 버킷 키 설정이 표시되지 않으면 사용자는 `s3:GetEncryptionConfiguration` 작업을 수행할 권한이 없을 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API Reference의 GetBucketEncryption을 참조하십시오.

객체에 대한 S3 버킷 키 설정 보기

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷(Buckets) 목록에서 S3 버킷 키를 사용 설정할 버킷을 선택합니다.
3. 객체(Objects) 목록에서 사용자의 객체 이름을 선택합니다.
4. 세부 정보(Details) 탭의 서버 측 암호화 설정(Server-side encryption settings)에서 편집(Edit)을 선택합니다.

   버킷 키에서 객체에 대한 S3 버킷 키 설정을 볼 수 있습니다. 이 설정은 편집할 수 없습니다.

AWS CLI 사용

버킷 수준의 S3 버킷 키 설정 반환

이 예제를 사용하려면 각 *user input placeholder*를 사용자의 정보로 대체합니다.

```
aws s3api get-bucket-encryption --bucket amzn-s3-demo-bucket1
```

자세한 내용은 AWS CLI Command Reference의 get-bucket-encryption을 참조하십시오.

객체 수준 S3 버킷 키 설정 반환

이 예제를 사용하려면 각 *user input placeholder*를 사용자의 정보로 대체합니다.

```
aws s3api head-object --bucket amzn-s3-demo-bucket1 --key my_images.tar.bz2
```

자세한 내용은 AWS CLI Command Reference의 head-object를 참조하십시오.

REST API 사용

버킷 수준의 S3 버킷 키 설정 반환

S3 버킷 키 설정을 포함하여 버킷에 대한 암호화 정보를 반환하려면 `GetBucketEncryption` 작업을 사용합니다. S3 버킷 키 설정은 `BucketKeyEnabled` 설정과 함께 `ServerSideEncryptionConfiguration` 요소의 응답 본문에 반환됩니다. 자세한 내용은 Amazon S3 API Reference의 GetBucketEncryption을 참조하십시오.

S3 버킷 키에 대한 객체 수준 설정 반환

객체에 대한 S3 버킷 키 상태를 반환하려면 `HeadObject` 작업을 사용합니다. `HeadObject`은(는) 객체에 대해 S3 버킷 키가 사용 설정되었는지 또는 사용 중지되었는지를 표시하는 `x-amz-server-side-encryption-bucket-key-enabled` 응답 헤더를 반환합니다. 자세한 내용은 Amazon S3 API Reference에서 HeadObject를 참조하십시오.

객체에 대해 S3 버킷 키가 구성된 경우 다음 API 작업도 `x-amz-server-side-encryption-bucket-key-enabled` 응답 헤더를 반환합니다.

- PutObject
- PostObject
- CopyObject
- CreateMultipartUpload

- UploadPartCopy
- UploadPart
- CompleteMultipartUpload
- GetObject

AWS KMS 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS) 사용

AWS Key Management Service(AWS KMS) 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)를 사용하면 Amazon S3에 객체를 업로드할 때 객체에 두 계층의 암호화가 적용됩니다. DSSE-KMS를 사용하면 데이터에 다중 계층 암호화를 적용하고 암호화 키를 완전하게 제어할 것을 요구하는 규정 준수 표준을 보다 쉽게 충족할 수 있습니다.

Amazon S3 버킷에 DSSE-KMS를 사용할 때 AWS KMS 키는 버킷과 동일한 리전에 있어야 합니다. 또한, 객체에 DSSE-KMS를 요청하면 객체 메타데이터의 일부인 S3 체크섬이 암호화된 형식으로 저장됩니다. 체크섬에 대한 자세한 내용은 Amazon S3에서 객체 무결성 확인 섹션을 참조하십시오.

DSSE-KMS 및 AWS KMS keys 사용에는 추가 요금이 부과됩니다. 자세한 내용은 AWS Key Management Service 개발자 가이드의 AWS KMS key 개념 및 AWS KMS 요금을 참조하십시오.

> **Note**
>
> S3 버킷 키는 DSSE-KMS에서 지원되지 않습니다.

AWS KMS keys를 사용한 이중 계층 서버 측 암호화(DSSE-KMS) 요구

특정 Amazon S3 버킷의 모든 객체에 대한 이중 계층 서버 측 암호화를 요구하려면 버킷 정책을 사용하면 됩니다. 예를 들어, 다음 버킷 정책은 요청에 DSSE-KMS를 사용한 서버 측 암호화를 요청하는 `x-amz-server-side-encryption` 헤더가 포함되지 않을 경우 모든 사용자에게 객체 업로드(`s3:PutObject`) 권한을 거부합니다.

```
{
            "Version":"2012-10-17",
            "Id":"PutObjectPolicy",
            "Statement":[{
                  "Sid":"DenyUnEncryptedObjectUploads",
                  "Effect":"Deny",
                  "Principal":"*",
                  "Action":"s3:PutObject",
                  "Resource":"arn:aws:s3:::amzn-s3-demo-bucket/*",
```

```
                "Condition":{
                   "StringNotEquals":{
                      "s3:x-amz-server-side-encryption":"aws:kms:dsse"
                   }
                }
             }
          ]
       }
```

주제

- AWS KMS 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS) 지정

AWS KMS 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS) 지정

새 객체를 업로드하거나 기존 객체를 복사할 때 암호화를 적용할 수 있습니다.

Amazon S3 콘솔, Amazon S3 REST API, AWS Command Line Interface(AWS CLI)를 사용하여 DSSE-KMS를 지정할 수 있습니다. 자세한 내용은 다음 항목을 참조하세요.

> **Note**
>
> Amazon S3에서 다중 리전 AWS KMS keys를 사용할 수 있습니다. 그러나 Amazon S3는 현재 다중 리전 키를 단일 리전 키인 것처럼 취급하며, 키의 다중 리전 기능을 사용하지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 다중 리전 키 사용을 참조하세요.

> **Note**
>
> 다른 계정에서 소유한 KMS 키를 사용하려면 먼저 해당 키에 대한 사용 권한이 있어야 합니다. KMS 키의 크로스 계정 권한에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서에서 다른 계정에서 사용할 수 있는 KMS 키 만들기를 참조하십시오.

S3 콘솔 사용

이 섹션에서는 Amazon S3 콘솔을 사용하여 AWS Key Management Service(AWS KMS) 키를 통한 이중 계층 서버 측 암호화(DSSE-KMS)를 사용하도록 객체 암호화 유형을 설정하거나 변경하는 방법을 설명합니다.

> **Note**
>
> - 객체의 용량이 5GB 미만인 경우 객체의 암호화를 변경할 수 있습니다. 객체가 5GB보다 큰 경우, 객체의 암호화를 변경하려면 AWS CLI 또는 AWS SDK를 사용해야 합니다.
> - 객체의 암호화를 변경하는 데 필요한 추가 권한 목록은 the section called “S3 API 작업에 필요한 권한” 섹션을 참조하세요. 이 권한을 부여하는 정책의 예시는 the section called “자격 증명 기반 정책 예제” 단원을 참조하세요.
> - 객체 암호화를 변경하면 새 객체가 생성되어 이전 객체를 대체합니다. S3 버전 관리가 사용 설정된 경우 객체의 새 버전이 생성되고 기존 객체는 이전 버전이 됩니다. 또한 속성을 변경하는 역할도 새 객체(또는 객체 버전)의 소유자가 됩니다.

객체에 대한 암호화 추가 또는 변경

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 탐색 창에서 버킷을 선택하고 범용 버킷 탭을 선택합니다. 변경할 객체가 포함된 Amazon S3 버킷 또는 폴더로 이동합니다.
3. 변경하려는 객체의 확인란을 선택합니다.
4. 작업 메뉴에 표시되는 옵션 목록에서 서버 측 암호화 편집을 선택합니다.
5. 서버 측 암호화 섹션으로 스크롤합니다.
6. 암호화 설정에서 버킷 기본 암호화 설정 사용 또는 기본 암호화에 버킷 설정 재정의를 선택합니다.
7. 기본 암호화에 버킷 설정 재정의를 선택한 경우 다음과 같은 암호화 설정을 구성해야 합니다.

   a. 암호화 유형에서 AWS Key Management Service 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)를 선택합니다.

   b. AWS KMS 키에서 다음 중 하나를 수행하여 KMS 키를 선택합니다.

      - 사용 가능한 KMS 키 목록에서 AWS KMS keys 중에서 선택을 선택한 다음, 사용 가능한 키 목록에서 KMS 키를 선택합니다.

        AWS 관리형 키(aws/s3)와 고객 관리형 키가 모두 목록에 표시됩니다. 고객 관리형 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 키 및 AWS 키를 참조하세요.

- KMS 키 ARN을 입력하려면 AWS KMS key ARN 입력을 선택한 다음 나타나는 필드에 KMS 키 ARN을 입력합니다.
- AWS KMS 콘솔에서 고객 관리형 키를 생성하려면 KMS 키 생성을 선택합니다.

  AWS KMS key 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 키 생성을 참조하십시오.

> **Important**
>
> 버킷과 동일한 AWS 리전에서 사용할 수 있는 KMS 키만 사용 가능합니다. Amazon S3 콘솔은 버킷과 동일한 리전에 있는 처음 100개의 KMS 키만 나열합니다. 목록에 없는 KMS 키를 사용하려면 KMS 키 ARN을 입력해야 합니다. 다른 계정에서 소유한 KMS 키를 사용하려면 먼저 해당 키에 대한 사용 권한이 있어야 하고, 다음 단계로 KMS 키 ARN을 입력해야 합니다.
> Amazon S3는 대칭 암호화 KMS 키만 지원하며 비대칭 KMS 키는 지원하지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 비대칭 KMS 키 식별을 참조하십시오.

8. 버킷 키에서 비활성화를 선택합니다. S3 버킷 키는 DSSE-KMS에서 지원되지 않습니다.
9. 추가 복사 설정에서 소스 설정 복사, 설정 지정 안 함 또는 설정 지정 중 원하는 것을 선택합니다. 소스 설정 복사가 기본 옵션입니다. 소스 설정 속성 없이 객체만 복사하려면 설정 지정 안 함을 선택합니다. 스토리지 클래스, ACL, 객체 태그, 메타데이터, 서버 측 암호화 및 추가 체크섬에 대한 설정을 지정하려면 설정 지정을 선택합니다.
10. 변경 사항 저장을 선택합니다.

> **Note**
>
> 이 작업은 지정된 모든 객체에 암호화를 적용합니다. 폴더를 암호화할 때 폴더에 새 객체를 추가하기 전에 저장 작업이 완료될 때까지 기다립니다.

REST API 사용

객체를 만들 때, 즉 새 객체를 업로드하거나 기존 객체를 복사할 때 AWS KMS keys를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)를 사용하여 데이터를 암호화하도록 지정할 수 있습니다. 이렇게 하려면 요청에 `x-amz-server-side-encryption` 헤더를 추가합니다. 암호화 알고리즘

`aws:kms:dsse`로 헤더의 값을 설정합니다. Amazon S3는 `x-amz-server-side-encryption` 응답 헤더를 반환하여 객체가 DSSE-KMS 암호화를 사용하여 저장되었는지 확인합니다.

`aws:kms:dsse`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: SSEKMSKeyId`
- `x-amz-server-side-encryption-context: SSEKMSEncryptionContext`

주제

- DSSE-KMS를 지원하는 Amazon S3 REST API 작업
- 암호화 컨텍스트(x-amz-server-side-encryption-context)
- AWS KMS 키 ID(x-amz-server-side-encryption-aws-kms-key-id)

DSSE-KMS를 지원하는 Amazon S3 REST API 작업

다음 REST API 작업은 `x-amz-server-side-encryption`, `x-amz-server-side-encryption-aws-kms-key-id` 및 `x-amz-server-side-encryption-context` 요청 헤더를 수락합니다.

- PutObject - PUT API 작업을 사용하여 데이터를 업로드할 때 이러한 요청 헤더를 지정할 수 있습니다.
- CopyObject - 객체를 복사할 때 소스 객체 및 대상 객체가 모두 있어야 합니다. `CopyObject` 작업을 사용하여 DSSE-KMS 헤더를 전달할 경우 헤더가 대상 객체에만 적용됩니다. 기존 객체를 복사할 때 소스 객체의 암호화 여부에 관계없이 명시적으로 서버 측 암호화를 요청하지 않는 한 대상 객체는 암호화되지 않습니다.
- POST 객체 - `POST` 작업을 사용하여 객체를 업로드할 경우에는 요청 헤더 대신 양식 필드에 동일한 정보를 제공합니다.
- CreateMultipartUpload - 멀티파트 업로드를 사용하여 대형 객체를 업로드할 때 `CreateMultipartUpload` 요청에 이러한 헤더를 지정할 수 있습니다.

객체가 서버 측 암호화를 사용하여 저장될 경우 다음 REST API 작업의 응답 헤더는 `x-amz-server-side-encryption` 헤더를 반환합니다.

- PutObject
- CopyObject

- POST 객체
- CreateMultipartUpload
- UploadPart
- UploadPartCopy
- CompleteMultipartUpload
- GetObject
- HeadObject

> **Important**
>
> - 보안 소켓 계층(SSL), 전송 계층 보안(TLS) 또는 서명 버전 4를 사용하여 요청을 수행하지 않으면 AWS KMS로 보호되는 객체에 대한 모든 GET 및 PUT 요청이 실패합니다.
> - 객체가 DSSE-KMS를 사용하는 경우 GET 요청 및 HEAD 요청에 대해 암호화 요청 헤더를 전송하지 마십시오. 전송하면 HTTP 400 (Bad Request) 오류가 발생합니다.

암호화 컨텍스트(**x-amz-server-side-encryption-context**)

`x-amz-server-side-encryption:aws:kms:dsse`를 지정하면 Amazon S3 API는 `x-amz-server-side-encryption-context` 헤더가 있는 암호화 컨텍스트를 지원합니다. 암호화 컨텍스트는 데이터에 대한 추가 컨텍스트 정보를 포함하는 키-값 페어 집합입니다.

Amazon S3는 자동으로 객체의 Amazon 리소스 이름(ARN)을 암호화 컨텍스트 쌍으로 사용합니다(예: `arn:aws:s3:::`*`object_ARN`*).

필요한 경우 `x-amz-server-side-encryption-context` 헤더를 사용하여 추가 암호화 컨텍스트 쌍을 제공할 수 있습니다. 그러나 암호화 컨텍스트는 암호화되지 않으므로 민감한 정보를 포함하지 않도록 해야 합니다. Amazon S3는 기본 암호화 컨텍스트와 함께 이 추가 키 페어를 저장합니다.

Amazon S3의 암호화 컨텍스트에 대한 자세한 내용은 암호화 컨텍스트 단원을 참조하십시오. 암호화 컨텍스트에 대한 일반 내용은 AWS Key Management Service 개발자 안내서의 AWS Key Management Service 개념 - 암호화 컨텍스트를 참조하십시오.

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 관리형 키의 ID를 지정할 수 있습니다. `x-amz-server-side-`

encryption:aws:kms:dsse 헤더를 지정하지만 x-amz-server-side-encryption-aws-kms-key-id 헤더를 제공하지 않는 경우 Amazon S3는 AWS 관리형 키(aws/s3)를 사용하여 데이터를 보호합니다. 고객 관리형 키를 사용하려면 고객 관리형 키의 x-amz-server-side-encryption-aws-kms-key-id 헤더를 제공해야 합니다.

> **Important**
>
> Amazon S3에서 서버 측 암호화에 AWS KMS key을 사용하는 경우 대칭 암호화 KMS 키를 선택해야 합니다. Amazon S3는 대칭 암호화 KMS 키만 지원합니다. 이들 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 대칭 암호화 KMS 키를 참조하십시오.

AWS CLI 사용

새 객체를 업로드하거나 기존 객체를 복사할 때 DSSE-KMS를 사용하여 데이터를 암호화하도록 지정할 수 있습니다. 이렇게 하려면 요청에 --server-side-encryption aws:kms:dsse 파라미터를 추가합니다. --ssekms-key-id *example-key-id* 파라미터를 사용하여, 생성해둔 고객 관리형 AWS KMS 키를 추가합니다. --server-side-encryption aws:kms:dsse를 지정하고 AWS KMS 키 ID를 제공하지 않으면 Amazon S3가 AWS 관리형 키(aws/s3)를 사용합니다.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key example-object-key --server-side-encryption aws:kms:dsse --ssekms-key-id example-key-id --body filepath
```

객체를 제자리에 복사하여 암호화되지 않은 객체가 DSSE-KMS를 사용하도록 암호화할 수 있습니다.

```
aws s3api copy-object --bucket amzn-s3-demo-bucket --key example-object-key --copy-source amzn-s3-demo-bucket/example-object-key --server-side-encryption aws:kms:dsse --ssekms-key-id example-key-id
```

고객 제공 키(SSE-C)로 서버 측 암호화 사용

서버 측 암호화는 저장된 데이터를 보호하기 위한 것입니다. 서버 측 암호화는 객체 메타데이터가 아닌 객체 데이터만 암호화합니다. 고객 제공 키(SSE-C)로 서버 측 암호화를 사용하여 자체 암호화 키로 암호화된 데이터를 저장할 수 있습니다. 요청의 일부로 제공한 암호화 키를 사용하여 Amazon S3는 디스크에 쓸 때 데이터 암호화를 관리하고 객체에 액세스할 때 데이터 암호 해독을 관리합니다. 따라서 데이터 암호화 및 암호 해독을 수행하기 위해 어떠한 코드도 사용할 필요가 없으며, 사용자는 자신이 제공한 암호화 키를 관리하기만 하면 됩니다.

객체를 업로드하면 Amazon S3는 제공된 암호화 키를 사용하여 AES-256 암호화를 데이터에 적용합니다. 그런 다음 Amazon S3는 메모리에서 암호화 키를 제거합니다. 객체를 검색할 경우 요청에 포함된 것과 동일한 암호화 키를 제공해야 합니다. Amazon S3는 제공된 암호화 키가 일치하는지 확인한 후 객체 데이터를 반환하기 전에 객체의 암호화를 해독합니다.

SSE-C 사용에 따르는 추가 비용은 없습니다. 그러나 SSE-C 구성 및 사용 요청에는 표준 Amazon S3 요청 요금이 발생합니다. 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

> **Note**
>
> Amazon S3는 제공된 암호화 키를 저장하지 않습니다. 대신에 임의로 암호화 키의 솔트 해시 기반 메시지 인증 코드(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.

S3 복제는 SSE-C로 암호화된 객체를 지원합니다. 암호화된 객체 복제에 대한 자세한 내용은 the section called “암호화된 객체 복제” 섹션을 참조하십시오.

SSE-C에 대한 자세한 내용은 다음 주제를 참조하십시오.

주제

- SSE-C 개요
- SSE-C 요구 및 제한
- 미리 서명된 URL 및 SSE-C
- 고객 제공 키를 사용한 서버 측 암호화 지정(SSE-C).

SSE-C 개요

이 섹션에서는 SSE-C의 개요를 다룹니다. SSE-C를 사용할 때는 다음 사항을 명심하십시오.

- HTTPS를 사용해야 합니다.

> **Important**
>
> Amazon S3는 SSE-C를 사용할 때 HTTP를 통해 전송된 모든 요청을 거부합니다. 보안상의 이유로 인해, 보안에 취약할 수 있는 HTTP를 통해 키를 보내면 오류가 발생할 수 있음을 유의하십시오. 키를 취소하고 적절하게 교체합니다.

- 응답의 엔터티 태그(ETag)는 객체 데이터의 MD5 해시가 아닙니다.
- 암호화 키가 사용되는 매핑을 관리하여 객체를 암호화합니다. Amazon S3는 암호화 키를 저장하지 않습니다. 객체에 대해 제공한 암호화 키는 직접 추적해야 합니다.
  - 버전 관리가 활성화된 버킷의 경우, 이 기능을 사용하여 업로드 하는 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.

SSE-C 요구 및 제한

특정 Amazon S3 버킷의 모든 객체에 대해 SSE-C를 요구하려면 정책을 사용하면 됩니다.

예를 들어 다음 버킷 정책은 SSE-C를 요청하는 `x-amz-server-side-encryption-customer-algorithm` 헤더가 포함되지 않은 모든 요청에 대해 객체 업로드(`s3:PutObject`) 권한을 거부합니다.

```
{
    "Version": "2012-10-17",
    "Id": "PutObjectPolicy",
    "Statement": [
        {
            "Sid": "RequireSSECObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption-customer-algorithm": "true"
                }
            }
        }
    ]
}
```

특정 Amazon S3 버킷의 모든 객체에 대한 서버 측 암호화를 제한하려는 경우에도 정책을 사용할 수 있습니다. 예를 들어, 다음 버킷 정책은 요청에 SSE-C를 요청하는 `x-amz-server-side-encryption-customer-algorithm` 헤더가 포함되는 경우 모든 사용자에게 객체 업로드(`s3:PutObject`) 권한을 거부합니다.

```
{
    "Version": "2012-10-17",
    "Id": "PutObjectPolicy",
    "Statement": [
        {
            "Sid": "RestrictSSECObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption-customer-algorithm": "false"
                }
            }
        }
    ]
}
```

Important

버킷 정책을 사용하여 `s3:PutObject`에서 SSE-C를 요구하는 경우, 모든 멀티파트 업로드 요청(CreateMultipartUpload, UploadPart, CompleteMultipartUpload)에 `x-amz-server-side-encryption-customer-algorithm` 헤더를 포함시켜야 합니다.

미리 서명된 URL 및 SSE-C

새로운 객체 업로드, 기존 객체 또는 객체 메타데이터 검색과 같은 작업에 사용되는 미리 서명된 URL을 생성할 수 있습니다. 미리 서명된 URL은 다음과 같이 SSE-C를 지원합니다.

- 미리 서명된 URL을 만들 때 서명 계산에 `x-amz-server-side-encryption-customer-algorithm` 헤더를 사용하여 알고리즘을 지정해야 합니다.
- 새로운 객체 업로드, 기존 객체 또는 객체 메타데이터 전용 검색에 미리 서명된 URL을 사용하는 경우, 클라이언트 애플리케이션의 요청에서 모든 암호화 헤더를 제공해야 합니다.

> **Note**
>
> 비 SSE-C 객체의 경우, 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 포함해야 하기 때문입니다. 따라서 프로그래밍 방식으로만 SSE-C 객체에 미리 서명된 URL을 생성할 수 있습니다.

미리 서명된 URL에 대한 자세한 내용은 the section called “미리 서명된 URL을 사용하여 객체 다운로드 및 업로드” 단원을 참조하십시오.

고객 제공 키를 사용한 서버 측 암호화 지정(SSE-C).

REST API를 사용하여 객체를 생성할 때 고객 제공 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 S3는 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 업로드하고 SSE-C를 요청할 수 없습니다. 또한, SSE-C를 사용하여 저장된 기존 객체를 업데이트할 수도 없습니다(예: 스토리지 클래스 변경 또는 메타데이터 추가).

REST API 사용

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(GetObject 참조)를 사용하여 객체를 검색할 때 요청 헤더를 지정할 수 있습니다.
- HEAD 작업 - HEAD API(HeadObject 참조)를 사용하여 객체 메타데이터를 검색하려면 이러한 요청 헤더를 지정하면 됩니다.
- PUT 작업 - PUT Object API(PutObject 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 시작에 관한 문서 참조) 각각의 후속 부분은 요청을 업로드합니다(UploadPart 또는 CopyObject 참조). 각 파트 업로드 요청에서 암호화 정보는 시작 멀티파트 업로드 요청의 시작에서 제공한 내용과 동일해야 합니다.
- POST 작업 - POST 작업(객체 POST에 관한 문서 참조)을 사용하여 객체를 업로드할 경우에는 요청 헤더 대신 양식 필드에 동일한 정보를 제공합니다.
- Copy 작업 - 객체를 복사(CopyObject 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 S3가 객체의 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.

AWS SDK를 사용하여 PUT, GET, Head 및 Copy 작업에 SSE-C 지정

다음 예제에서는 객체에 대해 고객 제공 키를 사용한 서버 측 암호화(SSE-C)를 요청하는 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 - 앞 단계에서 업로드한 객체를 다운로드합니다. 요청 시 객체가 업로드되었을 때 제공한 것과 동일한 암호화 정보를 제공합니다. Amazon S3는 객체의 암호를 해독하여 사용자에게 반환하기 위해 이 정보가 필요합니다.
- 객체 메타데이터 가져오기 - 객체의 메타데이터를 검색합니다. 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 S3는 사용자가 명시적으로 요청하는 경우에만 객체의 사본을 암호화합니다. 이 예제에서는 Amazon S3에 객체의 암호화된 사본을 저장하도록 지시합니다.

Java

> **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 업로드를 사용한 객체 업로드 섹션을 참조하십시오.

필요한 암호화 정보를 추가하려면 요청에 `SSECustomerKey`를 포함합니다. `SSECustomerKey` 클래스에 대한 자세한 내용은 REST API 섹션을 참조하십시오.

SSE-C에 대한 자세한 내용은 고객 제공 키(SSE-C)로 서버 측 암호화 사용 단원을 참조하십시오. 실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 AWS SDK for Java 개발자 안내서의 시작하기 섹션을 참조하세요.

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();
    }
}
```

.NET

> **Note**
>
> 멀티파트 업로드 API를 사용하여 대형 객체를 업로드하는 예제는 멀티파트 업로드를 사용한 객체 업로드 및 AWS SDK 사용(하위 수준 API) 단원을 참조하십시오.

SSE-C에 대한 자세한 내용은 고객 제공 키(SSE-C)로 서버 측 암호화 사용 단원을 참조하십시오. 코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

Example

```
using Amazon;
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);
        }
    }
}
```

AWS SDK를 사용하여 멀티파트 업로드를 위한 SSE-C 지정

앞 섹션의 예제에서는 PUT, GET, Head 및 Copy 작업에서 고객 제공 키(SSE-C)를 사용하는 서버 측 암호화를 요청하는 방법을 알아보았으며, 이 섹션에서는 SSE-C를 지원하는 기타 Amazon S3 API에 대해 소개합니다.

Java

멀티파트 업로드 API를 사용하여 대형 객체를 업로드할 수 있습니다(Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 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 업로드를 사용한 객체 업로드 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. 예시는 AWS SDK 사용(하위 수준 API) 섹션을 참조하세요.

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` 인스턴스에서 요청할 암호화 키 정보를 제공합니다. Amazon S3에서는 고객이 제공하는 키를 사용하여 객체를 암호화합니다.
- `TransferManager.copy()` 메서드를 호출하여 객체의 복사본을 생성합니다. 이 예제에서는 Amazon S3에 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();
        }
    }
}
```

.NET

멀티파트 업로드 API를 사용하여 대형 객체를 업로드할 수 있습니다(Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 참조). AWS SDK for .NET에서는 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 업로드를 사용한 객체 업로드 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. 예를 보려면 AWS SDK 사용(하위 수준 API) 섹션을 참조하십시오.

  다음은 기존 대형 객체의 복사본을 만드는 하위 수준의 멀티파트 업로드 예제입니다. 이 예제에서 객체는 SSE-C를 사용하여 Amazon S3에 저장되고, 대상 객체 또한 SSE-C를 사용하여 저장됩니다. 예제에서는 다음을 수행합니다.

- 암호화 키와 관련 정보를 제공하여 멀티파트 업로드 요청을 시작합니다.
- CopyPartRequest에 원본 및 대상 객체 암호화 키와 관련 정보를 제공합니다.
- 객체 메타데이터를 검색하여 복사할 원본 객체의 크기를 확인합니다.
- 5MB 단위로 객체를 업로드합니다.

Example

```
using Amazon;
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);
            }
        }
    }
}
```

클라이언트측 암호화를 사용하여 데이터 보호

클라이언트측 암호화는 전송 및 저장 시 보안을 보장하기 위해 로컬에서 데이터를 암호화하는 작업입니다. Amazon S3에 전송하기 전에 객체를 암호화하려면 Amazon S3 암호화 클라이언트를 사용하십시오. 이 방식으로 객체를 암호화하면 AWS를 포함한 제3자에게 객체가 노출되지 않습니다. Amazon S3는 이미 암호화된 객체를 수신하며, Amazon S3는 객체를 암호화하거나 해독하는 역할을 하지 않습니다. Amazon S3 암호화 클라이언트와 서버 측 암호화를 모두 사용하여 데이터를 암호화할 수 있습니다. 암호화된 객체를 Amazon S3에 전송할 때 Amazon S3는 해당 객체를 암호화된 것으로 인식하지 않고 일반적인 객체만 탐지합니다.

Amazon S3 암호화 클라이언트는 사용자와 Amazon S3 사이의 중개자 역할을 합니다. Amazon S3 암호화 클라이언트를 인스턴스화하고 나면 Amazon S3 `PutObject` 및 `GetObject` 요청의 일부로 객체가 자동으로 암호화되고 해독됩니다. 객체는 모두 고유한 데이터 키로 암호화됩니다. Amazon S3 암호

화 클라이언트는 KMS 키를 래핑 키로 지정하더라도 버킷 키를 사용하거나 버킷 키와 상호 작용하지 않습니다.

Amazon S3 암호화 클라이언트 개발자 안내서는 Amazon S3 암호화 클라이언트 버전 3.0 이상을 중점적으로 다루고 있습니다. 자세한 내용은 Amazon S3 암호화 클라이언트 개발자 안내서에서 Amazon S3 암호화 클라이언트란 무엇인가요?를 참조하십시오.

Amazon S3 Encryption 클라이언트의 이전 버전에 대한 자세한 내용은 해당 프로그래밍 언어의 AWS SDK 개발자 안내서를 참조하십시오.

- AWS SDK for Java
- AWS SDK for .NET
- AWS SDK for Go
- AWS SDK for PHP
- AWS SDK for Ruby
- AWS SDK for C++

인터네트워크 트래픽 개인 정보

이 주제에서는 Amazon S3가 해당 서비스에서 다른 위치로 향하는 연결을 보호하는 방법을 설명합니다.

서비스와 온프레미스 클라이언트 및 애플리케이션 간의 트래픽

다음 연결을 AWS PrivateLink와 결합하여 프라이빗 네트워크와 AWS 간의 연결을 제공할 수 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 전송 계층 보안(TLS) 1.2를 지원해야 합니다. TLS 1.3을 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 관련된 비밀 액세스 키를 사용하여 요청에 서명하거나 AWS

Security Token Service(STS)를 사용하여 요청에 서명할 수 있는 임시 보안 자격 증명을 생성할 수 있습니다.

같은 리전에 있는 AWS 리소스 사이의 트래픽

Amazon S3용 가상 프라이빗 클라우드(VPC) 엔드포인트는 VPC 내의 논리적 엔터티로서, Amazon S3에만 연결을 허용합니다. VPC는 Amazon S3로 요청을 라우팅하고, 응답을 다시 VPC로 라우팅합니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC 종단점을 참조하십시오. VPC 종단점에서 S3 버킷 액세스를 제어하는 데 사용할 수 있는 예제 버킷 정책은 버킷 정책을 사용하여 VPC 엔드포인트에서 액세스 제어 단원을 참조하십시오.

Amazon S3용 AWS PrivateLink

Amazon S3용 AWS PrivateLink를 사용하면 가상 프라이빗 클라우드(VPC)에서 인터페이스 VPC 엔드포인트(인터페이스 엔드포인트)를 프로비저닝할 수 있습니다. 이러한 엔드포인트는 VPN 및 AWS Direct Connect를 통해 온프레미스에 있거나 VPC 피어링을 통해 다른 AWS 리전에 있는 애플리케이션에서 직접 액세스할 수 있습니다.

인터페이스 엔드포인트는 VPC의 서브넷에서 프라이빗 IP 주소가 할당된 하나 이상의 탄력적 네트워크 인터페이스(ENI)로 표시됩니다. 인터페이스 엔드포인트를 통한 Amazon S3에 대한 요청은 Amazon 네트워크에 유지됩니다. 또한, AWS Direct Connect 또는 AWS Virtual Private Network(AWS VPN)을 통해 온프레미스 애플리케이션에서 VPC의 인터페이스 엔드포인트에 액세스할 수 있습니다. VPC를 온프레미스 네트워크에 연결하는 방법에 대한 자세한 내용은 AWS Direct Connect 사용 설명서 및 AWS Site-to-Site VPN 사용 설명서를 참조하십시오.

인터페이스 엔드포인트에 대한 일반적인 정보는 AWS PrivateLink 가이드의 인터페이스 VPC 엔드포인트(AWS PrivateLink)를 참조하십시오.

주제

- Amazon S3용 VPC 엔드포인트 유형
- Amazon S3용 AWS PrivateLink에 대한 규제 및 제한
- VPC 엔드포인트 생성
- Amazon S3 인터페이스 엔드포인트 액세스
- 프라이빗 DNS
- S3 인터페이스 엔드포인트에서 버킷, 액세스 포인트, Amazon S3 제어 API 작업에 액세스
- 온프레미스 DNS 구성 업데이트
- Amazon S3에 대한 VPC 엔드포인트 정책 생성

Amazon S3용 VPC 엔드포인트 유형

두 가지 유형의 VPC 엔드포인트, 즉 게이트웨이 엔드포인트와 인터페이스 엔드포인트(AWS PrivateLink 사용)를 사용하여 Amazon S3에 액세스할 수 있습니다. 게이트웨이 엔드포인트는 AWS 네트워크를 통해 VPC에서 Amazon S3에 액세스하기 위해 라우팅 테이블에 지정하는 게이트웨이입니다. 인터페이스 엔드포인트는 프라이빗 IP 주소를 통해 온프레미스의 VPC 내에서 또는 VPC 피어링이나 AWS Transit Gateway를 사용하여 다른 AWS 리전의 VPC에서 Amazon S3로 요청을 라우팅함으로써 게이트웨이 엔드포인트의 기능을 확장합니다. 자세한 내용은 VPC 피어링이란? 및 Transit Gateway 및 VPC 피어링을 참조하십시오.

인터페이스 엔드포인트는 게이트웨이 엔드포인트와 호환됩니다. VPC에 기존 게이트웨이 엔드포인트가 있는 경우, 동일한 VPC에서 두 가지 유형의 엔드포인트를 모두 사용할 수 있습니다.

Amazon S3용 게이트웨이 엔드포인트	Amazon S3용 인터페이스 엔드포인트
두 경우 모두, 네트워크 트래픽은 AWS 네트워크에 남아 있습니다.	
Amazon S3 퍼블릭 IP 주소 사용	VPC의 프라이빗 IP 주소를 사용하여 Amazon S3에 액세스
동일한 Amazon S3 DNS 이름 사용	엔드포인트별 Amazon S3 DNS 이름 필요
온프레미스에서의 액세스를 허용하지 않음	온프레미스에서의 액세스 허용
다른 AWS 리전에서의 액세스를 허용하지 않음	VPC 피어링 또는 AWS Transit Gateway를 사용하여 다른 AWS 리전의 VPC에서 액세스 허용
미청구	청구

게이트웨이 엔드포인트에 대한 자세한 내용은 AWS PrivateLink 가이드에서 게이트웨이 VPC 엔드포인트를 참조하십시오.

Amazon S3용 AWS PrivateLink에 대한 규제 및 제한

Amazon S3용 AWS PrivateLink에는 VPC 제한이 적용됩니다. 자세한 내용은 AWS PrivateLink 가이드의 인터페이스 엔드포인트 고려 사항 및 AWS PrivateLink 할당량을 참조하십시오. 또한 다음과 같은 제한 사항이 적용됩니다.

Amazon S3용 AWS PrivateLink는 다음을 지원하지 않습니다.

- Federal Information Processing Standard(FIPS) 엔드포인트
- 웹 사이트 엔드포인트
- 레거시 글로벌 엔드포인트
- S3 대시 리전 엔드포인트
- 듀얼 스택 엔드포인트
- 다른 AWS 리전에 있는 버킷 간 CopyObject 또는 UploadPartCopy 사용
- 전송 계층 보안(TLS) 1.0
- 전송 계층 보안(TLS) 1.1
- Transport Layer Security(TLS) 1.3

VPC 엔드포인트 생성

VPC 인터페이스 엔드포인트를 생성하려면 AWS PrivateLink 설명서의 VPC 엔드포인트 생성을 참조하십시오.

Amazon S3 인터페이스 엔드포인트 액세스

인터페이스 엔드포인트를 생성하면 Amazon S3는 두 가지 유형의 엔드포인트별 S3 DNS 이름인 Regional 및 zonal을 생성합니다.

- 리전별 DNS 이름에는 고유한 VPC 엔드포인트 ID, 서비스 식별자, AWS 리전, `vpce.amazonaws.com`이 해당 이름에 포함됩니다. 예를 들어, VPC 엔드포인트 ID *`vpce-1a2b3c4d`*의 경우, 생성된 DNS 이름은 *`vpce-1a2b3c4d-5e6f`*`.s3.us-east-1.vpce.amazonaws.com`과(와) 비슷할 수 있습니다.
- 영역별 DNS 이름에는 가용 영역이 포함됩니다(예: *`vpce-1a2b3c4d-5e6f`*`-us-east-1a.s3.us-east-1.vpce.amazonaws.com`). 아키텍처가 가용 영역을 분리하는 경우 이 옵션을 사용할 수 있습니다. 예를 들어, 오류를 제한하거나 리전별 데이터 전송 비용을 줄이는 데 사용할 수 있습니다.

엔드포인트별 S3 DNS 이름은 S3 퍼블릭 DNS 도메인에서 확인할 수 있습니다.

프라이빗 DNS

VPC 인터페이스 엔드포인트의 프라이빗 DNS 옵션은 VPC 엔드포인트를 통한 S3 트래픽 라우팅을 단순화하고 애플리케이션에서 사용할 수 있는 가장 저렴한 네트워크 경로를 활용할 수 있도록 도와줍니다. 프라이빗 DNS 옵션을 사용하면 인터페이스 엔드포인트의 엔드포인트별 DNS 이름을 사용하도록 S3 클라이언트를 업데이트하거나 DNS 인프라를 관리하지 않고도 리전별 S3 트래픽을 라우팅할

수 있습니다. 프라이빗 DNS 이름을 활성화하면 리전별 S3 DNS 쿼리는 다음 엔드포인트에 대한 AWS PrivateLink 프라이빗 IP 주소로 해결됩니다.

- 리전 버킷 엔드포인트(예: `s3.us-east-1.amazonaws.com`)
- 제어 엔드포인트(예: `s3-control.us-east-1.amazonaws.com`)
- 액세스 포인트 엔드포인트(예: `s3-accesspoint.us-east-1.amazonaws.com`)

AWS PrivateLink for Amazon S3는 Amazon S3 듀얼 스택 엔드포인트 사용을 지원하지 않습니다. S3 이중 스택 DNS 이름을 프라이빗 DNS 이름으로 사용하는 경우 IPv6 트래픽이 삭제되거나, 가상 프라이빗 클라우드(VPC)에 인터넷 게이트웨이가 있는 경우 IPv6 트래픽이 VPC의 인터넷 게이트웨이를 통해 라우팅됩니다.

VPC에 게이트웨이 엔드포인트가 있는 경우 VPC 내 요청은 기존 S3 게이트웨이 엔드포인트를 통해, 온프레미스 요청은 인터페이스 엔드포인트를 통해 자동으로 라우팅할 수 있습니다. 이 접근 방식을 사용하면 VPC 내 트래픽에 대해 요금이 청구되지 않는 게이트웨이 엔드포인트를 사용하여 네트워킹 비용을 최적화할 수 있습니다. 온프레미스 애플리케이션은 인바운드 해석기 엔드포인트의 도움을 받아 AWS PrivateLink를 사용할 수 있습니다. Amazon은 Route 53 Resolver라고 하는 VPC용 DNS 서버를 제공합니다. 인바운드 Resolver 엔드포인트는 온프레미스 네트워크 상의 DNS 쿼리를 Route 53 Resolver로 전달합니다.

> **Important**
>
> 인바운드 엔드포인트에만 프라이빗 DNS 활성화를 사용할 때 가장 저렴한 네트워크 경로를 활용하려면 VPC 게이트웨이 엔드포인트가 있어야 합니다. 게이트웨이 엔드포인트가 있으면 인바운드 엔드포인트에만 프라이빗 DNS 활성화 옵션을 선택한 경우 VPC 내 트래픽이 항상 AWS 프라이빗 네트워크를 통해 라우팅되도록 할 수 있습니다. 인바운드 엔드포인트에만 프라이빗 DNS 활성화 옵션을 선택한 상태에서 이 게이트웨이 엔드포인트를 유지 관리해야 합니다. 게이트웨이 엔드포인트를 삭제하려면 먼저 인바운드 엔드포인트에만 프라이빗 DNS 활성화를 선택 해제해야 합니다.
>
> 기존 인터페이스 엔드포인트를 인바운드 엔드포인트에만 프라이빗 DNS 활성화로 업데이트하려면 먼저 VPC에 S3 게이트웨이 엔드포인트가 있는지 확인하십시오. 게이트웨이 엔드포인트와 프라이빗 DNS 이름 관리에 대한 자세한 내용은 AWS PrivateLink 가이드의 게이트웨이 VPC 엔드포인트 및 DNS 이름 관리를 각각 참조하십시오.

인바운드 엔드포인트에만 프라이빗 DNS 활성화 옵션은 게이트웨이 엔드포인트를 지원하는 서비스에서만 사용할 수 있습니다.

인바운드 엔드포인트에만 프라이빗 DNS 활성화를 사용하는 VPC 엔드포인트를 만드는 방법에 대한 자세한 내용은 AWS PrivateLink 가이드의 인터페이스 엔드포인트 생성을 참조하십시오.

VPC 콘솔 사용

콘솔에는 DNS 이름 활성화와 인바운드 엔드포인트에만 프라이빗 DNS 활성화라는 두 가지 옵션이 있습니다. DNS 이름 활성화는 AWS PrivateLink에서 지원하는 옵션입니다. DNS 이름 활성화 옵션을 사용하면 기본 퍼블릭 엔드포인트 DNS 이름에 요청을 수행하면서 Amazon S3에 대한 Amazon의 프라이빗 연결을 사용할 수 있습니다. 이 옵션을 활성화하면 고객은 애플리케이션에서 사용할 수 있는 가장 저렴한 네트워크 경로를 활용할 수 있습니다.

Amazon S3에 대한 기존 또는 신규 VPC 인터페이스 엔드포인트에서 프라이빗 DNS 이름을 활성화하면 인바운드 엔드포인트에만 프라이빗 DNS 활성화 옵션이 기본적으로 선택됩니다. 이 옵션을 선택하면 애플리케이션은 온프레미스 트래픽에 인터페이스 엔드포인트만 사용합니다. 이 VPC 내 트래픽은 비용이 상대적으로 저렴한 게이트웨이 엔드포인트를 자동으로 사용합니다. 또는 인바운드 엔드포인트에만 프라이빗 DNS 활성화를 선택 해제하여 인터페이스 엔드포인트를 통해 모든 S3 요청을 라우팅할 수 있습니다.

AWS CLI 사용하기

`PrivateDnsOnlyForInboundResolverEndpoint`의 값을 지정하지 않으면 `true`이 기본값으로 사용됩니다. 그러나 VPC는 설정을 적용하기 전에 VPC에 게이트웨이 엔드포인트가 있는지 확인합니다. VPC 게이트웨이 엔드포인트가 있는 경우 호출은 성공합니다. 없다면 다음과 같은 오류 메시지가 나타납니다.

PrivateDnsOnlyForInboundResolverEndpoint를 참으로 설정하려면 VPC *vpce_id*에 해당 서비스의 게이트웨이 엔드포인트가 있어야 합니다.

신규 VPC 인터페이스 엔드포인트의 경우

`private-dns-enabled` 및 `dns-options` 특성을 사용하여 명령줄을 통해 프라이빗 DNS를 활성화합니다. `dns-options` 특성의 `PrivateDnsOnlyForInboundResolverEndpoint` 옵션은 `true`으로 설정해야 합니다. *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws ec2 create-vpc-endpoint \
--region us-east-1 \
--service-name s3-service-name \
--vpc-id client-vpc-id \
--subnet-ids client-subnet-id \
```

```
--vpc-endpoint-type Interface  \
--private-dns-enabled  \
--ip-address-type ip-address-type \
--dns-options PrivateDnsOnlyForInboundResolverEndpoint=true \
--security-group-ids client-sg-id
```

기존 VPC 엔드포인트의 경우

기존 VPC 엔드포인트에 프라이빗 DNS를 사용하려면 다음 예시 명령을 사용하되 *user input placeholders*를 실제 정보로 대체하십시오.

```
aws ec2 modify-vpc-endpoint \
--region us-east-1 \
--vpc-endpoint-id client-vpc-id \
--private-dns-enabled \
--dns-options PrivateDnsOnlyForInboundResolverEndpoint=false
```

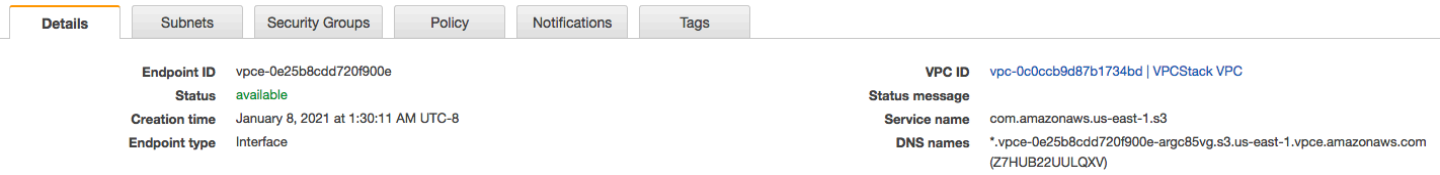
인바운드 해석기에서만 프라이빗 DNS를 사용하도록 기존 VPC 엔드포인트를 업데이트하려면 다음 예시를 사용하되 샘플 값을 실제 값으로 바꾸십시오.

```
aws ec2 modify-vpc-endpoint \
--region us-east-1 \
--vpc-endpoint-id client-vpc-id \
--private-dns-enabled \
--dns-options PrivateDnsOnlyForInboundResolverEndpoint=true
```

S3 인터페이스 엔드포인트에서 버킷, 액세스 포인트, Amazon S3 제어 API 작업에 액세스

AWS CLI 또는 AWS SDK를 사용하여 S3 인터페이스 엔드포인트를 통해 버킷, S3 액세스 포인트, Amazon S3 제어 API 작업에 액세스할 수 있습니다.

다음 이미지는 VPC 엔드포인트의 DNS 이름을 찾을 수 있는 VPC 콘솔 세부 정보 탭을 보여줍니다. 이 예에서 VPC 엔드포인트 ID(vpce-id)는 `vpce-0e25b8cdd720f900e`이고 DNS 이름은 `*.vpce-0e25b8cdd720f900e-argc85vg.s3.us-east-1.vpce.amazonaws.com`입니다.



DNS 이름을 사용하여 리소스에 액세스할 때는 *를 적절한 값으로 바꾸십시오. * 대신 사용할 적절한 값은 다음과 같습니다.

- `bucket`
- `accesspoint`
- `control`

예를 들어, 버킷에 액세스하려면 다음과 같은 DNS 이름을 사용합니다.

`bucket.vpce-0e25b8cdd720f900e-argc85vg.s3.us-east-1.vpce.amazonaws.com`

DNS 이름을 사용하여 버킷, 액세스 포인트 및 Amazon S3 제어 API 작업에 액세스하는 방법의 예는 AWS CLI 예제 및 AWS SDK 예제의 다음 섹션을 참조하십시오.

엔드포인트별 DNS 이름을 보는 방법에 대한 자세한 내용은 VPC 사용 설명서에서 엔드포인트 서비스 프라이빗 DNS 이름 구성 보기를 참조하십시오.

AWS CLI 예제

AWS CLI 명령에서 S3 인터페이스 엔드포인트를 통해 S3 버킷, S3 액세스 포인트 또는 S3 제어 API 작업에 액세스하려면 `--region` 및 `--endpoint-url` 파라미터를 사용합니다.

예: 엔드포인트 URL을 사용하여 버킷의 객체 나열

다음 예시에서 버킷 이름 *my-bucket*, 리전 *us-east-1*, VPC 엔드포인트 ID의 DNS 이름 *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com*을 실제 정보로 바꿉니다.

```
aws s3 ls s3://my-bucket/ --region us-east-1 --endpoint-url
 https://bucket.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com
```

예: 엔드포인트 URL을 사용하여 액세스 포인트의 객체 나열

- 방법 1 - 액세스 포인트의 Amazon 리소스 이름(ARN)을 액세스 포인트 엔드포인트에 사용

ARN *us-east-1:123456789012:accesspoint/accesspointexamplename*, 리전 *us-east-1* 및 VPC 엔드포인트 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com*을 실제 정보로 바꿉니다.

```
aws s3api list-objects-v2 --bucket arn:aws:s3:us-east-1:123456789012:accesspoint/
accesspointexamplename --region us-east-1 --endpoint-url
 https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com
```

명령을 성공적으로 실행할 수 없는 경우 AWS CLI를 최신 버전으로 업데이트하고 다시 시도하십시오. 업데이트 지침에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS CLI의 최신 버전 설치 또는 업데이트를 참조하십시오.

- 방법 2 - 액세스 포인트의 별칭을 리전 버킷 엔드포인트와 함께 사용

  다음 예시에서 액세스 포인트 별칭 *accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias*, 리전 *us-east-1*, VPC 엔드포인트 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com*을 실제 정보로 바꿉니다.

  ```
  aws s3api list-objects-v2 --
  bucket accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias
   --region us-east-1 --endpoint-url https://bucket.vpce-1a2b3c4d-5e6f.s3.us-
  east-1.vpce.amazonaws.com
  ```

- 방법 2 - 액세스 포인트의 별칭을 액세스 포인트 엔드포인트와 함께 사용

  먼저 호스트 이름에 버킷이 포함된 S3 엔드포인트를 구성하려면 `aws s3api`가 사용할 주소 지정 스타일을 `virtual`로 설정합니다. `AWS configure`에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 구성 및 자격 증명 파일 설정을 참조하십시오.

  ```
  aws configure set default.s3.addressing_style virtual
  ```

  그런 다음, 아래 예시에서 액세스 포인트 별칭 *accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias*, 리전 *us-east-1*, VPC 엔드포인트 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com*을 실제 정보로 바꿉니다. 액세스 포인트 별칭에 대한 자세한 내용은 액세스 포인트 별칭 섹션을 참조하세요.

```
aws s3api list-objects-v2 --
bucket accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias --
region us-east-1 --endpoint-url https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com
```

예: 엔드포인트 URL을 사용하여 S3 제어 API 작업이 있는 작업 나열

다음 예시에서 리전 *us-east-1*, VPC 엔드포인트 *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com*, 계정 ID *12345678*을 실제 정보로 바꿉니다.

```
aws s3control --region us-east-1 --endpoint-url
 https://control.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com list-jobs --
account-id 12345678
```

AWS SDK 예제

AWS SDK를 사용하여 S3 인터페이스 엔드포인트를 통해 S3 버킷, S3 액세스 포인트, Amazon S3 제어 API 작업에 액세스하려면 SDK를 최신 버전으로 업데이트하십시오. 그런 다음, S3 인터페이스 엔드포인트를 통해 버킷, 액세스 포인트 또는 Amazon S3 제어 API에 액세스하기 위해 엔드포인트 URL을 사용하도록 클라이언트를 구성합니다.

SDK for Python (Boto3)

예: 엔드포인트 URL을 사용하여 S3 버킷에 액세스

다음 예에서 리전 *us-east-1* 및 VPC 엔드포인트 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com*을 실제 정보로 바꿉니다.

```
s3_client = session.client(
service_name='s3',
region_name='us-east-1',
endpoint_url='https://bucket.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com'
)
```

예: 엔드포인트 URL을 사용하여 S3 액세스 포인트에 액세스

다음 예에서 리전 *us-east-1* 및 VPC 엔드포인트 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com*을 실제 정보로 바꿉니다.

```
ap_client = session.client(
service_name='s3',
region_name='us-east-1',
endpoint_url='https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com'
)
```

예: 엔드포인트 URL을 사용하여 Amazon S3 제어 API에 액세스

다음 예에서 리전 *us-east-1* 및 VPC 엔드포인트 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com*을 실제 정보로 바꿉니다.

```
control_client = session.client(
service_name='s3control',
region_name='us-east-1',
endpoint_url='https://control.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com'
)
```

SDK for Java 1.x

예: 엔드포인트 URL을 사용하여 S3 버킷에 액세스

다음 예에서 VPC 엔드포인트 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com*을 실제 정보로 바꿉니다.

```
// bucket client
final AmazonS3 s3 = AmazonS3ClientBuilder.standard().withEndpointConfiguration(
        new AwsClientBuilder.EndpointConfiguration(
                "https://bucket.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com",
                Regions.DEFAULT_REGION.getName()
        )
).build();
List<Bucket> buckets = s3.listBuckets();
```

예: 엔드포인트 URL을 사용하여 S3 액세스 포인트에 액세스

다음 예에서 VPC 엔드포인트 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 및 ARN *us-east-1:123456789012:accesspoint/prod*를 실제 정보로 바꿉니다.

```
// accesspoint client
```

```
final AmazonS3 s3accesspoint =
 AmazonS3ClientBuilder.standard().withEndpointConfiguration(
        new AwsClientBuilder.EndpointConfiguration(
                "https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com",
                Regions.DEFAULT_REGION.getName()
        )
).build();
ObjectListing objects = s3accesspoint.listObjects("arn:aws:s3:us-
east-1:123456789012:accesspoint/prod");
```

예: 엔드포인트 URL을 사용하여 Amazon S3 제어 API 작업에 액세스

다음 예에서 VPC 엔드포인트 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com*을 실제 정보로 바꿉니다.

```
// control client
final AWSS3Control s3control =
 AWSS3ControlClient.builder().withEndpointConfiguration(
        new AwsClientBuilder.EndpointConfiguration(
                "https://control.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com",
                Regions.DEFAULT_REGION.getName()
        )
).build();
final ListJobsResult jobs = s3control.listJobs(new ListJobsRequest());
```

SDK for Java 2.x

예: 엔드포인트 URL을 사용하여 S3 버킷에 액세스

다음 예에서 VPC 엔드포인트 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 및 리전 *Region.US_EAST_1*을 실제 정보로 바꿉니다.

```
// bucket client
Region region = Region.US_EAST_1;
s3Client = S3Client.builder().region(region)

 .endpointOverride(URI.create("https://bucket.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com"))
                    .build()
```

예: 엔드포인트 URL을 사용하여 S3 액세스 포인트에 액세스

다음 예에서 VPC 엔드포인트 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 및 리전 *Region.US_EAST_1*을 실제 정보로 바꿉니다.

```
// accesspoint client
Region region = Region.US_EAST_1;
s3Client = S3Client.builder().region(region)

 .endpointOverride(URI.create("https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com"))
                    .build()
```

예: 엔드포인트 URL을 사용하여 Amazon S3 제어 API에 액세스

다음 예에서 VPC 엔드포인트 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 및 리전 *Region.US_EAST_1*을 실제 정보로 바꿉니다.

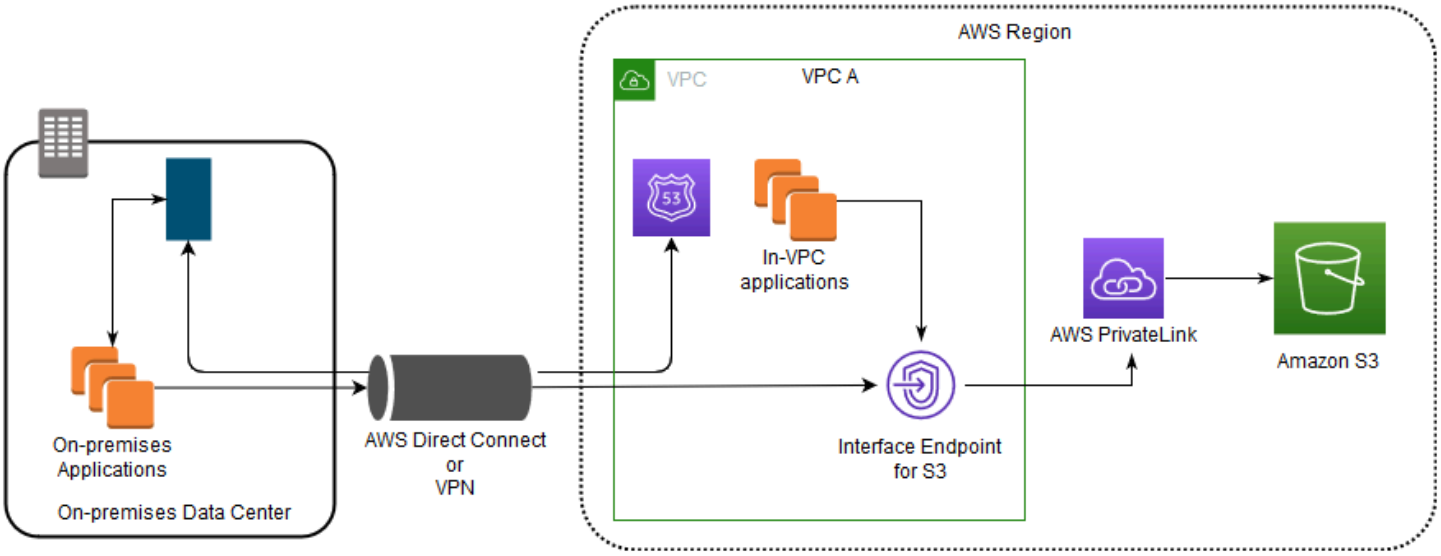
```
// control client
Region region = Region.US_EAST_1;
s3ControlClient = S3ControlClient.builder().region(region)

 .endpointOverride(URI.create("https://control.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com"))
                                  .build()
```

온프레미스 DNS 구성 업데이트

엔드포인트별 DNS 이름을 사용하여 Amazon S3의 인터페이스 엔드포인트에 액세스할 때는 온프레미스 DNS 확인자를 업데이트할 필요가 없습니다. 퍼블릭 Amazon S3 DNS 도메인의 인터페이스 엔드포인트의 프라이빗 IP 주소로 엔드포인트별 DNS 이름을 확인할 수 있습니다.

인터페이스 엔드포인트를 사용하여 VPC의 게이트웨이 엔드포인트 또는 인터넷 게이트웨이 없이 Amazon S3에 액세스

다음 다이어그램과 같이 VPC의 인터페이스 엔드포인트는 Amazon 네트워크를 통해 VPC 내 애플리케이션과 온프레미스 애플리케이션을 모두 Amazon S3로 라우팅할 수 있습니다.

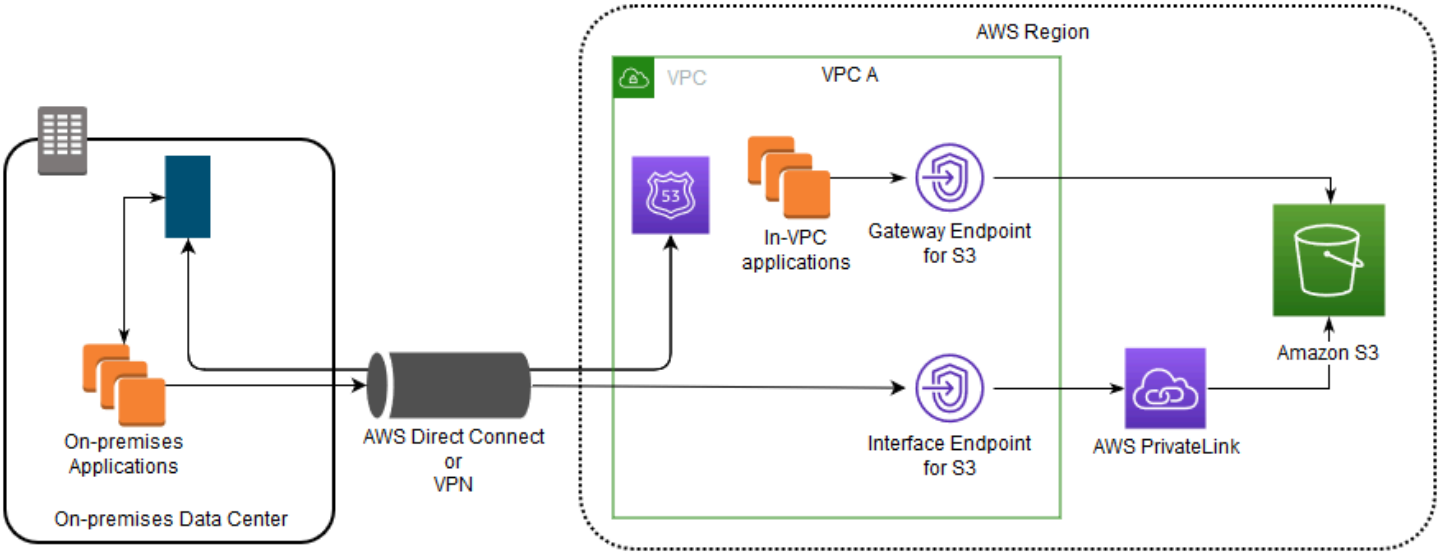


다이어그램은 다음을 보여 줍니다.

- 온프레미스 네트워크에서는 AWS Direct Connect 또는 AWS VPN을 사용하여 VPC A에 연결합니다.
- 온프레미스 및 VPC A의 애플리케이션은 엔드포인트별 DNS 이름을 사용하여 S3 인터페이스 엔드포인트를 통해 Amazon S3에 액세스합니다.
- 온프레미스 애플리케이션은 AWS Direct Connect(또는 AWS VPN)를 통해 VPC에서 인터페이스 엔드포인트로 데이터를 전송합니다. AWS PrivateLink는 AWS 네트워크를 통해 인터페이스 엔드포인트에서 Amazon S3로 데이터를 이동합니다.
- VPC 애플리케이션도 인터페이스 엔드포인트로 데이터를 전송합니다. AWS PrivateLink는 AWS 네트워크를 통해 인터페이스 엔드포인트에서 Amazon S3로 데이터를 이동합니다.

동일한 VPC에서 게이트웨이 엔드포인트와 인터페이스 엔드포인트를 함께 사용하여 Amazon S3에 액세스

다음 다이어그램과 같이 인터페이스 엔드포인트를 생성하고 기존 게이트웨이 엔드포인트를 동일한 VPC에 유지할 수 있습니다. 이 접근법을 사용하면 VPC 내 애플리케이션이 게이트웨이 엔드포인트를 통해 Amazon S3에 계속 액세스할 수 있으며, 요금은 청구되지 않습니다. 그런 다음, 온프레미스 애플리케이션만 인터페이스 엔드포인트를 사용하여 Amazon S3에 액세스합니다. 이러한 방식으로 Amazon S3에 액세스하려면 Amazon S3에 대한 엔드포인트별 DNS 이름을 사용하도록 온프레미스 애플리케이션을 업데이트해야 합니다.



다이어그램은 다음을 보여 줍니다.

- 온프레미스 애플리케이션은 엔드포인트별 DNS 이름을 사용하여 AWS Direct Connect(또는 AWS VPN)를 통해 VPC에서 인터페이스 엔드포인트로 데이터를 전송합니다. AWS PrivateLink는 AWS 네트워크를 통해 인터페이스 엔드포인트에서 Amazon S3로 데이터를 이동합니다.
- VPC 내 애플리케이션은 기본 리전 Amazon S3 이름을 사용하여 AWS 네트워크를 통해 Amazon S3에 연결하는 게이트웨이 엔드포인트로 데이터를 전송합니다.

게이트웨이 엔드포인트에 대한 자세한 내용은 VPC 사용 설명서에서 게이트웨이 VPC 엔드포인트를 참조하십시오.

Amazon S3에 대한 VPC 엔드포인트 정책 생성

Amazon S3에 대한 액세스를 제어하는 VPC 엔드포인트에 엔드포인트 정책을 연결할 수 있습니다. 이 정책은 다음 정보를 지정합니다.

- 작업을 수행할 수 있는 AWS Identity and Access Management(IAM) 보안 주체.
- 수행할 수 있는 작업
- 작업을 수행할 수 있는 리소스

Amazon S3 버킷 정책을 사용하여 버킷 정책의 `aws:sourceVpce` 조건을 통해 특정 VPC 엔드포인트의 특정 버킷에 대한 액세스를 제한할 수도 있습니다. 다음 예에서는 버킷 또는 엔드포인트에 대한 액세스를 제한하는 정책을 보여줍니다.

주제

- 예제: VPC 엔드포인트에서 특정 버킷에 대한 액세스 제한
- 예제: VPC 엔드포인트에서 특정 계정의 버킷에 대한 액세스 제한
- 예제: S3 버킷 정책에서 특정 VPC 엔드포인트에 대한 액세스 제한

예제: VPC 엔드포인트에서 특정 버킷에 대한 액세스 제한

특정 Amazon S3 버킷에 대한 액세스만 제한하는 엔드포인트 정책을 생성할 수 있습니다. 이 정책 유형은 VPC에 버킷을 사용하는 다른 AWS 서비스가 있을 경우 유용합니다. 다음 버킷 정책은 *amzn-s3-demo-bucket1*에 대한 액세스만 제한합니다. 이 엔드포인트 정책을 사용하려면 *amzn-s3-demo-bucket1*을 실제 버킷의 이름으로 대체합니다.

```
{
  "Version": "2012-10-17",
  "Id": "Policy1415115909151",
  "Statement": [
    { "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket1",
                   "arn:aws:s3:::amzn-s3-demo-bucket1/*"]
    }
  ]
}
```

예제: VPC 엔드포인트에서 특정 계정의 버킷에 대한 액세스 제한

특정 AWS 계정의 S3 버킷에 대해서만 액세스를 제한하는 엔드포인트 정책을 생성할 수 있습니다. VPC 내의 클라이언트가 사용자가 소유하지 않은 버킷에 액세스하지 못하게 하려면 엔드포인트 정책에서 다음 명령문을 사용합니다. 다음 예시 명령문에서는 하나의 AWS 계정 ID인 *111122223333*이 소유한 리소스에 대한 액세스를 제한하는 정책을 생성합니다.

```
{
  "Statement": [
    {
      "Sid": "Access-to-bucket-in-specific-account-only",
      "Principal": "*",
```

```
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Deny",
      "Resource": "arn:aws:s3:::*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": "111122223333"
        }
      }
    }
  ]
}
```

> **Note**
>
> 액세스되는 리소스의 AWS 계정 ID를 지정하려면 IAM 정책에서 `aws:ResourceAccount` 또는 `s3:ResourceAccount` 키를 사용하면 됩니다. 그러나 일부 AWS 서비스는 AWS 관리형 버킷에 대한 액세스에 의존합니다. 따라서 IAM 정책에서 `aws:ResourceAccount` 또는 `s3:ResourceAccount` 키를 사용하면 이러한 리소스에 대한 액세스에도 영향을 미칠 수 있습니다.

예제: S3 버킷 정책에서 특정 VPC 엔드포인트에 대한 액세스 제한

예제: S3 버킷 정책에서 특정 VPC 엔드포인트에 대한 액세스 제한

다음 Amazon S3 버킷 정책은 VPC 엔드포인트 *vpce-1a2b3c4d*에서만 특정 버킷 *amzn-s3-demo-bucket2*에 대한 액세스를 허용합니다. 이 정책은 지정된 엔드포인트가 사용되지 않으면 버킷에 대한 모든 액세스를 거부합니다. `aws:sourceVpce` 조건은 엔드포인트를 지정하며, VPC 엔드포인트 리소스에 대한 Amazon 리소스 이름(ARN)을 필요로 하지 않고 엔드포인트 ID만 필요로 합니다. 이 버킷 정책을 사용하려면 *amzn-s3-demo-bucket2* 및 *vpce-1a2b3c4d*를 실제 버킷의 이름과 엔드포인트로 대체합니다.

> **Important**
>
> - 특정 VPC 엔드포인트만 액세스할 수 있도록 다음 Amazon S3 버킷 정책을 적용할 경우 의도치 않게 버킷에 대한 액세스를 차단할 수 있습니다. VPC 엔드포인트에서 시작하는 연결에 대한 버킷 액세스를 특별히 제한하기 위한 버킷 정책은 버킷에 대한 모든 연결을 차단할 수

있습니다. 이 문제를 해결하는 방법에 대한 자세한 내용은 버킷 정책의 VPC 또는 VPC 엔드포인트 ID가 올바르지 않습니다.를 참조하십시오. 버킷에 액세스할 수 있도록 정책을 수정하는 방법은 무엇입니까?(지원 지식 센터)를 참조하세요.

- 다음 예제 정책을 사용하기 전에 VPC 엔드포인트 ID를 사용 사례에 적합한 값으로 바꾸세요. 그렇지 않으면 버킷에 액세스할 수 없습니다.
- 콘솔 요청은 지정된 VPC 종단점에서 발생하지 않으므로 이 정책은 지정된 버킷에 대한 콘솔 액세스를 사용 중지합니다.

```
{
  "Version": "2012-10-17",
  "Id": "Policy1415115909152",
  "Statement": [
    { "Sid": "Access-to-specific-VPCE-only",
      "Principal": "*",
      "Action": "s3:*",
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket2",
                   "arn:aws:s3:::amzn-s3-demo-bucket2/*"],
      "Condition": {"StringNotEquals": {"aws:sourceVpce": "vpce-1a2b3c4d"}}
    }
  ]
}
```

자세한 정책 예는 VPC 사용 설명서에서 Amazon S3용 엔드포인트를 참조하십시오.

VPC 연결에 대한 자세한 내용은 AWS 백서, Amazon Virtual Private Cloud(VPC) 연결 옵션에서 네트워크와 VPC 연결 옵션을 참조하십시오.

Amazon S3에 대한 규정 준수 확인

서드 파티 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 서비스 페이지에서 특정 규정 준수 프로그램 범위 내 자주 업데이트되는 AWS 서비스의 목록을 제공합니다.

AWS Artifact를 사용하여 다운로드할 수 있는 서드 파티 감사 보고서가 있습니다. 자세한 내용은 AWS Artifact에서 보고서 다운로드를 참조하세요.

AWS 규정 준수 프로그램에 대한 자세한 내용은 AWS 규정 준수 프로그램을 참조하십시오.

Amazon S3 사용 시 귀하의 규정 준수 책임은 데이터의 민감도, 조직의 규정 준수 목표, 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 환경을 AWS에 배포하기 위한 단계를 제공합니다.
- HIPAA 보안 및 규정 준수를 위한 설계에서는 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.
- Object Lock으로 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.

- S3 Inventory를 사용한 데이터 카탈로그화 및 분석를 사용하여 비즈니스, 규정 준수 및 규제 요건에 대한 객체의 복제 및 암호화 상태를 감사하고 보고할 수 있습니다.

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 대기 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 리전 내 및 리전 간 객체 복제 기능을 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 외에도 Amazon S3는 데이터 복원성과 백업 요구 사항을 지원하는 다양한 기능을 제공합니다.

수명 주기 구성

수명 주기 구성은 Amazon S3가 객체 그룹에 적용되는 작업을 정의하는 일련의 규칙입니다. 수명 주기 구성 규칙을 사용하면 Amazon S3가 객체를 더 저렴한 스토리지 클래스로 전환하거나 보관하거나 삭제하도록 유도할 수 있습니다. 자세한 내용은 객체 수명 주기 관리 섹션을 참조하세요.

버전 관리

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. 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하세요.

S3 객체 잠금

S3 객체 잠금을 사용하면 write once, read many(WORM) 모델을 사용하여 객체를 저장할 수 있습니다. S3 객체 잠금을 사용하면 고정된 시간 동안 또는 무기한으로 객체를 삭제하거나 덮어쓰지 않도록 할 수 있습니다. S3 객체 잠금을 사용하여 WORM 스토리지가 필요한 규제 요구 사항을 충족하거나 객체 변경 및 삭제에 대한 보호 계층을 추가할 수 있습니다. 자세한 내용은 Object Lock으로 객체 잠금 섹션을 참조하세요.

스토리지 클래스

Amazon S3는 워크로드 요구 사항에 따라 선택할 수 있는 다양한 스토리지 클래스를 제공합니다. S3 Standard-IA 및 S3 One Zone-IA 스토리지 클래스는 한 달에 한 번 정도 액세스하며 밀리초 단위의 액세스가 필요한 데이터용으로 설계되었습니다. S3 Glacier Instant Retrieval 스토리지 클래스는 분기에 한 번 정도 액세스하며 밀리초 단위의 액세스가 필요한 수명이 긴 아카이브 데이터용으로 설계되었습니다. 백업과 같이 즉각적인 액세스가 필요하지 않은 아카이브 데이터의 경우 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스를 사용할 수 있습니다. 자세한 내용은 Amazon S3 스토리지 클래스 이해 및 관리 섹션을 참조하세요.

다음 보안 모범 사례에서도 복원성을 다룹니다.

- Enable versioning
- Consider Amazon S3 cross-region replication
- Identify and audit all your Amazon S3 buckets

Amazon S3 백업의 암호화

Amazon S3를 사용하여 백업을 저장하는 경우 백업의 암호화는 해당 버킷의 구성에 따라 달라집니다. Amazon S3를 사용하면 S3 버킷의 기본 암호화 동작을 설정할 수 있습니다. 버킷에 저장되는 모든 객체를 암호화하도록 버킷에 대한 기본 암호화를 설정할 수 있습니다. 기본 암호화는 AWS KMS(SSE-KMS)에 저장된 키를 지원합니다. 자세한 내용은 Amazon S3 버킷에 대한 기본 서버 측 암호화 동작 설정 섹션을 참조하세요.

버전 관리 및 객체 잠금에 대한 자세한 내용은 다음 주제를 참조하십시오. S3 버전 관리로 여러 버전의 객체 유지 Object Lock으로 객체 잠금

Amazon S3의 인프라 보안

관리형 서비스인 Amazon S3는 AWS Well-Architected Framework의 보안 원칙에 설명된 AWS 글로벌 네트워크 보안 절차로 보호됩니다.

네트워크를 통한 Amazon S3 액세스는 AWS에서 게시한 API를 통해 이루어집니다. 클라이언트가 전송 계층 보안(TLS) 1.2를 지원해야 합니다. TLS 1.3도 지원하는 것이 좋습니다. (이 권장 사항에 대한 자세한 내용은 AWS 보안 블로그에서 TLS 1.3을 사용한 AWS 클라우드 연결 가속화를 참조하십시오.)

> **Note**
>
> TLS 1.3은 AWS PrivateLink for Amazon S3 및 다중 리전 액세스 포인트를 제외한 모든 S3 엔드포인트에서 지원됩니다.

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 서명 V4 또는 AWS 서명 V2를 사용하여 요청에 서명해야 합니다.

네트워크 위치에서 이 API를 호출할 수 있습니다. 다만 Amazon S3는 원본 IP 주소 기반의 제한을 포함할 수 있는 리소스 기반 액세스 정책을 지원합니다. Amazon S3 버킷 정책을 사용하여 특정 가상 프라이빗 클라우드(VPC) 엔드포인트 또는 특정 VPC의 버킷에 대한 액세스를 제어할 수도 있습니다. 그러면 AWS 네트워크의 특정 VPC에서만 특정 Amazon S3 버킷에 대한 네트워크 액세스가 효과적으로 격리됩니다. 자세한 내용은 버킷 정책을 사용하여 VPC 엔드포인트에서 액세스 제어 섹션을 참조하세요.

다음 보안 모범 사례에서도 Amazon S3의 인프라 보안을 다룹니다.

- Consider VPC endpoints for Amazon S3 access
- Identify and audit all your Amazon S3 buckets

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 참조하세요.

- Amazon S3에 대한 규정 준수 확인
- 공동 책임 모델
- Amazon Web Services: 보안 프로세스의 개요

다음 보안 모범 사례에서도 Amazon S3의 구성 및 취약성 분석을 다룹니다.

- Identify and audit all your Amazon S3 buckets
- AWS Config 사용 설정

액세스 관리

Amazon S3는 다양한 액세스 관리 도구를 제공합니다. 다음은 이러한 기능 및 도구의 목록입니다. 이러한 액세스 관리 도구가 모두 필요한 것은 아니지만, Amazon S3 버킷, 객체 및 기타 S3 리소스에 대한 액세스 권한을 부여하려면 하나 이상을 사용해야 합니다. 이러한 도구를 올바르게 적용하면 의도한 사용자만 리소스에 액세스하도록 할 수 있습니다.

가장 일반적으로 사용되는 액세스 관리 도구는 액세스 정책입니다. 액세스 정책은 AWS 리소스에 연결된 리소스 기반 정책일 수 있습니다(예: 버킷의 버킷 정책). 액세스 정책은 IAM 사용자, 그룹 또는 역할과 같은 AWS Identity and Access Management(IAM) ID에 연결된 ID 기반 정책일 수도 있습니다. 액세스 정책은 누가 무엇에 액세스 할 수 있는지를 나타냅니다. AWS 계정, IAM 사용자, 그룹, 역할에 리소스에서 작업을 수행할 수 있는 권한을 부여하는 액세스 정책을 작성하세요. 예를 들어, 다른 AWS 계정에서 내 버킷에 객체를 업로드할 수 있도록 다른 계정에 `PUT Object` 권한을 부여할 수 있습니다.

Amazon S3에서 제공되는 액세스 관리 도구는 다음과 같습니다. Amazon S3 액세스 제어에 대한 보다 포괄적인 가이드는 Amazon S3의 액세스 제어 섹션을 참조하세요.

버킷 정책

Amazon S3 버킷 정책은 특정 버킷에 연결된 JSON 형식 AWS Identity and Access Management(IAM) 리소스 기반 정책입니다. 버킷 정책을 사용하여 다른 AWS 계정 또는 IAM ID에 버킷과 버킷에 포함된 객체에 대한 권한을 부여합니다. 버킷 정책을 사용하면 많은 S3 액세스 관리 사용 사례를 충족할 수 있

습니다. 버킷 정책을 사용하면 버킷 액세스를 개인화하여 승인된 ID만 리소스에 액세스하고 리소스 내에서 작업을 수행하도록 할 수 있습니다. 자세한 내용은 Amazon S3의 버킷 정책 섹션을 참조하세요.

자격 증명 기반 정책

ID 기반 또는 IAM 사용자 정책은 일종의 AWS Identity and Access Management(IAM) 정책입니다. ID 기반 정책은 AWS 계정의 IAM 사용자, 그룹 또는 역할에 연결된 JSON 형식의 정책입니다. ID 기반 정책을 사용하여 버킷이나 객체에 대한 IAM ID 액세스 권한을 부여할 수 있습니다. 내 계정으로 IAM 사용자, 그룹, 역할을 만들고 액세스 정책을 연결할 수 있습니다. 그런 다음 Amazon S3 리소스를 포함한 AWS 리소스에 대한 액세스 권한을 부여할 수 있습니다. 자세한 내용은 Amazon S3의 ID 기반 정책 섹션을 참조하세요.

S3 Access Grants

S3 Access Grants를 사용하여 회사 ID 디렉터리의 ID(예: Active Directory) 및 AWS Identity and Access Management(IAM) ID 모두에 대해 Amazon S3 데이터에 대한 액세스 권한을 생성할 수 있습니다. S3 Access Grants는 대규모로 데이터 권한을 관리하는 데 도움이 됩니다. 또한 S3 Access Grants는 AWS CloudTrail의 S3 데이터에 액세스하는 데 사용된 최종 사용자 ID와 애플리케이션을 로깅합니다. 이를 통해 S3 버킷의 데이터에 대한 모든 액세스에 대해 최종 사용자 ID까지 상세한 감사 기록이 제공됩니다. 자세한 내용은 S3 Access Grants를 통한 액세스 관리 섹션을 참조하세요.

액세스 포인트

Amazon S3 액세스 포인트는 S3의 공유 데이터세트를 사용하는 애플리케이션에 대한 대규모 데이터 액세스 관리를 간소화합니다. 액세스 포인트는 버킷에 연결된 명명된 네트워크 엔드포인트입니다. 액세스 포인트를 사용하여 객체 업로드 및 검색과 같은 S3 객체 작업을 수행할 수 있습니다. 버킷에는 최대 1만 개의 액세스 포인트가 연결될 수 있으며, 각 액세스 포인트에 대해 고유한 권한과 네트워크 제어를 적용하여 S3 객체에 대한 액세스를 세부적으로 제어할 수 있습니다. S3 액세스 포인트는 동일한 계정 또는 다른 신뢰할 수 있는 계정의 버킷과 연결할 수 있습니다. 액세스 포인트 정책은 기본 버킷 정책과 함께 평가되는 리소스 기반 정책입니다. 자세한 내용은 액세스 포인트로 공유 데이터세트에 대한 액세스 관리 섹션을 참조하세요.

ACL(액세스 제어 목록)

ACL은 피부여자와 그에 부여된 권한을 식별하는 권한 부여 목록입니다. ACL은 다른 AWS 계정에 기본적인 읽기 또는 쓰기 권한을 부여합니다. ACL은 Amazon S3별 XML 스키마를 사용합니다. ACL은 일종의 AWS Identity and Access Management(IAM) 정책입니다. 객체 ACL은 객체에 대한 액세스를 관리하는 데 사용되고, 버킷 ACL은 버킷에 대한 액세스를 관리하는 데 사용됩니다. 버킷 정책의 경우 전체 버킷에 적용되는 하나의 버킷 정책이 있지만 객체 ACL은 객체별로 지정됩니다. 각 객체에 대해 액세스를 개별적으로 제어해야 하는 드문 상황을 제외하고는 ACL을 비활성화한 채로 두는 것이 좋습니

다. ACL 사용에 대한 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

> **Warning**
>
> Amazon S3의 최신 사용 사례 대부분은 ACL을 사용할 필요가 없습니다.

객체 소유권

객체에 대한 액세스를 관리하려면 객체 소유자여야 합니다. 객체 소유권 버킷 수준 설정을 사용하여 버킷에 업로드되는 객체의 소유권을 제어할 수 있습니다. 또한 ACL을 활성화하는 데도 객체 소유권을 사용합니다. 기본적으로 객체 소유권은 버킷 소유자 적용 설정으로 설정되며 모든 ACL이 비활성화되어 있습니다. ACL이 비활성화되면 버킷 소유자는 버킷의 모든 객체를 소유하고 데이터에 대한 액세스를 독점적으로 관리합니다. 액세스를 관리하기 위해 버킷 소유자는 정책이나 다른 액세스 관리 도구(ACL 제외)를 사용합니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

Amazon S3 액세스 제어에 대한 보다 포괄적인 가이드와 추가 모범 사례는 Amazon S3의 액세스 제어 섹션을 참조하세요.

Amazon S3의 데이터 보호

AWS 글로벌 인프라에서 제공하는 복원성과 더불어 Amazon S3는 실수로 인한 삭제나 리전 장애로부터 데이터를 보호하는 다양한 기능을 제공합니다.

S3 복제

라이브 복제를 사용하면 Amazon S3 버킷 전체에 걸쳐 객체를 비동기식으로 자동 복사할 수 있습니다. 객체 복제를 위해 구성된 버킷은 동일한 AWS 계정 또는 다른 계정이 소유할 수 있습니다. 단일 대상 버킷 또는 여러 대상 버킷에 객체를 복제할 수 있습니다. 대상 버킷은 다른 AWS 리전에 있을 수도 있고 소스 버킷과 동일한 리전 내에 있을 수도 있습니다. 장애 조치 제어를 활성화하려면 양방향 복제를 구성하여 리전별 장애 발생 시 소스 및 대상 버킷을 동기화된 상태로 유지하면 됩니다. 자세한 내용은 the section called “리전 내 및 리전 간 객체 복제” 섹션을 참조하세요.

다중 리전 액세스 포인트 및 장애 조치 제어

Amazon S3 다중 리전 액세스 포인트는 애플리케이션이 여러 AWS 리전에 있는 S3 버킷의 요청을 이행하는 데 사용할 수 있는 글로벌 엔드포인트를 제공합니다. 다중 리전 액세스 포인트를 사용하여 단일 리전에서 사용되는 것과 동일한 아키텍처로 다중 리전 애플리케이션을 빌드하면 전 세계 어디에서나 해당 애플리케이션을 실행할 수 있습니다. 다중 리전 액세스 포인트는 혼잡한 퍼블릭 인터넷을 통해 요청을 보내는 대신 Amazon S3에 대한 인터넷 기반 요청의 가속화를 통해 기본 제공 네트워크 복원력을 제공합니다. 다중 리전 액세스 포인트 글로벌 엔드포인트에 대한 애플리케이션 요청은 AWS Global Accelerator를 사용하여 AWS 글로벌 네트워크를 통해 라우팅 상태가 액티브인 가장 가까운 S3 버킷으로 자동으로 라우팅됩니다. 다중 리전 액세스 포인트에 대한 자세한 내용은 the section called “다중 리전 트래픽 관리 ” 섹션을 참조하세요.

Amazon S3 다중 리전 액세스 포인트 장애 조치 제어를 사용하면 리전의 트래픽이 중단되어도 비즈니스 연속성을 유지하는 동시에 애플리케이션에 다중 리전 아키텍처를 제공하여 규정 준수 및 이중화 요구 사항을 충족할 수 있습니다. 리전의 트래픽이 중단되는 경우 다중 리전 액세스 포인트 장애 조치 제어를 사용하여 Amazon S3 다중 리전 액세스 포인트 뒤에 있는 어떤 AWS 리전에서 데이터 액세스와 스토리지 요청을 처리할지 선택할 수 있습니다.

장애 조치를 지원하기 위해 정상 상태일 때는 트래픽이 액티브 리전으로 흐르게 하고 패시브 리전은 장애 조치를 위해 대기 상태로 두는 액티브-패시브 구성으로 다중 리전 액세스 포인트를 설정할 수 있습니다. 양방향 복제 규칙으로 S3 교차 리전 복제(CRR)를 사용 설정한 경우 장애 조치 중에 버킷을 동기화된 상태로 유지할 수 있습니다. 장애 조치 제어에 대한 자세한 내용은 the section called “장애 조치 구성” 섹션을 참조하세요.

S3 버전 관리

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. 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하세요.

S3 객체 잠금

S3 객체 잠금을 사용하면 write once, read many(WORM) 모델을 사용하여 객체를 저장할 수 있습니다. S3 객체 잠금을 사용하면 고정된 시간 동안 또는 무기한으로 객체를 삭제하거나 덮어쓰지 않도록 할 수 있습니다. S3 객체 잠금을 사용하여 WORM 스토리지가 필요한 규제 요구 사항을 충족하거나 객체 변경 및 삭제에 대한 보호 계층을 추가할 수 있습니다. 자세한 내용은 Object Lock으로 객체 잠금 섹션을 참조하세요.

AWS Backup

Amazon S3는 Amazon S3의 데이터를 보호하기 위해 백업 정책을 중앙에서 정의하는 데 사용할 수 있는 완전관리형 정책 기반 서비스인 AWS Backup과 기본적으로 통합됩니다. 백업 정책을 정의하고 Amazon S3 리소스를 정책에 할당하면 AWS Backup이 Amazon S3 백업을 자동으로 생성하고 사용자가 백업 계획에서 지정한 암호화된 백업 볼트에 백업을 안전하게 저장합니다. 자세한 내용은 the section called “데이터 백업” 섹션을 참조하세요.

이러한 기능 중 일부를 함께 사용하여 데이터를 보호하는 방법에 대한 자습서는 자습서: S3 버전 관리, S3 Object Lock, S3 복제를 사용하여 우발적인 삭제나 애플리케이션 버그로부터 Amazon S3의 데이터 보호를 참조하세요.

> **Important**
>
> 위의 기능을 사용하여 데이터를 보호하는 것 외에도 the section called “보안 모범 사례”에서 권장 사항을 검토하는 것이 좋습니다.

주제

- 리전 내 및 리전 간 객체 복제
- 다중 리전 액세스 포인트로 다중 리전 트래픽 관리
- S3 버전 관리로 여러 버전의 객체 유지
- Object Lock으로 객체 잠금

- Amazon S3 데이터 백업

리전 내 및 리전 간 객체 복제

복제를 사용하면 Amazon S3 버킷 전체에 걸쳐 객체를 비동기식으로 자동 복사할 수 있습니다. 객체 복제를 위해 구성된 버킷은 동일한 AWS 계정 또는 다른 계정이 소유할 수 있습니다. 단일 대상 버킷 또는 여러 대상 버킷에 객체를 복제할 수 있습니다. 대상 버킷은 다른 AWS 리전에 있을 수도 있고 소스 버킷과 동일한 리전 내에 있을 수도 있습니다.

복제에는 라이브 복제와 온디맨드 복제의 두 가지 유형이 있습니다.

- 라이브 복제 - 새 객체 및 업데이트된 객체가 원본 버킷에 기록될 때 자동으로 복제하려면 라이브 복제를 사용합니다. 라이브 복제는 복제를 설정하기 전에 버킷에 있던 객체를 자동으로 복제하지 않습니다. 복제를 설정하기 전에 존재한 객체를 복제하려면 온디맨드 복제를 사용합니다.
- 온디맨드 복제 - 요청 시 원본 버킷의 기존 객체를 하나 이상의 대상 버킷으로 복제하려면 S3 Batch Replication을 사용하세요. 기존 객체 복제에 대한 자세한 내용은 S3 Batch Replication을 사용하는 경우 섹션을 참조하세요.

라이브 복제에는 교차 리전 복제(CRR)와 단일 리전 복제(SRR)라는 2가지 형태가 있습니다.

- 교차 리전 복제(CRR) - CRR을 사용하여 서로 다른 AWS 리전의 Amazon S3 버킷에서 객체를 복사할 수 있습니다. CRR에 대한 자세한 내용은 the section called “교차 리전 복제를 사용하는 경우” 단원을 참조하십시오.
- 동일 리전 복제(SRR) - SRR을 사용하여 동일한 AWS 리전의 Amazon S3 버킷에서 객체를 복사할 수 있습니다. SRR에 대한 자세한 내용은 the section called “동일 리전 복제를 사용하는 경우” 섹션을 참조하세요.

주제

- 복제를 사용하는 이유
- 교차 리전 복제를 사용하는 경우
- 동일 리전 복제를 사용하는 경우
- 양방향 복제를 사용해야 하는 경우
- S3 Batch Replication을 사용하는 경우
- 워크로드 요구 사항 및 라이브 복제
- Amazon S3는 무엇을 복제합니까?

- 복제본에 대한 요구 사항 및 고려 사항
- 라이브 복제 설정 개요
- 라이브 복제 관리 또는 일시 중지
- 배치 복제를 사용한 기존 객체 복제
- 복제 문제 해결
- 지표, 이벤트 알림 및 상태를 통한 복제 모니터링

복제를 사용하는 이유

복제는 다음을 지원합니다.

- 메타데이터를 유지하면서 객체 복제 – 복제를 사용하여 소스 객체 생성 시간 및 버전 ID와 같은 모든 메타데이터를 유지하는 객체의 복사본을 생성할 수 있습니다. 이 기능은 복제본이 소스 객체와 동일한지 확인해야 하는 경우에 중요합니다.
- 서로 다른 스토리지 클래스로 객체 복제 – 복제를 사용하여 객체를 S3 Glacier Flexible Retrieval, 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 구성을 사용하여 객체를 오래된 스토리지 클래스로 이동시킬 수 있습니다.
- 다른 소유권으로 객체 복사본 유지 – 소스 객체의 소유자에 상관없이 대상 버킷을 소유한 AWS 계정로 복제본 소유권을 변경하도록 Amazon S3에 지시할 수 있습니다. 이를 사용자 재정의 옵션이라고 합니다. 이 옵션을 사용하여 객체 복제본에 대한 액세스를 제한할 수 있습니다.
- 여러 AWS 리전에 객체 저장 – 서로 다른 AWS 리전에 여러 대상 버킷을 설정하여 데이터가 보관된 곳의 지리적 차이를 보장할 수 있습니다. 이 기능은 특정 규정 준수 요구 사항을 충족하는 데 도움이 될 수 있습니다.
- 15분 이내에 객체 복제 - S3 Replication Time Control(S3 RTC)을 사용하면 예측 가능한 기간 내에 동일한 AWS 리전 또는 다른 리전 간에 데이터를 복제할 수 있습니다. S3 RTC는 Amazon S3에 저장된 새 객체의 99.99%를 15분 이내에 복제합니다(서비스 수준 계약에 따라 지원됨). 자세한 내용은 the section called “S3 Replication Time Control 사용” 섹션을 참조하세요.

> **Note**
>
> S3 RTC는 배치 복제에 적용되지 않습니다. 배치 복제는 온디맨드 복제 작업이며, S3 배치 작업을 통해 추적할 수 있습니다. 자세한 내용은 작업 상태 및 완료 보고서 추적 섹션을 참조하세요.

- 버킷 동기화, 기존 객체 복제, 이전에 실패 또는 복제된 객체 복제 - 버킷을 동기화하고 기존 객체를 복제하려면 배치 복제를 온디맨드 복제 작업으로 사용합니다. Batch Replication을 사용하는 경우에 대한 자세한 내용은 S3 Batch Replication을 사용하는 경우 섹션을 참조하세요.
- 객체 복제 및 다른 AWS 리전에 있는 버킷으로 장애 조치 - 데이터 복제 중에 모든 메타데이터와 객체를 버킷 간에 동기화된 상태로 유지하려면 Amazon S3 다중 리전 액세스 포인트 장애 조치 제어를 구성하기 전에 양방향 복제 규칙을 사용하세요. 양방향 복제 규칙은 트래픽이 장애 조치 목적지인 S3 버킷에 데이터를 쓸 때 해당 데이터가 원본 버킷으로 다시 복제되도록 하는 데 도움이 됩니다.

교차 리전 복제를 사용하는 경우

S3 교차 리전 복제(CRR)는 서로 다른 AWS 리전의 Amazon S3 버킷에서 객체를 복사하는 데 사용됩니다. CRR은 다음을 수행하는 데 도움이 됩니다.

- 규정 준수 요구 사항 충족 – 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. 이러한 요구 사항을 충족하려면 교차 리전 복제를 사용하여 먼 거리의 AWS 리전 간에 데이터를 복제합니다.
- 대기 시간 최소화 - 고객이 두 군데의 지리적 위치를 갖는 경우 사용자와 지리적으로 더 가까운 AWS 리전에 객체 복사본을 유지하여 객체 액세스 대기 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.

동일 리전 복제를 사용하는 경우

동일 리전 복제(SRR)는 동일한 AWS 리전의 Amazon S3 버킷에서 객체를 복사하는 데 사용됩니다. SRR은 다음을 수행하는 데 도움이 됩니다.

- 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, 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.

양방향 복제를 사용해야 하는 경우

- 여러 개의 AWS 리전 간에 공유 데이터 세트를 빌드하려는 경우 - 복제본 수정 동기화를 사용하면 복제 객체에 객체 액세스 제어 목록(ACL), 객체 태그 또는 객체 잠금과 같은 메타데이터 변경을 쉽게 복제할 수 있습니다. 이 양방향 복제는 모든 객체 및 객체 메타데이터 변경 사항을 동기화된 상태로 유지하려는 경우에 중요합니다. 동일하거나 다른 AWS 리전에 있는 둘 이상의 버킷 간에 양방향 복제를 수행할 때 새 복제 규칙이나 기존 복제 규칙에서 복제본 수정 동기화를 사용 설정할 수 있습니다.
- 장애 조치 중에 리전 간에 데이터를 동기화된 상태로 유지하려는 경우 - 다중 리전 액세스 포인트에서 직접 S3 교차 리전 복제(CRR)로 양방향 복제 규칙을 구성하여 AWS 리전 간에 버킷의 데이터를 동기화할 수 있습니다. 장애 조치를 언제 시작할지 정보에 입각한 결정을 내리려면 Amazon CloudWatch, S3 Replication Time Control(S3 RTC) 또는 다중 리전 액세스 포인트에서 복제를 모니터링하도록 S3 복제 지표를 활성화할 수도 있습니다.
- 애플리케이션의 가용성을 높이려는 경우 - 리전의 트래픽이 중단되더라도 양방향 복제 규칙을 사용하여 데이터를 복제하는 동안 모든 메타데이터와 객체를 버킷 간에 동기화된 상태로 유지할 수 있습니다.

S3 Batch Replication을 사용하는 경우

배치 복제는 기존 객체를 온디맨드 옵션으로 다른 버킷에 복제합니다. 라이브 복제와 달리 필요에 따라 이러한 작업을 실행할 수 있습니다. Batch Replication은 다음을 지원합니다.

- 기존 객체 복제 - 배치 복제를 사용하여 동일 리전 복제 또는 교차 리전 복제가 구성되기 전에 버킷에 추가되었던 객체를 복제할 수 있습니다.
- 이전에 복제 실패한 객체 복제 - 배치 복제 작업을 필터링하여 복제 상태가 실패인 객체의 복제를 시도할 수 있습니다.
- 이미 복제된 객체 복제 - 데이터의 여러 복사본을 별도의 AWS 계정 또는 AWS 리전에 저장해야 할 수도 있습니다. 배치 복제는 기존 객체를 새로 추가된 대상에 복제할 수 있습니다.
- 복제 규칙에서 생성되었던 객체의 복제본 복제 - 복제 구성은 대상 버킷에 객체의 복제본을 생성합니다. 객체의 복제본은 배치 복제를 통해서만 복제할 수 있습니다.

워크로드 요구 사항 및 라이브 복제

워크로드 요구 사항에 따라 일부 라이브 복제 유형이 다른 유형보다 사용 사례에 더 적합할 수 있습니다. 다음 표를 사용하여 상황에 사용할 복제 유형과 워크로드에 S3 Replication Time Control(S3 RTC)

을 사용할지 여부를 결정합니다. S3 RTC는 Amazon S3에 저장된 새 객체의 99.99%를 15분 이내에 복제합니다(서비스 수준 계약(SLA)에 따라 지원됨). 자세한 내용은 the section called “S3 Replication Time Control 사용” 섹션을 참조하세요.

워크로드 요구 사항	S3 RTC(15분 SLA)	리전 간 복제(CRR)	동일 리전 복제(SRR)
서로 다른 AWS 계정 간에 객체 복제	예	예	예
24~48시간 내에 동일한 AWS 리전 내에서 객체 복제(SLA 미지원)	아니요	아니요	예
24~48시간 내에 다른 AWS 리전 간에 객체 복제(SLA 미지원)	아니요	예	아니요
예측 가능한 복제 시간: SLA를 통해 15분 이내에 99.9%의 객체를 복제할 수 있습니다.	예	아니요	아니요

Amazon S3는 무엇을 복제합니까?

Amazon S3는 버킷에서 복제가 구성된 특정 항목만 복제합니다.

주제

- 복제 구성으로 복제되는 것은 무엇입니까?
- 복제 구성으로 복제되지 않는 것은 무엇입니까?

복제 구성으로 복제되는 것은 무엇입니까?

기본적으로 Amazon S3는 다음을 복제합니다.

- 복제 구성을 추가한 후에 생성된 객체입니다.

- 암호화되지 않은 객체입니다.
- 고객 제공 키(SSE-C)를 사용하여 암호화된 객체, Amazon S3 관리형 키(SSE-S3)를 사용하여 저장 시 암호화된 객체, AWS Key Management Service(SSE-KMS)에 저장된 KMS 키로 암호화된 객체. 자세한 내용은 the section called “암호화된 객체 복제” 섹션을 참조하세요.
- 원본 객체에서 복제본으로 가는 객체 메타데이터입니다. 복제본에서 원본 객체로 메타데이터를 복제하는 방법에 대한 자세한 내용은 복제본 수정 동기화를 사용하여 메타데이터 변경 복제 단원을 참조하세요.
- 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 소유권를 참조하세요.
- 객체 ACL 업데이트. 단, 원본 버킷과 대상 버킷을 동일한 계정이 소유하지 않은 경우 사용자가 Amazon S3에 복제본 소유권을 변경하도록 지시하지 않은 경우에 한합니다.

  자세한 내용은 복제본 소유자 변경 섹션을 참조하세요.

  Amazon S3에서 두 ACL을 동기화하려면 시간이 약간 걸릴 수 있습니다. 이 소유권 변경은 버킷에 복제 구성을 추가한 후 생성된 객체에만 적용됩니다.
- 객체 태그(있는 경우).
- 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 Object Lock으로 객체 잠금 섹션을 참조하세요.

삭제 작업이 복제에 미치는 영향

원본 버킷에서 객체를 삭제하는 경우 기본적으로 다음 작업이 이루어집니다.

- 객체 버전 ID를 지정하지 않고 DELETE 요청을 수행하면 Amazon S3는 삭제 마커를 추가합니다. Amazon S3는 다음과 같이 삭제 마커를 처리합니다.
  - 최신 버전의 복제 구성을 사용하는 경우(즉, 사용자가 복제 구성 규칙의 `Filter` 요소를 지정) 기본적으로 Amazon S3는 삭제 마커를 복제하지 않습니다. 그러나 태그 기반이 아닌 규칙에도 삭제 마커 복제를 추가할 수 있습니다. 자세한 내용은 버킷 간 삭제 마커 복제 섹션을 참조하세요.
  - `Filter` 요소를 지정하지 않으면 Amazon S3는 복제 구성이 버전 V1이라고 가정하고 사용자 작업에서 생성된 삭제 마커를 복제합니다. 그러나 Amazon S3가 수명 주기 작업 때문에 객체를 삭제하는 경우 삭제 마커는 대상 버킷에 복제되지 않습니다.

- 사용자가 DELETE 요청에서 삭제할 객체 버전 ID를 지정할 경우, Amazon S3는 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 구성으로 복제되지 않는 것은 무엇입니까?

기본적으로 Amazon S3는 다음을 복제하지 않습니다.

- 원본 버킷의 객체는 다른 복제 규칙에 따라 생성된 복제본입니다. 예를 들어, 버킷 A가 소스이고 버킷 B가 대상인 복제를 구성하는 경우. 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.

  복제본인 객체를 복제하려면 배치 복제를 사용합니다. 기존 객체 복제에서 배치 복제에 대해 자세히 알아보세요.
- 다른 대상에 이미 복제된 원본 버킷의 객체입니다. 예를 들어, 기존 복제 구성에서 대상 버킷을 변경하더라도 Amazon S3가 해당 객체를 다시 복제하지 않습니다.

  이전에 복제된 객체를 복제하려면 Batch Replication을 사용합니다. 기존 객체 복제에서 배치 복제에 대해 자세히 알아보세요.
- 배치 복제는 대상 버킷에서 객체의 버전 ID로 삭제된 객체를 다시 복제하는 것을 지원하지 않습니다. 이러한 객체를 다시 복제하려면 배치 복사 작업을 사용하여 소스 객체를 제자리에 복사할 수 있습니다. 이러한 객체를 제자리에 복사하면 소스 버킷에 객체의 새 버전이 생성되고 대상에 대한 복제가 자동으로 시작됩니다. 배치 복사 사용 방법에 대한 자세한 내용은 배치 작업을 사용하여 객체를 복사하는 예 섹션을 참조하세요.
- 기본적으로 다른 AWS 계정에서 복제하는 경우 소스 버킷에 추가된 삭제 마커는 복제되지 않습니다.

  삭제 마커를 복제하는 방법에 대한 자세한 내용은 버킷 간 삭제 마커 복제 섹션을 참조하세요.
- S3 Glacier Flexible Retrieval, S3 Glacier Deep Archive, S3 Intelligent-Tiering Archive Access 또는 S3 Intelligent-Tiering Deep Archive Access 스토리지 클래스 또는 계층에 저장된 객체입니다. 이러한 객체는 복원하여 다른 스토리지 클래스에 복사할 때까지 복제할 수 없습니다.

  S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive에 대한 자세한 내용은 거의 액세스하지 않는 객체를 위한 스토리지 클래스 섹션을 참조하세요.

  S3 Intelligent-Tiering에 대한 자세한 내용은 Amazon S3 Intelligent-Tiering을 사용한 스토리지 비용 관리 섹션을 참조하세요.

- 버킷 소유자에게 충분한 복제 권한이 없는 소스 버킷의 객체입니다.

  객체 소유자가 버킷 소유자에게 권한을 부여할 수 있는 방법에 대한 자세한 내용은 버킷 소유자가 완벽하게 제어할 수 있도록 보증하면서 객체에 업로드하는 크로스 계정 권한 부여 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. 이 기능을 사용하여 소스 버킷과 대상 버킷에 서로 다른 버킷 구성을 할 수 있습니다.

- 수명 주기 구성에 의해 수행되는 작업.

  예를 들어 원본 버킷에서만 수명 주기 구성이 사용 설정되면 Amazon S3는 만료된 객체에 대한 삭제 마커를 만들지만 그 마커를 복제하지는 않습니다. 소스 버킷과 대상 버킷에 동일한 수명 주기 구성을 적용하려는 경우 두 버킷 모두에서 동일한 수명 주기를 사용 설정합니다. 수명 주기 구성에 대한 자세한 내용은 객체 수명 주기 관리 섹션을 참조하십시오.

- 실시간 복제와 함께 태그 기반 복제 규칙을 사용하는 경우 `PutObject` 작업에서 매칭되는 복제 규칙 태그로 새 객체에 태그를 지정해야 합니다. 그러지 않으면 객체가 복제되지 않습니다. `PutObject` 작업 후 객체에 태그가 지정된 경우 해당 객체도 복제되지 않습니다.

  `PutObject` 작업을 수행한 후 태그가 지정된 객체를 복제하려면 S3 배치 복제를 사용해야 합니다. Batch Replication에 대한 자세한 내용은 기존 객체 복제 섹션을 참조하세요.

복제본에 대한 요구 사항 및 고려 사항

Amazon S3 복제에 관한 요구 사항은 다음과 같습니다.

- 소스 버킷 소유자는 자신의 계정에 대해 소스 및 대상 AWS 리전을 사용해야 합니다. 대상 버킷 소유자는 자신의 계정에 대해 대상 리전을 사용 설정해야 합니다.

  AWS 리전 활성화 또는 비활성화에 대한 자세한 내용은 AWS Account Management 참조 안내서의 계정에서 사용할 수 있는 AWS 리전 지정을 참조하세요.

- 소스 버킷과 대상 버킷 모두에서 버전 관리를 사용 설정해야 합니다. 버전 관리에 대한 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하세요.

- Amazon S3가 사용자를 대신해서 대상 버킷에 해당 원본 버킷의 객체를 복제할 권한을 가지고 있어야 합니다. 이러한 권한에 대한 자세한 내용은 라이브 복제에 대한 권한 설정 단원을 참조하십시오.

- 원본 버킷 소유자가 버킷 내 객체를 소유하지 않은 경우 객체 소유자는 객체 ACL(액세스 제어 목록)을 사용하여 버킷 소유자에게 READ 및 READ_ACP 권한을 부여해야 합니다. 자세한 내용은 ACL(액세스 제어 목록) 개요 섹션을 참조하세요.
- 원본 버킷에 S3 객체 잠금이 사용 설정되어 있는 경우, 대상 버킷에도 S3 객체 잠금이 사용 설정되어 있어야 합니다.

  객체 잠금이 활성화되어 있는 버킷에서 복제를 활성화하려면 AWS Command Line Interface, REST API 또는 AWS SDK를 사용해야 합니다. 더 일반적인 내용은 Object Lock으로 객체 잠금 섹션을 참조하세요.

  > **Note**
  >
  > 복제 설정에 사용할 AWS Identity and Access Management(IAM) 역할에서 소스 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 라이브 복제에 대한 권한 설정 섹션을 참조하세요.

자세한 내용은 라이브 복제 설정 개요 섹션을 참조하세요.

소스 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유하는 크로스 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 소스 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유할 경우 권한 부여 섹션을 참조하세요.
- 대상 버킷을 요청자 지불 버킷으로 구성할 수 없습니다. 자세한 내용은 스토리지 전송 및 사용량에 대한 요청자 지불 범용 버킷 사용 섹션을 참조하세요.

복제를 위한 고려 사항

복제 구성을 생성하기 전에 다음 고려 사항에 유의하세요.

주제

- 수명 주기 구성 및 객체 복제본
- 버전 관리 구성 및 복제 구성

- S3 Intelligent-Tiering과 함께 S3 복제 사용
- 로그 구성 및 복제 구성
- CRR 및 대상 리전
- S3 배치 복제
- S3 복제 시간 제어

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. 대상에서 복제본을 사용할 수 있기까지는 시간이 다소 걸리지만, 복제본 생성 시간은 원본 버킷에서 객체를 생성하는 데 걸린 시간과 동일합니다. 대상 버킷에서 수명 주기 구성이 활성화된 경우, 수명 주기 규칙은 대상 버킷에서 복제본을 사용할 수 있게 된 시간이 아니라 객체의 원본 생성 시간을 적용합니다.

복제 구성을 사용하려면 버킷에서 버전 관리를 사용해야 합니다. 버킷에서 버전 관리를 사용하는 경우 다음에 유의하세요.

- 객체 만료 수명 주기 구성이 있는 경우 버전 관리를 활성화한 후 `NonCurrentVersionExpiration` 정책을 추가하여 버전 관리를 사용 설정하기 전과 동일한 영구 삭제 동작을 유지합니다.
- 전환 수명 주기 구성이 있는 경우 버전 관리를 활성화한 후 `NonCurrentVersionTransition` 정책을 추가하는 것을 고려하세요.

버전 관리 구성 및 복제 구성

버킷에 대한 복제를 구성하려면 원본 및 대상 버킷 모두 버전 관리가 사용 설정되어 있어야 합니다. 원본 및 대상 버킷에 버전 관리를 사용 설정하고 원본 버킷의 복제를 구성한 후, 다음과 같은 문제가 발생할 수 있습니다.

- 원본 버킷의 버전 관리를 사용 중지하려는 경우, Amazon S3는 오류를 반환합니다. 원본 버킷의 버전 관리를 사용 중지하려면 우선 복제 구성을 삭제해야 합니다.
- 대상 버킷의 버전 관리를 사용 중지하는 경우, 복제가 실패합니다. 원본 객체의 복제 상태가 `FAILED`입니다.

S3 Intelligent-Tiering과 함께 S3 복제 사용

S3 Intelligent-Tiering은 가장 비용 효율적인 액세스 계층으로 데이터를 자동으로 이동하여 스토리지 비용을 최적화하도록 설계된 스토리지 클래스입니다. 약간의 월별 객체 모니터링 및 자동화 요금만 지불하면 S3 Intelligent-Tiering에서 액세스 패턴을 모니터링하고, 액세스되지 않은 객체를 저렴한 액세스 계층으로 자동으로 이동합니다.

S3 배치 복제를 사용하여 S3 Intelligent-Tiering에 저장된 객체를 복제하거나 CopyObject 또는 UploadPartCopy를 호출하는 것은 액세스를 구성합니다. 이러한 경우 복사 또는 복제 작업의 소스 객체가 계층화됩니다.

S3 Intelligent-Tiering에 대한 자세한 내용은 Amazon S3 Intelligent-Tiering을 사용한 스토리지 비용 관리 단원을 참조하세요.

로그 구성 및 복제 구성

Amazon S3가 복제가 사용 설정된 버킷에 대한 로그를 제공하는 경우, 해당 로그 객체가 복제됩니다.

소스 또는 대상 버킷에서 서버 액세스 로그 또는 AWS CloudTrail 로그가 사용된 경우 Amazon S3는 로그에 복제 관련 요청을 포함합니다. 예를 들어 Amazon S3는 복제되는 각 객체를 기록합니다.

CRR 및 대상 리전

Amazon S3 교차 리전 복제(CRR)는 서로 다른 AWS 리전의 S3 버킷에서 객체를 복사하는 데 사용됩니다. 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.

소스 버킷의 리전으로 미국 동부(버지니아 북부)(`us-east-1`)를 선택했다고 가정하겠습니다. 미국 서부(오리건)(`us-west-2`)를 대상 버킷의 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.

S3 배치 복제

배치 복제에 관한 고려 사항은 S3 배치 복제 고려 사항 섹션을 참조하세요.

S3 복제 시간 제어

S3 Replication Time Control(S3 RTC)에 대한 자세한 내용은 S3 RTC에 대한 모범 사례 및 지침 섹션을 참조하세요.

라이브 복제 설정 개요

> **Note**
>
> 복제를 설정하기 전에 존재한 객체는 자동으로 복제되지 않습니다. 즉, Amazon S3는 소급하여 객체를 복제하지 않습니다. 복제 구성 전에 생성된 객체를 복제하려면 S3 배치 복제를 사용합니다. 배치 복제 구성에 대한 자세한 내용은 기존 객체 복제 섹션을 참조하세요.

동일 리전 복제(SRR) 또는 교차 리전 복제(CRR)와 같은 라이브 복제를 활성화하려면 소스 버킷에 복제 구성을 추가합니다. 이 구성은 지정된 대로 객체를 복제하도록 Amazon S3에 지시합니다. 복제 구성에서는 다음을 제공해야 합니다.

- 대상 버킷 - Amazon S3가 객체를 복제할 버킷입니다.
- 복제할 객체 - 소스 버킷 또는 객체 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 S3는 `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.
- AWS Identity and Access Management(IAM) 역할 – Amazon S3는 사용자를 대신하여 객체를 복제하기 위해 이 역할을 맡습니다. 이 IAM 역할 생성 및 권한 관리에 대한 자세한 내용은 라이브 복제에 대한 권한 설정 섹션을 참조하세요.

이러한 최소 요건에 더해, 다음 옵션을 선택할 수 있습니다.

- 복제본 스토리지 클래스 - 기본적으로 Amazon S3는 소스 객체와 동일한 스토리지 클래스를 사용하여 객체 복제본을 저장합니다. 복제본에 대해 다른 스토리지 클래스를 지정할 수 있습니다.
- 복제본 소유권 - Amazon S3는 소스 객체 소유자가 객체 복제본을 계속 소유하게 된다고 가정합니다. 따라서 객체를 복제할 때 해당되는 객체 액세스 제어 목록(ACL) 또는 S3 객체 소유권 설정도 복제됩니다. 소스 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유하는 경우 복제본 소유자를 대상 버킷을 소유한 AWS 계정으로 변경하도록 복제를 구성할 수 있습니다. 자세한 내용은 the section called “복제본 소유자 변경” 섹션을 참조하세요.

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하여 복제를 구성할 수 있습니다. 복제 설정 방법에 대한 자세한 설명은 the section called “복제 실습 과정” 섹션을 참조하세요.

Amazon S3는 복제 규칙 설정을 지원하기 위해 REST API 작업을 제공합니다. 자세한 내용은 Amazon Simple Storage Service API 참조의 다음 주제들을 참조하십시오.

- PutBucketReplication
- GetBucketReplication
- DeleteBucketReplication

주제

- 복제 구성 파일 요소
- 라이브 복제에 대한 권한 설정
- 라이브 복제 구성 예제

복제 구성 파일 요소

Amazon S3는 복제 구성을 XML로 저장합니다. Amazon S3 REST API를 통해 프로그래밍 방식으로 복제를 구성하는 경우, 이 XML 파일에 복제 구성의 다양한 요소를 지정합니다. AWS Command Line Interface(AWS CLI)를 통해 복제를 구성하는 경우 JSON 형식을 사용하여 복제 구성을 지정합니다. JSON 예제는 the section called “복제 실습 과정”의 안내를 참조하세요.

> **Note**
>
> 최신 버전의 복제 구성 XML 형식은 V2입니다. XML V2 복제 구성은 규칙에 대한 `<Filter>` 요소 그리고 S3 Replication Time Control(S3 RTC)을 지정하는 규칙을 포함하는 구성입니다. 복제 구성 버전을 보려면 `GetBucketReplication` API 작업을 사용할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 GetBucketReplication를 참조하십시오. 이전 버전과의 호환성을 위해 Amazon S3는 XML V1 복제 구성 형식을 계속 지원합니다. XML V1 복제 구성 형식을 사용한 적이 있는 경우 이전 버전과의 호환성 고려 사항은 이전 버전과의 호환성 고려 사항 섹션을 참조하세요.

복제 구성 XML 파일에서 다음 예제와 같이 AWS Identity and Access Management(IAM) 역할과 하나 이상의 규칙을 지정해야 합니다.

```
<ReplicationConfiguration>
    <Role>IAM-role-ARN</Role>
    <Rule>
        ...
    </Rule>
    <Rule>
         ...
    </Rule>
     ...
</ReplicationConfiguration>
```

Amazon S3는 사용자가 부여한 권한 없이 객체를 복제할 수 없습니다. 복제 구성에서 지정한 IAM 역할로 Amazon S3에 대한 권한을 부여합니다. Amazon S3는 사용자를 대신하여 객체를 복제하기 위해 이 IAM 역할을 맡습니다. 먼저 IAM 역할에 필요한 권한을 부여해야 합니다. 권한 관리에 대한 자세한 내용은 라이브 복제에 대한 권한 설정 단원을 참조하십시오.

다음 시나리오에서는 복제 구성에 딱 한 가지 규칙을 추가합니다.

- 모든 객체를 복제하려는 경우
- 객체의 하위 집합을 하나만 복제하려는 경우. 규칙에 필터를 추가하여 객체 하위 집합을 식별하는 경우 필터에서 객체 키 접두사, 태그 또는 둘 모두를 지정하여 규칙이 적용되는 객체의 하위 집합을 식별합니다. 필터는 지정한 값과 정확히 일치하는 객체를 대상으로 합니다.

서로 다른 객체 하위 집합을 복제하려면 복제 구성에 여러 규칙을 추가하면 됩니다. 각 규칙에서 다른 하위 집합을 선택하는 필터를 지정합니다. 예를 들어 `tax/` 또는`document/` 키 접두사를 갖는 객체를 복제하도록 선택할 수 있습니다. 이렇게 하려면 두 가지 규칙을 추가합니다. 하나는 `tax/` 키 접두사 필터를 지정하고 다른 하나는 `document/` 키 접두사를 지정합니다. 객체 키 접두사에 대한 자세한 내용은 접두어를 사용한 객체 구성 섹션을 참조하세요.

다음 섹션에서 정보를 추가로 제공합니다.

주제

- 기본 규칙 구성
- 선택 사항: 필터 지정
- 추가 대상 구성
- 예제 복제 구성
- 이전 버전과의 호환성 고려 사항

기본 규칙 구성

각 규칙은 규칙의 상태 및 우선 순위를 포함해야 합니다. 규칙은 삭제 마커를 복제할지 여부를 나타내야 합니다.

- <Status> 요소는 Enabled 또는 Disabled 값을 사용하여 규칙이 활성화 또는 비활성화되는지 여부를 나타냅니다. 규칙이 사용 중지되면 Amazon S3는 규칙에 지정된 작업을 수행하지 않습니다.
- <Priority> 요소는 둘 이상의 복제 규칙이 충돌할 때마다 어떤 규칙의 우선순위가 높은지를 나타냅니다. Amazon S3는 모든 복제 규칙에 따라 객체 복제를 시도합니다. 그러나 동일한 대상 버킷이 있는 규칙이 둘 이상인 경우, 우선 순위가 가장 높은 규칙에 따라 객체가 복제됩니다. 숫자가 클수록 우선 순위가 높아집니다.
- <DeleteMarkerReplication> 요소는 Enabled 또는 Disabled 값을 사용하여 삭제 마커를 복제할지 여부를 나타냅니다.

<Destination> 요소 구성에서는 Amazon S3에서 객체를 복제할 하나 이상의 대상 버킷 이름을 입력해야 합니다.

다음 예제는 V2 규칙에 대한 최소 요구 사항을 보여줍니다. 이전 버전과의 호환성을 위해 Amazon S3는 XML V1 형식을 계속 지원합니다. 자세한 내용은 이전 버전과의 호환성 고려 사항 섹션을 참조하세요.

```
...
    <Rule>
        <ID>Rule-1</ID>
        <Status>Enabled-or-Disabled</Status>
        <Filter>
            <Prefix></Prefix>
        </Filter>
        <Priority>integer</Priority>
        <DeleteMarkerReplication>
           <Status>Enabled-or-Disabled</Status>
        </DeleteMarkerReplication>
        <Destination>
           <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</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 모두 가진 객체의 하위 집합에 적용됩니다.

객체 키 접두사를 기준으로 필터링

객체 키 접두사를 기반으로 한 필터를 사용하여 규칙을 지정하려면 다음 XML을 사용합니다. 규칙당 접두사는 하나만 지정할 수 있습니다.

```
<Rule>
    ...
    <Filter>
        <Prefix>key-prefix</Prefix>
    </Filter>
    ...
</Rule>
...
```

객체 태그를 기준으로 필터링

객체 태그를 기반으로 한 필터를 사용하여 규칙을 지정하려면 다음 XML을 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 접두사 및 객체 태그로 필터링

키 접두사와 객체 태그가 조합된 규칙 필터를 지정하려면 다음 XML을 사용합니다. <And> 상위 요소에서 이러한 필터를 래핑합니다. Amazon S3는 논리적 AND 작업을 수행하여 이들 필터를 조합합니다. 즉, 규칙은 특정 키 접두사와 특정 태그를 모두 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>
...
```

Note

- 빈 <Filter> 요소가 있는 규칙을 지정하면 규칙이 버킷의 모든 객체에 적용됩니다.
- 실시간 복제와 함께 태그 기반 복제 규칙을 사용하는 경우 PutObject 작업에서 매칭되는 복제 규칙 태그로 새 객체에 태그를 지정해야 합니다. 그러지 않으면 객체가 복제되지 않습니다. PutObject 작업 후 객체에 태그가 지정된 경우 해당 객체도 복제되지 않습니다.

> PutObject 작업을 수행한 후 태그가 지정된 객체를 복제하려면 S3 배치 복제를 사용해야 합니다. Batch Replication에 대한 자세한 내용은 기존 객체 복제 섹션을 참조하세요.

추가 대상 구성

대상 구성에서는 Amazon S3가 객체를 복제할 버킷을 지정합니다. 한 원본 버킷에서 하나 이상의 대상 버킷으로 객체를 복제하도록 복제를 구성할 수 있습니다.

```
...
<Destination>
    <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
</Destination>
...
```

<Destination> 요소에 다음 옵션을 추가할 수 있습니다.

주제

- 스토리지 클래스 지정
- 여러 대상 버킷 추가
- 여러 대상 버킷이 있는 각 복제 규칙에 대해 서로 다른 파라미터 지정
- 복제본 소유권 변경
- S3 복제 시간 제어 사용 설정
- AWS KMS를 사용하여 서버 측 암호화로 생성된 객체 복제

스토리지 클래스 지정

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:::amzn-s3-demo-destination-bucket</Bucket>
       <StorageClass>storage-class</StorageClass>
</Destination>
...
```

여러 대상 버킷 추가

다음과 같이 단일 복제 구성에서 여러 대상 버킷을 추가할 수 있습니다.

```
...
<Rule>
    <ID>Rule-1</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Enabled-or-Disabled</Status>
    </DeleteMarkerReplication>
    <Destination>
       <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket1</Bucket>
    </Destination>
</Rule>
<Rule>
    <ID>Rule-2</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Enabled-or-Disabled</Status>
    </DeleteMarkerReplication>
    <Destination>
       <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket2</Bucket>
    </Destination>
</Rule>
...
```

여러 대상 버킷이 있는 각 복제 규칙에 대해 서로 다른 파라미터 지정

단일 복제 구성에서 여러 대상 버킷을 추가할 때 다음과 같이 각 복제 규칙에 대해 서로 다른 파라미터를 지정할 수 있습니다.

```
...
<Rule>
    <ID>Rule-1</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Disabled</Status>
    </DeleteMarkerReplication>
      <Metrics>
```

```
    <Status>Enabled</Status>
    <EventThreshold>
      <Minutes>15</Minutes>
    </EventThreshold>
  </Metrics>
    <Destination>
       <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket1</Bucket>
    </Destination>
</Rule>
<Rule>
    <ID>Rule-2</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Enabled</Status>
    </DeleteMarkerReplication>
      <Metrics>
    <Status>Enabled</Status>
    <EventThreshold>
      <Minutes>15</Minutes>
    </EventThreshold>
  </Metrics>
  <ReplicationTime>
    <Status>Enabled</Status>
    <Time>
      <Minutes>15</Minutes>
    </Time>
  </ReplicationTime>
    <Destination>
       <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket2</Bucket>
    </Destination>
</Rule>
...
```

복제본 소유권 변경

소스 버킷과 대상 버킷을 동일한 계정에서 소유하지 않는 경우 복제본의 소유권을 대상 버킷을 소유한 AWS 계정로 변경할 수 있습니다. 이렇게 하려면 <AccessControlTranslation> 요소를 추가합니다. 이 요소는 Destination 값을 가져옵니다.

```
...
<Destination>
   <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
```

```
    <Account>destination-bucket-owner-account-id</Account>
    <AccessControlTranslation>
        <Owner>Destination</Owner>
    </AccessControlTranslation>
</Destination>
...
```

복제 구성에 <AccessControlTranslation> 요소를 추가하지 않으면 소스 객체를 소유한 계정과 동일한 AWS 계정에서 복제본을 소유합니다. 자세한 내용은 복제본 소유자 변경 섹션을 참조하세요.

S3 복제 시간 제어 사용 설정

복제 구성에서 S3 Replication Time Control(S3 RTC)을 사용 설정할 수 있습니다. S3 RTC는 대부분의 객체를 몇 초 만에 복제하고, 이러한 객체의 99.99%를 15분 내에 복제합니다(서비스 수준 계약에 따라).

> **Note**
>
> <Minutes>15</Minutes> 값은 <EventThreshold> 및 <Time> 요소에만 허용됩니다.

```
...
<Destination>
  <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</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 Control을 사용하여 규정 준수 요구 사항 충족 섹션을 참조하세요. API 예제는 Amazon Simple Storage Service API 참조의 PutBucketReplication 섹션을 참조하세요.

AWS KMS를 사용하여 서버 측 암호화로 생성된 객체 복제

AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 사용하는 서버 측 암호화로 생성되는 객체가 소스 버킷에 포함될 수 있습니다. 기본적으로 Amazon S3는 이러한 객체를 복제하지 않습니다. 필요하다면 Amazon S3에 이러한 객체를 복제하도록 지시할 수 있습니다. 이렇게 하려면 먼저 <SourceSelectionCriteria> 요소를 추가하여 이 기능을 명시적으로 선택합니다. 그런 다음 AWS KMS key(대상 버킷의 AWS 리전용)를 제공하여 객체 복제본 암호화에 사용합니다. 다음 예제에서는 이러한 요소를 지정하는 방법을 보여줍니다.

```
...
<SourceSelectionCriteria>
  <SseKmsEncryptedObjects>
    <Status>Enabled</Status>
  </SseKmsEncryptedObjects>
</SourceSelectionCriteria>
<Destination>
  <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
  <EncryptionConfiguration>
    <ReplicaKmsKeyID>AWS KMS key ID to use for encrypting object replicas</
ReplicaKmsKeyID>
  </EncryptionConfiguration>
</Destination>
...
```

자세한 내용은 암호화(SSE-S3, SSE-KMS, DSSE-KMS, SSE-C)된 객체 복제 섹션을 참조하세요.

예제 복제 구성

시작하려면, 다음 예제 복제 구성을 적절히 버킷에 추가할 수 있습니다.

> **Important**
>
> 버킷에 복제 구성을 추가하려면 `iam:PassRole` 권한이 있어야 합니다. 이 권한을 사용하면 Amazon S3에 복제 권한을 부여하는 IAM 역할을 전달할 수 있습니다. 복제 구성 XML의 <Role> 요소에서 사용되는 Amazon 리소스 이름(ARN)을 제공하여 IAM 역할을 지정합니다. 자세한 내용은 IAM 사용 설명서에서 사용자에게 AWS 서비스에 역할을 전달할 권한 부여를 참조하세요.

Example 1: 한 가지 규칙으로 이루어진 복제 구성

다음 기본 복제 구성은 한 규칙을 지정합니다. 이 규칙은 Amazon S3가 맡을 수 있는 IAM 역할과 객체 복제본의 단일 대상 버킷을 지정합니다. <Status> 요소 값이 Enabled인 경우 이 규칙이 유효한 상태임을 나타냅니다.

```
<?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:::amzn-s3-demo-destination-bucket</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::account-id: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:::amzn-s3-demo-destination-bucket</Bucket>
    </Destination>

  </Rule>
</ReplicationConfiguration>
```

<Filter> 요소를 지정하는 경우 <Priority> 및 <DeleteMarkerReplication> 요소도 포함시켜야 합니다. 이 예제에서는 규칙이 하나뿐이므로, <Priority> 요소에 설정한 값은 관련이 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 2개의 지정된 객체 태그가 있는 객체에 적용됩니다. 이 예제에서는 규칙이 하나뿐이므로, <Priority> 요소에 설정한 값은 관련이 없습니다.

```
<?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>
        <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:::amzn-s3-demo-destination-bucket</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:::amzn-s3-demo-destination-bucket</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 규칙은 다음을 지정합니다.

- 각 규칙은 객체의 고유한 하위 집합에 적용되도록 서로 다른 키 접두사를 필터링합니다. 예를 들어 Amazon S3는 키 이름이 *Tax/doc1.pdf* 및 *Project/project1.txt*인 객체는 복제하지만, 키 이름이 *PersonalDoc/documentA*인 객체는 복제하지 않습니다.
- 규칙 모두 <Priority> 요소 값을 지정하지만, 규칙이 서로 다른 객체 집합에 적용되어 규칙 우선순위는 관련이 없습니다. 다음 예제는 규칙 우선 순위가 적용되면 발생하는 상황을 보여 줍니다.
- 두 번째 규칙은 객체 복제본에 대한 S3 Standard-IA 스토리지 클래스를 지정합니다. Amazon S3는 해당 객체 복제본에 대해 지정된 스토리지 클래스를 사용합니다.

```
<?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:::amzn-s3-demo-destination-bucket</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:::amzn-s3-demo-destination-bucket</Bucket>
     <StorageClass>STANDARD_IA</StorageClass>
    </Destination>
     ...
  </Rule>


</ReplicationConfiguration>
```

Example 3: 접두사가 중첩되는 두 가지 규칙으로 이루어진 복제 구성

이 구성에는 *star/* 및 *starship/*과 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. 숫자가 클수록 우선 순위가 높아집니다.

```
<ReplicationConfiguration>

  <Role>arn:aws:iam::account-id: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:::amzn-s3-demo-destination-bucket</Bucket>
    </Destination>
  </Rule>
  <Rule>
    <Status>Enabled</Status>
    <Priority>2</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>starship</Prefix>
    </Filter>
    <Destination>
      <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

Example 4: 예제 안내

예제 안내에 대해서는 라이브 복제 구성 예제을 참조하십시오.

복제 구성의 XML 구조에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 PutBucketReplication을 참조하십시오.

이전 버전과의 호환성 고려 사항

최신 버전의 복제 구성 XML 형식은 V2입니다. XML V2 복제 구성은 규칙에 대한 `<Filter>` 요소 그리고 S3 Replication Time Control(S3 RTC)을 지정하는 규칙을 포함하는 구성입니다.

복제 구성 버전을 보려면 `GetBucketReplication` API 작업을 사용할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 GetBucketReplication를 참조하십시오.

이전 버전과의 호환성을 위해 Amazon S3는 XML V1 복제 구성 형식을 계속 지원합니다. XML V1 복제 구성 형식을 사용한 경우, 이전 버전과의 호환성에 영향을 미치는 다음 문제를 고려하세요.

- 복제 구성 XML V2 형식은 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 V1 형식은 키 접두사를 기반으로 한 필터링만 지원합니다. 이 경우 다음 예제와 같이 <Rule> 요소의 하위 요소로 <Prefix> 요소를 직접 추가합니다.

```
<?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>
    <Prefix>key-prefix</Prefix>
    <Destination>
       <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
    </Destination>

  </Rule>
</ReplicationConfiguration>
```

- 객체 버전 ID를 지정하지 않고 원본 버킷에서 객체를 삭제할 경우 Amazon S3는 삭제 마커를 추가합니다. 복제 구성 XML V1 형식을 사용하는 경우 Amazon S3는 사용자 작업으로 인해 발생하는 삭제 마커만 복제합니다. 즉, Amazon S3는 사용자가 객체를 삭제하는 경우에만 삭제 마커를 복제합니다. Amazon S3에서 만료된 객체를 제거하면(수명 주기 작업의 일부로) Amazon S3는 삭제 마커를 복제하지 않습니다.

  복제 구성 XML V2 형식에서 비태그 기반 규칙에 대해 삭제 마커 복제를 활성화할 수 있습니다. 자세한 내용은 버킷 간 삭제 마커 복제 섹션을 참조하세요.

라이브 복제에 대한 권한 설정

Amazon S3에서 라이브 복제를 설정할 때 다음과 같이 필요한 권한을 획득해야 합니다.

- Amazon S3는 사용자를 대신하여 객체를 복제할 수 있는 권한이 필요합니다. AWS Identity and Access Management(IAM) 역할을 생성하고 복제 구성에서 이 역할을 지정하여 이러한 권한을 부여합니다.
- 소스 버킷과 대상 버킷을 동일한 계정에서 소유하지 않는 경우 대상 버킷 소유자는 소스 버킷 소유자에게 복제본을 저장할 권한도 부여해야 합니다.

주제

- 복제 규칙을 생성할 수 있는 권한 설정
- IAM 역할 생성
- 소스 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유할 경우 권한 부여
- 복제본 소유권 변경
- S3 배치 작업에 대한 권한 부여

복제 규칙을 생성할 수 있는 권한 설정

복제 규칙을 생성하는 데 사용할 IAM 사용자 또는 역할에는 단방향 또는 양방향 복제에 대한 복제 규칙을 생성할 권한이 필요합니다. 사용자 또는 역할에 이러한 권한이 없는 경우 복제 규칙을 생성할 수 없습니다. 자세한 내용은 IAM 사용 설명서에서 IAM 자격 증명을 참조하세요.

사용자 또는 역할에는 다음 작업이 필요합니다.

- `iam:AttachRolePolicy`
- `iam:CreatePolicy`
- `iam:CreateServiceLinkedRole`
- `iam:PassRole`
- `iam:PutRolePolicy`
- `s3:GetBucketVersioning`
- `s3:GetObjectVersionAcl`
- `s3:GetObjectVersionForReplication`
- `s3:GetReplicationConfiguration`
- `s3:PutReplicationConfiguration`

다음은 이러한 작업을 포함하는 샘플 IAM 정책입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetAccessPoint",
                "s3:GetAccountPublicAccessBlock",
```

```
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetBucketPolicyStatus",
                "s3:GetBucketPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets",
                "s3:PutReplicationConfiguration",
                "s3:GetReplicationConfiguration",
                "s3:GetBucketVersioning",
                "s3:GetObjectVersionForReplication",
                "s3:GetObjectVersionAcl",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetObjectVersion",
                "s3:GetBucketOwnershipControls",
                "s3:PutBucketOwnershipControls",
                "s3:GetObjectLegalHold",
                "s3:GetObjectRetention",
                "s3:GetBucketObjectLockConfiguration"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1-*",
                "arn:aws:s3:::amzn-s3-demo-bucket2-*/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:List*AccessPoint*",
                "s3:GetMultiRegion*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:Get*",
                "iam:CreateServiceLinkedRole",
                "iam:CreateRole",
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::*:role/service-role/s3*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "iam:List*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy",
                "iam:CreatePolicy"
              ],
            "Resource": [
                "arn:aws:iam::*:policy/service-role/s3*",
                "arn:aws:iam::*:role/service-role/s3*"
            ]
        }
    ]
}
```

IAM 역할 생성

기본적으로 Amazon S3 리소스인 버킷, 객체 및 관련 하위 리소스는 모두 비공개이며 리소스 소유자만 리소스에 액세스할 수 있습니다. Amazon S3는 소스 버킷에서 객체를 읽고 복제할 수 있는 권한이 필요합니다. IAM 역할을 생성하고 복제 구성에서 이 역할을 지정하여 이러한 권한을 부여합니다.

이 섹션에서는 IAM 역할에 연결된 신뢰 정책과 최소한의 필수 권한 정책을 설명합니다. 연습 예제가 IAM 역할을 생성하는 단계별 지침을 제공합니다. 자세한 내용은 라이브 복제 구성 예제 섹션을 참조하세요.

신뢰 정책은 IAM 역할을 맡을 수 있는 보안 주체 ID를 식별합니다. 권한 정책은 IAM 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 지정합니다.

- 다음 예제는 역할을 맡을 수 있는 AWS 서비스 보안 주체로서 Amazon S3를 식별하는 신뢰 정책을 보여줍니다.

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

```
         "Effect":"Allow",
         "Principal":{
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

- 다음 예제는 Amazon S3 및 S3 Batch Operations를 역할을 맡을 수 있는 서비스 보안 주체로 식별하는 신뢰 정책을 보여줍니다. 배치 복제 작업을 생성하는 경우 이 접근 방식을 사용하세요. 자세한 내용은 새 복제 규칙 또는 대상에 대한 배치 복제 작업 만들기 섹션을 참조하세요.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service": [
              "s3.amazonaws.com",
              "batchoperations.s3.amazonaws.com"
           ]
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

IAM 역할에 대한 자세한 내용은 IAM 사용 설명서의 IAM 역할 단원을 참조하세요.

- 다음 예제는 사용자를 대신하여 복제 태스크를 수행할 수 있는 IAM 역할 권한을 부여하는 권한 정책을 보여줍니다. Amazon S3가 이 역할을 맡으면 이 정책에 지정된 권한을 보유하게 됩니다. 이 정책에서는 *amzn-s3-demo-source-bucket*이 소스 버킷이고, *amzn-s3-demo-destination-bucket*가 대상 버킷입니다.

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

```
            "s3:GetReplicationConfiguration",
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ReplicateTags"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
      }
   ]
}
```

권한 정책은 다음 작업에 대한 권한을 부여합니다.

- `s3:GetReplicationConfiguration` 및 `s3:ListBucket` - *amzn-s3-demo-source-bucket* 버킷에 대한 이러한 작업 권한을 통해 Amazon S3는 복제 구성을 검색하고 버킷 내용을 나열할 수 있습니다. (현재 권한 모델에는 삭제 마커에 액세스할 수 있는 `s3:ListBucket` 권한이 필요합니다.)
- `s3:GetObjectVersionForReplication` 및 `s3:GetObjectVersionAcl` - 모든 객체에 대해 부여되는 이러한 작업 권한을 사용하면 Amazon S3는 특정 객체 버전 및 객체와 연결된 ACL(액세스 제어 목록)을 가져올 수 있습니다.
- `s3:ReplicateObject` 및 `s3:ReplicateDelete` - *amzn-s3-demo-destination-bucket* 버킷의 모든 객체에 대한 이러한 작업 권한을 사용하면 Amazon S3는 대상 버킷에 객체 또는 삭제

마커를 복제할 수 있습니다. 삭제 마커에 대한 자세한 내용은 삭제 작업이 복제에 미치는 영향을 참조하십시오.

> **Note**
>
> *amzn-s3-demo-destination-bucket* 버킷에서의 `s3:ReplicateObject` 작업에 대한 권한은 객체 태그 및 ACL과 같은 메타데이터의 복제도 허용합니다. 따라서 `s3:ReplicateTags` 작업에 대한 권한을 명시적으로 부여할 필요가 없습니다.

- `s3:GetObjectVersionTagging` - *amzn-s3-demo-source-bucket* 버킷의 객체에 대해 이 작업 권한을 사용하면 Amazon S3는 복제 대상 객체 태그를 읽을 수 있습니다. 객체 태그에 대한 자세한 내용은 태그를 사용하여 스토리지 분류 섹션을 참조하십시오. `s3:GetObjectVersionTagging` 권한이 없을 경우 Amazon S3는 객체를 복제하지만, 객체 태그는 복제하지 않습니다.

Amazon S3 작업의 목록은 서비스 승인 참조의 Amazon S3에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

> **Important**
>
> IAM 역할을 소유한 AWS 계정은 해당 IAM 역할에 부여된 작업을 수행할 권한이 있어야 합니다.
> 예를 들어, 소스 버킷에 다른 AWS 계정이 소유한 객체가 포함되어 있다고 가정하겠습니다. 객체 소유자는 객체 액세스 제어 목록(ACL)을 통해 IAM 역할을 소유한 AWS 계정에 필요한 권한을 명시적으로 부여해야 합니다. 그렇지 않으면 Amazon S3는 객체에 액세스할 수 없으므로 객체의 복제가 실패합니다. ACL 권한에 대한 자세한 내용은 ACL(액세스 제어 목록) 개요 단원을 참조하십시오.
> 여기에서 설명하는 권한은 최소 복제 구성과 관련됩니다. 선택적 복제 구성을 추가하려면 Amazon S3에 추가 권한을 부여해야 합니다.
>
> - 암호화된 객체를 복제하려면 필요한 AWS Key Management Service(AWS KMS) 키 권한도 부여해야 합니다. 자세한 내용은 the section called “암호화된 객체 복제” 섹션을 참조하세요.
> - 복제에 객체 잠금을 사용하려면 복제 설정에 사용하는 AWS Identity and Access Management(IAM) 역할에서 소스 S3 버킷에 두 가지 권한을 추가로 부여해야 합니다. 두 가지 추가 권한은 `s3:GetObjectRetention`과 `s3:GetObjectLegalHold`입니다. 역

할에 s3:Get* 권한 문이 있으면 해당 문이 요구 사항을 충족합니다. 자세한 내용은 the section called “S3 복제에 객체 잠금 사용” 섹션을 참조하세요.

소스 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유할 경우 권한 부여

소스 버킷과 대상 버킷을 동일한 계정에서 소유하지 않는 경우, 다음 예제와 같이 대상 버킷 소유자가 소스 버킷 소유자에게 복제 작업을 수행할 권한을 부여하는 버킷 정책도 추가해야 합니다. 이 예제 정책에서는 *amzn-s3-demo-destination-bucket*가 대상 버킷입니다.

Amazon S3 콘솔을 사용하여 이 버킷 정책을 자동으로 생성할 수도 있습니다. 자세한 내용은 소스 버킷에서 복제된 객체 수신 활성화를 참조하세요.

> **Note**
>
> 역할의 ARN 형식이 다르게 보일 수 있습니다. 콘솔을 사용하여 역할을 생성한 경우 ARN 형식은 arn:aws:iam::*account-ID*:role/service-role/*role-name*입니다. AWS CLI를 사용하여 역할을 생성한 경우 ARN 형식은 arn:aws:iam::*account-ID*:role/*role-name*입니다. 자세한 내용은 IAM 사용 설명서에서 IAM 역할을 참조하세요.

```
{
   "Version":"2012-10-17",
   "Id":"PolicyForDestinationBucket",
   "Statement":[
      {
         "Sid":"Permissions on objects",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::source-bucket-account-ID:role/service-role/source-
account-IAM-role"
         },
         "Action":[
            "s3:ReplicateDelete",
            "s3:ReplicateObject"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
      },
      {
         "Sid":"Permissions on bucket",
         "Effect":"Allow",
```

```
         "Principal":{
            "AWS":"arn:aws:iam::source-bucket-account-ID:role/service-role/source-
account-IAM-role"
         },
         "Action": [
            "s3:List*",
            "s3:GetBucketVersioning",
            "s3:PutBucketVersioning"
         ],
         "Resource":"arn:aws:s3::amzn-s3-demo-destination-bucket"
      }
   ]
}
```

문제 해결 예는 서로 다른 계정의 버킷에 대한 복제 구성을(를) 참조하세요.

원본 버킷에 있는 객체에 태그가 지정된 경우, 다음에 주의하세요.

- 원본 버킷 소유자가 객체 태그를 복사할 수 있도록 `s3:GetObjectVersionTagging` 및 `s3:ReplicateTags` 작업 권한을 (IAM 역할을 통해) Amazon S3에 부여하는 경우, Amazon S3는 해당 객체와 함께 태그를 복제합니다. IAM 역할에 대한 상세 정보는 IAM 역할 생성 섹션을 참조하세요.
- 대상 버킷 소유자가 태그 복제를 원하지 않는 경우, 소유자는 다음의 명령문을 대상 버킷 정책에 추가해 `s3:ReplicateTags` 작업에 대한 권한을 명시적으로 거부할 수 있습니다. 이 정책에서는 *amzn-s3-demo-destination-bucket*가 대상 버킷입니다.

```
...
   "Statement":[
      {
         "Effect":"Deny",
         "Principal":{
            "AWS":"arn:aws:iam::source-bucket-account-id:role/service-role/source-
account-IAM-role"
         },
         "Action":"s3:ReplicateTags",
         "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
      }
   ]
...
```

> **Note**
>
> - 암호화된 객체를 복제하려면 필요한 AWS Key Management Service(AWS KMS) 키 권한도 부여해야 합니다. 자세한 내용은 the section called “암호화된 객체 복제” 섹션을 참조하세요.
> - 복제에 객체 잠금을 사용하려면 복제 설정에 사용하는 AWS Identity and Access Management(IAM) 역할에서 소스 S3 버킷에 두 가지 권한을 추가로 부여해야 합니다. 두 가지 추가 권한은 `s3:GetObjectRetention`과 `s3:GetObjectLegalHold`입니다. 역할에 `s3:Get*` 권한 문이 있으면 해당 문이 요구 사항을 충족합니다. 자세한 내용은 the section called “S3 복제에 객체 잠금 사용” 섹션을 참조하세요.

소스 버킷에서 복제된 객체 수신 활성화

이전 정책을 대상 버킷에 수동으로 추가하는 대신, Amazon S3 콘솔을 통해 소스 버킷에서 복제된 객체 수신을 활성화하는 데 필요한 정책을 신속하게 생성할 수 있습니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷(Buckets) 목록에서 대상 버킷으로 사용할 버킷을 선택합니다.
4. 관리(Management) 탭을 선택한 다음 아래로 스크롤하여 복제 규칙(Replication rules)으로 이동합니다.
5. 작업(Actions)으로 복제된 객체 수신(Receive replicated objects)을 선택합니다.

   프롬프트에 따라 소스 버킷 계정의 AWS 계정 ID를 입력하고 정책 생성을 선택합니다. 콘솔은 Amazon S3 버킷 정책과 KMS 키 정책을 생성합니다.
6. 이 정책을 기존 버킷 정책에 추가하려면 설정 적용(Apply settings)을 선택하거나 복사(Copy)를 선택하여 변경 사항을 수동으로 복사합니다.
7. (선택 사항) AWS Key Management Service 콘솔에서 원하는 KMS 키 정책에 AWS KMS 정책을 복사합니다.

복제본 소유권 변경

서로 다른 AWS 계정이 소스 버킷과 대상 버킷을 소유한 경우, Amazon S3에 복제본 소유권을 대상 버킷을 소유한 AWS 계정으로 변경하도록 지시할 수 있습니다. 소유자 재정의에 대한 자세한 내용은 복제본 소유자 변경 섹션을 참조하세요.

S3 배치 작업에 대한 권한 부여

S3 배치 복제은 다음 객체를 복제하는 방법을 제공합니다.

- 복제 구성이 적용되기 전에 존재했던 객체
- 이전에 복제된 객체
- 복제에 실패한 객체

새 복제 구성에서 첫 번째 규칙을 만들거나 Amazon S3 콘솔을 통해 기존 구성에 새 대상을 추가할 때 일회성 배치 복제 작업을 만들 수 있습니다. Batch Operations 작업을 만들어 기존 복제 구성에 대한 배치 복제를 시작할 수도 있습니다.

배치 복제 IAM 역할 및 정책 예제는 S3 배치 복제를 위한 IAM 역할 구성 섹션을 참조하세요.

라이브 복제 구성 예제

다음 예제에서는 일반적인 사용 사례에 맞게 라이브 복제를 구성하는 방법을 보여주는 단계별 실습 과정을 제공합니다.

> **Note**
>
> 라이브 복제는 동일 리전 복제(SRR) 및 교차 리전 복제(CRR)를 의미합니다. 라이브 복제는 복제를 설정하기 전에 버킷에 있던 객체를 자동으로 복제하지 않습니다. 복제를 설정하기 전에 존재한 객체를 복제하려면 온디맨드 복제를 사용합니다. 버킷을 동기화하고 기존 객체를 온디맨드로 복제하려면 기존 객체 복제 섹션을 참조하세요.

이 예제에서는 Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 및 AWS SDK(AWS SDK for Java 및 AWS SDK for .NET 예제 표시)를 사용하여 복제 구성을 만드는 방법을 보여 줍니다.

AWS CLI 설치 및 구성에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 다음 주제들을 참조하세요.

- AWS CLI 시작
- AWS CLI 구성 - 하나 이상의 프로파일을 설정해야 합니다. 교차 계정 시나리오에 대해 살펴보려면 두 개의 프로필을 설정해야 합니다.

AWS SDK에 대한 자세한 내용은 AWS SDK for Java 및 AWS SDK for .NET 섹션을 참조하세요.

> **Tip**
>
> 라이브 복제를 사용하여 데이터를 복제하는 방법을 보여주는 단계별 자습서는 자습서: S3 복제를 사용하여 AWS 리전 내 및 사이에 데이터 복제를 참조하세요.

주제

- 동일한 계정에서 버킷 복제 구성
- 서로 다른 계정의 버킷에 대한 복제 구성
- S3 Replication Time Control을 사용하여 규정 준수 요구 사항 충족
- 암호화(SSE-S3, SSE-KMS, DSSE-KMS, SSE-C)된 객체 복제
- 복제본 수정 동기화를 사용하여 메타데이터 변경 복제
- 버킷 간 삭제 마커 복제

동일한 계정에서 버킷 복제 구성

라이브 복제는 동일한 리전 또는 서로 다른 AWS 리전의 범용 버킷 간에 객체를 비동기식으로 자동 복사하는 것을 말합니다. 라이브 복제는 새로 생성된 객체 및 객체 업데이트를 소스 버킷에서 하나 이상의 대상 버킷으로 복사합니다. 자세한 내용은 리전 내 및 리전 간 객체 복제 섹션을 참조하세요.

복제를 구성할 때는 소스 버킷에 복제 규칙을 추가합니다. 복제 규칙은 복제할 원본 버킷 객체와 복제된 객체가 저장된 대상 버킷을 정의합니다. 특정 키 이름 접두사, 하나 이상의 객체 태그 또는 이 두 가지를 모두 포함하는 버킷 또는 객체 하위 집합에 있는 모든 객체를 복제하는 규칙을 작성할 수 있습니다. 대상 버킷은 소스 버킷과 동일한 AWS 계정에 있거나 다른 계정에 존재할 수 있습니다.

삭제할 객체 버전 ID를 지정하는 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.

버킷에 복제 규칙을 추가하면 이 규칙이 기본적으로 사용 설정되므로 이 규칙을 저장하면 그 즉시 작업이 시작됩니다.

이 예제에서는 동일한 AWS 계정에서 소유한 소스 버킷과 대상 버킷의 라이브 복제를 설정합니다. Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), AWS SDK for Java 및 AWS SDK for .NET을 사용하는 예제가 제공됩니다.

사전 조건

다음 절차를 사용하기 전에 소스 버킷과 대상 버킷을 동일한 계정에서 소유하는지 아니면 다른 계정에서 소유하는지에 따라 복제에 필요한 권한을 설정했는지 확인합니다. 자세한 내용은 the section called "권한 설정" 섹션을 참조하세요.

> **Note**
>
> - 암호화된 객체를 복제하려면 필요한 AWS Key Management Service(AWS KMS) 키 권한도 부여해야 합니다. 자세한 내용은 the section called "암호화된 객체 복제" 섹션을 참조하세요.
> - 복제에 객체 잠금을 사용하려면 복제 설정에 사용하는 AWS Identity and Access Management(IAM) 역할에서 소스 S3 버킷에 두 가지 권한을 추가로 부여해야 합니다. 두 가지 추가 권한은 `s3:GetObjectRetention`과 `s3:GetObjectLegalHold`입니다. 역할에 `s3:Get*` 권한 문이 있으면 해당 문이 요구 사항을 충족합니다. 자세한 내용은 the section called "S3 복제에 객체 잠금 사용" 섹션을 참조하세요.

S3 콘솔 사용

대상 버킷이 소스 버킷과 동일한 AWS 계정에 있는 경우 복제 규칙을 구성하려면 다음 단계를 따릅니다.

대상 버킷이 원본 버킷과 다른 계정에 있는 경우, 원본 버킷 계정의 소유자에게 대상 버킷의 객체를 복제할 수 있는 권한을 부여하려면 대상 버킷에 하나의 버킷 정책을 추가해야 합니다. 자세한 내용은 소스 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유할 경우 권한 부여 섹션을 참조하세요.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 원하는 버킷의 이름을 선택합니다.
4. 관리 탭을 선택하고 복제 규칙까지 아래로 스크롤한 다음 복제 규칙 생성을 선택합니다.
5. 복제 규칙 구성 섹션의 복제 규칙 이름에 나중에 규칙을 쉽게 식별할 수 있는 규칙 이름을 입력합니다. 이름은 필수 항목이며 버킷 내에서 고유해야 합니다.
6. 상태 아래의 활성이 기본적으로 선택됩니다. 사용 설정된 규칙은 저장하는 즉시 적용되기 시작합니다. 나중에 규칙을 활성화하고 싶다면 비활성화됨을 선택하세요.

7. 버킷에 기존 복제 규칙이 있는 경우 규칙에 우선 순위를 설정하라는 메시지가 표시됩니다. 여러 규칙의 범위에 포함된 객체로 인해 발생하는 충돌을 방지하기 위해 규칙의 우선 순위를 설정해야 합니다. 중첩 규칙의 경우, Amazon S3은 규칙 우선 순위를 사용하여 어느 규칙을 적용할지 결정합니다. 숫자가 클수록 우선 순위가 높아집니다. 규칙 우선 순위에 대한 자세한 내용은 복제 구성 파일 요소 단원을 참조하세요.
8. 소스 버킷에는 복제 소스를 설정하기 위한 다음과 같은 옵션이 있습니다.

   - 전체 버킷을 복제하려면 Apply to all objects in the bucket(버킷의 모든 객체에 적용)를 선택합니다.
   - 접두사가 같은 모든 객체를 복제하려면 하나 이상의 필터를 사용하여 이 규칙의 범위 제한(Limit the scope of this rule using one or more filters)을 선택합니다. 이렇게 하면 이름이 지정한 접두사(예: `pictures`)로 시작하는 모든 객체에 대한 복제가 제한됩니다. 접두사 상자에 접두사를 입력합니다.

     > **Note**
     >
     > 어떤 폴더의 이름에 해당하는 접두사를 입력할 경우, /(슬래시)를 마지막 문자(예: `pictures/`)로 사용해야 합니다.

   - 하나 이상의 객체 태그가 있는 모든 객체를 복제하려면 태그 추가를 선택하고 상자에 키 값 페어를 입력합니다. 절차를 반복하여 다른 태그를 추가합니다. 접두사와 태그를 결합할 수도 있습니다. 객체 태그에 대한 자세한 내용은 태그를 사용하여 스토리지 분류 섹션을 참조하십시오.

   새로운 복제 구성 XML 스키마는 접두사 및 태그 필터링과 규칙 우선 순위 지정을 지원합니다. 새 스키마에 대한 자세한 내용은 이전 버전과의 호환성 고려 사항 단원을 참조하세요. 사용자 인터페이스 뒤에서 작동하는 Amazon S3 API와 함께 사용되는 XML에 대한 자세한 내용은 복제 구성 파일 요소 섹션을 참조하세요. 새 스키마는 복제 구성 XML V2로 기술됩니다.
9. 대상에서 Amazon S3이 객체를 복제하게 할 버킷을 선택합니다.

   > **Note**
   >
   > 대상 버킷의 수는 지정된 파티션의 AWS 리전 수로 제한됩니다. 파티션은 리전 그룹입니다. AWS에는 `aws`(표준 리전), `aws-cn`(중국 리전) 및 `aws-us-gov`(AWS GovCloud (US) 리전), 이렇게 세 가지 파티션이 있습니다. 대상 버킷 할당량 증대를 요청하려면 서비스 할당량을 사용합니다.

- 계정의 버킷으로 복제하려면 이 계정의 버킷 선택을 선택하고 대상 버킷 이름을 입력하거나 찾아봅니다.
- 다른 AWS 계정의 버킷으로 복제하려면 다른 계정의 버킷 지정을 선택하고 대상 버킷 계정 ID와 버킷 이름을 입력합니다.

  대상이 원본 버킷과 다른 계정에 있는 경우, 원본 버킷 계정의 소유자에게 객체를 복제할 수 있는 권한을 부여하려면 대상 버킷에 하나의 버킷 정책을 추가해야 합니다. 자세한 내용은 소스 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유할 경우 권한 부여 섹션을 참조하세요.

  선택적으로, 대상 버킷에서 새 객체의 소유권을 표준화하고 싶다면 객체 소유권을 대상 버킷 소유자로 변경을 선택합니다. 이 옵션에 대한 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

> **Note**
>
> 대상 버킷에서 버전 관리가 사용 설정되어 있지 않을 경우, 버전 관리 사용 설정(Enable versioning) 버튼이 포함된 경고가 나타납니다. 이 버튼을 선택하면 버킷의 버전 관리가 사용 설정됩니다.

10. Amazon S3가 사용자 대신 객체를 복제하기 위해 수임할 수 있는 AWS Identity and Access Management(IAM) 역할을 설정합니다.

    IAM 역할을 설정하려면 IAM 역할 섹션의 IAM 역할 드롭다운 목록에서 다음 중 하나를 선택합니다.

    - 새 역할 생성(Create new role)을 선택하여 Amazon S3에서 자동으로 새 IAM 역할을 생성하도록 하는 것이 좋습니다. 규칙을 저장하면 선택한 원본 및 대상 버킷과 일치하는 IAM 역할에 대해 새 정책이 생성됩니다.
    - 기존 IAM 역할을 사용하도록 선택할 수 있습니다. 이 경우 복제에 필요한 권한을 Amazon S3에 부여하는 역할을 선택해야 합니다. 이 역할이 복제 규칙을 따를 수 있는 충분한 권한을 Amazon S3에 부여하지 않으면 복제가 실패합니다.

> **Important**
>
> 버킷에 복제 규칙을 추가하려면 Amazon S3에 복제 권한을 부여하는 IAM 역할을 전달할 수 있는 `iam:PassRole` 권한이 있어야 합니다. 자세한 내용은 IAM 사용 설명서에서 AWS 서비스에 역할을 전달하도록 사용자 권한 부여를 참조하세요.

11. AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 통한 서버 측 암호화로 암호화된 소스 버킷의 객체를 복제하려면 암호화에서 AWS KMS로 암호화된 객체 복제를 선택합니다. 대상 객체를 암호화하기 위한 AWS KMS 키에는 복제에 사용할 수 있는 소스 키가 있습니다. 기본값으로 모든 소스 KMS 키가 포함됩니다. KMS 키 선택 범위를 좁히려면 별칭이나 키 ID를 선택하면 됩니다.

    선택하지 않은 AWS KMS keys로 암호화된 객체는 복제되지 않습니다. KMS 키 또는 KMS 키 그룹이 자동으로 선택되지만, 원한다면 KMS 키를 선택할 수 있습니다. 복제와 함께 AWS KMS을(를) 사용하는 방법에 대한 자세한 내용은 암호화(SSE-S3, SSE-KMS, DSSE-KMS, SSE-C)된 객체 복제 단원을 참조하세요.

    > **Important**
    >
    > AWS KMS을(를) 사용하여 암호화된 객체를 복제하는 경우 AWS KMS 요청 빈도는 소스 리전에서 두 배가 되고 대상 리전에서는 같은 양만큼 늘어납니다. AWS KMS에 대해 늘어난 호출 빈도는 복제 대상 리전에 대해 정의한 KMS 키를 사용하여 데이터가 다시 암호화되는 방식 때문입니다. AWS KMS에는 리전당 호출 계정에 따른 요청 속도 할당량이 있습니다. 할당량 기본값에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 AWS KMS 할당량 - 초당 요청: 달라짐을 참조하세요.
    >
    > 복제 중 현재 Amazon S3 PUT 객체 요청 속도가 계정에 대한 기본 AWS KMS 속도 할당량의 절반보다 큰 경우 AWS KMS 요청 속도 제한 증가를 요청하는 것이 좋습니다. 증가를 요청하려면 문의처의 지원 Center에서 사례를 생성합니다. 예를 들어 현재 PUT 객체 요청 빈도가 초당 1,000개이고 AWS KMS를 사용하여 객체를 암호화한다고 가정합니다. 이 경우 AWS KMS에서 제한이 발생하지 않도록 지원에 소스 리전과 대상 리전 둘 다에서 AWS KMS 속도 제한을 초당 2,500개 요청으로 올려달라고 요청하는 것이 좋습니다.
    >
    > 소스 버킷에서 PUT 객체 요청 빈도를 확인하려면 Amazon S3에 대한 Amazon CloudWatch 요청 지표에서 `PutRequests`를 확인합니다. CloudWatch 지표를 확인하는 방법에 대한 자세한 내용은 S3 콘솔 사용 섹션을 참조하세요.

AWS KMS로 암호화된 객체를 복제하도록 선택한 경우 다음을 수행합니다.

- 대상 객체를 암호화하기 위한 AWS KMS key에서 다음 방법 중 하나로 KMS 키를 지정합니다.

  - 사용 가능한 KMS 키 목록에서 AWS KMS keys 중에서 선택을 선택하고 사용 가능한 키 목록에서 KMS 키를 선택합니다.

    AWS 관리형 키(aws/s3)와 고객 관리형 키가 모두 목록에 표시됩니다. 고객 관리형 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 키 및 AWS 키를 참조하세요.
  - KMS 키 Amazon 리소스 이름(ARN)을 입력하려면 AWS KMS key ARN 입력을 선택하고 나타나는 필드에 KMS 키 ARN을 입력합니다. 이렇게 하면 대상 버킷의 복제본이 암호화됩니다. IAM 콘솔의 암호화 키 아래에서 KMS 키에 대한 ARN을 찾을 수 있습니다.
  - AWS KMS 콘솔에서 고객 관리형 키를 생성하려면 KMS 키 생성을 선택합니다.

    AWS KMS key 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 키 생성을 참조하십시오.

  > **Important**
  >
  > 버킷과 동일한 AWS 리전에서 사용되는 KMS 키만 사용할 수 있습니다. KMS 키에서 선택을 선택하면 S3 콘솔에는 KMS 키를 리전당 100개씩만 나열합니다. 동일한 리전에 100개 이상의 KMS 키가 있는 경우, S3 콘솔에서 처음 100개의 KMS 키만 볼 수 있습니다. 콘솔에 나열되지 않은 KMS 키를 사용하려면 AWS KMS key ARN 입력을 선택하고 KMS 키 ARN을 입력합니다.
  > Amazon S3에서 서버 측 암호화에 AWS KMS key을 사용하는 경우 대칭 암호화 KMS 키를 선택해야 합니다. Amazon S3는 대칭 암호화 KMS 키만 지원하며 비대칭 KMS 키는 지원하지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 대칭 및 비대칭 KMS 키 식별을 참조하십시오.

AWS KMS key 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 키 생성을 참조하십시오. Amazon S3에서 AWS KMS을(를) 사용하는 방법에 대한 자세한 내용은 AWS KMS 키를 사용한 서버 측 암호화(SSE-KMS) 사용 단원을 참조하십시오.

12. 대상의 특정 스토리지 클래스로 데이터를 복제하려면 대상 스토리지 클래스에서 복제된 객체의 스토리지 클래스 변경을 선택합니다. 그런 다음, 대상의 복제된 객체에 사용할 스토리지 클래스를 선택합니다. 이 옵션을 선택하지 않을 경우, 복제된 객체의 스토리지 클래스는 원본 객체와 동일한 클래스에 해당됩니다.
13. 추가 복제 옵션을 설정할 때 다음과 같은 추가 옵션이 있습니다.

    - 복제 구성에서 S3 Replication Time Control (S3 RTC)를 활성화하려면 복제 시간 제어(RTC)를 선택합니다. 이 옵션에 대한 자세한 내용은 S3 Replication Time Control을 사용하여 규정 준수 요구 사항 충족 섹션을 참조하세요.
    - 복제 구성에서 S3 복제 지표를 활성화하려면 Replication metrics and events(복제 지표 및 이벤트)를 선택합니다. 자세한 내용은 지표, 이벤트 알림 및 상태를 통한 복제 모니터링 섹션을 참조하세요.
    - 복제 구성에서 삭제 마커 복제를 사용 설정하려면 삭제 마커 복제(Delete marker replication)를 선택합니다. 자세한 내용은 버킷 간 삭제 마커 복제 섹션을 참조하세요.
    - 복제 구성에서 Amazon S3 복제본 수정 동기화를 사용 설정하려면 복제본 수정 동기화(Replica modification sync)를 선택합니다. 자세한 내용은 복제본 수정 동기화를 사용하여 메타데이터 변경 복제 섹션을 참조하세요.

    > **Note**
    >
    > S3 RTC 또는 S3 복제 지표를 사용하면 추가 요금이 적용됩니다.

14. 완료하려면 저장(Save)을 선택합니다.
15. 규칙을 저장한 후 규칙을 선택하고 규칙 편집(Edit rule)을 선택하여 규칙을 편집, 사용 설정, 사용 중지 또는 삭제할 수 있습니다.

AWS CLI 사용

소스 버킷과 대상 버킷을 동일한 AWS 계정에서 소유한 경우 AWS CLI를 사용하여 복제를 설정하려면 다음을 수행합니다.

- 소스 및 대상 버킷을 만듭니다.
- 버킷의 버전 관리를 활성화합니다.
- Amazon S3에 객체 복제 권한을 제공하는 AWS Identity and Access Management(IAM) 역할을 만듭니다.

- 소스 버킷에 복제 구성을 추가합니다.

설정을 확인하려면 테스트합니다.

소스 버킷과 대상 버킷을 동일한 AWS 계정에서 소유한 경우 복제를 설정하려면

1. AWS CLI의 자격 증명 프로필을 설정합니다. 이 예시에서는 `acctA`라는 프로필 이름을 사용합니다. 자격 증명 프로파일 설정 및 명명된 프로파일 사용에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 구성 및 자격 증명 파일 설정을 참조하세요.

   > **Important**
   >
   > 이 예제에 사용하는 프로파일에 필요한 권한이 있어야 합니다. 예를 들어 복제 구성에서 Amazon S3가 맡을 수 있는 IAM 역할을 지정합니다. 사용하는 프로파일에 `iam:PassRole` 권한이 있는 경우에만 이 작업을 수행할 수 있습니다. 자세한 내용은 IAM 사용 설명서에서 사용자에게 AWS 서비스에 역할을 전달할 권한 부여를 참조하세요. 관리자 자격 증명을 사용하여 명명된 프로파일을 생성할 경우 모든 작업을 수행할 수 있습니다.

2. 다음 AWS CLI 명령을 사용하여 소스 버킷을 만들고 해당 버킷에서 버전 관리를 활성화합니다. 이 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체합니다.

   다음 `create-bucket` 명령은 미국 동부(버지니아 북부)(`us-east-1`) 리전에 이름이 *amzn-s3-demo-source-bucket*인 소스 버킷을 생성합니다.

   ```
   aws s3api create-bucket \
   --bucket amzn-s3-demo-source-bucket \
   --region us-east-1 \
   --profile acctA
   ```

   다음 `put-bucket-versioning` 명령은 *amzn-s3-demo-source-bucket* 버킷의 S3 버전 관리를 활성화합니다.

   ```
   aws s3api put-bucket-versioning \
   --bucket amzn-s3-demo-source-bucket \
   --versioning-configuration Status=Enabled \
   --profile acctA
   ```

3. 다음 AWS CLI 명령을 사용하여 대상 버킷을 만들고 해당 버킷에서 버전 관리를 활성화합니다. 이 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체합니다.

> **Note**
>
> 소스 버킷과 대상 버킷이 모두 동일한 AWS 계정에 있을 때 복제 구성을 설정하려면 소스 버킷과 대상 버킷에 대해 동일한 프로파일을 사용합니다. 이 예제에서는 `acctA`를 사용합니다.
>
> 버킷을 서로 다른 AWS 계정에서 소유하는 경우 복제 구성을 테스트하려면 각 계정에 대해 서로 다른 프로파일을 지정합니다. 예를 들어, 대상 버킷에 대해 `acctB` 프로파일을 사용합니다.

다음 `create-bucket` 명령은 미국 서부(오리건)(`us-west-2`) 리전에 이름이 *amzn-s3-demo-destination-bucket*인 대상 버킷을 생성합니다.

```
aws s3api create-bucket \
--bucket amzn-s3-demo-destination-bucket \
--region us-west-2 \
--create-bucket-configuration LocationConstraint=us-west-2 \
--profile acctA
```

다음 `put-bucket-versioning` 명령은 *amzn-s3-demo-destination-bucket* 버킷의 S3 버전 관리를 활성화합니다.

```
aws s3api put-bucket-versioning \
--bucket amzn-s3-demo-destination-bucket \
--versioning-configuration Status=Enabled \
--profile acctA
```

4. IAM 역할 생성. 나중에 소스 버킷에 추가하는 복제 구성에서 이 역할을 지정합니다. Amazon S3는 사용자를 대신하여 객체를 복제하기 위해 이 역할을 맡습니다. IAM 역할은 다음의 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 역할을 맡을 권한을 Amazon S3 서비스 보안 주체에 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",
            "s3:GetObjectVersionTagging"
         ],
         "Resource":[
```

```
            "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket",
            "s3:GetReplicationConfiguration"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ReplicateTags"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
      }
   ]
}
```

> **Note**
>
> - 암호화된 객체를 복제하려면 필요한 AWS Key Management Service(AWS KMS) 키 권한도 부여해야 합니다. 자세한 내용은 the section called “암호화된 객체 복제” 섹션을 참조하세요.
> - 복제에 객체 잠금을 사용하려면 복제 설정에 사용하는 AWS Identity and Access Management(IAM) 역할에서 소스 S3 버킷에 두 가지 권한을 추가로 부여해야 합니다. 두 가지 추가 권한은 `s3:GetObjectRetention`과 `s3:GetObjectLegalHold`입니다. 역할에 `s3:Get*` 권한 문이 있으면 해당 문이 요구 사항을 충족합니다. 자세한 내용은 the section called “S3 복제에 객체 잠금 사용” 섹션을 참조하세요.

ii. 다음 명령을 실행하여 정책을 생성하고 이를 역할에 연결합니다. *`user input placeholders`*를 사용자의 정보로 대체합니다.

```
$ aws iam put-role-policy \
--role-name replicationRole \
--policy-document file://s3-role-permissions-policy.json \
--policy-name replicationRolePolicy \
--profile acctA
```

5. 소스 버킷에 복제 구성을 추가합니다.

   a. Amazon S3 API에서는 복제 구성을 XML로 지정해야 하지만, AWS CLI에서는 복제 구성을 JSON으로 지정해야 합니다. 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:::amzn-s3-demo-destination-bucket"
         }
       }
     ]
   }
   ```

   b. *amzn-s3-demo-destination-bucket* 및 *IAM-role-ARN* 값을 자체 정보로 대체하여 JSON을 업데이트합니다. 변경 사항을 저장합니다.

   c. 다음 `put-bucket-replication` 명령을 실행하여 소스 버킷에 복제 구성을 추가합니다. 반드시 소스 버킷 이름을 제공해야 합니다.

   ```
   $ aws s3api put-bucket-replication \
   --replication-configuration file://replication.json \
   --bucket amzn-s3-demo-source-bucket \
   --profile acctA
   ```

   복제 구성을 검색하려면 `get-bucket-replication` 명령을 사용합니다.

```
$ aws s3api get-bucket-replication \
--bucket amzn-s3-demo-source-bucket \
--profile acctA
```

6. 다음 단계를 수행하여 Amazon S3 콘솔에서 설정을 테스트합니다.

   a. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.

   b. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다. 범용 버킷 목록에서 소스 버킷을 선택합니다.

   c. 소스 버킷에서 *Tax* 폴더를 만듭니다.

   d. 소스 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 가져오기 단원을 참조하십시오.

   ## 버킷에서 다음을 확인합니다.

   - Amazon S3가 객체를 복제함.
   - 객체를 복제했습니다. 객체의 속성 탭에서 아래로 스크롤하여 객체 관리 개요 섹션을 찾습니다. 관리 구성에서 복제 상태 아래의 값을 확인합니다. 이 값이 REPLICA로 설정되어 있는지 확인합니다.
   - 소스 버킷 계정이 복제본을 소유합니다. 객체의 권한 탭에서 객체 소유권을 확인할 수 있습니다.

     소스 버킷과 대상 버킷을 서로 다른 계정에서 소유하고 있는 경우, Amazon S3에 복제본 소유권을 대상 계정으로 변경하도록 지시하는 선택적 구성을 추가할 수 있습니다. 문제 해결 예는 복제본 소유자 변경 방법을(를) 참조하세요.

AWS SDK 사용

다음 코드 예제를 사용하여 AWS SDK for Java 및 AWS SDK for .NET를 각각 사용하여 버킷에 복제 구성을 추가합니다.

> **Note**
>
> - 암호화된 객체를 복제하려면 필요한 AWS Key Management Service(AWS KMS) 키 권한도 부여해야 합니다. 자세한 내용은 the section called “암호화된 객체 복제” 섹션을 참조하세요.
> - 복제에 객체 잠금을 사용하려면 복제 설정에 사용하는 AWS Identity and Access Management(IAM) 역할에서 소스 S3 버킷에 두 가지 권한을 추가로 부여해야 합니다. 두 가지 추가 권한은 `s3:GetObjectRetention`과 `s3:GetObjectLegalHold`입니다. 역할에 `s3:Get*` 권한 문이 있으면 해당 문이 요구 사항을 충족합니다. 자세한 내용은 the section called “S3 복제에 객체 잠금 사용” 섹션을 참조하세요.

Java

다음 예에서는 버킷에 복제 구성을 추가한 후 해당 구성을 검색 및 확인합니다. 실제 예제를 만들고 테스트하는 방법에 대한 자세한 내용은 AWS SDK for Java 개발자 안내서의 시작하기를 참조하세요.

```
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:%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 AWS SDK for .NET 코드의 예입니다. 이 코드를 사용하려면 버킷 이름과 IAM 역할의 Amazon 리소스 이름(ARN)을 제공합니다. 코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

```
using Amazon;
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);
            }
        }
    }
}
```

서로 다른 계정의 버킷에 대한 복제 구성

라이브 복제는 동일한 리전 또는 서로 다른 AWS 리전의 버킷 간에 객체를 비동기식으로 자동 복사하는 것을 말합니다. 라이브 복제는 새로 생성된 객체 및 객체 업데이트를 소스 버킷에서 하나 이상의 대상 버킷으로 복사합니다. 자세한 내용은 리전 내 및 리전 간 객체 복제 섹션을 참조하세요.

복제를 구성할 때는 소스 버킷에 복제 규칙을 추가합니다. 복제 규칙은 복제할 원본 버킷 객체와 복제된 객체가 저장된 대상 버킷을 정의합니다. 특정 키 이름 접두사, 하나 이상의 객체 태그 또는 이 두 가지를 모두 포함하는 버킷 또는 객체 하위 집합에 있는 모든 객체를 복제하는 규칙을 작성할 수 있습니다. 대상 버킷은 소스 버킷과 동일한 AWS 계정에 있거나 다른 계정에 존재할 수 있습니다.

삭제할 객체 버전 ID를 지정하는 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.

버킷에 복제 규칙을 추가하면 이 규칙이 기본적으로 사용 설정되므로 이 규칙을 저장하면 그 즉시 작업이 시작됩니다.

서로 다른 AWS 계정에서 소스 버킷과 대상 버킷을 소유한 경우의 라이브 복제 설정은 동일한 계정이 두 버킷을 모두 소유한 경우의 복제 설정과 비슷합니다. 하지만 교차 계정 시나리오에서 복제를 구성할 때는 몇 가지 차이점이 있습니다.

- 대상 버킷 소유자는 소스 버킷 소유자에게 대상 버킷 정책에서 객체를 복제할 수 있는 권한을 부여해야 합니다.
- 교차 계정 시나리오에서 AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 통한 서버 측 암호화로 암호화된 객체를 복제하는 경우 KMS 키 소유자가 소스 버킷 소유자에게 KMS 키를 사용할 수 있는 권한을 부여해야 합니다. 자세한 내용은 교차 계정 시나리오를 위한 추가 권한 부여 섹션을 참조하세요.
- 기본적으로 복제된 객체는 소스 버킷 소유자가 소유합니다. 교차 계정 시나리오에서는 복제된 객체의 소유권을 대상 버킷의 소유자로 변경하도록 복제를 구성할 수 있습니다. 자세한 내용은 복제본 소유자 변경 섹션을 참조하세요.

소스 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유한 경우 복제를 구성하려면

1. 이 예제에서는 소스 버킷과 대상 버킷을 서로 다른 두 AWS 계정에서 만듭니다. AWS CLI에 대해 2개의 자격 증명 프로파일을 설정해야 합니다. 이 예제에서는 해당 프로파일 이름에 `acctA` 및 `acctB`를 사용합니다. 자격 증명 프로파일 설정 및 명명된 프로파일 사용에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 구성 및 자격 증명 파일 설정을 참조하세요.
2. 동일한 계정에서 버킷 복제 구성의 단계별 지침을 따르되 다음 사항을 변경합니다.
   - 소스 버킷 활동(예: 소스 버킷 생성, 버전 관리 활성화, IAM 역할 생성)과 관련된 모든 AWS CLI 명령은 `acctA` 프로파일을 사용합니다. `acctB` 프로파일을 사용하여 대상 버킷을 만듭니다.
   - IAM 역할에 대한 권한 정책에 이 예제에서 만든 소스 및 대상 버킷이 지정되어 있는지 확인합니다.
3. 콘솔에서 ## 버킷에 다음 버킷 정책을 추가해 ## 버킷 소유자가 객체를 복제하도록 허용합니다. 지침은 Amazon S3 콘솔을 사용하여 버킷 정책 추가 섹션을 참조하세요. 소스 버킷 소유자의 AWS 계정 ID, IAM 역할 이름 및 대상 버킷 이름을 제공하여 정책을 편집해야 합니다.

> **Note**
>
> 다음 예시를 사용하려면 *user input placeholders*를 실제 정보로 대체하십시오. *amzn-s3-demo-destination-bucket*를 대상 버킷 이름으로 바꿉니다. IAM Amazon 리소스 이름(ARN)의 *source-bucket-account-ID*:role/service-role/*source-account-IAM-role*을 이 복제 구성에 사용 중인 IAM 역할로 바꿉니다.
> IAM 서비스 역할을 수동으로 만든 경우 다음 정책 예제와 같이 IAM ARN의 역할 경로를 role/service-role/로 설정합니다. 자세한 내용을 알아보려면 IAM 사용 설명서의 IAM ARN을 참조하세요.

```
{
   "Version":"2012-10-17",
   "Id":"",
   "Statement":[
      {
         "Sid":"Set-permissions-for-objects",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::source-bucket-account-ID:role/service-role/source-
account-IAM-role"
         },
         "Action":["s3:ReplicateObject", "s3:ReplicateDelete"],
         "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
      },
      {
         "Sid":"Set permissions on bucket",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::source-bucket-account-ID:role/service-role/source-
account-IAM-role"
         },
         "Action":["s3:GetBucketVersioning", "s3:PutBucketVersioning"],
         "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket"
      }
   ]
}
```

4. (선택 사항) SSE-KMS로 암호화된 객체를 복제하는 경우 KMS 키 소유자가 소스 버킷 소유자에게 KMS 키를 사용할 수 있는 권한을 부여해야 합니다. 자세한 내용은 교차 계정 시나리오를 위한 추가 권한 부여 섹션을 참조하세요.
5. (선택 사항) 복제에서는 기본적으로 소스 객체 소유자가 복제본을 소유합니다. 소스 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유한 경우 복제본 소유권을 대상 버킷을 소유한 AWS 계정으로 변경하는 선택적 구성 설정을 추가할 수 있습니다. 여기에는 `ObjectOwnerOverrideToBucketOwner` 권한 부여가 포함됩니다. 자세한 내용은 복제본 소유자 변경 섹션을 참조하세요.

복제본 소유자 변경

복제에서는 기본적으로 소스 객체 소유자가 복제본도 소유합니다. 단, 소스 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유한 경우 복제본 소유권을 변경할 수 있습니다. 예를 들어, 객체 복제본에 대한 액세스를 제한할 목적으로 소유권을 변경할 수 있습니다. 복제 구성에서 복제본 소유권을 대상 버킷을 소유한 AWS 계정으로 변경하는 선택적 구성 설정을 추가할 수 있습니다.

복제본 소유자를 변경하려면 다음을 수행합니다.

- 복제본 소유권을 변경하도록 Amazon S3에 지시하는 소유자 재정의 옵션을 복제 구성에 추가합니다.
- Amazon S3에 복제본 소유권을 변경할 수 있는 `s3:ObjectOwnerOverrideToBucketOwner` 권한을 부여합니다.
- 대상 버킷 정책에 복제본 소유권 변경을 허용할 `s3:ObjectOwnerOverrideToBucketOwner` 권한을 추가합니다. `s3:ObjectOwnerOverrideToBucketOwner` 권한을 통해 대상 버킷의 소유자가 객체 복제본의 소유권을 수락할 수 있습니다.

자세한 내용은 the section called “소유권 재정의 옵션에 대한 고려 사항” 및 복제 구성에 소유자 재정의 옵션 추가(을)를 참조하세요. 단계별 지침이 포함된 예제는 복제본 소유자 변경 방법 단원을 참조하십시오.

> **Important**
>
> 소유자 재정의 옵션을 사용하는 대신 객체 소유권에 대해 버킷 소유자 적용 설정을 사용할 수 있습니다. 복제를 사용하고 소스 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유하는 경우, 대상 버킷의 버킷 소유자는 객체 소유권에 대한 버킷 소유자 적용 설정을 사용하여 복제 소유권을 대상 버킷을 소유하는 AWS 계정으로 변경할 수 있습니다. 이 설정은 객체 액세스 제어 목록(ACL)을 비활성화합니다.

버킷 소유자 적용 설정은 `s3:ObjectOwnerOverrideToBucketOwner` 권한 없이도 기존 소유자 재정의 동작을 모방합니다. 버킷 소유자 시행 설정으로 대상 버킷에 복제되는 모든 객체는 대상 버킷 소유자가 소유합니다. 객체 소유권에 대한 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하십시오.

소유권 재정의 옵션에 대한 고려 사항

소유권 재정의 옵션을 구성할 때는 다음 사항을 고려해야 합니다.

- 기본적으로 원본 객체의 소유자도 복제본을 소유합니다. Amazon S3는 객체 버전 및 이와 연결된 ACL을 복제합니다.

  복제 구성에 소유자 재정의 옵션을 추가하면 Amazon S3는 ACL이 아닌 객체 버전만 복제합니다. 또한 Amazon S3는 원본 객체 ACL에 대한 후속 변경 사항을 복제하지 않습니다. Amazon S3는 복제본에서 대상 버킷 소유자에게 모든 권한을 부여하는 ACL을 설정합니다.
- 소유자 재정의를 활성화하거나 비활성화하도록 복제 구성을 업데이트하면 다음과 같은 동작이 발생합니다.
  - 복제 구성에 소유자 재정의 옵션을 추가할 경우:

    Amazon S3는 객체 버전을 복제할 때 소스 객체에 연결된 ACL을 무시합니다. 대신 Amazon S3는 복제본에서 대상 버킷 소유자에게 모든 권한을 부여하는 ACL을 설정합니다. Amazon S3는 소스 객체 ACL에 대한 후속 변경 사항을 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. 그 이유는 Amazon S3가 수행한 객체 소유권 변경이 유효하기 때문입니다. 즉, 소유자 재정의가 설정되었을 때 복제된 객체 버전에 적용된 ACL은 계속해서 복제되지 않습니다.

복제 구성에 소유자 재정의 옵션 추가

> **Warning**
>
> 소스 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유한 경우에만 소유자 재정의 옵션을 추가합니다. Amazon S3에서는 버킷을 소유한 계정이 동일한지 여부를 확인하지 않습니다. 두 버킷을 동일한 AWS 계정에서 모두 소유한 경우에도 소유자 재정의를 추가하면 Amazon S3가 소유자 재정의를 적용합니다. 이 옵션은 대상 버킷의 소유자에게 전체 권한을 부여하며 소스 객체의 액세스 제어 목록(ACL)에 대한 후속 업데이트를 복제하지 않습니다. 복제본 소유자는 `PutObjectAcl` 요청을 사용하여 복제본과 연결된 ACL을 직접 변경할 수 있지만, 복제를 통해서는 변경할 수 없습니다.

소유자 재정의 옵션을 지정하려면 다음을 각 `Destination` 요소에 추가합니다.

- 복제본 소유자 변경을 Amazon S3에 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 키 접두사가 *Tax*인 객체를 *amzn-s3-demo-destination-bucket* 대상 버킷으로 복제하고 복제본 소유권을 변경하도록 Amazon S3에 지시합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
<?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>
      <DeleteMarkerReplication>
         <Status>Disabled</Status>
      </DeleteMarkerReplication>
      <Filter>
         <Prefix>Tax</Prefix>
      </Filter>
      <Destination>
         <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
         <Account>destination-bucket-owner-account-id</Account>
         <AccessControlTranslation>
            <Owner>Destination</Owner>
         </AccessControlTranslation>
      </Destination>
   </Rule>
</ReplicationConfiguration>
```

복제본 소유권을 변경할 수 있는 Amazon S3 권한 부여

AWS Identity and Access Management(IAM) 역할과 연결된 권한 정책에 `s3:ObjectOwnerOverrideToBucketOwner` 작업에 대한 권한을 추가하여 복제본 소유권을 변경할 권한을 Amazon S3에 부여합니다. 이 역할은 Amazon S3가 사용자를 대신하여 객체를 맡고 복제할 수 있도록 복제 구성에서 지정한 IAM 역할입니다. 다음 예제를 사용하려면 *amzn-s3-demo-destination-bucket*을 대상 버킷의 이름으로 바꿉니다.

```
...
{
    "Effect":"Allow",
         "Action":[
       "s3:ObjectOwnerOverrideToBucketOwner"
    ],
    "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
}
...
```

대상 버킷 정책에 복제본 소유권 변경을 허용할 권한 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. 이 권한을 통해 대상 버킷 소유자는 객체 복제본의 소유권을 수락할 수 있습

니다. 다음 버킷 정책 문 예제에서는 이를 수행하는 방법을 보여줍니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
...
{
    "Sid":"1",
    "Effect":"Allow",
    "Principal":{"AWS":"source-bucket-account-id"},
    "Action":["s3:ObjectOwnerOverrideToBucketOwner"],
    "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
}
...
```

복제본 소유자 변경 방법

복제 구성의 소스 및 대상 버킷을 서로 다른 AWS 계정에서 소유하는 경우, Amazon S3에 대상 버킷을 소유한 AWS 계정으로 복제 소유권을 변경하도록 지시할 수 있습니다. 다음 예제에서는 Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 및 AWS SDK를 사용하여 복제본 소유권을 변경하는 방법을 보여줍니다.

S3 콘솔 사용

단계별 지침은 동일한 계정에서 버킷 복제 구성섹션을 참조하세요. 이 주제에서는 소스 및 대상 버킷을 동일하거나 서로 다른 AWS 계정에서 소유하는 경우 복제 구성을 설정하는 방법을 설명합니다.

AWS CLI 사용

다음 절차에서는 AWS CLI를 사용하여 복제본 소유권을 변경하는 방법을 설명합니다. 이 절차에서는 다음을 수행합니다.

- 소스 및 대상 버킷을 만듭니다.
- 버킷의 버전 관리를 활성화합니다.
- Amazon S3에 객체 복제 권한을 제공하는 AWS Identity and Access Management(IAM) 역할을 만듭니다.
- 소스 버킷에 복제 구성을 추가합니다.
- 복제 구성에서 복제본 소유권을 변경하도록 Amazon S3에 지시합니다.
- 복제 구성을 테스트합니다.

소스 버킷과 대상 버킷을 서로 다른 AWS 계정(AWS CLI)에서 소유한 경우 복제본 소유권을 변경하려면

이 절차에서 예제 AWS CLI 명령을 사용하려면 *user input placeholders*를 자체 정보로 대체합니다.

1. 이 예제에서는 소스 버킷과 대상 버킷을 서로 다른 두 AWS 계정에서 만듭니다. 이 두 계정으로 작업하려면 2개의 명명된 프로파일로 AWS CLI를 구성합니다. 이 예제에서는 이름이 *acctA* 및 *acctB*인 프로필을 사용합니다. 자격 증명 프로파일 설정 및 명명된 프로파일 사용에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 구성 및 자격 증명 파일 설정을 참조하세요.

   > **Important**
   >
   > 이 절차에 사용하는 프로파일에 필요한 권한이 있어야 합니다. 예를 들어 복제 구성에서 Amazon S3가 맡을 수 있는 IAM 역할을 지정합니다. 사용하는 프로파일에 `iam:PassRole` 권한이 있는 경우에만 이 작업을 수행할 수 있습니다. 관리자 사용자 자격 증명을 사용하여 명명된 프로파일을 만드는 경우 이 절차의 모든 태스크를 수행할 수 있습니다. 자세한 내용은 IAM 사용 설명서에서 사용자에게 AWS 서비스에 역할을 전달할 권한 부여를 참조하세요.

2. 원본 버킷을 생성하고 버전 관리를 사용 설정합니다. 이 예제에서는 미국 동부(버지니아 북부)(`us-east-1`) 리전에 *amzn-s3-demo-source-bucket*이라는 소스 버킷을 만듭니다.

   ```
   aws s3api create-bucket \
   --bucket amzn-s3-demo-source-bucket \
   --region us-east-1 \
   --profile acctA
   ```

   ```
   aws s3api put-bucket-versioning \
   --bucket amzn-s3-demo-source-bucket \
   --versioning-configuration Status=Enabled \
   --profile acctA
   ```

3. 대상 버킷을 생성하고 버전 관리를 사용 설정합니다. 이 예제에서는 미국 서부(오리건)(`us-west-2`) 리전에 *amzn-s3-demo-destination-bucket*이라는 이름의 대상 버킷을 만듭니다. 소스 버킷에 사용했던 것과는 다른 AWS 계정 프로파일을 사용합니다.

   ```
   aws s3api create-bucket \
   --bucket amzn-s3-demo-destination-bucket \
   ```

```
--region us-west-2 \
--create-bucket-configuration LocationConstraint=us-west-2 \
--profile acctB
```

```
aws s3api put-bucket-versioning \
--bucket amzn-s3-demo-destination-bucket \
--versioning-configuration Status=Enabled \
--profile acctB
```

4. ## 버킷 정책에 복제본 소유권 변경을 허용할 권한을 추가해야 합니다.

   a. 다음 정책을 *destination-bucket-policy*.json 이름의 파일에 저장합니다. *user input placeholders*를 사용자의 정보로 대체합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "destination_bucket_policy_sid",
      "Principal": {
        "AWS": "source-bucket-owner-account-id"
      },
      "Action": [
        "s3:ReplicateObject",
        "s3:ReplicateDelete",
        "s3:ObjectOwnerOverrideToBucketOwner",
        "s3:ReplicateTags",
        "s3:GetObjectVersionTagging"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
      ]
    }
  ]
}
```

   b. 다음 put-bucket-policy 명령을 사용하여 이전 정책을 대상 버킷에 추가합니다.

```
aws s3api put-bucket-policy --region $ {destination-region} --bucket $ {amzn-s3-demo-destination-bucket} --policy file://destination_bucket_policy.json
```

5. IAM 역할 생성. 나중에 소스 버킷에 추가하는 복제 구성에서 이 역할을 지정합니다. Amazon S3는 사용자를 대신하여 객체를 복제하기 위해 이 역할을 맡습니다. IAM 역할은 다음의 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*.json이라는 이름의 파일로 저장합니다. 이 정책은 역할을 맡을 권한을 Amazon S3에 부여합니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

      ii. 다음 AWS CLI `create-role` 명령을 실행하여 IAM 역할을 만듭니다.

```
$ aws iam create-role \
--role-name replicationRole \
--assume-role-policy-document file://s3-role-trust-policy.json  \
--profile acctA
```

      사용자가 만든 IAM 역할의 Amazon 리소스 이름(ARN)을 적어 둡니다. 이후 단계에서 이 ARN이 필요합니다.

   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 작

업에 대한 권한을 부여합니다. 다음 단계에서는 이 정책을 이전에 만든 IAM 역할에 연결합니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket",
            "s3:GetReplicationConfiguration"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ObjectOwnerOverrideToBucketOwner",
            "s3:ReplicateTags",
            "s3:GetObjectVersionTagging"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
      }
   ]
}
```

ii. 이전 권한 정책을 역할에 연결하려면 다음 `put-role-policy` 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 AccessControlTranslation은 소스 버킷 소유자에서 대상 버킷 소유자로의 복제본 소유권 변경을 지정합니다.

```
{
   "Role":"IAM-role-ARN",
   "Rules":[
      {
         "Status":"Enabled",
         "Priority":1,
         "DeleteMarkerReplication":{
            "Status":"Disabled"
         },
         "Filter":{
         },
         "Status":"Enabled",
         "Destination":{
            "Bucket":"arn:aws:s3:::amzn-s3-demo-destination-bucket",
            "Account":"destination-bucket-owner-account-id",
            "AccessControlTranslation":{
               "Owner":"Destination"
            }
         }
      }
   ]
}
```

   b. 대상 버킷 이름, 대상 버킷 소유자 계정 ID 및 *IAM-role-ARN*에 값을 입력하여 JSON을 편집합니다. *IAM-role-ARN*을 이전에 생성한 IAM 역할의 ARN으로 바꿉니다. 변경 사항을 저장합니다.

   c. 소스 버킷에 복제 구성을 추가하려면 다음 명령을 실행합니다.

```
$ aws s3api put-bucket-replication \
--replication-configuration file://replication.json \
```

```
--bucket amzn-s3-demo-source-bucket \
--profile acctA
```

7. Amazon S3 콘솔에서 복제본 소유권을 확인하여 복제 구성을 테스트합니다.

   a. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.

   b. ## 버킷에 객체를 추가합니다. 대상 버킷에 객체 복제본이 포함되어 있는지, 복제본의 소유권이 대상 버킷을 소유한 AWS 계정으로 변경되었는지 확인합니다.

AWS SDK 사용

복제 구성을 추가하는 코드 예제는 AWS SDK 사용 섹션을 참조하세요. 복제 구성을 적절히 수정해야 합니다. 개념적 정보는 복제본 소유자 변경 섹션을 참조하세요.

S3 Replication Time Control을 사용하여 규정 준수 요구 사항 충족

S3 Replication Time Control(S3 RTC)은 데이터 복제와 관련된 규정 준수 요구 사항이나 비즈니스 요구 사항을 충족할 수 있도록 지원하며 Amazon S3 복제 활동에 대한 가시성을 제공합니다. S3 RTC는 Amazon S3에 업로드하는 대부분의 객체를 몇 초 만에 복제하고 이러한 객체의 99.9%를 15분 내에 복제합니다.

기본적으로 S3 RTC에는 복제 진행 상황을 추적하는 2가지 방법이 있습니다.

- S3 복제 지표 – S3 복제 지표를 사용하여 복제 대기 중인 총 S3 API 작업 수, 복제 대기 중인 객체의 총 크기, 대상 리전으로의 최대 복제 시간, 복제에 실패한 총 작업 수를 모니터링할 수 있습니다. 그런 다음 별도로 복제하는 각 데이터 세트를 모니터링할 수 있습니다. S3 RTC와 독립적으로 S3 복제 지표를 활성화할 수도 있습니다. 자세한 내용은 the section called “S3 복제 지표 사용” 섹션을 참조하세요.

  S3 Replication Time Control(S3 RTC)이 활성화된 복제 규칙은 S3 복제 지표를 게시합니다. 복제 지표는 S3 RTC를 사용 설정한 후 15분 이내에 사용할 수 있습니다. 복제 지표는 Amazon S3 콘솔, Amazon S3 API, AWS SDK, AWS Command Line Interface(AWS CLI), Amazon CloudWatch를 통해 사용할 수 있습니다. CloudWatch 지표에 대한 자세한 내용은 Amazon CloudWatch를 사용한 지표 모니터링 섹션을 참조하세요. Amazon S3 콘솔을 통해 복제 지표를 보는 방법에 대한 자세한 내용은 복제 지표 보기 섹션을 참조하세요.

  S3 복제 지표는 Amazon CloudWatch 사용자 지정 지표와 동일한 요금으로 청구됩니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

- Amazon S3 이벤트 알림 - S3 RTC는 객체 복제가 15분의 임계값을 초과하거나 초과 후에 이루어지는 경우 버킷 소유자에게 알리는 `OperationMissedThreshold` 및 `OperationReplicatedAfterThreshold` 이벤트를 제공합니다. S3 RTC의 경우, Amazon S3 이벤트 알림은 드물지만 객체가 15분 이내에 복제되지 않는 경우와 해당 객체가 15분의 임계값 이후에 복제되는 경우를 알려줄 수 있습니다.

  복제 지표는 S3 RTC를 사용 설정한 후 15분 이내에 사용할 수 있습니다. Amazon S3 이벤트 알림은 Amazon SQS, Amazon SNS 또는 AWS Lambda를 통해 제공됩니다. 자세한 내용은 the section called “복제 실패 이벤트 수신” 섹션을 참조하세요.

S3 RTC에 대한 모범 사례 및 지침

S3 Replication Time Control(S3 RTC)을 활성화한 상태에서 Amazon S3에서 데이터를 복제할 때는 다음 모범 사례 지침을 따라 워크로드에 맞게 복제 성능을 최적화하세요.

주제

- Amazon S3 복제 및 요청 처리 성능 지침
- 복제 요청 속도 추정
- S3 RTC 데이터 전송 속도 할당량 초과
- AWS KMS 암호화된 객체 복제 요청 속도

Amazon S3 복제 및 요청 처리 성능 지침

애플리케이션은 Amazon S3에서 스토리지를 업로드하고 검색할 때 요청 성능에서 초당 수천 회의 트랜잭션을 구현할 수 있습니다. 예를 들어, 애플리케이션은 S3 복제가 사용자를 대신해서 수행하는 요청을 포함하여 S3 버킷의 접두사에 대해 초당 3,500개 이상의 `PUT/COPY/POST/DELETE` 또는 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 빈도 또는 `LIST` 요청과 동시에 이루어지는 연속 요청 빈도에 따라 자동으로 스케일 인됩니다. Amazon S3가 새로운 요청 빈도에 대해 내부적으로 최적화하는 동안 최적화가 완료될 때까지 일시적으로 HTTP 503 요청 응답을 받을 수 있습니다. 이 동작은 초당 요청 빈도가 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.

또한, 복제 데이터 전송 속도가 기본 초당 1기가비트(Gbps) 할당량을 초과하는 기간에는 S3 RTC SLA가 적용되지 않습니다. 복제 전송 속도가 1Gbps를 초과할 것으로 예상되는 경우 AWS Support Center에 문의하거나 Service Quotas를 사용하여 복제 전송 속도 할당량 증가를 요청할 수 있습니다.

복제 요청 속도 추정

사용자를 대신하여 Amazon S3 복제가 수행하는 요청을 포함한 총 요청 빈도는 복제 소스 및 대상 버킷 모두에 대한 Amazon S3 요청 빈도 지침 내에 있어야 합니다. 복제된 각 객체에 대해 Amazon S3 복제는 소스 버킷에 최대 5개의 GET/HEAD 요청과 1개의 PUT 요청을, 각 대상 버킷에 1개의 PUT 요청을 수행합니다.

예를 들어, 초당 100개의 객체를 복제하려는 경우 Amazon S3 복제는 소스 S3 버킷에 대해 초당 총 200개의 PUT 요청에 추가로 100개의 PUT 요청을 사용자 대신 수행할 수 있습니다. 또한, Amazon S3 복제는 최대 500개의 GET/HEAD 요청을 수행할 수 있습니다(복제되는 각 객체에 대해 5개의 GET/HEAD 요청)

> **Note**
>
> 복제된 객체당 하나의 PUT 요청에만 비용이 발생합니다. 자세한 내용은 복제 관련 Amazon S3 FAQ의 요금 정보를 참조하세요.

S3 RTC 데이터 전송 속도 할당량 초과

S3 RTC 데이터 전송 속도가 기본 1Gbps 할당량을 초과할 것으로 예상되는 경우 AWS Support Center에 문의하거나 Service Quotas를 사용하여 복제 전송 속도 할당량 증가를 요청하세요.

AWS KMS 암호화된 객체 복제 요청 속도

AWS Key Management Service(AWS KMS) 키를 사용한 서버 측 암호화로 암호화(SSE-KMS)된 객체를 복제할 때 초당 AWS KMS 요청 할당량이 적용됩니다. AWS KMS에서는 요청 빈도가 초당 요청 수 할당량을 초과하기 때문에 유효한 요청을 거부할 수 있습니다. 요청이 조절되면 AWS KMS이(가) `ThrottlingException` 오류를 반환합니다. AWS KMS 요청 빈도 할당량은 사용자가 직접 수행하는 요청과 사용자를 대신하여 Amazon S3 복제가 수행한 요청에 적용됩니다.

예를 들어, 초당 1,000개의 객체를 복제하려는 경우 AWS KMS 요청 빈도 할당량에서 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.

초당 AWS KMS 요청 할당량 증가를 요청하려면 AWS Support Center에 문의하거나 Service Quotas를 사용하세요.

S3 Replication Time Control 사용 설정

먼저 S3 Replication Time Control(S3 RTC)을 새 복제 규칙 또는 기존 복제 규칙과 함께 사용할 수 있습니다. 복제 규칙을 전체 버킷에 적용하거나 특정 접두사 또는 태그가 있는 객체에 적용하도록 선택할 수 있습니다. S3 RTC를 활성화하면 복제 규칙에서도 S3 복제 지표가 활성화됩니다.

Amazon S3 콘솔, Amazon S3 API, AWS SDK, AWS Command Line Interface(AWS CLI)를 사용하여 S3 RTC를 구성할 수 있습니다.

S3 콘솔 사용

단계별 지침은 동일한 계정에서 버킷 복제 구성섹션을 참조하세요. 이 주제에서는 소스 및 대상 버킷을 동일하거나 서로 다른 AWS 계정이 소유한 경우에 복제 구성에서 S3 RTC를 활성화하는 지침을 제공합니다.

AWS CLI 사용

AWS CLI를 사용하여 S3 RTC가 사용 설정된 객체를 복제하려면 버킷을 생성하고 버킷에서 버전 관리를 사용하며 객체를 복제할 권한을 Amazon S3에 부여하는 IAM 역할을 생성하고 복제 구성을 소스 버킷에 추가합니다. 다음 예제와 같이 복제 구성에 S3 RTC가 활성화되어 있어야 합니다.

AWS CLI를 사용하여 복제 구성을 설정하는 방법에 대한 단계적 지침은 동일한 계정에서 버킷 복제 구성 섹션을 참조하세요.

다음 예제 복제 구성은 복제 규칙의 `ReplicationTime` 및 `EventThreshold` 값을 활성화하고 설정합니다. 이 값을 활성화하고 설정하면 규칙에서 S3 RTC가 활성화됩니다.

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "DeleteMarkerReplication": {
                "Status": "Disabled"
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::amzn-s3-demo-destination-bucket",
                "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 값으로 15만 가질 수 있습니다.

Java용 AWS SDK 사용

다음 Java 예제에서는 S3 Replication Time Control(S3 RTC)을 활성화한 복제 구성을 추가합니다.

```
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);
  }
}
```

암호화(SSE-S3, SSE-KMS, DSSE-KMS, SSE-C)된 객체 복제

> **Important**
>
> 이제 Amazon S3가 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)를 Amazon S3 내 모든 버킷 암호화의 기본 수준으로 적용합니다. 2023년 1월 5일부터 Amazon S3로의 모든 새 객체 업로드는 추가 비용 없이 성능에 영향을 미치지 않고 자동으로 암호화됩니다. S3 버킷 기본 암호화 구성에 및 신규 객체 업로드에 대한 자동 암호화 상태는 AWS CloudTrail 로그, S3 인벤토리, S3 스토리지 렌즈, Amazon S3 콘솔에서 사용할 수 있으며, AWS Command Line Interface 및 AWS SDK에서 추가 Amazon S3 API 응답 헤더로도 사용할 수 있습니다. 자세한 내용은 기본 암호화 관련 FAQ를 참조하십시오.

서버 측 암호화를 사용하여 암호화된 객체를 복제할 때는 몇 가지 특별한 고려 사항이 있습니다. Amazon S3에서는 다음 유형의 서버 측 암호화를 지원합니다.

- Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)
- AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 사용한 서버 측 암호화

- AWS KMS 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)
- 고객 제공 키를 사용한 서버 측 암호화(SSE-C)

서버 측 암호화에 대한 자세한 정보는 the section called “서버 측 암호화” 섹션을 참조하십시오.

이 주제에서는 서버 측 암호화를 사용하여 암호화된 객체를 복제하도록 Amazon S3에 지시하는 데 필요한 권한에 대해 설명합니다. 이 주제에서는 추가할 수 있는 추가 구성 요소와 암호화된 객체를 복제하는 데 필요한 권한을 부여하는 AWS Identity and Access Management(IAM) 정책의 예도 제공합니다.

단계별 지침이 포함된 예제는 암호화된 객체에 대한 복제 활성화 단원을 참조하십시오. 복제 구성 생성에 대한 자세한 내용은 리전 내 및 리전 간 객체 복제 단원을 참조하십시오.

> **Note**
>
> Amazon S3에서 다중 리전 AWS KMS keys를 사용할 수 있습니다. 그러나 Amazon S3는 현재 다중 리전 키를 단일 리전 키인 것처럼 취급하며, 키의 다중 리전 기능을 사용하지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 다중 리전 키 사용을 참조하세요.

주제

- 기본 버킷 암호화가 복제에 미치는 영향
- SSE-C로 암호화된 객체 복제
- SSE-S3, SSE-KMS 또는 DSSE-KMS로 암호화된 객체 복제
- 암호화된 객체에 대한 복제 활성화

기본 버킷 암호화가 복제에 미치는 영향

복제 대상 버킷에 대한 기본 암호화를 사용 설정하면 다음 암호화 동작이 적용됩니다.

- 소스 버킷의 객체가 암호화되지 않은 경우 대상 버킷의 복제본 객체는 대상 버킷의 기본 암호화 설정을 사용하여 암호화됩니다. 결과적으로 소스 객체의 엔터티 태그(ETag)는 복제본 객체의 ETag와 다릅니다. ETag를 사용하는 애플리케이션이 있는 경우 이 차이를 고려하도록 해당 애플리케이션을 업데이트해야 합니다.
- 소스 버킷의 객체가 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3), AWS Key Management Service(AWS KMS) 키를 사용한 서버 측 암호화(SSE-KMS) 또는 AWS KMS 키를 사

용한 이중 계층 서브 측 암호화(DSSE-KMS)를 사용하여 암호화되는 경우, 대상 버킷의 복제본 객체는 소스 객체와 동일한 유형의 암호화를 사용합니다. 대상 버킷의 기본 암호화 설정은 사용되지 않습니다.

SSE-C로 암호화된 객체 복제

고객 제공 키(SSE-C)로 서버 측 암호화를 사용하면 자체 암호화 키를 관리할 수 있습니다. SSE-C를 사용하면 사용자는 키를 관리하고 Amazon S3는 암호화 및 암호 해독 프로세스를 관리합니다. 요청의 일부로 암호화 키를 제공해야 하지만 객체 암호화 또는 암호 해독을 수행하기 위해 코드를 작성할 필요는 없습니다. 객체를 업로드할 때 Amazon S3는 제공된 키를 사용하여 객체를 암호화합니다. 그런 다음 Amazon S3는 해당 키를 메모리에서 삭제합니다. 객체를 검색할 경우 요청에 포함된 것과 동일한 암호화 키를 제공해야 합니다. 자세한 내용은 the section called “고객 제공 암호화 키(SSE-C)” 섹션을 참조하세요.

S3 복제는 SSE-C로 암호화된 객체를 지원합니다. 암호화되지 않은 객체에 대한 복제를 구성하는 것과 동일한 방식으로 Amazon S3 콘솔 또는 AWS SDK를 사용하여 SSE-C 객체 복제를 구성할 수 있습니다. 현재 복제에 필요한 권한 이외의 추가 SSE-C 권한은 없습니다.

S3 복제는 가능한 경우 새로 업로드된 SSE-C 암호화된 객체를 S3 복제 구성에서 지정된 대로 자동으로 복제합니다. 버킷의 기존 객체를 복제하려면 S3 배치 복제를 사용합니다. 객체 복제에 대한 자세한 내용은 the section called “라이브 복제 설정” 및 the section called “기존 객체 복제” 섹션을 참조하세요.

SSE-C 객체 복제에 대한 추가 비용은 없습니다. 복제 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하세요.

SSE-S3, SSE-KMS 또는 DSSE-KMS로 암호화된 객체 복제

기본적으로 Amazon S3는 SSE-KMS 또는 DSSE-KMS로 암호화된 객체를 복제하지 않습니다. 이 섹션에서는 이러한 객체를 복제하도록 Amazon S3에 지시하기 위해 추가할 수 있는 추가 구성 요소를 설명합니다.

단계별 지침이 포함된 예제는 암호화된 객체에 대한 복제 활성화 단원을 참조하십시오. 복제 구성 생성에 대한 자세한 내용은 리전 내 및 리전 간 객체 복제 단원을 참조하십시오.

복제 구성에서 추가 정보 지정

복제 구성에서 다음을 수행합니다.

- 복제 구성의 `Destination` 요소에 다음 복제 구성 예시와 같이 Amazon S3에서 객체 복제본을 암호화하는 데 사용할 대칭 AWS KMS 고객 관리 키의 ID를 추가합니다.

- KMS 키를 사용하여 암호화(SSE-KMS 또는 DSSE-KMS)된 객체의 복제를 사용 설정함으로써 명시적으로 옵트인합니다. 옵트인하려면 다음 예제 복제 구성에 표시된 대로 `SourceSelectionCriteria` 요소를 추가합니다.

```
<ReplicationConfiguration>
   <Rule>
      ...
      <SourceSelectionCriteria>
         <SseKmsEncryptedObjects>
           <Status>Enabled</Status>
         </SseKmsEncryptedObjects>
      </SourceSelectionCriteria>

      <Destination>
          ...
          <EncryptionConfiguration>
             <ReplicaKmsKeyID>AWS KMS key ARN or Key Alias ARN that's in the same AWS #
# as the destination bucket.</ReplicaKmsKeyID>
          </EncryptionConfiguration>
       </Destination>
      ...
   </Rule>
</ReplicationConfiguration>
```

Important

- KMS 키는 대상 버킷과 동일한 AWS 리전에서 생성되어야 합니다.
- KMS 키가 유효해야 합니다. `PutBucketReplication` API 작업은 KMS 키가 유효한지 확인하지 않습니다. 잘못된 KMS 키를 사용할 경우 응답에서 HTTP `200 OK` 상태 코드를 받지만 복제는 실패합니다.

다음 예시는 선택적 구성 요소가 포함된 복제 구성을 보여줍니다. 이 복제 구성에는 한 가지 규칙이 있습니다. 이 규칙은 *Tax* 키 접두사를 포함하는 모든 객체에 적용됩니다. Amazon S3는 지정된 AWS KMS key ID를 사용하여 이러한 객체 복제본을 암호화합니다.

```
<?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:::amzn-s3-demo-destination-bucket</Bucket>
         <EncryptionConfiguration>
            <ReplicaKmsKeyID>AWS KMS key ARN or Key Alias ARN that's in the same AWS ##
 as the destination bucket.</ReplicaKmsKeyID>
         </EncryptionConfiguration>
      </Destination>
      <SourceSelectionCriteria>
         <SseKmsEncryptedObjects>
            <Status>Enabled</Status>
         </SseKmsEncryptedObjects>
      </SourceSelectionCriteria>
   </Rule>
</ReplicationConfiguration>
```

IAM 역할에 추가 권한 부여

SSE-S3, SSE-KMS 또는 DSSE-KMS를 사용하여 저장 시 암호화된 객체를 복제하려면 복제 구성에 지정하는 AWS Identity and Access Management(IAM) 역할에 다음 추가 권한을 부여합니다. IAM 역할과 연결된 권한 정책을 업데이트하여 이러한 권한을 부여합니다.

- 소스 객체에 대한 **`s3:GetObjectVersionForReplication`** 작업 – 이 작업은 Amazon S3가 SSE-S3, SSE-KMS 또는 DSSE-KMS를 사용한 서버 측 암호화로 생성된 객체와 암호화되지 않은 객체를 모두 복제할 수 있도록 허용합니다.

> **Note**
>
> `s3:GetObjectVersionForReplication`은 복제에 필요한 최소한의 권한만 Amazon S3에 제공하므로 `s3:GetObjectVersion` 작업 대신 `s3:GetObjectVersionForReplication` 작업을 사용하는 것이 좋습니다. 또한 `s3:GetObjectVersion` 작업을 사용하면 암호화되지 않은 객체와 SSE-S3로 암호화된 객

체를 복제할 수 있지만 KMS 키를 사용하여 암호화(SSE-KMS 또는 DSSE-KMS)된 객체는 복제할 수 없습니다.

- KMS 키에 대한 **`kms:Decrypt`** 및 **`kms:Encrypt`** AWS KMS 작업
  - 소스 객체의 암호를 해독하는 데 사용되는 AWS KMS key에 `kms:Decrypt` 권한을 부여해야 합니다.
  - 객체 복제본의 암호를 해독하는 데 사용되는 AWS KMS key에 `kms:Encrypt` 권한을 부여해야 합니다.
- 일반 텍스트 객체 복제를 위한 **`kms:GenerateDataKey`** 작업 - 기본적으로 SSE-KMS 또는 DSSE-KMS 암호화가 활성화된 버킷에 일반 텍스트 객체를 복제하는 경우, 대상 암호화 컨텍스트에 대한 `kms:GenerateDataKey` 권한과 KMS 키를 IAM 정책에 포함시켜야 합니다.

AWS KMS 조건 키를 사용하여 대상 버킷과 객체로만 이러한 권한을 제한하는 것이 좋습니다. IAM 역할을 소유한 AWS 계정은 정책에 나열된 KMS 키의 이러한 `kms:Encrypt` 및 `kms:Decrypt` 작업에 대한 권한이 있어야 합니다. 다른 AWS 계정이 KMS 키를 소유한 경우 KMS 키 소유자는 IAM 역할을 소유한 AWS 계정에 이러한 권한을 부여해야 합니다. 이러한 KMS 키에 대한 액세스를 관리하는 방법에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 AWS KMS에서 IAM 정책 사용을 참조하세요.

S3 버킷 키 및 복제

S3 버킷 키와 함께 복제를 사용하려면 객체 복제본을 암호화하는 데 사용되는 KMS 키에 대한 AWS KMS key 정책에 호출 보안 주체에 대한 `kms:Decrypt` 권한이 포함되어야 합니다. `kms:Decrypt`에 대한 호출은 S3 버킷 키를 사용하기 전에 S3 버킷 키의 무결성을 확인합니다. 자세한 내용은 복제와 함께 S3 버킷 키 사용 섹션을 참조하세요.

소스 또는 대상 버킷에 대해 S3 버킷 키가 활성화되면 암호화 컨텍스트는 객체 ARN(예: `arn:aws:s3:::`*`bucket_ARN`*)이 아니라 버킷 Amazon 리소스 이름(ARN)이 됩니다. 암호화 컨텍스트에 버킷 ARN을 사용하려면 IAM 정책을 업데이트해야 합니다.

```
"kms:EncryptionContext:aws:s3:arn": [
"arn:aws:s3:::bucket_ARN"
]
```

자세한 내용은 암호화 컨텍스트(x-amz-server-side-encryption-context)("REST API 사용" 섹션) 및 S3 버킷 키를 사용 설정하기 전에 유의할 변경 사항을 참조하세요.

예제 정책: 복제와 함께 SSE-S3 및 SSE-KMS 사용

다음 IAM 정책 예는 복제와 함께 SSE-S3 및 SSE-KMS를 사용하기 위한 스테이트먼트를 보여줍니다.

Example – 별도의 대상 버킷과 함께 SSE-KMS 사용

다음 정책 예는 별도의 대상 버킷과 함께 SSE-KMS를 사용하기 위한 스테이트먼트를 보여줍니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action": ["kms:Decrypt"],
         "Effect": "Allow",
         "Condition": {
             "StringLike": {
                  "kms:ViaService": "s3.source-bucket-region.amazonaws.com",
                  "kms:EncryptionContext:aws:s3:arn": [
                      "arn:aws:s3:::amzn-s3-demo-source-bucket/key-prefix1*"
                  ]
             }
          },
          "Resource": [
             "List of AWS KMS key ARNs that are used to encrypt source objects."
          ]
      },
      {
         "Action": ["kms:Encrypt"],
         "Effect": "Allow",
         "Condition": {
            "StringLike": {
               "kms:ViaService": "s3.destination-bucket-1-region.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn": [
                  "arn:aws:s3:::amzn-s3-demo-destination-bucket1/key-prefix1*"
               ]
         }
         },
         "Resource": [
            "AWS KMS key ARNs (in the same AWS ## as destination bucket 1). Used to
 encrypt object replicas created in destination bucket 1."
         ]
      },
      {
         "Action": ["kms:Encrypt"],
```

```
         "Effect": "Allow",
         "Condition": {
            "StringLike": {
               "kms:ViaService": "s3.destination-bucket-2-region.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn": [
                  "arn:aws:s3:::amzn-s3-demo-destination-bucket2/key-prefix1*"
               ]
         }
         },
         "Resource": [
            "AWS KMS key ARNs (in the same AWS ## as destination bucket 2). Used to
 encrypt object replicas created in destination bucket 2."
         ]
      }
   ]
}
```

Example – SSE-S3 및 SSE-KMS로 생성된 객체 복제

다음은 암호화되지 않은 객체, SSE-S3로 생성된 객체, SSE-KMS로 생성된 객체를 복제하는 데 필요한 권한을 부여하는 완전한 IAM 정책입니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket/key-prefix1*"
```

```
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-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:::amzn-s3-demo-source-bucket/key-prefix1*"
               ]
            }
         },
         "Resource":[
           "List of the AWS KMS key ARNs that are 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:::amzn-s3-demo-destination-bucket/prefix1*"
               ]
            }
         },
         "Resource":[
            "AWS KMS key ARNs (in the same AWS ## as the destination bucket) to use for
encrypting object replicas"
         ]
```

```
      }
   ]
}
```

Example – S3 버킷 키를 사용하여 객체 복제

다음은 S3 버킷 키로 객체를 복제하는 데 필요한 권한을 부여하는 전체 IAM 정책입니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket/key-prefix1*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-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:::amzn-s3-demo-source-bucket"
               ]
            }
         },
         "Resource":[
           "List of the AWS KMS key ARNs that are 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:::amzn-s3-demo-destination-bucket"
               ]
            }
         },
         "Resource":[
            "AWS KMS key ARNs (in the same AWS ## as the destination bucket) to use for
 encrypting object replicas"
         ]
      }
   ]
}
```

교차 계정 시나리오를 위한 추가 권한 부여

소스 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유하는 교차 계정 시나리오에서는 KMS 키를 사용하여 객체 복제본을 암호화할 수 있습니다. 그러나 KMS 키 소유자는 원본 버킷 소유자에게 KMS 키를 사용할 수 있는 권한을 부여해야 합니다.

> **Note**
>
> 계정 간 SSE-KMS 데이터를 복제해야 하는 경우 복제 규칙에 대상 계정에 대해 AWS KMS의 고객 관리형 키를 지정해야 합니다. AWS 관리형 키에서는 계정 간 사용을 허용하지 않으므로 계정 간 복제를 수행하는 데 사용할 수 없습니다.

소스 버킷 소유자에게 KMS 키 사용 권한 부여(AWS KMS 콘솔)

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/kms에서 AWS KMS 콘솔을 엽니다.
2. AWS 리전을 변경하려면 페이지의 오른쪽 상단 모서리에 있는 리전 선택기를 사용합니다.
3. 해당 계정에서 직접 생성하고 관리하는 키를 보려면 탐색 창에서 고객 관리형 키를 선택합니다.
4. KMS 키를 선택합니다.
5. 일반 구성 섹션에서 키 정책 탭을 선택합니다.
6. 아래로 스크롤하여 Other AWS 계정(기타 AWS 계정)로 이동합니다.
7. Add other AWS 계정(다른 AWS 계정 추가)를 선택합니다.

   기타 AWS 계정 대화 상자가 나타납니다.
8. 대화 상자에서 다른 AWS 계정 추가를 선택합니다. arn:aws:iam::의 경우 소스 버킷 계정 ID를 입력합니다.
9. 변경 사항 저장을 선택합니다.

원본 버킷 소유자에게 KMS 키 사용 권한 부여(AWS CLI)

- `put-key-policy` AWS Command Line Interface(AWS CLI) 명령에 대한 자세한 내용은 https://docs.aws.amazon.com/cli/latest/reference/kms/put-key-policy.htmlput-key-policy 명령 참조의 AWS CLI 섹션을 참조하세요. 기본 `PutKeyPolicy` API 작업에 대한 자세한 내용은 https://docs.aws.amazon.com/kms/latest/APIReference/AWS Key Management ServiceAPI 참조의 PutKeyPolicy 섹션을 참조하세요.

AWS KMS 트랜잭션 할당량 고려 사항

교차 지역 복제(CRR)를 사용 설정한 후 AWS KMS 암호화를 사용하여 새 객체를 많이 추가하면 조절(HTTP 503 `503 Service Unavailable` 오류)이 발생할 수 있습니다. 제한은 초당 AWS KMS

트랜잭션 수가 현재 할당량을 초과할 경우 발생합니다. 자세한 내용은 개발자 안내서에서 AWS Key Management Service 할당량을 참조하십시오.

할당량 증가를 요청하려면 Service Quotas를 사용합니다. 자세한 내용은 할당량 증가 요청을 참조하세요. 해당 리전에서 Service Quotas가 지원되지 않을 경우AWS Support 사례를 엽니다.

암호화된 객체에 대한 복제 활성화

기본적으로 Amazon S3는 AWS Key Management Service(AWS KMS) 키를 사용한 서버 측 암호화(SSE-KMS) 또는 AWS KMS 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)를 사용하여 암호화된 객체를 복제하지 않습니다. SSE-KMS 또는 DSS-KMS로 암호화된 객체를 복제하려면 버킷 복제 구성을 수정하여 이러한 객체를 복제하도록 Amazon S3에 지시해야 합니다. 이 예제에서는 Amazon S3 콘솔 및 AWS Command Line Interface(AWS CLI)를 사용해 암호화된 객체 복제를 사용하도록 버킷 복제 구성을 변경하는 방법을 설명합니다.

> **Note**
>
> 소스 또는 대상 버킷에 대해 S3 버킷 키를 사용하도록 설정하면 암호화 컨텍스트는 객체의 ARN이 아니라 버킷의 ARN(Amazon 리소스 이름)이 됩니다. 암호화 컨텍스트에 버킷 ARN을 사용하려면 IAM 정책을 업데이트해야 합니다. 자세한 내용은 S3 버킷 키 및 복제 섹션을 참조하세요.

> **Note**
>
> Amazon S3에서 다중 리전 AWS KMS keys를 사용할 수 있습니다. 그러나 Amazon S3는 현재 다중 리전 키를 단일 리전 키인 것처럼 취급하며, 키의 다중 리전 기능을 사용하지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 다중 리전 키 사용을 참조하세요.

S3 콘솔 사용

단계별 지침은 동일한 계정에서 버킷 복제 구성섹션을 참조하세요. 이 주제에서는 소스 및 대상 버킷을 동일하거나 서로 다른 AWS 계정에서 소유하는 경우 복제 구성을 설정하는 방법을 설명합니다.

AWS CLI 사용

AWS CLI를 사용하여 암호화된 객체를 복제하려면 다음을 수행합니다.

- 소스 및 대상 버킷을 생성하고 버킷에 버전 관리를 사용 설정합니다.
- Amazon S3에 객체 복제 권한을 제공하는 AWS Identity and Access Management(IAM) 역할을 생성합니다. IAM 역할의 권한은 암호화된 객체를 복제하는 데 필요한 권한을 포함합니다.
- 소스 버킷에 복제 구성을 추가합니다. 복제 구성이 KMS 키를 사용하여 암호화된 객체를 복제하는 데 관련된 정보를 제공합니다.
- 소스 버킷에 암호화된 객체를 추가합니다.
- 설정을 테스트하여 암호화된 객체가 대상 버킷에 복제되고 있는지 확인합니다.

아래에서 절차를 알아볼 수 있습니다.

서버 측 암호화된 객체 복제(AWS CLI)

이 절차에서 예제를 사용하려면 *user input placeholders*를 자체 정보로 대체합니다.

1. 이 예제에서는 동일한 AWS 계정에서 소스(*amzn-s3-demo-source-bucket*) 버킷과 대상(*amzn-s3-demo-destination-bucket*) 버킷을 모두 만듭니다. AWS CLI의 보안 인증 프로필도 설정합니다. 이 예시에서는 *acctA*라는 프로필 이름을 사용합니다.

   자격 증명 프로파일 설정 및 명명된 프로파일 사용에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 구성 및 자격 증명 파일 설정을 참조하세요.

2. 다음 명령을 사용하여 *amzn-s3-demo-source-bucket* 버킷을 생성하고 버킷에서 버전 관리를 활성화합니다. 다음 예시 명령은 미국 동부(버지니아 북부)(us-east-1) 리전에 *amzn-s3-demo-source-bucket* 버킷을 생성합니다.

   ```
   aws s3api create-bucket \
   --bucket amzn-s3-demo-source-bucket \
   --region us-east-1 \
   --profile acctA
   ```

   ```
   aws s3api put-bucket-versioning \
   --bucket amzn-s3-demo-source-bucket \
   --versioning-configuration Status=Enabled \
   --profile acctA
   ```

3. 다음 명령을 사용하여 *amzn-s3-demo-destination-bucket* 버킷을 생성하고 버킷에서 버전 관리를 활성화합니다. 다음 예시 명령은 미국 서부(오리건)(us-west-2) 리전에 *amzn-s3-demo-destination-bucket* 버킷을 생성합니다.

> **Note**
>
> *amzn-s3-demo-source-bucket* 및 *amzn-s3-demo-destination-bucket* 버킷이 모두 동일한 AWS 계정에 있을 때 복제 구성을 설정하려면 동일한 프로필을 사용합니다. 이 예제에서는 *acctA*를 사용합니다. 버킷을 서로 다른 AWS 계정에서 소유한 경우 복제 구성을 테스트하려면 각각 다른 프로필을 지정합니다.

```
aws s3api create-bucket \
--bucket amzn-s3-demo-destination-bucket \
--region us-west-2 \
--create-bucket-configuration LocationConstraint=us-west-2 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket amzn-s3-demo-destination-bucket \
--versioning-configuration Status=Enabled \
--profile acctA
```

4. IAM 서비스 역할을 생성합니다. 나중에 *amzn-s3-demo-source-bucket* 버킷에 추가하는 복제 구성에서 이 역할을 지정하게 됩니다. Amazon S3는 사용자를 대신하여 객체를 복제하기 위해 이 역할을 맡습니다. IAM 역할은 다음의 두 단계로 생성합니다.

   - 서비스 역할을 만듭니다.
   - 역할에 권한 정책을 연결합니다.

   a. IAM 서비스서비스 역할을 생성하려면 다음을 수행합니다.

      i. 다음 신뢰 정책을 복사하여 로컬 컴퓨터의 현재 디렉터리에 *s3-role-trust-policy-kmsobj*.json 파일로 저장합니다. 이 정책은 Amazon S3가 사용자를 대신하여 태스크를 수행할 수 있도록 역할을 맡을 권한을 Amazon S3 서비스 보안 주체에 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 replicationRolekmsobj \
--assume-role-policy-document file://s3-role-trust-policy-kmsobj.json  \
--profile acctA
```

b. 다음으로, 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 연결할 것입니다.

> **Important**
>
> 권한 정책에서 *amzn-s3-demo-source-bucket* 및 *amzn-s3-demo-destination-bucket* 버킷 암호화에 사용할 AWS KMS 키 ID를 지정합니다. *amzn-s3-demo-source-bucket* 및 *amzn-s3-demo-destination-bucket* 버킷에 대해 별도의 KMS 키 2개를 생성해야 합니다. AWS KMS keys는 생성된 AWS 리전 외부에서 공유되지 않습니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:ListBucket",
            "s3:GetReplicationConfiguration",
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"
```

```
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket",
            "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
         ]
      },
      {
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ReplicateTags"
         ],
         "Effect":"Allow",
         "Condition":{
            "StringLikeIfExists":{
               "s3:x-amz-server-side-encryption":[
                  "aws:kms",
                  "AES256",
                  "aws:kms:dsse"
               ],
               "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:::amzn-s3-demo-destination-bucket/*"
      },
      {
         "Action":[
            "kms:Decrypt"
         ],
         "Effect":"Allow",
         "Condition":{
            "StringLike":{
               "kms:ViaService":"s3.us-east-1.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn":[
                  "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
               ]
            }
         },
         "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:::amzn-s3-demo-destination-bucket/*"
               ]
            }
         },
         "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 *amzn-s3-demo-source-bucket* 버킷에 추가합니다. 이 구성은 *Tax/* 접두사가 있는 객체를 *amzn-s3-demo-destination-bucket* 버킷으로 복제하도록 Amazon S3에 지시합니다.

> **Important**
>
> 복제 구성에서 Amazon S3가 맡을 수 있는 IAM 역할을 지정합니다. `iam:PassRole` 권한이 있을 경우 이 권한만 사용할 수 있습니다. CLI 명령에서 지정한 프로필에 이 권한이 있

어야 합니다. 자세한 내용은 IAM 사용 설명서에서 사용자에게 AWS 서비스에 역할을 전달할 권한 부여를 참조하세요.

```
 <ReplicationConfiguration>
  <Role>IAM-Role-ARN</Role>
  <Rule>
    <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:::amzn-s3-demo-destination-bucket</Bucket>
      <EncryptionConfiguration>
        <ReplicaKmsKeyID>AWS KMS key IDs to use for encrypting object replicas</
ReplicaKmsKeyID>
      </EncryptionConfiguration>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

amzn-s3-demo-source-bucket 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:::amzn-s3-demo-destination-bucket",
            "EncryptionConfiguration":{
               "ReplicaKmsKeyID":"AWS KMS key IDs (in ARN format) to use for
 encrypting object replicas"
            }
         },
         "SourceSelectionCriteria":{
            "SseKmsEncryptedObjects":{
               "Status":"Enabled"
            }
         }
      }
   ]
}
```

b. JSON을 편집하여 *amzn-s3-demo-destination-bucket* 버킷, *AWS KMS key IDs (in ARN format)* 및 *IAM-role-ARN*에 대한 값을 제공합니다. 변경 사항을 저장합니다.

c. 다음 명령을 사용하여 *amzn-s3-demo-source-bucket* 버킷에 복제 구성을 추가합니다. 반드시 *amzn-s3-demo-source-bucket* 버킷 이름을 제공해야 합니다.

```
$ aws s3api put-bucket-replication \
--replication-configuration file://replication.json \
--bucket amzn-s3-demo-source-bucket \
--profile acctA
```

6. 구성을 테스트하여 암호화된 객체가 복제되는지 확인합니다. Amazon S3 콘솔에서 다음과 같이 수행합니다.

a. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.

b. *amzn-s3-demo-source-bucket* 버킷에서 이름이 *Tax*인 폴더를 생성합니다.

c. 폴더에 샘플 객체를 추가합니다. 반드시 암호화 옵션을 선택하고 객체를 암호화할 KMS 키를 지정해야 합니다.

d. *amzn-s3-demo-destination-bucket* 버킷에 객체 복제본이 있으며 해당 복제본이 구성에서 지정한 KMS 키를 사용하여 암호화되었는지 확인합니다. 자세한 내용은 the section called "복제 상태 가져오기" 섹션을 참조하세요.

AWS SDK 사용

복제 구성을 추가하는 코드 예시는 AWS SDK 사용 섹션을 참조하세요. 복제 구성을 적절히 수정해야 합니다.

복제본 수정 동기화를 사용하여 메타데이터 변경 복제

Amazon S3 복제본 수정 동기화를 사용하면 태그, 액세스 제어 목록(ACL), Object Lock 설정 등 객체 메타데이터를 복제본과 소스 객체 간에 복제할 수 있습니다. 기본적으로, Amazon S3는 원본 객체의 메타데이터를 복제본으로만 복제합니다. 복제본 수정 동기화가 활성화되면 Amazon S3는 복제본에 대한 메타데이터 변경 사항을 소스 객체로 다시 복제하므로, 복제가 양방향으로 수행됩니다.

복제본 수정 동기화 사용 설정

Amazon S3 복제본 수정 동기화를 새 복제 규칙이나 기존 복제 규칙과 함께 사용할 수 있습니다. 전체 버킷 또는 특정 접두사가 있는 객체에 이를 적용할 수 있습니다.

Amazon S3 콘솔을 사용하여 복제본 수정 동기화를 활성화하려면 라이브 복제 구성 예제 섹션을 참조하세요. 이 주제에서는 소스 및 대상 버킷을 동일하거나 서로 다른 AWS 계정에서 소유하고 있을 때 복제 구성에서 복제본 수정 동기화를 활성화하기 위한 지침을 제공합니다.

AWS Command Line Interface(AWS CLI)를 사용하여 복제본 수정 동기화를 활성화하려면 `ReplicaModifications`가 활성화된 상태의 복제본이 포함된 버킷에 복제 구성을 추가해야 합니다. 양방향 복제를 설정하려면 소스 버킷(*amzn-s3-demo-source-bucket*)에서 복제본이 포함된 버킷(*amzn-s3-demo-destination-bucket*)으로의 복제 규칙을 생성합니다. 그런 다음 복제본이 포함된 버킷(*amzn-s3-demo-destination-bucket*)에서 소스 버킷(*amzn-s3-demo-source-bucket*)으로의 두 번째 복제 규칙을 생성합니다. 소스 및 대상 버킷은 동일하거나 서로 다른 AWS 리전에 있을 수 있습니다.

> **Note**
>
> 복제된 객체의 객체 액세스 제어 목록(ACL), 객체 태그 또는 Object Lock 설정과 같은 복제본 메타데이터 변경 사항을 복제하려면 소스 및 대상 버킷 모두에서 복제본 수정 동기화를 활성화

해야 합니다. 모든 복제 규칙과 마찬가지로, 이러한 규칙은 전체 버킷이나 접두사 또는 객체 태그로 필터링된 객체의 하위 집합에 적용할 수 있습니다.

다음 예제 구성에서 Amazon S3는 *Tax* 접두사 아래에서 소스 객체가 포함된 *amzn-s3-demo-source-bucket* 버킷에 메타데이터 변경 사항을 복제합니다.

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "SourceSelectionCriteria": {
                "ReplicaModifications":{
                    "Status": "Enabled"
                }
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::amzn-s3-demo-source-bucket"
            },
            "Priority": 1
        }
    ],
    "Role": "IAM-Role-ARN"
}
```

AWS CLI를 사용한 복제 규칙 생성에 대한 전체 지침은 동일한 계정에서 버킷 복제 구성 섹션을 참조하세요.

버킷 간 삭제 마커 복제

기본적으로, S3 복제가 활성화되고 원본 버킷에서 객체가 삭제되면 Amazon S3는 원본 버킷에만 삭제 마커를 추가합니다. 이 작업은 우발적이거나 악의적인 삭제로부터 대상 버킷의 데이터를 보호하는 데 도움이 됩니다. 삭제 마커 복제를 활성화한 경우, 이러한 마커가 대상 버킷에 복사되고 Amazon S3는 소스 버킷과 대상 버킷 모두에서 객체가 삭제된 것처럼 동작합니다. 삭제 마커의 작동 원리에 대한 자세한 내용은 삭제 마커를 통한 작업 단원을 참조하세요.

> **Note**
>
> - 태그 기반 복제 규칙에는 삭제 마커 복제가 지원되지 않습니다. 삭제 마커 복제도 S3 Replication Time Control(S3 RTC)을 사용할 때 부여된 15분의 서비스 수준 계약(SLA)을 준수하지 않습니다.
> - 최신 복제 구성 XML 버전을 사용하지 않는 경우 삭제 작업은 복제에 다르게 영향을 줍니다. 자세한 내용은 삭제 작업이 복제에 미치는 영향 섹션을 참조하세요.
> - 삭제 마커 복제를 활성화하고 소스 버킷에 S3 수명 주기 만료 규칙이 있는 경우 S3 수명 주기 만료 규칙에 의해 추가된 삭제 마커는 대상 버킷에 복제되지 않습니다.

삭제 마커 복제 사용 설정

새 복제 규칙이나 기존 복제 규칙과 함께 삭제 마커 복제를 사용할 수 있습니다. 전체 버킷 또는 특정 접두사가 있는 객체에 삭제 마커 복제를 적용할 수 있습니다.

Amazon S3 콘솔을 사용하여 삭제 마커 복제를 활성화하려면 S3 콘솔 사용 섹션을 참조하세요. 이 주제에서는 소스 및 대상 버킷을 동일하거나 서로 다른 AWS 계정에서 소유하고 있을 때 복제 구성에서 삭제 마커 복제를 활성화하기 위한 지침을 제공합니다.

AWS Command Line Interface(AWS CLI)을 사용하여 삭제 마커 복제를 활성화하려면 다음 예제 구성과 같이 `DeleteMarkerReplication`을 활성화한 소스 버킷에 복제 구성을 추가해야 합니다.

다음 예제 복제 구성에서 접두사 *`Tax`* 아래의 객체에 대해 삭제 마커가 대상 버킷 *`amzn-s3-demo-destination-bucket`*에 복제됩니다.

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "DeleteMarkerReplication": {
                "Status": "Enabled"
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::amzn-s3-demo-destination-bucket"
            },
            "Priority": 1
```

```
        }
    ],
    "Role": "IAM-Role-ARN"
}
```

AWS CLI를 통한 복제 규칙 만들기에 대한 전체 지침은 동일한 계정에서 버킷 복제 구성 섹션을 참조하세요.

라이브 복제 관리 또는 일시 중지

라이브 복제는 동일한 리전 또는 서로 다른 AWS 리전의 버킷 간에 객체를 비동기식으로 자동 복사하는 것을 말합니다. 복제 구성을 설정하면 Amazon S3는 새로 생성된 객체 및 객체 업데이트를 원본 버킷에서 지정된 하나 이상의 대상 버킷으로 복제합니다.

Amazon S3 콘솔을 사용하면 원본 버킷에 복제 규칙을 추가할 수 있습니다. 복제 규칙은 복제할 원본 버킷 객체와 복제된 객체가 저장된 대상 버킷을 정의합니다. 복제에 대한 자세한 내용은 리전 내 및 리전 간 객체 복제 섹션을 참조하십시오.

Amazon S3 콘솔의 복제 페이지에서 복제 규칙을 관리할 수 있습니다. 복제 규칙은 추가, 확인, 편집, 활성화, 비활성화 또는 삭제할 수 있습니다. 복제 규칙의 우선순위를 변경할 수도 있습니다. 복제 규칙을 버킷에 추가하는 것에 관한 자세한 내용은 S3 콘솔 사용 단원을 참조하십시오.

Amazon S3 콘솔을 사용하여 버킷의 복제 규칙을 관리하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 범용 버킷 탭에서 확인하려는 버킷의 이름을 선택합니다.
4. 관리 탭을 선택한 다음 아래로 스크롤하여 복제 규칙으로 이동합니다.
5. 복제 규칙은 다음과 같은 방법으로 변경합니다.
    - 복제 규칙을 활성화하거나 비활성화하려면 규칙 왼쪽에 있는 옵션 버튼을 선택합니다. 작업 메뉴에서 규칙 활성화 또는 규칙 비활성화를 선택합니다. 작업 메뉴의 해당 버킷에 있는 모든 규칙들을 사용 중지 및 사용 설정하거나 삭제할 수도 있습니다.

> **Note**
>
> 복제 규칙을 비활성화했다가 나중에 다시 활성화하는 경우, 규칙이 비활성화된 동안 복제되지 않은 새 객체나 변경된 객체는 규칙을 다시 활성화해도 자동으로 복제되지

않습니다. 이러한 객체를 복제하려면 S3 배치 복제를 사용해야 합니다. 자세한 내용은 the section called “기존 객체 복제” 섹션을 참조하세요.

- 규칙의 우선 순위를 변경하려면 규칙 왼쪽에 있는 옵션 버튼을 선택한 다음 규칙 편집을 선택합니다.

  여러 규칙의 범위에 포함된 객체로 인해 발생하는 충돌을 방지하기 위해 규칙 우선 순위를 설정합니다. 중첩 규칙의 경우, Amazon S3은 규칙 우선 순위를 사용하여 어느 규칙을 적용할지 결정합니다. 숫자가 클수록 우선 순위가 높아집니다. 규칙 우선 순위에 대한 자세한 내용은 복제 구성 파일 요소 단원을 참조하세요.

복제 일시 중지 또는 중지

복제를 일시 중지했다가 나중에 자동으로 재개하도록 하려면 AWS Fault Injection Service에서 `aws:s3:bucket-pause-replication` 작업을 사용할 수 있습니다. 자세한 내용은 AWS Fault Injection Service 사용 설명서의 aws:s3:bucket-pause-replication 및 S3 복제 일시 중지를 참조하세요.

Amazon S3에서 복제를 중지하려면 복제 규칙을 비활성화하는 것이 좋습니다. 복제 규칙을 비활성화했다가 나중에 다시 활성화하는 경우, 규칙이 비활성화된 동안 복제되지 않은 새 객체나 변경된 객체는 규칙을 다시 활성화해도 자동으로 복제되지 않습니다. 이러한 객체를 복제하려면 S3 배치 복제를 사용해야 합니다. 자세한 내용은 the section called “기존 객체 복제” 섹션을 참조하세요.

AWS Identity and Access Management(IAM) 역할, AWS Key Management Service(AWS KMS) 권한 또는 Amazon S3에 필요한 권한을 부여하는 버킷 정책 권한을 제거하는 경우에도 복제가 중지됩니다. 그러나 이러한 방법을 사용하면 복제가 실패하므로 권장되지 않습니다. Amazon S3는 영향을 받는 객체의 복제 상태를 `FAILED`(으)로 보고합니다. 나중에 권한이 복원되는 경우 `FAILED`로 표시된 객체는 자동으로 복제되지 않습니다. 이러한 객체를 복제하려면 S3 배치 복제를 사용해야 합니다.

배치 복제를 사용한 기존 객체 복제

S3 배치 복제는 Amazon S3 버킷에서 새 객체를 지속적으로 자동으로 복사하는 실시간 복제와 다릅니다. 대신 S3 배치 복제는 기존 객체에서 요청 시 수행됩니다. S3 배치 복제를 사용하여 다음 유형의 객체를 복제할 수 있습니다.

- 복제 구성이 적용되기 전에 존재했던 객체
- 이전에 복제된 객체
- 복제에 실패한 객체

배치 작업을 사용하여 이러한 객체를 온디맨드로 복제할 수 있습니다.

다음을 수행하여 배치 복제를 시작할 수 있습니다.

- 새 복제 규칙 또는 대상에 대해 배치 복제 시작 - 새 복제 구성에서 첫 번째 규칙을 만들 때 또는 Amazon S3 콘솔을 통해 새 대상 버킷을 기존 구성에 추가할 때 일회성 배치 복제 작업을 만들 수 있습니다.
- 기존 복제 구성에 대해 배치 복제 시작 - Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), AWS SDK, 또는 Amazon S3 REST API를 통해 S3 Batch Operations를 사용하여 새 배치 복제 작업을 만들 수 있습니다.

배치 복제 작업이 완료되면 완료 보고서를 수신합니다. 보고서를 사용하여 작업을 검사하는 방법에 대한 자세한 정보는 작업 상태 및 완료 보고서 추적 섹션을 참조하세요.

S3 배치 복제 고려 사항

S3 배치 복제를 사용하기 전에 다음 고려 사항 목록을 검토합니다.

- 소스 버킷에는 기존 복제 구성이 있어야 합니다. 복제를 사용하려면 라이브 복제 설정 개요 및 라이브 복제 구성 예제 섹션을 참조하세요.
- 버킷에 대해 S3 수명 주기를 구성한 경우 배치 복제 작업이 활성 상태인 동안 수명 주기 규칙을 사용 중지하는 것이 좋습니다. 이렇게 하면 소스 버킷과 대상 버킷 간의 동등성이 보장됩니다. 이렇게 하지 않을 경우 두 버킷이 어긋날 수 있으며, 소스 버킷의 정확한 복제본인 대상 버킷을 만들 수 없습니다. 예를 들어 다음 시나리오를 고려해 보세요:
  - 소스 버킷에는 객체 및 객체의 삭제 마커에 대한 버전이 여러 개 있습니다.
  - 소스 및 대상 버킷에는 만료된 삭제 마커를 제거하는 수명 주기 구성이 있습니다.

  이 시나리오에서 배치 복제는 객체 버전을 복제하기 전에 삭제 마커를 대상 버킷에 복제할 수 있습니다. 이로 인해 수명 주기 구성이 삭제 마커를 만료된 것으로 표시하고 객체 버전이 복제되기 전에 대상 버킷에서 제거될 수 있습니다.
- 배치 작업을 실행하기 위해 지정하는 AWS Identity and Access Management(IAM) 역할에는 기본 배치 작업을 수행하는 데 필요한 권한이 있어야 합니다. IAM 역할 생성에 대한 자세한 내용은 S3 배치 복제를 위한 IAM 역할 구성 섹션을 참조하세요.
- 배치 복제에는 Amazon S3에서 생성할 수 있는 매니페스트가 필요합니다. 생성된 매니페스트는 소스 버킷과 동일한 AWS 리전에 저장되어야 합니다. 매니페스트를 생성하지 않는 경우 복제하려는 객체가 포함된 Amazon S3 인벤토리 보고서 또는 CSV 파일을 제공할 수 있습니다. 자세한 내용은 the section called “배치복제 작업에 대한 매니페스트 지정” 섹션을 참조하세요.

- 배치 복제는 대상 버킷에서 객체의 버전 ID를 지정하여 삭제한 객체를 다시 복제하는 것을 지원하지 않습니다. 이러한 객체를 다시 복제하려면 배치 복사 작업을 사용하여 소스 객체를 제자리에 복사할 수 있습니다. 이러한 객체를 제자리에 복사하면 소스 버킷에 객체의 새 버전이 생성되고 대상 버킷에 대한 복제가 자동으로 시작됩니다. 대상 버킷을 삭제하고 다시 생성하면 복제가 시작되지 않습니다.

  배치 복사에 대한 자세한 내용은 배치 작업을 사용하여 객체를 복사하는 예 섹션을 참조하세요.
- 소스 버킷에서 복제 규칙을 사용하는 경우 복제 구성을 업데이트하여 복제 규칙에 연결된 IAM 역할에 객체 복제를 위한 적절한 권한을 부여해야 합니다. 이 IAM 역할에는 소스 및 대상 버킷 모두에서 복제 작업을 수행하는 데 필요한 권한이 있어야 합니다.
- 짧은 시간 내에 동일한 버킷에 대해 여러 배치 복제 작업을 제출하면 Amazon S3는 해당 작업을 동시에 실행합니다.
- 서로 다른 두 버킷에 대해 여러 배치 복제 작업을 제출하는 경우 Amazon S3에서 모든 작업을 동시에 실행하지 않을 수 있다는 점을 유의하세요. 계정에서 한 번에 실행할 수 있는 배치 복제 작업 수를 초과할 경우 Amazon S3는 우선순위가 낮은 작업을 일시 중지하고 우선순위가 높은 작업을 진행합니다. 우선순위가 높은 작업이 완료되면 일시 중지된 모든 작업이 다시 활성화됩니다.
- 배치 복제는 S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive 스토리지 클래스에 저장된 객체에 대해서는 지원되지 않습니다.
- Archive Access 또는 Deep Archive Access 스토리지 계층에 저장된 S3 Intelligent-Tiering 객체를 배치 복제하려면 먼저 복원 요청을 시작하고 객체가 Frequent Access 계층으로 이동할 때까지 기다려야 합니다.
- 단일 배치 복제 작업은 최대 200억 개의 객체가 포함된 매니페스트를 지원할 수 있습니다.

배치복제 작업에 대한 매니페스트 지정

매니페스트는 Amazon S3가 작업을 수행할 객체 키를 포함하는 Amazon S3 객체입니다. 배치 복제 작업을 생성하려면 사용자 생성 매니페스트를 제공하거나 Amazon S3가 복제 구성을 기반으로 매니페스트를 생성하도록 해야 합니다.

사용자 생성 매니페스트를 제공하는 경우 Amazon S3 인벤토리 보고서 또는 CSV 파일 형식이어야 합니다. 매니페스트의 객체가 버전 지정된 버킷에 있는 경우 객체의 버전 ID를 지정해야 합니다. 매니페스트에 지정된 버전 ID를 가진 객체만 복제됩니다. 매니페스트 지정에 대한 자세한 내용은 매니페스트 지정 섹션을 참조하세요.

Amazon S3가 사용자를 대신하여 매니페스트 파일을 생성하도록 선택하면 나열된 객체는 소스 버킷의 복제 구성과 동일한 소스 버킷, 접두사 및 태그를 사용하게 됩니다. 매니페스트가 생성되면 Amazon S3가 모든 적합한 버전의 객체를 복제합니다.

> **Note**
>
> Amazon S3가 매니페스트를 생성하도록 하는 경우 매니페스트가 소스 버킷과 동일한 AWS 리전에 저장되어야 합니다.

배치 복제 작업에 대한 필터

배치 복제 작업을 생성할 때 선택적으로 객체 생성 날짜 및 복제 상태와 같은 추가 필터를 지정하여 작업 범위를 줄일 수 있습니다.

다음 값 중 하나 이상을 제공하여 `ObjectReplicationStatuses` 값을 기반으로 복제할 객체를 필터링할 수 있습니다.

- `"NONE"` - Amazon S3가 이전에 객체 복제를 시도한 적이 없음을 나타냅니다.
- `"FAILED"` - Amazon S3가 이전에 객체 복제를 시도했지만 실패했음을 나타냅니다.
- `"COMPLETED"` - Amazon S3가 이전에 객체를 성공적으로 복제했음을 나타냅니다.
- `"REPLICA"` - 이 객체가 Amazon S3가 다른 소스 버킷에서 복제한 복제본임을 나타냅니다.

복제 상태에 자세한 내용은 복제 상태 정보 가져오기 섹션을 참조하세요.

배치 복제 작업을 필터링하지 않으면 Batch Operations는 기본적으로 복제되지 않는 특정 객체를 제외하고 복제 구성의 규칙과 매칭되는 매니페스트의 모든 객체(`ObjectReplicationStatus` 무관)를 복제하려고 시도합니다. 자세한 내용은 the section called “복제 구성으로 복제되지 않는 것은 무엇입니까?” 섹션을 참조하세요.

목표에 따라 `ObjectReplicationStatuses`를 다음 값 중 하나 이상으로 설정할 수 있습니다.

- 복제된 적이 없는 기존 객체만 복제하려면 `"NONE"`만 포함합니다.
- 이전에 복제에 실패한 객체만 복제를 다시 시도하려는 경우 `"FAILED"`만 포함합니다.
- 기존 객체를 복제하고 이전에 복제에 실패한 객체 복제를 다시 시도하려면 `"NONE"` 및 `"FAILED"`를 모두 포함합니다.
- 다른 대상에 복제된 객체로 대상 버킷을 다시 채우려면 `"COMPLETED"`를 포함합니다.
- 이전에 복제된 객체를 복제하려면 `"REPLICA"`를 포함합니다.

배치 복제 완료 보고서

배치 복제 작업을 생성할 때 CSV 완료 보고서를 요청할 수 있습니다. 이 보고서에는 객체, 복제 성공 또는 실패 코드, 출력 및 설명이 표시됩니다. 작업 추적 및 완료 보고서에 대한 자세한 내용은 완료 보고서 섹션을 참조하세요.

복제 실패 코드 및 설명 목록은 Amazon S3 복제 실패 이유 섹션을 참조하세요.

배치 복제의 문제 해결에 대한 자세한 내용은 배치 복제 오류 섹션을 참조하세요.

배치 복제 시작

배치 복제를 사용하는 방법에 대해 자세히 알아보려면 자습서: S3 배치 복제를 사용하여 Amazon S3 버킷의 기존 객체 복제를 참조하세요.

S3 배치 복제를 위한 IAM 역할 구성

Amazon S3 배치 복제는 Batch Operations 작업의 일종이므로, 사용자를 대신하여 작업을 수행할 수 있는 Batch Operations 권한을 부여하려면 AWS Identity and Access Management(IAM) 역할을 만들어야 합니다. 또한 배치 복제 IAM 정책을 배치 작업 IAM 역할에 연결해야 합니다.

다음 절차에 따라 배치 복제 작업을 시작할 수 있는 Batch Operations 권한을 부여하는 정책과 IAM 역할을 만듭니다.

배치 복제 정책을 만들려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 액세스 관리(Access management)에서 정책(Policies)을 선택합니다.
3. 정책 생성을 선택합니다.
4. 권한 지정 페이지에서 JSON을 선택합니다.
5. Amazon S3에서 매니페스트를 생성했는지 아니면 자체 매니페스트를 제공하는지에 따라 다음 정책 중 하나를 삽입합니다. 매니페스트에 관한 자세한 내용은 배치복제 작업에 대한 매니페스트 지정 단원을 참조하십시오.

   이러한 정책을 사용하기 전에 다음 정책의 *user input placeholders*를 복제 소스 버킷, 매니페스트 버킷 및 완료 보고서 버킷의 이름으로 바꿉니다.

> **Note**
>
> 배치 복제의 IAM 역할에는 매니페스트를 생성하는지 아니면 제공하는지에 따라 다른 권한이 필요하므로, 다음 예제에서 적절한 정책을 선택해야 합니다.

Amazon S3 생성 매니페스트를 사용 및 저장하는 경우의 정책

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:InitiateReplication"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
         ]
      },
      {
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:PutInventoryConfiguration"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket"
         ]
      },
      {
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
         ]
      },
      {
```

```
         "Effect":"Allow",
         "Action":[
            "s3:PutObject"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*",
            "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
         ]
      }
   ]
}
```

사용자 제공 매니페스트를 사용하는 경우의 정책

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:InitiateReplication"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
         ]
      },
            {
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*"
```

```
            ]
        }
    ]
}
```

6. 다음을 선택합니다.
7. 정책 이름을 지정한 다음 정책 생성을 선택합니다.

배치 복제를 위한 IAM 역할을 만들려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 액세스 관리에서 역할을 선택합니다.
3. 역할 생성을 선택합니다.
4. 신뢰할 수 있는 엔터티 유형으로 AWS 서비스를 선택합니다. 사용 사례 섹션에서 S3를 서비스로, S3 Batch Operations를 사용 사례로 선택합니다.
5. 다음을 선택합니다. 권한 추가 페이지가 나타납니다. 검색 상자에서 이전 절차에서 만든 정책을 검색합니다. 정책 이름 옆에 있는 확인란을 선택하고 다음을 선택합니다.
6. 이름, 검토 및 만들기 페이지에서 IAM 역할의 이름을 지정합니다.
7. 1단계: 신뢰 ID 섹션에서 IAM 역할이 다음 신뢰 정책을 사용하고 있는지 확인합니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

8. 2단계: 권한 추가 섹션에서 IAM 역할이 이전에 만든 정책을 사용하고 있는지 확인합니다.
9. 역할 생성을 선택합니다.

새 복제 규칙 또는 대상에 대한 배치 복제 작업 만들기

Amazon S3에서 라이브 복제는 복제 구성을 만들기 전에 소스 버킷에 이미 존재했던 객체를 복제하지 않습니다. 라이브 복제는 복제 구성이 생성된 후 버킷에 기록된 새 객체와 업데이트된 객체만 자동으로 복제합니다. 이미 존재하는 객체를 복제하려면 S3 배치 복제를 사용하여 요청 시 복제할 수 있습니다.

새로운 라이브 복제 구성의 첫 번째 규칙을 만들거나 Amazon S3 콘솔을 통해 기존 복제 구성에 새 대상 버킷을 추가할 때는 필요에 따라 배치 복제 작업을 만들 수 있습니다. 이 배치 복제 작업을 사용하여 소스 버킷의 기존 객체를 대상 버킷에 복제할 수 있습니다.

새 대상 버킷을 추가하지 않고 기존 구성에 배치 복제를 사용하려면 기존 복제 규칙에 대한 배치 복제 작업 생성 섹션을 참조하세요.

사전 조건

배치 복제 작업을 만들기 전에 Batch Operations AWS Identity and Access Management(IAM) 역할을 만들어 사용자를 대신해서 작업을 수행할 수 있는 Amazon S3 권한을 부여해야 합니다. 자세한 내용은 S3 배치 복제를 위한 IAM 역할 구성 섹션을 참조하세요.

Amazon S3 콘솔을 통해 새 복제 규칙 또는 대상에 대한 배치 복제 사용

새 복제 구성의 첫 번째 규칙을 만들거나 Amazon S3 콘솔을 통해 기존 구성에 새 대상 버킷을 추가할 때 소스 버킷의 기존 객체를 복제하는 배치 복제 작업을 만들도록 선택할 수 있습니다.

복제 구성을 만들거나 업데이트할 때 배치 복제 작업을 만들려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 범용 버킷 목록에서 복제할 객체가 포함된 버킷 이름을 선택합니다.
4. 새 복제 규칙을 만들거나 기존 규칙을 편집하려면 관리 탭을 선택하고 아래로 스크롤하여 복제 규칙으로 이동합니다.
   - 새 복제 규칙을 생성하려면 Create replication rule(복제 규칙 생성)을 선택합니다. 기본 복제 규칙을 설정하는 방법에 대한 예제는 라이브 복제 구성 예제 섹션을 참조하세요.
   - 기존 복제 규칙을 편집하려면 규칙 이름 옆에 있는 옵션 버튼을 선택한 다음 규칙 편집을 선택합니다.
5. 새 복제 규칙을 생성하거나 기존 복제 규칙의 대상을 편집하고 Save(저장)를 선택합니다.

새 복제 구성에서 첫 번째 규칙을 생성하거나 기존 구성을 편집하여 새 대상을 추가하면 Replicate existing objects?(기존 객체 복제?) 대화 상자에 배치 복제 작업을 생성할 수 있는 옵션이 제공됩니다.

6. 지금 이 작업을 만들고 실행하려면 예, 기존 객체를 복제합니다를 선택합니다.

   나중에 배치 복제 작업을 만들려는 경우 아니요, 기존 객체를 복제하지 않습니다를 선택합니다.

7. 예, 기존 객체를 복제합니다를 선택한 경우 Batch Operations 작업 만들기 페이지가 나타납니다. S3 배치 복제 작업에는 다음과 같은 설정이 있습니다.

   작업 실행 옵션

   S3 배치 복제 작업을 즉시 실행하려면 준비가 되면 자동으로 작업 실행을 선택합니다. 나중에 작업을 실행하려면 준비가 되면 작업 실행 대기를 선택합니다.

   > **Note**
   >
   > 준비가 되면 자동으로 작업 실행을 선택한 경우 Batch Operations 매니페스트를 만들고 저장할 수 없습니다. Batch Operations 매니페스트를 저장하려면 준비가 되면 작업 실행 대기를 선택합니다.

   배치 작업 매니페스트

   준비가 되면 작업 실행 대기를 선택한 경우 Batch Operations 매니페스트 섹션이 나타납니다. 매니페스트는 지정된 작업을 실행해야 하는 모든 객체의 목록입니다. 매니페스트를 저장하도록 선택할 수 있습니다. S3 인벤토리 파일과 마찬가지로 매니페스트는 CSV 파일로 저장되고 버킷에 저장됩니다. 배치 작업 매니페스트에 대한 자세한 내용은 매니페스트 지정 섹션을 참조하세요.

   완료 보고서

   S3 Batch Operations는 매니페스트에 지정된 각 객체에 대해 하나의 태스크를 실행합니다. 완료 보고서를 사용하면 추가 설정을 하지 않아도 통합된 형식으로 작업 결과를 손쉽게 볼 수 있습니다. 모든 작업 또는 실패한 작업에만 대해 완료 보고서를 요청할 수 있습니다. 완료 보고서에 대한 자세한 내용은 완료 보고서 섹션을 참조하세요.

권한

복제 실패의 가장 일반적인 원인 중 하나는 제공된 AWS Identity and Access Management(IAM) 역할의 사용 권한이 충분하지 않은 것입니다. 이 역할을 생성하는 방법에 대한 상세 정보는 S3 배치 복제를 위한 IAM 역할 구성을(를) 참조하세요. 배치 복제에 필요한 권한을 가진 IAM 역할을 만들거나 선택해야 합니다.

8. 저장을 선택합니다.

기존 복제 규칙에 대한 배치 복제 작업 생성

Amazon S3에서 라이브 복제는 복제 구성을 만들기 전에 소스 버킷에 이미 존재했던 객체를 복제하지 않습니다. 라이브 복제는 복제 구성이 생성된 후 버킷에 기록된 새 객체와 업데이트된 객체만 자동으로 복제합니다. 이미 존재하는 객체를 복제하려면 S3 배치 복제를 사용하여 요청 시 복제할 수 있습니다.

AWS SDK, AWS Command Line Interface(AWS CLI) 또는 Amazon S3 콘솔을 사용하여 기존 복제 구성에 대해 S3 배치 복제를 구성할 수 있습니다. 배치 복제에 대한 개요는 배치 복제를 사용한 기존 객체 복제 섹션을 참조하세요.

배치 복제 작업이 완료되면 완료 보고서를 수신합니다. 보고서를 사용하여 작업을 검사하는 방법에 대한 자세한 정보는 작업 상태 및 완료 보고서 추적 섹션을 참조하세요.

사전 조건

배치 복제 작업을 만들기 전에 Batch Operations AWS Identity and Access Management(IAM) 역할을 만들어 사용자를 대신해서 작업을 수행할 수 있는 Amazon S3 권한을 부여해야 합니다. 자세한 내용은 S3 배치 복제를 위한 IAM 역할 구성 섹션을 참조하세요.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Batch Operations를 선택합니다.
3. 작업 생성을 선택합니다.
4. AWS 리전 섹션에 작업을 만들려는 리전이 표시되는지 확인합니다.
5. 매니페스트 섹션에서 사용하려는 매니페스트 형식을 지정합니다. 매니페스트는 지정된 작업을 실행해야 하는 모든 객체의 목록입니다. 배치 작업 매니페스트에 대한 자세한 내용은 매니페스트 지정 섹션을 참조하세요.

- 매니페스트를 준비한 경우 S3 인벤토리 보고서(manifest.json) 또는 CSV를 선택합니다. 매니페스트가 버전이 지정된 버킷에 있는 경우 매니페스트의 버전 ID를 지정할 수 있습니다. 버전 ID를 지정하지 않으면 Batch Operations에서는 매니페스트의 현재 버전을 사용합니다. 매니페스트 생성에 대한 자세한 내용은 매니페스트 지정 섹션을 참조하세요.

  > **Note**
  >
  > 매니페스트의 객체가 버전 지정된 버킷에 있는 경우 객체의 버전 ID를 지정해야 합니다. 자세한 내용은 매니페스트 지정 섹션을 참조하세요.

- 복제 구성을 기반으로 매니페스트를 생성하려면 S3 복제 구성을 사용한 매니페스트 생성(Create manifest using S3 Replication Configuration)을 선택합니다. 그런 다음 복제 구성의 소스 버킷을 선택합니다.

6. (선택 사항) S3 복제 구성을 사용하여 매니페스트 만들기를 선택한 경우 객체를 만든 날짜 및 복제 상태와 같은 추가 필터를 포함할 수 있습니다. 복제 상태별로 필터링하는 방법에 대한 예제는 배치 복제 작업에 대한 매니페스트 지정 섹션을 참조하세요.
7. (선택 사항) S3 복제 구성을 사용하여 매니페스트 만들기를 선택한 경우 생성된 매니페스트를 저장할 수 있습니다. 이 매니페스트를 저장하려면 Batch Operations 매니페스트 저장을 선택합니다. 그런 다음 매니페스트의 대상 버킷을 지정하고 매니페스트를 암호화할지 여부를 선택합니다.

   > **Note**
   >
   > 생성된 매니페스트는 소스 버킷과 동일한 AWS 리전에 저장되어야 합니다.

8. 다음을 선택합니다.
9. 작업 페이지에서 복제를 선택한 후 다음을 선택합니다.
10. (선택 사항) 설명을 입력합니다.
11. 필요한 경우 작업의 우선 순위를 조정합니다. 숫자가 높을 수록 우선순위가 높아집니다. Amazon S3는 우선 순위가 낮은 작업에 앞서 우선 순위가 높은 작업을 실행하려고 시도합니다. 작업 우선 순위에 대한 자세한 내용은 작업 우선 순위 지정 섹션을 참조하세요.
12. (선택 사항) 완료 보고서를 생성합니다. 이 보고서를 생성하려면 완료 보고서 생성을 선택합니다.

    완료 보고서를 생성하도록 선택한 경우 실패한 작업만(Failed tasks only) 또는 모든 작업(All tasks)을 선택하고 보고서의 대상 버킷을 제공해야 합니다.

13. 권한 섹션에서 배치 복제에 필요한 권한이 있는 IAM 역할을 선택해야 합니다. 복제가 실패하는 가장 흔한 이유 중 하나는 제공된 IAM 역할의 사용 권한이 충분하지 않기 때문입니다. 이 역할을 생성하는 방법에 대한 상세 정보는 S3 배치 복제를 위한 IAM 역할 구성을(를) 참조하세요.
14. (선택 사항) 배치 복제 작업에 작업 태그를 추가합니다.
15. 다음을 선택합니다.
16. 작업 구성을 살펴본 후 작업 만들기를 선택합니다.

S3 매니페스트와 AWS CLI 사용

다음 예제 `create-job` 명령에서는 AWS 계정 *111122223333*에 대해 S3 생성 매니페스트를 사용하여 S3 배치 복제 작업을 생성합니다. 이 예제에서는 기존 객체와 이전에 복제하지 못한 객체를 복제합니다. 복제 상태별 필터링에 대한 자세한 내용은 배치복제 작업에 대한 매니페스트 지정 섹션을 참조하세요.

이 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체합니다. `role/`*batch-Replication-IAM-policy* IAM 역할을 이전에 만든 IAM 역할로 대체합니다. 자세한 내용은 S3 배치 복제를 위한 IAM 역할 구성 섹션을 참조하세요.

```
aws s3control create-job --account-id 111122223333 \
--operation '{"S3ReplicateObject":{}}' \
--report '{"Bucket":"arn:aws:s3:::amzn-s3-demo-completion-report-bucket",\
"Prefix":"batch-replication-report", \
"Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}' \
--manifest-generator '{"S3JobManifestGenerator": {"ExpectedBucketOwner":
 "111122223333", \
"SourceBucket": "arn:aws:s3:::amzn-s3-demo-source-bucket", \
"EnableManifestOutput": false, "Filter": {"EligibleForReplication": true, \
"ObjectReplicationStatuses": ["NONE","FAILED"]}}}' \
--priority 1 \
--role-arn arn:aws:iam::111122223333:role/batch-Replication-IAM-policy \
--no-confirmation-required \
--region source-bucket-region
```

> **Note**
>
> 복제 소스 버킷과 동일한 AWS 리전에서 작업을 시작해야 합니다.

배치복제 작업을 성공적으로 시작한 후에는 작업 ID를 응답으로 받게 됩니다. 다음 `describe-job` 명령을 사용하여 이 작업을 모니터링할 수 있습니다. 이 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체합니다.

```
aws s3control describe-job --account-id 111122223333 --job-id job-id --region source-bucket-region
```

사용자 제공 매니페스트와 AWS CLI 사용

다음 예제에서는 AWS 계정 *111122223333*에 대해 사용자 정의 매니페스트를 사용하여 S3 배치 복제 작업을 생성합니다. 매니페스트의 객체가 버전 지정된 버킷에 있는 경우 객체의 버전 ID를 지정해야 합니다. 매니페스트에 지정된 버전 ID를 가진 객체만 복제됩니다. 매니페스트 생성에 대한 자세한 내용은 매니페스트 지정 섹션을 참조하세요.

이 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체합니다. `role/`*batch-Replication-IAM-policy* IAM 역할을 이전에 만든 IAM 역할로 대체합니다. 자세한 내용은 S3 배치 복제를 위한 IAM 역할 구성 섹션을 참조하세요.

```
aws s3control create-job --account-id 111122223333 \
--operation '{"S3ReplicateObject":{}}' \
--report '{"Bucket":"arn:aws:s3:::amzn-s3-demo-completion-report-bucket",\
"Prefix":"batch-replication-report", \
"Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}' \
--manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820",\
"Fields":["Bucket","Key","VersionId"]},\
"Location":{"ObjectArn":"arn:aws:s3:::amzn-s3-demo-manifest-bucket/manifest.csv",\
"ETag":"Manifest Etag"}}' \
--priority 1 \
--role-arn arn:aws:iam::111122223333:role/batch-Replication-IAM-policy \
--no-confirmation-required \
--region source-bucket-region
```

> **Note**
>
> 복제 소스 버킷과 동일한 AWS 리전에서 작업을 시작해야 합니다.

배치복제 작업을 성공적으로 시작한 후에는 작업 ID를 응답으로 받게 됩니다. 다음 `describe-job` 명령을 사용하여 이 작업을 모니터링할 수 있습니다.

```
aws s3control describe-job --account-id 111122223333 --job-id job-id --region source-bucket-region
```

복제 문제 해결

이 섹션에는 Amazon S3 복제에 대한 문제 해결 팁과 S3 배치 복제 오류에 대한 정보가 나와 있습니다.

주제

- S3 복제 문제 해결 팁
- 배치 복제 오류

S3 복제 문제 해결 팁

복제를 구성한 후 대상 버킷에 객체 복제본이 표시되지 않는다면 다음 문제 해결 도움말을 사용하여 문제를 해결합니다.

- 대다수의 객체는 15분 이내에 복제됩니다. Amazon S3가 객체를 복제하는 데 걸리는 시간은 소스 및 대상 리전 쌍, 객체 크기를 비롯한 다양한 요소에 따라 달라집니다. 큰 객체의 경우 복제에 몇 시간이 걸릴 수도 있습니다. 복제 시간에 대한 가시성을 확보하려면 S3 Replication Time Control(S3 RTC)을 사용하면 됩니다.

  복제하는 객체가 크다면 얼마간 기다렸다 대상에 해당 객체가 나타나는지 확인하세요. 또한 소스 객체의 복제 상태를 확인할 수도 있습니다. 객체 복제 상태가 `PENDING`이면 Amazon S3가 복제를 완료하지 않은 것입니다. 객체 복제 상태가 `FAILED`인 경우 소스 버킷에서 설정된 복제 구성을 확인하세요.

  또한 복제 중에 실패에 대한 정보를 받으려면 Amazon S3 이벤트 알림 복제를 설정하여 실패 이벤트를 받을 수 있습니다. 자세한 내용은 Amazon S3 이벤트 알림을 사용하여 복제 실패 이벤트 수신을 참조하세요.
- 객체의 복제 상태를 확인하려면 `HeadObject` API 작업을 직접 호출하면 됩니다. `HeadObject` API 작업은 객체의 `PENDING`, `COMPLETED` 또는 `FAILED` 복제 상태를 반환합니다. `HeadObject` API 직접 호출에 대한 응답으로 `x-amz-replication-status` 헤더에 복제 상태가 반환됩니다.

> **Note**
>
> `HeadObject`를 실행하려면 요청하는 객체에 대한 읽기 액세스 권한이 있어야 합니다. `HEAD` 요청에는 `GET` 작업 수행을 제외하고 `GET` 요청과 동일한 옵션이 있습니다. 예를 들어, AWS

Command Line Interface(AWS CLI)를 사용하여 HeadObject 요청을 실행하려면 다음 명령을 실행하면 됩니다. *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws s3api head-object --bucket amzn-s3-demo-source-bucket --key index.html
```

- HeadObject가 FAILED 복제 상태의 객체를 반환하는 경우 S3 배치 복제를 사용하여 실패한 객체를 복제할 수 있습니다. 자세한 내용은 the section called “기존 객체 복제” 섹션을 참조하세요. 또는 실패한 객체를 소스 버킷에 다시 업로드해도 됩니다. 이렇게 하면 새 객체에 대한 복제가 시작됩니다.
- 원본 버킷의 복제 구성에서 다음을 확인합니다.
  - 대상 버킷의 Amazon 리소스 이름(ARN)이 정확합니다.
  - 키 이름 접두사가 정확합니다. 예를 들어, 접두사 Tax로 객체 복제를 구성한 경우라면 Tax/document1 또는 Tax/document2와 같이 키 이름이 포함된 객체만 복제됩니다. 키 이름이 document3인 객체는 복제되지 않습니다.
  - 복제 규칙의 상태는 Enabled입니다.
- 복제 구성의 어떤 버킷에서도 버전 관리가 일시 중단되지 않았는지 확인합니다. 소스 버킷과 대상 버킷 모두에서 버전 관리를 활성화해야 합니다.
- 복제 규칙이 대상 버킷 소유자로 객체 소유권 변경으로 설정된 경우 복제에 사용되는 AWS Identity and Access Management(IAM) 역할에 s3:ObjectOwnerOverrideToBucketOwner 권한이 있어야 합니다. 이 권한은 리소스(이 경우 대상 버킷)에 부여됩니다. 예를 들어, 다음 Resource 명령문은 대상 버킷에 이 권한을 부여하는 방법을 보여줍니다.

```
{
  "Effect":"Allow",
  "Action":[
    "s3:ObjectOwnerOverrideToBucketOwner"
  ],
  "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
}
```

- 다른 계정이 대상 버킷을 소유한 경우, 대상 버킷 소유자가 대상 버킷 정책을 통해 소스 버킷 소유자에게 s3:ObjectOwnerOverrideToBucketOwner 권한도 부여해야 합니다. 다음 예시 버킷 정책을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
{
  "Version": "2012-10-17",
  "Id": "Policy1644945280205",
```

```
  "Statement": [
    {
      "Sid": "Stmt1644945277847",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789101:role/s3-replication-role"
      },
      "Action": [
        "s3:ReplicateObject",
        "s3:ReplicateTags",
        "s3:ObjectOwnerOverrideToBucketOwner"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
    }
  ]
}
```

> **Note**
>
> 대상 버킷의 객체 소유권 설정에 버킷 소유자 적용이 포함된 경우 복제 규칙에서 객체 소유권을 대상 버킷 소유자로 변경으로 설정을 업데이트할 필요가 없습니다. 객체 소유권 변경은 기본적으로 발생합니다. 복제본 소유권 변경에 대한 자세한 내용은 복제본 소유자 변경을 참조하세요.

- 소스 버킷과 대상 버킷을 서로 다른 AWS 계정에서 소유하는 교차 계정 시나리오에서 복제 구성을 설정하는 경우, 대상 버킷을 요청자 지불 버킷으로 구성할 수 없습니다. 자세한 내용은 스토리지 전송 및 사용량에 대한 요청자 지불 범용 버킷 사용 섹션을 참조하세요.
- 버킷의 소스 객체가 AWS Key Management Service(AWS KMS) 키를 사용한 서버 측 암호화(SSE-KMS)로 암호화된 경우, 복제 규칙은 AWS KMS로 암호화된 객체를 포함하도록 구성되어야 합니다. Amazon S3 콘솔의 암호화 설정에서 AWS KMS로 암호화된 객체 복제를 선택해야 합니다. 그런 다음, 대상 객체를 암호화하는 데 사용할 AWS KMS 키를 선택합니다.

> **Note**
>
> 대상 버킷이 다른 계정에 있는 경우 대상 계정에서 소유하는 AWS KMS 고객 관리형 키를 지정하세요. 기본 Amazon S3 관리 키(aws/s3)를 사용하지 마세요. 기본 키를 사용하면 소스 계정이 소유한 Amazon S3 관리형 키로 객체를 암호화하여 객체가 다른 계정과 공유되지 않도록 합니다. 따라서 대상 계정은 대상 버킷의 객체에 액세스할 수 없습니다.

대상 계정에 속한 AWS KMS 키를 사용하여 대상 객체를 암호화하려면 대상 계정이 KMS 키 정책에서 복제 역할에 kms:GenerateDataKey 및 kms:Encrypt 권한을 부여해야 합니다. KMS 키 정책에서 다음 예시 명령문을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
{
    "Sid": "AllowS3ReplicationSourceRoleToUseTheKey",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::123456789101:role/s3-replication-role"
    },
    "Action": ["kms:GenerateDataKey", "kms:Encrypt"],
    "Resource": "*"
}
```

AWS KMS 키 정책의 Resource 명령문에 별표(*)를 사용하는 경우 정책은 복제 역할에만 KMS 키를 사용할 수 있는 권한을 부여합니다. 정책은 복제 역할의 권한 상승을 허용하지 않습니다.

기본적으로 KMS 키 정책은 루트 사용자에게 키에 대한 전체 권한을 부여합니다. 이러한 권한은 동일한 계정의 다른 사용자에게 위임할 수 있습니다. 소스 KMS 키 정책에 Deny 명령문이 없는 한 IAM 정책을 사용하여 소스 KMS 키에 복제 역할 권한을 부여하는 것만으로도 충분합니다.

> **Note**
>
> 특정 CIDR 범위, 가상 프라이빗 클라우드(VPC) 엔드포인트 또는 S3 액세스 포인트에 대한 액세스를 제한하는 KMS 키 정책으로 인해 복제가 실패할 수 있습니다.

소스 또는 대상 KMS 키가 암호화 컨텍스트에 따라 권한을 부여하는 경우 Amazon S3 버킷 키가 버킷에 대해 활성화되어 있는지 확인하세요. 버킷에 S3 버킷 키가 켜져 있는 경우 암호화 컨텍스트는 다음과 같이 버킷 수준 리소스여야 합니다.

```
"kms:EncryptionContext:arn:aws:arn": [
     "arn:aws:s3:::amzn-s3-demo-source-bucket"
     ]
"kms:EncryptionContext:arn:aws:arn": [
     "arn:aws:s3:::amzn-s3-demo-destination-bucket"
     ]
```

소스 계정은 KMS 키 정책에서 부여한 권한 외에도 복제 역할의 IAM 정책에 다음과 같은 최소 권한을 추가해야 합니다.

```
{
    "Effect": "Allow",
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": [
        "Source-KMS-Key-ARN"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "kms:GenerateDataKey",
        "kms:Encrypt"
    ],
    "Resource": [
        "Destination-KMS-Key-ARN"
    ]
}
```

AWS KMS로 암호화된 객체를 복제하는 방법에 대한 자세한 내용은 암호화된 객체 복제를 참조하세요.

- 다른 AWS 계정이 대상 버킷을 소유한 경우, 해당 대상 버킷에 소스 버킷 소유자의 객체 복제를 허용하는 버킷 정책이 있는지 확인합니다. 문제 해결 예는 서로 다른 계정의 버킷에 대한 복제 구성을(를) 참조하세요.
- 복제에 객체 잠금을 사용하려면 복제 설정에 사용하는 AWS Identity and Access Management(IAM) 역할에서 소스 S3 버킷에 두 가지 권한을 추가로 부여해야 합니다. 두 가지 추가 권한은 `s3:GetObjectRetention`과 `s3:GetObjectLegalHold`입니다. 역할에 `s3:Get*` 권한 문이 있으면 해당 문이 요구 사항을 충족합니다. 자세한 내용은 the section called “S3 복제에 객체 잠금 사용” 섹션을 참조하세요.
- 권한을 검증한 후에도 객체가 여전히 복제되지 않는 경우 다음 위치에 명시적인 `Deny` 명령문이 있는지 확인하세요.
  - 소스 또는 대상 정책의 `Deny` 명령문. 버킷 정책이 다음 작업 중 하나에 대해 복제 역할에 대한 액세스를 거부하면 복제가 실패합니다.

원본 버킷:

```
            "s3:GetReplicationConfiguration",
            "s3:ListBucket",
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"
```

대상 버킷:

```
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ReplicateTags"
```

  - IAM 역할에 연결된 Deny 명령문 또는 권한 경계로 인해 복제가 실패할 수 있습니다.
  - 소스 또는 대상 계정에 연결된 AWS Organizations 서비스 제어 정책(SCP)의 Deny 문으로 인해 복제가 실패할 수 있습니다.
  - 소스 또는 대상 버킷에 연결된 AWS Organizations 리소스 제어 정책(RCP)의 Deny 문으로 인해 복제가 실패할 수 있습니다.
- 객체 복제본이 대상 버킷에 나타나지 않는 경우, 다음 문제로 복제가 차단된 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 S3는 버킷 B에 있는 객체 복제본을 버킷 C로 복제하지 않습니다.
  - 소스 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. 이 문제를 예방하기 위해 소스 버킷 소유자는 다른 AWS 계정에 해당 객체에 대한 명시적 액세스 권한을 조건부로 요구하는 객체 만들기 권한을 부여할 수 있습니다. 정책 예제는 버킷 소유자가 완벽하게 제어할 수 있도록 보증하면서 객체에 업로드하는 크로스 계정 권한 부여을 참조하세요.
- 복제 구성에서 객체 중 특정 태그를 갖는 하위 집합을 복제하는 규칙을 추가한다고 가정해 봅시다. 이 경우 Amazon S3가 객체를 복제하도록 하려면 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.
- Amazon S3 이벤트 알림을 사용하여 객체가 대상 AWS 리전으로 복제되지 않는 경우에 알림을 받으세요. Amazon S3 이벤트 알림은 Amazon Simple Queue Service(Amazon SQS), Amazon Simple

Notification Service(Amazon SNS) 또는 AWS Lambda를 통해 사용할 수 있습니다. 자세한 내용은 Amazon S3 이벤트 알림을 사용하여 복제 실패 이벤트 수신 섹션을 참조하세요.

Amazon S3 이벤트 알림을 사용하여 복제 실패 이유도 볼 수 있습니다. 실패 이유 목록을 검토하려면 Amazon S3 복제 실패 이유를 참조하세요.

배치 복제 오류

대상 버킷에 복제되지 않는 객체 문제를 해결하려면 배치 복제 작업을 만드는 데 사용되는 버킷, 복제 역할, IAM 역할에 대한 다양한 권한 유형을 확인하세요. 또한, 버킷의 퍼블릭 액세스 차단 설정과 S3 객체 소유권 설정을 확인해야 합니다.

Batch Operations 작업에 대한 추가 문제 해결 팁은 the section called “Batch Operations 문제 해결” 섹션을 참조하세요.

복제를 설정했는데 객체가 복제되지 않는 경우 AWS re:Post 지식 센터의 버킷 간에 복제를 설정해도 Amazon S3 객체가 복제되지 않는 이유는 무엇입니까?를 참조하세요.

배치 복제 사용 중에 다음 오류 중 하나가 발생할 수 있습니다.

- 매니페스트 생성에서 필터 기준과 일치하는 키를 찾지 못함.

  이 오류는 다음과 같은 이유로 발생할 수 있습니다.

  - 소스 버킷이 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스에 저장된 경우.

    이러한 객체에 배치 복제를 사용하려면 먼저 Batch Operations 작업에서 복원(`S3InitiateRestoreObjectOperation`) 작업을 사용하여 S3 Standard 스토리지 클래스로 복원해야 합니다. 자세한 내용은 아카이브된 객체 복원 및 객체 복원(배치 작업)을 참조하세요. 객체를 복원한 후에는 배치 복제 작업을 사용하여 객체를 복제할 수 있습니다.

  - 제공된 필터 기준이 소스 버킷의 유효한 객체와 일치하지 않는 경우.

    필터 기준을 확인하고 수정합니다. 예를 들어, 배치 복제 규칙에서 필터 기준은 소스 버킷에서 `Tax/` 접두사가 있는 모든 객체를 찾는 것입니다. 접두사 이름을 잘못 입력한 경우, 슬래시가 끝에만 있는 것이 아니라 처음과 끝에 있는 경우(`/Tax/`) S3 객체를 찾을 수 없습니다. 오류를 해결하려면 복제 규칙에서 접두사를 수정하세요(이 경우 `/Tax/`에서 `Tax/`로).

- 다음과 같은 이유로 배치 작업 상태가 실패했습니다. 작업 보고서를 보고서 버킷에 기록할 수 없습니다.

이 오류는 배치 작업에 사용되는 IAM 역할이 작업을 생성할 때 지정한 위치에 완료 보고서를 넣을 수 없을 때 발생합니다. 이 오류를 해결하려면 IAM 역할에 Batch Operations 완료 보고서를 저장하려는 버킷에 대한 `s3:PutObject` 권한이 있는지 확인하세요. 소스 버킷과 다른 버킷에 보고서를 전달하는 것이 좋습니다.

이 오류 해결에 대한 추가 팁은 the section called “권한 문제가 있거나 보존 모드가 활성화된 경우 작업 보고서가 전달되지 않음” 섹션을 참조하세요.

- 배치 작업이 실패로 완료되었으며 총 실패 개수가 0이 아닙니다.

  이 오류는 실행 중인 배치 복제 작업에 객체 사용 권한이 충분하지 않은 경우 발생합니다. 배치 복제 작업에 복제 규칙을 사용하는 경우 복제에 사용되는 IAM 역할에 소스 또는 대상 버킷의 객체에 액세스할 수 있는 적절한 권한이 있는지 확인하세요. 또한 배치 복제 완료 보고서를 확인하여 특정 Amazon S3 복제 실패 이유를 검토할 수도 있습니다.

- 배치 작업이 성공적으로 실행되었지만 대상 버킷에 예상되는 객체 수가 동일하지 않습니다.

  이 오류는 배치 복제 작업에 제공된 매니페스트에 나열된 객체와 작업을 만들 때 선택한 필터가 일치하지 않을 때 발생합니다. 소스 버킷의 객체와 매칭되는 복제 규칙이 없고 생성된 매니페스트에 소스 버킷의 객체가 포함되지 않은 경우에도 이 메시지가 표시될 수 있습니다.

기존 복제 구성에 새 복제 규칙을 추가한 후 배치 작업 오류가 발생함

배치 작업은 소스 버킷의 복제 구성에 있는 모든 규칙에 대해 기존 객체 복제를 시도합니다. 기존 복제 규칙에 문제가 있는 경우 오류가 발생할 수 있습니다.

배치 작업 작업의 완료 보고서에 작업 실패 이유가 설명되어 있습니다. 일반적인 오류 목록은 Amazon S3 복제 실패 이유 섹션을 참조하세요.

지표, 이벤트 알림 및 상태를 통한 복제 모니터링

다음 메커니즘을 통해 라이브 복제 구성과 S3 배치 복제 작업을 모니터링할 수 있습니다.

- S3 복제 지표 - S3 복제 지표를 활성화하면 Amazon CloudWatch는 복제 규칙 수준에서 보류 중인 바이트, 보류 중인 작업, 복제 지연 시간을 추적하는 데 사용할 수 있는 지표를 내보냅니다. Amazon S3 콘솔과 Amazon CloudWatch 콘솔을 통해 S3 복제 지표를 볼 수 있습니다. Amazon S3 콘솔에 있는 소스 버킷의 지표 탭에서 이러한 지표를 볼 수 있습니다. S3 복제 지표에 대한 자세한 내용은 the section called “S3 복제 지표 사용” 섹션을 참조하세요.

- S3 Storage Lens 지표 - S3 복제 지표 외에도 S3 Storage Lens 대시보드에서 제공하는 복제 관련 데이터 보호 지표를 사용할 수 있습니다. 예를 들어, S3 Storage Lens의 무료 지표를 사용하면 소스 버킷에서 복제된 총 바이트 수 또는 소스 버킷에서 복제된 객체 수 등의 지표를 볼 수 있습니다.

  S3 Storage Lens에서 고급 지표를 활성화하여 전체 복제 상태를 감사할 수 있습니다. S3 Storage Lens의 고급 지표를 사용하면 복제 대상이 유효하지 않은 복제 규칙 수를 포함한 여러 유형의 복제 규칙 수를 확인할 수 있습니다.

  S3 Storage Lens에서 복제 지표를 사용하는 방법에 대한 자세한 내용은 the section called “S3 Storage Lens 대시보드에서 복제 지표 보기” 섹션을 참조하세요.

- S3 이벤트 알림 - S3 이벤트 알림은 객체가 대상 AWS 리전으로 복제되지 않거나 특정 임계값 내에서 객체가 복제되지 않은 경우 객체 수준에서 알림을 보낼 수 있습니다. S3 이벤트 알림은 `s3:Replication:OperationFailedReplication`, `s3:Replication:OperationMissedThreshold`, `s3:Replication:OperationReplicatedAfterThreshold`, `s3:Replication:OperationNotTracked` 등의 복제 이벤트 유형을 제공합니다.

  Amazon S3 이벤트는 Amazon Simple Queue Service(Amazon SQS), Amazon Simple Notification Service(Amazon SNS) 또는 AWS Lambda을 통해 사용할 수 있습니다. 자세한 내용은 the section called “복제 실패 이벤트 수신” 섹션을 참조하세요.

- 복제 상태 값 - 객체의 복제 상태를 검색할 수도 있습니다. 복제 상태는 복제 중인 객체의 현재 상태를 확인하는 데 도움이 됩니다. 원본 객체의 복제 상태는 `PENDING`, `COMPLETED`, `FAILED` 중 하나를 반환합니다. 복제본의 복제 상태가 `REPLICA`을(를) 반환합니다.

  S3 배치 복제 작업을 만들 때 복제 상태 값을 사용할 수도 있습니다. 예를 들어, 이러한 상태 값을 사용하면 복제된 적이 없거나 복제에 실패한 객체를 복제할 수 있습니다.

  객체의 복제 상태를 검색하는 방법에 대한 자세한 내용은 the section called “복제 상태 가져오기” 섹션을 참조하세요. 배치 복제에서 이러한 값을 사용하는 방법에 대한 자세한 내용은 the section called “배치 복제 작업에 대한 필터” 섹션을 참조하세요.

주제

- S3 복제 지표 사용
- S3 Storage Lens 대시보드에서 복제 지표 보기
- Amazon S3 이벤트 알림을 사용하여 복제 실패 이벤트 수신
- 복제 상태 정보 가져오기

S3 복제 지표 사용

S3 복제 지표는 복제 구성의 복제 규칙에 대한 세부 지표를 제공합니다. 복제 지표를 사용하면 보류 중인 바이트, 보류 중인 작업, 복제 실패 작업, 복제 지연 시간을 추적하여 진행률을 분 단위로 모니터링할 수 있습니다.

> **Note**
>
> - S3 복제 지표는 Amazon CloudWatch 사용자 지정 지표와 동일한 요금으로 청구됩니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.
> - S3 Replication Time Control을 사용하는 경우 Amazon CloudWatch는 각 복제 규칙에서 S3 RTC를 활성화한 후 15분이 지나면 복제 지표 보고를 시작합니다.

S3 Replication Time Control(S3 RTC)을 활성화하면 S3 복제 지표가 자동으로 설정됩니다. 규칙을 만들거나 편집하는 동안 S3 RTC와 독립적으로 S3 복제 지표를 활성화할 수도 있습니다. S3 RTC에는 SLA(서비스 수준 계약) 및 누락된 임계값에 대한 알림 같은 기타 기능이 포함되어 있습니다. 자세한 내용은 S3 Replication Time Control을 사용하여 규정 준수 요구 사항 충족 섹션을 참조하세요.

S3 복제 지표가 활성화되면 Amazon S3는 다음 지표를 Amazon CloudWatch에 게시합니다. CloudWatch 지표를 전송하고자 최선을 다할 것입니다.

메트릭 이름	지표 설명	이 지표는 어떤 객체에 적용되나요?	이 지표는 어느 리전에 게시되나요?	대상 버킷이 삭제된 경우에도 지표가 계속 게시되나요?	복제가 이루어지지 않는 경우에도 지표가 계속 게시되나요?
복제 보류 중인 바이트	지정된 복제 규칙에 대해 복제 보류 중인 객체의 총 바이트 수입니다.	이 지표는 S3 Cross-Region Replication(S3 CRR) 또는 S3 Same-Region Replication(S3 SRR)으로 복제되	이 지표는 대상 버킷의 리전에 게시됩니다.	아니요	예

메트릭 이름	지표 설명	이 지표는 어떤 객체에 적용되나요?	이 지표는 어느 리전에 게시되나요?	대상 버킷이 삭제된 경우에도 지표가 계속 게시되나요?	복제가 이루어지지 않는 경우에도 지표가 계속 게시되나요?
		는 새 객체에만 적용됩니다.			
복제 지연 시간	지정된 복제 규칙에 대해 복제 대상 버킷이 소스 버킷보다 늦어지는 최대 시간(초)입니다.	이 지표는 S3 CRR 또는 S3 SRR로 복제된 새 객체에만 적용됩니다.	이 지표는 대상 버킷의 리전에 게시됩니다.	아니요	예
복제 보류 중인 작업	지정된 복제 규칙에 대해 복제 보류 중인 작업 수입니다. 이 지표는 객체, 삭제 마커, 태그, 액세스 제어 목록(ACL), S3 객체 잠금과 관련된 작업을 추적합니다.	이 지표는 S3 CRR 또는 S3 SRR로 복제된 새 객체에만 적용됩니다.	이 지표는 대상 버킷의 리전에 게시됩니다.	아니요	예

메트릭 이름	지표 설명	이 지표는 어떤 객체에 적용되나요?	이 지표는 어느 리전에 게시되나요?	대상 버킷이 삭제된 경우에도 지표가 계속 게시되나요?	복제가 이루어지지 않는 경우에도 지표가 계속 게시되나요?
복제 실패 작업	지정된 복제 규칙에 대해 복제에 실패한 작업 수입니다. 이 지표는 객체, 삭제 마커, 태그, 액세스 제어 목록(ACL), Object Lock과 관련된 작업을 추적합니다. 복제 실패 작업은 분 간격으로 집계된 S3 복제 실패를 추적합니다. 복제에 실패한 특정 객체와 실패 이유를 파악하려면 Amazon S3 이벤트 알림에서 `OperationFailedReplication` 이벤트를 구독하세요. 자	이 지표는 S3 CRR 또는 S3 SRR로 복제된 새 객체와 S3 배치 복제를 통해 복제된 기존 객체를 모두 추적합니다. **Note** S3 배치 복제 작업이 전혀 실행되지 않으면 지표가 Amazon CloudWatch로 전송되지 않습니다. 예를 들어,	이 지표는 소스 버킷의 리전에 게시됩니다.	예	아니요

메트릭 이름	지표 설명	이 지표는 어떤 객체에 적용되나요?	이 지표는 어느 리전에 게시되나요?	대상 버킷이 삭제된 경우에도 지표가 계속 게시되나요?	복제가 이루어지지 않는 경우에도 지표가 계속 게시되나요?
	세한 내용은 Amazon S3 이벤트 알림을 사용하여 복제 실패 이벤트 수신 섹션을 참조하세요.	S3 배치 복제 작업을 실행하는 데 필요한 권한이 없거나 복제 구성의 태그 또는 접두사가 일치하지 않는 경우 작업이 실행되지 않습니다.			

CloudWatch에서 이러한 지표를 사용하는 방법에 대한 자세한 내용은 the section called “CloudWatch의 S3 복제 지표” 섹션을 참조하세요.

S3 복제 지표 활성화

새 복제 규칙이나 기존 복제 규칙과 함께 S3 복제 지표를 사용할 수 있습니다. 복제 규칙 생성에 대한 전체 지침은 동일한 계정에서 버킷 복제 구성 단원을 참조하세요. 복제 규칙을 전체 S3 버킷에 적용하거나 특정 접두사 또는 태그가 있는 Amazon S3 객체에 적용하도록 선택할 수 있습니다.

이 주제에서는 소스 및 대상 버킷을 동일하거나 서로 다른 AWS 계정이 소유한 경우에 복제 구성에서 S3 복제 지표를 사용하는 지침을 제공합니다.

AWS Command Line Interface(AWS CLI)를 사용하여 복제 지표를 활성화하려면 `Metrics` 사용 상태의 소스 버킷에 복제 구성을 추가해야 합니다. 이 예시 구성에서 접두사 *`Tax`* 아래의 객체는 대상 버킷 *`amzn-s3-demo-bucket`*에 복제되고 해당 객체에 대한 지표가 생성됩니다.

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
                "Metrics": {
                    "Status": "Enabled"
                }
            },
            "Priority": 1
        }
    ],
    "Role": "IAM-Role-ARN"
}
```

복제 지표 보기

Amazon S3 콘솔에 있는 소스 범용 버킷의 지표 탭에서 S3 복제 지표를 볼 수 있습니다. 이러한 Amazon CloudWatch 지표는 Amazon CloudWatch 콘솔에서도 사용할 수 있습니다. S3 복제 지표를 활성화하면 Amazon CloudWatch는 복제 규칙 수준에서 보류 중인 바이트, 보류 중인 작업, 복제 지연 시간을 추적하는 데 사용할 수 있는 지표를 내보냅니다.

S3 복제 지표는 Amazon S3 콘솔 또는 Amazon S3 REST API를 사용하여 S3 Replication Time Control(S3 RTC)을 통해 복제를 활성화하면 자동으로 설정됩니다. 규칙을 만들거나 편집하는 동안 S3 RTC와 독립적으로 S3 복제 지표를 활성화할 수도 있습니다.

S3 Replication Time Control을 사용하는 경우 Amazon CloudWatch는 각 복제 규칙에서 S3 RTC를 활성화한 후 15분이 지나면 복제 지표 보고를 시작합니다. 자세한 내용은 S3 복제 지표 사용 섹션을 참조하세요.

복제 지표는 복제 구성의 규칙 ID를 추적합니다. 복제 규칙 ID는 접두사, 태그 또는 둘의 조합에 대해 고유할 수 있습니다.

Amazon S3용 CloudWatch 지표에 대한 자세한 내용은 Amazon CloudWatch를 사용한 지표 모니터링 섹션을 참조하십시오.

사전 조건

S3 복제 지표가 활성화된 복제 규칙을 만듭니다. 자세한 내용은 the section called “복제 지표 사용 설정” 섹션을 참조하세요.

소스 버킷의 지표 탭을 통해 S3 복제 지표를 보려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 요청 지표를 구성하려는 객체가 있는 소스 버킷의 이름을 선택합니다.
4. 지표 탭을 선택합니다.
5. 복제 지표에서 지표를 보려는 복제 규칙을 선택합니다.
6. 차트 표시를 선택합니다.

   Amazon S3는 선택한 규칙에 대한 복제 지연 시간, 복제 보류 중인 바이트, 복제 보류 중인 작업, 복제 실패 작업 차트를 표시합니다.

S3 Storage Lens 대시보드에서 복제 지표 보기

S3 복제 지표 외에도 S3 Storage Lens에서 제공하는 복제 관련 데이터 보호 지표를 사용할 수 있습니다. S3 스토리지 렌즈는 조직 전반에서 객체 스토리지 사용 및 활동에 대한 가시성을 확보하는 데 사용할 수 있는 클라우드 스토리지 분석 기능입니다. 자세한 내용은 Using S3 Storage Lens to protect your data(S3 스토리지 렌즈를 사용한 데이터 보호)를 참조하세요.

S3 Storage Lens에는 무료 지표와 고급 지표 및 권장 사항이라는 두 계층의 지표가 있는데, 추가 비용을 지불하고 업그레이드할 수 있습니다. 고급 지표 및 권장 사항을 통해 추가 지표 및 기능에 액세스하여 스토리지에 대한 인사이트를 얻을 수 있습니다. S3 스토리지 렌즈 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

S3 Storage Lens의 무료 지표를 사용하면 소스 버킷에서 복제된 총 바이트 수 또는 소스 버킷에서 복제된 객체 수 등의 지표를 볼 수 있습니다.

S3 Storage Lens에서 고급 지표를 활성화하여 전체 복제 상태를 감사할 수 있습니다. S3 Storage Lens의 고급 지표를 사용하면 복제 대상이 유효하지 않은 복제 규칙 수를 포함한 여러 유형의 복제 규칙 수를 확인할 수 있습니다.

각 계층에 있는 복제 지표를 포함한 S3 Storage Lens 지표의 전체 목록은 S3 Storage Lens 지표 용어집을 참조하세요.

사전 조건

라이브 복제 구성 또는 S3 배치 복제 작업을 만듭니다.

Amazon S3 Storage Lens에서 복제 지표를 보려면

1. S3 Storage Lens 대시보드를 만듭니다. 단계별 지침은 the section called “S3 콘솔 사용”섹션을 참조하세요.
2. (선택 사항) 대시보드 설정 중에 모든 S3 Storage Lens 복제 지표를 보려면 고급 지표 및 권장 사항을 선택한 다음 고급 데이터 보호 지표를 선택합니다. 지표의 전체 목록은 S3 Storage Lens 지표 용어집을 참조하세요.

   고급 지표 및 권장 사항을 활성화하면 복제 구성을 더 자세히 파악할 수 있습니다. 예를 들어, S3 Storage Lens 복제 규칙 수 지표를 사용하여 복제용으로 구성된 버킷에 대한 자세한 정보를 얻을 수 있습니다. 이 정보에는 버킷 및 리전 내부 및 리전 간의 복제 규칙이 포함됩니다. 자세한 내용은 the section called “각 버킷의 총 복제 규칙 수 계산” 섹션을 참조하세요.
3. 대시보드를 만든 후 대시보드를 열고 버킷 탭을 선택합니다.
4. 아래로 스크롤하여 Buckets(버킷) 섹션을 찾습니다. Metrics categories(지표 범주)에서 Data protection(데이터 보호)을 선택합니다. 그런 다음 Summary(요약)를 지웁니다.
5. 복제 지표만 표시하도록 버킷 목록을 필터링하려면 기본 설정 아이콘 ()을 선택합니다.
6. 복제 지표만 선택된 상태가 될 때까지 모든 데이터 보호 지표에 대한 토글을 지웁니다.

7. (선택 사항) Page size(페이지 크기)에서 목록에 표시할 버킷 수를 선택합니다.
8. 계속을 선택합니다.

Amazon S3 이벤트 알림을 사용하여 복제 실패 이벤트 수신

복제 구성에서 S3 복제 지표를 활성화한 경우 객체가 대상 AWS 리전으로 복제되지 않을 때 알리도록 Amazon S3 이벤트 알림을 설정할 수 있습니다. 복제 구성에서 S3 Replication Time Control(S3 RTC)을 활성화한 경우, 복제에 대한 15분 S3 RTC 임계값 내에 객체가 복제되지 않을 때도 알림을 받을 수 있습니다.

다음 `Replication` 이벤트 유형을 사용하여 보류 중인 바이트, 보류 중인 작업, 복제 지연 시간을 추적하면서 복제 이벤트의 진행 상황을 분 단위로 모니터링할 수 있습니다. S3 복제 지표에 대한 자세한 내용은 S3 복제 지표 사용 섹션을 참조하세요.

- `s3:Replication:OperationFailedReplication` 이벤트 유형은 복제에 적합한 객체가 복제에 실패할 때 알려줍니다.
- `s3:Replication:OperationMissedThreshold` 이벤트 유형은 S3 RTC를 사용하는 복제 대상 객체가 복제 임계값인 15분을 초과하면 알려줍니다.
- `s3:Replication:OperationReplicatedAfterThreshold` 이벤트 유형은 S3 RTC를 사용하는 복제 대상 객체가 15분 임계값 이후에 복제되면 알려줍니다.
- `s3:Replication:OperationNotTracked` 이벤트 유형은 라이브 복제 대상 객체(동일 리전 복제[SRR] 또는 교차 리전 복제[CRR])가 더 이상 복제 지표로 추적되지 않으면 알려줍니다.

지원되는 모든 복제 이벤트 유형에 대한 자세한 설명은 the section called “SQS, SNS 및 Lambda에 지원되는 이벤트 유형” 섹션을 참조하세요.

S3 이벤트 알림에서 캡처한 실패 코드 목록은 Amazon S3 복제 실패 이유 섹션을 참조하세요.

Amazon Simple Queue Service(Amazon SQS), Amazon Simple Notification Service(Amazon SNS) 또는 AWS Lambda를 통해 S3 이벤트 알림을 받을 수 있습니다. 자세한 내용은 the section called “Amazon S3 이벤트 알림” 섹션을 참조하세요.

Amazon S3 이벤트 알림을 구성하는 방법에 대한 지침은 이벤트 알림 사용 설정를 참조하세요.

> **Note**
>
> 이벤트 알림뿐만 아니라 S3 복제 지표도 활성화해야 합니다. 자세한 내용은 the section called "복제 지표 사용 설정" 섹션을 참조하세요.

다음은 Amazon S3에서 `s3:Replication:OperationFailedReplication` 이벤트를 게시하기 위해 전송하는 메시지의 예입니다. 자세한 내용은 the section called "이벤트 메시지 구조" 섹션을 참조하세요.

```
{
  "Records": [
    {
      "eventVersion": "2.2",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "2024-09-05T21:04:32.527Z",
      "eventName": "Replication:OperationFailedReplication",
      "userIdentity": {
        "principalId": "s3.amazonaws.com"
      },
      "requestParameters": {
        "sourceIPAddress": "s3.amazonaws.com"
      },
      "responseElements": {
        "x-amz-request-id": "123bf045-2b4b-4ca8-a211-c34a63c59426",
        "x-amz-id-2":
 "12VAWNDIHnwJsRhTccqQTeAPoXQmRt22KkewMV8G3XZihAuf9CLDdmkApgZzudaIe2KlLfDqGS0="
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "ReplicationEventName",
        "bucket": {
          "name": "amzn-s3-demo-bucket1",
          "ownerIdentity": {
            "principalId": "111122223333"
          },
          "arn": "arn:aws:s3:::amzn-s3-demo-bucket1"
        },
        "object": {
          "key": "replication-object-put-test.png",
          "size": 520080,
```

```
          "eTag": "e12345ca7e88a38428305d3ff7fcb99f",
          "versionId": "abcdeH0Xp66ep__QDjR76LK7Gc9X4wKO",
          "sequencer": "0066DA1CBF104C0D51"
        }
      },
      "replicationEventData": {
        "replicationRuleId": "notification-test-replication-rule",
        "destinationBucket": "arn:aws:s3:::amzn-s3-demo-bucket2",
        "s3Operation": "OBJECT_PUT",
        "requestTime": "2024-09-05T21:03:59.168Z",
        "failureReason": "AssumeRoleNotPermitted"
      }
    }
  ]
}
```


Amazon S3 복제 실패 이유

다음 테이블에는 Amazon S3 Replication 실패 이유가 나열되어 있습니다. Amazon S3 이벤트 알림이 포함된 `s3:Replication:OperationFailedReplication` 이벤트를 수신한 후 `failureReason` 값을 보면 그 이유를 알 수 있습니다.

S3 배치 복제 완료 보고서에서도 이러한 실패 이유를 확인할 수 있습니다. 자세한 내용은 배치 복제 완료 보고서 섹션을 참조하세요.

복제 실패 이유	설명
`AssumeRoleNotPermitted`	Amazon S3가 복제 구성 또는 배치 작업에 지정된 AWS Identity and Access Managemen t(IAM) 역할을 맡을 수 없습니다.
`DstBucketInvalidRegion`	대상 버킷의 위치가 배치 작업에 지정된 AWS 리전과 동일하지 않습니다. 이 오류는 배치 복제에만 해당합니다.
`DstBucketNotFound`	Amazon S3가 복제 구성에 지정된 대상 버킷을 찾을 수 없습니다.
`DstBucketObjectLockConfigMissing`	객체 잠금이 활성화된 소스 버킷에서 객체를 복제하려면 대상 버킷에도 객체 잠금이 활성화되어 있어야 합니다. 이 오류는 대상 버킷에서

복제 실패 이유	설명
	객체 잠금이 활성화되지 않았을 수 있다는 것을 나타냅니다. 자세한 내용은 객체 잠금 고려 사항 섹션을 참조하세요.
DstBucketUnversioned	S3 대상 버킷에서 버전 관리가 활성화되지 않았습니다. S3 복제로 객체를 복제하려면 대상 버킷에서 버전 관리를 활성화합니다.
DstDelObjNotPermitted	Amazon S3가 대상 버킷으로 삭제 마커를 복제할 수 없습니다. 대상 버킷에 대한 s3:ReplicateDelete 권한이 누락되었을 수 있습니다.
DstKmsKeyInvalidState	대상 버킷의 AWS Key Management Service(AWS KMS) 키가 유효하지 않은 상태입니다. 필요한 AWS KMS 키를 검토하고 활성화합니다. AWS KMS 키 관리에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 AWS KMS 키의 키 상태를 참조하세요.
DstKmsKeyNotFound	복제 구성에서 대상 버킷에 대해 구성된 AWS KMS 키가 존재하지 않습니다.
DstMultipartCompleteNotPermitted	Amazon S3가 대상 버킷에 있는 객체의 멀티파트 업로드를 완료할 수 없습니다. 대상 버킷에 대한 s3:ReplicateObject 권한이 누락되었을 수 있습니다.
DstMultipartInitNotPermitted	Amazon S3가 대상 버킷으로 객체의 멀티파트 업로드를 시작할 수 없습니다. 대상 버킷에 대한 s3:ReplicateObject 권한이 누락되었을 수 있습니다.

복제 실패 이유	설명
DstMultipartUploadNotPermitted	Amazon S3가 대상 버킷으로 멀티파트 업로드 객체를 업로드할 수 없습니다. 대상 버킷에 대한 s3:ReplicateObject 권한이 누락되었을 수 있습니다.
DstObjectHardDeleted	S3 배치 복제가 대상 버킷에서 객체의 버전 ID로 삭제된 객체를 다시 복제하는 것을 지원하지 않습니다. 이 오류는 배치 복제에만 해당합니다.
DstPutAclNotPermitted	Amazon S3가 대상 버킷으로 객체 액세스 제어 목록(ACL)을 복제할 수 없습니다. 대상 버킷에 대한 s3:ReplicateObject 권한이 누락되었을 수 있습니다.
DstPutLegalHoldNotPermitted	Amazon S3가 변경 불가능한 객체를 복제하는 동안 대상 객체에 Object Lock 법적 보류를 적용할 수 없습니다. 대상 버킷에 대한 s3:PutObjectLegalHold 권한이 누락되었을 수 있습니다. 자세한 내용은 법적 보존 섹션을 참조하세요.
DstPutObjectNotPermitted	Amazon S3가 대상 버킷으로 객체를 복제할 수 없습니다. 이는 대상 버킷에 필요한 권한(s3:ReplicateObject 또는 s3:ObjectOwnerOverrideToBucketOwner 권한)이 없거나 AWS KMS 키 정책에서 소스 버킷의 복제 역할이 대상 버킷에서 AWS KMS 키(kms:Decrypt 및 kms:GenerateDataKey* 작업)를 사용하도록 허용하지 않는 경우에 발생할 수 있습니다.

복제 실패 이유	설명
DstPutRetentionNotPermitted	Amazon S3가 변경 불가능한 객체를 복제하는 동안 대상 객체에 보존 기간을 적용할 수 없습니다. 대상 버킷에 대한 s3:PutObjectRetention 권한이 누락되었을 수 있습니다.
DstPutTaggingNotPermitted	Amazon S3가 대상 버킷으로 객체 태그를 복제할 수 없습니다. 대상 버킷에 대한 s3:ReplicateObject 권한이 누락되었을 수 있습니다.
DstVersionNotFound	Amazon S3가 대상 버킷에서 메타데이터를 복제해야 하는 필요한 객체 버전을 찾을 수 없습니다.
InitiateReplicationNotPermitted	Amazon S3가 객체에 복제를 시작할 수 없습니다. 배치 작업에 대한 s3:InitiateReplication 권한이 누락되었을 수 있습니다. 이 오류는 배치 복제에만 해당합니다.
SrcBucketInvalidRegion	소스 버킷의 위치가 배치 작업에 지정된 AWS 리전과 동일하지 않습니다. 이 오류는 배치 복제에만 해당합니다.
SrcBucketNotFound	Amazon S3가 소스 버킷을 찾을 수 없습니다.
SrcBucketReplicationConfigMissing	Amazon S3가 소스 버킷의 복제 구성을 찾을 수 없습니다.

복제 실패 이유	설명
SrcGetAclNotPermitted	Amazon S3가 복제를 위해 소스 버킷의 객체에 액세스할 수 없습니다. 소스 버킷 객체에 대한 s3:GetObjectVersionAcl 권한이 누락되었을 수 있습니다. 소스 버킷의 객체는 버킷 소유자가 소유하고 있어야 합니다. ACL이 사용 설정되어 있는 경우, 객체 소유권이 버킷 소유자 기본 설정 또는 객체 작성자로 설정되어 있는지 확인합니다. 객체 소유권이 버킷 소유자 선호로 설정된 경우, 버킷 소유자가 객체 소유자가 되려면 소스 버킷 객체에 bucket-owner-full-control ACL이 있어야 합니다. 소스 계정은 객체 소유권을 버킷 소유자 적용으로 설정하고 ACL을 사용 중지하여 버킷의 모든 객체에 대한 소유권을 가져올 수 있습니다.
SrcGetLegalHoldNotPermitted	Amazon S3가 S3 객체 잠금 법적 보존 정보에 액세스할 수 없습니다.
SrcGetObjectNotPermitted	Amazon S3가 복제를 위해 소스 버킷의 객체에 액세스할 수 없습니다. 소스 버킷에 대한 s3:GetObjectVersionForReplication 권한이 누락되었을 수 있습니다.
SrcGetRetentionNotPermitted	Amazon S3가 S3 객체 잠금 보존 기간 정보에 액세스할 수 없습니다.
SrcGetTaggingNotPermitted	Amazon S3가 소스 버킷의 객체 태그 정보에 액세스할 수 없습니다. 소스 버킷에 대한 s3:GetObjectVersionTagging 권한이 누락되었을 수 있습니다.

복제 실패 이유	설명
SrcHeadObjectNotPermitted	Amazon S3가 소스 버킷에서 객체 메타데이터를 검색할 수 없습니다. 소스 버킷에 대한 s3:GetObjectVersionForReplication 권한이 누락되었을 수 있습니다.
SrcKeyNotFound	Amazon S3가 복제할 소스 객체 키를 찾을 수 없습니다. 복제가 완료되기 전에 소스 객체가 삭제되었을 수 있습니다.
SrcKmsKeyInvalidState	소스 버킷의 AWS KMS 키가 유효하지 않은 상태입니다. 필요한 AWS KMS 키를 검토하고 활성화합니다. AWS KMS 키 관리에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 AWS KMS 키의 키 상태를 참조하세요.
SrcObjectNotEligible	일부 객체가 복제 요건을 충족하지 않습니다. 이는 객체의 스토리지 클래스 또는 객체 태그가 복제 구성과 일치하지 않기 때문일 수 있습니다.
SrcObjectNotFound	소스 객체가 존재하지 않습니다.
SrcReplicationNotPending	Amazon S3에서 이미 이 객체를 복제했습니다. 이 객체가 더 이상 복제 보류 중이 아닙니다.
SrcVersionNotFound	Amazon S3가 복제할 소스 객체 버전을 찾을 수 없습니다. 복제가 완료되기 전에 소스 객체 버전이 삭제되었을 수 있습니다.

관련 주제

라이브 복제에 대한 권한 설정

복제 문제 해결

복제 상태 정보 가져오기

복제 상태는 복제 중인 객체의 현재 상태를 확인하는 데 도움이 될 수 있습니다. 원본 객체의 복제 상태는 PENDING, COMPLETED, FAILED 중 하나를 반환합니다. 복제본의 복제 상태가 REPLICA을(를) 반환합니다.

S3 배치 복제 작업을 만들 때 복제 상태 값을 사용할 수도 있습니다. 예를 들어, 이러한 상태 값을 사용하면 복제된 적이 없거나 복제에 실패한 객체를 복제할 수 있습니다. 배치 복제에서 이러한 값을 사용하는 방법에 대한 자세한 내용은 the section called “배치 복제 작업에서 복제 상태 정보 사용” 섹션을 참조하세요.

주제

- 복제 상태 개요
- 여러 대상 버킷에 복제하는 경우 복제 상태
- Amazon S3 복제본 수정 동기화가 사용 설정된 경우 복제 상태
- 배치 복제 작업에서 복제 상태 정보 사용
- 복제 상태 찾기

복제 상태 개요

복제에는 복제를 구성하는 소스 버킷과 Amazon S3가 객체를 복제하는 하나 이상의 대상 버킷이 있습니다. 이러한 버킷에서 객체(GetObject 사용) 또는 객체 메타데이터(HeadObject 사용)를 요청하면 Amazon S3는 응답에서 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 TaxDocs를 지정해 키 이름 접두사가 TaxDocs인 객체만 복제하도록 Amazon S3에 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 FAILED가 포함된 x-amz-replication-status 헤더를 반환합니다.

  > **Note**
  >
  > 객체를 업로드한 후 객체 복제가 실패할 경우 복제를 재시도할 수 없습니다. 객체를 다시 업로드하거나 S3 배치 복제를 사용하여 실패한 객체를 복제해야 합니다. 배치 복제 사용 방법에 대한 자세한 내용은 the section called “기존 객체 복제” 섹션을 참조하세요.

복제 역할 권한, AWS Key Management Service(AWS KMS) 권한 또는 버킷 권한 누락과 같은 문제의 경우 객체가 FAILED 상태로 전환됩니다. 버킷 또는 리전을 사용할 수 없는 경우와 같은 일시적인 실패의 경우 복제 상태는 FAILED로 전환되지 않고 PENDING 상태 그대로 유지됩니다. 리소스가 다시 온라인 상태가 되면 Amazon S3는 해당 객체 복제를 재개합니다.

- 대상 버킷으로부터 객체를 요청하면 해당 객체가 Amazon S3에서 생성한 복제본인 경우 Amazon S3는 값이 REPLICA인 x-amz-replication-status 헤더를 반환합니다.

> **Note**
>
> 복제가 활성화된 소스 버킷에서 객체를 삭제하려면 먼저 해당 객체의 복제 상태를 통해 객체가 복제되었는지 확인해야 합니다.
> 소스 버킷에서 S3 수명 주기 구성이 활성화된 경우 객체 상태가 COMPLETED 또는 FAILED로 표시될 때까지 Amazon S3는 수명 주기 작업을 보류합니다.

여러 대상 버킷에 복제하는 경우 복제 상태

객체를 여러 대상 버킷에 복제하면 x-amz-replication-status 헤더가 다르게 작동합니다. 소스 객체의 헤더는 복제가 모든 대상에 대해 성공하는 경우에만 COMPLETED 값을 반환합니다. 헤더는 복제가 모든 대상에 대해 완료될 때까지 PENDING 값으로 유지됩니다. 하나 이상의 대상이 복제에 실패하면 헤더가 FAILED을(를) 반환합니다.

Amazon S3 복제본 수정 동기화가 사용 설정된 경우 복제 상태

복제 규칙에서 Amazon S3 복제본 수정 동기화를 사용 설정하면 복제본은 REPLICA 이외의 상태를 보고할 수 있습니다. 메타데이터 변경 사항이 복제 과정에 있는 경우 x-amz-replication-status 헤더가 PENDING을 반환합니다. 복제본 수정 동기화가 메타데이터 복제에 실패하면 헤더가 FAILED를 반환합니다. 메타데이터가 올바르게 복제되면 복제본이 헤더 REPLICA를 반환합니다.

배치 복제 작업에서 복제 상태 정보 사용

배치 복제 작업을 만들 때 필요에 따라 객체 생성 날짜 및 복제 상태와 같은 추가 필터를 지정하여 작업 범위를 줄일 수 있습니다.

다음 값 중 하나 이상을 제공하여 ObjectReplicationStatuses 값을 기반으로 복제할 객체를 필터링할 수 있습니다.

- "NONE" - Amazon S3가 이전에 객체 복제를 시도한 적이 없음을 나타냅니다.

- `"FAILED"` - Amazon S3가 이전에 객체 복제를 시도했지만 실패했음을 나타냅니다.
- `"COMPLETED"` - Amazon S3가 이전에 객체를 성공적으로 복제했음을 나타냅니다.
- `"REPLICA"` - Amazon S3가 다른 소스에서 복제한 복제본 객체임을 나타냅니다.

배치 복제에서 이러한 복제 상태 값을 사용하는 방법에 대한 자세한 내용은 the section called “배치 복제 작업에 대한 필터” 섹션을 참조하세요.

복제 상태 찾기

버킷에 있는 객체의 복제 상태를 가져오기 위해 Amazon S3 인벤토리 도구를 사용할 수 있습니다. Amazon S3는 인벤토리 구성에서 사용자가 지정한 대상 버킷으로 CSV 파일을 전송합니다. 또한 Amazon Athena를 사용하여 인벤토리 보고서의 복제 상태를 쿼리할 수도 있습니다. Amazon S3 인벤토리에 대한 자세한 정보는 S3 Inventory를 사용한 데이터 카탈로그화 및 분석 섹션을 참조하세요.

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 또는 AWS SDK를 사용하여 객체 복제 상태를 확인할 수도 있습니다.

S3 콘솔 사용

Amazon S3 콘솔의 객체 세부 정보 페이지에서 객체의 복제 상태를 볼 수 있습니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 범용 버킷 목록에서 복제 소스 버킷의 이름을 선택합니다.
4. 객체(Objects) 목록에서 객체 이름을 선택합니다. 객체 세부 정보 페이지가 나타납니다.
5. 속성 탭에서 아래로 스크롤하여 객체 관리 개요 섹션을 찾습니다. 관리 구성에서 복제 상태 아래의 값을 확인합니다.

AWS CLI 사용

다음 예제와 같이 AWS Command Line Interface(AWS CLI) `head-object` 명령을 사용하여 객체 메타데이터를 검색합니다. *`amzn-s3-demo-source-bucket1`*을 복제 소스 버킷 이름으로, 다른 *`user input placeholders`*를 사용자의 정보로 대체합니다.

```
aws s3api head-object --bucket amzn-s3-demo-source-bucket1 --key object-key --version-id object-version-id
```

이 명령은 다음의 응답 예와 같이 ReplicationStatus가 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 사용

다음 코드 조각은 각각 AWS SDK for Java 및 AWS SDK for .NET를 사용하여 복제 상태를 가져옵니다.

Java

```
GetObjectMetadataRequest metadataRequest = new GetObjectMetadataRequest(bucketName,
 key);
ObjectMetadata metadata = s3Client.getObjectMetadata(metadataRequest);

System.out.println("Replication Status : " +
 metadata.getRawMetadataValue(Headers.OBJECT_REPLICATION_STATUS));
```

.NET

```
GetObjectMetadataRequest getmetadataRequest = new GetObjectMetadataRequest
    {
         BucketName = sourceBucket,
         Key        = objectKey
    };

GetObjectMetadataResponse getmetadataResponse =
 client.GetObjectMetadata(getmetadataRequest);
Console.WriteLine("Object replication status: {0}",
 getmetadataResponse.ReplicationStatus);
```

다중 리전 액세스 포인트로 다중 리전 트래픽 관리

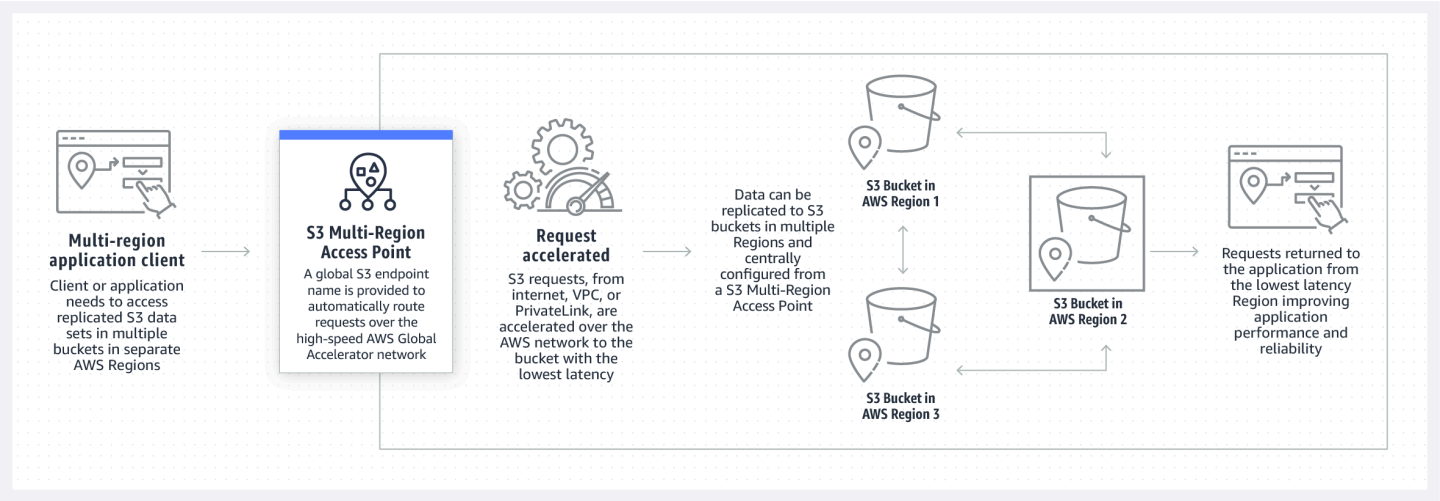
Amazon S3 다중 리전 액세스 포인트는 애플리케이션이 여러 AWS 리전에 있는 S3 버킷의 요청을 이행하는 데 사용할 수 있는 글로벌 엔드포인트를 제공합니다. 다중 리전 액세스 포인트를 사용하여 단일 리전에서 사용되는 것과 동일한 아키텍처로 다중 리전 애플리케이션을 구축하면 전 세계 어디에서나 해당 애플리케이션을 실행할 수 있습니다. 다중 리전 액세스 포인트는 혼잡한 퍼블릭 인터넷을 통해 요청을 보내는 대신 Amazon S3에 대한 인터넷 기반 요청의 가속화를 통해 기본 제공 네트워크 복원력을 제공합니다. 다중 리전 액세스 포인트 글로벌 엔드포인트에 대한 애플리케이션 요청은 AWS Global Accelerator를 사용하여 AWS 글로벌 네트워크를 통해 라우팅 상태가 액티브인 가장 가까운 S3 버킷으로 자동으로 라우팅됩니다.

리전별 트래픽 중단이 발생한 경우 다중 리전 액세스 포인트 장애 조치 제어를 사용하여 AWS 리전 간에 S3 데이터 요청 트래픽을 전환하고 몇 분 이내에 중단으로부터 S3 트래픽을 리디렉션할 수 있습니다. 또한, 중단에 대비해서 애플리케이션 복원력을 테스트하여 애플리케이션 장애 조치를 진행하고 재해 복구 시뮬레이션을 수행할 수 있습니다. VPC 외부에서 S3에 연결하고 요청을 가속화해야 하는 경우 Amazon S3 다중 리전 액세스 포인트를 사용하여 애플리케이션 및 네트워크 아키텍처를 단순화할 수 있습니다. 다중 리전 액세스 포인트 요청은 퍼블릭 인터넷을 통과할 필요 없이 AWS 글로벌 네트워크를 통해 라우팅된 다음 해당 AWS 리전 내 S3로 돌아갑니다. 따라서 더 많은 고가용성 애플리케이션을 구축할 수 있습니다.

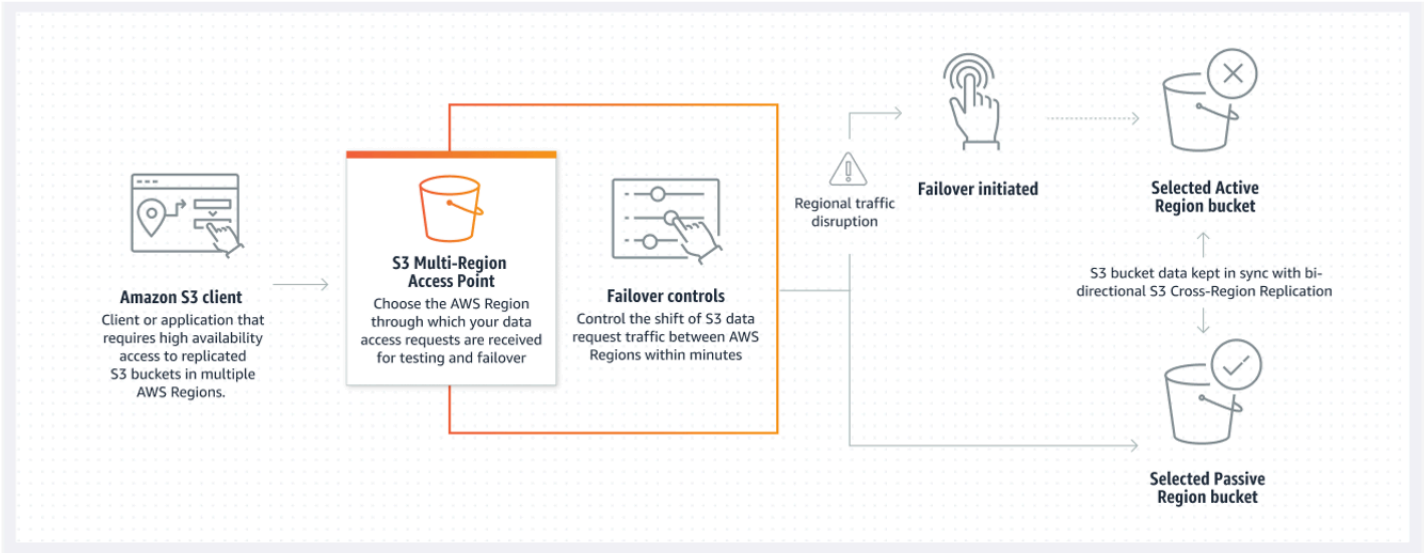
다중 리전 액세스 포인트를 만들고 설정하는 동안 해당 다중 리전 액세스 포인트를 통해 제공할 데이터를 저장할 AWS 리전 집합을 지정합니다. 제공된 다중 리전 액세스 포인트 엔드포인트 이름을 사용하여 클라이언트를 연결할 수 있습니다. 클라이언트 연결을 설정한 후 다중 리전 액세스 포인트 요청을 라우팅할 기존 또는 새 버킷을 선택할 수 있습니다. 그런 다음 S3 교차 리전 복제(CRR) 규칙을 사용하여 해당 리전의 버킷 간에 데이터를 동기화합니다.

다중 리전 액세스 포인트를 설정한 후 다중 리전 액세스 포인트 글로벌 엔드포인트를 통해 데이터를 요청하거나 작성할 수 있습니다. Amazon S3는 사용 가능한 가장 가까운 리전에서 복제된 데이터 세트에 대한 요청을 자동으로 이행합니다. 또한, AWS Management Console에서 다중 리전 액세스 포인트 요청과 관련된 기본 복제 토폴로지 및 복제 지표를 볼 수 있습니다. 이를 통해 다중 리전 애플리케이션용 스토리지를 훨씬 쉽게 구축, 관리 및 모니터링할 수 있습니다. 아니면 Amazon CloudFront를 사용하여 S3 다중 리전 액세스 포인트의 생성 및 구성을 자동화할 수 있습니다.

다음 이미지는 액티브-액티브로 구성된 Amazon S3 다중 리전 액세스 포인트를 그래픽으로 나타낸 것입니다. 이 그래픽은 Amazon S3 요청이 가장 가까운 액티브 AWS 리전의 버킷으로 자동 라우팅되는 방식을 보여줍니다.



다음 이미지는 액티브-패시브로 구성된 Amazon S3 다중 리전 액세스 포인트를 그래픽으로 나타낸 것입니다. 이 그래픽은 Amazon S3 데이터 액세스 트래픽이 액티브 및 패시브 AWS 리전 간에 장애 조치되도록 제어하는 방법을 보여줍니다.



다중 리전 액세스 포인트를 사용하는 방법에 대한 자세한 내용을 [자습서: Amazon S3 다중 리전 액세스 포인트 시작하기]를 참조하십시오.

주제

- [다중 리전 액세스 포인트 생성]
- [AWS PrivateLink와 함께 사용할 다중 리전 액세스 포인트 구성]
- [다중 리전 액세스 포인트를 통한 요청]

다중 리전 액세스 포인트 생성

Amazon S3에서 다중 리전 액세스 포인트를 생성하려면 다음을 수행합니다.

- 다중 리전 액세스 포인트의 이름을 지정합니다.
- 다중 리전 액세스 포인트에 대한 요청을 처리할 각 AWS 리전에서 버킷을 하나씩 선택합니다.
- 다중 리전 액세스 포인트에 Amazon S3 퍼블릭 액세스를 차단 설정을 구성합니다.

Amazon S3가 비동기적으로 처리하는 생성 요청에 이 정보를 모두 제공합니다. Amazon S3는 비동기 생성 요청의 상태를 모니터링하는 데 사용할 수 있는 토큰을 제공합니다.

정책을 저장하기 전에 AWS Identity and Access Management Access Analyzer의 보안 경고, 오류, 일반 경고 및 제안 사항을 해결해야 합니다. IAM Access Analyzer는 정책 확인은 실행하여 IAM 정책 문법 및 모범 사례에 대해 정책을 검증합니다. 이러한 확인은 결과를 생성하고 보안 모범 사례를 준수하고 작동하는 정책을 작성하는 데 도움이 되는 실행 가능한 권장 사항을 제공합니다. IAM 액세스 분석기를 사용한 정책 검증에 대한 자세한 내용은 IAM 사용 설명서의 IAM 액세스 분석기 정책 검증을 참조하세요. IAM Access Analyzer에서 반환된 경고, 오류 및 제안 사항 목록을 보려면 IAM Access Analyzer 정책 확인 참조를 참조하세요.

API를 사용하는 경우 다중 리전 액세스 포인트 생성 요청은 비동기적입니다. 다중 리전 액세스 포인트 생성 요청을 제출하면 Amazon S3가 요청을 동기적으로 승인합니다. 그런 다음 생성 요청의 진행 상황을 추적하는 데 사용할 수 있는 토큰을 즉시 반환합니다. 다중 리전 액세스 포인트를 생성 및 관리를 위한 비동기 요청을 추적에 대한 자세한 내용은 지원되는 API 작업으로 다중 리전 액세스 포인트 사용 섹션을 참조하세요.

다중 리전 액세스 포인트를 생성한 후 이에 대한 액세스 제어 정책을 생성할 수 있습니다. 각 다중 리전 액세스 포인트에는 연결된 정책이 있을 수 있습니다. 다중 리전 액세스 포인트 정책은 리소스, 사용자 또는 기타 조건별로 다중 리전 액세스 포인트 사용을 제한할 수 있는 리소스 기반 정책입니다.

> **Note**
>
> 애플리케이션 또는 사용자가 다중 리전 액세스 포인트를 통해 객체에 액세스할 수 있으려면 다음 두 정책이 모두 요청을 허용해야 합니다.
>
> - 다중 리전 액세스 포인트에 대한 액세스 정책
> - 객체가 포함된 기본 버킷에 대한 액세스 정책
>
> 두 정책이 서로 다른 경우 더 제한적인 정책이 우선합니다.

다중 리전 액세스 포인트에 대한 권한 관리를 단순화하려면 버킷의 액세스 제어를 다중 리전 액세스 포인트에 위임하면 됩니다. 자세한 내용은 the section called “다중 리전 액세스 포인트 정책 예시” 단원을 참조하십시오.

버킷에 다중 리전 액세스 포인트를 사용해도 버킷이 기존 버킷 이름이나 Amazon 리소스 이름(ARN)을 통해 액세스할 때 버킷의 동작은 변경되지 않습니다. 버킷에 대한 모든 기존 작업은 이전과 같이 계속 작동합니다. 다중 리전 액세스 포인트 정책에 포함시키는 제한은 해당 다중 리전 액세스 포인트를 통해 이루어진 요청에만 적용됩니다.

다중 리전 액세스 포인트를 생성한 후에는 정책을 업데이트할 수 있지만 정책을 삭제할 수는 없습니다. 하지만 모든 권한을 거부하도록 다중 리전 액세스 포인트 정책을 업데이트할 수 있습니다.

주제

- Amazon S3 다중 리전 액세스 포인트 이름 지정 규칙
- Amazon S3 다중 리전 액세스 포인트용 버킷 선택 규칙
- Amazon S3 다중 리전 액세스 포인트 생성
- Amazon S3 다중 리전 액세스 포인트를 사용하여 퍼블릭 액세스 차단
- Amazon S3 다중 리전 액세스 포인트 구성 세부 정보 보기
- 다중 리전 액세스 포인트 삭제

Amazon S3 다중 리전 액세스 포인트 이름 지정 규칙

다중 리전 액세스 포인트를 생성할 때, 선택한 문자열인 이름을 지정합니다. 다중 리전 액세스 포인트를 생성한 후에는 이름을 변경할 수 없습니다. 이름은 AWS 계정에서 고유해야 하며, 다중 리전 액세스 포인트 규제 및 제한에 나열된 이름 지정 요구 사항을 준수해야 합니다. 다중 리전 액세스 포인트를 쉽게 식별할 수 있도록 사용자 또는 조직에 의미 있는 이름 또는 시나리오를 반영하는 이름을 사용합니다.

`GetMultiRegionAccessPoint` 및 `PutMultiRegionAccessPointPolicy`와 같은 다중 리전 액세스 포인트 관리 작업을 호출할 때 이 이름을 사용합니다. 이 이름은 다중 리전 액세스 포인트로 요청을 보내는 데 사용되지 않으며 다중 리전 액세스 포인트를 사용하여 요청하는 클라이언트에게는 노출될 필요가 없습니다.

Amazon S3가 다중 리전 액세스 포인트를 생성하면 자동으로 별칭을 할당합니다. 이 별칭은 `.mrap`로 끝나는 고유한 영숫자 문자열입니다. 별칭은 다중 리전 액세스 포인트의 Amazon 리소스 이름(ARN)과

호스트 이름을 구성하는 데 사용됩니다. 또한 정규화된 이름은 다중 리전 액세스 포인트의 별칭을 기반으로 합니다.

별칭에서 다중 리전 액세스 포인트의 이름을 확인할 수 없으므로 다중 리전 액세스 포인트의 이름, 용도 또는 소유자가 노출될 위험 없이 별칭을 공개할 수 있습니다. Amazon S3는 각각의 새로운 다중 리전 액세스 포인트에 대한 별칭을 선택하며 별칭은 변경할 수 없습니다. 다중 리전 액세스 포인트 주소 지정에 대한 자세한 내용은 다중 리전 액세스 포인트를 통한 요청 섹션을 참조하세요.

다중 리전 액세스 포인트 별칭은 항상 고유하며 다중 리전 액세스 포인트의 이름이나 구성을 기반으로 하지 않습니다. 다중 리전 액세스 포인트를 생성한 후 삭제하고 동일한 이름과 구성의 다른 액세스 포인트를 생성하면 두 번째 다중 리전 액세스 포인트의 별칭은 첫 번째 액세스 포인트와 달라집니다. 새로운 다중 리전 액세스 포인트는 이전의 다중 리전 액세스 포인트와 동일한 별칭을 가질 수 없습니다.

Amazon S3 다중 리전 액세스 포인트용 버킷 선택 규칙

각 다중 리전 액세스 포인트는 요청을 이행할 리전과 연결됩니다. 다중 리전 액세스 포인트는 각 리전에서 정확히 하나의 버킷과 연결되어야 합니다. 요청에서 각 버킷의 이름을 지정하여 다중 리전 액세스 포인트를 생성합니다. 다중 리전 액세스 포인트를 지원하는 버킷은 다중 리전 액세스 포인트를 소유한 동일한 AWS 계정에 있거나 다른 AWS 계정에 있을 수 있습니다.

다중 리전 액세스 포인트에서 단일 버킷을 사용할 수 있습니다.

Important

- 다중 리전 액세스 포인트와 연결된 버킷은 생성할 때만 지정할 수 있습니다. 버킷을 생성한 후에는 다중 리전 액세스 포인트 구성에서 추가, 수정 또는 제거할 수 없습니다. 버킷을 변경하려면 전체 다중 리전 액세스 포인트를 삭제하고 새 액세스 포인트를 생성해야 합니다.
- 다중 리전 액세스 포인트의 일부인 버킷은 삭제할 수 없습니다. 다중 리전 액세스 포인트에 연결된 버킷을 삭제하려면 먼저 다중 리전 액세스 포인트를 삭제해야 합니다.
- 다른 계정이 소유한 버킷을 다중 리전 액세스 포인트에 추가하면 다중 리전 액세스 포인트에 대한 액세스 권한을 부여하도록 버킷 소유자가 버킷 정책도 업데이트해야 합니다. 그러지 않으면 다중 리전 액세스 포인트가 해당 버킷에서 데이터를 검색할 수 없습니다. 이러한 액세스 권한을 부여하는 방법을 보여주는 정책의 예시는 다중 리전 액세스 포인트 정책 예시를 참조하세요.
- 일부 리전에서는 다중 리전 액세스 포인트를 지원하지 않습니다. 지원되는 리전 목록을 보려면 다중 리전 액세스 포인트 규제 및 제한 섹션을 참조하세요.

복제 규칙을 생성하여 버킷 간에 데이터를 동기화할 수 있습니다. 이러한 규칙을 사용하면 소스 버킷에서 대상 버킷으로 데이터를 자동으로 복사할 수 있습니다. 버킷을 다중 리전 액세스 포인트에 연결해도 복제 작동 방식에는 영향을 미치지 않습니다. 이후 섹션에서 다중 리전 액세스 포인트를 사용한 복제 구성에 대해 설명합니다.

> **Important**
>
> 다중 리전 액세스 포인트에 요청하면 다중 리전 액세스 포인트는 다중 리전 액세스 포인트에 있는 버킷의 데이터 콘텐츠를 인식하지 못합니다. 따라서 요청을 받는 버킷에는 요청된 데이터가 포함되어 있지 않을 수 있습니다. 하나의 다중 리전 액세스 포인트와 연결된 Amazon S3 버킷에 일관된 데이터 세트를 생성하려면 S3 크로스 리전 복제(CRR)를 구성하는 것이 좋습니다. 자세한 내용은 다중 리전 액세스 포인트와 함께 사용할 복제 구성 단원을 참조하십시오.

Amazon S3 다중 리전 액세스 포인트 생성

다음 예시에서는 Amazon S3 콘솔을 사용하여 다중 리전 액세스 포인트를 생성하는 방법을 보여줍니다.

S3 콘솔 사용

다중 리전 액세스 포인트 생성

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 다중 리전 액세스 포인트)를 선택합니다.
3. 다중 리전 액세스 포인트 생성을 선택하여 다중 리전 액세스 포인트 생성을 시작합니다.
4. 다중 리전 액세스 포인트 페이지의 다중 리전 액세스 포인트 이름 필드에 다중 리전 액세스 포인트의 이름을 입력합니다.
5. 이 다중 리전 액세스 포인트와 연결할 버킷을 선택합니다. 계정에 있는 버킷을 선택하거나 다른 계정의 버킷을 선택할 수 있습니다.

> **Note**
>
> 사용자 계정 또는 다른 계정에서 하나 이상의 버킷을 추가해야 합니다. 또한 다중 리전 액세스 포인트는 AWS 리전당 한 개의 버킷만 지원한다는 점을 유의하세요. 따라서 동일한

리전에서 두 개의 버킷을 추가할 수 없습니다. 기본적으로 비활성화된 AWS 리전은 지원되지 않습니다.

- 사용자 계정에 있는 버킷을 추가하려면 버킷 추가를 선택합니다. 사용자 계정에 있는 모든 버킷의 목록이 표시됩니다. 이름으로 버킷을 검색하거나 버킷 이름을 알파벳순으로 정렬할 수 있습니다.
- 다른 계정에서 버킷을 추가하려면 다른 계정에서 버킷 추가를 선택합니다. 다른 계정에서는 버킷을 검색하거나 찾아볼 수 없으므로 정확한 버킷 이름과 AWS 계정 ID를 알고 있어야 합니다.

> **Note**
>
> 유효한 AWS 계정 ID와 버킷 이름을 입력해야 합니다. 또한 버킷은 지원되는 리전에 있어야 합니다. 그렇지 않으면 다중 리전 액세스 포인트를 만들려고 할 때 오류가 발생합니다. 다중 리전 액세스 포인트를 지원하는 리전 목록은 다중 리전 액세스 포인트 규제 및 제한을 참조하세요.

6. (선택 사항) 추가한 버킷을 제거해야 하는 경우 제거를 선택합니다.

> **Note**
>
> 다중 리전 액세스 포인트 생성을 완료한 후에는 이 다중 리전 액세스 포인트에 버킷을 추가하거나 버킷을 제거할 수 없습니다.

7. 이 다중 리전 액세스 포인트에 대한 퍼블릭 액세스 차단 설정(Block Public Access settings for this Multi-Region Access Point)에서 다중 리전 액세스 포인트에 적용할 퍼블릭 액세스 차단 설정을 선택합니다. 새 다중 리전 액세스 포인트에는 기본값으로 모든 퍼블릭 액세스 차단 설정이 사용 설정되어 있습니다. 특정 설정을 사용 중지해야 하는 경우가 아니면 모든 설정을 그대로 유지하는 것이 좋습니다.

> **Note**
>
> 다중 리전 액세스 포인트를 만든 후에는 다중 리전 액세스 포인트의 퍼블릭 액세스 차단 설정을 변경할 수 없습니다. 따라서 퍼블릭 액세스를 차단하려면 다중 리전 액세스 포인트를 생성하기 전에 퍼블릭 액세스 없이 애플리케이션이 제대로 작동하는지 확인하세요.

8. 다중 리전 액세스 포인트 생성(Create Multi-Region Access Point)을 선택합니다.

> **Important**
>
> 다른 계정이 소유한 버킷을 다중 리전 액세스 포인트에 추가할 때는 다중 리전 액세스 포인트에 대한 액세스 권한을 부여하도록 버킷 소유자가 버킷 정책도 업데이트해야 합니다. 그러지 않으면 다중 리전 액세스 포인트가 해당 버킷에서 데이터를 검색할 수 없습니다. 이러한 액세스 권한을 부여하는 방법을 보여주는 정책의 예시는 다중 리전 액세스 포인트 정책 예시를 참조하세요.

AWS CLI 사용

AWS CLI를 사용하여 다중 리전 액세스 포인트를 생성할 수 있습니다. 다중 리전 액세스 포인트를 생성할 때 다중 리전 액세스 포인트가 지원할 모든 버킷을 제공해야 합니다. 생성한 후에는 다중 리전 액세스 포인트에 버킷을 추가할 수 없습니다.

다음 예시에서는 AWS CLI를 사용하여 2개의 버킷이 있는 다중 리전 액세스 포인트를 생성하는 방법을 보여줍니다. 이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3control create-multi-region-access-point --account-id 111122223333 --details '{
        "Name": "simple-multiregionaccesspoint-with-two-regions",
        "PublicAccessBlock": {
            "BlockPublicAcls": true,
            "IgnorePublicAcls": true,
            "BlockPublicPolicy": true,
            "RestrictPublicBuckets": true
        },
        "Regions": [
            { "Bucket": "amzn-s3-demo-bucket1" },
            { "Bucket": "amzn-s3-demo-bucket2" }
        ]
    }' --region us-west-2
```

Amazon S3 다중 리전 액세스 포인트를 사용하여 퍼블릭 액세스 차단

각 다중 리전 액세스 포인트에는 Amazon S3 퍼블릭 액세스를 차단하는 고유한 설정이 있습니다. 이러한 설정은 다중 리전 액세스 포인트와 기본 버킷을 소유한 AWS 계정에 대한 퍼블릭 액세스 차단 설정과 함께 작동합니다.

Amazon S3가 요청을 승인하면 이러한 설정의 가장 제한적인 조합을 적용합니다. 이러한 리소스(다중 리전 액세스 포인트 소유자 계정, 기본 버킷 또는 버킷 소유자 계정)에 대한 퍼블릭 액세스 차단 설정이 요청된 작업 또는 리소스에 대한 액세스를 차단하는 경우 Amazon S3는 요청을 거부합니다.

특정 설정을 사용 중지해야 하는 경우가 아니면 모든 퍼블릭 액세스 차단 설정을 사용 설정하는 것이 좋습니다. 다중 리전 액세스 포인트에는 기본적으로 모든 퍼블릭 액세스 차단 설정이 사용 설정되어 있습니다. 퍼블릭 액세스 차단이 활성화된 경우 다중 리전 액세스 포인트는 인터넷 기반 요청을 수락할 수 없습니다.

> **Important**
>
> 다중 리전 액세스 포인트를 만든 후에는 다중 리전 액세스 포인트의 퍼블릭 액세스 차단 설정을 변경할 수 없습니다.

Amazon S3 퍼블릭 액세스 차단에 대한 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하세요.

Amazon S3 다중 리전 액세스 포인트 구성 세부 정보 보기

다음 예시에서는 Amazon S3 콘솔을 사용하여 다중 리전 액세스 포인트 구성 세부 정보를 확인하는 방법을 보여줍니다.

S3 콘솔 사용

다중 리전 액세스 포인트 생성

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 다중 리전 액세스 포인트)를 선택합니다.
3. 구성 세부 정보를 보려는 다중 리전 액세스 포인트의 이름을 선택합니다.

   - 속성 탭에는 다중 리전 액세스 포인트와 연결된 모든 버킷, 생성 날짜, Amazon 리소스 이름(ARN) 및 별칭이 나열됩니다. AWS 계정 ID 열에는 다중 리전 액세스 포인트와 연결된 외부 계정이 소유한 모든 버킷도 나열됩니다.
   - 권한 탭에는 이 다중 리전 액세스 포인트와 연결된 버킷에 적용되는 퍼블릭 액세스 차단 설정이 나열됩니다. 다중 리전 액세스 포인트를 생성한 경우 해당 다중 리전 액세스 포인트에 대한 다중 리전 액세스 포인트 정책도 볼 수 있습니다. 권한 페이지의 정보 알림에는 이 다중 리전 액세스 포인트에서 퍼블릭 액세스가 차단됨 설정이 활성화된 모든 버킷(사용자 계정 및 기타 계정의 버킷)도 나열됩니다.
   - 복제 및 장애 조치 탭에서는 다중 리전 액세스 포인트와 연결된 버킷과 해당 버킷이 있는 리전의 맵 보기를 제공합니다. 데이터를 가져올 권한이 없는 다른 계정의 버킷이 있는 경우 복제 요약약

맵에 리전이 빨간색으로 표시되어 복제 상태를 가져오는 중 오류가 발생한 AWS 리전을 나타냅니다.

> **Note**
>
> 외부 계정의 버킷에서 복제 상태 정보를 검색하려면 버킷 소유자가 버킷 정책에서 `s3:GetBucketReplication` 권한을 부여해야 합니다.

이 탭은 또한 다중 리전 액세스 포인트와 함께 사용되는 리전에 대한 복제 지표, 복제 규칙 및 장애 조치 상태를 제공합니다.

AWS CLI 사용

AWS CLI를 사용하여 다중 리전 액세스 포인트의 구성 세부 정보를 볼 수 있습니다.

다음 AWS CLI 예시 명령은 현재 다중 리전 액세스 포인트 구성을 가져옵니다. 이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3control get-multi-region-access-point --account-id 111122223333 --name amzn-s3-demo-bucket
```

다중 리전 액세스 포인트 삭제

다음 절차는 Amazon S3 콘솔을 사용하여 다중 리전 액세스 포인트를 삭제하는 방법을 보여줍니다.

다중 리전 액세스 포인트를 삭제해도 다중 리전 액세스 포인트와 연결된 버킷은 삭제되지 않으며 다중 리전 액세스 포인트 자체만 삭제됩니다.

S3 콘솔 사용

다중 리전 액세스 포인트를 삭제하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 다중 리전 액세스 포인트)를 선택합니다.
3. 다중 리전 액세스 포인트 이름 옆에 있는 옵션 버튼을 선택합니다.
4. Delete(삭제)를 선택합니다.

5. 다중 리전 액세스 포인트 삭제 대화 상자에서 삭제하려는 AWS 버킷의 이름을 입력합니다.

> **Note**
>
> 유효한 버킷 이름을 입력해야 합니다. 그러지 않으면 삭제 버튼이 비활성화됩니다.

6. 다중 리전 액세스 포인트의 삭제를 확인하려면 삭제를 선택합니다.

AWS CLI 사용

AWS CLI를 사용하여 다중 리전 액세스 포인트를 삭제할 수 있습니다. 이 작업은 다중 리전 액세스 포인트와 연결된 버킷은 삭제하지 않으며 다중 리전 액세스 포인트 자체만 삭제합니다. 이 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하십시오.

```
aws s3control delete-multi-region-access-point --account-id 123456789012 --details
 Name=example-multi-region-access-point-name
```

AWS PrivateLink와 함께 사용할 다중 리전 액세스 포인트 구성

다중 리전 액세스 포인트를 사용하여 AWS 리전 간에 Amazon S3 요청 트래픽을 라우팅할 수 있습니다. 각 다중 리전 액세스 포인트 글로벌 엔드포인트는 별도의 엔드포인트로 복잡한 네트워킹 구성을 구축할 필요 없이 여러 소스의 Amazon S3 데이터 요청 트래픽을 라우팅합니다. 이러한 데이터 요청 트래픽 소스에는 다음이 포함됩니다.

- Virtual Private Cloud(VPC)에서 시작되는 트래픽
- AWS PrivateLink를 통해 이동하는 온프레미스 데이터 센터에서 온 트래픽
- 퍼블릭 인터넷 트래픽

S3 다중 리전 액세스 포인트에 AWS PrivateLink 연결을 설정하면 간단한 네트워크 아키텍처 및 구성을 사용하여 프라이빗 연결을 통해 S3 요청을 AWS 또는 여러 AWS 리전에 라우팅할 수 있습니다. AWS PrivateLink를 사용하면 VPC 피어링 연결을 구성할 필요가 없습니다.

주제

- AWS PrivateLink와 함께 사용할 다중 리전 액세스 포인트 구성
- VPC 엔드포인트에서 다중 리전 액세스 포인트에 대한 액세스 제거

AWS PrivateLink와 함께 사용할 다중 리전 액세스 포인트 구성

AWS PrivateLink는 Virtual Private Cloud(VPC)의 프라이빗 IP 주소를 사용하여 Amazon S3에 대한 프라이빗 연결을 제공합니다. VPC 내부에 하나 이상의 인터페이스 엔드포인트를 프로비저닝하여 Amazon S3 다중 리전 액세스 포인트에 연결할 수 있습니다.

AWS Management Console, AWS CLI 또는 AWS SDK를 통해 다중 리전 액세스 포인트의 com.amazonaws.s3-global.accesspoint 엔드포인트를 생성할 수 있습니다. 다중 리전 액세스 포인트의 인터페이스 엔드포인트를 구성하는 방법에 대한 자세한 내용은 VPC 사용 설명서의 인터페이스 VPC 엔드포인트를 참조하세요.

인터페이스 엔드포인트를 통해 다중 리전 액세스 포인트에 요청하려면 다음 단계에 따라 VPC 와 다중 리전 액세스 포인트를 구성합니다.

AWS PrivateLink와 함께 사용할 다중 리전 액세스 포인트 구성

1. 다중 리전 액세스 포인트에 연결할 수 있는 적절한 VPC 엔드포인트를 생성하거나 보유합니다. VPC 엔드포인트 생성에 대한 자세한 내용은 VPC 사용 설명서의 VPC 엔드포인트를 참조하세요.

   > **Important**
   >
   > com.amazonaws.s3-global.accesspoint 엔드포인트를 생성해야 합니다. 다른 엔드포인트 유형은 다중 리전 액세스 포인트에 액세스할 수 없습니다.

   이 VPC 엔드포인트가 생성되면 엔드포인트에 프라이빗 DNS가 사용 설정된 경우 VPC의 모든 다중 리전 액세스 포인트 요청이 이 엔드포인트를 통해 라우팅됩니다. 이는 기본값으로 사용 설정되어 있습니다.
2. 다중 리전 액세스 포인트 정책이 VPC 엔드포인트의 연결을 지원하지 않는 경우 이를 업데이트해야 합니다.
3. 개별 버킷 정책이 다중 리전 액세스 포인트의 사용자에 대한 액세스를 허용하는지 확인합니다.

다중 지역 액세스 포인트는 요청 자체를 이행하는 것이 아니라 요청을 버킷으로 라우팅하여 작동한다는 점을 기억하세요. 요청 발신자는 다중 리전 액세스 포인트에 대한 권한을 가지고 있어야 하며 다중 리전 액세스 포인트의 개별 버킷에 액세스할 수 있어야 합니다. 그렇지 않은 경우, 요청 발신자에게 요청을 이행할 권한이 없는 버킷으로 요청이 라우팅될 수 있습니다. 다중 리전 액세스 포인트 및 다중 리전 액세스 포인트에 연결된 버킷은 같거나 다른 AWS에서 소유할 수 있습니다. 그러나 권한이 올바르게 구성된 경우 서로 다른 계정의 VPC는 다중 리전 액세스 포인트를 사용할 수 있습니다.

따라서 VPC 엔드포인트 정책은 다중 리전 액세스 포인트 및 요청을 이행할 수 있는 각 기본 버킷에 대한 액세스를 모두 허용해야 합니다. 예를 들어 별칭이 mfzwi23gnjvgw.mrap인 다중 리전 액세스 포인트가 있다고 가정해 보겠습니다. 이는 버킷 amzn-s3-demo-bucket1 및 amzn-s3-demo-bucket2에서 지원하며 모두 AWS 계정 123456789012에서 소유합니다. 이 경우 다음 VPC 엔드포인트트 정책은 mfzwi23gnjvgw.mrap에 대한 VPC의 GetObject 요청을 어느 것이든 하나의 백업 버킷에서 이행하도록 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Sid": "Read-buckets-and-MRAP-VPCE-policy",
        "Principal": "*",
        "Action": [
            "s3:GetObject"
        ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "arn:aws:s3:::amzn-s3-demo-bucket2/*",
            "arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap/object/*"
        ]
    }]
}
```

앞서 언급했듯이 다중 리전 액세스 포인트 정책이 VPC 엔드포인트를 통한 액세스를 지원하도록 구성되어 있는지 확인해야 합니다. 액세스를 요청하는 VPC 엔드포인트는 지정할 필요가 없습니다. 다음 샘플 정책은 GetObject 요청에 다중 리전 액세스 포인트를 사용하려는 모든 요청자에게 액세스 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Sid": "Open-read-MRAP-policy",
        "Effect": "Allow",
        "Principal": "*",
        "Action": [
            "s3:GetObject"
          ],
        "Resource": "arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap/object/*"
```

```
    }]
}
```

물론 개별 버킷에는 각각 VPC 엔드포인트를 통해 제출된 요청의 액세스를 지원하는 정책이 필요합니다. 다음 예제 정책은 모든 익명 사용자에게 읽기 액세스 권한을 부여하며, 여기에는 VPC 엔드포인트를 통한 요청이 포함됩니다.

```
{
    "Version":"2012-10-17",
   "Statement": [
   {
       "Sid": "Public-read",
       "Effect":"Allow",
       "Principal": "*",
       "Action": "s3:GetObject",
       "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket1",
           "arn:aws:s3:::amzn-s3-demo-bucket2/*"]
    }]
}
```

VPC 엔드포인트트 정책 편집에 대한 자세한 내용은 VPC 사용 설명서의 Control access to services with VPC endpoints(VPC 엔드포인트를 통해 서비스에 대한 액세스 제어)를 참조하세요.

VPC 엔드포인트에서 다중 리전 액세스 포인트에 대한 액세스 제거

다중 리전 액세스 포인트를 소유하고 있고 인터페이스 엔드포인트에서 액세스 권한을 제거하려면 VPC 엔드포인트를 통해 들어오는 요청에 대한 액세스를 차단하는 다중 리전 액세스 포인트에 새 액세스 정책을 제공해야 합니다. 그러나 다중 리전 액세스 포인트의 버킷이 VPC 엔드포인트를 통한 요청을 지원할 경우 이러한 요청을 계속 지원합니다. 이러한 지원을 방지하려면 버킷에 대한 정책도 업데이트해야 합니다. 다중 리전 액세스 포인트에 새 액세스 정책을 제공하면 기본 버킷이 아닌 다중 리전 액세스 포인트에 대한 액세스만 차단됩니다.

> **Note**
>
> 다중 리전 액세스 포인트에 대한 액세스 정책은 삭제할 수 없습니다. 다중 리전 액세스 포인트에 대한 액세스를 제거하려면 원하는 수정된 액세스를 포함하는 새 액세스 정책을 제공해야 합니다.

다중 리전 액세스 포인트의 액세스 정책을 업데이트하는 대신 버킷 정책을 업데이트하여 VPC 엔드포인트를 통한 요청을 방지할 수 있습니다. 이 경우에도 사용자는 여전히 VPC 엔드포인트를 통해 다중 리전 액세스 포인트에 액세스할 수 있습니다. 그러나 버킷 정책에서 액세스를 차단하는 버킷으로 다중 리전 액세스 포인트 요청이 라우팅되면 오류 메시지가 생성됩니다.

다중 리전 액세스 포인트를 통한 요청

다른 리소스와 마찬가지로 Amazon S3 다중 리전 액세스 포인트에는 Amazon 리소스 이름(ARN)이 있습니다. 이러한 ARN을 사용하면 AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 API를 사용하여 다중 리전 액세스 포인트로 요청을 보낼 수 있습니다. 또한 이러한 ARN을 사용하여 액세스 제어 정책에서 다중 리전 액세스 포인트를 식별할 수 있습니다. 다중 리전 액세스 ARN에는 다중 리전 액세스 포인트의 이름이 포함되거나 공개되지 않습니다. ARN에 대한 자세한 내용은 AWS 일반 참조의 Amazon 리소스 이름(ARN)을 참조하세요.

> **Note**
>
> 다중 리전 액세스 포인트 별칭과 ARN은 서로 바꿔서 사용할 수 없습니다.

다중 리전 액세스 포인트 ARN은 다음 형식을 사용합니다.

`arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias`

다중 리전 액세스 포인트 ARN의 예를 들면 같습니다.

- `arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap`는 별칭이 `mfzwi23gnjvgw.mrap`이고 AWS 계정 `123456789012`에서 소유한 다중 리전 액세스 포인트를 나타냅니다.
- `arn:aws:s3::123456789012:accesspoint/*`는 `123456789012` 계정에 있는 모든 다중 리전 액세스 포인트를 나타냅니다. 이 ARN은 `123456789012` 계정의 모든 다중 리전 액세스 포인트와 매칭하지만 ARN에 AWS 리전이 포함되어 있지 않기 때문에 리전 Amazon S3 액세스 포인트와는 매칭하지 않습니다. 대조적으로 `arn:aws:s3:us-west-2:123456789012:accesspoint/*` ARN은 계정 `123456789012`에 대한 `us-west-2` 리전의 모든 리전 액세스 포인트와 매칭하지만 다중 리전 액세스 포인트와는 매칭하지 않습니다.

다중 리전 액세스 포인트를 통해 액세스하는 객체의 ARN은 다음 형식을 사용합니다.

`arn:aws:s3::account_id:accesspoint/MultiRegionAccessPoint_alias//key`

다중 리전 액세스 포인트 ARN과 마찬가지로 다중 리전 액세스 포인트를 통해 액세스되는 객체의 ARN에는 AWS 리전 리전이 포함되지 않습니다. 여기 몇 가지 예가 있습니다.

- `arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap//-01`은 별칭이 `mfzwi23gnjvgw.mrap`이며 계정 `123456789012`에서 소유한 다중 리전 액세스 포인트를 통해 액세스되는 `-01`을 나타냅니다.
- `arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap//*`는 별칭이 `mfzwi23gnjvgw.mrap`이며 `123456789012` 계정에서 소유한 다중 리전 액세스 포인트를 통해 액세스될 수 있는 모든 객체를 나타냅니다.
- `arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap//-01/finance/*`는 별칭이 `mfzwi23gnjvgw.mrap`이며 `123456789012` 계정에서 소유한 다중 리전 액세스 포인트의 `-01/finance/`에서 액세스될 수 있는 모든 객체를 나타냅니다.

다중 리전 액세스 포인트 호스트 이름

다중 리전 액세스 포인트의 호스트 이름을 사용하여 다중 리전 액세스 포인트를 통해 Amazon S3 데이터에 액세스할 수 있습니다. 공용 인터넷에서 이 호스트 이름으로 요청을 보낼 수 있습니다. 다중 리전 액세스 포인트에 대해 하나 이상의 인터넷 게이트웨이를 구성한 경우 Virtual Private Cloud(VPC)에서 이 호스트 이름으로 요청을 보낼 수 있습니다. 다중 리전 액세스 포인트에 사용할 VPC 인터페이스 엔드포인트에 대한 자세한 내용은 AWS PrivateLink와 함께 사용할 다중 리전 액세스 포인트 구성 섹션을 참조하세요.

VPC 엔드포인트를 사용하여 VPC에서 다중 리전 액세스 포인트를 통해 요청하려면 AWS PrivateLink를 사용하면 됩니다. AWS PrivateLink를 사용한 다중 리전 액세스 포인트에 대한 요청에는 *region*`.vpce.amazonaws.com`으로 끝나는 엔드포인트별 리전 도메인 이름 시스템(DNS)을 직접 사용할 수 없습니다. 이 호스트 이름에는 연결된 인증서가 없으므로 직접 사용할 수 없습니다. VPC 엔드포인트의 퍼블릭 도메인 이름 시스템(DNS) 이름을 CNAME 또는 ALIAS 대상으로 계속 사용할 수 있습니다. 또는 엔드포인트에서 프라이빗 도메인 이름 시스템(DNS)을 사용하도록 설정하고 이 섹션에 설명된 대로 표준 다중 리전 액세스 포인트 *MultiRegionAccessPoint_alias*`.accesspoint.s3-global.amazonaws.com` 도메인 이름 시스템(DNS) 이름을 사용할 수 있습니다.

다중 리전 액세스 포인트를 통해 Amazon S3 데이터에 대한 API 작업을 사용하는 경우(예: `GetObject`) 요청의 호스트 이름은 다음과 같습니다.

MultiRegionAccessPoint_alias`.accesspoint.s3-global.amazonaws.com`

예를 들어, 별칭이 `mfzwi23gnjvgw.mrap`인 다중 리전 액세스 포인트를 통해 `GetObject` 요청을 하려면 `mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com` 호스트 이름에 대해 요

청합니다. 참고: 이 호스트 이름의 s3-global 부분은 이 호스트 이름이 특정 리전에 대한 이름이 아님을 나타냅니다.

다중 리전 액세스 포인트를 통한 요청은 단일 리전 액세스 포인트를 통한 요청과 유사합니다. 하지만 다음과 같은 차이점에 유의하세요.

- 다중 리전 액세스 포인트 ARN에는 AWS 리전이 포함되지 않습니다. 해당 액세스 포인트는 arn:aws:s3::*account-id*:accesspoint/*MultiRegionAccessPoint_alias* 형식을 따릅니다.
- API 작업을 통한 요청의 경우(ARN을 사용할 필요가 없는 요청), 다중 리전 액세스 포인트는 다른 엔드포인트 체계를 사용합니다. 해당 체계는 *MultiRegionAccessPoint_alias*.accesspoint.s3-global.amazonaws.com(예: mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com)입니다. 단일 리전 액세스 포인트와의 차이점에 유의하세요.
  - 다중 리전 액세스 포인트 호스트 이름은 다중 영역 액세스 포인트 이름이 아니라 해당 별칭을 사용합니다.
  - 다중 리전 액세스 포인트 호스트 이름에는 소유자의 AWS 계정 ID가 포함되지 않습니다.
  - 다중 리전 액세스 포인트 호스트 이름에는 AWS 리전이 포함되지 않습니다.
  - 다중 리전 액세스 포인트 호스트 이름에는 s3.amazonaws.com 대신 s3-global.amazonaws.com이 포함됩니다.
- 다중 리전 액세스 포인트 요청은 서명 버전 4A(Sigv4a)를 사용하여 서명해야 합니다. AWS SDK를 사용하는 경우 해당 SDK는 SigV4를 SigV4A로 자동으로 변환합니다. 따라서 AWS SDK가 글로벌 AWS 리전 요청에 서명하는 데 사용되는 서명 구현으로 SigV4A를 지원해야 합니다. SigV4A에 대한 자세한 내용은 AWS 일반 참조의 AWS API 요청에 서명을 참조하세요.

다중 리전 액세스 포인트 및 Amazon S3 Transfer Acceleration

Amazon S3 Transfer Acceleration은 버킷에 데이터를 더욱 빠르게 전송할 수 있는 기능입니다. Transfer Acceleration은 개별 버킷 수준에서 구성됩니다. Transfer Acceleration에 대한 자세한 내용은 Amazon S3 Transfer Acceleration을 사용하여 빠르고 안전한 파일 전송 구성 단원을 참조하세요.

다중 리전 액세스 포인트는 AWS 네트워크를 통해 크기가 큰 객체를 보내기 위해 Transfer Acceleration과 유사한 가속화된 전송 메커니즘을 사용합니다. 따라서 다중 리전 액세스 포인트를 통해 요청을 보낼 때 더Transfer Acceleration을 사용할 필요가 없습니다. 이렇게 향상된 전송 성능이 다중 리전 액세스 포인트에 자동으로 통합되어 있습니다.

주제

- 권한
- 다중 리전 액세스 포인트 규제 및 제한
- 다중 리전 액세스 포인트 요청 라우팅
- Amazon S3 다중 리전 액세스 포인트 장애 조치 제어
- 다중 리전 액세스 포인트와 함께 사용할 복제 구성
- 지원되는 API 작업으로 다중 리전 액세스 포인트 사용
- 다중 리전 액세스 포인트를 통해 기본 리소스에 대한 요청 모니터링 및 로깅

권한

Amazon S3 다중 리전 액세스 포인트는 여러 AWS 리전에 있는 Amazon S3 버킷에 대한 데이터 액세스를 간소화할 수 있습니다. 다중 리전 액세스 포인트는 `GetObject` 및 `PutObject` 등의 Amazon S3 데이터 액세스 객체 작업을 수행하는 데 사용할 수 있는 글로벌 엔드포인트입니다. 각 다중 리전 액세스 포인트는 글로벌 엔드포인트를 통해 이루어진 모든 요청에 대해 고유한 권한 및 네트워크 제어를 가질 수 있습니다.

또한 각 다중 리전 액세스 포인트는 기본 버킷에 연결된 버킷 정책과 함께 작동하는 사용자 지정 액세스 정책을 적용합니다. 교차 계정 요청이 성공하려면 다음 정책에서 해당 작업을 허용해야 합니다.

- 다중 리전 액세스 포인트 정책
- 기본 AWS Identity and Access Management(IAM) 정책
- 기본 버킷 정책(요청이 라우팅되는 대상 위치)

> **Note**
>
> 동일한 계정 요청의 경우 적절한 액세스 권한을 부여하는 기본 IAM 정책만 필요합니다.

특정 IAM 사용자 또는 그룹의 요청만 수락하도록 다중 리전 액세스 포인트 정책을 구성할 수 있습니다. 이렇게 하는 방법의 예는 the section called “다중 리전 액세스 포인트 정책 예시”의 예시 2를 참조하세요. Virtual Private Cloud(VPC)의 요청만 수락하도록 다중 리전 액세스 포인트를 구성하여 프라이빗 네트워크에 대한 Amazon S3 데이터 액세스를 제한할 수 있습니다.

예를 들어, AWS 계정의 `AppDataReader` 사용자를 이용하여 다중 리전 액세스 포인트를 통해 `GetObject` 요청을 한다고 가정하겠습니다. 요청이 거부되지 않도록 하려면 다중 리전 액세스 포인트

및 다중 리전 액세스 포인트의 기본 버킷 각각이 `s3:GetObject` 권한을 `AppDataReader` 사용자에게 부여해야 합니다. `AppDataReader`는 이 권한을 부여하지 않는 버킷에서 데이터를 검색할 수 없습니다.

> **Important**
>
> 버킷에 대한 액세스 제어를 위임해도 직접 버킷 이름을 사용하거나 Amazon 리소스 이름(ARN)을 통해 버킷에 액세스할 때 버킷의 동작은 변경되지 않습니다. 버킷에 대한 모든 직접적인 작업은 여전히 이전과 같이 작동합니다. 다중 리전 액세스 포인트 정책에 포함하는 제한은 해당 다중 리전 액세스 포인트를 통해 이루어진 요청에만 적용됩니다.

다중 리전 액세스 포인트에 대한 퍼블릭 액세스 관리

다중 리전 액세스 포인트는 각 다중 리전 액세스 포인트에 대해 독립적인 퍼블릭 액세스 차단 설정을 지원합니다. 다중 리전 액세스 포인트를 생성할 때 해당 다중 리전 액세스 포인트에 적용되는 퍼블릭 액세스 차단 설정을 지정할 수 있습니다.

> **Note**
>
> 이 계정의 퍼블릭 액세스 차단 설정(사용자 계정) 또는 외부 버킷에 대한 퍼블릭 액세스 차단 설정에서 활성화된 퍼블릭 액세스 차단 설정은 다중 리전 액세스 포인트에 대한 독립적인 퍼블릭 액세스 차단 설정이 비활성화된 경우에도 계속 적용됩니다.

다중 리전 액세스 포인트를 통해 이루어진 요청에 대해 Amazon S3는 다음에 대한 퍼블릭 액세스 차단 설정을 평가합니다.

- 다중 리전 액세스 포인트
- 기본 버킷(외부 버킷 포함)
- 다중 리전 액세스 포인트를 모두 소유하는 계정
- 기본 버킷을 소유하는 계정(외부 계정 포함)

이러한 설정 중 하나라도 요청이 차단되어야 함을 나타내는 경우, Amazon S3은 요청을 거부합니다. Amazon S3 퍼블릭 액세스 차단 기능에 대한 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하세요.

> **Important**
>
> 다중 리전 액세스 포인트에는 기본값으로 모든 퍼블릭 액세스 차단 설정이 사용 설정되어 있습니다. 다중 리전 액세스 포인트에 적용하지 않으려는 모든 설정을 명시적으로 꺼야 합니다. 다중 리전 액세스 포인트를 만든 후에는 다중 리전 액세스 포인트의 퍼블릭 액세스 차단 설정을 변경할 수 없습니다.

다중 리전 액세스 포인트에 대한 퍼블릭 액세스 차단 설정 보기

다중 리전 액세스 포인트에 대한 퍼블릭 액세스 차단 설정을 보는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 다중 리전 액세스 포인트)를 선택합니다.
3. 검토하려는 다중 리전 액세스 포인트의 이름을 선택합니다.
4. 권한 탭을 선택합니다.
5. Block Public Access settings for this Multi-Region Access Point(이 다중 리전 액세스 포인트에 대한 퍼블릭 액세스 차단 설정)에서 다중 리전 액세스 포인트에 적용할 퍼블릭 액세스 차단 설정을 선택합니다.

> **Note**
>
> 다중 리전 액세스 포인트를 생성한 후에는 퍼블릭 액세스 차단 설정을 편집할 수 없습니다. 따라서 퍼블릭 액세스를 차단하려면 다중 리전 액세스 포인트를 생성하기 전에 퍼블릭 액세스 없이 애플리케이션이 제대로 작동하는지 확인하세요.

다중 리전 액세스 포인트 정책 사용

다음 다중 리전 액세스 포인트 정책 예시는 IAM 사용자에게 다중 리전 액세스 포인트에서 파일을 나열하고 다운로드할 수 있는 액세스 권한을 부여합니다. 이 정책 예를 사용하려면 *user input placeholders*를 실제 정보로 바꾸세요.

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

```
      {
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::123456789012:user/JohnDoe"
         },
         "Action":[
            "s3:ListBucket",
            "s3:GetObject"
         ],
         "Resource":[
            "arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias",
            "arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias/object/
*"
         ]
      }
   ]
}
```

AWS Command Line Interface(AWS CLI)를 사용하여 다중 리전 액세스 포인트 정책을 지정된 다중 리전 액세스 포인트에 연결하려면 다음 `put-multi-region-access-point-policy` 명령을 사용하세요. 이 예 명령을 사용하려면 *`user input placeholders`*를 실제 정보로 대체하세요. 각 다중 리전 액세스 포인트는 하나의 정책만 가질 수 있으므로 `put-multi-region-access-point-policy` 작업에 대한 요청은 지정된 다중 리전 액세스 포인트와 연결된 기존 정책을 대체합니다.

AWS CLI

```
aws s3control put-multi-region-access-point-policy
--account-id 111122223333
--details { "Name": "amzn-s3-demo-bucket-MultiRegionAccessPoint",
 "Policy": "{ \"Version\": \"2012-10-17\", \"Statement\": { \"Effect\":
 \"Allow\", \"Principal\": { \"AWS\": \"arn:aws:iam::111122223333:root
\" }, \"Action\": [\"s3:ListBucket\", \"s3:GetObject\"], \"Resource\":
 [ \"arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias",
 \"arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias/object/*
\" ] } }" }
```

이전 작업에 대한 결과를 쿼리하려면 다음 명령을 사용하세요.

AWS CLI

```
aws s3control describe-multi-region-access-point-operation
```

```
--account-id 111122223333
--request-token-arn requestArn
```

다중 리전 액세스 포인트 정책을 검색하려면 다음 명령을 사용하세요.

AWS CLI

```
aws s3control get-multi-region-access-point-policy
--account-id 111122223333
--name=amzn-s3-demo-bucket-MultiRegionAccessPoint
```

다중 리전 액세스 포인트 정책 편집

다중 리전 액세스 포인트 정책(JSON으로 작성됨)은 이 다중 리전 액세스 포인트와 함께 사용되는 Amazon S3 버킷에 대한 스토리지 액세스를 제공합니다. 특정 주체가 다중 리전 액세스 포인트에서 다양한 작업을 수행하도록 허용하거나 거부할 수 있습니다. 요청이 다중 리전 액세스 포인트를 통해 버킷에 라우팅되면 다중 리전 액세스 포인트와 버킷의 액세스 정책이 모두 적용됩니다. 더 제한적인 액세스 정책이 항상 우선합니다.

> **Note**
>
> 버킷에 다른 계정이 소유한 객체가 포함된 경우 다중 리전 액세스 포인트 정책은 다른 AWS 계정이 소유한 객체에는 적용되지 않습니다.

다중 리전 액세스 포인트 정책을 적용한 후에는 정책을 삭제할 수 없습니다. 정책을 편집하거나 기존 정책을 덮어쓰는 새 정책을 만들 수 있습니다.

다중 리전 액세스 포인트 정책을 편집하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 다중 리전 액세스 포인트)를 선택합니다.
3. 정책을 편집할 다중 리전 액세스 포인트의 이름을 선택합니다.
4. 권한 탭을 선택합니다.
5. 아래로 스크롤하여 다중 리전 액세스 포인트 정책 섹션으로 이동합니다. 편집을 선택하여 정책을 업데이트합니다(JSON으로).

6. Edit Multi-Region Access Point policy(다중 리전 액세스 포인트 정책 편집) 페이지가 표시됩니다. 정책을 텍스트 필드에 직접 입력하거나, Add statement(문 추가)를 선택하여 드롭다운 목록에서 정책 요소를 선택할 수 있습니다.

> **Note**
>
> 콘솔에는 정책에서 사용할 수 있는 다중 리전 액세스 포인트의 Amazon 리소스 이름(ARN)이 자동으로 표시됩니다. 다중 리전 액세스 포인트 정책 예시는 the section called “다중 리전 액세스 포인트 정책 예시”를 참조하세요.

다중 리전 액세스 포인트 정책 예시

Amazon S3 다중 리전 액세스 포인트가 AWS Identity and Access Management(IAM) 리소스 정책을 지원합니다. 이러한 정책을 사용하여 리소스, 사용자 또는 기타 조건별로 다중 리전 액세스 포인트 사용을 제어할 수 있습니다. 애플리케이션 또는 사용자가 다중 리전 액세스 포인트를 통해 객체에 액세스할 수 있으려면 다중 리전 액세스 포인트와 기본 버킷 모두에서 동일한 액세스를 허용해야 합니다.

다중 리전 액세스 포인트와 기본 버킷에 대해 동일한 액세스를 허용하려면 다음 중 하나를 수행하세요.

- (권장) Amazon S3 다중 리전 액세스 포인트를 사용할 때 액세스 제어를 단순화하려면 Amazon S3 버킷에 대한 액세스 제어를 다중 리전 액세스 포인트에 위임하세요. 이렇게 하는 방법의 예는 이 섹션의 예시 1을 참조하세요.
- 다중 리전 액세스 포인트 정책에 포함된 동일한 권한을 기본 버킷 정책에 추가합니다.

> **Important**
>
> 버킷에 대한 액세스 제어를 위임해도 직접 버킷 이름을 사용하거나 Amazon 리소스 이름(ARN)을 통해 버킷에 액세스할 때 버킷의 동작은 변경되지 않습니다. 버킷에 대한 모든 직접적인 작업은 여전히 이전과 같이 작동합니다. 다중 리전 액세스 포인트 정책에 포함하는 제한은 해당 다중 리전 액세스 포인트를 통해 이루어진 요청에만 적용됩니다.

Example 1 - 버킷 정책에서 특정 다중 리전 액세스 포인트에 대한 액세스 권한 위임(동일한 계정 또는 크로스 계정에 대해)

다음 버킷 정책 예시는 특정 다중 리전 액세스 포인트에 대한 전체 버킷 액세스 권한을 부여합니다. 즉, 이 버킷에 대한 모든 액세스가 해당 다중 리전 액세스 포인트에 연결된 정책으로 제어됨을 의미합니다.

버킷에 직접 액세스할 필요가 없는 모든 사용 사례에 대해 이 방법으로 버킷을 구성하는 것이 좋습니다. 동일한 계정 또는 다른 계정의 다중 리전 액세스 포인트에 대해 이 버킷 정책 구조를 사용할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS": "*" },
        "Action" : "*",
        "Resource" : [ "Bucket ARN", "Bucket ARN/*"],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointArn" : "MultiRegionAccessPoint_ARN" }
        }
    }]
}
```

> **Note**
>
> 액세스 권한을 부여하려는 다중 리전 액세스 포인트가 여러 개 있는 경우 각 다중 리전 액세스 포인트를 나열해야 합니다.

Example 2 - 다중 리전 액세스 포인트 정책에서 다중 리전 액세스 포인트에 대한 계정 액세스 권한 부여

다음 다중 리전 액세스 포인트 정책은 *MultiRegionAccessPoint_ARN*으로 정의된 다중 리전 액세스 포인트에 포함된 객체를 나열하고 읽을 수 있는 권한을 계정 *123456789012*에 허용합니다.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
       "Effect": "Allow",
       "Principal": {
          "AWS":"arn:aws:iam::123456789012:user/JohnDoe"
       },
       "Action":[
          "s3:ListBucket",
          "s3:GetObject"
```

```
        ],
        "Resource":[
           "MultiRegionAccessPoint_ARN",
           "MultiRegionAccessPoint_ARN/object/*"
        ]
     }
  ]
}
```

Example 3 - 버킷 나열을 허용하는 다중 리전 액세스 포인트 정책

다음 다중 리전 액세스 포인트 정책은 *MultiRegionAccessPoint_ARN*으로 정의된 다중 리전 액세스 포인트에 포함된 객체를 나열할 수 있는 권한을 계정 *123456789012*에 허용합니다.

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
       "Effect": "Allow",
       "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/JohnDoe"
        },
        "Action": "s3:ListBucket",
        "Resource": "MultiRegionAccessPoint_ARN"
       }
    ]
}
```

다중 리전 액세스 포인트 규제 및 제한

Amazon S3의 다중 리전 액세스 포인트에는 다음과 같은 규제 및 제한 사항이 있습니다.

- 다중 리전 액세스 포인트 이름:
  - 하나의 AWS 계정 내에서 고유해야 함
  - 숫자 또는 소문자로 시작해야 함
  - 3~50자 이내여야 함
  - 하이픈(-)으로 시작하거나 끝날 수 없음
  - 밑줄(_), 대문자 또는 마침표(.)를 포함할 수 없음
  - 생성한 후에는 편집할 수 없음
- 다중 리전 액세스 포인트 별칭은 Amazon S3에서 생성되며 편집하거나 재사용할 수 없습니다.

- 게이트웨이 엔드포인트를 사용하여 다중 리전 액세스 포인트를 통해 데이터에 액세스할 수 없습니다. 그러나 인터페이스 엔드포인트를 사용하여 다중 리전 액세스 포인트를 통해 데이터에 액세스할 수 있습니다. AWS PrivateLink를 사용하려면 VPC 엔드포인트를 만들어야 합니다. 자세한 내용은 AWS PrivateLink와 함께 사용할 다중 리전 액세스 포인트 구성 섹션을 참조하세요.
- Amazon CloudFront에서 다중 리전 액세스 포인트를 사용하려면 다중 리전 액세스 포인트를 `Custom Origin` 배포 유형으로 구성해야 합니다. 다양한 오리진 유형에 대한 자세한 내용은 CloudFront 배포에 다양한 원본 사용을 참조하세요. Amazon CloudFront에서 다중 리전 액세스 포인트를 사용하는 방법에 대한 자세한 내용은 AWS 스토리지 블로그에서 다중 리전에 걸쳐 액티브-액티브, 근접성 기반 애플리케이션 구축을 참조하세요.
- 다중 리전 액세스 포인트 최소 요구 사항:
  - 전송 계층 보안(TLS) v1.2

    > **Note**
    >
    > Transport Layer Security(TLS) v1.3은 지원되지 않습니다.

  - 서명 버전 4(SigV4A)

    다중 리전 액세스 포인트는 서명 버전 4A 를 지원합니다. 이 버전의 SIGv4를 사용하면 여러 AWS 리전에 대한 요청에 서명할 수 있습니다. 이 기능은 여러 리전 중 하나에서 데이터 액세스가 발생할 수 있는 API 작업에 유용합니다. AWS SDK를 사용하는 경우 보안 인증 정보를 제공하면 다중 리전 액세스 포인트에 대한 요청은 추가 구성 없이 서명 버전 4A를 사용합니다. AWS SDK가 SigV4A 알고리즘과 호환되는지 확인하세요. SigV4A에 대한 자세한 내용은 AWS 일반 참조의 AWS API 요청에 서명을 참조하세요.

    > **Note**
    >
    > AWS Identity and Access Management(IAM) 역할을 사용하는 경우와 같이 임시 보안 자격 증명과 함께 SigV4A를 사용하려면 리전의 AWS Security Token Service(AWS STS) 엔드포인트에서 임시 자격 증명을 요청하면 됩니다. 글로벌 AWS STS 엔드포인트(`sts.amazonaws.com`)에서 임시 자격 증명을 요청하는 경우 먼저 글로벌 엔드포인트에 대한 세션 토큰의 리전 호환성이 모든 AWS 리전에서 유효하도록 설정해야 합니다. 자세한 내용은 IAM 사용 설명서의 AWS 리전에서 AWS STS 관리를 참조하세요.

- 다중 리전 액세스 포인트는 익명 요청을 지원하지 않습니다.
- 다중 리전 액세스 포인트 제한:
  - IPv6은 지원되지 않습니다.

  - Amazon S3 on Outposts 버킷은 지원되지 않습니다.
  - 다중 리전 액세스 포인트는 다중 리전 액세스 포인트 ARN을 사용할 때 목적지로만 다중 리전 액세스 포인트를 사용한 복사 작업을 지원합니다.
  - S3 배치 작업 기능은 지원되지 않습니다.
- 특정 AWS SDK는 지원되지 않습니다. 다중 리전 액세스 포인트에서 지원되는 AWS SDK를 확인하려면 AWS SDK와의 호환성을 참조하세요.
- 다중 리전 액세스 포인트와 함께 사용할 서비스 할당량은 다음과 같습니다.
  - 계정당 최대 100개의 다중 리전 액세스 포인트로 제한됩니다.
  - 하나의 다중 리전 액세스 포인트에는 17개의 리전으로 제한됩니다.
- 다중 리전 액세스 포인트를 생성한 후에는 다중 리전 액세스 포인트 구성에서 버킷을 추가, 수정 또는 제거할 수 없습니다. 버킷을 변경하려면 전체 다중 리전 액세스 포인트를 삭제하고 새 액세스 포인트를 생성해야 합니다. 다중 리전 액세스 포인트의 크로스 계정 버킷이 삭제된 경우 이 버킷을 다시 연결하는 유일한 방법은 해당 계정에서 동일한 이름 및 리전을 사용하여 버킷을 다시 만드는 것입니다.
- 다중 리전 액세스 포인트에서 사용하는 기본 버킷(동일 계정의 버킷)은 해당 다중 리전 액세스 포인트를 삭제한 후에만 삭제할 수 있습니다.
- 다중 리전 액세스 포인트를 생성하거나 유지하기 위한 모든 모든 컨트롤 플레인 요청은 `US West (Oregon)` 리전으로 라우팅되어야 합니다. 다중 리전 액세스 포인트 데이터 영역 요청의 경우 리전을 지정할 필요가 없습니다. 컨트롤 플레인 작업 목록은 Amazon Simple Storage Service API 참조에서 S3 API Reference를 참조하세요.
- 다중 리전 액세스 포인트 장애 조치 컨트롤 플레인의 경우 지원되는 다음 다섯 개 리전 중 하나로 요청을 라우팅해야 합니다.
  - `US East (N. Virginia)`
  - `US West (Oregon)`
  - `Asia Pacific (Sydney)`
  - `Asia Pacific (Tokyo)`
  - `Europe (Ireland)`
- 다중 리전 액세스 포인트는 다음 AWS 리전의 버킷만 지원합니다.
  - `US East (N. Virginia)`
  - `US East (Ohio)`
  - `US West (N. California)`
  - `US West (Oregon)`

- Asia Pacific (Mumbai)
- Asia Pacific (Osaka)
- Asia Pacific (Seoul)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Canada (Central)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Paris)
- Europe (Stockholm)
- South America (São Paulo)

다중 리전 액세스 포인트 요청 라우팅

다중 리전 액세스 포인트를 통해 요청하면 Amazon S3는 다중 리전 액세스 포인트와 연결된 버킷 중 가장 가까운 버킷을 판단합니다. 그런 다음 Amazon S3는 해당 버킷이 있는 AWS 리전에 관계없이 요청을 해당 버킷에 보냅니다.

다중 리전 액세스 포인트가 가장 가까운 버킷으로 요청을 라우팅하면 Amazon S3는 사용자가 해당 버킷에 직접 요청한 것처럼 요청을 처리합니다. 다중 리전 액세스 포인트는 Amazon S3 버킷의 데이터 콘텐츠를 인식하지 못합니다. 따라서 요청을 받는 버킷에는 요청된 데이터가 포함되어 있지 않을 수 있습니다. 하나의 다중 리전 액세스 포인트와 연결된 Amazon S3 버킷에 일관된 데이터 세트를 생성하려면 S3 크로스 리전 복제(CRR)를 구성할 수 있습니다. 그러면 모든 버킷이 요청을 성공적으로 이행할 수 있습니다.

Amazon S3는 다음 규칙에 따라 다중 리전 액세스 포인트 요청을 보냅니다.

- Amazon S3는 근접성에 따라 처리되도록 요청을 최적화합니다. 다중 리전 액세스 포인트가 지원하는 버킷을 살펴보고 가장 가까운 버킷으로 요청을 전달합니다.
- 요청이 기존 리소스를 지정하는 경우(예: `GetObject`), Amazon S3는 요청을 이행할 때 객체의 이름을 고려하지 않습니다. 즉, 객체가 다중 리전 액세스 포인트의 버킷 하나에 존재하더라도 요청이 객체를 포함하지 않는 버킷으로 라우팅될 수 있다는 뜻입니다. 그러면 404 오류 메시지가 클라이언트에 반환됩니다.

404 오류를 방지하려면 버킷에 S3 크로스 리전 복제(CRR)를 구성하는 것이 좋습니다. 복제는 원하는 객체가 다중 리전 액세스 포인트의 버킷에 있지만 요청이 라우팅된 특정 버킷에 없는 경우의 잠재적인 문제를 해결하는 데 도움이 됩니다. 복제 구성에 대한 자세한 내용은 다중 리전 액세스 포인트와 함께 사용할 복제 구성 섹션을 참조하세요.

원하는 특정 객체를 사용하여 요청이 이행되도록 하려면 버킷 버전 관리를 설정하고 요청에 버전 ID를 포함하는 것이 좋습니다. 그러면 찾고 있는 객체의 올바른 버전이 있는지 확인할 수 있습니다. 버전 관리가 활성화된 버킷을 사용하는 것도 실수로 덮어쓴 객체를 복구하는 데 도움이 될 수 있습니다. 자세한 내용은 S3 버킷에서 버전 관리 사용을 참조하세요.

- 리소스 생성 요청인 경우(예: `PutObject` 또는 `CreateMultipartUpload`) Amazon S3는 가장 가까운 버킷을 사용하여 요청을 이행합니다. 예를 들어, 전 세계 어디서나 동영상 업로드를 지원하고자 하는 동영상 회사를 가정해 보겠습니다. 다중 리전 액세스 포인트에 PUT 요청을 하면 객체가 가장 가까운 버킷에 배치됩니다. 그런 다음 업로드된 비디오를 전 세계 다른 사람들이 최저 지연 시간으로 다운로드할 수 있도록 양방향 복제와 함께 CRR을 사용하면 됩니다. 양방향 복제와 CRR을 사용하면 다중 리전 액세스 포인트와 연결된 모든 버킷의 콘텐츠가 동기화된 상태로 유지됩니다. 다중 리전 액세스 포인트를 사용한 복제에 대한 자세한 내용은 다중 리전 액세스 포인트와 함께 사용할 복제 구성 섹션을 참조하세요.

Amazon S3 다중 리전 액세스 포인트 장애 조치 제어

Amazon S3 다중 리전 액세스 포인트 장애 조치 제어를 사용하면 리전의 트래픽이 중단되어도 비즈니스 연속성을 유지하는 동시에 애플리케이션에 다중 리전 아키텍처를 제공하여 규정 준수 및 이중화 요구 사항을 충족할 수 있습니다. 리전의 트래픽이 중단되는 경우 다중 리전 액세스 포인트 장애 조치 제어를 사용하여 Amazon S3 다중 리전 액세스 포인트 뒤에 있는 어떤 AWS 리전에서 데이터 액세스와 스토리지 요청을 처리할지 선택할 수 있습니다.

장애 조치를 지원하기 위해 정상 상태일 때는 트래픽이 액티브 리전으로 흐르게 하고 패시브 리전은 장애 조치를 위해 대기 상태로 두는 액티브-패시브 구성으로 다중 리전 액세스 포인트를 설정할 수 있습니다.

예를 들어 원하는 AWS 리전으로 장애 조치를 수행하려면 트래픽을 기본(액티브) 리전에서 보조(패시브) 리전으로 이동합니다. 이와 같은 액티브-패시브 구성에서는 한 버킷이 액티브 상태로 트래픽을 받아들이고 다른 버킷은 패시브 상태로 트래픽을 받아들이지 않습니다. 패시브 버킷은 재해 복구에 사용됩니다. 장애 조치를 시작하면 모든 트래픽(예: `GET` 또는 `PUT` 요청)이 한 리전에 있는 액티브 상태의 버킷으로 전달되고 다른 리전에 있는 패시브 상태의 버킷에서 멀어집니다.

양방향 복제 규칙으로 S3 크로스 리전 복제(CRR)를 사용 설정한 경우 장애 조치 중에 버킷을 동기화된 상태로 유지할 수 있습니다. 또한 액티브-액티브 구성에서 CRR을 사용 설정한 경우 Amazon S3 다중 리전 액세스 포인트는 가장 가까운 버킷 위치에서 데이터를 가져올 수 있으며, 이 경우 애플리케이션 성능이 향상됩니다.

AWS 리전 지원

Amazon S3 다중 리전 액세스 포인트 장애 조치 제어를 사용하면 다중 리전 액세스 포인트가 지원되는 17개 리전 중 어디에나 S3 버킷을 배치할 수 있습니다. 한 번에 두 리전에서 장애 조치를 시작할 수 있습니다.

> **Note**
>
> 장애 조치는 한 번에 두 리전 사이에서만 시작되지만 다중 리전 액세스 포인트에서 동시에 여러 리전의 라우팅 상태를 개별적으로 업데이트할 수 있습니다.

다음 주제에서는 Amazon S3 다중 리전 액세스 포인트 장애 조치 제어를 사용하고 관리하는 방법을 설명합니다.

주제

- Amazon S3 다중 리전 액세스 포인트 라우팅 상태
- Amazon S3 다중 리전 액세스 포인트 장애 조치 제어 사용
- Amazon S3 다중 리전 액세스 포인트 장애 조치 제어 오류

Amazon S3 다중 리전 액세스 포인트 라우팅 상태

Amazon S3 다중 리전 액세스 포인트 장애 조치 구성은 다중 리전 액세스 포인트와 함께 사용되는 AWS 리전의 라우팅 상태를 결정합니다. Amazon S3 다중 리전 액세스 포인트를 액티브-액티브 상태 또는 액티브-패시브 상태로 구성할 수 있습니다.

- 액티브-액티브 - 액티브-액티브 구성에서는 모든 요청이 다중 리전 액세스 포인트의 가장 가까운 AWS 리전으로 자동 전송됩니다. 다중 리전 액세스 포인트가 액티브-액티브 상태로 구성된 후에는 모든 리전에서 트래픽을 수신할 수 있습니다. 액티브-액티브 구성에서 트래픽 중단이 발생하면 네트워크 트래픽이 액티브 리전 중 하나로 자동 리디렉션됩니다.
- 액티브-패시브 - 액티브-패시브 구성에서 다중 리전 액세스 포인트의 액티브 리전은 트래픽을 수신하고 패시브 리전은 수신하지 않습니다. 재해 상황에서 S3 장애 조치 제어를 사용하여 장애 조치를

시작하려는 경우 재해 복구 계획을 테스트하고 수행하는 동안 다중 리전 액세스 포인트를 액티브-패시브 구성으로 설정하세요.

Amazon S3 다중 리전 액세스 포인트 장애 조치 제어 사용

이 섹션에서는 AWS Management Console을 사용하여 Amazon S3 다중 리전 액세스 포인트 장애 조치 제어를 관리하고 사용하는 방법을 설명합니다.

AWS Management Console의 다중 리전 액세스 포인트 세부 정보 페이지에 있는 Failover configuration(장애 조치 구성) 섹션에는 Edit routing status(라우팅 상태 편집) 및 Failover(장애 조치)라는 두 가지 장애 조치 제어가 있습니다. 이러한 제어는 다음과 같이 사용할 수 있습니다.

- Edit routing status(라우팅 상태 편집) - Edit routing status(라우팅 상태 편집)를 선택하여 다중 리전 액세스 포인트에 대한 하나의 요청에서 최대 17개 AWS 리전의 라우팅 상태를 수동으로 편집할 수 있습니다. 다음과 같은 목적으로 Edit routing status(라우팅 상태 편집)를 사용할 수 있습니다.
  - 다중 리전 액세스 포인트에서 하나 이상의 리전에 대한 라우팅 상태를 설정하거나 편집하기 위해
  - 두 리전을 액티브-패시브 상태로 구성하여 다중 리전 액세스 포인트에 대한 장애 조치 구성을 만들기 위해
  - 리전을 수동으로 장애 조치하기 위해
  - 리전 간 트래픽을 수동으로 전환하기 위해
- Failover(장애 조치) - Failover(장애 조치)를 선택하여 장애 조치를 시작하면 이미 액티브-패시브 상태로 구성된 두 리전의 라우팅 상태만 업데이트됩니다. Failover(장애 조치)를 선택하여 시작한 장애 조치가 진행되는 동안 두 리전 간의 라우팅 상태가 자동으로 전환됩니다.

다중 리전 액세스 포인트에서 리전의 라우팅 상태 편집

다중 리전 액세스 포인트 세부 정보 페이지의 Failover configuration(장애 조치 구성) 섹션에서 Edit routing status(라우팅 상태 편집)를 선택하여 다중 리전 액세스 포인트에 대한 하나의 요청으로 최대 17개 AWS 리전의 라우팅 상태를 수동으로 업데이트할 수 있습니다. 그러나 Failover(장애 조치)를 선택하여 장애 조치를 시작하면 이미 액티브-패시브 상태로 구성된 두 리전의 라우팅 상태만 업데이트됩니다. Failover(장애 조치)를 선택하여 시작한 장애 조치가 진행되는 동안 두 리전 간의 라우팅 상태가 자동으로 전환됩니다.

다음 절차에 설명된 대로 Edit routing status(라우팅 상태 편집)를 다음과 같은 목적으로 사용할 수 있습니다.

- 다중 리전 액세스 포인트에서 하나 이상의 리전에 대한 라우팅 상태를 설정하거나 편집하기 위해

- 두 리전을 액티브-패시브 상태로 구성하여 다중 리전 액세스 포인트에 대한 장애 조치 구성을 만들기 위해
- 리전을 수동으로 장애 조치하기 위해
- 리전 간 트래픽을 수동으로 전환하기 위해

S3 콘솔 사용

다중 리전 액세스 포인트에서 리전의 라우팅 상태를 업데이트하는 방법

1. AWS Management Console에 로그인합니다.
2. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
3. 왼쪽 탐색 창에서 다중 리전 액세스 포인트를 선택합니다.
4. 업데이트하려는 다중 리전 액세스 포인트를 선택합니다.
5. Replication and failover(복제 및 장애 조치) 탭을 선택합니다.
6. 라우팅 상태를 편집할 리전을 하나 이상 선택합니다.

   > **Note**
   >
   > 장애 조치를 시작하려면 다중 리전 액세스 포인트에서 최소한 한 개의 AWS 리전을 Active(액티브)로 지정하고 한 리전을 Passive(패시브)로 지정해야 합니다.

7. Edit routing status(라우팅 상태 편집)를 선택합니다.
8. 나타나는 대화 상자에서 각 리전의 Routing status(라우팅 상태)를 Active(액티브) 또는 Passive(패시브)로 선택합니다.

   액티브 상태에서는 트래픽을 리전으로 라우팅할 수 있습니다. 패시브 상태에서는 리전으로의 트래픽 전달이 중지됩니다.

   다중 리전 액세스 포인트의 장애 조치 구성을 만들거나 장애 조치를 시작하려면 다중 리전 액세스 포인트에서 최소한 한 개의 AWS 리전을 Active(액티브)로 지정하고 한 리전을 Passive(패시브)로 지정해야 합니다.

9. Save routing status(라우팅 상태 저장)를 선택합니다. 트래픽이 리디렉션되는 데 약 2분이 걸립니다.

다중 리전 액세스 포인트에 대한 AWS 리전의 라우팅 상태를 제출한 후 라우팅 상태 변경 사항을 확인할 수 있습니다. 이러한 변경 사항을 확인하려면 Amazon CloudWatch(https://console.aws.amazon.com/cloudwatch/)로 이동하여 액티브 및 패시브 리전 간의 Amazon S3 데이터 요청 트래픽(예: GET 및 PUT 요청) 이동 변화를 모니터링하세요. 기존 연결은 장애 조치 중에 종료되지 않습니다. 기존 연결은 성공 또는 실패 상태에 도달할 때까지 계속됩니다.

AWS CLI 사용

> **Note**
>
> 다음 5개 리전 중 하나에 대해 다중 리전 액세스 포인트 AWS CLI 라우팅 명령을 실행할 수 있습니다.
>
> - ap-southeast-2
> - ap-northeast-1
> - us-east-1
> - us-west-2
> - eu-west-1

다음 예시 명령은 기존 다중 리전 액세스 포인트 라우팅 구성을 업데이트합니다. 버킷의 액티브 또는 패시브 상태를 업데이트하려면 TrafficDialPercentage 값을 액티브 상태의 경우 100, 패시브 상태의 경우 0으로 설정하세요. 이 예시에서는 *amzn-s3-demo-bucket1*이 액티브로, *amzn-s3-demo-bucket2*가 패시브로 설정되어 있습니다. 이 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3control submit-multi-region-access-point-routes
--region ap-southeast-2
--account-id 123456789012
--mrap MultiRegionAccessPoint_ARN
--route-updates Bucket=amzn-s3-demo-bucket1,TrafficDialPercentage=100
                Bucket=amzn-s3-demo-bucket2
,TrafficDialPercentage=0
```

다음 예시 명령은 업데이트된 다중 리전 액세스 포인트 라우팅 구성을 가져옵니다. 이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3control get-multi-region-access-point-routes
```

```
--region eu-west-1
--account-id 123456789012
--mrap MultiRegionAccessPoint_ARN
```

장애 조치 시작

다중 리전 액세스 포인트 세부 정보 페이지의 Failover configuration(장애 조치 구성) 섹션에서 Failover(장애 조치)를 선택하여 장애 조치를 시작하면 Amazon S3 요청 트래픽이 자동으로 대체 AWS 리전으로 이동합니다. 장애 조치 프로세스는 2분 이내에 완료됩니다.

다중 리전 액세스 포인트가 지원되는 17개 리전 중 한 번에 두 AWS 리전에서 장애 조치를 시작할 수 있습니다. 그러면 장애 조치 이벤트가 AWS CloudTrail에 로깅됩니다. 장애 조치가 완료되면 Amazon CloudWatch에서 새로운 액티브 리전에 대한 Amazon S3 트래픽 및 모든 트래픽 라우팅 업데이트를 모니터링할 수 있습니다.

> **Important**
>
> 데이터 복제 중에 모든 메타데이터와 객체를 버킷 간에 동기화된 상태로 유지하려면 장애 조치 제어를 구성하기 전에 양방향 복제 규칙을 생성하고 복제본 수정 동기화를 사용 설정하는 것이 좋습니다.
>
> 양방향 복제 규칙은 트래픽이 장애 조치 목적지인 Amazon S3 버킷에 데이터를 쓸 때 해당 데이터가 원본 버킷으로 다시 복제되도록 하는 데 도움이 됩니다. 복제본 수정 동기화는 양방향 복제 중에 객체 메타데이터도 버킷 간에 동기화되도록 하는 데 도움이 됩니다.
>
> 장애 조치를 지원하기 위한 복제 구성에 대한 자세한 내용은 the section called “버킷 복제” 섹션을 참조하세요.

복제된 버킷 간 장애 조치를 시작하는 방법

1. AWS Management Console에 로그인합니다.
2. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
3. 왼쪽 탐색 창에서 다중 리전 액세스 포인트)를 선택합니다.
4. 장애 조치를 시작하는 데 사용할 다중 리전 액세스 포인트를 선택합니다.
5. Replication and failover(복제 및 장애 조치) 탭을 선택합니다.
6. 아래로 스크롤하여 Failover configuration(장애 조치 구성) 섹션으로 이동한 다음 두 개의 AWS 리전을 선택합니다.

> **Note**
>
> 장애 조치를 시작하려면 다중 리전 액세스 포인트에서 최소한 한 개의 AWS 리전을 Active(액티브)로 지정하고 한 리전을 Passive(패시브)로 지정해야 합니다. 액티브 상태에서는 트래픽을 리전으로 보낼 수 있습니다. 패시브 상태에서는 리전으로의 트래픽 전달이 중지됩니다.

7. Failover(장애 조치)를 선택합니다.
8. 대화 상자에서 Failover(장애 조치)를 다시 선택하여 장애 조치를 시작합니다. 이 과정에서 두 리전의 라우팅 상태가 자동으로 전환됩니다. 모든 새 트래픽은 액티브 상태가 되는 리전으로 보내지고 패시브 상태가 되는 리전으로는 트래픽 전달이 중지됩니다. 트래픽이 리디렉션되는 데 약 2분이 걸립니다.

   장애 조치 프로세스를 시작한 후 트래픽 변경을 확인할 수 있습니다. 이러한 변경 사항을 확인하려면 Amazon CloudWatch(https://console.aws.amazon.com/cloudwatch/)로 이동하여 액티브 및 패시브 리전 간의 Amazon S3 데이터 요청 트래픽(예: GET 및 PUT 요청) 이동 변화를 모니터링하세요. 기존 연결은 장애 조치 중에 종료되지 않습니다. 기존 연결은 성공 또는 실패 상태에 도달할 때까지 계속됩니다.

Amazon S3 다중 리전 액세스 포인트 라우팅 제어 보기

S3 콘솔 사용

Amazon S3 다중 리전 액세스 포인트의 라우팅 제어를 보는 방법

1. AWS Management Console에 로그인합니다.
2. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
3. 왼쪽 탐색 창에서 다중 리전 액세스 포인트)를 선택합니다.
4. 검토하려는 다중 리전 액세스 포인트를 선택합니다.
5. Replication and failover(복제 및 장애 조치) 탭을 선택합니다. 이 페이지에는 다중 리전 액세스 포인트에 대한 라우팅 구성 세부 정보 및 요약, 관련 복제 규칙, 복제 지표가 표시됩니다. Failover configuration(장애 조치 구성) 섹션에서 리전의 라우팅 상태를 확인할 수 있습니다.

AWS CLI 사용

다음 예시 AWS CLI 명령은 지정된 리전의 현재 다중 리전 액세스 포인트 라우팅 구성을 가져옵니다. 이 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3control get-multi-region-access-point-routes
--region eu-west-1
--account-id 123456789012
--mrap MultiRegionAccessPoint_ARN
```

> **Note**
>
> 이 명령은 다음 5개 리전에 대해서만 실행할 수 있습니다.
>
> - ap-southeast-2
> - ap-northeast-1
> - us-east-1
> - us-west-2
> - eu-west-1

Amazon S3 다중 리전 액세스 포인트 장애 조치 제어 오류

다중 리전 액세스 포인트의 장애 조치 구성을 업데이트하면 다음 오류 중 하나가 발생할 수 있습니다.

- HTTP 400 잘못된 요청: 장애 조치 구성을 업데이트하는 동안 잘못된 다중 리전 액세스 포인트 ARN을 입력하면 이 오류가 발생할 수 있습니다. 다중 리전 액세스 포인트 정책을 검토하여 다중 리전 액세스 포인트 ARN을 확인할 수 있습니다. 다중 리전 액세스 포인트 정책을 검토하거나 업데이트하려면 Editing the Multi-Region Access Point policy(다중 리전 액세스 포인트 정책 편집)를 참조하세요. Amazon S3 다중 리전 액세스 포인트 장애 조치 제어를 업데이트하는 동안 빈 문자열이나 임의의 문자열을 사용하는 경우에도 이 오류가 발생할 수 있습니다. 다음과 같은 다중 리전 액세스 포인트 ARN 형식을 사용하세요.

  `arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias`

- HTTP 503 Slow Down(HTTP 503 속도 저하): 짧은 시간에 너무 많은 요청을 보내는 경우 이 오류가 발생합니다. 요청을 거부하면 오류가 발생합니다.

- HTTP 409 Conflict(HTTP 409 충돌): 이 오류는 두 개 이상의 동시 라우팅 구성 업데이트 요청이 하나의 다중 리전 액세스 포인트를 대상으로 하는 경우 발생합니다. 첫 번째 요청은 성공하지만 다른 요청은 오류와 함께 실패합니다.
- HTTP 405 Method Not Allowed(HTTP 405 메서드 허용되지 않음): 이 오류는 장애 조치를 시작할 때 하나의 AWS 리전과 다중 리전 액세스 포인트를 선택한 경우 발생합니다. 장애 조치를 시작하기 전에 두 리전을 선택하지 않으면 오류가 반환됩니다.

다중 리전 액세스 포인트와 함께 사용할 복제 구성

다중 리전 액세스 포인트 엔드포인트에 요청하면 Amazon S3는 사용자와 가장 가까운 버킷에 요청을 자동으로 라우팅합니다. Amazon S3가 이 결정을 내릴 때 요청의 내용은 고려하지 않습니다. 객체에 GET 요청을 하는 경우 해당 요청이 이 객체의 복사본이 없는 버킷으로 라우팅될 수 있습니다. 이런 경우에는 HTTP 상태 코드 404(찾을 수 없음) 오류 메시지가 수신됩니다. 다중 리전 액세스 포인트 요청 라우팅에 대한 자세한 내용은 the section called “라우팅 요청” 섹션을 참조하세요.

요청을 수신하는 버킷과 관계없이 다중 리전 액세스 포인트가 객체를 검색할 수 있게 하려면 Amazon S3 크로스 리전 복제(CRR)를 구성해야 합니다.

예를 들어, 다음의 세 버킷이 있는 다중 리전 액세스 포인트를 생각해 보세요.

- US West (Oregon) 리전에 있으며 my-image.jpg 객체가 포함된 버킷(이름: *amzn-s3-demo-bucket1*)
- Asia Pacific (Mumbai) 리전에 있으며 my-image.jpg 객체가 포함된 버킷(이름: *amzn-s3-demo-bucket2*)
- 이름이 *amzn-s3-demo-bucket*이고 Europe (Frankfurt) 리전에 있으며 my-image.jpg 객체가 포함되지 않은 버킷

이 상황에서, my-image.jpg 객체에 GetObject 요청을 하면 해당 요청의 성공 여부는 요청을 수신하는 버킷에 따라 달라집니다. Amazon S3는 요청의 내용을 고려하지 않으므로 GetObject 요청을 *amzn-s3-demo-bucket* 버킷으로 라우팅할 수 있습니다(해당 버킷이 가장 가까운 것에 응답하는 경우). 객체가 다중 리전 액세스 포인트의 버킷에 있더라도 요청을 받은 개별 버킷에 객체가 없으므로 404 찾을 수 없음 오류 메시지가 수신됩니다.

크로스 리전 복제(CRR)를 사용하면 이러한 결과를 피할 수 있습니다. 적절한 복제 규칙을 사용하면 my-image.jpg 객체가 *amzn-s3-demo-bucket* 버킷에 복사됩니다. 따라서 Amazon S3가 요청을 해당 버킷으로 라우팅하면 이제 객체를 검색할 수 있습니다.

복제는 다중 리전 액세스 포인트에 할당된 버킷에서 정상적으로 작동합니다. Amazon S3는 다중 리전 액세스 포인트에 있는 버킷으로 특별한 복제 처리를 수행하지 않습니다. 버킷에서 복제를 구성하는 방법에 대한 자세한 내용은 라이브 복제 설정 개요 섹션을 참조하세요.

다중 리전 액세스 포인트와 함께 복제를 사용할 때의 권장 사항

다중 리전 액세스 포인트로 작업할 때 최상의 복제 성능을 위해 다음을 권장합니다.

- S3 Replication Time Control(S3 RTC)을 구성합니다. 예측 가능한 기간 내에 여러 리전 간에 데이터를 복제하려면 S3 RTC를 사용하면 됩니다. S3 RTC는 Amazon S3에 저장된 새 객체의 99.99%를 15분 이내에 복제합니다(서비스 수준 계약에 따라 지원됨). 자세한 내용은 the section called “S3 Replication Time Control 사용” 단원을 참조하십시오. S3 RTC에는 추가 요금이 부과됩니다. 자세한 내용은 Amazon S3 요금을 참조하세요.
- 다중 리전 액세스 포인트를 통해 버킷이 업데이트될 때 버킷의 동기화를 유지할 수 있도록 양방향 복제를 활성화합니다. 자세한 내용은 the section called “다중 리전 액세스 포인트에 양방향 복제 규칙 생성” 단원을 참조하십시오.
- 크로스 계정 다중 리전 액세스 포인트를 생성하여 데이터를 별도의 AWS 계정에 복제합니다. 이 접근 방식은 계정 수준 분리를 제공하므로 소스 버킷이 아닌 다른 리전에 있는 여러 계정에서 데이터에 액세스하고 여러 계정 간에 데이터를 복제할 수 있습니다. 추가 비용 없이 크로스 계정 다중 리전 액세스 포인트를 설정할 수 있습니다. 버킷 소유자이지만 다중 리전 액세스 포인트를 소유하지 않은 경우 데이터 전송 및 요청 비용만 지불하면 됩니다. 다중 리전 액세스 포인트 소유자는 데이터 라우팅 및 인터넷 가속 비용을 지불합니다. 자세한 내용은 Amazon S3 요금을 참조하십시오.
- 각 복제 규칙에 대해 복제본 수정 동기화를 사용 설정하여 객체의 메타데이터 변경 사항도 동기화된 상태로 유지합니다. 자세한 내용은 복제본 수정 동기화 사용 설정을 참조하세요.
- 복제 이벤트를 모니터링하려면 Amazon CloudWatch 메트릭을 사용 설정하세요. CloudWatch 지표 요금이 부과됩니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

주제

- 다중 리전 액세스 포인트에 단방향 복제 규칙 생성
- 다중 리전 액세스 포인트에 양방향 복제 규칙 생성
- 다중 리전 액세스 포인트에 대한 복제 규칙 보기

다중 리전 액세스 포인트에 단방향 복제 규칙 생성

복제는 규칙을 사용하면 버킷 간에 객체를 비동기식으로 자동 복제할 수 있습니다. 단방향 복제 규칙은 데이터가 한 AWS 리전의 소스 버킷에서 다른 리전의 대상 버킷으로 완전히 복제되도록 하는 데 도

움이 됩니다. 단방향 복제를 설정하면 소스 버킷(*amzn-s3-demo-bucket*)에서 대상 버킷(*amzn-s3-demo-bucket*)으로 복제 규칙이 생성됩니다. 다른 복제 규칙과 마찬가지로 단방향 복제 규칙을 전체 Amazon S3 버킷에 적용하거나 접두사 또는 객체 태그로 필터링된 객체의 하위 집합에 적용할 수 있습니다.

> **Important**
>
> 사용자가 대상 버킷의 객체만 소비하는 경우 단방향 복제를 사용하는 것이 좋습니다. 사용자가 대상 버킷의 객체를 업로드하거나 수정할 경우 양방향 복제를 사용하여 모든 버킷을 동기화된 상태로 유지하세요. 또한 장애 조치에 다중 리전 액세스 포인트를 사용할 계획이라면 양방향 복제를 사용하는 것이 좋습니다. 양방향 복제를 설정하려면 the section called “다중 리전 액세스 포인트에 양방향 복제 규칙 생성” 섹션을 참조하세요.

다중 리전 액세스 포인트에 단방향 복제 규칙을 생성하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 다중 리전 액세스 포인트)를 선택합니다.
3. 다중 리전 액세스 포인트의 이름을 선택합니다.
4. Replication and failover(복제 및 장애 조치) 탭을 선택합니다.
5. 복제 규칙까지 아래로 스크롤한 다음 복제 규칙 생성을 선택합니다. 복제 규칙을 생성할 수 있는 충분한 권한이 있는지 확인하세요. 권한이 충분하지 않으면 버전 관리가 비활성화됩니다.

   > **Note**
   >
   > 사용자는 자신의 계정에 있는 버킷에 대해서만 복제 규칙을 생성할 수 있습니다. 외부 버킷에 대한 복제 규칙을 만들려면 버킷 소유자가 해당 버킷에 대한 복제 규칙을 만들어야 합니다.

6. 복제 규칙 생성 페이지에서 하나 이상의 소스 버킷에서 하나 이상의 대상 버킷으로 객체 복제 템플릿을 선택합니다.

   > **Important**
   >
   > 이 템플릿을 사용하여 복제 규칙을 만들면 해당 규칙이 이미 버킷에 할당된 기존 복제 규칙을 대체합니다.

기존 복제 규칙을 대체하는 대신 추가하거나 수정하려면 콘솔에서 각 버킷의 관리 탭으로 이동한 다음 복제 규칙 섹션에서 규칙을 편집하세요. AWS CLI, SDK 또는 REST API를 사용하여 기존 복제 규칙을 추가하거나 수정할 수도 있습니다. 자세한 내용은 복제 구성 파일 요소 단원을 참조하십시오.

7. 소스 및 대상 섹션의 소스 버킷에서 객체를 복제하려는 버킷을 하나 이상 선택합니다. 복제를 위해 선택한 모든 버킷(소스 및 대상)에는 S3 버전 관리가 활성화되어 있어야 하며 각 버킷은 서로 다른 AWS 리전에 있어야 합니다. S3 버전 관리에 대한 자세한 내용은 Amazon S3 버킷에서 버전 관리 사용을 참조하세요.

   대상 버킷에서 객체를 복제하려는 버킷을 하나 이상 선택합니다.

8. 복제 규칙 구성 섹션에서 복제 규칙을 생성할 때 해당 복제 규칙을 활성화할지 비활성화할지를 선택합니다.

   > **Note**
   >
   > 복제 규칙 이름 상자에 이름을 입력할 수 없습니다. 복제 규칙 이름은 복제 규칙을 생성할 때 구성을 기반으로 생성됩니다.

9. 범위 섹션에서 복제에 적합한 범위를 선택합니다.

   - 전체 버킷을 복제하려면 Apply to all objects in the bucket(버킷의 모든 객체에 적용)를 선택합니다.
   - 버킷 내의 객체 중 일부를 복제하려면 Limit the scope of this rule using one or more filters(하나 이상의 필터를 사용하여 이 규칙의 범위 제한)을 선택합니다.

     접두사, 객체 태그 또는 이 두 가지를 모두 사용하여 객체를 필터링할 수 있습니다.

     - 동일한 문자열로 시작하는 이름(예: `pictures`)을 가진 모든 객체로 복제를 제한하려면 접두사 상자에 저두사를 입력합니다.

       폴더의 이름에 해당하는 접두사를 입력할 경우, /(슬래시)와 같은 구분 기호로 계층 수준을 나타내야 합니다(예: `pictures/`) 접두사에 대한 자세한 정보는 접두사를 사용한 객체 구성 섹션을 참조하세요.

     - 하나 이상의 객체 태그가 있는 모든 객체를 복제하려면 태그 추가를 선택하고 상자에 키 값 페어를 입력합니다. 다른 태그를 추가하려면 이 절차를 반복합니다. 객체 태그에 대한 자세한 내용은 태그를 사용하여 스토리지 분류 섹션을 참조하십시오.

10. 아래로 스크롤하여 추가 복제 옵션 섹션으로 이동한 다음 적용할 복제 옵션을 선택합니다.

> **Note**
>
> 다음 옵션을 적용하는 것이 좋습니다.
>
> - 복제 시간 제어(RTC) - 예측 가능한 기간 내에 서로 다른 리전에서 데이터를 복제하려면 S3 Replication Time Control(S3 RTC)을 사용하면 됩니다. S3 RTC는 Amazon S3에 저장된 새 객체의 99.99%를 15분 이내에 복제합니다(서비스 수준 계약에 따라 지원됨). 자세한 내용은 the section called “S3 Replication Time Control 사용” 단원을 참조하십시오.
> - 복제 지표 및 알림 - Amazon CloudWatch 지표를 사용 설정하여 복제 이벤트를 모니터링합니다.
> - 삭제 마커 복제 - S3 삭제 작업으로 생성된 삭제 마커가 복제됩니다. 수명 주기 규칙에 의해 생성된 삭제 마커는 복제되지 않습니다. 자세한 내용은 버킷 간 삭제 마커 복제를 참조하세요.
>
> S3 RTC 및 CloudWatch 복제 지표와 알림에는 추가 요금이 부과됩니다. 자세한 내용은 Amazon S3 요금 및 Amazon CloudWatch 요금을 참조하세요.

11. 기존 복제 규칙을 대체하는 새 복제 규칙을 작성하는 경우 I acknowledge that by choosing Create replication rules, these existing replication rules will be overwritten(복제 규칙 생성을 선택하면 기존 복제 규칙을 덮어쓰게 됨을 이해합니다)을 선택합니다.
12. 복제 규칙 생성을 선택하여 새 단방향 복제 규칙을 생성하고 저장합니다.

다중 리전 액세스 포인트에 양방향 복제 규칙 생성

복제는 규칙을 사용하면 버킷 간에 객체를 비동기식으로 자동 복제할 수 있습니다. 양방향 복제 규칙을 사용하면 서로 다른 AWS 리전에 있는 두 개 이상의 버킷 간에 데이터를 완전히 동기화할 수 있습니다. 양방향 복제를 설정하면 원본 버킷(DOC-EXAMPLE-BUCKET-1)에서 복제본이 포함된 버킷(DOC-EXAMPLE-BUCKET-2)으로 복제 규칙이 생성됩니다. 그런 다음 복제본이 포함된 버킷(DOC-EXAMPLE-BUCKET-2)에서 원본 버킷(DOC-EXAMPLE-BUCKET-1)으로의 두 번째 복제 규칙이 생성됩니다.

다른 복제 규칙과 마찬가지로 양방향 복제 규칙을 전체 Amazon S3 버킷에 적용하거나 접두사 또는 객체 태그로 필터링된 객체의 하위 집합에 적용할 수 있습니다. 각 복제 규칙에 대해 복제본 수정 동기화를 사용 설정하여 객체의 메타데이터 변경 사항도 동기화된 상태로 유지할 수 있습니다. Amazon S3

콘솔, AWS CLI, AWS SDK, Amazon S3 REST API 또는 AWS CloudFormation를 통해 복제본 수정 동기화를 활성화할 수 있습니다.

Amazon CloudWatch에서 객체 및 객체 메타데이터의 복제 진행 상황을 모니터링하려면 S3 복제 지표 및 알림을 사용 설정하세요. 자세한 내용은 복제 지표 및 Amazon S3 이벤트 알림으로 진행 상태 모니터링을 참조하세요.

다중 리전 액세스 포인트에 양방향 복제 규칙을 생성하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 다중 리전 액세스 포인트)를 선택합니다.
3. 업데이트하려는 다중 리전 액세스 포인트의 이름을 선택합니다.
4. Replication and failover(복제 및 장애 조치) 탭을 선택합니다.
5. 복제 규칙까지 아래로 스크롤한 다음 복제 규칙 생성을 선택합니다.
6. 복제 규칙 생성 페이지에서 지정된 모든 버킷 간에 객체 복제 템플릿을 선택합니다. 지정된 모든 버킷 간에 객체 복제 템플릿은 버킷에 양방향 복제(장애 조치 기능 포함)를 설정합니다.

   > **Important**
   >
   > 이 템플릿을 사용하여 복제 규칙을 만들면 해당 규칙이 이미 버킷에 할당된 기존 복제 규칙을 대체합니다.
   > 기존 복제 규칙을 대체하는 대신 추가하거나 수정하려면 콘솔에서 각 버킷의 관리 탭으로 이동한 다음 복제 규칙 섹션에서 규칙을 편집하세요. AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용하여 기존 복제 규칙을 추가하거나 수정할 수도 있습니다. 자세한 내용은 복제 구성 파일 요소 단원을 참조하십시오.

7. 버킷 섹션에서 객체를 복제하려는 원본 버킷을 두 개 이상 선택합니다. 복제를 위해 선택한 모든 버킷에는 S3 버전 관리가 활성화되어 있어야 하며 각 버킷은 서로 다른 AWS 리전에 있어야 합니다. S3 버전 관리에 대한 자세한 내용은 Amazon S3 버킷에서 버전 관리 사용을 참조하세요.

   > **Note**
   >
   > 복제를 설정하는 데 필요한 읽기 및 복제 권한이 있는지 확인하세요. 권한이 없으면 오류가 발생합니다. 자세한 내용은 IAM 역할 생성을 참조하세요.

8. 복제 규칙 구성 섹션에서 복제 규칙을 생성할 때 해당 복제 규칙을 활성화할지 비활성화할지를 선택합니다.

> **Note**
>
> 복제 규칙 이름 상자에 이름을 입력할 수 없습니다. 복제 규칙 이름은 복제 규칙을 생성할 때 구성을 기반으로 생성됩니다.

9. 범위 섹션에서 복제에 적합한 범위를 선택합니다.

   - 전체 버킷을 복제하려면 Apply to all objects in the bucket(버킷의 모든 객체에 적용)를 선택합니다.
   - 버킷 내의 객체 중 일부를 복제하려면 Limit the scope of this rule using one or more filters(하나 이상의 필터를 사용하여 이 규칙의 범위 제한)을 선택합니다.

     접두사, 객체 태그 또는 이 두 가지를 모두 사용하여 객체를 필터링할 수 있습니다.

     - 동일한 문자열로 시작하는 이름(예: `pictures`)을 가진 모든 객체로 복제를 제한하려면 접두사 상자에 저두사를 입력합니다.

       어떤 폴더의 이름에 해당하는 접두사를 입력할 경우, /(슬래시)를 마지막 문자로 사용해야 합니다(예: `pictures/`).
     - 하나 이상의 객체 태그가 있는 모든 객체를 복제하려면 태그 추가를 선택하고 상자에 키 값 페어를 입력합니다. 다른 태그를 추가하려면 이 절차를 반복합니다. 객체 태그에 대한 자세한 내용은 태그를 사용하여 스토리지 분류 섹션을 참조하십시오.

10. 아래로 스크롤하여 추가 복제 옵션 섹션으로 이동한 다음 적용할 복제 옵션을 선택합니다.

> **Note**
>
> 특히 장애 조치를 지원하도록 다중 리전 액세스 포인트를 구성하려는 경우 다음 옵션을 적용하는 것이 좋습니다.
>
> - 복제 시간 제어(RTC) - 예측 가능한 기간 내에 서로 다른 리전에서 데이터를 복제하려면 S3 Replication Time Control(S3 RTC)을 사용하면 됩니다. S3 RTC는 Amazon S3에 저장된 새 객체의 99.99%를 15분 이내에 복제합니다(서비스 수준 계약에 따라 지원됨). 자세한 내용은 the section called “S3 Replication Time Control 사용” 단원을 참조하십시오.
> - 복제 지표 및 알림 - Amazon CloudWatch 지표를 사용 설정하여 복제 이벤트를 모니터링합니다.

- 삭제 마커 복제 - S3 삭제 작업으로 생성된 삭제 마커가 복제됩니다. 수명 주기 규칙에 의해 생성된 삭제 마커는 복제되지 않습니다. 자세한 내용은 버킷 간 삭제 마커 복제를 참조하세요.
- 복제본 수정 동기화 - 각 복제 규칙에 대해 복제본 수정 동기화를 사용 설정하여 객체의 메타데이터 변경 사항도 동기화된 상태로 유지합니다. 자세한 내용은 복제본 수정 동기화 사용 설정을 참조하세요.

S3 RTC 및 CloudWatch 복제 지표와 알림에는 추가 요금이 부과됩니다. 자세한 내용은 Amazon S3 요금 및 Amazon CloudWatch 요금을 참조하세요.

11. 기존 복제 규칙을 대체하는 새 복제 규칙을 작성하는 경우 I acknowledge that by choosing Create replication rules, these existing replication rules will be overwritten(복제 규칙 생성을 선택하면 기존 복제 규칙을 덮어쓰게 됨을 이해합니다)을 선택합니다.
12. 복제 규칙 생성을 선택하여 새 양방향 복제 규칙을 생성하고 저장합니다.

다중 리전 액세스 포인트에 대한 복제 규칙 보기

다중 리전 액세스 포인트를 사용하면 단방향 복제 규칙 또는 양방향 복제 규칙을 설정할 수 있습니다. 복제 규칙을 관리하는 방법에 대한 자세한 내용은 Amazon S3 콘솔을 사용하여 복제 규칙 관리를 참조하세요.

다중 리전 액세스 포인트에 대한 복제 규칙을 보는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 다중 리전 액세스 포인트)를 선택합니다.
3. 다중 리전 액세스 포인트의 이름을 선택합니다.
4. Replication and failover(복제 및 장애 조치) 탭을 선택합니다.
5. 복제 규칙 섹션까지 아래로 스크롤합니다. 이 섹션에는 다중 리전 액세스 포인트에 생성된 모든 복제 규칙이 나열됩니다.

> **Note**
>
> 다른 계정의 버킷을 이 다중 리전 액세스 포인트에 추가한 경우 해당 버킷의 복제 규칙을 보려면 버킷 소유자의 `s3:GetBucketReplication` 권한이 있어야 합니다.

지원되는 API 작업으로 다중 리전 액세스 포인트 사용

Amazon S3는 다중 리전 액세스 포인트를 관리하기 위한 일련의 작업을 제공합니다. Amazon S3는 이러한 작업 중 일부를 동기적으로 처리하고 일부는 비동기적으로 처리합니다. 비동기 작업을 호출하면 Amazon S3가 먼저 요청된 작업을 동기적으로 승인합니다. 권한이 부여되면 Amazon S3는 요청한 작업의 진행 상황 및 결과를 추적하는 데 사용할 수 있는 토큰을 반환합니다.

> **Note**
>
> Amazon S3 콘솔을 통한 요청은 항상 동기적입니다. 콘솔은 요청이 완료될 때까지 기다렸다가 다른 요청을 제출할 수 있도록 합니다.

콘솔을 사용하여 비동기 작업의 현재 상태 및 결과를 보거나 AWS CLI, AWS SDK 또는 REST API에서 `DescribeMultiRegionAccessPointOperation`을 사용할 수 있습니다. Amazon S3는 비동기 작업에 대한 응답으로 추적 토큰을 제공합니다. 해당 추적 토큰을 `DescribeMultiRegionAccessPointOperation`에 대한 인수로 포함합니다. 추적 토큰을 포함하는 경우 Amazon S3는 다음 단계로 오류나 관련 리소스 정보를 포함하여 지정된 작업의 현재 상태 및 결과를 반환합니다. Amazon S3는 `DescribeMultiRegionAccessPointOperation` 작업을 동기적으로 수행합니다.

다중 리전 액세스 포인트를 생성하거나 유지하기 위한 모든 모든 컨트롤 플레인 요청은 `US West (Oregon)` 리전으로 라우팅되어야 합니다. 다중 리전 액세스 포인트 데이터 영역 요청의 경우 리전을 지정할 필요가 없습니다. 다중 리전 액세스 포인트 장애 조치 컨트롤 플레인의 경우 지원되는 다섯 개 리전 중 하나로 요청을 라우팅해야 합니다. 다중 리전 액세스 포인트 지원 리전에 대한 자세한 내용은 다중 리전 액세스 포인트 규제 및 제한 섹션을 참조하세요.

이 밖에도 다중 리전 액세스 포인트 관리를 요청하는 사용자, 역할 또는 기타 AWS Identity and Access Management(IAM) 엔터티에 `s3:ListAllMyBuckets` 권한을 부여해야 합니다.

다음 예시에서는 Amazon S3에서 호환되는 작업으로 다중 리전 액세스 포인트를 사용하는 방법을 보여줍니다.

주제

- AWS 서비스 및 AWS SDK와 다중 리전 액세스 포인트의 호환성
- S3 작업과 다중 리전 액세스 포인트의 호환성
- 다중 리전 액세스 포인트 라우팅 구성 보기
- 기본 Amazon S3 버킷 정책을 업데이트합니다.

- 다중 리전 액세스 포인트 라우팅 구성 업데이트
- 다중 리전 액세스 포인트의 버킷에 객체 추가
- 다중 리전 액세스 포인트에서 객체 검색
- 다중 리전 액세스 포인트의 기반이 되는 버킷에 저장된 객체 목록
- 다중 리전 액세스 포인트와 함께 미리 서명된 URL 사용
- 다중 리전 액세스 포인트와 함께 요청자 지불로 구성된 버킷을 사용합니다.

AWS 서비스 및 AWS SDK와 다중 리전 액세스 포인트의 호환성

Amazon S3 버킷 이름이 필요한 애플리케이션과 함께 다중 리전 액세스 포인트를 사용하려면 AWS SDK를 사용하여 요청할 때 다중 리전 액세스 포인트의 Amazon 리소스 이름(ARN)을 사용합니다. 어떤 AWS SDK가 다중 리전 액세스 포인트와 호환되는지 확인하려면 AWS SDK와의 호환성을 참조하세요.

S3 작업과 다중 리전 액세스 포인트의 호환성

다음 Amazon S3 데이터 영역 API 작업을 사용하여 다중 리전 액세스 포인트와 연결된 버킷의 객체에 대한 작업을 수행할 수 있습니다. 다음 S3 작업에서는 다중 리전 액세스 포인트 ARN을 수락할 수 있습니다.

- AbortMultipartUpload
- CompleteMultipartUpload
- CreateMultipartUpload
- DeleteObject
- DeleteObjectTagging
- GetObject
- GetObjectAcl
- GetObjectLegalHold
- GetObjectRetention
- GetObjectTagging
- HeadObject
- ListMultipartUploads
- ListObjectsV2
- ListParts

- PutObject
- PutObjectAcl
- PutObjectLegalHold
- PutObjectRetention
- PutObjectTagging
- RestoreObject
- UploadPart

> **Note**
>
> 다중 리전 액세스 포인트는 다중 리전 액세스 포인트 ARN을 사용할 때 목적지로만 다중 리전 액세스 포인트를 사용한 복사 작업을 지원합니다.

다음과 같은 Amazon S3 컨트롤 플레인 작업을 사용하여 다중 리전 액세스 포인트를 생성하고 관리할 수 있습니다.

- CreateMultiRegionAccessPoint
- DescribeMultiRegionAccessPointOperation
- GetMultiRegionAccessPoint
- GetMultiRegionAccessPointPolicy
- GetMultiRegionAccessPointPolicyStatus
- GetMultiRegionAccessPointRoutes
- ListMultiRegionAccessPoints
- PutMultiRegionAccessPointPolicy
- SubmitMultiRegionAccessPointRoutes

다중 리전 액세스 포인트 라우팅 구성 보기

AWS CLI

다음 예시 명령은 다중 리전 액세스 포인트 라우팅 구성을 검색하여 버킷의 현재 라우팅 상태를 확인할 수 있도록 합니다. 이 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3control get-multi-region-access-point-routes
--region eu-west-1
--account-id 111122223333
--mrap arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap
```

SDK for Java

다음 SDK for Java 코드는 다중 리전 액세스 포인트 라우팅 구성을 검색하여 버킷의 현재 라우팅 상태를 확인할 수 있도록 합니다. 이 예시 구문을 사용하려면 *user input placeholders*를 사용자의 정보로 대체하세요.

```
S3ControlClient s3ControlClient = S3ControlClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(credentialsProvider)
    .build();

GetMultiRegionAccessPointRoutesRequest request =
 GetMultiRegionAccessPointRoutesRequest.builder()
    .accountId("111122223333")
    .mrap("arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap")
    .build();

GetMultiRegionAccessPointRoutesResponse response =
 s3ControlClient.getMultiRegionAccessPointRoutes(request);
```

SDK for JavaScript

다음 SDK for JavaScript 코드는 다중 리전 액세스 포인트 라우팅 구성을 검색하여 버킷의 현재 라우팅 상태를 확인할 수 있도록 합니다. 이 예시 구문을 사용하려면 *user input placeholders*를 사용자의 정보로 대체하세요.

```
const REGION = 'us-east-1'

const s3ControlClient = new S3ControlClient({
  region: REGION
})

export const run = async () => {
  try {
    const data = await s3ControlClient.send(
      new GetMultiRegionAccessPointRoutesCommand({
        AccountId: '111122223333',
```

```
        Mrap: 'arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap',
      })
    )
    console.log('Success', data)
    return data
  } catch (err) {
    console.log('Error', err)
  }
}

run()
```

SDK for Python

다음 SDK for Python 코드는 다중 리전 액세스 포인트 라우팅 구성을 검색하여 버킷의 현재 라우팅 상태를 확인할 수 있도록 합니다. 이 예시 구문을 사용하려면 *user input placeholders*를 사용자의 정보로 대체하세요.

```
s3.get_multi_region_access_point_routes(
        AccountId=111122223333,
        Mrap=arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap)['Routes']
```

기본 Amazon S3 버킷 정책을 업데이트합니다.

적절한 액세스 권한을 부여하려면 기본 Amazon S3 버킷 정책도 업데이트해야 합니다. 다음 예시는 다중 리전 액세스 포인트 정책에 액세스 제어를 위임합니다. 액세스 제어를 다중 리전 액세스 포인트 정책에 위임하고 나면 다중 리전 액세스 포인트를 통해 요청이 이루어진 경우 액세스 제어에 더 이상 버킷 정책을 사용할 수 없습니다.

다음은 액세스 제어를 다중 리전 액세스 포인트 정책에 위임하는 버킷 정책의 예시입니다. 이 버킷 정책 예시를 사용하려면 *user input placeholders*를 실제 정보로 대체하세요. AWS CLI `put-bucket-policy` 명령을 통해 이 정책을 적용하려면 다음 예와 같이 정책을 파일에 저장합니다(예: `policy.json`).

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Principal": { "AWS": "*" },
    "Effect": "Allow",
    "Action": ["s3:*"],
    "Resource": ["arn:aws:s3:::111122223333/*", "arn:aws:s3:::amzn-s3-demo-bucket"],
```

```
    "Condition": {
      "StringEquals": {
        "s3:DataAccessPointAccount": "444455556666"
      }
    }
  }
}
```

다음 put-bucket-policy 예시 명령은 업데이트된 S3 버킷 정책을 S3 버킷과 연결합니다.

```
aws s3api put-bucket-policy
  --bucket amzn-s3-demo-bucket
  --policy file:///tmp/policy.json
```

다중 리전 액세스 포인트 라우팅 구성 업데이트

다음 예시 명령은 다중 리전 액세스 포인트 라우팅 구성을 업데이트합니다. 다중 리전 액세스 포인트 라우팅 명령은 다음 5개 리전에 대해 실행할 수 있습니다.

- ap-southeast-2
- ap-northeast-1
- us-east-1
- us-west-2
- eu-west-1

다중 리전 액세스 포인트 라우팅 구성에서는 버킷을 액티브 또는 패시브 라우팅 상태로 설정할 수 있습니다. 액티브 버킷은 트래픽을 수신하지만 패시브 버킷은 그렇지 않습니다. 버킷의 TrafficDialPercentage 값을 액티브의 경우 100, 패시브의 경우 0으로 설정하여 버킷의 라우팅 상태를 설정할 수 있습니다.

AWS CLI

다음 예시 명령은 다중 리전 액세스 포인트 라우팅 구성을 업데이트합니다. 이 예시에서는 *amzn-s3-demo-bucket1*이 액티브 상태로, *amzn-s3-demo-bucket2*가 패시브로 설정되어 있습니다. 이 예시 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3control submit-multi-region-access-point-routes
--region ap-southeast-2
--account-id 111122223333
```

```
--mrap arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap
--route-updates Bucket=amzn-s3-demo-bucket1,TrafficDialPercentage=100
                Bucket=amzn-s3-demo-bucket2,TrafficDialPercentage=0
```

SDK for Java

다음 SDK for Java 코드는 다중 리전 액세스 포인트 라우팅 구성을 업데이트합니다. 이 예시 구문을 사용하려면 *user input placeholders*를 사용자의 정보로 대체하세요.

```
S3ControlClient s3ControlClient = S3ControlClient.builder()
    .region(Region.ap-southeast-2)
    .credentialsProvider(credentialsProvider)
    .build();

SubmitMultiRegionAccessPointRoutesRequest request =
 SubmitMultiRegionAccessPointRoutesRequest.builder()
    .accountId("111122223333")
    .mrap("arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap")
    .routeUpdates(
        MultiRegionAccessPointRoute.builder()
            .region("eu-west-1")
            .trafficDialPercentage(100)
            .build(),
        MultiRegionAccessPointRoute.builder()
            .region("ca-central-1")
            .bucket("111122223333")
            .trafficDialPercentage(0)
            .build()
    )
    .build();

SubmitMultiRegionAccessPointRoutesResponse response =
 s3ControlClient.submitMultiRegionAccessPointRoutes(request);
```

SDK for JavaScript

다음 SDK for JavaScript 코드는 다중 리전 액세스 포인트 라우팅 구성을 업데이트합니다. 이 예시 구문을 사용하려면 *user input placeholders*를 사용자의 정보로 대체하세요.

```
const REGION = 'ap-southeast-2'

const s3ControlClient = new S3ControlClient({
  region: REGION
```

```
})

export const run = async () => {
  try {
    const data = await s3ControlClient.send(
      new SubmitMultiRegionAccessPointRoutesCommand({
        AccountId: '111122223333',
        Mrap: 'arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap',
        RouteUpdates: [
          {
            Region: 'eu-west-1',
            TrafficDialPercentage: 100,
          },
          {
            Region: 'ca-central-1',
            Bucket: 'amzn-s3-demo-bucket1',
            TrafficDialPercentage: 0,
          },
        ],
      })
    )
    console.log('Success', data)
    return data
  } catch (err) {
    console.log('Error', err)
  }
}

run()
```

SDK for Python

다음 SDK for Python 코드는 다중 리전 액세스 포인트 라우팅 구성을 업데이트합니다. 이 예시 구문을 사용하려면 *user input placeholders*를 사용자의 정보로 대체하세요.

```
s3.submit_multi_region_access_point_routes(
        AccountId=111122223333,
        Mrap=arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap,
        RouteUpdates= [{
            'Bucket': amzn-s3-demo-bucket,
            'Region': ap-southeast-2,
            'TrafficDialPercentage': 10
        }])
```

다중 리전 액세스 포인트의 버킷에 객체 추가

다중 리전 액세스 포인트에 연결된 버킷에 객체를 추가하려면 PutObject 작업을 사용합니다. 다중 리전 액세스 포인트의 모든 버킷을 동기화된 상태로 유지하려면 크로스 리전 복제를 활성화합니다.

> **Note**
>
> 이 작업을 사용하려면 다중 리전 액세스 포인트에 대한 `s3:PutObject` 권한이 있어야 합니다. 다중 리전 액세스 포인트 권한 요구 사항에 대한 자세한 내용은 권한 섹션을 참조하세요.

AWS CLI

다음 예시 데이터 영역 요청은 지정된 다중 리전 액세스 포인트에 *example.txt*를 업로드합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws s3api put-object --bucket
 arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap --key example.txt --
body example.txt
```

SDK for Java

```
S3Client s3Client = S3Client.builder()
        .build();

PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket("arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap")
        .key("example.txt")
        .build();

s3Client.putObject(objectRequest, RequestBody.fromString("Hello S3!"));
```

SDK for JavaScript

```
const client = new S3Client({});

async function putObjectExample() {
    const command = new PutObjectCommand({
        Bucket: "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap",
        Key: "example.txt",
        Body: "Hello S3!",
```

```
    });

    try {
        const response = await client.send(command);
        console.log(response);
    } catch (err) {
        console.error(err);
    }
}
```

SDK for Python

```
import boto3

client = boto3.client('s3')
client.put_object(
    Bucket='arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap',
    Key='example.txt',
    Body='Hello S3!'
)
```

다중 리전 액세스 포인트에서 객체 검색

다중 리전 액세스 포인트에서 객체를 검색하려면 GetObject 작업을 사용할 수 있습니다.

> **Note**
>
> 이 API 작업을 사용하려면 다중 리전 액세스 포인트에 대한 `s3:GetObject` 권한이 있어야 합니다. 다중 리전 액세스 포인트 권한 요구 사항에 대한 자세한 내용은 권한 섹션을 참조하세요.

AWS CLI

다음 예시 데이터 영역 요청은 지정된 다중 리전 액세스 포인트에서 *example.txt*를 검색하여 *downloaded_example.txt*로 다운로드합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws s3api get-object --bucket
 arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap --
key example.txt downloaded_example.txt
```

SDK for Java

```
S3Client s3 = S3Client
   .builder()
   .build();

GetObjectRequest getObjectRequest = GetObjectRequest.builder()
    .bucket("arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap")
    .key("example.txt")
    .build();

s3Client.getObject(getObjectRequest);
```

SDK for JavaScript

```
const client = new S3Client({})

async function getObjectExample() {
    const command = new GetObjectCommand({
        Bucket: "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap",
        Key: "example.txt"
    });

    try {
        const response = await client.send(command);
        console.log(response);
    } catch (err) {
        console.error(err);
    }
}
```

SDK for Python

```
import boto3

client = boto3.client('s3')
client.get_object(
    Bucket='arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap',
    Key='example.txt'
)
```

다중 리전 액세스 포인트의 기반이 되는 버킷에 저장된 객체 목록

다중 리전 액세스 포인트의 기반이 되는 버킷에 저장된 객체 목록을 반환하려면 ListObjectsV2 작업을 사용합니다. 다음 예시 명령에서는 다중 리전 액세스 포인트에 대한 ARN을 사용하여 지정된 다중 리전 액세스 포인트에 대한 모든 객체가 나열됩니다. 이 경우 다중 리전 액세스 포인트 ARN은 다음과 같습니다.

`arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap`

> **Note**
>
> 이 API 작업을 사용하려면 다중 리전 액세스 포인트 및 기반 버킷에 대한 `s3:ListBucket` 권한이 있어야 합니다. 다중 리전 액세스 포인트 권한 요구 사항에 대한 자세한 내용은 권한 섹션을 참조하세요.

AWS CLI

다음 예시 데이터 영역 요청은 ARN에서 지정한 다중 리전 액세스 포인트의 기반이 되는 버킷의 객체를 나열합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws s3api list-objects-v2 --bucket
 arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap
```

SDK for Java

```
S3Client s3Client = S3Client.builder()
        .build();

String bucketName = "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap";

ListObjectsV2Request listObjectsRequest = ListObjectsV2Request
    .builder()
    .bucket(bucketName)
    .build();

 s3Client.listObjectsV2(listObjectsRequest);
```

SDK for JavaScript

```
const client = new S3Client({});

async function listObjectsExample() {
    const command = new ListObjectsV2Command({
        Bucket: "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap",
    });

    try {
        const response = await client.send(command);
        console.log(response);
    } catch (err) {
        console.error(err);
    }
}
```

SDK for Python

```
import boto3

client = boto3.client('s3')
client.list_objects_v2(
    Bucket='arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap'
)
```

다중 리전 액세스 포인트와 함께 미리 서명된 URL 사용

미리 서명된 URL을 사용하여 다른 사람이 Amazon S3 다중 리전 액세스 포인트를 통해 Amazon S3 버킷에 액세스하는 데 사용할 수 있는 URL을 생성할 수 있습니다. 미리 서명된 URL을 생성하면 S3 업로드(`PutObject`) 또는 S3 다운로드(`GetObject`)와 같은 특정 객체 작업과 연결됩니다. 미리 서명된 URL을 공유할 수 있으며 URL에 액세스할 수 있는 모든 사용자는 원래 서명 사용자인 것처럼 URL에 포함된 작업을 수행할 수 있습니다.

미리 서명된 URL에는 만료일이 있습니다. 만료 시간에 도달하면 URL이 더 이상 작동하지 않습니다.

미리 서명된 URL과 함께 S3 다중 리전 액세스 포인트를 사용하기 전에 SigV4A 알고리즘과의 AWS SDK 호환성을 확인하세요. 글로벌 AWS 리전 요청에 서명하는 데 사용되는 서명 구현으로 SigV4A 알고리즘을 지원하는 SDK 버전인지 확인하세요. Amazon S3와 함께 미리 서명된 URL을 사용하는 방법에 대한 자세한 내용은 미리 서명된 URL을 사용하여 객체 공유 섹션을 참조하세요.

다음 예시에서는 미리 서명된 URL과 함께 다중 리전 액세스 포인트를 사용하는 방법을 보여줍니다. 이러한 예시를 사용하려면 *user input placeholders*를 실제 정보로 대체하십시오.

AWS CLI

```
aws s3 presign
 arn:aws:s3::123456789012:accesspoint/MultiRegionAccessPoint_alias/example-file.txt
```

SDK for Python

```
import logging
import boto3
from botocore.exceptions import ClientError

s3_client = boto3.client('s3',aws_access_key_id='xxx',aws_secret_access_key='xxx')
s3_client.generate_presigned_url(HttpMethod='PUT',ClientMethod="put_object",
 Params={'Bucket':'arn:aws:s3::123456789012:accesspoint/
abcdef0123456.mrap','Key':'example-file'})
```

SDK for Java

```
S3Presigner s3Presigner = S3Presigner.builder()
    .credentialsProvider(StsAssumeRoleCredentialsProvider.builder()
        .refreshRequest(assumeRole)
        .stsClient(stsClient)
        .build())
    .build();

GetObjectRequest getObjectRequest = GetObjectRequest.builder()
    .bucket("arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap")
    .key("example-file")
    .build();

GetObjectPresignRequest preSignedReq = GetObjectPresignRequest.builder()
    .getObjectRequest(getObjectRequest)
    .signatureDuration(Duration.ofMinutes(10))
    .build();

PresignedGetObjectRequest presignedGetObjectRequest =
 s3Presigner.presignGetObject(preSignedReq);
```

> **Note**
>
> IAM 역할을 사용하는 경우와 같이 임시 보안 인증 정보와 함께 SigV4A를 사용하려면 글로벌 엔드포인트가 아닌 AWS Security Token Service(AWS STS)의 리전 엔드포인트에서 임시 보안 인증 정보를 요청해야 합니다. AWS STS(`sts.amazonaws.com`)에 대한 글로벌 엔드포인트를 사용하면 Sig4A에서 지원하지 않는 글로벌 엔드포인트에서 임시 보안 인증 정보가 생성됩니다. 따라서 오류가 발생합니다. 이 문제를 해결하려면 AWS STS 리전 엔드포인트에 나열된 것 중 하나를 사용하세요.

다중 리전 액세스 포인트와 함께 요청자 지불로 구성된 버킷을 사용합니다.

다중 리전 액세스 포인트와 연결된 S3 버킷이 요청자 지불을 사용하도록 구성된 경우 요청자는 버킷 요청과 다운로드, 다중 리전 액세스 포인트 관련 비용을 모두 지불합니다. 자세한 내용은 Amazon S3 요금을 참조하십시오.

다음은 요청자 지불 버킷에 연결된 다중 리전 액세스 포인트에 대한 데이터 영역 요청의 예입니다.

AWS CLI

요청자 지불 버킷에 연결된 다중 리전 액세스 포인트에서 객체를 다운로드하려면 get-object 요청의 일부로 `--request-payer requester`를 지정해야 합니다. 또한 버킷에 있는 파일의 이름과 다운로드되는 파일을 저장할 위치를 지정해야 합니다.

```
aws s3api get-object --bucket MultiRegionAccessPoint_ARN --request-payer requester
 --key example-file-in-bucket.txt example-location-of-downloaded-file.txt
```

SDK for Java

요청자 지불 버킷에 연결된 다중 리전 액세스 포인트에서 객체를 다운로드하려면 `GetObject` 요청의 일부로 `RequestPayer.REQUESTER`를 지정해야 합니다. 또한 버킷에 있는 파일의 이름과 파일을 저장할 위치를 지정해야 합니다.

```
GetObjectResponse getObjectResponse = s3Client.getObject(GetObjectRequest.builder()
    .key("example-file.txt")
    .bucket("arn:aws:s3::
123456789012:accesspoint/abcdef0123456.mrap")
    .requestPayer(RequestPayer.REQUESTER)
    .build()
).response();
```

다중 리전 액세스 포인트를 통해 기본 리소스에 대한 요청 모니터링 및 로깅

Amazon S3는 다중 리전 액세스 포인트를 통해 수행된 요청 및 이를 관리하는 API 작업에 대한 요청(예: `CreateMultiRegionAccessPoint` 및 `GetMultiRegionAccessPointPolicy`)을 로깅합니다. 다중 리전 액세스 포인트를 통해 수행된 Amazon S3에 대한 요청은 다중 리전 액세스 포인트의 호스트 이름과 함께 Amazon S3 서버 액세스 로그 및 AWS CloudTrail 로그에 표시됩니다. 액세스 포인트의 호스트 이름은 *`MRAP_alias`*`.accesspoint.s3-global.amazonaws.com` 형식을 사용합니다. 예를 들어 다음과 같은 버킷 및 다중 리전 액세스 포인트 구성이 있다고 가정합니다.

- 이름이 `my-bucket-usw2`이며 `us-west-2` 리전에 있고 `my-image.jpg` 객체가 포함된 버킷
- 이름이 `my-bucket-aps1`이며 `ap-south-1` 리전에 있고 `my-image.jpg` 객체가 포함된 버킷
- 이름이 `my-bucket-euc1`이며 `eu-central-1` 리전에 있고 `my-image.jpg` 객체가 포함되지 않은 버킷
- `mfzwi23gnjvgw.mrap` 별칭이 있는 `my-mrap` 다중 영역 액세스 포인트가 버킷 3개의 요청을 모두 이행하도록 구성되어 있습니다.
- 귀하의 AWS 계정 ID는 `123456789012`입니다.

버킷을 통해 `my-image.jpg`를 직접 검색하기 위해 이루어진 요청은 호스트 이름이 *`bucket_name`*`.s3.`*`Region`*`.amazonaws.com`인 로그에 표시됩니다.

다중 리전 액세스 포인트를 통해 요청하는 경우 Amazon S3는 먼저 서로 다른 리전의 버킷 중 가장 가까운 버킷을 판단합니다. Amazon S3가 요청을 이행하는 데 사용할 버킷을 결정하면 해당 버킷으로 요청을 보내고 다중 리전 액세스 포인트 호스트 이름을 사용하여 작업을 로깅합니다. 이 예에서 Amazon S3가 요청을 `my-bucket-aps1`로 릴레이하면 로그는 `mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com`의 호스트 이름을 사용하여 `my-bucket-aps1`에서 `my-image.jpg`에 대해 성공한 `GET` 요청을 반영합니다.

> **Important**
>
> 다중 리전 액세스 포인트는 기본 버킷의 데이터 콘텐츠를 인식하지 못합니다. 따라서 요청을 받는 버킷에는 요청된 데이터가 포함되어 있지 않을 수 있습니다. 예를 들어, Amazon S3가 `my-bucket-euc1` 버킷이 가장 가깝다고 판단한 경우 로그에는 `mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com`의 호스트 이름을 사용하여 `my-bucket-euc1`에서 `my-image.jpg`에 대해 실패한 `GET` 요청이 반영됩니다. 요청이 대신 `my-bucket-usw2`로 라우팅된 경우 로그에는 성공한 `GET` 요청이 표시됩니다.

Amazon S3 서버 액세스 로그에 대한 자세한 내용은 서버 액세스 로깅을 사용한 요청 로깅 섹션을 참조하세요. AWS CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 AWS CloudTrail이란 무엇인가요?를 참조하세요.

다중 리전 액세스 포인트 관리 API 작업에 대한 요청 모니터링 및 로깅

Amazon S3는 `CreateMultiRegionAccessPoint` 및 `GetMultiRegionAccessPointPolicy`와 같은 다중 리전 액세스 포인트를 관리하기 위한 여러 가지 API 작업을 제공합니다. AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하여 이러한 API 작업에 대해 요청을 수행하면 Amazon S3는 이러한 요청을 비동기적으로 처리합니다. 해당 요청에 대한 적절한 권한이 있는 경우 Amazon S3는 이러한 요청에 대한 토큰을 반환합니다. 이 토큰을 `DescribeAsyncOperation`과 같이 사용하면 진행 중인 비동기 작업의 상태를 확인할 수 있습니다. Amazon S3는 `DescribeAsyncOperation` 요청을 동기적으로 처리합니다. 비동기 요청의 상태를 확인하려면 Amazon S3 콘솔, AWS CLI, SDK 또는 REST API를 사용합니다.

> **Note**
>
> 콘솔에는 이전 14일 이내에 수행된 비동기 요청의 상태만 표시됩니다. AWS CLI, SDK 또는 REST API를 사용하여 기존의 요청 상태 확인

비동기 관리 작업은 다음 상태 중 하나일 수 있습니다.

NEW

Amazon S3가 요청을 받았으며 작업을 수행할 준비를 하고 있습니다.

IN_PROGRESS

Amazon S3가 현재 작업을 수행하고 있습니다.

SUCCESS

작업이 성공했습니다. 응답에는 `CreateMultiRegionAccessPoint` 요청에 대한 다중 리전 액세스 포인트 별칭과 같은 관련 정보가 포함됩니다.

FAILED

작업이 실패했습니다. 응답에는 요청 실패의 원인을 나타내는 오류 메시지가 포함됩니다.

AWS CloudTrail 및 다중 리전 액세스 포인트 사용

AWS CloudTrail을 사용하여 AWS 인프라 전반에서 계정 활동을 확인, 검색, 다운로드, 보관 및 응답할 수 있습니다. 다중 리전 액세스 포인트 및 CloudTrail 로깅을 통해 다음을 식별할 수 있습니다.

- 누가 또는 무엇이 어떤 조치를 취했는지
- 어떤 리소스가 사용되었는지
- 이벤트가 언제 발생했는지
- 이벤트에 관한 기타 세부 정보

이 로깅 정보를 사용하여 다중 리전 액세스 포인트를 통해 발생한 활동을 분석하고 이에 대응할 수 있습니다.

다중 리전 액세스 포인트에 AWS CloudTrail을 설정하는 방법

다중 리전 액세스 포인트 생성 또는 유지 관리와 관련된 모든 작업에 대해 CloudTrail 로깅을 활성화하려면 CloudTrail 로깅을 구성하여 미국 서부(오레곤) 리전의 이벤트를 기록해야 합니다. 요청을 할 때 사용자가 있는 리전 또는 다중 리전 액세스 포인트가 지원하는 리전과 관계없이 이 방법으로 로깅 구성을 설정해야 합니다. 다중 리전 액세스 포인트를 생성하거나 유지하기 위한 모든 요청은 미국 서부(오레곤) 리전을 통해 라우팅됩니다. 이 리전을 기존 추적에 추가하거나 이 리전 및 다중 리전 액세스 포인트와 연관된 모든 리전을 포함하는 새 추적을 만드는 것이 좋습니다.

Amazon S3는 다중 리전 액세스 포인트를 통해 수행된 모든 요청 및 액세스 포인트를 관리하는 API 작업에 대한 요청(예: `CreateMultiRegionAccessPoint` 및 `GetMultiRegionAccessPointPolicy`)을 로깅합니다. 다중 리전 액세스 포인트를 통해 이러한 요청을 기록하면 다중 리전 액세스 포인트의 호스트 이름으로 AWS CloudTrail 로그에 표시됩니다. 예를 들어 `mfzwi23gnjvgw.mrap` 별칭을 사용하여 다중 리전 액세스 포인트를 통해 버킷에 요청하는 경우 CloudTrail 로그 항목의 호스트 이름은 `mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com`입니다.

다중 리전 액세스 포인트는 가장 가까운 버킷에 요청을 라우팅합니다. 이 동작으로 인해 다중 리전 액세스 포인트에 대한 CloudTrail 로그를 보면 기본 버킷에 대해 요청이 이루어지는 것을 볼 수 있습니다. 이러한 요청 중 일부는 다중 리전 액세스 포인트를 통해 라우팅되지 않는 버킷에 대한 직접 요청일 수 있습니다. 트래픽을 검토할 때는 이 사실에 유의하시기 바랍니다. 버킷이 다중 리전 액세스 포인트에 있는 경우 다중 리전 액세스 포인트를 통하지 않고도 해당 버킷에 대한 요청을 직접 수행할 수 있습니다.

다중 리전 액세스 포인트 생성 및 관리와 관련된 비동기 이벤트가 있습니다. 비동기 요청의 경우 CloudTrail 로그에는 완료 이벤트가 없습니다. 비동기 요청에 대한 자세한 내용은 다중 리전 액세스 포인트 관리 API 작업에 대한 요청 모니터링 및 로깅 섹션을 참조하세요.

AWS CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 AWS CloudTrail이란 무엇인가요?를 참조하세요.

S3 버전 관리로 여러 버전의 객체 유지

Amazon S3의 버전 관리는 동일 버킷 내에 여러 개의 객체 변형을 보유하는 수단입니다. S3 버전 관리를 사용하면 버킷에 저장된 모든 버전의 객체를 모두 보존, 검색 및 복원할 수 있습니다. 또한 의도치 않은 사용자 작업 및 애플리케이션 장애로부터 더 쉽게 복구할 수 있습니다. 버킷에 대해 버전 관리를 사용 설정하면 Amazon S3가 동일한 객체에 대해 여러 쓰기 요청을 동시에 수신하는 경우 모든 객체가 저장됩니다.

버전 관리가 사용 설정된 버킷의 경우 실수로 삭제되거나 덮어쓴 객체를 복구할 수 있습니다. 예를 들어 객체를 삭제할 경우 Amazon S3는 객체를 영구적으로 제거하는 대신 삭제 마커를 삽입합니다. 이 삭제 마커가 현재 객체 버전이 됩니다. 객체를 덮어쓴 경우 버킷에 새 객체 버전이 생깁니다. 따라서 언제든지 이전 버전을 복원할 수 있습니다. 자세한 내용은 버전 관리가 사용 설정된 버킷에서 객체 버전 삭제 섹션을 참조하세요.

기본적으로 버킷에는 S3 버전 관리가 사용 중지되어 있으므로 명시적으로 사용 설정해야 합니다. 자세한 내용은 버킷에 버전 관리 사용 설정 섹션을 참조하세요.

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

버전 관리 미사용 버킷, 버전 관리가 사용 설정된 버킷 및 버전 관리가 일시 중지된 버킷

버킷은 다음 3가지 상태 중 하나일 수 있습니다.

- 버전 관리 미사용(기본값)
- 버전 관리 사용
- 버전 관리가 일시 중지됨

버전 관리는 버킷 수준에서 사용 설정하고 일시 중지합니다. 버킷의 버전 관리를 사용 설정한 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. 버킷에서 버전 관리를 사용 설정하면 모든 새 객체의 버전이 관리되고 고유한 버전 ID가 부여됩니다. 버전 관리가 사용 설정되었을 때 버킷에 이미 존재했던 객체는 향후 요청에 의해 수정될 때 항상 버전이 관리되고 고유한 버전 ID가 부여됩니다. 다음을 참조하세요.

- 버전 관리 상태로 설정하기 전에 버킷에 저장된 객체의 버전 ID는 `null`이 됩니다. 버전 관리를 사용 설정할 때 버킷의 기존 객체는 변경되지 않습니다. 이후 요청에서는 Amazon S3가 객체를 처리하는 방법만 변경됩니다. 자세한 내용은 버전 관리가 사용 설정된 버킷의 객체 작업 섹션을 참조하세요.
- 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 요청에서는 Amazon S3가 객체를 처리하는 방법만 변경됩니다. 자세한 내용은 버전 관리가 일시 중지된 버킷의 객체 작업 섹션을 참조하세요.

S3 수명 주기와 함께 S3 버전 관리 사용

데이터 보존 접근 방식을 사용자 지정하고 스토리지 비용을 관리하려면 S3 수명 주기와 함께 객체 버전 관리를 사용합니다. 자세한 내용은 객체 수명 주기 관리 섹션을 참조하세요. AWS Management Console, AWS CLI, AWS SDK 또는 REST API를 사용하여 S3 수명 주기 구성을 생성하는 방법에 대한 자세한 내용은 버킷에서 S3 수명 주기 구성 설정 섹션을 참조하세요.

> **Important**
>
> 버전이 지정되지 않은 버킷에 객체 만료 수명 주기 구성이 있고 버전 관리를 사용할 때 같은 영구 삭제 동작을 유지하고자 하는 경우에는 최신 버전이 아닌 만료 정책을 추가해야 합니다. 최신이 아닌 버전의 만료 수명 주기 구성은 버전 관리가 활성화된 버킷에서 최신이 아닌 객체 버전의 삭제를 관리합니다. 버전 관리가 활성화된 버킷은 하나의 현재 객체 버전과 0개 이상의 최신이 아닌 객체 버전을 유지합니다. 자세한 내용은 버킷에서 S3 수명 주기 구성 설정 섹션을 참조하세요.

S3 버전 관리 작업에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- S3 버전 관리 작동 방식
- 버킷에 버전 관리 사용 설정
- MFA Delete 구성
- 버전 관리가 사용 설정된 버킷의 객체 작업
- 버전 관리가 일시 중지된 버킷의 객체 작업
- 버전 관리 문제 해결

S3 버전 관리 작동 방식

S3 버전 관리를 사용하여 한 버킷에 여러 버전의 객체를 보관함으로써 실수로 삭제되거나 덮어써진 객체를 복원할 수 있습니다. 예를 들어, 버킷에 S3 버전 관리를 적용하는 경우 다음과 같은 변경 사항이 발생합니다.

- 객체를 영구적으로 제거하는 대신 삭제하는 경우, Amazon S3에서는 삭제 마커를 삽입하며 이는 현재 객체 버전이 됩니다. 따라서 나중에 이전 버전을 복원할 수 있습니다. 자세한 내용은 버전 관리가 사용 설정된 버킷에서 객체 버전 삭제 섹션을 참조하세요.
- 객체를 덮어쓴 경우 Amazon S3가 버킷에 새 객체 버전을 추가합니다. 이전 버전은 버킷에 남아 있고 현재 버전이 아닌 버전이 됩니다. 이전 버전을 복원할 수 있습니다.

> **Note**
>
> 저장되거나 전송된 객체의 모든 버전에는 일반 Amazon S3 요금이 적용됩니다. 각 객체 버전은 이전 버전과의 차이점만 포함된 것이 아닌 완전한 객체입니다. 따라서 세 가지 버전의 객체가 저장된 경우 객체 세 개에 대한 요금이 부과됩니다.

생성한 각 S3 버킷에는 버전 관리 하위 리소스가 연결되어 있습니다. (자세한 내용은 범용 버킷 구성 옵션 섹션을 참조하세요.) 기본적으로 버킷은 버전 관리 미사용 상태이며, 버전 관리 하위 리소스는 다음과 같이 빈 버전 관리 구성을 저장합니다.

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

버전 관리를 활성화하려면 Amazon S3에 Enabled 상태가 포함된 버전 관리 구성을 사용하여 요청을 보내면 됩니다.

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>Enabled</Status>
</VersioningConfiguration>
```

버전 관리를 일시 중지하려면 상태 값을 Suspended로 설정합니다.

> **Note**
>
> 버킷에서 버전 관리를 처음으로 활성화할 때 변경 사항이 완전히 전파되는 데 시간이 조금 걸릴 수 있습니다. 이 변경 사항이 전파되는 동안 버전 관리를 활성화한 후 생성되거나 업데이트된 객체에 대한 요청에 간헐적인 HTTP 404 NoSuchKey 오류가 발생할 수 있습니다. 버전 관리를 활성화하고 나서 15분 정도 기다린 후, 버킷의 객체에 대해 쓰기 작업(PUT 또는 DELETE)을 실행하는 것이 좋습니다.

버킷 소유자 및 모든 승인된 AWS Identity and Access Management(IAM) 사용자는 버전 관리를 활성화할 수 있습니다. 버킷 소유자는 버킷을 생성한 AWS 계정입니다. 권한에 대한 자세한 내용은 Amazon S3의 ID 및 액세스 관리을 참조하세요.

AWS Management Console, AWS Command Line Interface(AWS CLI) 또는 REST API를 사용하여 S3 버전 관리를 사용하고 사용 중지하는 방법에 대한 자세한 내용은 the section called “버킷에 버전 관리 사용 설정” 섹션을 참조하세요.

주제

- 버전 ID
- 버전 관리 워크플로

버전 ID

버킷에 대한 버전 관리를 사용 설정하면 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 키(객체 이름)는 동일하지만 버전 ID가 다른 두 개의 객체를 보유할 수 있습니다.
버전 관리가 활성화되어 있으며 키는 같지만 버전 ID가 다른 객체 두 개가 있는 버킷을 보여 주는 다이어그램.

각 객체에는 S3 버전 관리가 사용되는지 여부에 관계없이 버전 ID가 있습니다. S3 버전 관리를 활성화하지 않으면 Amazon S3가 버전 ID의 값을 null로 설정합니다. S3 버전 관리를 사용 설정한 경우 Amazon S3는 객체에 대한 버전 ID 값을 할당합니다. 이 값은 해당 객체를 동일한 키의 다른 버전과 구별해 줍니다.

기존 버킷에 S3 버전 관리를 사용 설정하는 경우 버킷에 이미 저장된 객체는 변경되지 않습니다. 버전 ID(null), 콘텐츠, 그리고 권한은 동일하게 유지됩니다. 버전 관리를 활성화하면 버킷에 추가된 각 객체가 동일한 키의 다른 버전과 구별되는 버전 ID를 가져옵니다.

Amazon S3는 버전 ID만 생성하며 버전 ID를 편집할 수 없습니다. 버전 ID는 유니코드, UTF-8 인코딩, URL 지원, 불투명 문자열이며 길이가 1,024바이트를 넘지 않습니다. 다음은 그 한 예입니다.

```
3sL4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo
```

> **Note**
>
> 간단한 설명을 위해 이 주제의 다른 예제에서는 훨씬 더 짧은 ID를 사용합니다.

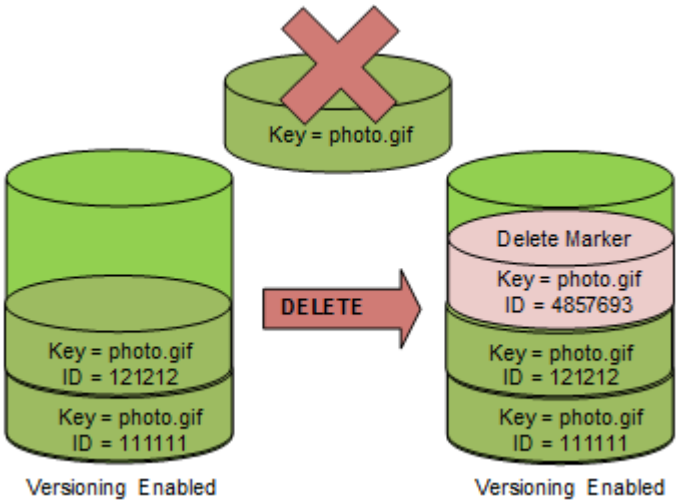
버전 관리 워크플로

버전 관리를 사용하는 버킷에 객체를 PUT할 때 비 최신 버전은 덮어쓰지 않습니다. 다음 그림과 같이 동일한 이름의 객체를 이미 보유하고 있는 버킷에 새 버전의 photo.gif를 PUT할 때 다음과 같은 동작이 발생합니다.

- 원래 객체(ID = 111111)는 버킷에 남아 있습니다.
- Amazon S3는 새 버전 ID(121212)를 생성하고 이 최신 버전의 객체를 버킷에 추가합니다.

이 기능을 사용하면 객체를 실수로 덮어쓰거나 삭제한 경우 객체의 이전 버전을 검색할 수 있습니다.

객체를 DELETE하면 다음 그림과 같이 버킷에 모든 버전이 계속 유지되며 Amazon S3는 삭제 마커를 삽입합니다.



삭제 마커가 객체의 최신 버전이 됩니다. 기본적으로 GET 요청은 가장 최근에 저장된 버전을 검색합니다. 최신 버전이 삭제 마커일 때 GET Object 요청을 수행하면 다음 그림과 같이 404 Not Found 오류가 반환됩니다.

하지만 버전 ID를 지정하여 최신 버전이 아닌 객체에 대해 GET을 실행할 수 있습니다. 다음 그림에서는 특정 객체 버전인 111111을 GET합니다. Amazon S3는 최신 버전이 아닌 해당 버전의 객체를 반환합니다.

자세한 내용은 버전 관리가 사용 설정된 버킷에서 객체 버전 검색 섹션을 참조하세요.

삭제하고자 하는 버전을 지정하여 객체를 영구적으로 삭제할 수 있습니다. Amazon S3 버킷 소유자 또는 권한이 부여된 IAM 사용자만이 영구적으로 버전을 삭제할 수 있습니다. DELETE 작업이 versionId를 지정하는 경우 해당 객체 버전은 영구 삭제되며 Amazon S3가 삭제 마커를 삽입하지 않습니다.

다중 인증(MFA) 삭제를 활성화하도록 버킷을 구성하여 보안을 강화할 수 있습니다. 버킷에 MFA 삭제를 활성화하는 경우 버킷 소유자는 버전을 삭제하거나 버킷의 버전 관리 상태를 변경하는 모든 요청에 두 가지 형식의 인증을 포함해야 합니다. 자세한 내용은 MFA Delete 구성 섹션을 참조하세요.

객체의 새 버전은 언제 생성됩니까?

객체의 새 버전은 새 객체에 PUT 작업을 수행할 때만 생성됩니다. CopyObject와 같은 특정 작업은 PUT 작업을 구현하여 작동한다는 점에 유의하세요.

현재 객체를 수정하는 일부 작업에서는 새 객체에 PUT 작업을 수행하지 않으므로 새 버전이 생성되지 않습니다. 여기에는 객체의 태그 변경과 같은 작업이 포함됩니다.

> **Important**
>
> S3 버전 관리를 사용하는 버킷에 대한 Amazon S3 객체 PUT 또는 DELETE 요청에서 받은 HTTP 503(서비스 사용 불가) 응답의 횟수가 크게 증가한다면 버전이 무수히 많은 객체가 하나 이상 버킷에 있을 것으로 추정됩니다. 자세한 내용은 버전 관리 문제 해결의 S3 버전 관리 섹션을 참조하세요.

버킷에 버전 관리 사용 설정

S3 버전 관리를 사용하면 한 버킷 내에 여러 개의 객체 버전을 유지할 수 있습니다. 이 섹션에서는 콘솔, REST API, AWS SDK 및 AWS Command Line Interface(AWS CLI)를 사용하여 버킷의 버전 관리를 사용하는 방법에 대한 예를 제공합니다.

> **Note**
>
> 버킷에서 버전 관리를 처음으로 활성화하면 변경 사항이 S3 시스템 전체에 완전히 전파되는 데 최대 15분이 걸릴 수 있습니다. 이 기간 동안 버전 관리를 활성화한 후 생성되거나 업데이트된 객체에 대한 GET 요청으로 인해 HTTP 404 NoSuchKey 오류가 발생할 수 있습니다. 버전 관리를 활성화하고 나서 15분 정도 기다린 후, 버킷의 객체에 대해 쓰기 작업(PUT 또는 DELETE)을 수행하는 것이 좋습니다. 이 대기 기간은 객체 가시성 및 버전 추적과 관련된 잠재적 문제를 방지하는 데 도움이 됩니다.

S3 버전 관리에 대한 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하세요. 버전 관리가 사용 설정된 버킷의 객체 작업에 대한 자세한 내용은 버전 관리가 사용 설정된 버킷의 객체 작업 섹션을 참조하세요.

S3 버전 관리를 사용하여 데이터를 보호하는 방법에 대한 자세한 내용은 Tutorial: Protecting data on Amazon S3 against accidental deletion or application bugs using S3 Versioning, S3 Object Lock, and S3 Replication(자습서: S3 버전 관리, S3 객체 잠금, S3 복제를 사용하여 우발적인 삭제나 애플리케이션 버그로부터 Amazon S3의 데이터 보호)을 참조하세요.

생성한 각 S3 버킷에는 버전 관리 하위 리소스가 연결되어 있습니다. (자세한 내용은 범용 버킷 구성 옵션 섹션을 참조하세요.) 기본적으로 버킷은 버전 관리 미사용 상태이며, 버전 관리 하위 리소스는 다음과 같이 빈 버전 관리 구성을 저장합니다.

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

```
</VersioningConfiguration>
```

버전 관리를 사용 설정하려면 Amazon S3에 상태가 포함된 버전 관리 구성을 사용하여 요청을 보내면 됩니다.

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>Enabled</Status>
</VersioningConfiguration>
```

버전 관리를 일시 중지하려면 상태 값을 Suspended로 설정합니다.

버킷 소유자 및 모든 승인된 사용자는 버전 관리를 사용 설정할 수 있습니다. 버킷 소유자는 버킷을 생성한 AWS 계정(루트 계정)입니다. 권한에 대한 자세한 내용은 Amazon S3의 ID 및 액세스 관리 섹션을 참조하세요.

다음 섹션에서는 콘솔, AWS CLI 및 AWS SDK를 사용하여 S3 버전 관리를 사용하는 방법에 대해 자세히 설명합니다.

S3 콘솔 사용

다음 단계에 따라 AWS Management Console을 사용하여 S3 버킷의 버전 관리를 사용합니다.

S3 범용 버킷의 버전 관리 사용 설정 또는 사용 중지

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 버전 관리를 사용 설정하려는 버킷의 이름을 선택합니다.
4. 속성을 선택합니다.
5. 버킷 버전 관리(Bucket Versioning)에서 편집을 선택합니다.
6. 일시 중지 또는 사용 설정를 선택한 다음 변경 사항 저장을 선택합니다.

> **Note**
>
> 버전 관리에 AWS 멀티 팩터 인증(MFA)을 사용할 수 있습니다. 버전 관리에 MFA를 사용하는 경우 객체 버전을 영구적으로 삭제하거나 버전 관리를 일시 중지 또는 다시 사용하려면 계정의 MFA 디바이스에서 유효한 코드와 AWS 계정의 액세스 키를 제공해야 합니다.

버전 관리에 MFA를 사용하려면 `MFA Delete`를 사용 설정합니다. 그러나 AWS Management Console을 사용하여 `MFA Delete`를 사용할 수 없습니다. AWS Command Line Interface(AWS CLI) 또는 API를 사용해야 합니다. 자세한 내용은 MFA Delete 구성 섹션을 참조하세요.

AWS CLI 사용

다음 예시에서는 S3 범용 버킷의 버전 관리를 활성화합니다.

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket1 --versioning-
configuration Status=Enabled
```

다음 예에서는 버킷에서 S3 버전 관리 및 다중 인증(MFA) 삭제를 활성화합니다.

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket1 --versioning-
configuration Status=Enabled,MFADelete=Enabled --mfa "SERIAL 123456"
```

Note

MFA Delete를 사용하려면 승인된 물리적 또는 가상 인증 디바이스가 필요합니다. Amazon S3에서 MFA Delete 사용에 대한 자세한 내용은 MFA Delete 구성 섹션을 참조하세요.

AWS CLI를 사용하여 버전 관리를 사용하는 방법에 대한 자세한 내용은 AWS CLI 명령 참조의 put-bucket-versioning을 참조하세요.

AWS SDK 사용

다음 예제에서는 버킷의 버전 관리를 사용한 다음 AWS SDK for Java 및 AWS SDK for .NET을 사용하여 버전 관리 상태를 검색합니다. 다른 AWS SDK 사용에 대한 자세한 내용은 AWS 개발자 센터를 참조하세요.

.NET

코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

```
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;
            }
    }
}
```

Java

실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 AWS SDK for Java 개발자 안내서에서 시작하기 섹션을 참조하세요.

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

Python

다음 Python 코드 예제는 Amazon S3 버킷을 생성하고 버전 관리를 위해 이를 사용하며 7일 이후 최신이 아닌 객체 버전을 만료하는 수명 주기를 구성합니다.

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
 lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
```

```
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket
```

MFA Delete 구성

Amazon S3 버킷에서 S3 버전 관리를 사용하는 경우 선택적으로 MFA(멀티 팩터 인증) Delete를 사용 설정하도록 버킷을 구성하여 다른 보안 계층을 추가할 수 있습니다. 이렇게 하면 버킷 소유자가 버전을 삭제하거나 버킷의 버전 관리 상태를 변경하는 모든 요청에 두 가지 형식의 인증을 포함해야 합니다.

MFA Delete는 다음 작업에 대해 추가 인증을 요구합니다.

- 버킷의 버전 관리 상태 변경
- 객체 버전 영구 삭제

MFA Delete는 다음 두 가지 인증 유형을 모두 요구합니다.

- 보안 자격 증명
- 승인된 인증 디바이스에 표시된 유효한 일련 번호, 공백, 6자리 코드 연결

MFA Delete는 예를 들어 보안 자격 증명이 손상된 경우에 대비하여 보안을 강화합니다. MFA Delete는 삭제 작업을 시작하는 사용자에게 MFA 코드를 통해 MFA 디바이스의 물리적 소유를 증명할 것을 요구하고 삭제 작업에 추가 마찰 및 보안 계층을 추가합니다. 따라서 실수로 인한 버킷 삭제를 방지하는 데 도움이 될 수 있습니다.

MFA 삭제가 활성화된 버킷을 식별하려면 Amazon S3 스토리지 렌즈 지표를 사용하면 됩니다. S3 스토리지 렌즈는 조직 전반에서 객체 스토리지 사용 및 활동에 대한 가시성을 확보하는 데 사용할 수 있는 클라우드 스토리지 분석 기능입니다. 자세한 내용은 S3 스토리지 렌즈를 사용한 스토리지 활동 및 사용량 평가를 참조하십시오. 지표의 전체 목록은 S3 스토리지 렌즈 지표 용어집을 참조하십시오.

버킷 소유자, 버킷을 생성한 AWS 계정(루트 계정) 및 승인된 모든 사용자는 버전 관리를 사용할 수 있습니다. 그러나 MFA 삭제는 버킷 소유자(루트 계정)만 사용 설정할 수 있습니다. 자세한 내용은 AWS 보안 블로그에서 MFA를 사용하여 AWS에 대한 액세스 보안을 참조하세요.

> **Note**
>
> 버전 관리에 MFA Delete를 사용하려면 `MFA Delete`를 사용 설정합니다. 그러나 AWS Management Console을 사용하여 `MFA Delete`를 사용할 수 없습니다. AWS Command Line Interface(AWS CLI) 또는 API를 사용해야 합니다.
>
> 버전 관리에서 MFA Delete를 사용하는 예는 주제 버킷에 버전 관리 사용 설정의 예제 섹션을 참조하세요.
>
> 수명 주기 구성과 함께 MFA 삭제를 사용할 수 없습니다. 수명 주기 구성 및 수명 주기 구성이 다른 구성과 상호 작용하는 방법에 대한 자세한 내용은 S3 수명 주기가 다른 버킷 구성과 상호 작용하는 방식 섹션을 참조하세요.

MFA Delete를 사용 설정 또는 사용 중지하려면 버킷에 대한 버전 관리를 구성하는 데 사용한 것과 동일한 API를 사용합니다. Amazon S3는 버킷의 버전 관리 상태를 저장하는 동일한 버전 관리 하위 리소스에 MFA Delete 구성을 저장합니다.

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>VersioningState</Status>
  <MfaDelete>MfaDeleteState</MfaDelete>
</VersioningConfiguration>
```

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.



MFA Delete 및 MFA 보호 API 액세스는 서로 다른 시나리오에 대한 보호를 제공하기 위해 마련된 기능입니다. 버킷의 데이터가 실수로 삭제되는 것을 방지하려면 버킷에 MFA Delete를 구성합니다. MFA 보호 API 액세스는 민감한 Amazon S3 리소스에 액세스할 때 다른 인증 요소(MFA 코드)를 적용하는 데 사용됩니다. 이 Amazon S3 리소스에 대한 모든 작업은 MFA를 사용하여 생성한 임시 자격 증명으로 수행하도록 요구할 수 있습니다. 관련 예제는 MFA 필요 섹션을 참조하세요

인증 디바이스를 구입하고 활성화하는 방법에 대한 자세한 내용은 멀티 팩터 인증을 참조하세요.

S3 버전 관리를 활성화하고 MFA 삭제를 구성하려면

AWS CLI 사용

일련 번호는 MFA 디바이스를 고유하게 식별하는 번호입니다. 물리적 MFA 디바이스의 경우, 디바이스와 함께 제공되는 고유 일련 번호입니다. 가상 MFA 디바이스의 경우 일련 번호는 디바이스 ARN입니다.

다음 예에서는 버킷에서 S3 버전 관리 및 다중 인증(MFA) 삭제를 활성화합니다.

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket1 --versioning-
configuration Status=Enabled,MFADelete=Enabled --mfa "SERIAL 123456"
```

REST API 사용

Amazon S3 REST API를 사용하여 MFA 삭제를 지정하는 방법에 대한 자세한 내용은 PutBucketVersioning Amazon Simple Storage Service API 참조에서 확인하세요.

버전 관리가 사용 설정된 버킷의 객체 작업

버전 관리 상태로 설정하기 전에 Amazon S3 버킷에 저장된 객체의 버전 ID는 null이 됩니다. 버전 관리를 사용 설정할 때 버킷의 기존 객체는 변경되지 않습니다. 이후 요청에서는 Amazon S3가 객체를 처리하는 방법만 변경됩니다.

객체 버전 이전

또한 수명 주기가 명확하게 정의된 객체에 대해 수명 주기 구성 규칙을 정의하여 객체 수명의 특정 시점에 S3 Glacier Flexible Retrieval 스토리지 클래스로 객체 버전을 이전할 수 있습니다. 자세한 내용은 객체 수명 주기 관리 섹션을 참조하세요.

이 섹션의 주제에서는 버전 관리를 사용하는 버킷의 다양한 객체 작업에 관해 설명합니다. 버전 관리에 대한 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하세요.

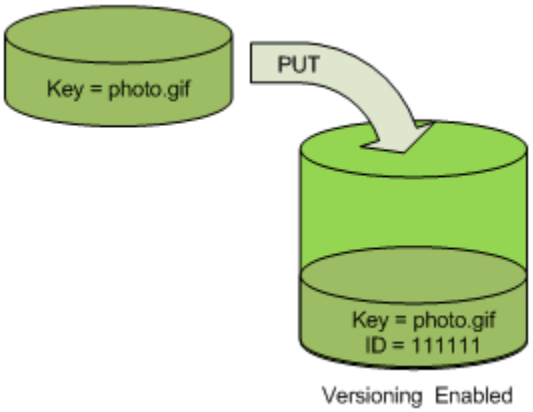
주제

- 버전 관리를 사용하는 버킷에 객체 추가
- 버전 관리를 사용하는 버킷의 객체 목록
- 버전 관리가 사용 설정된 버킷에서 객체 버전 검색
- 버전 관리가 사용 설정된 버킷에서 객체 버전 삭제
- 버전 관리되는 객체 사용 권한 구성

버전 관리를 사용하는 버킷에 객체 추가

버킷에 대한 버전 관리를 사용 설정하면 Amazon S3는 고유한 버전 ID를 버킷에 저장된 각 객체에 자동으로 추가합니다(PUT, POST 또는 CopyObject 사용).

다음 그림은 버전 관리를 사용하는 버킷에 객체를 추가할 때 Amazon S3에서 고유한 버전 ID를 객체에 추가하는 것을 보여줍니다.



> **Note**
>
> Amazon S3에서 할당하는 버전 ID 값은 URL로 사용 가능합니다(URI의 일부로 포함할 수 있음).

버전 관리에 대한 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하십시오. 콘솔, AWS SDK 및 REST API를 사용하여 버전 관리가 사용되는 버킷에 객체 버전을 추가할 수 있습니다.

콘솔 사용

지침은 객체 업로드 섹션을 참조하세요.

AWS SDK 사용

AWS SDK for Java/.NET/PHP를 사용한 객체 업로드 예제는 객체 업로드 섹션을 참조하세요. 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 AWS 개발자 센터를 참조하세요.

REST API 사용

버전 관리가 사용 설정된 버킷에 객체 추가

1. `PutBucketVersioning` 요청을 사용하여 버킷의 버전 관리를 사용 설정합니다.

   자세한 내용은 Amazon Simple Storage Service API 참조에서 PutBucketVersioning를 참조하십시오.

2. PUT, POST 또는 CopyObject 요청을 보내 버킷에 객체를 저장합니다.

버전 관리가 사용 설정된 버킷에 객체를 추가할 경우, Amazon S3는 다음 예와 같이 객체의 버전 ID를 x-amz-version-id 응답 헤더로 반환합니다.

```
x-amz-version-id: 3/L4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY
```

버전 관리를 사용하는 버킷의 객체 목록

이 섹션에서는 버전 관리를 사용하는 버킷의 객체 버전을 나열하는 예제를 제공합니다. Amazon S3에서는 객체 버전 정보를 버킷에 연결된 versions 하위 리소스에 저장합니다. 자세한 내용은 범용 버킷 구성 옵션 섹션을 참조하세요. 버전 관리가 활성화된 버킷의 객체를 나열하려면 ListBucketVersions 권한이 필요합니다.

S3 콘솔 사용

다음 단계에 따라 Amazon S3 콘솔을 사용하여 객체의 여러 버전을 확인합니다.

객체의 여러 버전 보기

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷 목록에서 객체가 포함된 버킷의 이름을 선택합니다.
3. 버킷에 있는 객체 버전 목록을 확인하려면 버전 표시 스위치를 선택합니다.

   각 객체 버전에 대해 콘솔에 고유한 버전 ID, 객체 버전이 생성된 날짜/시간 및 기타 속성이 표시됩니다. (버전 관리 상태로 설정하기 전에 버킷에 저장된 객체의 버전 ID는 null이 됩니다.)

   해당 버전이 없는 객체를 나열하려면 버전 나열(List versions) 스위치를 선택합니다.

또한, 콘솔의 객체 개요 패널에서도 객체 버전을 보고, 다운로드하며, 삭제할 수 있습니다. 자세한 내용은 Amazon S3 콘솔에서 객체 속성 보기 섹션을 참조하세요.

> **Note**
>
> 300개 버전 이전의 객체 버전에 액세스하려면 AWS CLI 또는 객체의 URL을 사용해야 합니다.

> **Important**
>
> 객체는 최신 버전으로 삭제한 경우에만 삭제를 취소할 수 있습니다. 삭제했던 객체의 이전 버전은 삭제를 취소할 수 없습니다. 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하세요.

AWS SDK 사용

이 섹션의 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 `null`의 버전 ID를 사용하여 객체 목록을 반환합니다.

다른 AWS SDK 사용에 대한 자세한 내용은 AWS 개발자 센터를 참조하세요.

Java

실제 예제를 작성 및 테스트하는 방법에 대한 자세한 내용은 AWS SDK for Java 개발자 안내서의 시작하기 섹션을 참조하세요.

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

.NET

코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

```
using Amazon;
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

Example - 버킷의 모든 객체 버전 나열

버킷에 있는 전체 객체의 버전을 모두 나열하려면 versions 요청에 GET Bucket 하위 리소스를 사용합니다. Amazon S3에서는 최대 1,000개의 객체를 검색할 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.gif의 990개 버전 모드를 가져오고 picture.jpg의 최신 버전 10개만 가져오게 됩니다.

Amazon S3에서는 객체 버전을 객체가 저장된 순서대로 가장 최근에 저장된 것을 먼저 반환합니다.

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=
```

Example - 키의 전체 버전 검색

객체 버전의 하위 집합을 가져오려면 GET Bucket에 요청 파라미터를 사용합니다. 자세한 내용은 GET Bucket 섹션을 참조하세요.

1. prefix 파라미터를 검색하려는 객체의 키로 설정합니다.
2. GET Bucket 하위 리소스 및 versions를 사용하여 prefix 요청을 보냅니다.

   ```
   GET /?versions&prefix=objectName HTTP/1.1
   ```

Example - 접두사를 사용하여 객체 검색

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 Amazon Simple Storage Service API 참조에서 GET Bucket를 참조하십시오.

Example - 응답이 잘린 경우 추가 객체의 목록 검색

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 Amazon Simple Storage Service API 참조의 GET Bucket을 참조하세요.

다음은 원래 GET 요청을 충족하는 추가 응답입니다.

- key-marker의 값을 이전 응답의 NextKeyMarker에서 반환된 키로 설정합니다.
- version-id-marker의 값을 이전 응답의 NextVersionIdMarker에서 반환된 버전 ID로 설정합니다.
- GET Bucket versions 및 key-marker를 사용하여 version-id-marker 요청을 보냅니다.

Example - 지정된 키 및 버전 ID로 시작하는 객체 검색

```
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=
```

AWS CLI 사용

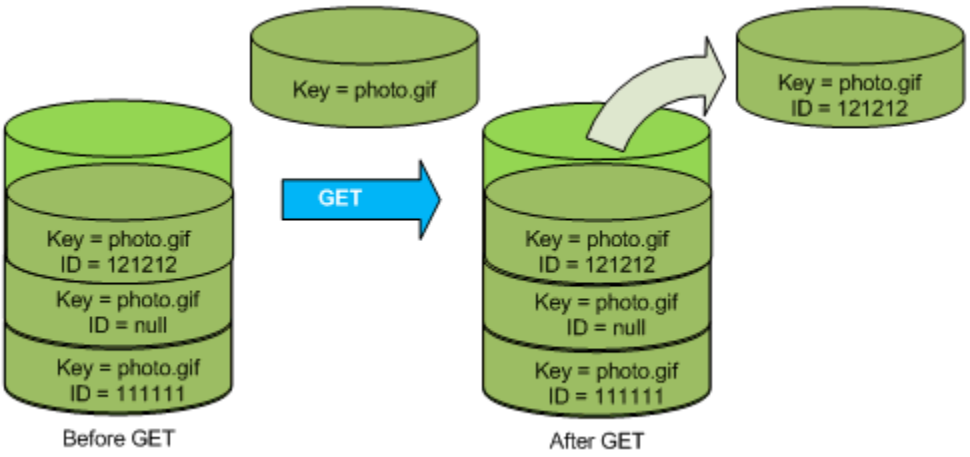
다음 명령은 버킷에 있는 객체의 모든 버전에 대한 메타데이터를 반환합니다.

```
aws s3api list-object-versions --bucket amzn-s3-demo-bucket1
```

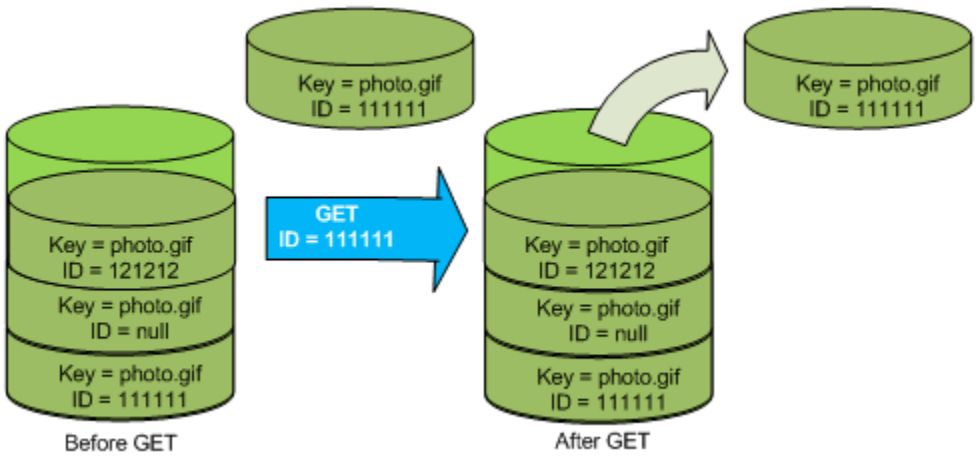
list-object-versions에 대한 자세한 정보는 AWS CLI 명령 참조의 list-object-versions 섹션을 참조하세요.

버전 관리가 사용 설정된 버킷에서 객체 버전 검색

Amazon S3에서의 버전 관리는 객체의 여러 변형을 동일한 버킷에 보관하는 방법입니다. 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, AWS SDK 또는 REST API를 사용하여 Amazon S3의 객체 버전을 검색할 수 있습니다.

> **Note**
>
> 300개 버전 이전의 객체 버전에 액세스하려면 AWS CLI 또는 객체의 URL을 사용해야 합니다.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷 목록에서 객체가 포함된 버킷의 이름을 선택합니다.
3. 객체 목록에서 객체의 이름을 선택합니다.
4. 버전을 선택합니다.

   Amazon S3가 객체의 모든 버전을 표시합니다.
5. 검색할 버전의 [버전 ID(Version ID)] 옆에 있는 확인란을 선택합니다.
6. [작업(Actions)]을 선택하고 [다운로드(Download)]를 선택한 다음 객체를 저장합니다.

또한 객체 개요 패널에서 객체 버전의 보기, 다운로드 및 삭제도 가능합니다. 자세한 내용은 Amazon S3 콘솔에서 객체 속성 보기 섹션을 참조하세요.

> **Important**
>
> 객체는 최신 버전으로 삭제한 경우에만 삭제를 취소할 수 있습니다. 삭제했던 객체의 이전 버전은 삭제를 취소할 수 없습니다. 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하세요.

AWS SDK 사용

버전 관리가 사용되지 않는 버킷과 버전 관리가 사용 설정된 버킷의 객체를 업로드하는 예는 동일합니다. 그러나 버전 관리가 사용 설정된 버킷의 경우 Amazon S3는 버전 번호를 할당합니다. 그렇지 않은 경우 버전 번호는 null이 됩니다.

AWS SDK for Java/.NET/PHP를 사용하여 객체를 다운로드하는 예제는 객체 다운로드를 참조하세요.

.NET 및 Rust용 AWS SDK를 사용하여 객체 버전을 나열하는 예시는 Amazon S3 버킷의 객체 버전 나열을 참조하세요.

REST API 사용

특정 객체 버전 가져오기

1. `versionId`를 검색하려는 객체의 버전 ID로 설정합니다.
2. `GET Object versionId` 요청을 보냅니다.

Example - 버전이 지정된 객체 검색

다음 요청은 L4kqtJlcpXroDTDmpUMLUo의 my-image.jpg 버전을 검색합니다.

```
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=
```

콘텐츠가 아니라 객체의 메타데이터만 검색할 수 있습니다. 자세한 정보는 the section called “버전 메타데이터 검색” 섹션을 참조하세요.

이전 객체 버전 복원에 대한 자세한 내용은 the section called “이전 버전 복원” 섹션을 참조하세요.

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 메타데이터 가져오기

1. 메타데이터를 가져오려는 객체의 버전 ID에 versionId를 설정합니다.
2. HEAD Object versionId 요청을 보냅니다.

Example - 버전이 지정된 객체의 메타데이터 검색

다음 요청은 my-image.jpg의 3HL4kqCxf3vjVBH40Nrjfkd 버전 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
```

이전 버전 복원

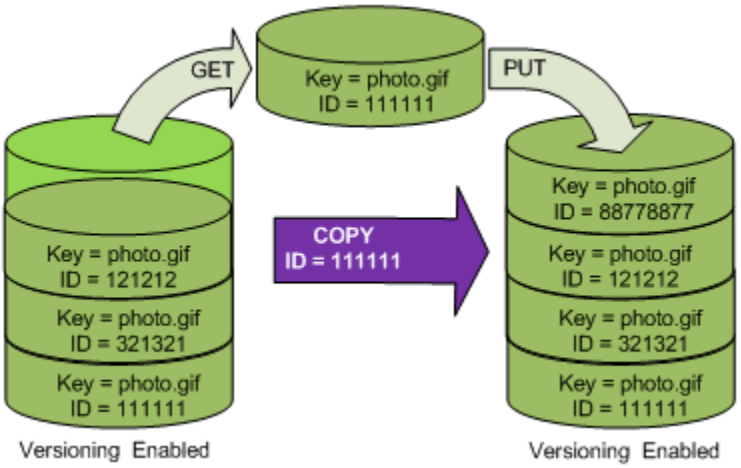
버전 관리를 사용하여 객체의 이전 버전을 검색할 수 있습니다. 이를 수행하기 위한 접근 방식으로 다음 두 가지가 있습니다.

- 이전 버전의 객체를 동일한 버킷으로 복사합니다.

  복사된 객체는 해당 객체의 현재 버전이 되고 모든 객체 버전은 유지됩니다.
- 객체의 현재 버전 영구 삭제

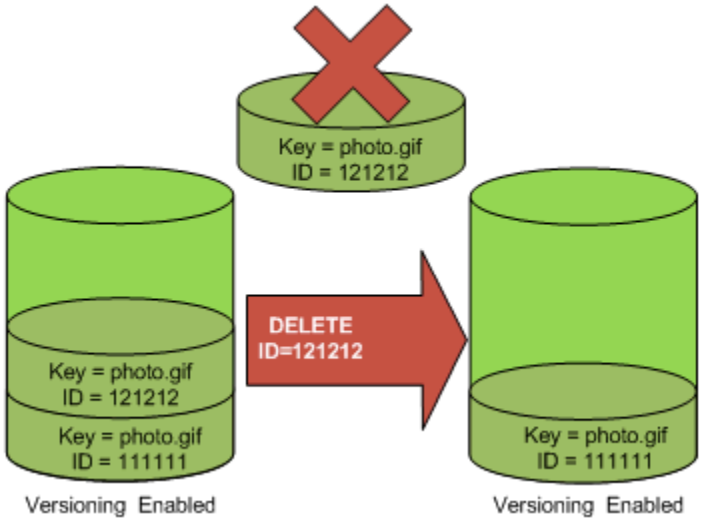
  현재 객체 버전을 삭제하는 것은 실제로 해당 객체의 이전 버전을 현재 버전으로 만드는 것과 같습니다.

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 S3에서는 새 ID(88778877)를 제공하며 이것이 객체의 현재 버전이 됩니다. 따라서 버킷에는 원래 객체 버전(111111)과 그 사본(88778877)이 모두 들어 있습니다. 이전 버전을 가져온 후 업로드하여 최신 버전으로 만드는 방법에 대한 자세한 내용은 versioning-enabled 버킷에서 객체 버전 검색 및 객체 업로드를 참조하세요.



후속 GET에서는 88778877을 검색합니다.

다음 그림은 현재 객체로 이전 버전(111111)을 남겨 둔 채로 객체의 현재 버전(121212)을 삭제하는 과정을 보여줍니다. 객체 삭제에 대한 자세한 내용은 단일 객체 삭제를 참조하세요.



후속 GET에서는 111111을 검색합니다.

> **Note**
>
> 객체 버전을 일괄적으로 복원하려면 `CopyObject` 작업을 사용할 수 있습니다. `CopyObject` 작업은 매니페스트에 지정된 각 객체를 복사합니다. 그러나 객체가 매니페스트에 나타나는 것과 반드시 동일한 순서로 복사되는 것은 아니라는 점을 유의하세요. 버전이 지정된 버킷에서 최신/이전 버전 순서를 유지하는 것이 중요한 경우 모든 이전 버전을 먼저 복사해야 합니다. 그런 다음 첫 번째 작업이 완료된 후 후속 작업에서 현재 버전을 복사합니다.

이전 객체 버전 복원

삭제된 객체 복원에 대한 자세한 지침은 AWS re:Post 지식 센터의 버전 관리가 활성화된 버킷에서 삭제된 Amazon S3 객체를 검색하려면 어떻게 해야 하나요?를 참조하세요.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷 목록에서 객체가 포함된 버킷의 이름을 선택합니다.
3. 객체 목록에서 객체의 이름을 선택합니다.
4. 버전을 선택합니다.

Amazon S3가 객체의 모든 버전을 표시합니다.

5. 검색할 버전의 [버전 ID(Version ID)] 옆에 있는 확인란을 선택합니다.
6. [작업(Actions)]을 선택하고 [다운로드(Download)]를 선택한 다음 객체를 저장합니다.

또한 객체 개요 패널에서 객체 버전의 보기, 다운로드 및 삭제도 가능합니다. 자세한 내용은 Amazon S3 콘솔에서 객체 속성 보기 섹션을 참조하세요.

> **Important**
>
> 객체는 최신 버전으로 삭제한 경우에만 삭제를 취소할 수 있습니다. 삭제했던 객체의 이전 버전은 삭제를 취소할 수 없습니다. 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하세요.

AWS SDK 사용

다른 AWS SDK 사용에 대한 자세한 내용은 AWS 개발자 센터를 참조하세요.

Python

다음 Python 코드 예제에서는 지정된 롤백 버전 이후에 발생한 모든 버전을 삭제하여 버전이 지정된 객체의 이전 버전을 복원합니다.

```
def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
```

```
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
f"{object_key}."
        )
```

버전 관리가 사용 설정된 버킷에서 객체 버전 삭제

언제든지 Amazon S3 버킷에서 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` 작업은 현재 객체 버전에 적용됩니다. 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.

S3 수명 주기에 대한 자세한 내용은 객체 수명 주기 관리 및 S3 수명 주기 구성의 예제 섹션을 참조하세요.

버킷에 있는 최신 및 비최신 객체 버전의 수를 확인하려면 Amazon S3 스토리지 렌즈 지표를 사용하면 됩니다. S3 스토리지 렌즈는 조직 전반에서 객체 스토리지 사용 및 활동에 대한 가시성을 확보하는 데 사용할 수 있는 클라우드 스토리지 분석 기능입니다. 자세한 내용은 Amazon S3 스토리지 렌즈를 사용한 스토리지 비용 최적화를 참조하세요. 지표의 전체 목록은 S3 스토리지 렌즈 지표 용어집을 참조하세요.

> **Note**
>
> 최신이 아닌 객체 버전을 포함하여 저장되거나 전송된 객체의 모든 버전에는 일반 Amazon S3 요금이 적용됩니다. 자세한 내용은 Amazon S3 요금을 참조하십시오.

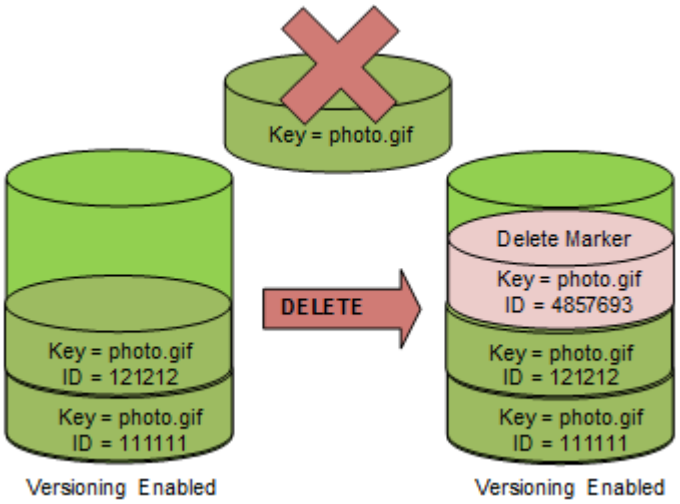
Delete 요청 사용 사례

`DELETE` 요청의 사용 사례는 다음과 같습니다.

- 버전 관리가 사용 설정된 경우, 단순 `DELETE` 요청은 객체를 영구적으로 삭제하지 않습니다. (단순 `DELETE` 요청은 버전 ID를 지정하지 않는 요청입니다.) 그 대신 Amazon S3는 삭제 마커를 버킷에 삽입하고 해당 마커는 새로운 ID를 가진 객체의 현재 버전이 됩니다.

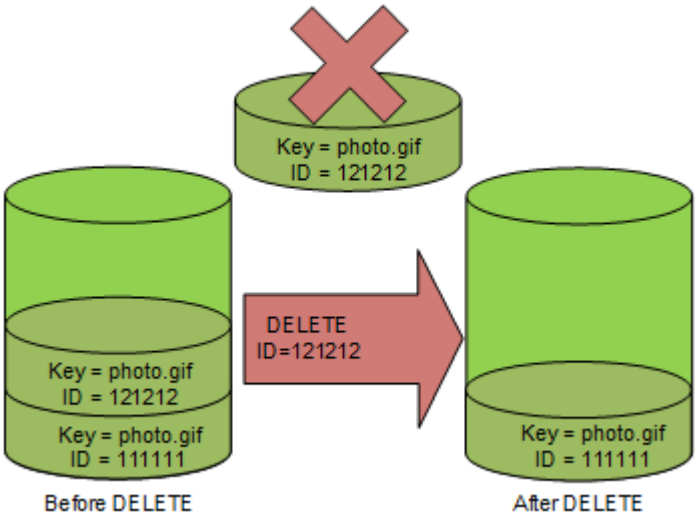
  현재 버전이 삭제 마커인 객체에 대해 `GET`을 시도하는 경우, Amazon S3는 객체가 삭제된 것처럼 동작하여 404 오류를 반환합니다. 자세한 내용은 삭제 마커를 통한 작업 섹션을 참조하세요.

  다음 그림은 단순한 `DELETE` 요청이 지정된 객체를 실제로 제거하지 않음을 보여 줍니다. 대신 Amazon S3에서 삭제 마커를 삽입합니다.



- 버전 관리되는 객체를 영구적으로 삭제하려면 `DELETE Object versionId`를 사용해야 합니다.

  다음 그림은 지정된 객체 버전을 삭제하면 해당 객체가 영구적으로 제거됨을 보여 줍니다.



객체 버전 삭제

콘솔, AWS SDK, REST API 또는 AWS Command Line Interface를 사용하여 Amazon S3에서 객체 버전을 삭제할 수 있습니다.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷 목록에서 객체가 포함된 버킷의 이름을 선택합니다.
3. 객체 목록에서 객체의 이름을 선택합니다.

4. 버전을 선택합니다.

   Amazon S3가 객체의 모든 버전을 표시합니다.
5. 영구적으로 삭제할 버전의 버전 ID(Version ID) 옆에 있는 확인란을 선택합니다.
6. 삭제를 선택합니다.
7. [객체를 영구적으로 삭제하시겠습니까?(Permanently delete objects?)]에 **permanently delete**를 입력합니다.

   > **Warning**
   >
   > 객체 버전을 영구적으로 삭제하면 작업을 실행 취소할 수 없습니다.

8. 객체 삭제를 선택합니다.

   Amazon S3가 객체 버전을 삭제합니다.

AWS SDK 사용

Java, .NET, PHP용 AWS SDK를 사용하여 객체를 삭제하는 예는 Amazon S3 객체 삭제 단원을 참조하세요. 버전 관리가 사용되지 않는 버킷과 버전 관리가 사용 설정된 버킷의 객체를 삭제하는 예는 동일합니다. 그러나 버전 관리가 사용 설정된 버킷의 경우 Amazon S3는 버전 번호를 할당합니다. 그렇지 않은 경우 버전 번호는 null이 됩니다.

다른 AWS SDK 사용에 대한 자세한 내용은 AWS 개발자 센터를 참조하세요.

Python

다음 Python 코드 예제에서는 모든 버전을 삭제하여 버전이 지정된 객체를 영구적으로 삭제합니다.

```
def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
```

```
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

REST API 사용

특정 버전의 객체 삭제

- DELETE에서 버전 ID를 지정합니다.

Example - 특정 버전 삭제

다음 예제에서는 UIORUnfnd89493jJFJ의 photo.gif 버전을 삭제합니다.

```
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
```

AWS CLI 사용

다음 명령은 이름이 *amzn-s3-demo-bucket1*인 버킷에서 test.txt라는 객체를 삭제합니다. 객체의 지정된 버전을 제거하려면 버킷 소유자여야 하고 버전 Id 하위 리소스를 사용해야 합니다.

```
aws s3api delete-object --bucket amzn-s3-demo-bucket1 --key test.txt --version-
id versionID
```

delete-object에 대한 자세한 정보는 AWS CLI 명령 참조의 delete-object 섹션을 참조하세요.

객체 버전 삭제에 대한 자세한 내용은 아래 주제를 참조하세요.

- 삭제 마커를 통한 작업
- 이전 버전을 현재 버전으로 만들기 위해 삭제 마커 제거
- MFA Delete 사용 설정 버킷에서 객체 삭제

삭제 마커를 통한 작업

Amazon S3의 삭제 마커는 단순 DELETE 요청에 지정된 버전이 지정된 객체에 대한 자리 표시자(또는 마커)입니다. 단순 DELETE 요청은 버전 ID를 지정하지 않는 요청입니다. 객체가 버전 관리를 사용하는 버킷에 있기 때문에 이 객체는 삭제되지 않습니다. 그러나 삭제 마커를 사용하면 Amazon S3가 객체가 삭제된 것처럼 작동합니다. 삭제 마커에 Amazon S3 API DELETE 호출을 사용할 수 있습니다. 이렇게 하려면 적절한 권한이 있는 AWS Identity and Access Management (IAM) 사용자 또는 역할을 활용하여 DELETE 요청을 수행해야 합니다.

삭제 마커에는 다른 객체들과 마찬가지로 키 이름(또는 키)과 버전 ID가 있습니다. 그러나 삭제 마커는 다음과 같은 부분에서 다른 객체와는 다릅니다.

- 삭제 마커에 연결된 데이터가 없습니다.
- 삭제 마커는 액세스 제어 목록(ACL) 값과 연결되어 있지 않습니다.
- 삭제 마커에 대한 GET 요청을 실행하면 삭제 마커에 데이터가 없기 때문에 GET 요청에서 아무 것도 검색하지 않습니다. 특히 GET 요청에 versionId가 지정되지 않은 경우 404(찾을 수 없음) 오류가 발생합니다.

삭제 마커는 Amazon S3의 스토리지에 대해 최소 요금을 발생시킵니다. 삭제 마커의 스토리지 크기는 삭제 마커의 키 이름 크기와 같습니다. 키 이름은 유니코드 문자열입니다. 키 이름에 대한 UTF-8 인코딩은 1~4바이트의 스토리지에서 이름의 각 문자별 버킷으로 추가됩니다. 삭제 마커는 S3 Standard 스토리지 클래스에 저장됩니다.

보유하고 있는 삭제 마커의 수와 마커가 어떤 스토리지 클래스에 저장되어 있는지 확인하려면 Amazon S3 Storage Lens를 사용하면 됩니다. 자세한 내용은 Amazon S3 스토리지 렌즈를 사용하여 스토리지 활동 및 사용량 평가 및 Amazon S3 스토리지 렌즈 지표 용어집(을)를 참조하세요.

키 이름에 대한 자세한 내용은 Amazon S3 객체 이름 지정 단원을 참조하십시오. 삭제 마커에 대한 자세한 내용은 삭제 마커 관리 섹션을 참조하세요.

Amazon S3에서만 삭제 마커를 생성할 수 있으며, 버전 관리가 사용 설정되었거나 일시 중지된 버킷의 객체에 대해 사용자가 DeleteObject 요청을 전송할 경우에만 삭제 마커를 생성합니다. DELETE 요청에 지정된 객체는 실제로 삭제되지 않습니다. 그 대신, 삭제 마커가 객체의 현재 버전이 됩니다. 객체의 키 이름(또는 키)은 삭제 마커의 키가 됩니다.

요청에 versionId를 지정하지 않고 객체를 가져올 때, 현재 버전이 삭제 마커인 경우 Amazon S3는 다음과 같이 응답합니다:

- 404 (찾을 수 없음) 오류

- 응답 헤더, `x-amz-delete-marker: true`

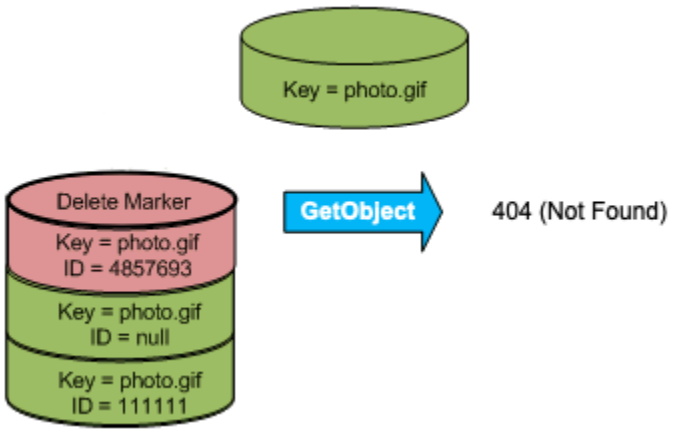
요청에 `versionId`를 지정하여 객체를 가져올 때, 지정된 버전이 삭제 마커인 경우 Amazon S3는 다음과 같이 응답합니다.

- 405(메서드 허용 안 됨) 오류
- 응답 헤더, `x-amz-delete-marker: true`
- 응답 헤더, `Last-Modified: timestamp`(HeadObject 또는 GetObject API 작업을 사용하는 경우에만)

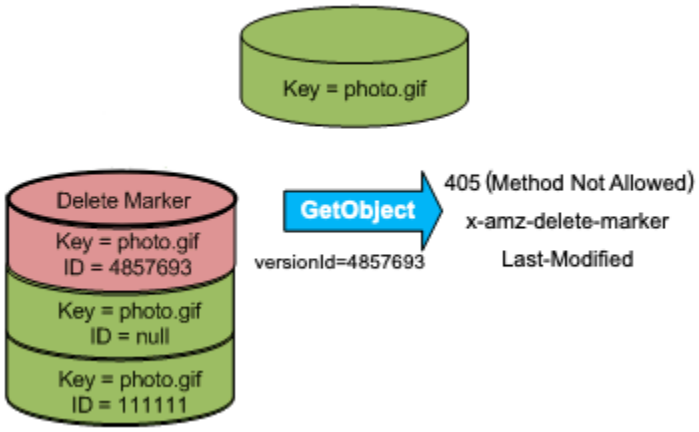
`x-amz-delete-marker: true` 응답 헤더는 액세스한 객체가 삭제 마커였음을 보고합니다. 값이 `false`이면 객체의 현재 버전 또는 지정된 버전이 삭제 마커가 아니므로 이 응답 헤더는 `false`를 반환하지 않습니다.

`Last-Modified` 응답 헤더는 삭제 마커의 생성 시간을 제공합니다.

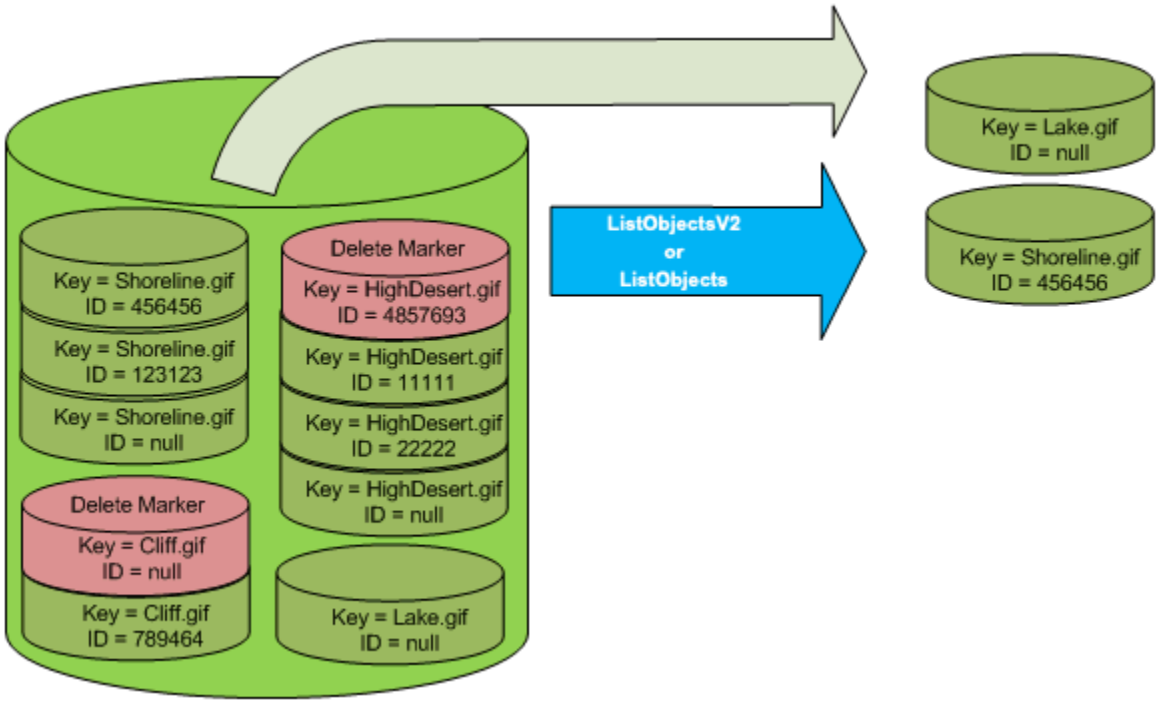
다음 그림은 현재 버전이 삭제 마커인 객체에 대한 `GetObject` API 호출이 404(찾을 수 없음) 오류로 응답하고 응답 헤더에 `x-amz-delete-marker: true`가 포함된 경우를 보여 줍니다.



요청에 `versionId`를 지정하여 객체에 대한 `GetObject` 호출을 수행하고 지정된 버전이 삭제 마커인 경우 Amazon S3는 405(메서드 허용 안 됨) 오류로 응답하고 응답 헤더에 `x-amz-delete-marker: true` 및 `Last-Modified: timestamp`를 포함합니다.



삭제 마커 목록(객체의 다른 버전 포함)을 표시하는 유일한 방법은 `versions` 요청에 ListObjectVersions 하위 리소스를 사용하는 것입니다. 다음 그림은 ListObjectsV2 또는 ListObjects 요청이 현재 버전이 삭제 마커인 객체를 반환하지 않음을 보여줍니다.



삭제 마커 관리

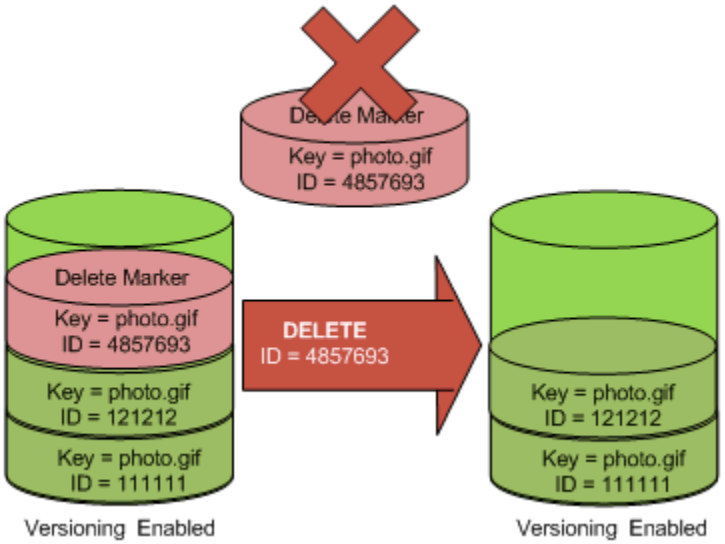
만료된 삭제 마커를 자동으로 정리하는 수명 주기 구성

만료된 객체 삭제 마커는 모든 객체 버전이 삭제되고 하나의 삭제 마커만 남은 마커입니다. 수명 주기 구성이 현재 버전을 삭제하도록 설정되어 있거나 `ExpiredObjectDeleteMarker` 작업이 명시적으로 설정된 경우 Amazon S3는 만료된 객체의 삭제 마커를 제거합니다. 관련 예제는 버전 관리를 사용하는 버킷에서 만료된 객체 삭제 마커 제거 섹션을 참조하세요

이전 버전을 현재 버전으로 만들기 위해 삭제 마커 제거

버전 관리가 사용 설정된 버킷에서 한 객체를 삭제하면 모든 버전이 버킷에 그대로 유지되며 Amazon S3는 해당 객체에 대한 삭제 마커를 생성합니다. 객체 삭제를 취소하려면 이 삭제 마커를 삭제해야 합니다. 버전 관리와 삭제 마커에 대한 자세한 정보는 S3 버전 관리로 여러 버전의 객체 유지을 참조하십시오.

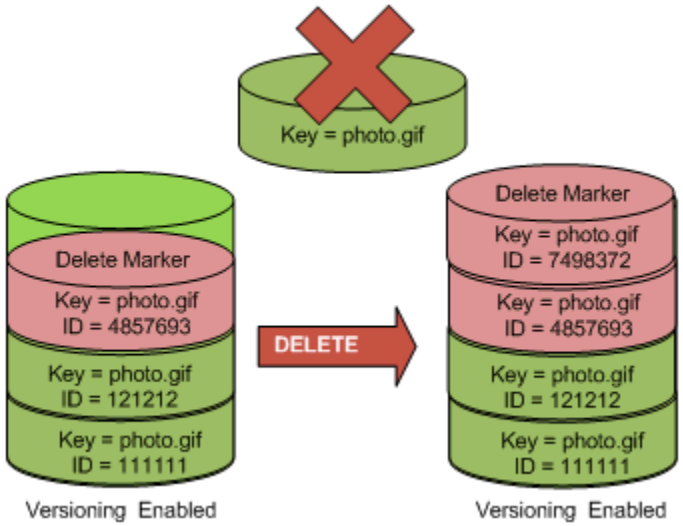
삭제 마커를 영구적으로 삭제하려면 `DeleteObject versionId` 요청에 삭제 마커의 버전 ID를 포함해야 합니다. 다음 그림은 `DeleteObject versionId` 요청이 어떻게 삭제 마커를 영구적으로 제거하는지를 보여 줍니다.



삭제 마커 제거의 영향으로 단순 `GET` 요청은 이제 현재 버전 ID(121212)의 객체를 가져옵니다.

> **Note**
>
> 현재 버전이 삭제 마커인 `DeleteObject` 요청을 사용하는 경우(삭제 마커의 버전 ID 지정 없이) Amazon S3은 삭제 마커를 삭제하지 않고 그 대신 다른 삭제 마커를 넣습니다(`PUT`s).

`NULL` 버전 ID를 사용해 삭제 마커를 삭제하려면 `DeleteObject` 요청에서 버전 ID로 `NULL`을 전달해야 합니다. 다음 그림은 버전 ID 없이 이루어진 단순 `DeleteObject` 요청(현재 버전이 삭제 마커임)으로 아무것도 제거되지 않고, 그 대신 고유한 버전 ID(7498372)의 추가적인 삭제 마커가 추가되는 과정을 보여주고 있습니다.



S3 콘솔 사용

다음 단계를 사용하여 S3 버킷의 폴더가 아닌 S3 버킷에서 삭제된 객체(해당 폴더 내에 있는 객체 포함)를 복원할 수 있습니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷(Buckets) 목록에서 원하는 버킷의 이름을 선택합니다.
3. 버킷에 있는 객체 버전 목록을 확인하려면 [버전 나열(List versions)] 스위치를 선택합니다. 삭제된 객체들에 대한 삭제 마커를 볼 수 있습니다.
4. 어떤 객체의 삭제를 취소하려면 삭제 마커를 삭제해야 합니다. 복원할 객체의 삭제 마커 옆에 있는 확인란을 선택한 후, 삭제(Delete)를 선택합니다.
5. 객체 삭제(Delete objects) 페이지에서 삭제를 확인합니다.
   a. Permanently delete objects?(객체를 영구적으로 삭제하시겠습니까?)에 **`permanently delete`**를 입력합니다.
   b. 객체 삭제(Delete objects)를 선택합니다.

> **Note**
>
> Amazon S3 콘솔을 사용하여 폴더의 삭제를 취소할 수 없습니다. AWS CLI 또는 SDK를 사용해야 합니다. 예를 들어 AWS 지식 센터에서 버전 관리 지원 버킷에서 삭제된 Amazon S3 객체를 검색하려면 어떻게 해야 합니까?를 참조하세요.

REST API 사용

영구적으로 삭제 마커 제거

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
```

AWS SDK 사용

다른 AWS SDK 사용에 대한 자세한 내용은 AWS 개발자 센터를 참조하세요.

Python

다음 Python 코드 예제에서는 객체에서 삭제 마커를 제거하여 현재 버전이 아닌 최신 버전을 객체의 현재 버전으로 만드는 방법을 보여 줍니다.

```
def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
 marker.
    By removing the delete marker, we make the previous version the latest version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.
```

```
    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s  with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
            )
    elif "Versions" in response:
        logger.warning("Got an active version for %s, nothing to do.", object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)
```

MFA Delete 사용 설정 버킷에서 객체 삭제

버킷의 버전 관리 구성에 MFA Delete가 사용 설정된 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 않으면 요청이 실패합니다.

인증 디바이스에 대한 자세한 내용은 멀티 팩터 인증을 참조하세요.

Example - MFA Delete 사용 설정 버킷에서 객체 삭제

다음 예제에서는 MFA Delete가 사용 설정된 버킷에서 지정된 버전의 `my-image.jpg`를 삭제합니다.

`SerialNumber` 및 *`AuthenticationCode`* 사이의 공백에 주의하세요. 자세한 내용은 Amazon Simple Storage Service API 참조에서 DeleteObject를 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 사용 설정에 대한 자세한 내용은 MFA Delete 구성 섹션을 참조하세요.

버전 관리되는 객체 사용 권한 구성

Amazon S3의 객체에 대한 권한은 버전 수준에서 설정됩니다. 각 버전에는 고유한 객체 소유자가 있습니다. 객체 버전을 생성하는 AWS 계정이 소유자입니다. 따라서 동일 객체의 서로 다른 버전에 대해 서로 다른 권한을 설정할 수 있습니다. 이렇게 하려면 `PUT Object versionId acl` 요청에 권한을 설정하려는 객체의 버전 ID를 지정해야 합니다. ACL 사용과 관련한 자세한 설명과 지침은 Amazon S3의 ID 및 액세스 관리 단원을 참조하십시오.

Example - 객체 버전에 대한 권한 설정

다음 요청은 피부여자 `BucketOwner@amazon.com`의 권한을 키 `my-image.jpg`, 버전 ID `3HL4kqtJvjVBH40Nrjfkd`에서 `FULL_CONTROL`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 검색

다음 예시에서 Amazon S3는 키 `my-image.jpg`, 버전 ID `DVBH40Nr8X8gUMLUo`에 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 Amazon Simple Storage Service API 참조에서 GetObjectAcl를 참조하십시오.

버전 관리가 일시 중지된 버킷의 객체 작업

Amazon S3에서 버킷 내 동일 객체의 새 버전이 발생하지 않도록 버전 관리를 일시 중지할 수 있습니다. 버킷에 단일 버전의 객체만 필요한 경우 이 작업을 수행할 수 있습니다. 또는 여러 버전에 대한 요금이 부과되는 것을 원하지 않을 경우에도 이 작업을 수행할 수 있습니다.

버전 관리를 일시 중지할 때 버킷의 기존 객체는 변경되지 않습니다. 이후 요청에서는 Amazon S3가 객체를 처리하는 방법만 변경됩니다. 이 섹션의 주제에서는 버전 관리가 일시 중지된 버킷의 다양한 객체 작업(예: 객체 추가, 검색 및 삭제)에 관해 설명합니다.

S3 버전 관리에 대한 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하십시오. 객체 버전 가져오기에 대한 자세한 내용은 버전 관리가 사용 설정된 버킷에서 객체 버전 검색 섹션을 참조하세요.

주제

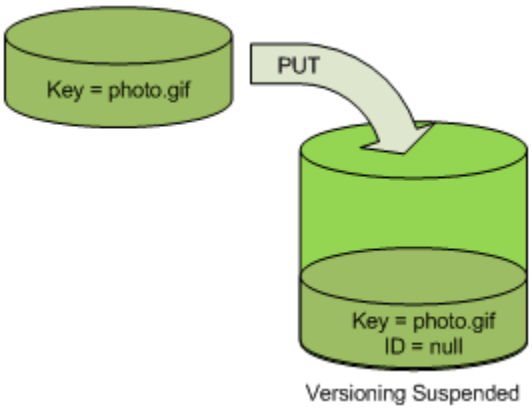
- 버전 관리가 일시 중지된 버킷에 객체 추가
- 버전 관리가 일시 중지된 버킷에서 객체 가져오기
- 버전 관리가 일시 중지된 버킷에서 객체 삭제

버전 관리가 일시 중지된 버킷에 객체 추가

Amazon S3에서 버전 관리가 일시 중지된 버킷에 객체를 추가하여 버전 ID가 null인 객체를 만들거나, 일치하는 버전 ID로 객체 버전을 덮어쓸 수 있습니다.

버킷에 대한 버전 관리를 일시 중지하면 Amazon S3에서는 버전 ID `null`을 해당 버킷에 이후로 저장되는 후속 객체 각각에 자동으로 추가합니다(PUT, POST 또는 `CopyObject` 사용).

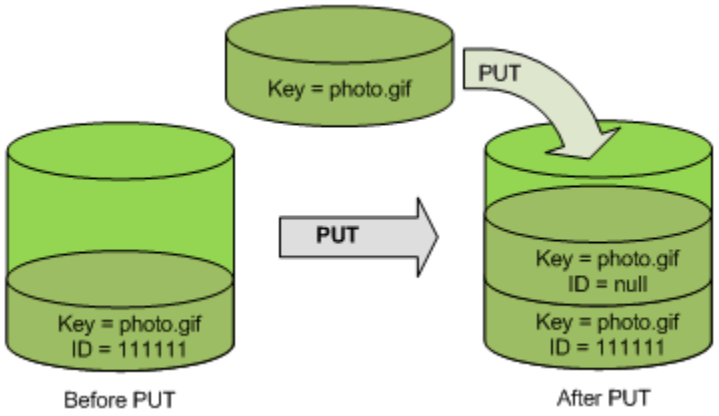
다음 그림은 버전 관리가 일시 중지된 버킷에 객체를 추가할 때 Amazon S3에서 버전 ID `null`을 객체에 추가하는 과정을 보여줍니다.



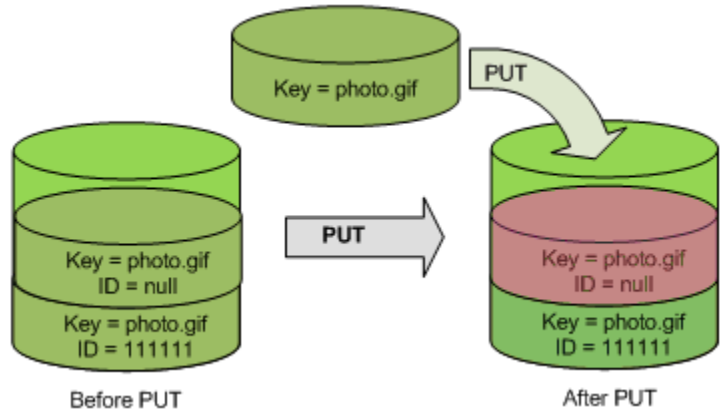
null 버전이 버킷에 이미 있는 경우 다른 객체를 같은 키로 추가하면 추가된 객체는 원래 null 버전을 덮어씁니다.

버전이 지정된 객체가 버킷에 있는 경우, PUT을 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 S3에서는 추가되는 객체에 null 버전 ID를 연결하고 이를 버킷에 저장합니다. 따라서 이는 버전 111111을 덮어쓰지 않습니다.



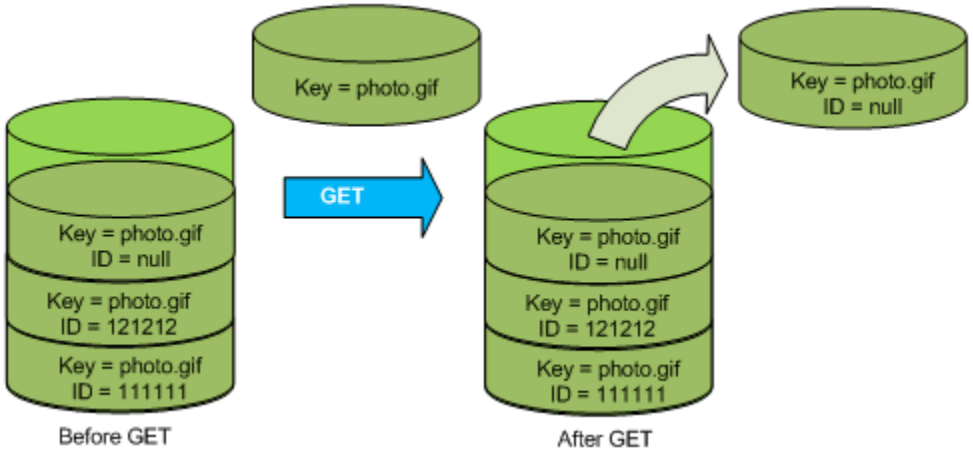
null 버전이 버킷에 이미 존재하는 경우, 다음 그림과 같이 이 null 버전을 덮어쓰게 됩니다.



Null 버전의 키 및 버전 ID(null)가 PUT 수행 이전과 이후에 서로 같더라도, 버킷에 원래 저장되어 있던 Null 버전의 콘텐츠는 버킷에 PUT을 수행한 객체의 콘텐츠로 대체됩니다.

버전 관리가 일시 중지된 버킷에서 객체 가져오기

GET Object 요청은 버킷에 대해 버전 관리를 사용 설정했는지 여부와 관계없이 현재 버전의 객체를 반환합니다. 다음 그림은 단순 GET 요청에서 현재 버전의 객체를 반환하는 과정을 보여 줍니다.



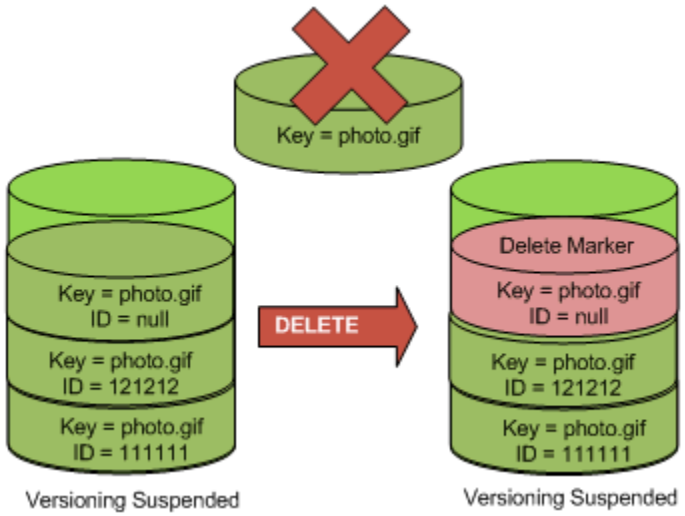
버전 관리가 일시 중지된 버킷에서 객체 삭제

버전 관리가 일시 정지된 버킷에서 객체를 삭제하여 null 버전 ID를 가진 객체를 제거할 수 있습니다.

버킷에 대해 버전 관리가 일시 중지된 경우 DELETE 요청은 다음과 같습니다.

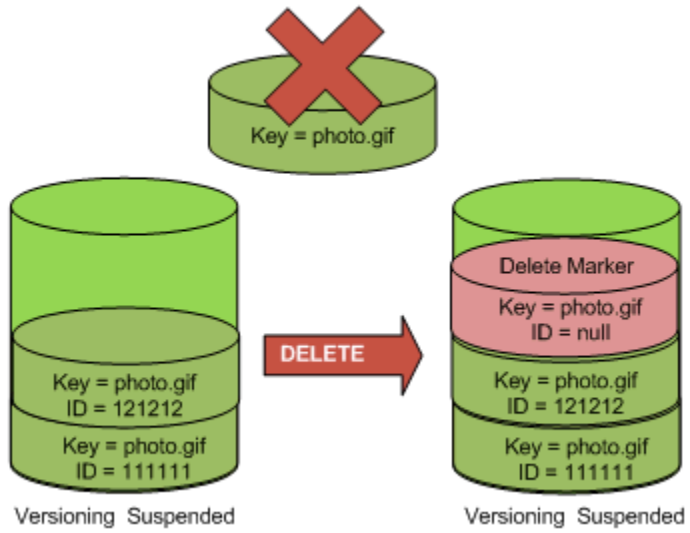
- 버전 ID가 null인 객체만 제거할 수 있습니다.
- 버킷에 null 버전의 객체가 없는 경우 어떤 것도 제거하지 않습니다.
- 버킷에 삭제 마커를 삽입합니다.

버킷 버전 관리가 일시 중단된 경우 작업은 null versionId가 있는 객체를 제거합니다. 버전 ID가 있는 경우 Amazon S3는 객체의 현재 버전이 되는 삭제 마커를 삽입합니다. 다음 그림은 단순 DELETE가 null 버전을 제거하고 Amazon S3가 대신 null 버전 ID와 함께 삭제 마커를 삽입하는 과정을 보여줍니다.

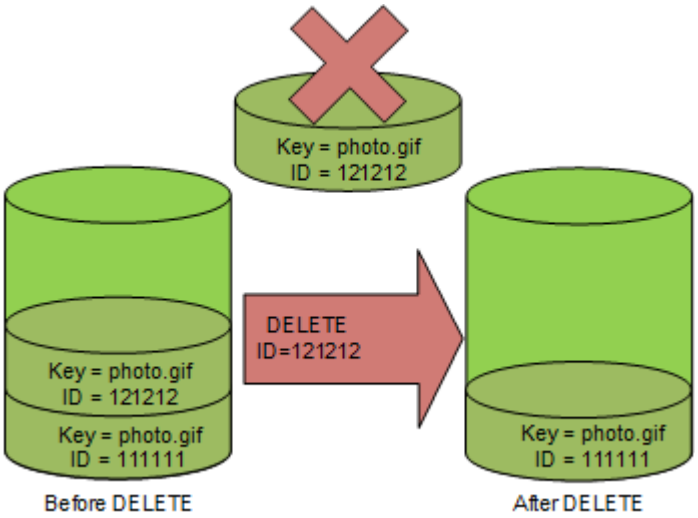


versionId가 있는 객체를 영구 삭제하려면 요청에 해당 객체 versionId를 포함해야 합니다. 삭제 마커에는 콘텐츠가 포함되어 있지 않으므로, 삭제 마커로 대체될 때 null 버전의 콘텐츠가 손실됩니다.

다음 그림은 null 버전이 없는 버킷을 보여 줍니다. 이 경우 DELETE는 아무것도 제거하지 않습니다. 대신 Amazon S3에서 삭제 마커를 삽입합니다.



버전 관리가 일시 정지된 버킷에서도 버킷 소유자는 DELETE 요청에 버전 ID를 포함하여 특정 버전을 영구적으로 삭제할 수 있습니다. 다음 그림은 지정된 객체 버전을 삭제하면 해당 객체의 버전이 영구적으로 제거됨을 보여 줍니다. 버킷 소유자만이 지정된 객체 버전을 삭제할 수 있습니다.



버전 관리 문제 해결

다음 주제는 몇 가지 일반적인 Amazon S3 버전 관리 문제를 해결하는 데 도움이 될 수 있습니다.

주제

- 버전 관리를 활성화한 버킷에서 실수로 삭제된 객체를 복구하려고 합니다.
- 버전이 지정된 객체를 영구적으로 삭제하고 싶습니다.
- 버킷 버전 관리를 활성화한 후 성능 저하가 발생했습니다.

버전 관리를 활성화한 버킷에서 실수로 삭제된 객체를 복구하려고 합니다.

일반적으로 S3 버킷에서 객체 버전이 삭제되면 Amazon S3에서 복구할 방법이 없습니다. 하지만 S3 버킷에 S3 버전 관리를 활성화한 경우 버전 ID를 지정하지 않는 DELETE 요청은 객체를 영구적으로 삭제할 수 없습니다. 대신 삭제 마커가 자리 표시자로 추가됩니다. 이 삭제 마커가 객체의 최신 버전이 됩니다.

삭제된 객체가 영구적으로 삭제되었는지 아니면 일시적으로 삭제되었는지(삭제 마커로 대체됨) 확인하려면 다음을 수행하세요.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 목록에서 객체가 포함된 버킷의 이름을 선택합니다.
4. 객체 목록에서 검색 창 오른쪽에 있는 버전 표시 토글을 켠 다음 검색 창에서 삭제된 객체를 검색합니다. 이 토글은 이전에 버킷에 버전 관리를 활성화한 경우에만 사용할 수 있습니다.

   S3 인벤토리를 사용하여 삭제된 객체를 검색할 수도 있습니다.
5. 버전 표시 토글을 켜거나 인벤토리 보고서를 만든 후에도 객체를 찾을 수 없고 객체의 삭제 마커도 찾을 수 없는 경우 영구적으로 삭제된 것이며 객체를 복구할 수 없습니다.

AWS Command Line Interface(AWS CLI)의 HeadObject API 작업을 사용하여 삭제된 객체의 상태를 확인할 수도 있습니다. 이 방법을 사용하려면 다음 head-object 명령을 사용하되 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key index.html
```

현재 버전이 삭제 마커인 버전이 지정된 객체에 대한 head-object 명령을 실행하는 경우 404 찾을 수 없음 오류가 발생합니다. 예시:

HeadObject 작업을 호출할 때 오류(404 찾을 수 없음)가 발생했습니다.

버전이 지정된 객체에서 head-object 명령을 실행하고 객체의 버전 ID를 제공하면 Amazon S3는 객체의 메타데이터를 검색하여 객체가 여전히 존재하며 영구적으로 삭제되지 않았음을 확인합니다.

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key index.html --
version-id versionID
```

```
{
"AcceptRanges": "bytes",
"ContentType": "text/html",
"LastModified": "Thu, 16 Apr 2015 18:19:14 GMT",
"ContentLength": 77,
"VersionId": "Zg5HyL7m.eZU9iM7AVlJkrqAiE.0UG4q",
"ETag": "\"30a6ec7e1a9ad79c203d05a589c8b400\"",
"Metadata": {}
}
```

객체가 검색되고 최신 버전이 삭제 마커인 경우 객체의 이전 버전이 여전히 존재합니다. 삭제 마커가 객체의 현재 버전이므로 삭제 마커를 삭제하여 객체를 복구할 수 있습니다.

삭제 마커를 영구적으로 제거하면 바로 이전 버전의 객체가 객체의 현재 버전이 되어 객체를 다시 사용할 수 있게 됩니다. 객체 복구 방법을 시각적으로 확인하려면 삭제 마커 제거를 참조하세요.

객체의 지정된 버전을 제거하려면 사용자가 버킷 소유자여야 합니다. 삭제 마커를 영구적으로 삭제하려면 `DeleteObject` 요청에 삭제 마커의 버전 ID를 포함해야 합니다. 이 방법을 사용하려면 다음 명령을 사용하되 *`user input placeholders`*를 실제 정보로 대체하세요.

```
aws s3api delete-object --bucket amzn-s3-demo-bucket --key index.html --
version-id versionID
```

`delete-object` 명령에 대한 자세한 내용은 AWS CLI 명령 참조의 delete-object 섹션을 참조하세요. 삭제 마커 영구 삭제에 대한 자세한 내용은 삭제 마커 관리 섹션을 참조하세요.

버전이 지정된 객체를 영구적으로 삭제하고 싶습니다.

버전 관리를 사용하는 버킷에서 버전 ID가 없는 `DELETE` 요청은 객체를 영구적으로 삭제할 수 없습니다. 대신 이러한 요청은 삭제 마커를 삽입합니다.

버전이 지정된 객체를 영구적으로 삭제하려면 다음 방법 중에서 선택할 수 있습니다.

- S3 수명 주기 규칙을 생성하여 비최신 버전을 영구적으로 삭제합니다. 비최신 버전을 영구적으로 삭제하려면 객체의 비최신 버전 영구 삭제를 선택하고 다음 일수 이후 객체가 오래된 것으로 간주됨에 숫자를 입력합니다. Number of newer versions to retain(유지할 새 버전 수) 아래에 값을 입력하여 유지할 최신 버전 수를 선택적으로 지정할 수 있습니다. 이 규칙을 생성하는 방법에 대한 자세한 내용은 S3 수명 주기 구성 설정을 참조하세요.
- `DELETE` 요청에 버전 ID를 포함하여 지정된 버전을 삭제합니다. 자세한 내용은 버전 지정된 객체를 영구적으로 삭제하는 방법을 참조하세요.

- 현재 버전을 만료시키는 수명 주기 규칙을 생성합니다. 객체의 현재 버전을 만료시키려면 객체의 현재 버전 만료를 선택한 후 객체 생성 후 일수에 숫자를 입력합니다. 이 수명 주기 규칙을 생성하는 방법에 대한 자세한 내용은 S3 수명 주기 구성 설정을 참조하세요.
- 버전이 지정된 모든 객체를 영구적으로 삭제하고 마커를 삭제하려면 두 개의 수명 주기 규칙을 만드세요. 하나는 현재 버전을 만료시키고 객체의 비최신 버전을 영구적으로 삭제하는 것이고 다른 하나는 만료된 객체 삭제 마커를 삭제하는 것입니다.

버전 관리를 사용하는 버킷에서 버전 ID를 지정하지 않은 DELETE 요청은 버전 ID가 NULL인 객체만 제거할 수 있습니다. 버전 관리가 활성화된 상태에서 객체를 업로드한 경우 버전 ID를 지정하지 않은 DELETE 요청은 해당 객체의 삭제 마커를 생성합니다.

> **Note**
>
> S3 객체 잠금이 활성화된 버킷의 경우 보호된 객체 버전 ID가 있는 DELETE 객체 요청은 403 액세스 거부 오류를 일으킵니다. 버전 ID가 없는 DELETE 객체 요청은 200 확인 응답과 함께 객체의 최신 버전으로 삭제 마커를 추가합니다. 객체 잠금으로 보호되는 객체는 보존 기간 및 법적 보존이 제거될 때까지 영구적으로 삭제할 수 없습니다. 자세한 내용은 the section called "S3 객체 잠금 작동 방식" 섹션을 참조하세요.

버킷 버전 관리를 활성화한 후 성능 저하가 발생했습니다.

삭제 마커나 버전이 지정된 객체가 너무 많고 모범 사례를 따르지 않을 경우 버전 관리를 사용하는 버킷에서 성능 저하가 발생할 수 있습니다.

삭제 마커가 너무 많음

버킷에 버전 관리를 활성화한 후 버전 ID가 없이 객체에 수행된 DELETE 요청은 고유한 버전 ID를 가진 삭제 마커를 생성합니다. 객체의 현재 버전 만료만료 규칙이 포함된 수명 주기 구성은 모든 객체에 고유한 버전 ID가 있는 삭제 마커를 추가합니다. 삭제 마커가 너무 많으면 버킷의 성능이 저하될 수 있습니다.

버전 관리가 중단된 버킷의 경우 Amazon S3는 새로 생성된 객체에 버전 ID를 NULL로 표시합니다. 버전 관리가 일시 중지된 버킷에서는 만료 작업으로 인해 Amazon S3가 버전 ID가 NULL인 삭제 마커를 생성합니다. 버전 관리가 일시 중단된 버킷에서는 모든 삭제 요청에 대해 NULL 삭제 마커가 생성됩니다. 모든 객체 버전이 삭제되고 하나의 삭제 마커만 남은 경우 이러한 NULL 삭제 마커는 만료된 객체 삭제 마커라고도 합니다. NULL 삭제 마커가 너무 많이 누적되면 버킷의 성능이 저하됩니다.

버전이 지정된 객체가 너무 많음

버전 관리를 사용하는 버킷에 수백만 개의 버전이 있는 객체가 포함된 경우 503 서비스 사용 불가 오류가 증가할 수 있습니다. S3 버전 관리를 사용하는 버킷에 대한 PUT 또는 DELETE 객체 요청에서 받은 HTTP 503 서비스 사용 불가 응답의 횟수가 크게 증가한다면 버전이 수백만 개인 객체가 하나 이상의 버킷에 있을 것으로 추정됩니다. 객체에 수백만 개의 버전이 있을 경우 Amazon S3는 자동으로 해당 버킷에 대한 요청을 제한합니다. 요청을 제한하면 과도한 횟수의 요청 트래픽으로부터 버킷이 보호되어 같은 버킷에 대한 다른 요청을 지연시킬 가능성이 있습니다.

수백만 개의 버전이 있는 S3 객체를 파악하려면 S3 인벤토리 도구를 사용하세요. S3 인벤토리는 버킷에 있는 객체의 플랫 파일 목록을 제공하는 보고서를 생성합니다. 자세한 내용은 S3 Inventory를 사용한 데이터 카탈로그화 및 분석 섹션을 참조하세요.

버킷에 버전이 지정된 객체 수가 많은지 확인하려면 S3 스토리지 렌즈 지표를 사용하여 현재 버전 객체 수, 최신이 아닌 버전 객체 수 및 삭제 마커 객체 수를 확인하세요. 스토리지 렌즈 지표에 대한 자세한 내용은 Amazon S3 스토리지 렌즈 지표 용어집 섹션을 참조하세요.

Amazon S3 팀은 같은 객체를 반복적으로 덮어써서 해당 객체에 수백만 개 버전이 생성될 가능성이 있는 애플리케이션을 조사할 것을 권장합니다. 조사를 통해 애플리케이션이 의도한 대로 작동하는지 판단할 수 있습니다. 예를 들어, 1주일 동안 1분마다 같은 객체를 덮어쓰는 애플리케이션은 1만 개 이상의 버전을 만들 수 있습니다. 각 객체에 대해 10만 개 미만의 버전을 저장하는 것이 좋습니다. 하나 이상의 객체에 수백만 개 버전이 필요한 사용 사례가 있는 경우, AWS Support 팀에 문의하여 보다 나은 솔루션을 찾는 데 도움을 받으세요.

모범 사례

버전 관리 관련 성능 저하 문제를 방지하려면 다음과 같은 모범 사례를 사용하는 것이 좋습니다.

- 객체의 이전 버전을 만료시키는 수명 주기 규칙을 활성화합니다. 예를 들어, 객체가 이전 버전이 된 지 30일이 지나면 비최신 버전을 만료시키는 수명 주기 규칙을 만들 수 있습니다. 모든 버전을 삭제하지 않고 싶다면 비최신 버전 여러 개를 보존할 수도 있습니다. 자세한 내용은 S3 수명 주기 구성 설정을 참조하세요.
- 버킷에 관련된 데이터 객체가 없는 경우 만료된 객체 삭제 마커를 삭제하는 수명 주기 규칙을 활성화합니다. 자세한 내용은 만료된 객체 삭제 마커 제거를 참조하세요.

추가 Amazon S3 성능 최적화 모범 사례는 모범 사례 설계 패턴을 참조하세요.

Object Lock으로 객체 잠금

S3 객체 잠금을 사용하면 고정된 시간 동안 또는 무기한으로 Amazon S3 객체의 삭제 또는 덮어쓰기를 방지하는 데 도움이 됩니다. 객체 잠금은 WORM(Write Once Read Many) 모델을 사용하여 객체를

저장합니다. 객체 잠금을 사용하면 WORM 스토리지가 필요한 규제 요구 사항을 충족하거나 객체 변경 또는 삭제에 대한 보호 계층을 추가하는 데 도움이 됩니다.

> **Note**
>
> S3 객체 잠금은 SEC 17a-4, CFTC 및 FINRA 규정의 적용을 받는 환경에서 Cohasset Associates가 평가했습니다. 객체 잠금과 이러한 규정의 관계에 대한 자세한 내용은 Cohasset Associates Compliance Assessment를 참조하세요.

객체 잠금은 보관 기간 및 법적 보존 등 객체 보관을 관리하는 2가지 방법을 제공합니다. 객체 버전에는 보존 기간과 법적 보존 또는 둘 다를 지정할 수 있습니다.

- 보존 기간 - 보존 기간은 객체 버전이 잠겨 있는 기간을 지정합니다. 개별 객체에 대해 고유한 보존 기간을 설정할 수 있습니다. 또한 S3 버킷에 기본 보존 기간을 설정할 수 있습니다. 버킷 정책의 `s3:object-lock-remaining-retention-days` 조건 키를 사용하여 최소 및 최대 허용 보존 기간을 제한할 수도 있습니다. 이를 통해 보존 기간의 범위를 설정하고 이 범위보다 짧거나 길 수 있는 보존 기간을 제한할 수 있습니다.
- 법적 보존 - 법적 보존은 보존 기간과 동일한 보호를 제공하지만 만료 날짜는 없습니다. 대신, 명시적으로 제거할 때까지 법적 보존이 유지됩니다. 법적 보존은 보존 기간과 별개이며 개별 객체 버전에 적용됩니다.

객체 잠금은 S3 버전 관리가 활성화된 버킷에서만 작동합니다. 객체 버전을 잠그면 Amazon S3는 해당 객체 버전의 메타데이터에 잠금 정보를 저장합니다. 객체에 보존 기간 또는 법적 보존을 설정하면 요청에 지정된 버전만 보호됩니다. 보존 기간과 법적 보존은 객체의 새로운 버전을 생성하거나 객체에 추가할 마커를 삭제하는 것을 차단하지 않습니다. S3 버전 관리에 대한 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하세요.

객체 키 이름이 동일한 기존의 보호된 객체가 있는 버킷에 객체를 넣으면 Amazon S3는 해당 객체의 새 버전을 생성합니다. 기존의 보호된 버전의 객체는 보관 구성에 따라 잠긴 상태로 유지됩니다.

S3 객체 잠금 작동 방식

주제

- 보관 기간
- 보존 모드
- 법적 보존

- S3 객체 잠금 사용의 모범 사례
- 필수 권한

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. 보존 기간이 만료된 후 객체 버전을 덮어쓰거나 삭제할 수 있습니다.

개별 객체 버전 또는 버킷의 속성에 명시적으로 보존 기간을 지정하여 버킷의 모든 객체에 자동으로 적용되도록 할 수 있습니다. 객체 버전에 명시적으로 보관 기간을 적용할 경우 객체 버전에 대해 보관 종료일을 지정합니다. Amazon S3는 객체 버전의 메타데이터에 이 날짜를 저장합니다.

또한 버킷 속성에 보존 기간을 설정할 수 있습니다. 버킷에 보존 기간을 설정할 때 버킷에 있는 모든 객체 버전을 보호해야 하는 기간을 일 또는 연 단위로 지정합니다. 객체를 버킷에 배치할 때 Amazon S3는 객체 버전의 생성 타임스탬프에 지정된 기간을 추가하여 객체 버전에 대한 보존 만료 날짜를 계산합니다. 그러면 객체 버전에 해당 보존 기간으로 개별 잠금을 명시적으로 설정한 것처럼 객체 버전이 보호됩니다.

> **Note**
>
> 버킷에 명시적인 개별 보존 모드 및 기간이 있는 객체 버전에 PUT 작업을 수행하는 경우 객체 버전의 개별 객체 잠금 설정이 모든 버킷 속성 보존 설정보다 우선합니다.

다른 모든 객체 잠금 설정과 마찬가지로 보관 기간은 개별 객체 버전에 적용됩니다. 단일 객체의 서로 다른 버전은 보관 모드 및 기간이 각기 다를 수 있습니다.

예를 들어 보관 기간이 15~30일인 객체가 있고 보관 기간이 60일인 동일한 이름의 객체를 Amazon S3에 PUT한다고 가정합니다. 이 경우 PUT 요청이 성공하면 Amazon S3에서는 보존 기간이 60일인 새 버전의 객체를 생성합니다. 이전 버전은 원래 보관 기간을 유지하고 15일 후에 삭제할 수 있게 됩니다.

객체 버전에 보존 설정을 적용한 후에 보존 기간을 연장할 수 있습니다. 이렇게 하려면 객체 버전에 대해 현재 구성된 객체 잠금 요청보다 이후에 해당되는 보존 만료 날짜를 지정하여 객체 버전에 대한 새 잠금 요청을 제출하세요. Amazon S3는 기존 보관 기간을 보다 연장된 새 기간으로 대체합니다. 객체 보존 기간을 설정할 권한이 있는 사용자는 객체 버전의 보존 기간을 연장할 수 있습니다. 보존 기간을 설정하려면 `s3:PutObjectRetention` 권한이 있어야 합니다.

객체 또는 S3 버킷에 보존 기간을 설정할 때는 규정 준수 또는 거버넌스라는 보존 모드 중 하나를 선택해야 합니다.

보존 모드

S3 객체 잠금은 객체에 각기 다른 보호 수준을 적용하는 두 가지 보존 모드를 제공합니다.

- 규정 준수 모드
- 거버넌스 모드

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 하는 데 도움이 됩니다.

> **Note**
>
> 보존 날짜가 만료되기 전에 규정 준수 모드에서 객체를 삭제하는 유일한 방법은 연결된 AWS 계정을 삭제하는 것입니다.

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 보존 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 보존 설정을 재정의하거나 제거하려면 `s3:BypassGovernanceRetention` 권한을 갖고 있어야 하며 거버넌스 모드 재정의를 요구하는 요청과 함께 요청 헤더로 `x-amz-bypass-governance-retention:true`를 명시적으로 포함해야 합니다.

> **Note**
>
> 기본적으로 Amazon S3 콘솔에는 `x-amz-bypass-governance-retention:true` 헤더가 포함됩니다. 거버넌스 모드로 보호되는 객체를 삭제하려는 경우 `s3:BypassGovernanceRetention` 권한이 있으면 작업이 성공합니다.

법적 보존

객체 잠금을 사용하면 객체 버전에 법적 보존을 적용할 수도 있습니다. 보관 기간과 마찬가지로 법적 보존을 사용하면 객체 버전을 덮어쓰거나 삭제할 수 없습니다. 그러나 법적 보존에는 고정된 기간이 없으며, 제거될 때까지 유효합니다. 법적 보존은 `s3:PutObjectLegalHold` 권한을 가진 사용자가 자유롭게 배치하고 제거할 수 있습니다.

법적 보존은 보관 기간과 독립적입니다. 객체 버전에 법적 보존을 설정해도 해당 객체 버전의 보관 모드 또는 보관 기간에는 영향을 주지 않습니다.

예를 들어, 객체 버전을 보존 기간으로도 보호하면서 객체 버전에 법적 보존을 적용한다고 가정해 봅시다. 보관 기간이 만료되면 객체의 WORM 보호가 손실되지 않습니다. 법적 보존은 권한 있는 사용자가 명시적으로 법적 보존을 제거할 때까지 객체를 계속 보호합니다. 마찬가지로 객체 버전의 보관 기간이 유효한 상태에서 법적 보존을 제거하면 보관 기간이 만료될 때까지 객체 버전이 보호됩니다.

S3 객체 잠금 사용의 모범 사례

미리 정의된 보존 기간 동안 대부분의 사용자가 객체를 삭제하지 못하도록 보호하는 동시에 특별한 권한이 있는 일부 사용자가 보존 설정을 변경하거나 객체를 삭제할 수 있도록 유연성을 부여하려면 거버넌스 모드를 사용하는 것이 좋습니다.

미리 정의된 보존 기간 동안 AWS 계정의 루트 사용자를 비롯한 어떤 사용자도 객체를 삭제할 수 없게 하려면 규정 준수 모드를 사용하는 것이 좋습니다. 규정 준수 데이터를 저장해야 하는 경우 이 모드를 사용할 수 있습니다.

객체를 변경 불가능한 상태로 유지하려는 기간이 확실하지 않은 경우 법적 보존을 사용할 수 있습니다. 데이터에 외부 감사가 예정되어 있고 감사가 완료될 때까지 객체를 변경할 수 없는 상태로 유지하고 싶기 때문일 수 있습니다. 또는 프로젝트가 완료될 때까지 변경 불가능으로 유지하려는 데이터 세트를 활용하는 진행 중인 프로젝트가 있을 수도 있습니다.

필수 권한

객체 잠금 작업에는 특정 권한이 필요합니다. 시도하려는 정확한 작업에 따라 다음 권한 중 하나가 필요할 수 있습니다.

- `s3:BypassGovernanceRetention`
- `s3:GetBucketObjectLockConfiguration`
- `s3:GetObjectLegalHold`
- `s3:GetObjectRetention`

- `s3:PutBucketObjectLockConfiguration`
- `s3:PutObjectLegalHold`
- `s3:PutObjectRetention`

설명이 포함된 Amazon S3 권한의 전체 목록은 서비스 승인 참조에서 Amazon S3에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.

권한과 함께 조건을 사용하는 방법에 대한 자세한 내용은 조건 키를 사용한 버킷 정책 예시 단원을 참조하십시오.

객체 잠금 고려 사항

Amazon S3 객체 잠금을 사용하면 고정된 시간 동안 또는 무기한으로 객체의 삭제 또는 덮어쓰기를 방지하는 데 도움이 됩니다.

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하여 객체 잠금 정보를 보거나 설정할 수 있습니다. S3 객체 잠금 기능에 대한 일반적인 내용은 Object Lock으로 객체 잠금 섹션을 참조하세요.

> **Important**
>
> - 버킷에 객체 잠금이 활성화하고 나면 해당 버킷에서 객체 잠금을 비활성화하거나 버전 관리를 일시 중지할 수 없습니다.
> - 객체 잠금이 설정된 S3 버킷은 서버 액세스 로그의 대상 버킷으로 사용할 수 없습니다. 자세한 내용은 the section called “서버 액세스 로깅” 단원을 참조하십시오.

주제

- 잠금 정보를 볼 수 있는 권한
- 거버넌스 모드 무시
- S3 복제에 객체 잠금 사용
- 암호화에 Object Lock 사용
- Amazon S3 인벤토리에서 객체 잠금 사용

- 객체 잠금을 통한 S3 수명 주기 정책 관리
- 객체 잠금으로 삭제 마커 관리
- 객체 잠금과 함께 S3 Storage Lens 사용
- 객체 잠금이 활성화된 버킷에 객체 업로드
- 이벤트 및 알림 구성
- 버킷 정책을 사용하여 보존 기간에 제한 설정

잠금 정보를 볼 수 있는 권한

HeadObject 또는 GetObject 작업을 사용하여 프로그래밍 방식으로 Amazon S3 객체 버전의 객체 잠금 상태를 볼 수 있습니다. 두 작업 모두 지정된 객체 버전에 대한 보존 모드, 보존 만료 날짜 및 법적 보존 상태를 반환합니다. 또한 S3 Inventory를 사용하여 S3 버킷의 여러 객체에 대한 객체 잠금 상태를 볼 수 있습니다.

객체 버전의 보관 모드 및 보관 기간을 보려면 `s3:GetObjectRetention` 권한이 있어야 합니다. 객체 버전의 법적 보존 상태를 보려면 `s3:GetObjectLegalHold` 권한이 있어야 합니다. 버킷의 기본 보존 구성을 보려면 `s3:GetBucketObjectLockConfiguration` 권한이 있어야 합니다. S3 객체 잠금이 활성화되지 않은 버킷에 대해 객체 잠금 구성을 요청하면 Amazon S3가 오류를 반환합니다.

거버넌스 모드 무시

`s3:BypassGovernanceRetention` 권한이 있는 경우, 거버넌스 모드에서 잠긴 객체 버전이 마치 보호되지 않은 것처럼 작업을 수행할 수 있습니다. 이러한 작업에는 객체 버전 삭제, 보존 기간 단축 또는 빈 파라미터와 함께 새로운 `PutObjectRetention` 요청을 배치하여 객체 잠금 보존 기간을 제거하는 작업이 포함됩니다.

거버넌스 모드를 무시하려면 요청에 이 모드를 무시할 것인지 명시해야 합니다. 이를 위해 `PutObjectRetention` API 요청에 `x-amz-bypass-governance-retention:true` 헤더를 포함하거나 AWS CLI 또는 AWS SDK를 통해 만든 요청과 동등한 파라미터를 사용합니다. `s3:BypassGovernanceRetention` 권한이 있는 경우, S3 콘솔은 콘솔을 통해 만든 요청에 이 헤더를 자동으로 적용합니다.

> **Note**
>
> 거버넌스 모드 무시는 객체 버전의 법적 보존 상태에 영향을 주지 않습니다. 객체 버전에 법적 보존이 활성화된 경우, 법적 보존은 유지되고, 객체 버전을 덮어쓰거나 삭제하는 요청을 차단합니다.

S3 복제에 객체 잠금 사용

S3 복제에 객체 잠금을 사용하여 잠긴 객체와 보존 메타데이터를 서로 다른 S3 버킷에 비동기식으로 자동 복사할 수 있습니다. 즉, 복제된 객체의 경우 Amazon S3는 소스 버킷의 객체 잠금 구성을 사용합니다. 다시 말해, 소스 버킷에 객체 잠금이 활성화되어 있는 경우, 대상 버킷에도 객체 잠금이 활성화되어 있어야 합니다. 객체가 대상 버킷(S3 복제 외부)에 직접 업로드되는 경우 대상 버킷에 설정된 객체 잠금을 사용합니다. 복제를 사용하면 소스 버킷의 객체가 하나 이상의 대상 버킷으로 복제됩니다.

객체 잠금이 활성화된 버킷에서 복제를 설정하려면 S3 콘솔, AWS CLI, Amazon S3 REST API 또는 AWS SDK를 사용할 수 있습니다.

> **Note**
>
> 복제에 객체 잠금을 사용하려면 복제 설정에 사용하는 AWS Identity and Access Management(IAM) 역할에서 소스 S3 버킷에 두 가지 권한을 추가로 부여해야 합니다. 두 가지 추가 권한은 `s3:GetObjectRetention`과 `s3:GetObjectLegalHold`입니다. 역할에 `s3:Get*` 권한 문이 있으면 해당 문이 요구 사항을 충족합니다. 자세한 내용은 라이브 복제에 대한 권한 설정 단원을 참조하십시오.
> S3 복제에 대한 일반적인 정보는 리전 내 및 리전 간 객체 복제 섹션을 참조하세요.
> S3 복제 설정의 예는 라이브 복제 구성 예제 섹션을 참조하세요.

암호화에 Object Lock 사용

Amazon S3는 기본적으로 모든 새 객체를 암호화합니다. 암호화된 객체에 Object Lock을 사용할 수 있습니다. 자세한 내용은 암호화로 데이터 보호 단원을 참조하십시오.

Object Lock을 사용하면 Amazon S3 객체의 삭제 또는 덮어쓰기를 방지하는 데 도움이 되지만, 삭제 중인 암호화 키 또는 암호화 키에 대한 액세스 권한 손실을 보호하지 못합니다. 예를 들어, AWS KMS 서버 측 암호화로 객체를 암호화하고 AWS KMS 키를 삭제하면 객체를 읽을 수 없게 될 수 있습니다.

Amazon S3 인벤토리에서 객체 잠금 사용

정해진 일정으로 S3 버킷의 객체 목록을 생성하도록 Amazon S3 인벤토리를 구성할 수 있습니다. 객체에 대한 다음 객체 잠금 메타데이터를 포함하도록 Amazon S3 인벤토리를 구성할 수 있습니다.

- 보존 만료 날짜
- 보존 모드
- 법적 보존 상태

자세한 내용은 S3 Inventory를 사용한 데이터 카탈로그화 및 분석 단원을 참조하십시오.

객체 잠금을 통한 S3 수명 주기 정책 관리

객체 수명 주기 관리 구성은 삭제 마커 배치를 포함하여 보호된 객체에서 계속 정상적으로 작동합니다. 하지만 객체의 잠긴 버전은 S3 수명 주기 만료 정책에 따라 삭제할 수 없습니다. 객체 잠금은 객체가 상주하는 스토리지 클래스와 관계없이 그리고 스토리지 클래스 간 S3 수명 주기 전환 전반에 걸쳐 유지됩니다.

객체 수명 주기 관리에 대한 자세한 내용은 객체 수명 주기 관리 단원을 참조하십시오.

객체 잠금으로 삭제 마커 관리

보호된 객체 버전은 삭제할 수 없지만 해당 객체에 대한 삭제 마커를 생성할 수 있습니다. 객체에 삭제 마커를 배치해도 객체 또는 해당 객체 버전은 삭제되지 않습니다. 그러나 대부분의 경우 객체가 삭제된 것처럼 Amazon S3가 동작을 수행하게 됩니다. 자세한 내용은 삭제 마커를 통한 작업 섹션을 참조하세요.

> **Note**
>
> 기본 객체에 설정된 보관 기간이나 법적 보존과 상관없이 삭제 마커는 WORM으로 보호되지 않습니다.

객체 잠금과 함께 S3 Storage Lens 사용

객체 잠금 지원 스토리지 바이트 및 객체 수에 대한 지표를 보려면 Amazon S3 스토리지 렌즈를 사용하면 됩니다. S3 스토리지 렌즈는 조직 전반에서 객체 스토리지 사용 및 활동에 대한 가시성을 확보하는 데 사용할 수 있는 클라우드 스토리지 분석 기능입니다.

자세한 내용은 S3 Storage Lens를 사용하여 데이터 보호 단원을 참조하십시오.

전체 지표 목록은 Amazon S3 스토리지 렌즈 지표 용어집 섹션을 참조하세요.

객체 잠금이 활성화된 버킷에 객체 업로드

`Content-MD5` 또는 `x-amz-sdk-checksum-algorithm` 헤더는 Object Lock을 사용하여 보존 기간이 구성된 객체를 업로드하는 모든 요청에 필요합니다. 이러한 헤더는 업로드 중에 객체의 무결성을 확인하는 방법입니다.

Amazon S3 콘솔로 객체를 업로드할 때 S3는 자동으로 `Content-MD5` 헤더를 추가합니다. 선택적으로 콘솔을 통해 `x-amz-sdk-checksum-algorithm` 헤더로 추가 체크섬 함수 및 체크섬 값을 지정할 수 있습니다. PutObject API를 사용하는 경우 `Content-MD5` 헤더, `x-amz-sdk-checksum-algorithm` 헤더 또는 둘 다 지정하여 Object Lock 보존 기간을 구성해야 합니다.

자세한 내용은 Amazon S3에서 객체 무결성 확인 단원을 참조하십시오.

이벤트 및 알림 구성

Amazon S3 이벤트 알림을 사용하면 AWS CloudTrail을 사용하여 객체 잠금 구성 및 데이터에 대한 액세스 및 변경 사항을 추적할 수 있습니다. CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 AWS CloudTrail이란 무엇입니까?를 참조하세요.

Amazon CloudWatch를 사용하여 이 데이터를 기반으로 알림을 생성할 수도 있습니다. CloudWatch에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch란 무엇인가요?를 참조하세요.

버킷 정책을 사용하여 보존 기간에 제한 설정

버킷 정책을 사용하여 버킷의 최소 및 최대 허용 보존 기간을 설정할 수 있습니다. 최대 보존 기간은 100년입니다.

다음 예에서는 `s3:object-lock-remaining-retention-days` 조건 키를 사용하여 최대 보관 기간을 10일로 설정하는 버킷 정책을 보여 줍니다.

```
{
    "Version": "2012-10-17",
    "Id": "SetRetentionLimits",
    "Statement": [
        {
            "Sid": "SetRetentionPeriod",
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "s3:PutObjectRetention"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "NumericGreaterThan": {
                    "s3:object-lock-remaining-retention-days": "10"
                }
            }
```

```
        }
    ]
}
```

> **Note**
>
> 버킷이 복제 구성의 대상 버킷인 경우 복제를 사용하여 생성된 객체 복제본에 대해 최소 및 최대 허용 보존 기간을 설정할 수 있습니다. 이렇게 하려면 버킷 정책에서 `s3:ReplicateObject` 작업을 허용해야 합니다. 복제 권한에 대한 자세한 내용은 the section called “권한 설정” 섹션을 참조하세요.

버킷 정책에 대한 자세한 내용은 다음 주제를 참조하세요.

- 서비스 승인 참조의 Amazon S3에 사용되는 작업, 리소스 및 조건 키

  S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.
- 객체 작업
- 조건 키를 사용한 버킷 정책 예시

S3 객체 잠금 구성

Amazon S3 Object Lock을 사용하면 Write Once Read Many(WORM) 모델을 사용하여 객체를 Amazon S3 범용 버킷에 저장할 수 있습니다. S3 객체 잠금을 사용하면 일정한 시간 동안 또는 무기한으로 객체를 삭제하거나 덮어쓰지 않도록 할 수 있습니다. 객체 잠금 기능에 대한 일반적인 내용은 Object Lock으로 객체 잠금 섹션을 참조하세요.

객체를 잠그기 전에 범용 버킷에 S3 버전 관리와 Object Lock을 활성화해야 합니다. 그 후에 보존 기간, 법적 보존 또는 둘 다를 설정할 수 있습니다.

객체 잠금을 사용하려면 특정 권한이 있어야 합니다. 다양한 객체 잠금 작업과 관련된 권한 목록은 the section called “필수 권한” 섹션을 참조하세요.

> **Important**
>
> - 버킷에 객체 잠금이 활성화하고 나면 해당 버킷에서 객체 잠금을 비활성화하거나 버전 관리를 일시 중지할 수 없습니다.

- 객체 잠금이 설정된 S3 버킷은 서버 액세스 로그의 대상 버킷으로 사용할 수 없습니다. 자세한 내용은 the section called “서버 액세스 로깅” 단원을 참조하십시오.

주제

- 새로운 S3 범용 버킷을 생성할 때 Object Lock을 활성화합니다.
- 기존 Amazon S3 버킷에 객체 잠금 활성화
- S3 객체에 법적 보존 설정 또는 수정
- S3 객체의 보존 기간 설정 또는 수정
- S3 버킷의 기본 보존 기간 설정 또는 수정

새로운 S3 범용 버킷을 생성할 때 Object Lock을 활성화합니다.

새 S3 범용 버킷을 생성할 때 Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하여 Object Lock을 활성화할 수 있습니다.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 만들기를 선택합니다.

   버킷 만들기 페이지가 열립니다.
4. [Bucket Name]에서 버킷 이름을 입력합니다.

   > **Note**
   >
   > 버킷을 생성한 후에는 이름을 변경할 수 없습니다. 버킷 이름 지정에 대한 자세한 내용은 범용 버킷 이름 지정 규칙 섹션을 참조하십시오.
5. 리전(Region)에서 버킷이 속할 AWS 리전을 선택합니다.
6. 객체 소유권에서 액세스 제어 목록(ACL)을 비활성화 또는 활성화하고 버킷에 업로드된 객체의 소유권을 제어합니다.
7. 퍼블릭 액세스 차단을 위한 버킷 설정에서 버킷에 적용할 퍼블릭 액세스 차단 설정을 선택합니다.
8. 버킷 버전 관리에서 활성화됨을 선택합니다.

객체 잠금은 버전이 지정된 버킷에서만 작동합니다.

9. (선택 사항) Tags(태그)에서 버킷에 태그를 추가할 수 있습니다. 태그는 스토리지를 분류하고 비용을 할당하는 데 사용되는 키-값 쌍입니다.
10. 고급 설정에서 객체 잠금을 찾아 활성화를 선택합니다.

    객체 잠금을 활성화하면 이 버킷의 객체가 영구적으로 잠기도록 허용한다는 점을 숙지해야 합니다.
11. 버킷 생성을 선택합니다.

AWS CLI 사용

다음 `create-bucket` 예시에서는 객체 잠금이 활성화되었으며 이름이 *amzn-s3-demo-bucket1*인 새 S3 버킷을 생성합니다.

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket1 --object-lock-enabled-for-bucket
```

자세한 내용과 예제는 AWS CLI 명령 참조에서 create-bucket를 참조하십시오.

> **Note**
>
> AWS CloudShell을 사용하여 콘솔에서 AWS CLI 명령을 실행할 수 있습니다. AWS CloudShell은 브라우저 기반의 사전 인증된 쉘로, AWS Management Console에서 직접 시작할 수 있습니다. 자세한 내용은 AWS CloudShell 사용 설명서에서 What is CloudShell?을 참조하세요.

REST API 사용

REST API를 사용하여 객체 잠금이 활성화된 새 S3 버킷을 생성할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 CreateBucket를 참조하십시오.

AWS SDK 사용

AWS SDK를 사용하여 새 S3 버킷을 만들 때 Object Lock을 활성화하는 방법에 대한 예제는 Amazon S3 API 참조의 코드 예제를 참조하세요.

AWS SDK를 사용하여 현재 Object Lock 구성을 가져오는 방법에 대한 예제는 Amazon S3 API 참조의 코드 예제를 참조하세요.

AWS SDK를 사용하여 다양한 Object Lock 기능을 보여주는 대화형 시나리오는 Amazon S3 API 참조의 코드 예제를 참조하세요.

다양한 AWS SDK 사용에 대한 일반적인 정보는 Amazon S3 API 참조의 AWS SDK를 사용하여 Amazon S3에서 개발을 참조하세요.

기존 Amazon S3 버킷에 객체 잠금 활성화

Amazon S3 콘솔, AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용하여 기존 S3 버킷에 객체 잠금을 활성화할 수 있습니다.

S3 콘솔 사용

> **Note**
>
> 객체 잠금은 버전이 지정된 버킷에서만 작동합니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 목록에서 객체 잠금을 활성화할 버킷의 이름을 선택합니다.
4. 속성(Properties) 탭을 선택합니다.
5. 속성에서 객체 잠금 섹션까지 아래로 스크롤한 다음 편집을 선택합니다.
6. 객체 잠금에서 활성화를 선택합니다.

   객체 잠금을 활성화하면 이 버킷의 객체가 영구적으로 잠기도록 허용한다는 점을 숙지해야 합니다.
7. Save changes(변경 사항 저장)를 선택합니다.

AWS CLI 사용

다음 `put-object-lock-configuration` 예시 명령은 이름이 *amzn-s3-demo-bucket1*인 버킷에 50일의 객체 잠금 보존 기간을 설정합니다.

```
aws s3api put-object-lock-configuration --bucket amzn-s3-demo-bucket1 --object-
lock-configuration='{ "ObjectLockEnabled": "Enabled", "Rule": { "DefaultRetention":
 { "Mode": "COMPLIANCE", "Days": 50 }}}'
```

자세한 내용과 예제는 AWS CLI 명령 참조에서 put-object-lock-configuration를 참조하십시오.

> **Note**
>
> AWS CloudShell을 사용하여 콘솔에서 AWS CLI 명령을 실행할 수 있습니다. AWS CloudShell은 브라우저 기반의 사전 인증된 쉘로, AWS Management Console에서 직접 시작할 수 있습니다. 자세한 내용은 AWS CloudShell 사용 설명서에서 What is CloudShell?을 참조하세요.

REST API 사용

Amazon S3 REST API를 사용하여 기존 S3 버킷에 객체 잠금을 활성화할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 PutObjectLockConfiguration를 참조하십시오.

AWS SDK 사용

AWS SDK를 사용하여 기존 S3 버킷에 대해 Object Lock을 활성화하는 방법에 대한 예제는 Amazon S3 API 참조의 코드 예제를 참조하세요.

AWS SDK를 사용하여 현재 Object Lock 구성을 가져오는 방법에 대한 예제는 Amazon S3 API 참조의 코드 예제를 참조하세요.

AWS SDK를 사용하여 다양한 Object Lock 기능을 보여주는 대화형 시나리오는 Amazon S3 API 참조의 코드 예제를 참조하세요.

다양한 AWS SDK 사용에 대한 일반적인 정보는 Amazon S3 API 참조의 AWS SDK를 사용하여 Amazon S3에서 개발을 참조하세요.

S3 객체에 법적 보존 설정 또는 수정

Amazon S3 콘솔, AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용하여 S3 객체에 법적 보존을 설정하거나 제거할 수 있습니다.

> **Important**
>
> - 객체에 법적 보존을 설정하려는 경우 해당 객체의 버킷에 이미 객체 잠금이 활성화되어 있어야 합니다.
> - 버킷에 명시적인 개별 보존 모드 및 기간이 있는 객체 버전에 PUT 작업을 수행하는 경우 객체 버전의 개별 객체 잠금 설정이 모든 버킷 속성 보존 설정보다 우선합니다.

자세한 내용은 the section called “법적 보존” 단원을 참조하십시오.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 목록에서 법적 보존을 설정하거나 수정할 객체가 들어 있는 버킷 이름을 선택합니다.
4. 객체 목록에서 법적 보존을 설정하거나 수정하려는 객체를 선택합니다.
5. 객체 속성 페이지에서 객체 잠금 법적 보존 섹션을 찾아 편집을 선택합니다.
6. 법적 보존을 설정하려면 활성화를 선택하고 법적 보존을 제거하려면 비활성화를 선택합니다.
7. Save changes(변경 사항 저장)를 선택합니다.

AWS CLI 사용

다음 `put-object-legal-hold` 예시에서는 이름이 *amzn-s3-demo-bucket1*인 버킷의 *my-image.fs* 객체에 법적 보존을 설정합니다.

```
aws s3api put-object-legal-hold --bucket amzn-s3-demo-bucket1 --key my-image.fs --
legal-hold="Status=ON"
```

다음 `put-object-legal-hold` 예시에서는 이름이 *amzn-s3-demo-bucket1*인 버킷의 *my-image.fs* 객체에서 법적 보존을 제거합니다.

```
aws s3api put-object-legal-hold --bucket amzn-s3-demo-bucket1 --key my-image.fs --
legal-hold="Status=OFF"
```

자세한 내용과 예제는 AWS CLI 명령 참조에서 put-object-legal-hold를 참조하십시오.

> **Note**
>
> AWS CloudShell을 사용하여 콘솔에서 AWS CLI 명령을 실행할 수 있습니다. AWS CloudShell은 브라우저 기반의 사전 인증된 쉘로, AWS Management Console에서 직접 시작할 수 있습니다. 자세한 내용은 AWS CloudShell 사용 설명서에서 What is CloudShell?을 참조하세요.

REST API 사용

REST API를 사용하여 객체에 법적 보존을 설정하거나 수정할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 PutObjectLegalHold를 참조하십시오.

AWS SDK 사용

AWS SDK를 사용하여 객체에 법적 보존을 설정하는 방법에 대한 예제는 Amazon S3 API 참조의 코드 예제를 참조하세요.

AWS SDK를 사용하여 현재 법적 보존 상태를 가져오는 방법에 대한 예제는 Amazon S3 API 참조의 코드 예제를 참조하세요.

AWS SDK를 사용하여 다양한 Object Lock 기능을 보여주는 대화형 시나리오는 Amazon S3 API 참조의 코드 예제를 참조하세요.

다양한 AWS SDK 사용에 대한 일반적인 정보는 Amazon S3 API 참조의 AWS SDK를 사용하여 Amazon S3에서 개발을 참조하세요.

S3 객체의 보존 기간 설정 또는 수정

Amazon S3 콘솔, AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용하여 S3 객체에 보존 기간을 설정하거나 수정할 수 있습니다.

> **Important**
>
> - 객체에 보존 기간을 설정하려는 경우 해당 객체의 버킷에 이미 객체 잠금이 활성화되어 있어야 합니다.
> - 버킷에 명시적인 개별 보존 모드 및 기간이 있는 객체 버전에 PUT 작업을 수행하는 경우 객체 버전의 개별 객체 잠금 설정이 모든 버킷 속성 보존 설정보다 우선합니다.
> - 보존 날짜가 만료되기 전에 규정 준수 모드에서 객체를 삭제하는 유일한 방법은 연결된 AWS 계정을 삭제하는 것입니다.

자세한 내용은 보관 기간 단원을 참조하십시오.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.

2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 목록에서 보존 기간을 설정하거나 수정할 객체가 들어 있는 버킷 이름을 선택합니다.
4. 객체 목록에서 보존 기간을 설정하거나 수정하려는 객체를 선택합니다.
5. 객체 속성 페이지에서 객체 잠금 보존 섹션을 찾아 편집을 선택합니다.
6. 보존에서 보존 기간을 설정하려면 활성화를 선택하고 보존 기간을 제거하려면 비활성화를 선택합니다.
7. 활성화를 선택한 경우 보존 모드에서 거버넌스 모드와 규정 준수 모드 중 하나를 선택합니다. 자세한 내용은 보존 모드 단원을 참조하십시오.
8. 보존 만료 날짜에서 보존 기간이 종료될 날짜를 선택합니다. 이 기간 동안 객체는 WORM으로 보호되며 덮어쓰거나 삭제할 수 없습니다. 자세한 내용은 보관 기간 단원을 참조하십시오.
9. [변경 사항 저장(Save changes)]을 선택합니다.

AWS CLI 사용

다음 `put-object-retention` 예시에서는 이름이 *amzn-s3-demo-bucket1*인 버킷의 *my-image.fs* 객체에 보존 기간을 2025년 1월 1일까지로 설정합니다.

```
aws s3api put-object-retention --bucket amzn-s3-demo-bucket1 --key my-image.fs --
retention='{ "Mode": "GOVERNANCE", "RetainUntilDate": "2025-01-01T00:00:00" }'
```

자세한 내용과 예제는 AWS CLI 명령 참조에서 put-object-retention를 참조하십시오.

> **Note**
>
> AWS CloudShell을 사용하여 콘솔에서 AWS CLI 명령을 실행할 수 있습니다. AWS CloudShell은 브라우저 기반의 사전 인증된 쉘로, AWS Management Console에서 직접 시작할 수 있습니다. 자세한 내용은 AWS CloudShell 사용 설명서에서 What is CloudShell?을 참조하세요.

REST API 사용

REST API를 사용하여 객체에 보존 기간을 설정할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 PutObjectRetention를 참조하십시오.

AWS SDK 사용

AWS SDK를 사용하여 객체에 보존 기간을 설정하는 방법에 대한 예제는 Amazon S3 API 참조의 코드 예제를 참조하세요.

AWS SDK를 사용하여 객체의 보존 기간을 가져오는 방법에 대한 예제는 Amazon S3 API 참조의 코드 예제를 참조하세요.

AWS SDK를 사용하여 다양한 Object Lock 기능을 보여주는 대화형 시나리오는 Amazon S3 API 참조의 코드 예제를 참조하세요.

다양한 AWS SDK 사용에 대한 일반적인 정보는 Amazon S3 API 참조의 AWS SDK를 사용하여 Amazon S3에서 개발을 참조하세요.

S3 버킷의 기본 보존 기간 설정 또는 수정

Amazon S3 콘솔, AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용하여 S3 버킷에 기본 보존 기간을 설정하거나 수정할 수 있습니다. 버킷에 있는 모든 객체 버전을 보호해야 하는 기간을 일 또는 연 단위로 지정합니다.

Important

- 버킷에 기본 보존 기간을 설정하려는 경우 버킷에 이미 객체 잠금이 활성화되어 있어야 합니다.
- 버킷에 명시적인 개별 보존 모드 및 기간이 있는 객체 버전에 PUT 작업을 수행하는 경우 객체 버전의 개별 객체 잠금 설정이 모든 버킷 속성 보존 설정보다 우선합니다.
- 보존 날짜가 만료되기 전에 규정 준수 모드에서 객체를 삭제하는 유일한 방법은 연결된 AWS 계정을 삭제하는 것입니다.

자세한 내용은 보관 기간 단원을 참조하십시오.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 목록에서 기본 보존 기간을 설정하거나 수정할 버킷 이름을 선택합니다.
4. 속성(Properties) 탭을 선택합니다.
5. 속성에서 객체 잠금 섹션까지 아래로 스크롤한 다음 편집을 선택합니다.
6. 기본 보존에서 기본 보존을 설정하려면 활성화를 선택하고 기본 보존을 제거하려면 비활성화를 선택합니다.

7. 활성화를 선택한 경우 보존 모드에서 거버넌스 모드와 규정 준수 모드 중 하나를 선택합니다. 자세한 내용은 보존 모드 단원을 참조하십시오.
8. 기본 보존 기간에서 보존 기간을 지속할 일수 또는 연수를 선택합니다. 이 버킷에 배치된 객체는 이 일수 또는 연수 동안 잠깁니다. 자세한 내용은 보관 기간 단원을 참조하십시오.
9. [변경 사항 저장(Save changes)]을 선택합니다.

AWS CLI 사용

다음 `put-object-lock-configuration` 예시 명령은 규정 준수 모드를 사용하여 이름이 *amzn-s3-demo-bucket1*인 버킷에 50일의 객체 잠금 보존 기간을 설정합니다.

```
aws s3api put-object-lock-configuration --bucket amzn-s3-demo-bucket1 --object-
lock-configuration='{ "ObjectLockEnabled": "Enabled", "Rule": { "DefaultRetention":
 { "Mode": "COMPLIANCE", "Days": 50 }}}'
```

다음 `put-object-lock-configuration` 예시에서는 버킷의 기본 보존 구성을 제거합니다.

```
aws s3api put-object-lock-configuration --bucket amzn-s3-demo-bucket1 --object-lock-
configuration='{ "ObjectLockEnabled": "Enabled"}'
```

자세한 내용과 예제는 AWS CLI 명령 참조에서 put-object-lock-configuration를 참조하십시오.

> **Note**
>
> AWS CloudShell을 사용하여 콘솔에서 AWS CLI 명령을 실행할 수 있습니다. AWS CloudShell은 브라우저 기반의 사전 인증된 쉘로, AWS Management Console에서 직접 시작할 수 있습니다. 자세한 내용은 AWS CloudShell 사용 설명서에서 What is CloudShell?을 참조하세요.

REST API 사용

REST API를 사용하여 기존 S3 버킷에 기본 보존 기간을 설정할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 PutObjectLockConfiguration를 참조하십시오.

AWS SDK 사용

AWS SDK를 사용하여 기존 S3 버킷에 기본 보존 기간을 설정하는 방법에 대한 예제는 Amazon S3 API 참조의 코드 예제를 참조하세요.

AWS SDK를 사용하여 다양한 Object Lock 기능을 보여주는 대화형 시나리오는 Amazon S3 API 참조의 코드 예제를 참조하세요.

다양한 AWS SDK 사용에 대한 일반적인 정보는 Amazon S3 API 참조의 Developing with Amazon S3 using the AWS SDKs를 참조하세요.

Amazon S3 데이터 백업

Amazon S3는 Amazon S3의 데이터를 보호하기 위해 백업 정책을 중앙에서 정의하는 데 사용할 수 있는 완전관리형 정책 기반 서비스인 AWS Backup과 기본적으로 통합됩니다. 백업 정책을 정의하고 Amazon S3 리소스를 정책에 할당하면 AWS Backup이 Amazon S3 백업을 자동으로 생성하고 사용자가 백업 계획에서 지정한 암호화된 백업 볼트에 백업을 안전하게 저장합니다.

Amazon S3에 AWS Backup을 사용할 때 다음 작업을 수행할 수 있습니다.

- 연속 백업 및 정기적 백업을 생성합니다. 연속 백업은 특정 시점으로 복원할 때 유용하며, 정기적 백업은 장기 데이터 보존 요구 사항을 충족하는 데 유용합니다.
- 백업 정책을 중앙에서 구성하여 백업 예약 및 보존을 자동화합니다.
- Amazon S3 데이터의 백업을 지정한 특정 시점으로 복원합니다.

AWS Backup과 함께 S3 버전 관리 및 S3 복제를 사용하여 우발적인 삭제로부터 복구하고 자체 복구 작업을 수행할 수 있습니다.

사전 조건

버킷에서 S3 버전 관리를 활성화해야 AWS Backup에서 백업할 수 있습니다.

> **Note**
>
> 백업할 버전 관리가 활성화된 버킷에 대한 수명 주기 만료 규칙을 설정하는 것이 좋습니다. 수명 주기 만료 기간을 설정하지 않으면 AWS Backup이 Amazon S3 데이터의 모든 버전을 유지하므로 Amazon S3 스토리지 비용이 증가할 수 있습니다.

시작하기

Amazon S3에 AWS Backup의 사용을 시작하려면 AWS Backup 개발자 가이드의 Amazon S3 백업 생성을 참조하세요.

규제 및 제한

제한 사항에 대해 알아보려면 AWS Backup 개발자 가이드의 Amazon S3 백업 생성을 참조하세요.

비용 최적화

Amazon S3는 데이터 수명 주기 전반에 걸쳐 비용을 최적화하는 데 도움이 되는 다양한 기능과 스토리지 클래스를 제공합니다. 스토리지 클래스는 선결제 비용이나 콘텐츠 저장량에 대한 약정 없이 해당 비용으로 다양한 데이터 액세스 수준을 지원하여 비용을 관리할 수 있는 유연성을 제공합니다. 다른 AWS 서비스와 마찬가지로 사용한 만큼만 종량제 방식으로 지불합니다.

Amazon S3 스토리지 클래스는 다양한 액세스 패턴에 맞게 가장 저렴한 스토리지를 제공하도록 특별히 구축되었습니다. 다음이 포함됩니다.

- 자주 액세스하는 데이터의 범용 스토리지를 위한 S3 Standard.
- 단일 가용 영역에서 자주 액세스하는 고성능 데이터를 위한 Amazon S3 Express One Zone.
- 액세스 패턴을 알 수 없거나 액세스 패턴이 달라지는 데이터에 대한 비용을 자동으로 최적화하는 S3 Intelligent-Tiering.
- 수명이 길지만 액세스 빈도가 낮은 데이터를 위한 S3 Standard-IA(S3 Standard-IA) 및 S3 One Zone-IA(S3 One Zone-IA).
- 즉각적인 액세스가 필요한 데이터 아카이브를 위한 S3 Glacier Instant Retrieval.
- 즉각적인 액세스가 필요하지는 않지만 대규모 데이터 세트를 무료로 검색할 수 있는 유연성이 필요한 데이터 아카이브를 위한 S3 Glacier.
- 클라우드에서 가장 저렴한 스토리지 비용으로 장기 아카이브 및 디지털 보존을 위한 S3 Glacier Deep Archive.

언제든지 객체를 가장 비용 효율적인 스토리지 클래스로 이동할 수 있습니다. 또한, Amazon S3는 데이터 수명 주기를 관리하는 기능을 제공합니다. 예를 들어, S3 수명 주기 구성을 사용하여 보다 비용 효율적인 스토리지 클래스로의 객체 이전을 자동화하거나 정의한 규칙에 따라 만료된 객체를 자동으로 삭제할 수 있습니다.

S3 스토리지 클래스 분석, 비용 할당 태그 지정, 청구 및 사용 보고서와 같은 기능을 통해 비용 및 사용 패턴을 분석할 수 있습니다.

주제

- Amazon S3에 결제 및 사용 보고
- Amazon S3 스토리지 클래스 이해 및 관리
- 객체 수명 주기 관리

Amazon S3에 결제 및 사용 보고

Amazon S3를 사용하는 경우 사용자는 선불 요금을 지불하거나 얼마나 많은 콘텐츠를 보유할지 약정하지 않아도 됩니다. 다른 AWS 서비스와 마찬가지로 사용하는 것에 대해 사용한 만큼만 지불합니다.

AWS은(는) Amazon S3에 대해 다음과 같은 보고서를 제공합니다.

- 결제 보고서 - Amazon S3를 포함하여 사용 중인 AWS 서비스의 모든 활동에 대한 개괄적인 보기를 제공하는 여러 보고서입니다. 버킷이 요청자 지불 버킷으로 생성되지 않은 경우 AWS는 항상 S3 버킷 소유자에게 Amazon S3 요금을 청구합니다. 요청자 지불에 대한 자세한 내용은 스토리지 전송 및 사용량에 대한 요청자 지불 범용 버킷 사용 섹션을 참조하세요. 결제 보고서에 대한 자세한 내용은 Amazon S3에 대한 AWS Billing 보고서 섹션을 참조하세요.
- 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 섹션을 참조하세요.

Amazon S3에 대한 결제 및 사용 보고에 관한 내용은 다음 주제를 참조하세요.

주제

- 비용 할당 S3 버킷 태그 사용
- Amazon S3에 대한 AWS Billing 보고서
- Amazon S3에 대한 AWS 사용 보고서
- Amazon S3에 대한 AWS 결제 및 사용 보고서 이해
- Amazon S3 오류 응답에 대한 청구

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 태그를 사용하여 스토리지 분류 섹션을 참조하세요.

비용 할당 보고서에는 계정에 대한 AWS 사용이 제품 범주 및 연결된 계정 사용자별로 나열됩니다. 이 보고서에는 세부 결제 보고서(Amazon S3에 대한 AWS 결제 및 사용 보고서 이해 참조)와 동일한 행 항목 및 태그 키에 대한 추가 열이 포함되어 있습니다.

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

Billing and Cost Management 콘솔에서 개별적으로 두 유형의 태그를 모두 활성화해야만 결제 보고서에 표시가 됩니다. AWS 생성 태그에 대한 자세한 내용은 AWS 생성 비용 할당 태그를 참조하세요.

- 콘솔에서 태그를 생성하려면 S3 범용 버킷에 대한 속성 보기 섹션을 참조하세요.
- Amazon S3 API를 사용하여 태그를 생성하려면 Amazon Simple Storage Service API 참조의 PUT 버킷 태그 지정을 참조하세요.
- AWS CLI를 사용하여 태그를 생성하려면 AWS CLI 명령 참조의 put-bucket-tagging을 참조하세요.
- 태그 활성화에 대한 자세한 내용은 AWS Billing 사용 설명서의 비용 할당 태그 사용을 참조하세요.

사용자 정의 비용 할당 태그

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 사용 설명서의 사용자 정의 태그 제한을 참조하세요. 사용자 정의 태그에 대한 자세한 내용은 AWS Billing 사용 설명서의 사용자 정의 비용 할당 태그를 참조하세요.

S3 버킷 태그

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.

AWS은(는) 태그에 의미론적 의미를 적용하지 않습니다. 태그는 엄격히 문자열로 해석됩니다.

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 또는 Amazon S3 API를 사용하여 태그를 추가, 나열, 편집 또는 삭제할 수 있습니다.

추가 정보

- AWS Billing 사용 설명서의 비용 할당 태그 사용
- Amazon S3에 대한 AWS 결제 및 사용 보고서 이해
- Amazon S3에 대한 AWS Billing 보고서

Amazon S3에 대한 AWS Billing 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 Billing 보고서가 제공되고 있습니다. 결제 보고서 확인 방법에 대한 자세한 내용은 AWS Billing 사용 설명서의 청구서 보기 섹션을 참조하세요.

AWS 사용 내역을 추적하고 계정의 예상 요금을 확인하려면 AWS Cost and Usage Reports를 설정하면 됩니다. 자세한 내용은 AWS 사용 설명서의 What are AWSCost and Usage Reports? 섹션을 참조하세요.

결제 보고서보다 Amazon S3 스토리지 사용에 대해 세부 정보를 제공하는 사용 보고서를 다운로드할 수도 있습니다. 자세한 내용은 Amazon S3에 대한 AWS 사용 보고서 섹션을 참조하세요.

다음 표에는 Amazon S3 사용과 관련된 요금이 나와 있습니다.

요금	설명
스토리지	S3 버킷에서의 객체 저장에 대한 요금이 부과됩니다. 부과되는 요금은 객체의 크기, 객체 저장 기간 및 스토리지 클래스에 따라 다릅니다. Amazon S3는 다음과 같은 스토리지 클래스를 제공합니다. S3 Standard, S3 Express One Zone, S3 Intelligent-Tiering, S3 Standard-IA(Infrequent Access용 IA), S3 One Zone-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, S3 Glacier Deep Archive 또는 Reduced Redundancy Storage(RRS). 스토

요금	설명
	리지 클래스에 대한 자세한 정보는 Amazon S3 스토리지 클래스 이해 및 관리를 참조하십시오. S3 버전 관리를 사용 설정한 경우 유지된 객체의 각 버전에 대해 요금이 부과됩니다. 버전 관리에 대한 자세한 내용은 S3 버전 관리 작동 방식 섹션을 참조하십시오.
범용 버킷	계정에서 생성한 첫 2,000개의 범용 버킷에 대해서는 요금이 청구되지 않습니다. 그러나 첫 2,000개를 초과하여 생성한 각 버킷에는 버킷당 요금이 부과됩니다. 이 요금은 버킷/월별로 청구됩니다. 범용 버킷 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하세요.
모니터링 및 자동화	액세스 패턴을 모니터링하고 S3 Intelligent-Tiering 스토리지 클래스의 액세스 계층 간에 객체를 이동하도록 S3 Intelligent-Tiering에 저장된 객체당 월별 모니터링 및 자동화 요금을 지불합니다.
요청	S3 버킷 및 객체에 대한 요청(예: 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 요금 섹션을 참조하세요.
검색	S3 Standard-IA, S3 One Zone-IA, S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive 스토리지에 저장된 객체 검색에 대한 요금이 부과됩니다.

요금	설명
조기 삭제	최소 스토리지 약정 기간이 지나기 전에 S3 Standard-IA, S3 One Zone-IA, S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지에 저장된 객체를 삭제하는 경우에는 해당 객체에 대한 조기 삭제 요금이 부과됩니다.
스토리지 관리	계정의 버킷에서 사용 설정된 스토리지 관리 기능(Amazon S3 인벤토리, 분석 및 객체 태깅)에 대해 요금이 부과됩니다.
대역폭	다음을 제외하면 Amazon S3의 모든 송수신 대역폭에 대해 요금을 부과됩니다. • 인터넷에서 전송되는 데이터 • Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스로 전송된 데이터(이 경우 인스턴스가 S3 버킷과 동일한 AWS 리전에 있음) • Amazon CloudFront(CloudFront)로 전송된 데이터 또한 Amazon S3 Transfer Acceleration을 사용하여 전송된 데이터에 대해서도 요금이 부과됩니다.

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

Amazon S3에 대한 결제 및 사용 보고서에서 사용되는 코드와 약어에 대한 내용은 Amazon S3에 대한 AWS 결제 및 사용 보고서 이해 섹션을 참조하세요.

추가 정보

- Amazon S3에 대한 AWS 사용 보고서

- 비용 할당 S3 버킷 태그 사용
- AWS Billing 및 비용 관리
- Amazon S3 요금

Amazon S3에 대한 AWS 사용 보고서

사용 보고서를 다운로드하면 사용 데이터를 시간, 일 또는 월 기준으로 집계할 수 있습니다. Amazon S3 사용 보고서에는 사용 유형 및 AWS 리전별로 작업이 나열됩니다. Amazon S3 스토리지 사용에 대한 자세한 보고서를 보려면 동적으로 생성된 AWS 사용 보고서를 다운로드합니다. 포함시킬 사용 유형, 작업 및 기간을 선택할 수 있습니다. 데이터가 축적되는 방식을 선택할 수도 있습니다. 사용 보고서에 대한 자세한 설명은 AWS 데이터 내보내기 사용 설명서의 AWS 사용 보고서를 참조하세요.

Amazon S3 사용 보고서에는 다음 정보가 포함됩니다.

- 서비스 – Amazon S3
- 작업 – 버킷 또는 객체에서 수행되는 작업입니다. Amazon S3 작업에 대한 자세한 설명은 사용 보고서의 추적 작업 섹션을 참조하세요.
- UsageType – 다음 값 중 하나입니다.
  - 스토리지의 유형을 식별하는 코드
  - 요청의 유형을 식별하는 코드
  - 검색의 유형을 식별하는 코드
  - 데이터 전송의 유형을 식별하는 코드입니다.
  - S3 Intelligent-Tiering, S3 Standard-IA, S3 One Zone-Infrequent Access(S3 One Zone-IA), S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지에서 초기 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 섹션을 참조하세요.
- 리소스 – 나열된 사용과 관련된 버킷 또는 테이블의 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. 데이터의 일반적인 측정 단위는 기가바이트(GB)입니다. 그러나 서비스 및 보고서에 따라 TB(테라바이트)가 대신 나타날 수 있습니다.
  - 지정된 기간 동안 요청 수

- 전송된 데이터의 양
- 지정된 시간에 저장된 데이터의 양
- S3 Standard-IA, S3 One Zone-IA, S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지에서 복원과 관련된 데이터의 양

> **Tip**
>
> Amazon S3가 객체에 대해 수신하는 모든 요청에 대한 세부 정보를 보려면 버킷에 대한 서버 액세스 로깅을 활성화하세요. 자세한 내용은 서버 액세스 로깅을 사용한 요청 로깅 섹션을 참조하세요.

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

자세한 내용은 Amazon S3에 대한 AWS 결제 및 사용 보고서 이해 섹션을 참조하세요.

AWS 사용 보고서 다운로드

사용 보고서를 XML 또는 CSV 파일로 다운로드할 수 있습니다.

사용 보고서 다운로드

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 제목 표시줄에서 사용자 이름 또는 계정 ID를 선택한 다음 결제 및 비용 관리를 선택합니다.
3. 탐색 창에서 레거시 페이지를 선택하고 비용 및 사용 보고서를 선택합니다.
4. AWS 사용 보고서에서 사용 보고서 생성을 선택합니다.
5. 사용 보고서 다운로드 페이지에서 다음 설정을 선택합니다.
   - 서비스 - Amazon Simple Storage Service를 선택합니다.

- 사용 유형 – Amazon S3 사용 유형에 관한 자세한 설명은 Amazon S3에 대한 AWS 결제 및 사용 보고서 이해 섹션을 참조하세요.
- 작업 – Amazon S3 작업에 관한 자세한 설명은 사용 보고서의 추적 작업 섹션을 참조하세요.
- 기간 – 보고서에서 평가하고자 하는 기간입니다.
- 보고서 세부 수준 – 보고서에 시간, 일 또는 월 기준으로 소계를 포함할지 여부를 선택합니다.

6. 다운로드를 선택하고 다운로드 형식(XML 보고서 또는 CSV 보고서)을 선택한 다음 메시지에 따라 보고서를 열거나 저장합니다.

추가 정보

- Amazon S3에 대한 AWS 결제 및 사용 보고서 이해
- Amazon S3에 대한 AWS Billing 보고서

Amazon S3에 대한 AWS 결제 및 사용 보고서 이해

Amazon S3 결제 및 사용 보고서는 코드와 약어를 사용합니다. 다음 표의 사용 유형에 대해 *region*, *region1* 및 *region2*를 이 목록의 약어로 바꿉니다.

- APE1: 아시아 태평양(홍콩)
- APN1: 아시아 태평양(도쿄)
- APN2: 아시아 태평양(서울)
- APN3: 아시아 태평양(오사카)
- APS1: 아시아 태평양(싱가포르)
- APS2: 아시아 태평양(시드니)
- APS3: 아시아 태평양(뭄바이)
- APS4: 아시아 태평양(자카르타)
- APS5: 아시아 태평양(하이데라바드)
- APS6: 아시아 태평양(멜버른)
- APS7: 아시아 태평양(말레이시아)
- APS9: 아시아 태평양(태국)
- CAN1: 캐나다(중부)
- CAN2: 캐나다 서부(캘거리)

- CNN1: 중국(베이징)
- CNW1: 중국(닝샤)
- AFS1: 아프리카(케이프타운)
- EUC2: 유럽(취리히)
- EUN1: 유럽(스톡홀름)
- EUS2: 유럽(스페인)
- EUC1: 유럽(프랑크푸르트)
- EU: 유럽(아일랜드)
- EUS1: 유럽(밀라노)
- EUW2: 유럽(런던)
- EUW3: 유럽(파리)
- ILC1: 이스라엘(텔아비브)
- MEC1: 중동(UAE)
- MES1: 중동(바레인)
- MXC1: 멕시코(중부)
- SAE1: 남아메리카(상파울루)
- UGW1: AWS GovCloud(미국 서부)
- UGE1: AWS GovCloud(미국 동부)
- USE1(또는 접두사 없음): 미국 동부(버지니아 북부)
- USE2: 미국 동부(오하이오)
- USW1: 미국 서부(캘리포니아 북부)
- USW2: 미국 서부(오레곤)

다음 표의 S3 다중 리전 액세스 포인트 사용 유형에 대해 *regiongroup1* 및 *regiongroup2*를 이 목록의 약어로 바꿉니다.

- AP: 아시아 태평양
- AU: 오스트레일리아
- EU: 유럽
- IN: 인도
- NA: 북미

- SA: 남미

리전 그룹은 여러 AWS 리전로 구성된 지리적 그룹입니다. 자세한 내용은 리전 및 가용 영역을 참조하세요. AWS 리전별 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하세요.

다음 표의 첫째 열에는 결제 및 사용 보고서에 나타나는 사용 유형이 나열됩니다. 데이터의 일반적인 측정 단위는 기가바이트(GB)입니다. 그러나 서비스 및 보고서에 따라 TB(테라바이트)가 대신 나타날 수 있습니다.

사용 유형	단위	Granularity	설명
region1-*region2*-AWS-In-ABytes	GB	시간당	*region2*에서 *region1*로 전송된 가속화된 데이터의 양
region1-*region2*-AWS-In-ABytes-T1	GB	시간당	*region2*에서 *region1*로 전송된 T1 가속화된 데이터의 양. 여기서 T1은 미국, 유럽, 일본의 존재 지점(POPs)에 대한 CloudFront 요청을 의미합니다.
region1-*region2*-AWS-In-ABytes-T2	GB	시간당	*region2*에서 *region1*로 전송된 T2 가속화된 데이터의 양. 여기서 T2는 다른 모든 AWS 엣지 로케이션의 POPs에 대한 CloudFront 요청을 의미합니다.
region1-*region2*-AWS-In-Bytes	GB	시간당	*region2*에서 *region1*로 전송된 데이터의 양
region1-*region2*-AWS-Out-ABytes	GB	시간당	*region1*에서 *region2*로 전송된 가속화된 데이터의 양

사용 유형	단위	Granularity	설명
region1-*region2*-AWS-Out-ABytes-T1	GB	시간당	*region1*에서 *region2*로 전송된 T1 가속화된 데이터의 양. 여기서 T1은 미국, 유럽, 일본의 POP에 대한 CloudFront 요청을 의미합니다.
region1-*region2*-AWS-Out-ABytes-T2	GB	시간당	*region1*에서 *region2*로 전송된 T2 가속화된 데이터의 양. 여기서 T2는 다른 모든 AWS 엣지 로케이션의 POP에 대한 CloudFront 요청을 의미합니다.
region1-*region2*-AWS-Out-Bytes	GB	시간당	*region1*에서 *region2*로 전송된 데이터의 양
region-BatchOperations-Jobs	개수	시간당	수행된 S3 배치 작업의 수입니다.
region-BatchOperations-Objects	개수	시간당	S3 배치 작업에 의해 수행된 객체 작업의 수입니다.
region-Bulk-Retrieval-Bytes	GB	시간당	Bulk S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 요청으로 검색된 데이터의 양
region-BytesDeleted-GDA	GB	매월	S3 Glacier Deep Archive 스토리지에서 DeleteObject 작업으로 삭제된 데이터의 양

사용 유형	단위	Granularity	설명
region-BytesDeleted-GIR	GB	매월	S3 Glacier Instant Retrieval 스토리지에서 DeleteObject 작업으로 삭제된 데이터의 양
region-BytesDeleted-GLACIER	GB	매월	S3 Glacier Flexible Retrieval 스토리지에서 DeleteObject 작업으로 삭제된 데이터의 양
region-BytesDeleted-INT	GB	매월	S3 Intelligent-Tiering 스토리지에서 DeleteObject 작업으로 삭제된 데이터의 양
region-BytesDeleted-RRS	GB	매월	Reduced Redundancy Storage(RRS) 스토리지에서 DeleteObject 작업으로 삭제된 데이터의 양
region-BytesDeleted-SIA	GB	매월	S3 Standard-IA 스토리지에서 DeleteObject 작업으로 삭제된 데이터의 양
region-BytesDeleted-STANDARD	GB	매월	S3 Standard 스토리지에서 DeleteObject 작업으로 삭제된 데이터의 양
region-BytesDeleted-ZIA	GB	매월	S3 One Zone-IA 스토리지에서 DeleteObject 작업으로 삭제된 데이터의 양

사용 유형	단위	Granularity	설명
region-C3DataTransfer-In-Bytes	GB	시간당	같은 AWS 리전 내에서 Amazon EC2로부터 Amazon S3로 전송된 데이터의 양
region-C3DataTransfer-Out-Bytes	GB	시간당	같은 AWS 리전 내에서 Amazon S3로부터 Amazon EC2로 전송된 데이터의 양
region-CloudFront-In-Bytes	GB	시간당	CloudFront 배포에서 AWS 리전으로 전송된 데이터의 양
region-CloudFront-Out-Bytes	GB	시간당	AWS 리전에서 CloudFront 배포로 전송된 데이터의 양
region-DataTransfer-In-Bytes	GB	시간당	인터넷에서 Amazon S3으로 전송된 데이터의 양
region-DataTransfer-Out-Bytes	GB	시간당	Amazon S3에서 인터넷으로 전송된 데이터의 양[1]
region-DataTransfer-Regional-Bytes	GB	시간당	같은 AWS 리전 내에서 Amazon S3에서 AWS 리소스로 전송된 데이터의 양
region-EarlyDelete-ByteHrs	GB-시간	시간당	최소 90일 약정 종료 전 S3 Glacier Flexible Retrieval 스토리지에서 삭제된 객체에 대해 비례 할당으로 계산된 스토리지 사용량[2]

사용 유형	단위	Granularity	설명
region-EarlyDelete-GDA	GB-시간	시간당	최소 180일 약정 종료 전 S3 Glacier Deep Archive 스토리지에서 삭제된 객체에 대해 비례 할당으로 계산된 스토리지 사용량2
region-EarlyDelete-GIR	GB-시간	시간당	최소 90일 약정 종료 전 S3 Glacier Instant Retrieval에서 삭제된 객체에 대해 비례 할당으로 계산된 스토리지 사용량
region-EarlyDelete-GIR-SmObjects	GB-시간	시간당	최소 90일 약정이 끝나기 전에 S3 Glacier Instant Retrieval에서 삭제된 작은 객체(128KB 미만)에 대해 비례 할당으로 계산된 스토리지 사용량
region-EarlyDelete-SIA	GB-시간	시간당	최소 30일 약정이 끝나기 전에 S3 Standard-IA에서 삭제된 객체에 대해 비례 할당으로 계산된 스토리지 사용량[3]
region-EarlyDelete-SIA-SmObjects	GB-시간	시간당	최소 30일 약정이 끝나기 전에 S3 Standard-IA에서 삭제된 작은 객체(128KB 미만)에 대해 비례 할당으로 계산된 스토리지 사용량[3]

사용 유형	단위	Granularity	설명
region-EarlyDelete-ZIA	GB-시간	시간당	최소 30일 약정이 끝나기 전에 S3 One Zone-IA에서 삭제된 객체에 대해 비례 할당으로 계산된 스토리지 사용량[3]
region-EarlyDelete-ZIA-SmObjects	GB-시간	시간당	최소 30일 약정이 끝나기 전에 S3 One Zone-IA에서 삭제된 작은 객체(128KB 미만)에 대해 비례 할당으로 계산된 스토리지 사용량[3]
region-Expedited-Retrieval-Bytes	GB	시간당	Expedited S3 Glacier Flexible Retrieval 요청으로 검색된 데이터의 양
Global-Bucket-Hrs-FreeTier	버킷	매월	2,000개 버킷 계정 수준 무료 티어 이내 계정의 범용 버킷 수
Global-Bucket-Hrs	버킷	매월	2,000개 버킷 계정 수준 무료 티어를 초과하는 계정의 범용 버킷 수
region-Inventory-Objects Listed	Objects	시간당	인벤토리 목록에서 객체 그룹(객체가 버킷 또는 접두사별로 그룹화)에 나열된 객체의 수
region-Metadata-Updates	업데이트	시간당	S3 메타데이터에서 처리한 업데이트에 대한 업데이트당 요금

사용 유형	단위	Granularity	설명
region-Monitoring-Automation-INT	Objects	시간당	S3 Intelligent-Tiering 스토리지 클래스에서 모니터링되고 자동으로 계층화된 고유한 객체 수
region-MRAP-Out-Bytes	GB	시간당	리전의 버킷 중 S3 멀티 리전 액세스 포인트 엔드포인트를 통해 전송되는 데이터의 양(MRAP 데이터 라우팅 가격)입니다.
region-MRAP-In-Bytes	GB	시간당	리전의 버킷 중 S3 멀티 리전 액세스 포인트 엔드포인트를 통해 전송되는 데이터의 양(MRAP 데이터 라우팅 가격)입니다.
regiongroup1-regiongroup2--MRAP-Out-Bytes	GB	시간당	*regiongroup1* 의 버킷에서 AWS 네트워크 외부에 위치한 *regiongroup2* 의 클라이언트로 S3 멀티 리전 액세스 포인트 엔드포인트를 통해 전송된 데이터의 양입니다.
regiongroup1-regiongroup2--MRAP-In-Bytes	GB	시간당	AWS 네트워크 외부에 위치한 *regiongroup2* 의 클라이언트에서 *regiongroup1* 의 버킷으로 S3 멀티 리전 액세스 포인트 엔드포인트를 통해 전송된 데이터의 양입니다.

사용 유형	단위	Granularity	설명
region-OverwriteBytes-Copy-GDA	GB	매월	S3 Glacier Deep Archive 스토리지에서 CopyObject 작업으로 덮어쓴 데이터의 양
region-OverwriteBytes-Copy-GIR	GB	매월	S3 Glacier Instant Retrieval 스토리지에서 CopyObject 작업으로 덮어쓴 데이터의 양
region-OverwriteBytes-Copy-GLACIER	GB	매월	S3 Glacier Flexible Retrieval 스토리지에서 CopyObject 작업으로 덮어쓴 데이터의 양
region-OverwriteBytes-Copy-INT	GB	매월	S3 Intelligent-Tiering 스토리지에서 CopyObject 작업으로 덮어쓴 데이터의 양
region-OverwriteBytes-Copy-RRS	GB	매월	Reduced Redundancy Storage(RRS) 스토리지에서 CopyObject 작업으로 덮어쓴 데이터의 양
region-OverwriteBytes-Copy-SIA	GB	매월	S3 Standard-IA 스토리지에서 CopyObject 작업으로 덮어쓴 데이터의 양
region-OverwriteBytes-Copy-STANDARD	GB	매월	S3 Standard 스토리지에서 CopyObject 작업으로 덮어쓴 데이터의 양
region-OverwriteBytes-Copy-ZIA	GB	매월	S3 One Zone-IA 스토리지에서 CopyObject 작업으로 덮어쓴 데이터의 양

사용 유형	단위	Granularity	설명
region-OverwriteBytes-Put-GDA	GB	매월	S3 Glacier Deep Archive 스토리지에서 PutObject 작업으로 덮어쓴 데이터의 양
region-OverwriteBytes-Put-GIR	GB	매월	S3 Glacier Instant Retrieval 스토리지에서 PutObject 작업으로 덮어쓴 데이터의 양
region-OverwriteBytes-Put-GLACIER	GB	매월	S3 Glacier Flexible Retrieval 스토리지에서 PutObject 작업으로 덮어쓴 데이터의 양
region-OverwriteBytes-Put-INT	GB	매월	S3 Intelligent-Tiering 스토리지에서 PutObject 작업으로 덮어쓴 데이터의 양
region-OverwriteBytes-Put-RRS	GB	매월	Reduced Redundancy Storage(RRS) 스토리지에서 PutObject 작업으로 덮어쓴 데이터의 양
region-OverwriteBytes-Put-SIA	GB	매월	S3 Standard-IA 스토리지에서 PutObject 작업으로 덮어쓴 데이터의 양
region-OverwriteBytes-Put-STANDARD	GB	매월	S3 Standard 스토리지에서 PutObject 작업으로 덮어쓴 데이터의 양
region-OverwriteBytes-Put-ZIA	GB	매월	S3 One Zone-IA 스토리지에서 PutObject 작업으로 덮어쓴 데이터의 양

사용 유형	단위	Granularity	설명
region1-*region2*-S3RTC-In-Bytes	GB	매월	PutObjectReplTime , GetObjectReplTime , InitiateMultipartUploadReplTime , UploadPartReplTime , CompleteMultipartUploadReplTime , WriteACLReplTime 작업을 사용하여 S3 Replication Time Control(S3 RTC)을 위해 *region2*에서 *region1*로 전송된 데이터의 양
region1-*region2*-S3RTC-Out-Bytes	GB	매월	PutObjectReplTime , GetObjectReplTime , InitiateMultipartUploadReplTime , UploadPartReplTime , CompleteMultipartUploadReplTime , WriteACLReplTime 작업을 사용하여 S3 Replication Time Control(S3 RTC)을 위해 *region1*에서 *region2*로 전송된 데이터의 양

사용 유형	단위	Granularity	설명
region-Requests-GDA-Tier1	개수	시간당	S3 Glacier Deep Archive 객체에 대한 PUT, COPY, POST, CreateMultipartUpload, UploadPart 또는 CompleteMultipartUpload 요청의 수 [6]
region-Requests-GDA-Tier2	개수	시간당	S3 Glacier Deep Archive 객체에 대한 GET 및 HEAD 요청의 수
region-Requests-GDA-Tier3	개수	시간당	S3 Glacier Deep Archive 스탠다드 복원 요청의 수
region-Requests-GDA-Tier5	개수	시간당	Bulk S3 Glacier Deep Archive 복원 요청의 수
region-Requests-GIR-Tier1	개수	시간당	S3 Glacier Instant Retrieval 객체에 대한 PUT, COPY 또는 POST 요청의 수
region-Requests-GIR-Tier2	개수	시간당	S3 Glacier Instant Retrieval 객체에 대한 GET 및 기타 모든 비 S3 Glacier Instant Retrieval-Tier1 요청의 수

사용 유형	단위	Granularity	설명
region-Requests-GLACIER-Tier1	개수	시간당	S3 Glacier Flexible Retrieval 객체에 대한 PUT, COPY, POST, CreateMultipartUpload , UploadPart 또는 CompleteMultipartUpload 요청의 수 [6]
region-Requests-GLACIER-Tier2	개수	시간당	S3 Glacier Flexible Retrieval 객체에 대해 나열되지 않은 GET 및 기타 모든 요청의 수
region-Requests-INT-Tier1	개수	시간당	S3 Intelligent-Tiering 객체에 대한 PUT, COPY 또는 POST 요청의 수
region-Requests-INT-Tier2	개수	시간당	S3 Intelligent-Tiering 객체에 대한 GET 및 기타 모든 비 Tier1 요청의 수
region-Requests-SIA-Tier1	개수	시간당	S3 Standard-IA 객체에 대한 PUT, COPY 또는 POST 요청의 수
region-Requests-SIA-Tier2	개수	시간당	S3 Standard-IA 객체에 대한 GET 및 기타 모든 비 S3 및 기타 모든 비 S3 Glacier Instant Retrieval-Tier1 요청의 수

사용 유형	단위	Granularity	설명
region-Requests-Tier1	개수	시간당	S3 STANDARD, RRS 및 태그에 대한 PUT, COPY 또는 POST 요청 수와 모든 버킷 및 객체에 대한 LIST 요청의 수
region-Requests-Tier2	개수	시간당	GET 및 기타 모든 비 Tier1 요청의 수
region-Requests-Tier3	개수	시간당	S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 및 표준 S3 Glacier Flexible Retrieval 복원 요청에 대한 수명 주기 요청의 수
region-Requests-Tier4	개수	시간당	S3 Glacier Instant Retrieval, S3 Intelligent-Tiering, S3 Standard-IA 또는 S3 One Zone-IA 스토리지로의 수명 주기 전환의 수
region-Requests-Tier5	개수	시간당	Bulk S3 Glacier Flexible Retrieval 복원 요청의 수
region-Requests-Tier6	개수	시간당	Expedited S3 Glacier Flexible Retrieval 복원 요청의 수
region-Requests-Tier8	개수	시간당	S3 Access Grants 요청의 수
region-Requests-XZ-Tier1	개수	시간당	S3 Express One Zone 객체에 대한 PUT 또는 COPY 요청의 수

사용 유형	단위	Granularity	설명
region-Requests-XZ-Tier2	개수	시간당	S3 Express One Zone 객체에 대한 GET 및 기타 모든 비 S3 Express One Zone-Tier1 요청의 수
region-Requests-ZIA-Tier1	개수	시간당	S3 One Zone-IA 객체에 대한 PUT, COPY 또는 POST 요청의 수
region-Requests-ZIA-Tier2	개수	시간당	S3 One Zone-IA 객체에 대한 GET 및 기타 모든 비 S3 One Zone-IA-Tier1 요청의 수
region-Retrieval-GIR	GB	시간당	S3 Glacier Instant Retrieval 스토리지에서 검색된 데이터의 양
region-Retrieval-SIA	GB	시간당	S3 Standard-IA 스토리지에서 검색된 데이터의 양
region-Retrieval-XZ	GB	시간당	S3 Express One Zone 스토리지를 사용한 특정 검색 요청(PUT 또는 COPY)에서 512KB를 초과하는 데이터 부분
region-Retrieval-ZIA	GB	시간당	S3 One Zone-IA 스토리지에서 검색된 데이터의 양
region-S3DSSE-In-Bytes	GB	매월	Amazon S3에 의해 이중 암호화된 데이터의 양
region-S3DSSE-Out-Bytes	GB	매월	Amazon S3에서 복호화한 이중 암호화된 데이터의 양

사용 유형	단위	Granularity	설명
region-S3G-DataTransfer-In-Bytes	GB	시간당	S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지에서 객체를 복원하기 위해 Amazon S3로 전송된 데이터의 양
region-S3G-DataTransfer-Out-Bytes	GB	시간당	Amazon S3에서 객체를 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지로 이전하기 위해 전송된 데이터 양
region-Select-Returned-Bytes	GB	시간당	S3 Standard 스토리지에서 Select 요청으로 반환되는 데이터의 양
region-Select-Returned-GIR-Bytes	GB	시간당	S3 Glacier Instant Retrieval 스토리지에서 Select 요청으로 반환되는 데이터의 양
region-Select-Returned-INT-Bytes	GB	시간당	S3 Intelligent-Tiering 스토리지에서 Select 요청으로 반환되는 데이터의 양
region-Select-Returned-SIA-Bytes	GB	시간당	S3 Standard-IA 스토리지에서 Select 요청으로 반환되는 데이터의 양
region-Select-Returned-ZIA-Bytes	GB	시간당	S3 One Zone-IA 스토리지에서 Select 요청과 함께 반환되는 데이터의 양

사용 유형	단위	Granularity	설명
region-Select-Scanned-Bytes	GB	시간당	S3 Standard 스토리지에서 Select 요청으로 스캔한 데이터의 양
region-Select-Scanned-GIR-Bytes	GB	시간당	S3 Glacier Instant Retrieval 스토리지에서 Select 요청으로 스캔되는 데이터의 양
region-Select-Scanned-INT-Bytes	GB	시간당	S3 Intelligent-Tiering 스토리지에서 Select 요청으로 스캔한 데이터의 양
region-Select-Scanned-SIA-Bytes	GB	시간당	S3 Standard-IA 스토리지의 선택 요청으로 스캔한 데이터의 양
region-Select-Scanned-ZIA-Bytes	GB	시간당	S3 One Zone-IA 스토리지에서 Select 요청으로 스캔한 데이터의 양
region-Standard-Retrieval-Bytes	GB	시간당	표준 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 요청으로 검색된 데이터의 양
region-StorageAnalytics-ObjCount	Objects	시간당	각 스토리지 클래스 분석 구성에서 모니터링되는 고유 객체 수입니다.
region-StorageLens-ObjCount	Objects	일별	S3 Storage Lens 고급 지표 및 권장 사항에 의해 추적되는 각 S3 Storage Lens 대시보드의 고유 객체 수입니다.

사용 유형	단위	Granularity	설명
region-StorageLensFreeTier-ObjCount	Objects	일별	S3 Storage Lens 사용량 지표에 의해 추적되는 각 S3 Storage Lens 대시보드의 고유 객체 수입니다.
StorageObjectCount	개수	일별	지정된 버킷 내에 저장되는 객체의 수
region-Tables-CompactedObjects	Objects	시간당	Amazon S3 테이블 버킷에 압축된 객체 수
region-Tables-MonitoredObjects	Objects	시간당	Amazon S3 테이블 버킷의 객체 수
region-Tables-ProcessedBytes	GB	시간당	Amazon S3 테이블 버킷에서 압축을 위해 처리되는 데이터의 양
region-Tables-Requests-Tier1	개수	시간당	Amazon S3 테이블 버킷의 PUT 요청 수
region-Tables-Requests-Tier2	개수	시간당	Amazon S3 테이블 버킷의 Tier1 요청이 아닌 기타 모든 요청 및 GET 요청의 수
region-Tables-TimedStorage-ByteHrs	GB-월	일별	Amazon S3 테이블 버킷에 데이터가 저장된 GB-월 수
region-TagStorage-TagHrs	Tag-Hours	일별	시간별로 보고된 버킷의 모든 객체에 대한 총 태그 수
region-TimedStorage-ByteHrs	GB-월	일별	S3 Standard 스토리지에 데이터가 저장된 GB-월 수

사용 유형	단위	Granularity	설명
region-TimedStorage-GDA-ByteHrs	GB-월	일별	S3 Glacier Deep Archive 스토리지에 데이터가 저장된 GB-월 수
region-TimedStorage-GDA-Staging	GB-월	일별	S3 Glacier Deep Archive 스테이징 스토리지에 데이터가 저장된 GB-월 수
region-TimedStorage-GIR-ByteHrs	GB-월	일별	S3 Glacier Instant Retrieval 스토리지에 데이터가 저장된 GB-월 수
region-TimedStorage-GIR-SmObjects	GB-월	일별	S3 Glacier Instant Retrieval 스토리지에 작은 객체(128KB 미만)가 저장된 GB-월 수
region-TimedStorage-GlacierByteHrs	GB-월	일별	S3 Glacier Flexible Retrieval 스토리지에 데이터가 저장된 GB-월 수
region-TimedStorage-GlacierStaging	GB-월	일별	S3 Glacier Flexible Retrieval 스테이징 스토리지에 데이터가 저장된 GB-월 수
region-TimedStorage-INT-FA-ByteHrs	GB-월	일별	S3 Intelligent-Tiering 스토리지의 Frequent Access 티어에 데이터가 저장되었던 GB-월 수[5]
region-TimedStorage-INT-IA-ByteHrs	GB-월	일별	S3 Intelligent-Tiering 스토리지의 Infrequent Access 티어에 데이터가 저장되었던 GB-월 수

사용 유형	단위	Granularity	설명
region-TimedStorage-INT-AA-ByteHrs	GB-월	일별	S3 Intelligent-Tiering 스토리지의 Archive Access 티어에 데이터가 저장되었던 GB-월 수
region-TimedStorage-INT-AIA-ByteHrs	GB-월	일별	S3 Intelligent-Tiering 스토리지의 Archive Instant Access 티어에 데이터가 저장되었던 GB-월 수
region-TimedStorage-INT-DAA-ByteHrs	GB-월	일별	S3 Intelligent-Tiering 스토리지의 Deep Archive Access 티어에 데이터가 저장되었던 GB-월 수
region-TimedStorage-RRS-ByteHrs	GB-월	일별	Reduced Redundancy Storage(RRS) 스토리지에 데이터가 저장된 GB-월 수
region-TimedStorage-SIA-ByteHrs	GB-월	일별	S3 Standard-IA 스토리지에 데이터가 저장된 GB-월 수
region-TimedStorage-SIA-SmObjects	GB-월	일별	S3 Standard-IA 스토리지에 작은 객체(128KB 미만)가 저장된 GB-월 수[4]
region-TimedStorage-XZ-ByteHrs	GB-월	일별	S3 Express One Zone 스토리지에 데이터가 저장된 GB-월 수
region-TimedStorage-ZIA-ByteHrs	GB-월	일별	S3 One Zone-IA 스토리지에 데이터가 저장된 GB-월 수

사용 유형	단위	Granularity	설명
region-TimedStorage-ZIA-SmObjects	GB-월	일별	S3 One Zone-IA 스토리지에 작은 객체(128KB 미만)가 저장된 GB-월 수
region-Upload-XZ	GB	시간당	S3 Express One Zone을 사용한 특정 업로드 요청(PUT 또는 COPY)에서 512KB를 초과하는 데이터의 양

Notes

1. Global-Bucket-Hrs 및 Global-Bucket-Hrs-FreeTier 사용 유형은 상용 AWS 리전 및 AWS GovCloud (US)의 범용 버킷에 적용됩니다.
2. 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.
3. S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스에 아카이브된 객체가 90일(S3 Glacier Instant Retrieval 및 S3 Glacier Flexible Retrieval의 경우) 또는 180일(S3 Glacier Deep Archive의 경우)의 최소 스토리지 약정 기간이 경과하기 전에 삭제되거나 덮어쓰거나 다른 스토리지 클래스로 이전되면 남은 기간(일) 동안 기가바이트당 요금이 비례 할당으로 계산됩니다.
4. S3 Standard-IA 또는 S3 One Zone-IA 스토리지에 보관되는 객체의 경우에는 30일 이전에 삭제 또는 덮어쓰기 되거나 다른 스토리지 클래스로 이전되면 남은 기간에 대해 기가바이트별로 요금이 비례 배분되어 계산됩니다.
5. S3 Standard-IA 또는 S3 One Zone-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. S3 Intelligent-Tiering 스토리지 클래스의 객체에 대해 청구 가능한 최소 객체 크기는 없습니다. 128KB보다 작은 객체는 모니터링되지 않거나 자동 계층화를 수행할 수 없습니다. 작은 객체는 항상 S3 Intelligent-Tiering Frequent Access 계층에 저장됩니다.

7. S3 Glacier Flexier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스에 `CreateMultipartUpload`, `UploadPart` 또는 `UploadPartCopy` 요청을 시작하면 멀티파트 업로드가 완료될 때까지 요청에 S3 Standard 요청 요금이 청구됩니다. 업로드가 완료되면 대상 S3 Glacier 스토리지에 해당하는 PUT 요금이 단일 `CompleteMultipartUpload` 요청에 청구됩니다. S3 Glacier Flexier Flexible Retrieval 스토리지 클래스로의 PUT에 멀티파트 업로드 파트가 진행 중인 경우, 업로드가 완료될 때까지 S3 Glacier Flexible Retrieval Staging Storage로 S3 Standard 스토리지 요금이 청구됩니다. 마찬가지로 S3 Glacier Deep Archive 스토리지 클래스로의 PUT에 멀티파트 업로드 파트가 진행 중인 경우, 업로드가 완료될 때까지 S3 Glacier Deep Archive Staging Storage로 S3 Standard 스토리지 요금이 청구됩니다.
8. S3 Express One Zone은 최대 512KB 크기의 요청에 대해 요청당 고정 요금을 적용합니다. 요청 중 512KB를 초과하는 부분에 대해서는 PUT 요청 및 GET 요청에 GB당 추가 요금이 적용됩니다.
9. S3 Express One Zone 스토리지 클래스에 지원되는 기능에 대한 자세한 내용은 디렉터리 버킷에서 지원되지 않는 Amazon S3 기능 섹션을 참조하세요.
10. GB 단위로 청구되는 단위를 포함한 사용 유형은 사용 보고서에서 바이트 단위로 계산됩니다.
11. GB-월은 한 달 동안의 총 GB-시간을 집계하여 총 GB-시간 수를 구한 다음 해당 월의 시간 수로 나누어 구합니다. 자세한 내용은 자주 묻는 질문: Amazon S3 사용에 대한 요금 및 청구는 어떻게 이루어지나요?를 참조하세요.

> **Note**
>
> 일반적으로 S3 버킷 소유자에게는 HTTP `200 OK` 성공 응답 및 HTTP `4XX` 클라이언트 오류 응답이 반환된 요청에 대해 요금이 청구됩니다. 버킷 소유자에게는 HTTP `503 Slow Down` 오류와 같은 HTTP `5XX` 서버 오류 응답에 대해서는 요금이 청구되지 않습니다. 요금이 청구되지 않는 HTTP `3XX` 및 `4XX` 상태 코드 아래의 S3 오류 코드에 대한 자세한 내용은 Amazon S3 오류 응답에 대한 청구 섹션을 참조하세요. 버킷이 요청자 지불 버킷으로 구성된 경우 청구 요금에 대한 자세한 내용은 요청자 지불 요금의 방식 섹션을 참조하세요.

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.

추가 정보

- Amazon S3에 대한 AWS 사용 보고서
- Amazon S3에 대한 AWS Billing 보고서
- Amazon S3 요금
- Amazon S3 FAQ

Amazon S3 오류 응답에 대한 청구

일반적으로 S3 버킷 소유자에게는 HTTP `200 OK` 성공 응답 및 HTTP `4XX` 클라이언트 오류 응답이 반환된 요청에 대해 요금이 청구됩니다. 버킷 소유자에게는 HTTP `503 Slow Down` 오류와 같은 HTTP `5XX` 서버 오류 응답에 대해서는 요금이 청구되지 않습니다. 버킷이 요청자 지불 버킷으로 구성된 경우 청구 요금에 대한 자세한 내용은 요청자 지불 요금의 방식 섹션을 참조하세요.

다음 테이블에는 요금이 청구되지 않는 HTTP `3XX` 및 `4XX` 상태 코드의 특정 오류 코드가 나열되어 있습니다. 웹 사이트 호스팅으로 구성된 버킷의 경우 S3가 사용자 지정 오류 문서를 반환하거나 사용자 지정 리디렉션을 반환할 때 해당 요청 및 기타 요금이 계속 적용됩니다.

> **Note**
>
> `AccessDenied`(HTTP `403 Forbidden`)의 경우 S3는 요청이 버킷 소유자의 개인 AWS 계정이나 버킷 소유자의 AWS 조직 외부에서 시작된 경우 버킷 소유자에게 요금을 청구하지 않습니다.

HTTP 상태 코드	오류 코드	오류 코드에 대한 설명	
301 Moved Permanently(301 영구 이동됨)	PermanentRedirect	액세스하려는 버킷이 지정된 엔트포인트를 사용하여 처리되어야 합니다. 이 엔드포인트로 모든 향후 요청을 보냅니다.	
	PermanentRedirectControlError	액세스하려는 API 작업이 지정된 엔트포인트를 사용하여 처리되어야 합니다.	

HTTP 상태 코드	오류 코드	오류 코드에 대한 설명
		이 엔드포인트로 모든 향후 요청을 보냅니다.
307 임시 리디렉션	TemporaryRedirect	도메인 이름 시스템(DNS) 서버가 업데이트되는 동안 버킷으로 리디렉션됩니다.
400 잘못된 요청	AuthorizationHeaderMalformed	제공한 권한 부여 헤더가 유효하지 않습니다.
	AuthorizationQueryParametersError	제공한 권한 부여 쿼리 매개변수가 유효하지 않습니다.
	ConnectionClosedByRequester	WriteGetObjectResponse 본문을 읽는 동안 오류가 발생하여 원래 호출자에게 반환되었습니다.
	DeviceNotActiveError	디바이스가 현재 활성화 상태가 아닙니다.
	EndpointNotFound	요청을 올바른 엔드포인트로 전달합니다.
	ExpiredToken	제공된 토큰이 만료되었습니다.

HTTP 상태 코드	오류 코드	오류 코드에 대한 설명	
	IllegalLocationConstraintException	버킷이 있는 리전과 다른 리전의 버킷에 액세스하려고 합니다. 이 오류를 방지하려면 `--region` 옵션을 사용하세요. 예를 들면 `aws s3 cp awsexample.txt s3://amzn-s3-demo-bucket/ --region ap-east-1` 입니다.	
	InvalidArgument	이 오류는 다음과 같은 이유로 발생할 수 있습니다. • 지정된 인수가 유효하지 않습니다. • 요청에 필수 헤더가 누락되었습니다. • 지정된 인수가 불완전하거나 형식이 잘못되었습니다. • 지정된 인수의 길이는 3 이상이어야 합니다.	
	InvalidBucketOwnerAWSAccountID	예상되는 버킷 소유자 파라미터의 값은 AWS 계정 ID여야 합니다.	
	InvalidDigest	지정한 Content-MD5 또는 체크섬 값이 유효하지 않습니다.	

HTTP 상태 코드	오류 코드	오류 코드에 대한 설명	
	InvalidEncryptionAlgorithmError	지정한 암호화 요청이 유효하지 않습니다. 유효한 값은 AES256입니다.	
	InvalidHostHeader	요청에 제공된 호스트 헤더가 잘못된 스타일 주소를 사용했습니다.	
	InvalidHttpMethod	예상치 못한 HTTP 메서드를 사용하여 요청했습니다.	

HTTP 상태 코드	오류 코드	오류 코드에 대한 설명	
	InvalidRequest	이 오류는 다음과 같은 이유로 발생할 수 있습니다. • 요청에서 잘못된 서명 버전을 사용하고 있습니다. AWS4-HMAC-SHA256(서명 버전 4)을 사용합니다. • 액세스 포인트는 기존 버킷에 대해서만 생성할 수 있습니다. • 액세스 포인트가 삭제할 수 있는 상태가 아닙니다. • 액세스 포인트는 기존 버킷에 대해서만 나열할 수 있습니다. • 다음 토큰이 유효하지 않습니다. • 수명 주기 규칙에 하나 이상의 작업을 지정해야 합니다. • 하나 이상의 수명 주기 규칙을 지정해야 합니다. • 수명 주기 규칙의 수는 허용된 규칙 수인 1,000	

HTTP 상태 코드	오류 코드	오류 코드에 대한 설명	
		개를 초과할 수 없습니다. • MaxResults 파라미터의 범위가 유효하지 않습니다. • SOAP 요청은 HTTPS 연결을 통해 이루어져야 합니다. • Amazon S3 Transfer Acceleration에서는 DNS를 준수하지 않는 이름의 버킷이 지원되지 않습니다. • Amazon S3 Transfer Acceleration에서는 이름에 마침표(.)가 포함된 버킷이 지원되지 않습니다. • Amazon S3 Transfer Acceleration 엔드포인트는 시각적 스타일 요청만 지원합니다. • Amazon S3 Transfer Acceleration이 이 버킷에서 구성되어 있지 않습니다. • Amazon S3 Transfer Acceleration이 이 버킷	

HTTP 상태 코드	오류 코드	오류 코드에 대한 설명	
		에서 비활성화되어 있습니다. • Amazon S3 Transfer Acceleration이 이 버킷에서 지원되지 않습니다. 도움이 필요하면 지원로 문의하십시오. • Amazon S3 Transfer Acceleration을 이 버킷에서 활성화할 수 없습니다. 도움이 필요하면 지원로 문의하십시오. • HTTP 헤더와 쿼리 파라미터에 제공된 값이 서로 충돌합니다. • HTTP 헤더와 POST 양식 필드에 제공된 값이 서로 충돌합니다. • 크기가 5GB보다 큰 객체에 대한 CopyObject 요청이 이루어졌습니다.	
	InvalidSessionException	제한 시간이 초과되었거나 만료되어 세션이 더 이상 존재하지 않는 경우 반환됩니다.	

HTTP 상태 코드	오류 코드	오류 코드에 대한 설명	
	InvalidSignature	서버가 계산한 요청 서명이 사용자가 제공한 서명과 일치하지 않습니다. AWS 비밀 액세스 키와 서명 방법을 확인합니다. 자세한 내용은 REST 요청 서명 및 인증을 참조하세요.	
	InvalidSOAPRequest	SOAP 요청 본문이 유효하지 않습니다.	
	InvalidStorageClass	지정한 스토리지 클래스가 유효하지 않습니다.	
	InvalidTag	요청에 유효하지 않은 태그 입력이 포함되어 있습니다. 예를 들어 요청에 중복된 키, 너무 긴 키 또는 값, 시스템 태그가 포함되어 있을 수 있습니다.	
	InvalidToken	입력된 토큰의 형식이 잘못되었거나 다른 이유로 유효하지 않습니다.	
	InvalidURI	지정된 URI를 구문 분석할 수 없습니다.	
	KeyTooLongError	키가 너무 깁니다.	
	KMS.DisabledException	지정한 KMS 키가 활성화되지 않아서 요청이 거부되었습니다.	

HTTP 상태 코드	오류 코드	오류 코드에 대한 설명	
	KMS.InvalidKeyUsageException	다음과 같은 이유 중 하나로 요청이 거부되었습니다. • KMS 키의 KeyUsage 값이 API 작업과 호환되지 않습니다. • 작업에 지정된 암호화 알고리즘 또는 서명 알고리즘이 KMS 키(KeySpec)의 키 구성 요소 유형과 호환되지 않습니다. 암호화, 암호 해독, 재암호화 및 데이터 키 생성을 위해서는 KeyUsage가 ENCRYPT_DECRYPT여야 합니다. 메시지 서명 및 검증을 위해서는 KeyUsage가 SIGN_VERIFY여야 합니다. 메시지 인증 코드(MAC)를 생성하고 확인하려면 KeyUsage가 GENERATE_VERIFY_MAC이어야 합니다. 키 일치 비밀을 도출하려면 KeyUsage가 KEY_AGREEMENT여야 합니다. KMS 키의 KeyUsage를 확인하려면 DescribeKey 작업을 사용합니다.	

HTTP 상태 코드	오류 코드	오류 코드에 대한 설명	
		특정 KMS 키에 지원되는 암호화 또는 서명 알고리즘을 찾으려면 DescribeKey 작업을 사용합니다.	
	KMS.KMSInvalidStateException	지정한 리소스의 상태가 이 요청에 유효하지 않아서 요청이 거부되었습니다. 이 예외는 다음 중 하나를 의미합니다. • KMS 키의 키 상태가 작업과 호환되지 않습니다. 키 상태를 확인하려면 DescribeKey 작업을 사용합니다. 각 KMS 작업과 호환되는 키 상태에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 AWS KMS keys 키 상태를 참조하세요. • 사용자 지정 키 저장소의 KMS 키에 대한 암호화 작업 시 이 예외는 여러 가지 원인이 있을 수 있는 일반적인 오류를 나타냅니다. 원인을 식별하려면 예외와 함께 제공되는 오류 메시지를 참조하세요.	

HTTP 상태 코드	오류 코드	오류 코드에 대한 설명	
	KMS.NotFoundException	지정한 엔터티 또는 리소스를 찾을 수 없으므로 요청이 거부되었습니다.	
	LambdaInvalidResponse	WriteGetObjectResponse가 AWS Lambda에 ValidationError로 응답하여 원래 호출자에게 반환되었습니다. 자세한 내용은 ValidationError 메시지를 참조하세요. 모든 ValidationError 오류로 인해 LambdaInvalidResponse 오류가 발생하는 것은 아닙니다.	
	LambdaInvocationFailed	Lambda 함수 간접 호출이 실패했습니다. S3 객체 Lambda가 구성된 Lambda 함수를 간접적으로 호출할 수 없는 경우 호출자에게 다음 오류가 표시될 수 있습니다. 오류 메시지에는 함수를 호출할 때 AWS Lambda 서비스에서 반환한 최종 오류에 대한 세부 정보(예: 상태 코드, 오류 코드, 오류 메시지, 요청 ID)가 포함될 수 있습니다.	

HTTP 상태 코드	오류 코드	오류 코드에 대한 설명	
	MalformedACLError	제공한 ACL이 잘못된 형식으로 되어 있고 게시된 스키마를 기준으로 검증되지 않았습니다.	
	MalformedPOSTRequest	POST 요청 본문이 올바른 형식의 멀티파트/양식 데이터가 아닙니다.	
	MalformedXML	제공한 XML이 잘못된 형식으로 되어 있고 게시된 스키마를 기준으로 검증되지 않았습니다.	
	MaxPostPreDataLengthExceededError	업로드 파일 앞에 있는 POST 요청 필드가 너무 컸습니다.	
	MetadataTooLarge	메타데이터 헤더가 최대 허용 메타데이터 크기를 초과합니다.	
	MissingAttachment	SOAP 첨부 파일이 필요했지만 해당 파일이 없었습니다.	
	MissingRequestBodyError	빈 XML 문서를 요청으로 보냈습니다.	
	MissingSecurityHeader	요청에 필수 헤더가 누락되었습니다.	
	NoLoggingStatusForKey	키에는 로깅 상태 하위 리소스라는 것이 없습니다.	

HTTP 상태 코드	오류 코드	오류 코드에 대한 설명	
	NotDeviceOwnerError	토큰을 생성한 디바이스가 인증된 사용자의 소유가 아닙니다.	
	ResponseInterrupted	WriteGetObjectResponse 본문을 읽는 동안 오류가 발생하여 원래 호출자에게 반환되었습니다.	
	RequestHeaderSectionTooLarge	요청을 만드는 데 사용된 요청 헤더 및 쿼리 파라미터가 최대 허용 크기를 초과합니다.	
	TokenCodeInvalidError	제공한 일련 번호 및/또는 토큰 코드가 유효하지 않습니다.	
	UnexpectedContent	이 요청에는 지원되지 않는 콘텐츠가 포함되어 있습니다.	
	UnsupportedArgument	요청에 지원되지 않는 인수가 포함되어 있습니다.	
	UnsupportedSignature	제공된 요청이 지원되지 않는 STS 토큰 버전으로 서명되었거나 해당 서명 버전이 지원되지 않습니다.	

HTTP 상태 코드	오류 코드	오류 코드에 대한 설명	
	UserKeyMustBeSpecified	버킷 POST 요청에는 지정된 필드 이름이 포함되어야 합니다. 필드 이름이 지정되었다면 필드의 순서를 확인합니다.	
	IncorrectEndpoint	지정된 버킷이 다른 리전에 있습니다. 요청을 올바른 엔드포인트로 전달합니다.	
	ValidationError	WriteGetObjectResponse API 작업에서 검증 오류가 반환될 수 있으며 이는 여러 가지 이유로 발생할 수 있습니다. 자세한 내용은 오류 메시지를 참조하세요.	
403 금지됨	RequestTimeTooSkewed	요청 시간과 서버 시간 간의 차이가 너무 큽니다.	
	SignatureDoesNotMatch	서버가 계산한 요청 서명이 사용자가 제공한 서명과 일치하지 않습니다. AWS 비밀 액세스 키와 서명 방법을 확인합니다. 자세한 내용은 REST 인증과 SOAP 인증을 확인하세요.	

HTTP 상태 코드	오류 코드	오류 코드에 대한 설명	
	NotSignedUp	계정이 Amazon S3 서비스에 가입되지 않았습니다. Amazon S3를 사용하려면 가입해야 합니다. 다음 URL에서 가입할 수 있습니다. https://aws.amazon.com/s3	
	InvalidSecurity	입력한 보안 자격 증명이 유효하지 않습니다.	
	InvalidPayer	이 개체에 대한 모든 액세스가 비활성화되었습니다. 추가 지원이 필요한 경우 문의처를 참조하세요.	
	InvalidAccessKeyId	입력한 AWS 액세스 키 ID가 AWS의 레코드에 존재하지 않습니다.	
	AccountProblem	AWS 계정에 해당 작업을 성공적으로 완료하지 못하도록 방해하는 문제가 있습니다. 추가 지원이 필요한 경우 문의처를 참조하세요.	
	UnauthorizedAccessError	중국 지역에만 적용됩니다. ICP 라이선스가 없는 버킷에 요청을 보냈을 때 반환됩니다. 자세한 내용은 ICP 기록을 참조하세요.	

HTTP 상태 코드	오류 코드	오류 코드에 대한 설명	
	UnexpectedIPError	중국 지역에만 적용됩니다. 예기치 않은 IP로 인해 요청이 거부되었습니다.	
	MissingAuthenticationToken	요청이 서명되지 않았습니다.	
	LambdaPermissionError	호출자에게 Lambda 함수를 간접적으로 호출할 권한이 없습니다. 호출자에게 Lambda 함수를 간접적으로 호출할 권한이 있어야 합니다. 호출자에게 연결된 정책을 검토하여 구성된 함수에 `lambda:InvokeFunction` 를 사용할 수 있도록 허용되었는지 확인하세요. 오류 메시지에는 함수를 호출할 때 Lambda 서비스에서 반환한 최종 오류에 대한 세부 정보(예: 상태 코드, 오류 코드, 오류 메시지, 요청 ID)가 포함될 수 있습니다.	

HTTP 상태 코드	오류 코드	오류 코드에 대한 설명	
404 Not Found(404 찾을 수 없음)	LambdaNotFound	AWS Lambda 함수를 찾을 수 없습니다. 간접 호출 시도 시 구성된 Lambda 함수, 버전 또는 별칭을 찾을 수 없습니다. S3 객체 Lambda 액세스 포인트 구성이 올바른 Lambda 함수 ARN을 가리키는지 확인하세요. 오류 메시지에는 함수를 호출할 때 AWS Lambda 서비스에서 반환한 최종 오류에 대한 세부 정보(예: 상태 코드, 오류 코드, 오류 메시지, 요청 ID)가 포함될 수 있습니다.	
	NoSuchAsyncRequest	지정된 리소스를 찾을 수 없습니다.	
	NoSuchObjectLockConfiguration	지정된 객체에 ObjectLock 구성이 없습니다.	
	NoSuchUpload	지정된 멀티파트 업로드가 존재하지 않습니다. 업로드 ID가 유효하지 않거나 멀티파트 업로드가 중단 또는 완료되었을 수 있습니다.	
	NoSuchWebsiteConfiguration	지정된 버킷에 웹 사이트 구성이 없습니다.	
	NoTransformationDefined	이 객체 Lambda 액세스 포인트에 대한 변환이 없습니다.	

HTTP 상태 코드	오류 코드	오류 코드에 대한 설명
	ObjectLockConfigurationNotFoundError	이 버킷에 대한 객체 잠금 구성이 존재하지 않습니다.
405 메서드 허용 안 함	MethodNotAllowed	지정된 메서드가 이 리소스에 허용되지 않습니다.
409 충돌	BucketAlreadyExists	요청된 버킷 이름을 사용할 수 없습니다. 버킷 네임스페이스는 시스템의 모든 사용자가 공유합니다. 다른 이름을 지정하고 다시 시도합니다.
	InvalidBucketState	요청이 버킷의 현재 상태에 대해 유효하지 않습니다.
	OperationAborted	현재 이 리소스에 대해 서로 충돌하는 조건부 작업이 진행 중입니다. 다시 시도하세요.
411 길이 필요	MissingContentLength	콘텐츠 길이 HTTP 헤더를 제공해야 합니다.
412 사전 조건 실패	RequestIsNotMultiPartContent	버킷 POST 요청은 엔클로저 유형의 멀티파트/양식 데이터여야 합니다.
4.16 요청 범위가 만족스럽지 않음	InvalidRange	요청한 범위가 요청에 유효하지 않습니다. 다른 범위를 사용해 보세요.

Amazon S3 스토리지 클래스 이해 및 관리

Amazon S3의 각 객체에는 그와 연결된 스토리지 클래스가 있습니다. 기본적으로 S3의 객체는 S3 Standard 스토리지 클래스에 저장되지만, Amazon S3는 저장하는 객체에 대한 여러 다른 스토리지 클래스를 제공합니다. 사용 사례 시나리오 및 성능 액세스 요구 사항에 따라 클래스를 선택합니다. 사용 사례에 맞게 설계된 스토리지 클래스를 선택하면 객체의 스토리지 비용, 성능 및 가용성을 최적화할 수 있습니다. 모든 스토리지 클래스는 내구성이 뛰어납니다.

다음 섹션에서는 다양한 스토리지 클래스에 대한 세부 정보 그리고 객체에 대한 스토리지 클래스를 설정하는 방법을 설명합니다.

주제

- 자주 액세스하는 객체를 위한 스토리지 클래스
- 변경되는 또는 알 수 없는 액세스 패턴으로 데이터를 자동으로 최적화하는 스토리지 클래스
- 자주 액세스하지 않는 객체를 위한 스토리지 클래스
- 거의 액세스하지 않는 객체를 위한 스토리지 클래스
- Amazon S3 on Outposts에 대한 스토리지 클래스
- Amazon S3 스토리지 클래스 비교
- 객체의 스토리지 클래스 설정
- Amazon S3 분석 - 스토리지 클래스 분석
- Amazon S3 Intelligent-Tiering을 사용한 스토리지 비용 관리
- 장기 데이터 스토리지를 위한 S3 Glacier 스토리지 클래스 이해하기
- 아카이브된 객체 작업

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(STANDARD) – 기본 스토리지 클래스입니다. 객체를 업로드할 때 스토리지 클래스를 지정하지 않으면 Amazon S3가 S3 Standard 스토리지 클래스를 할당합니다. Amazon S3 분석 - 스토리지 클래스 분석를 사용하여 S3 Standard 및 S3 Standard-IA 간의 비용을 최적화할 수 있습니다.
- S3 Express One Zone(EXPRESS_ONEZONE) - Amazon S3 Express One Zone은 지연 시간에 가장 민감한 애플리케이션에 대해 일관되게 10밀리초 미만의 데이터 액세스를 제공하도록 특별히 설계

된 고성능 단일 영역 Amazon S3 스토리지 클래스입니다. S3 Express One Zone은 현재 사용 가능한 클라우드 객체 스토리지 클래스 중 지연 시간이 가장 낮으며, S3 Standard보다 데이터 액세스 속도는 최대 10배 빠르고 요청 비용은 50% 저렴합니다. S3 Express One Zone 스토리지 클래스를 사용하면 단일 가용 영역 내의 여러 디바이스에 데이터가 중복으로 저장됩니다. 자세한 내용은 S3 Express One Zone 섹션을 참조하세요.

- Reduced Redundancy Storage(REDUCED_REDUNDANCY) – Reduced Redundancy Storage(RRS) 클래스는 S3 Standard 스토리지 클래스보다 더 적은 중복성으로 저장될 수 있는 데이터 중에서도 중요하지 않고 재현 가능한 데이터를 목적으로 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.

변경되는 또는 알 수 없는 액세스 패턴으로 데이터를 자동으로 최적화하는 스토리지 클래스

S3 Intelligent-Tiering(INTELLIGENT_TIERING)은 성능 영향 또는 운영 오버헤드 없이 가장 비용 효과적인 액세스 계층으로 데이터를 자동으로 이동하여 스토리지 비용을 최적화하기 위해 설계된 Amazon S3 스토리지 클래스입니다. S3 Intelligent-Tiering은 액세스 패턴이 변화할 때 액세스 계층 사이에서 세분화된 객체 수준으로 데이터를 이동하여 자동 비용 절감 효과를 제공하는 유일한 스토리지 클래스입니다. S3 Intelligent-Tiering은 액세스 패턴을 알 수 없거나 액세스 패턴이 변화하는 데이터에 대한 스토리지 비용을 최적화하려는 경우에 이상적인 스토리지 클래스입니다. S3 Intelligent-Tiering에는 검색 요금이 없습니다.

약간의 월별 객체 모니터링 및 자동화 요금만 지불하면 S3 Intelligent-Tiering에서 액세스 패턴을 모니터링하고, 액세스하지 않은 객체를 저렴한 액세스 계층으로 자동으로 이동합니다. S3 Intelligent-Tiering을 통해 지연 시간이 짧고 처리량이 높은 세 액세스 계층에서 스토리지 비용을 자동으로 절감할 수 있습니다. 비동기식으로 액세스할 수 있는 데이터의 경우 S3 Intelligent-Tiering 스토리지 클래스 내에서 자동 아카이브 기능을 활성화하도록 선택할 수 있습니다. S3 Intelligent-Tiering은 99.9% 의 가용성과 99.999999999%의 내구성을 제공하도록 설계되었습니다.

S3 Intelligent-Tiering은 3개의 액세스 계층에 객체를 자동으로 저장합니다.

- Frequent Access – S3 Intelligent-Tiering에 업로드하거나 이전한 객체는 Frequent Access 계층에 자동으로 저장됩니다.
- Infrequent Access – S3 Intelligent-Tiering은 30일 연속으로 액세스되지 않은 객체를 Infrequent Access 계층으로 이동합니다.
- Archive Instant Access – S3 Intelligent-Tiering을 사용하면 90일 연속으로 액세스되지 않은 기존 객체가 Archive Instant Access 계층으로 자동으로 이동됩니다.

이러한 세 가지 계층 외에도 S3 Intelligent-Tiering은 두 가지 선택적 아카이브 액세스 계층을 제공합니다.

- Archive Access – S3 Intelligent-Tiering은 비동기식으로 액세스할 수 있는 데이터에 대해 Archive Access 계층을 활성화하는 옵션을 제공합니다. 활성화 후 Archive Access 계층은 최소 연속 90일 동안 액세스하지 않은 객체를 자동으로 아카이브합니다.
- Deep Archive Access – S3 Intelligent-Tiering은 비동기식으로 액세스할 수 있는 데이터에 대해 Deep Archive Access 계층을 활성화하는 옵션을 제공합니다. 활성화 후 Deep Archive Access 계층은 최소 연속 180일 동안 액세스하지 않은 객체를 자동으로 아카이브합니다.

> **Note**
>
> - Archive Instant Access 계층을 우회하려는 경우에만 90일 동안 Archive Access 계층을 활성화합니다. Archive Access 계층은 몇 분에서 몇 시간의 검색 시간과 함께 약간 더 저렴한 스토리지 비용을 제공합니다. Archive Instant Access 계층은 밀리초 단위의 액세스와 높은 처리량 성능을 제공합니다.
> - 애플리케이션에서 객체에 비동기적으로 액세스할 수 있는 경우에만 Archive Access 및 Deep Archive Access 계층을 활성화합니다. 검색 중인 객체가 Archive Access 또는 Deep Archive Access 계층에 저장된 경우 먼저 `RestoreObject`를 사용하여 객체를 복원합니다.

새로 생성된 데이터를 S3 Intelligent-Tiering으로 이동하여 기본 스토리지 클래스로 구성할 수 있습니다. `PutBucketIntelligentTieringConfiguration` API 작업, AWS CLI 또는 Amazon S3 콘솔을 사용하여 아카이브 액세스 계층을 하나 또는 둘 다 활성화하도록 선택할 수도 있습니다. S3 Intelligent-Tiering 사용 및 아카이브 액세스 계층 활성화에 대한 자세한 내용은 S3 Intelligent-Tiering 사용 섹션을 참조하세요.

Archive Access 또는 Deep Archive Access 계층의 객체에 액세스하려면 먼저 객체를 복원해야 합니다. 자세한 내용은 S3 Intelligent-Tiering Archive Access 및 Deep Archive Access 계층에서 객체 복원 섹션을 참조하세요.

> **Note**
>
> 객체의 크기가 128KB 미만이면 자동 계층화를 모니터링 및 사용할 수 없습니다. 작은 객체는 항상 Frequent Access 계층에 저장됩니다. S3 Intelligent-Tiering에 대한 자세한 내용은 S3 Intelligent-Tiering 액세스 계층 섹션을 참조하세요.

자주 액세스하지 않는 객체를 위한 스토리지 클래스

S3 Standard-IA 및 S3 One Zone-IA 스토리지 클래스는 수명이 길고 자주 액세스하지 않는 데이터용으로 설계되었습니다. (IA는 Infrequent Access(빈번하지 않은 액세스)의 약어입니다.) S3 Standard-IA 및 S3 One Zone-IA 객체는 밀리초 단위로 액세스하는 데 사용할 수 있습니다(S3 Standard 스토리지 클래스와 유사). Amazon S3는 이들 객체에 대해 검색 비용을 부과하므로 이들 객체는 자주 액세스되지 않는 데이터에 가장 적합합니다. 요금 정보는 Amazon S3 요금을 참조하세요.

예를 들어, 다음의 경우에 S3 Standard-IA 및 S3 One Zone-IA 스토리지 클래스를 선택할 수 있을 것입니다.

- 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 S3가 객체들을 S3 Standard-IA 또는 S3 One Zone-IA 클래스로 전환하게 할 수 있을 것입니다.

  수명 주기 관리에 대한 자세한 내용은 객체 수명 주기 관리를 참조하세요.

> **Note**
>
> S3 Standard-IA 및 S3 One Zone-IA 스토리지 클래스는 사용자가 30일 이상 저장하려고 하는 128KB 이상의 객체에 적합합니다. 객체가 128KB 미만이더라도 Amazon S3는 128KB에 대한 요금을 부과합니다. 최소 스토리지 기간인 30일이 끝나기 전에 객체를 삭제하면 30일 요금이 부과됩니다. 30일 이전에 삭제되거나 덮어써지거나 다른 스토리지 클래스로 이전된 객체에는 일반 스토리지 사용 요금과 30일 최소 기간 중 남은 기간으로 비례 배분된 요금이 부과됩니다. 요금 정보는 Amazon S3 요금을 참조하세요.

이들 스토리지 클래스는 다음과 같은 차이가 있습니다.

- S3 Standard-IA(STANDARD_IA) – Amazon S3는 객체 데이터를 지리적으로 분리된 여러 개의 가용 영역(AZ)에 중복되게 저장합니다(S3 Standard 스토리지 클래스와 유사). S3 Standard-IA 객체는 가용 영역의 손실에 대한 복원력이 있습니다. 이 스토리지 클래스는 S3 One Zone-IA 클래스보다 뛰어난 가용성 및 복원성을 제공합니다. Amazon S3 분석 - 스토리지 클래스 분석를 사용하여 S3 Standard 및 S3 Standard-IA 간의 비용을 최적화하려면
- S3 One Zone-IA(ONEZONE_IA) – Amazon S3는 객체 데이터를 하나의 가용 영역에만 저장하므로 S3 Standard-IA보다 더 저렴합니다. 그러나 데이터는 지진 및 홍수와 같은 재해에 의한 가용 영역의 물리적 손실에 대해서는 복원성이 없습니다. S3 One Zone-IA 스토리지 클래스는 S3 Standard-IA만큼 내구성이 있으며 다만 가용성과 복원성은 더 낮습니다. 스토리지 클래스의 내구성과 가용성을 비교하려면 이 섹션 끝부분에 있는 Amazon S3 스토리지 클래스 비교을(를) 참조하세요. 요금 정보는 Amazon S3 요금을 참조하세요. 데이터 레지던시 및 격리 사용 사례의 경우 AWS 로컬 영역에서 디렉터리 버킷을 생성하고 S3 Express One Zone(EXPRESS_ONEZONE) 및 S3 One Zone-IA(ONEZONE_IA) 스토리지 클래스를 사용할 수 있습니다. 로컬 영역의 디렉터리 버킷에 대한 자세한 내용은 데이터 레지던시 워크로드 섹션을 참조하세요.

다음과 같이 하는 것이 좋습니다:

- S3 Standard-IA(STANDARD_IA) – 기본 데이터나 다시 생성할 수 없는 데이터의 유일한 복사본에만 사용합니다.
- S3 One Zone-IA(ONEZONE_IA) – 가용 영역에 장애 발생 시 데이터를 다시 생성할 수 있는 경우와 S3 교차 리전 복제(CRR)를 구성하는 경우 객체 복제본에 사용합니다. 또한 데이터 레지던시 및 격리의 경우 AWS 로컬 영역에서 디렉터리 버킷을 생성하고 S3 One Zone-IA 스토리지 클래스를 사용할 수 있습니다.

거의 액세스하지 않는 객체를 위한 스토리지 클래스

S3 Glacier Instant Retrieval(GLACIER_IR), S3 Glacier Flexible Retrieval(GLACIER) 및 S3 Glacier Deep Archive(DEEP_ARCHIVE) 스토리지 클래스는 저비용의 장기 데이터 저장 및 데이터 아카이브를 위해 설계되었습니다. 이러한 스토리지 클래스는 최소 스토리지 기간과 검색 요금이 필요하므로, 거의 액세스하지 않는 데이터에 가장 효과적입니다. S3 Glacier 스토리지 클래스에 대한 자세한 정보는 장기 데이터 스토리지를 위한 S3 Glacier 스토리지 클래스 이해하기 섹션을 참조하세요.

Amazon S3는 다음과 같은 S3 Glacier 스토리지 클래스를 제공합니다.

- S3 Glacier Instant Retrieval(GLACIER_IR) – 거의 액세스하지 않고 밀리초 단위로 검색해야 하는 장기 데이터에 사용합니다. 이 스토리지 클래스의 데이터는 실시간 액세스에 사용할 수 있습니다.
- S3 Glacier Flexible Retrieval(GLACIER) – 분 단위로 데이터의 일부를 검색해야 하는 아카이브에 사용합니다. 이 스토리지 클래스의 데이터는 아카이빙되며 실시간 액세스에 사용할 수 없습니다.
- S3 Glacier Deep Archive(DEEP_ARCHIVE) – 거의 액세스할 필요가 없는 데이터를 아카이브할 때 사용합니다. 이 스토리지 클래스의 데이터는 아카이빙되며 실시간 액세스에 사용할 수 없습니다.

아카이브된 객체 복원

객체의 스토리지 클래스 설정 섹션에서 설명한 것처럼 기타 스토리지 클래스와 동일한 방법으로 객체의 스토리지 클래스를 S3 Glacier Flexible Retrieval(GLACIER) 또는 S3 Glacier Deep Archive(DEEP_ARCHIVE)로 설정할 수 있습니다. 그러나 S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive 객체는 아카이빙되며 실시간 액세스에 사용할 수 없습니다. 자세한 내용은 S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive의 아카이브 스토리지 이해 섹션을 참조하세요.

> **Note**
>
> S3 Glacier 스토리지 클래스를 사용할 때 객체는 Amazon S3에 남아 있습니다. 별도의 Amazon S3 Glacier 서비스를 통해 객체에 직접 액세스할 수 없습니다. Amazon S3 Glacier 서비스에 대한 자세한 내용은 Amazon S3 Glacier 개발자 안내서를 참조하세요.

Amazon S3 on Outposts에 대한 스토리지 클래스

Amazon S3 on Outposts를 사용하면 AWS Outposts 리소스에서 S3 버킷을 생성하고 로컬 데이터 액세스, 로컬 데이터 처리 및 데이터 레지던시가 필요한 애플리케이션을 위해 온프레미스에서 객체를 저장하고 검색할 수 있습니다. 액세스 정책, 암호화, 태그 지정 등 Amazon S3에서와 같이 AWS Outposts에서도 동일한 API 작업 및 기능을 사용할 수 있습니다. AWS Management Console, AWS CLI, AWS SDK 또는 REST API를 통해 S3 on Outposts를 사용할 수 있습니다.

S3 on Outposts는 새로운 스토리지 클래스인 S3 Outposts(OUTPOSTS)를 제공합니다. S3 Outposts 스토리지 클래스는 Outposts의 버킷에 저장된 객체에만 사용할 수 있습니다. AWS 리전의 S3 버킷과 함께 이 스토리지 클래스를 사용하려고 하면 `InvalidStorageClass` 오류가 발생합니다. 또한 S3 on Outposts 버킷에 저장된 객체와 함께 다른 S3 스토리지 클래스를 사용하려고 하면 동일한 오류가 나타납니다.

S3 Outposts(OUTPOSTS) 스토리지 클래스에 저장된 객체는 항상 Amazon S3 관리형 암호화 키(SSE-S3)와 함께 서버 측 암호화를 사용하여 암호화됩니다. 자세한 내용은 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) 사용 섹션을 참조하세요.

또한 고객 제공 암호화 키(SSE-C)와 함께 서버 측 암호화를 사용하여 S3 Outposts 스토리지 클래스에 저장된 객체를 암호화하도록 명시적으로 선택할 수도 있습니다. 자세한 내용은 고객 제공 키(SSE-C)로 서버 측 암호화 사용 섹션을 참조하세요.

> **Note**
>
> S3 on Outposts는 AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 사용한 서버 측 암호화를 지원하지 않습니다.

S3 on Outposts에 대한 자세한 내용은 Amazon S3 on Outposts 사용 설명서의 S3 on Outposts란을 참조하세요.

Amazon S3 스토리지 클래스 비교

다음 표에서는 가용성, 내구성, 최소 스토리지 기간, 기타 고려 사항 등으로 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 Glacier Flexible Retrieval에는 아카이빙된 각 객체에 대해 40KB의 추가 메타데이터가 필요합니다. 여기에는 S3 Glacier Flexible Retrieval 요금(데이터 식별 및 검색에 필요)으로 청구되는 32KB의 메타데이터와 S3 Standard 요금으로 청구되는 추가 8KB의 데이터가 포함됩니다. S3 Glacier Flexible

Retrieval에 아카이빙된 객체의 사용자 정의 이름 및 메타데이터를 유지하려면 S3 Standard 요금이 필요합니다. 스토리지 클래스에 대한 자세한 내용은 Amazon S3 스토리지 클래스를 참조하세요.

** S3 Glacier Deep Archive에는 아카이빙된 각 객체에 대해 40KB의 추가 메타데이터가 필요합니다. 여기에는 S3 Glacier Deep Archive 요금(데이터 식별 및 검색에 필요)으로 청구되는 32KB의 메타데이터와 S3 Standard 요금으로 청구되는 추가 8KB의 데이터가 포함됩니다. Amazon S3 Glacier Deep Archive에 아카이빙된 객체의 사용자 정의 이름 및 메타데이터를 유지하려면 S3 Standard 요금이 필요합니다. 스토리지 클래스에 대한 자세한 내용은 Amazon S3 스토리지 클래스를 참조하세요.

S3 One Zone-IA(`ONEZONE_IA`) 및 S3 Express One Zone(`EXPRESS_ONEZONE`)을 제외한 모든 스토리지 클래스는 재해로 인한 가용 영역의 물리적 손실로부터 신속하게 복원할 수 있도록 설계되었다는 점을 유의하세요. 또한 애플리케이션 시나리오의 성능 요구 사항과 더불어 가격을 고려하세요. 스토리지 클래스의 요금에 대해서는 Amazon S3 요금을 참조하세요.

객체의 스토리지 클래스 설정

객체를 업로드할 때 객체의 스토리지 클래스를 지정할 수 있습니다. 지정하지 않는 경우 Amazon S3는 범용 버킷의 객체에 기본 Amazon S3 Standard 스토리지 클래스를 사용합니다. Amazon S3 콘솔, AWS SDK 또는 AWS Command Line Interface(AWS CLI)를 사용하여 Amazon S3 범용 버킷에 이미 저장된 객체의 스토리지 클래스를 다른 스토리지 클래스로 변경할 수도 있습니다. 이러한 모든 접근 방식은 Amazon S3 API 작업을 사용하여 Amazon S3에 요청을 전송합니다.

> **Note**
>
> 디렉터리 버킷에 저장된 객체의 스토리지 클래스는 변경할 수 없습니다.

하나의 버킷에 S3 수명 주기 구성을 추가하면 Amazon S3가 객체의 스토리지 클래스를 자동으로 변경하도록 지시할 수 있습니다. 자세한 내용은 객체 수명 주기 관리 섹션을 참조하세요.

S3 복제 구성을 설정할 때 복제된 객체의 스토리지 클래스를 다른 스토리지 클래스로 설정할 수 있습니다. 다만 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스에 저장된 객체는 복제할 수 없습니다. 자세한 내용은 복제 구성 파일 요소 섹션을 참조하세요.

프로그래밍 방식으로 스토리지 클래스를 설정할 때는 스토리지 클래스의 값을 제공합니다. 다음은 스토리지 클래스의 콘솔 이름과 해당 API 값의 목록입니다.

- Reduced Redundancy Storage – `REDUCED_REDUNDANCY`
- S3 Express One Zone – `EXPRESS_ONEZONE`

- S3 Glacier Deep Archive – `DEEP_ARCHIVE`
- S3 Glacier Flexible Retrieval – `GLACIER`
- S3 Glacier Instant Retrieval – `GLACIER_IR`
- S3 Intelligent-Tiering – `INTELLIGENT_TIERING`
- S3 One Zone-IA – `ONEZONE_IA`
- S3 Standard – `STANDARD`
- S3 Standard-IA – `STANDARD_IA`

새 객체에 대한 스토리지 클래스 설정

객체를 업로드할 때 스토리지 클래스를 설정하려면 다음 메서드를 사용하면 됩니다.

S3 콘솔 사용

콘솔에서 새 객체를 업로드할 때 스토리지 클래스를 설정하려면:

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 폴더 또는 파일을 업로드할 버킷 이름을 선택합니다.
4. 업로드를 선택합니다.
5. 업로드 창에서 속성을 선택합니다.
6. 스토리지 클래스에서 업로드하려는 파일의 스토리지 클래스를 선택합니다.
7. (선택 사항) 업로드하는 파일의 추가 속성을 구성합니다. 자세한 내용은 객체 업로드 섹션을 참조하세요.
8. 업로드 창에서 다음 중 하나를 수행합니다.
   - 파일 및 폴더를 업로드 창으로 끕니다.
   - 파일 추가 또는 폴더 추가를 선택하고 업로드할 파일 또는 폴더를 선택한 후 열기를 선택합니다.
9. 페이지 하단에서 업로드를 선택합니다.

REST API 사용

PutObject, POST Object Object 및 CreateMultipartUpload API 작업을 사용하여 객체를 만들 때 x-amz-storage-class 요청 헤더를 추가하여 객체의 스토리지 클래스를 지정할 수 있습니다. 이 헤더를 추가하지 않으면 Amazon S3는 기본 S3 Standard(STANDARD) 스토리지 클래스를 사용합니다.

이 예제 요청은 PutObject 명령을 사용하여 새 객체의 스토리지 클래스를 S3 Intelligent-Tiering으로 설정합니다.

```
PUT /my-image.jpg HTTP/1.1
Host: amzn-s3-demo-bucket1.s3.Region.amazonaws.com
Date: Wed, 12 Oct 2009 17:50:00 GMT
Authorization: authorization string
Content-Type: image/jpeg
Content-Length: 11434
Expect: 100-continue
x-amz-storage-class: INTELLIGENT_TIERING
```

AWS CLI 사용

이 예제에서는 put-object 명령을 사용하여 *my_images.tar.bz2*를 *GLACIER* 스토리지 클래스의 *amzn-s3-demo-bucket1*에 업로드합니다.

```
aws s3api put-object --bucket amzn-s3-demo-bucket1 --key dir-1/my_images.tar.bz2 --
storage-class GLACIER --body my_images.tar.bz2
```

객체 크기가 5GB를 초과하는 경우 다음 명령을 사용하여 스토리지 클래스를 설정합니다.

```
aws s3 cp large_test_file s3://amzn-s3-demo-bucket1 --storage-class GLACIER
```

기존 객체의 스토리지 클래스 변경

객체를 업로드할 때 스토리지 클래스를 설정하려면 다음 메서드를 사용하면 됩니다.

S3 콘솔 사용

객체 크기가 5GB 미만인 경우 Amazon S3 콘솔을 사용하여 객체의 스토리지 클래스를 변경할 수 있습니다. 크기가 더 큰 경우 S3 수명 주기 구성을 추가하여 객체의 스토리지 클래스를 변경하는 것이 좋습니다.

콘솔에서 객체의 스토리지 클래스를 변경하려면:

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 변경할 객체가 들어 있는 버킷 이름을 선택합니다.
4. 변경할 객체의 이름 왼쪽에 있는 확인란을 선택합니다.
5. 작업 메뉴에 표시되는 옵션 목록에서 스토리지 클래스 편집을 선택합니다.
6. 객체에 사용할 수 있는 스토리지 클래스 중에서 선택합니다.
7. 추가 복사 설정에서 소스 설정 복사, 설정 지정 안 함 또는 설정 지정 중 원하는 것을 선택합니다. 소스 설정 복사가 기본 옵션입니다. 소스 설정 속성 없이 객체만 복사하려면 설정 지정 안 함을 선택합니다. 스토리지 클래스, ACL, 객체 태그, 메타데이터, 서버 측 암호화 및 추가 체크섬에 대한 설정을 지정하려면 설정 지정을 선택합니다.
8. 오른쪽 하단 모서리에서 변경 사항 저장을 선택합니다. Amazon S3는 변경 사항을 저장합니다.

REST API 사용

기존 객체의 스토리지 클래스를 변경하려면 다음 메서드를 사용합니다.

이 예제 요청은 `PutObject` 명령을 사용하여 기존 객체의 스토리지 클래스를 S3 Intelligent-Tiering으로 설정합니다.

```
PUT /my-image.jpg HTTP/1.1
Host: amzn-s3-demo-bucket1.s3.Region.amazonaws.com
Date: Wed, 12 Oct 2009 17:50:00 GMT
Authorization: authorization string
Content-Type: image/jpeg
Content-Length: 11434
Expect: 100-continue
x-amz-storage-class: INTELLIGENT_TIERING
```

AWS CLI 사용

이 예시에서는 cp 명령을 사용하여 기존 객체의 스토리지 클래스를 현재 스토리지 클래스에서 *DEEP_ARCHIVE* 스토리지 클래스로 변경합니다.

```
aws s3 cp object_S3_URI object_S3_URI --storage-class DEEP_ARCHIVE
```

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` 권한을 부여하면 객체 업로드를 특정 스토리지 클래스로 제한할 수 있습니다. 정책 예제는 예: 특정 스토리지 클래스를 이용한 객체 업로드 제한을 참조하세요.

정책의 조건 및 Amazon S3 조건 키의 전체 목록을 사용하는 방법에 대한 자세한 내용은 다음 주제를 참조하세요.

- 서비스 승인 참조의 Amazon S3에 사용되는 작업, 리소스 및 조건 키

  S3 리소스 유형별 S3 API 작업 권한에 대한 자세한 내용은 Amazon S3 API 작업에 필요한 권한 섹션을 참조하세요.
- 조건 키를 사용한 버킷 정책 예시

Amazon S3 분석 - 스토리지 클래스 분석

Amazon S3 분석 스토리지 클래스 분석을 이용하면 스토리지 액세스 패턴을 분석해 올바른 데이터를 올바른 스토리지 클래스로 옮길 시점을 결정할 수 있습니다. 이 새로운 Amazon S3 분석 기능은 데이터 액세스 패턴을 관찰해 자주 액세스하지 않는 STANDARD 스토리지를 STANDARD_IA(IA는 자주 액세스하지 않는다는 뜻임) 스토리지 클래스로 옮길 시점을 알려줍니다. 스토리지 클래스에 대한 자세한 정보는 Amazon S3 스토리지 클래스 이해 및 관리를 참조하세요.

스토리지 클래스 분석 결과 필터링 데이터 세트가 일정 시간 동안 액세스 빈도가 떨어지는 패턴을 보인다면, 분석 결과를 이용해 수명 주기 구성을 개선할 수 있습니다. 버킷에 있는 모든 객체를 분석하도록 스토리지 클래스 분석을 구성할 수도 있습니다. 또는 분석을 위해 공통 접두사(이름이 공통 문자열

로 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 Standard - Standard IA 클래스에 대한 권장 사항만 제공합니다.

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 구성을 개선할 수 있습니다.

스토리지 클래스 분석은 매일 업데이트되는 스토리지 사용 시각화를 Amazon S3 콘솔에 표시합니다. 또한 이 일일 사용 데이터를 S3 버킷으로 내보내고 스프레드시트 애플리케이션이나 QuickSight와 같은 비즈니스 인텔리전스 도구를 사용하여 볼 수도 있습니다.

스토리지 클래스 분석과 관련된 비용이 있습니다. 요금 정보는 Amazon S3 요금의 관리 및 인사이트를 참조하세요.

주제

- 스토리지 클래스 분석을 설정하려면 어떻게 해야 합니까?
- 스토리지 클래스 분석을 사용하려면 어떻게 해야 합니까?
- 스토리지 클래스 분석 데이터를 내보내려면 어떻게 해야 합니까?
- 스토리지 클래스 분석 구성

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 분석 방법에 대한 자세한 내용은 스토리지 클래스 분석 구성 섹션을 참조하세요.
- Amazon S3 API를 사용하려면, AWS CLI 또는 AWS SDK에서 PutBucketAnalyticsConfiguration REST API 또는 이와 동등한 기능을 이용하세요.

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 처음 구성하면 Amazon S3 콘솔에서 데이터를 분석하는 데 시간이 걸릴 수 있습니다.

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 분석이 완료되었다는 메시지가 표시됩니다.

자주 액세스하지 않는 객체를 분석할 때, 스토리지 클래스는 Amazon S3에 업로드한 시점 이후의 연령을 기준으로 그룹화된 필터링 객체 세트를 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 지표에 표시된 수치 또는 자체 내부 시스템에 의해 추적된 수치와 차이가 날 수 있습니다.
- 실패한 GET 및 PUT 요청은 분석에 포함되지 않습니다. 하지만 스토리지 지표에는 실패한 요청이 표시됩니다.

내 스토리지에서 검색한 내용이 얼마나 되는지 확인

Amazon S3 콘솔은 관찰 기간에 필터링 데이터 세트의 서 스토리지에서 검색한 내용이 얼마나 되는지 그래프로 표시합니다.

내 스토리지에서 검색한 내용의 비율을 어떻게 확인합니까?

Amazon S3 콘솔은 관찰 기간에 필터링 데이터 세트의 스토리지에서 검색한 내용의 비율도 그래프로 표시합니다.

이 주제에서 앞서 언급했듯이, 자주 액세스하지 않는 객체를 분석할 때 스토리지 클래스는 Amazon S3에 업로드한 시점 이후의 연령을 기준으로 그룹화된 필터링 객체 세트를 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 나

면, 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

Amazon S3 콘솔은 사전 정의된 객체 연령 그룹을 기준으로 그룹화된 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 사용할 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 버킷에 생성해야 합니다. 정책에 대한 예는 S3 인벤토리 및 S3 분석 권한 부여을(를) 참조하세요.

스토리지 클래스 분석 보고서를 구성한 후 24시간이 지나면, 내보낸 보고서를 매일 받게 됩니다. 이후에는 Amazon S3가 모니터링을 계속 실시하고 매일 내보내기를 수행합니다.

CSV 파일은 스프레드시트 애플리케이션에서 열거나 Amazon QuickSight 같은 다른 애플리케이션에서 가져올 수 있습니다. Amazon QuickSight에서 &S3; 파일을 이용하는 방법은 Amazon QuickSight 사용 설명서의 Amazon S3 파일을 사용한 데이터 세트 생성 섹션을 참조하세요.

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
8/17/2021	SalesMaterial	SalesMaterial	STANDARD	000-014			0.4313			0		
9/2/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		
8/22/2021	SalesMaterial	SalesMaterial	STANDARD	000-014			0.4313			0		
8/25/2021	SalesMaterial	SalesMaterial	STANDARD	000-014			0.4313			0		
9/6/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		
8/30/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		
8/28/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		
8/21/2021	SalesMaterial	SalesMaterial	STANDARD	000-014			0.4313			0		
9/5/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		

보고서 끝에 있는 객체 연령 그룹은 ALL로 제공됩니다. ALL 행에는 128KB보다 작은 객체를 포함하여 해당 날짜의 모든 연령 그룹에 대한 누적 합계가 포함됩니다.

8/24/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599		0	000-014
9/3/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599		0.02426125	015-029
8/28/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599		0.03545875	015-029
8/17/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599		0	000-014
8/25/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599		0	000-014
9/6/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599		0.0209529	015-029
9/4/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599		0.02304819	015-029
8/22/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599		0	000-014
8/21/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599		0	000-014
8/30/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599		0.03073092	015-029
8/20/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599		0	000-014

다음 섹션에서는 보고서에서 사용한 행을 설명합니다.

내보낸 파일 레이아웃

다음 테이블에서는 Amazon S3 스토리지 클래스 분석 내보내기 파일 레이아웃을 설명합니다.

스토리지 클래스 분석 구성

Amazon S3 분석 스토리지 클래스 분석 도구를 사용하면 스토리지 액세스 패턴을 분석할 수 있어 적합한 데이터를 적절한 스토리지 클래스로 전환할 시기를 결정하는 데 도움이 됩니다. 스토리지 클래스 분석은 데이터 액세스 패턴을 관찰해 자주 액세스하지 않는 STANDARD 스토리지를 STANDARD_IA (IA는 자주 액세스하지 않는다는 뜻입니다) 스토리지 클래스로 옮길 시점을 알려줍니다. STANDARD_IA에 대한 자세한 내용은 Amazon S3 FAQ와 Amazon S3 스토리지 클래스 이해 및 관리 섹션을 참조하세요.

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.

> **Important**
>
> 스토리지 클래스 분석은 ONEZONE_IA 또는 S3 Glacier Flexible Retrieval 스토리지 클래스로의 전환에는 권장되지 않습니다.
> 검색 결과를 .csv 파일로 내보내도록 스토리지 클래스 분석을 구성하고 대상 버킷에서 AWS KMS key와 함께 기본 버킷 암호화를 사용하는 경우, AWS KMS 키 정책을 업데이트하여 Amazon S3에 .csv 파일을 암호화할 권한을 부여해야 합니다. 지침은 암호화에 고객 관리형 키를 사용하도록 Amazon S3에 권한 부여 섹션을 참조하세요.

분석에 대한 자세한 내용은 Amazon S3 분석 - 스토리지 클래스 분석 섹션을 참조하세요.

S3 콘솔 사용

스토리지 클래스 분석 구성

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷 또는 디렉터리 버킷을 선택합니다.
3. 버킷(Buckets) 목록에서 스토리지 클래스 분석을 구성할 버킷 이름을 선택합니다.
4. 지표 탭을 선택합니다.
5. 스토리지 클래스 분석(Storage Class Analysis)에서 분석 구성 생성(Create analytics configuration)을 선택합니다.
6. 필터 이름을 입력합니다. 버킷 전체를 분석하려면 접두사(Prefix) 필드에 아무것도 입력하지 마세요.
7. 접두사(Prefix) 필드에 분석할 객체의 접두사 텍스트를 입력합니다.
8. 태그를 추가하려면 태그 추가를 선택합니다. 해당 태그의 키와 값을 입력합니다. 접두사 한 개와 태그 여러 개를 입력할 수 있습니다.
9. 선택적으로 CSV 내보내기(Export CSV)에서 사용(Enable)을 선택하여 분석 보고서를 쉼표로 구분된 값(.csv) 플랫 파일로 내보낼 수 있습니다. 파일을 저장할 수 있는 대상 버킷을 선택합니다. 대상 버킷의 접두사를 입력하면 됩니다. 대상 버킷은 분석 대상 버킷과 같은 AWS 리전에 있어야 합니다. 대상 버킷은 다른 AWS 계정에 있을 수 있습니다.

.csv 파일의 대상 버킷에서 KMS 키와 함께 기본 버킷 암호화를 사용하는 경우, AWS KMS 키 정책을 업데이트하여 Amazon S3에 .csv 파일을 암호화할 권한을 부여해야 합니다. 지침은 암호화에 고객 관리형 키를 사용하도록 Amazon S3에 권한 부여 섹션을 참조하세요.

10. 구성 생성(Create Configuration)을 선택합니다.

Amazon S3는 대상 버킷에서 Amazon S3 쓰기 권한을 부여하는 버킷 정책을 생성합니다. 이렇게 하면 내보내기 데이터를 해당 버킷에 쓸 수 있습니다.

버킷 정책을 생성하는 동안 오류가 발생하는 경우, 해결 지침이 제시됩니다. 예를 들어, 다른 AWS 계정의 대상 버킷을 선택하는 바람에 해당 버킷 정책에 대한 읽기 및 쓰기 권한이 없는 경우, 다음과 같은 메시지가 나타납니다. 대상 버킷 소유자가 표시된 버킷 정책을 대상 버킷에 추가해 주어야만 합니다. Amazon S3은 대상 버킷에 대한 쓰기 권한이 없기 때문에, 이 정책을 대상 버킷에 추가하지 않으면 내보내기 데이터를 받을 수 없게 됩니다. 소스 버킷이 현재 사용자가 아닌 다른 계정의 소유물인 경우, 정책에서 소스 버킷의 올바른 계정 ID로 바꿔야 합니다.

내보낸 데이터와 필터 작동 방식에 대한 자세한 내용은 Amazon S3 분석 - 스토리지 클래스 분석 섹션을 참조하세요.

REST API 사용

REST API를 사용하여 스토리지 클래스 분석을 구성하려면 PutBucketAnalyticsConfiguration을 사용합니다. AWS CLI 또는 AWS SDK를 사용하여 동일한 작업을 수행할 수도 있습니다.

다음 REST API를 사용하여 스토리지 클래스 분석 작업을 수행할 수 있습니다.

- DELETE Bucket 분석 구성
- GET Bucket 분석 구성
- List Bucket 분석 구성

Amazon S3 Intelligent-Tiering을 사용한 스토리지 비용 관리

S3 Intelligent-Tiering 스토리지 클래스는 액세스 패턴이 변경될 때 운영 오버헤드나 성능에 대한 영향 없이 데이터를 가장 비용 효율적인 액세스 계층으로 자동 이동하여 스토리지 비용을 최적화하도록 설계되었습니다. 약간의 월별 객체 모니터링 및 자동화 요금만 지불하면 S3 Intelligent-Tiering에서 액세스 패턴을 모니터링하고, 액세스되지 않은 객체를 저렴한 액세스 계층으로 자동으로 이동합니다.

S3 Intelligent-Tiering을 통해 대기 시간이 짧고 처리량이 높은 세 액세스 계층에서 스토리지 비용을 자동으로 절감할 수 있습니다. 비동기식으로 액세스할 수 있는 데이터의 경우 S3 Intelligent-Tiering 스

토리지 클래스 내에서 자동 아카이브 기능을 활성화하도록 선택할 수 있습니다. S3 Intelligent-Tiering에는 검색 요금이 없습니다. 나중에 Infrequent Access 계층 또는 Archive Instant Access 계층의 객체에 액세스하면 이 객체는 자동으로 Frequent Access 계층으로 다시 이동합니다. 객체가 S3 Intelligent-Tiering 스토리지 클래스 내 액세스 계층 간에 이동될 때는 계층화 요금이 추가로 적용되지 않습니다.

S3 Intelligent-Tiering은 데이터 레이크, 데이터 분석 및 새 애플리케이션과 같이 객체 크기나 보존 기간에 관계없이 알 수 없거나 변경되거나 예측할 수 없는 액세스 패턴을 가진 데이터에 권장되는 스토리지 클래스입니다.

S3 Intelligent-Tiering 스토리지 클래스는 다음을 포함한 모든 Amazon S3 기능을 지원합니다.

- S3 인벤토리, 객체의 액세스 계층 확인용
- S3 복제, 모든 AWS 리전에 데이터 복제용
- S3 Storage Lens, 스토리지 사용량 및 활동 지표 보기용
- 서버 측 암호화, 객체 데이터 보호용
- S3 객체 잠금, 우발적인 데이터 삭제 방지용
- AWS PrivateLink, Virtual Private Cloud(VPC)의 프라이빗 엔드포인트를 통한 Amazon S3 액세스용

S3 Intelligent-Tiering 사용에 대한 자세한 내용은 다음 섹션을 참조하세요.

주제

- S3 Intelligent-Tiering 작동 방식
- S3 Intelligent-Tiering 사용
- S3 Intelligent-Tiering 관리

S3 Intelligent-Tiering 작동 방식

Amazon S3 Intelligent-Tiering 스토리지 클래스는 3개의 액세스 계층에 객체를 자동으로 저장합니다. 한 계층은 빈도가 높은 액세스에 최적화되어 있고, 하나의 저렴한 계층은 빈도가 낮은 액세스에 최적화되어 있고, 또 다른 매우 저렴한 계층은 거의 액세스하지 않는 데이터에 최적화되어 있습니다. S3 Intelligent-Tiering은 저렴한 월별 객체 모니터링 및 자동화 요금으로 액세스 패턴을 모니터링하고 연속 30일 동안 액세스하지 않은 객체를 Infrequent Access 계층으로 자동 이동합니다. 90일 동안 액세스한 적이 없으면 성능 영향이나 운영 오버헤드 없이 객체가 Archive Instant Access 계층으로 이동됩니다.

몇 분에서 몇 시간 내에 액세스할 수 있는 데이터에 대한 스토리지 비용을 최소화하려면 아카이빙 기능을 활성화하여 액세스 계층을 두 개 추가하세요. 객체의 계층을 Archive Access 계층, Deep Archive

Access 계층 또는 둘 다로 낮출 수 있습니다. Archive Access를 사용하면 S3 Intelligent-Tiering이 최소 연속 90일 동안 액세스하지 않은 객체를 Archive Access 계층으로 옮깁니다. Archive Access를 사용하면 S3 Intelligent-Tiering이 최소 연속 180일 동안 액세스하지 않은 객체를 Deep Archive Access 계층으로 옮깁니다. 두 계층 모두에 대해 필요에 따라 액세스되지 않아야 하는 기간(일)을 구성할 수 있습니다.

다음 작업은 객체를 Archive Access 계층 또는 Deep Archive Access 계층으로 계층화하지 못하게 하는 액세스에 해당합니다.

- Amazon S3 콘솔을 통해 객체 다운로드 또는 복사
- CopyObject, UploadPartCopy를 호출하거나 S3 Batch Replication을 사용하여 객체 복제. 이러한 경우 복사 또는 복제 작업의 소스 객체가 계층화됩니다.
- GetObject, PutObject, RestoreObject, CompleteMultipartUpload, ListParts 또는 SelectObjectContent 호출

예를 들어, 지정한 일수(예: 180일)가 지나기 전에 `SelectObjectContent`를 통해 객체에 액세스하는 경우 해당 작업으로 타이머가 재설정됩니다. 마지막 `SelectObjectContent` 요청이 지정한 기간(일)에 도달할 때까지 객체는 Archive Access 계층 또는 Deep Archive Access 계층으로 이동하지 않습니다.

나중에 Infrequent Access 계층 또는 Archive Instant Access 계층의 객체에 액세스하면 이 객체는 자동으로 Frequent Access 계층으로 다시 이동합니다.

다음 작업은 Infrequent Access 계층 또는 Archive Instant Access 계층에서 Frequent Access 계층으로 객체를 자동으로 이동하는 액세스를 구성합니다.

- Amazon S3 콘솔을 통해 객체 다운로드 또는 복사
- CopyObject, UploadPartCopy를 호출하거나 Batch Replication을 사용하여 객체 복제. 이러한 경우 복사 또는 복제 작업의 소스 객체가 계층화됩니다.
- GetObject, PutObject, RestoreObject, CompleteMultipartUpload 또는 ListParts 호출

이외 작업은 Infrequent Access 계층 또는 Archive Instant Access 계층에서 Frequent Access 계층으로 객체를 자동으로 이동하는 액세스를 구성하지 않습니다. 다음은 이러한 작업의 전체 목록이 아니라 샘플입니다.

- HeadObject, GetObjectTagging, PutObjectTagging, ListObjects, ListObjectsV2 또는 ListObjectVersions 호출

- SelectObjectContent를 호출해도 객체를 자주 액세스 계층까지 계층화하는 액세스는 구성되지 않음. 또한 Frequent Access 계층에서 Infrequent Access 계층으로 이어서 Archive Instant Access 계층으로 객체를 계층화하지 못하도록 막지는 않습니다.

PutObject, CopyObject 또는 CreateMultipartUpload 작업을 호출할 때 x-amz-storage-class 요청 헤더에 INTELLIGENT-TIERING을 지정하여 새로 생성된 데이터에 대한 기본 스토리지 클래스로 S3 Intelligent-Tiering을 사용할 수 있습니다. S3 Intelligent-Tiering은 99.9% 의 가용성과 99.9999999%의 내구성을 제공하도록 설계되었습니다.

> **Note**
>
> 객체의 크기가 128KB 미만이면 자동 계층화를 모니터링 및 사용할 수 없습니다. 작은 객체는 항상 Frequent Access 계층에 저장됩니다.

S3 Intelligent-Tiering 액세스 계층

다음 섹션에서는 다양한 자동 및 선택적 액세스 계층에 대해 설명합니다. 객체가 액세스 계층 간에 이동할 때 스토리지 클래스는 동일하게 유지됩니다(S3 Intelligent-Tiering).

Frequent Access 계층(자동)

S3 Intelligent-Tiering에서 생성하거나 전환한 객체가 수명 주기를 시작하는 기본 액세스 계층입니다. 객체가 액세스되는 동안 이 계층에 남아 있습니다. Frequent Access 계층은 짧은 지연 시간 및 높은 처리량 성능을 제공합니다.

Infrequent Access 계층(자동)

30일 연속으로 객체에 액세스하지 않으면 객체가 Infrequent Access 계층으로 이동합니다. Infrequent Access 계층은 짧은 지연 시간 및 높은 처리량 성능을 제공합니다.

Archive Instant Access 계층(자동)

90일 연속으로 객체에 액세스하지 않으면 객체가 Archive Instant Access 계층으로 이동합니다. Archive Instant Access 계층은 짧은 지연 시간과 높은 처리량 성능을 제공합니다.

Archive Access 계층(선택 사항)

S3 Intelligent-Tiering은 비동기식으로 액세스할 수 있는 데이터에 대해 Archive Access 계층을 활성화하는 옵션을 제공합니다. 활성화 후 Archive Access 계층은 최소 연속 90일 동안 액세스하지 않은 객체를 자동으로 아카이브합니다. 아카이브에 대한 마지막 액세스 시간을 최대 730일로 연장할

수 있습니다. Archive Access 계층의 성능은 S3 Glacier Flexible Retrieval 스토리지 클래스와 동일합니다.

이 액세스 계층의 표준 검색 시간은 3~5시간입니다. S3 배치 작업을 사용하여 복원 요청을 시작하는 경우 몇 분 내에 복원이 시작됩니다. 검색 옵션 및 시간에 대한 자세한 내용은 the section called "S3 Intelligent-Tiering Archive Access 및 Deep Archive Access 계층에서 객체 복원" 섹션을 참조하세요.

> **Note**
>
> Archive Instant Access 계층을 우회하려는 경우에만 90일 동안 Archive Access 계층을 활성화합니다. Archive Access 계층은 몇 분에서 몇 시간의 검색 시간과 함께 약간 더 저렴한 스토리지 비용을 제공합니다. Archive Instant Access 계층은 밀리초 단위의 액세스와 높은 처리량 성능을 제공합니다.

Deep Archive Access 계층(선택 사항)

S3 Intelligent-Tiering은 비동기식으로 액세스할 수 있는 데이터에 대해 Deep Archive Access 계층을 활성화하는 옵션을 제공합니다. 활성화 후 Deep Archive Access 계층은 최소 연속 180일 동안 액세스하지 않은 객체를 자동으로 아카이브합니다. 아카이브에 대한 마지막 액세스 시간을 최대 730일로 연장할 수 있습니다. Deep Archive Access 계층의 성능은 S3 Glacier Deep Archive 스토리지 클래스와 동일합니다.

이 액세스 계층에 있는 객체의 표준 검색은 12시간 이내에 발생합니다. S3 배치 작업을 사용하여 복원 요청을 시작하는 경우 9시간 내에 복원이 시작됩니다. 검색 옵션 및 시간에 대한 자세한 내용은 the section called "S3 Intelligent-Tiering Archive Access 및 Deep Archive Access 계층에서 객체 복원" 섹션을 참조하세요.

> **Note**
>
> 애플리케이션에서 객체에 비동기적으로 액세스할 수 있는 경우에만 Archive Access 및 Deep Archive Access 계층을 활성화합니다. 검색 중인 객체가 Archive Access 또는 Deep Archive Access 계층에 저장되어 있는 경우 먼저 `RestoreObject` 작업을 사용하여 객체를 복원해야 합니다.

S3 Intelligent-Tiering 사용

S3 Intelligent-Tiering 스토리지 클래스를 사용하여 스토리지 비용을 자동으로 최적화할 수 있습니다. S3 Intelligent-Tiering은 액세스 패턴이 변화할 때 액세스 계층 사이에서 세분화된 객체 수준으로 데이터를 이동하여 자동 비용 절감 효과를 제공합니다. 비동기식으로 액세스할 수 있는 데이터의 경우 AWS Management Console, AWS CLI 또는 Amazon S3 API를 사용하여 S3 Intelligent-Tiering 스토리지 클래스 내에서 자동 아카이브 기능을 사용 설정하도록 선택할 수 있습니다.

S3 Intelligent-Tiering으로 데이터 이동

S3 Intelligent-Tiering에는 두 가지의 데이터 이동 방법이 있습니다. 콘솔에서 S3 Intelligent-Tiering으로 객체를 직접 업로드하거나 PUT 작업을 사용하여 프로그래밍 방식으로 업로드할 수 있습니다. 자세한 내용은 객체의 스토리지 클래스 설정 단원을 참조하십시오. 또한, S3 수명 주기 구성을 구성하여 객체를 S3 Standard 또는 S3 Standard-Infrequent Access에서 S3 Intelligent-Tiering으로 이전할 수도 있습니다.

Direct PUT을 사용하여 S3 Intelligent-Tiering에 데이터 업로드

PUT API 작업을 사용하여 S3 Intelligent-Tiering 스토리지 클래스에 객체를 업로드할 경우 `x-amz-storage-class` 요청 헤더에 S3 Intelligent-Tiering을 지정합니다.

다음 요청은 `myBucket` 버킷에 이미지 `my-image.jpg`를 저장합니다. 요청은 `x-amz-storage-class` 헤더를 사용하여 객체가 S3 Intelligent-Tiering 스토리지 클래스를 사용하여 저장되도록 요청합니다.

Example

```
PUT /my-image.jpg HTTP/1.1
Host: myBucket.s3.<Region>.amazonaws.com (http://amazonaws.com/)
Date: Wed, 1 Sep 2021 17:50:00 GMT
Authorization: authorization string
Content-Type: image/jpeg
Content-Length: 11434
Expect: 100-continue
x-amz-storage-class: INTELLIGENT_TIERING
```

S3 수명 주기를 사용하여 S3 Standard 또는 S3 Standard - Infrequent Access에서 S3 Intelligent-Tiering으로 데이터 전환

S3 수명 주기 구성에 규칙을 추가하여 Amazon S3이 객체를 하나의 스토리지 클래스에서 다른 스토리지 클래스로 전환하도록 유도할 수 있습니다. 지원되는 전환 및 관련 제약 조건에 대한 자세한 내용은 S3 수명 주기를 사용하여 객체 전환을 참조하세요.

버킷 또는 접두사 수준에서 S3 수명 주기 구성을 지정할 수 있습니다. 이 S3 수명 주기 구성 규칙에서 필터는 키 접두사(`documents/`)를 지정합니다. 따라서 이 규칙은 `documents/` 및 `documents/doc1.txt`와 같은 키 이름 접두사 `documents/doc2.txt`가 있는 객체에 적용됩니다. 이 규칙은 생성된 지 0일 후에 객체를 S3 Intelligent-Tiering 스토리지 클래스로 전환하도록 Amazon S3에 지시하는 `Transition` 작업을 지정합니다. 이 경우 객체는 생성 후 자정 UTC에 S3 Intelligent-Tiering으로 전환할 수 있습니다.

Example

```
<LifecycleConfiguration>
  <Rule>
    <ID>ExampleRule</ID>
    <Filter>
       <Prefix>documents/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>INTELLIGENT_TIERING</StorageClass>
    </Transition>
 </Rule>
</LifecycleConfiguration>
```

버전 관리가 사용 설정된 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 단원을 참조하십시오.

S3 Intelligent-Tiering Archive Access 및 Deep Archive Access 계층 사용

몇 분에서 몇 시간 내에 액세스할 수 있는 데이터에 대한 가장 낮은 스토리지 비용을 얻기 위해 AWS Management Console, AWS CLI 또는 Amazon S3 API로 버킷, 접두사 또는 객체 태그별 구성을 생성하여 Archive Access 계층을 하나 또는 둘 다 활성화할 수 있습니다.

S3 콘솔 사용

S3 Intelligent-Tiering 자동 아카이브 사용 설정

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷 목록에서 원하는 버킷의 이름을 선택합니다.
3. [속성(Properties)]을 선택합니다.
4. S3 Intelligent-Tiering 아카이브 구성(S3 Intelligent-Tiering Archive configurations) 섹션으로 이동하여 구성 생성(Create configuration)을 선택합니다.
5. 아카이브 구성 설정(Archive configuration settings) 섹션에서 S3 Intelligent-Tiering 아카이브 구성에 대해 설명하는 구성 이름을 지정합니다.
6. 구성 범위 선택(Choose a configuration scope)에서 사용할 구성 범위를 선택합니다. 필요한 경우 공유 접두사, 객체 태그 또는 이 둘의 조합을 사용하여 버킷 내의 지정된 객체로 구성 범위를 제한할 수 있습니다.
   a. 구성 범위를 제한하려면 하나 이상의 필터를 사용하여 이 구성 범위 제한(Limit the scope of this configuration using one or more filters)을 선택합니다.
   b. 단일 접두사를 사용하여 구성의 범위를 제한하려면 접두사(Prefix) 아래에 접두사를 입력합니다.
   c. 객체 태그를 사용하여 구성의 범위를 제한하려면 태그 추가(Add tag)를 선택하고 키 값을 입력합니다.
7. 상태(Status)에서 사용(Enable)을 선택합니다.
8. 아카이브 설정(Archive settings) 섹션에서 사용 설정할 Archive Access 계층 중 하나 또는 둘 다를 선택합니다.
9. 생성(Create)을 선택합니다.

AWS CLI 사용

다음 AWS CLI 명령을 사용하여 S3 Intelligent-Tiering 구성을 관리할 수 있습니다.

- `delete-bucket-intelligent-tiering-configuration`
- `get-bucket-intelligent-tiering-configuration`
- `list-bucket-intelligent-tiering-configurations`
- `put-bucket-intelligent-tiering-configuration`

AWS CLI 설정에 대한 지침은 Amazon S3 API 참조의 AWS CLI를 사용하여 Amazon S3에서 개발을 참조하세요.

AWS CLI를 사용할 경우 구성을 XML 파일로 지정할 수 없습니다. 대신 JSON을 지정해야 합니다. 다음은 XML S3 Intelligent-Tiering 구성과 AWS CLI 명령에서 지정할 수 있는 그에 상당하는 JSON의 예입니다.

다음 예제에서는 지정된 버킷에 S3 Intelligent-Tiering 구성을 배치합니다.

Example **put-bucket-intelligent-tiering-configuration**

JSON

```
{
  "Id": "string",
  "Filter": {
    "Prefix": "string",
    "Tag": {
      "Key": "string",
      "Value": "string"
    },
    "And": {
      "Prefix": "string",
      "Tags": [
        {
          "Key": "string",
          "Value": "string"
        }
        ...
      ]
    }
  },
  "Status": "Enabled"|"Disabled",
  "Tierings": [
    {
      "Days": integer,
      "AccessTier": "ARCHIVE_ACCESS"|"DEEP_ARCHIVE_ACCESS"
    }
    ...
  ]
}
```

XML

```
PUT /?intelligent-tiering&id=Id HTTP/1.1
Host: Bucket.s3.amazonaws.com
<?xml version="1.0" encoding="UTF-8"?>
<IntelligentTieringConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Id>string</Id>
   <Filter>
      <And>
         <Prefix>string</Prefix>
         <Tag>
            <Key>string</Key>
            <Value>string</Value>
         </Tag>
         ...
      </And>
      <Prefix>string</Prefix>
      <Tag>
         <Key>string</Key>
         <Value>string</Value>
      </Tag>
   </Filter>
   <Status>string</Status>
   <Tiering>
      <AccessTier>string</AccessTier>
      <Days>integer</Days>
   </Tiering>
   ...
</IntelligentTieringConfiguration>
```

PUT API 작업 사용

지정된 버킷에 대해 PutBucketIntelligentTieringConfiguration 작업을 사용하고 버킷당 최대 1,000개의 S3 Intelligent-Tiering 구성을 구성할 수 있습니다. 공유 접두사 또는 객체 태그를 사용하여 버킷 내 객체를 Archive Access 계층에 적합하게 정의할 수 있습니다. 공유 접두사 또는 객체 태그로 사용함으로써 특정한 비즈니스 애플리케이션, 워크플로우, 또는 내부 조직에 맞춰 적용할 수 있습니다. 또한 Archive Access 계층, Deep Archive Access 계층 또는 둘 다를 활성화할 수 있습니다.

S3 Intelligent-Tiering 시작하기

S3 Intelligent-Tiering을 사용하는 방법에 대한 자세한 내용은 자습서: S3 Intelligent-Tiering 사용 시작하기를 참조하세요.

S3 Intelligent-Tiering 관리

S3 Intelligent-Tiering 스토리지 클래스를 통해 지연 시간이 짧고 처리량이 높은 세 액세스 계층에서 스토리지 비용을 자동으로 절감할 수 있습니다. 또한 선택적인 아카이브 기능을 제공하여 몇 분에서 몇 시간 내에 액세스할 수 있는 데이터에 대해 클라우드에서 스토리지 비용을 최소화할 수 있습니다.

저장되어 있는 S3 Intelligent-Tiering 액세스 계층 식별

Amazon S3 인벤토리를 사용하여 S3 Intelligent-Tiering 액세스 계층을 포함한 객체 및 해당 메타데이터의 목록을 얻을 수 있습니다. S3 인벤토리는 객체와 해당 메타데이터가 나열된 CSV, ORC 또는 Parquet 출력 파일을 제공합니다. Amazon S3 버킷 또는 공유 접두사에 대해 매일 또는 매주 이러한 인벤토리 보고서를 받을 수 있습니다. (공유 접두사는 이름이 공통 문자열로 시작하는 객체를 나타냅니다.)

S3 Intelligent-Tiering 내에서 객체의 아카이브 상태 보기

S3 Intelligent-Tiering 스토리지 클래스 내의 객체가 Archive Access 계층 또는 Deep Archive Access 계층으로 이동할 때 알림을 받도록 S3 이벤트 알림을 설정할 수 있습니다. 자세한 내용은 이벤트 알림 사용을 참조하세요.

Amazon S3는 Amazon Simple Notification Service(Amazon SNS) 주제, Amazon Simple Queue Service(Amazon SQS) 대기열 또는 AWS Lambda 함수에 이벤트 알림을 게시할 수 있습니다. 자세한 내용은 Amazon S3 이벤트 알림 단원을 참조하십시오.

다음은 Amazon S3에서 `s3:IntelligentTiering` 이벤트를 게시하기 위해 전송하는 메시지의 예입니다. 자세한 내용은 the section called “이벤트 메시지 구조” 단원을 참조하십시오.

```
{
   "Records":[
      {
         "eventVersion":"2.3",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"1970-01-01T00:00:00.000Z",
         "eventName":"IntelligentTiering",
         "userIdentity":{
            "principalId":"s3.amazonaws.com"
         },
         "requestParameters":{
            "sourceIPAddress":"s3.amazonaws.com"
         },
```

```
         "responseElements":{
            "x-amz-request-id":"C3D13FE58DE4C810",
            "x-amz-id-2":"FMyUVURIY8/IgAtTv8xRjskZQpcIZ9KG4V5Wp6S7S/
JRWeUWerMUE5JgHvANOjpD"
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"testConfigRule",
            "bucket":{
               "name":"amzn-s3-demo-bucket",
               "ownerIdentity":{
                  "principalId":"A3NL1KOZZKExample"
               },
               "arn":"arn:aws:s3:::amzn-s3-demo-bucket"
            },
            "object":{
               "key":"HappyFace.jpg",
               "size":1024,
               "eTag":"d41d8cd98f00b204e9800998ecf8427e",
            }
         },
         "intelligentTieringEventData":{
            "destinationAccessTier": "ARCHIVE_ACCESS"
            }
      }
   ]
}
```

또한 HEAD 객체 요청을 사용하여 객체의 아카이브 상태를 볼 수 있습니다. 객체가 S3 Intelligent-Tiering 스토리지 클래스에 저장되고 아카이브 계층 중 하나에 있는 경우 HEAD 객체 응답에 현재 아카이브 계층이 표시됩니다. 아카이브 계층을 표시하기 위해 요청은 x-amz-archive-status 헤더를 사용합니다.

다음 HEAD 객체 요청은 객체의 메타데이터(이 예에서 *my-image.jpg*)를 반환합니다.

Example

```
HEAD /my-image.jpg HTTP/1.1
Host: bucket.s3.region.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:02236Q3V0RonhpaBX5sCYVf1bNRuU=
```

또한 HEAD 객체 요청을 사용하여 `restore-object` 요청의 상태를 모니터링할 수 있습니다. 아카이브 복원이 진행 중인 경우 HEAD 객체 응답에는 x-amz-restore 헤더가 포함됩니다.

다음 샘플 HEAD 객체 응답에서는 복원 요청이 진행 중인 S3 Intelligent-Tiering을 사용하여 아카이브된 객체를 보여줍니다.

Example

```
HTTP/1.1 200 OK
x-amz-id-2: FSVaTMjrmBp3Izs1NnwBZeu7M19iI8UbxMbi0A8AirHANJBo+hEftBuiESACOMJp
x-amz-request-id: E5CEFCB143EB505A
Date: Fri, 13 Nov 2020 00:28:38 GMT
Last-Modified: Mon, 15 Oct 2012 21:58:07 GMT
ETag: "1accb31fcf202eba0c0f41fa2f09b4d7"
x-amz-storage-class: 'INTELLIGENT_TIERING'
x-amz-archive-status: 'ARCHIVE_ACCESS'
x-amz-restore: 'ongoing-request="true"'
x-amz-restore-request-date: 'Fri, 13 Nov 2020 00:20:00 GMT'
Accept-Ranges: bytes
Content-Type: binary/octet-stream
Content-Length: 300
Server: AmazonS3
```

S3 Intelligent-Tiering Archive Access 및 Deep Archive Access 계층에서 객체 복원

S3 Intelligent-Tiering Archive Access 및 Deep Archive Access 계층의 객체에 액세스하려면 복원 요청을 시작하고 객체가 Frequent Access 계층으로 이동할 때까지 기다려야 합니다. 아카이브된 객체에 대한 자세한 정보는 the section called “아카이브된 객체 작업” 단원을 참조하세요.

Archive Access 계층 또는 Deep Archive Access 계층에서 객체를 복원하면 객체가 Frequent Access 계층으로 다시 전환됩니다. 이후 30일 연속으로 객체에 액세스하지 않으면 자동으로 Infrequent Access 계층으로 이동합니다. 액세스하지 않은 기간이 최소 연속 90일이 넘으면 Archive Access 계층으로 이동합니다. 액세스하지 않은 기간이 최소 연속 180일이 넘으면 Deep Archive Access 계층으로 이동합니다. 자세한 내용은 the section called “S3 Intelligent-Tiering 작동 방식” 단원을 참조하십시오.

Amazon S3 콘솔, S3 배치 작업, Amazon S3 REST API, AWS SDK 또는 AWS Command Line Interface(AWS CLI)를 사용하여 아카이브된 객체를 복원할 수 있습니다. 자세한 내용은 the section called “아카이브된 객체 작업” 단원을 참조하십시오.

장기 데이터 스토리지를 위한 S3 Glacier 스토리지 클래스 이해하기

Amazon S3 S3 Glacier 스토리지 클래스를 사용하여 자주 액세스하지 않는 장기 데이터를 저장하는 비용 효율적인 솔루션을 제공할 수 있습니다. S3 Glacier 스토리지 클래스는 다음과 같습니다.

- S3 Glacier Instant Retrieval
- S3 Glacier Flexible Retrieval
- S3 Glacier Deep Archive

데이터에 액세스하는 빈도와 요구되는 데이터 검색 속도에 따라 이러한 스토리지 클래스 중 하나를 선택합니다. 각 스토리지 클래스는 S3 Standard 스토리지 클래스와 동일한 내구성 및 복원성을 제공하지만 스토리지 비용은 더 낮습니다. S3 Glacier 스토리지 클래스에 대한 자세한 내용은 https://aws.amazon.com/s3/storage-classes/glacier/를 참조하세요.

주제

- S3 Glacier 스토리지 클래스 비교
- S3 Glacier Instant Retrieval
- S3 Glacier Flexible Retrieval
- S3 Glacier Deep Archive
- S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive의 아카이브 스토리지 이해
- 이러한 스토리지 클래스와 S3 Glacier 서비스의 차이점

S3 Glacier 스토리지 클래스 비교

각 S3 Glacier 스토리지 클래스에는 모든 객체의 최소 저장 기간이 있습니다. 최소 기간 전에 객체를 삭제하거나 덮어쓰거나 다른 스토리지 클래스로 이전하는 경우, 해당 기간의 나머지 기간에 대한 요금이 청구됩니다.

일부 S3 Glacier 스토리지 클래스는 아카이브형입니다. 즉, 해당 클래스에 저장된 객체는 아카이빙되며 실시간 액세스에 사용할 수 없습니다. 자세한 내용은 S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive의 아카이브 스토리지 이해 섹션을 참조하세요.

액세스 빈도가 낮고 검색 시간이 길도록 설계된 스토리지 클래스는 스토리지 비용이 낮습니다. 요금에 대한 자세한 내용은 https://aws.amazon.com/s3/pricing/ 섹션을 참조하세요.

다음 표에는 S3 Glacier 스토리지 클래스를 선택할 때 고려해야 할 주요 사항이 요약되어 있습니다.

S3 Glacier Instant Retrieval

분기당 1회 액세스되고 검색 시간이 밀리초 단위여야 하는 장기 데이터에는 S3 Glacier Instant Retrieval을 사용하는 것이 좋습니다. 이 스토리지 클래스는 이미지 호스팅, 파일 공유 애플리케이션, 진료 중 액세스를 위한 의료 기록 저장 등 성능에 민감한 사용 사례에 적합합니다.

S3 Glacier Instant Retrieval 스토리지 클래스는 S3 Standard-IA 스토리지 클래스와 동일한 지연 시간 및 처리량 성능으로 객체에 대한 실시간 액세스를 제공합니다. S3 Standard-IA와 비교하면 S3 Glacier Instant Retrieval은 스토리지 비용은 더 낮지만 데이터 액세스 비용은 더 높습니다.

S3 Glacier Instant Retrieval 스토리지 클래스에 저장된 데이터의 최소 객체 크기는 128KB입니다. 또한 이 스토리지 클래스의 최소 저장 기간은 90일입니다.

S3 Glacier Flexible Retrieval

1년에 1~2회 액세스되며 즉시 액세스할 필요가 없는 아카이브 데이터에는 S3 Glacier Flexible Retrieval을 사용하는 것이 좋습니다. S3 Glacier Flexible Retrieval은 비용 균형을 맞출 수 있도록 액세스 시간이 몇 분에서 몇 시간에 이르는 유연한 검색 시간과 무료 대량 검색을 제공합니다. 이 스토리지 클래스는 백업 및 재해 복구에 적합합니다.

S3 Glacier Flexible Retrieval에 저장된 객체는 아카이빙되며 실시간 액세스에 사용할 수 없습니다. 자세한 내용은 S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive의 아카이브 스토리지 이해 섹션을 참조하세요. 이러한 객체에 액세스하려면 먼저 복원 요청을 시작하여 요청이 완료될 때 액세스할 수 있는 객체의 임시 사본을 생성합니다. 자세한 내용은 아카이브된 객체 작업 섹션을 참조하세요. 객체를 복원할 때 사용 사례에 맞는 검색 계층을 선택합니다. 복원 시간이 늘어나면 비용이 낮아집니다.

S3 Glacier Flexible Retrieval에 사용할 수 있는 검색 계층은 다음과 같습니다.

- 신속 검색 - 일반적으로 1~5분 내에 객체를 복원합니다. 신속 검색은 수요에 따라 달라지므로 안정적이고 예측 가능한 복원 시간을 확보하려면 프로비저닝된 검색 용량을 구매하는 것이 좋습니다. 자세한 내용은 프로비저닝된 용량 섹션을 참조하세요.
- 표준 검색 - 일반적으로 3~5시간 내에 객체를 복원합니다. S3 배치 작업을 사용하는 경우 1분~5시간 이내에 객체를 복원합니다. 자세한 내용은 배치 작업을 통한 객체 복원 섹션을 참조하세요.
- 대량 검색 - 일반적으로 5~12시간 내에 객체를 복원합니다. 대량 검색은 무료입니다.

S3 Glacier Flexible Retrieval 스토리지 클래스의 객체 최소 저장 기간은 90일입니다.

S3 Glacier Flexible Retrieval에는 각 객체에 대해 40KB의 추가 메타데이터가 필요합니다. 여기에는 데이터를 식별하고 검색하는 데 필요한 32KB의 메타데이터가 포함되며 S3 Glacier Flexible Retrieval의

기본 요금이 부과됩니다. 아카이빙된 객체의 사용자 정의 이름 및 메타데이터를 유지하기 위해 추가 8KB의 데이터가 필요하며 S3 Standard 요금이 부과됩니다.

S3 Glacier Deep Archive

액세스 횟수가 1년에 1회 미만인 아카이브 데이터에는 S3 Glacier Deep Archive를 사용하는 것이 좋습니다. 이 스토리지 클래스는 규정 준수 요구 사항을 충족하기 위해 데이터세트를 수년 동안 보존하도록 설계되었으며, 백업이나 재해 복구 또는 검색하는 데 최대 72시간까지 기다려도 되는 자주 액세스하지 않는 데이터에도 사용할 수 있습니다. S3 Glacier Deep Archive는 AWS에서 가장 저렴한 스토리지 옵션입니다.

S3 Glacier Deep Archive에 저장된 객체는 아카이빙되며 실시간 액세스에 사용할 수 없습니다. 자세한 내용은 S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive의 아카이브 스토리지 이해 섹션을 참조하세요. 이러한 객체에 액세스하려면 먼저 복원 요청을 시작하여 요청이 완료될 때 액세스할 수 있는 객체의 임시 사본을 생성합니다. 자세한 내용은 아카이브된 객체 작업 섹션을 참조하세요. 객체를 복원할 때 사용 사례에 맞는 검색 계층을 선택합니다. 복원 시간이 늘어나면 비용이 낮아집니다.

S3 Glacier Deep Archive에 사용할 수 있는 검색 계층은 다음과 같습니다.

- 표준 검색 - 일반적으로 12시간 내에 객체를 복원합니다. S3 배치 작업을 사용하는 경우 9~12시간 이내에 객체를 복원합니다. 자세한 내용은 배치 작업을 통한 객체 복원 섹션을 참조하세요.
- 대량 검색 - 일반적으로 Standard 검색 계층 비용보다 아주 낮은 비용으로 48시간 내에 객체를 복원합니다.

S3 Glacier Deep Archive 스토리지 클래스의 객체 최소 저장 기간은 180일입니다.

S3 Glacier Deep Archive에는 각 객체에 대해 40KB의 추가 메타데이터가 필요합니다. 여기에는 데이터를 식별하고 검색하는 데 필요한 32KB의 메타데이터가 포함되며 S3 Glacier Deep Archive의 기본 요금이 부과됩니다. 아카이빙된 객체의 사용자 정의 이름 및 메타데이터를 유지하기 위해 추가 8KB의 데이터가 필요하며 S3 Standard 요금이 부과됩니다.

S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive의 아카이브 스토리지 이해

S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive는 아카이브 스토리지 클래입니다. 즉, 이러한 스토리지 클래스에 객체를 저장하면 해당 객체는 아카이빙되며 직접 액세스할 수 없습니다. 아카이빙된 객체에 액세스하려면 객체에 대한 복원 요청을 제출한 다음 서비스가 객체를 복원할 때까지 기다려야 합니다. 복원 요청은 객체의 임시 사본을 복원하며, 요청에서 지정한 기간이 만료되면 해당 사본이 삭제됩니다. 자세한 내용은 아카이브된 객체 작업 섹션을 참조하세요.

객체를 S3 Glacier Deep Archive 스토리지 클래스로 전환할 때는 한 방향으로만 이동할 수 있습니다.

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 Instant Retrieval, S3 Glacier Flexible Retrieval 또는 Reduced Redundancy Storage를 스토리지 클래스로 지정하여 객체를 덮어씁니다.

> **Note**
>
> 복원된 객체에 대한 복사 작업은 Amazon S3 콘솔에서 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스에 있는 객체에 대해 지원되지 않습니다. 이러한 유형의 복사 작업에는 AWS Command Line Interface(AWS CLI), AWS SDK 또는 REST API를 사용하십시오.

이러한 스토리지 클래스에 아카이빙된 객체는 AWS 리전별로 계정당 최대 1,000TPS(초당 트랜잭션 수)의 객체 복원 요청으로 복원할 수 있습니다.

비용 고려 사항

자주 액세스하지 않는 데이터를 몇 달 또는 몇 년 동안 아카이브할 계획이라면 S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive 스토리지 클래스로 스토리지 비용을 절감할 수 있습니다. 그러나 S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive 스토리지 클래스가 적합한지 확인하려면 다음을 고려하세요.

- 스토리지 오버헤드 요금 - 아카이브된 각 객체에 대해 40KB의 추가 메타데이터가 필요합니다. 여기에는 데이터를 식별하고 검색하는 데 필요한 32KB의 메타데이터가 포함되며 해당 스토리지 클래스의 기본 요금이 부과됩니다. 아카이빙된 객체의 사용자 정의 이름 및 메타데이터를 유지하기 위해 추가 8KB의 데이터가 필요하며 S3 Standard 요금이 부과됩니다.

  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.
- 멀티파트 업로드 요금 - S3-storage-class-glacier의 객체입니다. S3 Glacier Deep Archive는 멀티파트 업로드를 사용하여 업로드할 때 S3 Standard 스토리지 클래스 요금으로 청구됩니다. 자세한 내용은 멀티파트 업로드 및 요금 섹션을 참조하세요.
- 최소 30일 스토리지 요금 - S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive는 장기 아카이브 솔루션입니다. 최소 스토리지 기간은 S3 Glacier Flexible Retrieval 스토리지 클래스의 경우 90일, S3 Glacier Deep Archive의 경우 180일입니다. 삭제한 객체가 최소 스토리지 기간보다 오래 아카이

브되는 경우 스토리지 클래스에 아카이브된 데이터를 삭제해도 요금이 부과되지 않습니다. 최소 기간 안에 아카이브된 객체를 삭제하거나 덮어쓰는 경우, Amazon S3는 해당 기간의 나머지 기간에 대한 요금을 청구합니다.

- 데이터 검색 요금 - 아카이브된 객체를 S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive Archive에 복원하는 경우 요청당 데이터 검색 요금이 부과됩니다. 이러한 요금은 복원을 시작할 때 선택한 검색 계층에 따라 달라집니다. 요금 정보는 Amazon S3 요금을 참조하세요.
- S3 수명 주기 – 아카이브된 객체를 S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive에 복원하면 요청당 데이터 검색 요금이 부과됩니다. 이러한 요금은 복원을 시작할 때 선택한 검색 계층에 따라 달라집니다. 요금 정보는 Amazon S3 요금을 참조하세요.

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 Flexible Retrieval 또는 S3 Glacier Deep Archive에 유지됩니다. (HeadObject 또는 GetObject API 작업 요청은 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive를 스토리지 클래스로 반환합니다.)

> **Note**
>
> 아카이브를 복원하면 아카이브(S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 요금) 및 임시로 복원한 복사본(S3 Standard 스토리지 요금) 모두에 대해 요금이 청구됩니다. 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하세요.

프로그래밍 방식으로 또는 Amazon S3 콘솔을 사용하여 객체 복사본을 복원할 수 있습니다. Amazon S3는 각 객체에 대한 복원 요청을 한 번에 하나씩만 처리합니다. 자세한 내용은 아카이브된 객체 복원 섹션을 참조하세요.

이러한 스토리지 클래스와 S3 Glacier 서비스의 차이점

S3 Glacier 스토리지 클래스는 Amazon S3 서비스의 일부이며 데이터를 S3 버킷에 객체로 저장합니다. S3 콘솔을 사용하거나 S3 API 또는 SDK를 사용하여 프로그래밍 방식으로 이러한 스토리지 클래스의 객체를 관리할 수 있습니다. S3 Glacier 스토리지 클래스에 객체를 저장하는 경우 고급 암호화, 객체 태그 지정, S3 수명 주기 구성과 같은 S3 기능을 사용하여 데이터 접근성과 비용을 관리할 수 있습니다.

> **Important**
>
> 모든 장기 데이터에 대해 Amazon S3 서비스 내의 S3 Glacier 스토리지 클래스를 사용하는 것이 좋습니다.

Amazon S3 Glacier(S3 Glacier) 서비스는 볼트에 데이터를 아카이브로 저장하는 별도의 서비스입니다. 이 서비스는 Amazon S3 기능을 지원하지 않으며 데이터 업로드 및 다운로드 작업에 콘솔 지원을 제공하지 않습니다. 장기 데이터에는 S3 Glacier 서비스를 사용하지 않는 것이 좋습니다. 이 서비스에 저장된 데이터는 Amazon S3 서비스에서 액세스할 수 없습니다. S3 Glacier 서비스에 대한 자세한 내용은 Amazon S3 Glacier 개발자 가이드를 참조하세요. Amazon S3 Glacier 서비스에서 Amazon S3의 스토리지 클래스로 데이터를 전송하려면 AWS 솔루션 라이브러리의 Data Transfer from Amazon S3 Glacier Vaults to Amazon S3를 참조하세요.

아카이브된 객체 작업

자주 액세스하지 않는 객체의 스토리지 비용을 줄이려면 해당 객체를 아카이브합니다. 객체를 아카이브하면 저비용 스토리지로 이동되므로 실시간으로 액세스할 수 없습니다.

아카이브된 객체에 실시간으로 액세스할 수는 없지만 스토리지 클래스에 따라 몇 분 또는 몇 시간 내에 복원할 수 있습니다. Amazon S3 콘솔, S3 배치 작업, REST API, AWS SDK 및 AWS Command Line Interface(AWS CLI)를 사용하여 아카이브된 객체를 복원할 수 있습니다. 지침은 아카이브된 객체 복원 단원을 참조하십시오.

다음 스토리지 클래스 또는 계층의 Amazon S3 객체는 아카이브되며 해당 객체에 실시간으로 액세스할 수 없습니다.

- S3 Glacier Flexible Retrieval 스토리지 클래스
- S3 Glacier Deep Archive 스토리지 클래스
- S3 Intelligent-Tiering Archive Access 계층
- S3 Intelligent-Tiering Deep Archive Access 계층

아카이브된 객체를 복원하려면 다음을 수행해야 합니다.

- S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive 스토리지 클래스에 있는 객체의 경우 복원 요청을 시작하고 객체의 임시 복사본을 사용할 수 있을 때까지 기다려야 합니다. 복원된 객체의 임시 사본이 생성된 경우 객체의 스토리지 클래스는 동일하게 유지됩니다. (HeadObject 또는 GetObject

API 작업 요청은 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive를 스토리지 클래스로 요청합니다.)

- S3 Intelligent-Tiering Archive Access 및 S3 Intelligent-Tiering Deep Archive Access 계층에 있는 객체의 경우 복원 요청을 시작하고 객체가 Frequent Access 계층으로 이동할 때까지 기다려야 합니다.

모든 Amazon S3 스토리지 클래스 비교에 대한 자세한 내용은 Amazon S3 스토리지 클래스 이해 및 관리 섹션을 참조하세요. S3 Intelligent-Tiering에 대한 자세한 내용은 the section called “S3 Intelligent-Tiering 작동 방식” 섹션을 참조하세요.

복원 작업을 완료하는 데 걸리는 시간은 사용하는 아카이브 스토리지 클래스 또는 스토리지 계층과 지정한 검색 옵션(긴급(S3 Glacier Flexible Retrieval 및 S3 Intelligent-Tiering Archive Access에만 사용 가능), 스탠다드 또는 대량)에 따라 달라집니다. 자세한 내용은 아카이브 가져오기 옵션 이해 단원을 참조하십시오.

복원이 완료되면 Amazon S3 이벤트 알림을 사용하여 알림을 받을 수 있습니다. 자세한 내용은 Amazon S3 이벤트 알림 단원을 참조하십시오.

S3 Glacier에서 객체 복원

S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive를 사용하는 경우 Amazon S3는 지정된 기간 동안만 객체의 임시 사본을 복원합니다. 그런 다음 복원된 객체 복사본을 삭제합니다. 복원 요청을 다시 발행하여 복원된 복사본의 만료 기간을 수정할 수 있습니다. 이 경우 Amazon S3는 현재 시간을 기준으로 만료 기간을 업데이트합니다.

> **Note**
>
> S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive에서 아카이브된 객체를 복원하면 아카이브된 객체와 임시로 복원한 사본 모두에 대해 요금이 청구됩니다. 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

Amazon S3는 복원 요청에 지정된 일수와 요청한 복원이 완료된 시간을 더하여 복원된 객체 복사본의 만료 시간을 계산합니다. 그런 다음 Amazon S3는 협정 세계시(UTC) 자정에 그 다음 날로 결과 시간을 반올림합니다. 예를 들어 복원된 객체가 2012년 10월 15일 오전 10시 30분(UTC)에 생성되었으며 복원 기간이 3일로 지정되었다고 가정하겠습니다. 이 경우 복원된 복사본은 2012년 10월 19일 00:00(UTC)에 만료되며 이 시간에 Amazon S3가 객체 복사본을 삭제합니다.

S3 Intelligent-Tiering에서 객체 복원

S3 Intelligent-Tiering Archive Access 계층 또는 S3 Intelligent-Tiering Deep Archive Access 계층에서 객체를 복원하는 경우 객체는 S3 Intelligent-Tiering Frequent Access 계층으로 이동합니다. 연속 30일이 지나는 동안 객체에 액세스하지 않으면 자동으로 Infrequent Access 계층으로 이동합니다. 액세스하지 않은 기간이 최소 연속 90일이 넘으면 객체가 자동으로 S3 Intelligent-Tiering Archive Access 계층으로 이동합니다. 최소 연속 180일 동안 객체에 액세스하지 않으면 객체가 Deep Archive Access 계층으로 이동합니다.

> **Note**
>
> S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive 스토리지 클래스와 달리 S3 Intelligent-Tiering 객체에 대한 복원 요청에는 `Days` 값을 사용할 수 없습니다.

복원 요청에 S3 배치 작업 사용

단일 요청으로 Amazon S3 객체를 하나 이상 복원하려면 S3 배치 작업을 사용할 수 있습니다. 작업할 객체 목록을 S3 배치 작업에 제공합니다. S3 배치 작업은 지정된 작업을 수행하기 위해 각 API 작업을 호출합니다. 단일 배치 작업 건으로 엑사바이트 규모의 데이터가 포함된 수십억 개의 객체에서 지정된 작업을 수행할 수 있습니다.

주제

- 아카이브 가져오기 옵션 이해
- 아카이브된 객체 복원

아카이브 가져오기 옵션 이해

다음은 Amazon S3에서 아카이브된 객체를 복원할 때 사용 가능한 검색 옵션입니다.

- 긴급 - S3 Glacier Flexible Retrieval 스토리지 클래스 또는 S3 Intelligent-Tiering Archive Access 계층에 저장된 데이터에 신속하게 액세스할 수 있습니다. 이 옵션은 간혹 아카이브의 하위 집합에 대한 긴급한 요청이 필요할 때 사용할 수 있습니다. 아카이브된 가장 큰 객체(250MB+)를 제외한 모든 경우, 긴급 검색을 사용하여 액세스된 데이터는 일반적으로 1~5분 안에 사용할 수 있게 됩니다.

  > **Note**
  >
  > 빠른 검색은 프리미엄 기능이며 빠른 요청 및 검색 요금이 부과됩니다.

Amazon S3 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하세요.

프로비저닝된 용량은 긴급 검색에 대한 검색 용량이 필요할 때 S3 Glacier Flexible Retrieval의 용량을 제공합니다. 자세한 내용은 프로비저닝된 용량 단원을 참조하십시오.

- 스탠다드 – 아카이브된 모든 객체에 몇 시간 내에 액세스할 수 있습니다. 검색 요청 시 검색 옵션을 지정하지 않을 경우 스탠다드가 기본 옵션이 됩니다. 표준 검색은 일반적으로 S3 Glacier Flexible Retrieval 스토리지 클래스 또는 S3 Intelligent-Tiering Archive Access 계층에 저장된 객체의 경우 3~5시간 이내에 완료됩니다. S3 Glacier Deep Archive 스토리지 클래스 또는 S3 Intelligent-Tiering Deep Archive Access 계층에 저장된 객체의 경우 이 검색은 일반적으로 12시간 이내에 완료됩니다. 표준 검색은 S3 Intelligent-Tiering에 저장된 객체에 대해 무료입니다.

  > **Note**
  >
  > - S3 Glacier Flexible Retrieval 스토리지 클래스 또는 S3 Intelligent-Tiering Archive Access 계층에 저장된 객체의 경우, S3 배치 작업의 복원 작업을 사용하여 시작된 표준 검색은 일반적으로 몇 분 이내에 시작되고 3~5시간 이내에 완료됩니다.
  > - S3 Glacier Deep Archive 스토리지 클래스 또는 S3 Intelligent-Tiering Deep Archive Access 계층에 저장된 객체의 경우, 배치 작업의 복원 작업을 사용하여 시작된 표준 검색은 일반적으로 9시간 이내에 시작되고 12시간 이내에 완료됩니다.

- 대량 - Amazon S3 Glacier에서 가장 저렴한 검색 옵션을 사용하여 데이터에 액세스할 수 있습니다. 대량 검색을 통해 심지어 페타바이트 단위의 대용량 데이터도 저렴하게 검색할 수 있습니다.

  S3 Glacier Flexible Retrieval 스토리지 클래스 또는 S3 Intelligent-Tiering Archive Access 계층에 저장된 객체의 경우, 대량 검색은 일반적으로 5~12시간 이내에 완료됩니다. S3 Glacier Deep Archive 스토리지 클래스 또는 S3 Intelligent-Tiering Deep Archive Access 계층에 저장된 객체의 경우, 이러한 검색은 일반적으로 48시간 이내에 완료됩니다.

  S3 Glacier Flexible Retrieval 또는 S3 Intelligent-Tiering 스토리지 클래스에 저장된 객체의 경우 대량 검색이 무료입니다.

다음 테이블에는 아카이브 검색 옵션이 요약되어 있습니다. 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

`Expedited`, `Standard` 또는 `Bulk` 검색을 실행하려면 RestoreObject REST API 작업 요청의 `Tier` 요청 요소를 원하는 옵션으로 설정하거나 AWS Command Line Interface(AWS CLI) 또는 AWS SDK에

서 동등한 옵션으로 설정합니다. 프로비저닝된 용량을 구매하였다면, 사용자의 프로비저닝된 용량을 통해 모든 신속 검색이 자동으로 수행됩니다.

프로비저닝된 용량

프로비저닝된 용량은 긴급 검색에 대한 검색 용량이 필요할 때 S3 Glacier Flexible Retrieval의 용량을 제공합니다. 각 용량 단위로 5분마다 긴급 검색을 최소 3회 수행할 수 있고, 초당 최대 150메가바이트(MBps)의 검색 처리량이 제공됩니다.

워크로드에 몇 분 내로 데이터의 하위 집합에 대한 매우 안전하고 예측 가능한 액세스가 필요한 경우 프로비저닝된 검색 용량을 구매하는 것이 좋습니다. 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.

프로비저닝된 용량 단위는 AWS 계정에 할당됩니다. 따라서 버킷 소유자가 아닌 긴급 데이터 검색의 요청자가 프로비저닝된 용량 단위를 구매해야 합니다.

Amazon S3 콘솔, Amazon S3 Glacier 콘솔, 프로비저닝된 용량 구매 REST API 작업, AWS SDK 또는 AWS CLI를 사용하면 프로비저닝된 용량을 구매할 수 있습니다. 프로비저닝된 용량의 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하세요.

S3 Glacier 복원 시작 요청 속도

S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스에 저장된 객체에 대해 복원 요청을 시작할 때 검색 요청 할당량이 AWS 계정에 적용됩니다. S3 Glacier는 초당 최대 1,000개의 트랜잭션 속도로 복원 요청을 지원합니다. 이 속도를 초과할 경우, 유효한 요청은 제한되거나 거부되며 Amazon S3에서 `ThrottlingException` 오류를 반환합니다.

선택적으로 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive에 저장된 다량의 객체를 하나의 요청으로 검색하기 위해 S3 배치 작업을 사용할 수도 있습니다. 자세한 내용은 Batch Operations를 사용하여 대량으로 객체 작업 수행 단원을 참조하십시오.

아카이브된 객체 복원

다음 스토리지 클래스 또는 계층의 Amazon S3 객체는 아카이브되며 해당 객체에 실시간으로 액세스할 수 없습니다.

- S3 Glacier Flexible Retrieval 스토리지 클래스
- S3 Glacier Deep Archive 스토리지 클래스
- S3 Intelligent-Tiering Archive Access 계층

- S3 Intelligent-Tiering Deep Archive Access 계층

S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스에 저장된 Amazon S3 객체에는 즉시 액세스할 수 없습니다. 이 스토리지 클래스의 객체에 액세스하려면 지정된 기간(일) 동안 객체의 임시 사본을 S3 버킷에 복원해야 합니다. 객체의 영구 사본이 필요한 경우, 객체를 복원한 후 Amazon S3 버킷에 객체의 사본을 만드세요. 복원된 객체 복사는 Amazon S3 콘솔에서 지원되지 않습니다. 이러한 유형의 복사 작업에는 AWS Command Line Interface(AWS CLI), AWS SDK 또는 REST API를 사용하세요. 사본을 만들고 스토리지 클래스를 변경하지 않는 한 객체는 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스에 계속 저장됩니다. 이 스토리지 클래스의 사용에 대한 자세한 내용은 거의 액세스하지 않는 객체를 위한 스토리지 클래스 섹션을 참조하세요.

S3 Intelligent-Tiering Archive Access 및 Deep Archive Access 계층의 객체에 액세스하려면 복원 요청을 시작하고 객체가 Frequent Access 계층으로 이동할 때까지 기다려야 합니다. Archive Access 계층 또는 Deep Archive Access 계층에서 객체를 복원하면 객체가 Frequent Access 계층으로 다시 전환됩니다. 이 스토리지 클래스의 사용에 대한 자세한 내용은 변경되는 또는 알 수 없는 액세스 패턴으로 데이터를 자동으로 최적화하는 스토리지 클래스 섹션을 참조하세요.

아카이브된 객체에 대한 일반적인 정보는 아카이브된 객체 작업 섹션을 참조하세요.

> **Note**
>
> - S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스에서 아카이브된 객체를 복원하는 경우, 아카이브된 객체와 임시로 복원한 사본 모두에 대해 요금이 청구됩니다.
> - S3 Intelligent-Tiering에서 객체를 복원하는 경우, 표준 또는 대량 검색에는 검색 요금이 부과되지 않습니다.
> - 이미 복원된 아카이브 객체에 대해 요청된 후속 복원은 GET 요청으로 청구됩니다. 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

아카이브된 객체 복원

Amazon S3 콘솔, Amazon S3 REST API, AWS SDK, AWS Command Line Interface(AWS CLI) 또는 S3 배치 작업을 사용하여 아카이브된 객체를 복원할 수 있습니다.

S3 콘솔 사용

Amazon S3 콘솔을 사용하여 객체 복원

S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스나 S3 Intelligent-Tiering Archive Access 또는 Deep Archive Access 스토리지 계층에 아카이브된 객체를 복원하려면 다음 절차를 사용합니다.

아카이브된 객체 복원

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 복원할 객체가 들어 있는 버킷 이름을 선택합니다.
4. [객체(Objects)] 목록에서 객체 또는 복원할 객체를 선택하고 [작업(Actions)]을 선택한 다음 [복원 시작(Initiate restore)]을 선택합니다.
5. S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive에서 복원하는 경우 복원된 사본을 사용할 수 있는 기간(일) 상자에 아카이브된 데이터에 액세스할 수 있는 기간(일)을 입력합니다.
6. 검색 티어에서 다음 중 하나를 수행합니다.

   - 대량 검색 또는 표준 검색을 선택하고 복원 시작을 선택합니다.
   - 긴급 검색(Expedited retrieval)을 선택합니다(S3 Glacier Flexible Retrieval 또는 S3 Intelligent-Tiering Archive Access에만 사용 가능). S3 Glacier Flexible Retrieval에서 객체를 복원하는 경우 긴급 검색을 위해 프로비저닝된 용량을 구매할지 여부를 선택할 수 있습니다. 프로비저닝된 용량을 구매하려는 경우 다음 단계로 진행합니다. 그렇지 않은 경우 복원 시작을 선택합니다.

   > **Note**
   >
   > S3 Intelligent-Tiering Archive Access 및 Deep Archive Access 계층의 객체는 Frequent Access 계층으로 자동으로 복원됩니다.

7. (선택 사항) S3 Glacier Flexible Retrieval에서 객체를 복원하며 긴급 검색을 선택한 경우 프로비저닝된 용량을 구매할지 여부를 선택할 수 있습니다. 프로비저닝된 용량은 S3 Glacier Flexible Retrieval의 객체에만 사용할 수 있습니다. 프로비저닝된 용량이 있는 경우, 복원 시작을 선택하여 프로비저닝된 검색을 시작합니다.

   프로비저닝된 용량이 있으면 모든 긴급 검색이 프로비저닝된 용량으로 처리됩니다. 자세한 내용은 프로비저닝된 용량 단원을 참조하십시오.

   - 프로비저닝된 용량이 없고 구매할 계획도 없는 경우, 복원 시작을 선택합니다.

- 프로비저닝된 용량이 없지만 프로비저닝된 용량 단위(PCU)를 구매하려는 경우 PCU 구매를 선택합니다. PCU 구매 대화 상자에서 구매하려는 PCU 수를 선택하고 구매를 확인한 다음 PCU 구매를 선택합니다. 구매 성공 메시지가 나타나면 복원 시작을 선택하여 프로비저닝된 검색을 시작합니다.

AWS CLI 사용

S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive의 객체 복원

다음 예제에서는 `restore-object` 명령을 사용하여 *amzn-s3-demo-bucket* 버킷의 *dir1/example.obj* 객체를 25일 동안 복원합니다.

```
aws s3api restore-object --bucket amzn-s3-demo-bucket --key dir1/example.obj --restore-request '{"Days":25,"GlacierJobParameters":{"Tier":"Standard"}}'
```

예제에 사용된 JSON 구문으로 인해 Windows 클라이언트에서 오류가 발생하는 경우 복원 요청을 다음 구문으로 바꿉니다.

```
--restore-request Days=25,GlacierJobParameters={"Tier"="Standard"}
```

S3 Intelligent-Tiering Archive Access 및 Deep Archive Access의 객체 복원

다음 예제에서는 `restore-object` 명령을 사용하여 *amzn-s3-demo-bucket* 버킷의 *dir1/example.obj* 객체를 Frequent Access 계층으로 복원합니다.

```
aws s3api restore-object --bucket amzn-s3-demo-bucket --key dir1/example.obj --restore-request '{}'
```

> **Note**
>
> S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive 스토리지 클래스와 달리 S3 Intelligent-Tiering 객체에 대한 복원 요청에는 `Days` 값을 사용할 수 없습니다.

복원 상태 모니터링

다음 `head-object` 명령을 실행하여 `restore-object` 요청 상태를 모니터링하려면 다음을 수행하세요.

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key dir1/example.obj
```

자세한 정보는 AWS CLI 명령 참조의 restore-object 섹션을 참조하세요.

REST API 사용

Amazon S3에서 제공한 API 작업으로 아카이브된 객체의 복원을 시작합니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 RestoreObject를 참조하십시오.

AWS SDK 사용

AWS SDK를 사용하여 S3 Glacier Flexier Retrieval 또는 S3 Glacier Deep Archive에 아카이브된 객체를 복원하는 방법의 예제는 Amazon S3 API 참조의 코드 예제를 참조하세요.

S3 배치 작업 사용

단일 요청으로 아카이브된 객체를 하나 이상 복원하려면 S3 배치 작업을 사용할 수 있습니다. 작업할 객체 목록을 S3 배치 작업에 제공합니다. S3 배치 작업은 지정된 작업을 수행하기 위해 각 API 작업을 호출합니다. 단일 배치 작업 건으로 엑사바이트 규모의 데이터가 포함된 수십억 개의 객체에서 지정된 작업을 수행할 수 있습니다.

배치 작업 건을 만들려면 복원할 개체만 포함된 매니페스트가 있어야 합니다. S3 인벤토리를 사용하여 매니페스트를 생성하거나 필요한 정보가 포함된 CSV 파일을 제공할 수 있습니다. 자세한 내용은 the section called “매니페스트 지정” 단원을 참조하십시오.

S3 배치 작업 건을 생성하고 실행하기 전에 Amazon S3에 사용자 대신 S3 배치 작업을 수행할 권한을 부여해야 합니다. 필요한 권한에 대해서는 the section called “권한 부여” 섹션을 참조하세요.

> **Note**
>
> 배치 작업 건은 S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive 스토리지 클래스 객체 또는 S3 Intelligent-Tiering Archive Access 및 Deep Archive Access 스토리지 계층 객체에서 작동할 수 있습니다. 배치 작업은 동일한 작업 건에 있는 두 가지 유형의 아카이브된 객체 모두에서 작동할 수 없습니다. 두 유형의 객체를 복원하려면 별도의 배치 작업을 생성해야 합니다.
>
> 아카이브 객체 복원에 배치 작업을 사용하는 것에 대한 자세한 내용은 the section called “객체 복원” 섹션을 참조하세요.

S3 객체 복원 시작 배치 작업을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Batch Operations를 선택합니다.
3. 작업 생성을 선택합니다.
4. AWS 리전은 작업을 생성하려는 리전을 선택합니다.
5. 매니페스트 형식에서 사용할 매니페스트의 형식을 선택합니다.

   - S3 인벤토리 보고서를 선택하는 경우 Amazon S3가 CSV 형식 인벤토리 보고서의 일부로 생성한 `manifest.json` 객체의 경로를 입력합니다. 최신 버전이 아닌 매니페스트 버전을 사용하려는 경우 `manifest.json` 객체의 버전 ID를 입력합니다.
   - CSV를 선택하는 경우 CSV 형식 매니페스트 객체의 경로를 입력합니다. 매니페스트 객체는 콘솔에 설명된 형식을 따라야 합니다. 최신 버전이 아닌 버전을 사용하려는 경우 선택적으로 매니페스트 객체의 버전 ID를 포함할 수 있습니다.
6. Next(다음)를 선택합니다.
7. 작업 섹션에서 복원을 선택합니다.
8. 복원 섹션에서 복원 소스로 Glacier Flexible Retrieval 또는 Glacier Deep Archive 또는 Intelligent-Tiering Archive Access 계층 또는 Deep Archive Access 계층을 선택합니다.

   Glacier Flexible Retrieval 또는 Glacier Deep Archive를 선택한 경우 복원된 사본을 사용할 수 있는 기간(일)에 숫자를 입력합니다.

   검색 티어에서 사용할 계층을 선택합니다.
9. Next(다음)를 선택합니다.
10. 추가 옵션 구성 페이지에서 다음 섹션을 작성합니다.

    - 추가 옵션 섹션에서 작업에 대한 설명을 제공하고 작업의 우선 순위 번호를 지정합니다. 숫자가 높을수록 우선 순위가 높습니다. 자세한 내용은 the section called “작업 우선 순위 지정” 단원을 참조하십시오.
    - 완료 보고서 섹션에서 배치 작업으로 완료 보고서를 생성할지 여부를 선택합니다. 완료 보고서에 대한 자세한 내용은 the section called “완료 보고서” 섹션을 참조하세요.
    - 권한 섹션에서 Amazon S3에 사용자 대신 배치 작업을 수행할 권한을 부여해야 합니다. 필요한 권한에 대해서는 the section called “권한 부여” 섹션을 참조하세요.

- (선택 사항) 작업 태그 섹션에서 키-값 페어로 태그를 추가합니다. 자세한 내용은 the section called “태그 사용” 단원을 참조하십시오.

마쳤으면 다음을 선택합니다.

11. 복습 페이지에서 설정을 확인합니다. 설정을 변경하려면 이전을 선택합니다 또는 작업 생성을 선택합니다.

배치 작업에 대한 자세한 내용은 배치 작업을 통한 객체 복원 및 S3 배치 작업 건 생성 섹션을 참조하세요.

복원 상태 및 만료 날짜 확인

Amazon S3 콘솔, Amazon S3 이벤트 알림, AWS CLI 또는 Amazon S3 REST API를 사용하여 복원 요청 상태 또는 만료 날짜를 확인할 수 있습니다.

> **Note**
>
> S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive 스토리지 클래스에서 복원된 객체는 지정된 일수 동안만 저장됩니다. 아래 절차에 따라 이러한 사본의 만료일이 반환됩니다.
> S3 Intelligent-Tiering Archive Access 및 Deep Archive Access 스토리지 계층에서 복원된 객체에는 만료 날짜가 없으며 대신 Frequent Access 계층으로 다시 이동됩니다.

S3 콘솔 사용

Amazon S3 콘솔에서 객체의 복원 상태 및 만료 날짜를 확인하려면

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 복원할 객체가 들어 있는 버킷 이름을 선택합니다.
4. 객체목록에서 복원할 개체를 선택합니다. 객체 세부 정보 페이지가 나타납니다.

   - 복원이 완료되지 않은 경우 페이지 상단에 복원 진행 중이라는 섹션이 표시됩니다.
   - 복원이 완료된 경우 페이지 상단에 복원 완료라는 섹션이 표시됩니다. S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive에서 복원하는 경우 이 섹션에는 복원 만료 날짜도 표시됩니다. 이 날짜는 Amazon S3가 아카이브된 객체의 복원된 사본을 제거하는 날짜입니다.

Amazon S3 이벤트 알림 사용

Amazon S3 이벤트 알림 기능과 함께 `s3:ObjectRestore:Completed` 작업을 사용하여 객체 복원 완료에 대해 알림을 받을 수 있습니다. 이벤트 알림 활성화에 대한 자세한 내용은 Amazon SQS, Amazon SNS 및 AWS Lambda를 사용한 알림 활성화를 참조하세요. 다양한 `ObjectRestore` 이벤트 유형에 대한 자세한 내용은 the section called “SQS, SNS 및 Lambda에 지원되는 이벤트 유형” 단원을 참조하세요.

AWS CLI 사용

AWS CLI를 사용하여 객체의 복원 상태 및 만료 날짜를 확인합니다.

다음 예제에서는 `head-object` 명령을 사용하여 *amzn-s3-demo-bucket* 버킷의 *dir1/example.obj* 객체에 대한 메타데이터를 확인합니다. 복원 중인 객체에 대해 이 명령을 실행하면 Amazon S3에서 복원이 진행 중인지 여부와 만료 날짜(해당하는 경우)를 반환합니다.

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key dir1/example.obj
```

예상 출력(복원 진행 중):

```
{
    "Restore": "ongoing-request=\"true\"",
    "LastModified": "2020-06-16T21:55:22+00:00",
    "ContentLength": 405,
    "ETag": "\"b662d79adeb7c8d787ea7eafb9ef6207\"",
    "VersionId": "wbYaE2vtOV0iIBXrOqGAJt3fP1cHB8Wi",
    "ContentType": "binary/octet-stream",
    "ServerSideEncryption": "AES256",
    "Metadata": {},
    "StorageClass": "GLACIER"
}
```

예상 출력(복원 완료):

```
{
    "Restore": "ongoing-request=\"false\", expiry-date=\"Wed, 12 Aug 2020 00:00:00 GMT
\"",
    "LastModified": "2020-06-16T21:55:22+00:00",
    "ContentLength": 405,
    "ETag": "\"b662d79adeb7c8d787ea7eafb9ef6207\"",
```

```
    "VersionId": "wbYaE2vtOV0iIBXrOqGAJt3fP1cHB8Wi",
    "ContentType": "binary/octet-stream",
    "ServerSideEncryption": "AES256",
    "Metadata": {},
    "StorageClass": "GLACIER"
}
```

`head-object`에 대한 자세한 정보는 AWS CLI 명령 참조의 head-object 단원을 참조하세요.

REST API 사용

Amazon S3는 객체 메타데이터를 검색할 수 있는 API 작업을 제공합니다. REST API를 사용하여 아카이브된 객체의 복원 상태 및 만료 날짜를 확인하려면 Amazon Simple Storage Service API 참조의 HeadObject 섹션을 참조하세요.

진행 중인 복원의 속도 업그레이드

복원 진행 중에 복원 속도를 업그레이드할 수 있습니다.

진행 중인 복원을 더 빠른 티어로 업그레이드

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷을 선택합니다.
3. 버킷(Buckets) 목록에서 복원할 객체가 들어 있는 버킷 이름을 선택합니다.
4. 객체목록에서 복원할 개체를 선택합니다. 객체 세부 정보 페이지가 나타납니다. 객체의 세부 정보 페이지에서 검색 계층 업그레이드를 선택합니다. 객체의 복원 상태 점검에 관한 내용은 복원 상태 및 만료 날짜 확인 단원을 참조하십시오.
5. 업그레이드할 계층을 선택한 후 복원 시작을 선택합니다.

객체 수명 주기 관리

S3 수명 주기를 사용하면 객체를 더 저렴한 스토리지 클래스로 이전하거나 자동으로 만료된 객체를 삭제하여 수명 주기 전반에 걸쳐 객체를 비용 효율적으로 저장할 수 있습니다. 객체의 수명 주기를 관리하려면 버킷에 대한 S3 수명 주기 구성을 만듭니다. S3 수명 주기 구성은 Amazon S3이 객체 그룹에 적용하는 작업을 정의하는 일련의 규칙입니다. 다음과 같은 두 가지 유형의 작업이 있습니다.

- 전환 작업 - 이 작업은 객체가 다른 스토리지 클래스로 전환되는 시기를 정의합니다. 예를 들어, 생성 후 30일이 지나면 객체를 S3 STANDARD-IA 스토리지 클래스로 전환하거나 생성 후 1년이 지나면

객체를 S3 Glacier Flexible Retrieval 스토리지 클래스에 아카이브하도록 선택할 수 있습니다. 자세한 내용은 Amazon S3 스토리지 클래스 이해 및 관리 섹션을 참조하세요.

수명 주기 전환 요청과 관련된 비용이 있습니다. 요금 정보는 Amazon S3 요금을 참조하세요.

- 만료 작업 - 이 작업은 객체가 만료되는 시기를 정의합니다. Amazon S3에서 만료된 객체를 자동으로 삭제합니다. 예를 들어, 규제 준수 기간 동안 객체를 저장한 후 객체를 만료하도록 선택할 수 있습니다. 자세한 내용은 객체 만료 섹션을 참조하세요.

  수명 주기 만료와 관련된 잠재적 비용은 스토리지 클래스의 객체를 최소 저장 기간으로 만료시키는 경우에만 발생합니다. 자세한 내용은 최소 스토리지 기간 요금 섹션을 참조하세요.

> **Important**
>
> 범용 버킷 - 버킷 정책을 사용하여 S3 수명 주기 규칙에 따른 삭제 또는 전환을 방지할 수는 없습니다. 예를 들어, 버킷 정책이 모든 보안 주체에 대한 모든 작업을 거부하더라도 S3 수명 주기 구성은 여전히 정상적으로 작동합니다.

기존 객체 및 새 객체

버킷에 수명 주기 구성을 추가할 경우 구성 규칙이 기존 객체는 물론 나중에 추가하는 객체에도 적용됩니다. 예를 들어, 객체를 생성한 후 30일이 경과하면 만료시키는 만료 작업을 포함하는 수명 주기 구성 규칙을 오늘 추가할 경우, Amazon S3는 30일 이상 경과한 모든 기존 객체를 삭제 대기열에 넣습니다.

결제 변경 사항

객체가 수명 주기 작업에 적합하게 되는 시점과 Amazon S3에서 객체를 이전하거나 만료하는 시점 사이에 지연이 있는 경우 객체가 수명 주기 작업에 적합한 상태가 되는 즉시 결제 변경 사항이 적용됩니다. 예를 들어 객체가 만료되도록 예약되어 있고 Amazon S3가 객체를 즉시 만료하지 않는 경우 만료 시간 이후에는 스토리지 요금이 부과되지 않습니다.

이 동작에 대한 한 가지 예외는 S3 Intelligent-Tiering 스토리지 클래스로 전환하는 수명 주기 규칙이 있는 경우입니다. 이 경우 객체가 S3 Intelligent-Tiering으로 전환될 때까지 결제 변경이 발생하지 않습니다. S3 수명 주기 규칙에 대한 자세한 내용은 수명 주기 구성의 요소 섹션을 참조하세요.

> **Note**
>
> 수명 주기 전환에는 데이터 검색 요금이 부과되지 않습니다. 하지만 PUT, COPY 또는 수명 주기 규칙을 사용하여 데이터를 S3 스토리지 클래스로 이동할 경우 요청당 수집 요금이 부과됩

니다. 객체를 스토리지 클래스로 이동하기 전에 수집 또는 전환 비용을 고려하세요. 비용 고려 사항에 대한 자세한 내용은 Amazon S3 요금을 참조하세요.

수명 주기 규칙의 영향 모니터링

활성 수명 주기 규칙에 의한 업데이트의 영향을 모니터링하려면 the section called “수명 주기 규칙에 따라 취해진 조치를 모니터링하려면 어떻게 해야 하나요?” 섹션을 참조하세요.

객체의 전체 수명 주기 관리

S3 수명 주기 구성 규칙을 사용하면 Amazon S3가 객체를 더 저렴한 스토리지 클래스로 전환하거나 아카이브하거나 삭제하도록 유도할 수 있습니다. 예시:

- 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 아카이브의 목적으로 Amazon S3에 업로드할 수도 있습니다. 예를 들어, 디지털 미디어, 금융 및 의료 기록, 가공되지 않은 유전체 염기서열 데이터, 장기 데이터베이스 백업 파일, 규제 준수를 위해 보존해야 하는 데이터 등을 보관할 수 있습니다.

S3 수명 주기 작업을 결합하여 객체의 전체 수명 주기를 관리합니다. 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 Flexible Retrieval 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 참조하세요.

주제

- Amazon S3 수명 주기를 사용하여 객체 전환
- 객체 만료

- 버킷에서 S3 수명 주기 구성 설정
- S3 수명 주기가 다른 버킷 구성과 상호 작용하는 방식
- S3 수명 주기 이벤트 알림 구성
- 수명 주기 구성의 요소
- S3가 수명 주기 구성의 충돌을 처리하는 방법
- S3 수명 주기 구성의 예제
- Amazon S3 수명 주기 문제 해결

Amazon S3 수명 주기를 사용하여 객체 전환

S3 수명 주기 구성에 이전 작업을 추가하여 Amazon S3에 다른 Amazon S3 스토리지 클래스로 객체를 이동하도록 지시할 수 있습니다. 스토리지 클래스에 대한 자세한 정보는 Amazon S3 스토리지 클래스 이해 및 관리를 참조하십시오. 이러한 방식으로 S3 수명 주기 구성을 사용할 수 있는 경우의 몇 가지 예에는 다음이 포함됩니다.

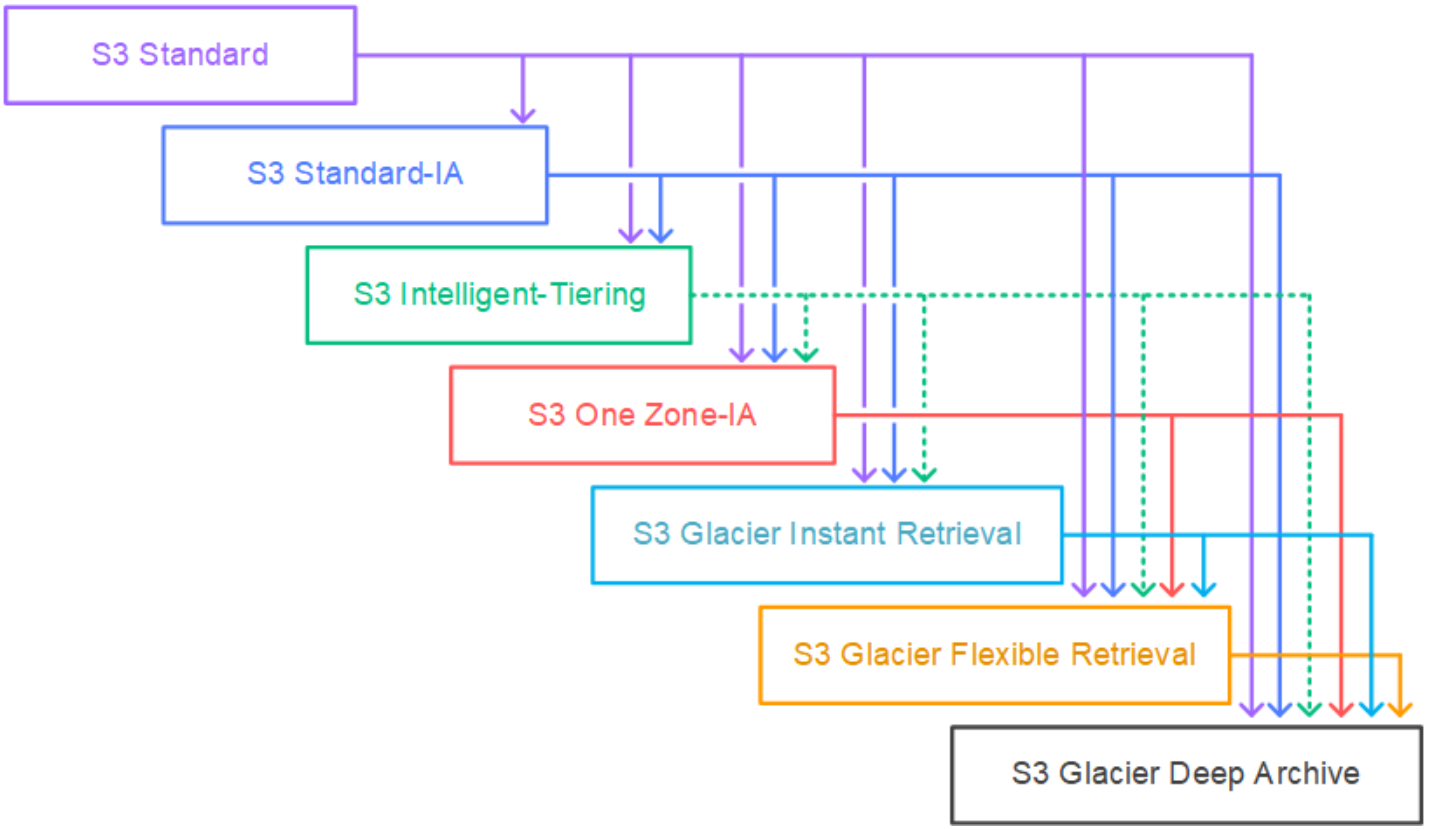
- 객체에 자주 액세스하지 않는 경우, 해당 객체를 S3 Standard-IA 스토리지 클래스로 전환할 수 있습니다.
- 실시간으로 액세스할 필요가 없는 객체를 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스에 아카이브할 수 있습니다.

> **Note**
>
> 암호화된 객체는 스토리지 클래스 전환 프로세스 전체에서 암호화된 상태로 유지됩니다.

Supported transitions

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 변하면 자동 비용 절감을 위해 객체를 S3 Intelligent-Tiering 스토리지 클래스로 전환할 수 있습니다. 스토리지 클래스에 대한 자세한 내용은 Amazon S3 스토리지 클래스 이해 및 관리 섹션을 참조하세요.

Amazon S3은 다음 다이어그램과 같이 스토리지 클래스 간 전환을 위한 폭포형(Waterfall) 모델을 지원합니다.



지원되는 수명 주기 전환

Amazon S3은 S3 수명 주기 구성을 사용하여 스토리지 클래스 간에 다음과 같은 수명 주기 전환을 지원합니다.

- S3 Standard 스토리지 클래스에서 S3 Standard-IA, S3 Intelligent-Tiering, S3 One Zone-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스로 전환
- S3 Standard-IA 스토리지 클래스에서 S3 Intelligent-Tiering, S3 One Zone-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스로 전환
- S3 Intelligent-Tiering 스토리지 클래스는 S3 Intelligent-Tiering 액세스 티어에 따라 다른 스토리지 클래스로 전환할 수 있습니다. 각 액세스 티어에 대해 다음과 같은 전환이 가능합니다.
  - Frequent Access 티어 또는 Infrequent Access 티어에서 S3 One Zone-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive로 전환
  - Archive Instant Access 티어에서 S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스로 전환
  - Archive Access 티어에서 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스로 전환

  - Deep Archive Access 티어에서 S3 Glacier Deep Archive 스토리지 클래스로 전환
- S3 One Zone-IA 스토리지 클래스에서 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스로 전환
- S3 Glacier Instant Retrieval 스토리지 클래스에서 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스로 전환
- S3 Glacier Flexible Retrieval 스토리지 클래스에서 S3 Glacier Deep Archive 스토리지 클래스로 전환

> **Note**
>
> 버전 관리가 활성화된 버킷 또는 버전 관리가 일시 중지된 버킷의 경우, 복제 상태가 `Pending`인 객체를 전환할 수 없습니다.

전환에 대한 제약 조건 및 고려 사항

수명 주기 스토리지 클래스 전환에는 다음과 같은 제약이 있습니다.

128KB 미만의 객체는 기본적으로 어떤 스토리지 클래스로도 전환되지 않습니다.

Amazon S3는 128KB 미만의 객체가 스토리지 클래스로 전환되지 않도록 하는 기본 동작을 S3 수명 주기 구성에 적용합니다. 각 객체마다 전환 요청 요금이 부과되므로, 128KB 미만의 객체는 전환하지 않는 것이 좋습니다. 즉, 소형 객체의 경우 전환 비용이 스토리지 절감액보다 클 수 있습니다. 전환 요청 비용에 대한 자세한 내용은 Amazon S3 요금 페이지의 스토리지 및 요청 탭에서 요청 및 데이터 검색을 참조하세요.

더 작은 객체가 전환되도록 하려면 사용자 지정 최소 크기(`ObjectSizeGreaterThan`) 또는 최대 크기(`ObjectSizeLessThan`)를 지정하는 객체 크기 필터를 수명 주기 전환 규칙에 추가할 수 있습니다. 자세한 내용은 예제: 128KB 미만의 객체 전환 허용 섹션을 참조하세요.

> **Note**
>
> 2024년 9월 Amazon S3는 소형 객체의 기본 전환 동작을 다음과 같이 업데이트했습니다.
>
> - 업데이트된 기본 전환 동작 - 2024년 9월부터 기본 동작은 128KB 미만의 객체가 스토리지 클래스로 전환되는 것을 금지합니다.
> - 이전의 기본 전환 동작 - 2024년 9월 이전의 기본 동작은 128KB 미만의 객체를 S3 Glacier 및 S3 Glacier Deep Archive 스토리지 클래스로는 전환할 수 있었습니다.

2024년 9월 이전에 생성된 구성은 수정하지 않는 한 이전 전환 동작을 유지합니다. 즉, 규칙을 생성, 편집 또는 삭제하면 구성의 기본 전환 동작이 업데이트된 동작으로 변경됩니다. 사용 사례에 필요한 경우 128KB 미만의 객체가 S3 Glacier 및 S3 Glacier Deep Archive로 전환되도록 기본 전환 동작을 변경할 수 있습니다. 이렇게 하려면 PutBucketLifecycleConfiguration 요청에서 선택적 `x-amz-transition-default-minimum-object-size` 헤더를 사용하세요.

객체를 S3 Standard-IA 또는 S3 One Zone-IA로 전환하기 전에 최소 30일 동안 저장해야 합니다.

객체를 S3 Standard-IA 또는 S3 One Zone-IA로 전환하기 전에 Amazon S3에 최소 30일 동안 해당 객체를 보관해야 합니다. 예를 들어, 생성 후 1일이 지난 객체를 S3 Standard-IA 스토리지 클래스로 전환하도록 요구하는 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 S3은 최초 30일 동안에는 이러한 전환을 지원하지 않습니다.

마찬가지로, 최신이 아닌 객체를 전환할 경우(버전이 지정된 버킷에서) 최소 30일이 경과한 최신 버전이 아닌 객체만 S3 Standard-IA 또는 S3 One Zone-IA 스토리지로 전환할 수 있습니다. 모든 스토리지 클래스의 최소 스토리지 기간 목록은 Amazon S3 스토리지 클래스 비교 섹션을 참조하세요.

최소 저장 기간 이전에 객체를 전환하면 요금이 부과됩니다.

특정 스토리지 클래스에는 최소 객체 저장 기간이 있습니다. 최소 기간 이전에 이러한 스토리지 클래스에서 객체를 전환하는 경우 해당 기간의 나머지 기간에 대해 요금이 부과됩니다. 최소 저장 기간이 있는 스토리지 클래스에 대한 자세한 내용은 Amazon S3 스토리지 클래스 비교 섹션을 참조하세요.

최소 저장 기간이 경과하기 전에 한 스토리지 클래스에서 다른 스토리지 클래스로 객체를 전환하는 단일 수명 주기 규칙은 만들 수 없습니다.

예를 들어, S3 Glacier Instant Retrieval의 경우 최소 저장 기간은 90일입니다. 4일 후에 객체를 S3 Glacier Instant Retrieval로 전환하고 20일 후에 객체를 S3 Glacier Deep Archive로 전환하는 수명 주기 규칙을 지정할 수 없습니다. 이 경우 S3 Glacier Deep Archive 전환은 최소 94일 후에 이루어져야 합니다.

이를 달성하기 위해 두 규칙을 지정할 수 있지만, 사용자는 최소 저장 기간 요금을 지불하게 됩니다. 비용 고려 사항에 대한 자세한 내용은 Amazon S3 요금을 참조하세요.

S3 수명 주기 생성 방법에 대한 자세한 내용은 버킷에서 S3 수명 주기 구성 설정 섹션을 참조하세요.

S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive 스토리지 클래스로 전환(객체 아카이브)

S3 수명 주기 구성을 사용하여 아카이브를 위해 객체를 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스로 전환할 수 있습니다.

객체를 보관하기 전에 이와 관련된 고려 사항에 대해 설명한 다음 섹션들을 살펴보세요.

일반적인 고려 사항

다음은 객체를 보관하기 전에 고려할 일반적인 사항들입니다.

- 암호화된 객체는 스토리지 클래스 전환 프로세스 전체에서 암호화된 상태로 유지됩니다.
- S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스에 저장된 객체는 실시간으로 사용할 수 없습니다.

  아카이브된 객체는 Amazon S3 객체이지만 아카이브된 객체에 액세스하려면 먼저 해당 객체의 임시 복사본을 복원해야 합니다. 복원된 객체 사본은 요청자가 복원 요청에서 지정한 기간 동안만 사용할 수 있습니다. 그런 다음 Amazon S3가 임시 복사본을 삭제하고 객체는 S3 Glacier Flexible Retrieval에 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 아카이브된 객체 복원 섹션을 참조하세요.
- S3 Glacier Flexible Retrieval 스토리지 클래스에 저장된 객체는 S3 Glacier Deep Archive 스토리지 클래스로만 전환할 수 있습니다.

  S3 수명 주기 구성 규칙을 사용하여 객체의 스토리지 클래스를 S3 Glacier Flexible Retrieval에서 S3 Glacier Deep Archive 스토리지 클래스로만 변환할 수 있습니다. S3 Glacier Flexible Retrieval에 저장된 객체의 스토리지 클래스를 S3 Glacier Deep Archive가 아닌 스토리지 클래스로 변경하려면 먼저 복원 작업을 사용하여 객체의 임시 복사본을 만들어야 합니다. 그런 다음 복사 작업을 통해 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 Instant Retrieval, S3 Glacier Flexible Retrieval 또는 Reduced Redundancy Storage를 스토리지 클래스로 지정하여 객체를 덮어씁니다.

> **Note**
>
> 복원된 객체에 대한 복사 작업은 Amazon S3 콘솔에서 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스에 있는 객체에 대해 지원되지 않습니다. 이러한 유형의 복사 작업에는 AWS Command Line Interface(AWS CLI), AWS SDK 또는 REST API를 사용하십시오.

S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive 스토리지 클래스에 저장된 객체는 Amazon S3를 통해서만 확인하고 사용할 수 있습니다. 별도의 Amazon S3 Glacier 서비스를 통해 사용할 수는 없습니다.

이러한 객체는 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 아카이브된 객체에 액세스할 수 없습니다.

비용 고려 사항

자주 액세스하지 않는 데이터를 몇 달 또는 몇 년 동안 아카이브할 계획이라면 S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive 스토리지 클래스로 스토리지 비용을 절감할 수 있습니다. 그러나 S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive 스토리지 클래스가 적합한지 확인하려면 다음을 고려하세요.

- 스토리지 오버헤드 요금 - 객체를 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 스토리지 클래스로 전환할 때 객체 관리를 위한 메타데이터를 수용하기 위해 고정된 양의 스토리지가 각 객체에 추가됩니다.
  - S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive에 아카이브된 각 객체에 대해 Amazon S3가 객체 이름 및 기타 메타데이터용으로 8KB의 스토리지를 사용합니다. Amazon S3은 이 메타데이터를 저장하므로 Amazon S3 API를 사용하여 아카이브된 객체의 실시간 목록을 얻을 수 있습니다. 자세한 내용은 Get Bucket(List Objects)을 참조하세요. 이 추가 스토리지에 대해 S3 Standard 요금이 청구됩니다.
  - S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive에 아카이브된 각 객체에 대해 Amazon S3가 인덱스 및 관련 메타데이터용으로 32KB의 스토리지를 추가합니다. 이 추가 데

이터는 객체를 식별하고 복원하는 데 필요합니다. 이 추가 스토리지에 대해 S3 Glacier Flexible Retrieval 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 아카이브 기간(일) - S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive는 장기 아카이브 솔루션입니다. 최소 스토리지 기간은 S3 Glacier Flexible Retrieval 스토리지 클래스의 경우 90일, S3 Glacier Deep Archive의 경우 180일입니다. 삭제한 객체가 최소 스토리지 기간보다 오래 아카이브되는 경우 Amazon S3 Glacier에 아카이브된 데이터를 삭제해도 요금이 부과되지 않습니다. 최소 기간 내에 아카이브된 객체를 삭제하거나 덮어쓸 경우 Amazon S3은 비례 할당으로 계산된 조기 삭제 요금을 부과합니다. 조기 삭제 요금에 대한 자세한 내용은 "Amazon S3 Glacier에서 90일 이전에 객체를 삭제하면 요금은 얼마나 청구되나요?"를 참조하세요. Amazon S3 FAQ에 나오는 질문입니다.
- S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive 전환 요청 요금 - S3 Glacier Flexible Retrieval 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. 작은 객체, 특히 128KB 미만의 객체를 포함하는 여러 객체를 보관하는 경우, 수명 주기 객체 크기 필터를 사용하여 전환 과정에서 작은 객체를 필터링하여 요청 비용을 줄이는 것이 좋습니다.
- S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive 데이터 복원 요금 - S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive는 자주 액세스하지 않는 데이터의 장기 아카이브를 위한 솔루션입니다. 데이터 복원 요금에 대한 자세한 내용은 "Amazon S3 Glacier에서 데이터를 검색하는 데 비용이 얼마나 드나요?"를 참조하세요. Amazon S3 FAQ에 나오는 질문입니다. Amazon S3 Glacier에서 데이터를 복원하는 방법에 대한 자세한 내용은 아카이브된 객체 복원 섹션을 참조하세요.

> **Note**
>
> S3 수명 주기 이전은 객체를 S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive로 비동기적으로 이전합니다. S3 수명 주기 구성 규칙의 전환 날짜와 실제 전환 작업 수행 날짜 간에 약간의 지연이 발생할 수 있습니다. 이 경우 규칙에 지정된 이전 날짜를 기준으로 이전한 스토리지 클래스의 기본 요금이 청구됩니다.

Amazon S3 제품 세부 정보 페이지에 자세한 요금 정보와 Amazon S3 객체 아카이브의 요금 계산 예제가 나와 있습니다. 자세한 정보는 다음 주제를 참조하세요.

- "Amazon S3 Glacier에 아카이브된 Amazon S3 객체에 대한 스토리지 요금은 어떻게 계산되나요?" Amazon S3 FAQ에 나옵니다.
- "Amazon S3 Glacier에서 90일 이전에 객체를 삭제하면 요금은 얼마나 청구되나요?" Amazon S3 FAQ에 나옵니다.
- "Amazon S3 Glacier에서 데이터를 검색하는 데 비용이 얼마나 드나요?" Amazon S3 FAQ에 나옵니다.
- 다양한 스토리지 클래스의 스토리지 비용에 대한 Amazon S3 요금

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 Flexible Retrieval 또는 S3 Glacier Deep Archive에 유지됩니다. (HeadObject 또는 GetObject API 작업 요청은 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive를 스토리지 클래스로 반환합니다.)

> **Note**
>
> 아카이브를 복원하면 아카이브(S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 요금) 및 임시로 복원한 복사본(S3 Standard 스토리지 요금) 모두에 대해 요금이 청구됩니다. 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하세요.

프로그래밍 방식으로 또는 Amazon S3 콘솔을 사용하여 객체 복사본을 복원할 수 있습니다. Amazon S3는 각 객체에 대한 복원 요청을 한 번에 하나씩만 처리합니다. 자세한 내용은 아카이브된 객체 복원 섹션을 참조하세요.

객체 만료

S3 수명 주기 구성에 이전 작업을 추가하여 Amazon S3가 수명 주기 종료 시 객체를 삭제하도록 지시할 수 있습니다. 수명 주기 구성을 기준으로 객체의 수명이 다하면 Amazon S3는 버킷이 어떤 S3 버전 관리 상태인지에 따라 `Expiration` 작업을 수행합니다.

- 버전 관리가 활성화되지 않은 버킷 - Amazon S3가 제거를 위해 객체를 대기열에 넣고 비동기식으로 제거하여 객체를 영구적으로 제거합니다.

- 버전 관리가 활성화된 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 - Amazon S3에서 버전 ID가 Null인 삭제 마커를 생성합니다. 삭제 마커는 버전 계층 구조에서 null 버전 ID로 모든 객체 버전을 대체함으로써 결과적으로 객체를 삭제합니다.

버전이 지정된 버킷(즉, 버전 관리를 사용하거나 버전 관리가 일시 중지된 버킷)의 경우, Amazon S3가 `Expiration` 작업을 처리하는 방식에 대한 몇 가지 고려 사항이 있습니다. 버전 관리가 활성화되거나 버전 관리가 일시 중지된 버킷에는 다음이 적용됩니다.

- 객체 만료가 객체의 현재 버전에만 적용됩니다(비최신 객체 버전에 영향이 없음).
- 객체 버전이 1개 이상이고 삭제 마커가 최신 버전이면 Amazon S3은 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 S3은 만료된 객체 삭제 마커를 제거합니다. `Expiration` 작업을 사용하여 Amazon S3에 만료된 객체 삭제 마커를 제거하도록 지시할 수도 있습니다. 예제는 버전 관리를 사용하는 버킷에서 만료된 객체 삭제 마커 제거을 참조하세요.
- `NoncurrentVersionExpiration` 작업 요소를 사용하여 Amazon S3가 비최신 객체 버전을 영구적으로 삭제하도록 지시할 수 있습니다. 이러한 삭제된 객체는 복구할 수 없습니다. 이 만료는 객체가 비최신 상태가 된 시점부터 특정 일수를 기준으로 할 수 있습니다. 일수 외에도 유지할 비최신 버전의 최대 개수를 제공할 수 있습니다(1~100개). 이 값은 Amazon S3가 특정 버전에서 관련 작업을 수행하기 전에 얼마나 많은 새로운 비최신 버전이 있어야 하는지를 지정합니다. 비최신 버전의 최대 개수를 지정하려면 `Filter` 요소 또한 제공해야 합니다. `Filter` 요소를 지정하지 않은 상태로 비최신 버전의 최대 수를 제공하는 경우 Amazon S3에서 `InvalidRequest` 오류가 발생합니다. `NoncurrentVersionExpiration` 작업 요소 사용에 대한 자세한 내용은 the section called “수명 주기 작업을 설명할 요소” 단원을 참조하세요.
- Amazon S3는 S3 객체 잠금 구성이 적용된 비최신 버전의 객체에 대해서는 어떠한 조치도 취하지 않습니다.
- 복제 상태가 `Pending`인 객체의 경우 Amazon S3는 최신 또는 비최신 버전의 객체에 대해 어떠한 조치도 취하지 않습니다.

자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하세요.

Important

S3 수명 주기 구성에 규칙이 여러 개인 경우 객체는 같은 날에 여러 가지 S3 수명 주기 작업을 수행할 수 있습니다. 이러한 경우 Amazon S3은 다음과 같은 일반 규칙을 따릅니다.

- 영구 삭제는 전환에 우선합니다.
- 이전은 삭제 마커 생성에 우선합니다.
- 객체에서 S3 Glacier Flexible Retrieval 및 S3 Standard-IA(또는 S3 One Zone-IA) 이전을 모두 사용할 수 있는 경우 Amazon S3가 S3 Glacier Flexible Retrieval 이전을 선택합니다.

예시는 중복 필터 및 충돌하는 수명 주기 작업의 예제 섹션을 참조하세요.

기존 객체 및 새 객체

버킷에 수명 주기 구성을 추가할 경우 구성 규칙이 기존 객체는 물론 나중에 추가하는 객체에도 적용됩니다. 예를 들어, 특정 접두사가 있는 객체를 생성한 후 30일이 경과하면 만료시키는 만료 작업을 포함하는 수명 주기 구성 규칙을 오늘 추가할 경우, Amazon S3는 30일 이상 경과하고 해당 접두사가 있는 모든 기존 객체를 삭제 대기열에 넣습니다.

Important

버킷 정책을 사용하여 S3 수명 주기 규칙에 따른 삭제 또는 이전을 방지할 수는 없습니다. 예를 들어, 버킷 정책이 모든 보안 주체에 대한 모든 작업을 거부하더라도 S3 수명 주기 구성은 여전히 정상적으로 작동합니다.

객체 만료 시기를 확인하는 방법

현재 버전의 객체가 만료될 예정인 시기를 알아보려면 HeadObject 또는 GetObject API 작업을 사용합니다. 이러한 API 작업은 현재 버전의 객체를 더 이상 캐시할 수 없는 날짜 및 시간을 제공하는 응답 헤더를 반환합니다.

Note

- 만료 날짜와 Amazon S3에서 객체를 제거하는 날짜 사이의 지연이 있을 수 있습니다. 만료된 객체와 관련된 만료 또는 스토리지 시간에 대해서는 요금이 청구되지 않습니다.

- 수명 주기 규칙을 업데이트, 비활성화 또는 삭제하기 전에 LIST API 작업(예: ListObjectsV2, ListObjectVersions, 및 ListMultipartUploads)을 사용하거나 S3 Inventory를 사용한 데이터 카탈로그화 및 분석를 사용하여 사용 사례에 따라 Amazon S3에서 적합한 객체를 전환 및 만료했는지 확인하세요.

최소 스토리지 기간 요금

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 Flexible Retrieval 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.

버킷에서 S3 수명 주기 구성 설정

버킷에 Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하여 Amazon S3 수명 주기 구성을 설정할 수 있습니다. S3 수명 주기 구성에 대해서는 객체 수명 주기 관리 섹션을 참조하세요.

> **Note**
>
> 디렉터리 버킷의 수명 주기 구성을 보거나 편집하려면 AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용합니다. 자세한 내용은 디렉터리 버킷의 S3 수명 주기 작업 섹션을 참조하세요.

S3 수명 주기 구성에서 수명 주기 규칙을 사용하여 객체의 수명 기간 동안 Amazon S3가 수행할 작업을 정의합니다. 예를 들어, 객체를 다른 스토리지 클래스로 이전하거나, 객체를 아카이브하거나, 지정된 시간이 지나면 객체를 만료(삭제)하는 규칙을 정의할 수 있습니다.

S3 수명 주기 고려 사항

수명 주기 구성을 설정하기 전에 다음 사항에 유의하세요.

수명 주기 구성 전파 지연

버킷에 S3 수명 주기 구성을 추가할 때 일반적으로 새로운 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.

이전 또는 만료 지연

수명 주기 규칙이 충족되는 시점과 규칙에 대한 작업이 완료되는 시점 사이에 지연이 있을 수 있습니다. 예를 들어, 수명 주기 규칙에 따라 1월 1일에 객체 세트가 만료된다고 가정해 보겠습니다. 1월 1일에 만료 규칙이 충족되었더라도 Amazon S3는 며칠 또는 몇 주가 지나야 이러한 객체를 실제로 삭제할 수 있습니다. 이러한 지연은 S3 Lifecycle이 이전 또는 만료를 위해 객체를 비동기적으로 대기시키기 때문에 발생합니다. 그러나 결제 변경 사항은 작업이 완료되지 않았더라도 수명 주기 규칙이 충족되는 즉시 적용됩니다. 자세한 내용은 결제 변경 사항 섹션을 참조하세요. 활성 수명 주기 규칙에 의한 업데이트의 영향을 모니터링하려면 the section called “수명 주기 규칙에 따라 취해진 조치를 모니터링하려면 어떻게 해야 하나요?” 섹션을 참조하세요.

수명 주기 규칙 업데이트, 사용 중지 또는 삭제

수명 주기 규칙을 사용 중지 또는 삭제하는 경우 Amazon S3가 새로운 객체에 대한 삭제 또는 이전 예약을 중지하기 전에 약간의 지연이 발생합니다. 이미 예약된 객체는 모두 예약이 취소되고 삭제 또는 전환되지 않습니다.

> **Note**
>
> 수명 주기 규칙을 업데이트, 비활성화 또는 삭제하기 전에 LIST API 작업(예: ListObjectsV2, ListObjectVersions, 및 ListMultipartUploads) 또는 S3 Inventory를 사용한 데이터 카탈로그화 및 분석를 사용하여 사용 사례에 따라 Amazon S3에서 적합한 객체를 이전 및 만료했는지 확인하세요. 수명 주기 규칙을 업데이트, 비활성화 또는 삭제하는 데 문제가 있는 경우 Amazon S3 수명 주기 문제 해결 섹션을 참조하세요.

기존 객체 및 새 객체

버킷에 수명 주기 구성을 추가할 경우 구성 규칙이 기존 객체는 물론 나중에 추가하는 객체에도 적용됩니다. 예를 들어, 특정 접두사가 있는 객체를 생성한 후 30일이 경과하면 만료시키는 만료 작업을 포함하는 수명 주기 구성 규칙을 오늘 추가할 경우, Amazon S3는 30일 이상 경과하고 해당 접두사가 있는 모든 기존 객체를 삭제 대기열에 넣습니다.

수명 주기 규칙의 영향 모니터링

활성 수명 주기 규칙에 의한 업데이트의 영향을 모니터링하려면 the section called “수명 주기 규칙에 따라 취해진 조치를 모니터링하려면 어떻게 해야 하나요?” 섹션을 참조하세요.

결제 변경 사항

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. 마찬가지로, 객체가 S3 Glacier Flexible Retrieval 스토리지 클래스로 즉시 전환되지 않은 경우에도 객체 전환 시간이 경과하는 순간부터 S3 Glacier Flexible Retrieval 스토리지 요금이 부과됩니다.

그러나 S3 Intelligent-Tiering 스토리지 클래스로의 수명 주기 전환은 예외입니다. 객체가 S3 Intelligent-Tiering 스토리지 클래스로 전환되기 전까지는 결제 변경이 발생하지 않습니다.

여러 규칙 또는 상충하는 규칙

S3 수명 주기 구성에 규칙이 여러 개인 경우 객체는 같은 날에 여러 가지 S3 수명 주기 작업을 수행할 수 있습니다. 이러한 경우 Amazon S3은 다음과 같은 일반 규칙을 따릅니다.

- 영구 삭제는 전환에 우선합니다.
- 이전은 삭제 마커 생성에 우선합니다.
- 객체에서 S3 Glacier Flexible Retrieval 및 S3 Standard-IA(또는 S3 One Zone-IA) 이전을 모두 사용할 수 있는 경우 Amazon S3가 S3 Glacier Flexible Retrieval 이전을 선택합니다.

예시는 중복 필터 및 충돌하는 수명 주기 작업의 예제 섹션을 참조하세요.

S3 수명 주기 구성을 설정하는 방법

범용 버킷에 Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하여 Amazon S3 수명 주기 구성을 설정할 수 있습니다.

AWS CloudFormation 템플릿에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS CloudFormation 템플릿 작업 및 AWS::S3::Bucket 섹션을 참조하세요.

S3 콘솔 사용

공유 접두사(공통 문자열로 시작하는 객체 이름) 또는 태그를 사용하여 버킷의 모든 객체 또는 일부 객체에 대해 수명 주기 규칙을 정의할 수 있습니다. 수명 주기 규칙에서는 현재 객체 버전과 비최신 객체 버전 관련 작업을 정의할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- 객체 수명 주기 관리
- S3 버전 관리로 여러 버전의 객체 유지

수명 주기 규칙 생성

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 수명 주기 규칙을 생성할 버킷의 이름을 선택합니다.
4. 관리(Management) 탭을 선택하고 수명 주기 규칙 생성(Create lifecycle rule)을 선택합니다.
5. 수명 주기 규칙 이름(Lifecycle rule name)에 규칙의 이름을 입력합니다.

   단, 버킷 내에서 고유한 이름을 갖도록 합니다.
6. 수명 주기 규칙의 범위를 선택합니다.

   - 특정 접두사나 태그가 있는 모든 객체에 이 수명 주기 규칙을 적용하려면 Limit the scope to specific prefixes or tags(범위를 특정 접두사 또는 태그로 제한)을 선택합니다.
     - 접두사로 범위를 제한하려면 접두사(Prefix)에 접두사를 입력합니다.
     - 태그로 범위를 제한하려면 태그 추가(Add tag)를 선택하고 태그 키와 값을 입력합니다.

     객체 이름 접두사에 대한 자세한 내용은 Amazon S3 객체 이름 지정 섹션을 참조하세요. 객체 태그에 대한 자세한 내용은 태그를 사용하여 스토리지 분류 섹션을 참조하세요.
   - 이 수명 주기 규칙을 버킷의 모든 객체에 적용하려면 이 규칙이 버킷의 모든 객체에 적용됨을 선택한 다음 이 규칙이 버킷의 모든 객체에 적용됨을 확인합니다를 선택합니다.
7. 객체 크기별로 규칙을 필터링하려면 최소 객체 크기 지정, 최대 객체 크기 지정 또는 두 옵션을 모두 선택할 수 있습니다.

   - 최소 객체 크기 또는 최대 객체 크기를 지정할 때 값은 0바이트보다 크고 최대 5TB여야 합니다. 이 값을 바이트, KB, MB 또는 GB 단위로 지정할 수 있습니다.
   - 둘 다 지정하는 경우 최대 객체 크기가 최소 객체 크기보다 커야 합니다.

> **Note**
>
> 최소 객체 크기 및 최대 객체 크기 필터는 지정된 값을 제외합니다. 예를 들어 최소 객체 크기가 128KB인 객체가 만료되도록 필터를 설정한 경우 정확히 128KB인 객체는 만료되지 않습니다. 대신 규칙은 크기가 128KB를 초과하는 객체에만 적용됩니다.

8. 수명 주기 규칙 작업(Lifecycle rule actions)에서 다음과 같은 수명 주기 규칙에서 수행할 작업을 선택합니다.

   - 스토리지 클래스 간에 객체의 현재 버전 이전
   - 스토리지 클래스 간에 객체의 이전 버전 이전
   - 객체의 현재 버전 만료

   > **Note**
   >
   > S3 버전 관리가 활성화되지 않은 버킷의 경우 현재 버전이 만료되면 Amazon S3에서 객체를 영구적으로 삭제합니다. 자세한 내용은 the section called “수명 주기 작업 및 버킷의 버전 관리 상태” 섹션을 참조하세요.

   - 객체의 이전 버전 영구 삭제
   - 만료된 삭제 마커 또는 불완전한 멀티파트 업로드 삭제

   선택한 작업에 따라 다른 옵션이 나타납니다.

9. 스토리지 클래스 간에 객체의 현재 버전을 이전하려면 스토리지 클래스 간에 객체의 현재 버전 이전에서 다음을 수행합니다.

   a. 스토리지 클래스 이전에서 이전할 대상 스토리지 클래스를 선택합니다. 가능한 이전 목록은 the section called “지원되는 수명 주기 전환” 섹션을 참조하세요. 사용자는 다음 스토리지 클래스에서 선택할 수 있습니다.

      - S3 Standard-IA
      - S3 Intelligent-Tiering
      - S3 One Zone-IA
      - S3 Glacier Instant Retrieval
      - S3 Glacier Flexible Retrieval

- S3 Glacier Deep Archive

b. 객체 생성일 후 일수(Days after object creation)에 생성일 후 객체를 이전할 일수를 입력합니다.

스토리지 클래스에 대한 자세한 정보는 Amazon S3 스토리지 클래스 이해 및 관리를 참조하세요. 최신 객체 버전 또는 이전 객체 버전의 이전 또는 두 버전 모두의 이전을 정의할 수도 있습니다. 버전 관리를 통해 하나의 버킷에서 객체의 여러 버전을 유지할 수 있습니다. 버전 관리에 대한 자세한 내용은 S3 콘솔 사용 섹션을 참조하세요.

> **Important**
>
> S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval 또는 Glacier Deep Archive 스토리지 클래스를 선택하면 객체가 Amazon S3에 그대로 유지됩니다. 별도의 Amazon S3 Glacier 서비스를 통해 객체에 직접 액세스할 수 없습니다. 자세한 내용은 Amazon S3 수명 주기를 사용하여 객체 전환 섹션을 참조하세요.

10. 스토리지 클래스 간에 비최신 객체 버전을 이전하려면 스토리지 클래스 간에 비최신 객체 버전 이전에서 다음을 수행합니다.

a. 스토리지 클래스 이전에서 이전할 대상 스토리지 클래스를 선택합니다. 가능한 이전 목록은 the section called “지원되는 수명 주기 전환” 섹션을 참조하세요. 사용자는 다음 스토리지 클래스에서 선택할 수 있습니다.

- S3 Standard-IA
- S3 Intelligent-Tiering
- S3 One Zone-IA
- S3 Glacier Instant Retrieval
- S3 Glacier Flexible Retrieval
- S3 Glacier Deep Archive

b. 객체가 비최신이 된 후의 일수에 생성일로부터 며칠 후에 객체를 전환할지 입력합니다.

11. 객체의 현재 버전을 만료하려면 Expire current versions of objects(객체의 현재 버전 만료)에서 Number of days after object creation(객체 생성 후 일 수에 일 수)를 입력합니다.

> **Important**
>
> 버전 관리를 사용하지 않는 버킷에서 만료 작업을 수행하면 Amazon S3에서는 객체를 영구적으로 제거합니다. 수명 주기 규칙에 대한 자세한 내용은 수명 주기 작업을 설명할 요소를 참조하세요.

12. 이전 객체 버전을 영구적으로 삭제하려면 이전 객체 버전 영구 삭제(Permanently delete noncurrent versions of objects) 아래의 객체가 이전 버전이 된 후 일수(Days after objects become noncurrent)에 일수를 입력합니다. Number of newer versions to retain(유지할 새 버전 수) 아래에 값을 입력하여 유지할 최신 버전 수를 선택적으로 지정할 수 있습니다.
13. 만료된 삭제 마커 또는 불완전한 멀티파트 업로드 삭제(Delete expired delete markers or incomplete multipart uploads)에서 만료된 객체 삭제 마커 삭제(Delete expired object delete markers) 및 불완전한 멀티파트 업로드 삭제(Delete incomplete multipart uploads)를 선택합니다. 그런 다음 멀티파트 업로드 시작일 이후 불완전한 멀티파트 업로드를 종료하고 정리하고자 하는 일수를 입력합니다.

    멀티파트 업로드에 대한 자세한 내용은 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 섹션을 참조하세요.
14. [Create rule]을 선택합니다.

    규칙에 아무런 오류가 없는 경우 Amazon S3에서 이를 사용 설정하고, 이 규칙은 수명 주기 규칙(Lifecycle rules)의 관리(Management) 탭에서 확인할 수 있습니다.

AWS CLI 사용

다음 AWS CLI 명령을 사용하여 S3 수명 주기 구성을 관리할 수 있습니다.

- `put-bucket-lifecycle-configuration`
- `get-bucket-lifecycle-configuration`
- `delete-bucket-lifecycle`

AWS CLI 설정에 대한 지침은 Amazon S3 API 참조의 AWS CLI를 사용하여 Amazon S3에서 개발을 참조하세요.

Amazon S3 수명 주기 구성은 XML 파일입니다. 그러나 AWS CLI를 사용할 때는 XML 형식을 지정할 수 없으며 대신 JSON 형식을 지정해야 합니다. 다음은 XML 수명 주기 구성과 AWS CLI 명령에서 지정할 수 있는 그에 상응하는 JSON 구성의 예시입니다.

다음 예제 S3 수명 주기 구성을 살펴보세요.

Example 예 1

Example

XML

```
<LifecycleConfiguration>
    <Rule>
        <ID>ExampleRule</ID>
        <Filter>
           <Prefix>documents/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
           <Days>365</Days>
           <StorageClass>GLACIER</StorageClass>
        </Transition>
        <Expiration>
             <Days>3650</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

JSON

```
{
    "Rules": [
        {
            "Filter": {
                "Prefix": "documents/"
            },
            "Status": "Enabled",
            "Transitions": [
                {
                    "Days": 365,
                    "StorageClass": "GLACIER"
```

```
                }
            ],
            "Expiration": {
                "Days": 3650
            },
            "ID": "ExampleRule"
        }
    ]
}
```

Example 예 2

Example

XML

```
<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>
```

JSON

```
{
    "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.

구성 테스트

1. 파일(예: *`lifecycle.json`*)에 JSON 수명 주기 구성을 저장합니다.
2. 다음 AWS CLI 명령을 실행하여 버킷에서 수명 주기 구성을 설정합니다. *`user input placeholders`*를 사용자의 정보로 대체합니다.

```
$ aws s3api put-bucket-lifecycle-configuration  \
--bucket amzn-s3-demo-bucket  \
--lifecycle-configuration file://lifecycle.json
```

3. 확인하려면 다음과 같이 `get-bucket-lifecycle-configuration` AWS CLI 명령을 사용해 S3 수명 주기 구성을 검색합니다.

```
$ aws s3api get-bucket-lifecycle-configuration  \
--bucket amzn-s3-demo-bucket
```

4. S3 수명 주기 구성을 삭제하려면 다음과 같이 `delete-bucket-lifecycle` AWS CLI 명령을 사용합니다.

```
aws s3api delete-bucket-lifecycle \
--bucket amzn-s3-demo-bucket
```

AWS SDK 사용

Java

AWS SDK for Java를 사용하여 버킷에서 S3 수명 주기 구성을 관리할 수 있습니다. S3 수명 주기 구성 관리에 대한 자세한 내용은 객체 수명 주기 관리 섹션을 참조하세요.

> **Note**
>
> 버킷에 S3 수명 주기 구성을 추가하면 Amazon S3가 버킷의 현재 수명 주기 구성(있는 경우)을 대체합니다. 구성을 업데이트하려면 검색한 후 원하는 대로 변경한 다음 수정한 구성을 버킷에 추가합니다.

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 S3에서는 기존 구성을 대체합니다.
- 구성을 다시 검색한 다음 규칙 수를 출력하여 규칙 개수가 올바른지 확인합니다.
- 수명 주기 구성을 삭제한 다음 다시 검색을 시도하여 삭제되었는지 확인합니다.

실제 예제를 만들고 테스트하는 방법에 대한 자세한 내용은 AWS SDK for Java 개발자 안내서의 시작하기를 참조하세요.

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

```
                }
            }
}
```

.NET

AWS SDK for .NET을 사용하여 버킷에서 S3 수명 주기 구성을 관리할 수 있습니다. 수명 주기 구성 관리에 대한 자세한 내용은 객체 수명 주기 관리 섹션을 참조하세요.

> **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.

다음 예제는 AWS SDK for .NET를 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 S3에서는 기존 수명 주기 구성을 대체합니다.
- 구성을 다시 검색한 다음 구성의 규칙 수를 인쇄하여 구성을 확인합니다.
- 수명 주기 구성을 삭제하고 삭제되었는지 확인합니다.

코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

```
using Amazon;
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);
        }
    }
}
```

Ruby

AWS::S3::BucketLifecycleConfiguration 클래스를 통해 AWS SDK for Ruby을 사용하여 버킷에서 S3 수명 주기 구성을 관리할 수 있습니다. S3 수명 주기 구성 관리에 대한 자세한 내용은 객체 수명 주기 관리 섹션을 참조하세요.

REST API 사용

Amazon Simple Storage Service API 참조의 다음 주제에서는 S3 수명 주기 구성과 관련된 REST API 작업을 설명합니다.

- PutBucketLifecycleConfiguration
- GetBucketLifecycleConfiguration
- DeleteBucketLifecycle

S3 수명 주기 문제 해결

S3 수명 주기 작업 시 발생할 수 있는 일반적인 문제는 the section called “수명 주기 문제 해결” 섹션을 참조하세요.

S3 수명 주기가 다른 버킷 구성과 상호 작용하는 방식

S3 수명 주기 구성 외에 다른 구성도 버킷에 연결할 수 있습니다. 이 섹션에서는 S3 수명 주기 구성과 다른 버킷 구성과의 관계를 설명합니다.

S3 수명 주기 및 S3 버전 관리

버전이 지정되지 않은 버킷과 버전 관리가 사용 설정된 버킷에 S3 수명 주기 구성을 추가할 수 있습니다. 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하세요.

버전 관리가 사용 설정된 버킷은 하나의 현재 객체 버전과 0개 이상의 최신이 아닌 객체 버전을 유지합니다. 최신 버전과 최신이 아닌 객체 버전에 대해 별도의 수명 주기 규칙을 정의할 수 있습니다.

자세한 내용은 수명 주기 구성의 요소 섹션을 참조하세요.

> **Important**
>
> S3 수명 주기 구성에 규칙이 여러 개인 경우 객체는 같은 날에 여러 가지 S3 수명 주기 작업을 수행할 수 있습니다. 이러한 경우 Amazon S3은 다음과 같은 일반 규칙을 따릅니다.
>
> - 영구 삭제는 전환에 우선합니다.
> - 이전은 삭제 마커 생성에 우선합니다.
> - 객체에서 S3 Glacier Flexible Retrieval 및 S3 Standard-IA(또는 S3 One Zone-IA) 전환을 모두 사용할 수 있는 경우 Amazon S3가 S3 Glacier 전환을 선택합니다.

예시는 중복 필터 및 충돌하는 수명 주기 작업의 예제 섹션을 참조하세요.

MFA 활성화 버킷에 대한 S3 수명 주기 구성

MFA가 구성된 다중 인증 버킷에 대한 S3 수명 주기 구성은 지원되지 않습니다. 자세한 내용은 MFA Delete 구성 섹션을 참조하세요.

S3 수명 주기 및 로깅

Amazon S3 수명 주기 작업은 AWS CloudTrail 객체 수준 로깅에서 캡처되지 않습니다. CloudTrail은 외부 Amazon S3 엔드포인트에 대한 API 요청을 캡처하는 반면, S3 수명 주기 작업은 내부 Amazon S3 엔드포인트를 사용하여 수행됩니다.

S3 버킷의 Amazon S3 서버 액세스 로그를 활성화하여 다른 스토리지 클래스로의 객체 이전 및 영구 삭제 또는 논리적 삭제를 초래하는 객체 만료와 같은 S3 수명 주기 관련 작업을 캡처할 수 있습니다. 자세한 내용은 the section called “서버 액세스 로깅” 섹션을 참조하세요.

버킷에서 로깅을 사용 설정한 경우 Amazon S3 서버 액세스 로그가 다음 작업의 결과를 보고합니다.

작업 로그	설명
`S3.EXPIRE.OBJECT`	Amazon S3는 수명 주기 `Expiration` 작업으로 인해 객체를 영구적으로 삭제합니다.
`S3.CREATE.DELETEMARKER`	Amazon S3는 버전 관리 활성화 버킷에 삭제 마커를 추가하여 현재 버전을 논리적으로 삭제합니다.
`S3.TRANSITION_SIA.OBJECT`	Amazon S3이 S3 Standard-IA 스토리지 클래스로 객체를 전환합니다.
`S3.TRANSITION_ZIA.OBJECT`	Amazon S3이 S3 One Zone-IA 스토리지 클래스로 객체를 전환합니다.
`S3.TRANSITION_INT.OBJECT`	Amazon S3가 S3 Intelligent-Tiering 스토리지 클래스로 객체를 전환합니다.

작업 로그	설명
S3.TRANSITION_GIR.OBJECT	Amazon S3가 S3 Glacier Instant Retrieval 스토리지 클래스로 객체 전환을 시작합니다.
S3.TRANSITION.OBJECT	Amazon S3가 S3 Glacier Flexible Retrieval 스토리지 클래스로 객체 전환을 시작합니다.
S3.TRANSITION_GDA.OBJECT	Amazon S3가 S3 Glacier Deep Archive 스토리지 클래스로 객체 전환을 시작합니다.
S3.DELETE.UPLOAD	Amazon S3가 불완전한 멀티파트 업로드를 중단합니다.

> **Note**
>
> Amazon S3 서버 액세스 로그 레코드는 최대한 전달되며, 모든 Amazon S3 요청의 완벽한 기록에 사용할 수는 없습니다.

S3 수명 주기 이벤트 알림 구성

S3 수명 주기 규칙에 따라 Amazon S3가 객체를 삭제하거나 다른 Amazon S3 스토리지 클래스로 이전할 때 알림을 받으려면 Amazon S3 이벤트 알림을 설정하면 됩니다.

다음과 같은 S3 수명 주기 이벤트에 대한 알림을 받을 수 있습니다.

- 이전 이벤트 – `s3:LifecycleTransition` 이벤트 유형을 사용하면 S3 수명 주기 구성에 의해 객체가 Amazon S3 스토리지 클래스 간에 이전될 때 알림을 받을 수 있습니다.
- 만료(삭제) 이벤트 - `LifecycleExpiration` 이벤트 유형을 사용하면 Amazon S3가 S3 수명 주기 구성에 따라 객체를 삭제할 때마다 알림을 받을 수 있습니다.

  만료 이벤트 유형에는 2가지가 있습니다.

  - `s3:LifecycleExpiration:Delete` 이벤트 유형은 버전이 지정되지 않은 버킷의 객체가 삭제될 때 알려줍니다. `s3:LifecycleExpiration:Delete`에서는 S3 수명 주기 구성으로 인해 객체 버전이 영구적으로 삭제될 때도 알려줍니다.

- `s3:LifecycleExpiration:DeleteMarkerCreated` 이벤트 유형은 버전이 지정된 버킷에 있는 객체의 현재 버전이 삭제된 후 S3 수명 주기가 삭제 마커를 생성할 때 알려줍니다. S3 수명 주기는 삭제 마커의 생성 시간을 현재 날짜의 00:00 UTC(자정)로 설정합니다. 이 생성 시간은 S3가 보내는 `s3:LifecycleExpiration:DeleteMarkerCreated` 알림의 이벤트 시간과 다를 수 있습니다. 자세한 내용은 the section called “객체 버전 삭제” 섹션을 참조하세요.

Amazon S3는 Amazon Simple Notification Service(Amazon SNS) 주제, Amazon Simple Queue Service(Amazon SQS) 대기열 또는 AWS Lambda 함수에 이벤트 알림을 게시할 수 있습니다. 자세한 내용은 Amazon S3 이벤트 알림 섹션을 참조하세요.

Amazon S3 이벤트 알림을 구성하는 방법에 대한 지침은 Amazon SQS, Amazon SNS 및 AWS Lambda를 사용한 이벤트 알림 활성화를 참조하세요.

다음은 Amazon S3에서 `s3:LifecycleExpiration:Delete` 이벤트를 게시하기 위해 전송하는 메시지의 예입니다. 자세한 내용은 the section called “이벤트 메시지 구조” 섹션을 참조하세요.

```
{
   "Records":[
      {
         "eventVersion":"2.3",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"1970-01-01T00:00:00.000Z",
         "eventName":"LifecycleExpiration:Delete",
         "userIdentity":{
            "principalId":"s3.amazonaws.com"
         },
         "requestParameters":{
            "sourceIPAddress":"s3.amazonaws.com"
         },
         "responseElements":{
            "x-amz-request-id":"C3D13FE58DE4C810",
            "x-amz-id-2":"FMyUVURIY8/IgAtTv8xRjskZQpcIZ9KG4V5Wp6S7S/
JRWeUWerMUE5JgHvANOjpD"
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"testConfigRule",
            "bucket":{
               "name":"amzn-s3-demo-bucket",
               "ownerIdentity":{
```

```
                "principalId":"A3NL1KOZZKExample"
             },
             "arn":"arn:aws:s3:::amzn-s3-demo-bucket"
          },
          "object":{
             "key":"expiration/delete",
             "sequencer":"0055AED6DCD90281E5",
          }
       }
    }
 ]
}
```

Amazon S3가 `s3:LifecycleTransition` 이벤트를 게시하기 위해 전송하는 메시지에는 다음 정보도 포함됩니다.

```
"lifecycleEventData":{
    "transitionEventData": {
        "destinationStorageClass": the destination storage class for the object
    }
}
```

수명 주기 구성의 요소

S3 수명 주기 구성은 Amazon S3가 객체 수명 주기 동안 수행하는 작업을 설명하는 다양한 요소를 포함하는 수명 주기 규칙으로 구성됩니다. Amazon S3 수명 주기 구성을 하나 이상의 수명 주기 규칙으로 구성된 XML로 지정합니다. 이때 각 규칙은 하나 이상의 요소로 이루어집니다.

```
<LifecycleConfiguration>
    <Rule>
         <Element>
    </Rule>
    <Rule>
         <Element>
         <Element>
    </Rule>
</LifecycleConfiguration>
```

각 규칙은 다음과 같은 요소로 구성됩니다.

- 규칙 메타데이터에는 규칙 ID와 규칙의 활성화 여부를 나타내는 상태가 포함됩니다. 규칙이 사용 중지되면 Amazon S3은 규칙에 지정된 작업을 수행하지 않습니다.
- 규칙이 적용되는 객체를 식별하는 필터. 객체 크기, 객체 키 접두사, 하나 이상의 객체 태그 또는 필터 조합을 사용하여 필터를 지정할 수 있습니다.
- Amazon S3이 객체 수명 주기의 특정 날짜 또는 기간 동안 지정된 작업을 수행하기를 원하는 하나 이상의 전환 또는 만료 작업

주제

- ID 요소
- 상태 요소
- 필터 요소
- 수명 주기 작업을 설명할 요소
- 수명 주기 규칙에 필터 추가

다음 섹션에서는 S3 수명 주기 구성의 XML 요소에 대해 설명합니다. 구성에 대한 예시는 S3 수명 주기 구성의 예제 섹션을 참조하십시오.

ID 요소

한 S3 수명 주기 구성에서 최대 1,000개의 규칙을 설정할 수 있으며, 이 제한은 조정할 수 없습니다. `<ID>` 요소는 각 규칙을 고유하게 식별합니다. ID 길이는 255자로 제한됩니다.

상태 요소

이때 `<Status>` 요소 값은 `Enabled` 또는 `Disabled`가 될 수 있습니다. 규칙이 사용 중지되면 Amazon S3은 규칙에 정의된 작업을 수행하지 않습니다.

필터 요소

S3 수명 주기 규칙은 규칙에서 지정하는 `<Filter>` 요소에 기반하여 버킷 내 객체의 모든 하위 집합 또는 하나의 하위 집합에 적용될 수 있습니다.

키 접두사, 객체 태그 또는 이 둘의 조합으로 객체를 필터링할 수 있습니다. 조합하는 경우 Amazon S3가 논리적 `AND`를 사용하여 필터를 조합합니다. 필터에 대한 예제 및 자세한 정보는 수명 주기 규칙에 필터 추가 섹션을 참조하세요.

- 키 접두사를 사용하여 필터 지정 – 이 예시는 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.txt에 적용됩니다. 이 규칙은 객체 example.jpg에는 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 Amazon S3 객체 이름 지정 섹션을 참조하십시오.

- 객체 태그에 기반한 필터 지정 - 다음 예시에서 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. 다음 예시와 같이 <And> 요소로 태그를 묶어야 합니다. 규칙은 2개의 태그(특정 태그 키와 값)를 가진 객체에서 수명 주기 작업을 수행하도록 Amazon S3에 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 태그 2개가 모두 지정된 객체에 적용됩니다. Amazon S3가 논리적 AND를 수행합니다. 다음 사항에 유의하세요.

- 각 태그는 키 및 값 모두와 정확히 일치해야 합니다. <Key> 요소만 지정하고 <Value> 요소는 지정하지 않는 경우, 규칙은 태그 키와 일치하고 값이 지정되지 않은 객체에만 적용됩니다.
- 이 규칙은 규칙에 모든 태그가 지정되는 객체의 하위 집합에 적용됩니다. 객체에 추가 태그가 지정되어 있더라도 규칙은 계속 적용됩니다.

> **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 모든 필터 요소를 <And> 요소로 묶어야 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 S3가 논리적 AND를 사용하여 이러한 필터를 조합합니다. 즉, 규칙은 특정 키 접두사와 특정 태그를 가진 객체의 하위 집합에 적용됩니다. 필터는 단 1개의 접두사와 0개 이상의 태그를 가질 수 있습니다.

- 빈 필터(empty filter)를 지정할 수도 있는데, 이 경우에 규칙은 버킷 내 모든 객체에 적용됩니다.

```
<LifecycleConfiguration>
```

```
    <Rule>
        <Filter>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

- 객체 크기(object size)별로 규칙을 필터링하려면 최소 크기(ObjectSizeGreaterThan) 또는 최대 크기(ObjectSizeLessThan)를 지정하거나 객체 크기 범위를 지정할 수 있습니다.

  객체 크기 값은 바이트 단위입니다. 기본적으로 128KB 미만의 객체는 최소 크기(ObjectSizeGreaterThan) 또는 최대 크기(ObjectSizeLessThan)를 더 작게 지정하지 않는 한 스토리지 클래스로 전환되지 않습니다. 자세한 내용은 예제: 128KB 미만의 객체 전환 허용 단원을 참조하십시오.

```
                <LifecycleConfiguration>
    <Rule>
        <Filter>
            <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

> **Note**
>
> ObjectSizeGreaterThan 및 ObjectSizeLessThan 필터는 지정된 값을 제외합니다. 예를 들어 128KB에서 1024KB로 객체를 설정하여 S3 Standard 스토리지 클래스에서 S3 Standard-IA 스토리지 클래스로 이동할 때 크기가 정확히 1024KB와 128KB인 객체는 S3 Standard-IA로 이전되지 않습니다. 대신 규칙은 크기가 128KB보다 크고 1024KB 미만인 객체에만 적용됩니다.

객체 크기 범위를 지정하는 경우 ObjectSizeGreaterThan 정수는 ObjectSizeLessThan 값보다 작아야 합니다. 둘 이상의 필터를 사용하는 경우 <And> 요소로 필터를 묶어야 합니다. 다음 예에서는 500~64,000바이트 범위의 객체를 지정하는 방법을 보여 줍니다.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
            <And>
                <Prefix>key-prefix</Prefix>
                <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
                <ObjectSizeLessThan>64000</ObjectSizeLessThan>
            </And>
        </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.

- **`Transition`** 작업 요소 – 한 스토리지 클래스에서 다른 스토리지 클래스로 객체를 전환하려면 `Transition` 작업을 지정합니다. 객체 전환에 대한 자세한 내용은 Supported transitions 섹션을 참조하십시오. 객체의 수명 주기에서 지정된 날짜 또는 기간이 도래하면 Amazon S3이 전환을 수행합니다.

  버전이 지정된 버킷(버전 관리를 사용하거나 버전 관리가 일시 중지된 버킷)의 경우 `Transition` 작업은 현재 객체 버전에 적용됩니다. 최신이 아닌 버전을 관리하기 위해 Amazon S3에서는 `NoncurrentVersionTransition` 작업을 정의합니다(이 주제의 뒷부분에서 설명).

- **`Expiration`** 작업 요소 - `Expiration` 작업은 규칙에서 식별된 객체를 만료시키고 모든 Amazon S3 스토리지 클래스의 적격 객체에 적용됩니다. 스토리지 클래스에 대한 자세한 정보는 Amazon S3 스토리지 클래스 이해 및 관리를 참조하십시오. Amazon S3에 의해 만료된 모든 객체는 사용할 수 없게 되고, 버킷의 버전 관리 상태에 따라 객체의 영구 삭제 여부가 결정됩니다.
  - 버전이 지정되지 않은 버킷 - `Expiration` 작업을 수행하면 Amazon S3가 객체를 영구적으로 제거합니다.

- 버전이 지정된 버킷 - 버전이 지정된 버킷(즉, 버전 관리를 사용하거나 버전 관리가 일시 중지된 버킷)의 경우, Amazon S3이 `Expiration` 작업을 처리하는 방식에 대한 몇 가지 고려 사항이 있습니다. 버전 관리가 활성화되거나 버전 관리가 일시 중지된 버킷에는 다음이 적용됩니다.
  - `Expiration` 작업은 최신 버전에만 적용됩니다(비 최신 버전에는 영향이 없음).
  - 객체 버전이 1개 이상이고 삭제 마커가 최신 버전이면 Amazon S3은 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 S3은 만료된 객체 삭제 마커를 제거합니다. 만료 행위를 사용하여 Amazon S3에게 만료된 객체 삭제 마커를 제거하도록 지시할 수도 있습니다. 관련 예제는 버전 관리를 사용하는 버킷에서 만료된 객체 삭제 마커 제거 섹션을 참조하십시오

  자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하십시오.

  Amazon S3을 설정하여 만료를 관리할 때는 다음 사항도 고려하십시오.

  - 버전 관리를 사용하는 버킷

    최신 객체 버전이 삭제 마커가 아닌 경우 Amazon S3은 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 S3가 버전 ID가 `null`인 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 최신이 아닌 객체 버전을 관리하는 데 사용할 수 있는 다음 작업을 제공합니다.

- **`NoncurrentVersionTransition`** 작업 요소 – 이 작업을 사용하여 언제 Amazon S3에서 객체를 지정된 스토리지 클래스로 전환할지 지정합니다. 이 전환은 객체가 비최신(`<NoncurrentDays>`) 상태가 된 시점부터 특정 일수를 기준으로 할 수 있습니다. 일수 외에도 유지할 비최신 버전(`<NewerNoncurrentVersions>`)의 개수를 지정할 수 있습니다(1~100개). 이 값은 Amazon S3가 특정 버전을 전환하기 전에 얼마나 많은 새로운 비최신 버전이 있어야 하는지를 판단합니다. Amazon S3는 보존하도록 지정된 수를 초과하는 모든 추가 비최신 버전을 전환합니다. 전환을 수행하려면 `<NoncurrentDays>` 및 `<NewerNoncurrentVersions>` 값을 모두 초과해야 합니다.

유지할 비최신 버전의 개수를 지정하려면 <Filter> 요소 또한 제공해야 합니다. <Filter> 요소를 지정하지 않은 상태로 유지할 비최신 버전의 수를 지정하는 경우 Amazon S3에서 InvalidRequest 오류가 발생합니다.

객체 전환에 대한 자세한 내용은 Supported transitions 섹션을 참조하십시오. NoncurrentVersionTransition 작업에서 일 수를 지정할 때 Amazon S3가 날짜를 계산하는 방법에 대한 자세한 내용은 수명 주기 규칙: 객체 기간 기반 섹션을 참조하십시오.

- **NoncurrentVersionExpiration** 작업 요소 – 이 작업을 사용하여 Amazon S3가 비최신 객체 버전을 영구적으로 삭제하도록 지시합니다. 이러한 삭제된 객체는 복구할 수 없습니다. 이 만료는 객체가 비최신(<NoncurrentDays>) 상태가 된 시점부터 특정 일수를 기준으로 할 수 있습니다. 일수 외에도 유지할 비최신 버전(<NewerNoncurrentVersions>)의 개수를 지정할 수 있습니다(1~100개). 이 값은 Amazon S3가 특정 버전을 만료하기 전에 얼마나 많은 새로운 비최신 버전이 있어야 하는지를 지정합니다. Amazon S3는 보존하도록 지정된 수를 초과하는 모든 추가 비최신 버전을 영구적으로 삭제합니다. 삭제를 수행하려면 <NoncurrentDays> 및 <NewerNoncurrentVersions> 값을 모두 초과해야 합니다.

  유지할 비최신 버전의 개수를 지정하려면 <Filter> 요소 또한 제공해야 합니다. <Filter> 요소를 지정하지 않은 상태로 유지할 비최신 버전의 수를 지정하는 경우 Amazon S3에서 InvalidRequest 오류가 발생합니다.

  최신 버전이 아닌 객체를 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. 비최신 버전이 된 지 5일 후인 2014년 1월 8일 0시(UTC)에 만료 수명 주기 규칙이 실행되어 photo.gif(버전 ID 111111)를 영구 삭제합니다.

  NoncurrentVersionExpiration 작업에서 일 수를 지정할 때 Amazon S3가 날짜를 계산하는 방법에 대한 자세한 내용은 수명 주기 규칙: 객체 기간 기반 섹션을 참조하십시오.

> **Note**
>
> 객체 만료 수명 주기 구성은 완료되지 않은 멀티파트 업로드를 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 미완료 멀티파트 업로드를 중지하거나 만료된 객체 삭제 마커를 제거하도록 Amazon S3에 지시할 수 있습니다.

- **`AbortIncompleteMultipartUpload`** 작업 요소 - 이 요소를 사용해 멀티파트 업로드가 진행 상태에 있길 원하는 최대 시간(일수)을 설정합니다. 해당 멀티파트 업로드(수명 주기 규칙에서 지정된 키 이름 `prefix`에 의해 결정)가 사전 지정된 기간 내에 성공적으로 완료되지 않으면, Amazon S3는 불완전 멀티파트 업로드를 중지합니다. 자세한 내용은 멀티파트 업로드 중단 단원을 참조하십시오.

  > **Note**
  >
  > 객체 태그를 사용하는 필터가 있는 규칙에서는 이 수명 주기 작업을 지정할 수 없습니다.

- **`ExpiredObjectDeleteMarker`** 작업 요소 – 버전 관리가 활성화된 버킷에서 비최신 버전이 없는 삭제 마커는 만료된 객체 삭제 마커라고 부릅니다. 이 수명 주기 작업을 사용하여 Amazon S3에 만료된 객체 삭제 마커를 제거하도록 지시할 수 있습니다. 예시는 버전 관리를 사용하는 버킷에서 만료된 객체 삭제 마커 제거을 확인하세요.

  > **Note**
  >
  > 객체 태그를 사용하는 필터가 있는 규칙에서는 이 수명 주기 작업을 지정할 수 없습니다.

Amazon S3이 최신이 아닌 객체 버전이 된 기간을 계산하는 방법

버전 관리가 사용 설정된 버킷에는 여러 버전의 객체가 있을 수 있습니다. 항상 하나의 현재 버전과 0개 이상의 이전 버전이 있습니다. 새로 객체를 업로드할 때마다 새로 업로드된 후임 버전이 최신 버전이 되고, 기존의 최신 버전은 비 최신 버전이 됩니다. 최신이 아닌 객체 버전이 된 기간을 계산하기 위해 Amazon S3은 후속 객체가 생성된 날짜를 확인합니다. Amazon S3은 후속 객체의 생성 일수를 사용하여 최신이 아닌 객체 버전이 된 기간을 계산합니다.

S3 수명 주기 구성을 사용한 이전 버전의 객체 복원

이전 버전 복원의 설명과 같이 다음 두 가지 방법 중 하나를 사용하여 이전 버전의 객체를 검색할 수 있습니다.

- 방법 1 – 비최신 버전의 객체를 동일한 버킷으로 복사합니다. 복사된 객체는 해당 객체의 최신 버전이 되고 모든 객체 버전은 유지됩니다.
- 방법 2 – 객체의 현재 버전을 영구적으로 삭제합니다. 최신 객체 버전을 삭제하는 것은 실제로 해당 객체의 비 최신 버전을 최신 버전으로 만드는 것과 같습니다.

버전 관리가 활성화된 버킷에 S3 수명 주기 구성 규칙을 사용할 경우 방법 1을 사용하는 것이 모범 사례입니다.
S3 수명 주기는 최종 일관성 모델로 작동합니다. 영구적으로 삭제한 최신 버전은 변경 사항이 모든 Amazon S3 시스템에 전파될 때까지 사라지지 않을 수 있습니다. (따라서 Amazon S3가 이 삭제를 일시적으로 인식하지 못할 수 있습니다.) 그 동안 최신 버전이 아닌 객체를 만료시키도록 구성된 수명 주기 규칙에 의해 복구하려는 객체를 포함하여 최신 버전이 아닌 객체가 영구 제거될 수 있습니다. 따라서 방법 1에서 권장하는 대로 이전 버전을 복사하는 것이 더 안전합니다.

수명 주기 작업 및 버킷의 버전 관리 상태

수명 주기 규칙: 객체 기간 기반

객체 생성 또는 수정 시점을 기준으로 Amazon S3가 지정된 작업을 수행할 수 있는 기간(일수)을 지정할 수 있습니다.

S3 수명 주기 구성에서 `Transition` 및 `Expiration` 작업 일수를 지정할 경우, 다음에 유의하십시오.

- 지정하는 값은 작업이 이루어질 객체 생성 이후의 일수입니다.
- Amazon S3는 객체 생성 시간을 기준으로 규칙에 지정된 일수를 계산하며, 익일 자정(UTC)으로 계산된 시간을 상향 조정합니다. 예를 들어, 객체가 2014년 1월 15일 오전 10시 30분에 생성되고, 전환 규칙에 3일을 지정한 경우 객체의 전환일은 2014년 1월 19일 0시가 됩니다.

> **Note**
>
> Amazon S3은 각 객체에 대해 최종 수정일만 유지합니다. 예를 들어, Amazon S3 콘솔은 객체의 속성 창에 마지막 수정 날짜를 표시합니다. 새 객체를 처음 생성할 때 이 날짜는 객체가 생성된 날짜가 되고, 객체를 변경하면 날짜도 이에 따라 바뀝니다. 따라서 생성 날짜는 마지막 수정 날짜와 동일합니다.

수명 주기 구성에서 `NoncurrentVersionTransition` 및 `NoncurrentVersionExpiration` 작업의 일 수를 지정할 경우 다음에 유의하십시오.

- 지정하는 값은 Amazon S3가 지정된 객체에 대해 작업을 수행할 기간으로 객체 버전이 비최신이 된 이후의 일수(즉, 객체를 덮어쓰거나 삭제한 후 경과한 일수)입니다.
- Amazon S3는 후속 버전의 새 객체가 생성된 시간을 기준으로 규칙에 지정된 일수를 계산하며, 익일 자정(UTC)으로 계산된 시간을 상향 조정합니다. 예를 들어 버킷에서 2014년 1월 1일 오전 10시30분(UTC)에 생성된 객체가 현재 버전이라고 가정합시다. 현재 버전을 대체한 새 버전이 2014년 1월 15일 오전 10시 30분(UTC)에 생성되었고 전환 규칙에 3일을 지정했다면, 해당 객체의 전환일은 2014년 1월 19일 0시(UTC)로 계산됩니다.

수명 주기 규칙: 지정된 날짜 기반

S3 수명 주기 규칙에서 작업을 지정할 때 Amazon S3에서 작업을 수행하기를 원하는 날짜를 지정할 수 있습니다. 특정 데이터가 도착하면 Amazon S3이 필터 기준에 부합하는 모든 객체에 작업을 적용합니다.

과거 날짜를 사용하여 S3 수명 주기 작업을 지정할 경우 기준을 충족하는 모든 객체가 즉시 해당 수명 주기 작업의 대상이 됩니다.

> **Important**
>
> 날짜 기반 작업은 일회용 작업이 아닙니다. Amazon S3은 날짜가 경과한 이후에도 해당 규칙 상태가 `Enabled`로 유지되는 한 데이터 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 Amazon S3는 사용자가 버킷에서 만드는 새 객체도 모두 만료시킵니다. 수명 주기 작업을 중지하려면 작업을 수명 주기 규칙에서 제거하거나, 규칙을 비활성화하거나, 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 확인, 비활성화 또는 삭제할 수는 있습니다.

수명 주기 규칙에 필터 추가

필터는 규칙이 적용되는 객체를 지정하는 데 사용할 수 있는 선택적 수명 주기 규칙 요소입니다.

다음 요소를 사용하여 객체를 필터링할 수 있습니다.

키 접두사

접두사를 기준으로 객체를 필터링할 수 있습니다. 다른 접두사를 사용하여 객체의 하위 집합에 수명 주기 작업을 적용하려면 각 작업에 대한 별도의 규칙을 생성합니다.

객체 태그

하나 이상의 태그를 기준으로 객체를 필터링할 수 있습니다. 각 태그는 키와 값이 모두 정확히 일치해야 하며, 복수의 태그를 지정할 경우 각각의 태그 키가 고유해야 합니다. 객체 태그가 여러 개 있는 필터는 지정된 모든 태그가 있는 객체의 하위 집합에 적용됩니다. 객체에 추가 태그가 지정되어 있더라도 필터는 계속 적용됩니다.

> **Note**
>
> `Key` 요소만 지정하고 `Value` 요소는 지정하지 않는 경우, 규칙은 태그 키와 일치하고 값이 지정되지 않은 객체에만 적용됩니다.

최소 또는 최대 객체 크기

크기를 기준으로 객체를 필터링할 수 있습니다. 최소 크기(`ObjectSizeGreaterThan`) 또는 최대 크기(`ObjectSizeLessThan`)를 지정하거나 동일한 필터에서 객체 크기 범위를 지정할 수 있습니다. 객체 크기 값은 바이트 단위입니다. 최대 필터 크기는 5TB입니다. Amazon S3는 기본 최소 객체 크기를 수명 주기 구성에 적용합니다. 자세한 내용은 예제: 128KB 미만의 객체 전환 허용 섹션을 참조하세요.

서로 다른 필터 요소를 결합할 수 있으며, 이 경우 Amazon S3는 논리적 `AND`를 사용합니다.

필터링 예제

다음은 서로 다른 필터 요소를 사용할 수 있는 방법을 보여주는 예제입니다.

- 키 접두사를 사용하여 필터 지정 – 이 예시는 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>
```

> **Note**
>
> 동일한 문자로 시작하는 접두사가 하나 이상 있는 경우 필터에 후행 슬래시(/)가 없는 부분 접두사를 지정하여 규칙에 해당 접두사를 모두 포함할 수 있습니다. 예를 들어, 다음과 같은 접두사가 있다고 가정해 보세요.
>
> ```
> sales1999/
>                sales2000/
>                sales2001/
> ```
>
> 규칙에 세 접두사를 모두 포함하려면 수명 주기 규칙에서 `sales`를 접두사로 지정합니다.

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 Amazon S3 객체 이름 지정 섹션을 참조하십시오.

- 객체 태그에 기반한 필터 지정 - 다음 예시에서 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. 다음 예시와 같이 <And> 요소로 태그를 묶어야 합니다. 규칙은 2개의 태그(특정 태그 키와 값)를 가진 객체에서 수명 주기 작업을 수행하도록 Amazon S3에 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 태그 2개가 모두 지정된 객체에 적용됩니다. Amazon S3가 논리적 AND를 수행합니다. 다음 사항에 유의하세요.

- 각 태그는 키 및 값 모두와 정확히 일치해야 합니다. <Key> 요소만 지정하고 <Value> 요소는 지정하지 않는 경우, 규칙은 태그 키와 일치하고 값이 지정되지 않은 객체에만 적용됩니다.
- 이 규칙은 규칙에 모든 태그가 지정되는 객체의 하위 집합에 적용됩니다. 객체에 추가 태그가 지정되어 있더라도 규칙은 계속 적용됩니다.

> **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 모든 필터 요소를 <And> 요소로 묶어야 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 S3가 논리적 AND를 사용하여 이러한 필터를 조합합니다. 즉, 규칙은 특정 키 접두사와 특정 태그를 가진 객체의 하위 집합에 적용됩니다. 필터는 단 1개의 접두사와 0개 이상의 태그를 가질 수 있습니다.

- 빈 필터 지정 – 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>
```

- >객체 크기 필터 지정 - 객체 크기별로 규칙을 필터링하려면 최소 크기(ObjectSizeGreaterThan) 또는 최대 크기(ObjectSizeLessThan)를 지정하거나 객체 크기 범위를 지정하면 됩니다.

  객체 크기 값은 바이트 단위입니다. 최대 필터 크기는 5TB입니다. 일부 스토리지 클래스에는 최소 객체 크기 제한이 있습니다. 자세한 내용은 Amazon S3 스토리지 클래스 비교 섹션을 참조하세요.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
            <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

> **Note**
>
> `ObjectSizeGreaterThan` 및 `ObjectSizeLessThan` 필터는 지정된 값을 제외합니다. 예를 들어 128KB에서 1024KB로 객체를 설정하여 S3 Standard 스토리지 클래스에서 S3 Standard-IA 스토리지 클래스로 이동할 때 크기가 정확히 1024KB와 128KB인 객체는 S3 Standard-IA로 이전되지 않습니다. 대신 규칙은 크기가 128KB보다 크고 1024KB 미만인 객체에만 적용됩니다.

객체 크기 범위를 지정하는 경우 `ObjectSizeGreaterThan` 정수는 `ObjectSizeLessThan` 값보다 작아야 합니다. 둘 이상의 필터를 사용하는 경우 <And> 요소로 필터를 묶어야 합니다. 다음 예에서는 500~64,000바이트 범위의 객체를 지정하는 방법을 보여 줍니다.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
            <And>
                <Prefix>key-prefix</Prefix>
                <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
                <ObjectSizeLessThan>64000</ObjectSizeLessThan>
            </And>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

S3가 수명 주기 구성의 충돌을 처리하는 방법

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 일반 규칙의 예외는 S3 Intelligent-Tiering 스토리지 클래스입니다. S3 Intelligent-Tiering은 S3 Glacier 및 S3 Glacier Deep Archive 스토리지 클래스를 제외한 모든 스토리지 클래스보다 S3 수명 주기에서 우선 적용됩니다.

S3 수명 주기 구성에 규칙이 여러 개인 경우 객체는 같은 날에 여러 가지 S3 수명 주기 작업을 수행할 수 있습니다. 이러한 경우 Amazon S3은 다음과 같은 일반 규칙을 따릅니다.

- 영구 삭제는 전환에 우선합니다.
- 이전은 삭제 마커 생성에 우선합니다.
- 객체에서 S3 Glacier Flexible Retrieval 및 S3 Standard-IA(또는 S3 One Zone-IA) 이전을 모두 사용할 수 있는 경우 Amazon S3가 S3 Glacier Flexible Retrieval 이전을 선택합니다.

중복 필터 및 충돌하는 수명 주기 작업의 예제

S3 수명 주기 구성을 지정하여 중복 접두사 또는 작업을 지정할 수 있습니다. 다음 예에서는 Amazon S3가 어떻게 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 키 이름 접두사(`logs/`)를 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>
```

이 경우 충돌이 없으므로 Amazon S3는 `logs/` 접두사가 있는 객체를 생성 후 30일이 지나면 S3 Standard-IA 스토리지 클래스로 전환합니다. 생성 후 1년이 지나면 객체가 삭제됩니다.

Example 2: 서로 충돌하는 수명 주기 작업

이 구성 예에는 객체 수명 주기 중 같은 시각에 동일한 객체 집합에 대해 두 가지 작업을 수행하도록 Amazon S3에 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 S3가 이 객체들에 대해 만료 작업을 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 S3 수명 주기 작업이 적용됩니다. 한 규칙은 생성 후 10일이 지난 객체를 전환하도록 Amazon S3에 지시하고, 다른 규칙은 생성 후 365일이 지난 객체를 전환하도록 Amazon S3에 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 S3은 생성 후 10일 지난 객체를 전환하는 쪽을 선택합니다.

Example 4: 태그 기반 필터링과 그로 인해 서로 충돌하는 수명 주기 작업

다음과 같이 각각 태그 필터를 지정하는 두 가지 규칙이 있는 S3 수명 주기 구성이 있다고 가정해 보겠습니다.

- 규칙 1은 태그 기반 필터(`tag1/value1`)를 지정합니다. 이 규칙은 생성 후 365일이 지난 객체를 S3 Glacier Flexible Retrieval 스토리지 클래스로 전환하도록 Amazon S3에 지시합니다.
- 규칙 2는 태그 기반 필터(`tag2/value2`)를 지정합니다. 이 규칙은 생성 후 14일이 지난 객체를 만료 처리하도록 Amazon S3에 지시합니다.

S3 수명 주기 구성은 다음 예에 나와 있습니다.

```
<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>value2</Value>
      </Tag>
    </Filter>
    <Status>Enabled</Status>
    <Expiration>
      <Days>14</Days>
```

```
    </Expiration>
   </Rule>
</LifecycleConfiguration>
```

객체에 두 태그가 모두 있는 경우 Amazon S3는 따라야 할 규칙을 결정해야 합니다. 이 경우, Amazon S3은 생성 후 14일 지난 객체를 만료 처리합니다. 객체가 제거되므로 전환 작업이 적용되지 않습니다.

S3 수명 주기 구성의 예제

이 섹션에서는 S3 수명 주기 구성의 예를 제공합니다. 각 예제에서는 예제 시나리오별로 XML을 지정하는 방법을 보여줍니다.

주제

- 생성 후 1일 이내에 모든 객체 아카이브
- 수명 주기 규칙 임시 비활성화
- 객체의 수명 주기 동안 스토리지 클래스 티어 다운
- 여러 규칙 지정
- 버전 관리를 사용하는 버킷에 대한 수명 주기 규칙 지정
- 버전 관리를 사용하는 버킷에서 만료된 객체 삭제 마커 제거
- 멀티파트 업로드 중단을 위한 수명 주기 구성
- 데이터가 없는 비최신 객체 만료
- 예제: 128KB 미만의 객체 전환 허용

생성 후 1일 이내에 모든 객체 아카이브

각 S3 수명 주기 규칙에는 S3 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 S3 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 S3에 다음을 수행하도록 지시하는 두 가지 작업을 지정합니다.

  - 생성 후 365일(1년)이 지난 객체를 S3 Glacier Flexible Retrieval 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>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 Date와 Days를 모두 사용할 수는 없습니다.

- S3 수명 주기 규칙을 버킷의 모든 객체에 적용하려면 빈 접두사를 지정합니다. 다음 구성에서 규칙은 생성 후 0일이 지나면 객체를 S3 Glacier Flexible Retrieval 스토리지 클래스로 전환하도록 Amazon S3에 지시하는 Transition 작업을 지정합니다. 이 규칙은 객체를 생성한 후 UTC 자정에 S3 Glacier Flexible Retrieval에 아카이브할 수 있음을 의미합니다. 수명 주기 제한 사항에 대한 자세한 내용은 전환에 대한 제약 조건 및 고려 사항 섹션을 참조하세요.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Archive all object same-day upon creation</ID>
    <Filter>
      <Prefix></Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>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 S3 수명 주기 규칙을 적용합니다. 둘 이상의 필터를 지정하는 경우 예시와 같이 <And> 요소를 포함해야 합니다(Amazon S3가 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, 다음 S3 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 같은 날에 여러 가지 S3 수명 주기 작업을 수행할 수 있습니다. 이러한 경우 Amazon S3은 다음과 같은 일반 규칙을 따릅니다.

- 영구 삭제는 전환에 우선합니다.
- 이전은 삭제 마커 생성에 우선합니다.
- 객체에서 S3 Glacier Flexible Retrieval 및 S3 Standard-IA(또는 S3 One Zone-IA) 전환을 모두 사용할 수 있는 경우 Amazon S3가 S3 Glacier 전환을 선택합니다.

예시는 중복 필터 및 충돌하는 수명 주기 작업의 예제 섹션을 참조하세요.

수명 주기 규칙 임시 비활성화

`status` 요소를 사용하여 S3 수명 주기 규칙을 일시적으로 비활성화할 수 있습니다. 이는 기존 규칙을 덮어쓰지 않고 새 규칙을 테스트하거나 구성 관련 문제를 해결하려는 경우에 유용할 수 있습니다. 다음 S3 수명 주기 구성에는 두 가지 규칙을 지정합니다.

- 규칙 1은 객체 생성 직후 `logs/` 접두사가 있는 객체를 S3 Glacier Flexible Retrieval 스토리지 클래스로 전환하도록 Amazon S3에 지시합니다.
- 규칙 2는 객체 생성 직후 `documents/` 접두사가 있는 객체를 S3 Glacier Flexible Retrieval 스토리지 클래스로 전환하도록 Amazon S3에 지시합니다.

구성에서 규칙 1은 활성화되어 있고 규칙 2는 비활성화되어 있습니다. Amazon S3는 비활성화된 규칙을 무시합니다.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule1</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>GLACIER</StorageClass>
```

```
    </Transition>
  </Rule>
  <Rule>
    <ID>Rule2</ID>
    <Filter>
      <Prefix>documents/</Prefix>
    </Filter>
    <Status>Disabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

객체의 수명 주기 동안 스토리지 클래스 티어 다운

이 예제에서는 S3 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 참조하세요.

다음 S3 수명 주기 구성에서는 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 Flexible Retrieval 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 S3 수명 주기 작업을 설명할 수 있습니다. 또는 각각 다른 필터를 지정하는 여러 규칙을 추가할 수 있습니다.

> **Important**
>
> S3 수명 주기 구성에 규칙이 여러 개인 경우 객체는 같은 날에 여러 가지 S3 수명 주기 작업을 수행할 수 있습니다. 이러한 경우 Amazon S3은 다음과 같은 일반 규칙을 따릅니다.
>
> - 영구 삭제는 전환에 우선합니다.
> - 이전은 삭제 마커 생성에 우선합니다.
> - 객체에서 S3 Glacier Flexible Retrieval 및 S3 Standard-IA(또는 S3 One Zone-IA) 전환을 모두 사용할 수 있는 경우 Amazon S3가 S3 Glacier 전환을 선택합니다.
>
> 예시는 중복 필터 및 충돌하는 수명 주기 작업의 예제 섹션을 참조하세요.

여러 규칙 지정

객체마다 다른 S3 수명 주기 작업을 원할 경우 여러 규칙을 지정할 수 있습니다. 다음 S3 수명 주기 구성에는 두 가지 규칙이 있습니다.

- 규칙 1은 키 이름 접두사가 `classA/`인 객체에 적용되며, 생성 후 1년이 지난 객체를 S3 Glacier Flexible Retrieval 스토리지 클래스로 전환하고 10년이 지난 객체를 만료 처리하도록 Amazon S3에 지시합니다.

- 규칙 2는 키 이름 접두사가 `classB/`인 객체에 적용되며, 생성 후 90일이 지난 객체를 S3 Standard-IA 스토리지 클래스로 전환하고 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>
```

Important

S3 수명 주기 구성에 규칙이 여러 개인 경우 객체는 같은 날에 여러 가지 S3 수명 주기 작업을 수행할 수 있습니다. 이러한 경우 Amazon S3은 다음과 같은 일반 규칙을 따릅니다.

- 영구 삭제는 전환에 우선합니다.
- 이전은 삭제 마커 생성에 우선합니다.

- 객체에서 S3 Glacier Flexible Retrieval 및 S3 Standard-IA(또는 S3 One Zone-IA) 전환을 모두 사용할 수 있는 경우 Amazon S3가 S3 Glacier 전환을 선택합니다.

예시는 중복 필터 및 충돌하는 수명 주기 작업의 예제 섹션을 참조하세요.

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. S3 버전 관리에 대한 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하세요.

다음 예에서는 1년간의 기록을 유지하고 최신이 아닌 버전 5개를 보존하려고 합니다. S3 수명 주기 구성은 모든 객체의 1~100개 버전을 유지할 수 있도록 지원합니다. Amazon S3가 특정 버전을 만료하려면 5개 이상의 최신 버전이 있어야 합니다. Amazon S3는 보존하도록 지정된 수를 초과하는 모든 추가 비최신 버전을 영구적으로 삭제합니다. 삭제를 수행하려면 `NoncurrentDays` 및 `NewerNoncurrentVersions` 값을 모두 초과해야 합니다.

스토리지 비용을 절약하기 위해 최신이 아닌 버전이 된 후 30일 후에 S3 Glacier Flexible Retrieval로 이전 버전을 이동하려고 합니다(이러한 최신이 아닌 객체가 실시간 액세스가 필요 없는 콜드 데이터라고 가정). 뿐만 아니라 생성 후 90일이 지나면 최신 버전에 대한 액세스 빈도가 감소할 것을 예상하여 해당 객체를 S3 Standard-IA 스토리지 클래스로 이동하는 쪽을 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>GLACIER</StorageClass>
        </NoncurrentVersionTransition>
       <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>5</NewerNoncurrentVersions>
```

```
            <NoncurrentDays>365</NoncurrentDays>
        </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

버전 관리를 사용하는 버킷에서 만료된 객체 삭제 마커 제거

버전 관리를 사용하는 버킷에는 최신 버전의 객체 1개와 각 객체에 대한 0개 이상의 비 최신 버전이 존재합니다. 객체를 삭제할 때는 다음 사항에 유의하세요.

- 삭제 요청에서 버전 ID를 지정하지 않는 경우, Amazon S3에서는 그 객체를 삭제하는 대신 삭제 마커를 추가합니다. 최신 객체 버전이 최신이 아닌 버전이 되고 삭제 마커가 최신 버전이 됩니다.
- 삭제 요청에서 버전 ID를 지정하는 경우, Amazon S3에서는 그 객체 버전을 영구적으로 삭제합니다(삭제 마커는 생성되지 않음).
- 최신이 아닌 버전이 없는 삭제 마커는 만료된 객체 삭제 마커라고 부릅니다.

이 예제에서는 버킷에 만료된 객체 삭제 마커를 생성할 수 있는 시나리오와 S3 수명 주기 구성을 사용해 만료된 객체 삭제 마커를 제거하도록 Amazon S3에 지시하는 방법을 보여줍니다.

다음 예와 같이 `NoncurrentVersionExpiration` 작업을 사용하여 비최신 버전이 된 후 30일이 지나면 비최신 버전을 제거하고 최대 10개의 비최신 버전을 보존하는 S3 수명 주기 구성을 작성한다고 가정합니다. Amazon S3가 특정 버전을 만료하려면 10개 이상의 최신 버전이 있어야 합니다. Amazon S3는 보존하도록 지정된 수를 초과하는 모든 추가 비최신 버전을 영구적으로 삭제합니다. 삭제를 수행하려면 `NoncurrentDays` 및 `NewerNoncurrentVersions` 값을 모두 초과해야 합니다.

```
<LifecycleConfiguration>
    <Rule>
        ...
        <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
            <NoncurrentDays>30</NoncurrentDays>
        </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

`NoncurrentVersionExpiration` 작업은 현재 객체 버전에는 적용되지 않습니다. 최신이 아닌 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.

- 최신 객체 버전은 잘 정의된 수명 주기를 따릅니다.

  이 경우 다음 예시와 같이 Amazon S3에 최신 버전을 제거하도록 지시하는 `Expiration` 작업과 함께 S3 수명 주기 구성을 사용할 수 있습니다.

```
<LifecycleConfiguration>
    <Rule>
        ...
        <Expiration>
           <Days>60</Days>
        </Expiration>
        <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
            <NoncurrentDays>30</NoncurrentDays>
        </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

  이 예에서 Amazon S3가 최신 버전의 객체 각각에 삭제 마커를 추가함으로써 생성된 후 60일이 지난 최신 버전을 제거합니다. 이 프로세스를 통해 최신 버전이 최신이 아닌 버전이 되고 삭제 마커가 최신 버전이 됩니다. 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 단원을 참조하십시오.

> **Note**
>
> 동일한 규칙에 `Days` 및 `ExpiredObjectDeleteMarker` 태그를 모두 지정할 수는 없습니다. `Days` 태그를 지정하면 삭제 마커의 기간이 사용 기간 기준을 충족할 때 Amazon S3가 자동으로 `ExpiredObjectDeleteMarker` 정리를 수행합니다. 삭제 마커가 유일한 버전이 되는 즉시 정리하려면 `ExpiredObjectDeleteMarker` 태그만 있는 별도의 규칙을 생성합니다.

  동일한 S3 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. 이 예시에서는 Amazon S3가 특정 버전을 만료하려면 10개 이상의 최신 버전이 있어야 합니다. Amazon S3는 보존하도록 지정된 수를 초과하는 모든 추가 비최신 버전을 영구적으

로 삭제합니다. 삭제를 수행하려면 NoncurrentDays 및 NewerNoncurrentVersions 값을 모두 초과해야 합니다.

만료된 객체 삭제 마커는 이 과정에서 생성되지만 Amazon S3에서 만료된 객체 삭제 마커를 자동 감지하여 제거합니다.

- 객체의 최신 버전은 잘 정의된 수명 주기를 따르지 않습니다.

  이 경우 객체가 필요 없는 경우 1개 이상의 최신이 아닌 버전에 삭제 마커를 생성하여 수동으로 객체를 제거할 수 있습니다. S3 수명 주기 구성에 NoncurrentVersionExpiration 작업을 사용하여 최신이 아닌 버전을 모두 제거하면 만료된 객체 삭제 마커가 생깁니다.

  특히 이 시나리오의 경우 S3 수명 주기 구성은 만료된 객체 삭제 마커를 제거하는 데 사용할 수 있는 Expiration 작업을 제공합니다.

```
<LifecycleConfiguration>
    <Rule>
       <ID>Rule 1</ID>
        <Filter>
          <Prefix>logs/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Expiration>
           <ExpiredObjectDeleteMarker>true</ExpiredObjectDeleteMarker>
        </Expiration>
        <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
            <NoncurrentDays>30</NoncurrentDays>
        </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

Expiration 작업에서 ExpiredObjectDeleteMarker 요소를 true로 설정하여 Amazon S3에 만료된 객체 삭제 마커를 제거하도록 지시합니다.

> **Note**
>
> ExpiredObjectDeleteMarker S3 수명 주기 작업을 사용하면 이 규칙은 태그 기반 필터를 지정할 수 없습니다.

멀티파트 업로드 중단을 위한 수명 주기 구성

Amazon S3 멀티파트 업로드 REST API 작업을 사용하여 대형 객체를 여러 부분으로 나누어 업로드할 수 있습니다. 멀티파트 업로드에 대한 자세한 내용은 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 섹션을 참조하십시오.

멀티파트 업로드가 시작된 후 지정된 일수 내에 완료되지 않은 경우 S3 수명 주기 구성을 사용하여 미완료 멀티파트 업로드(규칙에 지정된 키 이름 접두사로 식별)를 중단하도록 Amazon S3에 지시할 수 있습니다. Amazon S3가 멀티파트 업로드를 중단할 때 멀티파트 업로드와 관련된 모든 부분을 삭제합니다. 이 프로세스는 Amazon S3에 저장된 부분으로 불완전한 멀티파트 업로드가 없도록 하여 스토리지 비용을 조정하는 데 도움이 됩니다.

> **Note**
>
> `AbortIncompleteMultipartUpload` S3 수명 주기 작업을 사용하면 이 규칙은 태그 기반 필터를 지정할 수 없습니다.

다음은 `AbortIncompleteMultipartUpload` 작업으로 규칙을 지정하는 S3 수명 주기 구성의 예시입니다. 이 작업은 시작 후 7일이 지나면 미완료된 멀티파트 업로드를 중단하도록 Amazon S3에 지시합니다.

```
<LifecycleConfiguration>
    <Rule>
        <ID>sample-rule</ID>
        <Filter>
           <Prefix>SomeKeyPrefix/</Prefix>
        </Filter>
        <Status>rule-status</Status>
        <AbortIncompleteMultipartUpload>
          <DaysAfterInitiation>7</DaysAfterInitiation>
        </AbortIncompleteMultipartUpload>
    </Rule>
</LifecycleConfiguration>
```

데이터가 없는 비최신 객체 만료

크기만 기준으로 객체를 전환하는 규칙을 생성할 수 있습니다. 최소 크기(`ObjectSizeGreaterThan`) 또는 최대 크기(`ObjectSizeLessThan`)를 지정하거나 객체 크기 범

위를 바이트 단위로 지정할 수 있습니다. 접두사 및 크기 규칙과 같은 필터를 여러 개 사용하는 경우 <And> 요소로 필터를 묶어야 합니다.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Transition with a prefix and based on size</ID>
    <Filter>
       <And>
          <Prefix>tax/</Prefix>
          <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
       </And>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

ObjectSizeGreaterThan 요소와 ObjectSizeLessThan 요소를 둘 다 사용하여 범위를 지정하는 경우 최대 객체 크기가 최소 객체 크기보다 커야 합니다. 둘 이상의 필터를 사용하는 경우 <And> 요소로 필터를 묶어야 합니다. 다음 예에서는 500~64,000바이트 범위의 객체를 지정하는 방법을 보여 줍니다. 범위를 지정하는 경우 ObjectSizeGreaterThan 및 ObjectSizeLessThan 필터는 지정된 값을 제외합니다. 자세한 내용은 the section called “필터 요소” 단원을 참조하십시오.

```
<LifecycleConfiguration>
    <Rule>
        ...
          <And>
             <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
             <ObjectSizeLessThan>64000</ObjectSizeLessThan>
          </And>
    </Rule>
</LifecycleConfiguration>
```

버전 관리를 사용하는 버킷에서 만든 비최신 삭제 마커 객체를 포함하여 데이터가 없는 비최신 객체를 특별히 만료시키는 규칙을 만들 수도 있습니다. 다음 예시에서는 NoncurrentVersionExpiration 작업을 사용하여 비최신 버전이 된 후 30일이 지나면 제거하고 비최신 버전을 최대 10개까지 보존합니다. 또한 이 예시에서는 ObjectSizeLessThan 요소를 사용하여 데이터가 없는 객체만 필터링합니다.

Amazon S3가 특정 버전을 만료하려면 10개 이상의 최신 버전이 있어야 합니다. Amazon S3는 보존하도록 지정된 수를 초과하는 모든 추가 비최신 버전을 영구적으로 삭제합니다. 삭제를 수행하려면 NoncurrentDays 및 NewerNoncurrentVersions 값을 모두 초과해야 합니다.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Expire noncurrent with size less than 1 byte</ID>
    <Filter>
       <ObjectSizeLessThan>1</ObjectSizeLessThan>
    </Filter>
    <Status>Enabled</Status>
    <NoncurrentVersionExpiration>
       <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
       <NoncurrentDays>30</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
</LifecycleConfiguration>
```

예제: 128KB 미만의 객체 전환 허용

Amazon S3는 128KB 미만의 객체가 스토리지 클래스로 전환되지 않도록 하는 기본 동작을 수명 주기 구성에 적용합니다. 구성에 더 작은 크기를 지정하는 최소 크기(ObjectSizeGreaterThan) 또는 최대 크기(ObjectSizeLessThan) 필터를 추가하여 더 작은 객체가 전환되도록 할 수 있습니다. 다음 예제에서는 128KB보다 작은 객체의 S3 Glacier Instant Retrieval 스토리지 클래스 전환을 허용합니다.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Allow small object transitions</ID>
    <Filter>
          <ObjectSizeGreaterThan>1</ObjectSizeGreaterThan>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER_IR</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

> **Note**
>
> 2024년 9월 Amazon S3는 소형 객체의 기본 전환 동작을 다음과 같이 업데이트했습니다.
>
> - 업데이트된 기본 전환 동작 - 2024년 9월부터 기본 동작은 128KB 미만의 객체가 스토리지 클래스로 전환되는 것을 금지합니다.
> - 이전의 기본 전환 동작 - 2024년 9월 이전의 기본 동작은 128KB 미만의 객체를 S3 Glacier 및 S3 Glacier Deep Archive 스토리지 클래스로는 전환할 수 있었습니다.
>
> 2024년 9월 이전에 생성된 구성은 수정하지 않는 한 이전 전환 동작을 유지합니다. 즉, 규칙을 생성, 편집 또는 삭제하면 구성의 기본 전환 동작이 업데이트된 동작으로 변경됩니다. 사용 사례에 필요한 경우 128KB 미만의 객체가 S3 Glacier 및 S3 Glacier Deep Archive로 전환되도록 기본 전환 동작을 변경할 수 있습니다. 이렇게 하려면 PutBucketLifecycleConfiguration 요청에서 선택적 `x-amz-transition-object-size-minimum-default` 헤더를 사용합니다.

다음 예시는 PutBucketLifecycleConfiguration 요청에서 `x-amz-transition-object-size-minimum-default` 헤더를 사용하여 S3 수명 주기 구성에 `varies_by_storage_class` 기본 전환 동작을 적용하는 방법을 보여줍니다. 이 동작을 통해 128KB보다 작은 객체를 S3 Glacier 또는 S3 Glacier Deep Archive 스토리지 클래스로 전환할 수 있습니다. 기본적으로 다른 모든 스토리지 클래스는 128KB 미만의 전환을 허용하지 않습니다. 그래도 사용자 지정 필터를 사용하여 모든 스토리지 클래스의 최소 전환 크기를 변경할 수 있습니다. 사용자 지정 필터는 항상 기본 전환 동작보다 우선합니다.

```
HTTP/1.1 200
x-amz-transition-object-size-minimum-default: varies_by_storage_class
<?xml version="1.0" encoding="UTF-8"?>
...
```

Amazon S3 수명 주기 문제 해결

다음 정보는 Amazon S3 수명 주기 규칙과 관련된 문제를 해결하는 데 도움이 될 수 있습니다.

주제

- 버킷에서 목록 작업을 실행한 결과, 수명 주기 규칙에 따라 만료되었거나 전환된 것으로 생각되는 객체가 있었습니다.
- 수명 주기 규칙에 따라 취해진 조치를 모니터링하려면 어떻게 해야 하나요?
- 버전 관리가 활성화된 버킷에 수명 주기 규칙을 설정한 후에도 S3 객체 수가 계속 증가합니다.

- 수명 주기 규칙을 사용하여 S3 버킷을 비우려면 어떻게 해야 하나요?
- 객체를 더 저렴한 스토리지 클래스로 전환한 후 Amazon S3 청구액이 늘었습니다.
- 버킷 정책을 업데이트했지만 만료된 수명 주기 규칙으로 인해 S3 객체가 여전히 삭제되고 있습니다.
- S3 수명 주기 규칙으로 인해 만료된 S3 객체를 복구할 수 있나요?
- 만료 및 전환 수명 주기 작업이 발생하지 않는 이유는 무엇입니까?
- 수명 주기 규칙에서 접두사를 제외하려면 어떻게 해야 하나요?
- 수명 주기 규칙에 여러 접두사를 포함하려면 어떻게 해야 하나요?

버킷에서 목록 작업을 실행한 결과, 수명 주기 규칙에 따라 만료되었거나 전환된 것으로 생각되는 객체가 있었습니다.

S3 수명 주기 객체 전환 및 객체 만료는 비동기 작업입니다. 따라서 객체가 만료 또는 전환될 수 있는 시점과 실제로 전환되거나 만료되는 시점 사이에 지연이 있을 수 있습니다. 결제 변경 사항은 작업이 완료되지 않았더라도 수명 주기 규칙이 충족되는 즉시 적용됩니다. 이 동작에 대한 한 가지 예외는 S3 Intelligent-Tiering 스토리지 클래스로 전환하도록 수명 주기 규칙을 설정한 경우입니다. 이 경우 객체가 S3 Intelligent-Tiering으로 전환될 때까지 결제 변경이 발생하지 않습니다. 결제 변경에 대한 자세한 내용은 버킷의 수명 주기 구성 설정을 참조하세요.

> **Note**
>
> Amazon S3은 128KB보다 작은 객체를 S3 Standard 또는 S3 Standard-IA 스토리지 클래스에서 S3 Intelligent-Tiering, S3 Standard-IA 또는 S3 One Zone-IA 스토리지 클래스로 전환하지 않습니다.

수명 주기 규칙에 따라 취해진 조치를 모니터링하려면 어떻게 해야 하나요?

다음 기능을 사용하여 수명 주기 규칙이 수행한 작업을 모니터링할 수 있습니다.

- S3 이벤트 알림 - S3 수명 주기 만료 또는 이전 이벤트에 대한 알림을 받도록 S3 이벤트 알림을 설정할 수 있습니다.
- S3 서버 액세스 로그 - S3 버킷에 대한 서버 액세스 로그를 활성화하여 다른 스토리지 클래스로의 객체 이전 및 객체 만료와 같은 S3 수명 주기 관련 작업을 캡처합니다. 자세한 내용은 수명 주기 및 로깅 섹션을 참조하세요.

수명 주기 작업으로 인한 스토리지 변경을 매일 확인하려면 Amazon CloudWatch 지표를 사용하는 대신 S3 Storage Lens 대시보드를 사용하는 것이 좋습니다. Storage Lens 대시보드에서 객체 개수 또는 크기를 모니터링하는 다음 지표를 볼 수 있습니다.

- 최신 버전 바이트
- 최신 버전 객체 수
- 비최신 버전 바이트
- 비최신 버전 객체 수
- 삭제 마커 객체 수
- 삭제 마커 스토리지 바이트
- 불완전 멀티파트 업로드 바이트
- 불완전 멀티파트 업로드 객체 수

버전 관리가 활성화된 버킷에 수명 주기 규칙을 설정한 후에도 S3 객체 수가 계속 증가합니다.

버전 관리를 사용하는 버킷에서 객체가 만료되도록 설정된 경우 객체가 버킷에서 완전히 삭제되지는 않습니다. 대신 삭제 마커가 객체의 최신 버전으로 만들어집니다. 삭제 마커는 여전히 객체로 집계됩니다. 따라서 현재 버전만 만료하도록 수명 주기 규칙을 생성하면 S3 버킷의 객체 수가 줄어들지 않고 실제로 증가합니다.

예를 들어 버전 관리가 활성화된 S3 버킷에 100개의 객체가 있고 7일 후에 객체의 현재 버전이 만료되도록 수명 주기 규칙이 설정되어 있다고 가정해 보겠습니다. 7일이 지나면 현재 비최신 버전인 원래 객체 100개 외에 100개의 삭제 마커가 생성되므로 객체 수가 200개로 늘어납니다. 버전 관리가 활성화된 버킷의 S3 수명 주기 구성 규칙 작업에 대한 자세한 내용은 버킷에서 수명 주기 구성 설정을 참조하세요.

객체를 영구적으로 제거하려면 객체의 이전 버전, 만료된 삭제 마커 및 불완전 멀티파트 업로드를 삭제하는 수명 주기 구성을 추가하세요. 새 수명 주기 규칙을 만드는 방법에 대한 지침은 버킷에 수명 주기 구성 설정을 참조하세요.

> **Note**
>
> - Amazon S3는 객체의 전환 또는 만료 날짜를 다음 날 자정(UTC)으로 반올림합니다.
>
> 수명 주기 작업에 사용할 객체를 평가할 때 Amazon S3는 객체 생성 시간(UTC)을 사용합니다. 예를 들어 하루가 지나면 객체가 만료되도록 구성된 수명 주기 규칙이 있는 버전이 지

정되지 않은 버킷을 생각해 보세요. 1월 1일 17:05 태평양 표준시(PDT)에 객체가 생성되었다고 가정해 보겠습니다. 이는 1월 2일 00:05 UTC에 해당합니다. 이 객체는 1월 3일 00:05 UTC에 하루가 지나고 나면 S3 라이프사이클에서 1월 4일 00:00 UTC에 객체를 평가할 때 만료될 수 있습니다.

Amazon S3 수명 주기 작업은 비동기적으로 발생하기 때문에 수명 주기 규칙에 지정된 날짜와 객체의 실제 물리적 이전 사이에 약간의 지연이 있을 수 있습니다. 자세한 내용은 이전 또는 만료 지연을 참조하세요.

자세한 내용은 수명 주기 규칙: 객체 기간 기반을 참조하세요.

- 객체 잠금으로 보호되는 S3 객체의 경우 현재 버전은 영구적으로 삭제되지 않습니다. 대신 객체에 삭제 마커가 추가되어 객체가 비최신 상태가 됩니다. 그런 다음, 비최신 버전이 보존되며 영구적으로 만료되지 않습니다.

수명 주기 규칙을 사용하여 S3 버킷을 비우려면 어떻게 해야 하나요?

S3 수명 주기 규칙은 수백만 개의 객체가 포함된 S3 버킷을 비우는 효과적인 도구입니다. S3 버킷에서 많은 수의 객체를 삭제하려면 다음 두 쌍의 수명 주기 규칙을 사용해야 합니다.

- 객체의 현재 버전 만료 및 이전 버전의 객체 영구 삭제
- 만료된 삭제 마커 삭제 및 불완전 멀티파트 업로드 삭제

새 수명 주기 규칙을 만드는 방법에 대한 지침은 버킷에서 수명 주기 구성 설정을 참조하세요.

> **Note**
>
> 객체 잠금으로 보호되는 S3 객체의 경우 현재 버전은 영구적으로 삭제되지 않습니다. 대신 객체에 삭제 마커가 추가되어 객체가 비최신 상태가 됩니다. 그런 다음, 비최신 버전이 보존되며 영구적으로 만료되지 않습니다.

객체를 더 저렴한 스토리지 클래스로 전환한 후 Amazon S3 청구액이 늘었습니다.

객체를 더 저렴한 스토리지 클래스로 전환한 후 청구액이 늘어나는 데는 몇 가지 이유가 있습니다.

- 작은 객체에 대한 S3 Glacier 오버헤드 요금

S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive에 각 객체를 전환할 때 이 스토리지 업데이트에는 총 40KB의 총 오버헤드가 있습니다. 40KB 오버헤드의 일부로 8KB는 메타데이터와 객체 이름을 저장하는 데 사용됩니다. 이 8KB는 S3 Standard 요금에 따라 요금이 청구됩니다. 나머지 32KB는 인덱싱 및 관련 메타데이터에 사용됩니다. 이 32KB는 S3 Glacier Flexible Retrieval 또는 S3 Glacier Deep Archive 요금에 따라 요금이 청구됩니다.

따라서 크기가 작은 객체를 여러 개 저장하는 경우 수명 주기 전환을 사용하지 않는 것이 좋습니다. 오버헤드 요금을 줄이려면 비교적 작은 객체 여러 개를 Amazon S3에 저장하기 전에 개수가 더 적은 큰 객체로 취합하는 것이 좋습니다. 비용 고려 사항과 관련한 자세한 내용은 S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive 스토리지 클래스로 전환(객체 아카이브)을 참조하세요.

- 최소 스토리지 요금

  일부 S3 스토리지 클래스에는 최소 스토리지 기간 요구 사항이 있습니다. 최소 기간이 충족되기 전에 해당 클래스에서 삭제, 덮어쓰기 또는 전환된 객체에는 비례 계산된 조기 전환 또는 삭제 요금이 부과됩니다. 이러한 최소 스토리지 기간 요구 사항은 다음과 같습니다.

  - S3 Standard-IA 및 S3 One Zone-IA - 30일
  - S3 Glacier Flexible Retrieval 및 S3 Glacier Instant Retrieval – 90일
  - S3 Glacier Deep Archive - 180일

  이러한 요구 사항에 대한 자세한 내용은 S3 수명 주기를 사용하여 객체 전환의 제약 조건 섹션을 참조하세요. 일반 S3 요금 정보는 Amazon S3 요금 및 AWS 요금 계산기를 참조하세요.

- 수명 주기 전환 비용

  수명 주기 규칙에 따라 객체가 다른 스토리지 클래스로 전환될 때마다 Amazon S3는 해당 전환을 하나의 전환 요청으로 간주합니다. 이러한 전환 요청에 대한 비용은 이러한 스토리지 클래스의 비용에 추가됩니다. 많은 수의 객체를 전환하려는 경우 낮은 계층으로 전환할 때 요청 요금을 고려해야 합니다. 자세한 내용은 Amazon S3 요금을 참조하십시오.

버킷 정책을 업데이트했지만 만료된 수명 주기 규칙으로 인해 S3 객체가 여전히 삭제되고 있습니다.

버킷 정책의 `Deny` 명령문은 수명 주기 규칙에 정의된 객체의 만료를 방지하지 않습니다. 수명 주기 작업(예: 전환 또는 만료)은 S3 `DeleteObject` 작업을 사용하지 않습니다. 대신 S3 수명 주기 작업은 내부 S3 엔드포인트를 사용하여 수행됩니다. (자세한 내용은 수명 주기 및 로깅을 참조하세요.)

수명 주기 규칙이 작업을 수행하지 못하도록 하려면 규칙을 편집, 삭제 또는 비활성화해야 합니다.

S3 수명 주기 규칙으로 인해 만료된 S3 객체를 복구할 수 있나요?

S3 수명 주기에 의해 만료된 객체를 복구하는 유일한 방법은 버전 관리를 사용하는 것인데, 객체가 만료 대상이 되기 전에 버전 관리를 적용해야 합니다. 수명 주기 규칙으로 인해 수행된 만료 작업은 취소할 수 없습니다. 설정된 S3 수명 주기 규칙에 따라 객체가 영구적으로 삭제된 경우 해당 객체를 복구할 수 없습니다. 버킷의 버전 관리를 활성화하려면 the section called “여러 버전의 객체 유지” 섹션을 참조하세요.

버킷에 버전 관리를 적용했는데 객체의 비최신 버전이 여전히 그대로 있는 경우 만료된 객체의 이전 버전을 복원할 수 있습니다. S3 수명 주기 규칙 작업의 동작 및 버전 관리 상태에 대한 자세한 내용은 수명 주기 작업을 설명할 요소의 수명 주기 작업 및 버킷의 버전 관리 상태 표를 참조하세요.

> **Note**
>
> S3 버킷이 AWS 백업 또는 S3 복제로 보호되는 경우 이러한 기능을 사용하여 만료된 객체를 복구할 수도 있습니다.

만료 및 전환 수명 주기 작업이 발생하지 않는 이유는 무엇입니까?

버전 관리를 사용하거나 버전 관리가 일시 중지된 버킷의 경우, Amazon S3가 만료 작업을 처리하는 방식에 대한 몇 가지 고려 사항이 있습니다.

- 객체 만료가 객체의 현재 버전에만 적용됩니다(비최신 객체 버전에 영향이 없음).
- 객체 버전이 1개 이상이고 삭제 마커가 최신 버전이면 Amazon S3은 작업을 수행하지 않습니다.
- Amazon S3는 S3 Object Lock이 적용된 비최신 버전의 객체에 대해서는 어떠한 조치도 취하지 않습니다.
- 복제 상태가 `PENDING`인 객체의 경우 Amazon S3는 최신 또는 비최신 버전의 객체에 대해 어떠한 조치도 취하지 않습니다.

수명 주기 스토리지 클래스 전환에는 다음과 같은 제약이 있습니다.

- 기본적으로 128KB 미만의 객체는 어떤 스토리지 클래스로도 전환되지 않습니다.
- 객체를 S3 Standard-IA 또는 S3 One Zone-IA로 전환하기 전에 최소 30일 동안 저장해야 합니다.
- 버전 관리를 사용하거나 버전 관리가 일시 중지된 버킷의 경우, 복제 상태가 `PENDING`인 객체를 전환할 수 없습니다.

수명 주기 규칙에서 접두사를 제외하려면 어떻게 해야 하나요?

S3 수명 주기는 규칙에서 접두사를 제외하는 것을 지원하지 않습니다. 대신 태그를 사용하여 규칙에 포함하려는 모든 객체에 태그를 지정합니다. 수명 주기 규칙에서 태그 사용에 대한 자세한 내용은 the section called “생성 후 1일 이내에 모든 객체 아카이브” 섹션을 참조하세요.

수명 주기 규칙에 여러 접두사를 포함하려면 어떻게 해야 하나요?

S3 수명 주기는 규칙에 여러 접두사를 포함하는 것을 지원하지 않습니다. 대신 태그를 사용하여 규칙에 포함하려는 모든 객체에 태그를 지정합니다. 수명 주기 규칙에서 태그 사용에 대한 자세한 내용은 the section called “생성 후 1일 이내에 모든 객체 아카이브” 섹션을 참조하세요.

하지만 동일한 문자로 시작하는 접두사가 하나 이상 있는 경우 필터에 후행 슬래시(/)가 없는 부분 접두사를 지정하여 규칙에 해당 접두사를 모두 포함할 수 있습니다. 예를 들어, 다음과 같은 접두사가 있다고 가정해 보세요.

```
sales1999/
sales2000/
sales2001/
```

규칙에 세 접두사를 모두 포함하려면 수명 주기 규칙에 `<Prefix>sales</Prefix>`를 지정합니다.

Amazon S3의 로깅 및 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 수집하는 것이 좋습니다. Amazon S3에 대한 모니터링을 시작하기 전에 다음 질문에 대한 답변을 비롯한 모니터링 계획을 세웁니다.

- 모니터링의 목표
- 모니터링할 리소스
- 이러한 리소스를 모니터링하는 빈도
- 사용할 모니터링 도구
- 모니터링 작업을 수행할 사람
- 문제 발생 시 알려야 할 대상

Amazon S3의 로깅 및 모니터링에 대한 자세한 내용은 다음 주제를 참조하십시오.

> **Note**
>
> Amazon S3 Express One Zone 스토리지 클래스를 디렉터리 버킷과 함께 사용하는 방법에 대한 자세한 내용은 S3 Express One Zone 및 디렉터리 버킷 작업 섹션을 참조하세요.

모니터링은 Amazon S3와 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다. 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 정책으로 알림이 전송됩니다. CloudWatch 경보는 단순히 특정 상태에 있다고 해서 작업을 호출하지 않습니다. 대신, 상태가 변경되어 지정된 기간 동안 유지되어야 합니다. 자세한 내용은 Amazon CloudWatch를 사용한 지표 모니터링 섹션을 참조하세요.

AWS CloudTrail 로그

CloudTrail은 Amazon S3에서 사용자, 역할 또는 AWS 서비스가 수행한 작업의 기록을 제공합니다. CloudTrail에서 수집한 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 섹션을 참조하세요.

Amazon GuardDuty

Amazon GuardDuty는 계정, 컨테이너, 워크로드 및 AWS 환경 내 데이터를 지속적으로 모니터링하여 S3 버킷에 대한 잠재적 위협 또는 보안 위험을 식별하는 위협 탐지 서비스입니다. 또한 GuardDuty는 탐지한 위협에 대한 풍부한 컨텍스트를 제공합니다. GuardDuty는 위협에 대한 AWS CloudTrail 관리 로그를 모니터링하고 보안 관련 정보를 표시합니다. 예를 들어 GuardDuty는 API 요청 중 요청한 사용자, 요청이 이루어진 위치, 요청된 특정 API 등 사용자 환경에서 일반적이지 않을 수 있는 요소를 포함됩니다. GuardDuty S3 Protection은 CloudTrail에서 수집한 S3 데이터 이벤트를 모니터링하고 사용자 환경의 모든 S3 버킷에서 잠재적으로 비정상적이고 악의적인 동작을 식별합니다.

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 서버 액세스 로깅을 사용한 요청 로깅 섹션을 참조하세요.

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
- 공개 액세스 권한을 가진 Amazon S3 버킷에 대한 보안 검사
- 버전 관리가 사용 설정되지 않았거나 버전 관리가 중지된 Amazon S3 버킷에 대한 내결함성 검사

자세한 내용은 지원 사용 설명서의 AWS Trusted Advisor 섹션을 참조하십시오.

Amazon S3 Storage Lens

Amazon S3 스토리지 렌즈는 객체 스토리지 사용 및 활동에 대한 조직 전반의 가시성을 확보하는 데 사용할 수 있는 클라우드 스토리지 분석 기능입니다. S3 스토리지 렌즈 지표를 사용하여 조직 전반의 스토리지 용량 또는 가장 빠르게 증가하는 버킷과 접두사 등의 요약 인사이트를 생성할 수 있습니다. 또한 S3 스토리지 렌즈 지표를 사용하여 비용 최적화 기회를 식별하고, 데이터 보호 및 보안 모범 사례를 구현하고, 애플리케이션 워크로드의 성능을 개선할 수 있습니다.

S3 Storage Lens는 지표를 집계하고 Amazon S3 콘솔 버킷 페이지의 계정 스냅샷 섹션에 이 정보를 표시합니다. S3 스토리지 렌즈는 또한 인사이트와 추세를 시각화하고, 이상치에 플래그를 지정하고, 스토리지 비용 최적화와 데이터 보호 모범 사례 적용을 위한 권장 사항을 수신하는 데 사용할 수 있는 대화형 대시보드를 제공합니다. 대시보드에 있는 드릴다운 옵션을 통해 조직, 계정, AWS 리전, 스토리지 클래스, 버킷, 접두사 또는 스토리지 렌즈 그룹 수준에서 인사이트를 생성하거나 시각화할 수 있습니다. 자세한 내용은 Amazon S3 스토리지 렌즈 이해 섹션을 참조하세요.

Amazon S3 Inventory

Amazon S3 Inventory는 객체를 쿼리하고 관리하는 데 사용할 수 있는 객체 및 메타데이터 목록을 생성합니다. 이 인벤토리 보고서를 사용하여 객체 크기, 마지막으로 수정된 날짜, 암호화 상태 및 기타 필드와 같은 세분화된 데이터를 생성할 수 있습니다. 이러한 보고서는 최신 목록을 자동으로 제공하기 위해 매일 또는 매주 사용할 수 있습니다.

예를 들어, Amazon S3 Inventory를 사용하면 비즈니스, 규정 준수 및 규제 요건에 관하여 객체의 복제 및 암호화 상태를 감사하고 보고할 수 있습니다. 또한 비즈니스 워크플로와 빅 데이터 작업을 간소화하고 속도를 높이는 데도 Amazon S3 Inventory를 사용할 수 있습니다. 이는 Amazon S3 동기식 `List` API 작업에 대한 예약된 대안을 제공합니다. Amazon S3 Inventory는 객체를 감사하기 위해 `List` API 작업을 사용하지 않으며 버킷의 요청 속도에 영향을 주지 않습니다. 자세한 내용은 S3 Inventory를 사용한 데이터 카탈로그화 및 분석 섹션을 참조하세요.

Amazon S3 이벤트 알림

Amazon S3 이벤트 알림 기능을 사용하면 S3 버킷에서 특정 이벤트가 발생하면 알림을 받을 수 있습니다. 알림을 사용 설정하려면 Amazon S3에서 게시하려는 이벤트를 식별하는 알림 구성을 추가합니다. 자세한 내용은 Amazon S3 이벤트 알림 섹션을 참조하세요.

Amazon S3 및 AWS X-Ray

AWS X-Ray는 Amazon S3와 통합되어 업스트림 요청을 추적하여 애플리케이션의 S3 버킷을 업데이트합니다. 서비스가 X-Ray SDK를 사용하여 요청을 추적하는 경우, Amazon S3는 추적 헤더를 Λ, Amazon SQS, Amazon SNS와 같은 다운스트림 이벤트 구독자에게 보낼 수 있습니다. X-Ray는 Amazon S3 이벤트 알림에 대한 추적 메시지를 활성화합니다. X-Ray 트레이스 맵을 사용하여

Amazon S3와 애플리케이션에서 사용하는 다른 서비스의 연결을 확인할 수 있습니다. 자세한 내용은 Amazon S3 and X-Ray를 참조하세요.

다음 보안 모범 사례에서도 로깅 및 모니터링을 다룹니다.

- Identify and audit all your Amazon S3 buckets
- Implement monitoring using Amazon Web Services monitoring tools
- AWS Config 사용 설정
- Enable Amazon S3 server access logging
- Use CloudTrail
- Monitor Amazon Web Services security advisories

주제

- 모니터링 도구
- Amazon S3에 대한 로깅 옵션
- AWS CloudTrail을 사용하여 Amazon S3 API 호출 로깅
- 서버 액세스 로깅을 사용한 요청 로깅
- Amazon CloudWatch를 사용한 지표 모니터링
- Amazon S3 이벤트 알림
- Amazon S3 스토리지 렌즈를 사용하여 스토리지 활동 및 사용량 평가
- S3 Inventory를 사용한 데이터 카탈로그화 및 분석

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 작업은 최대한 자동화하는 것이 좋습니다.

자동 모니터링 도구

다음과 같은 자동 모니터링 도구를 사용하여 Amazon S3를 관찰하고 문제 발생 시 보고할 수 있습니다.

- Amazon CloudWatch 경보 – 지정한 기간 동안 단일 지표를 감시하고, 여러 기간에 대해 지정된 임계값과 관련하여 지표 값을 기준으로 하나 이상의 작업을 수행합니다. 이 작업은 Amazon Simple Notification Service(Amazon SNS) 주제 또는 Amazon EC2 Auto Scaling 정책에 전송되는 알림입니다. CloudWatch 경보는 단순히 특정 상태에 있다고 해서 작업을 호출하지 않습니다. 상태가 변경되어 지정된 기간 수 동안 유지되어야 합니다. 자세한 내용은 Amazon CloudWatch를 사용한 지표 모니터링 섹션을 참조하세요.
- AWS CloudTrail Log Monitoring – 계정 간에 로그 파일을 공유하고, CloudTrail 로그 파일을 CloudWatch Logs에 전송하여 실시간으로 모니터링하며, Java에서 로그 처리 애플리케이션을 작성하고, CloudTrail에서 전송한 후 로그 파일이 변경되지 않았는지 확인합니다. 자세한 내용은 AWS CloudTrail을 사용하여 Amazon S3 API 호출 로깅 섹션을 참조하세요.

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 Console 대시보드에서는 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 서버 액세스 로깅을 사용한 요청 로깅 섹션을 참조하세요.

- Amazon S3 대시보드에 다음이 표시됩니다.
  - 버킷 및 객체와 해당 속성
- CloudWatch 홈 페이지에 다음이 표시됩니다.
  - 현재 경보 및 상태
  - 경보 및 리소스 그래프
  - 서비스 상태

  또한 CloudWatch를 사용하여 다음을 수행할 수 있습니다.
  - 사용자 지정 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 Advisor 사용을 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 내용은 AWS Trusted Advisor 섹션을 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 관리가 사용 설정되지 않았거나 버전 관리가 중지된 Amazon S3 버킷에 대한 내결함성 검사

Amazon S3에 대한 로깅 옵션

Amazon S3 리소스에 대해 사용자, 역할 또는 AWS 서비스가 수행하는 작업을 기록하고 감사 및 규정 준수 목적으로 로그 레코드를 유지 관리할 수 있습니다. 서버 액세스 로깅, AWS CloudTrail 로깅 또는 둘 다를 사용할 수 있습니다. Amazon S3 리소스의 버킷 수준 및 객체 수준 작업 로깅에 CloudTrail을 사용하는 것이 좋습니다. 각 옵션에 대한 자세한 내용은 다음 섹션을 참조하십시오.

- 서버 액세스 로깅을 사용한 요청 로깅
- AWS CloudTrail을 사용하여 Amazon S3 API 호출 로깅

다음 표에 CloudTrail 로그와 Amazon S3 서버 액세스 로그의 주요 속성이 나와 있습니다. CloudTrail이 보안 요구 사항을 충족하는지 확인하려면 표 및 참고 사항을 검토하세요.

로그 속성	AWS CloudTrail	Amazon S3 서버 로그
다른 시스템(Amazon CloudWatch Logs, Amazon CloudWatch Events)으로 전달 가능	예	No
둘 이상의 대상으로 로그 전송(예: 서로 다른 두 버킷으로 동일한 로그 전송)	예	No
객체의 하위 세트(접두사)에 대해 로그 켜기	예	No

로그 속성	AWS CloudTrail	Amazon S3 서버 로그
교차 계정 로그 전송(서로 다른 계정이 소유한 대상 및 소스 버킷)	예	No
디지털 서명 또는 해싱을 사용하여 로그 파일의 무결성 검증	예	No
로그 파일을 기본으로 또는 선택적으로 암호화	예	No
객체 작업(Amazon S3 API 사용)	예	예
버킷 작업(Amazon S3 API 사용)	예	예
로그에 대해 검색 가능한 UI	예	No
객체 잠금 파라미터의 필드, 로그 레코드에 대한 Amazon S3 Select 속성	예	No
로그 레코드의 `Object Size`, `Total Time`, `Turn-Around Time` , `HTTP Referer`의 필드	아니요	예
수명 주기 전환, 만료, 복원	아니요	예
일괄 삭제 작업에서 키 로깅	예	예
인증 실패[1]	아니요	예
로그가 전달되는 계정	버킷 소유자[2] 및 요청자	버킷 소유자만
Performance and Cost	AWS CloudTrail	Amazon S3 Server Logs

로그 속성	AWS CloudTrail	Amazon S3 서버 로그
가격	관리 이벤트(첫 전달)는 무료, 로그 저장 외에 데이터 이벤트에도 요금 발생	로그 저장 외에 다른 비용 없음
로그 전달 속도	5분마다 데이터 이벤트, 15분마다 관리 이벤트	몇 시간 내에
로그 형식	JSON	공백으로 구분되고 줄 바꿈으로 구분된 레코드가 있는 로그 파일

Notes

1. CloudTrail은 인증에 실패한 요청(제공된 자격 증명이 유효하지 않은 요청) 또는 리디렉션으로 인해 실패한 요청(오류 코드 `301 Moved Permanently`)에 대한 로그를 전달하지 않습니다. 그러나 권한 부여에 실패한 요청(`AccessDenied`)과 익명 사용자의 요청에 대한 로그는 포함합니다.
2. S3 버킷 소유자 계정이 요청내 객체에 대한 모든 액세스를 보유했을 때 CloudTrail 로그를 수신합니다. 자세한 내용은 크로스 계정 시나리오에서의 Amazon S3 객체 수준 작업 섹션을 참조하세요.
3. S3는 VPC 엔드포인트 정책에서 거부하는 경우 VPC 엔드포인트 요청에 대해, 또는 VPC 정책 평가 전에 실패하는 요청에 대해 요청자 또는 버킷 소유자에게 CloudTrail 로그 또는 서버 액세스 로그를 전송하는 것을 지원하지 않습니다.

AWS CloudTrail을 사용하여 Amazon S3 API 호출 로깅

Amazon S3는 사용자, 역할 또는 AWS 서비스가 수행한 작업에 대한 레코드를 제공하는 서비스인 AWS CloudTrail과 통합됩니다. CloudTrail은 Amazon S3에 대한 모든 API 직접 호출을 이벤트로 캡처합니다. 캡처되는 직접 호출에는 Amazon S3 콘솔로부터의 직접 호출과 Amazon S3 API 작업에 대한 코드 직접 호출이 포함됩니다. CloudTrail에서 수집한 정보를 사용하여 Amazon S3에 전송된 요청, 요청이 이루어진 IP 주소, 요청이 이루어진 시간, 추가 세부 정보를 확인할 수 있습니다.

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

- 요청을 루트 사용자로 했는지 사용자 보안 인증으로 했는지 여부.
- IAM Identity Center 사용자를 대신하여 요청이 이루어졌는지 여부입니다.

- 역할 또는 페더레이션 사용자에 대한 임시 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지 여부.

계정을 생성할 때 AWS 계정에서 CloudTrail이 활성 상태이며, CloudTrail 이벤트 기록에 자동으로 액세스할 수 있습니다. CloudTrail 이벤트 기록은 지난 90일 간 AWS 리전의 관리 이벤트에 대해 보기, 검색 및 다운로드가 가능하고, 수정이 불가능한 레코드를 제공합니다. 자세한 설명은 AWS CloudTrail 사용 설명서의 CloudTrail 이벤트 기록 작업을 참조하세요. Event history(이벤트 기록) 보기는 CloudTrail 요금이 부과되지 않습니다.

지난 90일 동안 AWS 계정에서 진행 중인 이벤트 기록을 보려면 추적 또는 CloudTrail Lake 이벤트 데이터 스토어를 생성합니다.

CloudTrail 추적

CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. AWS Management Console을 사용하여 만든 추적은 모두 다중 리전입니다. AWS CLI를 사용하여 단일 리전 또는 다중 리전 추적을 생성할 수 있습니다. 계정의 모든 AWS 리전에서 활동을 캡처하므로, 다중 리전 추적 생성이 권장됩니다. 단일 리전 추적을 생성하는 경우 추적의 AWS 리전에 로깅된 이벤트만 볼 수 있습니다. 추적에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 Creating a trail for your AWS 계정 및 Creating a trail for an organization을 참조하세요.

CloudTrail에서 추적을 생성하여 진행 중인 관리 이벤트의 사본 하나를 Amazon S3 버킷으로 무료로 전송할 수는 있지만, Amazon S3 스토리지 요금이 부과됩니다. CloudTrail 요금에 대한 자세한 내용은 AWS CloudTrail 요금을 참조하세요. Amazon S3 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하세요.

CloudTrail Lake 이벤트 데이터 스토어

CloudTrail Lake를 사용하면 이벤트에 대해 SQL 기반 쿼리를 실행할 수 있습니다. CloudTrail Lake는 행 기반 JSON 형식의 기존 이벤트를 Apache ORC 형식으로 변환합니다. ORC는 빠른 데이터 검색에 최적화된 열 기반 스토리지 형식입니다. 이벤트는 이벤트 데이터 스토어로 집계되며, 이벤트 데이터 스토어는 고급 이벤트 선택기를 적용하여 선택한 기준을 기반으로 하는 변경 불가능한 이벤트 컬렉션입니다. 이벤트 데이터 스토어에 적용하는 선택기는 어떤 이벤트가 지속되고 쿼리할 수 있는지 제어합니다. CloudTrail Lake에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 Working with AWS CloudTrail Lake를 참조하세요.

CloudTrail Lake 이벤트 데이터 스토어 및 쿼리에는 비용이 발생합니다. 이벤트 데이터 스토어를 생성할 때 이벤트 데이터 스토어에 사용할 요금 옵션을 선택합니다. 요금 옵션에 따라 이벤트 모으기

및 저장 비용과 이벤트 데이터 스토어의 기본 및 최대 보존 기간이 결정됩니다. CloudTrail 요금에 대한 자세한 내용은 AWS CloudTrail 요금을 참조하세요.

원하는 기간만큼 버킷에 로그 파일을 저장할 수 있습니다. 그러나 Amazon S3 수명 주기 규칙을 정의하여 자동으로 로그 파일을 아카이브하거나 삭제할 수도 있습니다. 기본적으로 로그 파일은 Amazon S3 서버 쪽 암호화(SSE)를 사용하여 암호화합니다.

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 “로깅 옵션” 섹션을 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 서버 액세스 로깅을 사용한 요청 로깅을 참조하십시오.

CloudTrail 로그를 Amazon S3용 Amazon CloudWatch와 함께 사용할 수도 있습니다. CloudTrail을 CloudWatch Logs와 통합하면 CloudTrail에서 캡처한 S3 버킷 수준 API 활동을 사용자가 지정한 CloudWatch 로그 그룹의 CloudWatch 로그 스트림에 전달할 수 있습니다. 특정 API 동작을 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 사용 설명서를 참조하십시오. Amazon S3에서 CloudWatch를 사용하는 방법에 대한 자세한 내용은 Amazon CloudWatch를 사용한 지표 모니터링 단원을 참조하십시오.

> **Note**
>
> S3는 VPC 엔드포인트 정책에서 거부하는 경우 VPC 엔드포인트 요청에 대해 요청자 또는 버킷 소유자에게 CloudTrail 로그를 전송하는 것을 지원하지 않습니다.

Amazon S3 SOAP API 호출을 통한 CloudTrail 추적

CloudTrail은 Amazon S3 SOAP API 호출을 추적합니다. HTTP를 통한 Amazon S3 SOAP 지원은 중단되었지만 HTTPS를 통해 계속해서 사용할 수 있습니다. Amazon S3 SOAP 지원에 대한 자세한 내용은 Amazon S3 API 참조의 부록: SOAP API를 참조하세요.

> **Important**
>
> 최신 Amazon S3 기능은 SOAP를 지원하지 않습니다. REST API 또는 AWS SDK를 사용하는 것이 좋습니다.

다음 테이블은 CloudTrail 로깅으로 추적된 Amazon S3 SOAP 작업을 보여줍니다.

SOAP API 이름	CloudTrail 로그에 사용되는 API 이벤트 이름
ListAllMyBuckets	`ListBuckets`
CreateBucket	`CreateBucket`
DeleteBucket	`DeleteBucket`
GetBucketAccessControlPolicy	`GetBucketAcl`
SetBucketAccessControlPolicy	`PutBucketAcl`
GetBucketLoggingStatus	`GetBucketLogging`
SetBucketLoggingStatus	`PutBucketLogging`

CloudTrail 및 Amazon S3에 대한 자세한 내용은 다음 주제를 참조하십시오.

주제

- Amazon S3 CloudTrail 이벤트
- Amazon S3 및 S3 on Outposts에 대한 CloudTrail 로그 파일 항목
- S3 버킷 및 객체에 대한 CloudTrail 이벤트 로깅 사용 설정
- CloudTrail을 사용하여 Amazon S3 요청 식별

Amazon S3 CloudTrail 이벤트

> **Important**
>
> 이제 Amazon S3가 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)를 Amazon S3 내 모든 버킷 암호화의 기본 수준으로 적용합니다. 2023년 1월 5일부터 Amazon S3로의 모든 새 객체 업로드는 추가 비용 없이 성능에 영향을 미치지 않고 자동으로 암호화됩니다. S3 버킷 기본 암호화 구성에 및 신규 객체 업로드에 대한 자동 암호화 상태는 AWS CloudTrail 로그, S3 인벤토리, S3 스토리지 렌즈, Amazon S3 콘솔에서 사용할 수 있으며, AWS Command Line Interface 및 AWS SDK에서 추가 Amazon S3 API 응답 헤더로도 사용할 수 있습니다. 자세한 내용은 기본 암호화 관련 FAQ를 참조하십시오.

이 섹션에서는 S3가 CloudTrail에 기록하는 이벤트에 대한 정보를 제공합니다.

CloudTrail의 Amazon S3 데이터 이벤트

데이터 이벤트는 리소스 기반 또는 리소스에서 수행된 리소스 작업에 대한 정보를 제공합니다(예: Amazon S3 객체 읽기 또는 쓰기). 이를 데이터 영역 작업이라고도 합니다. 데이터 이벤트가 대량 활동인 경우도 있습니다. 기본적으로 CloudTrail은 데이터 이벤트를 로깅하지 않습니다. CloudTrail 이벤트 기록은 데이터 이벤트를 기록하지 않습니다.

데이터 이벤트에는 추가 요금이 적용됩니다. CloudTrail 요금에 대한 자세한 내용은 AWS CloudTrail 요금을 참조하세요.

CloudTrail 콘솔, AWS CLI 또는 CloudTrail API 작업을 사용하여 Amazon S3 리소스 유형에 대한 데이터 이벤트를 로깅할 수 있습니다. 데이터 이벤트를 로깅하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 Logging data events with the AWS Management Console 및 Logging data events with the AWS Command Line Interface를 참조하세요.

다음 테이블에는 데이터 이벤트를 로깅할 수 있는 Amazon S3 유형이 나열되어 있습니다. 데이터 이벤트 유형(콘솔) 열에는 CloudTrail 콘솔의 데이터 이벤트 유형 목록에서 선택할 값이 표시됩니다. resources.type 값 열에는 AWS CLI 또는 CloudTrail API를 사용하여 고급 이벤트 선택기를 구성할 때 지정하는 `resources.type` 값이 표시됩니다. CloudTrail에 로깅되는 데이터 API 열에는 리소스 유형에 대해 CloudTrail에 로깅된 API 호출이 표시됩니다.

데이터 이벤트 유형(콘솔)	resources.type 값	CloudTrail에 로깅되는 데이터 API
S3	`AWS::S3::Object`	• AbortMultipartUpload • CompleteMultipartUpload • CopyObject • CreateMultipartUpload • DeleteObject • DeleteObjectTagging • DeleteObjects • GetObject • GetObjectAcl • GetObjectAttributes • GetObjectLegalHold • GetObjectRetention • GetObjectTagging • GetObjectTorrent • HeadObject • HeadBucket • ListObjectVersions • ListObjects • ListParts • PutObject • PutObjectAcl • PutObjectLegalHold • PutObjectRetention • PutObjectTagging • RestoreObject • SelectObjectContent • UploadPart

데이터 이벤트 유형(콘솔)	resources.type 값	CloudTrail에 로깅되는 데이터 API
		• UploadPartCopy
S3 Express One Zone	`AWS::S3Express::Object`	• AbortMultipartUpload • CompleteMultipartUpload • CreateSession • CopyObject • CreateMultipartUpload • DeleteObject • DeleteObjects • GetObject • GetObjectAttributes • HeadBucket • HeadObject • ListObjectsV2 • ListParts • PutObject • UploadPart • UploadPartCopy

데이터 이벤트 유형(콘솔)	resources.type 값	CloudTrail에 로깅되는 데이터 API
S3 액세스 포인트	`AWS::S3::Access Point`	• AbortMultipartUpload • CompleteMultipartUpload • CopyObject(동일한 리전 사본만 해당) • CreateMultipartUpload • DeleteObject • DeleteObjectTagging • GetBucketAcl • GetBucketCors • GetBucketLocation • GetBucketNotificationConfiguration • GetBucketPolicy • GetObject • GetObjectAcl • GetObjectAttributes • GetObjectLegalHold • GetObjectRetention • GetObjectTagging • HeadBucket • HeadObject • ListObjects • ListObjectsV2 • ListObjectVersions • ListParts • Presign • PutObject • PutObjectLegalHold

데이터 이벤트 유형(콘솔)	resources.type 값	CloudTrail에 로깅되는 데이터 API
		• PutObjectRetention • PutObjectAcl • PutObjectTagging • RestoreObject • UploadPart • UploadPartCopy(동일한 리전 사본만 해당)

데이터 이벤트 유형(콘솔)	resources.type 값	CloudTrail에 로깅되는 데이터 API
S3 객체 Lambda	`AWS::S3ObjectLambda::AccessPoint`	• AbortMultipartUpload • CompleteMultipartUpload • CopyObject(동일한 리전 사본만 해당) • CreateMultipartUpload • DeleteObject • DeleteObjectTagging • GetObject • GetObjectAcl • GetObjectLegalHold • GetObjectRetention • GetObjectTagging • HeadObject • ListObjects • ListObjectVersions • ListParts • PutObject • PutObjectLegalHold • PutObjectRetention • PutObjectAcl • PutObjectTagging • RestoreObject • UploadPart • WriteGetObjectResponse

데이터 이벤트 유형(콘솔)	resources.type 값	CloudTrail에 로깅되는 데이터 API
S3 Outposts	`AWS::S3Outposts::Object`	• AbortMultipartUpload • CompleteMultipartUpload • CopyObject(동일한 리전 사본만 해당) • CreateMultipartUpload • DeleteObject • DeleteObjectTagging • GetObject • GetObjectTagging • HeadObject • ListObjects • ListObjectsV2 • ListParts • PutObject • PutObjectTagging • UploadPart • UploadPartCopy

`eventName`, `readOnly` 및 `resources.ARN` 필드를 필터링하여 중요한 이벤트만 로깅하도록 고급 이벤트 선택기를 구성할 수 있습니다. 이러한 필드에 대한 자세한 내용은 AWS CloudTrail API 참조의 AdvancedFieldSelector 섹션을 참조하세요.

CloudTrail의 Amazon S3 관리 이벤트

Amazon S3는 모든 컨트롤 플레인 작업을 관리 이벤트로 로깅합니다. S3 API 작업에 대한 자세한 내용은 Amazon S3 API 참조를 참조하세요.

CloudTrail이 Amazon S3에 대한 요청을 캡처하는 방법

기본적으로 CloudTrail은 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 데이터 이벤트(Amazon S3 객체 수준 호출)는 볼 수 없으므로 CloudTrail 로그를 구문 분석하거나 쿼리해야 합니다.

AWS CloudTrail을 사용하여 데이터 활동을 로깅하는 경우 Amazon S3 DeleteObjects 데이터 이벤트에 대한 이벤트 레코드에는 DeleteObjects 이벤트 및 해당 작업의 일부로 삭제된 각 객체에 대한 DeleteObject 이벤트가 모두 포함됩니다. 이벤트 레코드에서 삭제된 객체에 대한 추가 가시성을 제외할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서의 AWS CLI examples for filtering data events를 참조하세요.

CloudTrail 로깅을 통해 추적되는 Amazon S3 계정 수준 작업

CloudTrail은 계정 수준 작업을 기록합니다. Amazon S3 레코드가 다른 AWS 서비스 레코드와 함께 로그 파일에 기록됩니다. CloudTrail은 기간 및 파일 크기를 기준으로 새 파일을 만들고 기록하는 시점을 결정합니다.

이 섹션의 표에는 CloudTrail 로깅이 지원되는 Amazon S3 계정 수준 작업이 나와 있습니다.

CloudTrail 로깅을 통해 추적되는 Amazon S3 계정 수준 API 작업은 다음 이벤트 이름으로 표시됩니다. CloudTrail 이벤트 이름은 API 작업 이름과 다릅니다. 예를 들어 DeletePublicAccessBlock은 DeleteAccountPublicAccessBlock입니다.

- DeleteAccountPublicAccessBlock
- GetAccountPublicAccessBlock
- PutAccountPublicAccessBlock

CloudTrail 로깅을 통해 추적되는 Amazon S3 버킷 수준 작업

기본적으로 CloudTrail은 범용 버킷의 버킷 수준 작업을 로깅합니다. Amazon S3 레코드가 다른 AWS 서비스 레코드와 함께 로그 파일에 기록됩니다. CloudTrail은 기간 및 파일 크기를 기준으로 새 파일을 만들고 기록하는 시점을 결정합니다.

이 섹션에는 CloudTrail 로깅이 지원되는 Amazon S3 버킷 수준 작업이 나와 있습니다.

CloudTrail 로깅을 통해 추적되는 Amazon S3 버킷 수준 API 작업은 다음 이벤트 이름으로 표시됩니다. 경우에 따라 CloudTrail 이벤트 이름은 API 작업 이름과 다릅니다. 예를 들어 PutBucketLifecycleConfiguration은 PutBucketLifecycle입니다.

- CreateBucket

- CreateBucketMetadataTableConfiguration
- DeleteBucket
- DeleteBucketAnalyticsConfiguration
- DeleteBucketCors
- DeleteBucketEncryption
- DeleteBucketIntelligentTieringConfiguration
- DeleteBucketInventoryConfiguration
- DeleteBucketLifecycle
- DeleteBucketMetadataTableConfiguration
- DeleteBucketMetricsConfiguration
- DeleteBucketOwnershipControls
- DeleteBucketPolicy
- DeleteBucketPublicAccessBlock
- DeleteBucketReplication
- DeleteBucketTagging
- GetAccelerateConfiguration
- GetBucketAcl
- GetBucketAnalyticsConfiguration
- GetBucketCors
- GetBucketEncryption
- GetBucketIntelligentTieringConfiguration
- GetBucketInventoryConfiguration
- GetBucketLifecycle
- GetBucketLocation
- GetBucketLogging
- GetBucketMetadataTableConfiguration
- GetBucketMetricsConfiguration
- GetBucketNotification
- GetBucketObjectLockConfiguration

- GetBucketOwnershipControls
- GetBucketPolicy
- GetBucketPolicyStatus
- GetBucketPublicAccessBlock
- GetBucketReplication
- GetBucketRequestPayment
- GetBucketTagging
- GetBucketVersioning
- GetBucketWebsite
- HeadBucket
- ListBuckets
- PutAccelerateConfiguration
- PutBucketAcl
- PutBucketAnalyticsConfiguration
- PutBucketCors
- PutBucketEncryption
- PutBucketIntelligentTieringConfiguration
- PutBucketInventoryConfiguration
- PutBucketLifecycle
- PutBucketLogging
- PutBucketMetricsConfiguration
- PutBucketNotification
- PutBucketObjectLockConfiguration
- PutBucketOwnershipControls
- PutBucketPolicy
- PutBucketPublicAccessBlock
- PutBucketReplication
- PutBucketRequestPayment
- PutBucketTagging
- PutBucketVersioning

- PutBucketWebsite

이러한 API 작업 이외에 OPTIONS 객체 객체 수준 작업을 사용할 수도 있습니다. 이 작업은 버킷의 CORS 구성을 확인하므로 CloudTrail 로깅에서 버킷 수준 작업처럼 간주됩니다.

CloudTrail 로깅으로 추적되는 Amazon S3 Express One Zone 버킷 수준(리전 API 엔드포인트) 작업

기본적으로 CloudTrail은 디렉터리 버킷에 대한 버킷 수준 작업을 관리 이벤트로 로깅합니다. S3 Express One Zone의 CloudTrail 관리 이벤트용 `eventsource`는 `s3express.amazonaws.com`입니다.

다음과 같은 리전 엔드포인트 API 작업이 CloudTrail에 로깅됩니다.

- CreateBucket
- DeleteBucket
- DeleteBucketPolicy
- GetBucketPolicy
- PutBucketPolicy
- ListDirectoryBuckets
- PutBucketEncryption
- GetBucketEncryption
- DeleteBucketEncryption

자세한 내용은 S3 Express One Zone에 대한 AWS CloudTrail 로깅을 참조하세요.

크로스 계정 시나리오에서의 Amazon S3 객체 수준 작업

다음은 교차 계정 시나리오에서의 객체 수준 API 호출과 관련한 특별한 사용 사례로, CloudTrail 로그가 어떻게 보고되는지 보여줍니다. CloudTrail은 로그 항목이 삭제되거나 생략되어 액세스가 거부되는 일부 경우를 제외하고 요청자(API 호출을 수행한 계정)에게 로그를 전달합니다. 교차 계정 액세스를 설정할 때, 이 섹션의 예시들을 고려하십시오.

> **Note**
>
> 이 예에서는 CloudTrail 로그가 올바르게 구성되어 있다고 가정합니다.

예 1: CloudTrail이 버킷 소유자에게 로그를 전송함

버킷 소유자가 동일한 객체 API 작업에 대한 권한을 가지지 못한 경우에라도 CloudTrail이 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.
- 계정 C가 객체를 소유하고 있습니다. 계정 C가 계정 A와 동일한 계정일 수도, 아닐 수도 있습니다.

> **Note**
>
> CloudTrail은 항상 객체 수준 API 로그를 요청자(계정 B)에게 전송합니다. 또한, 버킷 소유자가 객체를 소유하지 않았거나(계정 C) 또는 해당 객체에 동일한 API 작업 권한을 가지지 않은 경우에도 CloudTrail이 버킷 소유자(계정 A)에게 동일한 로그를 전송합니다.

예 2: CloudTrail이 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 제어 목록) 개요 단원을 참조하십시오.

이 요청자는 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 이메일 주소와 권한 부여를 가져오지 않습니다. 로그가 버킷 소유자에게 제공하는 유일한 정보는 ACL API 호출이 계정 B에 의해 실행되었다는 것입니다.

Amazon S3 및 S3 on Outposts에 대한 CloudTrail 로그 파일 항목

> **Important**
>
> 이제 Amazon S3가 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)를 Amazon S3 내 모든 버킷 암호화의 기본 수준으로 적용합니다. 2023년 1월 5일부터 Amazon S3로의 모든 새 객체 업로드는 추가 비용 없이 성능에 영향을 미치지 않고 자동으로 암호화됩니다. S3 버킷 기본 암호화 구성에 및 신규 객체 업로드에 대한 자동 암호화 상태는 AWS CloudTrail 로그, S3 인벤토리, S3 스토리지 렌즈, Amazon S3 콘솔에서 사용할 수 있으며, AWS Command Line

Interface 및 AWS SDK에서 추가 Amazon S3 API 응답 헤더로도 사용할 수 있습니다. 자세한 내용은 기본 암호화 관련 FAQ를 참조하십시오.

이벤트는 모든 소스로부터의 단일 요청을 나타내며 요청된 API 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보가 들어 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출의 주문 스택 추적이 아니므로 이벤트가 특정 순서로 표시되지 않습니다.

> **Note**
>
> Amazon S3 Express One Zone에 대한 CloudTrail 로그 파일 예시를 보려면 S3 Express One Zone에 대한 CloudTrail 로그 파일 예시를 참조하세요.

자세한 정보는 다음 예를 참조하십시오.

주제

- 예제: Amazon S3에 대한 CloudTrail 로그 파일 항목

예제: Amazon S3에 대한 CloudTrail 로그 파일 항목

다음은 GET 서비스, PutBucketAcl, GetBucketVersioning 작업을 보여 주는 CloudTrail 로그 항목이 나타낸 예입니다.


```
{
    "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",
            "aclRequired": "Yes"
    },
        "requestID": "47B8E8D397DCE7A6",
        "eventID": "cdc4b7ed-e171-4cef-975a-ad829d4123e8",
        "eventType": "AwsApiCall",
        "recipientAccountId": "444455556666",
        "tlsDetails": {
            "tlsVersion": "TLSv1.2",
            "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
            "clientProvidedHostHeader": "s3.amazonaws.com"
    }
    },
    {
       "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",
    "tlsDetails": {
          "tlsVersion": "TLSv1.2",
          "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
          "clientProvidedHostHeader": "s3.amazonaws.com"
  }
  },
  {
    "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": "amzn-s3-demo-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",
        "tlsDetails": {
              "tlsVersion": "TLSv1.2",
              "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
              "clientProvidedHostHeader": "s3.amazonaws.com"
    }
    }
  ]
}
```


S3 버킷 및 객체에 대한 CloudTrail 이벤트 로깅 사용 설정

CloudTrail 데이터 이벤트를 사용하여 Amazon S3의 버킷 및 객체 수준 요청에 대한 정보를 가져올 수 있습니다. 모든 버킷 또는 특정 버킷 목록에 CloudTrail 데이터 이벤트를 활성화하려면 CloudTrail에서 수동으로 추적을 생성해야 합니다.

> **Note**
>
> - CloudTrail의 기본 설정은 관리 이벤트만 찾는 것입니다. 계정에 데이터 이벤트가 사용 설정되어 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 사용 설정할 것인지, 주의해서 선택해야 합니다.

CloudTrail은 사용자가 선택한 S3 버킷에 Amazon S3 데이터 이벤트 로그를 저장합니다. 보다 간편한 쿼리 및 분석을 위해 중앙 위치에 소유할 수 있는 여러 버킷의 이벤트를 효율적으로 구성하도록 별도의 AWS 계정에서 버킷을 사용하는 방법을 고려합니다. AWS Organizations를 사용하면 모니터링하는 버킷을 소유한 계정과 연결된 AWS 계정을 쉽게 만들 수 있습니다. 자세한 내용은 AWS Organizations 사용 설명서의 AWS Organizations(이)란 무엇입니까? 섹션을 참조하십시오.

CloudTrail에서 추적을 위해 데이터 이벤트를 로깅할 때 고급 이벤트 선택기 또는 기본 이벤트 선택기를 사용하여 범용 버킷에 저장된 객체에 대한 데이터 이벤트를 로깅할 수 있습니다. 디렉터리 버킷에 저장된 객체의 데이터 이벤트를 로깅하려면 고급 이벤트 선택기를 사용해야 합니다. 자세한 내용은 S3 Express One Zone에 대한 AWS CloudTrail 로깅을 참조하세요.

고급 이벤트 선택기를 사용하여 CloudTrail 콘솔에서 추적을 생성할 때 데이터 이벤트 섹션에서 로그 선택기 템플릿의 모든 이벤트 로깅을 선택하여 모든 객체 수준 이벤트를 기록할 수 있습니다. 기본 이벤트선택기를 사용하여 CloudTrail 콘솔에서 추적을 생성할 때 데이터 이벤트 섹션에서 계정의 모든 S3 버킷 선택 확인란을 선택하여 모든 객체 수준 이벤트를 로깅할 수 있습니다.

> **Note**
>
> - AWS CloudTrail 데이터 이벤트 버킷에 대한 수명 주기 구성을 생성하는 것이 모범 사례입니다. 감사에 필요하다고 생각되는 기간이 지나면 로그 파일을 주기적으로 제거하도록 수명 주기 구성을 설정합니다. 그러면 Athena에서 쿼리마다 분석하는 데이터의 양이 줄어듭니다. 자세한 내용은 버킷에서 S3 수명 주기 구성 설정 섹션을 참조하세요.
> - 로깅 형식에 대한 자세한 내용은 AWS CloudTrail을 사용하여 Amazon S3 API 호출 로깅 섹션을 참조하십시오.
> - CloudTrail 로그를 쿼리하는 방법의 예제는 AWS 빅 데이터 블로그 AWS CloudTrail 및 Amazon Athena를 사용하여 보안, 규정 준수 및 운영 활동 분석을 참조하십시오.

콘솔을 사용하여 버킷의 객체에 대한 로깅 사용 설정

Amazon S3 콘솔에서 AWS CloudTrail 추적을 구성하여 S3 버킷의 객체에 대한 데이터 이벤트를 로깅할 수 있습니다. CloudTrail은 `GetObject`, `DeleteObject`, `PutObject` 같은 Amazon S3 객체 수준 API 작업 로깅을 지원합니다. 이 이벤트를 데이터 이벤트라고 합니다.

기본적으로 CloudTrail 추적은 데이터 이벤트를 로깅하지 않지만 지정한 S3 버킷에 대한 데이터 이벤트를 로깅하거나 AWS 계정의 모든 Amazon S3 버킷에 대한 데이터 이벤트를 로깅하도록 추적을 구성할 수 있습니다. 자세한 내용은 AWS CloudTrail을 사용하여 Amazon S3 API 호출 로깅 섹션을 참조하세요.

CloudTrail은 CloudTrail 이벤트 기록의 데이터 이벤트를 채우지 않습니다. 또한 모든 버킷 수준 작업이 CloudTrail 이벤트 기록에 채워지는 것은 아닙니다. CloudTrail 로깅을 통해 추적되는 Amazon S3 버킷 수준 API 작업에 대한 자세한 내용은 CloudTrail 로깅을 통해 추적되는 Amazon S3 버킷 수준 작업 단원을 참조하십시오. CloudTrail 로그를 쿼리하는 방법에 대한 자세한 내용은 Amazon CloudWatch Logs 필터 패턴 및 Amazon Athena를 사용하여 CloudTrail 로그 쿼리에 대한 AWS 지식 센터 문서를 참조하십시오.

> **Note**
>
> AWS CloudTrail을 사용하여 데이터 활동을 로깅하는 경우 Amazon S3 `DeleteObjects` 데이터 이벤트에 대한 이벤트 레코드에는 `DeleteObjects` 이벤트 및 해당 작업의 일부로 삭제된 각 객체에 대한 `DeleteObject` 이벤트가 모두 포함됩니다. 이벤트 레코드에서 삭제된 객체에 대한 추가 가시성을 제외할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서의 AWS CLI examples for filtering data events를 참조하세요.

S3 버킷에 대한 데이터 이벤트를 기록하도록 추적을 구성하기 위해 AWS CloudTrail 콘솔 또는 Amazon S3 콘솔을 사용할 수 있습니다. AWS 계정의 모든 Amazon S3 버킷에 대해 데이터 이벤트를 기록하도록 추적을 구성하는 경우 CloudTrail 콘솔을 사용하는 편이 더 편리합니다. CloudTrail 콘솔을 사용하여 S3 데이터 이벤트를 로깅하도록 추적을 구성하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서에서 데이터 이벤트를 참조하십시오.

> **Important**
>
> 데이터 이벤트에는 추가 요금이 적용됩니다. 자세한 내용은 AWS CloudTrail 요금을 참조하십시오.

다음 절차는 Amazon S3 콘솔을 사용하여 CloudTrail 추적이 S3 버킷에 대한 데이터 이벤트를 기록할 수 있도록 구성하는 방법을 보여 줍니다.

S3 범용 버킷 또는 S3 디렉터리 버킷의 객체에 대해 CloudTrail 데이터 이벤트 로깅을 활성화하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 버킷 목록에서 버킷의 이름을 선택합니다.
3. [속성(Properties)]을 선택합니다.
4. AWS CloudTrail 데이터 이벤트에서 CloudTrail에서 구성을 선택합니다.

   새 CloudTrail 추적을 생성하거나 기존 추적을 재사용하고 추적에 기록되도록 Amazon S3 데이터 이벤트를 구성할 수 있습니다. CloudTrail 콘솔에서 추적을 생성하는 자세한 방법은 AWS CloudTrail 사용 설명서에서 콘솔을 사용하여 추적 생성 및 업데이트를 참조하십시오. CloudTrail 콘솔을 사용하여 Amazon S3 데이터 이벤트 로깅을 구성하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서에서 Amazon S3 객체의 데이터 이벤트 로깅을 참조하십시오.

   > **Note**
   >
   > CloudTrail 콘솔이나 Amazon S3 콘솔을 사용하여 S3 버킷에 대해 데이터 이벤트를 기록하도록 추적을 구성하면 Amazon S3 콘솔에는 해당 버킷에 대해 객체 수준 로깅이 사용 설정된 것으로 표시됩니다.

S3 버킷의 객체에 대해 CloudTrail 데이터 이벤트 로깅 사용 중지

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/cloudtrail/에서 CloudTrail 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 추적을 선택합니다.
3. 버킷에 이벤트를 로깅하기 위해 생성한 추적의 이름을 선택합니다.
4. 추적의 세부 정보 페이지에서 오른쪽 상단의 로깅 중지를 선택합니다.
5. 표시되는 대화 상자에서 로깅 중지를 선택합니다.

S3 버킷을 생성할 때 객체 수준 로깅을 사용 설정하는 자세한 방법은 범용 버킷 생성 섹션을 참조하십시오.

S3 버킷을 사용한 CloudTrail 로깅에 대한 자세한 내용은 다음 주제를 참조하십시오.

- S3 범용 버킷에 대한 속성 보기
- AWS CloudTrail을 사용하여 Amazon S3 API 호출 로깅
- AWS CloudTrail 사용 설명서의 CloudTrail 로그 파일 작업

CloudTrail을 사용하여 Amazon S3 요청 식별

Amazon S3에서는 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 “S3 요청 식별” 섹션을 참조하십시오.

주제

- CloudTrail 로그에서 Amazon S3에 대한 요청 식별
- CloudTrail을 사용하여 Amazon S3 서명 버전 2 요청 식별
- CloudTrail을 사용하여 S3 객체에 대한 액세스 식별

CloudTrail 로그에서 Amazon S3에 대한 요청 식별

버킷으로 이벤트를 전달하도록 CloudTrail을 설정한 후 Amazon S3 콘솔에서 객체가 대상 버킷으로 이동하는지 확인해야 합니다. 형식은 다음과 같습니다.

`s3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/Region/yyyy/mm/dd`

CloudTrail에서 기록한 이벤트는 S3 버킷에 gzipped JSON 객체로 저장됩니다. 요청을 효율적으로 찾으려면 Amazon Athena와 같은 서비스를 사용하여 CloudTrail 로그를 인덱싱하고 쿼리해야 합니다.

CloudTrail 및 Athena에 대한 자세한 내용은 Amazon Athena 사용 설명서의 파티션 프로젝션을 사용하여 Athena에서 AWS CloudTrail 로그에 대한 테이블 생성을 참조하십시오.

CloudTrail을 사용하여 Amazon S3 서명 버전 2 요청 식별

CloudTrail 이벤트 로그를 사용하여 Amazon S3의 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.

CloudTrail을 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 re:Post의 Announcement: AWS CloudTrail for Amazon S3 adds new fields for enhanced security auditing(공지: Amazon S3용 CloudTrail, 향상된 보안 감사를 위해 새로운 필드 추가)를 참조하십시오.

> **Note**
>
> Amazon S3에 대한 CloudTrail 이벤트에는 'additionalEventData' 키 이름 아래에 요청 세부 정보의 서명 버전이 포함되어 있습니다. GET, PUT, DELETE 요청 등 Amazon S3 객체에 대한 요청의 서명 버전을 찾으려면 CloudTrail 데이터 이벤트를 활성화해야 합니다. (이 기능은 기본적으로 꺼져 있습니다.)

서명 버전 2 요청을 식별하는 데 사용되는 기본 방법은 AWS CloudTrail입니다. Amazon S3 서버 액세스 로그를 사용하는 경우 Amazon S3 액세스 로그를 사용하여 Signature Version 2 요청 식별 섹션을 참조하십시오.

주제

- Amazon S3 서명 버전 2 요청을 식별하기 위한 Athena 쿼리 예제
- 서명 버전 2 데이터 분할

Amazon S3 서명 버전 2 요청을 식별하기 위한 Athena 쿼리 예제

Example - 모든 서명 버전 2 이벤트 선택, **EventTime**, **S3_Action**, **Request_Parameters**, **Region**, **SourceIP**, **UserAgent**만 인쇄

다음 Athena 쿼리에서 *s3_cloudtrail_events_db.cloudtrail_table*을 Athena 세부 정보로 대체하고 필요에 따라 한도를 늘리거나 제거합니다.

```
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_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_table
WHERE eventsource='s3.amazonaws.com'
    and json_extract_scalar(additionalEventData, '$.SignatureVersion')='SigV2'
Group by useridentity.arn
```

서명 버전 2 데이터 분할

쿼리할 데이터가 많은 경우 분할된 표를 생성하면 Athena의 비용과 실행 시간을 줄일 수 있습니다.

이를 위해 다음과 같이 파티션이 있는 새 테이블을 만드십시오.

```
  CREATE EXTERNAL TABLE s3_cloudtrail_events_db.cloudtrail_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 '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.HiveIgnoreKeyTextOutputFormat'
    LOCATION 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/';
```

그런 다음 개별적으로 파티션을 만듭니다. 생성하지 않은 날짜의 결과는 가져올 수 없습니다.

```
ALTER TABLE s3_cloudtrail_events_db.cloudtrail_table_partitioned ADD
    PARTITION (region= 'us-east-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/us-east-1/2019/02/19/'
    PARTITION (region= 'us-west-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/us-west-1/2019/02/19/'
    PARTITION (region= 'us-west-2', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/us-west-2/2019/02/19/'
    PARTITION (region= 'ap-southeast-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/ap-southeast-1/2019/02/19/'
    PARTITION (region= 'ap-southeast-2', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/ap-southeast-2/2019/02/19/'
    PARTITION (region= 'ap-northeast-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/ap-northeast-1/2019/02/19/'
    PARTITION (region= 'eu-west-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/eu-west-1/2019/02/19/'
    PARTITION (region= 'sa-east-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/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_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
```

CloudTrail을 사용하여 S3 객체에 대한 액세스 식별

AWS CloudTrail 이벤트 로그를 사용하여 `GetObject`, `DeleteObject` 및 `PutObject`와 같은 데이터 이벤트에 대한 Amazon S3 객체 액세스 요청을 식별하고 해당 요청에 대한 추가 정보를 검색할 수 있습니다.

> **Note**
>
> AWS CloudTrail을 사용하여 데이터 활동을 로깅하는 경우 Amazon S3 `DeleteObjects` 데이터 이벤트에 대한 이벤트 레코드에는 `DeleteObjects` 이벤트 및 해당 작업의 일부로 삭제된 각 객체에 대한 `DeleteObject` 이벤트가 모두 포함됩니다. 이벤트 레코드에서 삭제된 객체에 대한 추가 가시성을 제외할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서의 AWS CLI examples for filtering data events를 참조하세요.

다음 예제는 AWS CloudTrail 이벤트 로그에서 Amazon S3에 대한 모든 PUT 객체 요청을 가져오는 방법을 보여줍니다.

주제

- Amazon S3 객체 액세스 요청을 식별하기 위한 Athena 쿼리 예제

Amazon S3 객체 액세스 요청을 식별하기 위한 Athena 쿼리 예제

다음 Athena 쿼리 예제에서 *s3_cloudtrail_events_db.cloudtrail_table*을 Athena 세부 정보로 대체하고 필요에 따라 날짜 범위를 수정합니다.

Example - **PUT** 객체 액세스 요청이 있는 모든 이벤트 선택, **EventTime**, **EventSource**, **SourceIP**, **UserAgent**, **BucketName**, **object**, **UserARN**만 인쇄

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  json_extract_scalar(requestParameters, '$.key') as object,
  userIdentity.arn as userArn
FROM
  s3_cloudtrail_events_db.cloudtrail_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 이벤트 선택, **EventTime**, **EventSource**, **SourceIP**, **UserAgent**, **BucketName**, **object**, **UserARN**만 인쇄

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  json_extract_scalar(requestParameters, '$.key') as object,
  userIdentity.arn as userArn
FROM
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
  eventName = 'GetObject'
  AND eventTime BETWEEN '2019-07-05T00:00:00Z' and '2019-07-06T00:00:00Z'
```

Example - 특정 기간 내 버킷으로의 모든 익명 요청자 이벤트 선택, **EventTime**, **EventName**, **EventSource**, **SourceIP**, **UserAgent**, **BucketName**, **UserARN**, **AccountID**만 인쇄

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  userIdentity.arn as userArn,
  userIdentity.accountId
FROM
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
  userIdentity.accountId = 'anonymous'
  AND eventTime BETWEEN '2019-07-05T00:00:00Z' and '2019-07-06T00:00:00Z'
```

Example - 인증을 위해 ACL이 필요한 모든 요청 식별

다음 Amazon Athena 쿼리 예제에서는 인증을 위해 액세스 제어 목록 (ACL)이 필요한 S3 버킷에 대한 모든 요청을 식별하는 방법을 보여줍니다. 승인을 위해 요청에 ACL이 필요한 경우, additionalEventData 내 aclRequired 값은 Yes입니다. ACL이 필요하지 않은 경우 aclRequired가 표시되지 않습니다. 이 정보를 사용하여 해당 ACL 권한을 적절한 버킷 정책으로 마이그레이션할 수 있습니다. 이러한 버킷 정책을 생성한 후에는 해당 버킷에 ACL을 비활성화할 수 있습니다. ACL에 대한 자세한 내용은 ACL 사용 중지를 위한 사전 조건 페이지를 참조하십시오.

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  userIdentity.arn as userArn,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  json_extract_scalar(requestParameters, '$.key') as object,
  json_extract_scalar(additionalEventData, '$.aclRequired') as aclRequired
FROM
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
  json_extract_scalar(additionalEventData, '$.aclRequired') = 'Yes'
```

```
  AND eventTime BETWEEN '2022-05-10T00:00:00Z' and '2022-08-10T00:00:00Z'
```

> **Note**
>
> - 이 쿼리 예제는 보안 모니터링에도 유용할 수 있습니다. 예상치 못하거나 승인되지 않은 IP 주소 또는 요청자의 `PutObject` 또는 `GetObject` 호출 결과를 검토하고 버킷에 대한 익명 요청을 식별할 수 있습니다.
> - 이 쿼리는 로깅이 사용 설정된 시간부터의 정보만 검색합니다.

Amazon S3 서버 액세스 로그를 사용하는 경우 Amazon S3 액세스 로그를 사용하여 객체 액세스 요청 식별 단원을 참조하십시오.

서버 액세스 로깅을 사용한 요청 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. 이 정보를 통해 고객 기반을 이해하고 Amazon S3 청구 비용을 파악할 수도 있습니다.

> **Note**
>
> 서버 액세스 로그는 2019년 3월 20일 이후에 시작된 리전의 잘못된 리전 리디렉션 오류에 대한 정보를 기록하지 않습니다. 버킷이 존재하는 리전 외부에서 객체 또는 버킷 요청이 생성되면 잘못된 리전 리디렉션 오류가 발생합니다.

로그 전송을 사용 설정하려면 어떻게 해야 합니까?

로그 전송을 사용 설정하려면 다음 기본 단계를 수행합니다. 자세한 내용은 Amazon S3 서버 액세스 로깅 사용 설정을 참조하세요.

1. 대상 버킷의 이름을 입력합니다. 이 버킷은 Amazon S3에서 액세스 로그를 객체로 저장하는 곳입니다. 소스 및 대상 버킷 모두 동일한 AWS 리전에 있어야 하며 동일한 계정에서 소유하고 있어야 합니다. 대상 버킷에는 S3 객체 잠금 기본 보존 기간이 구성되어 있지 않아야 합니다. 또한 대상 버킷에는 요청자 지불이 활성화되어 있지 않아야 합니다.

소스 버킷 자체를 포함하여 소스 버킷과 동일한 리전에 있는 자신의 고유 버킷에 로그를 전달할 수 있습니다. 그러나 로그 관리를 간소화하기 위해서는 액세스 로그를 다른 버킷에 저장하는 것이 좋습니다.

소스 버킷과 대상 버킷이 동일한 버킷이면 버킷에 작성되는 로그에 대해 추가 로그가 생성되므로 로그의 무한 루프가 만들어집니다. 이러한 방식은 스토리지 결제 요금이 약간 증가할 수 있으므로 이 작업을 수행하지 않는 것이 좋습니다. 또한 로그에 대한 추가 로그로 인해 원하는 로그를 찾기가 힘들어질 수 있습니다.

소스 버킷에 액세스 로그를 저장하려는 경우 모든 로그 객체 키에 대상 접두사를 지정하는 것이 좋습니다. 접두사를 지정하면 모든 로그 객체 이름이 공통의 문자열로 시작되므로 로그 객체를 쉽게 식별할 수 있습니다.

2. (선택 사항) 모든 Amazon S3 로그 객체 키에 대상 접두사를 지정합니다. 대상 접두사를 사용하면 더 쉽게 로그 객체를 찾을 수 있습니다. 예를 들어 접두사 값을 `logs/`로 지정할 경우 Amazon S3가 만드는 각 로그 객체의 키는 `logs/` 접두사로 시작합니다.

```
logs/2013-11-01-21-32-16-E568B2907131C0C0
```

접두사 값을 `logs`로 지정하는 경우 로그 객체는 다음과 같이 나타납니다.

```
logs2013-11-01-21-32-16-E568B2907131C0C0
```

접두사는 여러 버킷이 동일한 대상 버킷에 로깅할 때 소스 버킷을 서로 구별하는 데 유용합니다.

로그를 삭제할 경우에도 이 접두사가 유용하게 사용됩니다. 예를 들어 Amazon S3가 특정 접두사를 가진 객체를 삭제하도록 수명 주기 구성 규칙을 설정할 수 있습니다. 자세한 내용은 Amazon S3 로그 파일 삭제 섹션을 참조하세요.

3. (선택 사항) 다른 사용자가 생성된 로그에 액세스할 수 있도록 권한을 설정합니다. 기본적으로 버킷 소유자에게만 항상 로그 객체에 대한 모든 액세스 권한이 부여됩니다. 대상 버킷이 S3 객체 소유권에 버킷 소유자 적용 설정을 사용하여 액세스 제어 목록(ACL)을 비활성화하는 경우 ACL을 사용하는 대상 권한 부여에서 권한을 부여할 수 없습니다. 그러나 대상 버킷에 대한 버킷 정책을 업데이트하여 다른 사용자에게 액세스 권한을 부여할 수 있습니다. 자세한 내용은 Amazon S3의 ID 및 액세스 관리 및 로그 전달을 위한 권한(을)를 참조하세요.

4. (선택 사항) 로그 파일에 로그 객체 키 형식을 설정합니다. 로그 객체 키 형식(대상 객체 키 형식이라고도 함)에는 두 가지 옵션이 있습니다.

- 날짜 기반이 아닌 분할 - 원래 로그 객체 키 형식입니다. 이 형식을 선택하면 로그 파일 키 형식이 다음과 같이 나타납니다.

  ```
  [DestinationPrefix][YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
  ```

  예를 들어 접두사를 `logs/`로 지정하는 경우 로그 객체의 이름은 다음과 같이 지정됩니다.

  ```
  logs/2013-11-01-21-32-16-E568B2907131C0C0
  ```

- 날짜 기반 분할 - 날짜 기반 분할을 선택하는 경우 로그 파일의 이벤트 시간이나 전송 시간을 로그 형식에 사용되는 날짜 소스로 선택할 수 있습니다. 이 형식을 사용하면 로그를 더 쉽게 쿼리할 수 있습니다.

  날짜 기반 분할을 선택하면 로그 파일 키 형식이 다음과 같이 나타납니다.

  ```
  [DestinationPrefix][SourceAccountId]/[SourceRegion]/[SourceBucket]/[YYYY]/[MM]/
  [DD]/[YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
  ```

  예를 들어 대상 접두사를 `logs/`로 지정하는 경우 로그 객체의 이름은 다음과 같이 지정됩니다.

  ```
  logs/123456789012/us-west-2/amzn-s3-demo-source-
  bucket/2023/03/01/2023-03-01-21-32-16-E568B2907131C0C0
  ```

  전송 시간 전송의 경우 로그 파일 이름의 시간은 로그 파일의 전송 시간에 해당합니다.

  이벤트 시간 전송의 경우 연도, 월, 일은 이벤트가 발생한 날에 해당하며 시, 분, 초는 키에서 `00`으로 설정됩니다. 이러한 로그 파일에 전송되는 로그는 특정 날짜에만 제공됩니다.

AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 통해 로그를 구성하는 경우 `TargetObjectKeyFormat`을 사용하여 로그 객체 키 형식을 지정하세요. 날짜 기반이 아닌 분할을 지정하려면 `SimplePrefix`를 사용하세요. 날짜 기반 분할을 지정하려면 `PartitionedPrefix`를 사용하세요. `PartitionedPrefix`를 사용하는 경우 `PartitionDateSource`를 사용하여 `EventTime` 또는 `DeliveryTime`을 지정하세요.

`SimplePrefix`의 경우 로그 파일 키 형식은 다음과 같이 나타납니다.

```
[TargetPrefix][YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
```

이벤트 시간이나 전송 시간이 포함된 PartitionedPrefix의 경우 로그 파일 키 형식은 다음과 같이 나타납니다.

```
[TargetPrefix][SourceAccountId]/[SourceRegion]/[SourceBucket]/[YYYY]/[MM]/[DD]/
[YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
```

로그 객체 키 형식

Amazon S3는 대상 버킷에 업로드하는 로그 객체에 대해 다음과 같은 객체 키 형식을 사용합니다.

- 날짜 기반이 아닌 분할 - 원래 로그 객체 키 형식입니다. 이 형식을 선택하면 로그 파일 키 형식이 다음과 같이 나타납니다.

  ```
  [DestinationPrefix][YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
  ```

- 날짜 기반 분할 - 날짜 기반 분할을 선택하는 경우 로그 파일의 이벤트 시간이나 전송 시간을 로그 형식에 사용되는 날짜 소스로 선택할 수 있습니다. 이 형식을 사용하면 로그를 더 쉽게 쿼리할 수 있습니다.

  날짜 기반 분할을 선택하면 로그 파일 키 형식이 다음과 같이 나타납니다.

  ```
  [DestinationPrefix][SourceAccountId]/[SourceRegion]/[SourceBucket]/[YYYY]/[MM]/[DD]/
  [YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
  ```

로그 객체 키에서 YYYY, MM, DD, hh, mm 및 ss는 각각 연도, 월, 일, 시, 분, 초의 숫자를 나타냅니다. 이러한 날짜 및 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.

로그 전송 방법

Amazon S3는 주기적으로 액세스 로그 레코드를 수집하여 로그 파일에 레코드를 통합한 다음 대상 버킷의 로그 객체로 로그 파일을 업로드합니다. 여러 소스 버킷에서 로깅을 활성화하고 동일한 대상 버킷

을 지정한 경우 대상 버킷에 이러한 모든 소스 버킷의 액세스 로그가 저장됩니다. 그러나 각 로그 객체는 특정 원본 버킷의 액세스 로그 레코드만 보고합니다.

Amazon S3는 특별 로그 전달 계정을 사용하여 서버 액세스 로그를 씁니다. 이 쓰기는 일반적인 액세스 제어 제약 조건을 따릅니다. 액세스 로그 전송을 위해 로깅 서비스 보안 주체(`logging.s3.amazonaws.com`)에 액세스 권한을 부여하도록 대상 버킷의 버킷 정책을 업데이트하는 것이 좋습니다. 버킷 액세스 제어 목록(ACL)을 통해 S3 로그 전송 그룹에 액세스 로그 전송을 위한 액세스 권한을 부여할 수도 있습니다. 그러나 버킷 ACL을 사용하여 S3 로그 전송 그룹에 액세스 권한을 부여하는 것은 권장되지 않습니다.

대상 버킷 정책을 통해 서버 액세스 로깅을 활성화하고 액세스 로그 전송을 위한 액세스 권한을 부여할 때 정책을 업데이트하여 로깅 서비스 보안 주체에게 `s3:PutObject` 액세스를 허용해야 합니다. Amazon S3 콘솔을 사용하여 서버 액세스 로깅을 활성화하는 경우 콘솔은 대상 버킷 정책을 자동으로 업데이트하여 이러한 권한을 로깅 서비스 보안 주체에 부여합니다. 서버 액세스 로그 전달을 위한 권한 부여에 대한 자세한 내용은 로그 전달을 위한 권한 섹션을 참조하십시오.

> **Note**
>
> S3는 VPC 엔드포인트 정책에서 거부하는 경우 VPC 엔드포인트 요청에 대해, 또는 VPC 정책 평가 전에 실패하는 요청에 대해 요청자 또는 버킷 소유자에게 CloudTrail 로그 또는 서버 액세스 로그를 전송하는 것을 지원하지 않습니다.

S3 객체 소유권에 대한 버킷 소유자 시행 설정

대상 버킷이 객체 소유권에 버킷 소유자 적용 설정을 사용하는 경우 ACL이 비활성화되고 더 이상 권한에 영향을 주지 않습니다. 로깅 서비스 보안 주체에 액세스 권한을 부여하려면 대상 버킷의 버킷 정책을 업데이트해야 합니다. 객체 소유권에 대한 자세한 내용은 서버 액세스 로깅을 위해 S3 로그 전송 그룹에 대한 액세스 권한 부여 섹션을 참조하십시오.

서버 로그 전송이 항상 보장되지는 않음

서버 액세스 로그 레코드는 최대한 전송을 시도하지만 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. 로그 레코드가 중복되

는 경우가 발생할 수도 있습니다. 서버 로그는 버킷에 대한 트래픽의 특성을 파악할 용도로 제공되며, 로그 레코드가 손실되거나 중복되는 경우는 매우 드물지만 서버 로깅은 모든 요청을 완벽하게 기록할 목적으로 제공되는 것이 아니라는 점에 유의해야 합니다.

완벽한 전송을 보장할 수 없는 서버 로깅의 특성 때문에 사용 보고서에는 전송된 서버 로그에 포함되지 않은 액세스 요청이 하나 이상 포함될 수 있습니다. 이 사용 보고서는 AWS 결제 및 비용 관리 콘솔의 비용 및 사용량 보고서에서 찾을 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 로깅을 활성화할 경우 이후 1시간 동안 이루어진 요청 중 일부는 로깅되지만 일부는 로깅되지 않을 수도 있습니다. 로깅의 대상 버킷을 버킷 A에서 버킷 B로 변경한다고 가정하겠습니다. 변경 후 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 로그가 전송됩니다.

로깅 및 로그 파일에 대한 자세한 내용은 다음 섹션을 참조하십시오.

주제

- Amazon S3 서버 액세스 로깅 사용 설정
- Amazon S3 서버 액세스 로그 형식
- Amazon S3 로그 파일 삭제
- Amazon S3 서버 액세스 로그를 사용하여 요청 식별
- 서버 액세스 로깅 문제 해결

Amazon S3 서버 액세스 로깅 사용 설정

서버 액세스 로깅은 Amazon S3 버킷에 수행된 요청에 대한 상세 레코드를 제공합니다. 서버 액세스 로그는 많은 애플리케이션에 있어 유용합니다. 예를 들어 액세스 로그 정보는 보안 및 액세스 감사에 유용할 수 있습니다. 이 정보를 통해 고객 기반을 이해하고 Amazon S3 청구 비용을 파악할 수도 있습니다.

Amazon S3는 기본적으로 서버 액세스 로그를 수집하지 않습니다. 로깅을 활성화하면 Amazon S3는 선택된 대상 버킷에 소스 버킷에 대한 액세스 로그를 전송합니다. 대상 버킷은 소스 버킷과 AWS 리전 및 AWS 계정이 동일해야 합니다.

액세스 로그 레코드에는 버킷에 대한 요청 내역이 자세히 나와 있습니다. 이 정보에는 요청 유형, 요청에 지정된 리소스, 요청을 처리한 날짜 및 시간 등이 포함됩니다. 로깅 기초 사항에 대한 자세한 내용은 서버 액세스 로깅을 사용한 요청 로깅 단원을 참조하십시오.

Important

- Amazon S3 버킷에서 서버 액세스 로그를 사용하는 데 따른 별도의 요금은 청구되지 않습니다. 단, 시스템이 사용자에게 전달하는 로그 파일에 대해서는 일반적인 스토리지 요금이 발생합니다. (로그 파일은 언제든지 삭제할 수 있습니다.) 로그 파일 전송에 따른 데이터 전송 요금은 발생하지 않지만 로그 파일 액세스에 따른 일반 데이터 전송 요금은 부과됩니다.
- 대상 버킷에는 서버 액세스 로깅을 활성화해서는 안 됩니다. 소스 버킷 자체를 포함하여 소스 버킷과 동일한 리전에 있는 자신의 고유 버킷에 로그를 전달할 수 있습니다. 그러나 원본 버킷에 로그를 전송하면 로그의 무한 루프가 발생하므로 권장되지 않습니다. 로그 관리를 간소화하기 위해서는 액세스 로그를 다른 버킷에 저장하는 것이 좋습니다. 자세한 내용은 로그 전송을 사용 설정하려면 어떻게 해야 합니까? 섹션을 참조하세요.
- S3 객체 잠금이 활성화된 S3 버킷은 서버 액세스 로그의 대상 버킷으로 사용할 수 없습니다. 대상 버킷에는 기본 보존 기간이 구성되어 있지 않아야 합니다.
- 대상 버킷에는 요청자 지불이 활성화되어 있지 않아야 합니다.
- 256비트 고급 암호화 표준(AES-256)을 사용하는 Amazon S3 관리형 키(SSE-S3)를 통한 서버 측 암호화를 사용할 경우에만 대상 버킷에 기본 버킷 암호화를 사용할 수 있습니다. AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 사용한 기본 서버 측 암호화는 지원되지 않습니다.

Amazon S3 콘솔, Amazon S3 API, AWS Command Line Interface(AWS CLI) 또는 AWS SDK를 사용하여 서버 액세스 로깅을 사용하거나 사용 중지할 수 있습니다.

로그 전달을 위한 권한

Amazon S3는 특별 로그 전달 계정을 사용하여 서버 액세스 로그를 씁니다. 이 쓰기는 일반적인 액세스 제어 제약 조건을 따릅니다. 액세스 로그를 전송하려면 로깅 서비스 보안 주체(`logging.s3.amazonaws.com`)에 대상 버킷에 대한 액세스 권한을 부여해야 합니다.

Amazon S3에 로그 전송 권한을 부여하려면 대상 버킷의 S3 객체 소유권 설정에 따라 버킷 정책 또는 버킷 액세스 제어 목록(ACL)을 사용할 수 있습니다. 그러나 ACL 대신 버킷 정책을 사용하는 것이 좋습니다.

S3 객체 소유권에 대한 버킷 소유자 시행 설정

대상 버킷이 객체 소유권에 버킷 소유자 적용 설정을 사용하는 경우 ACL이 비활성화되고 더 이상 권한에 영향을 주지 않습니다. 이 경우, 대상 버킷에서 로깅 서비스 보안 주체에 액세스 권한을 부여하도록 버킷 정책을 업데이트해야 합니다. S3 로그 전달 그룹에 액세스 권한을 부여하기 위해 버킷 ACL을 업데이트할 수 없습니다. 또한 PutBucketLogging 구성에 대상 권한 부여를 포함할 수 없습니다.

버킷 정책으로 액세스 로그 전달을 위해 기존 버킷 ACL을 마이그레이션하는 방법에 대한 자세한 내용은 서버 액세스 로깅을 위해 S3 로그 전송 그룹에 대한 액세스 권한 부여 섹션을 참조하십시오. 객체 소유권에 대한 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하십시오. 새 버킷을 만들 때 기본적으로 ACL이 비활성화됩니다.

버킷 정책을 사용하여 액세스 권한 부여

대상 버킷에 대한 버킷 정책을 사용하여 액세스 권한을 부여하려면 로깅 서비스 보안 주체에 `s3:PutObject` 권한을 부여하도록 버킷 정책을 업데이트합니다. Amazon S3 콘솔을 사용하여 서버 액세스 로깅을 활성화하는 경우 콘솔은 대상 버킷에 대한 버킷 정책을 자동으로 업데이트하여 이 권한을 로깅 서비스 보안 주체에 부여합니다. 프로그래밍 방식으로 서버 액세스 로깅을 활성화하는 경우 대상 버킷에서 로깅 서비스 보안 주체에 액세스 권한을 부여하도록 버킷 정책을 수동으로 업데이트해야 합니다.

로깅 서비스 보안 주체에 액세스 권한을 부여하는 버킷 정책의 예시는 the section called “버킷 정책을 사용하여 로깅 서비스 보안 주체에 권한 부여” 섹션을 참조하세요.

버킷 ACL을 사용하여 액세스 권한 부여

또는 버킷 ACL을 사용하여 액세스 로그 전달을 위한 액세스 권한을 부여할 수 있습니다. S3 로그 전달 그룹에 `WRITE` 및 `READ_ACP` 권한을 부여하는 권한 부여 항목을 버킷 ACL에 추가합니다. 그러나 버킷 ACL을 사용하여 S3 로그 전달 그룹에 액세스 권한을 부여하는 것은 권장되지 않습니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요. 버킷 정책으로 액세스 로그 전달을 위해 기존 버킷 ACL을 마이그레이션하는 방법에 대한 자세한 내용은 서버 액세스 로깅을 위해 S3 로그 전송 그룹에 대한 액세스 권한 부여 섹션을 참조하십시오. 로깅 서비스 보안 주체에 액세스 권한을 부여하는 ACL의 예시는 the section called “버킷 ACL을 사용하여 로그 전달 그룹에 권한 부여” 섹션을 참조하세요.

버킷 정책을 사용하여 로깅 서비스 보안 주체에 권한 부여

이 예시 버킷 정책은 로깅 서비스 보안 주체(`logging.s3.amazonaws.com`)에 `s3:PutObject` 권한을 부여합니다. 이 버킷 정책을 사용하려면 *`user input placeholders`*를 사용자의 정보로 대체합니다. 다음 정책에서 *`amzn-s3-demo-destination-bucket`*은 서버 액세스 로그가 전송되는 대상 버킷이고 *`amzn-s3-demo-source-bucket`*은 소스 버킷입니다. *`EXAMPLE-LOGGING-PREFIX`*는 원

하는 경우 로그 객체에 선택적으로 사용할 수 있는 대상 접두사입니다. *SOURCE-ACCOUNT-ID*는 소스 버킷을 소유한 AWS 계정입니다.

> **Note**
>
> 버킷 정책에 Deny 명령문이 있는 경우 Amazon S3가 액세스 로그를 전송하는 것을 차단하지 않도록 해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ServerAccessLogsPolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/EXAMPLE-LOGGING-
PREFIX*",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::amzn-s3-demo-source-bucket"
                },
                "StringEquals": {
                    "aws:SourceAccount": "SOURCE-ACCOUNT-ID"
                }
            }
        }
    ]
}
```

버킷 ACL을 사용하여 로그 전달 그룹에 권한 부여

> **Note**
>
> 보안 모범 사례로 Amazon S3는 기본적으로 모든 새 버킷에서 액세스 제어 목록(ACL)을 비활성화합니다. Amazon S3 콘솔의 ACL 권한에 대한 자세한 내용은 ACL 구성 단원을 참조하십시오.

권장하지는 않는 방식이지만 버킷 ACL을 사용하여 로그 전송 그룹에 권한을 부여할 수 있습니다. 그러나 대상 버킷이 객체 소유권에 버킷 소유자 적용 설정을 사용하는 경우 버킷 또는 객체 ACL을 설정할 수 없습니다. 또한 PutBucketLogging 구성에 대상 권한 부여를 포함할 수 없습니다. 대신, 로깅 서비스 보안 주체(`logging.s3.amazonaws.com`)에 액세스 권한을 부여하려면 버킷 정책을 사용해야 합니다. 자세한 내용은 로그 전달을 위한 권한 섹션을 참조하세요.

버킷 ACL에서 로그 전송 그룹은 다음 URL로 표시됩니다.

```
http://acs.amazonaws.com/groups/s3/LogDelivery
```

`WRITE` 및 `READ_ACP`(ACL 읽기) 권한을 부여하려면 대상 버킷 ACL에 다음 권한을 추가합니다.

```
<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>
```

프로그래밍 방식으로 ACL 권한을 추가하는 예는 ACL 구성 단원을 참조하십시오.

> **Important**
>
> 버킷에서 AWS CloudFormation을 사용하여 Amazon S3 서버 액세스 로깅을 활성화하고 ACL을 사용하여 S3 로그 전달 그룹에 액세스 권한을 부여하는 경우 CloudFormation 템플릿에

`"AccessControl": "LogDeliveryWrite"`도 추가해야 합니다. 이것이 중요한 이유는 버킷에 대해 ACL을 생성해야만 이러한 권한을 부여할 수 있지만 CloudFormation에서 버킷에 대한 사용자 지정 ACL을 생성할 수 없기 때문입니다. 미리 준비된 ACL만 CloudFormation에서 사용할 수 있습니다.

서버 액세스 로깅 사용 설정

Amazon S3 콘솔, Amazon S3 REST API, AWS SDK, AWS CLI를 사용하여 서버 액세스 로깅을 활성화하려면 다음 절차를 수행합니다.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 서버 액세스 로깅을 사용 설정하려는 버킷의 이름을 선택합니다.
4. [속성(Properties)]을 선택합니다.
5. 서버 액세스 로깅 섹션에서 편집을 선택합니다.
6. 서버 액세스 로깅에서 활성화를 선택합니다.
7. 대상 버킷에서 버킷과 원하는 경우 접두사를 지정합니다. 접두사를 지정하는 경우 로그를 더 쉽게 찾을 수 있도록 접두사 뒤에 슬래시(/)를 포함하는 것이 좋습니다.

   Note

   접두사를 슬래시(/)와 함께 지정하면 로그 객체를 더 쉽게 구분할 수 있습니다. 예를 들어 접두사 값을 `logs/`로 지정할 경우 Amazon S3가 만드는 각 로그 객체의 키는 다음과 같이 `logs/` 접두사로 시작합니다.

   ```
   logs/2013-11-01-21-32-16-E568B2907131C0C0
   ```

   접두사 값을 `logs`로 지정하는 경우 로그 객체는 다음과 같이 나타납니다.

   ```
   logs2013-11-01-21-32-16-E568B2907131C0C0
   ```

8. 로그 객체 키 형식에서 다음 중 하나를 수행합니다.

- 날짜 기반이 아닌 분할을 선택하려면 [DestinationPrefix][YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]을 선택합니다.
- 날짜 기반 분할을 선택하려면 [DestinationPrefix][SourceAccountId]/[SourceRegion]/[SourceBucket]/[YYYY]/[MM]/[DD]/[YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]을 선택한 다음 S3 이벤트 시간 또는 로그 파일 전송 시간을 선택합니다.

9. 변경 사항 저장을 선택합니다.

   버킷에서 서버 액세스 로깅을 활성화하면 콘솔이 소스 버킷에서 로깅을 사용 설정하고 대상 버킷에서 로깅 서비스 보안 주체(`logging.s3.amazonaws.com`)에 `s3:PutObject` 권한을 부여하도록 버킷 정책을 업데이트합니다. 이 버킷 정책에 대한 자세한 내용은 버킷 정책을 사용하여 로깅 서비스 보안 주체에 권한 부여 섹션을 참조하십시오.

   대상 버킷에서 로그를 볼 수 있습니다. 서버 액세스 로깅을 사용 설정하면 로그가 대상 버킷에 전달되기까지 몇 시간이 소요될 수 있습니다. 로그가 전송되는 방법 및 시기에 대한 자세한 내용은 로그 전송 방법 섹션을 참조하십시오.

자세한 내용은 S3 범용 버킷에 대한 속성 보기 섹션을 참조하세요.

REST API 사용

로깅을 활성화하기 위해 소스 버킷에 로깅 구성을 추가하는 PutBucketLogging 요청을 제출합니다. 요청에 대상 버킷, 그리고 선택 사항으로 모든 로그 객체 키에 사용할 접두사를 지정합니다.

다음 예시에서는 대상 버킷으로 *amzn-s3-demo-destination-bucket*을, 접두사로 *logs/*를 지정합니다.

```
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <LoggingEnabled>
    <TargetBucket>amzn-s3-demo-destination-bucket</TargetBucket>
    <TargetPrefix>logs/</TargetPrefix>
  </LoggingEnabled>
</BucketLoggingStatus>
```

다음 예시에서는 대상 버킷으로 *amzn-s3-demo-destination-bucket*을, 접두사로 *logs/*를, 로그 객체 키 형식으로 `EventTime`을 지정합니다.

```
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <LoggingEnabled>
    <TargetBucket>amzn-s3-demo-destination-bucket</TargetBucket>
```

```
    <TargetPrefix>logs/</TargetPrefix>
    <TargetObjectKeyFormat>
      <PartitionedPrefix>
         <PartitionDateSource>EventTime</PartitionDateSource>
      </PartitionedPrefix>
  </TargetObjectKeyFormat>
  </LoggingEnabled>
</BucketLoggingStatus>
```

S3 로그 전달 계정이 로그 객체를 작성하고 소유하며, 버킷 소유자에게는 로그 객체에 대한 모든 권한이 부여됩니다. 필요한 경우 대상 권한 부여를 사용하여 로그에 액세스할 수 있도록 다른 사용자에게 권한을 부여할 수 있습니다. 자세한 내용은 PutBucketLogging 섹션을 참조하세요.

> **Note**
>
> 대상 버킷이 객체 소유권에 버킷 소유자 적용 설정을 사용하는 경우 대상 권한 부여를 사용하여 다른 사용자에게 권한을 부여할 수 없습니다. 다른 사용자에게 권한을 부여하기 위해 대상 버킷에서 버킷 정책을 업데이트할 수 있습니다. 자세한 내용은 로그 전달을 위한 권한 섹션을 참조하세요.

버킷의 로깅 구성을 검색하려면 GetBucketLogging API 작업을 사용하세요.

로깅 구성을 삭제하려면 빈 `BucketLoggingStatus`로 `PutBucketLogging` 요청을 보냅니다.

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

버킷에서 로깅을 활성화하려면 Amazon S3 API 또는 AWS SDK 래퍼 라이브러리를 사용할 수 있습니다.

AWS SDK 사용

다음은 버킷에서 로깅을 활성화하는 예시입니다. 소스 버킷 하나와 대상 버킷 하나를 생성해야 합니다. 예시는 대상 버킷의 버킷 ACL을 먼저 업데이트합니다. 그런 다음, 먼저 대상 버킷에 로그를 쓰는 데 필요한 권한을 로그 전송 그룹에 부여하고 나서 소스 버킷에서 로깅을 활성화합니다.

이 예시는 객체 소유권에 버킷 소유자 적용 설정을 사용하는 대상 버킷에서는 작동하지 않습니다.

대상 버킷이 객체 소유권에 버킷 소유자 강제 설정을 사용하는 경우 버킷 또는 객체 ACL을 설정할 수 없습니다. 또한 PutBucketLogging 구성에 대상 권한 부여를 포함할 수 없습니다. 로깅 서비스 보안 주

체(logging.s3.amazonaws.com)에 액세스 권한을 부여하려면 버킷 정책을 사용해야 합니다. 자세한 내용은 로그 전달을 위한 권한 섹션을 참조하세요.

.NET

SDK for .NET

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

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

/// <summary>
/// This example shows how to enable logging on an Amazon Simple Storage
/// Service (Amazon S3) bucket. You need to have two Amazon S3 buckets for
/// this example. The first is the bucket for which you wish to enable
/// logging, and the second is the location where you want to store the
/// logs.
/// </summary>
public class ServerAccessLogging
{
    private static IConfiguration _configuration = null!;

    public static async Task Main()
    {
        LoadConfig();

        string bucketName = _configuration["BucketName"];
        string logBucketName = _configuration["LogBucketName"];
        string logObjectKeyPrefix = _configuration["LogObjectKeyPrefix"];
        string accountId = _configuration["AccountId"];

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
```

```
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
            IAmazonS3 client = new AmazonS3Client();

            try
            {
                // Update bucket policy for target bucket to allow delivery of
logs to it.
                await SetBucketPolicyToAllowLogDelivery(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix,
                    accountId);

                // Enable logging on the source bucket.
                await EnableLoggingAsync(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error: {e.Message}");
            }
        }

        /// <summary>
        /// This method grants appropriate permissions for logging to the
        /// Amazon S3 bucket where the logs will be stored.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
used
        /// to apply the bucket policy.</param>
        /// <param name="sourceBucketName">The name of the source bucket.</param>
        /// <param name="logBucketName">The name of the bucket where logging
        /// information will be stored.</param>
        /// <param name="logPrefix">The logging prefix where the logs should be
delivered.</param>
        /// <param name="accountId">The account id of the account where the
source bucket exists.</param>
        /// <returns>Async task.</returns>
        public static async Task SetBucketPolicyToAllowLogDelivery(
```

```
            IAmazonS3 client,
            string sourceBucketName,
            string logBucketName,
            string logPrefix,
            string accountId)
        {
            var resourceArn = @"""arn:aws:s3:::" + logBucketName + "/" +
logPrefix + @"*""";

            var newPolicy = @"{
                                ""Statement"":[{
                                ""Sid"": ""S3ServerAccessLogsPolicy"",
                                ""Effect"": ""Allow"",
                                ""Principal"": { ""Service"":
""logging.s3.amazonaws.com"" },
                                ""Action"": [""s3:PutObject""],
                                ""Resource"": [" + resourceArn + @"],
                                ""Condition"": {
                                ""ArnLike"": { ""aws:SourceArn"":
""arn:aws:s3:::" + sourceBucketName + @""" },
                                ""StringEquals"": { ""aws:SourceAccount"": """ +
accountId + @""" }
                                        }
                                    }]
                                }";
            Console.WriteLine($"The policy to apply to bucket {logBucketName} to
enable logging:");
            Console.WriteLine(newPolicy);

            PutBucketPolicyRequest putRequest = new PutBucketPolicyRequest
            {
                BucketName = logBucketName,
                Policy = newPolicy,
            };
            await client.PutBucketPolicyAsync(putRequest);
            Console.WriteLine("Policy applied.");
        }

        /// <summary>
        /// This method enables logging for an Amazon S3 bucket. Logs will be
stored
        /// in the bucket you selected for logging. Selected prefix
        /// will be prepended to each log object.
        /// </summary>
```

```
        /// <param name="client">The initialized Amazon S3 client which will be
 used
        /// to configure and apply logging to the selected Amazon S3 bucket.</
param>
        /// <param name="bucketName">The name of the Amazon S3 bucket for which
 you
        /// wish to enable logging.</param>
        /// <param name="logBucketName">The name of the Amazon S3 bucket where
 logging
        /// information will be stored.</param>
        /// <param name="logObjectKeyPrefix">The prefix to prepend to each
        /// object key.</param>
        /// <returns>Async task.</returns>
        public static async Task EnableLoggingAsync(
            IAmazonS3 client,
            string bucketName,
            string logBucketName,
            string logObjectKeyPrefix)
        {
            Console.WriteLine($"Enabling logging for bucket {bucketName}.");
            var loggingConfig = new S3BucketLoggingConfig
            {
                TargetBucketName = logBucketName,
                TargetPrefix = logObjectKeyPrefix,
            };

            var putBucketLoggingRequest = new PutBucketLoggingRequest
            {
                BucketName = bucketName,
                LoggingConfig = loggingConfig,
            };
            await client.PutBucketLoggingAsync(putBucketLoggingRequest);
            Console.WriteLine($"Logging enabled.");
        }

        /// <summary>
        /// Loads configuration from settings files.
        /// </summary>
        public static void LoadConfig()
        {
            _configuration = new ConfigurationBuilder()
                .SetBasePath(Directory.GetCurrentDirectory())
                .AddJsonFile("settings.json") // Load settings from .json file.
```

```
                .AddJsonFile("settings.local.json", true) // Optionally, load
 local settings.
                .Build();
        }
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 PutBucketLogging을 참조하십시오.

Java

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLoggingStatus;
import software.amazon.awssdk.services.s3.model.LoggingEnabled;
import software.amazon.awssdk.services.s3.model.PartitionedPrefix;
import software.amazon.awssdk.services.s3.model.PutBucketLoggingRequest;
import software.amazon.awssdk.services.s3.model.TargetObjectKeyFormat;

// Class to set a bucket policy on a target S3 bucket and enable server access
 logging on a source S3 bucket.
public class ServerAccessLogging {
    private static S3Client s3Client;

    public static void main(String[] args) {
        String sourceBucketName = "SOURCE-BUCKET";
        String targetBucketName = "TARGET-BUCKET";
        String sourceAccountId = "123456789012";
        String targetPrefix = "logs/";

        // Create S3 Client.
        s3Client = S3Client.builder().
                region(Region.US_EAST_2)
                .build();

        // Set a bucket policy on the target S3 bucket to enable server access
 logging by granting the
        // logging.s3.amazonaws.com principal permission to use the PutObject
 operation.
        ServerAccessLogging serverAccessLogging = new ServerAccessLogging();
        serverAccessLogging.setTargetBucketPolicy(sourceAccountId, sourceBucketName,
 targetBucketName);
```

```
        // Enable server access logging on the source S3 bucket.
        serverAccessLogging.enableServerAccessLogging(sourceBucketName,
 targetBucketName,
                targetPrefix);

    }

    // Function to set a bucket policy on the target S3 bucket to enable server
 access logging by granting the
    // logging.s3.amazonaws.com principal permission to use the PutObject operation.
    public void setTargetBucketPolicy(String sourceAccountId, String
 sourceBucketName, String targetBucketName) {
        String policy = "{\n" +
                "    \"Version\": \"2012-10-17\",\n" +
                "    \"Statement\": [\n" +
                "        {\n" +
                "            \"Sid\": \"S3ServerAccessLogsPolicy\",\n" +
                "            \"Effect\": \"Allow\",\n" +
                "            \"Principal\": {\"Service\": \"logging.s3.amazonaws.com
\"},\n" +
                "            \"Action\": [\n" +
                "                \"s3:PutObject\"\n" +
                "            ],\n" +
                "            \"Resource\": \"arn:aws:s3:::" + targetBucketName + "/*
\",\n" +
                "            \"Condition\": {\n" +
                "                \"ArnLike\": {\n" +
                "                    \"aws:SourceArn\": \"arn:aws:s3:::" +
 sourceBucketName + "\"\n" +
                "                },\n" +
                "                \"StringEquals\": {\n" +
                "                    \"aws:SourceAccount\": \"" + sourceAccountId +
 "\"\n" +
                "                }\n" +
                "            }\n" +
                "        }\n" +
                "    ]\n" +
                "}";
        s3Client.putBucketPolicy(b -> b.bucket(targetBucketName).policy(policy));
    }

    // Function to enable server access logging on the source S3 bucket.
```

```
    public void enableServerAccessLogging(String sourceBucketName, String
 targetBucketName,
            String targetPrefix) {
        TargetObjectKeyFormat targetObjectKeyFormat =
 TargetObjectKeyFormat.builder()

 .partitionedPrefix(PartitionedPrefix.builder().partitionDateSource("EventTime").build())
                .build();
        LoggingEnabled loggingEnabled = LoggingEnabled.builder()
                .targetBucket(targetBucketName)
                .targetPrefix(targetPrefix)
                .targetObjectKeyFormat(targetObjectKeyFormat)
                .build();
        BucketLoggingStatus bucketLoggingStatus = BucketLoggingStatus.builder()
                .loggingEnabled(loggingEnabled)
                .build();
        s3Client.putBucketLogging(PutBucketLoggingRequest.builder()
                .bucket(sourceBucketName)
                .bucketLoggingStatus(bucketLoggingStatus)
                .build());
    }

}
```

AWS CLI 사용

S3 버킷이 있는 각 AWS 리전에 전용 로깅 버킷을 생성하는 것이 좋습니다. 그런 다음 해당 S3 버킷에 Amazon S3 액세스 로그를 전송하세요. 자세한 내용과 예제는 AWS CLI 참조의 put-bucket-logging 섹션을 참조하십시오.

대상 버킷이 객체 소유권에 버킷 소유자 강제 설정을 사용하는 경우 버킷 또는 객체 ACL을 설정할 수 없습니다. 또한 PutBucketLogging 구성에 대상 권한 부여를 포함할 수 없습니다. 로깅 서비스 보안 주체(`logging.s3.amazonaws.com`)에 액세스 권한을 부여하려면 버킷 정책을 사용해야 합니다. 자세한 내용은 로그 전달을 위한 권한 섹션을 참조하세요.

Example - 2개 리전에 걸쳐 5개 버킷으로 액세스 로그 사용 설정

이 예제에서는 다음과 같은 다섯 개 버킷이 있습니다.

- *amzn-s3-demo-source-bucket*-us-east-1
- *amzn-s3-demo-source-bucket1*-us-east-1

- *amzn-s3-demo-source-bucket2*-us-east-1
- *amzn-s3-demo-bucket1*-us-west-2
- *amzn-s3-demo-bucket2*-us-west-2

> **Note**
>
> 다음 절차의 마지막 단계에서는 로깅 버킷을 생성하고 해당 버킷에서 서버 액세스 로깅을 활성화하는 데 사용할 수 있는 예시 bash 스크립트를 제공합니다. 이러한 스크립트를 사용하려면 다음 절차에 설명된 대로 policy.json 및 logging.json 파일을 만들어야 합니다.

1. 미국 서부(오리건) 및 미국 동부(버지니아 북부) 리전에 로깅 대상 버킷 두 개를 만들고 다음과 같이 이름을 지정합니다.

   - *amzn-s3-demo-destination-bucket*-logs-us-east-1
   - *amzn-s3-demo-destination-bucket1*-logs-us-west-2
2. 이 단계 후반부에서 다음과 같이 서버 액세스 로깅을 활성화합니다.

   - *amzn-s3-demo-source-bucket*-us-east-1은 접두사가 *amzn-s3-demo-source-bucket*-us-east-1인 S3 버킷 *amzn-s3-demo-destination-bucket*-logs-us-east-1에 로깅합니다.
   - *amzn-s3-demo-source-bucket1*-us-east-1은 접두사가 *amzn-s3-demo-source-bucket1*-us-east-1인 S3 버킷 *amzn-s3-demo-destination-bucket*-logs-us-east-1에 로깅합니다.
   - *amzn-s3-demo-source-bucket2*-us-east-1은 접두사가 *amzn-s3-demo-source-bucket2*-us-east-1인 S3 버킷 *amzn-s3-demo-destination-bucket*-logs-us-east-1에 로깅합니다.
   - *amzn-s3-demo-bucket1*-us-west-2은 접두사가 *amzn-s3-demo-bucket1*-us-west-2인 S3 버킷 *amzn-s3-demo-destination-bucket1*-logs-us-west-2에 로깅합니다.
   - *amzn-s3-demo-bucket2*-us-west-2는 접두사가 *amzn-s3-demo-bucket2*-us-west-2인 S3 버킷 *amzn-s3-demo-destination-bucket1*-logs-us-west-2에 로깅합니다.
3. 대상 로깅 버킷 각각에 대해 버킷 ACL 또는 버킷 정책을 사용하여 서버 액세스 로그 전송에 대한 권한을 부여합니다.

- 버킷 정책 업데이트(권장) - 로깅 서비스 보안 주체에 권한을 부여하려면 다음 put-bucket-policy 명령을 사용합니다. *amzn-s3-demo-destination-bucket*-logs를 대상 버킷의 이름으로 바꿉니다.

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-destination-bucket-logs --
policy file://policy.json
```

Policy.json은 다음 버킷 정책이 포함된 현재 폴더의 JSON 문서입니다. 이 버킷 정책을 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다. 다음 정책에서 *amzn-s3-demo-destination-bucket-logs*는 서버 액세스 로그가 전송될 대상 버킷이고 *amzn-s3-demo-source-bucket*은 소스 버킷입니다. *SOURCE-ACCOUNT-ID*는 소스 버킷을 소유한 AWS 계정입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ServerAccessLogsPolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket-logs/*",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::amzn-s3-demo-source-bucket"
                },
                "StringEquals": {
                    "aws:SourceAccount": "SOURCE-ACCOUNT-ID"
                }
            }
        }
    ]
}
```

- 버킷 ACL 업데이트 – S3 로그 전달 그룹에 권한을 부여하려면 다음 `put-bucket-acl` 명령을 사용합니다. *amzn-s3-demo-destination-bucket-logs*를 대상 버킷의 이름으로 바꾸세요.

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-destination-bucket-logs  --
grant-write URI=http://acs.amazonaws.com/groups/s3/LogDelivery --grant-read-acp
 URI=http://acs.amazonaws.com/groups/s3/LogDelivery
```

4. 그런 다음, 아래 세 가지 예시 중 하나를 기반으로 로깅 구성이 포함된 `logging.json` 파일을 생성합니다. `logging.json` 파일을 만든 후 다음 `put-bucket-logging` 명령을 사용하여 로깅 구성을 적용할 수 있습니다. *amzn-s3-demo-destination-bucket-logs*를 대상 버킷의 이름으로 바꾸세요.

```
aws s3api put-bucket-logging --bucket amzn-s3-demo-destination-bucket-logs --
bucket-logging-status file://logging.json
```

> **Note**
>
> 이 `put-bucket-logging` 명령을 사용하여 각 대상 버킷에 로깅 구성을 적용하는 대신 다음 단계에서 제공하는 bash 스크립트 중 하나를 사용할 수 있습니다. 이러한 스크립트를 사용하려면 이 절차에 설명된 대로 `policy.json` 및 `logging.json` 파일을 만들어야 합니다.

`logging.json` 파일은 로깅 구성이 포함된 현재 폴더의 JSON 문서입니다. 대상 버킷이 객체 소유권에 버킷 소유자 적용 설정을 사용하는 경우 로깅 구성에 대상 권한 부여가 포함될 수 없습니다. 자세한 내용은 로그 전달을 위한 권한 섹션을 참조하세요.

Example - 대상 권한 부여가 없는 **`logging.json`**

다음 예시 `logging.json` 파일에는 대상 권한 부여가 없습니다. 따라서 객체 소유권에 대해 버킷 소유자 적용 설정을 사용하는 대상 버킷에 이 구성을 적용할 수 있습니다.

```
{
```

```
    "LoggingEnabled": {
        "TargetBucket": "amzn-s3-demo-destination-bucket-logs",
        "TargetPrefix": "amzn-s3-demo-destination-bucket/"
     }
 }
```

Example - 대상 권한 부여가 있는 **logging.json**

다음 예시 logging.json 파일에는 대상 권한 부여가 포함되어 있습니다.

대상 버킷이 객체 소유권에 버킷 소유자 적용 설정을 사용하는 경우 PutBucketLogging 구성에 대상 권한 부여를 포함할 수 없습니다. 자세한 내용은 로그 전달을 위한 권한 섹션을 참조하세요.

```
{
    "LoggingEnabled": {
        "TargetBucket": "amzn-s3-demo-destination-bucket-logs",
        "TargetPrefix": "amzn-s3-demo-destination-bucket/",
        "TargetGrants": [
             {
                "Grantee": {
                    "Type": "AmazonCustomerByEmail",
                    "EmailAddress": "user@example.com"
                 },
                "Permission": "FULL_CONTROL"
             }
         ]
     }
 }
```

피부여자 값

다음과 같은 방법으로 액세스 권한을 할당할 사람(피부여자)을 지정할 수 있습니다(요청 요소 사용).

- 개인의 ID:

```
{
  "Grantee": {
    "Type": "CanonicalUser",
```

```
    "ID": "ID",
    "DisplayName": "GranteesEmail"
  }
}
```

DisplayName은 선택 사항이며 요청에서 무시됩니다.

- 이메일 주소:

```
{
  "Grantee": {
    "Type": "AmazonCustomerByEmail",
    "EmailAddress": "username@example.com"
  }
}
```

피부여자는 CanonicalUser로 확인되고 GetObjectAcl 요청에 대한 응답에서 CanonicalUser로 표시됩니다.

> **Note**
>
> 피부여자를 지정하기 위해 이메일 주소를 사용하는 것은 일부 AWS 리전에서만 지원됩니다. 자세한 내용은 Amazon S3 API 참조의 Grantee 섹션을 참조하세요.

- URI:

```
{
  "Grantee": {
    "Type": "Group",
    "URI": "http://acs.amazonaws.com/groups/global/AuthenticatedUsers"
  }
}
```

Example - 로그 객체 키 형식이 S3 이벤트 시간으로 설정된 **logging.json**

다음 logging.json 파일은 로그 객체 키 형식을 S3 이벤트 시간으로 변경합니다. 로그 객체 키 형식 설정에 대한 자세한 내용은 the section called “로그 전송을 사용 설정하려면 어떻게 해야 합니까?” 섹션을 참조하세요.

```
  {
    "LoggingEnabled": {
        "TargetBucket": "amzn-s3-demo-destination-bucket-logs",
        "TargetPrefix": "amzn-s3-demo-destination-bucket/",
        "TargetObjectKeyFormat": {
            "PartitionedPrefix": {
                "PartitionDateSource": "EventTime"
            }
         }
    }
}
```

5. 다음 bash 스크립트 중 하나를 사용하여 계정의 모든 버킷에 대한 액세스 로깅을 추가합니다. *amzn-s3-demo-destination-bucket-logs*를 대상 버킷의 이름으로 바꾸고, *us-west-2*를 버킷이 위치한 리전의 이름으로 바꾸세요.

> **Note**
>
> 이 스크립트는 모든 버킷이 동일한 리전에 있는 경우에만 작동합니다. 여러 리전에 버킷이 있는 경우 스크립트를 조정해야 합니다.

Example – 버킷 정책으로 액세스 권한 부여 및 계정의 버킷에 대한 로깅 추가

```
loggingBucket='amzn-s3-demo-destination-bucket-logs'
region='us-west-2'


# Create the logging bucket.
aws s3 mb s3://$loggingBucket --region $region

aws s3api put-bucket-policy --bucket $loggingBucket --policy file://policy.json

# List the buckets in this account.
buckets="$(aws s3 ls | awk '{print $3}')"

# Put a bucket logging configuration on each bucket.
for bucket in $buckets
    do
      # This if statement excludes the logging bucket.
```

```
        if [ "$bucket" == "$loggingBucket" ] ; then
            continue;
        fi
        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"
```

Example – 버킷 ACL로 액세스 권한 부여 및 계정의 버킷에 대한 로깅 추가

```
  loggingBucket='amzn-s3-demo-destination-bucket-logs'
  region='us-west-2'


  # Create the 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 the buckets in this account.
  buckets="$(aws s3 ls | awk '{print $3}')"

  # Put a bucket logging configuration on each bucket.
  for bucket in $buckets
      do
        # This if statement excludes the logging bucket.
        if [ "$bucket" == "$loggingBucket" ] ; then
            continue;
        fi
        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"
```

서버 액세스 로그 설정 확인

서버 액세스 로깅을 활성화한 후 다음 단계를 완료합니다.

- 대상 버킷에 액세스하여 로그 파일이 전송되고 있는지 확인합니다. 액세스 로그가 설정되면 Amazon S3는 즉시 요청 캡처 및 로깅을 시작합니다. 하지만 로그가 대상 버킷으로 전송되려면 몇 시간이 소요될 수 있습니다. 자세한 내용은 the section called “버킷 로깅 상태 변경 시 일정 기간에 걸쳐 단계적으로 반영됨” 및 the section called “서버 로그 전송이 항상 보장되지는 않음”(을)를 참조하세요.

  또한 Amazon S3 요청 지표를 사용하고 이러한 지표에 대한 Amazon CloudWatch 경보를 설정하여 로그 전달을 자동으로 확인할 수 있습니다. 자세한 내용은 Amazon CloudWatch를 사용한 지표 모니터링 섹션을 참조하세요.
- 로그 파일의 내용을 열고 읽을 수 있는지 확인합니다.

서버 액세스 로깅 문제 해결 정보는 서버 액세스 로깅 문제 해결 섹션을 참조하십시오.

Amazon S3 서버 액세스 로그 형식

서버 액세스 로깅은 Amazon S3 버킷에 수행된 요청에 대한 상세 레코드를 제공합니다. 서버 액세스 로그는 다음과 같은 목적으로 사용 가능합니다.

- 보안 및 액세스 감사 수행
- 고객층 조사
- Amazon S3 청구서 이해

이 섹션에서는 Amazon S3 서버 액세스 로그 파일에 대한 형식과 기타 세부 정보를 설명합니다.

서버 액세스 로그 파일은 줄 바꿈으로 구분되는 로그 레코드의 시퀀스로 구성됩니다. 각 로그 레코드는 하나의 요청을 표시하며 공백으로 구분된 필드로 구성됩니다.

다음은 5개 로그 레코드로 구성된 로그 예제입니다.

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 amzn-s3-demo-bucket1 [06/Feb/2019:00:00:38 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 3E57427F3EXAMPLE
 REST.GET.VERSIONING - "GET /amzn-s3-demo-bucket1?versioning HTTP/1.1" 200 - 113 - 7 -
 "-" "S3Console/0.4" - s9lzHYrFp76ZVxRcpX9+5cjAnEH2ROuNkd2BHfIa6UkFVdtjf5mKR3/eTPFvsiP/
XV/VLi31234= SigV4 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader amzn-s3-demo-bucket1.s3.us-
west-1.amazonaws.com TLSV1.2 arn:aws:s3:us-west-1:123456789012:accesspoint/example-AP
 Yes
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 amzn-s3-demo-bucket1 [06/Feb/2019:00:00:38 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 891CE47D2EXAMPLE
 REST.GET.LOGGING_STATUS - "GET /amzn-s3-demo-bucket1?logging HTTP/1.1" 200 -
 242 - 11 - "-" "S3Console/0.4" - 9vKBE6vMhrNiWHZmb2L0mXOcqPGzQOI5XLnCtZNPxev+Hf
+7tpT6sxDwDty4LHBUOZJG96N1234= SigV4 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader amzn-s3-
demo-bucket1.s3.us-west-1.amazonaws.com TLSV1.2 - -
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 amzn-s3-demo-bucket1 [06/Feb/2019:00:00:38 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be A1206F460EXAMPLE
 REST.GET.BUCKETPOLICY - "GET /amzn-s3-demo-bucket1?policy HTTP/1.1" 404
 NoSuchBucketPolicy 297 - 38 - "-" "S3Console/0.4" - BNaBsXZQQDbssi6xMBdBU2sLt
+Yf5kZDmeBUP35sFoKa3sLLeMC78iwEIWxs99CRUrbS4n11234= SigV4 ECDHE-RSA-AES128-GCM-SHA256
 AuthHeader amzn-s3-demo-bucket1.s3.us-west-1.amazonaws.com TLSV1.2 - Yes
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 amzn-s3-demo-bucket1 [06/Feb/2019:00:01:00 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 7B4A0FABBEXAMPLE
 REST.GET.VERSIONING - "GET /amzn-s3-demo-bucket1?versioning HTTP/1.1" 200 -
 113 - 33 - "-" "S3Console/0.4" - Ke1bUcazaN1jWuUlPJaxF64cQVpUEhoZKEG/hmy/gijN/
I1DeWqDfFvnpybfEseEME/u7ME1234= SigV4 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader amzn-s3-
demo-bucket1.s3.us-west-1.amazonaws.com TLSV1.2 - -
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 amzn-s3-demo-bucket1 [06/Feb/2019:00:01:57 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 DD6CC733AEXAMPLE REST.PUT.OBJECT s3-dg.pdf "PUT /amzn-s3-demo-bucket1/
s3-dg.pdf HTTP/1.1" 200 - - 4406583 41754 28 "-" "S3Console/0.4" -
 10S62Zv81kBW7BB6SX4XJ48o6kpcl6LPwEoizZQQxJd5qDSCTLX0TgS37kYUBKQW3+bPdrg1234= SigV4
```

```
ECDHE-RSA-AES128-SHA AuthHeader amzn-s3-demo-bucket1.s3.us-west-1.amazonaws.com
TLSV1.2 - Yes
```

> **Note**
>
> 아무 필드나 -로 설정하여 데이터를 알 수 없거나 사용할 수 없음 또는 해당 필드에 이 요청이 적용되지 않음을 표시할 수 있습니다.

주제

- 로그 레코드 필드
- 복사 작업을 위한 추가 로깅
- 사용자 지정 액세스 로그 정보
- 확장 가능한 서버 액세스 로그 형식에 대한 프로그래밍 고려 사항

로그 레코드 필드

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 일반 참조에서 AWS 계정 식별자를 참조하십시오. 계정의 정식 사용자 ID를 찾는 방법에 대한 자세한 내용은 AWS 계정에 대한 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

```
amzn-s3-demo-bucket1
```

시간

요청이 수신된 시간입니다. 이 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 IP 주소. 중간 프록시 및 방화벽이 요청 시스템의 실제 IP 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 계정와 함께 요청자의 IAM 사용자 이름을 반환합니다. 이 식별자는 액세스 제어 목적으로 사용되는 것과 동일합니다.

입력 예

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

요청자가 수임된 역할을 사용하는 경우 이 필드는 수임된 IAM 역할을 반환합니다.

입력 예

```
arn:aws:sts::123456789012:assumed-role/roleName/test-role
```

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 선언되거나 S3 수명 주기 및 로깅의 경우 `S3.action.resource_type`으로 선언됩니다.

입력 예

```
REST.PUT.OBJECT
```

키

요청의 키(객체 이름) 부분입니다.

입력 예

```
/photos/2019/08/puppy.jpg
```

Request-URI

HTTP 요청 메시지의 `Request-URI` 부분.

입력 예

```
"GET /amzn-s3-demo-bucket1/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 응답, 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 밀리초 단위 시간. 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 S3이 요청을 처리하는 데 소비한 시간(밀리초). 이 값은 요청의 마지막 바이트가 수신된 시간부터 응답의 첫 바이트가 전송된 시간까지 측정됩니다.

입력 예

```
10
```

Referer

HTTP `Referer` 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.example.com/webservices"
```

User-Agent

HTTP `User-Agent` 헤더의 값.

입력 예

```
"curl/7.15.1"
```

버전 ID

요청의 버전 ID. 작업이 `versionId` 파라미터를 사용하지 않는 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 Transport Layer Security(TLS) 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)의 경우 `QueryString`, 미인증 요청의 경우 `-`.

입력 예

```
AuthHeader
```

호스트 헤더

Amazon S3에 연결하는 데 사용된 엔드포인트

입력 예

```
s3.us-west-2.amazonaws.com
```

일부 초기 리전은 레거시 엔드포인트를 지원합니다. 서버 액세스 로그 또는 AWS CloudTrail 로그에 이러한 엔드포인트가 표시될 수 있습니다. 자세한 내용은 레거시 엔드포인트 섹션을 참조하세요. Amazon S3 리전 및 엔드포인트의 전체 목록은 Amazon Web Services 일반 참조에서 Amazon S3 엔드포인트 및 할당량을 참조하십시오.

TLS 버전

클라이언트가 협상한 TLS(전송 계층 보안) 버전. 값은 `TLSv1.1`, `TLSv1.2`, `TLSv1.3` 또는 `-`(TLS가 사용되지 않은 경우) 중 하나입니다.

입력 예

```
TLSv1.2
```

액세스 포인트 ARN

요청 액세스 포인트의 Amazon 리소스 이름(ARN) 입니다. 액세스 포인트 ARN의 형식이 잘못되었거나 이를 사용하지 않으면 필드에 `-`가 포함됩니다. 액세스 포인트에 대한 자세한 내용은 범용 버킷의 Amazon S3 Access Points 사용 섹션을 참조하십시오. ARN에 대한 자세한 내용은 AWS 참조 안내서의 Amazon 리소스 이름(ARN)을 참조하십시오.

입력 예

```
arn:aws:s3:us-east-1:123456789012:accesspoint/example-AP
```

aclRequired

승인을 위해 요청에 액세스 제어 목록(ACL)이 필요한지 여부를 나타내는 문자열입니다. 승인을 위해 요청에 ACL이 필요한 경우, 문자열은 `Yes` 입니다. ACL이 필요하지 않은 경우 문자열은 `-`입니다. ACL에 대한 자세한 내용은 ACL(액세스 제어 목록) 개요 단원을 참조하십시오. `aclRequired` 필드를 사용하여 ACL을 비활성화하는 방법에 대한 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 페이지를 참조하십시오.

입력 예

```
Yes
```

복사 작업을 위한 추가 로깅

복사 작업에는 `GET` 및 `PUT`이 관련됩니다. 그러므로 복사 작업을 수행할 때 2개의 레코드가 로그됩니다. 작업의 `PUT` 부분과 관련된 필드는 이전 표에 설명되어 있습니다. 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 일반 참조에서 AWS 계정 식별자를 참조하십시오. 계정의 정식 사용자 ID를 찾는 방법에 대한 자세한 내용은 AWS 계정에 대한 정식 사용자 ID 찾기를 참조하십시오.

입력 예

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

버킷

복사 중인 객체가 저장되어 있는 버킷의 이름.

입력 예

```
amzn-s3-demo-bucket1
```

시간

요청이 수신된 시간입니다. 이 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 IP 주소. 중간 프록시 및 방화벽이 요청 시스템의 실제 IP 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 계정 루트 사용자와 함께 요청자의 IAM 사용자 이름을 반환합니다. 이 식별자는 액세스 제어 목적으로 사용되는 것과 동일합니다.

입력 예

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

요청자가 수임된 역할을 사용하는 경우 이 필드는 수임된 IAM 역할을 반환합니다.

입력 예

```
arn:aws:sts::123456789012:assumed-role/roleName/test-role
```

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.OBJECT로 선언됩니다.

입력 예

```
REST.COPY.OBJECT_GET
```

키

복사 중인 객체의 키(객체 이름). 작업에 키 파라미터가 없을 경우 -.

입력 예

```
/photos/2019/08/puppy.jpg
```

Request-URI

HTTP 요청 메시지의 Request-URI 부분.

입력 예

```
"GET /amzn-s3-demo-bucket1/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 응답, 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 밀리초 단위 시간. 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 S3이 요청을 처리하는 데 소비한 시간(밀리초). 이 값은 요청의 마지막 바이트가 수신된 시간부터 응답의 첫 바이트가 전송된 시간까지 측정됩니다.

입력 예

```
10
```

Referer

HTTP `Referer` 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.example.com/webservices"
```

User-Agent

HTTP `User-Agent` 헤더의 값.

입력 예

```
"curl/7.15.1"
```

버전 ID

복사 중인 객체의 버전 ID. `x-amz-copy-source` 헤더가 복사 소스의 일부로 `versionId` 파라미터를 지정하지 않은 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`. 미인증 요청의 경우 `-`.

입력 예

```
SigV4
```

암호 그룹

HTTPS 요청에 대해 협상된 Transport Layer Security(TLS) 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)의 경우 `QueryString`, 미인증 요청의 경우 `-`.

입력 예

```
AuthHeader
```

호스트 헤더

Amazon S3에 연결하는 데 사용된 엔드포인트.

입력 예

```
s3.us-west-2.amazonaws.com
```

일부 초기 리전은 레거시 엔드포인트를 지원합니다. 서버 액세스 로그 또는 AWS CloudTrail 로그에 이러한 엔드포인트가 표시될 수 있습니다. 자세한 내용은 레거시 엔드포인트 섹션을 참조하세요. Amazon S3 리전 및 엔드포인트의 전체 목록은 Amazon Web Services 일반 참조에서 Amazon S3 엔드포인트 및 할당량을 참조하십시오.

TLS 버전

클라이언트가 협상한 TLS(전송 계층 보안) 버전. 값은 `TLSv1.1`, `TLSv1.2`, `TLSv1.3` 또는 `-`(TLS가 사용되지 않은 경우) 중 하나입니다.

입력 예

```
TLSv1.2
```

액세스 포인트 ARN

요청 액세스 포인트의 Amazon 리소스 이름(ARN) 입니다. 액세스 포인트 ARN의 형식이 잘못되었거나 이를 사용하지 않으면 필드에 `-`가 포함됩니다. 액세스 포인트에 대한 자세한 내용은 범용 버킷의 Amazon S3 Access Points 사용 섹션을 참조하십시오. ARN에 대한 자세한 내용은 AWS 참조 안내서의 Amazon 리소스 이름(ARN)을 참조하십시오.

입력 예

```
arn:aws:s3:us-east-1:123456789012:accesspoint/example-AP
```

aclRequired

승인을 위해 요청에 액세스 제어 목록(ACL)이 필요한지 여부를 나타내는 문자열입니다. 승인을 위해 요청에 ACL이 필요한 경우, 문자열은 `Yes` 입니다. ACL이 필요하지 않은 경우 문자열은 `-`입니다. ACL에 대한 자세한 내용은 ACL(액세스 제어 목록) 개요 단원을 참조하십시오. `aclRequired` 필드를 사용하여 ACL을 비활성화하는 방법에 대한 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 페이지를 참조하십시오.

입력 예

```
Yes
```

사용자 지정 액세스 로그 정보

요청에 대한 액세스 로그 레코드에 저장할 사용자 지정 정보를 포함할 수 있습니다. 이렇게 하려면 요청의 URL에 사용자 지정 쿼리 문자열 파라미터를 추가합니다. Amazon S3은 `x-`로 시작하는 쿼리 문자열 파라미터를 무시하지만 해당 파라미터를 요청에 대한 액세스 로그 레코드에 로그 레코드 `Request-URI` 필드의 일부로 포함시킵니다.

예를 들어 `GET`에 대한 `"s3.amazonaws.com/amzn-s3-demo-bucket1/photos/2019/08/puppy.jpg?x-user=johndoe"` 요청은 관련 로그 레코드의 `"s3.amazonaws.com/amzn-s3-demo-bucket1/photos/2019/08/puppy.jpg"` 필드에 `"x-user=johndoe"` 문자열이 포함된다는 점을 제외하고 `Request-URI`에 대한 요청과 동일하게 작동합니다. 이 기능은 REST 인터페이스에서만 사용할 수 있습니다.

확장 가능한 서버 액세스 로그 형식에 대한 프로그래밍 고려 사항

때때로 각 줄의 끝에 새로운 필드를 추가하여 액세스 로그 레코드 형식을 확장할 수 있습니다. 따라서, 서버 액세스 로그를 파싱하여 이해되지 못할 수 있는 후행 필드를 처리하는 코드를 작성해야 합니다.

Amazon S3 로그 파일 삭제

시간이 지나면 서버 액세스 로깅이 사용 설정된 Amazon S3 버킷에 수많은 서버 로그 객체가 누적될 수 있습니다. 생성 후 일정 기간 동안 애플리케이션에 이 액세스 로그가 필요할 수 있지만, 이후 더 이상 필요가 없어지면 삭제하는 것이 좋습니다. Amazon S3 수명 주기 구성을 사용하여 Amazon S3가 자동으로 수명이 끝난 이러한 객체를 삭제하도록 규칙을 설정할 수 있습니다.

공유 접두사를 사용하여 버킷의 일부 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 객체에 `logs/` 접두사가 있다고 가정해 보세요. 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 섹션을 참조하십시오.

서버 액세스 로깅에 대한 자세한 내용은 서버 액세스 로깅을 사용한 요청 로깅 섹션을 참조하십시오.

Amazon S3 서버 액세스 로그를 사용하여 요청 식별

Amazon S3 서버 액세스 로그를 사용하여 Amazon S3 요청을 식별할 수 있습니다.

> **Note**
>
> - Amazon S3 요청을 식별하려면 Amazon S3 서버 액세스 로그 대신 AWS CloudTrail 데이터 이벤트를 사용하는 것이 좋습니다. CloudTrail 데이터 이벤트는 더 쉽게 설정할 수 있으며 더 많은 정보를 포함합니다. 자세한 내용은 CloudTrail을 사용하여 Amazon S3 요청 식별 섹션을 참조하세요.
> - 받는 액세스 요청의 수에 따라 CloudTrail 데이터 이벤트를 사용하는 것보다 로그를 분석하는 데 더 많은 리소스 또는 시간이 필요할 수 있습니다.

주제

- Amazon Athena를 사용하여 요청에 대한 액세스 로그 쿼리
- Amazon S3 액세스 로그를 사용하여 Signature Version 2 요청 식별
- Amazon S3 액세스 로그를 사용하여 객체 액세스 요청 식별

Amazon Athena를 사용하여 요청에 대한 액세스 로그 쿼리

Amazon Athena를 사용하여 Amazon S3 액세스 로그로 Amazon S3 요청을 식별할 수 있습니다.

Amazon S3에서는 서버 액세스 로그를 S3 버킷에 객체로 저장합니다. Amazon S3의 로그를 분석할 수 있는 도구를 사용하는 것이 보통 더 쉽습니다. Athena는 S3 객체 분석을 지원하며 Amazon S3 액세스 로그를 쿼리하는 데 사용할 수 있습니다.

Example

다음 예에서는 Amazon Athena에서 Amazon S3 서버 액세스 로그를 쿼리하는 방법을 보여 줍니다. 다음 예시에 사용된 *user input placeholders*를 실제 정보로 바꾸세요.

> **Note**
>
> Athena 쿼리에서 Amazon S3 위치를 지정하려면 로그가 전송되는 버킷 이름에 대한 S3 URI를 제공해야 합니다. 이 URI에는 버킷 이름과 접두사가 `s3://`*amzn-s3-demo-bucket1-logs/prefix*`/` 형식으로 포함되어야 합니다.

1. https://console.aws.amazon.com/athena/에서 Athena 콘솔을 엽니다.
2. 쿼리 편집기에서 다음과 유사한 명령을 실행합니다. *s3_access_logs_db*를 데이터베이스에 부여하려는 이름으로 바꿉니다.

```
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. *s3_access_logs_db.mybucket_logs*를 테이블에 부여하려는 이름으로 바꿉니다. `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.

Date-based partitioning

```
CREATE EXTERNAL TABLE s3_access_logs_db.mybucket_logs(
 `bucketowner` STRING,
 `bucket_name` STRING,
 `requestdatetime` STRING,
 `remoteip` STRING,
 `requester` STRING,
 `requestid` STRING,
 `operation` STRING,
 `key` STRING,
 `request_uri` 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,
 `accesspointarn` STRING,
 `aclrequired` STRING)
 PARTITIONED BY (
   `timestamp` string)
ROW FORMAT SERDE
 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
 'input.regex'='([^ ]*) ([^ ]*) \\[(.*?)\\] ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*) (\"[^\"]*\"|-) (-|[0-9]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*) (\"[^\"]*\"|-) ([^ ]*)(?: ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*) ([^ ]*) ([^ ]*))?.*$')
STORED AS INPUTFORMAT
 'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
 's3://bucket-name/prefix-name/account-id/region/source-bucket-name/'
 TBLPROPERTIES (
  'projection.enabled'='true',
  'projection.timestamp.format'='yyyy/MM/dd',
  'projection.timestamp.interval'='1',
  'projection.timestamp.interval.unit'='DAYS',
  'projection.timestamp.range'='2024/01/01,NOW',
  'projection.timestamp.type'='date',
  'storage.location.template'='s3://bucket-name/prefix-name/account-id/region/
source-bucket-name/${timestamp}')
```

Non-date-based partitioning

```
CREATE EXTERNAL TABLE `s3_access_logs_db.mybucket_logs`(
  `bucketowner` STRING,
  `bucket_name` STRING,
  `requestdatetime` STRING,
  `remoteip` STRING,
  `requester` STRING,
  `requestid` STRING,
  `operation` STRING,
  `key` STRING,
  `request_uri` 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,
  `accesspointarn` STRING,
  `aclrequired` STRING)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
  'input.regex'='([^ ]*) ([^ ]*) \\[(.*?)\\] ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*) (\"[^\"]*\"|-) (-|[0-9]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*) (\"[^\"]*\"|-) ([^ ]*)(?: ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*) ([^ ]*) ([^ ]*))?.*$')
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  's3://amzn-s3-demo-bucket1-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 수행한 모든 작업 표시

```
SELECT *
FROM s3_access_logs_db.mybucket_logs
WHERE requester='arn:aws:iam::123456789123:user/user_name';
```

Example - 특정 기간에 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 기간에 특정 IP 주소로 전송한 데이터의 양 표시

```
SELECT coalesce(SUM(bytessent), 0) AS bytessenttotal
FROM s3_access_logs_db.mybucket_logs
WHERE remoteip='192.0.2.1'
```

```
AND parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2022-06-01','yyyy-MM-dd')
AND parse_datetime('2022-07-01','yyyy-MM-dd');
```

Example - 특정 기간의 HTTP 5xx 오류에 대한 요청 ID 찾기

```
SELECT requestdatetime, key, httpstatus, errorcode, requestid, hostid
FROM s3_access_logs_db.mybucket_logs
WHERE httpstatus like '5%' AND timestamp
BETWEEN '2024/01/29'
AND '2024/01/30'
```

> **Note**
>
> 로그를 보존하는 시간을 줄이려면 서버 액세스 로그 버킷에 대해 S3 수명 주기 구성을 생성할 수 있습니다. 주기적으로 로그 파일을 제거하는 수명 주기 구성 규칙을 생성합니다. 그러면 Athena에서 쿼리마다 분석하는 데이터의 양이 줄어듭니다. 자세한 내용은 버킷에서 S3 수명 주기 구성 설정 섹션을 참조하세요.

Amazon S3 액세스 로그를 사용하여 Signature Version 2 요청 식별

서명 버전 2에 대한 Amazon S3 지원이 비활성화됩니다(사용되지 않음). 그 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 Signature Version 2 액세스 요청을 식별할 수 있습니다.

> **Note**
>
> Signature Version 2 요청을 식별하려면 Amazon S3 서버 액세스 로그 대신 AWS CloudTrail 데이터 이벤트를 사용하는 것이 좋습니다. CloudTrail 데이터 이벤트는 서버 액세스 로그보다 더 쉽게 설정할 수 있으며 더 많은 정보를 포함합니다. 자세한 내용은 CloudTrail을 사용하여 Amazon S3 서명 버전 2 요청 식별 섹션을 참조하세요.

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 예시에서는 서버 액세스 로그에서 Amazon S3에 대한 모든 PUT 객체 요청을 가져오는 방법을 보여줍니다.

Example - 특정 기간에 **PUT** 객체 요청을 보내는 모든 요청자 표시

```
SELECT bucket_name, 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 예제에서는 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_name, 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 예제에서는 서버 액세스 로그에서 S3 버킷에 대한 모든 익명 요청을 가져오는 방법을 보여 줍니다.

Example - 특정 기간에 버킷에 요청하는 모든 익명 요청자 표시

```
SELECT bucket_name, 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')
```

다음 Amazon Athena 쿼리에서는 인증을 위해 액세스 제어 목록 (ACL)이 필요한 S3 버킷에 대한 모든 요청을 식별하는 방법을 보여줍니다. 이 정보를 사용하여 해당 ACL 권한을 적절한 버킷 정책으로 마이그레이션하고 ACL을 비활성화할 수 있습니다. 이러한 버킷 정책을 생성한 후에는 해당 버킷에 ACL을 비활성화할 수 있습니다. ACL에 대한 자세한 내용은 ACL 사용 중지를 위한 사전 조건 페이지를 참조하십시오.

Example - 인증을 위해 ACL이 필요한 모든 요청 식별

```
SELECT bucket_name, requester, key, operation, aclrequired, requestdatetime
FROM s3_access_logs_db
WHERE aclrequired = 'Yes' AND
parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2022-05-10:00:00:00','yyyy-MM-dd:HH:mm:ss')
AND parse_datetime('2022-08-10:00:00:00','yyyy-MM-dd:HH:mm:ss')
```

Note

- 필요에 따라 요구에 맞게 날짜 범위를 수정할 수 있습니다.
- 이 쿼리 예제는 보안 모니터링에도 유용할 수 있습니다. 예상치 못하거나 승인되지 않은 IP 주소 또는 요청자의 `PutObject` 또는 `GetObject` 호출 결과를 검토하고 버킷에 대한 익명 요청을 식별할 수 있습니다.
- 이 쿼리는 로깅이 사용 설정된 시간부터의 정보만 검색합니다.
- AWS CloudTrail 로그를 사용하는 경우, CloudTrail을 사용하여 S3 객체에 대한 액세스 식별 섹션을 참조하세요.

서버 액세스 로깅 문제 해결

다음 주제는 Amazon S3를 사용하여 로깅을 설정할 때 발생할 수 있는 문제를 해결하는 데 도움이 됩니다.

주제

- 로깅 설정 시 자주 발생하는 오류 메시지
- 전달 실패 문제 해결

로깅 설정 시 자주 발생하는 오류 메시지

AWS Command Line Interface(AWS CLI) 및 AWS SDK를 통해 로깅을 활성화할 때 다음과 같은 일반적인 오류 메시지가 나타날 수 있습니다.

오류: 크로스 S3 위치 로깅이 허용되지 않음

대상 버킷이 소스 버킷과 다른 리전에 있는 경우 크로스 S3 위치 로깅이 허용되지 않음 오류가 발생합니다. 이 오류를 해결하려면 액세스 로그를 수신하도록 구성된 대상 버킷이 소스 버킷과 동일한 AWS 리전 및 AWS 계정에 있어야 합니다.

오류: 로깅할 버킷의 소유자와 대상 버킷이 동일해야 함

서버 액세스 로깅을 활성화할 때 지정된 대상 버킷이 다른 계정에 속할 경우 이 오류가 발생합니다. 이 오류를 해결하려면 대상 버킷이 소스 버킷과 동일한 AWS 계정에 있어야 합니다.

> **Note**
>
> 소스 버킷과 다른 대상 버킷을 선택하는 것이 좋습니다. 소스 버킷과 대상 버킷이 동일하면 버킷에 작성되는 로그에 대해 추가 로그가 생성되어 스토리지 요금이 증가할 수 있습니다. 로그에 대한 이러한 추가 로그로 인해 원하는 특정 로그를 찾기가 어려울 수도 있습니다. 로그 관리를 간소화하기 위해서는 액세스 로그를 다른 버킷에 저장하는 것이 좋습니다. 자세한 내용은 the section called “로그 전송을 사용 설정하려면 어떻게 해야 합니까?” 섹션을 참조하세요.

오류: 로깅을 위한 대상 버킷이 없음

구성을 설정하기 전에 대상 버킷이 있어야 합니다. 이 오류는 대상 버킷이 존재하지 않거나 찾을 수 없음을 나타냅니다. 버킷 이름의 철자가 정확한지 확인한 다음 다시 시도하세요.

오류: 버킷 소유자 적용 버킷에 대상 부여가 허용되지 않음

이 오류는 대상 버킷이 S3 객체 소유권에 대해 버킷 소유자 적용 설정을 사용함을 나타냅니다. 버킷 소유자 적용 설정은 대상 권한 부여를 지원하지 않습니다. 자세한 내용은 로그 전달을 위한 권한 섹션을 참조하세요.

전달 실패 문제 해결

서버 액세스 로깅 문제를 방지하려면 다음 모범 사례를 따라야 합니다.

- S3 로그 전송 그룹에는 대상 버킷에 대한 쓰기 권한이 있어야 함 - S3 로그 전송 그룹은 대상 버킷에 서버 액세스 로그를 전송합니다. 버킷 정책 또는 버킷 액세스 제어 목록(ACL)을 사용하여 대상 버킷에 쓰기 액세스 권한을 부여할 수 있습니다. 그러나 ACL 대신 버킷 정책을 사용하는 것이 좋습니다. 대상 버킷에 액세스 권한을 부여하는 방법에 대한 자세한 내용은 로그 전달을 위한 권한 섹션을 참조하세요.

  > **Note**
  >
  > 대상 버킷이 객체 소유권에 대해 버킷 소유자 적용 설정을 사용하는 경우 다음 사항에 유의하세요.
  >
  > - ACL이 비활성화되어 더 이상 권한에 영향을 미치지 않습니다. S3 로그 전달 그룹에 액세스 권한을 부여하도록 버킷 ACL을 업데이트할 수 없다는 뜻입니다. 로깅 서비스 보안 주체에 액세스 권한을 부여하려면 대상 버킷에 대한 버킷 정책을 업데이트해야 합니다.
  > - `PutBucketLogging` 구성에 대상 권한 부여를 포함할 수 없습니다.

- 대상 버킷의 버킷 정책이 로그에 대한 액세스를 허용해야 함 - 대상 버킷의 버킷 정책을 확인합니다. 버킷 정책에서 `"Effect": "Deny"`를 포함하는 명령문을 검색합니다. 그런 다음, Deny 명령문이 액세스 로그가 버킷에 기록되는 것을 막고 있지 않은지 확인합니다.
- 대상 버킷에서 S3 객체 잠금이 활성화되어 있지 않아야 함 - 대상 버킷에 객체 잠금이 활성화되어 있는지 확인합니다. 오브젝트 잠금은 서버 액세스 로그 전달을 차단합니다. 객체 잠금이 활성화되지 않은 대상 버킷을 선택해야 합니다.
- 대상 버킷에 기본 암호화가 활성화된 경우 Amazon S3 관리형 키(SSE-S3)를 선택해야 함 - Amazon S3 관리형 키를 통한 서버 측 암호화(SSE-S3)를 사용하는 경우에만 대상 버킷에 기본 버킷 암호화를 사용할 수 있습니다. AWS Key Management Service(AWS KMS) 키를 사용한 기본 서버 측 암호화(SSE-KMS)는 서버 액세스 로깅 대상 버킷에 지원되지 않습니다. 기본 암호화를 활성화하는 방법에 대한 자세한 내용은 기본 암호화 구성 섹션을 참조하세요.
- 대상 버킷에 요청자 지불이 활성화되어 있지 않아야 함 - 서버 액세스 로깅을 위해 요청자 지불 버킷을 대상 버킷으로 사용하는 것은 지원되지 않습니다. 서버 액세스 로그 전송을 허용하려면 대상 버킷에서 요청자 지불 옵션을 비활성화하세요.

- AWS Organizations 서비스 제어 정책(SCP) 및 리소스 제어 정책(RCP) 검토 – AWS Organizations를 사용할 때는 서비스 제어 정책과 리소스 제어 정책을 확인하여 Amazon S3 액세스가 허용되는지 확인하세요. 해당 정책은 영향을 받는 계정의 보안 주체 및 리소스에 대한 최대 권한을 지정합니다. 정책에 `"Effect": "Deny"`가 포함된 명령문을 검색하고 `Deny` 명령문이 버킷에 대한 액세스 로그 쓰기를 방해하지 않는지 확인합니다. 자세한 내용은 AWS Organizations 사용 설명서에서 AWS Organizations의 권한 부여 정책을 참조하세요.
- 최근 로깅 구성 변경 사항이 적용될 때까지 대기 - 서버 액세스 로깅을 처음으로 활성화하거나 로그의 대상 버킷을 변경하는 경우 완전히 적용되려면 시간이 필요합니다. 모든 요청이 제대로 로깅되고 전달되려면 한 시간 이상 걸릴 수 있습니다.

  로그 전달 실패를 확인하려면 Amazon CloudWatch에서 요청 지표를 활성화하세요. 몇 시간 내에 로그가 전송되지 않는 경우 로그 전달 실패를 나타낼 수 있는 `4xxErrors` 지표를 찾아보세요. 요청 지표 활성화에 대한 자세한 내용은 the section called “모든 객체에 대한 지표 구성 생성” 섹션을 참조하세요.

Amazon CloudWatch를 사용한 지표 모니터링

Amazon S3용 Amazon CloudWatch 지표는 Amazon S3를 사용하는 애플리케이션의 성능을 이해하고 개선하는 데 도움이 됩니다. Amazon S3에서 CloudWatch를 사용할 수 있는 방법에는 여러 가지가 있습니다.

버킷에 대한 일별 스토리지 지표

Amazon S3에서 스토리지 데이터를 수집하여 읽을 수 있는 일간 지표로 처리하는 CloudWatch를 사용하여 버킷 스토리지를 모니터링합니다. Amazon S3에 대한 이러한 스토리지 지표는 하루에 한 번 보고되고 추가 비용 없이 모든 고객에게 제공됩니다.

요청 지표

지표 요청 - 운영 문제를 신속하게 확인하여 조치하기 위해 Amazon S3 요청을 모니터링합니다. 지표는 약간의 처리 지연시간 후에 1분 간격으로 제공됩니다. 이러한 CloudWatch 지표는 Amazon CloudWatch 사용자 지정 지표와 동일한 요금으로 청구됩니다. CloudWatch 요금에 대한 정보는 Amazon CloudWatch 요금을 참조하십시오. 옵트인하여 이들 지표를 얻는 방법에 대해 알아보려면 CloudWatch 지표 구성 항목을 참조하십시오.

사용 설정되면, 모든 객체 운영에 대한 요청 지표들이 보고됩니다. 기본 설정상, 이 1분 지표들은 Amazon S3 버킷 수준에서 제공됩니다. 공유 접두사, 객체 태그 또는 액세스 포인트를 사용하여 지표에 대한 필터를 정의할 수도 있습니다.

- 액세스 포인트 - 액세스 포인트는 버킷에 연결되어 있는 명명된 네트워크 엔드포인트이며, S3의 공유 데이터 집합에 대한 대규모 데이터 액세스 관리를 간소화합니다. 액세스 포인트 필터를 사용하면 액세스 포인트 사용에 대한 인사이트를 얻을 수 있습니다. 액세스 포인트에 대한 자세한 내용은 액세스 포인트 모니터링 및 로깅 섹션을 참조하십시오.
- 접두사 - Amazon S3 데이터 모델은 단일 구조이긴 하지만, 접두사를 사용하여 계층을 유추할 수 있습니다. 접두사는 유사한 객체를 단일 버킷으로 그룹화하는 데 사용할 수 있는 디렉터리 이름과 유사합니다. S3 콘솔에서는 폴더 개념에 접두사를 지원합니다. 접두사로 필터링할 경우, 동일한 접두사를 갖는 객체들이 지표 구성에 포함됩니다. 접두사에 대한 자세한 정보는 접두어를 사용한 객체 구성 섹션을 참조하세요.
- 태그 - 태그는 객체에 추가할 수 있는 키 값 이름 페어입니다. 태그를 사용하면 객체들을 쉽게 찾아서 조직할 수 있습니다. 또한 태그를 지표 구성에 대한 필터로 사용할 수도 있습니다. 그러면 해당 태그가 있는 객체만 지표 구성에 포함됩니다. 객체 태그에 대한 자세한 내용은 태그를 사용하여 스토리지 분류 섹션을 참조하십시오.

  이러한 지표를 특정한 비즈니스 애플리케이션, 워크플로 또는 내부 조직에 맞추기 위해 공유 접두사, 객체 태그 또는 액세스 포인트를 필터링할 수 있습니다.

복제 지표

  복제가 보류 중인 총 S3 API 작업 수, 복제가 보류 중인 총 객체 크기, 대상 AWS 리전에 대한 최대 복제 시간, 복제가 실패한 총 작업 수를 모니터링합니다. S3 Replication Time Control(S3 RTC) 또는 S3 복제 지표가 활성화된 복제 규칙은 복제 지표를 게시합니다.

  자세한 내용은 지표, 이벤트 알림 및 상태를 통한 복제 모니터링 또는 S3 Replication Time Control을 사용하여 규정 준수 요구 사항 충족 섹션을 참조하십시오.

Amazon S3 스토리지 렌즈 지표

  S3 스토리지 렌즈 사용량 및 활동 지표를 Amazon CloudWatch에 게시하여 CloudWatch 대시보드에서 운영 상태에 대한 통합 보기를 생성할 수 있습니다. S3 스토리지 렌즈 지표는 `AWS/S3/Storage-Lens` 네임스페이스에 사용 가능합니다. CloudWatch 게시 옵션은 고급 지표 및 권장 사항으로 업그레이드된 S3 스토리지 렌즈 대시보드에 사용할 수 있습니다. S3 스토리지 렌즈의 신규 또는 기존 대시보드 구성에 CloudWatch 게시 옵션을 사용하도록 설정할 수 있습니다.

  자세한 내용은 CloudWatch의 S3 Storage Lens 지표 모니터링 섹션을 참조하세요.

모든 CloudWatch 통계는 15개월간 유지되므로 기록 정보를 보고 웹 애플리케이션이나 서비스가 어떻게 실행되고 있는지 전체적으로 더 잘 파악할 수 있습니다. CloudWatch에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch란 무엇입니까?를 참조하십시오. 사용 사례에 따라

CloudWatch 알람에 몇 가지 추가 구성이 필요할 수 있습니다. 예를 들어 지표 수학 표현식을 사용하여 알람을 만들 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서에서 CloudWatch 지표 사용, 지표 수학 사용, Amazon CloudWatch 알람 사용 및 지표 수학 표현식에 기반한 CloudWatch 알람 생성을 참조하십시오.

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. 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 결제 및 비용 관리 대시보드에 제공되는 보고서에는 버킷 지표에 나타나지 않는 액세스 요청이 하나 이상 포함될 수 있습니다.

자세한 내용은 다음 항목을 참조하십시오.

주제

- 지표 및 차원
- CloudWatch 지표 액세스
- CloudWatch 지표 구성

지표 및 차원

다음 테이블에는 Amazon S3에서 Amazon CloudWatch로 전송하는 스토리지 지표와 차원이 나열되어 있습니다.

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. 어떤 시점의 데이터 요소는 CloudWatch를 통해 제공되기 전에 지연될 수도 있으며 어떤 경우에는 아예 전송되지 않을 수도 있

습니다. 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 결제 및 비용 관리 대시보드에 제공되는 보고서에는 버킷 지표에 나타나지 않는 액세스 요청이 하나 이상 포함될 수 있습니다.

주제

- CloudWatch의 버킷에 대한 Amazon S3 일별 스토리지 지표
- CloudWatch의 Amazon S3 요청 지표
- CloudWatch의 S3 복제 지표
- CloudWatch의 S3 스토리지 렌즈 지표
- CloudWatch의 S3 객체 Lambda 요청 지표
- CloudWatch의 Amazon S3 차원
- CloudWatch의 S3 복제 차원
- CloudWatch의 S3 스토리지 렌즈 차원
- CloudWatch의 S3 객체 Lambda 요청 차원
- Amazon S3 사용량 지표

CloudWatch의 버킷에 대한 Amazon S3 일별 스토리지 지표

AWS/S3 네임스페이스에는 버킷에 대한 다음 일간 스토리지 지표가 포함되어 있습니다.

측정치	설명
BucketSizeBytes	다음 스토리지 클래스의 버킷에 저장된 데이터의 양(바이트): • Reduced Redundancy Storage(RRS)(REDUCED_REDUNDANCY) • S3 Express One Zone(EXPRESS_ONEZONE) • S3 Glacier Deep Archive(DEEP_ARCHIVE) • S3 Glacier Flexible Retrieval(GLACIER) • S3 Glacier Instant Retrieval(GLACIER_IR) • S3 Intelligent-Tiering(INTELLIGENT_TIERING) • S3 One Zone-Infrequent Access(ONEZONE_IA) • S3 Standard(STANDARD)

측정치	설명
	• S3 Standard-Infrequent Access(`STANDARD_IA`)

이 값은 버킷에 대한 모든 불완전 멀티파트 업로드의 모든 부분 크기를 포함하여 버킷(현재 객체 및 현재가 아닌 객체 모두)의 모든 객체 및 메타데이터(예: 버킷 이름) 크기를 합산하여 계산됩니다.

> **Note**
>
> S3 Express One Zone 스토리지 클래스는 디렉터리 버킷에만 사용할 수 있습니다.

유효한 스토리지 유형 필터(`StorageType` 차원 참조):

- Reduced Redundancy Storage (RRS): `ReducedRedundancyStorage`
- S3 Express One Zone: `ExpressOneZoneStorage`
- S3 Glacier Deep Archive: `DeepArchiveObjectOverhead`, `DeepArchiveS3ObjectOverhead`, `DeepArchiveStagingStorage`, `DeepArchiveStorage`
- S3 Glacier Flexible Retrieval: `GlacierObjectOverhead`, `GlacierS3ObjectOverhead`, `GlacierStagingStorage`, `GlacierStorage`
- S3 Glacier Instant Retrieval: `GlacierInstantRetrievalStorage`, `GlacierIRSizeOverhead`
- S3 Intelligent-Tiering: `IntelligentTieringAAStorage`, `IntelligentTieringAIAStorage`, `IntelligentTieringDAAStorage`, `IntelligentTieringFAStorage`, `IntelligentTieringIAStorage`
- S3 One Zone-Infrequent Access: `OneZoneIASizeOverhead`, `OneZoneIAStorage`
- S3 Standard: `StandardStorage`

측정치	설명
	• S3 Standard-Infrequent Access: `StandardIAObjectOverhead` , `StandardIASizeOverhead` , `StandardIAStorage` 단위: 바이트 유효 통계: Average `StorageType` 차원에 대한 자세한 내용은 the section called “CloudWatch의 Amazon S3 차원” 섹션을 참조하세요.
`NumberOfObjects`	모든 스토리지 클래스에서 범용 버킷에 저장된 총 객체 수입니다. 이 값은 현재 및 비현재 객체, 삭제 마커, 버킷에 업로드된 모든 미완성 멀티파트 업로드의 총 파트 수를 포함하여 버킷의 모든 객체를 계산하여 계산됩니다. S3 Express One Zone 스토리지 클래스에 객체가 있는 디렉터리 버킷의 경우 이 값은 버킷의 모든 객체 수를 합산하여 계산되지만 버킷에 대한 불완전 다중 업로드는 포함되지 않습니다. 올바른 스토리지 유형 필터 : `AllStorageTypes` (`StorageType` 차원 참조) 단위: 개수 유효 통계: Average

CloudWatch의 Amazon S3 요청 지표

`AWS/S3` 네임스페이스에는 다음 요청 지표가 포함되어 있습니다. 이러한 지표에는 청구 불가능한 요청(`CopyObject` 및 복제를 통한 `GET` 요청의 경우)이 포함됩니다.

> **Note**
>
> CloudWatch의 Amazon S3 요청 지표는 디렉터리 버킷에는 지원되지 않습니다.

지표	설명
AllRequests	유형에 관계 없이 Amazon S3 버킷에 실행된 HTTP 요청의 총 횟수입니다. 필터와 함께 지표 구성을 사용하는 경우, 이 지표는 그 필터의 요건을 충족하는 HTTP 요청만 반환합니다. 단위: 개 유효 통계: Sum
GetRequests	Amazon S3 버킷 내 객체들에 대해 실행된 HTTP GET 요청의 횟수입니다. 여기에는 나열 작업이 포함되지 않습니다. 이 지표는 각 CopyObject 요청의 소스에 대해 증가합니다. 단위: 개 유효 통계: Sum Note ListMultipartUploads, ListParts, ListObjectVersions 등과 같은 페이지가 지정된 목록 기반 요청은 이 지표에 포함되지 않습니다.
PutRequests	Amazon S3 버킷 내 객체들에 대해 실행된 HTTP PUT 요청의 횟수입니다. 이 지표는 각 CopyObject 요청의 대상에 대해 증가합니다. 단위: 개 유효 통계: Sum
DeleteRequests	Amazon S3 버킷 내 객체들에 대해 실행된 HTTP DELETE 요청의 횟수입니다. 이 지표에는 DeleteObjects 요청도 포함됩니다. 이 지표는 삭제된 객체 수가 아니라 실행된 요청 수를 나타냅니다. 단위: 개 유효 통계: Sum
HeadRequests	Amazon S3 버킷에 대해 실행된 HTTP HEAD 요청의 횟수입니다.

지표	설명
	단위: 개 유효 통계: Sum
PostRequests	Amazon S3 버킷에 대해 실행된 HTTP POST 요청의 횟수입니다. 단위: 개 유효 통계: Sum **Note** DeleteObjects 및 SelectObjectContent 요청은 이 지표에 포함되지 않습니다.
SelectRequests	Amazon S3 버킷 내 객체에 대해 실행된 Amazon S3 SelectObjectContent 요청의 횟수입니다. 단위: 개 유효 통계: Sum
SelectBytesScanned	Amazon S3 버킷의 Amazon S3 SelectObjectContent 요청으로 스캔된 데이터 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
SelectBytesReturned	Amazon S3 버킷의 Amazon S3 SelectObjectContent 요청으로 반환된 데이터 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

지표	설명
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 실행된 요청에 대해 업로드된 바이트 수로, 여기서 요청에는 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
4xxErrors	0 또는 1의 값을 가진 Amazon S3 버킷에 실행된 HTTP 4xx 클라이언트 오류 상태 코드 요청 수입니다. Average 통계는 오류율을 보여주고 Sum 통계는 각 기간 동안 그 유형의 오류 수를 보여줍니다. 단위: 개 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 수를 보여줍니다. 단위: 개 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. 이 지표에는 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

CloudWatch의 S3 복제 지표

보류 중인 바이트, 보류 중인 작업, 복제 지연 시간을 추적하여 S3 복제 지표를 사용한 복제 진행률을 모니터링할 수 있습니다. 자세한 내용은 복제 지표로 진행률 모니터링을 참조하세요.

> **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.

측정치	설명
`ReplicationLatency`	지정된 복제 규칙에 대해 복제 대상 AWS 리전이 소스 AWS 리전보다 늦은 최대 시간(초)입니다. 단위: 초

측정치	설명
	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: 개 유효한 통계: Max
`OperationsFailedReplication`	지정된 복제 규칙에 대해 복제에 실패한 작업 수입니다. 단위: 개 유효한 통계: 합계(실패한 총 작업 수), 평균(실패율), 샘플 수(총 복제 작업 수)

CloudWatch의 S3 스토리지 렌즈 지표

S3 스토리지 렌즈 사용량 및 활동 지표를 Amazon CloudWatch에 게시하여 CloudWatch 대시보드에서 운영 상태에 대한 통합 보기를 생성할 수 있습니다. S3 스토리지 렌즈 지표는 CloudWatch의 `AWS/S3/Storage-Lens` 네임스페이스에 게시됩니다. CloudWatch 게시 옵션은 고급 지표 및 권장 사항으로 업그레이드된 S3 스토리지 렌즈 대시보드에 사용할 수 있습니다.

CloudWatch에 게시되는 S3 스토리지 렌즈 지표 목록은 Amazon S3 스토리지 렌즈 지표 용어집 섹션을 참조하십시오. 전체 차원 목록은 Dimensions 섹션을 참조하십시오.

CloudWatch의 S3 객체 Lambda 요청 지표

S3 객체 Lambda에는 다음 요청 지표가 포함되어 있습니다.

지표	설명
`AllRequests`	객체 Lambda 액세스 포인트를 사용하여 Amazon S3 버킷에 실행된 HTTP 요청의 총 횟수입니다.

지표	설명
	단위: 개 유효 통계: Sum
GetRequests	객체 Lambda 액세스 포인트를 사용하여 객체에 대해 실행된 HTTP GET 요청의 횟수입니다. 이 지표에는 나열 작업이 포함되지 않습니다. 단위: 개 유효 통계: Sum
BytesUploaded	객체 Lambda 액세스 포인트를 사용하여 Amazon S3 버킷에 업로드된 바이트 수로, 여기서 요청에는 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
PostRequests	객체 Lambda 액세스 포인트를 사용하여 Amazon S3 버킷에 실행된 HTTP POST 요청의 횟수입니다. 단위: 개 유효 통계: Sum
PutRequests	객체 Lambda 액세스 포인트를 사용하여 Amazon S3 버킷의 객체에 실행된 HTTP PUT 요청의 횟수입니다. 단위: 개 유효 통계: Sum

지표	설명
DeleteRequests	객체 Lambda 액세스 포인트를 사용하여 Amazon S3 버킷의 객체에 실행된 HTTP DELETE 요청의 횟수입니다. 이 지표에는 DeleteObjects 요청이 포함됩니다. 이 지표는 삭제된 객체 수가 아니라 실행된 요청 수를 나타냅니다. 단위: 개 유효 통계: Sum
BytesDownloaded	객체 Lambda 액세스 포인트를 사용하여 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
FirstByteLatency	객체 Lambda 액세스 포인트를 통해 Amazon S3 버킷이 전체 요청을 수신할 때부터 응답이 반환되기 시작하는 때까지의 요청당 시간입니다. 이 지표는 AWS Lambda 함수가 객체 Lambda 액세스 포인트에 바이트를 반환하기 전에 객체를 변환하는 데 걸리는 함수의 실행 시간에 따라 달라집니다. 단위: 밀리초 유효한 통계: Average, Sum, Min, Max(p100과 동일), Sample Count, p0.0과 p100 사이의 모든 백분위수
TotalRequestLatency	최초 수신된 바이트부터 객체 Lambda 액세스 포인트로 전송된 마지막 바이트까지의 요청당 경과 시간입니다. 이 지표에는 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

지표	설명
HeadRequests	객체 Lambda 액세스 포인트를 사용하여 Amazon S3 버킷에 실행된 HTTP HEAD 요청의 횟수입니다. 단위: 개 유효 통계: Sum
ListRequests	Amazon S3 버킷의 콘텐츠를 나열하는 HTTP GET 요청의 수입니다. 이 지표에는 ListObjects 작업과 ListObjectsV2 작업도 포함됩니다. 단위: 개 유효 통계: Sum
4xxErrors	값이 0 또는 1인 객체 Lambda 액세스 포인트를 사용하여 Amazon S3 버킷에 실행된 HTTP 4xx 클라이언트 오류 상태 코드 요청의 수입니다. Average 통계는 오류율을 보여주고 Sum 통계는 각 기간 동안 그 유형의 오류 수를 보여줍니다. 단위: 개 유효 통계: Average(요청당 보고), Sum(기간당 보고), Min, Max, Sample Count
5xxErrors	값이 0 또는 1인 객체 Lambda 액세스 포인트를 사용하여 Amazon S3 버킷에 실행된 HTTP 5xx 서버 오류 상태 코드 요청 수입니다. Average 통계는 오류율을 보여주고 Sum 통계는 각 기간 동안 그 유형의 오류 수를 보여줍니다. 단위: 개 유효 통계: Average(요청당 보고), Sum(기간당 보고), Min, Max, Sample Count

지표	설명
ProxiedRequests	표준 Amazon S3 API 응답을 반환하는 객체 Lambda 액세스 포인트에 대한 HTTP 요청 수입니다. (이러한 요청에는 Lambda 함수가 구성되어 있지 않습니다.) 단위: 개 유효 통계: Sum
InvokedLambda	Lambda 함수가 호출된 S3 객체에 대한 HTTP 요청 수입니다. 단위: 개 유효 통계: Sum
LambdaResponseRequests	Lambda 함수의 WriteGetObjectResponse 요청 수입니다. 이 지표는 GetObject 요청에만 적용됩니다.
LambdaResponse4xx	Lambda 함수에서 WriteGetObjectResponse 를 호출할 때 발생하는 HTTP 4xx 클라이언트 오류 수입니다. 이 지표는 4xxErrors 와 동일한 정보를 제공하지만 WriteGetObjectResponse 호출에만 해당합니다.
LambdaResponse5xx	Lambda 함수에서 WriteGetObjectResponse 를 호출할 때 발생하는 HTTP 5xx 서버 오류 수입니다. 이 지표는 5xxErrors 와 동일한 정보를 제공하지만 WriteGetObjectResponse 호출에만 해당합니다.

CloudWatch의 Amazon S3 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.

차원	설명
	• DeepArchiveObjectOverhead - 보관된 각 객체에 대해 S3 Glacier는 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 S3는 객체 이름 및 기타 메타데이터용으로 8KB의 스토리지를 사용합니다. 이 추가 스토리지에 대해 S3 Standard 요금이 청구됩니다. • DeepArchiveStagingStorage - S3 Glacier Deep Archive 스토리지 클래스에서 객체에 대한 CompleteMultipartUpload 요청이 완료되기 전 멀티파트 업로드 객체의 일부에 사용되는 바이트 수입니다. • DeepArchiveStorage - S3 Glacier Deep Archive 스토리지 클래스의 객체에 사용된 바이트 수입니다. • ExpressOneZoneStorage - S3 Express One Zone 스토리지 클래스의 객체에 사용되는 바이트 수입니다. • GlacierInstantRetrievalStorage - S3 Glacier Instant Retrieval 스토리지 클래스의 객체에 사용된 바이트 수입니다. • GlacierIRSizeOverhead - S3 Glacier Instant Retrieval 스토리지 클래스에서 128KB보다 작은 객체에 사용되는 바이트 수입니다. • GlacierObjectOverhead - 보관된 각 객체에 대해 S3 Glacier는 인덱스 및 관련 메타데이터용으로 32KB의 스토리지를 추가합니다. 이 추가 데이터는 객체를 식별하고 복원하는 데 필요합니다. 이 추가 스토리지에 대해 S3 Glacier Flexible Retrieval 요금이 청구됩니다. • GlacierS3ObjectOverhead - S3 Glacier Flexible Retrieval에 보관되는 각 객체에 대해 Amazon S3는 객체 이름 및 기타 메타데이터용으로 8KB의 스토리지를 사용합니다. 이 추가 스토리지에 대해 S3 Standard 요금이 청구됩니다.

차원	설명
	• `GlacierStagingStorage` - S3 Glacier Flexible Retrieval 스토리지 클래스에서 객체에 대한 `CompleteMultipartUpload` 요청이 완료되기 전 멀티파트 업로드 객체의 일부에 사용되는 바이트 수입니다. • `GlacierStorage` - S3 Glacier Flexible Retrieval 스토리지 클래스의 객체에 사용된 바이트 수입니다. • `IntAAObjectOverhead` - Archive Access 계층의 `INTELLIGENT_TIERING` 스토리지 클래스에 있는 각 객체의 경우, S3 Glacier는 인덱스 및 관련 메타데이터에 대해 32KB의 스토리지를 추가합니다. 이 추가 데이터는 객체를 식별하고 복원하는 데 필요합니다. 이 추가 스토리지에 대해 S3 Glacier Flexible Retrieval 요금이 청구됩니다. • `IntAAS3ObjectOverhead` - Archive Access 계층의 `INTELLIGENT_TIERING` 스토리지 클래스에 있는 각 객체의 경우, Amazon S3는 객체 및 기타 메타데이터의 이름으로 8KB의 스토리지를 사용합니다. 이 추가 스토리지에 대해 S3 Standard 요금이 청구됩니다. • `IntDAAObjectOverhead` - Deep Archive Access 계층의 `INTELLIGENT_TIERING` 스토리지 클래스에 있는 각 객체의 경우, S3 Glacier는 인덱스 및 관련 메타데이터에 대해 32KB의 스토리지를 추가합니다. 이 추가 데이터는 객체를 식별하고 복원하는 데 필요합니다. 이 추가 스토리지에 대해 S3 Glacier Deep Archive 스토리지 요금이 청구됩니다. • `IntDAAS3ObjectOverhead` - Deep Archive Access 계층의 `INTELLIGENT_TIERING` 스토리지 클래스에 있는 각 객체의 경우, Amazon S3는 인덱스 및 관련 메타데이터에 대해 8KB의 스토리지를 추가합니다. 이 추가 데이터는 객체를 식별하고 복원하는 데 필요합니다. 이 추가 스토리지에 대해 S3 Standard 요금이 청구됩니다. • `IntelligentTieringAAStorage` - `INTELLIGENT_TIERING` 스토리지 클래스의 Archive Access 티어에 있는 객체에 사용되는 바이트 수입니다.

차원	설명
	• IntelligentTieringAIAStorage - INTELLIGENT_TIERING 스토리지 클래스의 Archive Instant Access 티어에 있는 객체에 사용되는 바이트 수입니다. • IntelligentTieringDAAStorage - INTELLIGENT_TIERING 스토리지 클래스의 Deep Archive Access 티어에 있는 객체에 사용되는 바이트 수입니다. • IntelligentTieringFAStorage - INTELLIGENT_TIERING 스토리지 클래스의 Frequent Access 티어에 있는 객체에 사용되는 바이트 수입니다. • IntelligentTieringIAStorage - INTELLIGENT_TIERING 스토리지 클래스의 Infrequent Access 티어에 있는 객체에 사용되는 바이트 수입니다. • OneZoneIASizeOverhead - ONEZONE_IA 스토리지 클래스에서 128KB보다 작은 객체에 사용되는 바이트 수입니다. • OneZoneIAStorage - S3 One Zone-Infrequent Access(ONEZONE_IA) 스토리지 클래스의 객체에 사용되는 바이트 수입니다. • ReducedRedundancyStorage - Reduced Redundancy Storage(RRS) 클래스의 객체에 사용된 바이트 수입니다. • StandardIASizeOverhead - STANDARD_IA 스토리지 클래스에서 128KB보다 작은 객체에 사용되는 바이트 수입니다. • StandardIAStorage - S3 Standard-Infrequent Access(STANDARD_IA) 스토리지 클래스의 객체에 사용되는 바이트 수입니다. • StandardStorage - STANDARD 스토리지 클래스의 객체에 사용되는 바이트 수입니다.
FilterId	이 차원은 버킷에서 사용자가 요청 지표에 대해 지정하는 지표 구성을 필터링합니다. 지표 구성을 생성할 때 필터 ID(예: 접두사, 태그 또는 액세스 지점)를 지정합니다. 자세한 내용은 지표 구성 생성을 참조하십시오.

CloudWatch의 S3 복제 차원

다음 차원은 S3 복제 지표 필터링에 사용됩니다.

차원	설명
`SourceBucket`	버킷 개체의 이름이 복제되는 위치입니다.
`DestinationBucket`	버킷 개체의 이름을 복제하는 대상입니다.
`RuleId`	이 복제 지표를 업데이트하도록 트리거한 규칙의 고유 식별자입니다.

CloudWatch의 S3 스토리지 렌즈 차원

CloudWatch의 S3 스토리지 렌즈 지표를 필터링하는 데 사용되는 차원 목록은 Dimensions 섹션을 참조하십시오.

CloudWatch의 S3 객체 Lambda 요청 차원

다음 차원은 객체 Lambda 액세스 포인트에서 데이터를 필터링하는 데 사용됩니다.

차원	설명
`AccessPointName`	요청이 이루어지는 액세스 지점의 이름입니다.
`DataSourceARN`	객체 Lambda 액세스 포인트가 데이터를 검색하는 소스입니다. 요청이 Lambda 함수를 간접적으로 호출하는 경우 이는 Lambda Amazon 리소스 이름(ARN)을 나타냅니다. 그 외의 경우는 액세스 지점 ARN을 의미합니다.

Amazon S3 사용량 지표

CloudWatch 사용량 지표를 사용하여 계정의 리소스 사용량을 확인할 수 있습니다. 이러한 지표를 사용하여 CloudWatch 그래프 및 대시보드에서 현재 서비스 사용량을 시각화합니다.

Amazon S3 사용량 지표는 AWS 서비스 할당량에 해당합니다. 사용량이 서비스 할당량에 가까워지면 경고하는 경보를 구성할 수 있습니다. CloudWatch Service Quotas 통합에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 AWS 사용량 지표를 참조하세요.

Amazon S3는 AWS/Usage 네임스페이스에 다음 지표를 게시합니다.

지표	설명
ResourceCount	계정에서 실행 중인 지정된 리소스의 수입니다. 리소스는 지표와 연결된 차원에 의해 정의됩니다.

다음 차원은 Amazon S3에 의해 게시되는 사용량 지표를 구체화하는 데 사용됩니다.

차원	설명
Service	리소스가 포함된 AWS 서비스의 이름 Amazon S3 사용량 지표의 경우 이 차원 값은 S3입니다.
Type	보고되는 엔터티의 유형입니다. 현재 Amazon S3 사용량 지표에 대한 유일한 유효 값은 Resource입니다.
Resource	실행 중인 리소스의 유형입니다. 현재 Amazon S3 사용량 지표의 유효 값은 AWS 계정의 범용 버킷 수를 반환하는 GeneralPurposeBuckets 뿐입니다. 범용 버킷을 사용하면 S3 Express One Zone을 제외한 모든 스토리지 클래스에 객체를 저장할 수 있습니다.

CloudWatch 지표 액세스

다음 절차에 따라 Amazon S3에 대한 스토리지 지표를 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 및 BucketName 및 StorageType 차원을 설정하십시오.

AWS CLI 사용

예를 들어, AWS CLI를 사용하여 특정 버킷의 평균 크기를 바이트 단위로 받으려는 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=amzn-s3-demo-bucket
 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"
}
```

S3 콘솔 사용

Amazon CloudWatch 콘솔을 사용한 지표 확인

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Metrics를 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 지표 목록 보기

- 명령 프롬프트에서 다음 명령을 사용합니다.

```
aws cloudwatch list-metrics --namespace "AWS/S3"
```

CloudWatch 대시보드에 액세스하는 데 필요한 권한에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch 대시보드 권한에 관한 문서를 참조하십시오.

CloudWatch 지표 구성

Amazon S3에 대한 Amazon CloudWatch 요청 지표를 사용하면 1분 간격 CloudWatch 지표를 수신하고, CloudWatch 경보를 설정하고, CloudWatch 대시보드에 액세스하여 Amazon S3 스토리지의 운영 및 성능을 거의 실시간으로 볼 수 있습니다. 클라우드 스토리지에 의존하는 애플리케이션의 경우, 이 지표를 사용하면 운영 문제를 신속하게 확인하여 조치할 수 있습니다. 이것을 사용 설정하면 이 1분 지표들이 기본적으로 Amazon S3 버킷 수준에서 제공됩니다.

버킷의 객체에 대한 CloudWatch 요청 지표를 얻으려면 그 버킷에 대해 측정 구성을 생성해야 합니다. 자세한 정보는 버킷의 모든 객체에 대한 CloudWatch 지표 구성 생성을 참조하세요.

공유 접두사, 객체 태그 또는 액세스 포인트를 사용하여 수집한 지표에 대한 필터를 정의할 수도 있습니다. 이 필터 정의 방법을 사용하면 특정한 비즈니스 애플리케이션, 워크플로 또는 내부 조직에 맞춰 지표 필터를 적용할 수 있습니다. 자세한 정보는 접두사, 객체 태그 또는 액세스 포인트로 필터링하는 지표 구성 생성을 참조하세요. 사용 가능한 CloudWatch 지표 및 스토리지와 요청 지표의 차이점에 대한 자세한 내용은 Amazon CloudWatch를 사용한 지표 모니터링 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. 지표 필터를 특정한 비즈니스 애플리케이션, 워크플로 또는 내부 조직에 맞추기 위해 공유 접두사, 객체 태그 또는 액세스 포인트를 필터링할 수 있습니다. 전체 버킷에 대한 지표를 요청하려면 필터 없이 지표 구성을 생성합니다.
- 지표 구성은 요청 지표를 사용 설정할 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 전체를 사용 설정합니다. 작업별 지표(예: `PostRequests`)는 해당하는 작업 유형에 대한 요청이 버킷 또는 필터에 들어올 때만 보고됩니다.
- 객체 수준 운영에 대한 요청 지표들이 보고됩니다. 또한 GET Bucket(List Objects), GET Bucket Object Versions 및 List Multipart Uploads와 같이 버킷 콘텐츠를 나열하는 작업에 대해서도 보고되지만 버킷의 다른 작업에 대해서는 보고되지 않습니다.
- 요청 지표는 접두사, 객체 태그 또는 액세스 포인트별 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. 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 결제 및 비용 관리 대시보드에 제공되는 보고서에는 버킷 지표에 나타나지 않는 액세스 요청이 하나 이상 포함될 수 있습니다.

Amazon S3에서 CloudWatch 지표를 사용하는 방법에 대한 자세한 내용은 다음 주제를 참조하십시오.

주제

- 버킷의 모든 객체에 대한 CloudWatch 지표 구성 생성
- 접두사, 객체 태그 또는 액세스 포인트로 필터링하는 지표 구성 생성
- 지표 필터 삭제

버킷의 모든 객체에 대한 CloudWatch 지표 구성 생성

요청 지표를 구성하는 경우 버킷의 모든 객체에 대한 CloudWatch 지표 구성을 생성하거나 접두사, 객체 태그 또는 액세스 포인트로 필터링할 수 있습니다. 이 주제의 절차에서는 버킷의 모든 객체에 대한 구성을 생성하는 방법을 보여 줍니다. 객체 태그 또는 접두사 또는 액세스 포인트로 필터링하는 구성을 생성하려면 접두사, 객체 태그 또는 액세스 포인트로 필터링하는 지표 구성 생성 섹션을 참조하십시오.

Amazon S3용 Amazon CloudWatch 지표는 크게 세 가지 유형 즉, 스토리지 지표, 요청 지표 및 복제 지표로 구분됩니다. 스토리지 지표는 하루에 한 번 보고되고 추가 비용 없이 모든 고객에게 제공됩니다. 요청 지표는 약간의 처리 지연 시간 후에 1분 간격으로 제공됩니다. 요청 지표는 표준 CloudWatch 요금으로 청구됩니다. 콘솔에서 또는 Amazon S3 API를 사용해 요청 지표를 구성하여 이 지표를 선택해야 합니다. S3 복제 지표는 복제 구성의 복제 규칙에 대한 세부 지표를 제공합니다. 복제 지표를 사용하면 보류 중인 바이트, 보류 중인 작업, 복제 실패 작업, 복제 지연 시간을 추적하여 진행률을 분 단위로 모니터링할 수 있습니다.

Amazon S3용 CloudWatch 지표에 대한 자세한 내용은 Amazon CloudWatch를 사용한 지표 모니터링 섹션을 참조하세요.

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 또는 Amazon S3 REST API를 사용하여 버킷에 지표 구성을 추가할 수 있습니다.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 요청 지표가 필요한 객체가 있는 버킷 이름을 선택합니다.
4. 지표 탭을 선택합니다.
5. 버킷 지표(Bucket metrics)에서 추가 차트 보기(Bucket metrics)를 선택합니다.
6. 요청 지표 탭을 선택합니다.
7. 필터 생성을 선택합니다.
8. 필터 이름 상자에 필터 이름을 입력합니다.

   이름은 문자, 숫자, 마침표, 대시 및 밑줄만 포함할 수 있습니다. 모든 객체에 적용되는 필터의 이름은 `EntireBucket`을 사용하는 것이 좋습니다.
9. 필터 범위(Filter scope)에서 이 필터는 버킷의 모든 객체에 적용됨(This filter applies to all objects in the bucket)을 선택합니다.

   지표가 버킷의 객체 하위 세트에 대해서만 수집 및 보고되도록 필터를 정의할 수도 있습니다. 자세한 정보는 접두사, 객체 태그 또는 액세스 포인트로 필터링하는 지표 구성 생성을 참조하세요.
10. 변경 사항 저장을 선택합니다.
11. 요청 지표 탭의 필터에서 방금 생성한 필터를 선택합니다.

    약 15분 후 CloudWatch에서 이러한 요청 지표를 추적하기 시작합니다. 요청 지표 탭에서 지표를 확인할 수 있습니다. 지표에 대한 그래프는 Amazon S3 또는 CloudWatch 콘솔에서 볼 수 있습니다. 요청 지표는 표준 CloudWatch 요금으로 청구됩니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

REST API 사용

Amazon S3 REST API를 이용한 프로그래밍 방법으로 지표 구성을 추가할 수도 있습니다. 지표 구성 추가 및 작업에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 다음 주제를 참조하십시오.

- 버킷 지표 구성 적용
- 버킷 지표 구성 가져오기
- 버킷 지표 구성 목록 조회

- 버킷 지표 구성 삭제

AWS CLI 사용

1. AWS CLI를 설치 및 설정합니다. 자세한 내용은 AWS Command Line Interface 사용 설명서에서 AWS CLI 설치, 업데이트 및 제거를 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"}'
```

접두사, 객체 태그 또는 액세스 포인트로 필터링하는 지표 구성 생성

Amazon S3용 Amazon CloudWatch 지표는 크게 세 가지 유형 즉, 스토리지 지표, 요청 지표 및 복제 지표로 구분됩니다. 스토리지 지표는 하루에 한 번 보고되고 추가 비용 없이 모든 고객에게 제공됩니다. 요청 지표는 약간의 처리 지연 시간 후에 1분 간격으로 제공됩니다. 요청 지표는 표준 CloudWatch 요금으로 청구됩니다. 콘솔에서 또는 Amazon S3 API를 사용해 요청 지표를 구성하여 이 지표를 선택해야 합니다. S3 복제 지표는 복제 구성의 복제 규칙에 대한 세부 지표를 제공합니다. 복제 지표를 사용하면 보류 중인 바이트, 보류 중인 작업, 복제 실패 작업, 복제 지연 시간을 추적하여 진행률을 분 단위로 모니터링할 수 있습니다.

Amazon S3용 CloudWatch 지표에 대한 자세한 내용은 Amazon CloudWatch를 사용한 지표 모니터링 섹션을 참조하십시오.

CloudWatch 지표를 구성하는 경우 버킷의 모든 객체에 대한 필터를 생성하거나 구성을 단일 버킷 내의 관련 객체 그룹으로 필터링할 수 있습니다. 다음 필터 유형 중 하나 이상에 기초하여 버킷 내의 객체들이 한 지표 구성에 포함되도록 필터링할 수 있습니다.

- 객체 키 이름 접두사 - Amazon S3 데이터 모델은 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. 접두사에 대한 자세한 정보는 접두어를 사용한 객체 구성 섹션을 참조하십시오.
- 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. 객체 태그에 대한 자세한 내용은 태그를 사용하여 스토리지 분류 섹션을 참조하십시오.

- 액세스 포인트 - S3 액세스 포인트는 버킷에 연결되어 있는 명명된 네트워크 엔드포인트이며, S3의 공유 데이터 집합에 대한 대규모 데이터 액세스 관리를 간소화합니다. 액세스 포인트 필터를 생성할 때 Amazon S3는 지표 구성에서 지정한 액세스 포인트에 대한 요청을 포함합니다. 자세한 정보는 액세스 포인트 모니터링 및 로깅을 참조하세요.

> **Note**
>
> 액세스 포인트별로 필터링하는 지표 구성을 생성할 때는 액세스 포인트 별칭이 아닌 액세스 포인트 Amazon 리소스 이름(ARN)을 사용해야 합니다. 특정 객체에 대한 ARN이 아닌 액세스 포인트 자체에 ARN을 사용해야 합니다. 액세스 포인트 ARN에 대한 자세한 내용은 범용 버킷의 Amazon S3 Access Points 사용 단원을 참조하십시오.

필터를 지정하는 경우, 단일 객체에서 작동하는 요청만 해당 필터와 일치하므로 보고된 지표에 그 요청만 포함됩니다. DeleteObjects 및 `ListObjects` 요청과 같은 요청은 필터가 있는 구성에 대한 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.

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 요청 지표가 필요한 객체가 있는 버킷 이름을 선택합니다.
4. 지표 탭을 선택합니다.
5. 버킷 지표(Bucket metrics)에서 추가 차트 보기(Bucket metrics)를 선택합니다.
6. 요청 지표 탭을 선택합니다.
7. 필터 생성을 선택합니다.
8. 필터 이름 상자에 필터 이름을 입력합니다.

   이름은 문자, 숫자, 마침표, 대시 및 밑줄만 포함할 수 있습니다.
9. 필터 범위(Filter scope)에서, 접두사, 객체 태그 및 S3 액세스 포인트 또는 세 가지 조합을 사용하여 이 필터 범위 제한(Limit the scope of this filter using a prefix, object tags, and an S3 Access Point, or a combination of all three)을 선택합니다.

10. 필터 유형(Filter type)에서, 필터 유형(접두사, 객체 태그 또는 액세스 포인트)을 하나 이상 선택합니다.
11. 접두사 필터를 정의하고 필터 범위를 단일 경로로 제한하려면 접두사(Prefix) 상자에 접두사를 입력합니다.
12. 객체 태그 필터를 정의하려면 객체 태그(Object tags)에서태그 추가(Add tag)를 선택한 후 태그 키(Key) 및 값(Value)을 입력합니다.
13. 액세스 포인트 필터를 정의하려면 S3 액세스 포인트(S3 Access Point) 필드에서 액세스 포인트 ARN 입력하거나 S3 찾아보기(Browse S3)를 선택하여 액세스 포인트로 이동합니다.

    > **Important**
    >
    > 액세스 포인트 별칭을 입력할 수 없습니다. 특정 객체에 대한 ARN이 아닌 액세스 포인트 자체에 ARN을 입력해야 합니다.

14. 변경 사항 저장을 선택합니다.

    Amazon S3는 사용자가 지정한 접두사, 태그 또는 액세스 포인트를 사용하는 필터를 생성합니다.
15. 요청 지표 탭의 필터에서 방금 생성한 필터를 선택합니다.

    이제 접두사, 객체 태그 또는 액세스 포인트로 요청 지표 범위를 제한하는 필터를 만들었습니다. CloudWatch가 이러한 요청 지표를 추적하기 시작한 지 약 15분이 지나면 Amazon S3와 CloudWatch 콘솔에서 해당 지표의 차트를 확인할 수 있습니다. 요청 지표는 표준 CloudWatch 요금으로 청구됩니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

    버킷 수준에서도 요청 지표를 구성할 수 있습니다. 자세한 내용은 버킷의 모든 객체에 대한 CloudWatch 지표 구성 생성을 참조하십시오.

AWS CLI 사용

1. AWS CLI를 설치 및 설정합니다. 자세한 내용은 AWS Command Line Interface 사용 설명서에서 AWS CLI 설치, 업데이트 및 제거를 참조하십시오.
2. 터미널을 엽니다.
3. 지표 구성을 추가하려면 다음 명령 중 하나를 실행하십시오.

Example : 접두사로 필터링

```
aws s3api put-bucket-metrics-configuration --bucket amzn-s3-demo-bucket --
id metrics-config-id --metrics-configuration '{"Id":"metrics-config-id", "Filter":
{"Prefix":"prefix1"}} '
```

Example : 태그로 필터링

```
aws s3api put-bucket-metrics-configuration --bucket amzn-s3-demo-bucket --
id metrics-config-id --metrics-configuration '{"Id":"metrics-config-id", "Filter":
{"Tag": {"Key": "string", "Value": "string"}} '
```

Example : 액세스 포인트로 필터링

```
aws s3api put-bucket-metrics-configuration --bucket amzn-s3-demo-bucket --
id metrics-config-id --metrics-configuration '{"Id":"metrics-config-id", "Filter":
{"AccessPointArn":"arn:aws:s3:Region:account-id:accesspoint/access-point-name"}} '
```

Example : 접두사, 태그 및 액세스 포인트로 필터링

```
aws s3api put-bucket-metrics-configuration --endpoint https://
s3.Region.amazonaws.com --bucket amzn-s3-demo-bucket --id metrics-config-id --
metrics-configuration '
{
    "Id": "metrics-config-id",
    "Filter": {
        "And": {
            "Prefix": "string",
            "Tags": [
                {
                    "Key": "string",
                    "Value": "string"
                }
            ],
            "AccessPointArn": "arn:aws:s3:Region:account-id:accesspoint/access-
point-name"
        }
    }
}'
```

REST API 사용

Amazon S3 REST API를 이용한 프로그래밍 방법으로 지표 구성을 추가할 수도 있습니다. 지표 구성 추가 및 작업에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 다음 주제를 참조하십시오.

- 버킷 지표 구성 적용
- 버킷 지표 구성 가져오기
- 버킷 지표 구성 목록 조회
- 버킷 지표 구성 삭제

지표 필터 삭제

더 이상 필요하지 않은 경우 Amazon CloudWatch 요청 지표 필터를 삭제할 수 있습니다. 필터를 삭제하면 해당 특정 필터를 사용하는 요청 지표에 대해 더 이상 요금이 청구되지 않습니다. 그러나 존재하는 다른 필터 구성에 대해서는 계속 요금이 청구됩니다.

필터를 삭제하면 더 이상 요청 지표에 해당 필터를 사용할 수 없습니다. 필터 삭제는 실행 취소할 수 없습니다.

요청 지표 필터 생성에 대한 자세한 내용은 다음 주제를 참조하십시오.

- 버킷의 모든 객체에 대한 CloudWatch 지표 구성 생성
- 접두사, 객체 태그 또는 액세스 포인트로 필터링하는 지표 구성 생성

S3 콘솔 사용

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 요청 지표 필터를 삭제하려는 객체가 있는 버킷 이름을 선택합니다.
4. 지표 탭을 선택합니다.
5. 버킷 지표(Bucket metrics)에서 추가 차트 보기(Bucket metrics)를 선택합니다.
6. 요청 지표 탭을 선택합니다.
7. 필터 관리(Manage filters)를 선택합니다.
8. 필터를 선택합니다.

> **Important**
>
> 필터 삭제는 실행 취소할 수 없습니다.

9. 삭제를 선택합니다.

   Amazon S3가 필터를 삭제합니다.

REST API 사용

Amazon S3 REST API를 이용한 프로그래밍 방법으로 지표 구성을 추가할 수도 있습니다. 지표 구성 추가 및 작업에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 다음 주제를 참조하십시오.

- 버킷 지표 구성 적용
- 버킷 지표 구성 가져오기
- 버킷 지표 구성 목록 조회
- 버킷 지표 구성 삭제

Amazon S3 이벤트 알림

Amazon S3 이벤트 알림 기능을 사용하면 S3 버킷에서 특정 이벤트가 발생할 때 알림을 받을 수 있습니다. 알림을 사용 설정하려면 Amazon S3에서 게시하려는 이벤트를 식별하는 알림 구성을 추가합니다. 또한 해당 알림 구성이 Amazon S3에서 알림을 보낼 대상도 식별하는지 확인합니다. 버킷에 연결된 알림 하위 리소스에 이 구성을 저장합니다. 자세한 내용은 범용 버킷 구성 옵션 섹션을 참조하세요. Amazon S3는 이 하위 리소스를 관리하기 위한 API를 제공합니다.

> **Important**
>
> Amazon S3 이벤트 알림은 한 번 이상 전송되도록 설계되었습니다. 이벤트 알림은 일반적으로 몇 초 안에 전송되지만 1분 이상 소요되는 경우도 있습니다.

Amazon S3 이벤트 알림 개요

현재 Amazon S3은 다음 이벤트에 대한 알림을 게시할 수 있습니다.

- 새 객체 생성 이벤트
- 객체 제거 이벤트
- 객체 이벤트 복원
- RRS(Reduced Redundancy Storage) 객체 손실 이벤트
- 복제 이벤트
- S3 수명 주기 만료 이벤트
- S3 수명 주기 전환 이벤트
- S3 Intelligent-Tiering 자동 아카이브 이벤트
- 객체 태깅 이벤트
- 객체 ACL PUT 이벤트

지원되는 모든 이벤트 유형에 대한 자세한 설명은 SQS, SNS 및 Lambda에 지원되는 이벤트 유형 섹션을 참조하십시오.

Amazon S3은 다음과 같은 대상으로 이벤트 알림 메시지를 보낼 수 있습니다. 알림 구성에서 이 대상의 Amazon 리소스 이름(ARN) 값을 지정합니다.

- Amazon Simple Notification Service(Amazon SNS) 주제
- Amazon Simple Queue Service(Amazon SQS) 대기열
- AWS Lambda 함수
- Amazon EventBridge

자세한 내용은 지원되는 이벤트 대상 섹션을 참조하세요.

> **Note**
>
> Amazon Simple Queue Service FIFO(선입선출) 대기열은 Amazon S3 이벤트 알림 대상으로 지원되지 않습니다. Amazon S3 이벤트에 대한 알림을 Amazon SQS FIFO 대기열로 보내려면 Amazon EventBridge를 사용하면 됩니다. 자세한 내용은 Amazon EventBridge 사용 설정 섹션을 참조하세요.

> **Warning**
>
> 알림을 트리거하는 동일한 버킷에 알림이 기록되면 실행 루프가 발생할 수 있습니다. 예를 들어 객체가 업로드될 때마다 버킷이 Lambda 함수를 트리거하고 그 함수가 객체를 버킷에 업로드하는 경우, 함수는 간접적으로 자체 트리거됩니다 이렇게 되지 않도록 하려면 두 개의 버킷을 사용하거나, 수신 객체에 사용되는 접두사에만 적용되도록 트리거를 구성합니다.
> AWS Lambda에서 Amazon S3 알림을 사용하는 방법에 대한 자세한 내용과 예제는 AWS Lambda 개발자 안내서의 Amazon S3에서 AWS Lambda 사용을 참조하십시오.

버킷별로 생성할 수 있는 이벤트 알림 구성 수에 대한 자세한 내용은 AWS 일반 참조의 Amazon S3 서비스 할당량을 참조하십시오.

이벤트 알림에 대한 자세한 내용은 다음 섹션을 참조하십시오.

주제

- 이벤트 알림 유형 및 대상
- Amazon SQS, Amazon SNS 및 Lambda 사용
- EventBridge 사용

이벤트 알림 유형 및 대상

Amazon S3는 알림을 게시할 수 있는 여러 이벤트 알림 유형 및 대상을 지원합니다. 이벤트 알림을 구성할 때 이벤트 유형 및 대상을 지정할 수 있습니다. 각 이벤트 알림에는 대상을 하나만 지정할 수 있습니다. Amazon S3 이벤트 알림은 각 알림 메시지에 대해 하나의 이벤트 항목을 전송합니다.

주제

- 지원되는 이벤트 대상
- SQS, SNS 및 Lambda에 지원되는 이벤트 유형
- Amazon EventBridge에 지원되는 이벤트 유형
- 이벤트 주문 및 중복 이벤트

지원되는 이벤트 대상

Amazon S3은 다음과 같은 대상으로 이벤트 알림 메시지를 보낼 수 있습니다.

- Amazon Simple Notification Service(Amazon SNS) 주제
- Amazon Simple Queue Service(Amazon SQS) 대기열
- AWS Lambda
- Amazon EventBridge

그러나 각 이벤트 알림에는 대상 유형을 하나만 지정할 수 있습니다.

> **Note**
>
> Amazon SNS 주제 또는 Amazon SQS 대기열에 메시지를 게시할 수 있는 권한을 Amazon S3에 부여해야 합니다. 또한, 사용자를 대신하여 AWS Lambda 함수를 호출할 수 있는 권한을 Amazon S3에 부여해야 합니다. 이러한 권한을 부여하는 방법에 대한 지침은 대상에 이벤트 알림 메시지를 게시할 권한 부여 섹션을 참조하십시오.

Amazon SNS 주제

Amazon SNS는 유연한 완전 관리형 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. 현재 표준 SNS는 S3 이벤트 알림 대상으로만 허용되지만 SNS FIFO는 허용되지 않습니다.

Amazon SNS는 구독 엔드포인트 또는 클라이언트로의 메시지 전송 및 전달을 조정하고 관리합니다. Amazon SNS 콘솔을 사용하여 알림을 수신할 Amazon SNS 주제를 만들 수 있습니다.

주제는 Amazon S3 버킷과 동일한 AWS 리전에 있어야 합니다. Amazon SNS 주제를 생성하는 방법에 대한 지침은 Amazon Simple Notification Service 개발자 안내서의 Amazon SNS 시작하기와 Amazon SNS FAQ를 참조하십시오.

이벤트 알림 대상으로 생성한 Amazon SNS 주제를 사용하려면 다음 항목이 필요합니다.

- Amazon SNS 주제에 대한 Amazon 리소스 이름(ARN)
- 유효한 Amazon SNS 주제 구독입니다. 이를 통해 Amazon SNS 주제에 메시지가 게시되면 해당 주제 구독자에게 알림이 전송됩니다.

Amazon SQS 대기열

Amazon SQS는 컴퓨터 간에 주고 받는 메시지를 저장하기 위한 안정적이고 확장성이 뛰어난 호스팅 대기열을 제공합니다. Amazon SQS를 사용하면 다른 서비스를 항상 가용 상태로 유지하지 않고도 모든 데이터 볼륨을 전송할 수 있습니다. Amazon SQS 콘솔을 사용하여 알림을 수신할 Amazon SQS 대기열을 만들 수 있습니다.

Amazon SQS 대기열은 Amazon S3 버킷과 동일한 AWS 리전에 있어야 합니다. Amazon SQS 대기열을 생성하는 방법에 대한 지침은 Amazon Simple Queue Service 개발자 안내서의 Amazon Simple Queue Service란? 및 Amazon SQS 시작하기를 참조하십시오.

이벤트 알림 대상으로 만든 Amazon SQS 대기열 사용하려면 다음 항목이 필요합니다.

- Amazon SQS 대기열에 대한 Amazon 리소스 이름(ARN)

> **Note**
>
> Amazon Simple Queue Service FIFO(선입선출) 대기열은 Amazon S3 이벤트 알림 대상으로 지원되지 않습니다. Amazon S3 이벤트에 대한 알림을 Amazon SQS FIFO 대기열로 보내려면 Amazon EventBridge를 사용하면 됩니다. 자세한 내용은 Amazon EventBridge 사용 설정 섹션을 참조하세요.

Lambda 함수

AWS Lambda을 사용하여 사용자 지정 로직으로 다른 AWS 서비스를 확장하거나 AWS 규모와 성능, 보안에 따라 작동하는 자체 백엔드를 만들 수 있습니다. Lambda를 사용하면 필요할 때만 실행되는 개별 이벤트 중심 애플리케이션을 생성할 수 있습니다. 또한 이를 통해 하루에 몇 번의 요청에서 초당 수천 개의 요청으로 이러한 애플리케이션을 자동으로 확장할 수 있습니다.

Lambda는 Amazon S3 버킷 이벤트에 대한 응답으로 사용자 지정 코드를 실행할 수 있습니다. 사용자 정의 코드를 Lambda에 업로드하여 Lambda 함수라는 것을 생성합니다. Amazon S3가 특정 유형의 이벤트를 감지하면 AWS Lambda에 이벤트를 게시하고 Lambda에서 함수를 호출할 수 있습니다. 이에 대한 응답으로 Lambda는 함수를 실행합니다. 예를 들어 이러한 애플리케이션에서 탐지할 수 있는 이벤트 유형 중 하나는 객체 생성 이벤트입니다.

AWS Lambda 콘솔에서 AWS 인프라를 사용하여 사용자 대신해 코드를 실행하는 Lambda 함수를 생성할 수 있습니다. Lambda 함수는 S3 버킷과 동일한 리전에 있어야 합니다. Lambda 함수를 이벤트 알림 대상으로 설정하려면 Lambda 함수의 이름이나 ARN도 있어야 합니다.

> **Warning**
>
> 알림을 트리거하는 동일한 버킷에 알림이 기록되면 실행 루프가 발생할 수 있습니다. 예를 들어 객체가 업로드될 때마다 버킷이 Lambda 함수를 트리거하고 그 함수가 객체를 버킷에 업로드하는 경우, 함수는 간접적으로 자체 트리거됩니다 이렇게 되지 않도록 하려면 두 개의 버킷을 사용하거나, 수신 객체에 사용되는 접두사에만 적용되도록 트리거를 구성합니다.
> AWS Lambda에서 Amazon S3 알림을 사용하는 방법에 대한 자세한 내용과 예제는 AWS Lambda 개발자 안내서의 Amazon S3에서 AWS Lambda 사용을 참조하십시오.

Amazon EventBridge

Amazon EventBridge는 AWS 서비스로부터 이벤트를 수신하는 서버리스 이벤트 버스입니다. 이벤트를 일치시키고 AWS 서비스 또는 HTTP 엔드포인트와 같은 대상에 전달하도록 규칙을 설정할 수 있습니다. 자세한 내용은 Amazon EventBridge 사용 설명서의 EventBridge란?을 참조하세요.

다른 대상과 달리 버킷에 대해 EventBridge로 이벤트 전송을 사용 설정하거나 사용 중지할 수 있습니다. 전송을 사용 설정하면 모든 이벤트가 EventBridge로 전송됩니다. 또한 EventBridge 규칙을 사용하여 이벤트를 추가 대상으로 라우팅할 수 있습니다.

SQS, SNS 및 Lambda에 지원되는 이벤트 유형

Amazon S3은 다음과 같은 유형의 이벤트를 게시할 수 있습니다. 알림 구성에 이러한 이벤트 유형을 지정합니다.

이벤트 유형	설명
`s3:TestEvent`	알림이 사용 설정되면 Amazon S3는 테스트 알림을 게시합니다. 이는 주제가 존재하고 버킷 소유자에게 지정된 주제를 게시할 권한이 있는지 확인하기 위한 것입니다. 알림 사용 설정이 실패하면 테스트 알림이 수신되지 않습니다.

`s3:ObjectCreated:*`
 	
`s3:ObjectCreated:Put`
 	
`s3:ObjectCreated:Post`
	PUT, POST 및 COPY와 같은 Amazon S3 API 작업은 객체를 생성할 수 있습니다. 이 이벤트 유형을 사용하면 특정 API 작업을 사용하여 객체가 생성될 때 알림을 사용 설정할 수 있습니다. 또는 `s3:ObjectCreated:*` 이벤트 유형을 사용

이벤트 유형	설명
s3:ObjectCreated:Copy s3:ObjectCreated:CompleteMultipartUpload	하여 객체를 생성하는 데 사용된 API와 관계없이 알림을 요청할 수 있습니다. s3:ObjectCreated:CompleteMultipartUpload 에는 복사 작업을 위해 UploadPartCopy를 사용하여 생성된 객체가 포함되어 있습니다.
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:DeleteMarkerCreated 를 사용하여 버전이 지정된 객체에 대해 삭제 마커가 생성될 경우 알림을 요청할 수 있습니다. 버전이 지정된 객체를 삭제하는 방법에 대한 지침은 버전 관리가 사용 설정된 버킷에서 객체 버전 삭제 섹션을 참조하십시오. s3:ObjectRemoved:* 와일드카드를 사용하여 객체가 삭제되는 모든 경우에 알림을 요청할 수도 있습니다. 이러한 이벤트 알림은 수명 주기 구성 또는 실패한 작업의 자동 삭제에 대해 경고하지 않습니다.

이벤트 유형	설명
s3:ObjectRestore:* s3:ObjectRestore:Post s3:ObjectRestore:Completed s3:ObjectRestore:Delete	ObjectRestore 이벤트 유형을 사용하여 S3 Glacier Flexible Retrieval 스토리지 클래스, S3 Glacier Deep Archive 스토리지 클래스, S3 Intelligent Tiering Archive Access 계층, S3 Intelligent-Tiering Deep Archive Access 계층에서 객체를 복원할 때 이벤트 초기화 및 완료 알림을 받을 수 있습니다. 객체의 복원된 복사본이 만료될 때 알림을 받을 수도 있습니다. s3:ObjectRestore:Post 이벤트 유형은 객체 복원 시작을 알립니다. s3:ObjectRestore:Completed 이벤트 유형은 복원 완료를 알립니다. s3:ObjectRestore:Delete 이벤트 유형은 복원된 객체의 임시 복사본이 만료될 때 알려줍니다.
s3:ReducedRedundancyLostObject	Amazon S3가 RRS 스토리지 클래스의 객체가 손실되었음을 감지하면 이 알림 이벤트를 수신합니다.

이벤트 유형	설명

`s3:Replication:*`
 	
`s3:Replication:OperationFailedReplication`
 	
`s3:Replication:OperationMissedThreshold`
 	
`s3:Replication:OperationReplicatedAfterThreshold`
 	
`s3:Replication:OperationNotTracked`
	`Replication` 이벤트 유형을 사용하여 S3 복제 지표 또는 S3 Replication Time Control(S3 RTC)이 활성화된 복제 구성에 대해 알림을 받을 수 있습니다. 보류 중인 바이트, 보류 중인 작업, 복제 대기 시간을 추적하여 복제 이벤트의 진행률을 분 단위로 모니터링할 수 있습니다. 복제 지표에 대한 자세한 내용은 지표, 이벤트 알림 및 상태를 통한 복제 모니터링 단원을 참조하세요.
	
	• `s3:Replication:OperationFailedReplication` 이벤트 유형은 복제에 적합한 객체가 복제에 실패할 때 알려줍니다.
	• `s3:Replication:OperationMissedThreshold` 이벤트 유형은 S3 RTC를 사용하는 복제 대상 객체가 복제 임계값인 15분을 초과하면 알려줍니다.
	• `s3:Replication:OperationReplicatedAfterThreshold` 이벤트 유형은 S3 RTC를 사용하는 복제 대상 객체가 15분 임계값 이후에 복제되면 알려줍니다.
	• `s3:Replication:OperationNotTracked` 이벤트 유형은 라이브 복제 대상 객체(동일 리전 복제[SRR] 또는 교차 리전 복제[CRR])가 더 이상 복제 지표로 추적되지 않으면 알려줍니다.

`s3:LifecycleExpiration:*`
 	
`s3:LifecycleExpiration:Delete`
 	
`s3:LifecycleExpiration:DeleteMarkerCreated`
	`LifecycleExpiration` 이벤트 유형을 사용하면 Amazon S3가 S3 수명 주기 구성에 따라 객체를 삭제할 때 알림을 받을 수 있습니다.
	
	`s3:LifecycleExpiration:Delete` 이벤트 유형은 버전이 지정되지 않은 버킷의 객체가 삭제될 때 알려줍니다. S3 수명 주기 구성에 의해 객체 버전이 영구적으로 삭제될 때도 알려줍니다. `s3:LifecycleExpiration:DeleteMarkerCreated` 이벤트 유형은 버전이 지정된 버킷에 있는 객체의 현재 버전이 삭제되는 경우 S3 수명 주기가 삭제 마커를 생성할 때 알려줍니다.

이벤트 유형	설명
`s3:LifecycleTransition`	S3 수명 주기 구성에 의해 객체가 다른 Amazon S3 스토리지 클래스로 전환될 때 이 알림 이벤트를 수신합니다.
`s3:IntelligentTiering`	S3 Intelligent-Tiering 스토리지 클래스 내의 객체가 Archive Access 계층 또는 Deep Archive Access 계층으로 이동할 때 이 알림 이벤트를 수신합니다.

`s3:ObjectTagging:*`
 	
`s3:ObjectTagging:Put`
 	
`s3:ObjectTagging:Delete`
	`ObjectTagging` 이벤트 유형을 사용하면 객체 태그가 객체에서 추가 또는 삭제될 때 알림을 활성화할 수 있습니다.
	
	`s3:ObjectTagging:Put` 이벤트 유형은 태그가 객체에 PUT되거나 기존 태그가 업데이트될 때 알려줍니다.
	`s3:ObjectTagging:Delete` 이벤트 유형은 객체에서 태그가 제거될 때 알려줍니다.

`s3:ObjectAcl:Put`	ACL이 객체에 PUT되거나 기존 ACL이 변경될 때 이 알림 이벤트를 수신합니다. 요청으로 인해 객체의 ACL이 변경되지 않으면 이벤트가 생성되지 않습니다.

Amazon EventBridge에 지원되는 이벤트 유형

Amazon S3가 Amazon EventBridge로 전송할 이벤트 유형 목록은 EventBridge 사용 섹션을 참조하십시오.

이벤트 주문 및 중복 이벤트

Amazon S3 이벤트 알림은 알림을 한 번 이상 전송하도록 설계되었지만 이벤트가 발생한 순서대로 도착하지 않을 수 있습니다. 드문 경우이긴 하지만 Amazon S3의 재시도 메커니즘으로 인해 동일한 객체 이벤트에 대해 중복된 S3 이벤트 알림이 발생할 수 있습니다. 중복 또는 비순차적 이벤트의 처리에 대한 자세한 내용은 AWS 스토리지 블로그의 Amazon S3 이벤트 알림을 사용하여 이벤트 주문 및 중복 이벤트 관리를 참조하세요.

Amazon SQS, Amazon SNS 및 Lambda 사용

알림 사용 설정은 버킷 수준 작업입니다. 알림 구성 정보는 버킷에 연결된 알림 하위 리소스에 저장됩니다. 버킷 알림 구성을 생성하거나 변경한 후 변경 사항이 적용되려면 일반적으로 5분 정도 걸립니다.

알림이 처음으로 사용 설정될 때 `s3:TestEvent`가 발생합니다. 다음과 같은 메서드를 사용하여 알림 구성을 관리할 수 있습니다.

- Amazon S3 콘솔 사용 - 콘솔 UI를 사용하면 코드를 작성할 필요 없이 버킷에서 알림 구성을 설정할 수 있습니다. 자세한 내용은 Amazon S3 콘솔을 사용하여 이벤트 알림 사용 설정 및 구성 섹션을 참조하세요.
- AWS SDK를 사용한 프로그래밍 방식 - 내부적으로는 콘솔이나 SDK 모두 Amazon S3 REST API를 호출하여 버킷과 연결된 알림 하위 리소스를 관리합니다. AWS SDK를 사용하는 알림 구성의 예는 연습: 알림용 버킷 구성(SNS 주제 또는 SQS 대기열) 섹션을 참조하십시오.

> **Note**
>
> 코드에서 직접 Amazon S3 REST API를 호출할 수도 있습니다. 그러나 이렇게 하려면 요청 인증을 위한 코드를 작성해야 하므로 번거로울 수 있습니다.

사용하는 방법에 관계없이 Amazon S3가 버킷과 연결된 알림 하위 리소스에 알림 구성을 XML로 저장합니다. 버킷 하위 리소스에 대한 자세한 내용은 범용 버킷 구성 옵션 단원을 참조하십시오.

> **Note**
>
> 삭제된 대상으로 인해 실패한 이벤트 알림이 여러 개 있는 경우 이를 삭제하려고 하면 다음 대상 구성을 검증할 수 없음이라는 메시지가 표시될 수 있습니다. S3 콘솔에서 모든 실패한 알림을 동시에 삭제하여 이 문제를 해결할 수 있습니다.

주제

- 대상에 이벤트 알림 메시지를 게시할 권한 부여
- Amazon S3 콘솔을 사용하여 이벤트 알림 사용 설정 및 구성
- 프로그래밍 방식으로 이벤트 알림 구성
- 연습: 알림용 버킷 구성(SNS 주제 또는 SQS 대기열)
- 객체 키 이름 필터링을 사용하여 이벤트 알림 구성
- 이벤트 메시지 구조

대상에 이벤트 알림 메시지를 게시할 권한 부여

관련 API를 호출하여 SNS 주제, SQS 대기열 또는 Lambda 함수에 메시지를 게시할 수 있는 권한을 Amazon S3 보안 주체에 부여해야 합니다. 이는 Amazon S3가 이벤트 알림 메시지를 대상에 게시할 수 있도록 하기 위한 것입니다.

대상에 이벤트 알림 메시지를 게시하는 문제를 해결하려면 Amazon Simple Notification Service 주제에 Amazon S3 이벤트 알림을 게시할 때의 문제 해결을 참조하십시오.

주제

- AWS Lambda 함수 호출 권한 부여
- SNS 주제 또는 SQS 대기열로 메시지를 게시할 권한 부여

AWS Lambda 함수 호출 권한 부여

Amazon S3는 Lambda 함수를 호출하고 이벤트 메시지를 인수로 제공하여 AWS Lambda에 이벤트 메시지를 게시합니다.

Amazon S3 콘솔을 사용하여 Amazon S3 버킷에서 Lambda 함수에 대한 이벤트 알림을 구성할 때 콘솔은 Lambda 함수에 대해 필요한 권한을 설정합니다. 이는 Amazon S3가 버킷에서 함수를 호출할 수 있는 권한을 갖도록 하기 위한 것입니다. 자세한 내용은 Amazon S3 콘솔을 사용하여 이벤트 알림 사용 설정 및 구성 섹션을 참조하세요.

또한, AWS Lambda에서 Lambda 함수를 호출할 수 있는 권한을 Amazon S3에 부여할 수도 있습니다. 자세한 내용은 AWS Lambda 개발자 안내서의 자습서: Amazon S3에서 AWS Lambda 사용을 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 연습: 알림용 버킷 구성(SNS 주제 또는 SQS 대기열) 단원을 참조하십시오. 권한에 대한 자세한 내용은 다음 섹션을 참조하십시오.

- Amazon Simple Notification Service 개발자 안내서의 Amazon SNS 액세스 제어 사례
- Amazon Simple Queue Service 개발자 안내서의 Amazon SQS의 Identity and Access Management

대상 SNS 주제에 대한 IAM 정책

다음은 대상 SNS 주제에 연결하는 AWS Identity and Access Management(IAM) 정책의 예제입니다. 이 정책을 사용하여 이벤트 알림에 대한 대상 Amazon SNS 주제를 설정하는 방법에 대한 지침은 연습: 알림용 버킷 구성(SNS 주제 또는 SQS 대기열) 섹션을 참조하십시오.

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "Example SNS topic policy",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SNS:Publish"
            ],
            "Resource": "SNS-topic-ARN",
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:s3:::amzn-s3-demo-bucket"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-account-id"
                }
            }
        }
    ]
}
```

대상 SQS 대기열에 대한 IAM 정책

다음은 대상 SQS 대기열에 연결하는 IAM 정책의 예제입니다. 이 정책을 사용하여 이벤트 알림에 대한 대상 Amazon SQS 대기열을 설정하는 방법에 대한 지침은 연습: 알림용 버킷 구성(SNS 주제 또는 SQS 대기열) 섹션을 참조하십시오.

이 정책을 사용하려면 Amazon SQS 대기열 ARN, 버킷 이름 및 버킷 소유자의 AWS 계정 ID를 업데이트해야 합니다.

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

```
    "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:queue-name",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:awsexamplebucket1"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-account-id"
                }
            }
        }
    ]
}
```

Amazon SNS 및 Amazon SQS IAM 정책 모두 `StringLike` 조건 대신, `ArnLike` 조건을 정책에 지정할 수 있습니다.

`ArnLike`를 사용하는 경우 ARN의 파티션, 서비스, 계정 ID, 리소스 유형 및 리소스 ID 부분이 요청 컨텍스트의 ARN과 정확히 일치해야 합니다. 리전 및 리소스 경로만 부분 매칭이 허용됩니다.

`ArnLike` 대신 `StringLike`를 사용하는 경우 매칭에서는 ARN 구조를 무시하고 와일드카드가 적용된 부분에 관계없이 부분 일치를 허용합니다. 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소를 참조하십시오.

```
"Condition": {
  "StringLike": { "aws:SourceArn": "arn:aws:s3:*:*:amzn-s3-demo-bucket" }
  }
```

AWS KMS 키 정책

SQS 대기열 또는 SNS 주제가 AWS Key Management Service(AWS KMS) 고객 관리형 키로 암호화되는 경우 Amazon S3 서비스 보안 주체에게 암호화된 주제 또는 대기열을 사용할 수 있는 권한을 부

여해야 합니다. Amazon S3 서비스 보안 주체에게 권한을 부여하려면 고객 관리형 키의 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 개발자 안내서에서 AWS KMS에서 키 정책 사용을 참조하십시오.

Amazon SQS 및 Amazon SNS에 대해 AWS KMS의 서버 측 암호화를 사용하는 방법에 대한 자세한 내용은 다음을 참조하십시오.

- Amazon Simple Notification Service 개발자 안내서의 키 관리
- Amazon Simple Queue Service 개발자 안내서의 키 관리
- AWS 컴퓨팅 블로그의 AWS KMS를 활용하여 Amazon SNS에 게시된 메시지 암호화

Amazon S3 콘솔을 사용하여 이벤트 알림 사용 설정 및 구성

특정 Amazon S3 범용 버킷 이벤트를 사용 설정해 이벤트가 발생할 때마다 대상에 알림을 보낼 수 있습니다. 이 섹션에서는 Amazon S3 콘솔로 이벤트 알림을 사용 설정하는 방법을 살펴봅니다. AWS SDK 및 Amazon S3 REST API를 통해 이벤트 알림을 사용하는 방법에 대한 자세한 내용은 프로그래밍 방식으로 이벤트 알림 구성 섹션을 참조하십시오.

사전 조건: 버킷에 대한 이벤트 알림을 사용 설정하려면 먼저 대상 유형 중 하나를 설정한 후 권한을 구성해야 합니다. 자세한 내용은 지원되는 이벤트 대상 및 대상에 이벤트 알림 메시지를 게시할 권한 부여(을)를 참조하세요.

> **Note**
>
> Amazon Simple Queue Service FIFO(선입선출) 대기열은 Amazon S3 이벤트 알림 대상으로 지원되지 않습니다. Amazon S3 이벤트에 대한 알림을 Amazon SQS FIFO 대기열로 보내려면 Amazon EventBridge를 사용하면 됩니다. 자세한 내용은 Amazon EventBridge 사용 설정 섹션을 참조하세요.

주제

- Amazon S3 콘솔로 Amazon SNS, Amazon SQS 또는 Lambda 알림 사용 설정

Amazon S3 콘솔로 Amazon SNS, Amazon SQS 또는 Lambda 알림 사용 설정

S3 버킷에 대한 이벤트 알림 사용 설정 및 구성

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 이벤트를 활성화하려는 버킷의 이름을 선택합니다.
4. [속성(Properties)]을 선택합니다.
5. 이벤트 알림(Event Notifications) 섹션으로 이동하여 이벤트 알림 생성(Create event notification)을 선택합니다.
6. 일반 구성(General configuration) 섹션에서 이벤트 알림을 설명하는 이벤트 이름을 지정합니다. 선택적으로 접두사와 접미사를 지정하여 지정된 문자로 끝나는 키가 있는 객체로 알림을 제한할 수도 있습니다.

   a. 이벤트 이름(Event name)에 대한 설명을 입력합니다.

      이름을 입력하지 않으면 전역 고유 식별자(GUID)가 생성되어 이름에 사용됩니다.

   b. (선택 사항) 접두사를 기준으로 이벤트 알림을 필터링하려면 접두사(Prefix)를 입력합니다.

      예를 들어 특정 폴더에 파일이 추가될 때만 알림을 받도록 접두사 필터를 설정할 수 있습니다(예: `images/`).

c. (선택 사항) 접미사를 기준으로 이벤트 알림을 필터링하려면 접미사(Suffix)를 입력합니다.

자세한 내용은 객체 키 이름 필터링을 사용하여 이벤트 알림 구성 섹션을 참조하세요.

7. 이벤트 유형(Event types) 섹션에서 알림을 받을 이벤트 유형을 하나 이상 선택합니다.

다양한 이벤트 유형 목록은 SQS, SNS 및 Lambda에 지원되는 이벤트 유형 섹션을 참조하십시오.

8. 대상(Destination) 섹션에서 이벤트 알림 대상을 선택합니다.

> **Note**
>
> 이벤트 알림을 게시하려면 관련 API를 호출하는 데 필요한 권한을 Amazon S3 보안 주체에 부여해야 합니다. 이는 Lambda 함수, SNS 주제 또는 SQS 대기열에 알림을 게시할 수 있도록 하기 위한 것입니다.

a. Lambda 함수, SNS 주제 또는 SQS 대기열과 같은 대상 유형을 선택합니다.

b. 대상 유형을 선택한 후 목록에서 함수, 주제 또는 대기열을 선택합니다.

c. 또는 Amazon 리소스 이름(ARN)을 지정하려는 경우 [ARN 입력(Enter ARN)]을 선택하고 ARN을 입력합니다.

자세한 내용은 지원되는 이벤트 대상 섹션을 참조하세요.

9. [변경 사항 저장(Save changes)]을 선택하면 Amazon S3가 이벤트 알림 대상으로 테스트 메시지를 보냅니다.

프로그래밍 방식으로 이벤트 알림 구성

기본적으로 모든 유형의 이벤트에 대해 알림이 사용 중지되어 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 S3이 게시할 이벤트 유형 및 이벤트 게시할 대상을 지정하는 구성으로 이 XML을 교체합니다. 각 대상에 대해 해당 XML 구성을 추가해야 합니다.

SQS 대기열에 이벤트 메시지 게시

SQS 대기열을 하나 이상의 이벤트 유형에 대한 알림 대상으로 설정하려면 QueueConfiguration을 추가합니다.

```
<NotificationConfiguration>
  <QueueConfiguration>
    <Id>optional-id-string</Id>
    <Queue>sqs-queue-arn</Queue>
    <Event>event-type</Event>
    <Event>event-type</Event>
     ...
  </QueueConfiguration>
   ...
</NotificationConfiguration>
```

SNS 주제에 이벤트 메시지 게시

SNS 주제를 특정 이벤트 유형의 알림 대상으로 설정하려면 TopicConfiguration을 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 인수로 이벤트 메시지 제공

Lambda 함수를 특정 이벤트 유형에 대한 알림 대상으로 설정하려면 CloudFunctionConfiguration을 추가합니다.

```
<NotificationConfiguration>
  <CloudFunctionConfiguration>
     <Id>optional-id-string</Id>
     <CloudFunction>cloud-function-arn</CloudFunction>
     <Event>event-type</Event>
     <Event>event-type</Event>
      ...
  </CloudFunctionConfiguration>
```

```
   ...
</NotificationConfiguration>
```

버킷에 구성된 모든 알림 제거

버킷에 구성된 모든 알림을 제거하려면 알림 하위 리소스에 빈 `<NotificationConfiguration/>` 요소를 저장합니다.

Amazon S3이 특정 유형의 이벤트를 감지하면 이벤트 정보와 함께 메시지를 게시합니다. 자세한 내용은 이벤트 메시지 구조 섹션을 참조하세요.

이벤트 알림 구성에 대한 자세한 내용은 다음 주제를 참조하십시오.

- 연습: 알림용 버킷 구성(SNS 주제 또는 SQS 대기열).
- 객체 키 이름 필터링을 사용하여 이벤트 알림 구성

연습: 알림용 버킷 구성(SNS 주제 또는 SQS 대기열)

Amazon Simple Notification Service(Amazon SNS) 또는 Amazon Simple Queue Service(Amazon SQS)를 통해 Amazon S3 알림을 받을 수 있습니다. 이 연습에서는 Amazon SNS 주제 및 Amazon SQS 대기열을 사용하여 버킷에 알림 구성을 추가합니다.

> **Note**
>
> Amazon Simple Queue Service FIFO(선입선출) 대기열은 Amazon S3 이벤트 알림 대상으로 지원되지 않습니다. Amazon S3 이벤트에 대한 알림을 Amazon SQS FIFO 대기열로 보내려면 Amazon EventBridge를 사용하면 됩니다. 자세한 내용은 Amazon EventBridge 사용 설정 섹션을 참조하세요.

주제

- 연습 요약
- 1단계: Amazon SQS 대기열 생성
- 2단계: Amazon SNS 주제 생성
- 3단계: 버킷에 알림 구성 추가
- 4단계: 설정 테스트

연습 요약

이 연습은 다음을 수행하는 데 도움이 됩니다.

- Amazon SQS 대기열에 `s3:ObjectCreated:*` 유형의 이벤트를 게시합니다.
- Amazon SNS 주제에 `s3:ReducedRedundancyLostObject` 유형의 이벤트를 게시합니다.

알림 구성에 대한 자세한 내용은 Amazon SQS, Amazon SNS 및 Lambda 사용 단원을 참조하십시오.

콘솔을 사용하면 코드를 생성하지 않고도 이러한 모든 단계를 수행할 수 있습니다. 또한, AWS SDK for Java 및 .NET을 사용하는 코드 예제가 제공되므로 프로그래밍 방식으로 알림 구성을 추가할 수 있습니다.

이 절차에는 다음 단계가 포함됩니다.

1. Amazon SQS 대기열 생성

   Amazon SQS 콘솔을 사용하여 SQS 대기열을 생성합니다. 프로그래밍 방식으로 Amazon S3이 대기열로 전송하는 모든 메시지에 액세스할 수 있습니다. 그러나 이 시연에서는 콘솔에서 알림 메시지를 확인합니다.

   대기열에 액세스 정책을 연결하여 메시지를 게시할 수 있는 권한을 Amazon S3에 부여합니다.
2. Amazon SNS 주제 생성

   Amazon SNS 콘솔을 사용하여 SNS 주제를 생성하고 주제를 구독합니다. 그렇게 하면 게시된 모든 이벤트가 사용자에게 전송됩니다. 통신 프로토콜로 이메일을 지정합니다. 주제를 생성하면 Amazon SNS에서 이메일을 전송합니다. 이메일에 포함된 링크를 사용하여 주제 구독을 확인합니다.

   주제에 액세스 정책을 연결하여 메시지를 게시할 수 있는 권한을 Amazon S3에 부여합니다.
3. 버킷에 알림 구성을 추가합니다.

1단계: Amazon SQS 대기열 생성

단계에 따라 Amazon Simple Queue Service(Amazon SQS) 대기열을 생성하고 구독합니다.

1. Amazon SQS 콘솔을 사용하여 대기열을 생성합니다. 자세한 내용은 Amazon Simple Queue Service 개발자 안내서의 Amazon SQS 시작하기를 참조하십시오.
2. 대기열에 연결된 액세스 정책을 다음 정책으로 바꿉니다.

   a. Amazon SQS 콘솔의 대기열 목록에서 대기열 이름을 선택합니다.

b. 액세스 정책 탭에서 편집을 선택합니다.

c. 대기열에 연결된 액세스 정책을 바꿉니다. 여기에 Amazon SQS ARN, 원본 버킷 이름 및 버킷 소유자 계정 ID를 제공합니다.

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SQS:SendMessage"
            ],
            "Resource": "SQS-queue-ARN",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:awsexamplebucket1"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-account-id"
                }
            }
        }
    ]
}
```

d. 저장을 선택합니다.

3. (선택 사항) AWS Key Management Service(AWS KMS)를 사용하여 Amazon SQS 대기열 또는 Amazon SNS 주제에 서버 측 암호화가 사용되어 있는 경우 연결된 대칭 암호화 고객 관리형 키에 다음 정책을 추가합니다.

   Amazon SQS 또는 Amazon SNS용 AWS 관리형 키는 수정할 수 없으므로 고객 관리형 키에 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 SNS용 SSE를 사용하는 방법에 대한 자세한 내용은 다음을 참조하십시오.

- Amazon Simple Notification Service 개발자 안내서의 키 관리
- Amazon Simple Queue Service 개발자 안내서의 키 관리

4. 대기열 ARN을 기록합니다.

   생성한 SQS 대기열은 AWS 계정에 있는 또 다른 리소스입니다. 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
   ```

2단계: Amazon SNS 주제 생성

다음 단계에 따라 Amazon SNS 주제를 생성하고 구독합니다.

1. Amazon SNS 콘솔을 사용하여 주제를 생성합니다. 자세한 내용은 Amazon Simple Notification Service 개발자 안내서에서 Amazon SNS 주제 생성을 참조하십시오.
2. 주제를 구독합니다. 이 실습에서는 통신 프로토콜로 전자 메일을 사용합니다. 자세한 내용은 Amazon Simple Notification Service 개발자 안내서에서 Amazon SNS 주제 구독을 참조하십시오.

   주제 구독을 확인하도록 요청하는 전자 메일이 전송되면 구독을 확인합니다.

3. 주제에 연결된 액세스 정책을 다음 정책으로 교체합니다. 여기에 SNS 주제 ARN, 버킷 이름 및 버킷 소유자의 계정 ID를 제공합니다.

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "Example SNS topic policy",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SNS:Publish"
            ],
            "Resource": "SNS-topic-ARN",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:amzn-s3-demo-bucket"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-account-id"
                }
            }
        }
    ]
}
```

4. 주제 ARN을 기록합니다.

   생성된 SNS 주제는 AWS 계정의 리소스이며 따라서 고유한 ARN을 가집니다. 다음 단계에서 이 ARN을 사용합니다. ARN의 형식은 다음과 같습니다.

```
arn:aws:sns:aws-region:account-id:topic-name
```

3단계: 버킷에 알림 구성 추가

Amazon S3 콘솔 또는 AWS SDK를 사용한 프로그래밍 방식으로 버킷 알림을 사용할 수 있습니다. 다음 옵션 중 하나를 선택하여 버킷에 알림을 구성합니다. 이 섹션에서는 AWS SDK for Java 및 .NET을 사용한 코드 예제를 제공합니다.

옵션 A: 콘솔을 사용하여 버킷에 대한 알림 사용 설정

Amazon S3 콘솔을 사용하여 다음 작업을 Amazon S3에 요청하는 알림 구성을 추가합니다.

- Amazon SQS 대기열에 모든 객체 생성 이벤트 유형의 이벤트를 게시합니다.
- Amazon SNS 주제에 RRS 객체 손실 유형의 이벤트를 게시합니다.

알림 구성을 저장하면 Amazon S3이 테스트 메시지를 게시하고 이메일을 통해 이 테스트 메시지가 전송됩니다.

지침은 Amazon S3 콘솔을 사용하여 이벤트 알림 사용 설정 및 구성 단원을 참조하십시오.

옵션 B: AWS SDK를 사용하여 버킷에 대한 알림 사용

.NET

다음 C# 코드 예제는 버킷에 알림 구성을 추가하는 전체 코드를 제공합니다. 이 코드를 업데이트하여 버킷 이름과 SNS 주제 ARN을 제공해야 합니다. 코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

```
using Amazon;
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);
            }
        }
    }
}
```

Java

다음 예제에서는 버킷에 알림 구성을 추가하는 방법을 보여줍니다. 실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 AWS SDK for Java 개발자 안내서에서 시작하기 섹션을 참조하세요.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
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 내용은 Amazon Simple Queue Service 개발자 안내서의 '시작하기' 섹션에서 메시지 수신을 참조하십시오.

객체 키 이름 필터링을 사용하여 이벤트 알림 구성

Amazon S3 이벤트 알림을 구성할 때 Amazon S3에서 알림을 전송하도록 하는 지원되는 Amazon S3 이벤트 유형을 지정해야 합니다. 지정하지 않은 이벤트 유형이 S3 버킷에 발생하면 Amazon S3에서 알림을 전송하지 않습니다.

객체 키 이름의 접두사 및 접미사로 알림을 필터링하도록 구성할 수 있습니다. 예를 들어, 파일 이름 확장명이 ".jpg"인 이미지 파일이 버킷에 추가될 경우에만 알림을 받는 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.

> **Note**
>
> 필터에서 접두사 또는 접미사로 와일드카드 문자("*")를 사용할 수 없습니다. 접두사 또는 접미사에 공백이 포함된 경우 공백을 "+" 문자로 대체해야 합니다. 접두사 또는 접미사 값에 다른 특수 문자를 사용하는 경우 URL 인코딩(퍼센트 인코딩) 형식으로 입력해야 합니다. 이벤트 알림의 접두사 또는 접미사에 사용할 때 URL 인코딩 형식으로 변환해야 하는 특수 문자의 전체 목록은 사용 가능 문자 페이지를 참조하십시오.

Amazon S3 콘솔에서 객체 키 이름 필터링을 사용하는 알림 구성을 설정할 수 있습니다. AWS SDK 또는 REST API를 통해 Amazon S3 API를 직접 사용하면 됩니다. 콘솔 UI를 사용하여 버킷에 알림 구성을 설정하는 방법에 대한 자세한 내용은 Amazon S3 콘솔을 사용하여 이벤트 알림 사용 설정 및 구성 섹션을 참조하십시오.

Amazon SQS, Amazon SNS 및 Lambda 사용에서 설명한 대로, Amazon S3은 버킷과 연결된 알림 하위 리소스에 XML로 알림 구성을 저장합니다. 알림의 규칙을 정의하여 객체 키 이름의 접두사 또는 접미사로 알림을 필터링하려면 `Filter` XML 구조를 사용합니다. `Filter` XML 구조에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 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 섹션에는 객체 키 이름 필터링을 사용하는 유효한 알림 구성의 예가 나와 있습니다. 또한 접두사 및 접미사 중첩으로 인해 유효하지 않은 알림 구성 예제가 포함되어 있습니다.

주제

- 객체 키 이름 필터링을 포함하는 유효한 알림 구성 예제
- 유효하지 않은 접두사 및 접미사 중첩을 포함하는 알림 구성 예제

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 정의합니다. .png 및 .jpg 접미사는 마지막 문자가 동일해도 중첩되지 않습니다. 지정된 문자열이 두 접미사로 끝날 수 있으면 2개의 접미사가 겹치는 것으로 간주됩니다. 문자열이 .png와 .jpg로 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>
    <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-A</
CloudFunction>
    <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
  <CloudFunctionConfiguration>
    <Id>2</Id>
    <Filter>
           <S3Key>
               <FilterRule>
                   <Name>suffix</Name>
                   <Value>.png</Value>
               </FilterRule>
           </S3Key>
    </Filter>
    <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-B</
CloudFunction>
    <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
</NotificationConfiguration>
```

Filter를 사용하는 알림 구성은 동일한 이벤트 유형에 대해 중첩 접두사를 포함하는 필터링 규칙을 정의할 수 없습니다. 겹치지 않는 접미사와 함께 사용되는 겹치는 접두사가 있는 경우에만 그렇게 할 수 있습니다. 다음 예제 구성은 일반 접두사를 갖지만 접미사가 중첩되지 않게 생성된 객체가 어떻게 다른 대상으로 전송될 수 있는지를 보여줍니다.

```
<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>
     <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-A</
CloudFunction>
     <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>
     <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-B</
CloudFunction>
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
```

```
</NotificationConfiguration>
```

유효하지 않은 접두사 및 접미사 중첩을 포함하는 알림 구성 예제

Filter를 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 필터링을 사용하여 이벤트 알림 구성 섹션을 참조하십시오

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 ObjectRemoved:* 접두사를 사용하는 알림 구성을 생성할 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 섹션에서는 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 접두사가 중첩되기 때문에 유효하지 않습니다. 특히 루트 접두사는 다른 접두사와 겹칩니다. 이 예제에서 접두사 대신 접미사를 사용하는 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. 지정된 문자열이 두 접미사로 끝날 수 있으면 2개의 접미사가 겹치는 것으로 간주됩니다. 문자열은 jpg와 pg로 끝날 수 있습니다. 따라서 접미사가 겹칩니다. 접두사의 경우에도 마찬가지입니다. 지정된 문자열이 두 접두사로 시작할 수 있으면 2개의 접두사가 겹치는 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>
```

이벤트 메시지 구조

이벤트를 게시하기 위해 Amazon S3이 전송하는 알림 메시지는 JSON 형식입니다.

이벤트 알림 구성에 대한 일반 개요와 지침은 Amazon S3 이벤트 알림 섹션을 참조하십시오.

이 예에서는 이벤트 알림 JSON 구조의 버전 2.2를 보여줍니다. Amazon S3는 이 이벤트 구조의 버전 2.1, 2.2 및 2.3을 사용합니다. Amazon S3는 리전 간 복제 이벤트 알림에 버전 2.2를 사용합니다. S3 수명 주기, S3 Intelligent-Tiering, 객체 ACL, 객체 태깅 및 객체 복원 삭제 이벤트에는 버전 2.3을 사용합니다. 이러한 버전에는 작업과 관련된 추가 정보가 포함되어 있습니다. 버전 2.2 및 2.3은 Amazon S3가 현재 다른 모든 이벤트 알림 유형에 사용하는 버전 2.1과 호환됩니다.

```
{
   "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":"amzn-s3-demo-bucket",
               "ownerIdentity":{
                  "principalId":"Amazon-customer-ID-of-the-bucket-owner"
               },
               "arn":"bucket-ARN"
            },
            "object":{
               "key":"object-key",
               "size":"object-size in bytes",
               "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"
            }
         }
      }
   ]
}
```

이벤트 메시지 구조에 대한 다음 사항에 유의하십시오.

- `eventVersion` 키 값에는 `<major>.<minor>` 형식의 메이저 및 마이너 버전이 포함됩니다.

  Amazon S3가 이전 버전과 호환되지 않는 이벤트 구조를 변경하는 경우 주 버전이 증가합니다. 여기에는 이미 존재하는 JSON 필드를 제거하거나 필드의 콘텐츠가 표시되는 방식(예: 날짜 형식)을 변경하는 것이 포함됩니다.

  Amazon S3이 이벤트 구조에 새 필드를 추가하는 경우 마이너 버전이 증가합니다. 이는 일부 또는 모든 기존 이벤트에 새 정보가 제공되는 경우 발생할 수 있습니다. 이는 새로 도입된 이벤트 유형에 대해서만 새 정보가 제공되는 경우에도 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.
- `eventName`은 이벤트 알림 유형 목록을 참조하지만 `s3:` 접두사는 포함하지 않습니다.
- 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 S3이 개별 요청을 추적하는 데 도움이 됩니다. 이러한 값은 이벤트를 시작한 요청에 대한 응답에서 Amazon S3가 반환하는 값과 동일합니다. 이는 이벤트를 요청에 일치시키는 데 사용할 수 있도록 하기 위한 것입니다.
- `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 S3은 응답의 콘텐츠 유형으로 'application/x-www-form-urlencoded'를 반환함).
- `sequencer` 키는 이벤트 시퀀스를 정의하는 방법을 제공합니다. 이벤트 알림은 이벤트가 발생한 순서대로 도착하지 않을 수 있습니다. 단, 객체를 생성(PUT)하고 객체를 삭제하는 이벤트의 알림에는 `sequencer`가 포함됩니다. 이는 지정된 객체 키에 대한 이벤트 순서를 결정하는 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 저장하는 것이 좋습니다.

  다음 사항에 유의하세요.

  - `sequencer`는 다른 객체 키의 이벤트에 대한 순서를 정의하는 데 사용할 수 없습니다.

  - 시퀀서는 길이가 서로 다를 수 있습니다. 따라서 이러한 값을 비교하려면 먼저 오른쪽 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.
- `intelligentTieringEventData` 키는 S3 Intelligent-Tiering 이벤트에만 표시됩니다.
- `lifecycleEventData` 키는 S3 수명 주기 전환 이벤트에만 표시됩니다.

예제 메시지

다음은 Amazon S3 이벤트 알림 메시지의 예입니다.

Amazon S3 테스트 메시지

버킷에 이벤트 알림을 구성한 후 Amazon S3가 다음과 같은 테스트 메시지를 전송합니다.

```
{
   "Service":"Amazon S3",
   "Event":"s3:TestEvent",
   "Time":"2014-10-13T15:57:02.089Z",
   "Bucket":"amzn-s3-demo-bucket",
   "RequestId":"5582815E1AEA5ADF",
   "HostId":"8cLeGAmw098X5cv4Zkwcmo8vvZa3eH3eKxsPzbB9wrR+YstdA6Knx4Ip8EXAMPLE"
}
```

> **Note**
>
> `s3:TestEvent` 메시지는 일반 S3 이벤트 알림과 다른 형식을 사용합니다. 위에 표시된 `Records` 배열 구조를 사용하는 다른 이벤트 알림과 달리 테스트 이벤트는 직접 필드가 있는 단순화된 형식을 사용합니다. 이벤트 처리를 구현할 때 코드가 두 메시지 형식을 구분하고 올바르게 처리할 수 있는지 확인합니다.

PUT 요청을 사용하여 객체를 생성할 때의 예제 메시지

다음 메시지는 Amazon S3에서 `s3:ObjectCreated:Put` 이벤트를 게시하기 위해 보내는 메시지의 예입니다.

```
{
   "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":"amzn-s3-demo-bucket",
               "ownerIdentity":{
                  "principalId":"A3NL1KOZZKExample"
               },
               "arn":"arn:aws:s3:::amzn-s3-demo-bucket"
            },
            "object":{
               "key":"HappyFace.jpg",
               "size":1024,
               "eTag":"d41d8cd98f00b204e9800998ecf8427e",
               "versionId":"096fKKXTRTtl3on89fVO.nfljtsv6qko",
               "sequencer":"0055AED6DCD90281E5"
            }
         }
      }
   ]
}
```

각 IAM 식별 접두사(예: AIDA, AROA, AGPA)에 대한 자세한 내용은 IAM 사용 설명서의 IAM 식별자를 참조하십시오.

EventBridge 사용

Amazon S3는 버킷에서 특정 이벤트가 발생할 때마다 Amazon EventBridge에 이벤트를 보낼 수 있습니다. 다른 대상과 달리 전송할 이벤트 유형을 선택할 필요가 없습니다. EventBridge가 사용 설정되면 아래의 모든 이벤트가 EventBridge로 전송됩니다. EventBridge 규칙을 사용하여 이벤트를 추가 대상으로 라우팅할 수 있습니다. 다음은 Amazon S3가 EventBridge에 보내는 이벤트 목록입니다.

이벤트 유형	설명
객체 생성됨(Object Created)	객체가 생성되었습니다. 이벤트 메시지 구조의 이유 필드는 객체 생성에 사용된 S3 API(PutObject, POST Object, CopyObject 또는 CompleteMultipartUpload)를 나타냅니다.
객체 삭제됨(DeleteObject)(Object Deleted (DeleteObject)) 객체 삭제됨(수명 주기 만료)(Object Deleted (Lifecycle expiration))	객체가 삭제되었습니다. S3 API 호출을 사용하여 객체를 삭제하면 이유 필드가 DeleteObject로 설정됩니다. S3 수명 주기 만료 규칙에 의해 객체가 삭제되면 이유 필드가 수명 주기 만료(Lifecycle Expiration)로 설정됩니다. 자세한 내용은 객체 만료 섹션을 참조하세요. 버전이 지정되지 않은 객체가 삭제되거나 버전이 지정된 객체가 영구적으로 삭제되면 deletion-type 필드가 영구 삭제됨(Permanently Deleted)으로 설정됩니다. 버전이 지정된 객체에 대해 삭제 마커가 생성되면 deletion-type 필드가 삭제 마커 생성됨(Delete Marker Created)으로 설정됩니다. 자세한 내용은 버전 관리가 사용 설정된 버킷에서 객체 버전 삭제 섹션을 참조하세요.
객체 복원 시작됨	객체 복원이 S3 Glacier 또는 S3 Glacier Deep Archive 스토리지 클래스나 S3 Intelligent-Tiering Archive Access 또는 Deep Archive Access 계층에서 시작되었습니다. 자세한 내용은 아카이브된 객체 작업 섹션을 참조하세요.
객체 복원 완료됨(Object Restore Completed)	객체 복원이 완료되었습니다.

이벤트 유형	설명
객체 복원 만료됨(Object Restore Expired)	S3 Glacier 또는 S3 Glacier Deep Archive에서 복원된 객체의 임시 사본이 만료되어 삭제되었습니다.
객체 스토리지 클래스 변경됨(Object Storage Class Changed)	객체가 다른 스토리지 클래스로 전환되었습니다. 자세한 내용은 Amazon S3 수명 주기를 사용하여 객체 전환 섹션을 참조하세요.
객체 액세스 계층 변경됨(Object Access Tier Changed)	객체가 S3 Intelligent-Tiering Archive Access 계층 또는 Deep Archive Access 계층으로 전환되었습니다. 자세한 내용은 Amazon S3 Intelligent-Tiering을 사용한 스토리지 비용 관리 섹션을 참조하세요.
객체 ACL 업데이트됨(Object ACL Updated)	PutObjectACL을 사용하여 객체의 액세스 제어 목록(ACL)이 설정되었습니다. 요청으로 인해 객체의 ACL이 변경되지 않으면 이벤트가 생성되지 않습니다. 자세한 내용은 ACL(액세스 제어 목록) 개요 섹션을 참조하세요.
객체 태그 추가됨(Object Tags Added)	PutObjectTagging을 사용하여 태그 집합이 객체에 추가되었습니다. 자세한 내용은 태그를 사용하여 스토리지 분류 섹션을 참조하세요.
객체 태그 삭제됨(Object Tags Deleted)	DeleteObjectTagging을 사용하여 객체에서 모든 태그가 제거되었습니다. 자세한 내용은 태그를 사용하여 스토리지 분류 섹션을 참조하세요.

> **Note**
>
> Amazon S3 이벤트 유형이 EventBridge 이벤트 유형에 매핑되는 방법에 대한 자세한 내용은 Amazon EventBridge 매핑 및 문제 해결 섹션을 참조하십시오.

EventBridge와 함께 Amazon S3 이벤트 알림을 사용하여 버킷에서 이벤트가 발생할 때 조치를 취하는 규칙을 작성할 수 있습니다. 예를 들어 알림이 전송되도록 할 수 있습니다. 자세한 내용은 Amazon EventBridge 사용 설명서의 EventBridge란?을 참조하십시오.

EventBridge API를 사용하여 상호 작용할 수 있는 작업 및 데이터 유형에 대한 자세한 내용은 Amazon EventBridge API 참조의 Amazon EventBridge API 참조를 참조하세요.

요금에 대한 자세한 내용은 Amazon EventBridge 요금을 참조하십시오.

주제

- Amazon EventBridge 권한
- Amazon EventBridge 사용 설정
- EventBridge 이벤트 메시지 구조
- Amazon EventBridge 매핑 및 문제 해결

Amazon EventBridge 권한

Amazon S3는 이벤트를 Amazon EventBridge에 전송하기 위해 추가 권한이 필요하지 않습니다.

Amazon EventBridge 사용 설정

S3 콘솔, AWS Command Line Interface(AWS CLI) 또는 Amazon S3 REST API를 사용하여 Amazon EventBridge를 사용 설정할 수 있습니다.

> **Note**
>
> EventBridge를 사용 설정한 후에는 변경 사항이 적용되는 데 5분 정도 걸립니다.

S3 콘솔 사용

S3 콘솔에서 EventBridge 이벤트 전송 사용 설정

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 이벤트를 활성화하려는 버킷의 이름을 선택합니다.
4. [속성(Properties)]을 선택합니다.
5. 이벤트 알림(Event Notifications) 섹션으로 이동하고 Amazon EventBridge 하위 섹션을 찾습니다. 편집을 선택합니다.

6. 이 버킷의 모든 이벤트에 대해 Amazon EventBridge에 알림 보내기(Send notifications to Amazon EventBridge for all events in this bucket)에서 설정(On)을 선택합니다.

AWS CLI 사용

다음 예에서는 Amazon EventBridge가 사용 설정된 버킷 `amzn-s3-demo-bucket1`에 대한 버킷 알림 구성을 생성합니다.

```
aws s3api put-bucket-notification-configuration --bucket amzn-s3-demo-bucket1 --
notification-configuration='{ "EventBridgeConfiguration": {} }'
```

REST API 사용

Amazon S3 REST API를 호출하여 프로그래밍 방식으로 버킷에서 Amazon EventBridge를 사용 설정할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조의 PutBucketNotificationConfiguration을 참조하세요.

다음 예에서는 Amazon EventBridge가 사용 설정된 버킷 알림 구성을 생성하는 데 사용된 XML을 보여줍니다.

```
<NotificationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <EventBridgeConfiguration>
  </EventBridgeConfiguration>
</NotificationConfiguration>
```

EventBridge 규칙 생성

사용 설정되면 특정 작업에 대한 Amazon EventBridge 규칙을 생성할 수 있습니다. 예를 들어 객체가 생성될 때 이메일 알림을 보낼 수 있습니다. 전체 자습서는 Amazon EventBridge 사용 설명서의 자습서: Amazon S3 객체가 생성될 때 알림 보내기를 참조하십시오.

EventBridge 이벤트 메시지 구조

이벤트를 게시하기 위해 Amazon S3이 전송하는 알림 메시지는 JSON 형식입니다. Amazon S3가 이벤트를 Amazon EventBridge로 전송하는 경우 때 다음 필드가 있습니다.

- 버전(version) - 현재 모든 이벤트에 대해 0(영)입니다.
- id - 모든 이벤트에 대해 생성되는 UUID입니다.
- detail-type - 전송 중인 이벤트의 유형입니다. 이벤트 유형 목록은 EventBridge 사용 섹션을 참조하십시오.

- 소스(source) - 이벤트를 생성한 서비스를 식별합니다.
- 계정(account) - S3 버킷 소유자의 12자리 AWS 계정 ID입니다.
- time(시간) - 이벤트가 발생한 시간입니다.
- 리전(region) - 버킷의 AWS 리전을 식별합니다.
- resource(리소스) - 버킷의 Amazon 리소스 이름(ARN)을 포함하는 JSON 배열입니다.
- 세부 정보(detail) - 이벤트에 대한 정보를 포함하는 JSON 객체입니다. 이 필드에 포함될 수 있는 항목에 대한 자세한 내용은 이벤트 메시지 세부 정보 필드 섹션을 참조하십시오.

이벤트 메시지 구조 예제

다음은 Amazon EventBridge로 전송할 수 있는 일부 Amazon S3 이벤트 알림 메시지의 예입니다.

객체 생성됨(Object created)


```
{
  "version": "0",
  "id": "17793124-05d4-b198-2fde-7ededc63b103",
  "detail-type": "Object Created",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
  "resources": [
    "arn:aws:s3:::amzn-s3-demo-bucket1"
  ],
  "detail": {
    "version": "0",
    "bucket": {
      "name": "amzn-s3-demo-bucket1"
    },
    "object": {
      "key": "example-key",
      "size": 5,
      "etag": "b1946ac92492d2347c6235b4d2611184",
      "version-id": "IYV3p45BT0ac8hjHg1houSdS1a.Mro8e",
      "sequencer": "617f08299329d189"
    },
    "request-id": "N4N7GDK58NMKJ12R",
    "requester": "123456789012",
    "source-ip-address": "1.2.3.4",
    "reason": "PutObject"
```

```
  }
}
```

객체 삭제됨(DeleteObject)(Object deleted (using DeleteObject))

```
{
  "version": "0",
  "id": "2ee9cc15-d022-99ea-1fb8-1b1bac4850f9",
  "detail-type": "Object Deleted",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
  "resources": [
    "arn:aws:s3:::amzn-s3-demo-bucket1"
  ],
  "detail": {
    "version": "0",
    "bucket": {
      "name": "amzn-s3-demo-bucket1"
    },
    "object": {
      "key": "example-key",
      "etag": "d41d8cd98f00b204e9800998ecf8427e",
      "version-id": "1QW9g1Z99LUNbvaaYVpW9xDlOLU.qxgF",
      "sequencer": "617f0837b476e463"
    },
    "request-id": "0BH729840619AG5K",
    "requester": "123456789012",
    "source-ip-address": "1.2.3.4",
    "reason": "DeleteObject",
    "deletion-type": "Delete Marker Created"
  }
}
```

객체 삭제됨(수명 주기 만료)(Object deleted (using lifecycle expiration))

```
{
  "version": "0",
  "id": "ad1de317-e409-eba2-9552-30113f8d88e3",
  "detail-type": "Object Deleted",
```

```
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
  "resources": [
    "arn:aws:s3:::amzn-s3-demo-bucket1"
  ],
  "detail": {
    "version": "0",
    "bucket": {
      "name": "amzn-s3-demo-bucket1"
    },
    "object": {
      "key": "example-key",
      "etag": "d41d8cd98f00b204e9800998ecf8427e",
      "version-id": "mtB0cV.jejK63XkRNceanNMC.qXPWLeK",
      "sequencer": "617b398000000000"
    },
    "request-id": "20EB74C14654DC47",
    "requester": "s3.amazonaws.com",
    "reason": "Lifecycle Expiration",
    "deletion-type": "Delete Marker Created"
  }
}
```

객체 복원 완료됨(Object restore completed)

```
{
  "version": "0",
  "id": "6924de0d-13e2-6bbf-c0c1-b903b753565e",
  "detail-type": "Object Restore Completed",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
  "resources": [
    "arn:aws:s3:::amzn-s3-demo-bucket1"
  ],
  "detail": {
    "version": "0",
    "bucket": {
      "name": "amzn-s3-demo-bucket1"
```

```
    },
    "object": {
      "key": "example-key",
      "size": 5,
      "etag": "b1946ac92492d2347c6235b4d2611184",
      "version-id": "KKsjUC1.6gIjqtvhfg5AdMI0eCePIiT3"
    },
    "request-id": "189F19CB7FB1B6A4",
    "requester": "s3.amazonaws.com",
    "restore-expiry-time": "2021-11-13T00:00:00Z",
    "source-storage-class": "GLACIER"
  }
}
```


이벤트 메시지 세부 정보 필드

세부 정보 필드에는 이벤트에 대한 정보가 포함된 JSON 객체가 있습니다. 세부 정보 필드에 다음 필드가 있을 수 있습니다.

- 버전(version) - 현재 모든 이벤트에 대해 0(영)입니다.
- 버킷(bucket) - 이벤트와 관련된 Amazon S3 버킷에 대한 정보입니다.
- 객체(object) - 이벤트와 관련된 Amazon S3 객체에 대한 정보입니다.
- request-id - S3 응답의 요청 ID입니다.
- 요청자(requester) - AWS 계정 ID 또는 요청자의 AWS 서비스 보안 주체입니다.
- source-ip-address - S3 요청의 소스 IP 주소입니다. S3 요청에 의해 트리거된 이벤트의 경우에만 있습니다.
- 이유(reason) - 객체 생성됨(Object Created) 이벤트의 경우 객체 생성에 사용된 S3 API(PutObject, POST Object, CopyObject 또는 CompleteMultipartUpload)입니다. 객체 삭제됨(Object Deleted) 이벤트의 경우 객체가 S3 API 호출에 의해 삭제되면 DeleteObject로 설정되고, S3 수명 주기 만료 규칙에 의해 객체가 삭제되면 수명 주기 만료(Lifecycle Expiration)로 설정됩니다. 자세한 내용은 객체 만료 섹션을 참조하세요.
- deletion-type - 객체 삭제됨(Object Deleted) 이벤트의 경우 버전이 지정되지 않은 객체가 삭제되거나 버전이 지정된 객체가 영구적으로 삭제되면 영구 삭제됨(Permanently Deleted)으로 설정됩니다. 버전이 지정된 객체에 대해 삭제 마커가 생성되면 삭제 마커 생성됨(Delete Marker Created)으로 설정됩니다. 자세한 내용은 버전 관리가 사용 설정된 버킷에서 객체 버전 삭제 섹션을 참조하세요.

> **Note**
>
> 일부 객체 속성(예: `etag` 및 `size`)은 삭제 마커가 생성된 경우에만 표시됩니다.

- restore-expiry-time - 객체 복원 완료됨(Object Restore Completed) 이벤트의 경우 객체의 임시 복사본이 S3에서 삭제되는 시간입니다. 자세한 내용은 아카이브된 객체 작업 섹션을 참조하세요.
- source-storage-class - 객체 복원 시작됨(Object Restore Initiated) 및 객체 복원 완료됨(Object Restore Completed) 이벤트의 경우 복원 중인 객체의 스토리지 클래스입니다. 자세한 내용은 아카이브된 객체 작업 섹션을 참조하세요.
- destination-storage-class - 객체 스토리지 클래스 변경됨(Object Storage Class Changed) 이벤트의 경우 객체의 새 스토리지 클래스입니다. 자세한 내용은 Amazon S3 수명 주기를 사용하여 객체 전환 섹션을 참조하세요.
- destination-access-tier - 객체 액세스 계층 변경됨(Object Access Tier Changed) 이벤트의 경우 객체의 새 액세스 계층입니다. 자세한 내용은 Amazon S3 Intelligent-Tiering을 사용한 스토리지 비용 관리 섹션을 참조하세요.

Amazon EventBridge 매핑 및 문제 해결

다음 표에서는 Amazon S3 이벤트 유형이 Amazon EventBridge 이벤트 유형에 매핑되는 방법을 설명합니다.

S3 이벤트 유형	Amazon EventBridge 세부 정보 유형
ObjectCreated:Put ObjectCreated:Post ObjectCreated:Copy ObjectCreated:CompleteMultipartUpload	객체 생성됨(Object Created)
ObjectRemoved:Delete ObjectRemoved:DeleteMarkerCreated	객체 삭제됨(Object Deleted)

S3 이벤트 유형	Amazon EventBridge 세부 정보 유형
LifecycleExpiration:Delete LifecycleExpiration:DeleteMarkerCreated	
ObjectRestore:Post	객체 복원 시작됨(Object Restore Initiated)
ObjectRestore:Completed	객체 복원 완료됨(Object Restore Completed)
ObjectRestore:Delete	객체 복원 만료됨(Object Restore Expired)
LifecycleTransition	객체 스토리지 클래스 변경됨(Object Storage Class Changed)
IntelligentTiering	객체 액세스 계층 변경됨(Object Access Tier Changed)
ObjectTagging:Put	객체 태그 추가됨(Object Tags Added)
ObjectTagging:Delete	객체 태그 삭제됨(Object Tags Deleted)
ObjectAcl:Put	객체 ACL 업데이트됨(Object ACL Updated)

Amazon EventBridge 문제 해결

Eventbridge 문제 해결 방법에 대한 자세한 내용은 Amazon EventBridge 사용 설명서의 Amazon EventBridge 문제 해결을 참조하십시오.

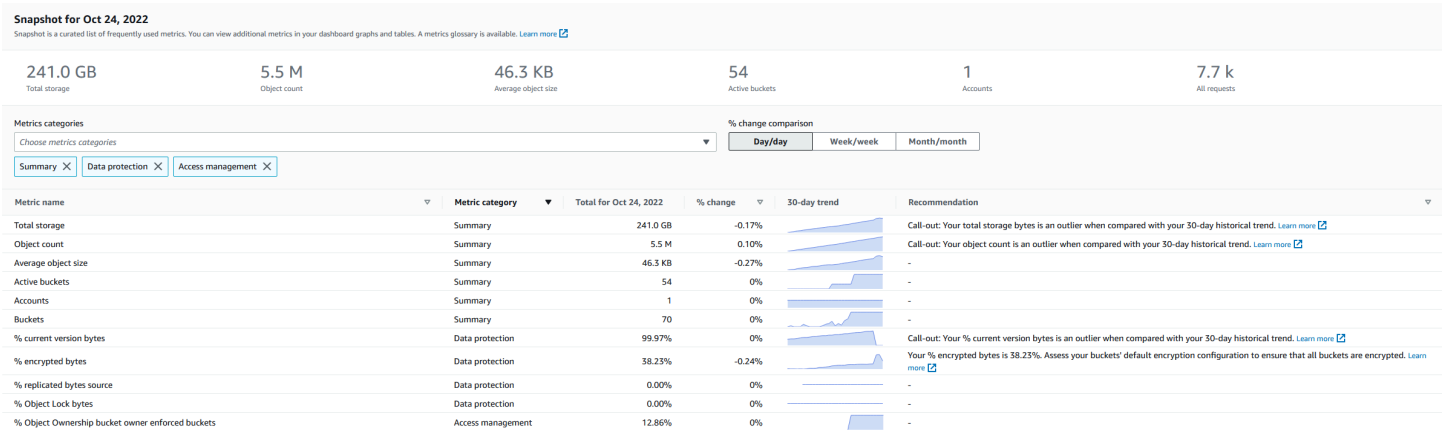
Amazon S3 스토리지 렌즈를 사용하여 스토리지 활동 및 사용량 평가

Amazon S3 스토리지 렌즈는 객체 스토리지 및 활동에 대한 조직 전반의 가시성을 확보하는 데 사용할 수 있는 클라우드 스토리지 분석 기능입니다. S3 스토리지 렌즈는 또한 지표를 분석해 스토리지 비용을 최적화하고 데이터 보호에 대한 모범 사례를 적용하는 데 사용할 수 있는 상황별 권장 사항을 제공합니다.

S3 Storage Lens 지표를 사용하여 요약 인사이트를 생성할 수 있습니다. 예를 들어, 조직 전반의 스토리지 용량 또는 가장 빠르게 증가하는 버킷과 접두사 등을 확인할 수 있습니다. 또한 S3 스토리지 렌즈

지표를 사용하여 비용 최적화 기회를 식별하고, 데이터 보호 및 액세스 관리 모범 사례를 구현하고, 애플리케이션 워크로드의 성능을 개선할 수 있습니다. 예를 들어 S3 수명 주기 규칙 설정이 없는 버킷을 식별하여 7일이 넘게 경과한 미완료 멀티파트 업로드를 만료시킬 수 있습니다. 또한 S3 복제 또는 S3 버전 관리 사용과 같은 데이터 보호 모범 사례를 따르지 않는 버킷을 식별할 수 있습니다.

S3 스토리지 렌즈는 지표를 집계하고 Amazon S3 콘솔 Buckets(버킷) 페이지의 Account snapshot(계정 스냅샷) 섹션에 이 정보를 표시합니다. S3 스토리지 렌즈는 또한 인사이트와 추세를 시각화하고, 이상치에 플래그를 지정하고, 스토리지 비용 최적화와 데이터 보호 모범 사례 적용을 위한 권장 사항을 수신하는 데 사용할 수 있는 대화형 대시보드를 제공합니다. 대시보드에 있는 드릴다운 옵션을 통해 조직, 계정, AWS 리전, 스토리지 클래스, 버킷, 접두사 또는 스토리지 렌즈 그룹 수준에서 인사이트를 생성하거나 시각화할 수 있습니다. 또한 CSV 또는 Parquet 형식의 일일 지표 내보내기를 S3 버킷으로 전송할 수 있습니다.



S3 스토리지 렌즈의 지표 및 기능

S3 스토리지 렌즈는 매일 업데이트되는 대화형 기본 대시보드를 제공합니다. S3 스토리지 렌즈는 대시보드를 사전 구성하여 전체 계정에 대한 요약된 인사이트와 추세를 시각화하고, 이를 S3 콘솔에 매일 업데이트합니다. 이 대시보드의 지표도 Buckets(버킷) 페이지의 계정 스냅샷에 요약됩니다. 자세한 내용은 기본 대시보드 섹션을 참조하십시오.

다른 대시보드를 생성하고 AWS 리전, S3 버킷 또는 계정(AWS Organizations의 경우)을 기준으로 범위를 지정하려면 S3 스토리지 렌즈 대시보드 구성을 생성합니다. Amazon S3 콘솔, AWS Command Line Interface, (AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하여 S3 스토리지 렌즈 대시보드 구성을 생성하고 관리할 수 있습니다. S3 스토리지 렌즈 대시보드를 생성하거나 편집할 때 대시보드 범위와 지표 선택을 정의합니다.

S3 스토리지 렌즈는 무료 지표를 제공하며 추가 비용을 지불하면 고급 지표 및 권장 사항으로 업그레이드할 수 있습니다. 고급 지표 및 권장 사항을 통해 추가 지표 및 기능에 액세스하여 스토리지에 대한 인사이트를 얻을 수 있습니다. 이러한 기능에는 고급 지표 범주, 접두사 집계, 상황별 권장 사항 및

Amazon CloudWatch 게시가 포함됩니다. 접두사 집계 및 상황별 권장 사항은 Amazon S3 콘솔에서만 사용할 수 있습니다. S3 스토리지 렌즈 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

지표 범주

무료 및 고급 티어 내에서 지표는 비용 최적화 및 데이터 보호와 같은 주요 사용 사례에 맞는 범주로 정리되어 있습니다. 무료 지표에는 요약, 비용 최적화, 데이터 보호, 액세스 관리, 성능 및 이벤트 지표가 포함됩니다. 고급 지표 및 권장 사항으로 업그레이드하면 고급 비용 최적화 및 데이터 보호 지표를 사용할 수 있습니다. 이러한 고급 지표를 사용하여 S3 스토리지 비용을 추가로 줄이고 데이터 보호 상태를 개선할 수 있습니다. 또한 활동 지표 및 세부 상태 코드 지표를 활성화하여 S3 버킷에 액세스하는 애플리케이션 워크로드의 성능을 개선할 수 있습니다. 무료 및 고급 지표 범주에 대한 자세한 내용은 지표 선택 섹션을 참조하십시오.

암호화, S3 객체 잠금 또는 S3 버전 관리가 활성화된 버킷의 백분율을 분석하는 등 S3 모범 사례를 기반으로 스토리지를 평가할 수 있습니다. 또한 잠재적인 비용 절감 기회를 식별할 수 있습니다. 예를 들어, S3 수명 주기 규칙 수 지표를 사용하여 수명 주기 만료 또는 전환 규칙이 누락된 버킷을 식별할 수 있습니다. 또한 버킷당 요청 활동을 분석하여 객체를 비용이 낮은 스토리지 클래스로 전환할 수 있는 버킷을 찾을 수도 있습니다. 자세한 내용은 Amazon S3 스토리지 렌즈 지표 사용 사례 섹션을 참조하십시오.

지표 내보내기

S3 콘솔에서 대시보드를 표시하는 것에 더해, 지표를 CSV 또는 Parquet 형식으로 원하는 S3 버킷으로 내보내 원하는 분석 도구를 사용하여 추가로 분석할 수 있습니다. 자세한 내용은 데이터 내보내기를 사용하여 Amazon S3 스토리지 렌즈 지표 보기 섹션을 참조하십시오.

Amazon CloudWatch 게시

S3 스토리지 렌즈 사용량 및 활동 지표를 Amazon CloudWatch에 게시하여 CloudWatch 대시보드에서 운영 상태에 대한 통합 보기를 생성할 수 있습니다. 또한 경보 및 트리거된 작업, 지표 수학, 이상 감지와 같은 CloudWatch 기능을 사용하여 S3 스토리지 렌즈 지표를 모니터링하고 조치를 취할 수 있습니다. 또한 CloudWatch API 작업을 사용하면 서드 파티 공급자를 포함한 애플리케이션이 S3 스토리지 렌즈 지표에 액세스할 수 있습니다. CloudWatch 게시 옵션은 S3 스토리지 렌즈 고급 지표 및 권장 사항으로 업그레이드된 대시보드에 사용할 수 있습니다. CloudWatch의 S3 스토리지 렌즈 지표 지원에 대한 자세한 내용은 CloudWatch의 S3 Storage Lens 지표 모니터링 섹션을 참조하십시오.

S3 스토리지 렌즈 사용에 대한 자세한 내용은 다음 주제를 참조하십시오.

주제

- Amazon S3 스토리지 렌즈 이해

- Amazon S3 스토리지 렌즈 지표 용어집
- Amazon S3 Storage Lens 권한 설정
- 콘솔과 API를 사용하여 Amazon S3 스토리지 렌즈 작업
- Amazon S3 스토리지 렌즈에서 지표 보기
- (으)로 Amazon S3 스토리지 렌즈 사용AWS Organizations
- S3 Storage Lens 그룹을 사용하여 지표 필터링 및 집계

Amazon S3 스토리지 렌즈 이해

> **Important**
>
> 이제 Amazon S3가 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)를 Amazon S3 내 모든 버킷 암호화의 기본 수준으로 적용합니다. 2023년 1월 5일부터 Amazon S3로의 모든 새 객체 업로드는 추가 비용 없이 성능에 영향을 미치지 않고 자동으로 암호화됩니다. S3 버킷 기본 암호화 구성에 및 신규 객체 업로드에 대한 자동 암호화 상태는 AWS CloudTrail 로그, S3 인벤토리, S3 스토리지 렌즈, Amazon S3 콘솔에서 사용할 수 있으며, AWS Command Line Interface 및 AWS SDK에서 추가 Amazon S3 API 응답 헤더로도 사용할 수 있습니다. 자세한 내용은 기본 암호화 관련 FAQ를 참조하십시오.

Amazon S3 스토리지 렌즈는 객체 스토리지 사용 및 활동에 대한 조직 전반의 가시성을 확보하는 데 사용할 수 있는 클라우드 스토리지 분석 기능입니다. S3 스토리지 렌즈 지표를 사용하여 조직 전반의 스토리지 용량 또는 가장 빠르게 증가하는 버킷과 접두사 등의 요약 인사이트를 생성할 수 있습니다. 또한 S3 스토리지 렌즈 지표를 사용하여 비용 최적화 기회를 식별하고, 데이터 보호 및 보안 모범 사례를 구현하고, 애플리케이션 워크로드의 성능을 개선할 수 있습니다. 예를 들어 S3 수명 주기 규칙이 없는 버킷을 식별하여 7일이 넘게 경과한 미완료 멀티파트 업로드를 만료시킬 수 있습니다. 또한 S3 복제 또는 S3 버전 관리 사용과 같은 데이터 보호 모범 사례를 따르지 않는 버킷을 식별할 수 있습니다. S3 스토리지 렌즈는 또한 지표를 분석해 스토리지 비용을 최적화하고 데이터 보호에 대한 모범 사례를 적용하는 데 사용할 수 있는 상황별 권장 사항을 제공합니다.

S3 스토리지 렌즈는 지표를 집계하고 Amazon S3 콘솔 버킷 페이지의 계정 스냅샷 섹션에 이 정보를 표시합니다. S3 스토리지 렌즈는 또한 인사이트와 추세를 시각화하고, 이상치에 플래그를 지정하고, 스토리지 비용 최적화와 데이터 보호 모범 사례 적용을 위한 권장 사항을 수신하는 데 사용할 수 있는 대화형 대시보드를 제공합니다. 대시보드에 있는 드릴다운 옵션을 통해 조직, 계정, AWS 리전, 스토리지 클래스, 버킷, 접두사 또는 스토리지 렌즈 그룹 수준에서 인사이트를 생성하거나 시각화할 수 있습

니다. 또한 CSV 또는 Parquet 형식의 일일 지표 내보내기를 S3 버킷으로 전송할 수 있습니다. Amazon S3 콘솔, AWS Command Line Interface, (AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하여 S3 스토리지 렌즈 대시보드를 생성하고 관리할 수 있습니다.

S3 스토리지 렌즈 개념 및 용어

이 섹션에는 Amazon S3 스토리지 렌즈를 성공적으로 이해하고 사용하는 데 꼭 필요한 용어와 개념이 포함되어 있습니다.

주제

- 대시보드 구성
- 기본 대시보드
- 대시보드
- 계정 스냅샷
- 지표 내보내기
- 홈 리전
- 보존 기간
- 지표 범주
- 추천
- 지표 선택
- S3 스토리지 렌즈 및 AWS Organizations

대시보드 구성

S3 스토리지 렌즈를 사용하려면 단일 대시보드 또는 내보내기를 위해 사용자 대신 지표를 집계하는 데 필요한 속성이 들어 있는 구성이 필요합니다. 구성을 만들 때 대시보드 이름과 홈 리전을 선택합니다. 이 이름은 대시보드를 만든 후에는 변경할 수 없습니다. 선택적으로 태그를 추가하고 지표를 CSV 또는 Parquet 형식으로 내보내도록 구성할 수 있습니다.

대시보드 구성에서는 대시보드 범위와 지표 선택도 정의합니다. 범위에는 조직 계정의 모든 스토리지 또는 리전, 버킷 및 계정별로 필터링된 섹션이 포함될 수 있습니다. 지표 선택을 구성할 때 무료 지표와 추가 비용을 지불하고 업그레이드할 수 있는 고급 지표 및 권장 사항 중에서 선택하게 됩니다. 고급 지표 및 권장 사항을 통해 추가 지표 및 기능에 액세스할 수 있습니다. 이러한 기능에는 고급 지표 범주, 접두사 수준 집계, 상황별 권장 사항 및 Amazon CloudWatch 게시가 포함됩니다. S3 스토리지 렌즈 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

기본 대시보드

콘솔의 S3 스토리지 렌즈 기본 대시보드의 이름은 default-account-dashboard로 지정됩니다. S3는 대시보드를 사전 구성하여 전체 계정에 대한 요약된 인사이트와 추세를 시각화하고, 이를 S3 콘솔에 매일 업데이트합니다. 기본 대시보드의 구성 범위는 수정할 수 없지만 무료 지표에서 고급 지표 및 권장 사항으로 지표 선택을 업그레이드할 수 있습니다. 선택적 지표 내보내기를 구성할 수도 있고 대시보드를 비활성화할 수도 있습니다. 그러나 기본 대시보드는 삭제할 수 없습니다.

> **Note**
>
> 기본 대시보드를 비활성화하면 더 이상 업데이트되지 않습니다. S3 스토리지 렌즈 대시보드, 지표 내보내기 또는 S3 Buckets 페이지의 계정 스냅샷에 더 이상 새로운 일일 지표가 수신되지 않습니다. 대시보드에서 고급 지표 및 권장 사항을 사용하는 경우 더 이상 요금이 청구되지 않습니다. 14일간의 데이터 쿼리 기간이 만료될 때까지 대시보드에서 기록 데이터를 계속해서 볼 수 있습니다. 고급 지표 및 권장 사항을 활성화한 경우 이 기간은 15개월입니다. 만료 기간 내에 대시보드를 다시 활성화하여 기록 데이터에 액세스할 수 있습니다.

대시보드

추가 S3 스토리지 렌즈 대시보드를 생성하고 AWS 리전, S3 버킷 또는 계정(AWS Organizations의 경우)을 기준으로 범위를 지정할 수 있습니다. S3 스토리지 렌즈 대시보드를 생성하거나 편집할 때 대시보드 범위와 지표 선택을 정의합니다. S3 스토리지 렌즈는 무료 지표를 제공하며 추가 비용을 지불하면 고급 지표 및 권장 사항으로 업그레이드할 수 있습니다. 고급 지표 및 권장 사항을 통해 추가 지표 및 기능에 액세스하여 스토리지에 대한 인사이트를 얻을 수 있습니다. 여기에는 고급 지표 범주, 접두사 수준 집계, 상황별 권장 사항 및 Amazon CloudWatch 게시가 포함됩니다. S3 스토리지 렌즈 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

대시보드를 사용 중지하거나 삭제할 수도 있습니다. 대시보드를 사용 중지하면 더 이상 업데이트되지 않으며, 사용자는 더 이상 새로운 일일 지표를 수신하지 않습니다. 14일간의 만료 기간이 끝날 때까지 기록 데이터를 계속해서 볼 수 있습니다. 해당 대시보드에 고급 지표 및 권장 사항을 활성화한 경우 이 기간은 15개월입니다. 만료 기간 내에 대시보드를 다시 활성화하여 기록 데이터에 액세스할 수 있습니다.

대시보드를 삭제하면 모든 대시보드 구성 설정이 손실됩니다. 사용자는 더 이상 새로운 일일 지표를 수신하지 않으며 해당 대시보드와 연결된 기록 데이터에도 액세스할 수 없게 됩니다. 삭제된 대시보드의 기록 데이터에 액세스하려면 동일한 홈 리전에서 동일한 이름의 다른 대시보드를 만들어야 합니다.

> **Note**
>
> - S3 스토리지 렌즈를 사용하여 홈 리전당 최대 50개의 대시보드를 생성할 수 있습니다.
> - 조직 수준의 대시보드는 리전 범위로만 제한할 수 있습니다.

계정 스냅샷

S3 스토리지 렌즈 Account snapshot(계정 스냅샷)은 기본 대시보드의 지표를 요약하여 S3 콘솔 Buckets(버킷) 페이지에 총 스토리지, 객체 수 및 평균 객체 크기를 표시합니다. 계정 스냅샷을 사용하면 Buckets(버킷) 페이지에서 나가지 않고도 스토리지에 대한 인사이트를 빠르게 확인할 수 있습니다. 또한 계정 스냅샷을 통해 대화형 S3 스토리지 렌즈 대시보드에 클릭 한 번으로 액세스할 수 있습니다.

대시보드를 사용하여 인사이트와 추세를 시각화하고, 이상치에 플래그를 지정하고, 스토리지 비용 최적화와 데이터 보호 모범 사례 적용을 위한 권장 사항을 수신할 수 있습니다. 대시보드에 있는 드릴다운 옵션을 통해 조직, 계정, 버킷, 객체 또는 접두사 수준에서 인사이트를 생성할 수 있습니다. 또한 하루에 한 번 지표를 CSV 또는 Parquet 형식으로 내보내서 S3 버킷으로 전송할 수 있습니다.

default-account dashboard(기본 계정 대시보드)의 대시보드 범위는 Account snapshot(계정 스냅샷)에 연결되어 있으므로 수정할 수 없습니다. default-account-dashboard(기본 계정 대시보드)의 지표 선택을 무료 지표에서 유료 고급 지표 및 권장 사항으로 업그레이드할 수 있습니다. 업그레이드한 다음 S3 스토리지 렌즈 Account snapshot(계정 스냅샷)에서 모든 요청, 업로드된 바이트 및 다운로드된 바이트를 표시할 수 있습니다.

> **Note**
>
> 기본 대시보드를 비활성화하면 Account snapshot(계정 스냅샷)이 더 이상 업데이트되지 않습니다. default-account-dashboard(기본 계정 대시보드)를 다시 활성화하여 Account snapshot(계정 스냅샷)에 지표 표시를 계속할 수 있습니다.

지표 내보내기

S3 스토리지 렌즈 지표 내보내기는 S3 스토리지 렌즈 구성에서 식별된 모든 지표를 포함한 파일입니다. 이 정보는 CSV 또는 Parquet 형식으로 매일 생성되며 S3 버킷으로 전송됩니다. 지표를 내보내면 원하는 지표 도구를 사용하여 추가로 분석할 수 있습니다. 지표 내보내기를 위한 S3 버킷은 S3 스토리지 렌즈 구성과 동일한 리전에 있어야 합니다. 대시보드 구성을 편집하여 S3 콘솔에서 S3 스토리지 렌

즈 지표 내보내기를 생성할 수 있습니다. AWS CLI 및 AWS SDK를 사용하여 지표 내보내기를 구성할 수도 있습니다.

홈 리전

홈 리전은 지정된 대시보드 구성에 대한 모든 S3 스토리지 렌즈 지표가 저장되는 AWS 리전입니다. S3 스토리지 렌즈 대시보드 구성을 생성할 때 홈 리전을 선택해야 합니다. 홈 리전을 선택한 후에는 변경할 수 없습니다. 또한 Storage Lens 그룹을 생성하는 경우 Storage Lens 대시보드와 동일한 홈 리전을 선택하는 것이 좋습니다.

> **Note**
>
> 다음 리전 중 하나를 홈 리전으로 선택할 수 있습니다.
>
> - 미국 동부(버지니아 북부) – `us-east-1`
> - 미국 동부(오하이오) – `us-east-2`
> - 미국 서부(캘리포니아 북부) – `us-west-1`
> - 미국 서부(오리건) – `us-west-2`
> - 아시아 태평양(뭄바이) – `ap-south-1`
> - 아시아 태평양(서울) – `ap-northeast-2`
> - 아시아 태평양(싱가포르) – `ap-southeast-1`
> - 아시아 태평양(시드니) – `ap-southeast-2`
> - 아시아 태평양(도쿄) – `ap-northeast-1`
> - 캐나다(중부) – `ca-central-1`
> - 중국(베이징) – `cn-north-1`
> - 중국(닝샤) – `cn-northwest-1`
> - 유럽(프랑크푸르트) – `eu-central-1`
> - 유럽(아일랜드) – `eu-west-1`
> - 유럽(런던) – `eu-west-2`
> - 유럽(파리) – `eu-west-3`
> - 유럽(스톡홀름) – `eu-north-1`
> - 남아메리카(상파울루) – `sa-east-1`

보존 기간

S3 스토리지 렌즈 지표는 보존되므로 사용자는 과거 추세를 확인하고 시간 경과에 따른 스토리지와 활동의 차이를 비교할 수 있습니다. 과거 추세를 확인하고 시간 경과에 따른 스토리지 사용량과 활동의 차이를 비교할 수 있도록 쿼리에 Amazon S3 스토리지 렌즈 지표를 사용할 수 있습니다.

모든 S3 스토리지 렌즈 지표는 15개월 동안 보존됩니다. 하지만 지표는 선택한 지표에 따라 달라지는 특정 기간 동안의 쿼리에만 사용할 수 있습니다. 이 기간은 수정할 수 없습니다. 무료 지표는 14일 동안 쿼리에 사용할 수 있으며 고급 지표는 15개월 동안 쿼리에 사용할 수 있습니다.

지표 범주

무료 및 고급 티어 내에서 S3 스토리지 지표는 비용 최적화 및 데이터 보호와 같은 주요 사용 사례에 맞는 범주로 정리되어 있습니다. 무료 지표에는 요약, 비용 최적화, 데이터 보호, 액세스 관리, 성능 및 이벤트 지표가 포함됩니다. 고급 지표 및 권장 사항으로 업그레이드하면 추가 비용 최적화 및 데이터 보호 지표를 활성화하여 S3 스토리지 비용을 추가로 절감하고 데이터를 확실하게 보호할 수 있습니다. 또한 활동 지표 및 세부 상태 코드 지표를 활성화하여 애플리케이션 워크플로우의 성능을 개선할 수 있습니다.

다음 목록은 무료 및 고급 지표 범주를 모두 보여줍니다. 각 범주에 포함된 개별 지표의 전체 목록은 지표 용어집을 참조하십시오.

요약 지표

요약 지표는 총 스토리지 바이트 및 객체 수를 포함하여 S3 스토리지에 대한 일반적인 인사이트를 제공합니다.

비용 최적화 지표

비용 최적화 지표는 스토리지 비용을 관리하고 최적화하는 데 사용할 수 있는 인사이트를 제공합니다. 예를 들어 7일이 넘게 경과한 미완료 멀티파트 업로드가 있는 버킷을 식별할 수 있습니다.

고급 지표 및 권장 사항을 사용하면 고급 비용 최적화 지표를 사용할 수 있습니다. 이러한 지표에는 버킷당 만료 및 전환 S3 수명 주기 규칙 수를 확인하는 데 사용할 수 있는 S3 수명 주기 규칙 수 지표가 포함됩니다.

데이터 보호 지표

데이터 보호 지표는 암호화 및 S3 버전 관리와 같은 데이터 보호 기능에 대한 인사이트를 제공합니다. 이러한 지표를 사용하여 데이터 보호 모범 사례를 따르지 않는 버킷을 식별할 수 있습니다. 예를 들어 AWS Key Management Service 키(SSE-KMS)가 포함된 기본 암호화 또는 S3 버전 관리를 사용하지 않는 버킷을 식별할 수 있습니다.

고급 지표 및 권장 사항을 사용하면 고급 데이터 보호 지표를 사용할 수 있습니다. 이러한 지표에는 버킷별 복제 규칙 수 지표가 포함됩니다.

액세스 관리 지표

액세스 관리 지표는 S3 객체 소유권에 대한 인사이트를 제공합니다. 이러한 지표를 사용하여 버킷에서 사용하는 객체 소유권 설정을 확인할 수 있습니다.

이벤트 지표

이벤트 지표는 S3 이벤트 알림에 대한 인사이트를 제공합니다. 이벤트 지표를 사용하면 S3 이벤트 알림이 구성된 버킷을 확인할 수 있습니다.

성능 지표

성능 지표는 S3 Transfer Accelation에 대한 인사이트를 제공합니다. 성능 지표를 사용하면 Transfer Acceleration이 활성화된 버킷을 확인할 수 있습니다.

활동 지표(고급)

대시보드를 고급 지표 및 권장 사항으로 업그레이드하면 활동 지표를 활성화할 수 있습니다. 활동 지표는 스토리지 요청 방식(예: 모든 요청, Get 요청, Put 요청), 업로드 또는 다운로드된 바이트, 오류에 대한 세부 정보를 제공합니다.

접두사 수준의 활동 지표를 사용하면 자주 사용되지 않는 접두사를 파악할 수 있으므로 S3 Lifecycle을 사용하여 보다 최적의 스토리지 클래스로 전환할 수 있습니다.

세부 상태 코드 지표(고급)

대시보드를 고급 지표 및 권장 사항으로 업그레이드하면 세부 상태 코드 지표를 활성화할 수 있습니다. 세부 상태 코드 지표는 액세스 또는 성능 문제를 해결하는 데 사용할 수 있는 403 금지됨 및 503 서비스 사용 불가와 같은 HTTP 상태 코드에 대한 인사이트를 제공합니다. 예를 들어 403 금지됨 오류 수 지표를 살펴보면 올바른 권한이 적용되지 않은 상태에서 버킷에 액세스하는 워크로드를 식별할 수 있습니다.

접두사 수준의 세부 상태 코드 지표를 사용하면 접두사별 HTTP 상태 코드 발생을 더 잘 이해할 수 있습니다. 예를 들어 503 오류 수 지표를 사용하면 데이터 모으기 중에 제한 요청을 받는 접두사를 식별할 수 있습니다.

추천

S3 스토리지 렌즈는 스토리지를 최적화하는 데 도움이 되는 자동화된 권장 사항을 제공합니다. 권장 사항은 S3 스토리지 렌즈 대시보드의 관련 지표와 함께 상황별로 배치됩니다. 권장 사항은 가장 최근

기간에 발생한 일과 관련이 있으므로 기록 데이터는 권장 사항에 적합하지 않습니다. 권장 사항은 관련성이 있는 경우에만 나타납니다.

S3 스토리지 렌즈 권장 사항은 다음 형식으로 제공됩니다.

- 제안

  제안은 스토리지 사용량 및 활동의 추세를 알려주므로 스토리지 비용 최적화 기회 또는 데이터 보호 모범 사례를 시사할 수 있습니다. Amazon S3 사용 설명서 및 S3 스토리지 렌즈 대시보드의 제안된 주제를 사용하여 특정 리전, 버킷 또는 접두사에 대한 자세한 내용을 드릴다운할 수 있습니다.

- 콜아웃

  콜아웃은 추가 주의나 모니터링이 필요할 수 있는 기간 동안 스토리지 및 활동 내에서 흥미로운 변칙 상황을 알려주는 권장 사항입니다.

  - 이상 콜아웃

    S3 스토리지 렌즈는 최근 30일 추세를 기반으로 이상인 지표에 대한 콜아웃을 제공합니다. 이상은 표준 점수(z 점수라고 함)를 사용하여 계산됩니다. 이 점수는 먼저 현재 날짜의 지표를 해당 지표에 대한 지난 30일 평균에서 뺍니다. 그런 다음, 현재 날짜의 지표를 지난 30일 동안의 해당 지표에 대한 표준 편차로 나눕니다. 그렇게 해서 나오는 점수는 일반적으로 -3과 +3 사이입니다. 이 수치는 현재 날짜의 지표가 평균에서 벗어난 표준 편차의 수를 나타냅니다.

    S3 스토리지 렌즈에서는 점수가 2보다 크거나 -2보다 작은 지표를 이상으로 간주합니다. 이러한 수치는 정규 분포 데이터의 95%보다 높거나 낮기 때문입니다.

  - 중요한 변경 콜아웃

    중요한 변경 콜아웃은 자주 변경되지 않을 것으로 예상되는 지표에 적용됩니다. 따라서 이상 계산보다 높은 민감도로 설정되며, 일반적으로 전날, 전주 또는 전월과 비교하여 +/-20%의 범위에 있습니다.

    스토리지 및 활동에서 콜아웃 해결 – 중요한 변경 콜아웃을 받았다고 해서 반드시 문제가 되는 것은 아닙니다. 콜아웃은 스토리지의 예상 변경으로 인한 것일 수 있습니다. 예를 들어, 최근에 많은 수의 새 객체를 추가했거나 많은 수의 객체를 삭제했거나 유사한 계획된 변경을 수행했을 수 있습니다.

    대시보드에 중요한 변경 콜아웃이 표시되면 이를 기록하고 최근 상황으로 설명 가능한지를 판단합니다. 설명되지 않는 경우, S3 스토리지 렌즈 대시보드를 사용해 세부 정보를 드릴다운하여 변동을 유발하는 특정 리전, 버킷 또는 접두사를 알아봅니다.

- 미리 알림

미리 알림은 Amazon S3의 작동 방식에 대한 통찰력을 제공합니다. 미리 알림은 S3 기능을 사용해 스토리지 비용을 절감하거나 데이터 보호 모범 사례를 적용하는 방법에 대해 자세히 알 수 있도록 도와줍니다.

지표 선택

S3 스토리지 렌즈는 대시보드 및 내보내기에 대해 선택할 수 있는 두 가지 지표, 즉 무료 지표 및 고급 지표 및 권장 사항을 제공합니다.

- 무료 지표

  S3 스토리지 렌즈는 모든 대시보드 및 구성에 대한 무료 지표를 제공합니다. 무료 지표에는 계정에 있는 버킷 수 및 객체와 같은 스토리지와 관련된 지표가 포함됩니다. 무료 지표에는 스토리지가 S3 모범 사례에 따라 구성되었는지 여부를 조사하는 데 사용할 수 있는 사용 사례 기반 지표(예: 비용 최적화 및 데이터 보호 지표)도 포함되어 있습니다. 모든 무료 지표는 매일 수집됩니다. 데이터는 14일 동안 쿼리에 사용할 수 있습니다. 무료 지표에서 사용할 수 있는 지표에 대한 자세한 내용은 Amazon S3 스토리지 렌즈 지표 용어집 섹션을 참조하십시오.

- 고급 지표 및 권장 사항

  S3 스토리지 렌즈는 고급 지표 및 권장 사항으로 업그레이드할 수 있는 옵션과 함께 모든 대시보드 및 구성에 대한 무료 지표를 제공합니다. 추가 요금이 발생합니다. 자세한 내용은 Amazon S3 요금을 참조하십시오.

  고급 지표 및 권장 사항에는 고급 데이터 보호 및 비용 최적화 지표, 활동 지표 및 세부 상태 코드 지표와 같은 추가 지표와 함께 무료 지표의 모든 지표가 포함됩니다. 고급 지표 및 권장 사항을 통해 스토리지를 최적화하는 데 도움이 되는 권장 사항도 제공됩니다. 권장 사항은 대시보드의 관련 지표와 함께 상황별로 배치됩니다.

  고급 지표 및 권장 사항에는 다음 기능도 포함됩니다.

  - 고급 지표 – 추가 지표를 생성합니다. 고급 지표 범주 전체 목록은 지표 범주 섹션을 참조하십시오. 전체 지표 목록은 Amazon S3 스토리지 렌즈 지표 용어집 섹션을 참조하십시오.
  - Amazon CloudWatch 게시 – S3 스토리지 렌즈 지표를 CloudWatch에 게시하여 CloudWatch 대시보드에서 운영 상태에 대한 통합 보기를 생성합니다. 또한 경보 및 트리거된 작업, 지표 수학, 이상 감지와 같은 CloudWatch API 작업 및 기능을 사용하여 S3 스토리지 렌즈 지표를 모니터링하고 조치를 취할 수 있습니다. 자세한 내용은 CloudWatch의 S3 Storage Lens 지표 모니터링 섹션을 참조하십시오.

- 접두사 집계 - 접두사 수준에서 지표를 수집합니다. 접두사 집계를 활성화하면 대시보드 구성에 포함된 모든 지표가 접두사 수준에서 확장됩니다. 지표는 구성된 임계값을 충족하는 접두사에 대해서만 생성됩니다. 참고로, 접두사 수준에서 적용할 수 있는 지표는 접두사 집계와 함께 사용할 수 있습니다. 단, 버킷 수준 설정 및 규칙 수 지표는 예외입니다. 접두사 수준 지표는 CloudWatch에 게시되지 않습니다.
- Storage Lens 그룹 집계 - Storage Lens 그룹 수준에서 지표를 수집합니다. 고급 지표 및 권장 사항과 Storage Lens 그룹 집계를 활성화한 후 Storage Lens 대시보드에 포함하거나 제외할 Storage Lens 그룹을 지정할 수 있습니다. Storage Lens 그룹을 하나 이상 지정해야 합니다. 또한 지정된 Storage Lens 그룹은 대시보드 계정의 지정된 홈 리전 내에 있어야 합니다. 스토리지 렌즈 그룹 수준 지표는 CloudWatch에 게시되지 않습니다.

모든 고급 지표는 매일 수집됩니다. 데이터는 15개월 동안 쿼리에 사용할 수 있습니다. S3 스토리지 렌즈로 집계된 스토리지 지표에 대한 자세한 내용은 Amazon S3 스토리지 렌즈 지표 용어집 섹션을 참조하십시오.

> **Note**
>
> 권장 사항은 Amazon S3 콘솔에서 S3 스토리지 렌즈 대시보드를 사용하는 경우에만 제공됩니다.

S3 스토리지 렌즈 및 AWS Organizations

AWS Organizations는 하나의 조직 계층에서 모든 AWS 계정을 집계하는 데 도움이 되는 AWS 서비스입니다. Amazon S3 스토리지 렌즈는 AWS Organizations와 함께 Amazon S3 스토리지 전반의 객체 스토리지 및 활동을 한곳에서 볼 수 있게 해줍니다.

자세한 내용은 (으)로 Amazon S3 스토리지 렌즈 사용AWS Organizations 섹션을 참조하십시오.

- 트러스트된 액세스

  조직의 관리 계정을 사용할 때, S3 스토리지 렌즈에 대한 트러스트된 액세스를 활성화하여 조직의 모든 구성원 계정에 대한 스토리지 지표와 사용량 데이터를 집계해야 합니다. 그런 다음 관리 계정을 사용하거나 조직의 다른 계정에 대해 위임된 관리자 액세스 권한을 부여하여 조직에 대한 대시보드 또는 내보내기를 생성할 수 있습니다.

  언제든지 S3 스토리지 렌즈에 대해 트러스트된 액세스를 사용 중지할 수 있습니다. 그러면 S3 스토리지 렌즈가 조직의 지표를 집계하지 못합니다.

- 위임된 관리자

  AWS Organizations 관리 계정을 사용하거나 조직의 다른 계정에 대해 위임된 관리자 액세스 권한을 부여하여 조직의 S3 스토리지 렌즈에 대한 대시보드 및 지표를 생성할 수 있습니다. 위임된 관리자의 등록을 언제든지 취소할 수 있습니다. 또한 위임된 관리자의 등록을 취소하면 위임된 관리자가 생성한 모든 조직 수준의 대시보드가 새 스토리지 지표를 집계하는 것이 자동으로 중지됩니다.

자세한 내용은 AWS Organizations 사용 설명서의 Amazon S3 스토리지 렌즈 및 AWS Organizations를 참조하십시오.

Amazon S3 스토리지 렌즈 서비스 연결 역할

Amazon S3 스토리지 렌즈는 AWS Organizations의 신뢰할 수 있는 액세스와 함께 AWS Identity and Access Management(IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 S3 스토리지 렌즈에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 S3 스토리지 렌즈가 미리 정의하며 여기에는 조직의 구성원 계정에서 일일 스토리지 및 활동 지표를 수집하는 데 필요한 모든 권한이 포함되어 있습니다.

자세한 내용은 Amazon S3 스토리지 렌즈에 서비스 연결 역할 사용을 참조하십시오.

Amazon S3 스토리지 렌즈 지표 용어집

Amazon S3 스토리지 렌즈 지표 용어집은 S3 스토리지 렌즈용 무료 및 고급 지표의 전체 목록을 제공합니다.

S3 스토리지 렌즈는 고급 지표로 업그레이드할 수 있는 옵션과 함께 모든 대시보드 및 구성에 대한 무료 지표를 제공합니다.

- 무료 지표에는 계정에 있는 버킷 수 및 객체와 같은 스토리지 사용량과 관련된 지표가 포함됩니다. 무료 지표에는 비용 최적화 및 데이터 보호 지표와 같은 사용 사례 기반 지표도 포함되어 있습니다. 모든 무료 지표는 매일 수집되며 데이터는 최장 14일 동안 쿼리에 사용할 수 있습니다.
- 고급 지표 및 권장 사항에는 고급 데이터 보호 및 비용 최적화 지표와 같은 추가 지표와 함께 무료 지표의 모든 지표가 포함됩니다. 고급 지표에는 활동 지표 및 세부 상태 코드 지표와 같은 추가 지표 범주도 포함됩니다. 고급 지표 데이터는 15개월 동안 쿼리에 사용할 수 있습니다.

  고급 지표 및 권장 사항과 함께 S3 스토리지 렌즈를 사용하면 추가 요금이 부과됩니다. 자세한 내용은 Amazon S3 요금을 참조하세요. 고급 지표 및 권장 사항 기능에 대한 자세한 내용은 지표 선택 섹션을 참조하십시오.

> **Note**
>
> Storage Lens 그룹의 경우 프리 티어 스토리지 지표만 사용할 수 있습니다. Storage Lens 그룹 수준에서는 고급 티어 지표를 사용할 수 없습니다.

지표 이름

다음 표의 지표 이름 열에는 S3 콘솔에 있는 각 S3 스토리지 렌즈의 이름이 나와 있습니다. CloudWatch 및 내보내기 열에는 Amazon CloudWatch의 각 지표 이름과 S3 스토리지 렌즈 대시보드에서 구성할 수 있는 지표 내보내기 파일이 표시됩니다.

파생된 지표 공식

파생된 지표는 지표 내보내기 및 CloudWatch 게시 옵션에 사용할 수 없습니다. 그러나 Derived metrics formula(파생된 지표 수학) 열에 표시된 지표 수학을 사용하여 계산할 수 있습니다.

Amazon S3 스토리지 렌즈 지표 단위 배수용 접두사 기호(K, M, G 등) 해석

S3 스토리지 렌즈 지표 단위 배수는 접두사 기호와 함께 작성됩니다. 이러한 접두사 기호는 국제도량형국(BIPM)에서 표준화한 국제단위계(SI) 기호와 일치합니다. 이러한 기호는 UCUM(측정 단위에 대한 통합 코드)에도 사용됩니다. 자세한 내용은 SI 접두사 기호 목록을 참조하십시오.

> **Note**
>
> - S3 스토리지 바이트의 측정 단위는 바이너리 기가바이트(GB) 단위이며, 여기서 1GB는 2^{30}바이트, 1TB는 2^{40}바이트, 1PB는 2^{50}바이트입니다. 이 측정 단위는 국제전기기술위원회(IEC)에서 정의한 대로 기비바이트(GiB)라고도 합니다.
> - 수명 주기 구성에 따라 객체가 수명 주기의 끝에 도달한 경우, Amazon S3에서는 제거를 위해 객체를 대기열에 넣고 비동기 방식으로 제거합니다. 따라서 만료 날짜와 Amazon S3에서 객체를 제거하는 날짜 사이에 지연이 있을 수 있습니다. 만료되었지만 제거되지 않은 객체에 대한 지표는 S3 스토리지 렌즈에 포함되지 않습니다. S3 수명 주기에서 만료 작업에 대한 자세한 내용은 객체 만료 섹션을 참조하십시오.

다음 테이블에는 S3 Storage Lens 지표 용어집이 나와 있습니다.

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
총 스토리지	StorageBytes	전체 스토리지(불완전한 멀티파트 업로드, 객체 메타데이터 및 삭제 마커 포함)	무료	요약	N	-
객체 수	ObjectCount	총 객체 수	무료	요약	N	-
평균 객체 크기	-	평균 객체 크기	무료	요약	Y	sum(StorageBytes)/sum(ObjectCount)
활성 버킷	-	스토리지가 0바이트를 초과하는 총 버킷 수	무료	요약	Y	-
버킷	-	버킷의 총수	무료	요약	Y	-
Accounts	-	스토리지가 범위 내에 있는 계정 수	무료	요약	Y	-
최신 버전 바이트	CurrentVersionStorageBytes	객체의 최신 버전인 바이트 수	무료	비용 최적화	N	-

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
최신 버전 바이트 비율(%)	-	객체의 최신 버전인 범위 내 바이트의 백분율	무료	비용 최적화	Y	sum(CurrentVersionStorageBytes)/sum(StorageBytes)
최신 버전 객체 수	CurrentVersionObjectCount	현재 버전 객체 수	무료	데이터 보호	N	-
최신 버전 객체 비율(%)	-	최신 버전인 범위 내 객체의 백분율	무료	비용 최적화	Y	sum(CurrentVersionObjectCount)/sum(ObjectCount)
비최신 버전 바이트	NonCurrentVersionStorageBytes	비최신 버전 바이트 수	무료	비용 최적화	N	-

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
비최신 버전 바이트 비율	-	비최신 버전인 범위 내 바이트의 백분율	무료	비용 최적화	Y	sum(NonCurrentVersionStorageBytes)/ sum(StorageBytes)
비최신 버전 객체 수	NonCurrentVersionObjectCount	비최신 버전 객체 수	무료	비용 최적화	N	-
비최신 버전 객체 비율(%)	-	최신 버전이 아닌 범위 내 객체의 백분율	무료	비용 최적화	Y	sum(NonCurrentVersionObjectCount)/ sum(ObjectCount)
삭제 마커 바이트	DeleteMarkerStorageBytes	삭제 마커인 범위 내 바이트의 수	무료	비용 최적화	N	-

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
삭제 마커 바이트 비율(%)	-	삭제 마커인 범위 내 바이트의 백분율	무료	비용 최적화	Y	sum(DeleteMarkerStorageBytes)/sum(StorageBytes)
삭제 마커 객체 수	DeleteMarkerObjectCount	삭제 마커가 있는 객체의 총 수	무료	비용 최적화	N	-
삭제 마커 객체 비율(%)	-	삭제 마커가 있는 범위 내 객체의 백분율	무료	비용 최적화	Y	sum(DeleteMarkerObjectCount)/sum(ObjectCount)
미완료 멀티파트 업로드 바이트	IncompleteMultipartUploadStorageBytes	미완료 멀티파트 업로드의 범위 내 총 바이트	무료	비용 최적화	N	-

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
미완료 멀티파트 업로드 바이트 비율(%)	-	미완료 멀티파트 업로드의 결과인 범위 내 바이트의 백분율	무료	비용 최적화	Y	sum(IncompleteMultipartUploadStorageBytes)/sum(StorageBytes)
미완료 멀티파트 업로드 객체 수	IncompleteMultipartUploadObjectCount	미완료 멀티파트 업로드인 범위 내 객체의 수	무료	비용 최적화	N	-
미완료 멀티파트 업로드 객체 비율(%)	-	미완료 멀티파트 업로드인 범위 내 객체의 백분율	무료	비용 최적화	Y	sum(IncompleteMultipartUploadObjectCount)/sum(ObjectCount)
7일이 넘게 경과한 미완료 멀티파트 업로드 스토리지 바이트	IncompleteMPUStorageBytesOlderThan7Days	7일이 넘게 경과한 미완료 멀티파트 업로드의 범위 내 총 바이트	무료	비용 최적화	N	-

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
7일이 넘게 경과한 미완료 멀티파트 업로드 스토리지 바이트 비율(%)	-	7일이 넘게 경과한 미완료 멀티파트 업로드의 바이트 백분율	무료	비용 최적화	Y	sum(IncompleteMPUStorageBytesOlderThan7Days)/sum(StorageBytes)
7일이 넘게 경과한 미완료 멀티파트 업로드 객체 수	IncompleteMPUObjectCountOlderThan7Days	7일이 넘게 경과한 미완료 멀티파트 업로드인 객체의 수	무료	비용 최적화	N	-
7일이 넘게 경과한 미완료 멀티파트 업로드 객체 수 비율(%)	-	7일이 넘게 경과한 미완료 멀티파트 업로드인 객체의 백분율	무료	비용 최적화	Y	sum(IncompleteMPUObjectCountOlderThan7Days)/sum(ObjectCount)
전환 수명 주기 규칙 수	TransitionLifecycleRuleCount	객체를 다른 스토리지 클래스로 전환하기 위한 수명 주기 규칙의 수	고급	비용 최적화	N	-

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
버킷당 평균 전환 수명 주기 규칙	-	객체를 다른 스토리지 클래스로 전환하기 위한 수명 주기 규칙의 평균 수	고급	비용 최적화	Y	sum(TransitionLifecycleRuleCount)/sum(DistinctNumberOfBuckets)
만료 수명 주기 규칙 수	ExpirationLifecycleRuleCount	객체를 만료시키기 위한 수명 주기 규칙의 수	고급	비용 최적화	N	-
버킷당 평균 만료 수명 주기 규칙	-	객체를 만료시키기 위한 수명 주기 규칙의 평균 수	고급	비용 최적화	Y	sum(ExpirationLifecycleRuleCount)/sum(DistinctNumberOfBuckets)
비최신 버전 전환 수명 주기 규칙 수	NoncurrentVersionTransitionLifecycleRuleCount	비최신 버전 객체를 다른 스토리지 클래스로 전환하기 위한 수명 주기 규칙의 수	고급	비용 최적화	N	

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
버킷당 평균 비최신 버전 전환 수명 주기 규칙	-	비최신 버전 객체를 다른 스토리지 클래스로 전환하기 위한 수명 주기 규칙의 평균 수	고급	비용 최적화	Y	sum(NoncurrentVersionTransitionLifecycleRuleCount)/sum(DistinctNumberOfBuckets)
비최신 버전 만료 수명 주기 규칙 수	NoncurrentVersionExpirationLifecycleRuleCount	비최신 객체를 만료시키기 위한 수명 주기 규칙의 수	고급	비용 최적화	N	-
버킷당 평균 비최신 버전 만료 수명 주기 규칙	-	비최신 객체를 만료시키기 위한 수명 주기 규칙의 평균 수	고급	비용 최적화	Y	sum(NoncurrentVersionExpirationLifecycleRuleCount)/sum(DistinctNumberOfBuckets)

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
미완료 멀티파트 업로드 중단 수명 주기 규칙 개수	AbortIncompleteMPULifecycleRuleCount	미완료 멀티파트 업로드를 삭제하기 위한 수명 주기 규칙의 수	고급	비용 최적화	N	-
버킷당 평균 미완료 멀티파트 업로드 중단 수명 주기 규칙	-	미완료 멀티파트 업로드를 삭제하기 위한 수명 주기 규칙의 평균 수	고급	비용 최적화	Y	sum(AbortIncompleteMPULifecycleRuleCount)/sum(DistinctNumberOfBuckets)
만료된 객체 삭제 마커 수명 주기 규칙 수	ExpiredObjectDeleteMarkerLifecycleRuleCount	만료된 객체 삭제 마커를 제거하기 위한 수명 주기 규칙의 수	고급	비용 최적화	N	-

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
버킷당 평균 만료된 객체 삭제 마커 수명 주기 규칙	-	만료된 객체 삭제 마커를 제거하기 위한 수명 주기 규칙의 평균 수	고급	비용 최적화	Y	sum(ExpiredObjectDeleteMarkerLifecycleRuleCount)/ sum(DistinctNumberOfBuckets)
총 수명 주기 규칙 수	TotalLifecycleRuleCount	라이프사이클 규칙의 총수	고급	비용 최적화	N	-
버킷당 평균 라이프사이클 규칙 수	-	수명 주기 규칙의 평균 수	고급	비용 최적화	Y	sum(TotalLifecycleRuleCount)/ sum(DistinctNumberOfBuckets)

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
암호화된 바이트	EncryptedStorageBytes	암호화된 총 바이트 수	무료	데이터 보호	N	-
암호화된 바이트 비율(%)	-	암호화된 총 바이트의 백분율	무료	데이터 보호	Y	sum(EncryptedObjectCount)/sum(StorageBytes)
암호화된 객체 수	EncryptedObjectCount	암호화된 객체의 총수	무료	데이터 보호	N	-
암호화된 객체 비율(%)	-	암호화된 객체의 백분율	무료	데이터 보호	Y	sum(EncryptedStorageBytes)/sum(ObjectCount)

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
암호화되지 않은 바이트	UnencryptedStorageBytes	암호화되지 않은 바이트의 수	무료	데이터 보호	Y	sum(StorageBytes) - sum(EncryptedStorageBytes)
암호화되지 않은 바이트 비율(%)	-	암호화되지 않은 바이트의 백분율	무료	데이터 보호	Y	sum(UnencryptedStorageBytes)/ sum(StorageBytes)
암호화되지 않은 객체 수	UnencryptedObjectCount	암호화되지 않은 객체의 총수	무료	데이터 보호	Y	sum(ObjectCount) - sum(EncryptedObjectCount)
암호화되지 않은 객체 비율(%)	-	암호화되지 않은 객체의 백분율	무료	데이터 보호	Y	sum(UnencryptedStorageBytes)/ sum(ObjectCount)

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
복제된 스토리지 바이트 소스	ReplicatedStorageBytesSource	소스 버킷에서 복제된 바이트의 총 수	무료	데이터 보호	N	-
복제된 바이트 소스 비율(%)	-	소스 버킷에서 복제된 총 바이트의 백분율	무료	데이터 보호	Y	sum(ReplicatedStorageBytesSource)/sum(StorageBytes)
복제된 객체 수 소스	ReplicatedObjectCountSource	원본 버킷에서 복제된 객체의 수	무료	데이터 보호	N	-
복제된 객체 소스 비율(%)	-	소스 버킷에서 복제된 총 객체의 백분율	무료	데이터 보호	Y	sum(ReplicatedStorageObjectCount)/sum(ObjectCount)

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
복제된 스토리지 바이트 대상	ReplicatedStorageBytes	대상 버킷에 복제된 바이트의 총수	무료	데이터 보호	N	-
복제된 바이트 대상 비율(%)	-	대상 버킷에 복제된 총 바이트의 백분율	무료	데이터 보호	Y	sum(ReplicatedStorageBytes) / sum(StorageBytes)
복제된 객체 수 대상	ReplicatedObjectCount	대상 버킷에 복제된 객체의 수	무료	데이터 보호	N	-
복제된 객체 대상 비율(%)	-	대상 버킷에 복제된 총 객체의 백분율	무료	데이터 보호	Y	sum(ReplicatedObjectCount)/ sum(ObjectCount)

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
객체 잠금 바이트	ObjectLockEnabledStorageBytes	객체 잠금이 활성화된 스토리지 바이트의 총수	무료	데이터 보호	N	sum(UnencryptedStorageBytes)/ sum(ObjectLockEnabledStorageCount) - sum(ObjectLockEnabledStorageBytes)

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
객체 잠금 바이트 비율(%)	-	객체 잠금이 활성화된 스토리지 바이트의 백분율	무료	데이터 보호	Y	sum(오브젝트 잠금 지원 스토리지 바이트) / sum(스토리지 바이트)

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
객체 잠금 객체 수	ObjectLockEnabledObjectCount	객체 잠금 객체의 총수	무료	데이터 보호	N	-

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
객체 잠금 객체 비율(%)	-	객체 잠금이 사용 설정된 총 객체의 백분율	무료	데이터 보호	Y	sum(오브젝트 잠금이 활성화된 오브젝트 수) / sum(오브젝트 수)

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
버전 관리가 활성화된 버킷 수	VersioningEnabledBucketCount	S3 버전 관리가 활성화된 버킷의 수	무료	데이터 보호	N	-
버전 관리가 활성화된 버킷 비율(%)	-	S3 버전 관리가 활성화된 버킷의 백분율	무료	데이터 보호	Y	sum(VersioningEnabledBucketCount)/ sum(DistinctNumberOfBuckets)
MFA 삭제가 활성화된 버킷 수	MFADeleteEnabledBucketCount	MFA(멀티 팩터 인증) 삭제가 활성화된 버킷의 수	무료	데이터 보호	N	-
MFA 삭제가 활성화된 버킷 비율(%)	-	MFA(멀티 팩터 인증) 삭제가 활성화된 버킷의 백분율	무료	데이터 보호	Y	sum(MFADeleteEnabledBucketCount)/ sum(DistinctNumberOfBuckets)

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
SSE-KMS가 활성화된 버킷 수	SSEKMSEnabledBucketCount	기본 버킷 암호화에AWS Key Management Service 키가 포함된 서버 측 암호화(SSE-KMS)를 사용하는 버킷의 수	무료	데이터 보호	N	-
SSE-KMS가 활성화된 버킷 비율(%)	-	기본 버킷 암호화에 SSE-KMS를 사용하는 버킷의 백분율	무료	데이터 보호	Y	sum(SSEKMSEnabledBucketCount)/sum(DistinctNumberOfBuckets)
지원되지 않는 모든 서명 요청	AllUnsupportedSignatureRequests	지원되지 않는 AWS 서명 버전을 사용하는 요청의 총수	고급	데이터 보호	N	-

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
지원되지 않는 모든 서명 요청 비율	-	지원되지 않는 AWS 서명 버전을 사용하는 요청의 백분율	고급	데이터 보호	Y	sum(AllUnsupportedSignatureRequests) / sum(AllRequests)
지원되지 않는 모든 TLS 요청	AllUnsupportedTLSRequests	지원되지 않는 전송 계층 보안(TLS) 버전을 사용하는 요청의 수	고급	데이터 보호	N	-
지원되지 않는 모든 TLS 요청 비율	-	지원되지 않는 TLS 버전을 사용하는 요청의 백분율	고급	데이터 보호	Y	sum(AllUnsupportedTLSRequests)/ sum(AllRequests)
모든 SSE-KMS 요청	AllSSEKMSRequests	SSE-KMS를 지정하는 요청의 총수	고급	데이터 보호	N	-

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
모든 SSE-KMS 요청 비율	-	SSE-KMS를 지정하는 요청의 백분율	고급	데이터 보호	Y	sum(AllSSEKMSRequests)/sum(AllRequests)
동일 리전 복제 규칙 수	SameRegionReplicationRuleCount	동일 리전 복제(SRR)에 대한 복제 규칙 수	고급	데이터 보호	N	-
버킷당 평균 동일 리전 복제 규칙 수	-	SRR에 대한 복제 규칙의 평균 수	고급	데이터 보호	Y	sum(SameRegionReplicationRuleCount)/sum(DistinctNumberOfBuckets)
교차 리전 복제 규칙 수	CrossRegionReplicationRuleCount	교차 리전 복제(SRR)에 대한 복제 규칙 수	고급	데이터 보호	N	-

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
버킷당 평균 교차 리전 복제 규칙	-	CRR에 대한 복제 규칙의 평균 수	고급	데이터 보호	Y	sum(CrossRegionReplicationRuleCount) / sum(DistinctNumberOfBuckets)
동일 계정 복제 규칙 수	SameAccountReplicationRuleCount	동일한 계정 내 복제를 위한 복제 규칙의 수	고급	데이터 보호	N	-
버킷당 평균 크로스 계정 복제 규칙	-	동일한 계정 내 복제를 위한 복제 규칙의 평균 수	고급	데이터 보호	Y	sum(SameAccountReplicationRuleCount) / sum(DistinctNumberOfBuckets)

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
크로스 계정 복제 규칙 수	CrossAccountReplicationRuleCount	크로스 계정 복제를 위한 복제 규칙의 수	고급	데이터 보호	N	-
버킷당 평균 크로스 계정 복제 규칙	-	크로스 계정 복제를 위한 복제 규칙의 평균 수	고급	데이터 보호	Y	sum(CrossAccountReplicationRuleCount)/sum(DistinctNumberOfBuckets)
잘못된 대상 복제 규칙 수	InvalidDestinationReplicationRuleCount	잘못된 복제 대상이 있는 복제 규칙의 수	고급	데이터 보호	N	-

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
버킷당 평균 잘못된 대상 복제 규칙	-	잘못된 복제 대상이 있는 복제 규칙의 평균 수	고급	데이터 보호	Y	sum(InvalidReplicationRuleCount)/sum(DistinctNumberOfBuckets)
총 복제 규칙 수	-	총 복제 규칙 수	고급	데이터 보호	Y	-
버킷당 평균 복제 규칙	-	총 평균 복제 규칙 수	고급	데이터 보호	Y	sum(all replication rule count metrics)/sum(DistinctNumberOfBuckets)

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
객체 소유권 버킷 소유자 강제 적용 버킷 수	ObjectOwnershipBucketOwnerEnforcedBucketCount	객체 소유권에 버킷 소유자 강제 적용 설정을 사용하여 액세스 제어 목록(ACL)을 비활성화한 버킷의 총수	무료	액세스 관리	N	-
객체 소유권 버킷 소유자 강제 적용 버킷 비율	-	객체 소유권에 버킷 소유자 강제 적용 설정을 사용하여 ACL을 비활성화한 버킷의 백분율	무료	액세스 관리	Y	sum(ObjectOwnershipBucketOwnerEnforcedBucketCount)/ sum (Distinct NumberOfBuckets)
객체 소유권 버킷 소유자 선호 버킷 수	ObjectOwnershipBucketOwnerPreferredBucketCount	객체 소유권에 버킷 소유자 선호 설정을 사용하는 버킷의 총수	무료	액세스 관리	N	-

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
객체 소유권 버킷 소유자 선호 버킷 비율	-	객체 소유권에 버킷 소유자 선호 설정을 사용하는 버킷의 백분율	무료	액세스 관리	Y	sum(ObjectOwnershipBucketOwnerPreferredBucketCount)/sum(DistinctNumberOfBuckets)
객체 소유권 객체 작성자 버킷 수	ObjectOwnershipObjectWriterBucketCount	객체 소유권에 객체 작성자 설정을 사용하는 버킷의 총수	무료	액세스 관리	N	-
객체 소유권 객체 작성자 버킷 비율	-	객체 소유권에 객체 작성자 설정을 사용하는 버킷의 백분율	무료	액세스 관리	Y	sum(ObjectOwnershipObjectWriterBucketCount)/sum(DistinctNumberOfBuckets)

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
전송 가속화가 활성화된 버킷 수	TransferAccelerationEnabledBucketCount	전송 가속화가 활성화된 버킷의 총 수	무료	성능	N	-
전송 가속화가 활성화된 버킷 비율	-	전송 가속화가 활성화된 버킷의 백분율	무료	성능	Y	sum(TransferAccelerationEnabledBucketCount)/sum(DistinctNumberOfBuckets)
이벤트 알림 활성화 버킷 수	EventNotificationEnabledBucketCount	이벤트 알림이 활성화된 버킷의 총 수	무료	이벤트	N	
이벤트 알림 활성화 버킷 비율	-	이벤트 알림이 활성화된 버킷의 백분율	무료	이벤트	Y	sum(EventNotificationEnabledBucketCount)/sum(DistinctNumberOfBuckets)

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
모든 요청	AllRequests	생성된 요청의 총수	고급	활동	N	-
GET 요청	GetRequests	생성된 GET 요청의 총수	고급	활동	N	-
Put 요청	PutRequests	생성된 PUT 요청의 총수	고급	활동	N	-
Head 요청	HeadRequests	생성된 HEAD 요청의 총수	고급	활동	N	-
Delete 요청	DeleteRequests	생성된 DELETE 요청의 총수	고급	활동	N	-
List 요청	ListRequests	생성된 LIST 요청의 총수	고급	활동	N	-
Post 요청	PostRequests	생성된 POST 요청의 총수	고급	활동	N	-
Select 요청	SelectRequests	S3 Select 요청의 총수	고급	활동	N	-
Select 스캔 바이트	SelectScannedBytes	스캔된 S3 Select 바이트의 수	고급	활동	N	-
Select 반환 바이트	SelectReturnedBytes	반환된 S3 Select 바이트의 수	고급	활동	N	-

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
다운로드된 바이트	BytesDownloaded	다운로드된 바이트의 수	고급	활동	N	-
검색 비율(%)	-	다운로드된 바이트의 백분율	고급	활동	Y	sum(BytesDownloaded)/sum(StorageBytes)
업로드된 바이트	BytesUploaded	업로드된 바이트 수	고급	활동	N	-
수집 비율(%)	-	다운로드된 바이트의 백분율	고급	활동	Y	sum(BytesUploaded)/sum(StorageBytes)
4xx 오류	4xxErrors	HTTP 4xx 상태 코드의 총수	고급	활동	N	-
5xx 오류	5xxErrors	HTTP 5xx 상태 코드의 총수	고급	활동	N	-
총 오류	-	모든 4xx 및 5xx 오류의 합계	고급	활동	Y	sum(4xxErrors) + sum(5xxErrors)

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
오류 발생률(%)	-	총 요청의 백분율로 나타낸 4xx 및 5xx 오류의 총수	고급	활동	Y	sum(TotalErrors)/sum(TotalRequests)
200 OK 상태 수	200OKStatusCount	200 OK 상태 코드의 총수	고급	세부 상태 코드	N	-
200 OK 상태 비율(%)	-	총 요청의 백분율로 나타낸 200 OK 상태 코드의 총수	고급	세부 상태 코드	Y	sum(200OKStatusCount)/sum(AllRequests)
206 일부 콘텐츠 상태 수	206PartialContentStatusCount	206 일부 콘텐츠 상태 코드의 총수	고급	세부 상태 코드	N	-

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
206 일부 콘텐츠 상태 비율 (%)	-	총 요청의 백분율로 나타낸 206 일부 콘텐츠 상태 코드의 총수	고급	세부 상태 코드	Y	sum(206PartialContentStatusCount)/sum(AllRequests)
400 잘못된 요청 오류 수	400BadRequestErrorCount	400 잘못된 요청 상태 코드의 총수	고급	세부 상태 코드	N	-
400 잘못된 요청 오류 비율	-	총 요청의 백분율로 나타낸 400 잘못된 요청 상태 코드의 총수	고급	세부 상태 코드	Y	sum(400BadRequestErrorCount)/sum(AllRequests)
403 금지됨 오류 수	403ForbiddenErrorCount	403 금지됨 상태 코드의 총수	고급	세부 상태 코드	N	-

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
403 금지됨 오류 비율(%)	-	총 요청의 백분율로 나타낸 403 금지됨 상태 코드의 총수	고급	세부 상태 코드	Y	sum(403ForbiddenErrorCount) / sum(AllRequests)
404 찾을 수 없음 오류 수	404NotFoundErrorCount	404 찾을 수 없음 상태 코드의 총수	고급	세부 상태 코드	N	-
404 찾을 수 없음 오류 비율(%)	-	총 요청의 백분율로 나타낸 404 찾을 수 없음 상태 코드의 총수	고급	세부 상태 코드	Y	sum(404NotFoundErrorCount)/ sum(AllRequests)
500 내부 서버 오류 수	500InternalServerErrorCount	500 내부 서버 오류 상태 코드의 총수	고급	세부 상태 코드	N	-

메트릭 이름	CloudWatch와 내보내기	설명	티어[1]	카테고리[2]	파생 여부	파생된 지표 공식
500 내부 서버 오류 비율(%)	-	총 요청의 백분율로 나타낸 500 내부 서버 오류 상태 코드의 총수	고급	세부 상태 코드	Y	sum(500InternalServerErrorCount)/sum(AllRequests)
503 서비스 사용 불가 오류 수	503ServiceUnavailableErrorCount	503 서비스 사용 불가 상태 코드의 총수	고급	세부 상태 코드	N	-
503 서비스 사용 불가 오류 비율(%)	-	총 요청의 백분율로 나타낸 503 서비스 사용 불가 상태 코드의 총수	고급	세부 상태 코드	Y	sum(503ServiceUnavailableErrorCount)/sum(AllRequests)

[1] 모든 프리 티어 스토리지 지표는 Storage Lens 그룹 수준에서 사용할 수 있습니다. Storage Lens 그룹 수준에서는 고급 티어 지표를 사용할 수 없습니다.

[2] 규칙 수 지표와 버킷 설정 지표는 접두사 수준에서 사용할 수 없습니다.

Amazon S3 Storage Lens 권한 설정

Amazon S3 스토리지 렌즈를 사용하려면 S3 스토리지 렌즈 작업에 액세스할 수 있도록 AWS Identity and Access Management(IAM)에 새 권한이 있어야 합니다. 이러한 권한을 부여하려면 자격 증명 기반 IAM 정책을 사용할 수 있습니다. 권한을 부여하려면 이 정책을 IAM 사용자, 그룹 또는 역할에 연결해야 합니다. 이러한 권한에는 S3 Storage Lens를 활성화 또는 비활성화하거나 S3 Storage Lens 대시보드 또는 구성에 액세스할 수 있는 기능이 포함될 수 있습니다.

다음 두 조건 모두에 해당하는 경우가 아니면, IAM 사용자 또는 역할은 대시보드 또는 구성을 생성하거나 소유한 계정에 속해야 합니다.

- 귀하의 계정이 AWS Organizations의 구성원입니다.
- 관리 계정을 위임된 관리자로 사용하여 조직 수준의 대시보드를 생성할 수 있는 액세스 권한을 부여받았습니다.

> **Note**
>
> - 계정의 루트 사용자 자격 증명을 사용하여 Amazon S3 스토리지 렌즈 대시보드를 볼 수는 없습니다. S3 스토리지 렌즈 대시보드에 액세스하려면 신규 또는 기존 IAM 사용자에게 필요한 IAM 권한을 부여해야 합니다. 그런 다음 해당 사용자 자격 증명으로 로그인하여 S3 스토리지 렌즈 대시보드에 액세스합니다. 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하십시오.
> - Amazon S3 콘솔에서 S3 스토리지 렌즈를 사용하려면 여러 권한이 필요할 수 있습니다. 예를 들어, 콘솔에서 대시보드를 편집하려면 다음 권한이 필요합니다.
>   - `s3:ListStorageLensConfigurations`
>   - `s3:GetStorageLensConfiguration`
>   - `s3:PutStorageLensConfiguration`

주제

- S3 스토리지 렌즈를 사용하기 위한 계정 권한 설정
- S3 스토리지 렌즈 그룹을 사용하기 위한 계정 권한 설정
- 에서 Amazon S3 스토리지 렌즈를 사용하기 위한 권한 설정AWS Organizations

S3 스토리지 렌즈를 사용하기 위한 계정 권한 설정

S3 Storage Lens 대시보드 및 Storage Lens 대시보드 구성을 생성하고 관리하려면 수행하려는 작업에 따라 다음과 같은 권한이 있어야 합니다.

다음 테이블에서는 Amazon S3 Storage Lens 관련 IAM 권한을 보여줍니다.

작업	IAM 권한

Amazon S3 콘솔에서 S3 스토리지 렌즈 대시보드를 생성하거나 업데이트합니다.	`s3:ListStorageLensConfigurations`
	`s3:GetStorageLensConfiguration`
	`s3:GetStorageLensConfigurationTagging`
	`s3:PutStorageLensConfiguration`
	`s3:PutStorageLensConfigurationTagging`

Amazon S3 콘솔에서 S3 스토리지 렌즈 대시보드의 태그를 가져옵니다.	`s3:ListStorageLensConfigurations` `s3:GetStorageLensConfigurationTagging`

Amazon S3 콘솔에서 S3 스토리지 렌즈 대시보드를 봅니다.	`s3:ListStorageLensConfigurations`
	`s3:GetStorageLensConfiguration`
	`s3:GetStorageLensDashboard`

Amazon S3 콘솔에서 S3 스토리지 렌즈 대시보드를 삭제합니다.	`s3:ListStorageLensConfigurations`
	`s3:GetStorageLensConfiguration`
	`s3:DeleteStorageLensConfiguration`

AWS CLI 또는 AWS SDK를 사용하여 S3 스토리지 렌즈 구성을 생성하거나 업데이트합니다.	`s3:PutStorageLensConfiguration`

작업	IAM 권한
	`s3:PutStorageLensConfigurationTagging`
AWS CLI 또는 AWS SDK를 사용하여 S3 스토리지 렌즈 구성의 태그를 가져옵니다.	`s3:GetStorageLensConfigurationTagging`
AWS CLI 또는 AWS SDK를 사용하여 S3 스토리지 렌즈 구성을 봅니다.	`s3:GetStorageLensConfiguration`
AWS CLI 또는 AWS SDK를 사용하여 S3 스토리지 렌즈 구성을 삭제합니다.	`s3:DeleteStorageLensConfiguration`

Note

- IAM 정책에서 리소스 태그를 사용하여 권한을 관리할 수 있습니다.
- 이러한 권한이 있는 IAM 사용자 또는 역할은 객체를 읽거나 나열할 수 있는 직접적인 권한이 없는 버킷 및 접두사의 지표를 볼 수 있습니다.
- 접두사 수준 지표가 활성화된 S3 Storage Lens 대시보드의 경우 선택한 접두사 경로가 객체 키와 일치하면 대시보드에 객체 키가 다른 접두사로 표시될 수 있습니다.
- 계정의 버킷에 저장되는 지표 내보내기의 경우, IAM 정책의 기존 `s3:GetObject` 권한을 사용하여 권한이 부여됩니다. 마찬가지로, AWS Organizations 엔터티의 경우, 조직 관리 계정 또는 위임된 관리자 계정은 IAM 정책을 사용하여 조직 수준의 대시보드 및 구성에 대한 액세스 권한을 관리할 수 있습니다.

S3 스토리지 렌즈 그룹을 사용하기 위한 계정 권한 설정

S3 Storage Lens 그룹을 사용하면 접두사, 접미사, 객체 태그, 객체 크기 또는 객체 수명을 기반으로 버킷 내 스토리지의 분포를 이해할 수 있습니다. Storage Lens 그룹을 대시보드에 연결하여 집계된 지표를 볼 수 있습니다.

Storage Lens 그룹을 사용하려면 특정 권한이 필요합니다. 자세한 내용은 the section called “스토리지 렌즈 그룹 권한” 섹션을 참조하세요.

에서 Amazon S3 스토리지 렌즈를 사용하기 위한 권한 설정AWS Organizations

Amazon S3 스토리지 렌즈를 사용하여 AWS Organizations 계층 구조의 일부인 계정 모두에 대한 스토리지 지표와 사용량 데이터를 수집할 수 있습니다. 다음 테이블에서는 조직에서 S3 Storage Lens를 사용하는 것과 관련된 작업 및 권한을 보여줍니다.

작업	IAM 권한
조직의 S3 스토리지 렌즈에 대한 트러스트된 액세스를 사용 설정합니다.	`organizations:EnableAWSServiceAccess`
조직의 S3 스토리지 렌즈에 대한 트러스트된 액세스를 사용 중지합니다.	`organizations:DisableAWSServiceAccess`
위임된 관리자를 등록하여 조직에 대한 S3 스토리지 렌즈 대시보드 또는 구성을 생성합니다.	`organizations:RegisterDelegatedAdministrator`
위임된 관리자의 등록을 취소하여 조직에 대한 S3 스토리지 렌즈 대시보드 또는 구성을 더 이상 생성할 수 없도록 합니다.	`organizations:DeregisterDelegatedAdministrator`

S3 스토리지 렌즈 조직 전체 구성을 생성할 수 있는 추가 권한을 부여합니다.	`organizations:DescribeOrganization`
	`organizations:ListAccounts`
	`organizations:ListAWSServiceAccessForOrganization`
	`organizations:ListDelegatedAdministrators`
	`iam:CreateServiceLinkedRole`

콘솔과 API를 사용하여 Amazon S3 스토리지 렌즈 작업

Amazon S3 스토리지 렌즈는 객체 스토리지 사용 및 활동에 대한 조직 전반의 가시성을 확보하는 데 사용할 수 있는 클라우드 스토리지 분석 기능입니다. S3 스토리지 렌즈 지표를 사용하여 조직 전반의 스토리지 용량 또는 가장 빠르게 증가하는 버킷과 접두사 등의 요약 인사이트를 생성할 수 있습니다.

또한 S3 스토리지 렌즈 지표를 사용하여 비용 최적화 기회를 식별하고, 데이터 보호 및 보안 모범 사례를 구현하고, 애플리케이션 워크로드의 성능을 개선할 수 있습니다. 예를 들어 S3 수명 주기 규칙이 없는 버킷을 식별하여 7일이 넘게 경과한 미완료 멀티파트 업로드를 만료시킬 수 있습니다. 또한 S3 복제 또는 S3 버전 관리 사용과 같은 데이터 보호 모범 사례를 따르지 않는 버킷을 식별할 수 있습니다. S3 스토리지 렌즈는 또한 지표를 분석해 스토리지 비용을 최적화하고 데이터 보호에 대한 모범 사례를 적용하는 데 사용할 수 있는 상황별 권장 사항을 제공합니다.

S3 스토리지 렌즈는 지표를 집계하고 Amazon S3 콘솔 버킷 페이지의 계정 스냅샷 섹션에 이 정보를 표시합니다. S3 스토리지 렌즈는 또한 인사이트와 추세를 시각화하고, 이상치에 플래그를 지정하고, 스토리지 비용 최적화와 데이터 보호 모범 사례 적용을 위한 권장 사항을 수신하는 데 사용할 수 있는 대화형 대시보드를 제공합니다. 대시보드에 있는 드릴다운 옵션을 통해 조직, 계정, AWS 리전, 스토리지 클래스, 버킷, 접두사 또는 스토리지 렌즈 그룹 수준에서 인사이트를 생성하거나 시각화할 수 있습니다. 또한 CSV 또는 Parquet 형식의 일일 지표 내보내기를 S3 버킷으로 전송할 수 있습니다.

이 섹션에는 S3 스토리지 렌즈 구성을 생성하고 업데이트하고 보는 예제 그리고 해당 기능과 관련된 작업을 수행하는 예제가 포함되어 있습니다. 사용자가AWS Organizations에서 S3 스토리지 렌즈를 사용하는 경우, 이러한 예제에서 해당 사용 사례도 볼 수 있습니다. 예제에서 변수 값을 사용자 고유의 값으로 바꿉니다.

주제

- Amazon S3 Storage Lens 대시보드 만들기
- Amazon S3 Storage Lens 대시보드 업데이트
- Amazon S3 Storage Lens 대시보드 비활성화
- Amazon S3 Storage Lens 대시보드 삭제
- Amazon S3 Storage Lens 대시보드 나열
- Amazon S3 Storage Lens 대시보드 구성 세부 정보 보기
- S3 Storage Lens로 AWS 리소스 태그 관리
- Amazon S3 스토리지 렌즈 사용을 위한 도우미 파일

Amazon S3 Storage Lens 대시보드 만들기

AWS Organizations의 조직 또는 계정 내의 특정 AWS 리전이나 버킷으로 범위를 지정할 수 있는 추가 S3 Storage Lens 사용자 지정 대시보드를 만들 수 있습니다.

> **Note**
>
> 대시보드 구성을 업데이트하는 경우 올바르게 표시하거나 시각화하는 데 최장 48시간이 걸릴 수 있습니다.

S3 콘솔 사용

다음 단계에 따라 Amazon S3 콘솔에서 Amazon S3 스토리지 렌즈 대시보드를 생성합니다.

1단계: 대시보드 범위 정의

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 전환하려는 리전을 선택합니다.
3. 왼쪽 탐색 창의 S3 스토리지 렌즈에서 대시보드를 선택합니다.
4. 대시보드 생성을 선택합니다.
5. 대시보드(Dashboard) 페이지의 일반(General) 섹션에서 다음을 수행합니다.

   a. 대시보드의 홈 리전을 확인합니다. 홈 리전은 이 Storage Lens 대시보드에 대한 구성 및 지표가 저장되는 AWS 리전입니다.

   b. 대시보드 이름을 입력합니다.

   대시보드 이름은 65자 미만이어야 하며 특수 문자나 공백을 포함할 수 없습니다.

   > **Note**
   >
   > 대시보드를 생성한 후에는 이 대시보드 이름을 변경할 수 없습니다.

   c. 원하면 대시보드에 태그(Tags)를 추가하도록 선택할 수 있습니다. 태그를 사용하여 대시보드에 대한 권한을 관리하고 S3 스토리지 렌즈에 대한 비용을 추적할 수 있습니다.

   자세한 내용은 IAM 사용 설명서의 리소스 태그를 사용한 액세스 제어 및 AWS Billing 사용 설명서의 AWS에서 생성된 비용 할당 태그를 참조하십시오.

> **Note**
>
> 대시보드 구성에 최대 50개의 태그를 추가할 수 있습니다.

6. 대시보드 범위(Dashboard scope) 섹션에서 다음을 수행합니다.

   a. S3 스토리지 렌즈가 대시보드에 포함하거나 제외할 리전과 버킷을 선택합니다.

   b. S3 스토리지 렌즈에 포함하거나 제외하도록 사용자가 선택한 리전에서 버킷을 선택합니다. 버킷을 포함하거나 제외할 수 있지만, 둘 다 할 수는 없습니다. 조직 수준의 대시보드를 생성할 때는 이 옵션을 사용할 수 없습니다.

> **Note**
>
> - 리전과 버킷을 포함하거나 제외할 수 있습니다. 이 옵션은 조직의 구성원 계정에서 조직 수준의 대시보드를 생성할 때만 리전으로 제한됩니다.
> - 포함하거나 제외할 버킷을 최대 50개까지 선택할 수 있습니다.

2단계: 지표 선택 구성

1. 지표 선택(Metrics selection) 섹션에서 이 대시보드에 대해 집계할 지표 유형을 선택합니다.

   - 버킷 수준에서 집계되고 14일 동안 쿼리에 사용할 수 있는 무료 지표를 포함하려면 Free metrics(무료 지표)를 선택합니다.
   - 고급 지표 및 기타 고급 옵션을 활성화하려면 Advanced metrics and recommendations(고급 지표 및 권장 사항)를 선택합니다. 이러한 옵션에는 고급 접두사 집계, Amazon CloudWatch 게시 및 상황별 권장 사항이 포함됩니다. 데이터는 15개월 동안 쿼리에 사용할 수 있습니다. 고급 지표 및 권장 사항에는 추가 비용이 부과됩니다. 자세한 내용은 Amazon S3 요금을 참조하십시오.

   고급 지표 및 무료 지표에 대한 자세한 내용은 지표 선택 섹션을 참조하십시오.

2. Select advanced metrics and recommendations features(고급 지표 및 권장 사항 기능 선택)에서 활성화할 옵션을 선택합니다.

   - Advanced metrics(고급 지표)
   - CloudWatch 게시(CloudWatch publishing)
   - 접두사 집계(Prefix aggregation)

> **Important**
>
> S3 스토리지 렌즈 구성에 접두사 집계를 사용하면 접두사 수준 지표가 CloudWatch에 게시되지 않습니다. 버킷, 계정 및 조직 수준 S3 스토리지 렌즈 지표만 CloudWatch에 게시됩니다.

3. Advanced metrics(고급 지표)를 활성화한 경우 S3 스토리지 렌즈 대시보드에 표시하려는 Advanced metrics categories(고급 지표 범주)를 선택합니다.

   - 활동 지표
   - 세부 상태 코드 지표
   - 고급 비용 최적화 지표
   - 고급 데이터 보호 지표

   비용 범주에 관한 자세한 내용은 지표 범주 섹션을 참조하십시오. 전체 지표 목록은 Amazon S3 스토리지 렌즈 지표 용어집 섹션을 참조하세요.

4. 접두사 집계를 사용하도록 선택한 경우 다음을 구성합니다.

   a. 이 대시보드의 최소 접두사 임계값 크기를 선택합니다.

      예를 들어, 접두사 임계값이 5%라는 것은 버킷의 총 스토리지 크기에서 5% 이상을 차지하는 접두사가 집계됨을 의미합니다.

   b. 접두사 깊이를 선택합니다.

      이 설정은 접두사가 평가되는 최대 레벨 수를 나타냅니다. 접두사 깊이는 10보다 작아야 합니다.

   c. 접두사 구분 기호 문자를 입력합니다.

      이 값은 각 접두사 수준을 식별하는 데 사용되는 값입니다. Amazon S3의 기본값은 / 문자이지만, 사용자 스토리지 구조에서 다른 구분 기호를 사용할 수 있습니다.

(선택 사항) 3단계: 대시보드에 대한 지표 내보내기

1. Metrics export(지표 내보내기) 섹션에서 선택한 대상 버킷에 매일 배치될 지표 내보내기를 생성하려면 Enable(활성화)을 선택합니다.

지표 내보내기는 CSV 또는 Apache Parquet 형식입니다. 그리고 권장 사항 없이 S3 스토리지 렌즈 대시보드 데이터와 동일한 데이터 범위를 나타냅니다.

2. 지표 내보내기가 사용 설정되었으면 일일 지표 내보내기의 출력 형식을 CSV 또는 Apache Parquet 중에 선택합니다.

   Parquet는 플랫 열 형식으로 중첩된 데이터를 저장하는 Hadoop의 오픈 소스 파일 형식입니다.

3. 지표 내보내기에 사용할 대상 S3 버킷을 선택합니다.

   S3 스토리지 렌즈 대시보드의 현재 계정에서 버킷을 선택할 수 있습니다. 또는 대상 버킷 권한과 대상 버킷 소유자 계정 ID가 있으면 다른 AWS 계정을 선택할 수 있습니다.

4. 대상 S3 버킷(형식: `s3://`*`bucket-name`*`/`*`prefix`*)을 선택합니다.

   버킷은 S3 스토리지 렌즈 대시보드의 홈 리전에 있어야 합니다. S3 콘솔에는 Amazon S3가 대상 버킷 정책에 추가할 대상 버킷 권한이 표시됩니다. Amazon S3는 대상 버킷의 버킷 정책을 업데이트하여 S3가 해당 버킷에 데이터를 배치할 수 있게 합니다.

5. (선택 사항) 지표 내보내기에 서버 측 암호화를 활성화하려면 Specify an encryption key(암호화 키 지정)를 선택합니다. 그러고 나서 암호화 유형을 Amazon S3 관리형 키(SSE-S3) 또는 AWS Key Management Service 키(SSE-KMS) 중에서 선택합니다.

   Amazon S3 관리형 키(SSE-S3)와 AWS Key Management Service(AWS KMS) 키(SSE-KMS) 중에 선택할 수 있습니다.

6. (선택 사항) AWS KMS 키를 지정하려면 KMS 키를 선택하거나 키 Amazon 리소스 이름(ARN) 을 입력해야 합니다.

   고객 관리형 키를 선택하면 AWS KMS 키 정책을 사용하여 암호화할 권한을 S3 스토리지 렌즈에 부여해야 합니다. 자세한 내용은 AWS KMS key를 사용하여 지표 내보내기 암호화 섹션을 참조하세요.

7. 대시보드 생성(Create dashboard)을 선택합니다.

AWS CLI 사용

Example

다음 예제 명령에서는 태그가 있는 Amazon S3 Storage Lens 구성을 만듭니다. 이러한 예시를 사용하려면 *`user input placeholders`*를 실제 정보로 대체하십시오.

```
aws s3control put-storage-lens-configuration --account-id=111122223333 --
config-id=example-dashboard-configuration-id --region=us-east-1 --storage-lens-
configuration=file://./config.json --tags=file://./tags.json
```

Example

다음 예제 명령에서는 태그가 없는 Amazon S3 Storage Lens 구성을 만듭니다. 이러한 예시를 사용하려면 *user input placeholders*를 실제 정보로 대체하십시오.

```
aws s3control put-storage-lens-configuration --account-id=222222222222 --config-
id=your-configuration-id --region=us-east-1 --storage-lens-configuration=file://./
config.json
```

Java용 AWS SDK 사용

Example – Amazon S3 Storage Lens 구성 만들기 및 업데이트

다음 예제에서는 SDK for Java에서 Amazon S3 Storage Lens 구성을 만들고 업데이트합니다.

```
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.AccountLevel;
import com.amazonaws.services.s3control.model.ActivityMetrics;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.CloudWatchMetrics;
import com.amazonaws.services.s3control.model.Format;
import com.amazonaws.services.s3control.model.Include;
import com.amazonaws.services.s3control.model.OutputSchemaVersion;
import com.amazonaws.services.s3control.model.PrefixLevel;
import com.amazonaws.services.s3control.model.PrefixLevelStorageMetrics;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.S3BucketDestination;
import com.amazonaws.services.s3control.model.SSES3;
import com.amazonaws.services.s3control.model.SelectionCriteria;
import com.amazonaws.services.s3control.model.StorageLensAwsOrg;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensDataExport;
import com.amazonaws.services.s3control.model.StorageLensDataExportEncryption;
```

```
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateAndUpdateDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "111122223333";
        String exportAccountId = "Destination Account ID";
        String exportBucketArn = "arn:aws:s3:::destBucketName"; // The destination
 bucket for your metrics export must be in the same Region as your S3 Storage Lens
 configuration.
        String awsOrgARN = "arn:aws:organizations::123456789012:organization/o-
abcdefgh";
        Format exportFormat = Format.CSV;

        try {
            SelectionCriteria selectionCriteria = new SelectionCriteria()
                    .withDelimiter("/")
                    .withMaxDepth(5)
                    .withMinStorageBytesPercentage(10.0);
            PrefixLevelStorageMetrics prefixStorageMetrics = new
 PrefixLevelStorageMetrics()
                    .withIsEnabled(true)
                    .withSelectionCriteria(selectionCriteria);
            BucketLevel bucketLevel = new BucketLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withAdvancedCostOptimizationMetrics(new
 AdvancedCostOptimizationMetrics().withIsEnabled(true))
                    .withAdvancedDataProtectionMetrics(new
 AdvancedDataProtectionMetrics().withIsEnabled(true))
                    .withDetailedStatusCodesMetrics(new
 DetailedStatusCodesMetrics().withIsEnabled(true))
                    .withPrefixLevel(new
 PrefixLevel().withStorageMetrics(prefixStorageMetrics));
            AccountLevel accountLevel = new AccountLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withAdvancedCostOptimizationMetrics(new
 AdvancedCostOptimizationMetrics().withIsEnabled(true))
```

```
                    .withAdvancedDataProtectionMetrics(new
AdvancedDataProtectionMetrics().withIsEnabled(true))
                    .withDetailedStatusCodesMetrics(new
DetailedStatusCodesMetrics().withIsEnabled(true))
                    .withBucketLevel(bucketLevel);

            Include include = new Include()
                    .withBuckets(Arrays.asList("arn:aws:s3:::bucketName"))
                    .withRegions(Arrays.asList("us-west-2"));

            StorageLensDataExportEncryption exportEncryption = new
StorageLensDataExportEncryption()
                    .withSSES3(new SSES3());
            S3BucketDestination s3BucketDestination = new S3BucketDestination()
                    .withAccountId(exportAccountId)
                    .withArn(exportBucketArn)
                    .withEncryption(exportEncryption)
                    .withFormat(exportFormat)
                    .withOutputSchemaVersion(OutputSchemaVersion.V_1)
                    .withPrefix("Prefix");
            CloudWatchMetrics cloudWatchMetrics = new CloudWatchMetrics()
                    .withIsEnabled(true);
            StorageLensDataExport dataExport = new StorageLensDataExport()
                    .withCloudWatchMetrics(cloudWatchMetrics)
                    .withS3BucketDestination(s3BucketDestination);

            StorageLensAwsOrg awsOrg = new StorageLensAwsOrg()
                    .withArn(awsOrgARN);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withInclude(include)
                    .withDataExport(dataExport)
                    .withAwsOrg(awsOrg)
                    .withIsEnabled(true);

            List<StorageLensTag> tags = Arrays.asList(
                    new StorageLensTag().withKey("key-1").withValue("value-1"),
                    new StorageLensTag().withKey("key-2").withValue("value-2")
            );

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
```

```
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
                    .withTags(tags)
            );
        } 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();
        }
    }
}
```

스토리지에 대한 가시성을 높이기 위해 하나 이상의 S3 Storage Lens 그룹을 생성하여 대시보드에 연결할 수 있습니다. S3 Storage Lens 그룹은 접두사, 접미사, 객체 태그, 객체 크기, 객체 수명 또는 이러한 필터의 조합을 기반으로 객체에 대해 사용자 지정 정의된 필터입니다.

S3 Storage Lens 그룹을 사용하면 데이터 레이크와 같은 대규모 공유 버킷에 대한 세분화된 가시성을 확보하여 정보에 입각한 비즈니스 결정을 내릴 수 있습니다. 예를 들어, 버킷 내 또는 여러 버킷에 걸친 개별 프로젝트 및 비용 센터의 특정 객체 그룹으로 스토리지 사용을 분류하여 스토리지 할당을 간소화하고 비용 보고를 최적화할 수 있습니다.

S3 Storage Lens 그룹을 사용하려면 고급 지표 및 권장 사항을 사용하도록 대시보드를 업그레이드해야 합니다. S3 스토리지 렌즈 사용에 대한 자세한 내용은 the section called “S3 Storage Lens 그룹 사용” 섹션을 참조하십시오.

Amazon S3 Storage Lens 대시보드 업데이트

Amazon S3 Storage Lens 기본 대시보드는 `default-account-dashboard`입니다. 이 대시보드는 Amazon S3에서 사전 구성되어, 사용자가 콘솔에서 전체 계정의 집계된 무료 및 고급 지표에 대해 요약된 인사이트와 추세를 시각화하도록 도와줍니다. 기본 대시보드의 구성 범위는 수정할 수 없지만, 무료 지표에서 유료 고급 지표 및 권장 사항으로 지표 선택을 업그레이드하거나 선택적 지표 내보내기를

구성하거나 심지어는 기본 대시보드를 비활성화할 수는 있습니다. 기본 대시보드는 삭제할 수 없으며 비활성화할 수만 있습니다. 자세한 내용은 S3 콘솔 사용 섹션을 참조하세요.

S3 콘솔 사용

Amazon S3 콘솔에서 Amazon S3 스토리지 렌즈 대시보드를 업데이트하려면 다음 단계를 따르십시오.

1단계: 대시보드 범위 업데이트

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 스토리지 렌즈, Dashboards(스토리지 렌즈, 대시보드)를 선택합니다.
3. 편집할 대시보드를 선택한 다음 편집(Edit)을 선택합니다.

   Edit dashboard(대시보드 수정) 페이지가 열립니다.

   > **Note**
   >
   > 다음 항목은 변경할 수 없습니다.
   >
   > - 대시보드 이름
   > - 홈 리전
   > - 전체 계정의 스토리지로 범위가 지정된 기본 대시보드의 대시보드 범위

4. (선택 사항) 대시보드 구성 페이지의 General(일반) 섹션에서 대시보드에 태그를 업데이트하고 추가할 수 있습니다.

   태그를 사용하여 대시보드에 대한 권한을 관리하고 S3 스토리지 렌즈에 대한 비용을 추적할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 리소스 태그를 사용한 액세스 제어 및 AWS Billing 사용 설명서의 AWS에서 생성된 비용 할당 태그를 참조하십시오.

   > **Note**
   >
   > 대시보드 구성에 최대 50개의 태그를 추가할 수 있습니다.

5. 대시보드 범위(Dashboard scope) 섹션에서 다음을 수행합니다.

   a. S3 스토리지 렌즈가 대시보드에 포함하거나 제외할 리전과 버킷을 업데이트합니다.

> **Note**
>
> - 리전과 버킷을 포함하거나 제외할 수 있습니다. 이 옵션은 조직의 구성원 계정에서 조직 수준의 대시보드를 생성할 때만 리전으로 제한됩니다.
> - 포함하거나 제외할 버킷을 최대 50개까지 선택할 수 있습니다.

b. S3 스토리지 렌즈에 포함하거나 제외하도록 사용자가 선택한 리전에서 버킷을 업데이트합니다. 버킷을 포함하거나 제외할 수 있지만, 둘 다 할 수는 없습니다. 조직 수준의 대시보드를 생성할 때는 이 옵션이 표시되지 않습니다.

2단계: 지표 선택 업데이트

1. 지표 선택(Metrics selection) 섹션에서 이 대시보드에 대해 집계할 지표 유형을 선택합니다.

   - 버킷 수준에서 집계되고 14일 동안 쿼리에 사용할 수 있는 무료 지표를 포함하려면 Free metrics(무료 지표)를 선택합니다.
   - 고급 지표 및 기타 고급 옵션을 활성화하려면 Advanced metrics and recommendations(고급 지표 및 권장 사항)를 선택합니다. 이러한 옵션에는 고급 접두사 집계, Amazon CloudWatch 게시 및 상황별 권장 사항이 포함됩니다. 데이터는 15개월 동안 쿼리에 사용할 수 있습니다. 고급 지표 및 권장 사항에는 추가 비용이 부과됩니다. 자세한 내용은 Amazon S3 요금을 참조하십시오.

     고급 지표 및 무료 지표에 대한 자세한 내용은 지표 선택 섹션을 참조하십시오.

2. Select advanced metrics and recommendations features(고급 지표 및 권장 사항 기능 선택)에서 활성화할 옵션을 선택합니다.

   - Advanced metrics(고급 지표)
   - CloudWatch 게시(CloudWatch publishing)
   - 접두사 집계(Prefix aggregation)

> **Important**
>
> S3 스토리지 렌즈 구성에 접두사 집계를 사용하면 접두사 수준 지표가 CloudWatch에 게시되지 않습니다. 버킷, 계정 및 조직 수준 S3 스토리지 렌즈 지표만 CloudWatch에 게시됩니다.

3. Advanced metrics(고급 지표)를 활성화한 경우 S3 스토리지 렌즈 대시보드에 표시하려는 Advanced metrics categories(고급 지표 범주)를 선택합니다.

   - 활동 지표
   - 세부 상태 코드 지표
   - 고급 비용 최적화 지표
   - 고급 데이터 보호 지표

   지표 범주에 대한 자세한 내용은 지표 범주 섹션을 참조하십시오. 전체 지표 목록은 Amazon S3 스토리지 렌즈 지표 용어집 섹션을 참조하세요.

4. 접두사 집계를 사용하도록 선택한 경우 다음을 구성합니다.

   a. 이 대시보드의 최소 접두사 임계값 크기를 선택합니다.

      예를 들어, 접두사 임계값이 5%라는 것은 버킷의 총 스토리지 크기에서 5% 이상을 차지하는 접두사가 집계됨을 의미합니다.

   b. 접두사 깊이를 선택합니다.

      이 설정은 접두사가 평가되는 최대 레벨 수를 나타냅니다. 접두사 깊이는 10보다 작아야 합니다.

   c. 접두사 구분 기호 문자를 입력합니다.

      이 값은 각 접두사 수준을 식별하는 데 사용되는 값입니다. Amazon S3의 기본값은 / 문자이지만, 사용자 스토리지 구조에서 다른 구분 기호를 사용할 수 있습니다.

(선택 사항) 3단계: 대시보드에 대한 지표 내보내기

1. Metrics export(지표 내보내기) 섹션에서 선택한 대상 버킷에 매일 배치될 지표 내보내기를 생성하려면 Enable(활성화)을 선택합니다. 지표 내보내기를 비활성화하려면 Disable(비활성화)을 선택합니다.

   지표 내보내기는 CSV 또는 Apache Parquet 형식입니다. 그리고 권장 사항 없이 S3 스토리지 렌즈 대시보드 데이터와 동일한 데이터 범위를 나타냅니다.

2. 사용 설정되었으면 일일 지표 내보내기의 출력 형식을 CSV 또는 Apache Parquet 중에 선택합니다.

   Parquet는 플랫 열 형식으로 중첩된 데이터를 저장하는 Hadoop의 오픈 소스 파일 형식입니다.

3. 지표 내보내기에 사용할 대상 S3 버킷을 선택합니다.

   S3 스토리지 렌즈 대시보드의 현재 계정에서 버킷을 선택할 수 있습니다. 또는 대상 버킷 권한과 대상 버킷 소유자 계정 ID가 있으면 다른 AWS 계정을 선택할 수 있습니다.

4. 대상 S3 버킷(형식: `s3://bucket-name/prefix`)을 선택합니다.

   버킷은 S3 스토리지 렌즈 대시보드의 홈 리전에 있어야 합니다. S3 콘솔에는 Amazon S3가 대상 버킷 정책에 추가할 대상 버킷 권한이 표시됩니다. Amazon S3는 대상 버킷의 버킷 정책을 업데이트하여 S3가 해당 버킷에 데이터를 배치할 수 있게 합니다.

5. (선택 사항) 지표 내보내기에 서버 측 암호화를 활성화하려면 Specify an encryption key(암호화 키 지정)를 선택합니다. 그러고 나서 암호화 유형을 Amazon S3 관리형 키(SSE-S3) 또는 AWS Key Management Service 키(SSE-KMS) 중에서 선택합니다.

   Amazon S3 관리형 키(SSE-S3)와 AWS Key Management Service(AWS KMS) 키(SSE-KMS) 중에 선택할 수 있습니다.

6. (선택 사항) AWS KMS 키를 지정하려면 KMS 키를 선택하거나 키 Amazon 리소스 이름(ARN) 을 입력해야 합니다. AWS KMS 키에서 다음 방법 중 하나로 KMS 키를 지정합니다.

   - 사용 가능한 KMS 키 목록에서 AWS KMS keys 중에서 선택을 선택하고 사용 가능한 키 목록에서 KMS 키를 선택합니다.

     AWS 관리형 키(`aws/s3`)와 고객 관리형 키가 모두 목록에 표시됩니다. 고객 관리형 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 키 및 AWS 키를 참조하세요.

     > **Note**
     >
     > AWS 관리형 키(`aws/S3`)는 S3 스토리지 렌즈를 사용한 SSE-KMS 암호화에 지원되지 않습니다.

   - KMS 키 ARN을 입력하려면 AWS KMS key ARN 입력을 선택하고 나타나는 필드에 KMS 키 ARN을 입력합니다.
   - AWS KMS 콘솔에서 고객 관리형 키를 생성하려면 KMS 키 생성을 선택합니다.

     고객 관리형 키를 선택하면 AWS KMS 키 정책을 사용하여 암호화할 권한을 S3 스토리지 렌즈에 부여해야 합니다. 자세한 내용은 AWS KMS key를 사용하여 지표 내보내기 암호화 섹션을 참조하세요.

AWS KMS key 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 키 생성을 참조하십시오.

7. 변경 사항 저장을 선택합니다.

스토리지에 대한 가시성을 높이기 위해 하나 이상의 S3 Storage Lens 그룹을 생성하여 대시보드에 연결할 수 있습니다. S3 Storage Lens 그룹은 접두사, 접미사, 객체 태그, 객체 크기, 객체 수명 또는 이러한 필터의 조합을 기반으로 객체에 대해 사용자 지정 정의된 필터입니다.

S3 Storage Lens 그룹을 사용하면 데이터 레이크와 같은 대규모 공유 버킷에 대한 세분화된 가시성을 확보하여 정보에 입각한 비즈니스 결정을 내릴 수 있습니다. 예를 들어, 버킷 내 또는 여러 버킷에 걸친 개별 프로젝트 및 비용 센터의 특정 객체 그룹으로 스토리지 사용을 분류하여 스토리지 할당을 간소화하고 비용 보고를 최적화할 수 있습니다.

S3 Storage Lens 그룹을 사용하려면 고급 지표 및 권장 사항을 사용하도록 대시보드를 업그레이드해야 합니다. S3 스토리지 렌즈 사용에 대한 자세한 내용은 the section called “S3 Storage Lens 그룹 사용” 섹션을 참조하십시오.

AWS CLI 사용

Example

다음 예제 명령에서는 Amazon S3 Storage Lens 대시보드 구성을 업데이트합니다. 이러한 예시를 사용하려면 *user input placeholders*를 실제 정보로 대체하십시오.

```
aws s3control put-storage-lens-configuration --account-id=111122223333 --
config-id=example-dashboard-configuration-id --region=us-east-1 --storage-lens-
configuration=file://./config.json --tags=file://./tags.json
```

Java용 AWS SDK 사용

Example – 고급 지표 및 권장 사항으로 Amazon S3 Storage Lens 구성 업데이트

다음 예제에서는 SDK for Java에서 고급 지표 및 권장 사항으로 기본 S3 Storage Lens 구성을 업데이트하는 방법을 보여줍니다.

```
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.AccountLevel;
import com.amazonaws.services.s3control.model.ActivityMetrics;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.Format;
import com.amazonaws.services.s3control.model.Include;
import com.amazonaws.services.s3control.model.OutputSchemaVersion;
import com.amazonaws.services.s3control.model.PrefixLevel;
import com.amazonaws.services.s3control.model.PrefixLevelStorageMetrics;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.S3BucketDestination;
import com.amazonaws.services.s3control.model.SSES3;
import com.amazonaws.services.s3control.model.SelectionCriteria;
import com.amazonaws.services.s3control.model.StorageLensAwsOrg;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensDataExport;
import com.amazonaws.services.s3control.model.StorageLensDataExportEncryption;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class UpdateDefaultConfigWithPaidFeatures {

    public static void main(String[] args) {
        String configurationId = "default-account-dashboard"; // This configuration ID
 cannot be modified.
        String sourceAccountId = "111122223333";

        try {
            SelectionCriteria selectionCriteria = new SelectionCriteria()
                    .withDelimiter("/")
                    .withMaxDepth(5)
                    .withMinStorageBytesPercentage(10.0);
            PrefixLevelStorageMetrics prefixStorageMetrics = new
 PrefixLevelStorageMetrics()
                    .withIsEnabled(true)
                    .withSelectionCriteria(selectionCriteria);
            BucketLevel bucketLevel = new BucketLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
```

```
                    .withPrefixLevel(new
 PrefixLevel().withStorageMetrics(prefixStorageMetrics));
            AccountLevel accountLevel = new AccountLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withBucketLevel(bucketLevel);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );

        } 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();
        }
    }
}
```

> **Note**
>
> 고급 지표 및 권장 사항에 대해서는 추가 요금이 부과됩니다. 자세한 내용은 고급 지표 및 권장 사항을 참조하십시오.

Amazon S3 Storage Lens 대시보드 비활성화

Amazon S3 콘솔에서 Amazon S3 Storage Lens 대시보드를 비활성화할 수 있습니다. 대시보드를 비활성화하면 나중에 지표를 생성할 수 없습니다. 다시 활성화하면 쉽게 재시작할 수 있도록 비활성화된 대시보드가 여전히 구성 정보를 보존합니다. 사용 중지된 대시보드는 쿼리에 더 이상 사용할 수 없을 때까지 기록 데이터를 유지합니다.

S3 콘솔 사용

Amazon S3 콘솔에서 Amazon S3 Storage Lens 대시보드를 비활성화하려면 다음 단계를 따르세요.

Amazon S3 Storage Lens 대시보드를 비활성화하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens(스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. 대시보드(Dashboards) 목록에서 사용 중지할 대시보드를 선택한 후, 목록 맨 위에서 사용 중지(Disable)를 선택합니다.
4. 확인 페이지에서 텍스트 필드에 대시보드 이름을 입력하여 대시보드를 비활성화하겠다고 확인한 후 확인을 선택합니다.

Amazon S3 Storage Lens 대시보드 삭제

기본 대시보드는 삭제할 수 없습니다. 그러나 비활성화는 가능합니다. 생성한 대시보드를 삭제하기 전에 다음을 고려하십시오.

- 대시보드를 삭제하는 대신, 대시보드를 사용 중지하여 나중에 다시 사용 설정하도록 설정할 수 있습니다. 자세한 내용은 S3 콘솔 사용 섹션을 참조하세요.
- 대시보드를 삭제하면 대시보드와 연결된 모든 구성 설정이 삭제됩니다.
- 대시보드를 삭제하면 모든 기록 지표 데이터를 사용할 수 없게 됩니다. 이 기록 데이터는 여전히 15개월 동안 보존됩니다. 이 데이터에 다시 액세스하려면 삭제된 리전과 동일한 홈 리전에서 이름이 같은 대시보드를 생성합니다.

S3 콘솔 사용

Amazon S3 콘솔에서 Amazon S3 Storage Lens 대시보드를 삭제할 수 있습니다. 그러나 대시보드를 삭제하면 나중에 지표를 생성할 수 없습니다.

Amazon S3 스토리지 렌즈 대시보드 삭제

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 스토리지 렌즈(스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. 대시보드(Dashboards) 목록에서 삭제할 대시보드를 선택한 후, 목록 맨 위에서 삭제(Delete)를 선택합니다.
4. 대시보드 삭제 페이지에서 텍스트 필드에 대시보드 이름을 입력하여 대시보드를 삭제하겠다고 확인합니다. 그 다음 확인을 선택합니다.

AWS CLI 사용

Example

다음 예제에서는 S3 Storage Lens 구성을 삭제합니다. 이러한 예시를 사용하려면 *user input placeholders*를 실제 정보로 대체하십시오.

```
aws s3control delete-storage-lens-configuration --account-id=222222222222 --region=us-east-1 --config-id=your-configuration-id
```

Java용 AWS SDK 사용

Example – Amazon S3 Storage Lens 대시보드 구성 삭제

다음 예제에서는 SDK for Java를 사용하여 S3 Storage Lens 구성을 삭제하는 방법을 보여줍니다.

```
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.DeleteStorageLensConfigurationRequest;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class DeleteDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
```

```
        String sourceAccountId = "111122223333";
        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.deleteStorageLensConfiguration(new
 DeleteStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
            );
        } 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();
        }
    }
}
```

Amazon S3 Storage Lens 대시보드 나열

S3 콘솔 사용

S3 Storage Lens 대시보드를 나열하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens로 이동합니다.
3. 대시보드를 선택합니다. 이제 AWS 계정에서 대시보드를 볼 수 있습니다.

AWS CLI 사용

Example

다음 예제 명령은 AWS 계정에 있는 S3 Storage Lens 대시보드를 나열합니다. 이러한 예시를 사용하려면 *user input placeholders*를 실제 정보로 대체하십시오.

```
aws s3control list-storage-lens-configurations --account-id=222222222222 --region=us-east-1 --next-token=abcdefghij1234
```

Example

아래 예제에서는 다음 토큰이 없는 S3 Storage Lens 구성을 나열합니다. 이러한 예시를 사용하려면 *user input placeholders*를 실제 정보로 대체하십시오.

```
aws s3control list-storage-lens-configurations --account-id=222222222222 --region=us-east-1
```

Java용 AWS SDK 사용

Example – S3 Storage Lens 대시보드 구성 나열

다음 예제에서는 SDK for Java에 S3 Storage Lens 구성을 나열하는 방법을 보여줍니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다." 각 예제 설명에

```
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.ListStorageLensConfigurationEntry;
import com.amazonaws.services.s3control.model.ListStorageLensConfigurationsRequest;

import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class ListDashboard {

    public static void main(String[] args) {
        String sourceAccountId = "111122223333";
        String nextToken = "nextToken";

        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
```

```
                    .withRegion(US_WEST_2)
                    .build();

            final List<ListStorageLensConfigurationEntry> configurations =
                    s3ControlClient.listStorageLensConfigurations(new
 ListStorageLensConfigurationsRequest()
                            .withAccountId(sourceAccountId)
                            .withNextToken(nextToken)
                    ).getStorageLensConfigurationList();

            System.out.println(configurations.toString());
        } 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();
        }
    }
}
```

Amazon S3 Storage Lens 대시보드 구성 세부 정보 보기

Amazon S3 콘솔, AWS CLI, SDK for Java에서 Amazon S3 Storage Lens 대시보드를 볼 수 있습니다.

S3 콘솔 사용

S3 Storage Lens 대시보드 구성 세부 정보를 보려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens로 이동합니다.
3. 대시보드를 선택합니다.
4. 대시보드 목록에서 보고자 하는 대시보드를 클릭합니다. 이제 Storage Lens 대시보드의 세부 정보를 확인합니다.

AWS CLI 사용

Example

다음 예제에서는 구성 세부 정보를 볼 수 있도록 S3 Storage Lens 구성을 검색합니다. 이러한 예시를 사용하려면 *user input placeholders*를 실제 정보로 대체하십시오.

```
aws s3control get-storage-lens-configuration --account-id=222222222222 --config-
id=your-configuration-id --region=us-east-1
```

Java용 AWS SDK 사용

Example - S3 Storage Lens 구성 검색 및 보기

다음 예제에서는 구성 세부 정보를 볼 수 있도록 SDK for Java에서 S3 Storage Lens 구성을 검색하는 방법을 보여줍니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
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.GetStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.GetStorageLensConfigurationResult;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class GetDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "111122223333";

        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();
```

```
            final StorageLensConfiguration configuration =
                    s3ControlClient.getStorageLensConfiguration(new
 GetStorageLensConfigurationRequest()
                            .withAccountId(sourceAccountId)
                            .withConfigId(configurationId)
                    ).getStorageLensConfiguration();

            System.out.println(configuration.toString());
        } 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();
        }
    }
}
```

S3 Storage Lens로 AWS 리소스 태그 관리

각 Amazon S3 Storage Lens 대시보드는 고유한 Amazon 리소스 이름(ARN)을 갖고 있는 AWS 리소스로 계산됩니다. 따라서 Storage Lens 대시보드를 구성할 때 필요에 따라 대시보드에 AWS 리소스 태그를 추가할 수 있습니다. 각 Storage Lens 대시보드에 최대 50개의 태그를 추가할 수 있습니다. 태그가 있는 Storage Lens 대시보드를 만들려면 다음과 같은 S3 Storage Lens 권한이 있어야 합니다.

- `s3:ListStorageLensConfigurations`
- `s3:GetStorageLensConfiguration`
- `s3:GetStorageLensConfigurationTagging`
- `s3:PutStorageLensConfiguration`
- `s3:PutStorageLensConfigurationTagging`

AWS 리소스 태그를 사용하여 부서, 사업부 또는 프로젝트에 따라 리소스를 분류할 수 있습니다. 이는 동일한 유형의 리소스가 많을 때 유용합니다. 태그를 적용하면 지정한 태그를 기반으로 특정 S3 Storage Lens 대시보드를 빠르게 식별할 수 있습니다. 또한 태그를 사용하여 비용을 추적 및 할당할 수 있습니다.

또한, Storage Lens 대시보드에 AWS 리소스 태그를 추가하면 속성 기반 액세스 제어(ABAC)가 활성화됩니다. ABAC는 태그와 같은 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. 또한 IAM 정책

의 리소스 태그를 지정하는 조건을 사용하여 해당 리소스의 태그를 기반으로 AWS 리소스에 대한 액세스를 제어할 수 있습니다.

태그 키와 값을 편집할 수 있으며 언제든지 리소스에서 태그를 제거할 수 있습니다. 또한 다음 제한 사항을 고려해야 합니다.

- 태그 키와 태그 값은 대/소문자를 구분합니다.
- 해당 리소스에 대해 키가 기존 태그와 동일한 태그를 추가하는 경우 새 값이 이전 값을 덮어씁니다.
- 리소스를 삭제하면 리소스 태그도 삭제됩니다.
- AWS 리소스 태그에 비공개 또는 민감한 데이터를 포함하지 마십시오.
- 시스템 태그(`aws:`로 시작하는 태그 키 포함)는 지원되지 않습니다.
- 각 태그 키의 길이는 128자를 초과할 수 없습니다. 각 태그 값의 길이는 256자를 초과할 수 없습니다.

다음 예제에서는 Storage Lens 대시보드에서 AWS 리소스 태그를 사용하는 방법을 보여줍니다.

주제

- Storage Lens 대시보드에 AWS 리소스 태그 추가
- 스토리지 렌즈 대시보드의 AWS 리소스 태그 검색
- Storage Lens 대시보드 태그 업데이트
- S3 Storage Lens 대시보드에서 AWS 리소스 태그 삭제

Storage Lens 대시보드에 AWS 리소스 태그 추가

다음 예제에서는 S3 Storage Lens 대시보드에 AWS 리소스 태그를 추가하는 방법을 보여줍니다. Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 및 AWS SDK for Java을 사용하여 리소스 태그를 추가할 수 있습니다.

S3 콘솔 사용

Storage Lens 대시보드에 AWS 리소스 태그를 추가하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 왼쪽 탐색 패널의 Storage Lens로 이동합니다.
3. 대시보드를 선택합니다.

4. 업데이트하려는 Storage Lens 대시보드의 라디오 버튼을 선택합니다. 그런 다음 편집(Edit)을 선택합니다.
5. 일반에서 태그 추가를 선택합니다.
6. 태그 추가 페이지에서 새 키-값 쌍을 추가합니다.

> **Note**
>
> 기존 태그와 동일한 키가 있는 새 태그를 추가하면 이전 태그 값을 덮어씁니다.

7. (선택 사항) 새 태그를 두 개 이상 추가하려면 태그 추가를 다시 선택하여 새 항목을 계속 추가합니다. Storage Lens 대시보드에 최대 50개의 AWS 리소스 태그를 추가할 수 있습니다.
8. (선택 사항) 새로 추가된 항목을 제거하려면 제거하려는 태그 옆의 제거를 선택합니다.
9. 변경 사항 저장을 선택합니다.

AWS CLI 사용

Example

다음 예제 명령은 S3 Storage Lens 대시보드 구성에 태그를 추가합니다. 이러한 예시를 사용하려면 *user input placeholders*를 실제 정보로 대체하십시오.

```
aws s3control put-storage-lens-configuration-tagging --account-id=222222222222 --
region=us-east-1 --config-id=your-configuration-id --tags=file://./tags.json
```

Java용 AWS SDK 사용

다음 예제에서는 SDK for Java의 Amazon S3 Storage Lens 구성에 태그를 추가합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

Example - S3 Storage Lens 구성에 태그 추가

```
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.PutStorageLensConfigurationTaggingRequest;
```

```
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class PutDashboardTagging {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "111122223333";

        try {
            List<StorageLensTag> tags = Arrays.asList(
                    new StorageLensTag().withKey("key-1").withValue("value-1"),
                    new StorageLensTag().withKey("key-2").withValue("value-2")
            );

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfigurationTagging(new
 PutStorageLensConfigurationTaggingRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withTags(tags)
            );
        } 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();
        }
    }
}
```

스토리지 렌즈 대시보드의 AWS 리소스 태그 검색

다음 예제에서는 S3 Storage Lens 대시보드의 AWS 리소스 태그를 검색하는 방법을 보여줍니다. Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 및 AWS SDK for Java를 사용하여 리소스 태그를 가져올 수 있습니다.

S3 콘솔 사용

Storage Lens 대시보드의 AWS 리소스 태그를 검색하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens로 이동합니다.
3. 대시보드를 선택합니다.
4. 보려는 Storage Lens 대시보드 구성의 라디오 버튼을 선택합니다. 그런 다음 대시보드 구성 보기를 선택합니다.
5. 태그에서 대시보드와 연결된 태그를 검토합니다.
6. (선택 사항) 새 태그를 추가하려면 편집을 선택합니다. 그런 다음 태그 추가를 선택합니다. 태그 추가 페이지에서 새 키-값 쌍을 추가합니다.

   > **Note**
   >
   > 기존 태그와 동일한 키가 있는 새 태그를 추가하면 이전 태그 값을 덮어씁니다.

7. (선택 사항) 새로 추가된 항목을 제거하려면 제거하려는 태그 옆의 제거를 선택합니다.
8. 변경 사항 저장을 선택합니다.

AWS CLI 사용

Example

다음 예제 명령은 S3 Storage Lens 대시보드 구성에 대한 태그를 검색합니다. 이러한 예시를 사용하려면 *user input placeholders*를 실제 정보로 대체하십시오.

```
aws s3control get-storage-lens-configuration-tagging --account-id=222222222222 --
region=us-east-1 --config-id=your-configuration-id --tags=file://./tags.json
```

Java용 AWS SDK 사용

Example – S3 Storage Lens 대시보드 구성에 대한 태그 가져오기

다음 예제에서는 SDK for Java에서 S3 Storage Lens 대시보드 구성에 대한 태그를 검색하는 방법을 보여줍니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
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.DeleteStorageLensConfigurationRequest;
import
 com.amazonaws.services.s3control.model.GetStorageLensConfigurationTaggingRequest;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class GetDashboardTagging {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "111122223333";
        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            final List<StorageLensTag> s3Tags = s3ControlClient
                    .getStorageLensConfigurationTagging(new
 GetStorageLensConfigurationTaggingRequest()
                            .withAccountId(sourceAccountId)
                            .withConfigId(configurationId)
                    ).getTags();

            System.out.println(s3Tags.toString());
        } 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();
        }
    }
}
```

Storage Lens 대시보드 태그 업데이트

다음 예제는 Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 및 AWS SDK for Java을 사용하여 Storage Lens 대시보드 태그를 업데이트하는 방법을 보여줍니다.

S3 콘솔 사용

Storage Lens 대시보드의 AWS 리소스 태그를 업데이트하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens로 이동합니다.
3. 대시보드를 선택합니다.
4. 보려는 Storage Lens 대시보드 구성의 라디오 버튼을 선택합니다. 그런 다음 대시보드 구성 보기를 선택합니다.
5. 태그에서 대시보드와 연결된 태그를 검토합니다.
6. (선택 사항) 새 태그를 추가하려면 편집을 선택합니다. 그런 다음 태그 추가를 선택합니다. 태그 추가 페이지에서 새 키-값 쌍을 추가합니다.

   > **Note**
   >
   > 기존 태그와 동일한 키가 있는 새 태그를 추가하면 이전 태그 값을 덮어씁니다.

7. (선택 사항) 새로 추가된 항목을 제거하려면 제거하려는 태그 옆의 제거를 선택합니다.
8. 변경 사항 저장을 선택합니다.

AWS CLI 사용

Example

다음 예제 명령은 기존 Amazon S3 Storage Lens 대시보드 구성에 태그를 추가하거나 교체합니다. 이러한 예시를 사용하려면 *user input placeholders*를 실제 정보로 대체하십시오.

```
aws s3control put-storage-lens-configuration-tagging --account-id=111122223333 --
config-id=example-dashboard-configuration-id --region=us-east-1 --config-id=your-
configuration-id
```

Java용 AWS SDK 사용

다음 AWS SDK for Java 예제는 기존 Storage Lens 대시보드의 AWS 리소스 태그를 업데이트합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

Example - 기존 Storage Lens 대시보드 구성의 태그 업데이트

```
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.PutStorageLensConfigurationTaggingRequest;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class PutDashboardTagging {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "111122223333";

        try {
            List<StorageLensTag> tags = Arrays.asList(
                    new StorageLensTag().withKey("key-1").withValue("value-1"),
```

```
                    new StorageLensTag().withKey("key-2").withValue("value-2")
            );

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfigurationTagging(new
 PutStorageLensConfigurationTaggingRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withTags(tags)
            );
        } 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();
        }
    }
}
```

S3 Storage Lens 대시보드에서 AWS 리소스 태그 삭제

다음 예제에서는 기존 Storage Lens 대시보드에서 AWS 리소스 태그를 삭제하는 방법을 보여줍니다. Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 및 AWS SDK for Java를 사용하여 태그를 삭제할 수 있습니다.

S3 콘솔 사용

기존 Storage Lens 대시보드에서 AWS 리소스 태그를 삭제하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens로 이동합니다.
3. 대시보드를 선택합니다.
4. 보려는 Storage Lens 대시보드 구성의 라디오 버튼을 선택합니다. 그런 다음 대시보드 구성 보기를 선택합니다.

5. 태그에서 대시보드와 연결된 태그를 검토합니다.
6. 제거할 태그 옆에 있는 제거를 선택합니다.
7. 변경 사항 저장을 선택합니다.

AWS CLI 사용

다음 AWS CLI 명령은 기존 Storage Lens 대시보드에서 AWS 리소스 태그를 삭제합니다. 이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

Example

```
aws s3control delete-storage-lens-configuration-tagging --account-id=222222222222 --
config-id=your-configuration-id --region=us-east-1
```

Java용 AWS SDK 사용

다음 AWS SDK for Java 예제에서는 *111122223333* 계정에 지정한 Amazon 리소스 이름(ARN)을 사용하여 Storage Lens 대시보드에서 AWS 리소스 태그를 삭제합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

Example - S3 Storage Lens 대시보드 구성에 대한 태그 삭제

```
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.DeleteStorageLensConfigurationTaggingRequest;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class DeleteDashboardTagging {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "111122223333";
        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
```

```
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.deleteStorageLensConfigurationTagging(new
 DeleteStorageLensConfigurationTaggingRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
            );
        } 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();
        }
    }
}
```

Amazon S3 스토리지 렌즈 사용을 위한 도우미 파일

예제에 다음 JSON 파일과 키 입력을 사용하십시오.

JSON의 S3 스토리지 렌즈 예제 구성

Example **`config.json`**

`config.json` 파일에는 S3 스토리지 렌즈 조직 수준의 고급 지표 및 권장 사항 구성에 대한 세부 정보가 들어 있습니다. 다음 예시를 사용하려면 *`user input placeholders`*를 실제 정보로 대체하십시오.

> **Note**
>
> 고급 지표 및 권장 사항에 대해서는 추가 요금이 부과됩니다. 자세한 내용은 고급 지표 및 권장 사항을 참조하십시오.

```
{
  "Id": "SampleS3StorageLensConfiguration", //Use this property to identify your S3
 Storage Lens configuration.
```

```
  "AwsOrg": { //Use this property when enabling S3 Storage Lens for AWS Organizations.
    "Arn": "arn:aws:organizations::123456789012:organization/o-abcdefgh"
  },
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "DetailedStatusCodesMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
        "IsEnabled":true
      },
      "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
      },
      "AdvancedCostOptimizationMetrics": {
        "IsEnabled":true
      },
      "DetailedStatusCodesMetrics": {
        "IsEnabled":true
      },
      "PrefixLevel":{
        "StorageMetrics":{
          "IsEnabled":true,
          "SelectionCriteria":{
            "MaxDepth":5,
            "MinStorageBytesPercentage":1.25,
            "Delimiter":"/"
          }
        }
      }
    }
  },
  "Exclude": { //Replace with "Include" if you prefer to include Regions.
    "Regions": [
      "eu-west-1"
```

```
    ],
    "Buckets": [ //This attribute is not supported for AWS Organizations-level
 configurations.
      "arn:aws:s3:::amzn-s3-demo-source-bucket"
    ]
  },
  "IsEnabled": true, //Whether the configuration is enabled
  "DataExport": { //Details about the metrics export
    "S3BucketDestination": {
      "OutputSchemaVersion": "V_1",
      "Format": "CSV", //You can add "Parquet" if you prefer.
      "AccountId": "111122223333",
      "Arn": "arn:aws:s3:::
amzn-s3-demo-destination-bucket", // The destination bucket for your metrics export
 must be in the same Region as your S3 Storage Lens configuration.
      "Prefix": "prefix-for-your-export-destination",
      "Encryption": {
        "SSES3": {}
      }
    },
    "CloudWatchMetrics": {
      "IsEnabled": true
    }
  }
}
```

JSON의 Storage Lens 그룹을 사용한 S3 Storage Lens 예제 구성

Example **config.json**

config.json 파일에는 스토리지 렌즈 그룹을 사용할 때 스토리지 렌즈 구성에 적용하려는 세부 정보가 들어 있습니다. 예제를 사용하려면 *user input placeholders*를 실제 정보로 대체하십시오.

모든 Storage Lens 그룹을 대시보드에 연결하려면 다음 구문으로 Storage Lens 구성을 업데이트하십시오.

```
{
  "Id": "ExampleS3StorageLensConfiguration",
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
```

```
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
      "IsEnabled":true
      },
    "StorageLensGroupLevel": {},
  "IsEnabled": true
}
```

Storage Lens 대시보드 구성(*slg-1* 및 *slg-2*)에 Storage Lens 그룹을 두 개만 포함하려면 다음 구문을 사용하십시오.

```
{
  "Id": "ExampleS3StorageLensConfiguration",
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
      "IsEnabled":true
      },
   "StorageLensGroupLevel": {
        "SelectionCriteria": {
            "Include": [
                "arn:aws:s3:us-east-1:111122223333:storage-lens-group/slg-1",
                "arn:aws:s3:us-east-1:444455556666:storage-lens-group/slg-2"
            ]
    },
  "IsEnabled": true
}
```

특정 Storage Lens 그룹만 대시보드 구성에 연결되지 않도록 제외하려면 다음 구문을 사용하십시오.

```
{
  "Id": "ExampleS3StorageLensConfiguration",
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
      "IsEnabled":true
      },
   "StorageLensGroupLevel": {
        "SelectionCriteria": {
            "Exclude": [
                "arn:aws:s3:us-east-1:111122223333:storage-lens-group/slg-1",
                "arn:aws:s3:us-east-1:444455556666:storage-lens-group/slg-2"
            ]
    },
  "IsEnabled": true
}
```

JSON의 S3 스토리지 렌즈 예제 태그 구성

Example **tags.json**

tags.json 파일에는 S3 스토리지 렌즈 구성에 적용하려는 태그가 들어 있습니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
[
    {
        "Key": "key1",
        "Value": "value1"
    },
    {
        "Key": "key2",
        "Value": "value2"
    }
```

```
]
```

S3 스토리지 렌즈 예제 구성 IAM 권한

Example **permissions.json** - 특정 대시보드 이름

이 정책 예시에서는 대시보드 이름이 지정된 S3 스토리지 렌즈 IAM permissions.json 파일을 보여줍니다. *value1*, *us-east-1*, *your-dashboard-name* 및 *example-account-id*를 실제 값으로 바꾸십시오.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetStorageLensConfiguration",
                "s3:DeleteStorageLensConfiguration",
                "s3:PutStorageLensConfiguration"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/key1": "value1"
                }
            },
            "Resource": "arn:aws:s3:us-east-1:example-account-id:storage-lens/your-
dashboard-name"
        }
    ]
}
```

Example **permissions.json** - 특정 대시보드 이름 없음

이 정책 예시에서는 대시보드 이름이 지정되지 않은 S3 스토리지 렌즈 IAM permissions.json 파일을 보여줍니다. *value1*, *us-east-1* 및 *example-account-id*를 사용자의 값으로 바꿉니다.

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

```
            "Action": [
                "s3:GetStorageLensConfiguration",
                "s3:DeleteStorageLensConfiguration",
                "s3:PutStorageLensConfiguration"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/key1": "value1"
                }
            },
            "Resource": "arn:aws:s3:us-east-1:example-account-id:storage-lens/*"
        }
    ]
}
```

Amazon S3 스토리지 렌즈에서 지표 보기

S3 스토리지 렌즈는 지표를 집계하고 Amazon S3 콘솔 Buckets(버킷) 페이지의 Account snapshot(계정 스냅샷) 섹션에 이 정보를 표시합니다. S3 스토리지 렌즈는 또한 인사이트와 추세를 시각화하고, 이상치에 플래그를 지정하고, 스토리지 비용 최적화와 데이터 보호 모범 사례 적용을 위한 권장 사항을 수신하는 데 사용할 수 있는 대화형 대시보드를 제공합니다. 대시보드에 있는 드릴다운 옵션을 통해 조직, 계정, AWS 리전, 스토리지 클래스, 버킷, 접두사 또는 스토리지 렌즈 그룹 수준에서 인사이트를 생성하거나 시각화할 수 있습니다. 또한 CSV 또는 Parquet 형식의 일일 지표 내보내기를 S3 버킷으로 전송할 수 있습니다.

기본적으로 모든 대시보드는 무료 지표로 구성되며, 여기에는 S3 스토리지 전반의 사용량 및 활동을 이해하고, 스토리지 비용을 최적화하고, 데이터 보호 및 액세스 관리 모범 사례를 구현하는 데 사용할 수 있는 지표가 포함됩니다. 무료 지표는 버킷 수준까지 집계됩니다. 무료 지표를 사용하는 경우 데이터는 최장 14일 동안 쿼리에 사용할 수 있습니다.

고급 지표 및 권장 사항에는 스토리지 전반의 사용량 및 활동과 스토리지 최적화를 위한 모범 사례를 자세히 파악하는 데 사용할 수 있는 다음과 같은 추가 기능이 포함되어 있습니다.

- 상황별 권장 사항(대시보드에서만 사용할 수 있음)
- 고급 지표(버킷별로 집계된 활동 지표 포함)
- 접두사 집계(Prefix aggregation)
- Storage Lens 그룹 집계
- Storage Lens 그룹 집계
- Amazon CloudWatch 게시

고급 지표 데이터는 15개월 동안 쿼리에 사용할 수 있습니다. 고급 지표와 함께 S3 스토리지 렌즈를 사용하는 경우 추가 요금이 부과됩니다. 자세한 내용은 Amazon S3 요금을 참조하십시오. 무료 및 고급 지표에 대한 자세한 정보는 지표 선택 섹션을 참조하십시오.

주제

- 대시보드에서 S3 스토리지 렌즈 지표 보기
- 데이터 내보내기를 사용하여 Amazon S3 스토리지 렌즈 지표 보기
- CloudWatch의 S3 Storage Lens 지표 모니터링
- Amazon S3 스토리지 렌즈 지표 사용 사례

대시보드에서 S3 스토리지 렌즈 지표 보기

Amazon S3 콘솔에서 S3 Storage Lens는 데이터의 인사이트와 추세를 시각화하는 데 사용할 수 있는 대화형 기본 대시보드를 제공합니다. 대시보드를 사용하여 이상치에 플래그를 지정하고, 스토리지 비용 최적화와 데이터 보호 모범 사례 적용을 위한 권장 사항을 수신할 수 있습니다. 대시보드에 있는 드릴다운 옵션을 통해 계정, 버킷, AWS 리전, 접두사 또는 스토리지 렌즈 그룹 수준에서 인사이트를 생성할 수 있습니다. S3 Storage Lens를 AWS Organizations와 함께 사용하도록 설정한 경우, 조직 수준에서 인사이트 (예: AWS Organizations 계층 구조의 일부인 계정 모두에 대한 데이터)를 생성할 수도 있습니다. 대시보드는 항상 지표를 사용할 수 있는 최신 날짜에 대해 로드됩니다.

콘솔의 S3 스토리지 렌즈 기본 대시보드의 이름은 default-account-dashboard로 지정됩니다. Amazon S3는 대시보드를 사전 구성하여 전체 계정에 대한 요약된 인사이트와 추세를 시각화하고, 이를 S3 콘솔에 매일 업데이트합니다. 기본 대시보드의 구성 범위는 수정할 수 없지만 무료 지표에서 유료 고급 지표 및 권장 사항으로 지표 선택을 업그레이드할 수 있습니다. 고급 지표 및 권장 사항을 통해 추가 지표 및 기능에 액세스할 수 있습니다. 이러한 기능에는 고급 지표 범주, 접두사 수준 집계, 상황별 권장 사항 및 Amazon CloudWatch 게시가 포함됩니다.

기본 대시보드를 비활성화할 수 있지만 삭제할 수는 없습니다. 기본 대시보드를 비활성화하면 더 이상 업데이트되지 않습니다. S3 스토리지 렌즈 또는 버킷 페이지의 계정 스냅샷에서 새로운 일일 지표를 더 이상 수신하지 않게 됩니다. 14일간의 데이터 쿼리 기간이 만료될 때까지 기본 대시보드에서 기록 데이터를 계속해서 볼 수 있습니다. 고급 지표 및 권장 사항을 활성화한 경우 이 기간은 15개월입니다. 만료 기간 내에 기본 대시보드를 다시 활성화하여 이 데이터에 액세스할 수 있습니다.

추가 S3 스토리지 렌즈 대시보드를 생성하고 AWS 리전, S3 버킷 또는 계정을 기준으로 범위를 지정할 수 있습니다. Storage Lens를 AWS Organizations에 사용할 수 있도록 설정한 경우 조직별로 대시보드 범위를 지정할 수도 있습니다. S3 스토리지 렌즈 대시보드를 생성하거나 편집할 때 대시보드 범위와 지표 선택을 정의합니다.

생성한 추가 대시보드를 비활성화하거나 삭제할 수 있습니다.

- 대시보드를 사용 중지하면 더 이상 업데이트되지 않으며, 사용자는 더 이상 새로운 일일 지표를 수신하지 않습니다. 14일간의 만료 기간이 끝날 때까지 무료 지표에 대한 기록 데이터를 계속해서 볼 수 있습니다. 해당 대시보드에 고급 지표 및 권장 사항을 활성화한 경우 이 기간은 15개월입니다. 만료 기간 내에 대시보드를 활성화하여 이 데이터에 액세스할 수 있습니다.
- 대시보드를 삭제하면 모든 대시보드 구성 설정이 손실됩니다. 사용자는 더 이상 새로운 일일 지표를 수신하지 않으며 해당 대시보드와 연결된 기록 데이터에도 액세스할 수 없게 됩니다. 삭제된 대시보드의 기록 데이터에 액세스하려면 동일한 홈 리전에서 동일한 이름의 다른 대시보드를 만들어야 합니다.

주제

- Amazon S3 스토리지 렌즈 대시보드 보기
- S3 스토리지 렌즈 대시보드 이해

Amazon S3 스토리지 렌즈 대시보드 보기

다음 절차는 S3 콘솔에서 S3 스토리지 렌즈 대시보드를 보는 방법을 보여줍니다. 대시보드를 사용하여 비용을 최적화하고, 모범 사례를 구현하고, S3 버킷에 액세스하는 애플리케이션의 성능을 개선하는 방법을 보여주는 사용 사례 기반 안내는 Amazon S3 스토리지 렌즈 지표 사용 사례 섹션을 참조하십시오.

> **Note**
>
> 계정의 루트 사용자 자격 증명을 사용하여 Amazon S3 스토리지 렌즈 대시보드를 볼 수는 없습니다. S3 스토리지 렌즈 대시보드에 액세스하려면 신규 또는 기존 IAM 사용자에게 필요한 AWS Identity and Access Management(IAM) 권한을 부여해야 합니다. 그런 다음 해당 사용자 자격 증명으로 로그인하여 S3 스토리지 렌즈 대시보드에 액세스합니다. 자세한 내용은 IAM 사용 설명서의 Amazon S3 Storage Lens 권한 설정 및 IAM의 보안 모범 사례를 참조하십시오.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens(스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. 대시보드(Dashboards) 목록에서 보고자 하는 대시보드를 선택합니다.

대시보드가 S3 스토리지 렌즈에서 열립니다. Snapshot for date(날짜에 대한 스냅샷) 섹션에는 S3 스토리지 렌즈가 최근에 지표를 수집한 날짜가 표시됩니다. 대시보드는 항상 지표를 사용할 수 있는 최신 날짜에 대해 로드됩니다.

4. (선택 사항) S3 스토리지 렌즈 대시보드의 날짜를 변경하려면 오른쪽 상단의 날짜 선택기에서 새 날짜를 선택합니다.
5. (선택 사항) 임시 필터를 적용하여 대시보드 데이터의 범위를 추가로 제한하려면 다음을 수행합니다.

   a. 필터 섹션을 확장합니다.

   b. 특정 계정, AWS 리전, 스토리지 클래스, 버킷, 접두사 또는 스토리지 렌즈 그룹별로 필터링하려면 필터링 기준으로 사용할 옵션을 선택합니다.

   > **Note**
   >
   > 접두사 필터와 Storage Lens 그룹 필터를 동시에 적용할 수는 없습니다.

   c. 필터를 업데이트하려면 적용을 선택합니다.

   d. 필터를 제거하려면 필터 옆에 있는 X를 클릭합니다.
6. S3 스토리지 렌즈 대시보드의 모든 섹션에서 특정 지표에 대한 데이터를 보려면 지표에서 지표 이름을 선택합니다.
7. S3 스토리지 렌즈 대시보드의 모든 차트 또는 시각화에서 계정, AWS 리전, 스토리지 클래스, 버킷, 프리픽스 또는 스토리지 렌즈 그룹 탭을 사용하여 더 심층적인 집계 수준으로 드릴다운할 수 있습니다. 문제 해결 예는 Amazon S3 콜드 버킷 발견을(를) 참조하세요.

S3 스토리지 렌즈 대시보드 이해

S3 스토리지 렌즈 대시보드에는 기본 Overview(개요) 탭과 각 집계 수준을 나타내는 최대 5개의 추가 탭이 있습니다.

- Accounts(계정)
- AWS 리전
- 스토리지 클래스
- 버킷
- 접두사
- Storage Lens 그룹

개요 탭에서 대시보드 데이터는 날짜 스냅샷, 추세 및 분포 및 상위 N개 개요의 세 가지 섹션으로 집계됩니다.

S3 스토리지 렌즈 대시보드에 대한 자세한 내용은 다음 섹션을 참조하십시오.

스냅샷

날짜 스냅샷 섹션은 S3 스토리지 렌즈가 선택한 날짜에 대해 집계한 요약 지표를 보여 줍니다. 이러한 요약 지표에는 다음과 같은 지표가 포함됩니다.

- 총 스토리지 - 사용된 스토리지의 총 양(바이트)
- 객체 수 - 내 AWS 계정의 총 객체 수
- 평균 객체 크기 - 평균 객체 크기입니다.
- 활성 버킷 – 계정에서 스토리지가 0바이트를 초과하는 활성 사용 중인 총 활성 버킷 수입니다.
- 계정 – 스토리지가 범위 내에 있는 계정의 수입니다. 사용자가 AWS Organizations를 사용하고 있고 S3 스토리지 렌즈에 유효한 서비스 연결 역할로 트러스트된 액세스 권한이 있는 경우가 아니면, 이 값은 1입니다. 자세한 내용은 Amazon S3 스토리지 렌즈에 대한 서비스 연결 역할 사용 섹션을 참조하세요.
- 버킷 - 계정 내 총 버킷 수

지표 데이터

스냅샷에 나타나는 각 지표에 대해 다음 데이터를 볼 수 있습니다.

- 지표 이름 - 지표의 이름입니다.
- 지표 범주 - 지표가 구성되는 범주입니다.
- 날짜 합계 - 선택한 날짜의 총수입니다.
- 변화 비율 - 마지막 스냅샷 날짜로부터의 백분율 변화입니다.
- 30일 추세 - 30일 기간 동안의 지표 변화를 보여주는 추세선입니다.
- 권장 사항 - 스냅샷에 제공된 데이터를 기반으로 하는 상황별 권장 사항입니다. 권장 사항은 고급 지표 및 권장 사항과 함께 사용할 수 있습니다. 자세한 내용은 추천 섹션을 참조하세요.

지표 범주

필요에 따라 대시보드 날짜 스냅샷 섹션을 업데이트하여 다른 범주의 지표를 표시할 수 있습니다. 추가 지표에 대한 스냅샷 데이터를 보려면 다음 지표 범주 중에서 선택할 수 있습니다.

- 비용 최적화
- 데이터 보호
- 활동(고급 지표와 함께 사용 가능)
- 액세스 관리
- 성능
- 이벤트

Snapshot for date(날짜 스냅샷) 섹션에는 각 범주에 대해 선택한 지표만 표시됩니다. 특정 범주에 대한 모든 지표를 보려면 Trends and distributions(추세 및 분포) 또는 Top N overview(상위 N개 개요) 섹션에서 지표를 선택하십시오. 비용 범주에 관한 자세한 내용은 지표 범주 섹션을 참조하십시오. 전체 S3 스토리지 렌즈 지표 목록은 Amazon S3 스토리지 렌즈 지표 용어집 섹션을 참조하십시오.

추세 및 분포

Overview(개요) 탭의 두 번째 섹션은 Trends and distribution(추세 및 분포)입니다. Trends and distributions(추세 및 분포) 섹션에서는 정의한 날짜 범위에서 비교할 두 개의 지표를 선택할 수 있습니다. Trends and distributions(추세 및 분포) 섹션은 시간 경과에 따른 두 지표 간의 관계를 보여줍니다. 이 섹션에는 추적 중인 두 추세 사이의 스토리지 클래스와 리전 분포를 확인하는데 사용할 수 있는 차트가 표시됩니다. 필요에 따라 차트 중 하나에 있는 데이터 요소를 드릴다운하여 심층 분석을 수행할 수 있습니다.

Trends and distributions(추세 및 분포) 섹션을 사용하는 방법에 대한 안내는 기본 암호화에 AWS KMS 이(가) 포함된 서버 측 암호화(SSE-KMS)를 사용하지 않는 버킷 식별 섹션을 참조하십시오.

상위 N개 개요

S3 스토리지 렌즈 대시보드의 세 번째 섹션은 상위 N개 개요(Top N overview)(오름차순 또는 내림차순으로 정렬)입니다. 이 섹션은 상위 계정 수, AWS 리전, 버킷, 프리픽스 또는 스토리지 렌즈 그룹 전반에서 선택한 지표를 표시합니다. S3 Storage Lens를 AWS Organizations와 함께 사용하도록 설정한 경우 조직 전체에서 사용자가 선택한 지표를 볼 수도 있습니다.

상위 N개 개요 섹션을 사용하는 방법에 대한 안내는 가장 큰 S3 버킷 식별 섹션을 참조하십시오.

옵션별 드릴다운 및 분석

분석 수행 시 원활한 경험을 제공하기 위해 S3 스토리지 렌즈 대시보드에는 차트 값을 선택할 때 나타나는 작업 메뉴가 제공됩니다. 이 메뉴를 사용하려면 차트 값을 선택하여 연결된 지표 값을 확인한 다음 표시되는 상자에서 다음 두 가지 옵션 중에서 선택합니다.

- Drill down(드릴다운) 작업은 선택한 값을 대시보드의 모든 탭에서 필터로 적용합니다. 그런 다음 심층 분석을 위해 해당 값을 드릴다운할 수 있습니다.
- 분석 기준 작업을 수행하면 선택한 차원 탭으로 이동되고 해당 탭 값을 필터로 적용합니다. 이러한 탭에는 계정, AWS 리전, 스토리지 클래스, 버킷, 접두사(고급 지표 및 접두사 집계가 활성화된 대시보드용), Storage Lens 그룹(고급 지표 및 Storage Lens 그룹 집계가 활성화된 대시보드용)이 포함됩니다. 분석 기준을 사용하여 심층 분석을 위해 새 차원의 컨텍스트에서 데이터를 볼 수 있습니다.

결과가 비논리적이거나 결과 값이 없는 경우에는 드릴다운 및 분석 기준 작업이 비활성화될 수 있습니다. 드릴다운 및 분석 기준 작업 둘 다 대시보드의 모든 탭에서 기존 필터 위에 필터가 적용됩니다. 필요에 따라 필터를 삭제할 수도 있습니다.

탭

Dimension level(차원 레벨) 탭은 특정 차원 내의 모든 값에 대한 세부 보기를 제공합니다. 예를 들어, AWS 리전 탭에는 모든 AWS 리전에 대한 지표가 표시되고 버킷 탭에는 모든 버킷에 대한 지표가 표시됩니다. 각 차원 탭에는 다음과 같은 4개의 섹션으로 구성된 동일한 레이아웃이 있습니다.

- 선택한 지표에 대해 지난 30일 동안 차원 내에서 상위 N개 항목을 표시하는 추세 차트입니다. 기본적으로 이 차트에는 상위 10개 항목이 표시되지만 항목 수는 최소 3개까지 줄이거나 최대 50개까지 늘릴 수 있습니다.
- 히스토그램 차트는 선택한 날짜와 지표에 대한 세로 막대형 차트를 보여줍니다. 이 차트에 표시할 항목이 많으면 가로로 스크롤해야 할 수 있습니다.
- 차원 내의 모든 항목을 표시하는 버블 분석 차트입니다. 이 차트는 x축의 첫 번째 지표와 y축의 두 번째 지표를 나타냅니다. 세 번째 지표는 거품의 크기로 표시됩니다.
- 지표 그리드 보기에는 행에 나열된 차원의 각 항목이 포함됩니다. 열은 사용 가능한 각 지표를 나타내며, 더 쉽게 탐색할 수 있도록 지표 카테고리 탭에 정리되어 있습니다.

데이터 내보내기를 사용하여 Amazon S3 스토리지 렌즈 지표 보기

Amazon S3 스토리지 렌즈 지표는 CSV 또는 Apache Parquet 형식의 지표 내보내기 파일로 매일 생성되어 계정의 S3 버킷에 배치됩니다. 거기서 지표 내보내기를 원하는 분석 도구(예: Amazon QuickSight 및 Amazon Athena)로 수집하여, 스토리지 사용량 및 활동 추세를 분석할 수 있습니다.

주제

- AWS KMS key를 사용하여 지표 내보내기 암호화
- S3 스토리지 렌즈 내보내기 매니페스트란 무엇입니까?

- Amazon S3 스토리지 렌즈 내보내기 스키마 이해

AWS KMS key를 사용하여 지표 내보내기 암호화

고객 관리형 키를 사용하여 지표 내보내기를 암호화할 수 있는 권한을 Amazon S3 스토리지 렌즈에 부여하려면 키 정책을 사용해야 합니다. KMS 키를 사용하여 S3 스토리지 렌즈 지표 내보내기를 암호화할 수 있도록 키 정책을 업데이트하려면 다음 단계를 따릅니다.

KMS 키를 사용한 데이터 암호화 권한을 S3 스토리지 렌즈에 부여

1. 고객 관리형 키를 소유한 AWS 계정을 사용하여 AWS Management Console에 로그인합니다.
2. AWS KMS 콘솔(https://console.aws.amazon.com/kms)을 엽니다.
3. AWS 리전을 변경하려면 페이지의 오른쪽 상단에 있는 리전 선택기를 사용합니다.
4. 탐색 창에서 고객 관리형 키를 선택합니다.
5. 고객 관리형 키(Customer managed keys)에서 지표 내보내기를 암호화하는 데 사용할 키를 선택합니다. AWS KMS keys는 리전별로 다르며 지표 내보내기 대상 S3 버킷과 동일한 리전에 있어야 합니다.
6. 키 정책(Key policy)에서 정책 보기로 전환(Switch to policy view)을 선택합니다.
7. 키 정책을 업데이트하려면 편집(Edit)을 선택합니다.
8. 키 정책 편집에서 기존 키 정책에 다음 키 정책을 추가합니다. 이 정책을 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
{
    "Sid": "Allow Amazon S3 Storage Lens use of the KMS key",
     "Effect": "Allow",
    "Principal": {
        "Service": "storage-lens.s3.amazonaws.com"
    },
    "Action": [
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
       "StringEquals": {
           "aws:SourceArn": "arn:aws:s3:us-east-1:source-account-id:storage-
lens/your-dashboard-name",
           "aws:SourceAccount": "source-account-id"
        }
     }
```

```
}
```

9. 변경 사항 저장을 선택합니다.

고객 관리형 키 생성 및 키 정책 사용에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 다음 주제를 참조하십시오.

- 시작하기
- AWS KMS에서 키 정책 사용

AWS KMS PUT 키 정책 API 작업(PutKeyPolicy)을 사용하면 REST API, AWS CLI 및 SDK를 사용하여 지표 내보내기를 암호화하는 데 사용할 키 정책을 고객 관리형 키에 복사할 수 있습니다.

S3 스토리지 렌즈 내보내기 매니페스트란 무엇입니까?

많은 양의 데이터가 집계되면 S3 스토리지 렌즈 일일 지표 내보내기를 여러 파일로 나눌 수 있습니다. `manifest.json` 매니페스트 파일은 해당 날짜의 지표 내보내기 파일이 있는 위치를 설명합니다. 새 내보내기가 전달될 때마다 새 매니페스트가 함께 제공됩니다. `manifest.json` 파일에 포함된 각 매니페스트는 내보내기에 대한 메타데이터 및 기타 기본 정보를 제공합니다.

매니페스트 정보에는 다음 속성이 포함됩니다.

- `sourceAccountId` – 구성 소유자의 계정 ID입니다.
- `configId` – 대시보드의 고유 식별자입니다.
- `destinationBucket` – 지표 내보내기가 배치되는 대상 버킷 Amazon 리소스 이름(ARN)입니다.
- `reportVersion` – 내보내기의 버전입니다.
- `reportDate` – 보고서의 날짜입니다.
- `reportFormat` – 보고서의 형식입니다.
- `reportSchema` – 보고서의 스키마입니다.
- `reportFiles` – 대상 버킷에 있는 내보내기 보고서 파일의 실제 목록입니다.

다음은 CSV 형식 내보내기에 대해 `manifest.json` 파일에 있는 매니페스트의 예입니다.

```
{
   "sourceAccountId":"123456789012",
```

```
   "configId":"my-dashboard-configuration-id",
   "destinationBucket":"arn:aws:s3:::amzn-s3-demo-destination-bucket",
   "reportVersion":"V_1",
   "reportDate":"2020-11-03",
   "reportFormat":"CSV",

 "reportSchema":"version_number,configuration_id,report_date,aws_account_number,aws_region,stor
   "reportFiles":[
        {
            "key":"DestinationPrefix/StorageLens/123456789012/my-dashboard-
configuration-id/V_1/reports/dt=2020-11-03/a38f6bc4-2e3d-4355-ac8a-e2fdcf3de158.csv",
            "size":1603959,
            "md5Checksum":"2177e775870def72b8d84febe1ad3574"
        }
      ]
}
```

다음은 Parquet 형식 내보내기에 대해 `manifest.json` 파일에 있는 매니페스트의 예입니다.

```
{
   "sourceAccountId":"123456789012",
   "configId":"my-dashboard-configuration-id",
   "destinationBucket":"arn:aws:s3:::amzn-s3-demo-destination-bucket",
   "reportVersion":"V_1",
   "reportDate":"2020-11-03",
   "reportFormat":"Parquet",
   "reportSchema":"message s3.storage.lens { required string version_number;
 required string configuration_id; required string report_date; required string
 aws_account_number; required string aws_region; required string storage_class;
 required string record_type; required string record_value; required string
 bucket_name; required string metric_name; required long metric_value; }",
   "reportFiles":[
      {
         "key":"DestinationPrefix/StorageLens/123456789012/my-dashboard-configuration-
id/V_1/reports/dt=2020-11-03/bd23de7c-b46a-4cf4-bcc5-b21aac5be0f5.par",
         "size":14714,
         "md5Checksum":"b5c741ee0251cd99b90b3e8eff50b944"
      }
}
```

Amazon S3 콘솔에서 또는 Amazon S3 REST API, AWS CLI 및 SDK를 사용하여 대시보드 구성의 일부로 생성되도록 지표 내보내기를 구성할 수 있습니다.

Amazon S3 스토리지 렌즈 내보내기 스키마 이해

다음 표에는 S3 스토리지 렌즈 지표 내보내기의 스키마가 나와 있습니다.

속성 이름	데이터 형식	열 이름	설명
VersionNumber	문자열	version_number	사용 중인 S3 스토리지 렌즈 지표의 버전입니다.
ConfigurationId	String	configuration_id	S3 스토리지 렌즈 구성의 configuration_id 입니다.
ReportDate	String	report_date	지표가 추적된 날짜입니다.
AwsAccountNumber	String	aws_account_number	AWS 계정 번호.
AwsRegion	문자열	aws_region	지표가 추적되는 AWS 리전입니다.
StorageClass	문자열	storage_class	해당 버킷의 스토리지 클래스입니다.
RecordType	ENUM	record_type	보고되는 아티팩트의 유형(ACCOUNT, BUCKET 또는 PREFIX)입니다.
RecordValue	String	record_value	RecordType 아티팩트의 값입니다. **Note** record_value 는 URL

속성 이름	데이터 형식	열 이름	설명
			로 인코딩됩니다.
BucketName	String	bucket_name	보고되는 버킷의 이름입니다.
MetricName	문자열	metric_name	보고되는 지표의 이름입니다.
MetricValue	Long	metric_value	보고되는 지표의 값입니다.

S3 스토리지 렌즈 지표 내보내기의 예

다음은 이 스키마를 기반으로 하는 S3 스토리지 렌즈 지표 내보내기의 예입니다.

> **Note**
>
> record_type 열에서 STORAGE_LENS_GROUP_BUCKET 또는 STORAGE_LENS_GROUP_ACCOUNT 값을 찾아 Storage Lens 그룹의 지표를 식별할 수 있습니다. record_value 열에는 Storage Lens 그룹의 Amazon 리소스 이름(ARN)이 표시됩니다(예: arn:aws:s3:us-east-1:123456789012:storage-lens-group/slg-1).

version_r	configuration_id	report_date	aws_account_number	aws_region	storage_class	record_type	record_value	bucket_name	metric_name	metric_value
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			StorageBytes	2478830621
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ObjectCount	1598962
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ReplicatedStorageBytes	20000
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ReplicatedObjectCount	20
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			EncryptedStorageBytes	2478828742
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			EncryptedObjectCount	1598961
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			DeleteMarkerObjectCount	1500
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ObjectLockEnabledStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ObjectLockEnabledObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			CurrentVersionStorageBytes	2478830621
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			CurrentVersionObjectCount	1598962
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			NonCurrentVersionStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			NonCurrentVersionObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			IncompleteMultipartUploadStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			IncompleteMultipartUploadObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	StorageBytes	29996800
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	ObjectCount	12264
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	ReplicatedStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	ReplicatedObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	EncryptedStorageBytes	29996800
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	EncryptedObjectCount	12264
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	DeleteMarkerObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	ObjectLockEnabledStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	ObjectLockEnabledObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	CurrentVersionStorageBytes	29996800
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	CurrentVersionObjectCount	12264
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	NonCurrentVersionStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	NonCurrentVersionObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	IncompleteMultipartUploadStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	IncompleteMultipartUploadObjectCount	0

다음은 S3 Storage Lens 지표 내보내기의 예입니다(Storage Lens 그룹 데이터 포함).

version_numbe	configuration_id	report_date	aws_account_num	aws_region	storage_class	record_type	record_value	bucket_name	metric_name	metric_value
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		StorageBytes	3128548856
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ObjectCount	4440
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ReplicatedStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ReplicatedObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		EncryptedStorageBytes	3128548856
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		EncryptedObjectCount	4440
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		DeleteMarkerObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ObjectLockEnabledStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ObjectLockEnabledObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		CurrentVersionStorageBytes	3128548856
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		CurrentVersionObjectCount	4440
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		NonCurrentVersionStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		NonCurrentVersionObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		IncompleteMultipartUploadStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		IncompleteMultipartUploadObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		DeleteMarkerStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ReplicatedStorageBytesSource	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ReplicatedObjectCountSource	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		IncompleteMPUStorageBytesOlderThan7Days	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		IncompleteMPUObjectCountOlderThan7Days	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	StorageBytes	676863200
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ObjectCount	3000
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ReplicatedStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ReplicatedObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	EncryptedStorageBytes	676863200
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	EncryptedObjectCount	3000
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	DeleteMarkerObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ObjectLockEnabledStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ObjectLockEnabledObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	CurrentVersionStorageBytes	676863200
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	CurrentVersionObjectCount	3000
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	NonCurrentVersionStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	NonCurrentVersionObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	IncompleteMultipartUploadStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	IncompleteMultipartUploadObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	DeleteMarkerStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ReplicatedStorageBytesSource	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ReplicatedObjectCountSource	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	IncompleteMPUStorageBytesOlderThan7Days	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	IncompleteMPUObjectCountOlderThan7Days	0

CloudWatch의 S3 Storage Lens 지표 모니터링

S3 스토리지 렌즈 지표를 Amazon CloudWatch에 게시하여 CloudWatch 대시보드에서 운영 상태에 대한 통합 보기를 생성할 수 있습니다. 또한 경보 및 트리거된 작업, 지표 수학, 이상 감지와 같은 CloudWatch 기능을 사용하여 S3 스토리지 렌즈 지표를 모니터링하고 조치를 취할 수 있습니다. 또한 CloudWatch API 작업을 사용하면 서드 파티 공급자를 포함한 애플리케이션이 S3 스토리지 렌즈 지표에 액세스할 수 있습니다. CloudWatch 기능에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하세요.

Amazon S3 콘솔, Amazon S3 REST API, AWS CLI 및 AWS SDK로 신규 또는 기존 대시보드 구성에 CloudWatch 게시 옵션을 사용할 수 있습니다. S3 Storage Lens 고급 지표 및 권장 사항으로 업그레이드된 대시보드는 CloudWatch 게시 옵션을 사용할 수 있습니다. S3 스토리지 렌즈 고급 지표 및 권장 사항 요금은 Amazon S3 요금을 참조하세요. 추가 CloudWatch 지표 게시 요금은 적용되지 않습니다. 그러나 대시보드, 경보, API 호출 등의 다른 CloudWatch 요금이 적용됩니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

S3 Storage Lens 지표는 S3 Storage Lens 구성을 소유한 계정의 CloudWatch에 게시됩니다. 고급 지표 및 권장 사항 내에서 CloudWatch 게시 옵션을 사용하면 CloudWatch에서 조직, 계정, 버킷 수준 지표에 액세스할 수 있습니다. CloudWatch에서는 접두사 수준 지표를 사용할 수 없습니다.

> **Note**
>
> S3 Storage Lens 지표는 일일 지표이며 하루에 한 번 CloudWatch에 게시됩니다. CloudWatch에서 S3 Storage Lens 지표를 쿼리할 때 쿼리 기간은 1일(86,400초)이어야 합니다. 일일 S3 Storage Lens 지표가 Amazon S3 콘솔의 S3 Storage Lens 대시보드에 표시된 후 동일한 지표가 CloudWatch에 표시되는 데 몇 시간이 걸릴 수 있습니다. S3 Storage Lens 지표에 대해 CloudWatch 게시 옵션을 처음 사용하면 지표가 CloudWatch에 게시되는 데 최대 24시간이 걸릴 수 있습니다.

CloudWatch 게시 옵션을 사용 설정한 후 다음 CloudWatch 기능으로 S3 Storage Lens 데이터를 모니터링하고 분석할 수 있습니다.

- 대시보드 - CloudWatch 대시보드를 사용하여 사용자 지정된 S3 Storage Lens 대시보드를 생성합니다. CloudWatch 대시보드를 AWS 계정에 직접 액세스할 수 없는 사람들, 팀 전체, 이해 관계자 및 조직 외부의 사람들과 공유하세요.
- 경보 및 트리거된 작업 - 지표를 감시하고 임계값 위반 시 조치를 취하는 경보를 구성합니다. 예를 들어 미완료 멀티파트 업로드 바이트가 3일 연속 1GB를 초과할 때 Amazon SNS 알림을 전송하는 경보를 구성할 수 있습니다.
- 이상 탐지 - 이상 탐지를 사용하여 지표를 지속적으로 분석하고, 정상 기준을 결정하고, 이상을 표시합니다. 지표의 예상 값을 기반으로 이상 탐지 경보를 생성할 수 있습니다. 예를 들어 객체 잠금이 활성화된 바이트의 이상을 모니터링하여 객체 잠금 설정의 무단 제거를 탐지할 수 있습니다.
- 지표 수학 – 지표 수학을 사용하여 여러 S3 Storage Lens 지표를 쿼리하고, 수학 표현식을 사용하여 이러한 지표에 기반한 새로운 시계열을 생성할 수도 있습니다. 예를 들어 새 지표를 생성하여 `StorageBytes`를 `ObjectCount`로 나눠서 평균 객체 크기를 구할 수 있습니다.

S3 Storage Lens 지표용 CloudWatch 게시 옵션에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- S3 Storage Lens 지표 및 차원
- S3 Storage Lens에 CloudWatch 게시 사용
- CloudWatch의 S3 Storage Lens 지표 작업

S3 Storage Lens 지표 및 차원

S3 Storage Lens 지표를 CloudWatch로 전송하려면 S3 Storage Lens 고급 지표 및 권장 사항 내에서 CloudWatch 게시 옵션을 사용해야 합니다. 고급 지표를 활성화하면 CloudWatch 대시보드로 다른 애플리케이션 지표와 함께 S3 스토리지 렌즈 지표를 모니터링하고 운영 상태에 대한 통합 보기를 생성할 수 있습니다. 차원을 사용하여 조직, 계정, 버킷, 스토리지 클래스, 리전 및 지표 구성 ID별로 CloudWatch의 S3 Storage Lens 지표를 필터링할 수 있습니다.

S3 Storage Lens 지표는 S3 Storage Lens 구성을 소유한 계정의 CloudWatch에 게시됩니다. 고급 지표 및 권장 사항 내에서 CloudWatch 게시 옵션을 사용하면 CloudWatch에서 조직, 계정, 버킷 수준 지표에 액세스할 수 있습니다. CloudWatch에서는 접두사 수준 지표를 사용할 수 없습니다.

> **Note**
>
> S3 Storage Lens 지표는 일일 지표이며 하루에 한 번 CloudWatch에 게시됩니다. CloudWatch에서 S3 Storage Lens 지표를 쿼리할 때 쿼리 기간은 1일(86,400초)이어야 합니다. 일일 S3 Storage Lens 지표가 Amazon S3 콘솔의 S3 Storage Lens 대시보드에 표시된 후 동일한 지표가 CloudWatch에 표시되는 데 몇 시간이 걸릴 수 있습니다. S3 Storage Lens 지표에 대해 CloudWatch 게시 옵션을 처음 사용하면 지표가 CloudWatch에 게시되는 데 최대 24시간이 걸릴 수 있습니다.

CloudWatch의 S3 Storage Lens 지표 및 차원에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- Metrics
- Dimensions

Metrics

S3 스토리지 렌즈 지표는 CloudWatch 내에서 지표로 사용할 수 있습니다. S3 Storage Lens 지표는 `AWS/S3/Storage-Lens` 네임스페이스에 게시됩니다. 이 네임스페이스는 S3 Storage Lens 지표 전용입니다. Amazon S3 버킷, 요청 및 복제 지표는 `AWS/S3` 네임스페이스에 게시됩니다.

S3 Storage Lens 지표는 S3 Storage Lens 구성을 소유한 계정의 CloudWatch에 게시됩니다. 고급 지표 및 권장 사항 내에서 CloudWatch 게시 옵션을 사용하면 CloudWatch에서 조직, 계정, 버킷 수준 지표에 액세스할 수 있습니다. CloudWatch에서는 접두사 수준 지표를 사용할 수 없습니다.

S3 Storage Lens에서 지표는 지정된 홈 리전에서만 집계되고 저장됩니다. S3 Storage Lens 지표는 S3 Storage Lens 구성에서 지정하는 홈 리전의 CloudWatch에도 게시됩니다.

CloudWatch에서 사용 가능한 지표 목록을 포함한 S3 Storage Lens 지표의 전체 목록은 Amazon S3 스토리지 렌즈 지표 용어집 섹션을 참조하세요.

> **Note**
>
> CloudWatch의 S3 Storage Lens 지표에 대한 유효한 통계는 Average입니다. CloudWatch의 통계에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 CloudWatch 통계 정의를 참조하세요.

CloudWatch의 S3 Storage Lens 지표 세부 수준

S3 Storage Lens는 조직, 계정, 버킷 및 접두사 세부 수준의 지표를 제공합니다. S3 Storage Lens는 CloudWatch에 조직, 계정 및 버킷 수준 S3 Storage Lens 지표를 게시합니다. CloudWatch에서는 접두사 수준 S3 Storage Lens 지표를 사용할 수 없습니다.

CloudWatch에서 사용할 수 있는 S3 Storage Lens 지표의 세부 수준에 대한 자세한 내용은 다음 목록을 참조하세요.

- 조직 - 조직의 멤버 계정 전체에서 집계된 지표입니다. S3 Storage Lens는 관리 계정의 CloudWatch에 멤버 계정에 대한 지표를 게시합니다.
  - 조직 및 계정 - 조직의 멤버 계정에 대한 지표입니다.
  - 조직 및 버킷 - 조직의 멤버 계정에 있는 Amazon S3 버킷에 대한 지표입니다.
- 계정(비조직 수준) – 계정의 버킷 전체에 걸쳐 집계된 지표입니다.

- 버킷(비조직 수준) - 특정 버킷에 대한 지표입니다. CloudWatch에서 S3 Storage Lens는 이러한 지표를 S3 Storage Lens 구성을 생성한 AWS 계정에 게시합니다. S3 Storage Lens는 비조직 구성에 대해서만 이러한 지표를 게시합니다.

Dimensions

S3 스토리지 렌즈는 CloudWatch에 데이터를 전송할 때 각 지표에 차원을 연결합니다. 차원은 지표의 특성을 설명하는 범주입니다. 측정기준을 사용하여 CloudWatch가 반환하는 결과를 필터링할 수 있습니다.

예를 들어 CloudWatch의 모든 S3 Storage Lens 지표에는 `configuration_id` 차원이 있습니다. 이 차원을 사용하여 특정 S3 Storage Lens 구성과 연결된 지표를 구분할 수 있습니다. `organization_id`는 조직 수준 지표를 식별합니다. CloudWatch의 차원에 대한 자세한 내용은 CloudWatch 사용 설명서의 Dimensions를 참조하세요.

지표의 세부 수준에 따라 S3 스토리지 렌즈 지표에 서로 다른 차원을 사용할 수 있습니다. 예를 들어 `organization_id` 차원을 사용하여 AWS Organizations ID로 조직 수준 지표를 필터링할 수 있습니다. 그러나 버킷 및 계정 수준 지표에는 이 차원을 사용할 수 없습니다. 자세한 내용은 차원을 사용하여 지표 필터링 섹션을 참조하세요.

S3 Storage Lens 구성에 사용할 수 있는 차원을 확인하려면 다음 표를 참조하세요.

차원	설명	버킷	Account	조직	조직 및 버킷	조직 및 계정
`configuration_id`	지표에 보고된 S3 스토리지 렌즈 구성에 대한 대시보드 이름	.	.	.	.	.
`metrics_version`	S3 스토리지 렌즈 지표의 버전. 지표 버전의 고정 값은 `1.0`입니다.	.	.	.	.	.
`organization_id`	지표의 AWS Organizations ID	.	.	.	.	.

차원	설명	버킷	계정	조직	조직 및 버킷	조직 및 계정
aws_account_number	지표에 연결된 AWS 계정	·	·	·	·	·
aws_region	지표의 AWS 리전	·	·	·	·	·
bucket_name	지표에 보고된 S3 버킷의 이름	·	·	·	·	·
storage_class	지표에 보고된 버킷에 대한 스토리지 클래스	·	·	·	·	·
record_type	지표의 세부 수준: ORGANIZATION, ACCOUNT, BUCKET	BUCKET	ACCOUNT	ORGANIZATION	·	·

S3 Storage Lens에 CloudWatch 게시 사용

S3 스토리지 렌즈 지표를 Amazon CloudWatch에 게시하여 CloudWatch 대시보드에서 운영 상태에 대한 통합 보기를 생성할 수 있습니다. 또한 경보 및 트리거된 작업, 지표 수학, 이상 감지와 같은 CloudWatch 기능을 사용하여 S3 스토리지 렌즈 지표를 모니터링하고 조치를 취할 수 있습니다. 또한 CloudWatch API 작업을 사용하면 서드 파티 공급자를 포함한 애플리케이션이 S3 스토리지 렌즈 지표에 액세스할 수 있습니다. CloudWatch 기능에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하세요.

S3 Storage Lens 지표는 S3 Storage Lens 구성을 소유한 계정의 CloudWatch에 게시됩니다. 고급 지표 및 권장 사항 내에서 CloudWatch 게시 옵션을 사용하면 CloudWatch에서 조직, 계정, 버킷 수준 지표에 액세스할 수 있습니다. CloudWatch에서는 접두사 수준 지표를 사용할 수 없습니다.

S3 콘솔, Amazon S3 REST API, AWS CLI 및 AWS SDK로 신규 또는 기존 대시보드 구성에 CloudWatch 지원을 사용할 수 있습니다. CloudWatch 게시 옵션은 S3 스토리지 렌즈 고급 지표 및 권

장 사항으로 업그레이드된 대시보드에 사용할 수 있습니다. S3 스토리지 렌즈 고급 지표 및 권장 사항 요금은 Amazon S3 요금을 참조하세요. 추가 CloudWatch 지표 게시 요금은 적용되지 않습니다. 그러나 대시보드, 경보, API 호출 등의 다른 CloudWatch 요금이 적용됩니다.

S3 Storage Lens 지표용 CloudWatch 게시 옵션을 사용하려면 다음 주제를 참조하세요.

> **Note**
>
> S3 Storage Lens 지표는 일일 지표이며 하루에 한 번 CloudWatch에 게시됩니다. CloudWatch에서 S3 Storage Lens 지표를 쿼리할 때 쿼리 기간은 1일(86,400초)이어야 합니다. 일일 S3 Storage Lens 지표가 Amazon S3 콘솔의 S3 Storage Lens 대시보드에 표시된 후 동일한 지표가 CloudWatch에 표시되는 데 몇 시간이 걸릴 수 있습니다. S3 Storage Lens 지표에 대해 CloudWatch 게시 옵션을 처음 사용하면 지표가 CloudWatch에 게시되는 데 최대 24시간이 걸릴 수 있습니다.
> 현재 S3 스토리지 렌즈 지표는 CloudWatch 스트림을 통해 사용할 수 없습니다.

S3 콘솔 사용

S3 스토리지 렌즈 대시보드를 업데이트할 때 대시보드 이름이나 홈 리전을 변경할 수 없습니다. 또한 전체 계정의 스토리지로 범위가 지정된 기본 대시보드의 범위를 변경할 수 없습니다.

CloudWatch 게시를 사용하도록 S3 스토리지 렌즈 대시보드 업데이트

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 S3 스토리지 렌즈(S3 스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. 편집할 대시보드를 선택한 다음 편집(Edit)을 선택합니다.
4. 지표 선택(Metrics selection)에서 고급 지표 및 권장 사항(Advanced metrics and recommendations)을 선택합니다.

   고급 지표 및 권장 사항을 추가 요금에 사용할 수 있습니다. 고급 지표 및 권장 사항에는 15개월의 데이터 쿼리 기간, 접두사 수준에서 집계된 사용량 지표, 버킷별로 집계된 활동 지표, CloudWatch 게시 옵션, 스토리지 비용을 최적화하고 데이터 보호 모범 사례를 적용하는 데 도움이 되는 상황별 권장 사항이 포함됩니다. 자세한 내용은 Amazon S3 요금을 참조하십시오.
5. 고급 지표 및 권장 사항 기능 선택(Select Advanced metrics and recommendations features)에서 CloudWatch 게시(CloudWatch publishing)를 선택합니다.

> **Important**
>
> 구성에서 사용량 지표에 접두사 집계를 사용하면 접두사 수준 지표가 CloudWatch에 게시되지 않습니다. 버킷, 계정 및 조직 수준 S3 Storage Lens 지표만 CloudWatch에 게시됩니다.

6. 변경 사항 저장을 선택합니다.

CloudWatch 지원을 사용하는 새 S3 Storage Lens 대시보드 생성

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 스토리지 렌즈(스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. 대시보드 생성(Create dashboard)을 선택합니다.
4. General(일반)에서 다음 구성 옵션을 정의합니다.

   a. Dashboard name(대시보드 이름)에 대시보드 이름을 입력합니다.

      대시보드 이름은 65자 미만이어야 하며 특수 문자나 공백을 포함할 수 없습니다. 대시보드를 생성한 후에는 대시보드 이름을 변경할 수 없습니다.

   b. 대시보드에 대해 홈 리전(Home Region)을 선택합니다.

      이 대시보드 범위에 포함된 모든 리전에 대한 대시보드 지표는 지정된 홈 리전에 중앙 집중식으로 저장됩니다. CloudWatch에서 S3 Storage Lens 지표는 홈 리전에서도 사용할 수 있습니다. 대시보드를 생성한 후에는 홈 리전을 변경할 수 없습니다.

5. (선택 사항) 태그를 추가하려면 태그 추가(Add tag)를 선택하고 태그 키(Key)와 값(Value)을 입력합니다.

> **Note**
>
> 대시보드 구성에 최대 50개의 태그를 추가할 수 있습니다.

6. 구성 범위를 정의합니다.

   a. 조직 수준 구성을 생성할 때 Include all accounts in your configuration(구성에 모든 계정 포함) 또는 Limit the scope to your signed-in account(로그인한 계정으로 범위 제한)를 사용하여 구성에 포함할 계정을 선택합니다.

> **Note**
>
> 모든 계정을 포함하는 조직 수준 구성을 생성할 때 버킷이 아닌 리전만 포함하거나 제외할 수 있습니다.

b. 다음에 따라 S3 스토리지 렌즈가 대시보드 구성에 포함할 리전과 버킷을 선택합니다.

- 모든 리전을 포함하려면 리전 및 버킷 포함(Include Regions and buckets)을 선택합니다.
- 특정 리전을 포함하려면 모든 리전 포함(Include all Regions)을 선택 취소합니다. 포함할 리전 선택(Choose Regions to include)에서 S3 Storage Lens가 대시보드에 포함할 리전을 선택합니다.
- 특정 버킷을 포함하려면 모든 버킷 포함(Include all buckets)을 선택 취소합니다. 포함할 버킷 선택(Choose buckets to include)에서 S3 Storage Lens가 대시보드에 포함할 버킷을 선택합니다.

> **Note**
>
> 버킷을 최대 50개까지 선택할 수 있습니다.

7. Metrics selection(지표 선택)에서 Advanced metrics and recommendations(고급 지표 및 권장 사항)를 선택합니다.

   고급 지표 및 권장 사항 요금에 대한 자세한 내용은 Amazon S3 요금 섹션을 참조하세요.

8. Select advanced metrics and recommendations features(고급 지표 및 권장 사항 기능 선택)에서 활성화할 옵션을 선택합니다.

- Advanced metrics(고급 지표)
- CloudWatch 게시(CloudWatch publishing)

> **Important**
>
> S3 스토리지 렌즈 구성에 접두사 집계를 사용하면 접두사 수준 지표가 CloudWatch에 게시되지 않습니다. 버킷, 계정 및 조직 수준 S3 스토리지 렌즈 지표만 CloudWatch에 게시됩니다.

- 접두사 집계(Prefix aggregation)

> **Note**
>
> 고급 지표 및 권장 사항 기능에 대한 자세한 내용은 지표 선택 섹션을 참조하세요.

9. Advanced metrics(고급 지표)를 활성화한 경우 S3 스토리지 렌즈 대시보드에 표시하려는 Advanced metrics categories(고급 지표 범주)를 선택합니다.

   - 활동 지표
   - 세부 상태 코드 지표
   - 고급 비용 최적화 지표
   - 고급 데이터 보호 지표

   비용 범주에 관한 자세한 내용은 지표 범주 섹션을 참조하십시오. 전체 지표 목록은 Amazon S3 스토리지 렌즈 지표 용어집 섹션을 참조하세요.

10. (선택 사항) 지표 내보내기를 구성합니다.

    지표 내보내기를 구성하는 방법에 대한 자세한 내용은 S3 콘솔 사용 단계를 참조하세요.

11. 대시보드 생성(Create dashboard)을 선택합니다.

AWS CLI 사용

다음 AWS CLI 예에서는 S3 스토리지 렌즈 조직 수준 고급 지표 및 권장 사항 구성으로 CloudWatch 게시 옵션을 사용하도록 설정합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws s3control put-storage-lens-configuration --account-id=555555555555 --config-
id=your-configuration-id --region=us-east-1 --storage-lens-configuration=file://./
config.json

config.json
{
  "Id": "SampleS3StorageLensConfiguration", //Use this property to identify your S3
 Storage Lens configuration.
  "AwsOrg": { //Use this property when enabling S3 Storage Lens for AWS Organizations.
    "Arn": "arn:aws:organizations::123456789012:organization/o-abcdefgh"
  },
  "AccountLevel": {
```

```
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "DetailedStatusCodesMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
        "IsEnabled":true //Mark this as false if you want only free metrics.
      },
      "ActivityMetrics": {
        "IsEnabled":true //Mark this as false if you want only free metrics.
      },
      "AdvancedCostOptimizationMetrics": {
        "IsEnabled":true //Mark this as false if you want only free metrics.
      },
      "DetailedStatusCodesMetrics": {
        "IsEnabled":true //Mark this as false if you want only free metrics.
      },
      "PrefixLevel":{
        "StorageMetrics":{
          "IsEnabled":true, //Mark this as false if you want only free metrics.
          "SelectionCriteria":{
            "MaxDepth":5,
            "MinStorageBytesPercentage":1.25,
            "Delimiter":"/"
          }
        }
      }
    }
  },
  "Exclude": { //Replace with "Include" if you prefer to include Regions.
    "Regions": [
      "eu-west-1"
    ],
    "Buckets": [ //This attribute is not supported for AWS Organizations-level
configurations.
      "arn:aws:s3:::amzn-s3-demo-source-bucket "
```

```
    ]
  },
  "IsEnabled": true, //Whether the configuration is enabled
  "DataExport": { //Details about the metrics export
    "S3BucketDestination": {
      "OutputSchemaVersion": "V_1",
      "Format": "CSV", //You can add "Parquet" if you prefer.
      "AccountId": "111122223333",
      "Arn": "arn:aws:s3:::amzn-s3-demo-destination-bucket", // The destination
 bucket for your metrics export must be in the same Region as your S3 Storage Lens
 configuration.
      "Prefix": "prefix-for-your-export-destination",
      "Encryption": {
        "SSES3": {}
      }
    },
    "CloudWatchMetrics": {
      "IsEnabled": true //Mark this as false if you want to export only free metrics.
    }
  }
}
```

Java용 AWS SDK 사용

```
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.AccountLevel;
import com.amazonaws.services.s3control.model.ActivityMetrics;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.CloudWatchMetrics;
import com.amazonaws.services.s3control.model.Format;
import com.amazonaws.services.s3control.model.Include;
import com.amazonaws.services.s3control.model.OutputSchemaVersion;
import com.amazonaws.services.s3control.model.PrefixLevel;
import com.amazonaws.services.s3control.model.PrefixLevelStorageMetrics;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.S3BucketDestination;
import com.amazonaws.services.s3control.model.SSES3;
```

```
import com.amazonaws.services.s3control.model.SelectionCriteria;
import com.amazonaws.services.s3control.model.StorageLensAwsOrg;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensDataExport;
import com.amazonaws.services.s3control.model.StorageLensDataExportEncryption;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateAndUpdateDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";
        String exportAccountId = "Destination Account ID";
        String exportBucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket"; //
 The destination bucket for your metrics export must be in the same Region as your S3
 Storage Lens configuration.
        String awsOrgARN = "arn:aws:organizations::123456789012:organization/o-
abcdefgh";
        Format exportFormat = Format.CSV;

        try {
            SelectionCriteria selectionCriteria = new SelectionCriteria()
                    .withDelimiter("/")
                    .withMaxDepth(5)
                    .withMinStorageBytesPercentage(10.0);
            PrefixLevelStorageMetrics prefixStorageMetrics = new
 PrefixLevelStorageMetrics()
                    .withIsEnabled(true)
                    .withSelectionCriteria(selectionCriteria);
            BucketLevel bucketLevel = new BucketLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withAdvancedCostOptimizationMetrics(new
 AdvancedCostOptimizationMetrics().withIsEnabled(true))
                    .withAdvancedDataProtectionMetrics(new
 AdvancedDataProtectionMetrics().withIsEnabled(true))
                    .withDetailedStatusCodesMetrics(new
 DetailedStatusCodesMetrics().withIsEnabled(true))
                    .withPrefixLevel(new
 PrefixLevel().withStorageMetrics(prefixStorageMetrics));
```

```
            AccountLevel accountLevel = new AccountLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withAdvancedCostOptimizationMetrics(new
AdvancedCostOptimizationMetrics().withIsEnabled(true))
                    .withAdvancedDataProtectionMetrics(new
AdvancedDataProtectionMetrics().withIsEnabled(true))
                    .withDetailedStatusCodesMetrics(new
DetailedStatusCodesMetrics().withIsEnabled(true))
                    .withBucketLevel(bucketLevel);

            Include include = new Include()
                    .withBuckets(Arrays.asList("arn:aws:s3:::amzn-s3-demo-bucket"))
                    .withRegions(Arrays.asList("us-west-2"));

            StorageLensDataExportEncryption exportEncryption = new
StorageLensDataExportEncryption()
                    .withSSES3(new SSES3());
            S3BucketDestination s3BucketDestination = new S3BucketDestination()
                    .withAccountId(exportAccountId)
                    .withArn(exportBucketArn)
                    .withEncryption(exportEncryption)
                    .withFormat(exportFormat)
                    .withOutputSchemaVersion(OutputSchemaVersion.V_1)
                    .withPrefix("Prefix");
            CloudWatchMetrics cloudWatchMetrics = new CloudWatchMetrics()
                    .withIsEnabled(true);
            StorageLensDataExport dataExport = new StorageLensDataExport()
                    .withCloudWatchMetrics(cloudWatchMetrics)
                    .withS3BucketDestination(s3BucketDestination);

            StorageLensAwsOrg awsOrg = new StorageLensAwsOrg()
                    .withArn(awsOrgARN);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withInclude(include)
                    .withDataExport(dataExport)
                    .withAwsOrg(awsOrg)
                    .withIsEnabled(true);

            List<StorageLensTag> tags = Arrays.asList(
                    new StorageLensTag().withKey("key-1").withValue("value-1"),
                    new StorageLensTag().withKey("key-2").withValue("value-2")
```

```
            );

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
                    .withTags(tags)
            );
        } 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();
        }
    }
}
```

REST API 사용

Amazon S3 REST API로 CloudWatch 게시 옵션을 사용하도록 설정하려면 PutStorageLensConfiguration을 사용합니다.

다음 단계

CloudWatch 게시 옵션을 사용하도록 설정한 후 CloudWatch의 S3 Storage Lens 지표에 액세스할 수 있습니다. CloudWatch 기능을 활용하여 CloudWatch에서 S3 Storage Lens 데이터를 모니터링하고 분석할 수도 있습니다. 자세한 정보는 다음의 주제를 참조하세요.

- S3 Storage Lens 지표 및 차원
- CloudWatch의 S3 Storage Lens 지표 작업

CloudWatch의 S3 Storage Lens 지표 작업

S3 스토리지 렌즈 지표를 Amazon CloudWatch에 게시하여 CloudWatch 대시보드에서 운영 상태에 대한 통합 보기를 생성할 수 있습니다. 또한 경보 및 트리거된 작업, 지표 수학, 이상 감지와 같은 CloudWatch 기능을 사용하여 S3 스토리지 렌즈 지표를 모니터링하고 조치를 취할 수 있습니다. 또한 CloudWatch API 작업을 사용하면 서드 파티 공급자를 포함한 애플리케이션이 S3 스토리지 렌즈 지표에 액세스할 수 있습니다. CloudWatch 기능에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하세요.

Amazon S3 콘솔, Amazon S3 REST API, AWS CLI 및 AWS SDK로 신규 또는 기존 대시보드 구성에 CloudWatch 게시 옵션을 사용할 수 있습니다. CloudWatch 게시 옵션은 S3 스토리지 렌즈 고급 지표 및 권장 사항으로 업그레이드된 대시보드에 사용할 수 있습니다. S3 스토리지 렌즈 고급 지표 및 권장 사항 요금은 Amazon S3 요금을 참조하세요. 추가 CloudWatch 지표 게시 요금은 적용되지 않습니다. 그러나 대시보드, 경보, API 호출 등의 다른 CloudWatch 요금이 적용됩니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

S3 Storage Lens 지표는 S3 Storage Lens 구성을 소유한 계정의 CloudWatch에 게시됩니다. 고급 지표 및 권장 사항 내에서 CloudWatch 게시 옵션을 사용하면 CloudWatch에서 조직, 계정, 버킷 수준 지표에 액세스할 수 있습니다. CloudWatch에서는 접두사 수준 지표를 사용할 수 없습니다.

> **Note**
>
> S3 Storage Lens 지표는 일일 지표이며 하루에 한 번 CloudWatch에 게시됩니다. CloudWatch에서 S3 Storage Lens 지표를 쿼리할 때 쿼리 기간은 1일(86,400초)이어야 합니다. 일일 S3 Storage Lens 지표가 Amazon S3 콘솔의 S3 Storage Lens 대시보드에 표시된 후 동일한 지표가 CloudWatch에 표시되는 데 몇 시간이 걸릴 수 있습니다. S3 Storage Lens 지표에 대해 CloudWatch 게시 옵션을 처음 사용하면 지표가 CloudWatch에 게시되는 데 최대 24시간이 걸릴 수 있습니다.
>
> 현재 S3 스토리지 렌즈 지표는 CloudWatch 스트림을 통해 사용할 수 없습니다.

CloudWatch에서 S3 Storage Lens 지표를 사용하는 방법에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- CloudWatch 대시보드 사용
- 경보 설정, 작업 트리거 및 이상 탐지 사용
- 차원을 사용하여 지표 필터링

- 지표 수학으로 새 지표 계산
- 그래프에서 검색 표현식 사용

CloudWatch 대시보드 사용

CloudWatch 대시보드로 다른 애플리케이션 지표와 함께 S3 Storage Lens 지표를 모니터링하고 운영 상태에 대한 통합 보기를 생성할 수 있습니다. 대시보드는 CloudWatch 콘솔에서 사용자 지정이 가능한 홈 페이지로, 단일 보기에서 리소스를 모니터링하는 데 사용할 수 있습니다.

CloudWatch에는 특정 지표 또는 차원 집합에 대한 액세스 제한을 지원하지 않는 광범위한 권한 제어 기능이 있습니다. CloudWatch에 액세스할 수 있는 계정 또는 조직의 사용자는 CloudWatch 지원 옵션이 사용되는 모든 S3 Storage Lens 구성에 대한 지표에 액세스할 수 있습니다. S3 Storage Lens에서와 같이 특정 대시보드에 대한 권한을 관리할 수 없습니다. CloudWatch 권한에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 CloudWatch 리소스에 대한 액세스 권한 관리를 참조하세요.

CloudWatch 대시보드 사용 및 권한 구성에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch 대시보드 사용과 CloudWatch 대시보드 공유를 참조하세요.

경보 설정, 작업 트리거 및 이상 탐지 사용

CloudWatch의 S3 Storage Lens 지표를 감시하고 임계값 위반 시 조치를 취하는 CloudWatch 경보를 구성할 수 있습니다. 예를 들어 미완료 멀티파트 업로드 바이트가 3일 연속 1GB를 초과할 때 Amazon SNS 알림을 전송하는 경보를 구성할 수 있습니다.

또한 이상 탐지를 사용하여 S3 Storage Lens 지표를 지속적으로 분석하고, 정상 기준을 결정하고, 이상을 표시할 수 있습니다. 지표의 기댓값에 기반하여 이상 탐지 경보를 생성할 수 있습니다. 예를 들어 객체 잠금이 활성화된 바이트의 이상을 모니터링하여 객체 잠금 설정의 무단 제거를 탐지할 수 있습니다.

자세한 내용과 예는 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch 경보 사용과 그래프의 지표에서 경보 생성을 참조하세요.

차원을 사용하여 지표 필터링

차원을 사용하여 CloudWatch 콘솔에서 S3 Storage Lens 지표를 필터링할 수 있습니다. 예를 들어 `configuration_id`, `aws_account_number`, `aws_region`, `bucket_name` 등으로 필터링할 수 있습니다.

S3 Storage Lens는 계정당 여러 대시보드 구성을 지원합니다. 즉, 서로 다른 구성이 동일한 버킷을 포함할 수 있습니다. 이러한 지표가 CloudWatch에 게시되면 버킷은 CloudWatch 내에서 중복 지표를 갖

게 됩니다. CloudWatch에서 특정 S3 Storage Lens 구성에 대한 지표만 보려면 `configuration_id` 차원을 사용합니다. `configuration_id`로 필터링하면 식별한 구성과 관련된 지표만 표시됩니다.

구성 ID로 필터링에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 사용 가능한 지표 검색을 참조하세요.

지표 수학으로 새 지표 계산

지표 수학을 사용하여 여러 S3 Storage Lens 지표를 쿼리하고, 수학 표현식을 사용하여 이러한 지표에 기반한 새로운 시계열을 생성할 수 있습니다. 예를 들어 객체 수에서 암호화된 객체를 빼서 암호화되지 않은 객체에 대한 새 지표를 생성할 수 있습니다. `StorageBytes`를 `ObjectCount`로 나누어 평균 객체 크기를 구하거나 `BytesDownloaded`를 `StorageBytes`로 나누어 하루에 액세스한 바이트 수를 구하는 지표를 생성할 수도 있습니다.

자세한 내용은 Amazon CloudWatch 사용 설명서의 지표 수학 사용을 참조하세요.

그래프에서 검색 표현식 사용

S3 스토리지 렌즈 지표를 사용하면 검색 표현식을 생성할 수 있습니다. 예를 들어 IncompleteMultipartUploadStorageBytes라는 모든 지표에 대한 검색 표현식을 생성하고 표현식에 `SUM`을 추가할 수 있습니다. 이 검색 표현식을 사용하면 스토리지의 모든 차원에서 총 미완료 멀티파트 업로드 바이트 수를 단일 지표로 볼 수 있습니다.

이 예에서는 IncompleteMultipartUploadStorageBytes라는 모든 지표에 대한 검색 표현식을 생성하는 데 사용할 구문을 보여줍니다.

```
SUM(SEARCH('{AWS/S3/Storage-
Lens,aws_account_number,aws_region,configuration_id,metrics_version,record_type,storage_class}
 MetricName="IncompleteMultipartUploadStorageBytes"', 'Average',86400))
```

이 구문에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 CloudWatch 검색 표현식 구문을 참조하세요. 검색 표현식으로 CloudWatch 그래프를 생성하려면 Amazon CloudWatch 사용 설명서의 검색 표현식이 있는 CloudWatch 그래프 생성을 참조하세요.

Amazon S3 스토리지 렌즈 지표 사용 사례

Amazon S3 스토리지 렌즈 대시보드를 사용하여 인사이트와 추세를 시각화하고, 이상치에 플래그를 지정하고, 권장 사항을 수신할 수 있습니다. S3 스토리지 렌즈 지표는 주요 사용 사례에 맞는 범주로 정리되어 있습니다. 이러한 지표를 사용하여 다음을 수행할 수 있습니다.

- 비용 최적화 기회 파악

- 데이터 보호 모범 사례 적용
- 액세스 관리 모범 사례 적용
- 애플리케이션 워크로드의 성능 향상

예를 들어, 비용 최적화와 지표를 사용하면 Amazon S3 스토리지 비용 절감을 위한 기회를 식별할 수 있습니다. 7일이 넘게 경과했으며 멀티파트 업로드가 포함된 버킷이나 비최신 버전이 누적되고 있는 버킷을 식별할 수 있습니다.

마찬가지로 데이터 보호 지표를 사용하여 조직 내 데이터 보호 모범 사례를 따르지 않는 버킷을 식별할 수 있습니다. 예를 들어 기본 암호화에 AWS Key Management Service 키(SSE-KMS)를 사용하지 않거나 S3 버전 관리가 활성화되지 않은 버킷을 식별할 수 있습니다.

S3 스토리지 렌즈 액세스 관리 지표를 사용하면 S3 객체 소유권에 대한 버킷 설정을 식별하여 액세스 제어 목록(ACL) 권한을 버킷 정책으로 마이그레이션하고 ACL을 비활성화할 수 있습니다.

S3 스토리지 렌즈 고급 지표가 활성화되어 있다면 세부 상태 코드 지표를 사용하여 액세스 또는 성능 문제를 해결하는 데 사용할 수 있는 성공 또는 실패한 요청의 수를 파악할 수 있습니다.

또한 고급 지표를 사용하면 추가 비용 최적화 및 데이터 보호 지표에 액세스하여 전체 S3 스토리지 비용을 추가로 절감하고 데이터 보호 모범 사례에 더 잘 부합할 기회를 식별할 수 있습니다. 예를 들어, 고급 비용 최적화 지표에는 7일이 넘게 경과된 미완료 멀티파트 업로드를 만료시키는 수명 주기 규칙이 없는 버킷을 식별하는 데 사용할 수 있는 수명 주기 규칙 수가 포함됩니다. 고급 데이터 보호 지표에는 복제 규칙 수가 포함됩니다.

지표 범주에 대한 자세한 내용은 지표 범주 섹션을 참조하세요. S3 스토리지 렌즈 지표의 전체 목록은 Amazon S3 스토리지 렌즈 지표 용어집 섹션을 참조하세요.

주제

- Amazon S3 Storage Lens를 사용한 스토리지 비용 최적화
- S3 Storage Lens를 사용하여 데이터 보호
- S3 스토리지 렌즈를 사용하여 객체 소유권 설정 감사
- 성능 향상을 위한 S3 스토리지 렌즈 지표 사용

Amazon S3 Storage Lens를 사용한 스토리지 비용 최적화

S3 스토리지 렌즈 비용 최적화 지표를 사용하여 S3 스토리지의 전체 비용을 줄일 수 있습니다. 비용 최적화 지표를 통해 모범 사례에 따라 Amazon S3를 비용 효율적으로 구성했는지 확인할 수 있습니다. 예를 들어, 다음과 같은 비용 최적화 기회를 식별할 수 있습니다.

- 7일이 넘게 경과한 미완료 멀티파트 업로드가 있는 버킷
- 여러 비최신 버전을 누적하고 있는 버킷
- 미완료 멀티파트 업로드를 중단하는 수명 주기 규칙이 없는 버킷
- 비최신 버전 객체를 만료시키는 수명 주기 규칙이 없는 버킷
- 객체를 다른 스토리지 클래스로 전환하기 위한 수명 주기 규칙이 없는 버킷

그런 다음 이 데이터를 사용하여 버킷에 추가 수명 주기 규칙을 추가할 수 있습니다.

다음 예제에서는 S3 스토리지 렌즈 대시보드에서 비용 최적화 지표를 사용하여 스토리지 비용을 최적화하는 방법을 보여 줍니다.

주제

- 가장 큰 S3 버킷 식별
- Amazon S3 콜드 버킷 발견
- 불완전한 멀티파트 업로드 찾기
- 최신이 아닌 보존 버전 수 축소
- 수명 주기 규칙이 없는 버킷을 식별하고 수명 주기 규칙 수 검토

가장 큰 S3 버킷 식별

요금은 S3 버킷에 저장된 객체에 대해 부과됩니다. 부과되는 요금은 객체의 크기, 객체 저장 기간 및 스토리지 클래스에 따라 다릅니다. S3 스토리지 렌즈를 사용하면 계정의 모든 버킷을 중앙에서 볼 수 있습니다. AWS Organizations-레벨 S3 Storage Lens를 구성하면 조직의 모든 계정에 속한 모든 버킷을 볼 수 있습니다. 이 대시보드 보기에서 가장 큰 버킷을 식별할 수 있습니다.

1단계: 가장 큰 버킷 식별

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens(스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. 대시보드(Dashboards) 목록에서 보고자 하는 대시보드를 선택합니다.

   대시보드가 열리면 S3 스토리지 렌즈가 최근에 지표를 수집한 날짜를 볼 수 있습니다. 대시보드는 항상 지표를 사용할 수 있는 최신 날짜에 대해 로드됩니다.
4. 선택된 날짜 범위에 대한 Total storage(전체 스토리지) 지표를 기준으로 가장 큰 버킷의 순위를 보려면 아래로 스크롤하여 Top N overview for date(날짜의 상위 N개 개요) 섹션으로 이동합니다.

정렬 순서를 전환하여 가장 작은 버킷을 표시할 수 있습니다. Metric(지표) 선택을 조정하여 사용 가능한 지표를 기준으로 버킷 순위를 매길 수도 있습니다. Top N overview for date(날짜에 대한 상위 N개 개요) 섹션에는 이전 날짜 또는 이전 주로부터의 변동 백분율과 추세를 시각화하는 스파크라인도 표시됩니다. 이 추세는 무료 지표의 경우 14일 추세이고 고급 지표 및 권장 사항의 경우 30일 추세입니다.

> **Note**
>
> S3 스토리지 렌즈 고급 지표 및 권장 사항 을 통해 15개월 동안 쿼리에 지표를 사용할 수 있습니다. 자세한 내용은 지표 선택 섹션을 참조하세요.

5. 버킷에 대한 자세한 정보를 보려면 페이지 상단으로 스크롤한 다음 Bucket(버킷) 탭을 선택하세요.

   Bucket(버킷) 탭에서는 최근 성장률, 평균 객체 크기, 가장 큰 접두사, 객체 수와 같은 세부 정보를 볼 수 있습니다.

2단계: 버킷으로 이동하여 조사

가장 큰 S3 버킷을 식별한 다음 S3 콘솔 내의 각 버킷으로 이동하여 버킷의 객체를 보고 관련 워크로드를 파악하고 버킷의 내부 소유자를 식별할 수 있습니다. 버킷 소유자로에게 연락하면 성장이 예상되는지 여부나 성장에 추가 모니터링 및 관리가 필요한지를 확인할 수 있습니다.

Amazon S3 콜드 버킷 발견

S3 Storage Lens 고급 지표가 사용 설정되어 있다면 활동 지표를 사용해 S3 버킷의 활용도를 파악할 수 있습니다. “콜드” 버킷은 스토리지가 전혀 또는 거의 액세스되지 않는 버킷입니다. 이러한 활동 부족은 일반적으로 버킷의 객체가 자주 액세스되지 않음을 나타냅니다.

활동 지표(예: GET 요청 및 다운로드 바이트 등)는 매일 버킷이 액세스되는 빈도를 나타냅니다. 액세스 패턴의 지속성을 파악하고 더 이상 액세스되지 않는 버킷을 확인하려면 몇 개월 동안 이러한 데이터의 추세를 알아야 합니다. 다운로드 바이트/총 스토리지로 계산되는 검색 비율 지표는 매일 액세스되는 버킷 내 스토리지의 비중을 나타냅니다.

> **Note**
>
> 다운로드 바이트는 같은 개체가 하루 동안 여러 번 다운로드되는 경우 중복 집계됩니다.

전제 조건

S3 스토리지 렌즈 대시보드에서 활동 지표를 보려면 S3 스토리지 렌즈 Advanced metrics and recommendations(고급 지표 및 권장 사항)를 활성화한 다음 Activity metrics(활동 지표)를 선택해야 합니다. 자세한 내용은 S3 콘솔 사용 섹션을 참조하세요.

1단계: 활성 버킷 식별

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens(스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. 대시보드(Dashboards) 목록에서 보고자 하는 대시보드를 선택합니다.
4. Bucket(버킷) 탭을 선택하고 아래로 스크롤하여 Bubble analysis by buckets for date(날짜에 대한 버킷별 거품형 분석) 섹션으로 이동합니다.

   Bubble analysis by buckets for date(날짜에 대한 버킷별 거품형 분석) 섹션에서는 X-axis(X축), Y-axis(Y축) 및 Size(크기)를 나타내는 지표 3개를 사용한 다차원 도표에 버킷을 표현할 수 있습니다.
5. 콜드 상태가 된 버킷을 찾으려면 X-axis(X축), Y-axis(Y축) 및 Size(크기)에 Total storage(총 스토리지), % retrieval rate(검색 비율) 및 Average object size(평균 객체 크기) 지표를 선택합니다.
6. Bubble analysis by buckets for date(날짜에 대한 버킷별 거품형 분석) 섹션에서 검색 비율이 0이고(또는 0에 가깝고) 상대적으로 스토리지 크기가 큰 버킷을 찾아 해당 버킷을 나타내는 거품을 선택합니다.

   보다 세부적인 인사이트를 얻을 수 있는 선택 항목이 포함된 상자가 나타납니다. 다음 중 하나를 수행합니다.

   a. 선택한 버킷에 대한 지표만 표시하도록 Bucket(버킷) 탭을 업데이트하려면 Drill down(드릴다운)을 선택한 다음 Apply(적용)를 선택합니다.

   b. 버킷 수준 데이터를 계정, AWS 리전, 스토리지 클래스 또는 버킷별로 집계하려면 Analyze by(분석 기준)를 선택한 다음 Dimension(차원)을 선택합니다. 예를 들어, 스토리지 클래스별로 집계하려면 Dimension(차원)에 Storage class(스토리지 클래스)를 선택합니다.

   자주 사용되지 않는 버킷을 찾으려면 총 스토리지, 검색 비율(%), 평균 객체 크기 지표를 사용한 거품형 분석을 구성합니다. 검색 비율이 0이고(또는 0에 가깝고) 상대적으로 스토리지 크기가 큰 버킷을 찾습니다.

대시보드의 Bucket(버킷) 탭이 업데이트되어 선택한 집계 또는 필터에 대한 데이터가 표시됩니다. 스토리지 클래스 또는 다른 차원별로 집계한 경우 대시보드에서 새 탭이 열립니다(예: Storage class(스토리지 클래스) 탭).

2단계: 콜드 버킷 조사

이때 계정 또는 조직의 콜드 버킷 소유자를 식별하고 해당 스토리지가 계속 필요한지 여부를 확인할 수 있습니다. 그런 다음 이러한 버킷에 대한 수명 주기 만료 구성을 설정하거나 Amazon S3 Glacier 스토리지 클래스 중 하나에 데이터를 보관하여 비용을 최적화할 수 있습니다.

향후 콜드 버킷 문제를 방지하려면 버킷에 대해 S3 수명 주기 구성을 사용해 데이터를 자동으로 이동하거나 S3 Intelligent-Tiering을 사용한 자동 보관을 활성화할 수 있습니다.

1단계를 사용하여 핫 버킷을 식별할 수도 있습니다. 그런 다음 이러한 버킷이 올바른 S3 스토리지 클래스를 사용하는지 확인하여 성능 및 비용 측면에서 요청을 가장 효과적으로 처리하도록 할 수 있습니다.

불완전한 멀티파트 업로드 찾기

멀티파트 업로드를 사용하면 매우 큰 객체(최대 5TB)를 하나의 파트 집합으로 업로드하여 처리량을 개선하고 네트워크 문제를 신속하게 복구할 수 있습니다. 멀티파트 업로드 프로세스가 완료되지 않는 경우 미완료 파트가 (사용할 수 없는 상태로) 버킷에 남아 있게 됩니다. 이렇게 미완료 파트가 발생할 경우 업로드 프로세스가 완료되거나 미완료 파트가 제거될 때까지 스토리지 비용이 발생합니다. 자세한 내용은 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 섹션을 참조하세요.

S3 스토리지 렌즈를 사용하면 계정 또는 조직 전체에서 7일이 넘게 경과한 미완료 멀티파트 업로드를 포함한 미완료 멀티파트 업로드 바이트 수를 파악할 수 있습니다. 미완료 멀티파트 업로드 지표의 전체 목록은 Amazon S3 스토리지 렌즈 지표 용어집 섹션을 참조하세요.

특정 일수보다 오래된 미완료 멀티파트 업로드를 만료시키도록 수명 주기 규칙을 구성하는 것이 가장 좋습니다. 미완료 멀티파트 업로드를 만료시키도록 수명 주기 규칙을 만들 때는 7일부터 시작하는 것이 좋습니다.

1단계: 미완료 멀티파트 업로드에 대한 전반적인 추세 검토

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens(스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. 대시보드(Dashboards) 목록에서 보고자 하는 대시보드를 선택합니다.

4. Snapshot for date(날짜 스냅샷) 섹션에서 Metrics categories(지표 범주) 아래의 Cost optimization(비용 최적화)을 선택합니다.

   Snapshot for date(날짜 스냅샷) 섹션이 업데이트되어 Cost optimization(비용 최적화) 지표가 표시됩니다. 여기에는 7일이 넘게 경과된 미완료 멀티파트 업로드 바이트가 포함됩니다.

   S3 스토리지 렌즈 대시보드 내 모든 차트에서 미완료 멀티파트 업로드에 대한 지표를 볼 수 있습니다. 이러한 지표를 사용하면 미완료 멀티파트 업로드 바이트가 스토리지에 미치는 영향(전체 성장 추세에 대한 기여도 포함)을 추가로 평가할 수 있습니다. 또한 Account(계정), AWS 리전, Bucket(버킷) 또는 Storage class(스토리지 클래스) 탭을 사용하여 더 심층적인 집계 수준으로 드릴다운함으로써 데이터를 심층 분석할 수 있습니다. 문제 해결 예는 Amazon S3 콜드 버킷 발견을(를) 참조하세요.

2단계: 미완료 멀티파트 업로드 바이트가 가장 많지만 미완료 멀티파트 업로드를 중단하기 위한 수명 주기 규칙이 없는 버킷 식별

전제 조건

S3 스토리지 렌즈 대시보드에서 Abort incomplete multipart upload lifecycle rule count(미완료 멀티파트 업로드 중단 수명 주기 규칙 개수) 지표를 보려면 S3 스토리지 렌즈 Advanced metrics and recommendations(고급 지표 및 권장 사항)를 활성화한 다음 Advanced cost optimization metrics(고급 비용 최적화 지표)를 선택해야 합니다. 자세한 내용은 S3 콘솔 사용 섹션을 참조하세요.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens(스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. 대시보드(Dashboards) 목록에서 보고자 하는 대시보드를 선택합니다.
4. 7일이 넘게 경과된 미완료 멀티파트 업로드가 누적되고 있는 특정 버킷을 식별하려면 Top N overview for date(날짜에 대한 상위 N개 개요) 섹션으로 이동합니다.

   Top N overview for date(날짜에 대한 상위 N개 개요) 섹션에는 기본적으로 상위 3개 버킷에 대한 지표가 표시됩니다. Top N(상위 N개) 필드에서 버킷 수를 늘리거나 줄일 수 있습니다. Top N overview for date(날짜에 대한 상위 N개 개요) 섹션에는 이전 날짜 또는 이전 주로부터의 변동 백분율과 추세를 시각화하는 스파크라인도 표시됩니다. (이 추세는 무료 지표의 경우 14일 추세이고 고급 지표 및 권장 사항의 경우 30일 추세입니다.)

> **Note**
>
> S3 스토리지 렌즈 고급 지표 및 권장 사항 을 통해 15개월 동안 쿼리에 지표를 사용할 수 있습니다. 자세한 내용은 지표 선택 섹션을 참조하세요.

5. Metric(지표)의 Cost optimization(비용 최적화) 카테고리에서 Incomplete multipart upload bytes greater than 7 days old(7일이 넘게 경과한 미완료 멀티파트 업로드 바이트)를 선택합니다.

   Top number buckets(상위 개수 버킷) 아래에서 7일이 넘게 경과된 미완료 멀티파트 업로드 스토리지 바이트가 가장 큰 버킷을 확인할 수 있습니다.

6. 미완료 멀티파트 업로드에 대한 자세한 버킷 수준 지표를 보려면 페이지 상단으로 스크롤한 다음 Bucket(버킷) 탭을 선택합니다.

7. 아래로 스크롤하여 Buckets(버킷) 섹션을 찾습니다. Metrics categories(지표 범주)에서 Cost optimization(비용 최적화)을 선택합니다. 그런 다음 Summary(요약)를 지웁니다.

   Buckets(버킷) 목록이 업데이트되어 표시된 버킷에 사용 가능한 모든 Cost optimization(비용 최적화 보호) 지표가 표시됩니다.

8. 특정 비용 최적화 지표만 표시하도록 Buckets(버킷) 목록을 필터링하려면 기본 설정 아이콘 ( ) 을 선택합니다.

9. Incomplete multipart upload bytes greater than 7 days old(7일이 넘게 경과한 미완료 멀티파트 업로드 바이트) 및 Abort incomplete multipart upload lifecycle rule count(미완료 멀티파트 업로드 중단 수명 주기 규칙 개수)만 선택되어 있을 때까지 모든 비용 최적화 지표에 대한 토글을 지웁니다.

10. (선택 사항) Page size(페이지 크기)에서 목록에 표시할 버킷 수를 선택합니다.

11. 확인을 선택합니다.

    Buckets(버킷) 목록이 업데이트되어 미완료 멀티파트 업로드 및 수명 주기 규칙 수에 대한 버킷 수준 지표가 표시됩니다. 이 데이터를 사용하여 7일이 넘게 경과한 미완료 멀티파트 업로드 바이트가 가장 많으며 미완료 멀티파트 업로드를 중단하기 위한 수명 주기 규칙이 누락된 버킷을 식별할 수 있습니다. 그런 다음 S3 콘솔에서 이러한 버킷으로 이동하고 수명 주기 규칙을 추가하여 중단된 미완료 멀티파트 업로드를 삭제할 수 있습니다.

3단계: 7일 후에 미완료 멀티파트 업로드를 삭제하는 수명 주기 규칙 추가

불완전 멀티파트 업로드를 자동으로 관리하려면, 지정된 일수가 지난 후에 버킷에서 불완전한 멀티파트 업로드 바이트를 만료시키는 수명 주기 구성을 S3 콘솔을 사용하여 만들면 됩니다. 자세한 내용은 불완전한 멀티파트 업로드를 삭제하도록 버킷 수명 주기 구성 설정 섹션을 참조하세요.

최신이 아닌 보존 버전 수 축소

S3 버전 관리를 사용 설정하면 동일한 객체의 서로 다른 여러 사본을 보존하기 때문에 객체가 실수로 삭제되거나 덮어쓰여진 경우 데이터를 신속하게 복구할 수 있습니다. 비최신 버전을 전환하거나 만료시키도록 수명 주기 규칙을 구성하지 않고 S3 버전 관리를 활성화한 경우, 이전 비최신 버전이 대량으로 누적되어 스토리지 비용에 영향을 미칠 수 있습니다. 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하세요.

1단계: 비최신 객체 버전이 가장 많은 버킷 식별

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens(스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. 대시보드(Dashboards) 목록에서 보고자 하는 대시보드를 선택합니다.
4. Snapshot for date(날짜 스냅샷) 섹션에서 Metric categories(지표 범주) 아래의 Cost optimization(비용 최적화)을 선택합니다.

   Snapshot for date(날짜 스냅샷) 섹션이 업데이트되어 Cost optimization(비용 최적화) 지표가 표시되며, 여기에는 % noncurrent version bytes(비최신 버전 바이트 비율)에 대한 지표가 포함됩니다. % noncurrent version bytes(비최신 버전 바이트 비율) 지표는 선택한 날짜 및 대시보드 범위 내에 비최신 버전이 총 스토리지 바이트에서 차지하는 비율을 나타냅니다.

   > **Note**
   >
   > % noncurrent version bytes(비최신 버전 바이트 비율)가 계정 수준의 스토리지 중 10%보다 높으면 너무 많은 객체 버전이 저장된 것일 수 있습니다.

5. 여러 비최신 버전이 누적되고 있는 특정 버킷 식별

   a. 아래로 스크롤하여 Top N overview for date(날짜에 대한 상위 N개 개요) 섹션으로 이동합니다. Top N(상위 N개)에서 데이터를 보려는 버킷의 수를 입력합니다.

   b. Metric(지표)에서 % noncurrent version bytes(비최신 버전 바이트 비율)를 선택합니다.

Top number buckets(상위 개수 버킷) 아래에서 % noncurrent version bytes(비최신 버전 바이트 비율)가 가장 높은 버킷을 지정한 숫자만큼 볼 수 있습니다. Top N overview for date(날짜에 대한 상위 N개 개요) 섹션에는 이전 날짜 또는 이전 주로부터의 변동 백분율과 추세를 시각화하는 스파크라인도 표시됩니다. 이 추세는 무료 지표의 경우 14일 추세이고 고급 지표 및 권장 사항의 경우 30일 추세입니다.

> **Note**
>
> S3 스토리지 렌즈 고급 지표 및 권장 사항 을 통해 15개월 동안 쿼리에 지표를 사용할 수 있습니다. 자세한 내용은 지표 선택 섹션을 참조하세요.

c. 비최신 객체 버전에 대한 자세한 버킷 수준 지표를 보려면 페이지 상단으로 스크롤한 다음 Bucket(버킷) 탭을 선택합니다.

   S3 스토리지 렌즈 대시보드의 모든 차트 또는 시각화에서 Account, AWS 리전, Storage class(스토리지 클래스) 또는 Bucket(버킷) 탭을 사용하여 더 심층적인 집계 수준으로 드릴다운할 수 있습니다. 문제 해결 예는 Amazon S3 콜드 버킷 발견을(를) 참조하세요.

d. Buckets(버킷) 섹션에서 Metric categories(지표 범주)에 대해 Cost optimization(비용 최적화)를 선택합니다. 그런 다음 Summary(요약)를 지웁니다.

   이제 % noncurrent version bytes(비최신 버전 바이트 비율) 지표를 비최신 버전과 관련된 다른 지표와 함께 볼 수 있습니다.

2단계: 비최신 버전을 관리하기 위한 전환 및 만료 수명 주기 규칙이 누락된 버킷 식별

전제 조건

S3 스토리지 렌즈 대시보드에서 Noncurrent version transition lifecycle rule count(비최신 버전 전환 수명 주기 규칙 수) 및 Noncurrent version expiration lifecycle rule count(비최신 버전 만료 수명 주기 규칙 수) 지표를 보려면 S3 스토리지 렌즈 Advanced metrics and recommendations(고급 지표 및 권장 사항)를 활성화한 다음 Advanced cost optimization metrics(고급 비용 최적화 지표)를 선택해야 합니다. 자세한 내용은 S3 콘솔 사용 섹션을 참조하세요.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens(스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. 대시보드(Dashboards) 목록에서 보고자 하는 대시보드를 선택합니다.

4. 스토리지 렌즈 대시보드에서 Bucket(버킷) 탭을 선택합니다.
5. 아래로 스크롤하여 Buckets(버킷) 섹션을 찾습니다. Metrics categories(지표 범주)에서 Cost optimization(비용 최적화)을 선택합니다. 그런 다음 Summary(요약)를 지웁니다.

   Buckets(버킷) 목록이 업데이트되어 표시된 버킷에 사용 가능한 모든 Cost optimization(비용 최적화 보호) 지표가 표시됩니다.
6. 특정 비용 최적화 지표만 표시하도록 Buckets(버킷) 목록을 필터링하려면 기본 설정 아이콘 ()을 선택합니다.
7. 다음 항목만 선택될 때까지 모든 비용 최적화 지표에 대한 토글을 지웁니다.

   - % noncurrent version bytes(비최신 버전 바이트 비율)
   - Noncurrent version transition lifecycle rule count(비최신 버전 전환 수명 주기 규칙 수)
   - Noncurrent version expiration lifecycle rule count(비최신 버전 만료 수명 주기 규칙 수)
8. (선택 사항) Page size(페이지 크기)에서 목록에 표시할 버킷 수를 선택합니다.
9. 확인을 선택합니다.

   Buckets(버킷) 목록이 업데이트되어 비최신 버전 바이트 및 비최신 버전 수명 주기 규칙 수에 대한 지표가 표시됩니다. 이 데이터를 사용하여 비최신 버전 바이트의 비율이 높지만 전환 및 만료 수명 주기 규칙이 없는 버킷을 식별할 수 있습니다. 그런 다음 S3 콘솔에서 해당 버킷으로 이동하여 해당 버킷에 수명 주기 규칙을 추가할 수 있습니다.

3단계: 비최신 객체 버전을 전환하거나 만료시키기 위한 수명 주기 규칙 추가

추가 조사가 필요한 버킷을 판단했다면 S3 콘솔 내의 해당 버킷으로 이동해 특정 일 수가 지난 비최신 버전을 만료시키는 수명 주기 정책을 추가할 수 있습니다. 또는 비최신 버전을 보존하면서 비용을 줄이려면 비최신 버전을 Amazon S3 Glacier 스토리지 클래스로 전환하도록 수명 주기 정책을 구성할 수 있습니다. 자세한 내용은 버전 관리를 사용하는 버킷에 대한 수명 주기 규칙 지정 섹션을 참조하세요.

수명 주기 규칙이 없는 버킷을 식별하고 수명 주기 규칙 수 검토

S3 스토리지 렌즈는 수명 주기 규칙이 누락된 버킷을 식별하는 데 사용할 수 있는 S3 수명 주기 규칙 수 지표를 제공합니다. 수명 주기 규칙이 없는 버킷을 찾으려면 Total buckets without lifecycle rules(수명 주기 규칙이 없는 총 버킷 수) 지표를 사용하면 됩니다. S3 수명 주기 구성이 없는 버킷에는 더 이상 필요하지 않거나 더 저렴한 스토리지 클래스로 마이그레이션할 수 있는 스토리지가 있을 수 있습니다. 또한 수명 주기 규칙 수 지표를 사용하여 만료 또는 전환 규칙과 같은 특정 유형의 수명 주기 규칙이 누락된 버킷을 식별할 수 있습니다.

전제 조건

S3 스토리지 렌즈 대시보드에서 수명 주기 규칙 수 지표와 Total buckets without lifecycle rules(수명 주기 규칙이 없는 총 버킷 수) 지표를 보려면 S3 스토리지 렌즈 Advanced metrics and recommendations(고급 지표 및 권장 사항)를 활성화한 다음 Advanced cost optimization metrics(고급 비용 최적화 지표)를 선택해야 합니다. 자세한 내용은 S3 콘솔 사용 섹션을 참조하세요.

1단계: 수명 주기 규칙이 없는 버킷 식별

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens(스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. 대시보드(Dashboards) 목록에서 보고자 하는 대시보드를 선택합니다.
4. 수명 주기 규칙이 없는 특정 버킷을 식별하려면 아래로 스크롤하여 Top N overview for date(날짜에 대한 상위 N개 개요) 섹션으로 이동합니다.

   Top N overview for date(날짜에 대한 상위 N개 개요) 섹션에는 기본적으로 상위 3개 버킷에 대한 지표가 표시됩니다. Top N(상위 N개) 필드에서 버킷 수를 늘릴 수 있습니다. Top N overview for date(날짜에 대한 상위 N개 개요) 섹션에는 이전 날짜 또는 이전 주로부터의 변동 백분율과 추세를 시각화하는 스파크라인도 표시됩니다. 이 추세는 무료 지표의 경우 14일 추세이고 고급 지표 및 권장 사항의 경우 30일 추세입니다.

   > **Note**
   >
   > S3 스토리지 렌즈 고급 지표 및 권장 사항 을 통해 15개월 동안 쿼리에 지표를 사용할 수 있습니다. 자세한 내용은 지표 선택 섹션을 참조하세요.

5. Metric(지표)의 Cost optimization(비용 최적화) 범주에서 Total buckets without lifecycle rules(수명 주기 규칙이 없는 총 버킷 수)를 선택합니다.
6. Total buckets without lifecycle rules(수명 주기 규칙이 없는 총 버킷 수)에 대한 다음 데이터를 검토하세요.

   - Top number accounts(상위 개수 계정) - 수명 주기 규칙이 없는 버킷이 가장 많은 계정을 확인합니다.
   - Top number Regions(상위 개수 리전) - 수명 주기 규칙이 없는 버킷을 리전별로 분류하여 확인합니다.
   - Top number buckets(상위 개수 버킷) - 수명 주기 규칙이 없는 버킷을 확인합니다.

S3 스토리지 렌즈 대시보드의 모든 차트 또는 시각화에서 Account, AWS 리전, Storage class(스토리지 클래스) 또는 Bucket(버킷) 탭을 사용하여 더 심층적인 집계 수준으로 드릴다운할 수 있습니다. 문제 해결 예는 Amazon S3 콜드 버킷 발견을(를) 참조하세요.

수명 주기 규칙이 없는 버킷을 식별한 후에는 버킷의 특정 수명 주기 규칙 수를 검토할 수도 있습니다.

2단계: 버킷의 수명 주기 규칙 수 검토

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens(스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. 대시보드(Dashboards) 목록에서 보고자 하는 대시보드를 선택합니다.
4. S3 스토리지 렌즈 대시보드에서 Bucket(버킷) 탭을 선택합니다.
5. 아래로 스크롤하여 Buckets(버킷) 섹션을 찾습니다. Metrics categories(지표 범주) 아래에서 Cost optimization(비용 최적화)을 선택합니다. 그런 다음 Summary(요약)를 지웁니다.

   Buckets(버킷) 목록이 업데이트되어 표시된 버킷에 사용 가능한 모든 Cost optimization(비용 최적화 보호) 지표가 표시됩니다.
6. 특정 비용 최적화 지표만 표시하도록 Buckets(버킷) 목록을 필터링하려면 기본 설정 아이콘 () 을 선택합니다.
7. 다음 항목만 선택될 때까지 모든 비용 최적화 지표에 대한 토글을 지웁니다.

   - Transition lifecycle rule count(전환 수명 주기 규칙 수)
   - Expiration lifecycle rule count(만료 수명 주기 규칙 수)
   - Noncurrent version transition lifecycle rule count(비최신 버전 전환 수명 주기 규칙 수)
   - Noncurrent version expiration lifecycle rule count(비최신 버전 만료 수명 주기 규칙 수)
   - Abort incomplete multipart upload lifecycle rule count(미완료 멀티파트 업로드 중단 수명 주기 규칙 개수)
   - Total lifecycle rule count(총 수명 주기 규칙 수)
8. (선택 사항) Page size(페이지 크기)에서 목록에 표시할 버킷 수를 선택합니다.
9. 확인을 선택합니다.

Buckets(버킷) 목록이 업데이트되어 버킷의 수명 주기 규칙 수 지표가 표시됩니다. 이 데이터를 사용하여 수명 주기 규칙이 없는 버킷이나 특정 유형의 수명 주기 규칙(예: 만료 또는 전환 규칙)이 누락된 버킷을 식별할 수 있습니다. 그런 다음 S3 콘솔에서 해당 버킷으로 이동하여 해당 버킷에 수명 주기 규칙을 추가할 수 있습니다.

3단계: 수명 주기 규칙 추가

수명 주기 규칙이 없는 버킷을 식별한 후 수명 주기 규칙을 추가할 수 있습니다. 자세한 내용은 버킷에서 S3 수명 주기 구성 설정 및 S3 수명 주기 구성의 예제(을)를 참조하세요.

S3 Storage Lens를 사용하여 데이터 보호

Amazon S3 스토리지 렌즈 데이터 보호 지표를 사용하여 데이터 보호 모범 사례가 적용되지 않은 버킷을 식별할 수 있습니다. 이러한 지표를 사용하여 조치를 취하고 계정 또는 조직의 전체 버킷에서 데이터를 보호하기 위한 모범 사례에 맞는 표준 설정을 적용할 수 있습니다. 예를 들어 데이터 보호 지표를 사용하여 기본 암호화에 AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 사용하지 않는 버킷이나 AWS Signature Version 2(SigV2)를 사용하는 요청을 식별할 수 있습니다.

다음 사용 사례는 S3 스토리지 렌즈 대시보드를 사용하여 S3 버킷 전체에서 이상값을 식별하고 데이터 보호 모범 사례를 적용하기 위한 전략을 제공합니다.

주제

- 기본 암호화에 AWS KMS이(가) 포함된 서버 측 암호화(SSE-KMS)를 사용하지 않는 버킷 식별
- S3 버전 관리가 활성화된 버킷 식별
- AWS Signature Version 2(SigV2)를 사용하는 요청 식별
- 각 버킷의 총 복제 규칙 수 계산
- 객체 잠금 바이트의 백분율 확인

기본 암호화에 AWS KMS이(가) 포함된 서버 측 암호화(SSE-KMS)를 사용하지 않는 버킷 식별

Amazon S3 기본 암호화를 사용하면 S3 버킷의 기본 암호화 동작을 설정할 수 있습니다. 자세한 내용은 the section called “기본 버킷 암호화 설정” 섹션을 참조하세요.

SSE-KMS enabled bucket count(SSE-KMS 활성화 버킷 수) 및 % SSE-KMS enabled buckets(SSE-KMS 활성화 버킷 비율) 지표를 사용하여 기본 암호화에 AWS KMS 키가 포함된 서버 측 암호화(SSE-KMS)를 사용하는 버킷을 식별할 수 있습니다. 또한 S3 스토리지 렌즈는 암호화되지 않은 바이트, 암

호화되지 않은 객체, 암호화된 바이트 및 암호화된 객체에 대한 지표를 제공합니다. 전체 지표 목록은 Amazon S3 스토리지 렌즈 지표 용어집 섹션을 참조하세요.

일반 암호화 지표의 컨텍스트에서 SSE-KMS 암호화 지표를 분석하여 SSE-KMS를 사용하지 않는 버킷을 식별할 수 있습니다. 계정 또는 조직의 모든 버킷에 SSE-KMS를 사용하려면 SSE-KMS를 사용하도록 해당 버킷의 기본 암호화 설정을 업데이트할 수 있습니다. SSE-KMS 외에도 Amazon S3 관리형 키(SSE-S3) 또는 고객 제공 키(SSE-C)가 포함된 서버 측 암호화를 사용할 수 있습니다. 자세한 내용은 암호화로 데이터 보호 섹션을 참조하세요.

1단계: 기본 암호화에 SSE-KMS를 사용하는 버킷 식별

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens(스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. Dashboards(대시보드) 목록에서 보고자 하는 대시보드의 이름을 선택합니다.
4. Trends and distributions(추세 및 배포) 섹션에서 기본 지표로 % SSE-KMS enabled bucket count(SSE-KMS 활성화 버킷 수 비율)를 선택하고 보조 지표로 % encrypted bytes(암호화된 바이트 비율)를 선택합니다.

   Trend for date(날짜에 대한 추세) 차트가 업데이트되어 SSE-KMS 및 암호화된 바이트에 대한 추세가 표시됩니다.
5. SSE-KMS에 대한 보다 세분화된 버킷 수준 인사이트 보기:

   a. 차트에서 지점을 선택합니다. 보다 세부적인 인사이트를 얻을 수 있는 선택 항목이 포함된 상자가 나타납니다.

   b. Buckets(버킷) 차원을 선택합니다. 그런 다음, 적용을 선택합니다.
6. Distribution by buckets for date(날짜에 대한 버킷 분포) 차트에서 SSE-KMS enabled bucket count(SSE-KMS 활성화 버킷 수) 지표를 선택합니다.
7. 이제 SSE-KMS가 활성화된 버킷과 활성화되지 않은 버킷을 확인할 수 있습니다.

2단계: 버킷의 기본 암호화 설정 업데이트

% encrypted bytes(암호화된 바이트 비율)의 컨텍스트에서 SSE-KMS를 사용하는 버킷을 확인했으므로 이제 SSE-KMS를 사용하지 않는 버킷을 식별할 수 있습니다. 그런 다음 선택적으로 S3 콘솔 내에서 이러한 버킷으로 이동하고 SSE-KMS 또는 SSE-S3를 사용하도록 기본 암호화 설정을 업데이트할 수 있습니다. 자세한 내용은 기본 암호화 구성 섹션을 참조하세요.

S3 버전 관리가 활성화된 버킷 식별

S3 버전 관리 기능을 사용 설정하면 동일한 객체의 여러 버전을 보존하기 때문에 객체가 실수로 삭제되거나 덮어쓰여진 경우 데이터를 신속하게 복구할 수 있습니다. Versioning-enabled bucket count(버전 관리 활성화 버킷 수) 지표를 사용하여 S3 버전 관리를 사용하는 버킷을 확인할 수 있습니다. 그런 다음 S3 콘솔에서 조치를 취하여 다른 버킷에 S3 버전 관리를 활성화할 수 있습니다.

1단계: S3 버전 관리가 활성화된 버킷 식별

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 탐색 창에서 Storage Lens(스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. Dashboards(대시보드) 목록에서 보고자 하는 대시보드의 이름을 선택합니다.
4. Trends and distributions(추세 및 배포) 섹션에서 기본 지표로 Versioning-enabled bucket count(버전 관리 활성화 버킷 수)를 선택하고 보조 지표로 Buckets(버킷)를 선택합니다.

   Trend for date(날짜에 대한 추세) 차트가 업데이트되어 S3 버전 관리가 활성화된 버킷에 대한 추세가 표시됩니다. 추세선 바로 아래에서 Storage class distribution(스토리지 클래스 분포) 및 Region distribution(리전 분포) 하위 섹션을 볼 수 있습니다.
5. Trend for date(날짜에 대한 추세) 차트에 표시되는 특정 버킷에 대한 보다 세분화된 인사이트를 보고 보다 심층적인 분석을 수행하려면 다음과 같이 하세요.

   a. 차트에서 지점을 선택합니다. 보다 세부적인 인사이트를 얻을 수 있는 선택 항목이 포함된 상자가 나타납니다.

   b. 보다 심층적인 분석을 위해 데이터에 적용할 차원(Account(계정), AWS 리전, Storage class(스토리지 클래스) 또는 Bucket(버킷))을 선택합니다. 그런 다음, 적용을 선택합니다.
6. Bubble analysis by buckets for date(날짜에 대한 버킷별 거품형 분석) 섹션에서 Versioning-enabled bucket count(버전 관리 활성화 버킷 수), Buckets(버킷) 및 Active buckets(활성 버킷) 지표를 선택합니다.

   Bubble analysis by buckets for date(날짜에 대한 버킷별 거품형 분석) 섹션이 업데이트되어 선택한 지표에 대한 데이터가 표시됩니다. 이 데이터를 사용하여 총 버킷 수의 컨텍스트에서 S3 버전 관리가 활성화된 버킷을 확인할 수 있습니다. Bubble analysis by buckets for date(날짜에 대한 버킷별 거품형 분석) 섹션에서는 X-axis(X축), Y-axis(Y축) 및 Size(크기)를 나타내는 지표 3개를 사용한 다차원 도표에 버킷을 표현할 수 있습니다.

2단계: S3 버전 관리 활성화

S3 버전 관리가 활성화된 버킷을 식별한 후에는 S3 버전 관리가 활성화된 적이 없거나 버전 관리가 일시 중단된 버킷을 식별할 수 있습니다. 그런 다음 S3 콘솔에서 이러한 버킷의 버전 관리를 선택적으로 활성화할 수 있습니다. 자세한 내용은 버킷에 버전 관리 사용 설정 섹션을 참조하세요.

AWS Signature Version 2(SigV2)를 사용하는 요청 식별

All unsupported signature requests(지원되지 않는 모든 서명 요청) 지표를 사용하여 AWS Signature Version 2(SigV2)를 사용하는 요청을 식별할 수 있습니다. 이 데이터는 SigV2를 사용하는 특정 애플리케이션을 식별하는 데 도움이 될 수 있습니다. 그런 다음 이러한 애플리케이션을 AWS Signature Version 4(SigV4)로 마이그레이션할 수 있습니다.

SigV4는 모든 새 S3 애플리케이션에 권장되는 서명 방법입니다. SigV4는 향상된 보안을 제공하며 모든 AWS 리전에서 지원됩니다. 자세한 내용은 Amazon S3 업데이트 - SigV2 사용 중단 기간 연장 및 수정을 참조하세요.

전제 조건

S3 스토리지 렌즈 대시보드에서 All unsupported signature requests(지원되지 않는 모든 서명 요청)를 보려면 S3 스토리지 렌즈 Advanced metrics and recommendations(고급 지표 및 권장 사항)를 활성화한 다음 Advanced data protection metrics(고급 데이터 보호 지표)를 선택해야 합니다. 자세한 내용은 S3 콘솔 사용 섹션을 참조하세요.

1단계: AWS 계정, 리전 및 버킷별 SigV2 서명 추세 검토

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens(스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. Dashboards(대시보드) 목록에서 보고자 하는 대시보드의 이름을 선택합니다.
4. SigV2를 사용하는 요청이 있는 특정 버킷, 계정 및 리전 식별:

    a. Top N(상위 N개)의 Top N overview for date(날짜에 대한 상위 N개 개요) 아래에서 데이터를 보려는 버킷의 수를 입력합니다.

    b. Metric(지표)에는 Data protection(데이터 보호) 범주의 All unsupported signature requests(지원되지 않는 모든 서명 요청)를 선택합니다.

    Top N overview for date(날짜에 대한 상위 N개 개요)가 업데이트되어 계정, AWS 리전 및 버킷별 SigV2 요청에 대한 데이터가 표시됩니다. Top N overview for date(날짜에 대한 상위 N

개 개요) 섹션에는 이전 날짜 또는 이전 주로부터의 변동 백분율과 추세를 시각화하는 스파크라인도 표시됩니다. 이 추세는 무료 지표의 경우 14일 추세이고 고급 지표 및 권장 사항의 경우 30일 추세입니다.

> **Note**
>
> S3 스토리지 렌즈 고급 지표 및 권장 사항 을 통해 15개월 동안 쿼리에 지표를 사용할 수 있습니다. 자세한 내용은 지표 선택 섹션을 참조하세요.

2단계: 애플리케이션이 SigV2 요청을 통해 액세스하는 버킷 식별

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens(스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. Dashboards(대시보드) 목록에서 보고자 하는 대시보드의 이름을 선택합니다.
4. 스토리지 렌즈 대시보드에서 Bucket(버킷) 탭을 선택합니다.
5. 아래로 스크롤하여 Buckets(버킷) 섹션을 찾습니다. Metrics categories(지표 범주)에서 Data protection(데이터 보호)을 선택합니다. 그런 다음 Summary(요약)를 지웁니다.

   Buckets(버킷) 목록이 업데이트되어 표시된 버킷에 사용 가능한 모든 Data protection(데이터 보호) 지표가 표시됩니다.
6. 특정 데이터 보호 지표만 표시하도록 Buckets(버킷) 목록을 필터링하려면 기본 설정 아이콘 ( ) 을 선택합니다.
7. 다음 지표만 선택된 상태가 될 때까지 모든 데이터 보호 지표에 대한 토글을 지웁니다.

   - All unsupported signature requests(지원되지 않는 모든 서명 요청)
   - % all unsupported signature requests(지원되지 않는 모든 서명 요청 비율)
8. (선택 사항) Page size(페이지 크기)에서 목록에 표시할 버킷 수를 선택합니다.
9. 확인을 선택합니다.

   Buckets(버킷) 목록이 업데이트되어 SigV2 요청에 대한 버킷 수준 지표가 표시됩니다. 이 데이터를 사용하여 SigV2 요청이 있는 특정 버킷을 식별할 수 있습니다. 그런 다음 이 정보를 사용하여 애플리케이션을 SigV4로 마이그레이션할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조의 요청 인증(AWS 서명 버전 4)을 참조하세요.

각 버킷의 총 복제 규칙 수 계산

S3 복제를 사용하면 Amazon S3 버킷 전체에 걸쳐 객체를 비동기식으로 자동 복사할 수 있습니다. 객체 복제를 위해 구성된 버킷은 동일한 AWS 계정 또는 다른 계정이 소유할 수 있습니다. 자세한 내용은 리전 내 및 리전 간 객체 복제 섹션을 참조하세요.

S3 스토리지 렌즈 복제 규칙 수 지표를 사용하여 복제용으로 구성된 버킷에 대한 자세한 버킷별 정보를 얻을 수 있습니다. 이 정보에는 버킷 및 리전 내부 및 리전 간의 복제 규칙이 포함됩니다.

전제 조건

S3 스토리지 렌즈 대시보드에서 복제 규칙 수 지표를 보려면 S3 스토리지 렌즈 Advanced metrics and recommendations(고급 지표 및 권장 사항)를 활성화한 다음 Advanced data protection metrics(고급 데이터 보호 지표)를 선택해야 합니다. 자세한 내용은 S3 콘솔 사용 섹션을 참조하세요.

1단계: 각 버킷의 총 복제 규칙 수 계산

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens(스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. Dashboards(대시보드) 목록에서 보고자 하는 대시보드의 이름을 선택합니다.
4. 스토리지 렌즈 대시보드에서 Bucket(버킷) 탭을 선택합니다.
5. 아래로 스크롤하여 Buckets(버킷) 섹션을 찾습니다. Metrics categories(지표 범주)에서 Data protection(데이터 보호)을 선택합니다. 그런 다음 Summary(요약)를 지웁니다.
6. 복제 규칙 수 지표만 표시하도록 Buckets(버킷) 목록을 필터링하려면 기본 설정 아이콘 () 을 선택합니다.
7. 복제 규칙 수 지표만 선택된 상태가 될 때까지 모든 데이터 보호 지표에 대한 토글을 지웁니다.
   - Same-Region Replication rule count(동일 리전 복제 규칙 수)
   - Cross-Region Replication rule count(교차 리전 복제 규칙 수)
   - Same-account replication rule count(동일 계정 복제 규칙 수)
   - Cross-account replication rule count(크로스 계정 복제 규칙 수)
   - Total replication rule count(총 복제 규칙 수)
8. (선택 사항) Page size(페이지 크기)에서 목록에 표시할 버킷 수를 선택합니다.
9. 확인을 선택합니다.

2단계: 복제 규칙 추가

버킷별 복제 규칙 수를 확인한 후 선택적으로 추가 복제 규칙을 생성할 수 있습니다. 자세한 내용은 라이브 복제 구성 예제 섹션을 참조하세요.

객체 잠금 바이트의 백분율 확인

S3 객체 잠금을 사용하면 write-once-read-many(WORM) 모델을 사용하여 객체를 저장할 수 있습니다. 객체 잠금을 사용하여 고정된 시간 동안 또는 무기한으로 객체의 삭제 또는 덮어쓰기를 방지할 수 있습니다. 버킷을 생성할 때만 객체 잠금을 활성화하고 S3 버전 관리도 활성화할 수 있습니다. 하지만 개별 객체 버전의 보존 기간을 편집하거나 객체 잠금이 활성화된 버킷에 법적 보류를 적용할 수 있습니다. 자세한 내용은 Object Lock으로 객체 잠금 섹션을 참조하세요.

S3 스토리지 렌즈의 객체 잠금 지표를 사용하여 계정 또는 조직의 % Object Lock bytes(객체 잠금 바이트 비율) 지표를 확인할 수 있습니다. 이 정보를 사용하여 계정 또는 조직에서 데이터 보호 모범 사례를 따르지 않는 버킷을 식별할 수 있습니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens(스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. Dashboards(대시보드) 목록에서 보고자 하는 대시보드의 이름을 선택합니다.
4. Snapshot(스냅샷) 섹션의 Metrics categories(지표 범주)에서 Data protection(데이터 보호)을 선택합니다.

   Snapshot(스냅샷) 섹션이 업데이트되어 % Object Lock bytes(객체 잠금 바이트 비율) 지표를 비롯한 데이터 보호 지표가 표시됩니다. 계정 또는 조직에 대한 객체 잠금 바이트의 전체 백분율을 볼 수 있습니다.
5. 버킷당 % Object Lock bytes(객체 잠금 바이트 비율)를 보려면 아래로 스크롤하여 Top N overview(상위 N개 개요) 섹션으로 이동합니다.

   객체 잠금에 대한 객체 수준 데이터를 가져오려면 Object Lock object count(객체 잠금 객체 수) 및 % Object Lock objects(객체 잠금 객체 비율)를 사용할 수도 있습니다.
6. Metric(지표)에는 Data protection(데이터 보호) 범주의 % Object Lock bytes(객체 잠금 바이트 비율)를 선택합니다.

   Top N overview for date(날짜에 대한 상위 N개 개요) 섹션에는 기본적으로 상위 3개 버킷에 대한 지표가 표시됩니다. Top N(상위 N개) 필드에서 버킷 수를 늘릴 수 있습니다. Top N overview for date(날짜에 대한 상위 N개 개요) 섹션에는 이전 날짜 또는 이전 주로부터의 변동 백분율과 추세를

시각화하는 스파크라인도 표시됩니다. 이 추세는 무료 지표의 경우 14일 추세이고 고급 지표 및 권장 사항의 경우 30일 추세입니다.

> **Note**
>
> S3 스토리지 렌즈 고급 지표 및 권장 사항 을 통해 15개월 동안 쿼리에 지표를 사용할 수 있습니다. 자세한 내용은 지표 선택 섹션을 참조하세요.

7. % Object Lock bytes(객체 잠금 바이트 비율)에 대한 다음 데이터를 검토합니다.

   - Top number accounts(상위 개수 계정) - % Object Lock bytes(객체 잠금 바이트 비율)이 가장 높은 계정과 가장 낮은 계정을 확인합니다.
   - Top number Regions(상위 개수 리전) - % Object Lock bytes(객체 잠금 바이트 비율)의 리전별 분석을 확인합니다.
   - Top number buckets(상위 개수 버킷) - % Object Lock bytes(객체 잠금 바이트 비율)이 가장 높은 버킷과 가장 낮은 버킷을 확인합니다.

S3 스토리지 렌즈를 사용하여 객체 소유권 설정 감사

S3 객체 소유권은 액세스 제어 목록(ACL)을 비활성화하고 버킷에 있는 객체의 소유권을 제어하는 데 사용할 수 있는 S3 버킷 수준 설정입니다. 객체 소유권을 버킷 소유자 강제 적용으로 설정하면 액세스 제어 목록(ACL)을 비활성화하고 버킷의 모든 객체에 대한 소유권을 가져옵니다. 이 접근 방식은 Amazon S3에 저장된 데이터에 대한 액세스 관리를 간소화합니다.

기본적으로 다른 AWS 계정이 S3 버킷에 객체를 업로드하면 해당 계정(객체 작성자)이 객체를 소유하고 객체에 액세스할 수 있으며 ACL을 통해 다른 사용자에게 객체에 대한 액세스 권한을 부여할 수 있습니다. 객체 소유권을 사용하여 이 기본 동작을 변경할 수 있습니다.

Amazon S3의 최신 사용 사례 대부분은 더 이상 ACL을 사용할 필요가 없습니다. 따라서 각 객체에 대해 액세스를 개별적으로 제어해야 하는 비정상적인 상황을 제외하고는 ACL을 비활성화하는 것이 좋습니다. 객체 소유권을 버킷 소유자 강제 적용으로 설정하면 ACL을 비활성화하고 정책에 의존해 액세스를 제어할 수 있습니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

S3 스토리지 렌즈 액세스 관리 지표를 사용하면 비활성화된 ACL이 없는 버킷을 식별할 수 있습니다. 이러한 버킷을 식별한 후 ACL 권한을 정책으로 마이그레이션하고 해당 버킷에 대한 ACL을 비활성화할 수 있습니다.

주제

- 1단계: 객체 소유권 설정의 일반 추세 파악
- 2단계: 객체 소유권 설정의 버킷 수준 추세 파악
- 3단계: ACL을 비활성화하기 위해 객체 소유권 설정을 버킷 소유자 강제 적용으로 업데이트

1단계: 객체 소유권 설정의 일반 추세 파악

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens(스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. Dashboards(대시보드) 목록에서 보고자 하는 대시보드의 이름을 선택합니다.
4. Snapshot for date(날짜 스냅샷) 섹션에서 Metrics categories(지표 범주) 아래의 Access management(액세스 관리)를 선택합니다.

   Snapshot for date(날짜 스냅샷) 섹션이 업데이트되어 % Object Ownership bucket owner enforced(객체 소유권 버킷 소유자 강제 적용 비율) 지표가 표시됩니다. 계정 또는 조직에서 객체 소유권에 버킷 소유자 강제 적용 설정을 사용하여 ACL을 사용 중지하는 버킷의 전체 비율을 볼 수 있습니다.

2단계: 객체 소유권 설정의 버킷 수준 추세 파악

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens(스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. Dashboards(대시보드) 목록에서 보고자 하는 대시보드의 이름을 선택합니다.
4. 더 자세한 버킷 수준 지표를 보려면 Bucket(버킷) 탭을 선택합니다.
5. Distribution by buckets for date(날짜에 대한 버킷 분포) 섹션에서 % Object Ownership bucket owner enforced(객체 소유권 버킷 소유자 강제 적용 비율) 지표를 선택합니다.

   차트가 업데이트되어 % Object Ownership bucket owner enforced(객체 소유권 버킷 소유자 강제 적용 비율)에 대한 버킷별 분석이 표시됩니다. ACL을 사용 중지하기 위해 객체 소유권에 버킷 소유자 강제 적용 설정을 사용하는 버킷을 볼 수 있습니다.
6. 버킷 소유자 강제 적용 설정을 컨텍스트에서 보려면 아래로 스크롤하여 Buckets(버킷) 섹션으로 이동하세요. Metrics categories(지표 범주)에서 Access management(액세스 관리)를 선택합니다. 그런 다음 Summary(요약)를 지웁니다.

Buckets(버킷) 목록에는 세 가지 객체 소유권 설정(버킷 소유자 강제 적용, 버킷 소유자 선호, 객체 작성자) 모두에 대한 데이터가 표시됩니다.

7. 특정 객체 소유권 설정에 대한 지표만 표시하도록 Buckets(버킷) 목록을 필터링하려면 기본 설정 아이콘 ( ) 을 선택합니다.
8. 보고 싶지 않은 지표를 지웁니다.
9. (선택 사항) Page size(페이지 크기)에서 목록에 표시할 버킷 수를 선택합니다.
10. 확인을 선택합니다.

3단계: ACL을 비활성화하기 위해 객체 소유권 설정을 버킷 소유자 강제 적용으로 업데이트

객체 소유권에 객체 작성자와 버킷 소유자 선호 설정을 사용하는 버킷을 식별한 후에는 ACL 권한을 버킷 정책으로 마이그레이션할 수 있습니다. ACL 권한 마이그레이션을 마치면 객체 소유권 설정을 버킷 소유자 강제 적용으로 업데이트하여 ACL을 비활성화할 수 있습니다. 자세한 내용은 ACL 사용 중지를 위한 사전 조건 섹션을 참조하세요.

성능 향상을 위한 S3 스토리지 렌즈 지표 사용

S3 스토리지 렌즈 고급 지표가 활성화되어 있다면 세부 상태 코드 지표를 사용해 성공 또는 실패한 요청의 수를 파악할 수 있습니다. 이 정보를 사용하여 액세스 또는 성능 문제를 해결할 수 있습니다. 세부 상태 코드 지표는 403 금지됨 및 503 서비스 사용 불가와 같은 HTTP 상태 코드의 수를 보여 줍니다. S3 버킷, 계정 및 조직 전반의 세부 상태 코드 지표에 대한 전반적인 추세를 살펴볼 수 있습니다. 그런 다음 버킷 수준 지표를 드릴다운하여 현재 이러한 버킷에 액세스하여 오류를 일으키는 워크로드를 식별할 수 있습니다.

예를 들어 403 Forbidden error count(403 금지됨 오류 수) 지표를 살펴보면 올바른 권한이 적용되지 않은 상태에서 버킷에 액세스하는 워크로드를 식별할 수 있습니다. 이러한 워크로드를 식별한 후에는 S3 스토리지 렌즈 외부에서 심층적으로 분석하여 403 금지됨 오류를 해결할 수 있습니다.

이 예에서는 403 Forbidden error count(404 금지됨 오류 수) 및 % 403 Forbidden errors(403 금지됨 오류 비율) 지표를 사용하여 403 금지됨 오류에 대한 추세 분석을 수행하는 방법을 보여 줍니다. 이러한 지표를 사용하여 올바른 권한이 적용되지 않은 상태에서 버킷에 액세스하는 워크로드를 식별할 수 있습니다. 다른 세부 상태 코드 지표에 대해서도 유사한 추세 분석을 수행할 수 있습니다. 자세한 내용은 Amazon S3 스토리지 렌즈 지표 용어집 섹션을 참조하세요.

전제 조건

S3 스토리지 렌즈 대시보드에서 세부 상태 코드 지표를 보려면 S3 스토리지 렌즈 고급 지표 및 권장 사항을 활성화한 다음 세부 상태 코드 지표를 선택해야 합니다. 자세한 내용은 S3 콘솔 사용 섹션을 참조하세요.

주제

- 1단계: 개별 HTTP 상태 코드에 대한 추세 분석 수행
- 2단계: 버킷별 오류 수 분석
- 3단계: 오류 해결

1단계: 개별 HTTP 상태 코드에 대한 추세 분석 수행

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens(스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. Dashboards(대시보드) 목록에서 보고자 하는 대시보드의 이름을 선택합니다.
4. Trends and distributions(추세 및 분포) 섹션의 Primary metric(기본 지표)에서 Detailed status codes(세부 상태 코드) 범주의 403 Forbidden error count(403 금지됨 오류 수)를 선택합니다. Secondary metric(보조 지표)의 경우 % 403 Forbidden errors(403 금지됨 오류 비율)를 선택합니다.
5. 아래로 스크롤하여 Top N overview for date(날짜에 대한 상위 N개 개요) 섹션으로 이동합니다. Metrics(지표)의 경우 Detailed status codes(세부 상태 코드) 범주의 403 Forbidden error count(403 금지됨 오류 수) 또는 % 403 Forbidden errors(403 금지됨 오류 비율)를 선택합니다.

   Top N overview for date(날짜에 대한 상위 N개 개요) 섹션이 업데이트되어 계정, AWS 리전 및 버킷별 상위 403 금지됨 오류 수가 표시됩니다.

2단계: 버킷별 오류 수 분석

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens(스토리지 렌즈), Dashboards(대시보드)를 선택합니다.
3. Dashboards(대시보드) 목록에서 보고자 하는 대시보드의 이름을 선택합니다.
4. 스토리지 렌즈 대시보드에서 Bucket(버킷) 탭을 선택합니다.
5. 아래로 스크롤하여 Buckets(버킷) 섹션을 찾습니다. Metrics categories(지표 범주)에서 Detailed status code(세부 상태 코드) 측정항목을 선택합니다. 그런 다음 Summary(요약)를 지웁니다.

Buckets(버킷) 목록이 업데이트되어 사용 가능한 모든 세부 상태 코드 지표가 표시됩니다. 이 정보를 사용하여 특정 HTTP 상태 코드가 많은 버킷과 버킷 간에 공통적인 상태 코드를 확인할 수 있습니다.

6. 특정 세부 상태 코드 지표만 표시하도록 Buckets(버킷) 목록을 필터링하려면 기본 설정 아이콘 ( ) 을 선택합니다.
7. Buckets(버킷) 목록에서 보고 싶지 않은 세부 상태 코드 지표에 대한 토글을 지웁니다.
8. (선택 사항) Page size(페이지 크기)에서 목록에 표시할 버킷 수를 선택합니다.
9. 확인을 선택합니다.

   Buckets(버킷) 목록에는 지정한 버킷 수에 대한 오류 수 지표가 표시됩니다. 이 정보를 사용하여 오류가 많이 발생하는 특정 버킷을 식별하고 버킷별로 오류를 해결할 수 있습니다.

3단계: 오류 해결

특정 HTTP 상태 코드의 비율이 높은 버킷을 식별한 후에는 이러한 오류를 해결할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- Amazon S3에서 파일을 업로드하려고 할 때 403 금지됨 오류가 발생하는 이유는 무엇인가요?
- Amazon S3에서 버킷 정책을 수정하려고 할 때 403 금지됨 오류가 발생하는 이유는 무엇인가요?
- 모든 리소스의 출처가 동일한 AWS 계정인 Amazon S3 버킷에서 발생한 403 금지됨 오류를 해결하려면 어떻게 해야 하나요?
- Amazon S3에서 발생한 HTTP 500 또는 503 오류를 해결하려면 어떻게 해야 하나요?

(으)로 Amazon S3 스토리지 렌즈 사용AWS Organizations

Amazon S3 스토리지 렌즈는 조직 전반에서 객체 스토리지 사용 및 활동에 대한 가시성을 확보하는 데 사용할 수 있는 클라우드 스토리지 분석 기능입니다. S3 스토리지 렌즈 지표를 사용하여 조직 전반의 스토리지 용량 또는 가장 빠르게 증가하는 버킷과 접두사 등의 요약 인사이트를 생성할 수 있습니다. Amazon S3 Storage Lens를 사용하여 AWS Organizations 계층 구조의 일부인 AWS 계정 모두에 대한 스토리지 지표와 사용량 데이터를 수집할 수도 있습니다. 이렇게 하려면 AWS Organizations를 사용해야 하며 AWS Organizations 관리 계정을 사용하여 S3 스토리지 렌즈 신뢰할 수 있는 액세스를 사용해야 합니다.

신뢰할 수 있는 액세스를 활성화한 후, 조직의 계정에 위임된 관리자 액세스 권한을 추가합니다. 위임된 관리자 계정은 조직 전체의 스토리지 지표와 사용자 데이터를 수집하는 S3 Storage Lens 구성 및 대시보드를 만드는 데 사용됩니다. 신뢰할 수 있는 액세스를 사용하는 방법에 대한 자세한 내용은 AWS Organizations 사용 설명서의 Amazon S3 스토리지 렌즈 및 AWS Organizations를 참조하십시오.

주제

- S3 스토리지 렌즈에 대한 트러스트된 액세스 사용 설정
- S3 스토리지 렌즈에 대해 트러스트된 액세스 사용 중지
- S3 스토리지 렌즈에 대해 위임된 관리자 등록
- S3 스토리지 렌즈에 대해 위임된 관리자 등록 취소

S3 스토리지 렌즈에 대한 트러스트된 액세스 사용 설정

신뢰할 수 있는 액세스를 사용하면 Amazon S3 Storage Lens가 AWS Organizations API 작업을 통해 AWS Organizations 계층 구조, 멤버십 및 구조에 액세스할 수 있습니다. S3 스토리지 렌즈는 전체 조직의 구조에 대한 트러스트된 서비스가 됩니다.

대시보드 구성이 생성될 때마다 S3 스토리지 렌즈가 조직의 관리 또는 위임된 관리자 계정에 서비스 연결 역할을 생성합니다. 서비스 연결 역할은 S3 Storage Lens에 다음 작업을 수행할 수 있는 권한을 부여합니다.

- 조직 설명
- 계정 나열
- 조직의 AWS 서비스 액세스 목록 확인
- 조직의 위임 관리자 가져오기

그런 다음 S3 Storage Lens는 조직의 계정에 대한 교차 계정 지표를 수집할 수 있는 액세스 권한이 있는지 확인할 수 있습니다. 자세한 내용은 Amazon S3 스토리지 렌즈에 서비스 연결 역할 사용을 참조하십시오.

트러스트된 액세스를 사용 설정한 후, 조직의 계정에 위임된 관리자 액세스 권한을 할당할 수 있습니다. 계정이 서비스에 대한 위임된 관리자로 표시되면 해당 계정은 모든 읽기 전용 조직 API에 액세스할 수 있는 권한을 받습니다. 이렇게 하면 위임된 관리자에게 조직의 구성원 및 구조에 대한 가시성이 제공되어 S3 스토리지 렌즈 대시보드를 만들 수 있습니다.

> **Note**
>
> - 트러스트된 액세스는 관리 계정에서만 사용 설정할 수 있습니다.
> - 관리 계정 및 위임된 관리자만 조직에 대한 S3 Storage Lens 대시보드 또는 구성을 생성할 수 있습니다.

S3 콘솔 사용

S3 Storage Lens가 AWS Organizations 신뢰할 수 있는 액세스 권한을 갖도록 활성화하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens로 이동합니다.
3. AWS Organizations 설정을 선택합니다. Storage Lens에 대한 AWS Organizations 액세스 페이지가 표시됩니다.
4. AWS Organizations 신뢰할 수 있는 액세스에서 편집을 선택합니다.

   AWS Organizations 액세스 페이지가 표시됩니다.
5. S3 Storage Lens 대시보드에 대해 신뢰할 수 있는 액세스를 활성화하려면 활성화를 선택합니다.
6. 변경 사항 저장을 선택합니다.

AWS CLI 사용

Example

다음 예제에서는 AWS CLI에서 S3 Storage Lens에 대한 AWS Organizations 신뢰할 수 있는 액세스를 활성화하는 방법을 보여줍니다.

```
aws organizations enable-aws-service-access --service-principal storage-
lens.s3.amazonaws.com
```

Java용 AWS SDK 사용

Example - SDK for Java를 사용하여 S3 Storage Lens에 대한 AWS Organizations 신뢰할 수 있는 액세스 활성화

다음 예제에서는 SDK for Java에서 S3 Storage Lens에 대한 신뢰할 수 있는 액세스를 활성화하는 방법을 보여줍니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import com.amazonaws.services.organizations.model.EnableAWSServiceAccessRequest;

public class EnableOrganizationsTrustedAccess {
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";

 public static void main(String[] args) {
  try {
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            organizationsClient.enableAWSServiceAccess(new
 EnableAWSServiceAccessRequest()
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
 }
}
```

S3 스토리지 렌즈에 대해 트러스트된 액세스 사용 중지

위임된 관리자로 표시된 계정을 제거하거나 신뢰할 수 있는 액세스를 비활성화하면 계정 소유자의 S3 Storage Lens 대시보드 지표가 계정 수준에서만 작동하도록 제한됩니다. 그러면 각 계정 소유자는 전체 조직이 아닌 제한된 계정 범위를 통해서만 S3 Storage Lens의 이점을 확인할 수 있습니다.

S3 Storage Lens에서 신뢰할 수 있는 액세스를 비활성화하면 신뢰할 수 있는 액세스가 필요한 대시보드는 더 이상 업데이트되지 않습니다. 생성된 조직 대시보드도 더 이상 업데이트되지 않습니다. 대신 데이터를 사용할 수 있는 동안 S3 Storage Lens 대시보드의 기록 데이터만 쿼리할 수 있습니다.

> **Note**
>
> - 또한 S3 스토리지 렌즈에 대한 트러스트된 액세스를 비활성화하면 모든 조직 수준의 대시보드가 스토리지 지표를 수집하고 집계하는 것이 자동으로 중지됩니다. 이는 S3 Storage Lens가 더 이상 조직 계정에 대한 신뢰할 수 있는 액세스 권한을 갖지 않기 때문입니다.
> - 관리 계정과 위임 관리자 계정은 비활성화된 대시보드에 대한 기록 데이터를 계속 볼 수 있습니다. 또한, 기록 데이터가 아직 사용 가능한 상태일 경우 해당 기록 데이터를 쿼리할 수도 있습니다.

S3 콘솔 사용

S3 스토리지 렌즈에 대해 트러스트된 액세스 사용 중지

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens로 이동합니다.
3. AWS Organizations 설정을 선택합니다. Storage Lens에 대한 AWS Organizations 액세스 페이지가 표시됩니다.
4. AWS Organizations 신뢰할 수 있는 액세스에서 편집을 선택합니다.

   AWS Organizations 액세스 페이지가 표시됩니다.
5. S3 Storage Lens 대시보드에 대해 신뢰할 수 있는 액세스를 비활성화하려면 비활성화를 선택합니다.
6. 변경 사항 저장을 선택합니다.

AWS CLI 사용

Example

다음 예제에서는 AWS CLI를 사용하여 S3 Storage Lens에 대한 신뢰할 수 있는 액세스를 비활성화합니다.

```
aws organizations disable-aws-service-access --service-principal storage-
lens.s3.amazonaws.com
```

Java용 AWS SDK 사용

Example – S3 Storage Lens에 대한 AWS Organizations 신뢰할 수 있는 액세스 비활성화

다음 예제에서는 SDK for Java에서 S3 Storage Lens에 대한 AWS Organizations 신뢰할 수 있는 액세스를 비활성화하는 방법을 보여줍니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import com.amazonaws.services.organizations.model.DisableAWSServiceAccessRequest;

public class DisableOrganizationsTrustedAccess {
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";

 public static void main(String[] args) {
  try {
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            // Make sure to remove any existing delegated administrator for S3 Storage
 Lens
            // before disabling access; otherwise, the request will fail.
            organizationsClient.disableAWSServiceAccess(new
 DisableAWSServiceAccessRequest()
```

```
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
 }
}
```

S3 스토리지 렌즈에 대해 위임된 관리자 등록

조직의 관리 계정 또는 위임된 관리자 계정을 사용하여 조직 수준의 대시보드를 생성할 수 있습니다. 위임된 관리자 계정을 사용하면 관리 계정 이외의 다른 계정에서 조직 수준의 대시보드를 생성할 수 있습니다. 조직의 관리 계정만 조직의 위임 관리자로서 다른 계정을 등록 및 등록 취소할 수 있습니다.

신뢰할 수 있는 액세스를 활성화한 후 관리 계정의 AWS Organizations REST API, AWS CLI 또는 SDK를 사용하여 조직 내 계정에 위임 관리자 액세스 권한을 등록할 수 있습니다. (자세한 내용을 알아보려면 AWS Organizations API 참조의 RegisterDelegatedAdministrator 섹션을 참조하세요.) 계정이 위임된 관리자로 등록되면 해당 계정은 모든 읽기 전용 AWS Organizations API 작업에 액세스할 수 있는 권한을 받습니다. 이렇게 하면 조직의 구성원 및 구조에 대한 가시성을 제공하므로 사용자를 대신하여 S3 스토리지 렌즈 대시보드를 만들 수 있습니다.

> **Note**
>
> AWS Organizations REST API, AWS CLI 또는 SDK를 사용하여 위임된 관리자를 지정하기 전에, 먼저 EnableAWSOrganizationsAccess 작업을 호출해야 합니다.

S3 콘솔 사용

S3 스토리지 렌즈에 대해 위임된 관리자 등록

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens로 이동합니다.

3. AWS Organizations 설정을 선택합니다.
4. 위임된 관리자에서 계정 등록을 선택합니다.
5. AWS 계정 ID를 추가하여 계정을 위임된 관리자로 등록합니다. 위임된 관리자는 조직의 모든 계정과 스토리지에 대한 조직 수준의 대시보드를 만들 수 있습니다.
6. 계정 등록을 선택합니다.

AWS CLI 사용

Example

다음 예제는 AWS CLI를 사용하여 S3 Storage Lens에 대한 Organizations 위임 관리자를 등록하는 방법을 보여줍니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
aws organizations register-delegated-administrator --service-principal storage-
lens.s3.amazonaws.com --account-id 111122223333
```

Java용 AWS SDK 사용

Example - S3 Storage Lens에 대한 Organizations 위임 관리자 등록

다음 예제는 SDK for Java에서 S3 Storage Lens에 대한 AWS Organizations 위임 관리자를 등록하는 방법을 보여줍니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import
 com.amazonaws.services.organizations.model.RegisterDelegatedAdministratorRequest;

public class RegisterOrganizationsDelegatedAdministrator {
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";

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

```
 try {
            String delegatedAdminAccountId = "111122223333"; // Account Id for the
delegated administrator.
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            organizationsClient.registerDelegatedAdministrator(new
RegisterDelegatedAdministratorRequest()
                .withAccountId(delegatedAdminAccountId)
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
 }
}
```

S3 스토리지 렌즈에 대해 위임된 관리자 등록 취소

신뢰할 수 있는 액세스를 활성화한 후, 조직의 계정에 위임된 관리자 액세스 권한을 등록 취소할 수도 있습니다. 위임된 관리자 계정을 사용하면 관리 계정 이외의 다른 계정에서 조직 수준의 대시보드를 생성할 수 있습니다. 조직의 관리 계정만 조직의 위임된 관리자로 계정의 등록을 취소할 수 있습니다.

관리 계정의 AWS Organizations AWS Management Console, REST API, AWS CLI 또는 AWS SDK를 사용하여 위임된 관리자의 등록을 취소할 수 있습니다. 자세한 내용을 알아보려면 AWS Organizations API 참조의 DeregisterDelegatedAdministrator 섹션을 참조하세요.

위임된 관리자 등록이 취소된 계정은 다음 액세스 권한을 잃게 됩니다.

- 조직의 구성원 및 구조를 볼 수 있는 모든 읽기 전용 AWS Organizations API 작업.
- 위임된 관리자가 만드는 모든 조직 수준의 대시보드. 또한 위임된 관리자의 등록을 취소하면 위임된 관리자가 생성한 모든 조직 수준의 대시보드가 새 스토리지 지표를 집계하는 것이 자동으로 중지됩니다.

> **Note**
>
> 등록 취소된 위임 관리자는 데이터를 여전히 쿼리할 수 있다면 직접 만든 비활성화된 대시보드의 기록 데이터를 계속 볼 수 있습니다.

S3 콘솔 사용

S3 Storage Lens에 대해 위임된 관리자 등록을 취소하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Storage Lens로 이동합니다.
3. AWS Organizations 설정을 선택합니다.
4. 위임된 관리자에서 등록 취소할 계정을 선택합니다.
5. 계정 등록 취소를 선택합니다. 등록 취소된 계정은 더 이상 위임된 관리자가 아니므로, 조직의 모든 계정과 스토리지에 대한 조직 수준의 대시보드를 만들 수 없습니다.
6. 계정 등록을 선택합니다.

AWS CLI 사용

Example

다음 예제는 AWS CLI를 사용하여 S3 Storage Lens에 대한 Organizations 위임 관리자 등록을 취소하는 방법을 보여줍니다. 이 예제를 사용하려면 각 *111122223333*를 사용자의 AWS 계정 ID로 대체합니다.

```
aws organizations deregister-delegated-administrator --service-principal storage-lens.s3.amazonaws.com --account-id 111122223333
```

Java용 AWS SDK 사용

Example - S3 Storage Lens에 대한 Organizations 위임 관리자 등록 취소

다음 예제는 SDK for Java를 사용하여 S3 Storage Lens에 대한 Organizations 위임 관리자 등록을 취소하는 방법을 보여줍니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import
 com.amazonaws.services.organizations.model.DeregisterDelegatedAdministratorRequest;

public class DeregisterOrganizationsDelegatedAdministrator {
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";

 public static void main(String[] args) {
  try {
            String delegatedAdminAccountId = "111122223333"; // Account Id for the
 delegated administrator.
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            organizationsClient.deregisterDelegatedAdministrator(new
 DeregisterDelegatedAdministratorRequest()
                .withAccountId(delegatedAdminAccountId)
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
 }
}
```

S3 Storage Lens 그룹을 사용하여 지표 필터링 및 집계

Amazon S3 Storage Lens 그룹은 객체 메타데이터를 기반으로 사용자 지정 필터를 사용하여 지표를 집계합니다. Storage Lens 그룹을 사용하면 연령별 객체 분포, 가장 일반적인 파일 유형 등과 같은 데이터의 특성을 자세히 살펴볼 수 있습니다. 예를 들어 객체 태그별로 지표를 필터링하여 가장 빠르게 성장하는 데이터 세트를 식별하거나 객체 크기 및 기간을 기준으로 스토리지를 시각화하여 스토리지 아카이브 전략을 수립할 수 있습니다. 따라서 Amazon Storage Lens 그룹은 S3 스토리지를 더 잘 이해하고 최적화하는 데 도움이 됩니다.

Storage Lens 그룹을 사용하면 접두사, 접미사, 객체 태그, 객체 크기 또는 객체 수명과 같은 객체 메타데이터를 사용하여 S3 Storage Lens 지표를 분석하고 필터링할 수 있습니다. 이러한 필터를 조합하여 적용할 수도 있습니다. Storage Lens 그룹을 S3 Storage Lens 대시보드에 연결하면 Amazon S3 Storage Lens 그룹별로 집계된 S3 Storage Lens 지표를 대시보드에서 직접 볼 수 있습니다.

예를 들어, 객체 크기나 사용 기간 범위를 기준으로 지표를 필터링하여 스토리지의 어느 부분이 소형 객체로 구성되어 있는지 확인할 수도 있습니다. 그런 다음 이 정보를 S3 Intelligent-Tiering 또는 S3 Lifecycle과 함께 사용하여 비용 및 스토리지 최적화를 위해 소형 객체를 다른 스토리지 클래스로 전환할 수 있습니다.

주제

- S3 스토리지 렌즈 그룹 작동 방식
- Storage Lens 그룹 사용

S3 스토리지 렌즈 그룹 작동 방식

Storage Lens 그룹을 사용하면 객체 메타데이터를 기반으로 하는 사용자 지정 필터를 사용하여 지표를 집계할 수 있습니다. 사용자 지정 필터를 정의할 때 접두사, 접미사, 객체 태그, 객체 크기, 객체 수명 또는 이러한 사용자 지정 필터의 조합을 사용할 수 있습니다. Storage Lens 그룹을 만들 때 단일 필터 또는 여러 필터 조건을 포함할 수도 있습니다. 필터 조건을 여러 개 지정하려면 `And` 또는 `Or` 논리 연산자를 사용합니다.

Storage Lens 그룹을 만들고 구성하면 Storage Lens 그룹 자체가 그룹을 연결하는 대시보드에서 사용자 지정 필터 역할을 합니다. 그런 다음 대시보드에서 Storage Lens 그룹 필터를 사용하여 그룹에서 정의한 사용자 지정 필터를 기반으로 스토리지 지표를 가져올 수 있습니다.

S3 Storage Lens 대시보드에서 Storage Lens 그룹의 데이터를 보려면 그룹을 만든 후 그룹을 대시보드에 연결해야 합니다. Storage Lens 그룹이 Storage Lens 대시보드에 연결되면 대시보드는 48시간

이내에 스토리지 사용 지표를 수집합니다. 그런 다음 Storage Lens 대시보드에서 이 데이터를 시각화하거나 지표 내보내기를 통해 내보낼 수 있습니다. Storage Lens 그룹을 대시보드에 연결하는 것을 잊어버리면 Storage Lens 그룹 데이터가 캡처되거나 표시되지 않습니다.

> **Note**
>
> - S3 Storage Lens 그룹을 만들면 AWS 리소스가 생성됩니다. 따라서 각 Storage Lens 그룹에는 고유한 Amazon 리소스 이름(ARN)이 있으며, 이를 S3 Storage Lens 대시보드에 연결하거나 S3 Storage Lens 대시보드에서 제외할 때 지정할 수 있습니다.
> - Storage Lens 그룹이 대시보드에 연결되어 있지 않은 경우 Storage Lens 그룹을 만드는 데 추가 요금이 발생하지 않습니다.
> - S3 Storage Lens는 일치하는 모든 Storage Lens 그룹에 속한 객체의 사용량 지표를 집계합니다. 따라서 객체가 두 개 이상의 Storage Lens 그룹에 대한 필터 조건과 일치하면 스토리지 사용량 전체에서 동일한 객체에 대한 반복 횟수가 표시됩니다.

지원되는 AWS 리전 목록에서 지정된 홈 리전의 계정 수준에서 Storage Lens 그룹을 만들 수 있습니다. 그런 다음 대시보드가 동일한 AWS 계정 및 홈 리전에 있는 한 Storage Lens 그룹을 여러 Storage Lens 대시보드에 연결할 수 있습니다. 각 AWS 계정에서 홈 리전마다 최대 50개의 Storage Lens 그룹을 만들 수 있습니다.

Amazon S3 콘솔, AWS Command Line Interface(AWS CLI), AWS SDK 또는 Amazon S3 REST API를 사용하여 S3 스토리지 렌즈 그룹을 만들고 관리할 수 있습니다.

주제

- Storage Lens 그룹 집계 지표 보기
- 스토리지 렌즈 그룹 권한
- Storage Lens 그룹 구성
- AWS 리소스 태그
- Storage Lens 그룹 지표 내보내기

Storage Lens 그룹 집계 지표 보기

Storage Lens 그룹을 대시보드에 연결하여 Storage Lens 그룹의 집계된 지표를 볼 수 있습니다. 연결하려는 Storage Lens 그룹은 대시보드 계정의 지정된 홈 리전 내에 있어야 합니다.

Storage Lens 그룹을 대시보드에 연결하려면 대시보드 구성의 Storage Lens 그룹 집계 섹션에서 그룹을 지정해야 합니다. Storage Lens 그룹이 여러 개 있는 경우 Storage Lens 그룹 집계 결과를 필터링하여 원하는 그룹만 포함하거나 제외할 수 있습니다. 대시보드에 그룹을 연결하는 방법에 대한 자세한 내용은 the section called “Storage Lens 그룹 연결 또는 제거”를 참조하십시오.

그룹을 연결하면 48시간 이내에 대시보드에 추가 Storage Lens 그룹 집계 데이터가 표시됩니다.

> **Note**
>
> Storage Lens 그룹의 집계된 지표를 보려면 그룹을 S3 Storage Lens 대시보드에 연결해야 합니다.

스토리지 렌즈 그룹 권한

스토리지 렌즈 그룹을 사용하려면 S3 스토리지 렌즈 그룹 작업에 액세스할 수 있도록 AWS Identity and Access Management(IAM)에 특정 권한이 있어야 합니다. 이러한 권한을 부여하려면 자격 증명 기반 IAM 정책을 사용할 수 있습니다. 이 정책을 IAM 사용자, 그룹 또는 역할에 연결하여 권한을 부여할 수 있습니다. 이러한 권한에는 Storage Lens 그룹을 만들거나 삭제하고, 구성을 보거나, 태그를 관리하는 기능이 포함될 수 있습니다.

권한을 부여하는 IAM 사용자 또는 역할은 Storage Lens 그룹을 만들거나 소유한 계정에 속해야 합니다.

Storage Lens 그룹을 사용하고 Storage Lens 그룹 지표를 보려면 먼저 S3 Storage Lens를 사용할 수 있는 적절한 권한이 있어야 합니다. 자세한 내용은 the section called “권한 설정” 섹션을 참조하십시오.

S3 Storage Lens 그룹을 만들고 관리하려면 수행하려는 작업에 따라 다음과 같은 IAM 권한이 있어야 합니다.

작업	IAM 권한
새 Storage Lens 그룹 만들기	`s3:CreateStorageLensGroup`
태그를 사용하여 새 Storage Lens 그룹 만들기	`s3:CreateStorageLensGroup` , `s3:TagResource`
기존 Storage Lens 그룹 업데이트	`s3:UpdateStorageLensGroup`
Storage Lens 그룹 구성의 세부 정보 반환	`s3:GetStorageLensGroup`

작업	IAM 권한
홈 리전의 모든 Storage Lens 그룹 나열	`s3:ListStorageLensGroups`
Storage Lens 그룹 삭제	`s3:DeleteStorageLensGroup`
Storage Lens 그룹에 추가된 태그 나열	`s3:ListTagsForResource`
기존 Storage Lens 그룹의 Storage Lens 그룹 태그 추가 또는 업데이트	`s3:TagResource`
Storage Lens 그룹에서 태그 삭제	`s3:UntagResource`

다음은 Storage Lens 그룹을 만드는 계정에서 IAM 정책을 구성하는 방법의 예입니다. 이 정책을 사용하려면 *us-east-1*을 Storage Lens 그룹이 위치한 홈 리전으로 바꾸십시오. *111122223333*을 AWS 계정 ID로 바꾸고 *example-storage-lens-group*을 Storage Lens 그룹의 이름으로 바꾸십시오. 이러한 권한을 모든 Storage Lens 그룹에 적용하려면 *example-storage-lens-group*을 *로 바꾸십시오.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EXAMPLE-Statement-ID",
            "Effect": "Allow",
            "Action": [
                "s3:CreateStorageLensGroup",
                "s3:UpdateStorageLensGroup",
                "s3:GetStorageLensGroup",
                "s3:ListStorageLensGroups",
                "s3:DeleteStorageLensGroup,
                "s3:TagResource",
                "s3:UntagResource",
                "s3:ListTagsForResource"
                ],
            "Resource": "arn:aws:s3:us-east-1:111122223333:storage-lens-group/example-storage-lens-group"
        }
    ]
}
```

S3 스토리지 렌즈 권한에 대한 자세한 내용은 Amazon S3 Storage Lens 권한 설정 섹션을 참조하십시오. IAM 정책 언어에 대한 자세한 내용은 Amazon S3의 정책 및 권한를 참조하십시오.

Storage Lens 그룹 구성

S3 Storage Lens 그룹 이름

대시보드에 연결할 그룹을 쉽게 결정할 수 있도록 Storage Lens 그룹에 용도를 나타내는 이름을 지정하는 것이 좋습니다. Storage Lens 그룹을 대시보드에 연결하려면 대시보드 구성의 Storage Lens 그룹 집계 섹션에서 그룹을 지정해야 합니다.

스토리지 렌즈 그룹 이름은 계정 내에서 고유해야 합니다. 64자를 넘지 않아야 하며, 문자(a~z, A~Z), 숫자(0~9), 하이픈(-) 및 밑줄(_)만 포함할 수 있습니다.

홈 리전

홈 리전은 Storage Lens 그룹이 만들어지고 유지되는 AWS 리전입니다. Storage Lens 그룹은 Amazon S3 Storage Lens 대시보드와 동일한 홈 리전에 만들어집니다. Storage Lens 그룹 구성 및 지표도 이 리전에 저장됩니다. 홈 리전에 최대 50개의 Storage Lens 그룹을 만들 수 있습니다.

Storage Lens 그룹을 만든 후에는 홈 리전을 편집할 수 없습니다.

범위

Storage Lens 그룹에 객체를 포함하려면 해당 객체가 Amazon S3 Storage Lens 대시보드의 범위 내에 있어야 합니다. Storage Lens 대시보드의 범위는 S3 Storage Lens 대시보드 구성의 대시보드 범위에 포함한 버킷에 따라 결정됩니다.

객체에 다양한 필터를 사용하여 Storage Lens 그룹의 범위를 정의할 수 있습니다. S3 Storage Lens 대시보드에서 이러한 Storage Lens 그룹 지표를 보려면 객체가 Storage Lens 그룹에 포함된 필터와 일치해야 합니다. 예를 들어 Storage Lens 그룹에 접두사 `marketing`과 접미사 `.png`가 있는 객체가 포함되어 있지만 해당 기준과 일치하는 객체가 없다고 가정해 보겠습니다. 이 경우 이 Storage Lens 그룹의 지표는 일일 지표 내보내기에서 생성되지 않으며 이 그룹의 지표는 대시보드에 표시되지 않습니다.

필터

S3 Storage Lens 그룹에서는 다음 필터를 사용할 수 있습니다.

- 접두사 - 포함된 객체의 접두사를 지정합니다. 접두사는 객체 키 이름의 시작 부분에 있는 문자열입니다. 예를 들어, 접두사 필터의 `images` 값에는 `images/`, `images-marketing` 및 `images/production` 접두사 중 하나가 있는 객체가 포함됩니다. 접두사의 최대 길이는 1,024바이트입니다.

- 접미사 - 포함된 객체의 접미사(예: `.png`, `.jpeg` 또는 `.csv`)를 지정합니다. 접미사의 최대 길이는 1,024바이트입니다.
- 객체 태그 - 필터링 기준으로 사용할 객체 태그 목록을 지정합니다. 태그 키는 유니코드 128자를 초과할 수 없으며 태그 값은 유니코드 256자를 초과할 수 없습니다. 객체 태그 값 필드를 비워 두면 S3 Storage Lens 그룹은 객체를 태그 값이 비어 있는 다른 객체하고만 매칭한다는 점에 유의하세요.
- 수명 - 포함된 객체의 객체 연령 범위를 일 단위로 지정합니다. 정수만 지원됩니다.
- 크기 - 포함된 객체의 객체 크기 범위를 바이트 단위로 지정합니다. 정수만 지원됩니다. 허용되는 최대값은 5TB입니다.

Storage Lens 그룹 객체 태그

최대 10개의 객체 태그 필터를 포함하는 Storage Lens 그룹을 만들 수 있습니다. 다음 예제는 *Marketing-Department*라는 이름의 Storage Lens 그룹에 대한 필터로 두 개의 객체 태그 키-값 페어를 포함합니다. 이 예제를 사용하려면 *Marketing-Department*를 그룹 이름으로 바꾸고, *object-tag-key-1*, *object-tag-value-1* 등을 필터링하려는 객체 태그 키-값 페어로 바꿉니다.

```
{
    "Name": "Marketing-Department",
    "Filter": {
     "MatchAnyTag":[
                {
                    "Key": "object-tag-key-1",
                    "Value": "object-tag-value-1"
                },
                {
                    "Key": "object-tag-key-2",
                    "Value": "object-tag-value-2"
                }
            ]
    }
}
```

논리 연산자(**And** 또는 **Or**)

Storage Lens 그룹에 여러 필터 조건을 포함하려면 논리 연산자(`And`또는`Or`)를 사용할 수 있습니다. 다음 예제에서 *Marketing-Department*라는 이름의 Storage Lens 그룹에는 `Prefix`, `ObjectAge`, `ObjectSize` 필터를 포함하는 `And` 연산자가 있습니다. `And` 연산자가 사용되므로 이러한 필터 조건을 모두 충족하는 객체만 Storage Lens 그룹의 범위에 포함됩니다.

이 예제를 사용하려면 *user input placeholders*를 필터링 기준으로 사용할 값으로 바꾸십시오.

```
{
    "Name": "Marketing-Department",
    "Filter": {
        "And": {
            "MatchAnyPrefix": [
                "prefix-1",
                "prefix-2",
                "prefix-3/sub-prefix-1"
            ],
             "MatchObjectAge": {
                "DaysGreaterThan": 10,
                "DaysLessThan": 60
            },
            "MatchObjectSize": {
                "BytesGreaterThan": 10,
                "BytesLessThan": 60
            }
        }
    }
}
```

> **Note**
>
> 필터의 조건 중 하나와 일치하는 객체를 포함하려면 이 예제에서 `And` 논리 연산자를 `Or` 논리 연산자로 바꾸십시오.

AWS 리소스 태그

각 S3 Storage Lens 그룹은 고유한 Amazon 리소스 이름(ARN)이 있는 AWS 리소스로 간주됩니다. 따라서 Storage Lens 그룹을 구성할 때 선택적으로 그룹에 AWS 리소스 태그를 추가할 수 있습니다. 각 Storage Lens 그룹에 최대 50개의 태그를 추가할 수 있습니다. 태그가 있는 Storage Lens 그룹을 만들려면 `s3:CreateStorageLensGroup` 및 `s3:TagResource` 권한이 있어야 합니다.

AWS 리소스 태그를 사용하여 부서, 사업부 또는 프로젝트에 따라 리소스를 분류할 수 있습니다. 이는 동일한 유형의 리소스가 많을 때 유용합니다. 태그를 적용하면 지정한 태그를 기반으로 특정 Storage Lens 그룹을 빠르게 식별할 수 있습니다. 또한 태그를 사용하여 비용을 추적 및 할당할 수 있습니다.

또한 Storage Lens 그룹에 AWS 리소스 태그를 추가하면 속성 기반 액세스 제어(ABAC)가 활성화됩니다. ABAC는 속성(이 경우 태그)을 기반으로 권한을 정의하는 권한 부여 전략입니다. 또한 IAM 정책의 리소스 태그를 지정하는 조건을 사용하여 해당 리소스의 태그를 기반으로 AWS 리소스에 대한 액세스를 제어할 수 있습니다.

태그 키와 값을 편집할 수 있으며 언제든지 리소스에서 태그를 제거할 수 있습니다. 또한 다음 제한 사항을 고려해야 합니다.

- 태그 키와 태그 값은 대/소문자를 구분합니다.
- 해당 리소스에 대해 키가 기존 태그와 동일한 태그를 추가하는 경우 새 값이 이전 값을 덮어씁니다.
- 리소스를 삭제하면 리소스 태그도 삭제됩니다.
- AWS 리소스 태그에 비공개 또는 민감한 데이터를 포함하지 마십시오.
- 시스템 태그(또는 `aws:`로 시작하는 태그 키를 포함하는 태그)는 지원되지 않습니다.
- 각 태그 키의 길이는 128자를 초과할 수 없습니다. 각 태그 값의 길이는 256자를 초과할 수 없습니다.

Storage Lens 그룹 지표 내보내기

S3 Storage Lens 그룹 지표는 Storage Lens 그룹이 연결된 대시보드에 대한 Amazon S3 Storage Lens 지표 내보내기에 포함됩니다. Storage Lens 지표 내보내기 기능에 대한 일반 정보는 데이터 내보내기를 사용하여 Amazon S3 스토리지 렌즈 지표 보기를 참조하십시오.

Storage Lens 그룹의 지표 내보내기에는 Storage Lens 그룹을 연결한 대시보드의 범위 내에 있는 모든 S3 Storage Lens 지표가 포함됩니다. 내보내기에는 Storage Lens 그룹에 대한 추가 지표 데이터도 포함됩니다.

Storage Lens 그룹을 만든 후에는 그룹이 연결된 대시보드의 지표 내보내기를 구성할 때 선택한 버킷으로 매일 지표 내보내기가 전송됩니다. 첫 번째 지표 내보내기를 받는 데 최장 48시간이 걸릴 수 있습니다.

일일 내보내기에서 지표를 생성하려면 객체가 Storage Lens 그룹에 포함하는 필터와 일치해야 합니다. Storage Lens 그룹에 포함한 필터와 일치하는 객체가 없는 경우 지표가 생성되지 않습니다. 하지만 객체가 두 개 이상의 Storage Lens 그룹과 일치하는 경우 해당 객체가 지표 내보내기에 표시될 때 각 그룹별로 별도로 나열됩니다.

대시보드에 대한 지표 내보내기의 `record_type` 열에서 다음 값 중 하나를 찾아 Storage Lens 그룹의 지표를 식별할 수 있습니다.

- `STORAGE_LENS_GROUP_BUCKET`

- STORAGE_LENS_GROUP_ACCOUNT

record_value 열에는 Storage Lens 그룹(예: arn:aws:s3:*us-east-1*:*111122223333*:storage-lens-group/*Marketing-Department*)의 리소스 ARN이 표시됩니다.

Storage Lens 그룹 사용

Amazon S3 Storage Lens는 객체 메타데이터를 기반으로 사용자 지정 필터를 사용하여 지표를 집계합니다. 접두사, 접미사, 객체 태그, 객체 크기 또는 객체 수명을 사용하여 S3 Storage Lens 지표를 분석하고 필터링할 수 있습니다. Amazon S3 Storage Lens 3 버킷에서 사용량을 분류할 수도 있습니다. 따라서 S3 스토리지를 더 잘 이해하고 최적화할 수 있습니다.

Storage Lens 그룹의 데이터 시각화를 시작하려면 먼저 Storage Lens 그룹을 S3 Storage Lens 대시보드에 연결해야 합니다. 대시보드에서 Storage Lens 그룹을 관리해야 하는 경우 대시보드 구성을 편집할 수 있습니다. 계정에 속한 Storage Lens 그룹을 확인하려면 해당 그룹을 나열할 수 있습니다. 대시보드에 연결된 Storage Lens 그룹을 확인하려면 대시보드의 Storage Lens 그룹 탭을 언제든지 확인할 수 있습니다. 기존 Storage Lens 그룹의 범위를 검토하거나 업데이트하려면 해당 세부 정보를 볼 수 있습니다. Storage Lens 그룹을 영구적으로 삭제할 수도 있습니다.

권한을 관리하기 위해 사용자 정의 AWS 리소스 태그를 만들어 Storage Lens 그룹에 추가할 수 있습니다. AWS 리소스 태그를 사용하여 부서, 사업부 또는 프로젝트에 따라 리소스를 분류할 수 있습니다. 이는 동일한 유형의 리소스가 많을 때 유용합니다. 태그를 적용하면 지정한 태그를 기반으로 특정 스토리지 렌즈 그룹을 빠르게 식별할 수 있습니다.

또한 스토리지 렌즈 그룹에 AWS 리소스 태그를 추가하면 속성 기반 액세스 제어(ABAC)가 활성화됩니다. ABAC는 속성(이 경우 태그)을 기반으로 권한을 정의하는 권한 부여 전략입니다. 또한 IAM 정책의 리소스 태그를 지정하는 조건을 사용하여 해당 리소스의 태그를 기반으로 AWS 리소스에 대한 액세스를 제어할 수 있습니다.

주제

- Storage Lens 그룹 만들기
- 대시보드에 S3 Storage Lens 그룹 연결 또는 제거
- Storage Lens 그룹 데이터 시각화하기
- Storage Lens 그룹 업데이트
- Storage Lens 그룹을 통한 AWS 리소스 태그 관리
- 모든 Storage Lens 그룹 목록

- Storage Lens 그룹 세부 정보 보기
- Storage Lens 그룹 삭제

Storage Lens 그룹 만들기

다음 예제는 Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 및 AWS SDK for Java를 사용하여 Amazon S3 Storage Lens 그룹을 만드는 방법을 보여줍니다.

S3 콘솔 사용

Storage Lens 그룹을 만들려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 전환하려는 리전을 선택합니다.
3. 왼쪽 탐색 창에서 스토리지 렌즈 그룹을 선택합니다.
4. Storage Lens 그룹 생성을 선택합니다.
5. 일반에서 홈 리전을 확인하고 Storage Lens 그룹 이름을 입력합니다.
6. 범위에서 Storage Lens 그룹에 적용하려는 필터를 선택합니다. 여러 필터를 적용하려면 필터를 선택한 다음 AND 또는 OR 논리 연산자를 선택합니다.
   - 접두사 필터의 경우 접두사를 선택하고 접두사 문자열을 입력합니다. 여러 접두사를 추가하려면 접두사 추가를 선택합니다. 접두사를 제거하려면 제거할 접두사 옆에 있는 제거를 선택합니다.
   - 객체 태그 필터의 경우 객체 태그를 선택하고 객체의 키-값 쌍을 입력합니다. 그런 다음 태그 추가를 선택합니다. 태그를 제거하려면 제거하려는 태그 옆에 있는 제거를 선택합니다.
   - 접미사 필터의 경우 접미사를 선택하고 접미사 문자열을 입력합니다. 접미사를 여러 개 추가하려면 접미사 추가를 선택합니다. 접미사를 제거하려면 제거할 지문 옆에 있는 제거를 선택합니다.
   - 연령 필터의 경우 객체 연령 범위를 일 단위로 지정합니다. 최소 객체 연령 지정을 선택하고 최소 객체 연령을 입력합니다. 그런 다음 최대 객체 연령 지정을 선택하고 최대 객체 수명을 입력합니다.
   - 크기 필터의 경우 객체 크기 범위와 측정 단위를 지정합니다. 최소 객체 크기 지정을 선택하고 최소 객체 크기를 입력합니다. 최대 객체 크기 지정을 선택하고 최대 객체 크기를 입력합니다.
7. (선택 사항) AWS 리소스 태그의 경우 키-값 쌍을 추가한 다음 태그 추가를 선택합니다.

8. Storage Lens 그룹 생성을 선택합니다.

AWS CLI 사용

다음 예시 AWS CLI 명령은 스토리지 렌즈 그룹을 생성합니다. 이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3control create-storage-lens-group --account-id 111122223333 \
--region us-east-1 --storage-lens-group=file://./marketing-department.json
```

다음 예시 AWS CLI 명령은 두 개의 AWS 리소스 태그가 있는 Storage Lens 그룹을 만듭니다. 이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3control create-storage-lens-group --account-id 111122223333 \
--region us-east-1 --storage-lens-group=file://./marketing-department.json \
--tags Key=k1,Value=v1 Key=k2,Value=v2
```

JSON 구성에 대한 예제는 Storage Lens 그룹 구성 섹션을 참조하십시오.

Java용 AWS SDK 사용

다음 AWS SDK for Java 예제에서는 Storage Lens 그룹을 생성합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

Example - 단일 필터를 사용하여 Storage Lens 그룹 만들기

다음 예제에서는 *Marketing-Department*라는 이름의 스토리지 렌즈 그룹을 만듭니다. 이 그룹에는 연령 범위를 *30*~*90*로 지정하는 객체 연령 필터가 있습니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.MatchObjectAge;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;
```

```
public class CreateStorageLensGroupWithObjectAge {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            StorageLensGroupFilter objectAgeFilter = StorageLensGroupFilter.builder()
                    .matchObjectAge(MatchObjectAge.builder()
                            .daysGreaterThan(30)
                            .daysLessThan(90)
                            .build())
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(objectAgeFilter)
                    .build();

            CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
        } 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();
        }
    }
}
```

Example - 여러 필터를 포함하는 **AND** 연산자를 사용하여 Storage Lens 그룹을 만듭니다.

다음 예제에서는 *Marketing-Department*라는 이름의 스토리지 렌즈 그룹을 만듭니다. 이 그룹은 AND 연산자를 사용하여 객체가 모든 필터 조건과 일치해야 함을 나타냅니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.MatchObjectAge;
import software.amazon.awssdk.services.s3control.model.MatchObjectSize;
import software.amazon.awssdk.services.s3control.model.S3Tag;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupAndOperator;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;


public class CreateStorageLensGroupWithAndFilter {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            // Create object tags.
            S3Tag tag1 = S3Tag.builder()
                    .key("object-tag-key-1")
                    .value("object-tag-value-1")
                    .build();
            S3Tag tag2 = S3Tag.builder()
                    .key("object-tag-key-2")
                    .value("object-tag-value-2")
                    .build();

            StorageLensGroupAndOperator andOperator =
 StorageLensGroupAndOperator.builder()
                    .matchAnyPrefix("prefix-1", "prefix-2", "prefix-3/sub-prefix-1")
                    .matchAnySuffix(".png", ".gif", ".jpg")
                    .matchAnyTag(tag1, tag2)
```

```
                    .matchObjectAge(MatchObjectAge.builder()
                            .daysGreaterThan(30)
                            .daysLessThan(90).build())
                    .matchObjectSize(MatchObjectSize.builder()
                            .bytesGreaterThan(1000L)
                            .bytesLessThan(6000L).build())
                    .build();

            StorageLensGroupFilter andFilter = StorageLensGroupFilter.builder()
                    .and(andOperator)
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(andFilter)
                    .build();

            CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
        } 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();
        }
    }
}
```

Example - 여러 필터를 포함하는 **OR** 연산자를 사용하여 Storage Lens 그룹을 만듭니다.

다음 예제에서는 *Marketing-Department*라는 이름의 스토리지 렌즈 그룹을 만듭니다. 이 그룹은 OR 연산자를 사용하여 접두사 필터(*prefix-1*,*prefix-2*,*prefix3/sub-prefix-1*) 또는 크기 범

위가 *1000*바이트에서 *6000*바이트 사이인 객체 크기 필터를 적용합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.MatchObjectSize;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupOrOperator;

public class CreateStorageLensGroupWithOrFilter {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            StorageLensGroupOrOperator orOperator =
 StorageLensGroupOrOperator.builder()
                    .matchAnyPrefix("prefix-1", "prefix-2", "prefix-3/sub-prefix-1")
                    .matchObjectSize(MatchObjectSize.builder()
                            .bytesGreaterThan(1000L)
                            .bytesLessThan(6000L)
                            .build())
                    .build();

            StorageLensGroupFilter orFilter = StorageLensGroupFilter.builder()
                    .or(orOperator)
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(orFilter)
                    .build();

            CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
```

```
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
        } 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();
        }
    }
}
```

Example - 단일 필터와 두 개의 AWS 리소스 태그를 사용하여 Storage Lens 그룹을 만듭니다.

다음 예제에서는 접미사 필터가 있는 *Marketing-Department*라는 이름의 Storage Lens 그룹을 만듭니다. 또한 이 예제는 Storage Lens 그룹에 두 개의 AWS 리소스 태그를 추가합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;
import software.amazon.awssdk.services.s3control.model.Tag;

public class CreateStorageLensGroupWithResourceTags {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
```

```
            // Create AWS resource tags.
            Tag resourceTag1 = Tag.builder()
                    .key("resource-tag-key-1")
                    .value("resource-tag-value-1")
                    .build();
            Tag resourceTag2 = Tag.builder()
                    .key("resource-tag-key-2")
                    .value("resource-tag-value-2")
                    .build();

            StorageLensGroupFilter suffixFilter = StorageLensGroupFilter.builder()
                    .matchAnySuffix(".png", ".gif", ".jpg")
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(suffixFilter)
                    .build();

            CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
                    .tags(resourceTag1, resourceTag2)
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
        } 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();
        }
    }
}
```

JSON 구성에 대한 예제는 Storage Lens 그룹 구성 섹션을 참조하십시오.

대시보드에 S3 Storage Lens 그룹 연결 또는 제거

Amazon S3 Storage Lens의 고급 티어로 업그레이드한 후 Storage Lens 그룹을 대시보드에 연결할 수 있습니다. Storage Lens 그룹이 여러 개인 경우 원하는 그룹을 포함하거나 제외할 수 있습니다.

Storage Lens 그룹은 대시보드 계정의 지정된 홈 리전 내에 있어야 합니다. Storage Lens 그룹을 대시보드에 연결하면 48시간 이내에 지표 내보내기에서 추가 Storage Lens 그룹 집계 데이터를 받게 됩니다.

> **Note**
>
> Storage Lens 그룹의 집계된 지표를 보려면 Storage Lens 대시보드에 연결해야 합니다. Storage Lens 그룹 JSON 구성 파일의 예는 JSON의 Storage Lens 그룹을 사용한 S3 Storage Lens 예제 구성를 참조하십시오.

S3 콘솔 사용

Storage Lens 그룹을 Storage Lens 대시보드에 연결하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창의 스토리지 렌즈에서 대시보드를 선택합니다.
3. Storage Lens 그룹을 연결하려는 Storage Lens 대시보드의 옵션 버튼을 선택합니다.
4. 편집을 선택합니다.
5. 지표 선택(Metrics selection)에서 고급 지표 및 권장 사항(Advanced metrics and recommendations)을 선택합니다.
6. Storage Lens 그룹 집계를 선택합니다.

   > **Note**
   >
   > 기본적으로 고급 지표도 선택됩니다. 하지만 Storage Lens 그룹 데이터를 집계하는 데 필요하지 않으므로 이 설정을 선택 취소할 수도 있습니다.

7. Storage Lens 그룹 집계까지 아래로 스크롤하여 데이터 집계에 포함하거나 제외하려는 Storage Lens 그룹 또는 그룹을 지정합니다. 다음 필터링 옵션을 사용할 수 있습니다.

- 특정 Storage Lens 그룹을 포함하려면 Storage Lens 그룹 포함을 선택합니다. 포함할 Storage Lens 그룹에서 Storage Lens 그룹을 선택합니다.
- 모든 Storage Lens 그룹을 포함하려면 이 계정에서 홈 리전의 모든 Storage Lens 그룹 포함을 선택합니다.
- 특정 Storage Lens 그룹을 제외하려면 Storage Lens 그룹 제외를 선택합니다. 제외할 Storage Lens 그룹에서 제외하려는 Storage Lens 그룹을 선택합니다.

8. 변경 사항 저장을 선택합니다. Storage Lens 그룹을 올바르게 구성한 경우 48시간 내에 대시보드에 추가 Storage Lens 그룹 집계 데이터가 표시됩니다.

S3 Storage Lens 대시보드에서 Storage Lens 그룹을 제거하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창의 스토리지 렌즈에서 대시보드를 선택합니다.
3. Storage Lens 그룹을 제거하려는 Storage Lens 대시보드의 옵션 버튼을 선택합니다.
4. 대시보드 구성 보기를 선택합니다.
5. 편집을 선택합니다.
6. 아래로 스크롤하여 지표 선택 섹션을 찾습니다.
7. Storage Lens 그룹 집계에서 제거하려는 Storage Lens 그룹 옆의 X를 선택합니다. 그러면 Storage Lens 그룹이 제거됩니다.

   대시보드에 모든 Storage Lens 그룹을 포함한 경우 이 계정에 홈 리전의 모든 Storage Lens 그룹 포함 옆의 확인란을 선택 취소하십시오.
8. 변경 사항 저장을 선택합니다.

> **Note**
>
> 대시보드에 구성 업데이트가 반영되는 데 최장 48시간이 걸립니다.

Java용 AWS SDK 사용

Example - 모든 Storage Lens 그룹을 대시보드에 연결

다음 Java용 SDK 예제는 계정 *111122223333*의 모든 Storage Lens 그룹을 *DashBoardConfigurationId* 대시보드에 연결합니다.

```
package aws.example.s3control;


import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensGroupLevel;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateDashboardWithStorageLensGroups {
    public static void main(String[] args) {
        String configurationId = "ExampleDashboardConfigurationId";
        String sourceAccountId = "111122223333";

        try {
            StorageLensGroupLevel storageLensGroupLevel = new StorageLensGroupLevel();

            AccountLevel accountLevel = new AccountLevel()
                    .withBucketLevel(new BucketLevel())
                    .withStorageLensGroupLevel(storageLensGroupLevel);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();
```

```
            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );
        } 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();
        }
    }
}
```

Example - 2개의 Storage Lens 그룹을 대시보드에 연결

다음 AWS SDK for Java 예제는 두 개의 Storage Lens 그룹(*StorageLensGroupName1* 및 *StorageLensGroupName2*)을 *ExampleDashboardConfigurationId* 대시보드에 연결합니다.

```
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.AccountLevel;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensGroupLevel;
import com.amazonaws.services.s3control.model.StorageLensGroupLevelSelectionCriteria;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateDashboardWith2StorageLensGroups {
    public static void main(String[] args) {
        String configurationId = "ExampleDashboardConfigurationId";
```

```
        String storageLensGroupName1 = "StorageLensGroupName1";
        String storageLensGroupName2 = "StorageLensGroupName2";
        String sourceAccountId = "111122223333";

        try {
            StorageLensGroupLevelSelectionCriteria selectionCriteria = new
StorageLensGroupLevelSelectionCriteria()
                    .withInclude(
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
+ ":storage-lens-group/" + storageLensGroupName1,
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
+ ":storage-lens-group/" + storageLensGroupName2);

            System.out.println(selectionCriteria);
            StorageLensGroupLevel storageLensGroupLevel = new StorageLensGroupLevel()
                    .withSelectionCriteria(selectionCriteria);

            AccountLevel accountLevel = new AccountLevel()
                    .withBucketLevel(new BucketLevel())
                    .withStorageLensGroupLevel(storageLensGroupLevel);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );
        } 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();
        }
    }
}
```

Example - 제외 항목 이외의 모든 Storage Lens 그룹 연결

다음 Java용 SDK 예제는 지정된 두 개의 그룹(*StorageLensGroupName1* and *StorageLensGroupName2*)을 제외한 모든 Storage Lens 그룹을 *ExampleDashboardConfigurationId* 대시보드에 연결합니다.

```
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.AccountLevel;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensGroupLevel;
import com.amazonaws.services.s3control.model.StorageLensGroupLevelSelectionCriteria;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateDashboardWith2StorageLensGroupsExcluded {
    public static void main(String[] args) {
        String configurationId = "ExampleDashboardConfigurationId";
        String storageLensGroupName1 = "StorageLensGroupName1";
        String storageLensGroupName2 = "StorageLensGroupName2";
        String sourceAccountId = "111122223333";

        try {
            StorageLensGroupLevelSelectionCriteria selectionCriteria = new
 StorageLensGroupLevelSelectionCriteria()
                    .withInclude(
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
 + ":storage-lens-group/" + storageLensGroupName1,
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
 + ":storage-lens-group/" + storageLensGroupName2);
```

```
            System.out.println(selectionCriteria);
            StorageLensGroupLevel storageLensGroupLevel = new StorageLensGroupLevel()
                    .withSelectionCriteria(selectionCriteria);

            AccountLevel accountLevel = new AccountLevel()
                    .withBucketLevel(new BucketLevel())
                    .withStorageLensGroupLevel(storageLensGroupLevel);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );
        } 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();
        }
    }
}
```

Storage Lens 그룹 데이터 시각화하기

Amazon S3 Storage Lens 대시보드에 그룹을 연결하여 Storage Lens 그룹 데이터를 시각화할 수 있습니다. 대시보드 구성의 Storage Lens 그룹 집계에 Storage Lens 그룹을 포함한 후 Storage Lens 그룹 데이터가 대시보드에 표시되는 데 최장 48시간이 걸릴 수 있습니다.

대시보드 구성이 업데이트되면 새로 연결된 모든 Storage Lens 그룹이 Storage Lens 그룹 탭 아래의 사용 가능한 리소스 목록에 나타납니다. 개요 탭에서 데이터를 다른 차원으로 분할하여 스토리지 사용량을 더 자세히 분석할 수도 있습니다. 예를 들어 상위 3개 범주에 나열된 항목 중 하나를 선택하고 분석 기준을 선택하여 데이터를 다른 차원으로 분할할 수 있습니다. 필터 자체와 동일한 차원을 적용할 수는 없습니다.

> **Note**
>
> Storage Lens 그룹 필터를 접두사 필터와 함께 적용하거나 그 반대로 적용할 수 없습니다. 또한 프리픽스 필터를 사용하여 Storage Lens 그룹을 더 자세히 분석할 수 없습니다.

Amazon S3 Storage Lens 대시보드의 Storage Lens 그룹 탭을 사용하여 대시보드에 연결된 Storage Lens 그룹의 데이터 시각화를 사용자 지정할 수 있습니다. 대시보드에 연결된 일부 Storage Lens 그룹 또는 모든 Storage Lens 그룹의 데이터를 시각화할 수 있습니다.

S3 Storage Lens 대시보드의 Storage Lens 그룹 데이터를 시각화할 때는 다음 사항에 유의하십시오.

- S3 Storage Lens는 일치하는 모든 Storage Lens 그룹에 속한 객체의 사용량 지표를 집계합니다. 따라서 객체가 두 개 이상의 Storage Lens 그룹에 대한 필터 조건과 일치하면 스토리지 사용량 전체에서 동일한 객체에 대한 반복 횟수가 표시됩니다.
- 객체는 Storage Lens 그룹에 포함된 필터와 일치해야 합니다. Storage Lens 그룹에 포함한 필터와 일치하는 객체가 없으면 지표가 생성되지 않습니다. 할당되지 않은 객체가 있는지 확인하려면 대시보드의 계정 수준 및 버킷 수준에서 총 객체 수를 확인하십시오.

Storage Lens 그룹 업데이트

다음 예제에서는 Amazon S3 스토리지 렌즈 그룹을 업데이트하는 방법을 보여줍니다. Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 및 AWS SDK for Java을 사용하여 Storage Lens 그룹을 업데이트할 수 있습니다.

S3 콘솔 사용

Storage Lens 그룹을 업데이트하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 스토리지 렌즈 그룹을 선택합니다.
3. 스토리지 렌즈 그룹에서 업데이트하려는 스토리지 렌즈 그룹을 선택합니다.

4. 범위에서 편집을 선택합니다.
5. 범위 페이지에서 Storage Lens 그룹에 적용하려는 필터를 선택합니다. 여러 필터를 적용하려면 필터를 선택하고 AND 또는 OR 논리 연산자를 선택합니다.

   - 접두사 필터의 경우 접두사를 선택하고 접두사 문자열을 입력합니다. 여러 접두사를 추가하려면 접두사 추가를 선택합니다. 접두사를 제거하려면 제거할 접두사 옆에 있는 제거를 선택합니다.
   - 객체 태그 필터에 객체의 키-값 페어를 입력합니다. 그런 다음 태그 추가를 선택합니다. 태그를 제거하려면 제거하려는 태그 옆에 있는 제거를 선택합니다.
   - 접미사 필터의 경우 접미사를 선택하고 접미사 문자열을 입력합니다. 접미사를 여러 개 추가하려면 접미사 추가를 선택합니다. 접미사를 제거하려면 제거할 지문 옆에 있는 제거를 선택합니다.
   - 연령 필터의 경우 객체 연령 범위를 일 단위로 지정합니다. 최소 객체 연령 지정을 선택하고 최소 객체 연령을 입력합니다. 최대 객체 연령 지정에는 최대 객체 연령을 입력합니다.
   - 크기 필터의 경우 객체 크기 범위와 측정 단위를 지정합니다. 최소 객체 크기 지정을 선택하고 최소 객체 크기를 입력합니다. 최대 객체 크기 지정에는 최대 객체 크기를 입력합니다.
6. 변경 사항 저장을 선택합니다. Storage Lens 그룹에 대한 세부 정보 페이지가 나타납니다.
7. (선택 사항) 새 AWS 리소스 태그를 추가하려면 AWS 리소스 태그 섹션으로 스크롤한 다음 태그 추가를 선택합니다. 태그 추가 페이지가 나타납니다.

   새 키-값 페어를 추가한 다음 변경 내용 저장을 선택합니다. Storage Lens 그룹에 대한 세부 정보 페이지가 나타납니다.
8. (선택 사항) 기존 AWS 리소스 태그를 제거하려면 AWS 리소스 태그 섹션으로 스크롤하여 리소스 태그를 선택합니다. 그런 다음 삭제를 선택합니다. AWS 태그 삭제 대화 상자가 나타납니다.

   AWS 리소스 태그를 영구 삭제하려면 삭제를 다시 선택합니다.

> **Note**
>
> AWS 리소스 태그를 영구 삭제한 후에는 복원할 수 없습니다.

AWS CLI 사용

다음 AWS CLI 예제 명령은 *marketing-department*라는 Storage Lens 그룹의 구성 세부 정보를 반환합니다. 이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3control get-storage-lens-group --account-id 111122223333 \
--region us-east-1 --name marketing-department
```

다음 AWS CLI 예제에서는 Storage Lens 그룹을 업데이트합니다. 이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3control update-storage-lens-group --account-id 111122223333 \
--region us-east-1 --storage-lens-group=file://./marketing-department.json
```

JSON 구성에 대한 예제는 Storage Lens 그룹 구성 섹션을 참조하십시오.

Java용 AWS SDK 사용

다음 AWS SDK for Java 예제에서는 계정 *111122223333* 내의 *Marketing-Department* Storage Lens 그룹의 구성 세부 정보를 반환합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupResponse;

public class GetStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            GetStorageLensGroupRequest getRequest =
 GetStorageLensGroupRequest.builder()
                    .name(storageLensGroupName)
                    .accountId(accountId).build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
```

```
            GetStorageLensGroupResponse response =
 s3ControlClient.getStorageLensGroup(getRequest);
            System.out.println(response);
        } 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();
        }
    }
}
```

다음 예제에서는 계정 *111122223333* 내의 *Marketing-Department*라는 이름의 Storage Lens 그룹을 업데이트합니다. 이 예제에서는 *.png*, *.gif*, *.jpg* 또는 *.jpeg* 접미사와 일치하는 객체를 포함하도록 대시보드 범위를 업데이트합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;
import software.amazon.awssdk.services.s3control.model.UpdateStorageLensGroupRequest;

public class UpdateStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            // Create updated filter.
            StorageLensGroupFilter suffixFilter = StorageLensGroupFilter.builder()
                    .matchAnySuffix(".png", ".gif", ".jpg", ".jpeg")
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
```

```
                    .name(storageLensGroupName)
                    .filter(suffixFilter)
                    .build();

            UpdateStorageLensGroupRequest updateStorageLensGroupRequest =
 UpdateStorageLensGroupRequest.builder()
                    .name(storageLensGroupName)
                    .storageLensGroup(storageLensGroup)
                    .accountId(accountId)
                    .build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.updateStorageLensGroup(updateStorageLensGroupRequest);
        } 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();
        }
    }
}
```

JSON 구성에 대한 예제는 Storage Lens 그룹 구성 섹션을 참조하십시오.

Storage Lens 그룹을 통한 AWS 리소스 태그 관리

각 Amazon S3 Storage Lens 그룹은 고유한 Amazon 리소스 이름(ARN)을 갖고 있는 AWS 리소스로 계산됩니다. 따라서 Storage Lens 그룹을 구성할 때 선택적으로 그룹에 AWS 리소스 태그를 추가할 수 있습니다. 각 Storage Lens 그룹에 최대 50개의 태그를 추가할 수 있습니다. 태그가 있는 Storage Lens 그룹을 만들려면 `s3:CreateStorageLensGroup` 및 `s3:TagResource` 권한이 있어야 합니다.

AWS 리소스 태그를 사용하여 부서, 사업부 또는 프로젝트에 따라 리소스를 분류할 수 있습니다. 이는 동일한 유형의 리소스가 많을 때 유용합니다. 태그를 적용하면 지정한 태그를 기반으로 특정 Storage Lens 그룹을 빠르게 식별할 수 있습니다. 또한 태그를 사용하여 비용을 추적 및 할당할 수 있습니다.

또한 Storage Lens 그룹에 AWS 리소스 태그를 추가하면 속성 기반 액세스 제어(ABAC)가 활성화됩니다. ABAC는 속성(이 경우 태그)을 기반으로 권한을 정의하는 권한 부여 전략입니다. 또한 IAM 정책의

리소스 태그를 지정하는 조건을 사용하여 해당 리소스의 태그를 기반으로 AWS 리소스에 대한 액세스를 제어할 수 있습니다.

태그 키와 값을 편집할 수 있으며 언제든지 리소스에서 태그를 제거할 수 있습니다. 또한 다음 제한 사항을 고려해야 합니다.

- 태그 키와 태그 값은 대/소문자를 구분합니다.
- 해당 리소스에 대해 키가 기존 태그와 동일한 태그를 추가하는 경우 새 값이 이전 값을 덮어씁니다.
- 리소스를 삭제하면 리소스 태그도 삭제됩니다.
- AWS 리소스 태그에 비공개 또는 민감한 데이터를 포함하지 마십시오.
- 시스템 태그(`aws:`로 시작하는 태그 키 포함)는 지원되지 않습니다.
- 각 태그 키의 길이는 128자를 초과할 수 없습니다. 각 태그 값의 길이는 256자를 초과할 수 없습니다.

다음 예제에서는 Storage Lens 그룹에서 AWS 리소스 태그를 사용하는 방법을 보여줍니다.

주제

- Storage Lens 그룹에 AWS 리소스 태그 추가
- Storage Lens 그룹 태그 값 업데이트
- Storage Lens 그룹에서 AWS 리소스 태그 삭제
- Storage Lens 그룹 태그 목록

Storage Lens 그룹에 AWS 리소스 태그 추가

다음 예제에서는 Amazon S3 Storage Lens 그룹에 AWS 리소스 태그를 추가하는 방법을 보여줍니다. Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 및 AWS SDK for Java을 사용하여 리소스 태그를 추가할 수 있습니다.

S3 콘솔 사용

Storage Lens 그룹에 AWS 리소스 태그를 추가하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 스토리지 렌즈 그룹을 선택합니다.

3. Storage Lens 그룹에서 업데이트하려는 Storage Lens 그룹을 선택합니다.
4. AWS 리소스 태그 아래에서 태그 추가를 선택합니다.
5. 태그 추가 페이지에서 새 키-값 쌍을 추가합니다.

> **Note**
>
> 기존 태그와 동일한 키가 있는 새 태그를 추가하면 이전 태그 값을 덮어씁니다.

6. (선택 사항) 새 태그를 두 개 이상 추가하려면 태그 추가를 다시 선택하여 새 항목을 계속 추가합니다. Storage Lens 그룹에 최대 50개의 AWS 리소스 태그를 추가할 수 있습니다.
7. (선택 사항) 새로 추가된 항목을 제거하려면 제거하려는 태그 옆의 제거를 선택합니다.
8. 변경 사항 저장을 선택합니다.

AWS CLI 사용

다음 예제 AWS CLI 명령은 *marketing-department*라는 이름의 기존 Storage Lens 그룹에 두 개의 리소스 태그를 추가합니다. 이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3control tag-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/marketing-
department \
--region us-east-1 --tags Key=k1,Value=v1 Key=k2,Value=v2
```

Java용 AWS SDK 사용

다음 AWS SDK for Java 예제에서는 기존 Storage Lens 그룹에 두 개의 AWS 리소스 태그를 추가합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.Tag;
import software.amazon.awssdk.services.s3control.model.TagResourceRequest;
```

```
public class TagResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            Tag resourceTag1 = Tag.builder()
                .key("resource-tag-key-1")
                .value("resource-tag-value-1")
                .build();
            Tag resourceTag2 = Tag.builder()
                    .key("resource-tag-key-2")
                    .value("resource-tag-value-2")
                    .build();
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .tags(resourceTag1, resourceTag2)
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.tagResource(tagResourceRequest);
        } 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();
        }
    }
}
```

Storage Lens 그룹 태그 값 업데이트

다음 예제는 Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 및 AWS SDK for Java을 사용하여 Storage Lens 그룹 태그 값을 업데이트하는 방법을 보여줍니다.

S3 콘솔 사용

Storage Lens 그룹의 AWS 리소스 태그를 업데이트하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 스토리지 렌즈 그룹을 선택합니다.
3. Storage Lens 그룹에서 업데이트하려는 Storage Lens 그룹을 선택합니다.
4. AWS 리소스 태그에서 업데이트하려는 태그를 선택합니다.
5. 업데이트하려는 키-값 페어와 동일한 키를 사용하여 새 태그 값을 추가합니다. 확인 표시 아이콘을 선택하여 태그 값을 업데이트합니다.

   > **Note**
   >
   > 기존 태그와 동일한 키가 있는 새 태그를 추가하면 이전 태그 값을 덮어씁니다.

6. (선택 사항) 새 태그를 추가하려면 태그 추가를 선택하여 새 항목을 추가합니다. 태그 추가 페이지가 나타납니다.

   Storage Lens 그룹에 최대 50개의 AWS 리소스 태그를 추가할 수 있습니다. 새 태그 추가가 완료되면 변경 사항 저장을 선택합니다.
7. (선택 사항) 새로 추가된 항목을 제거하려면 제거하려는 태그 옆의 제거를 선택합니다. 태그 제거를 마쳤으면 변경 사항 저장을 선택합니다.

AWS CLI 사용

다음 예제 AWS CLI 명령은 *marketing-department*라는 이름의 Storage Lens 그룹에 대한 두 개의 태그 값을 업데이트합니다. 이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3control tag-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/marketing-
department \
--region us-east-1 --tags Key=k1,Value=v3 Key=k2,Value=v4
```

Java용 AWS SDK 사용

다음 AWS SDK for Java 예제에서는 두 개의 Storage Lens 그룹 태그 값을 업데이트합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.Tag;
import software.amazon.awssdk.services.s3control.model.TagResourceRequest;

public class UpdateTagsForResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            Tag updatedResourceTag1 = Tag.builder()
                .key("resource-tag-key-1")
                .value("resource-tag-updated-value-1")
                .build();
            Tag updatedResourceTag2 = Tag.builder()
                    .key("resource-tag-key-2")
                    .value("resource-tag-updated-value-2")
                    .build();
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .tags(updatedResourceTag1, updatedResourceTag2)
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.tagResource(tagResourceRequest);
        } 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();
        }
    }
}
```

Storage Lens 그룹에서 AWS 리소스 태그 삭제

다음 예제에서는 Storage Lens 그룹에서 AWS 리소스 태그를 삭제하는 방법을 보여줍니다. Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 및 AWS SDK for Java를 사용하여 태그를 삭제할 수 있습니다.

S3 콘솔 사용

Storage Lens 그룹에서 AWS 리소스 태그를 삭제하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 스토리지 렌즈 그룹을 선택합니다.
3. Storage Lens 그룹에서 업데이트하려는 Storage Lens 그룹을 선택합니다.
4. AWS 리소스 태그 아래에서 삭제하려는 키-값 페어를 선택합니다.
5. 삭제를 선택합니다. AWS 리소스 태그 삭제 대화 상자가 나타납니다.

> **Note**
>
> 태그를 사용하여 액세스를 제어하는 경우 이 작업을 진행하면 관련 리소스에 영향이 미칠 수 있습니다. 영구 삭제한 태그는 복원할 수 없습니다.

6. 삭제를 선택하여 키-값 페어를 영구적으로 삭제합니다.

AWS CLI 사용

다음 AWS CLI 명령은 기존 Storage Lens 그룹에서 두 개의 AWS 리소스 태그를 삭제합니다. 이 예제 명령을 사용하려면 자체 정보로 *user input placeholders*를 바꿉니다.

```
aws s3control untag-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/Marketing-
Department \
```

```
--region us-east-1 --tag-keys k1 k2
```

Java용 AWS SDK 사용

다음 AWS SDK for Java 예제에서는 Storage Lens 그룹 Amazon 리소스 이름(ARN)에서 계정 *111122223333*에서 지정한 AWS 리소스 태그 두 개를 삭제합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.UntagResourceRequest;

public class UntagResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            String tagKey1 = "resource-tag-key-1";
            String tagKey2 = "resource-tag-key-2";
            UntagResourceRequest untagResourceRequest = UntagResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .tagKeys(tagKey1, tagKey2)
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.untagResource(untagResourceRequest);
        } 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();
```

```
        }
    }
}
```

Storage Lens 그룹 태그 목록

다음 예는 Storage Lens 그룹과 관련된 AWS 리소스 태그를 나열하는 방법을 보여줍니다. Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 및 AWS SDK for Java을 사용하여 태그를 나열할 수 있습니다.

S3 콘솔 사용

Storage Lens 그룹의 태그 및 태그 값 목록을 검토하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 스토리지 렌즈 그룹을 선택합니다.
3. Storage Lens 그룹에서 관심 있는 Storage Lens 그룹을 선택합니다.
4. 아래로 스크롤하여 AWS 리소스 태그 섹션으로 이동합니다. Storage Lens 그룹에 추가된 모든 사용자 정의 AWS 리소스 태그가 해당 태그 값과 함께 나열됩니다.

AWS CLI 사용

다음 AWS CLI 예제 명령은 *marketing-department*라는 이름의 Storage Lens 그룹의 모든 Storage Lens 그룹 태그 값을 나열합니다. 이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3control list-tags-for-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/marketing-
department \
--region us-east-1
```

Java용 AWS SDK 사용

다음 AWS SDK for Java 예제는 지정한 Storage Lens 그룹 Amazon 리소스 이름(ARN)의 Storage Lens 그룹 태그 값을 나열합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
package aws.example.s3control;
```

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.ListTagsForResourceRequest;
import software.amazon.awssdk.services.s3control.model.ListTagsForResourceResponse;

public class ListTagsForResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            ListTagsForResourceRequest listTagsForResourceRequest =
 ListTagsForResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            ListTagsForResourceResponse response =
 s3ControlClient.listTagsForResource(listTagsForResourceRequest);
            System.out.println(response);
        } 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();
        }
    }
}
```

모든 Storage Lens 그룹 목록

다음 예제에서는 AWS 계정 및 홈 리전 내의 모든 Amazon S3 Storage Lens 그룹을 나열하는 방법을 보여줍니다. 이 예제는 Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 및 AWS SDK for Java를 사용하여 모든 Storage Lens 그룹을 나열하는 방법을 보여줍니다.

S3 콘솔 사용

계정 및 홈 리전의 모든 Storage Lens 그룹을 나열하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 스토리지 렌즈 그룹을 선택합니다.
3. Storage Lens 그룹 아래에 계정의 Storage Lens 그룹 목록이 표시됩니다.

AWS CLI 사용

다음 AWS CLI 예제는 계정의 모든 Storage Lens 그룹을 나열합니다. 이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3control list-storage-lens-groups --account-id 111122223333 \
--region us-east-1
```

Java용 AWS SDK 사용

다음 AWS SDK for Java 예제는 계정 *111122223333*의 Storage Lens 그룹을 나열합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.ListStorageLensGroupsRequest;
import software.amazon.awssdk.services.s3control.model.ListStorageLensGroupsResponse;

public class ListStorageLensGroups {
    public static void main(String[] args) {
        String accountId = "111122223333";

        try {
            ListStorageLensGroupsRequest listStorageLensGroupsRequest =
 ListStorageLensGroupsRequest.builder()
                    .accountId(accountId)
```

```
                .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                .region(Region.US_WEST_2)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();
            ListStorageLensGroupsResponse response =
 s3ControlClient.listStorageLensGroups(listStorageLensGroupsRequest);
            System.out.println(response);
        } 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();
        }
    }
}
```

Storage Lens 그룹 세부 정보 보기

다음 예제는 Amazon S3 Storage Lens 그룹 구성 세부 정보를 보는 방법을 보여줍니다. Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 및 AWS SDK for Java를 사용하여 이러한 세부 정보를 볼 수 있습니다.

S3 콘솔 사용

Storage Lens 그룹 구성 세부 정보를 보려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 스토리지 렌즈 그룹을 선택합니다.
3. 스토리지 렌즈 그룹에서 관심 있는 스토리지 렌즈 그룹 옆의 옵션 버튼을 선택합니다.
4. 세부 정보 보기를 선택합니다. 이제 Storage Lens 그룹의 세부 정보를 검토합니다.

AWS CLI 사용

다음 AWS CLI 예제는 Storage Lens 그룹의 구성 세부 정보를 반환합니다. 이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3control get-storage-lens-group --account-id 111122223333 \
--region us-east-1 --name marketing-department
```

Java용 AWS SDK 사용

다음 AWS SDK for Java 예제에서는 계정 *111122223333*에 *Marketing-Department*라는 이름의 Storage Lens 그룹의 구성 세부 정보를 반환합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupResponse;

public class GetStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            GetStorageLensGroupRequest getRequest =
 GetStorageLensGroupRequest.builder()
                    .name(storageLensGroupName)
                    .accountId(accountId).build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            GetStorageLensGroupResponse response =
 s3ControlClient.getStorageLensGroup(getRequest);
            System.out.println(response);
        } 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();
        }
    }
}
```

Storage Lens 그룹 삭제

다음 예제는 Amazon S3 콘솔, AWS Command Line Interface(AWS CLI) 및 AWS SDK for Java을 사용하여 Amazon S3 Storage Lens 그룹을 삭제하는 방법을 보여줍니다.

S3 콘솔 사용

Storage Lens 그룹을 삭제하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 스토리지 렌즈 그룹을 선택합니다.
3. Storage Lens 그룹에서 삭제할 Storage Lens 그룹 옆에 있는 옵션 버튼을 선택합니다.
4. 삭제를 선택합니다. Storage Lens 그룹 삭제 대화 상자가 표시됩니다.
5. Storage Lens 그룹을 영구 삭제하려면 삭제를 다시 선택합니다.

> **Note**
>
> Storage Lens 그룹을 삭제한 후에는 복원할 수 없습니다.

AWS CLI 사용

다음 AWS CLI 예제에서는 *marketing-department* 라는 이름의 Storage Lens 그룹을 삭제합니다. 이 예 명령을 사용하려면 *user input placeholders*를 실제 정보로 대체하세요.

```
aws s3control delete-storage-lens-group --account-id 111122223333 \
--region us-east-1 --name marketing-department
```

Java용 AWS SDK 사용

다음 AWS SDK for Java 예제에서는 계정 *111122223333*에 *Marketing-Department*라는 이름의 Storage Lens 그룹을 삭제합니다. 이 예제를 사용하려면 *user input placeholders*를 사용자의 정보로 대체합니다.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.DeleteStorageLensGroupRequest;

public class DeleteStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            DeleteStorageLensGroupRequest deleteStorageLensGroupRequest =
 DeleteStorageLensGroupRequest.builder()
                    .name(storageLensGroupName)
                    .accountId(accountId).build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.deleteStorageLensGroup(deleteStorageLensGroupRequest);
        } 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();
        }
    }
}
```

S3 Inventory를 사용한 데이터 카탈로그화 및 분석

Amazon S3 인벤토리를 사용하여 스토리지를 관리할 수 있습니다. 예를 들어 이 인벤토리를 사용하면 비즈니스, 규정 준수 및 규제 요건에 대한 객체의 복제 및 암호화 상태를 감사하고 보고할 수 있습니다. 또한 Amazon S3 동기식 `List` API 작업에 대한 예약된 대안을 제공하는 Amazon S3 인벤토리를 사용하여 비즈니스 워크플로와 빅데이터 작업을 간소화하고 속도를 높일 수 있습니다. Amazon S3 인벤토

리는 객체를 감사하기 위해 `List` API 작업을 사용하지 않으며 버킷의 요청 속도에 영향을 주지 않습니다.

Amazon S3 인벤토리는 S3 버킷 또는 공유 접두사가 있는 객체(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 파일 출력을 가능하게 합니다. 주간 인벤토리를 설정하면 초기 보고서가 생성된 후 매주 일요일(UTC 시간대)에 보고서가 생성됩니다. Amazon S3 인벤토리 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하세요.

하나의 버킷에 대해 복수의 인벤토리 목록을 구성할 수 있습니다. 인벤토리 목록을 구성할 때 다음 항목을 지정할 수 있습니다.

- 인벤토리에 포함할 객체 메타데이터
- 모든 객체 버전을 나열할지 아니면 현재 버전만 나열할지 여부
- 인벤토리 목록 파일 출력을 저장할 위치
- 인벤토리를 일일 또는 주간 단위로 생성할지 여부
- 인벤토리 목록 파일을 암호화할지 여부

Amazon Athena, Amazon Redshift Spectrum 및 기타 도구(예: Presto, Apache Hive, Apache Spark)를 사용하여 표준 SQL 쿼리로 Amazon S3 인벤토리를 쿼리할 수 있습니다. Athena를 사용하여 데이터를 쿼리하는 방법에 대한 자세한 내용은 the section called “Athena로 인벤토리 쿼리” 단원을 참조하세요.

> **Note**
>
> Amazon S3에서 첫 번째 인벤토리 보고서를 전달하는 데 최대 48시간이 걸릴 수 있습니다.

원본 및 대상 버킷

인벤토리가 객체를 열거하는 버킷을 소스 버킷이라고 합니다. 인벤토리 목록 파일이 저장되는 버킷을 대상 버킷이라고 합니다.

소스 버킷

인벤토리는 원본 버킷에 저장되어 있는 객체를 열거합니다. 전체 버킷에 대한 인벤토리 목록을 가져오거나 객체 키 이름 접두사를 기준으로 목록을 필터링할 수 있습니다.

소스 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 접두사를 지정할 수 있습니다.

대상 버킷:

- 인벤토리 파일 목록을 포함합니다.
- 대상 버킷에 저장되어 있는 모든 인벤토리 목록 파일이 열거된 매니페스트 파일을 포함합니다. 자세한 내용은 인벤토리 매니페스트 섹션을 참조하세요.
- Amazon S3가 버킷의 소유권을 검증하고 파일을 버킷에 쓸 수 있도록 허용하는 버킷 정책이 반드시 있어야 합니다.
- 소스 버킷과 동일한 AWS 리전에 있어야 합니다.
- 원본 버킷과 같을 수 있습니다.
- 소스 버킷을 소유한 계정과 다른 AWS 계정에서 소유할 수 있습니다.

Amazon S3 인벤토리 목록

인벤토리 목록 파일에는 원본 버킷에 있는 객체의 목록과 각 객체의 메타데이터가 포함됩니다. 인벤토리 목록 파일은 다음 형식 중 하나로 대상 버킷에 저장됩니다.

- GZIP으로 압축된 CSV 파일
- ZLIB로 압축된 Apache 최적화 행 열 형식(ORC) 파일
- Snappy로 압축된 Apache Parquet 파일

> **Note**
>
> Amazon S3 인벤토리 보고서의 객체는 어떤 순서로 정렬되는지 보장되지 않습니다.

인벤토리 목록 파일에는 소스 버킷에 있는 객체의 목록과 나열된 각 객체의 메타데이터가 포함됩니다.

- Bucket name – 인벤토리 대상 버킷의 이름.

- 키 이름 – 버킷에 있는 객체를 고유하게 식별하는 객체 키 이름(또는 키). CSV 파일 형식을 사용하고 있는 경우 키 이름이 URL로 암호화되므로 사용하려면 디코딩해야 합니다.
- 버전 ID – 객체 버전 ID. 버전 관리를 사용하는 버킷의 경우 Amazon S3는 버킷에 추가된 객체에 버전 번호를 지정합니다. 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하세요. (목록이 최신 버전의 객체에 대해서만 구성된 경우 이 필드는 포함되지 않습니다.)
- IsLatest – 객체가 최신 버전인 경우 `True`로 설정됩니다. (목록이 최신 버전의 객체에 대해서만 구성된 경우 이 필드는 포함되지 않습니다.)
- 삭제 마커 – 객체가 삭제 마커인 경우 `True`로 설정됩니다. 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하세요. (모든 버전의 객체를 포함하도록 보고서를 구성한 경우 이 필드는 보고서에 자동으로 추가됩니다.)
- 크기 - 객체 크기(바이트)를 나타내며, 불완전한 멀티파트 업로드, 객체 메타데이터 및 삭제 마커의 크기는 포함되지 않습니다.
- 마지막 수정 날짜(Last modified date) - 객체 생성일 또는 최종 수정일 중 최근 날짜
- ETag - 엔터티 태그(ETag)는 객체의 해시입니다. ETag는 객체의 콘텐츠에 대한 변경 사항만 반영하고 메타데이터에 대한 변경을 반영하지 않습니다. ETag는 객체 데이터의 MD5 다이제스트일 수 있습니다. 다이제스트인지 여부는 객체 생성 방식 및 암호화 방식에 좌우됩니다. 자세한 내용은 Amazon Simple Storage Service API 참조에서 Object를 참조하십시오.
- 스토리지 클래스 - 객체 저장에 사용되는 스토리지 클래스입니다. `STANDARD`, `REDUCED_REDUNDANCY`, `STANDARD_IA`, `ONEZONE_IA`, `INTELLIGENT_TIERING`, `GLACIER`, `DEEP_ARCHIVE`, `OUTPOSTS`, `GLACIER_IR`, `SNOW`로 설정됩니다. 자세한 내용은 Amazon S3 스토리지 클래스 이해 및 관리 섹션을 참조하세요.

> **Note**
>
> S3 인벤토리는 S3 Express One Zone을 지원하지 않습니다.

- Multipart upload flag – 객체가 멀티파트 업로드로 업로드된 경우 `True`로 설정됩니다. 자세한 내용은 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 섹션을 참조하세요.
- 복제 상태 — `PENDING`, `COMPLETED`, `FAILED` 또는 `REPLICA`로 설정됩니다. 자세한 내용은 복제 상태 정보 가져오기 섹션을 참조하세요.
- 암호화 상태 - Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3), AWS Key Management Service(AWS KMS) 키를 사용한 서버 측 암호화(SSE-KMS), AWS KMS KMS 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS), 고객 제공형 키를 사용한 서버 측 암호화(SSE-C) 등 사용되는 암호화 키의 종류에 따른 서버 측 암호화 상태입니다. `SSE-S3`, `SSE-KMS`, `DSSE-KMS`, `SSE-C`, `NOT-`

SSE로 설정됩니다. NOT-SSE 상태는 객체가 서버 측 암호화를 사용하여 암호화되지 않은 것을 의미합니다. 자세한 내용은 암호화로 데이터 보호 섹션을 참조하세요.

- S3 객체 잠금 보관 종료일 – 잠긴 객체를 삭제할 수 없는 기한입니다. 자세한 내용은 Object Lock으로 객체 잠금 섹션을 참조하세요.
- S3 객체 잠금 보존 모드 – 잠긴 객체에 대해 Governance 또는 Compliance로 설정됩니다. 자세한 내용은 Object Lock으로 객체 잠금 섹션을 참조하세요.
- S3 객체 잠금 법적 보존 상태 – 법적 보존이 객체에 적용된 경우 , 그렇지 않으면 On으로 설정됩니다. 그렇지 않으면 Off로 설정됩니다. 자세한 내용은 Object Lock으로 객체 잠금 섹션을 참조하세요.
- S3 Intelligent-Tiering 액세스 티어 – 객체가 S3 Intelligent-Tiering 스토리지 클래스에 저장된 경우 객체의 액세스 계층(Frequent Access 또는 Infrequent Access)입니다. FREQUENT, INFREQUENT, ARCHIVE_INSTANT_ACCESS, ARCHIVE, DEEP_ARCHIVE로 설정됩니다. 자세한 내용은 변경되는 또는 알 수 없는 액세스 패턴으로 데이터를 자동으로 최적화하는 스토리지 클래스 섹션을 참조하세요.
- S3 버킷 키 상태 — ENABLED 또는 DISABLED로 설정됩니다. 객체가 SSE-KMS에 S3 버킷 키를 사용하는지 여부를 나타냅니다. 자세한 내용은 Amazon S3 버킷 키 사용 섹션을 참조하세요.
- 체크섬 알고리즘 – 객체의 체크섬을 생성하는 데 사용된 알고리즘을 나타냅니다. 자세한 내용은 지원되는 체크섬 알고리즘 사용 섹션을 참조하세요.
- 객체 액세스 제어 목록 - 이 객체에 대한 액세스 권한이 부여된 AWS 계정 또는 그룹과 부여된 액세스 유형을 정의하는 각 객체에 대한 액세스 제어 목록(ACL)입니다. 객체 ACL 필드는 JSON 형식으로 정의됩니다. S3 인벤토리 보고서에는 소스 버킷의 객체와 연결된 ACL이 포함되며, 이는 버킷에 대해 ACL이 비활성화된 경우에도 마찬가지입니다. 자세한 내용은 객체 ACL 필드 작업 및 ACL(액세스 제어 목록) 개요(을)를 참조하세요.

> **Note**
>
> 객체 ACL 필드는 JSON 형식으로 정의됩니다. 인벤토리 보고서에는 객체 ACL 필드의 값이 base64로 인코딩된 문자열로 표시됩니다.
> 예를 들어 다음과 같은 JSON 형식의 객체 ACL 필드가 있다고 가정해 보겠습니다.

```
{
        "version": "2022-11-10",
        "status": "AVAILABLE",
        "grants": [{
            "canonicalId": "example-canonical-user-ID",
            "type": "CanonicalUser",
            "permission": "READ"
        }]
```

```
}
```

객체 ACL 필드는 다음과 같이 base64로 인코딩된 문자열로 인코딩되어 표시됩니다.

```
eyJ2ZXJzaW9uIjoiMjAyMi0xMS0xMCIsInN0YXR1cyI6IkFWQUlMQUJMRSIsImdyYW50cyI6W3siY2Fub25pY2Fs
```

객체 ACL 필드의 디코딩된 값을 JSON으로 가져오려면 Amazon Athena에서 이 필드를 쿼리하면 됩니다. 쿼리 예제를 보려면 Amazon Athena로 Amazon S3 인벤토리 쿼리 단원을 참조하세요.

- 객체 소유자 - 객체 소유자의 표준 사용자 ID입니다. 자세한 내용은 AWS 계정 관리 참조 안내서의 내 AWS 계정의 표준 사용자 ID 찾기를 참조하세요.

> **Note**
>
> 수명 주기 구성에 따라 객체가 수명 주기의 끝에 도달한 경우, Amazon S3에서는 제거를 위해 객체를 대기열에 넣고 비동기 방식으로 제거합니다. 따라서 만료 날짜와 Amazon S3에서 객체를 제거하는 날짜 사이에 지연이 있을 수 있습니다. 인벤토리 보고서에는 만료되었지만 아직 제거되지 않은 객체가 포함됩니다. S3 수명 주기에서 만료 작업에 대한 자세한 내용은 객체 만료 섹션을 참조하십시오.

다음은 4개의 레코드로 구성된 추가 메타데이터 필드가 포함된 예제 인벤토리 보고서입니다.

```
amzn-s3-demo-bucket1    example-object-1    EXAMPLEDC8l.XJCENlF7LePaNIIvs001    TRUE
     1500    2024-08-15T15:28:26.0004    EXAMPLE21e1518b92f3d92773570f600    STANDARD
  FALSE    COMPLETED    SSE-KMS    2025-01-25T15:28:26.000Z    COMPLIANCE    Off
 ENABLED
 eyJ2ZXJzaW9uIjoiMjAyMi0xMS0xMCIsInN0YXR1cyI6IkFWQUlMQUJMRSIsImdyYW50cyI6W3sicGVybWlzc2lvbiI6Ik
    EXAMPLE766e8f6b115d93d41df2eac420aa4a465d177c1398bc6a088c76b7000
amzn-s3-demo-bucket1    example-object-2    EXAMPLEDC8l.XJCENlF7LePaNIIvs002
 TRUE        200    2024-08-21T15:28:26.000Z    EXAMPLE21e1518b92f3d92773570f601
  INTELLIGENT_TIERING    FALSE    COMPLETED    SSE-KMS    2025-01-25T15:28:26.000Z
    COMPLIANCE    Off    INFREQUENT    ENABLED    SHA-256
 eyJ2ZXJzaW9uIjoiMjAyMi0xMS0xMCIsInN0YXR1cyI6IkFWQUlMQUJMRSIsImdyYW50cyI6W3sicGVybWlzc2lvbiI6Ik
    EXAMPLE766e8f6b115d93d41df2eac420aa4a465d177c1398bc6a088c76b7001
amzn-s3-demo-bucket1    example-object-3    EXAMPLEDC8l.XJCENlF7LePaNIIvs003    TRUE
      12500    2023-01-15T15:28:30.000Z    EXAMPLE21e1518b92f3d92773570f602    STANDARD
    FALSE    REPLICA    SSE-KMS    2025-01-25T15:28:26.000Z    GOVERNANCE    On
```

```
 ENABLED
 eyJ2ZXJzaW9uIjoiMjAyMi0xMS0xMCIsInN0YXR1cyI6IkFWQUlMQUJMRSIsImdyYW50cyI6W3sicGVybWlzc2lvbiI6Ik
    EXAMPLE766e8f6b115d93d41df2eac420aa4a465d177c1398bc6a088c76b7002
amzn-s3-demo-bucket1    example-object-4    EXAMPLEDC8l.XJCENlF7LePaNIIvs004    TRUE
      100    2021-02-15T15:28:27.000Z    EXAMPLE21e1518b92f3d92773570f603    STANDARD
  FALSE    COMPLETED    SSE-KMS    2025-01-25T15:28:26.000Z    COMPLIANCE    Off
 ENABLED
 eyJ2ZXJzaW9uIjoiMjAyMi0xMS0xMCIsInN0YXR1cyI6IkFWQUlMQUJMRSIsImdyYW50cyI6W3sicGVybWlzc2lvbiI6Ik
    EXAMPLE766e8f6b115d93d41df2eac420aa4a465d177c1398bc6a088c76b7003
```

오래된 인벤토리 목록을 삭제하는 수명 주기 정책을 생성할 것을 권장합니다. 자세한 내용은 객체 수명 주기 관리 섹션을 참조하세요.

`s3:PutInventoryConfiguration` 권한을 통해 사용자는 인벤토리 목록을 구성할 때 각 객체에 대해 앞서 나열된 모든 메타데이터 필드를 선택하고 인벤토리를 저장할 대상 버킷을 지정할 수 있습니다. 대상 버킷의 객체에 대한 읽기 권한이 있는 사용자는 인벤토리 목록에서 사용할 수 있는 모든 객체 메타데이터 필드에 액세스할 수 있습니다. 인벤토리 보고서에 대한 액세스를 제한하려면 S3 인벤토리 및 S3 분석 권한 부여 섹션을 참조하세요.

인벤토리 일관성

모든 객체가 각 인벤토리 목록에 나타나지 않을 수도 있습니다. 인벤토리 목록은 새 객체 및 덮어쓰기 모두의 PUT 요청과 DELETE 요청에 대한 최종 일관성을 제공합니다. 버킷의 각 인벤토리 목록은 버킷 항목의 스냅샷입니다. 이러한 목록은 결국 일관성이 있습니다(즉, 최근에 추가되거나 삭제된 객체가 목록에 포함되지 않을 수 있습니다).

객체에 대한 작업을 수행하기 전에 객체 상태의 유효성을 검증하려면 객체의 메타데이터를 가져오도록 `HeadObject` REST API 요청을 수행하거나 Amazon S3 콘솔에서 객체의 속성을 확인하는 것이 좋습니다. AWS CLI 또는 AWS SDK를 사용하여 객체 메타데이터를 확인할 수도 있습니다. 자세한 내용은 Amazon Simple Storage Service API 참조의 `HeadObject`를 참조하세요.

Amazon S3 인벤토리 작업에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- Amazon S3 인벤토리 구성
- 인벤토리 목록 찾기
- 인벤토리 완료에 대한 Amazon S3 이벤트 알림 설정
- Amazon Athena로 Amazon S3 인벤토리 쿼리
- Amazon S3 인벤토리 보고서의 빈 버전 ID 문자열을 null 문자열로 변환

- 객체 ACL 필드 작업

Amazon S3 인벤토리 구성

Amazon S3 인벤토리는 사용자가 정의한 일정에 따라 객체 및 메타데이터의 플랫 파일 목록을 제공합니다. Amazon S3 동기식 `List` API 작업 대신 S3 인벤토리를 예약된 대안으로 사용할 수 있습니다. S3 인벤토리는 객체 및 해당 메타데이터를 나열하는 쉼표로 구분된 값(CSV), Apache Optimized Row Columnar(ORC) 또는 Apache Parquet (Parquet) 출력 파일을 제공합니다.

S3 버킷 또는 접두사를 공유하는 객체(이름이 동일한 문자열로 시작하는 객체)에 대해 일간 또는 주간 기준으로 인벤토리 목록을 생성하도록 S3 인벤토리를 구성할 수 있습니다. 자세한 내용은 S3 Inventory를 사용한 데이터 카탈로그화 및 분석 섹션을 참조하세요.

이 섹션에서는 인벤토리 원본 및 대상 버킷에 대한 세부 정보를 비롯하여 인벤토리를 구성하는 방법에 대해 설명합니다.

주제

- 개요
- 대상 버킷 정책 생성
- 암호화에 고객 관리형 키를 사용하도록 Amazon S3에 권한 부여
- S3 콘솔을 사용하여 인벤토리 구성
- REST API를 사용한 S3 인벤토리 작업

개요

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 Amazon S3 콘솔을 사용하는 것이지만, Amazon S3 REST API, AWS Command Line Interface(AWS CLI) 또는 AWS SDK를 사용할 수도 있습니다. 콘솔에서는 대상 버킷에 버킷 정책을 추가하는 절차의 첫 번째 단계를 자동으로 수행합니다.

S3 버킷의 Amazon S3 인벤토리 설정

1. 대상 버킷의 버킷 정책을 추가합니다.

객체를 정해진 위치에 있는 버킷에 쓸 수 있는 권한을 Amazon S3에 부여하는 버킷 정책을 대상 버킷에 생성해야 합니다. 정책에 대한 예는 S3 인벤토리 및 S3 분석 권한 부여을 참조하세요.

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 모든 객체 버전을 목록에 포함시킬지 아니면 최신 버전만을 목록에 포함시킬지 구성하고 포함시킬 객체 메타데이터를 구성할 수 있습니다.

   S3 인벤토리 보고서 구성의 일부 객체 메타데이터 필드는 선택 사항입니다. 즉, 기본적으로 사용할 수 있지만 사용자에게 `s3:PutInventoryConfiguration` 권한을 부여하면 제한될 수 있습니다. `s3:InventoryAccessibleOptionalFields` 조건 키를 사용하여 사용자가 보고서에 이러한 선택적 메타데이터 필드를 포함할 수 있는지를 제어할 수 있습니다.

   S3 인벤토리에서 사용할 수 있는 선택적 메타데이터 필드에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 OptionalFields 섹션을 참조하세요. 인벤토리 구성의 특정 선택적 메타데이터 필드 대상 액세스 제한에 대한 자세한 정보는 S3 인벤토리 보고서 구성 생성 제어 섹션을 참조하세요.

   Amazon S3 관리형 키(SSE-S3) 또는 AWS Key Management Service(AWS KMS) 고객 관리 키(SSE-KMS)를 사용한 서버 측 암호화를 사용하여 인벤토리 목록 파일을 암호화하도록 지정할 수 있습니다.

   > **Note**
   >
   > AWS 관리형 키(`aws/s3`)는 S3 인벤토리를 사용하는 SSE-KMS 암호화에 지원되지 않습니다.

   SSE-S3 및 SSE-KMS에 대한 자세한 내용은 서버 측 암호화를 사용하여 데이터 보호 섹션을 참조하세요. SSE-KMS 암호화를 사용하려면 3단계를 참조하세요.

   - 콘솔을 사용하여 인벤토리 목록을 구성하는 방법에 대한 자세한 내용은 S3 콘솔을 사용하여 인벤토리 구성 섹션을 참조하세요.
   - Amazon S3 API를 사용하여 인벤토리 목록을 구성하려면 PutBucketInventoryConfiguration REST API 작업을 사용하거나 AWS CLI 또는 AWS SDK의 동일한 기능을 사용하면 됩니다.

3. SSE-KMS를 사용하여 인벤토리 목록 파일을 암호화하려면 AWS KMS key 사용 권한을 Amazon S3에 부여합니다.

Amazon S3 콘솔, Amazon S3 REST API, AWS CLI 또는 AWS SDK를 사용하여 인벤토리 목록 파일에 대한 암호화를 구성할 수 있습니다. 어느 방법을 선택하든 인벤토리 파일을 암호화하려면 Amazon S3에 고객 관리형 키 사용 권한을 부여해야 합니다. 인벤토리 파일을 암호화하는 데 사용할 고객 관리형 키에 대한 키 정책을 수정하여 Amazon S3에 권한을 부여합니다. 자세한 내용은 암호화에 고객 관리형 키를 사용하도록 Amazon S3에 권한 부여 섹션을 참조하세요.

인벤토리 목록 파일을 저장하는 대상 버킷은 소스 버킷을 소유한 계정과 다른 AWS 계정 계정에서 소유할 수 있습니다. Amazon S3 인벤토리의 계정 간 작업에 SSE-KMS 암호화를 사용하는 경우 S3 인벤토리를 구성할 때 정규화된 KMS 키 ARN을 사용하는 것이 좋습니다. 자세한 내용은 Amazon Simple Storage Service API 참조의 크로스 계정 작업에 SSE-KMS 암호화 사용 및 ServerSideEncryptionByDefault을 참조하세요.

대상 버킷 정책 생성

Amazon S3 콘솔을 통해 인벤토리 구성을 생성하면 Amazon S3가 대상 버킷에 Amazon S3 쓰기 권한을 부여하는 버킷 정책을 자동으로 생성합니다. 그러나 AWS CLI, AWS SDK 또는 Amazon S3 REST API를 통해 인벤토리 구성을 생성하는 경우에는 대상 버킷에 버킷 정책을 수동으로 추가해야 합니다. S3 인벤토리 대상 버킷 정책은 Amazon S3가 인벤토리 보고서에 대한 데이터를 해당 버킷에 쓰도록 허용합니다.

다음은 버킷 정책의 예제입니다.

```
{
      "Version": "2012-10-17",
      "Statement": [
        {
            "Sid": "InventoryExamplePolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-DESTINATION-BUCKET/*"
            ],
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET"
                },
                "StringEquals": {
```

```
                    "aws:SourceAccount": "source-account-id",
                    "s3:x-amz-acl": "bucket-owner-full-control"
                }
            }
        }
    ]
}
```

자세한 내용은 S3 인벤토리 및 S3 분석 권한 부여 섹션을 참조하세요.

버킷 정책을 생성하는 동안 오류가 발생하는 경우, 해결 지침이 제시됩니다. 예를 들어 다른 AWS 계정의 대상 버킷을 선택하는 바람에 해당 버킷 정책에 대한 읽기 및 쓰기 권한이 없는 경우 오류 메시지가 나타납니다.

이 경우 대상 버킷 소유자는 버킷 정책을 대상 버킷에 추가해야 합니다. Amazon S3는 대상 버킷에 대한 쓰기 권한이 없기 때문에 이 정책을 대상 버킷에 추가하지 않으면 인벤토리 보고서를 받을 수 없게 됩니다. 소스 버킷이 현재 사용자가 아닌 다른 계정의 소유물인 경우, 정책에서 소스 버킷 소유자의 올바른 계정 ID로 바꿔야 합니다.

> **Note**
>
> 인벤토리 보고서가 이 버킷으로 전송되지 않도록 하는 Deny 문이 대상 버킷 정책에 추가되지 않았는지 확인하세요. 자세한 내용은 Why can't I generate an Amazon S3 Inventory Report?를 참조하세요.

암호화에 고객 관리형 키를 사용하도록 Amazon S3에 권한 부여

서버 측 암호화에 AWS Key Management Service(AWS KMS) 고객 관리형 키를 사용하도록 Amazon S3에 권한을 부여하려면 키 정책을 사용해야 합니다. 고객 관리형 키를 사용할 수 있도록 키 정책을 업데이트하려면 아래 절차를 따르세요.

고객 관리형 키를 사용하여 Amazon S3에 암호화 권한을 부여하는 방법

1. 고객 관리형 키를 소유한 AWS 계정을 사용하여 AWS Management Console에 로그인합니다.
2. AWS KMS 콘솔(https://console.aws.amazon.com/kms)을 엽니다.
3. AWS 리전을 변경하려면 페이지의 오른쪽 상단 모서리에 있는 리전 선택기를 사용합니다.
4. 탐색 창에서 고객 관리형 키를 선택합니다.
5. 고객 관리형 키에서 인벤토리 파일을 암호화하는 데 사용할 고객 관리형 키를 선택합니다.

6. 키 정책(Key policy) 섹션에서 정책 보기로 전환(Switch to policy view)을 선택합니다.
7. 키 정책을 업데이트하려면 편집(Edit)을 선택합니다.
8. 키 정책 편집 페이지에서 기존 키 정책에 다음 줄을 추가합니다. *source-account-id* 및 *amzn-s3-demo-source-bucket*에 사용 사례에 적합한 값을 입력합니다.

```
{
    "Sid": "Allow Amazon S3 use of the customer managed key",
    "Effect": "Allow",
    "Principal": {
        "Service": "s3.amazonaws.com"
    },
    "Action": [
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition":{
      "StringEquals":{
         "aws:SourceAccount":"source-account-id"
     },
      "ArnLike":{
        "aws:SourceARN": "arn:aws:s3:::amzn-s3-demo-source-bucket"
     }
   }
}
```

9. 변경 사항 저장을 선택합니다.

고객 관리형 키 생성 및 키 정책 사용에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 다음 링크를 참조하세요.

- 키 관리
- AWS KMS의 키 정책

> **Note**
>
> 인벤토리 보고서가 이 버킷으로 전송되지 않도록 하는 Deny 문이 대상 버킷 정책에 추가되지 않았는지 확인하세요. 자세한 내용은 Why can't I generate an Amazon S3 Inventory Report?를 참조하세요.

S3 콘솔을 사용하여 인벤토리 구성

다음 지침에 따라 S3 콘솔을 사용하여 인벤토리를 구성합니다.

> **Note**
>
> Amazon S3에서 첫 번째 인벤토리 보고서를 전달하는 데 최대 48시간이 걸릴 수 있습니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 Amazon S3 Inventory를 구성할 버킷의 이름을 선택합니다.
4. [Management] 탭을 선택한 후
5. 인벤토리 구성(Inventory configurations)에서 인벤토리 구성 생성(Create inventory configuration)을 선택합니다.
6. 인벤토리 구성 이름에 이름을 입력합니다.
7. 인벤토리 범위에서는 다음을 수행합니다.
   - 선택적 접두사를 입력합니다.
   - 현재 버전만 또는 모든 버전 중에서 포함할 객체 버전을 선택합니다.
8. [보고서 세부 정보(Report details)]에서 보고서를 저장할 AWS 계정의 위치를 [이 계정(This account)] 또는 [다른 계정(A different account)]으로 선택합니다.
9. 대상에서 인벤토리 보고서를 저장할 버킷을 선택합니다.

   대상 버킷은 인벤토리를 설정할 버킷과 같은 AWS 리전에 있어야 합니다. 대상 버킷은 다른 AWS 계정에 있을 수 있습니다. 대상 버킷을 지정할 때 선택적 접두사를 포함하여 인벤토리 보고서를 함께 그룹화할 수도 있습니다.

   대상 버킷 필드 아래에 대상 버킷 정책에 추가되어 Amazon S3가 해당 버킷에 데이터를 배치할 수 있도록 허용하는 대상 버킷 권한 문이 표시됩니다. 자세한 내용은 대상 버킷 정책 생성 섹션을 참조하세요.
10. 빈도에서 보고서 생성 빈도를 매일 또는 매주를 선택합니다.
11. 출력 형식에서 다음 보고서 형식 중 하나를 선택합니다.

- CSV - 이 인벤토리 보고서를 S3 배치 작업과 함께 사용하거나 Microsoft Excel과 같은 다른 도구에서 이 보고서를 분석하려는 경우 CSV를 선택합니다.
- Apache ORC
- Apache Parquet

12. 상태에서 활성화 또는 비활성화를 선택합니다.
13. 서버 측 암호화를 구성하려면 인벤토리 보고서 암호화에서 다음 단계를 따르세요.

    a. 서버 측 암호화에서 암호화를 지정하지 마십시오 또는 암호화 키 지정을 선택하여 데이터를 암호화합니다.

    - Amazon S3에 객체를 저장할 때 객체의 기본 서버 측 암호화에 대한 버킷 설정을 유지하려면 암호화 키를 지정하지 마십시오를 선택합니다. 버킷 대상에서 S3 버킷 키가 사용되는 한 복사 작업은 대상 버킷에 S3 버킷 키를 적용합니다.

      > **Note**
      >
      > 지정된 대상의 버킷 정책에서 Amazon S3에 저장하기 전에 객체를 암호화하도록 요구하는 경우 암호화 키 지정을 선택해야 합니다. 지정하지 않으면 대상에 대한 객체 복사가 실패합니다.

    - Amazon S3에 저장하기 전에 객체를 암호화하려면 암호화 키 지정을 선택합니다.

    b. 암호화 키 지정을 선택한 경우, 암호화 유형에서 Amazon S3 관리형 키(SSE-S3) 또는 AWS Key Management Service 키(SSE-KMS) 중 하나를 선택해야 합니다.

    SSE-S3는 가장 강력한 블록 암호 중 하나인 256비트 Advanced Encryption Standard(AES-256)을 사용하여 각 객체를 암호화합니다. SSE-KMS는 키에 대한 더 많은 제어 기능을 제공합니다. SSE-S3에 대한 자세한 내용은 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) 사용 섹션을 참조하세요. SSE-KMS에 대한 자세한 내용은 AWS KMS 키를 사용한 서버 측 암호화(SSE-KMS) 사용 단원을 참조하십시오.

    > **Note**
    >
    > SSE-KMS를 사용하여 인벤토리 목록 파일을 암호화하려면 고객 관리형 키를 사용하는 권한을 Amazon S3에 부여해야 합니다. 자세한 내용은 Amazon S3에 KMS 키를 사용하여 암호화할 수 있는 권한 부여를 참조하세요.

c. AWS Key Management Service 키(SSE-KMS)를 선택한 경우, AWS KMS key에서 다음 옵션 중 하나를 통해 AWS KMS 키를 지정할 수 있습니다.

> **Note**
>
> 인벤토리 목록 파일을 저장하는 대상 버킷을 다른 AWS 계정에서 소유한 경우 정규화된 KMS 키 ARN을 사용하여 KMS 키를 지정하세요.

- 사용 가능한 KMS 키 목록에서 선택하려면 AWS KMS 키 중에서 선택을 선택한 다음, 사용 가능한 키 중 대칭 암호화 KMS 키를 선택하세요. KMS 키가 버킷과 같은 리전에 있어야 합니다.

> **Note**
>
> AWS 관리형 키(aws/s3)와 고객 관리형 키가 모두 목록에 표시됩니다. 그러나 AWS 관리형 키(aws/s3)는 S3 인벤토리를 사용하는 SSE-KMS 암호화에 지원되지 않습니다.

- KMS 키 ARN을 입력하려면 AWS KMS 키 ARN 입력을 선택하고 나타나는 필드에 KMS 키 ARN을 입력합니다.
- AWS KMS 콘솔에서 고객 관리형 키를 생성하려면 KMS 키 생성을 선택합니다.

14. 추가 메타데이터 필드에 대해 다음 중 하나 이상을 선택하여 인벤토리 보고서에 추가합니다.

- 크기 - 객체 크기(바이트)를 나타내며, 불완전한 멀티파트 업로드, 객체 메타데이터 및 삭제 마커의 크기는 포함되지 않습니다.
- 마지막 수정 날짜(Last modified date) - 객체 생성일 또는 최종 수정일 중 최근 날짜
- 멀티파트 업로드(Multipart upload) - 객체가 멀티파트 업로드로 업로드되도록 지정합니다. 자세한 내용은 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 섹션을 참조하세요.
- 복제 상태(Replication status) - 객체의 복제 상태입니다. 자세한 내용은 복제 상태 정보 가져오기 섹션을 참조하세요.
- 암호화 상태 - 객체를 암호화하는 데 사용된 서버 측 암호화 유형입니다. 자세한 내용은 서버 측 암호화를 사용하여 데이터 보호 섹션을 참조하세요.
- 버킷 키 상태 — AWS KMS에 의해 생성된 버킷 레벨 키가 객체에 적용되는지 여부를 나타냅니다. 자세한 내용은 Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감 섹션을 참조하세요.

- 객체 액세스 제어 목록 - 이 객체에 대한 액세스 권한이 부여된 AWS 계정 또는 그룹과 부여된 액세스 유형을 정의하는 각 객체에 대한 액세스 제어 목록(ACL)입니다. 이 필드에 대한 자세한 내용은 객체 ACL 필드 작업 단원을 참조하세요. ACL에 대한 자세한 내용은 ACL(액세스 제어 목록) 개요 단원을 참조하십시오.
- 객체 소유자 - 객체 소유자입니다.
- 스토리지 클래스 - 객체 저장에 사용되는 스토리지 클래스입니다.
- Intelligent-Tiering 액세스 티어 – 객체가 S3 Intelligent-Tiering 스토리지 클래스에 저장된 경우 객체의 액세스 계층(Frequent Access 또는 Infrequent Access)을 나타냅니다. 자세한 내용은 변경되는 또는 알 수 없는 액세스 패턴으로 데이터를 자동으로 최적화하는 스토리지 클래스 섹션을 참조하세요.
- ETag - 엔터티 태그(ETag)는 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. 자세한 내용은 Amazon Simple Storage Service API 참조에서 Object를 참조하십시오.
- 체크섬 알고리즘 – 객체의 체크섬을 생성하는 데 사용된 알고리즘을 나타냅니다. 자세한 내용은 지원되는 체크섬 알고리즘 사용 섹션을 참조하세요.
- 모든 객체 잠금 구성 - 다음 설정을 포함한 객체의 객체 잠금 상태입니다.
  - 객체 잠금: 보관 모드 - 객체에 적용된 보호 수준(거버넌스 또는 규정 준수)입니다.
  - 객체 잠금: 다음 날짜까지 보존 – 잠긴 객체를 삭제할 수 없는 기한입니다.
  - 객체 잠금: 법적 보존 상태 - 잠긴 객체의 법적 보존 상태입니다.

  S3 객체 잠금에 대한 자세한 내용은 S3 객체 잠금 작동 방식 단원을 참조하세요.

인벤토리 보고서의 내용에 대한 자세한 내용은 Amazon S3 인벤토리 목록 단원을 참조하세요.

인벤토리 구성의 특정 선택적 메타데이터 필드 대상 액세스 제한에 대한 자세한 정보는 S3 인벤토리 보고서 구성 생성 제어 섹션을 참조하세요.

15. 생성(Create)을 선택합니다.

REST API를 사용한 S3 인벤토리 작업

다음은 Amazon S3 인벤토리 작업에 사용할 수 있는 REST 작업입니다.

- DeleteBucketInventoryConfiguration
- GetBucketInventoryConfiguration

- ListBucketInventoryConfigurations
- PutBucketInventoryConfiguration

인벤토리 목록 찾기

매니페스트 파일은 인벤토리 목록이 게시될 때 대상 버킷의 다음 위치에 게시됩니다.

```
destination-prefix/amzn-s3-demo-source-bucket/config-ID/YYYY-MM-DDTHH-MMZ/manifest.json
 destination-prefix/amzn-s3-demo-source-bucket/config-ID/YYYY-MM-DDTHH-MMZ/
manifest.checksum
 destination-prefix/amzn-s3-demo-source-bucket/config-ID/hive/dt=YYYY-MM-DD-HH-MM/
symlink.txt
```

- *destination-prefix*는 인벤토리 구성에서 선택적으로 지정되는 객체 키 이름 접두사이며, 이 접두사를 사용하여 대상 버킷 내의 일반적인 위치에 있는 모든 인벤토리 목록 파일을 그룹화할 수 있습니다.
- *amzn-s3-demo-source-bucket*은 인벤토리 목록에 객체가 열거되는 버킷입니다. 여러 소스 버킷에서 복수의 인벤토리 보고서가 동일한 대상 버킷으로 전송되는 경우 충돌을 방지하기 위해 소스 버킷 이름이 추가됩니다.
- *config-ID*는 동일한 소스 버킷에서 동일한 대상 버킷으로 복수의 인벤토리 보고서가 전송되는 경우 이들의 충돌을 방지하기 위해 추가됩니다. *config-ID*는 인벤토리 보고서 구성에서 비롯되며 설정 중에 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.json은 매니페스트 파일입니다.
- manifest.checksum은 manifest.json 파일 콘텐츠의 MD5 해시입니다.
- symlink.txt는 Apache Hive 호환 매니페스트 파일입니다.

인벤토리 목록은 매일 또는 매주 대상 버킷의 다음 위치에 게시됩니다.

```
destination-prefix/amzn-s3-demo-source-bucket/config-ID/data/example-file-name.csv.gz
...
destination-prefix/amzn-s3-demo-source-bucket/config-ID/data/example-file-name-1.csv.gz
```

- *destination-prefix*는 인벤토리 구성에서 선택적으로 지정되는 객체 키 이름 접두사이며, 대상 버킷 내의 일반적인 위치에 있는 모든 인벤토리 목록 파일을 그룹화하는 데 사용할 수 있습니다.

- *amzn-s3-demo-source-bucket*은 인벤토리 목록에 객체가 열거되는 버킷입니다. 여러 소스 버킷에서 복수의 인벤토리 보고서가 동일한 대상 버킷으로 전송되는 경우 충돌을 방지하기 위해 소스 버킷 이름이 추가됩니다.
- *example-file-name*.csv.gz는 CSV 인벤토리 파일 중 하나입니다. ORC 인벤토리 이름은 파일 이름 확장자 .orc로 끝나고, Parquet 인벤토리 이름은 파일 이름 확장자 .parquet로 끝납니다.

인벤토리 매니페스트

매니페스트 파일 manifest.json 및 symlink.txt는 인벤토리 파일의 위치에 대해 설명합니다. 새 인벤토리 목록이 제공될 때마다 새 매니페스트 파일 세트가 함께 제공됩니다. 이러한 파일은 서로 덮어쓸 수도 있습니다. Amazon S3는 버전 관리를 사용하는 버킷에서 매니페스트 파일의 새 버전을 생성합니다.

manifest.json 파일에 포함된 각 매니페스트에는 인벤토리에 대한 기타 기본 정보와 메타데이터가 들어 있습니다. 이 정보에는 다음 사항이 포함됩니다.

- 소스 버킷 이름
- 대상 버킷 이름
- 인벤토리 버전
- 시작 시간과 인벤토리 보고서 생성 프로세스가 버킷 스캔을 시작하는 날짜로 구성된 epoch 날짜 형식의 생성 타임스탬프
- 인벤토리 파일의 형식 및 스키마
- 대상 버킷에 있는 인벤토리 파일의 목록

manifest.json 파일이 쓰여질 때마다 manifest.checksum 파일 콘텐츠의 MD5 해시인 manifest.json 파일이 수반됩니다.

Example **manifest.json** 파일의 인벤토리 매니페스트

다음 예제에서는 CSV, ORC, Parquet 형식 인벤토리에 대한 manifest.json 파일의 인벤토리 매니페스트를 보여 줍니다.

CSV

다음은 CSV 형식 인벤토리에 대해 manifest.json 파일에 있는 매니페스트의 예입니다.

```
{
    "sourceBucket": "amzn-s3-demo-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, IntelligentTieringAccessTier, BucketKeyStatus,
 ChecksumAlgorithm, ObjectAccessControlList, ObjectOwner",
    "files": [
        {
            "key": "Inventory/amzn-s3-demo-source-bucket/2016-11-06T21-32Z/
files/939c6d46-85a9-4ba8-87bd-9db705a579ce.csv.gz",
            "size": 2147483647,
            "MD5checksum": "f11166069f1990abeb9c97ace9cdfabc"
        }
    ]
}
```

ORC

다음은 ORC 형식 인벤토리에 대해 `manifest.json` 파일에 있는 매니페스트의 예입니다.

```
{
    "sourceBucket": "amzn-s3-demo-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:boole
    "files": [
        {
            "key": "inventory/amzn-s3-demo-source-bucket/data/
d794c570-95bb-4271-9128-26023c8b4900.orc",
            "size": 56291,
            "MD5checksum": "5925f4e78e1695c2d020b9f6eexample"
        }
    ]
}
```

PARQUET

다음은 Parquet 인벤토리에 대해 `manifest.json` 파일에 있는 매니페스트의 예입니다.

```
{
    "sourceBucket": "amzn-s3-demo-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); optional int64 object_lock_retain_until_date (TIMESTAMP_MILLIS); optional
 binary object_lock_mode (UTF8); optional binary object_lock_legal_hold_status
 (UTF8); optional binary intelligent_tiering_access_tier (UTF8); optional binary
 bucket_key_status (UTF8); optional binary checksum_algorithm (UTF8); optional
 binary object_access_control_list (UTF8); optional binary object_owner (UTF8);}",
    "files": [
        {
           "key": "inventory/amzn-s3-demo-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` 파일 읽기는 ORC 및 Parquet 형식 인벤토리 파일에 지원되지 않습니다.

인벤토리 완료에 대한 Amazon S3 이벤트 알림 설정

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 Amazon Simple Notification Service(Amazon SNS) 주제, Amazon Simple Queue Service(Amazon SQS) 대기열 또는 AWS Lambda 함수에 이벤트를 게시할 수 있습니다. 자세한 내용은 Amazon S3 이벤트 알림 섹션을 참조하세요.

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>
      <CloudFunction>arn:aws:lambda:us-west-2:222233334444:cloud-function-list-write</
CloudFunction>
      <Event>s3:ObjectCreated:*</Event>
  </QueueConfiguration>
  </NotificationConfiguration>
```

자세한 내용은 AWS Lambda 개발자 안내서의 Amazon S3에서 AWS Lambda 사용을 참조하세요.

Amazon Athena로 Amazon S3 인벤토리 쿼리

Athena를 사용할 수 있는 모든 리전에서 Amazon Athena를 사용하여 표준 SQL 쿼리로 Amazon S3 인벤토리 파일을 쿼리할 수 있습니다. AWS 리전 가용성을 확인하려면 AWS 리전 표를 참조하세요.

Athena는 Apache 최적화된 행 열 형식(ORC), Apache Parquet 또는 쉼표로 구분된 값(CSV) 형식의 Amazon S3 인벤토리 파일을 쿼리할 수 있습니다. Athena를 사용하여 인벤토리 파일을 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. ORC와 Parquet은 자체 설명이 가능한 유형 인식 열 형식의 파일 형식으로, Apache Hadoop용으로 설계되었습니다. 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 파일을 쿼리하려면

1. Athena 테이블을 만듭니다. 테이블 생성에 대한 자세한 내용은 Amazon Athena 사용 설명서의 Amazon Athena에서 테이블 생성을 참조하세요.
2. 쿼리하려는 인벤토리 보고서가 ORC 형식인지, Parquet 형식인지, CSV 형식인지에 따라 다음 샘플 쿼리 템플릿 중 하나를 사용하여 쿼리를 생성합니다.

   - Athena를 사용하여 ORC 형식의 인벤토리 보고서를 쿼리하는 경우 다음 샘플 쿼리를 템플릿으로 사용합니다.

     다음 쿼리 예제에는 ORC 형식 인벤토리 보고서의 모든 선택적 필드가 포함되어 있습니다.

     이 샘플 쿼리를 사용하려면 다음을 수행하세요.

     - *your_table_name*을 생성한 Athena 테이블의 이름으로 바꿉니다.
     - 쿼리가 인벤토리에 대해 선택한 필드와 일치하도록 인벤토리에 대해 선택하지 않은 선택적 필드를 제거합니다.
     - 다음 버킷 이름과 인벤토리 위치(구성 ID)를 구성에 맞게 적절하게 바꿉니다.

       s3://*amzn-s3-demo-bucket*/*config-ID*/hive/

     - *2022-01-01-00-00* 아래의 `projection.dt.range` 날짜를 Athena에서 데이터를 파티션하는 시간 범위의 첫날로 바꿉니다. 자세한 내용은 Athena에서 데이터 파티셔닝을 참조하세요.

```
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 bigint,
         object_lock_mode string,
         object_lock_legal_hold_status string,
         intelligent_tiering_access_tier string,
         bucket_key_status string,
         checksum_algorithm string,
         object_access_control_list string,
         object_owner 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://source-bucket/config-ID/hive/'
  TBLPROPERTIES (
    "projection.enabled" = "true",
    "projection.dt.type" = "date",
    "projection.dt.format" = "yyyy-MM-dd-HH-mm",
    "projection.dt.range" = "2022-01-01-00-00,NOW",
    "projection.dt.interval" = "1",
    "projection.dt.interval.unit" = "HOURS"
  );
```

- Athena를 사용하여 Parquet 형식의 인벤토리 보고서를 쿼리하는 경우 ORC 형식의 보고서에 대한 샘플 쿼리를 사용합니다. 단, `ROW FORMAT SERDE` 문에서 ORC SerDe 대신 다음 Parquet SerDe를 사용하세요.

```
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
```

- Athena를 사용하여 CSV 형식의 인벤토리 보고서를 쿼리하는 경우 다음 샘플 쿼리를 템플릿으로 사용합니다.

  다음 쿼리 예제에는 CSV 형식 인벤토리 보고서의 모든 선택적 필드가 포함되어 있습니다.

  이 샘플 쿼리를 사용하려면 다음을 수행하세요.

  - *your_table_name*을 생성한 Athena 테이블의 이름으로 바꿉니다.

- 쿼리가 인벤토리에 대해 선택한 필드와 일치하도록 인벤토리에 대해 선택하지 않은 선택적 필드를 제거합니다.
- 다음 버킷 이름과 인벤토리 위치(구성 ID)를 구성에 맞게 적절하게 바꿉니다.

  s3://*amzn-s3-demo-bucket*/*config-ID*/hive/

- *2022-01-01-00-00* 아래의 `projection.dt.range` 날짜를 Athena에서 데이터를 파티션하는 시간 범위의 첫날로 바꿉니다. 자세한 내용은 Athena에서 데이터 파티셔닝을 참조하세요.

```
CREATE EXTERNAL TABLE your_table_name(
         bucket string,
         key string,
         version_id string,
         is_latest boolean,
         is_delete_marker boolean,
         size string,
         last_modified_date string,
         e_tag string,
         storage_class string,
         is_multipart_uploaded boolean,
         replication_status string,
         encryption_status string,
         object_lock_retain_until_date string,
         object_lock_mode string,
         object_lock_legal_hold_status string,
         intelligent_tiering_access_tier string,
         bucket_key_status string,
         checksum_algorithm string,
         object_access_control_list string,
         object_owner string
) PARTITIONED BY (
        dt string
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
  STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
  OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'
  LOCATION 's3://source-bucket/config-ID/hive/'
  TBLPROPERTIES (
    "projection.enabled" = "true",
    "projection.dt.type" = "date",
    "projection.dt.format" = "yyyy-MM-dd-HH-mm",
    "projection.dt.range" = "2022-01-01-00-00,NOW",
```

```
    "projection.dt.interval" = "1",
    "projection.dt.interval.unit" = "HOURS"
  );
```

3. 이제 다음 예제와 같이 인벤토리에서 다양한 쿼리를 실행할 수 있습니다. 각 *user input placeholder*를 사용자의 정보로 바꿉니다.

```
# Get a list of the latest inventory report dates available.
SELECT DISTINCT dt FROM your_table_name ORDER BY 1 DESC limit 10;

# Get the 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 the encryption status for inventory 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;
```

인벤토리 보고서에 객체 액세스 제어 목록(ACL) 필드를 추가하도록 S3 인벤토리를 구성하면 보고서에 객체 ACL 필드의 값이 base64로 인코딩된 문자열로 표시됩니다. 객체 ACL 필드의 디코딩된 값을 JSON으로 가져오려면 Athena를 사용해 이 필드를 쿼리하면 됩니다. 다음 쿼리 예제를 참조하세요. 객체 ACL 필드에 대한 자세한 내용은 객체 ACL 필드 작업 단원을 참조하세요.

```
# Get the S3 keys that have Object ACL grants with public access.
WITH grants AS (
    SELECT key,
        CAST(
            json_extract(from_utf8(from_base64(object_access_control_list)),
 '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))
        ) AS grants_array
    FROM your_table_name
)
SELECT key,
       grants_array,
       grant
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE element_at(grant, 'uri') = 'http://acs.amazonaws.com/groups/global/AllUsers'
```

```
# Get the S3 keys that have Object ACL grantees in addition to the object owner.
```

```
WITH grants AS
    (SELECT key,
    from_utf8(from_base64(object_access_control_list)) AS
 object_access_control_list,
         object_owner,
         CAST(json_extract(from_utf8(from_base64(object_access_control_list)),
         '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))) AS grants_array
    FROM your_table_name)
SELECT key,
       grant,
       objectowner
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE cardinality(grants_array) > 1 AND element_at(grant, 'canonicalId') !=
 object_owner;
```

```
# Get the S3 keys with READ permission that is granted in the Object ACL.
WITH grants AS (
    SELECT key,
        CAST(
            json_extract(from_utf8(from_base64(object_access_control_list)),
 '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))
        ) AS grants_array
    FROM your_table_name
)
SELECT key,
       grants_array,
       grant
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE element_at(grant, 'permission') = 'READ';
```

```
# Get the S3 keys that have Object ACL grants to a specific canonical user ID.
WITH grants AS (
    SELECT key,
        CAST(
            json_extract(from_utf8(from_base64(object_access_control_list)),
 '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))
        ) AS grants_array
    FROM your_table_name
)
SELECT key,
```

```
       grants_array,
       grant
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE element_at(grant, 'canonicalId') = 'user-canonical-id';
```

```
# Get the number of grantees on the Object ACL.
SELECT key,
       object_access_control_list,
       json_array_length(json_extract(object_access_control_list,'$.grants')) AS
 grants_count
FROM your_table_name;
```

Athena 사용에 대한 자세한 내용은 Amazon Athena 사용 설명서를 참조하십시오.

Amazon S3 인벤토리 보고서의 빈 버전 ID 문자열을 null 문자열로 변환

> **Note**
>
> 다음 절차는 모든 버전이 포함된 Amazon S3 인벤토리 보고서에만 적용되며 "모든 버전" 보고서가 S3 버전 관리가 사용 설정된 버킷에서 S3 배치 작업의 매니페스트로 사용되는 경우에만 적용됩니다. 현재 버전만 지정하는 S3 인벤토리 보고서의 문자열을 변환할 필요가 없습니다.

S3 인벤토리 보고서를 S3 배치 작업의 매니페스트로 사용할 수 있습니다. 그러나 버킷에서 S3 버전 관리가 사용 설정된 경우 모든 버전을 포함하는 S3 인벤토리 보고서는 버전 ID 필드에 빈 문자열이 있는 모든 null 버전이 지정된 객체를 표시합니다. 인벤토리 보고서에 모든 객체 버전 ID가 포함된 경우 배치 작업은 `null` 문자열을 버전 ID로 인식하지만 빈 문자열은 인식하지 않습니다.

S3 배치 작업이 "모든 버전" S3 인벤토리 보고서를 매니페스트로 사용하는 경우 버전 ID 필드에 빈 문자열이 있는 객체에 대한 모든 태스크가 실패합니다. S3 인벤토리 보고서의 버전 ID 필드에 있는 빈 문자열을 배치 작업용 `null` 문자열로 변환하려면 다음 절차를 따르세요.

배치 작업에 사용할 Amazon S3 인벤토리 보고서 업데이트

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.

2. S3 인벤토리 보고서로 이동합니다. 인벤토리 보고서는 인벤토리 보고서를 구성하는 동안 지정한 대상 버킷에 있습니다. 인벤토리 보고서 찾기에 대한 자세한 내용은 인벤토리 목록 찾기 섹션을 참조하세요.

   a. 대상 버킷을 선택합니다.

   b. 폴더를 선택합니다. 폴더 이름은 원래 원본 버킷의 이름을 따서 지정됩니다.

   c. 인벤토리 구성의 이름을 따서 명명된 폴더를 선택합니다.

   d. hive라는 폴더 옆의 확인란을 선택합니다. 페이지 상단에서 S3 URI 복사(Copy S3 URI)를 선택하여 폴더의 S3 URI를 복사합니다.

3. https://console.aws.amazon.com/athena/에서 Amazon Athena 콘솔을 엽니다.

4. 쿼리 편집기에서 설정(Settings)을 선택한 다음 관리(Manage)를 선택합니다. 설정 관리(Manage settings) 페이지의 쿼리 결과 위치(Location of query result)에서 쿼리 결과를 저장할 S3 버킷을 선택합니다.

5. 쿼리 편집기에서 다음 명령을 사용하여 인벤토리 보고서의 데이터를 보관할 Athena 테이블을 생성합니다. *table_name*을 원하는 이름으로 바꾸고, LOCATION 절에는 앞에서 복사한 S3 URI를 삽입합니다. 그런 다음 실행(Run)을 선택하여 쿼리를 실행합니다.

```
CREATE EXTERNAL TABLE table_name(bucket string, key string,
 version_id string) PARTITIONED BY (dt string)ROW FORMAT SERDE
 'org.apache.hadoop.hive.serde2.OpenCSVSerde'STORED AS INPUTFORMAT
 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat' OUTPUTFORMAT
 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat' LOCATION 'Copied S3 URI';
```

6. 쿼리 편집기를 지우려면 지우기(Clear)를 선택합니다. 그리고 다음 명령을 사용하여 인벤토리 보고서를 테이블에 로드합니다. *table_name*을 이전 단계에서 선택한 이름으로 바꿉니다. 그런 다음 실행(Run)을 선택하여 쿼리를 실행합니다.

```
MSCK REPAIR TABLE table_name;
```

7. 쿼리 편집기를 지우려면 지우기(Clear)를 선택합니다. 다음 SELECT 쿼리를 실행하여 원본 인벤토리 보고서의 모든 항목을 검색하고 빈 버전 ID를 null 문자열로 바꿉니다. *table_name*을 이전에 선택한 이름으로 바꾸고 WHERE 절의 *YYYY-MM-DD-HH-MM*을 이 도구를 실행하려는 인벤토리 보고서의 날짜로 바꿉니다. 그런 다음 실행(Run)을 선택하여 쿼리를 실행합니다.

```
SELECT bucket as Bucket, key as Key, CASE WHEN version_id = '' THEN 'null' ELSE
 version_id END as VersionId FROM table_name WHERE dt = 'YYYY-MM-DD-HH-MM';
```

8. Amazon S3 콘솔(https://console.aws.amazon.com/s3/)로 돌아가 이전에 쿼리 결과 위치(Location of query result)에서 선택한 S3 버킷으로 이동합니다. 내부에는 날짜로 끝나는 일련의 폴더가 있어야 합니다.

   예를 들어, s3://***amzn-s3-demo-bucket/query-result-location***/Unsaved/2021/10/07/과 같은 항목이 표시되어야 합니다. 실행한 SELECT 쿼리의 결과가 포함된 .csv 파일이 표시되어야 합니다.

   가장 최근에 수정된 날짜의 CSV 파일을 선택합니다. 다음 단계를 위해 이 파일을 로컬 시스템에 다운로드합니다.

9. 생성된 CSV 파일에는 헤더 행이 포함되어 있습니다. 배치 작업은 CSV 매니페스트의 헤더 행을 지원하지 않으므로 이 CSV 파일을 S3 배치 작업에 대한 입력으로 사용하려면 헤더 행을 제거해야 합니다.

   헤더 행을 제거하려면 파일에서 다음 명령 중 하나를 실행합니다. *file.csv*를 CSV 파일의 이름으로 바꿉니다.

   macOS 및 Linux 시스템의 경우 터미널 창에서 tail 명령을 실행합니다.

   ```
   tail -n +2 file.csv > tmp.csv && mv tmp.csv file.csv
   ```

   Windows 시스템의 경우 Windows PowerShell 창에서 다음 스크립트를 실행합니다. *File-location*을 파일 경로로, *file.csv*를 파일 이름으로 바꿉니다.

   ```
   $ins = New-Object System.IO.StreamReader File-location\file.csv
   $outs = New-Object System.IO.StreamWriter File-location\temp.csv
   try {
       $skip = 0
       while ( !$ins.EndOfStream ) {
           $line = $ins.ReadLine();
           if ( $skip -ne 0 ) {
               $outs.WriteLine($line);
           } else {
               $skip = 1
           }
       }
   } finally {
       $outs.Close();
       $ins.Close();
   }
   ```

```
Move-Item File-location\temp.csv File-location\file.csv -Force
```

10. CSV 파일에서 헤더 행을 제거한 후에는 S3 배치 작업에서 이를 매니페스트로 사용할 수 있습니다. CSV 파일을 선택한 S3 버킷 또는 위치에 업로드한 다음 CSV 파일을 매니페스트로 사용하여 배치 작업을 생성합니다.

    배치 작업 생성에 대한 자세한 내용은 S3 배치 작업 건 생성 섹션을 참조하세요.

객체 ACL 필드 작업

Amazon S3 인벤토리 보고서에는 S3 소스 버킷의 객체 목록과 각 객체의 메타데이터가 포함됩니다. 객체 액세스 제어 목록(ACL) 필드는 Amazon S3 인벤토리에서 사용할 수 있는 메타데이터 필드입니다. 특히 객체 ACL 필드에는 각 객체에 대한 액세스 제어 목록(ACL)이 포함되어 있습니다. 객체의 ACL은 이 객체에 대한 액세스 권한이 부여되는 AWS 계정 계정 또는 그룹과 부여되는 액세스 유형을 정의합니다. 자세한 내용은 ACL(액세스 제어 목록) 개요 및 Amazon S3 인벤토리 목록(을)를 참조하세요.

Amazon S3 인벤토리 보고서의 객체 ACL 필드는 JSON 형식으로 정의됩니다. JSON 데이터에는 다음 필드가 포함됩니다.

- `version` - 인벤토리 보고서의 객체 ACL 필드 형식 버전입니다. 데이터 형식 `yyyy-mm-dd`입니다.
- `status` - 가능한 값은 `AVAILABLE` 또는 `UNAVAILABLE`이며 객체에 대해 객체 ACL을 사용할 수 있는지 여부를 나타냅니다. 체 ACL의 상태가 `UNAVAILABLE`인 경우 인벤토리 보고서의 객체 소유자 필드 값도 `UNAVAILABLE`입니다.
- `grants` - 객체 ACL에 의해 부여된 각 피부여자의 권한 상태를 나열하는 피부여자-권한 쌍입니다. 피부여자가 사용할 수 있는 값은 `CanonicalUser` 및 `Group`입니다. 피부여자에 대한 자세한 내용은 액세스 제어 목록의 피부여자를 참조하세요.

  `Group` 형식이 있는 피부여자의 경우 피부여자-권한 쌍에는 다음 속성이 포함됩니다.

  - `uri` - 사전 정의된 Amazon S3 그룹입니다.
  - `permission` - 객체에 부여되는 ACL 권한입니다. 자세한 내용은 객체에 대한 ACL 권한을 참조하세요.
  - `type` - `Group` 형식으로 피부여자가 그룹임을 나타냅니다.

  `CanonicalUser` 형식이 있는 피부여자의 경우 피부여자-권한 쌍에는 다음 속성이 포함됩니다.

  - `canonicalId` - AWS 계정 ID의 난독화된 형식입니다. AWS 계정의 정식 사용자 ID는 해당 계정에만 적용됩니다. 표준 사용자 ID를 검색할 수 있습니다. 자세한 내용은 AWS 계정 관리 참조 안내서의 내 AWS 계정의 표준 사용자 ID 찾기를 참조하세요.

> **Note**
>
> ACL의 피부여자가 AWS 계정의 이메일 주소인 경우 S3 인벤토리는 해당 AWS 계정의 `canonicalId`와 `CanonicalUser` 유형을 사용하여 이 피부여자를 지정합니다. 자세한 내용은 액세스 제어 목록의 피부여자를 참조하세요.

- `permission` - 객체에 부여되는 ACL 권한입니다. 자세한 내용은 객체에 대한 ACL 권한을 참조하세요.
- `type` - `CanonicalUser` 형식으로 피부여자가 그AWS 계정임을 나타냅니다.

다음 예제에서는 JSON 형식의 객체 ACL 필드에 대해 가능한 값을 보여줍니다.

```
{
    "version": "2022-11-10",
    "status": "AVAILABLE",
    "grants": [{
        "uri": "http://acs.amazonaws.com/groups/global/AllUsers",
        "permission": "READ",
        "type": "Group"
    }, {
        "canonicalId": "example-canonical-id",
        "permission": "FULL_CONTROL",
        "type": "CanonicalUser"
    }]
}
```

> **Note**
>
> 객체 ACL 필드는 JSON 형식으로 정의됩니다. 인벤토리 보고서에는 객체 ACL 필드의 값이 base64로 인코딩된 문자열로 표시됩니다.
> 예를 들어 다음과 같은 JSON 형식의 객체 ACL 필드가 있다고 가정해 보겠습니다.
>
> ```
> {
>         "version": "2022-11-10",
>         "status": "AVAILABLE",
>         "grants": [{
>             "canonicalId": "example-canonical-user-ID",
>             "type": "CanonicalUser",
>             "permission": "READ"
> ```

```
        }]
}
```

객체 ACL 필드는 다음과 같이 base64로 인코딩된 문자열로 인코딩되어 표시됩니다.

```
eyJ2ZXJzaW9uIjoiMjAyMi0xMS0xMCIsInN0YXR1cyI6IkFWQUlMQUJMRSIsImdyYW50cyI6W3siY2Fub25pY2FsSW
```

객체 ACL 필드의 디코딩된 값을 JSON으로 가져오려면 Amazon Athena에서 이 필드를 쿼리하면 됩니다. 쿼리 예제를 보려면 Amazon Athena로 Amazon S3 인벤토리 쿼리 단원을 참조하세요.

모범 사례 설계 패턴: Amazon S3 성능 최적화

애플리케이션은 Amazon S3의 스토리지를 업로드하고 검색할 때 요청 성능에서 초당 수천 회의 트랜잭션을 쉽게 달성할 수 있습니다. Amazon S3는 높은 요청 빈도로 자동으로 조정됩니다. 예를 들어, 애플리케이션이 분할된 Amazon S3 접두사별로 초당 최소 3,500개의 PUT/COPY/POST/DELETE 또는 5,500개의 GET/HEAD 요청을 달성할 수 있습니다. 버킷의 접두사 수에는 제한이 없습니다. 병렬화를 사용하여 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. 마찬가지로, 여러 접두사에 쓰면 쓰기 작업의 크기를 조정할 수 있습니다. 읽기 및 쓰기 작업 모두의 경우 규모 조정은 즉각적이지 않고 점진적으로 수행되며 실제 성능은 특정 워크로드 특성, 사용 패턴 및 시스템 구성에 따라 달라집니다. Amazon S3가 더 높은 요청 비율에 맞춰 확장하는 동안 503(속도 저하) 오류가 발생할 수 있습니다. 이러한 오류는 크기 조정이 완료되면 사라집니다. 접두사 생성 및 사용에 대한 자세한 내용은 접두어를 사용한 객체 구성 단원을 참조하세요.

Amazon S3의 일부 데이터 레이크 애플리케이션은 페타바이트 이상의 데이터를 실행하는 쿼리에 대한 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 한 자릿수 밀리초 지연 시간에 비해 더 높은 전송 속도를 원한다면 Amazon S3로 캐싱하기 위해 Amazon CloudFront 또는 Amazon ElastiCache를 사용합니다.

또한 클라이언트와 S3 버킷 간 장거리에 걸쳐 고속 데이터 전송을 원할 경우 Amazon S3 Transfer Acceleration을 사용하여 빠르고 안전한 파일 전송 구성를 사용하십시오. Transfer Acceleration은 CloudFront에서 전 세계에 분산된 엣지 로케이션을 사용하여 지리적으로 먼 거리 간 데이터 전송을 가속화합니다. Amazon S3 워크로드에서 AWS KMS를 사용한 서버 측 암호화를 사용하는 경우 사용 사례에 지원되는 요청 빈도에 대한 자세한 정보는 AWS Key Management Service 개발자 안내서의 AWS KMS 제한을 참조하세요.

다음 주제에서는 Amazon S3를 사용하는 애플리케이션의 성능을 최적화하기 위한 모범 사례 지침과 설계 패턴에 대해 설명합니다. Amazon S3 성능 최적화에 대한 최신 정보는 Amazon S3 성능 지침 및 Amazon S3의 성능 디자인 패턴 섹션을 참조하십시오.

> **Note**
>
> Amazon S3 Express One Zone 스토리지 클래스를 디렉터리 버킷과 함께 사용하는 방법에 대한 자세한 내용은 S3 Express One Zone 및 디렉터리 버킷 작업 섹션을 참조하세요.

주제

- Amazon S3 성능 지침
- Amazon S3의 성능 디자인 패턴

Amazon S3 성능 지침

Amazon S3에서 객체를 업로드 및 검색하는 애플리케이션을 빌드할 때 모범 사례 지침에 따라 성능을 최적화합니다. 자세한 내용은 Amazon S3의 성능 디자인 패턴 섹션을 참조하십시오.

Amazon S3에서 애플리케이션의 성능을 최상으로 유지하려면 다음 지침을 따르는 것이 좋습니다.

주제

- 성능 측정
- 스토리지 연결 수평 규모 조정
- 바이트 범위 가져오기 사용
- 지연 시간에 민감한 애플리케이션 요청 재시도
- 동일한 AWS 리전에서 Amazon S3(스토리지)와 Amazon EC2(컴퓨팅) 결합
- 거리에 의해 발생하는 지연 시간을 최소화하기 위해 Amazon S3 Transfer Acceleration 사용
- 최신 AWS SDK 버전 사용

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 Amazon EC2 사용 설명서의 인스턴스 유형을 참조하세요.

성능을 측정할 때 HTTP 분석 도구를 사용하여 DNS 조회 시간, 지연 시간 및 데이터 전송 속도를 살펴보는 것도 도움이 됩니다.

성능 요구 사항을 이해하고 애플리케이션의 성능을 최적화하기 위해 수신되는 503 오류 응답을 모니터링할 수도 있습니다. 특정 성과 지표를 모니터링하면 추가 비용이 발생할 수 있습니다. 자세한 내용은 Amazon S3 요금을 참조하십시오.

503(속도 저하) 상태 오류 응답 수 모니터링

표시되는 503 상태 오류 응답 수를 모니터링하려면 다음 옵션 중 하나를 사용할 수 있습니다.

- Amazon S3에 대한 Amazon CloudWatch 요청 지표 사용 CloudWatch 요청 지표에는 5xx 상태 응답에 대한 지표가 포함됩니다. CloudWatch 요청 지표에 대한 자세한 내용은 Amazon CloudWatch를 사용한 지표 모니터링 섹션을 참조하세요.
- Amazon S3 스토리지 렌즈의 고급 지표 섹션에서 제공되는 503(서비스 사용 불가) 오류 수를 사용합니다. 자세한 내용은 성능 향상을 위한 S3 스토리지 렌즈 지표 사용 섹션을 참조하세요.
- Amazon S3 서버 액세스 로깅 사용 서버 액세스 로깅을 사용하면 503(내부 오류) 응답을 받는 모든 요청을 필터링하고 검토할 수 있습니다. 또한 Amazon Athena를 사용하여 로그를 구문 분석할 수도 있습니다. 서버 액세스 로깅에 대한 자세한 내용은 서버 액세스 로깅을 사용한 요청 로깅 단원을 참조하십시오.

HTTP 503 상태 오류 코드의 수를 모니터링하면 제한 요청이 가장 많이 발생하는 접두사, 키 또는 버킷에 대한 유용한 인사이트를 얻을 수 있습니다.

스토리지 연결 수평 규모 조정

여러 연결에 걸쳐 요청을 분산시키는 것은 성능을 수평으로 확장하는 일반적인 설계 패턴입니다. 고성능 애플리케이션을 빌드할 때는 Amazon S3를 기존의 스토리지 서버와 같은 단일 네트워크 엔드포인트가 아닌 아주 큰 분산 시스템처럼 생각하십시오. Amazon S3로 여러 건의 동시 요청을 보내 최상의 성능을 달성할 수 있습니다. Amazon S3에서 액세스할 수 있는 대역폭을 최대화하기 위해 이러한 요청을 별도의 연결로 분산합니다. Amazon S3는 버킷에 대한 연결 수 제한이 없습니다.

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 다운로드 섹션을 참조하세요.

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)와 Amazon EC2(컴퓨팅) 결합

S3 버킷 이름은 전 세계적으로 고유하지만 각 버킷은 버킷을 만들 때 선택한 리전에 저장됩니다. 버킷 이름 지정 지침에 대해 자세히 알아보려면 버킷 개요 및 버킷 이름 지정 규칙을 참조하세요. 성능을 최적화하려면, 가급적 같은 AWS 리전의 Amazon EC2 인스턴스에서 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 Acceleration을 사용하여 빠르고 안전한 파일 전송 구성을 사용하면 지리적으로 거리가 먼 클라이언트와 S3 버킷 간에 파일을 빠르고 쉽고 안전하게 전송할 수 있습니다. 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 AWS SDK 버전 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 따르

기 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 조정 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 참조를 참조하세요.

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 Amazon S3 성능 지침 도 제공합니다.

성능을 최적화하려면 다음과 같은 설계 패턴을 사용할 수 있습니다.

주제

- 자주 액세스하는 콘텐츠에 캐싱 사용
- 지연 시간에 민감한 애플리케이션의 제한 시간 및 재시도
- 높은 처리량을 위한 수평적 스케일링 및 요청 병렬화
- Amazon S3 Transfer Acceleration을 사용하여 지리적으로 다른 데이터 전송 가속화

자주 액세스하는 콘텐츠에 캐싱 사용

Amazon S3에 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 MediaStore와 같은 캐시를 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 인 메모리 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 캐시를 핫 객체로 채우고 Amazon S3에서 요청하기 전에 캐시에 핫 객체가 있는지 확인하도록 애플리케이션 로직을 수정합니다. ElastiCache를 사용하여 Amazon S3 GET 성능을 향상시키는 예는 블로그 게시물 Amazon ElastiCache for Redis를 사용한 Amazon S3 가속화를 참조하십시오.

AWS Elemental MediaStore는 Amazon S3의 비디오 워크플로와 미디어 전송 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 설명서를 참조하세요.

지연 시간에 민감한 애플리케이션의 제한 시간 및 재시도

애플리케이션이 Amazon S3에서 재시도를 해야 한다는 응답을 수신하는 경우가 있습니다. Amazon S3는 버킷 및 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 AWS SDK 및 도구 참조 가이드에서 동작 재시도를 참조하세요.

Amazon S3는 새로운 연속 요청 속도에 따라 자동으로 조정되어 동적으로 성능을 최적화합니다. Amazon S3가 새로운 요청 속도에 대해 내부적으로 최적화하는 동안 최적화가 완료될 때까지 일시적으로 HTTP 503 요청 응답을 받게 됩니다. Amazon S3가 새로운 요청 속도에 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 개발자 안내서의 할당량을 참조하세요.

높은 처리량을 위한 수평적 스케일링 및 요청 병렬화

Amazon S3는 매우 큰 분산 시스템입니다. 규모를 활용하려면 병렬 요청을 Amazon S3 서비스 엔드포인트까지 수평으로 조정하는 것이 좋습니다. 이 유형의 조정 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 EC2로 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%가 되며 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 피하

면 요청마다 TCP slow-start 및 SSL(Secure Sockets Layer) 핸드셰이크를 수행할 필요가 없습니다. REST API 사용에 대한 자세한 내용은 Amazon Simple Storage Service API 참조를 참조하세요.

마지막으로, DNS에 주의를 기울여 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 Amazon S3 API 참조에서 요청을 참조하세요.

Amazon S3 Transfer Acceleration을 사용하여 지리적으로 다른 데이터 전송 가속화

Amazon S3 Transfer Acceleration을 사용하여 빠르고 안전한 파일 전송 구성은 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 위치에 상호 접속 위치(POP)를 보유합니다. 현재는 CloudFront를 통해 콘텐츠를 배포하고 Amazon Route 53에 대해 수행된 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 Acceleration을 사용하면 더 높은 속도 향상을 기대할 수 있습니다.

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 성능이 향상될 가능성이 있는 전송에 대해서만 요금이 부과됩니다. 다른 AWS SDK와 함께 Transfer Acceleration을 사용하는 방법에 대한 정보는 S3 Transfer Acceleration 사용 설정 및 사용 섹션을 참조하세요.

Amazon S3를 사용하여 정적 웹 사이트 호스팅

Amazon S3을 사용하여 정적 웹 사이트를 호스팅할 수 있습니다. 정적 웹 사이트에서 개별 웹 페이지는 정적 콘텐츠를 포함합니다. 클라이언트 측 스크립트를 포함할 수도 있습니다.

> **Note**
>
> AWS Amplify Hosting을 사용하여 S3에 저장된 정적 웹 사이트 콘텐츠를 호스팅하는 것이 좋습니다. Amplify Hosting은 Amazon CloudFront로 구동되는 전 세계적으로 사용 가능한 콘텐츠 전송 네트워크(CDN)에 웹 사이트를 쉽게 배포할 수 있도록 하는 완전 관리형 서비스이므로 안전한 정적 웹 사이트 호스팅이 가능합니다.
>
> AWS Amplify Hosting을 사용하면 범용 버킷 내에서 객체의 위치를 선택하고, 관리형 CDN에 콘텐츠를 배포하고, 어디서나 웹 사이트에 액세스할 수 있는 퍼블릭 HTTPS URL을 생성할 수 있습니다. Amplify Hosting에 대한 자세한 내용은 AWS Amplify 콘솔 사용 설명서의 S3 범용 버킷에서 AWS Amplify Hosting에 정적 웹 사이트 배포 및 Amplify 콘솔을 사용하여 S3에서 정적 웹 사이트 배포를 참조하세요.

지침 및 단계별 연습을 포함하여 Amazon S3에서 정적 웹 사이트를 호스팅하는 방법에 대한 자세한 내용은 다음 주제를 참조하십시오.

주제

- 웹 사이트 엔드포인트
- 웹 사이트 호스팅 사용 설정
- 인덱스 문서 구성
- 사용자 지정 오류 문서 구성
- 웹 사이트 액세스에 대한 권한 설정
- (선택 사항) 웹 트래픽 로깅
- (선택 사항) 웹 페이지 리디렉션 구성
- 교차 오리진 리소스 공유(CORS) 사용
- 정적 웹 사이트 자습서

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 차이점을 참조하십시오.

리전에 따라 Amazon S3 웹 사이트 엔드포인트는 다음 두 형식 중 하나를 따릅니다.

- s3-website 대시(-) 리전 - `http://bucket-name.s3-website-Region.amazonaws.com`
- s3-website 점(.) 리전 - `http://bucket-name.s3-website.Region.amazonaws.com`

이러한 URL은 사용자가 웹 사이트에 대해 구성한 기본 인덱스 문서를 반환합니다. Amazon S3 웹 사이트 엔드포인트의 전체 목록은 Amazon S3 웹 사이트 엔드포인트를 참조하십시오.

> **Note**
>
> Amazon S3 정적 웹 사이트의 보안을 강화하기 위해 Amazon S3 웹 사이트 엔드포인트 도메인(예: s3-website-us-east-1.amazonaws.com 또는 s3-website.ap-south-1.amazonaws.com)은 공개 접미사 목록(PSL)에 등록되어 있습니다. 보안 강화를 위해 Amazon S3 정적 웹 사이트의 도메인 이름에 민감한 쿠키를 설정해야 하는 경우 `__Host-` 접두사가 있는 쿠키를 사용하는 것이 좋습니다. 이렇게 쿠키를 설정하면 교차 사이트 요청 위조 시도(CSRF)로부터 도메인을 보호하는 데 도움이 됩니다. 자세한 내용은 Mozilla 개발자 네트워크의 Set-Cookie 페이지를 참조하십시오.

웹 사이트를 퍼블릭으로 설정하려면 고객이 웹 사이트 엔드포인트의 콘텐츠에 액세스할 수 있도록 모든 콘텐츠를 공개적으로 읽기 가능하도록 만들어야 합니다. 자세한 정보는 웹 사이트 액세스에 대한 권한 설정 섹션을 참조하세요.

> **Important**
>
> - Amazon S3 웹 사이트 엔드포인트는 HTTPS 또는 액세스 포인트를 지원하지 않습니다. HTTPS를 사용하려면 다음 중 하나를 수행하세요.
>   - (권장) AWS Amplify Hosting을 사용하여 S3에 저장된 정적 웹 사이트 콘텐츠를 호스팅하세요. Amplify Hosting은 Amazon CloudFront로 구동되는 전 세계적으로 사용 가능한 콘텐

츠 전송 네트워크(CDN)에 웹 사이트를 쉽게 배포할 수 있도록 하는 완전 관리형 서비스이므로 안전한 정적 웹 사이트 호스팅이 가능합니다.

AWS Amplify Hosting을 사용하면 범용 버킷 내에서 객체의 위치를 선택하고, 관리형 CDN에 콘텐츠를 배포하고, 어디서나 웹 사이트에 액세스할 수 있는 퍼블릭 HTTPS URL을 생성할 수 있습니다. Amplify Hosting에 대한 자세한 내용은 AWS Amplify 콘솔 사용 설명서의 S3 범용 버킷에서 AWS Amplify Hosting에 정적 웹 사이트 배포 및 Amplify 콘솔을 사용하여 S3에서 정적 웹 사이트 배포를 참조하세요.

- Amazon CloudFront를 사용하여 Amazon S3에서 호스팅되는 정적 웹 사이트를 제공. 자세한 내용은 CloudFront를 사용하여 Amazon S3 버킷에 대한 HTTPS 요청을 처리하려면 어떻게 해야 하나요?를 참조하십시오 사용자 지정 도메인에서 HTTPS를 사용하려면 Route 53에 등록된 사용자 지정 도메인을 사용하여 정적 웹 사이트 구성을 참조하십시오.

- 요청자 지불 버킷은 웹 사이트 엔드포인트를 통한 액세스를 허용하지 않습니다. 해당 버킷으로의 모든 요청은 403 Access Denied 응답을 수신합니다. 자세한 내용은 스토리지 전송 및 사용량에 대한 요청자 지불 범용 버킷 사용 섹션을 참조하세요.

주제

- 웹 사이트 엔드포인트 예제
- DNS CNAME 추가
- Route 53에서 사용자 지정 도메인 사용
- 웹 사이트 엔드포인트와 REST API 엔드포인트 간의 주요 차이점

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 도메인 www.example-bucket.com이 있다면 www.example-bucket.com 버킷을 생성하고 www.example-bucket.com.s3-website.*Region*.amazonaws.com을 가리키는 DNS CNAME 레코드를 추가할 수 있습니다. http://www.example-bucket.com으로의 모든 요청은 www.example-bucket.com.s3-website.*Region*.amazonaws.com으로 라우팅됩니다.

자세한 내용은 CNAME 레코드를 사용하여 Amazon S3 URL 사용자 지정 섹션을 참조하세요.

Route 53에서 사용자 지정 도메인 사용

Amazon S3 웹 사이트 엔드포인트를 사용하여 웹 사이트에 액세스하는 대신, Amazon Route 53에 등록된 자체 도메인(예: example.com)을 사용하여 콘텐츠를 처리할 수 있습니다. Route 53과 함께 Amazon S3을 사용하여 루트 도메인에서 웹 사이트를 호스팅할 수 있습니다. 예를 들어, 루트 도메인 example.com을 보유하고 Amazon S3에서 웹 사이트를 호스팅하는 경우 웹 사이트 방문자는 http://www.example.com 또는 http://example.com을 입력하여 브라우저에서 사이트에 액세스할 수 있습니다.

예제 연습은 섹션을 참조하십시오자습서: Route 53에 등록된 사용자 지정 도메인을 사용하여 정적 웹 사이트 구성

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` 요청만 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 일반 참조에서 Amazon S3 엔드포인트 및 할당량을 참조하십시오.

웹 사이트 호스팅 사용 설정

버킷을 정적 웹 사이트로 구성할 때 정적 웹 사이트 호스팅을 사용 설정하고, 인덱스 문서를 추가하고, 권한을 설정해야 합니다.

Amazon S3 콘솔, REST API, AWS SDK, AWS CLI 또는 AWS CloudFormation을 사용하여 정적 웹 사이트 호스팅을 사용할 수 있습니다.

사용자 지정 도메인으로 웹 사이트를 구성하려면 자습서: Route 53에 등록된 사용자 지정 도메인을 사용하여 정적 웹 사이트 구성 섹션을 참조하십시오.

S3 콘솔 사용

정적 웹 사이트 호스팅 사용 설정

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 정적 웹 사이트 호스팅을 사용 설정하려는 버킷의 이름을 선택합니다.
4. [속성(Properties)]을 선택합니다.
5. 정적 웹 사이트 호스팅(Static website hosting)에서 편집(Edit)을 선택합니다.
6. 이 버킷을 사용하여 웹 사이트를 호스팅합니다.를 선택합니다.
7. 정적 웹 사이트 호스팅에서 사용을 선택합니다.
8. 인덱스 문서(Index document)에 인덱스 문서 이름을 입력합니다(일반적으로 `index.html`).

   인덱스 문서 이름은 대소문자를 구분하며 S3 버킷에 업로드하려는 HTML 인덱스 문서의 파일 이름과 정확히 일치해야 합니다. 웹 사이트 호스팅용 버킷을 구성하는 경우 인덱스 문서를 지정해야 합니다. 루트 도메인이나 임의의 하위 폴더로 요청이 전송되면 Amazon S3가 이 인덱스 문서를 반환합니다. 자세한 내용은 인덱스 문서 구성 섹션을 참조하세요.
9. 4XX 클래스 오류에 대한 사용자 지정 오류 문서를 제공하려면 [오류 문서(Error document)]에 사용자 지정 오류 문서 파일 이름을 입력합니다.

   오류 문서 이름은 대소문자를 구분하며 S3 버킷에 업로드하려는 HTML 오류 문서의 파일 이름과 정확히 일치해야 합니다. 사용자 지정 오류 문서를 지정하지 않았는데 오류가 발생하면 Amazon S3에서 기본 HTML 오류 문서를 반환합니다. 자세한 내용은 사용자 지정 오류 문서 구성 섹션을 참조하세요.
10. (선택 사항) 고급 리디렉션 규칙을 지정하려면 리디렉션 규칙(Redirection rules)에 JSON을 입력하여 규칙을 설명합니다.

    예를 들어, 요청의 특정 객체 키 이름 또는 접두사에 따라 조건부로 요청을 라우팅할 수 있습니다. 자세한 내용은 고급 조건부 리디렉션을 사용하도록 리디렉션 규칙 구성 섹션을 참조하세요.
11. [변경 사항 저장(Save changes)]을 선택합니다.

    Amazon S3는 버킷에 대한 정적 웹 사이트 호스팅을 지원합니다. 페이지 하단의 정적 웹 사이트 호스팅(Static website hosting)에 버킷의 웹 사이트 엔드포인트가 표시됩니다.
12. 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.

REST API 사용

정적 웹 사이트 호스팅을 사용 설정하기 위해 REST 요청을 직접 보내는 방법에 대한 자세한 내용은 Amazon Simple Storage Service API Reference의 다음 섹션을 참조하십시오.

- PUT 버킷 웹 사이트
- GET 버킷 웹 사이트
- DELETE 버킷 웹 사이트

AWS SDK 사용

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.

.NET

다음 예제는 AWS SDK for .NET를 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 S3를 사용하여 정적 웹 사이트 호스팅를 참조하십시오.

다음 C# 코드로 지정된 버킷에 웹 사이트 구성을 추가하는 예를 살펴봅니다. 이 구성은 인덱스 문서와 오류 문서 이름을 모두 지정합니다. 코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

```
using Amazon;
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);
            }
        }
    }
}
```

PHP

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 S3를 사용하여 정적 웹 사이트 호스팅를 참조하십시오.

AWS SDK for Ruby API에 대한 자세한 내용은 AWS SDK for Ruby – 버전 2를 참조하세요.

```
 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
]);
```

AWS CLI 사용

AWS CLI를 사용하여 S3 버킷을 정적 웹 사이트로 구성하는 방법에 대한 자세한 내용은 AWS CLI 명령 참조의 웹 사이트를 참조하십시오.

그런 다음 인덱스 문서를 구성하고 권한을 설정해야 합니다. 자세한 내용은 인덱스 문서 구성 및 웹 사이트 액세스에 대한 권한 설정 섹션을 참조하십시오.

오류 문서, 웹 트래픽 로깅 또는 리디렉션을 선택적으로 구성할 수도 있습니다.

인덱스 문서 구성

웹 사이트 호스팅을 사용 설정하는 경우 인덱스 문서도 구성하고 업로드해야 합니다. 인덱스 문서는 웹 사이트의 루트나 임의의 하위 폴더로 요청이 전달되는 경우에 Amazon S3이 반환하는 웹 페이지입니다. 예를 들어, 사용자가 브라우저에 `http://www.example.com`을 입력하는 경우 사용자는 특정 페이지를 요청하는 것입니다. 이 경우 Amazon S3이 기본 페이지라고도 하는 인덱스 문서를 표시합니다.

버킷용 정적 웹 사이트 호스팅을 사용 설정할 때 인덱스 문서의 이름(예: `index.html`)을 입력합니다. 버킷용 정적 웹 사이트 호스팅을 사용 설정한 후 인덱스 문서 이름이 있는 HTML 파일을 버킷에 업로드합니다.

루트 수준 URL의 후행 슬래쉬는 선택 사항입니다. 예를 들어, 인덱스 문서가 index.html인 웹 사이트를 구성하는 경우 다음과 같은 URL 중 하나에서 index.html이 반환됩니다.

```
http://example-bucket.s3-website.Region.amazonaws.com/
http://example-bucket.s3-website.Region.amazonaws.com
```

Amazon S3 웹 사이트 엔드포인트에 대한 자세한 내용은 웹 사이트 엔드포인트 섹션을 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 - 객체가 버킷의 루트에 있습니다.
- photos/2006/Jan/sample2.jpg - 객체가 photos/2006/Jan 하위 폴더에 있습니다.
- photos/2006/Feb/sample3.jpg - 객체가 photos/2006/Feb 하위 폴더에 있습니다.

Amazon S3 콘솔에서 버킷에 폴더를 생성할 수도 있습니다. 예를 들어 photos라는 폴더를 만들 수 있습니다. 이 버킷에 혹은 버킷 내부의 photos 폴더에 객체를 업로드할 수 있습니다. 버킷에 객체 sample.jpg를 추가하는 경우 키 이름은 sample.jpg입니다. photos 폴더에 객체를 업로드하는 경우 객체 키 이름은 photos/sample.jpg입니다.

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 photos를 찾습니다. photos 객체를 찾을 수 없는 경우 인덱스 문서인 photos/index.html이 검색 대상입니다. 해당 문서를 찾은 경우 Amazon S3이 302 Found 메시지를 반환하고 photos/ 키를 가리킵니다.

photos/에 대한 후속 요청의 경우 Amazon S3이 photos/index.html을 반환합니다. 인덱스 문서를 찾을 수 없는 경우 Amazon S3가 오류를 반환합니다.

인덱스 문서 구성

S3 콘솔을 사용하여 인덱스 문서를 구성하려면 다음 절차를 사용합니다. REST API, AWS SDK, AWS CLI 또는 AWS CloudFormation을 사용하여 인덱스 문서를 구성할 수도 있습니다.

> **Note**
>
> 버전 관리가 활성화된 버킷에서는 index.html의 사본을 여러 개 업로드할 수 있지만 최신 버전만 업로드할 수 있습니다. S3 버전 관리에 대한 자세한 내용은 S3 버전 관리로 여러 버전의 객체 유지 섹션을 참조하십시오.

버킷용 정적 웹 사이트 호스팅을 사용 설정할 때 인덱스 문서의 이름(예: **index.html**)을 입력합니다. 버킷용 정적 웹 사이트 호스팅을 사용 설정한 후 인덱스 문서 이름이 있는 HTML 파일을 버킷에 업로드합니다.

인덱스 문서 구성

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 저장합니다.

   인덱스 문서 파일 이름은 정적 웹 사이트 호스팅 대화 상자에 입력한 인덱스 문서 이름과 정확히 일치해야 합니다. 인덱스 문서 이름은 대/소문자를 구분합니다. 예를 들어 정적 웹 사이트 호스팅 대화 상자에서 인덱스 문서 이름에 index.html을 입력하는 경우, 인덱스 문서 파일은 index.html이 아니라 Index.html이어야 합니다.

3. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
4. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
5. 버킷 목록에서 정적 웹 사이트를 호스팅하는 데 사용할 버킷의 이름을 선택합니다.
6. 버킷에 정적 웹 사이트 호스팅을 사용 설정하고 인덱스 문서의 정확한 이름(예: `index.html`)을 입력합니다. 자세한 내용은 웹 사이트 호스팅 사용 설정 섹션을 참조하세요.

   정적 웹 사이트 호스팅을 사용 설정한 후 6단계로 이동합니다.
7. 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 객체 업로드 섹션을 참조하세요.
8. (선택 사항) 버킷에 다른 웹 사이트 콘텐츠를 업로드합니다.

그런 다음 웹 사이트 액세스에 대한 권한을 설정해야 합니다. 자세한 정보는 웹 사이트 액세스에 대한 권한 설정 섹션을 참조하세요.

오류 문서, 웹 트래픽 로깅 또는 리디렉션을 선택적으로 구성할 수도 있습니다.

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**
>
> 오류가 발생할 때 일부 브라우저는 자체 오류 메시지를 표시하므로 Amazon S3이 반환하는 오류 문서는 무시됩니다. 예를 들어, HTTP 404 찾을 수 없음 오류가 발생할 때 Google Chrome은 Amazon S3이 반환하는 오류 문서를 무시하고 자체 오류 메시지를 표시할 수도 있습니다.

주제

- Amazon S3 HTTP 응답 코드

- 사용자 지정 오류 문서 구성

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 S3은 301 영구 이동됨 응답을 반환하고 해당 요청을 `https://aws.amazon.com/s3/` 으로 리디렉션합니다.
302 Found(302 찾음)	x의 `http://`*`bucket-name`*` .s3-website. `*`Region`*`.amazonaws.com/x` 키에 대한 요청이 후행 슬래시 없이 Amazon S3에 수신되는 경우, 키 이름이 x인 객체가 첫 검색 대상이 됩니다. 객체를 찾을 수 없는 경우 Amazon S3은 해당 요청이 하위 폴더 x에 대한 것으로 판단하므로 맨 뒤에 슬래시를 추가하여 요청을 리디렉션하고 302 찾음을 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.
[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.

HTTP 오류 코드	설명
404 Not Found(404 찾을 수 없음)	웹 사이트 엔드포인트가 404 Not Found(404 찾을 수 없음)로 응답하는 이유는 다음과 같습니다. • Amazon S3에서 웹 사이트 URL이 존재하지 않는 객체 키를 참조한다고 판단함 • Amazon S3이 존재하지 않는 인덱스 문서에 대한 요청이라고 유추함 • 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 방법에 대한 자세한 내용은 인덱스 문서 구성 섹션을 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 빈도를 줄여야 한다고 Amazon S3이 판단하는 경우 웹 사이트 엔드포인트가 503 서비스 사용 불가로 응답합니다.

이러한 각 오류에 대해 Amazon S3은 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 제공할 수 있습니다. Amazon S3은 HTTP 4XX 클래스 오류 코드에 대한 사용자 지정 오류 문서만 반환합니다.

S3 콘솔을 사용하여 사용자 지정 오류 문서를 구성하려면 아래 단계를 따릅니다. REST API, AWS SDK, AWS CLI 또는 AWS CloudFormation을 사용하여 오류 문서를 구성할 수도 있습니다. 자세한 내용은 다음 자료를 참조하십시오.

- Amazon Simple Storage Service API Reference의 PutBucketWebsite
- AWS CloudFormation 사용 설명서의 AWS::S3::Bucket WebsiteConfiguration
- AWS CLI 명령 참조의 put-bucket-website

버킷용 정적 웹 사이트 호스팅을 사용 설정할 때 오류 문서의 이름(예: **`404.html`**)을 입력합니다. 버킷용 정적 웹 사이트 호스팅을 사용 설정한 후 오류 문서 이름이 있는 HTML 파일을 버킷에 업로드합니다.

오류 문서 구성

1. 오류 문서를 생성합니다(예: `404.html`).
2. 오류 문서 파일을 로컬에 저장합니다.

오류 문서 이름은 대/소문자를 구분하며 정적 웹 사이트 호스팅을 사용하도록 설정할 때 입력한 이름과 정확히 일치해야 합니다. 예를 들어 정적 웹 사이트 호스팅 대화 상자에서 오류 문서 이름에 `404.html`을 입력하는 경우, 오류 문서 파일은 `404.html`이어야 합니다.

3. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
4. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
5. 버킷 목록에서 정적 웹 사이트를 호스팅하는 데 사용할 버킷의 이름을 선택합니다.
6. 버킷에 정적 웹 사이트 호스팅을 사용 설정하고 오류 문서의 정확한 이름(예: `404.html`)을 입력합니다. 자세한 내용은 웹 사이트 호스팅 사용 설정 및 사용자 지정 오류 문서 구성(을)를 참조하세요.

   정적 웹 사이트 호스팅을 사용 설정한 후 6단계로 이동합니다.
7. 버킷에 오류 문서를 업로드하려면 다음 중 하나를 수행합니다.

   - 오류 문서 파일을 콘솔 버킷 목록으로 끌어다 놓습니다.
   - 업로드를 선택하고 프롬프트의 메시지에 따라 인덱스 파일을 선택하고 업로드합니다.

   단계별 지침은 객체 업로드 섹션을 참조하세요.

웹 사이트 액세스에 대한 권한 설정

버킷을 정적 웹 사이트로 구성할 때, 웹 사이트를 퍼블릭으로 설정하려면 퍼블릭 읽기 액세스 권한을 부여할 수 있습니다. 버킷을 공개적으로 읽기 가능하게 만들려면 버킷에 대해 퍼블릭 액세스 차단 설정을 사용 중지하고 퍼블릭 읽기 액세스 권한을 부여하는 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.

버킷에 대한 퍼블릭 액세스 차단 설정을 비활성화하지 않으면서 웹 사이트를 퍼블릭으로 설정하려면 Amazon CloudFront 배포를 생성하여 정적 웹 사이트를 제공할 수 있습니다. 자세한 내용은 Amazon CloudFront로 웹사이트 속도 향상 또는 Amazon Route 53 개발자 안내서에서 Amazon CloudFront 배포를 사용하여 정적 웹 사이트 제공을 참조하십시오.

> **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.

주제

- 1단계: S3 퍼블릭 액세스 차단 설정 편집
- 2단계: 버킷 정책 추가
- 객체 ACL(액세스 제어 목록)

1단계: S3 퍼블릭 액세스 차단 설정 편집

기존 버킷을 퍼블릭 액세스를 포함하는 정적 웹 사이트로 구성하려면 해당 버킷에 대한 퍼블릭 액세스 차단 설정을 편집해야 합니다. 계정 수준의 퍼블릭 액세스 차단 설정을 편집해야 할 수도 있습니다. Amazon S3은 버킷 수준 및 계정 수준 퍼블릭 액세스 차단 설정의 가장 제한적인 조합을 적용합니다.

예를 들어 버킷에 대한 퍼블릭 액세스는 허용하지만 계정 수준의 모든 퍼블릭 액세스는 차단하는 경우 Amazon S3에서 해당 버킷에 대한 퍼블릭 액세스를 계속 차단합니다. 이 시나리오에서는 버킷 수준 및 계정 수준의 퍼블릭 액세스 차단 설정을 편집해야 할 수도 있습니다. 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하세요.

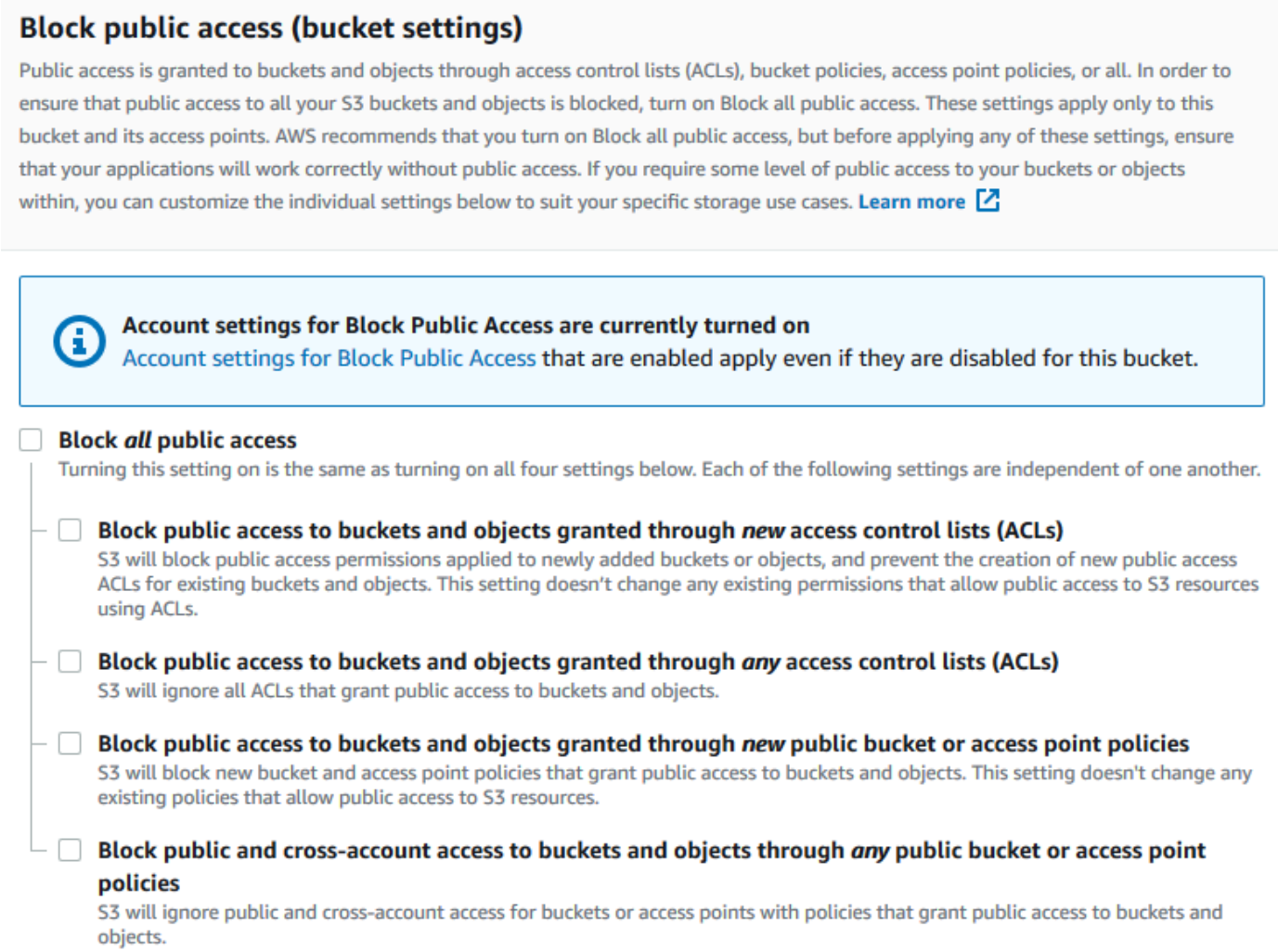
기본적으로 Amazon S3은 계정 및 버킷에 대한 퍼블릭 액세스를 차단합니다. 버킷을 사용하여 정적 웹 사이트를 호스팅하려는 경우 이러한 단계를 사용하여 퍼블릭 액세스 차단 설정을 편집할 수 있습니다.

Warning

이러한 단계를 완료하기 전에 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단을 검토하여 퍼블릭 액세스 허용과 관련된 위험을 이해하고 이에 동의하는지 확인합니다. 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. 퍼블릭 액세스 차단(버킷 설정)(Block public access (bucket settings))에서 편집(Edit)을 선택합니다.

5. 모든 퍼블릭 액세스 차단(Block all public access)을 선택 취소하고 변경 사항 저장(Save changes)을 선택합니다.



Amazon S3은 버킷에 대한 퍼블릭 액세스 차단 설정을 끕니다. 정적 퍼블릭 웹 사이트를 생성하려면 버킷 정책을 추가하기 전에 계정에 대한 퍼블릭 액세스 차단 설정을 편집해야 할 수도 있습니다. 계정에 대한 퍼블릭 액세스 차단 설정이 현재 켜져 있는 경우 퍼블릭 액세스 차단(버킷 설정) 아래에 메모가 표시됩니다.

2단계: 버킷 정책 추가

버킷의 객체를 공개적으로 읽기 가능하게 만들려면 모든 사용자에게 `s3:GetObject` 권한을 부여하는 버킷 정책을 작성해야 합니다.

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 버킷에 있는 파일을 보호하려면 어떻게 해야 하나요?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. 버킷 정책(Bucket Policy)에서 편집(Edit)을 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:::Bucket-Name/*"
            ]
        }
    ]
}
```

5. `Resource`를 버킷 이름으로 업데이트합니다.

   앞의 버킷 정책 예제에서 *Bucket-Name*은 버킷 이름의 자리 표시자입니다. 자체 버킷에 이 버킷 정책을 사용하려면 자체 버킷 이름과 일치하도록 이 이름을 업데이트해야 합니다.

6. [변경 사항 저장(Save changes)]을 선택합니다.

버킷 정책이 성공적으로 추가되었음을 나타내는 메시지가 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 하나요?를 참조하십시오.

오류 메시지가 나타나고 버킷 정책을 저장할 수 없는 경우 계정 및 버킷의 퍼블릭 액세스 차단 설정에서 버킷에 대한 퍼블릭 액세스를 허용하는지 확인합니다.

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.

S3 객체 소유권은 버킷에 업로드되는 객체의 소유권을 제어하고 ACL을 비활성화 또는 활성화하는 데 사용할 수 있는 Amazon S3 버킷 수준 설정입니다. 기본적으로 객체 소유권은 버킷 소유자 적용 설정으로 설정되며 모든 ACL이 비활성화되어 있습니다. ACL이 비활성화되면 버킷 소유자는 버킷의 모든 객체를 소유하고 액세스 관리 정책을 사용하여 객체에 대한 액세스를 독점적으로 관리합니다.

Amazon S3의 최신 사용 사례 대부분은 더 이상 ACL을 사용할 필요가 없습니다. 각 객체에 대해 액세스를 개별적으로 제어할 필요가 있는 드문 상황을 제외하고는 ACL을 비활성화한 채로 두는 것이 좋습니다. ACL을 비활성화하면 누가 객체를 버킷에 업로드했는지에 관계없이 정책을 사용하여 버킷의 모든 객체에 대한 액세스를 제어할 수 있습니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 섹션을 참조하세요.

> **Important**
>
> 범용 버킷이 S3 객체 소유권에 대해 버킷 소유자 적용 설정을 사용하는 경우 정책을 사용하여 범용 버킷과 버킷의 객체에 대한 액세스 권한을 부여해야 합니다. 버킷 소유자 적용 설정이 활성화된 상태에서 액세스 제어 목록(ACL) 설정 또는 ACL 업데이트 요청은 실패하고 `AccessControlListNotSupported` 오류 코드를 반환합니다. ACL 읽기 요청은 계속 지원됩니다.

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 ACL(액세스 제어 목록) 개요 섹션을 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

선택적으로 정적 웹 사이트로 구성된 버킷에 대한 Amazon S3 서버 액세스 로깅을 사용 설정할 수 있습니다. 서버 액세스 로깅은 버킷에 대해 이루어진 요청에 따른 상세 레코드를 제공합니다. 자세한 내용은 서버 액세스 로깅을 사용한 요청 로깅 섹션을 참조하세요. Amazon CloudFront를 사용하여 웹 사이트 속도를 높이려는 경우 CloudFront 로깅을 사용할 수도 있습니다. 자세한 내용은 Amazon CloudFront 개발자 안내서의 액세스 로그 구성 및 사용을 참조하십시오.

정적 웹 사이트 버킷에 대한 서버 액세스 로깅 사용 설정

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 정적 웹 사이트로 구성된 버킷을 생성한 리전에서 로깅용 범용 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.

   자세한 내용은 Amazon CloudFront로 웹사이트 속도 향상 섹션을 참조하세요.
5. 버킷 목록에서 버킷을 선택합니다.
6. [속성(Properties)]을 선택합니다.
7. 서버 액세스 로깅(Server access logging)에서 편집(Edit)을 선택합니다.
8. 사용을 선택합니다.
9. 대상 버킷(Target bucket)에서 서버 액세스 로그의 버킷과 폴더 대상을 선택합니다.

   - 폴더 및 버킷 위치를 찾습니다.
     1. S3 찾아보기(Browse S3)를 선택합니다.
     2. 버킷 이름을 선택한 다음 로그 폴더를 선택합니다.
     3. 경로 선택(Choose path)을 선택합니다.

- S3 버킷 경로(예: **s3://logs.example.com/logs/**)를 입력합니다.

10. [변경 사항 저장(Save changes)]을 선택합니다.

    이제 로그 버킷에서 로그에 액세스할 수 있습니다. Amazon S3은 2시간마다 웹 사이트 액세스 로그를 로그 버킷에 기록합니다.

(선택 사항) 웹 페이지 리디렉션 구성

웹 사이트 호스팅용으로 Amazon S3 버킷이 구성된 경우 버킷 또는 버킷 내 객체에 대한 리디렉션을 구성할 수 있습니다. 리디렉션 구성에 대해 다음과 같은 옵션이 있습니다.

주제

- 버킷의 웹 사이트 엔드포인트에 대한 요청을 다른 버킷이나 도메인으로 리디렉션
- 고급 조건부 리디렉션을 사용하도록 리디렉션 규칙 구성
- 객체에 대한 요청 리디렉션

버킷의 웹 사이트 엔드포인트에 대한 요청을 다른 버킷이나 도메인으로 리디렉션

버킷의 웹 사이트 엔드포인트에 대한 모든 요청을 다른 버킷 또는 도메인으로 리디렉션할 수 있습니다. 모든 요청을 리디렉션하면 웹 사이트 엔드포인트로 전송된 요청이 지정된 버킷 또는 도메인으로 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`이라는 이름의 버킷을 2개 만들어야 합니다. 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 요청 리디렉션

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷(Buckets) 아래에서 요청을 리디렉션할 버킷의 이름(예: `www.example.com`)을 선택합니다.
3. [속성(Properties)]을 선택합니다.
4. 정적 웹 사이트 호스팅(Static website hosting)에서 편집(Edit)을 선택합니다.

5. 객체에 대한 요청 리디렉션(Redirect requests for an object)을 선택합니다.
6. 호스트 이름(Host name) 상자에 버킷 또는 사용자 지정 도메인의 웹 사이트 엔드포인트를 입력합니다.

   예를 들어, 루트 도메인 주소로 리디렉션하려면 **example.com**을 입력해야 합니다.
7. 프로토콜(Protocol)에서는 리디렉션된 요청에 대한 프로토콜(없음, http 또는 https)을 선택합니다.

   프로토콜을 지정하지 않으면 기본 옵션은 없음입니다.
8. [변경 사항 저장(Save changes)]을 선택합니다.

고급 조건부 리디렉션을 사용하도록 리디렉션 규칙 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 가정해 봅시다. 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.

버킷용 정적 웹 사이트 호스팅을 사용하도록 설정할 때 고급 리디렉션 규칙을 선택적으로 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 S3 콘솔 사용 섹션을 참조하세요.

REST API를 사용하여 라우팅 규칙을 구성하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 PutBucketWebsite를 참조하십시오.

> **Important**
>
> 새 Amazon S3 콘솔에서 리디렉션 규칙을 생성하려면 JSON을 사용해야 합니다. JSON 예제는 리디렉션 규칙 예제 섹션을 참조하세요.

정적 웹 사이트에 대한 리디렉션 규칙 구성

정적 웹 사이트 호스팅이 이미 사용 설정되어 있는 버킷에 대한 리디렉션 규칙을 추가하려면 다음 단계를 따르십시오.

1. https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.

2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 정적 웹 사이트로 구성한 버킷의 이름을 선택합니다.
4. [속성(Properties)]을 선택합니다.
5. 정적 웹 사이트 호스팅(Static website hosting)에서 편집(Edit)을 선택합니다.
6. 리디렉션 규칙(Redirection rules) 상자에서 JSON에 리디렉션 규칙을 입력합니다.

   S3 콘솔에서는 JSON을 사용하여 규칙을 설명합니다. JSON 예제는 리디렉션 규칙 예제 섹션을 참조하세요. Amazon S3에는 웹 사이트 구성당 50개의 라우팅 규칙 제한이 있습니다.
7. [변경 사항 저장(Save changes)]을 선택합니다.

라우팅 규칙 요소

다음은 JSON 및 XML의 웹 사이트 구성에서 라우팅 규칙을 정의하기 위한 일반 구문입니다. 새 S3 콘솔에서 리디렉션 규칙을 구성하려면 JSON을 사용해야 합니다. JSON 예제는 리디렉션 규칙 예제 섹션을 참조하십시오.

JSON

```
[
    {
      "Condition": {
        "HttpErrorCodeReturnedEquals": "string",
        "KeyPrefixEquals": "string"
      },
      "Redirect": {
        "HostName": "string",
        "HttpRedirectCode": "string",
        "Protocol": "http"|"https",
        "ReplaceKeyPrefixWith": "string",
        "ReplaceKeyWith": "string"
      }
    }
  ]

Note: Redirect must each have at least one child element. You can have either
 ReplaceKeyPrefix with or ReplaceKeyWith but not both.
```

XML

```
<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. 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:

이름	설명
	• 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 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.

이름	설명
Protocol	응답에서 반환된 http 헤더에 사용할 https 또는 Location 프로토콜입니다. 해당 형제 요소 중 하나가 제공되는 경우 Protocol는 필요하지 않습니다.
HostName	응답에서 반환된 Location 헤더에 사용할 호스트 이름입니다. 해당 형제 요소 중 하나가 제공되는 경우 HostName는 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.

리디렉션 규칙 예제

다음 예제는 다음과 같은 일반적인 리디렉션 작업을 설명합니다.

> **Important**
>
> 새 Amazon S3 콘솔에서 리디렉션 규칙을 생성하려면 JSON을 사용해야 합니다.

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.html에 대한 요청은 documents/article1.html로 리디렉션됩니다.

이 경우 웹 사이트 구성에 다음 라우팅 규칙을 추가합니다.

JSON

```
[
    {
        "Condition": {
            "KeyPrefixEquals": "docs/"
        },
        "Redirect": {
            "ReplaceKeyPrefixWith": "documents/"
        }
    }
]
```

XML

```
  <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.html라는 이름의 페이지로 리디렉션하는 라우팅 규칙을 추가할 수 있습니다.

JSON

```
[
    {
        "Condition": {
            "KeyPrefixEquals": "images/"
        },
        "Redirect": {
            "ReplaceKeyWith": "folderdeleted.html"
        }
    }
]
```

XML

```
  <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 요청을 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스로 리디렉션하려고 한다고 가정합니다. HTTP 상태 코드 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 요청했지만 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.

JSON

```
[
    {
        "Condition": {
            "HttpErrorCodeReturnedEquals": "404"
        },
        "Redirect": {
            "HostName": "ec2-11-22-333-44.compute-1.amazonaws.com",
            "ReplaceKeyPrefixWith": "report-404/"
        }
    }
]
```

XML

```
  <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>
```

객체에 대한 요청 리디렉션

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 API를 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 섹션을 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.

S3 콘솔 사용

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷(Buckets) 목록에서 정적 웹 사이트로 구성한 버킷의 이름(예: `example.com`)을 선택합니다.
3. 객체(Objects)에서 객체를 선택합니다.
4. 작업(Actions)을 선택하고 메타데이터 편집(Edit metadata)을 선택합니다.
5. 메타데이터를 선택합니다.
6. 메타데이터 추가(Add Metadata)를 선택합니다.
7. 유형(Type)에서 시스템 정의(System Defined)를 선택합니다.
8. 키(Key)에서 x-amz-website-redirect-location을 선택합니다.
9. 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`).
10. 메타데이터 편집(Edit metadata)을 선택합니다.

REST API 사용

다음 Amazon S3 API 작업은 요청의 `x-amz-website-redirect-location` 헤더를 지원합니다. Amazon S3은 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 S3이 `x-amz-website-redirect-location` 속성 값에 따라 페이지 요청을 리디렉션합니다.
- REST 엔드포인트 - Amazon S3이 페이지 요청을 리디렉션하지 않습니다. 이 경우 요청된 객체가 반환됩니다.

엔드포인트에 대한 자세한 내용은 웹 사이트 엔드포인트와 REST API 엔드포인트 간의 주요 차이점 섹션을 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 기존 `page1.html` 객체를 원본으로 사용하는 새 `page1.html` 객체를 생성할 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.

교차 오리진 리소스 공유(CORS) 사용

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 S3으로 다양한 기능의 클라이언트 측 웹 애플리케이션을 구축하고, Amazon S3 리소스에 대한 Cross-Origin 액세스를 선택적으로 허용할 수 있습니다.

이 섹션에서는 CORS 개요를 다룹니다. 하위 섹션에서는 Amazon S3 콘솔을 사용하거나 프로그래밍 방식으로 Amazon S3 REST API 및 AWS SDK를 사용하여 CORS를 사용하는 방법을 설명합니다.

CORS(Cross-Origin Resource Sharing): 사용 사례 시나리오

다음은 CORS 사용에 대한 예제 시나리오입니다.

시나리오 1

Amazon S3를 사용하여 정적 웹 사이트 호스팅에서 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 S3의 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 사용 설정하도록 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.

Amazon S3은 버킷의 CORS 구성을 어떻게 평가하나요?

Amazon S3이 브라우저에서 preflight 요청을 받으면 버킷에 대한 CORS 구성을 평가하고 수신된 브라우저 요청과 일치하는 첫 번째 `CORSRule` 규칙을 사용하여 Cross-Origin 요청을 허용합니다. 규칙이 일치하려면 다음 조건이 충족되어야 합니다.

- 버킷에 대한 CORS 요청의 `Origin` 헤더는 CORS 구성의 `AllowedOrigins` 요소에 있는 오리진과 일치해야 합니다.
- 버킷에 대한 CORS 요청의 `Access-Control-Request-Method`에 지정된 HTTP 메서드는 CORS 구성의 `AllowedMethods` 요소에 나열된 메서드와 일치해야 합니다.
- 사전 요청의 `Access-Control-Request-Headers` 헤더에 나열된 헤더는 CORS 구성의 `AllowedHeaders` 요소에 있는 헤더와 일치해야 합니다.

> **Note**
>
> 버킷에 대해 CORS를 허용하면 ACL과 정책이 계속 적용됩니다.

객체 Lambda 액세스 포인트가 CORS를 지원하는 방법

브라우저로부터 요청을 받거나 요청에 `Origin` 헤더가 포함된 경우 S3 Object Lambda는 항상 `"AllowedOrigins":"*"` 헤더 필드를 추가합니다.

CORS 사용에 대한 자세한 내용은 다음 주제를 참조하십시오.

주제

- CORS 구성의 요소
- 교차 오리진 리소스 공유(CORS) 구성
- CORS 테스트
- CORS 문제 해결

CORS 구성의 요소

Cross-Origin 요청을 허용하도록 버킷을 구성하려면 CORS 구성을 생성합니다. CORS 구성은 버킷에 액세스할 수 있도록 허용할 오리진, 각 오리진에 대해 지원되는 작업(HTTP 메서드) 및 기타 작업별 정

보를 식별하는 요소가 포함된 문서입니다. 구성당 최대 100개의 규칙을 추가할 수 있습니다. CORS 구성을 `cors` 하위 리소스로 버킷에 추가할 수 있습니다.

S3 콘솔에서 CORS를 구성하는 경우 JSON을 사용하여 CORS 구성을 생성해야 합니다. 새로운 S3 콘솔은 JSON CORS 구성만 지원합니다.

CORS 구성 및 구성 내 요소에 대한 자세한 내용은 아래 주제를 참조하십시오. CORS 구성을 추가하는 방법에 대한 지침은 교차 오리진 리소스 공유(CORS) 구성 섹션을 참조하십시오.

> **Important**
>
> S3 콘솔에서 CORS 구성은 JSON이어야 합니다.

주제

- AllowedMethods 요소
- AllowedOrigins 요소
- AllowedHeaders 요소
- ExposeHeaders 요소
- MaxAgeSeconds 요소
- CORS 구성의 예

`AllowedMethods` 요소

CORS 구성에서 `AllowedMethods` 요소에 대한 다음 값을 지정할 수 있습니다.

- GET
- PUT
- POST
- DELETE
- HEAD

`AllowedOrigins` 요소

`AllowedOrigins` 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 https를 지정하여 안전한 오리진만 허용할 수도 있습니다.

AllowedHeaders 요소

AllowedHeaders 요소는 Access-Control-Request-Headers 헤더를 통해 preflight 요청에 허용되는 헤더를 지정합니다. Access-Control-Request-Headers 헤더의 각 헤더 이름은 요소의 해당 항목과 일치해야 합니다. Amazon S3은 요청된 응답에서 허용된 헤더만을 보냅니다. Amazon S3에 대한 요청에 사용할 수 있는 샘플 헤더 목록을 보려면 Amazon Simple Storage Service API 참조 가이드의 공통 요청 헤더를 참조하십시오.

구성의 각 AllowedHeaders 문자열에는 최대 한 개의 * 와일드카드 문자를 포함할 수 있습니다. 예를 들어, <AllowedHeader>x-amz-*</AllowedHeader>는 모든 Amazon별 헤더를 허용합니다.

ExposeHeaders 요소

각 ExposeHeader 요소는 응답에서 고객이 해당 애플리케이션(예: JavaScript XMLHttpRequest 객체)으로부터 액세스할 수 있도록 하려는 헤더를 식별합니다. 공통 Amazon S3 응답 헤더 목록을 보려면 Amazon Simple Storage Service API 참조 가이드의 공통 응답 헤더를 참조하십시오.

MaxAgeSeconds 요소

MaxAgeSeconds 요소는 브라우저가 리소스, HTTP 메서드, 오리진으로 식별되는 preflight 요청에 대한 응답을 캐시할 수 있는 시간(초)을 지정합니다.

CORS 구성의 예

Amazon S3 웹 사이트 엔드포인트를 사용하여 웹 사이트에 액세스하는 대신, example1.com과 같은 자체 도메인을 사용하여 콘텐츠를 제공할 수 있습니다. 자체 도메인 사용에 대한 자세한 내용은 자습서: Route 53에 등록된 사용자 지정 도메인을 사용하여 정적 웹 사이트 구성 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. Amazon S3은 preflight OPTIONS 요청에 대한 응답으로 요청된 헤더를 반환합니다.
- 두 번째 규칙은 첫 번째 규칙과 같이 cross-origin 요청을 허용하지만 다른 오리진인 http://www.example2.com에 적용됩니다.

- 세 번째 규칙은 모든 오리진에서 cross-origin GET 요청을 허용합니다. * 와일드카드 문자는 모든 오리진을 나타냅니다.

JSON

```
[
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "http://www.example1.com"
        ],
        "ExposeHeaders": []
    },
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "http://www.example2.com"
        ],
        "ExposeHeaders": []
    },
    {
        "AllowedHeaders": [],
        "AllowedMethods": [
            "GET"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": []
```

```
    }
]
```

XML

```
<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.

JSON

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

```
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "http://www.example.com"
        ],
        "ExposeHeaders": [
            "x-amz-server-side-encryption",
            "x-amz-request-id",
            "x-amz-id-2"
        ],
        "MaxAgeSeconds": 3000
    }
]
```

XML

```
<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 Amazon S3에 preflight 요청을 보낼 필요가 없습니다.

- `ExposeHeaders` - 고객이 자체 애플리케이션(예: JavaScript `x-amz-server-side-encryption` 객체)에서 액세스할 수 있는 응답 헤더(이 예에서는 `x-amz-request-id`, `x-amz-id-2`, `XMLHttpRequest`)를 식별합니다.

교차 오리진 리소스 공유(CORS) 구성

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 S3으로 다양한 기능의 클라이언트 측 웹 애플리케이션을 구축하고, Amazon S3 리소스에 대한 Cross-Origin 액세스를 선택적으로 허용할 수 있습니다.

이 섹션에서는 Amazon S3 콘솔, Amazon S3 REST API 및 AWS SDK를 사용하여 CORS를 사용하는 방법을 보여줍니다. 교차 오리진 요청을 허용하도록 버킷을 구성하려면 해당 버킷에 CORS 구성을 추가합니다. CORS 구성은 버킷에 대한 액세스를 허용할 오리진과 각 오리진에 대해 지원되는 작업(HTTP 메서드) 그리고 기타 작업별 정보를 각각 식별하는 규칙과 기타 작업별 정보를 포함하는 문서입니다. S3 콘솔에서 CORS 구성은 JSON 문서여야 합니다.

JSON 및 XML의 CORS 구성 예제는 CORS 구성의 요소 섹션을 참조하십시오.

S3 콘솔 사용

이 섹션에서는 Amazon S3 콘솔을 사용하여 S3 버킷에 CORS(Cross-Origin 리소스 공유) 구성을 추가하는 방법을 설명합니다.

버킷에 대해 CORS를 사용 설정하면 ACL(액세스 제어 목록)과 기타 액세스 권한 정책이 계속 적용됩니다.

> **Important**
>
> S3 콘솔에서 CORS 구성은 JSON이어야 합니다. JSON 및 XML의 CORS 구성 예제는 CORS 구성의 요소 섹션을 참조하십시오.

S3 버킷에 CORS 구성 추가

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 버킷 정책을 만들 버킷의 이름을 선택합니다.

4. Permissions를 선택합니다.
5. CORS(Cross-Origin 리소스 공유) 섹션에서 편집을 선택합니다.
6. CORS 구성 편집기 텍스트 상자에 새 CORS 구성을 입력하거나 복사하여 붙여넣거나, 기존 구성을 편집합니다.

   CORS 구성은 JSON 파일입니다. 편집기에 입력하는 텍스트는 유효한 JSON이어야 합니다. 자세한 내용은 CORS 구성의 요소 단원을 참조하십시오.
7. [변경 사항 저장(Save changes)]을 선택합니다.

> **Note**
>
> Amazon S3는 CORS 구성 편집기 제목 옆에 버킷의 Amazon 리소스 이름(ARN)을 표시합니다. ARN에 대한 자세한 내용은 Amazon 리소스 이름(ARN) 및 AWS 서비스 네임스페이스를 참조하십시오.

AWS SDK 사용

AWS SDK를 사용하여 버킷에 대한 cross-origin 리소스 공유(CORS)를 관리할 수 있습니다. CORS에 대한 자세한 내용은 교차 오리진 리소스 공유(CORS) 사용 섹션을 참조하십시오.

다음 예제를 고려하십시오.

- CORS 구성을 만들고 버킷에 구성 설정
- 구성을 검색하고 규칙을 추가하여 구성 수정
- 버킷에 수정된 구성 추가
- 구성 삭제

Java

Example

Example

실제 예제를 작성하여 테스트하는 방법에 대한 자세한 내용은 AWS SDK for Java 개발자 안내서에서 시작하기 섹션을 참조하세요.

```
import com.amazonaws.AmazonServiceException;
```

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

.NET

Example

코드 예제 설정 및 실행에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 AWS SDK for .NET 시작하기를 참조하세요.

```
using Amazon;
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 API 사용

버킷에 CORS 구성을 설정하려면 AWS Management Console을 사용할 수 있습니다. 하지만 애플리케이션에서 요구할 경우 REST 요청을 직접 전송할 수도 있습니다. Amazon Simple Storage Service API 참조의 다음 섹션에서는 CORS 구성과 관련된 REST API 작업에 대해 설명합니다.

- PutBucketCors
- GetBucketCors
- DeleteBucketCors
- OPTIONS 객체

CORS 테스트

CORS 구성을 테스트하려면 OPTIONS 메서드와 함께 CORS 사전 요청을 전송하여 요청 전송이 허용되는 경우 서버가 응답하도록 할 수 있습니다. Amazon S3가 사전 요청을 받으면 S3에서 버킷에 대한

CORS 구성을 평가하고 수신된 요청과 일치하는 첫 번째 CORSRule 규칙을 사용하여 교차 오리진 요청을 허용합니다. 규칙이 일치하려면 다음 조건이 충족되어야 합니다.

- 버킷에 대한 CORS 요청의 Origin 헤더는 CORS 구성의 AllowedOrigins 요소에 있는 오리진과 일치해야 합니다.
- 버킷에 대한 CORS 요청의 Access-Control-Request-Method에 지정된 HTTP 메서드는 CORS 구성의 AllowedMethods 요소에 나열된 메서드와 일치해야 합니다.
- 사전 요청의 Access-Control-Request-Headers 헤더에 나열된 헤더는 CORS 구성의 AllowedHeaders 요소에 있는 헤더와 일치해야 합니다.

다음은 CORS 구성 예제입니다. CORS 구성을 만들려면 CORS 구성을 참조하세요. CORS 구성의 자세한 예제는 CORS 구성의 요소를 참조하세요.

CORS 규칙 구성 및 문제 해결에 대한 지침은 AWS re:Post 지식 센터의 Amazon S3에서 CORS를 구성하고 cURL을 사용하여 CORS 규칙을 확인하려면 어떻게 해야 하나요?를 참조하세요.

JSON

```
[
    {
        "AllowedHeaders": [
            "Authorization"
        ],
        "AllowedMethods": [
            "GET",
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "http://www.example1.com"
        ],
        "ExposeHeaders":  [
             "x-amz-meta-custom-header"
        ]

    }
]
```

CORS 구성을 테스트하려면 다음 CURL 명령을 사용하여 사전 요청 OPTIONS 검사를 보낼 수 있습니다. CURL은 S3와 상호 작용하는 데 사용할 수 있는 명령줄 도구입니다. 자세한 내용은 CURL을 참조하세요.

```
curl -v -X OPTIONS \
 -H "Origin: http://www.example1.com" \
 -H "Access-Control-Request-Method: PUT" \
 -H "Access-Control-Request-Headers: Authorization" \
 -H "Access-Control-Expose-Headers: x-amz-meta-custom-header"\
    "http://bucket_name.s3.amazonaws.com/object_prefix_name"
```

위 예제에서 `curl -v -x OPTIONS` 명령은 S3에 사전 요청을 보내 S3가 교차 오리진 `http://www.example1.com`에서 객체에 대한 PUT 요청을 보내는 것을 허용하는지 문의하는 데 사용됩니다. 헤더 `Access-Control-Request-Headers` 및 `Access-Control-Expose-Headers`는 선택 사항입니다.

- Amazon S3는 사전 OPTIONS 요청의 `Access-Control-Request-Method` 헤더에 대한 응답으로 요청된 메서드가 일치하는 경우 허용된 메서드 목록을 반환합니다.
- Amazon S3는 사전 OPTIONS 요청의 `Access-Control-Request-Headers` 헤더에 대한 응답으로 요청된 헤더가 일치하는 경우 허용된 헤더 목록을 반환합니다.
- Amazon S3는 사전 OPTIONS 요청의 `Access-Control-Expose-Headers` 헤더에 대한 응답으로 요청된 헤더가 브라우저에서 실행되는 스크립트에서 액세스할 수 있는 허용 헤더와 일치하는 경우 허용된 헤더 목록을 반환합니다.

> **Note**
>
> 사전 요청을 전송할 때 CORS 요청 헤더 중 하나라도 허용되지 않는 경우 응답 CORS 헤더는 반환되지 않습니다.

이 사전 OPTIONS 요청에 대한 응답으로 `200 OK` 응답을 받게 됩니다. CORS를 테스트할 때 수신되는 일반적인 오류 코드와 CORS 관련 문제 해결을 위한 자세한 내용은 CORS 문제 해결을 참조하세요.

```
< HTTP/1.1 200 OK
< Date: Fri, 12 Jul 2024 00:23:51 GMT
< Access-Control-Allow-Origin: http://www.example1.com
< Access-Control-Allow-Methods: GET, PUT, POST, DELETE
< Access-Control-Allow-Headers: Authorization
```

```
< Access-Control-Expose-Headers: x-amz-meta-custom-header
< Access-Control-Allow-Credentials: true
< Vary: Origin, Access-Control-Request-Headers, Access-Control-Request-Method
< Server: AmazonS3
< Content-Length: 0
```

CORS 문제 해결

다음 주제는 S3와 관련된 몇 가지 일반적인 CORS 문제를 해결하는 데 도움이 될 수 있습니다.

주제

- 403 Forbidden 오류: 이 버킷에 대해 CORS가 활성화되지 않음
- 403 Forbidden 오류: 이 CORS 요청은 허용되지 않음
- CORS 응답에서 헤더를 찾을 수 없음
- S3 프록시 통합의 CORS 관련 고려 사항

403 Forbidden 오류: 이 버킷에 대해 CORS가 활성화되지 않음

교차 오리진 요청이 Amazon S3로 전송되었지만 CORS가 S3 버킷에 구성되지 않은 경우 다음 `403 Forbidden` 오류가 발생합니다.

오류: HTTP/1.1 403 Forbidden CORS Response: CORS is not enabled for this bucket.

CORS 구성은 버킷에 액세스할 수 있도록 허용할 오리진, 각 오리진에 대해 지원되는 작업(HTTP 메서드) 및 기타 작업별 정보를 식별하는 규칙이 포함된 문서 또는 정책입니다. Amazon S3 콘솔, AWS SDK 및 REST API를 사용하여 S3에 CORS를 구성하는 방법을 알아보세요. CORS에 대한 자세한 내용과 CORS 구성의 예는 CORS의 요소를 참조하세요.

403 Forbidden 오류: 이 CORS 요청은 허용되지 않음

CORS 구성의 CORS 규칙이 요청의 데이터와 일치하지 않는 경우 다음 `403 Forbidden` 오류가 발생합니다.

오류: HTTP/1.1 403 Forbidden CORS Response: This CORS request is not allowed.

따라서 다음과 같은 여러 가지 이유로 이 `403 Forbidden` 오류가 발생할 수 있습니다.

- 오리진이 허용되지 않습니다.
- 메서드가 허용되지 않습니다.

- 요청된 헤더가 허용되지 않습니다.

Amazon S3가 받은 각 요청에 대해 해당 요청의 데이터와 일치하는 CORS 규칙이 CORS 구성에 있어야 합니다.

오리진이 허용되지 않음

버킷에 대한 CORS 요청의 `Origin` 헤더는 CORS 구성의 `AllowedOrigins` 요소에 있는 오리진과 일치해야 합니다. `AllowedOrigins` 요소의 와일드카드 문자("*")는 모든 HTTP 메서드와 일치합니다. `AllowedOrigins` 요소를 업데이트하는 방법에 대한 자세한 내용은 교차 오리진 리소스 공유 (CORS) 구성을 참조하세요.

예를 들어 `AllowedOrigins` 요소에 `http://www.example1.com` 도메인만 포함된 경우 `http://www.example2.com` 도메인에서 보낸 CORS 요청에서 `403 Forbidden` 오류가 발생합니다.

다음 예시에서는 `AllowedOrigins` 요소에 `http://www.example1.com` 도메인이 포함된 CORS 구성의 일부를 보여줍니다.

```
"AllowedOrigins":[
   "http://www.example1.com"
]
```

`http://www.example2.com` 도메인에서 보낸 CORS 요청이 성공하려면 해당 `http://www.example2.com` 도메인이 CORS 구성의 `AllowedOrigins` 요소에 포함되어야 합니다.

```
"AllowedOrigins":[
   "http://www.example1.com"
   "http://www.example2.com"
]
```

메서드가 허용되지 않음

버킷에 대한 CORS 요청의 `Access-Control-Request-Method`에 지정된 HTTP 메서드는 CORS 구성의 `AllowedMethods` 요소에 나열된 메서드와 일치해야 합니다. `AllowedMethods`의 와일드카드 문자("*")는 모든 HTTP 메서드와 일치합니다. `AllowedOrigins` 요소를 업데이트하는 방법에 대한 자세한 내용은 교차 오리진 리소스 공유(CORS) 구성을 참조하세요.

CORS 구성에서 `AllowedMethods` 요소에 다음 메서드를 지정할 수 있습니다.

- `GET`

- PUT
- POST
- DELETE
- HEAD

다음 예시에서는 AllowedMethods 요소에 GET 메서드가 포함된 CORS 구성의 일부를 보여줍니다. GET 메서드를 포함한 요청만 성공합니다.

```
"AllowedMethods":[
   "GET"
]
```

HTTP 메서드(예:PUT)가 CORS 요청에 사용되었거나 버킷에 대한 사전 CORS 요청에 포함되었으나 해당 메서드가 CORS 구성에 없는 경우 요청에서 403 Forbidden 오류가 발생합니다. 이 CORS 요청 또는 CORS 사전 요청을 허용하려면 PUT 메서드를 CORS 구성에 추가해야 합니다.

```
"AllowedMethods":[
   "GET"
   "PUT"
]
```

요청된 헤더가 허용되지 않음

사전 요청의 Access-Control-Request-Headers 헤더에 나열된 헤더는 CORS 구성의 AllowedHeaders 요소에 있는 헤더와 일치해야 합니다. Amazon S3에 대한 요청에 사용할 수 있는 공통 헤더 목록은 공통 요청 헤더를 참조하세요. AllowedHeaders 요소를 업데이트하는 방법에 대한 자세한 내용은 교차 오리진 리소스 공유(CORS) 구성을 참조하세요.

다음 예시에서는 AllowedHeaders 요소에 Authorization 헤더가 포함된 CORS 구성의 일부를 보여줍니다. Authorization 헤더에 대한 요청만 성공합니다.

```
"AllowedHeaders":  [
    "Authorization"
]
```

헤더(예: Content-MD5)가 CORS 요청에 포함되었지만 해당 헤더가 CORS 구성에 없는 경우 요청에서 403 Forbidden 오류가 발생합니다. 이 CORS 요청을 허용하려면 Content-MD5 헤더를 CORS

구성에 추가해야 합니다. CORS 요청의 Authorization 헤더와 Content-MD5 헤더를 모두 버킷으로 전달하려면 두 헤더가 모두 CORS 구성의 AllowedHeaders 요소에 포함되어 있는지 확인합니다.

```
"AllowedHeaders":  [
    "Authorization"
    "Content-MD5"
]
```

CORS 응답에서 헤더를 찾을 수 없음

CORS 구성의 ExposeHeaders 요소는 CORS 요청에 대한 응답으로 브라우저에서 실행되는 스크립트와 애플리케이션이 액세스할 수 있도록 설정할 응답 헤더를 식별합니다.

S3 버킷에 저장된 객체에 응답 데이터와 함께 사용자 정의 메타데이터(예:x-amz-meta-custom-header)가 있는 경우 이 사용자 지정 헤더에는 클라이언트측 JavaScript 코드에서 액세스할 추가 메타데이터 또는 정보가 포함될 수 있습니다. 하지만 기본적으로 브라우저는 보안상의 이유로 사용자 지정 헤더에 대한 액세스를 차단합니다. 클라이언트측 JavaScript가 사용자 지정 헤더에 액세스할 수 있도록 하려면 CORS 구성에 해당 헤더를 포함해야 합니다.

아래 예시에서 x-amz-meta-custom-header1 헤더는 ExposeHeaders 요소에 포함되어 있습니다. x-amz-meta-custom-header2는 ExposeHeaders 요소에 포함되지 않았으며 CORS 구성에서 누락되었습니다. 응답에서는 ExposeHeaders 요소에 포함된 값만 반환됩니다. 요청에서 Access-Control-Expose-Headers 헤더에 x-amz-meta-custom-header2 헤더가 포함된 경우에도 응답은 여전히 200 OK를 반환합니다. 하지만 허용된 헤더(예: x-amz-meta-custom-header)만 반환되어 응답에 표시됩니다.

```
"ExposeHeaders":  [
    "x-amz-meta-custom-header1"
]
```

모든 헤더가 응답에 표시되도록 하려면 아래와 같이 허용된 모든 헤더를 CORS 구성의 ExposeHeaders 요소에 추가합니다.

```
"ExposeHeaders":  [
    "x-amz-meta-custom-header1",
    "x-amz-meta-custom-header2"
]
```

S3 프록시 통합의 CORS 관련 고려 사항

오류가 발생하며 이미 S3 버킷의 CORS 구성을 확인했고 교차 오리진 요청이 AWS CloudFront와 같은 프록시로 전송되는 경우 다음 작업을 시도해 봅니다.

- `OPTIONS` 메서드에서 HTTP 요청을 허용하도록 설정을 구성합니다.
- `Origin`, `Access-Control-Request-Headers` 및 `Access-Control-Request-Method` 헤더를 전달하도록 프록시를 구성합니다.
- 캐시 키에 오리진 헤더를 포함하도록 프록시 설정을 구성합니다. 이는 캐시 키에 오리진 헤더를 포함하지 않는 캐싱 프록시가 다른 오리진에 적절한 CORS 헤더를 포함하지 않는 캐싱된 응답을 제공할 수 있기 때문에 중요합니다.

일부 프록시는 CORS 요청에 대한 사전 정의된 기능을 제공합니다. 예를 들어 CloudFront에서 헤더를 포함하는 정책을 구성할 수 있습니다.

이러한 헤더는 오리진이 Amazon S3 버킷일 때 Cross-Origin Resource Sharing(CORS) 요청을 활성화하는 헤더입니다.

이 정책에는 다음 설정이 포함되어 있습니다.

- 오리진 요청에 포함된 헤더:

  `Origin`

  `Access-Control-Request-Headers`

  `Access-Control-Request-Method`
- 오리진 요청에 포함된 쿠키: 없음
- 오리진 요청에 포함된 쿼리 문자열: 없음

자세한 내용은 Amazon CloudFront 개발자 안내서의 정책을 통한 오리진 요청 제어 또는 관리형 오리진 요청 정책 사용을 참조하세요.

정적 웹 사이트 자습서

다음 튜토리얼 또는 연습에서는 정적 웹 사이트 호스팅 및 온디맨드 비디오 스트리밍 호스팅을 위한 Amazon S3 범용 버킷을 만들고 구성하는 방법에 대한 절차 전체를 안내합니다. 이 자습서 목적은 일

반적인 지침을 제공하는 것입니다. 이러한 자습서는 예제 버킷 이름 및 사용자 이름 등을 통해 랩 유형의 환경에 맞게 고안되었습니다. 조직의 고유한 요구 사항에 맞는지 주의 깊은 검토 및 조정 없이 프로덕션 환경에 바로 사용할 수 있도록 고안된 것이 아닙니다.

- Amazon S3, Amazon CloudFront 및 Amazon Route 53로 온디맨드 스트리밍 비디오 호스팅 – Amazon S3를 Amazon CloudFront와 함께 사용하여 안전하고 확장 가능한 방식으로 온디맨드 보기 용도의 비디오를 호스팅할 수 있습니다. 비디오를 올바른 포맷으로 패키징했으면 서버나 S3 범용 버킷에 저장한 후 최종 사용자들이 요청할 때 CloudFront를 사용하여 제공할 수 있습니다. 이 자습서에서는 전송을 위한 CloudFront와 도메인 이름 시스템(DNS) 및 사용자 지정 도메인 관리를 위한 Amazon Route 53를 사용하여 온디맨드 비디오 스트리밍을 호스팅하도록 범용 버킷을 구성하는 방법을 배울 수 있습니다. CloudFront는 해당 캐시에서 비디오를 제공하고, 아직 캐싱되지 않은 경우에만 범용 버킷에서 비디오를 검색합니다. 이 캐싱 관리 기능은 짧은 대기 시간, 높은 처리량, 빠른 전송 속도로 전 세계 뷰어에게 비디오를 빠르게 제공할 수 있습니다. CloudFront 캐싱 관리에 대한 자세한 내용은 Amazon CloudFront 개발자 가이드의 캐싱 및 가용성 최적화를 참조하십시오.
- 정적 웹 사이트 구성 – 웹 사이트처럼 작동하도록 범용 버킷을 구성할 수 있습니다. 이 자습서에서는 버킷 만들기, S3 콘솔에서의 정적 웹 사이트 호스팅 활성화, 인덱스 문서 만들기, 오류 문서 만들기 등 Amazon S3에 웹 사이트를 호스팅하는 단계를 안내합니다. 자세한 내용은 Amazon S3를 사용하여 정적 웹 사이트 호스팅을 참조하세요.
- Route 53에 등록된 사용자 지정 도메인을 사용하여 정적 웹 사이트 구성 - 정적 웹 사이트를 호스팅하는 범용 버킷을 생성 및 구성하고 S3에서 Amazon Route 53에 등록된 사용자 지정 도메인 이름의 웹 사이트에 대한 리디렉션을 만들 수 있습니다. Route 53를 사용하여 도메인을 등록하고 도메인의 인터넷 트래픽을 라우팅할 위치를 정의합니다. 이 자습서에서는 도메인 및 하위 도메인의 트래픽을 HTML 파일이 포함된 범용 버킷으로 라우팅하는 Route 53 별칭 레코드를 만드는 방법을 보여줍니다. 자세한 내용은 Amazon Route 53 개발자 안내서의 Amazon S3 버킷의 정적 웹 사이트에 대한 도메인 사용을 참조하세요. 이 자습서를 완료한 후에는 필요에 따라 CloudFront를 사용하여 웹 사이트의 성능을 향상시킬 수 있습니다. 자세한 내용은 Amazon CloudFront로 웹 사이트 속도 향상을 참조하세요.
- S3 범용 버킷에서 AWS Amplify Hosting에 정적 웹 사이트 배포 - AWS Amplify Hosting을 사용하여 S3에 저장된 정적 웹 사이트 콘텐츠를 호스팅하는 것이 좋습니다. Amplify Hosting은 Amazon CloudFront로 구동되는 전 세계적으로 사용 가능한 콘텐츠 전송 네트워크(CDN)에 웹 사이트를 쉽게 배포할 수 있도록 하는 완전 관리형 서비스이므로 광범위한 설정을 하지 않아도 안전한 정적 웹 사이트 호스팅이 가능합니다. AWS Amplify Hosting을 사용하면 범용 버킷 내에서 객체의 위치를 선택하고, 관리형 CDN에 콘텐츠를 배포하고, 어디서나 웹 사이트에 액세스할 수 있는 퍼블릭 HTTPS URL을 생성할 수 있습니다. 자세한 내용은 AWS Amplify Hosting 사용 설명서의 Amplify 콘솔을 사용하여 S3에서 정적 웹 사이트 배포를 참조하세요.

자습서: Amazon S3, Amazon CloudFront 및 Amazon Route 53로 온디맨드 스트리밍 비디오 호스팅

Amazon S3를 Amazon CloudFront와 함께 사용하여 보안 및 확장성이 있는 방식의 온디맨드 보기 용도로 비디오를 호스팅할 수 있습니다. 온디맨드 비디오(VOD) 스트리밍은 비디오 콘텐츠가 서버에 저장되고 뷰어가 언제든지 비디오 콘텐츠를 볼 수 있음을 의미합니다.

CloudFront는 빠르고 안전하며 프로그래밍 가능한 콘텐츠 전송 네트워크(CDN) 서비스입니다. CloudFront는 전세계적으로 모든 CloudFront의 엣지 로케이션에서 HTTPS를 통해 콘텐츠를 안전하게 전송할 수 있습니다. CloudFront 에 대한 자세한 내용은 Amazon CloudFront 개발자 가이드의 Amazon CloudFront란 무엇입니까? 단원을 참조하십시오.

CloudFront 캐싱은 오리진 서버에서 직접 응답해야 하는 요청의 수를 줄입니다. 뷰어(최종 사용자)의 경우 CloudFront에서 제공하는 비디오를 요청하면 해당 요청은 뷰어의 위치와 가까운 엣지 로케이션으로 라우팅됩니다. CloudFront는 해당 캐시에서 비디오를 제공하고, 아직 캐싱되지 않은 경우에만 S3 버킷에서 비디오를 검색합니다. 이 캐싱 관리 기능은 짧은 대기 시간, 높은 처리량, 빠른 전송 속도로 전 세계 뷰어에게 비디오를 빠르게 제공할 수 있습니다. CloudFront 캐싱 관리에 대한 자세한 내용은 Amazon CloudFront 개발자 가이드의 캐싱 및 가용성 최적화를 참조하십시오.



목표

이 자습서에서는 전송을 위해 CloudFront를 사용하고 도메인 이름 시스템(DNS) 및 사용자 지정 도메인 관리를 위해 Amazon Route 53를 사용하여 온디맨드 비디오 스트리밍을 호스팅하도록 S3 버킷을 구성합니다.

주제

- 사전 조건: Route 53에 사용자 지정 도메인 등록 및 구성
- 1단계: S3 버킷 생성
- 2단계: S3 버킷에 파일 업로드
- 3단계: CloudFront 원본 액세스 자격 증명 생성
- 4단계: CloudFront 배포 생성
- 5단계: CloudFront 배포를 통해 비디오에 액세스
- 6단계: 사용자 지정 도메인 이름을 사용하도록 CloudFront 배포 구성
- 7단계: 사용자 지정 도메인 이름을 사용하여 CloudFront 배포를 통해 S3 비디오에 액세스
- 8단계: CloudFront 배포에서 수신한 요청에 대한 데이터 보기(선택 사항)
- 9단계: 정리
- 다음 단계

사전 조건: Route 53에 사용자 지정 도메인 등록 및 구성

이 자습서를 시작하기 전에, 나중에 사용자 지정 도메인 이름을 사용하도록 CloudFront 배포를 구성하려면 Route 53에 사용자 지정 도메인(예: **example.com**)을 등록하고 구성해야 합니다.

사용자 지정 도메인 이름이 없으면 S3 비디오가 다음과 유사한 URL에서 CloudFront를 통해 공개적으로 액세스되고 호스팅됩니다.

```
https://CloudFront distribution domain name/Path to an S3 video
```

예: **https://d111111abcdef8.cloudfront.net/sample.mp4**.

Route 53으로 구성된 사용자 지정 도메인 이름을 사용하도록 CloudFront 배포를 구성하면 S3 비디오가 다음과 유사한 URL에서 CloudFront를 통해 공개적으로 액세스되고 호스팅됩니다.

```
https://CloudFront distribution alternate domain name/Path to an S3 video
```

예: **https://www.example.com/sample.mp4**. 사용자 지정 도메인 이름은 뷰어가 보다 간단하고 직관적으로 사용할 수 있습니다.

사용자 지정 도메인을 등록하려면 Amazon Route 53 개발자 안내서의 Route 53을 사용하여 새 도메인 이름 등록을 참조하십시오.

Route 53을 사용하여 도메인 이름을 등록하면 Route 53은 이 자습서의 뒷부분에서 사용할 호스팅 영역을 생성합니다. 이 호스팅 영역은 트래픽을 도메인(예: Amazon EC2 인스턴스 또는 CloudFront 배포)으로 라우팅하는 방법에 대한 정보를 저장하는 곳입니다.

도메인 등록, 호스팅 영역 및 도메인에서 수신한 DNS 쿼리와 관련된 요금이 있습니다. 자세한 내용은 Amazon Route 53 요금을 참조하십시오.

> **Note**
>
> 도메인을 등록하면 즉시 비용이 청구되며, 이는 취소할 수 없습니다. 도메인을 자동 갱신하지 않도록 선택할 수 있지만, 선불로 비용을 지불하며 1년 동안 소유합니다. 자세한 내용은 Amazon Route 53 개발자 안내서의 새 도메인 등록을 참조하십시오.

1단계: S3 버킷 생성

스트리밍하려는 원본 비디오를 저장할 버킷을 생성합니다.

버킷을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 버킷을 생성하려는 리전을 선택합니다.

   > **Note**
   >
   > 지연 시간과 요금을 최소화하고 규제 요건을 충족하려면 가장 가까운 리전을 선택하십시오. 특정 리전에 저장된 객체는 사용자가 명시적으로 객체를 다른 리전으로 전송하지 않는 한 해당 리전을 벗어나지 않습니다. Amazon S3 AWS 리전 목록은 Amazon Web Services 일반 참조의 AWS 서비스 엔드포인트를 참조하십시오.
3. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
4. 버킷 만들기를 선택합니다. 버킷 만들기 페이지가 열립니다.
5. 버킷 이름에 버킷 이름을 입력합니다(예: **`tutorial-bucket`**).

Amazon S3의 버킷 이름 지정 규칙에 대한 자세한 내용은 범용 버킷 이름 지정 규칙 섹션을 참조하십시오.

6. 리전(Region)에서 버킷이 속할 AWS 리전을 선택합니다.

   가능하면 뷰어의 대다수와 가장 가까운 리전을 선택해야 합니다. 버킷 리전에 대한 자세한 내용은 범용 버킷 개요 섹션을 참조하십시오.

7. 이 버킷에 대한 퍼블릭 액세스 차단 설정(Block Public Access settings for this bucket)에서 해당 설정을 기본값으로 유지합니다(모든 퍼블릭 액세스 차단(Block all public access)이 활성화됨).

   모든 퍼블릭 액세스 차단을 활성화한 경우에도 뷰어는 CloudFront를 통해 업로드된 비디오에 계속 액세스할 수 있습니다. 이 기능은 CloudFront를 사용하여 S3에 저장된 비디오를 호스팅할 때의 주요 이점입니다.

   해당 사용 사례에 대해 하나 이상의 설정을 해제해야 하는 경우가 아니라면 모든 설정을 활성화 상태로 유지하는 것이 좋습니다. 퍼블릭 액세스 차단에 대한 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 섹션을 참조하십시오.

8. 나머지 설정은 기본값으로 유지합니다.

   특정 사용 사례에 대한 추가 버킷 설정을 구성하려면 범용 버킷 생성 섹션을 참조하십시오(선택 사항).

9. 버킷 생성을 선택합니다.

2단계: S3 버킷에 파일 업로드

다음 절차에서는 콘솔을 사용하여 S3 버킷에 비디오 파일을 업로드하는 방법을 설명합니다. S3에 많은 대용량 비디오파일을 업로드하는 경우 Amazon S3 Transfer Acceleration을 사용하여 빠르고 안전한 파일 전송을 구성할 수 있습니다. Transfer Acceleration을 사용하면 S3 버킷에 비디오를 빠르게 업로드하여 대용량의 비디오를 장거리 전송할 수 있습니다. 자세한 내용은 Amazon S3 Transfer Acceleration을 사용하여 빠르고 안전한 파일 전송 구성 단원을 참조하십시오.

버킷에 파일을 업로드하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 범용 버킷 목록에서 1단계에서 생성한 버킷의 이름(예: **`tutorial-bucket`**)을 선택하여 파일을 업로드합니다.

4. 버킷의 객체(Objects) 탭에서 업로드(Upload)를 선택합니다.
5. 업로드(Upload) 페이지의 파일 및 폴더(Files and Folders)에서 파일 추가(Add Files)를 선택합니다.
6. 업로드할 파일을 선택한 후 열기를 선택합니다.

   예를 들어 `sample.mp4` 비디오 파일을 업로드할 수 있습니다.
7. 업로드를 선택합니다.

3단계: CloudFront 원본 액세스 자격 증명 생성

S3 버킷에서 비디오에 대한 직접 액세스를 제한하려면, 원본 액세스 ID(OAI)라는 특별한 CloudFront 사용자를 생성합니다. 이 자습서의 후반에서는 OAI를 배포와 연결합니다. OAI를 사용하면 CloudFront를 우회하여 S3 버킷에서 직접 비디오를 가져올 수 없도록 할 수 있습니다. CloudFront OAI만 S3 버킷의 파일에 액세스할 수 있습니다. 자세한 내용은 Amazon CloudFront 개발자 안내서의 OAI를 사용하여 Amazon S3 콘텐츠에 대한 액세스 제한 단원을 참조하십시오.

CloudFront OAI를 생성하려면

1. AWS Management Console에 로그인한 다음 https://console.aws.amazon.com/cloudfront/v4/home에서 CloudFront 콘솔을 엽니다.
2. 왼쪽 탐색 창의 보안 섹션에서 오리진 액세스를 선택합니다.
3. 자격 증명 탭에서 오리진 액세스 ID 생성을 선택합니다.
4. 새로운 원본 액세스 ID에 이름을 입력합니다(예: **`S3-OAI`**).
5. 생성(Create)을 선택합니다.

4단계: CloudFront 배포 생성

CloudFront를 사용하여 S3 버킷에서 비디오를 제공하고 배포하려면 CloudFront 배포를 생성해야 합니다.

하위 단계

- CloudFront 배포를 생성합니다.
- 버킷 정책 검토

CloudFront 배포를 생성합니다.

1. AWS Management Console에 로그인한 다음 https://console.aws.amazon.com/cloudfront/v4/home에서 CloudFront 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 배포를 선택합니다.
3. 배포 생성을 선택합니다.
4. 원본 섹션의 원본 도메인에서, 1단계에서 생성한 S3 버킷의 이름으로 시작하는 S3 원본의 도메인 이름을 선택합니다(예: **`tutorial-bucket`**).
5. 원본 액세스에서 레거시 액세스 ID를 선택합니다.
6. 원본 액세스 ID에서, 3단계에서 생성한 원본 액세스 ID를 선택합니다(예: **`S3-OAI`**).
7. 버킷 정책(Bucket policy)에서, 예, 버킷 정책을 업데이트합니다(Yes, update the bucket policy)를 선택합니다.
8. 기본 캐시 동작 섹션의 뷰어 프로토콜 정책에서 HTTP를 HTTPS로 리디렉션을 선택합니다.

   이 기능을 선택하는 경우, 웹 사이트를 보호하고 뷰어의 데이터를 보호하기 위해 HTTP 요청이 자동으로 HTTPS로 리디렉션됩니다.
9. 기본 캐시 동작 섹션의 기타 설정은 기본값을 유지하십시오.

   (선택 사항) CloudFront에서 다른 요청을 원본에 전달하기 전에 파일을 CloudFront 캐시에 보관하는 기간을 제어할 수 있습니다. 이 기간을 단축함으로써 동적 콘텐츠를 제공할 수 있습니다. 이 기간이 늘어나면 파일이 엣지 캐시에서 바로 제공될 가능성이 높으므로 뷰어에게 제공되는 성능이 향상됩니다. 보관 기간이 늘어나면 오리진에 걸리는 부하 역시 줄어듭니다. 자세한 내용은 Amazon CloudFront 개발자 안내서의 콘텐츠가 엣지 캐시에 유지되는 기간 관리(만료)를 참조하십시오.
10. 다른 섹션의 경우 나머지 설정을 기본값으로 유지합니다.

    다양한 옵션에 대한 자세한 내용은 Amazon CloudFront 개발자 안내서의 배포의 생성 또는 업데이트 시 지정하는 값을 참조하십시오.
11. 페이지 맨 아래에서 배포 생성(Create distribution)을 선택합니다.
12. CloudFront 배포에 대한 일반 탭의 세부 정보에서, 배포에 대한 마지막 수정 날짜 열 값이 배포 중에서 배포가 마지막으로 수정된 타임스탬프로 변경됩니다. 이 작업은 일반적으로 몇 분 정도 걸립니다.

버킷 정책 검토

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷목록에서, 이전에 CloudFront 배포의 원본으로 사용한 버킷의 이름을 선택합니다(예: **`tutorial-bucket`**).
4. 권한 탭을 선택합니다.
5. 버킷 정책 섹션에서 버킷 정책 텍스트 상자에 다음과 비슷한 문이 표시되는지 확인합니다.

```
{
    "Version": "2008-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
            "Sid": "1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
 Identity EH1HDMB1FH2TC"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::tutorial-bucket/*"
        }
    ]
}
```

이는 이전에 예, 버킷 정책을 업데이트합니다를 선택했을 때CloudFront 배포가 버킷 정책에 추가한 문입니다.

이 버킷 정책 업데이트는 S3 버킷에 대한 액세스를 제한하도록 CloudFront 배포를 성공적으로 구성했음을 나타냅니다. 이러한 제한으로 인해 CloudFront 배포를 통해서만 버킷의 객체에 액세스할 수 있습니다.

5단계: CloudFront 배포를 통해 비디오에 액세스

이제 CloudFront는 S3 버킷에 저장된 비디오를 제공할 수 있습니다. CloudFront를 통해 비디오에 액세스하려면 CloudFront 배포 도메인 이름을 S3 버킷의 비디오 경로와 결합해야 합니다.

CloudFront 배포 도메인 이름을 사용하여 S3 비디오에 대한 URL을 생성하려면

1. AWS Management Console에 로그인한 다음 https://console.aws.amazon.com/cloudfront/v4/home에서 CloudFront 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 배포를 선택합니다.
3. 배포 도메인 이름을 가져오려면 다음을 수행합니다.
   a. 원본 열에서, 1단계에서 생성한 S3 버킷으로 시작되는 원본 이름을 찾아 올바른 CloudFront 배포를 찾습니다(예: **tutorial-bucket**).
   b. 목록에서 배포를 찾은 후 도메인 이름 열을 확장하여 CloudFront 배포의 도메인 이름 값을 복사합니다.
4. 새 브라우저 탭에서, 복사한 배포 도메인 이름을 붙여 넣습니다.
5. 이전 브라우저 탭으로 돌아가 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
6. 왼쪽 탐색 창에서 버킷을 선택합니다.
7. 버킷(Buckets) 목록에서, 1단계에서 생성한 버킷의 이름을 선택합니다(예: **tutorial-bucket**).
8. 객체 목록에서, 2단계에서 업로드한 비디오의 이름을 선택합니다(예: `sample.mp4`).
9. 객체 세부 정보 페이지의 객체 개요 섹션에서 키 값을 복사합니다. 이 값은 S3 버킷에서 업로드된 비디오 객체로의 경로입니다.
10. 이전에 배포 도메인 이름을 붙여 넣은 브라우저 탭으로 돌아가서 배포 도메인 이름 뒤에 사선(**/**)을 입력한 다음, 앞서 복사한 비디오로의 경로를 붙여 넣습니다(예: `sample.mp4`).

    이제 S3 비디오가 다음과 유사한 URL에서 CloudFront를 통해 공개적으로 액세스되고 호스팅됩니다.

    ```
    https://CloudFront distribution domain name/Path to the S3 video
    ```

    CloudFront ## ### ## 및 *S3 ##### ##*를 적절한 값으로 바꿉니다. 예제 URL은 **https://d111111abcdef8.cloudfront.net/sample.mp4**입니다.

6단계: 사용자 지정 도메인 이름을 사용하도록 CloudFront 배포 구성

URL에서 CloudFront 도메인 이름 대신에 고유의 도메인 이름을 사용하여 S3 비디오에 액세스하려면, 대체 도메인 이름을 CloudFront 배포에 추가합니다.

하위 단계

- SSL 인증서 요청

- CloudFront 배포에 대체 도메인 이름을 추가합니다.
- 대체 도메인 이름에서 트래픽을 CloudFront 배포의 도메인 이름으로 라우팅하는 DNS 레코드를 생성합니다.
- 배포에 대해 IPv6이 활성화되어 있는지 확인하고, 필요한 경우 다른 DNS 레코드를 만듭니다.

SSL 인증서 요청

뷰어가 동영상 비디오 스트리밍용 URL에 HTTPS 및 사용자 지정 도메인 이름을 사용할 수 있도록 하려면 AWS Certificate Manager(ACM)을(를) 사용하여 보안 소켓 계층(SSL) 인증서를 요청합니다. SSL 인증서는 웹 사이트에 대한 암호화된 네트워크 연결을 설정합니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/acm/에서 ACM 콘솔을 엽니다.
2. 소개 페이지가 나타나면 인증서 프로비저닝(Provision certificates)에서 시작하기(Get Started)를 선택합니다.
3. 인증서 요청 페이지에서 퍼블릭 인증서 요청을 선택한 후 인증서 요청을 선택합니다.
4. 도메인 이름 추가 페이지에서 SSL/TLS 인증서로 보안을 설정할 사이트의 정규화된 도메인 이름(FQDN)을 입력합니다. 별표(*)를 사용하여 같은 도메인 내의 여러 사이트를 보호하는 와일드카드 인증서를 요청할 수 있습니다. 이 자습서에서는 **`*`** 및 사전 조건에서 구성한 사용자 지정 도메인 이름을 입력합니다. 이 예제의 경우 **`*.example.com`**을 입력한 다음 다음을 선택합니다.

   자세한 내용은 AWS Certificate Manager 사용 설명서의 ACM 퍼블릭 인증서를 요청하려면(콘솔)을 참조하십시오.
5. 검증 방법 선택(Select validation method) 페이지에서 DNS 검증(DNS validation)을 선택합니다. 그리고 다음을 선택합니다.

   사용자 DNS 환경 설정을 편집할 수 있다면, 이메일 검증보다는 DNS 검증을 사용하는 것을 권장합니다. DNS 검증은 이메일 검증에 비해 다양한 이점이 있습니다. 자세한 내용은 AWS Certificate Manager 사용 설명서의 옵션 1: DNS 유효성 검사 단원을 참조하십시오.
6. (선택 사항) 태그 추가 페이지에서 인증서에 메타데이터로 태그를 지정할 수 있습니다.
7. 검토를 선택합니다.
8. 검토 페이지에서 도메인 이름 및 메서드 검증의 정보가 올바른지 확인합니다. 그런 다음 확인 및 요청(Confirm and request)을 선택합니다.

   검증 페이지에서는 요청이 처리 중이며 인증서 도메인이 검증되고 있음을 보여줍니다. 검증 대기 중인 인증서는 검증 보류(Pending validation) 상태입니다.

9. 검증 페이지에서 사용자 지정 도메인 이름의 왼쪽에 있는 아래쪽 화살표를 선택한 후 Route 53에서 레코드 생성을 선택하여 DNS를 통해 도메인 소유권을 검증합니다.

   그러면 AWS Certificate Manager에서 제공하는 CNAME 레코드가 DNS 구성에 추가됩니다.

10. Route 53에서 레코드 생성(Create record in Route 53) 대화 상자에서 생성(Create)을 선택합니다.

    검증 페이지의 하단에 성공이라는 상태 알림이 표시되어야 합니다.

11. 계속(Continue)을 선택하여 인증서(Certificates) 목록 페이지를 확인합니다.

    새 인증서에 대한 상태가 30분 이내에 검증 보류에서 발급 완료로 변경됩니다.

CloudFront 배포에 대체 도메인 이름을 추가합니다.

1. AWS Management Console에 로그인한 다음 https://console.aws.amazon.com/cloudfront/v4/home에서 CloudFront 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 배포를 선택합니다.
3. 4단계에서 생성한 배포의 ID를 선택합니다.
4. 일반탭에서 설정섹션으로 이동하고 편집을 선택합니다.
5. 설정 편집 페이지에서 대체 도메인 이름(CNAME) - 선택 사항의 경우, 항목 추가를 선택하여 이 CloudFront 배포에서 제공하는 S3 비디오의 URL에 사용하려는 사용자 지정 도메인 이름을 추가합니다.

   이 자습서에서는 예를 들어 하위 도메인(예: `www.example.com`)에 대한 트래픽을 라우팅하려는 경우 도메인 이름(`example.com`)과 함께 하위 도메인 이름(`www`)을 입력합니다. 구체적으로 **`www.example.com`**을 입력합니다.

   > **Note**
   >
   > 추가하는 대체 도메인 이름(CNAME)은 이전에 CloudFront 배포에 연결한 SSL 인증서의 적용을 받아야 합니다.

6. 사용자 지정 SSL 인증서 - 선택 사항의 경우 이전에 요청한 SSL 인증서를 선택합니다(예: **`*.example.com`**).

> **Note**
>
> SSL 인증서를 요청한 직후에 인증서가 표시되지 않으면 인증서를 선택할 수 있을 때까지 30분간 기다린 다음 목록을 새로 고칩니다.

7. 나머지 설정은 기본값으로 유지합니다. Save changes(변경 사항 저장)를 선택합니다.
8. 해당 배포에 대한 일반 탭에서, 마지막 수정 날짜 값이 배포 중에서 배포가 마지막으로 수정된 타임스탬프로 변경될 때까지 기다립니다.

대체 도메인 이름에서 트래픽을 CloudFront 배포의 도메인 이름으로 라우팅하는 DNS 레코드를 생성합니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/route53/에서 Route 53 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Hosted Zones(호스팅 영역)를 선택합니다.
3. 호스팅 영역 페이지에서, 사전 조건에서 Route 53가 생성한 호스팅 영역의 이름을 선택합니다(예: **`example.com`**).
4. 레코드 생성을 선택한 다음 빠른 레코드 생성 메서드를 사용합니다.
5. 레코드 이름의 경우 레코드 이름의 값을 이전에 추가한 CloudFront 배포의 대체 도메인 이름과 동일하게 유지합니다.

   이 자습서에서는 하위 도메인(예: `www.example.com`)에 대한 트래픽을 라우팅하기 위해 도메인 이름 없이 하위 도메인 이름을 입력합니다. 예를 들어, 사용자 지정 도메인 이름 앞의 텍스트 필드에 **`www`**를 입력합니다.
6. 레코드 유형(Record type)에서 A - IPv4 주소 및 일부 AWS 리소스로 트래픽 라우팅(A - Routes traffic to an IPv4 address and some AWS resource)을 선택합니다.
7. 값에서 별칭 리소스를 활성화하기 위해 별칭 토글을 선택합니다.
8. 다음으로 트래픽 라우팅의 드롭다운 목록에서 CloudFront 배포에 대한 별칭을 선택합니다.
9. 배포 선택이라는 검색 상자에서, 4단계에서 생성한 CloudFront 배포의 도메인 이름을 선택합니다.

   CloudFront 배포의 도메인 이름을 찾으려면 다음을 수행합니다.

   a. 새 브라우저 탭에서 AWS Management Console에 로그인한 후 https://console.aws.amazon.com/cloudfront/v3/home에서 CloudFront 콘솔을 엽니다.
   b. 왼쪽 탐색 창에서 배포를 선택합니다.

c. 원본 열에서, 1단계에서 생성한 S3 버킷으로 시작되는 원본 이름을 찾아 올바른 CloudFront 배포를 찾습니다(예: **`tutorial-bucket`**).

d. 목록에서 배포를 찾은 후 도메인 이름 열을 확장하여 CloudFront 배포의 도메인 이름 값을 확인합니다.

10. Route 53 콘솔의 레코드 생성 페이지에서 나머지 설정은 기본값을 유지합니다.

11. 레코드 생성을 선택합니다.

배포에 대해 IPv6이 활성화되어 있는지 확인하고, 필요한 경우 다른 DNS 레코드를 만듭니다.

배포에 대해 IPv6이 활성화되어 있다면 다른 DNS 레코드를 만들어야 합니다.

1. 배포에 대해 IPv6이 활성화되어 있는지 확인하려면 다음을 수행합니다.

   a. AWS Management Console에 로그인한 다음 https://console.aws.amazon.com/cloudfront/v4/home에서 CloudFront 콘솔을 엽니다.

   b. 왼쪽 탐색 창에서 배포를 선택합니다.

   c. 4단계에서 생성한 CloudFront 배포의 ID를 선택합니다.

   d. 일반 탭의 설정에서, IPv6이 활성화됨으로 설정되어 있는지 확인합니다.

   배포에 대해 IPv6이 활성화되어 있다면 다른 DNS 레코드를 만들어야 합니다.

2. 배포에 대해 IPv6이 활성화되어 있다면 다음을 수행하여 다른 DNS 레코드를 만들어야 합니다.

   a. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/route53/에서 Route 53 콘솔을 엽니다.

   b. 왼쪽 탐색 창에서 Hosted Zones(호스팅 영역)를 선택합니다.

   c. 호스팅 영역 페이지에서, 사전 조건에서 Route 53가 생성한 호스팅 영역의 이름을 선택합니다(예: **`example.com`**).

   d. 레코드 생성을 선택한 다음 빠른 레코드 생성 메서드를 사용합니다.

   e. 레코드 이름의 경우, 사용자 지정 도메인 이름 앞의 텍스트 필드에서 이전에 IPv4 DNS 레코드를 생성했을 때 입력한 것과 동일한 값을 입력합니다. 예를 들어 이 자습서에서는 하위 도메인 `www.example.com`의 트래픽을 라우팅하기 위해 **`www`**만 입력합니다.

   f. 레코드 유형(Record type)에서 AAAA - IPv6 주소 및 일부 AWS 리소스로 트래픽 라우팅(AAAA - Routes traffic to an IPv6 address and some AWS resource)을 선택합니다.

   g. 값에서 별칭 리소스를 활성화하기 위해 별칭 토글을 선택합니다.

   h. 다음으로 트래픽 라우팅의 드롭다운 목록에서 CloudFront 배포에 대한 별칭을 선택합니다.

i. 배포 선택이라는 검색 상자에서, 4단계에서 생성한 CloudFront 배포의 도메인 이름을 선택합니다.

j. 나머지 설정은 기본값으로 유지합니다.

k. 레코드 생성을 선택합니다.

7단계: 사용자 지정 도메인 이름을 사용하여 CloudFront 배포를 통해 S3 비디오에 액세스

사용자 지정 URL을 사용하여 S3 비디오에 액세스하려면 대체 도메인 이름을 S3 버킷의 비디오 경로와 결합해야 합니다.

CloudFront 배포를 통해 S3 비디오에 액세스하기 위한 사용자 지정 URL을 생성하려면

1. AWS Management Console에 로그인한 다음 https://console.aws.amazon.com/cloudfront/v4/home에서 CloudFront 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 배포를 선택합니다.
3. CloudFront 배포의 대체 도메인 이름을 가져오려면 다음을 수행합니다.

   a. 원본 열에서, 1단계에서 생성한 버킷의 S3 버킷 이름으로 시작되는 원본 이름을 찾아 CloudFront 배포를 찾고 수정합니다(예: **`tutorial-bucket`**).

   b. 목록에서 배포를 찾은 후 대체 도메인 이름 열을 확장하여 CloudFront 배포의 대체 도메인 이름 값을 복사합니다.
4. 새 브라우저 탭에서, CloudFront 배포의 대체 도메인 이름을 붙여 넣습니다.
5. 이전 브라우저 탭으로 돌아가 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
6. 5단계에서 설명하는 S3 비디오의 경로를 찾습니다.
7. 이전에 대체 도메인 이름을 붙여 넣은 브라우저 탭으로 돌아가서 사선(**`/`**)를 입력하고 S3 비디오의 경로를 붙여 넣습니다(예: `sample.mp4`).

   이제 S3 비디오가 다음과 유사한 사용자 지정 URL에서 CloudFront를 통해 공개적으로 액세스되고 호스팅됩니다.

   ```
   https://CloudFront distribution alternate domain name/Path to the S3 video
   ```

   `CloudFront ## ## ### ##` 및 *`S3 ### ##`*를 적절한 값으로 바꿉니다. 예제 URL은 **`https://www.example.com/sample.mp4`**입니다.

8단계: CloudFront 배포에서 수신한 요청에 대한 데이터 보기(선택 사항)

8단계: CloudFront 배포에서 수신한 요청에 대한 데이터를 보려면

1. AWS Management Console에 로그인한 다음 https://console.aws.amazon.com/cloudfront/v4/home에서 CloudFront 콘솔을 엽니다.
2. 왼쪽 탐색 창의 보고서 및 분석에서, 캐시 통계, 인기 객체, 상위 참조자, 사용량 및 뷰어의 범위에서 콘솔의 보고서를 선택합니다.

   각 보고서 대시보드를 필터링할 수 있습니다. 자세한 내용은 Amazon CloudFront 개발자 안내서에서 콘솔의 CloudFront 보고서를 참조하십시오.
3. 데이터를 필터링하려면 4단계에서 생성한 CloudFront 배포의 ID를 선택합니다.

9단계: 정리

CloudFront 및 Route 53을 학습 연습으로만 사용하여 S3 스트리밍 비디오를 호스팅한 경우 더 이상 요금이 발생하지 않도록 할당한 AWS 리소스를 삭제합니다.

> **Note**
>
> 도메인을 등록하면 즉시 비용이 청구되며, 이는 취소할 수 없습니다. 도메인을 자동 갱신하지 않도록 선택할 수 있지만, 선불로 비용을 지불하며 1년 동안 소유합니다. 자세한 내용은 Amazon Route 53 개발자 안내서의 새 도메인 등록을 참조하십시오.

하위 단계

- CloudFront 배포 삭제
- DNS 레코드 삭제
- 사용자 지정 도메인의 퍼블릭 호스팅 영역 삭제
- Route 53에서 사용자 지정 도메인 이름 삭제
- S3 소스 버킷의 원본 비디오 삭제
- S3 소스 버킷 삭제

CloudFront 배포 삭제

1. AWS Management Console에 로그인한 다음 https://console.aws.amazon.com/cloudfront/v4/home에서 CloudFront 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 배포를 선택합니다.
3. 원본 열에서, 1단계에서 생성한 버킷의 S3 버킷 이름으로 시작되는 원본 이름을 찾아 CloudFront 배포를 찾고 수정합니다(예: **`tutorial-bucket`**).
4. CloudFront 배포를 삭제하려면 먼저 배포를 비활성화해야 합니다.
   - 상태 열 값이 활성화됨이고 마지막 수정 날짜 값이 배포가 마지막으로 수정되었을 때의 타임스탬프이면 삭제하기 전에 배포를 비활성화합니다.
   - 상태 값이 활성화됨이고 마지막 수정 날짜 값이 배포 중이면 상태 값이 배포가 마지막으로 수정되었을 때의 타임스탬프로 변경될 때까지 기다립니다. 그런 다음 삭제하기 전에 배포를 비활성화합니다.
5. CloudFront 배포를 비활성화하려면 다음을 수행합니다.
   a. 배포 목록에서 삭제할 배포에 대한 ID 옆의 확인란을 선택합니다.
   b. 배포를 비활성화하려면 비활성화를 선택한 후 비활성화를 선택하여 확인합니다.

      대체 도메인 이름이 연결된 배포를 비활성화하면, 동일한 도메인과 일치하는 와일드카드(*)가 있는 대체 도메인 이름(예: `*.example.com`)이 다른 배포에 있더라도 CloudFront가 해당 도메인 이름(예: `www.example.com`)에 대한 트래픽 수신을 중지합니다.
   c. 상태(Status) 값이 비활성화됨(Disabled)으로 즉시 변경됩니다. 마지막 수정 날짜 값이 배포 중에서 배포가 마지막으로 수정된 타임스탬프로 변경될 때까지 기다립니다.

      CloudFront는 이 변경 사항을 모든 엣지 로케이션에 전파해야 하므로, 업데이트가 완료되어 배포를 삭제할 수 있는 삭제(Delete) 옵션이 사용 가능해질 때까지 몇 분이 걸릴 수 있습니다.
6. 비활성화된 배포를 삭제하려면 다음을 수행합니다.
   a. 삭제하려는 배포의 ID 옆 확인란을 선택합니다.
   b. 삭제를 선택한 후 삭제를 선택하여 확인합니다.

DNS 레코드 삭제

도메인의 퍼블릭 호스팅 영역(DNS 레코드 포함)을 삭제하려면 Amazon Route 53 개발자 안내서의 사용자 지정 도메인의 퍼블릭 호스팅 영역 삭제를 참조하십시오. 6단계에서 생성한 DNS 레코드만 삭제하려면 다음을 수행합니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/route53/에서 Route 53 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Hosted Zones(호스팅 영역)를 선택합니다.
3. 호스팅 영역 페이지에서, 사전 조건에서 Route 53이 생성한 호스팅 영역의 이름을 선택합니다(예: **example.com**).
4. 레코드 목록에서, 삭제하려는 레코드(6단계에서 생성한 레코드) 옆의 확인란을 선택합니다.

   > **Note**
   >
   > 유형 값이 NS 또는 SOA인 레코드는 삭제할 수 없습니다.

5. 레코드 삭제(Delete record)를 선택합니다.
6. 삭제를 확인하려면 삭제를 선택합니다.

   레코드 변경 내용이 Route 53 DNS 서버로 전파되려면 시간이 걸립니다. 현재 변경 사항의 전파 여부를 확인하는 유일한 방법은 GetChange API 작업을 사용하는 것입니다. 변경 사항은 일반적으로 60초 이내에 모든 Route 53 이름 서버로 전파됩니다.

사용자 지정 도메인의 퍼블릭 호스팅 영역 삭제

> **Warning**
>
> 도메인 등록을 유지하면서 웹 사이트 또는 웹 애플리케이션으로 인터넷 트래픽이 라우팅되는 것을 중지하려면, 호스팅 영역을 삭제하는 대신 호스팅 영역에서 레코드를 삭제하는 것이 좋습니다(이전 섹션에 설명됨).
> 호스팅 영역을 삭제하면 다른 사람이 이 도메인을 사용할 수 있으며 사용자의 도메인 이름을 사용하여 트래픽을 그들의 리소스로 라우팅할 수 있습니다.
> 또한 호스팅 영역을 삭제하면 삭제를 취소할 수 없습니다. 새 호스팅 영역을 만들고 도메인 등록을 위한 이름 서버를 업데이트해야 합니다. 도메인 등록은 효력이 발생하려면 최대 48시간이 걸립니다.
> 도메인을 인터넷에서 사용 불가능하게 만들려면 먼저 DNS 서비스를 무료 DNS 서비스로 이전한 후 Route 53 호스팅 영역을 삭제합니다. 이렇게 하면 이후의 DNS 쿼리가 잘못 라우팅되지 않도록 할 수 있습니다.
>
> 1. 도메인이 Route 53에 등록되어 있는 경우 Route 53 이름 서버를 새로운 DNS 서비스의 이름 서버로 바꾸는 방법은 Amazon Route 53 개발자 안내서의 도메인의 이름 서버 및 글루 레코드 추가 또는 변경을 참조하십시오.

2. 도메인이 다른 등록 대행자에 등록되어 있는 경우 등록 대행자가 제공한 방법을 사용하여 도메인에 대한 이름 서버를 변경하십시오.

> **Note**
>
> 하위 도메인(`www.example.com`)의 호스팅 영역을 삭제하는 경우에는 도메인(`example.com`)의 이름 서버를 변경할 필요가 없습니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/route53/에서 Route 53 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Hosted Zones(호스팅 영역)를 선택합니다.
3. 호스팅 영역(Hosted zones) 페이지에서, 삭제할 호스팅 영역의 이름을 선택합니다.
4. 호스팅 영역의 레코드(Records) 탭에서, 삭제할 호스팅 영역에 NS 및 SOA레코드만 포함되어 있는지 확인합니다.

   추가 레코드가 있는 경우 먼저 삭제합니다.

   호스팅 영역의 하위 도메인에 대한 NS 레코드를 생성한 경우, 해당 레코드 역시 삭제합니다.
5. 호스팅 영역의 DNSSEC 서명(DNSSEC signing) 탭에서 DNNSSEC 서명을 활성화한 경우 비활성화합니다. 자세한 내용은 Amazon Route 53 개발자 안내서의 DNSSEC 서명 비활성화를 참조하십시오.
6. 호스팅 영역의 세부 정보 페이지 상단에서 영역 삭제를 선택합니다.
7. 삭제를 확인하려면 **`delete`**을(를) 입력한 후 삭제를 선택합니다.

Route 53에서 사용자 지정 도메인 이름 삭제

TLD(최상위 도메인)에 대해 등록을 더 이상 원하지 않는 경우 등록을 삭제할 수 있습니다. 등록이 만료되기 전에 Route 53에서 도메인 이름 등록을 삭제할 경우 AWS에서는 등록 요금을 환불하지 않습니다. 자세한 내용은 Amazon Route 53 개발자 가이드의 도메인 이름 등록 삭제를 참조하십시오.

> **Important**
>
> AWS 계정 간에 도메인을 이전하거나 도메인을 다른 등록 대행사로 이전하려는 경우 도메인을 삭제해서 즉시 재등록하려고 하지 마십시오. 대신 Amazon Route 53 개발자 가이드의 해당 문서를 참조하십시오.
>
> - 도메인을 다른 AWS 계정로 이전하기
> - Amazon Route 53에서 다른 등록 기관으로 도메인 이전하기

S3 소스 버킷의 원본 비디오 삭제

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 이름 목록에서, 2단계에서 동영상을 업로드한 버킷의 이름을 선택합니다(예: **`tutorial-bucket`**).
4. 객체 탭에서 삭제하려는 객체의 이름 옆에 있는 확인란을 선택합니다(예: `sample.mp4`).
5. Delete(삭제)를 선택합니다.
6. 객체를 영구적으로 삭제하시겠습니까?에서 **`permanently delete`**를 입력하여 객체를 삭제할 것인지 확인합니다.
7. 객체 삭제를 선택합니다.

S3 소스 버킷 삭제

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.
3. 버킷 목록에서, 1단계에서 생성한 버킷의 이름 옆에 있는 옵션 버튼을 선택합니다(예: **`tutorial-bucket`**).
4. Delete(삭제)를 선택합니다.
5. 버킷 삭제(Delete bucket) 페이지의 텍스트 필드에 버킷 이름을 입력하여 버킷의 삭제 여부를 확인한 다음 버킷 삭제(Delete bucket)를 선택합니다.

다음 단계

이 자습서를 완료한 후에는 다음과 같은 관련 사용 사례를 더 자세히 탐색할 수 있습니다.

- CloudFront 배포로 S3 비디오를 호스팅하기 전에 특정 TV 또는 커넥티드 디바이스에 필요한 스트리밍 형식으로 트랜스코딩합니다.

  Amazon S3 배치 작업, AWS Lambda 및 AWS Elemental MediaConvert를 사용하여 비디오 모음을 다양한 출력 미디어 형식으로 일괄 트랜스코딩하려면 자습서: S3 Batch Operations를 통해 비디오 일괄 트랜스코딩 섹션을 참조하십시오.
- CloudFront 및 Route 53을 사용하여 이미지, 오디오, 모션 그래픽, 스타일시트, HTML, JavaScript, React 앱 등 S3에 저장된 다른 객체를 호스팅합니다.

  예제는 자습서: Route 53에 등록된 사용자 지정 도메인을 사용하여 정적 웹 사이트 구성 및 Amazon CloudFront로 웹사이트 속도 향상 섹션을 참조하십시오.
- Amazon S3 Transfer Acceleration을 사용하여 빠르고 안전한 파일 전송을 구성할 수 있습니다. Transfer Acceleration을 사용하면 S3 버킷에 비디오를 빠르게 업로드하여 대용량의 비디오를 장거리 전송할 수 있습니다. Transfer Acceleration은 CloudFront의 전역으로 배포된 엣지 로케이션 및 AWS 백본 네트워크를 통해 트래픽을 라우팅하여 전송 성능을 향상시킵니다. 또한 네트워크 프로토콜 최적화를 활용합니다. 자세한 내용은 Amazon S3 Transfer Acceleration을 사용하여 빠르고 안전한 파일 전송 구성 단원을 참조하십시오.

자습서: Amazon S3에서 정적 웹 사이트 구성

> **Important**
>
> 이제 Amazon S3가 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)를 Amazon S3 내 모든 버킷 암호화의 기본 수준으로 적용합니다. 2023년 1월 5일부터 Amazon S3로의 모든 새 객체 업로드는 추가 비용 없이 성능에 영향을 미치지 않고 자동으로 암호화됩니다. S3 버킷 기본 암호화 구성에 및 신규 객체 업로드에 대한 자동 암호화 상태는 AWS CloudTrail 로그, S3 인벤토리, S3 스토리지 렌즈, Amazon S3 콘솔에서 사용할 수 있으며, AWS Command Line Interface 및 AWS SDK에서 추가 Amazon S3 API 응답 헤더로도 사용할 수 있습니다. 자세한 내용은 기본 암호화 관련 FAQ를 참조하십시오.

웹 사이트처럼 작동하도록 Amazon S3 버킷을 구성할 수 있습니다. 이 예제에서는 Amazon S3에서 웹 사이트를 호스팅하는 절차를 단계별로 살펴봅니다.

> **Important**
>
> 다음 자습서에서는 퍼블릭 액세스 차단을 비활성화해야 합니다. 퍼블릭 액세스 차단 설정을 활성화 상태로 유지하는 것이 좋습니다. 네 개의 퍼블릭 액세스 차단 설정을 모두 활성화하여 정적 웹 사이트를 호스팅하려는 경우 Amazon CloudFront 원본 액세스 제어(OAC)를 사용할 수 있습니다. Amazon CloudFront는 안전한 정적 웹 사이트를 설정하는 데 필요한 기능을 제공합니다. Amazon S3 정적 웹 사이트는 HTTP 엔드포인트만 지원합니다. Amazon CloudFront는 Amazon S3의 내구성 있는 스토리지를 사용하면서 HTTPS와 같은 추가 보안 헤더를 제공합니다. HTTPS는 일반적인 HTTP 요청을 암호화하고 일반적인 사이버 공격으로부터 보호함으로써 보안을 강화합니다. 자세한 내용은 Amazon CloudFront 개발자 안내서의 안전한 정적 웹 사이트 시작하기를 참조하십시오.

주제

- 1단계: 버킷 만들기
- 2단계: 정적 웹 사이트 호스팅 활성화
- 3단계: 퍼블릭 액세스 차단 설정 편집
- 4단계: 버킷 콘텐츠를 공개적으로 사용 가능하도록 설정하는 버킷 정책 추가
- 5단계: 인덱스 문서 구성
- 6단계: 오류 문서 구성
- 7단계: 웹 사이트 엔드포인트 테스트
- 8단계: 정리

1단계: 버킷 만들기

아래 지침에서는 웹 사이트 호스팅용 버킷을 생성하는 방법에 대한 개요를 제공합니다. 버킷 생성에 대한 자세한 단계별 지침은 범용 버킷 생성 섹션을 참조하세요.

버킷을 만들려면

1. AWS Management Console에 로그인한 후 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.

지리적으로 가까운 리전을 선택하면 지연 시간과 요금을 최소화하고, 규제 요건을 해결할 수 있습니다. 선택한 리전에 따라 Amazon S3 웹 사이트 엔드포인트가 결정됩니다. 자세한 내용은 웹 사이트 엔드포인트 섹션을 참조하세요.

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 Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 정적 웹 사이트 호스팅을 사용 설정하려는 버킷의 이름을 선택합니다.
4. [속성(Properties)]을 선택합니다.
5. 정적 웹 사이트 호스팅(Static website hosting)에서 편집(Edit)을 선택합니다.
6. 이 버킷을 사용하여 웹 사이트를 호스팅합니다.를 선택합니다.
7. 정적 웹 사이트 호스팅에서 사용을 선택합니다.
8. 인덱스 문서(Index document)에 인덱스 문서 이름을 입력합니다(일반적으로 `index.html`).

   인덱스 문서 이름은 대소문자를 구분하며 S3 버킷에 업로드하려는 HTML 인덱스 문서의 파일 이름과 정확히 일치해야 합니다. 웹 사이트 호스팅용 버킷을 구성하는 경우 인덱스 문서를 지정해야 합니다. 루트 도메인이나 임의의 하위 폴더로 요청이 전송되면 Amazon S3가 이 인덱스 문서를 반환합니다. 자세한 내용은 인덱스 문서 구성 섹션을 참조하세요.
9. 4XX 클래스 오류에 대한 사용자 지정 오류 문서를 제공하려면 [오류 문서(Error document)]에 사용자 지정 오류 문서 파일 이름을 입력합니다.

   오류 문서 이름은 대소문자를 구분하며 S3 버킷에 업로드하려는 HTML 오류 문서의 파일 이름과 정확히 일치해야 합니다. 사용자 지정 오류 문서를 지정하지 않았는데 오류가 발생하면 Amazon S3에서 기본 HTML 오류 문서를 반환합니다. 자세한 내용은 사용자 지정 오류 문서 구성 단원을 참조하십시오.
10. (선택 사항) 고급 리디렉션 규칙을 지정하려면 리디렉션 규칙(Redirection rules)에 JSON을 입력하여 규칙을 설명합니다.

예를 들어, 요청의 특정 객체 키 이름 또는 접두사에 따라 조건부로 요청을 라우팅할 수 있습니다. 자세한 내용은 고급 조건부 리디렉션을 사용하도록 리디렉션 규칙 구성 섹션을 참조하세요.

11. [변경 사항 저장(Save changes)]을 선택합니다.

    Amazon S3는 버킷에 대한 정적 웹 사이트 호스팅을 지원합니다. 페이지 하단의 정적 웹 사이트 호스팅(Static website hosting)에 버킷의 웹 사이트 엔드포인트가 표시됩니다.

12. 정적 웹 사이트 호스팅에서 엔드포인트를 기록합니다.

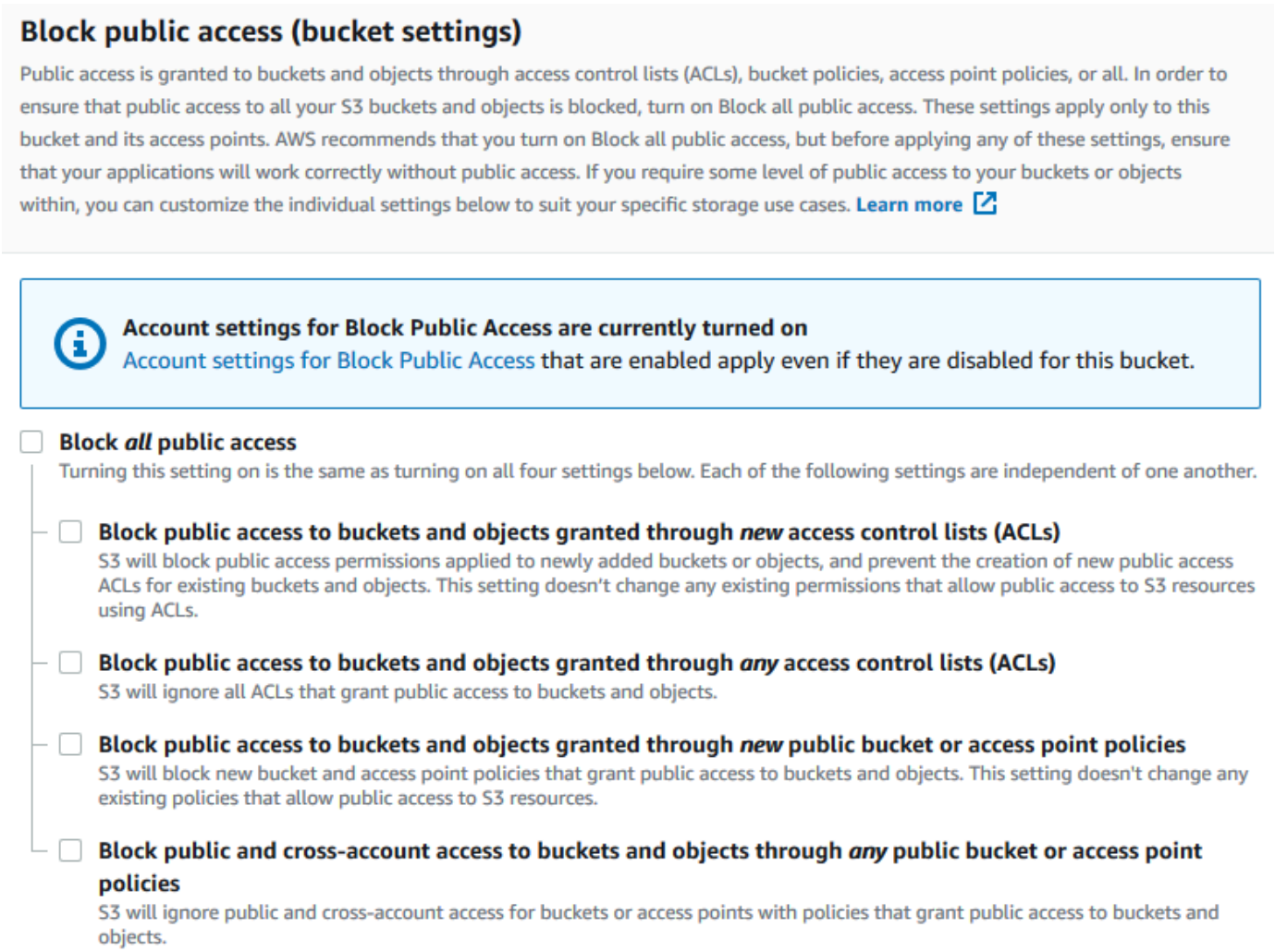
    엔드포인트는 버킷의 Amazon S3 웹 사이트 엔드포인트입니다. 버킷을 정적 웹 사이트로 구성한 후 이 엔드포인트를 사용하여 웹 사이트를 테스트할 수 있습니다.

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**
>
> 이러한 단계를 완료하기 전에 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단을 검토하여 퍼블릭 액세스 허용과 관련된 위험을 이해하고 이에 동의하는지 확인합니다. 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. 퍼블릭 액세스 차단(버킷 설정)(Block public access (bucket settings))에서 편집(Edit)을 선택합니다.
5. 모든 퍼블릭 액세스 차단(Block all public access)을 선택 취소하고 변경 사항 저장(Save changes)을 선택합니다.



Amazon S3은 버킷에 대한 퍼블릭 액세스 차단 설정을 끕니다. 정적 퍼블릭 웹 사이트를 생성하려면 버킷 정책을 추가하기 전에 계정에 대한 퍼블릭 액세스 차단 설정을 편집해야 할 수도 있습니다. 계정에 대한 퍼블릭 액세스 차단 설정이 현재 켜져 있는 경우 퍼블릭 액세스 차단(버킷 설정) 아래에 메모가 표시됩니다.

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 버킷에 있는 파일을 보호하려면 어떻게 해야 하나요?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. 버킷 정책(Bucket Policy)에서 편집(Edit)을 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:::Bucket-Name/*"
            ]
        }
    ]
}
```

5. `Resource`를 버킷 이름으로 업데이트합니다.

   앞의 버킷 정책 예제에서 *`Bucket-Name`*은 버킷 이름의 자리 표시자입니다. 자체 버킷에 이 버킷 정책을 사용하려면 자체 버킷 이름과 일치하도록 이 이름을 업데이트해야 합니다.

6. [변경 사항 저장(Save changes)]을 선택합니다.

   버킷 정책이 성공적으로 추가되었음을 나타내는 메시지가 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 하나요?를 참조하십시오.

오류 메시지가 나타나고 버킷 정책을 저장할 수 없는 경우 계정 및 버킷의 퍼블릭 액세스 차단 설정에서 버킷에 대한 퍼블릭 액세스를 허용하는지 확인합니다.

5단계: 인덱스 문서 구성

버킷용 정적 웹 사이트 호스팅을 활성화할 때 인덱스 문서의 이름(예: **`index.html`**)을 입력합니다. 버킷용 정적 웹 사이트 호스팅을 사용 설정한 후 인덱스 문서 이름이 있는 HTML 파일을 버킷에 업로드합니다.

인덱스 문서 구성

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 저장합니다.

   인덱스 문서 파일 이름은 정적 웹 사이트 호스팅 대화 상자에 입력한 인덱스 문서 이름과 정확히 일치해야 합니다. 인덱스 문서 이름은 대/소문자를 구분합니다. 예를 들어 정적 웹 사이트 호스팅 대화 상자에서 인덱스 문서 이름에 `index.html`을 입력하는 경우, 인덱스 문서 파일은 `index.html`이 아니라 `Index.html`이어야 합니다.

3. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
4. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
5. 버킷 목록에서 정적 웹 사이트를 호스팅하는 데 사용할 버킷의 이름을 선택합니다.

6. 버킷에 정적 웹 사이트 호스팅을 사용 설정하고 인덱스 문서의 정확한 이름(예: `index.html`)을 입력합니다. 자세한 내용은 웹 사이트 호스팅 사용 설정 섹션을 참조하세요.

   정적 웹 사이트 호스팅을 사용 설정한 후 6단계로 이동합니다.
7. 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 객체 업로드 섹션을 참조하세요.
8. (선택 사항) 버킷에 다른 웹 사이트 콘텐츠를 업로드합니다.

6단계: 오류 문서 구성

버킷용 정적 웹 사이트 호스팅을 활성화할 때 오류 문서의 이름(예: **`404.html`**)을 입력합니다. 버킷용 정적 웹 사이트 호스팅을 사용 설정한 후 오류 문서 이름이 있는 HTML 파일을 버킷에 업로드합니다.

오류 문서 구성

1. 오류 문서를 생성합니다(예: `404.html`).
2. 오류 문서 파일을 로컬에 저장합니다.

   오류 문서 이름은 대/소문자를 구분하며 정적 웹 사이트 호스팅을 사용하도록 설정할 때 입력한 이름과 정확히 일치해야 합니다. 예를 들어 정적 웹 사이트 호스팅 대화 상자에서 오류 문서 이름에 `404.html`을 입력하는 경우, 오류 문서 파일은 `404.html`이어야 합니다.
3. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
4. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
5. 버킷 목록에서 정적 웹 사이트를 호스팅하는 데 사용할 버킷의 이름을 선택합니다.
6. 버킷에 정적 웹 사이트 호스팅을 사용 설정하고 오류 문서의 정확한 이름(예: `404.html`)을 입력합니다. 자세한 내용은 웹 사이트 호스팅 사용 설정 및 사용자 지정 오류 문서 구성 단원을 참조하세요.

   정적 웹 사이트 호스팅을 사용 설정한 후 6단계로 이동합니다.
7. 버킷에 오류 문서를 업로드하려면 다음 중 하나를 수행합니다.

   - 오류 문서 파일을 콘솔 버킷 목록으로 끌어다 놓습니다.

- 업로드를 선택하고 프롬프트의 메시지에 따라 인덱스 파일을 선택하고 업로드합니다.

  단계별 지침은 객체 업로드 섹션을 참조하세요.

7단계: 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 CloudFront를 사용하여 Amazon S3에 호스팅된 정적 웹 사이트를 제공하려면 어떻게 해야 하나요? 및 최종 사용자와 CloudFront 간의 통신에 HTTPS 요구를 참조하십시오.

1. 버킷에서 버킷의 이름을 선택합니다.
2. [속성(Properties)]을 선택합니다.
3. 페이지 하단의 정적 웹 사이트 호스팅(Static website hosting)에서 버킷 웹 사이트 엔드포인트(Bucket website endpoint)를 선택합니다.

   인덱스 문서가 별도의 브라우저 창에서 열립니다.

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. 관련 예제는 자습서: Route 53에 등록된 사용자 지정 도메인을 사용하여 정적 웹 사이트 구성 섹션을 참조하세요

8단계: 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 범용 버킷 삭제 단원을 참조하십시오.

자습서: 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` 및 `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 호스팅할 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 CloudFront로 웹사이트 속도 향상 섹션을 참조하세요.

> **Note**
>
> Amazon S3 웹 사이트 엔드포인트는 HTTPS 또는 액세스 포인트를 지원하지 않습니다. HTTPS를 사용하려는 경우 Amazon CloudFront를 사용하여 Amazon S3에서 호스팅되는 정적 웹 사이트를 제공할 수 있습니다.
>
> CloudFront 및 Amazon S3를 사용하여 콘텐츠를 안전하게 호스팅하는 방법에 대한 자세한 내용은 자습서: Amazon S3, Amazon CloudFront 및 Amazon Route 53로 온디맨드 스트리밍 비디오 호스팅 섹션을 참조하세요. 자세한 내용은 CloudFront를 사용하여 Amazon S3에 호스팅된 정적 웹 사이트를 제공하려면 어떻게 해야 하나요? 및 최종 사용자와 CloudFront 간의 통신에 HTTPS 요구를 참조하세요.

AWS CloudFormation 템플릿으로 정적 웹 사이트 설정 자동화

AWS CloudFormation 템플릿을 사용하여 정적 웹 사이트 설정을 자동화할 수 있습니다. AWS CloudFormation 템플릿은 안전한 정적 웹 사이트를 호스팅하는 데 필요한 구성 요소를 설정하므로 구성 요소의 구성에 대한 비중은 낮추고 웹 사이트의 콘텐츠에 더 중점을 둘 수 있습니다.

AWS CloudFormation 템플릿에는 다음 구성 요소가 포함됩니다.

- Amazon S3 - 정적 웹 사이트를 호스팅할 Amazon S3 버킷을 생성합니다.
- CloudFront - 정적 웹 사이트의 속도를 높이기 위해 CloudFront 배포를 생성합니다.
- Lambda@Edge - Lambda@Edge를 사용하여 모든 서버 응답에 보안 헤더를 추가합니다. 보안 헤더는 웹 서버 응답에서 웹 브라우저에 추가 보안 예방 조치를 취하도록 알려주는 헤더 그룹입니다. 자

세한 내용은 Lambda@Edge 및 Amazon CloudFront를 사용하여 HTTP 보안 헤더 추가 블로그 게시물을 참조하십시오.

이 AWS CloudFormation 템플릿은 다운로드하여 사용할 수 있습니다. 자세한 내용과 지침은 Amazon CloudFront 개발자 안내서의 안전한 정적 웹 사이트 시작하기를 참조하십시오.

주제

- 시작하기 전에
- 1단계: Route 53에 사용자 지정 도메인 등록
- 2단계: 두 개의 버킷 생성
- 3단계: 웹 사이트 호스팅용 루트 도메인 버킷 구성
- 4단계: 웹 사이트 리디렉션용 하위 도메인 버킷 구성
- 5단계: 웹 사이트 트래픽용 로깅 구성
- 6단계: 인덱스 및 웹 사이트 콘텐츠 업로드
- 7단계: 오류 문서 업로드
- 8단계: S3 퍼블릭 액세스 차단 설정 편집
- 9단계: 버킷 정책 연결
- 10단계: 도메인 엔드포인트 테스트
- 11단계: 도메인 및 하위 도메인에 대한 별칭 레코드 추가
- 12단계: 웹 사이트 테스트
- Amazon CloudFront로 웹사이트 속도 향상
- 예제 리소스 정리

시작하기 전에

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 S3을 사용하여 버킷을 생성하고 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 등록을 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 생성합니다.

- 도메인 버킷 - `example.com`
- 하위 도메인 버킷 - `www.example.com`

이러한 버킷 이름은 도메인 이름과 정확히 일치해야 합니다. 이 예제에서 도메인 이름은 `example.com`입니다. 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 버킷 생성

아래 지침에서는 웹 사이트 호스팅용 버킷을 생성하는 방법에 대한 개요를 제공합니다. 버킷 생성에 대한 자세한 단계별 지침은 범용 버킷 생성 섹션을 참조하십시오.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 루트 도메인 버킷을 생성합니다.

   a. 페이지 상단의 탐색 모음에서 현재 표시된 AWS 리전의 이름을 선택합니다. 그런 다음 버킷을 생성하려는 리전을 선택합니다.

   > **Note**
   >
   > 지연 시간과 요금을 최소화하고 규제 요건을 충족하려면 가장 가까운 리전을 선택하십시오. 특정 리전에 저장된 객체는 사용자가 명시적으로 객체를 다른 리전으로 전송하지 않는 한 해당 리전을 벗어나지 않습니다. Amazon S3 AWS 리전 목록은 Amazon Web Services 일반 참조의 AWS 서비스 엔드포인트를 참조하십시오.

   b. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.

c. 버킷 만들기를 선택합니다. 버킷 만들기 페이지가 열립니다.

d. 버킷 이름(예: **example.com**)을 입력합니다.

e. 버킷을 생성하려는 리전을 선택합니다.

지리적으로 가까운 리전을 선택하면 지연 시간과 요금을 최소화하고, 규제 요건을 해결할 수 있습니다. 선택한 리전에 따라 Amazon S3 웹 사이트 엔드포인트가 결정됩니다. 자세한 내용은 웹 사이트 엔드포인트 섹션을 참조하세요.

f. 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.

지리적으로 가까운 리전을 선택하면 지연 시간과 요금을 최소화하고, 규제 요건을 해결할 수 있습니다. 선택한 리전에 따라 Amazon S3 웹 사이트 엔드포인트가 결정됩니다. 자세한 내용은 웹 사이트 엔드포인트 섹션을 참조하세요.

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 엔드포인트를 사용하여 웹 사이트에 액세스할 수 있습니다.

정적 웹 사이트 호스팅 사용 설정

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
3. 버킷 목록에서 정적 웹 사이트 호스팅을 사용 설정하려는 버킷의 이름을 선택합니다.
4. [속성(Properties)]을 선택합니다.
5. 정적 웹 사이트 호스팅(Static website hosting)에서 편집(Edit)을 선택합니다.
6. 이 버킷을 사용하여 웹 사이트를 호스팅합니다.를 선택합니다.

7. 정적 웹 사이트 호스팅에서 사용을 선택합니다.
8. 인덱스 문서(Index document)에 인덱스 문서 이름을 입력합니다(일반적으로 `index.html`).

   인덱스 문서 이름은 대소문자를 구분하며 S3 버킷에 업로드하려는 HTML 인덱스 문서의 파일 이름과 정확히 일치해야 합니다. 웹 사이트 호스팅용 버킷을 구성하는 경우 인덱스 문서를 지정해야 합니다. 루트 도메인이나 임의의 하위 폴더로 요청이 전송되면 Amazon S3가 이 인덱스 문서를 반환합니다. 자세한 내용은 인덱스 문서 구성 섹션을 참조하세요.
9. 4XX 클래스 오류에 대한 사용자 지정 오류 문서를 제공하려면 [오류 문서(Error document)]에 사용자 지정 오류 문서 파일 이름을 입력합니다.

   오류 문서 이름은 대소문자를 구분하며 S3 버킷에 업로드하려는 HTML 오류 문서의 파일 이름과 정확히 일치해야 합니다. 사용자 지정 오류 문서를 지정하지 않았는데 오류가 발생하면 Amazon S3에서 기본 HTML 오류 문서를 반환합니다. 자세한 내용은 사용자 지정 오류 문서 구성 단원을 참조하십시오.
10. (선택 사항) 고급 리디렉션 규칙을 지정하려면 리디렉션 규칙(Redirection rules)에 JSON을 입력하여 규칙을 설명합니다.

    예를 들어, 요청의 특정 객체 키 이름 또는 접두사에 따라 조건부로 요청을 라우팅할 수 있습니다. 자세한 내용은 고급 조건부 리디렉션을 사용하도록 리디렉션 규칙 구성 섹션을 참조하세요.
11. [변경 사항 저장(Save changes)]을 선택합니다.

    Amazon S3는 버킷에 대한 정적 웹 사이트 호스팅을 지원합니다. 페이지 하단의 정적 웹 사이트 호스팅(Static website hosting)에 버킷의 웹 사이트 엔드포인트가 표시됩니다.
12. 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.

퍼블릭 액세스 차단 설정을 편집하고 퍼블릭 읽기 액세스를 허용하는 버킷 정책을 추가한 후 웹 사이트 엔드포인트를 사용하여 웹 사이트에 액세스할 수 있습니다.

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 요청 구성

1. Amazon S3 콘솔의 범용 버킷 목록에서 하위 도메인 버킷 이름(이 예시에서는 `www.example.com`)을 선택합니다.
2. [속성(Properties)]을 선택합니다.
3. 정적 웹 사이트 호스팅(Static website hosting)에서 편집(Edit)을 선택합니다.
4. 객체에 대한 요청 리디렉션(Redirect requests for an object)을 선택합니다.
5. 대상 버킷(Target bucket) 상자에 루트 도메인(예: **`example.com`**)을 입력합니다.
6. 프로토콜(Protocol)에서 http를 선택합니다.
7. [변경 사항 저장(Save changes)]을 선택합니다.

5단계: 웹 사이트 트래픽용 로깅 구성

웹 사이트에 접속하는 방문자들의 수를 추적하려는 경우 필요에 따라 루트 도메인 버킷에 대한 로깅을 사용 설정할 수 있습니다. 자세한 내용은 서버 액세스 로깅을 사용한 요청 로깅 섹션을 참조하세요. Amazon CloudFront를 사용하여 웹 사이트 속도를 높이려는 경우 CloudFront 로깅을 사용할 수도 있습니다.

루트 도메인 버킷에 대한 서버 액세스 로깅 사용 설정

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.

> **Important**
>
> 배포를 생성 또는 업데이트하고 CloudFront 로깅을 사용 설정하면 CloudFront는 버킷 액세스 제어 목록(ACL)을 업데이트하여 버킷에 로그를 쓸 수 있는 `FULL_CONTROL` 권한을

awslogsdelivery 계정에 부여합니다. 자세한 내용은 Amazon CloudFront 개발자 안내서의 표준 로깅 구성 및 로그 파일 액세스에 필요한 권한을 참조하세요. 로그를 저장하는 버킷이 S3 객체 소유권에 대해 버킷 소유자 적용 설정을 사용하여 ACL을 비활성화하면 CloudFront에서 버킷에 로그를 쓸 수 없습니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 단원을 참조하십시오.

5. 버킷(Buckets) 목록에서 루트 도메인 버킷을 선택합니다.
6. [속성(Properties)]을 선택합니다.
7. 서버 액세스 로깅(Server access logging)에서 편집(Edit)을 선택합니다.
8. 사용을 선택합니다.
9. 대상 버킷(Target bucket)에서 서버 액세스 로그의 버킷과 폴더 대상을 선택합니다.

   - 폴더 및 버킷 위치를 찾습니다.
     1. S3 찾아보기(Browse S3)를 선택합니다.
     2. 버킷 이름을 선택한 다음 로그 폴더를 선택합니다.
     3. 경로 선택(Choose path)을 선택합니다.
   - S3 버킷 경로(예: s3://logs.example.com/logs/)를 입력합니다.
10. [변경 사항 저장(Save changes)]을 선택합니다.

    이제 로그 버킷에서 로그에 액세스할 수 있습니다. Amazon S3은 2시간마다 웹 사이트 액세스 로그를 로그 버킷에 기록합니다.

6단계: 인덱스 및 웹 사이트 콘텐츠 업로드

이 단계에서 인덱스 문서와 선택적 웹 사이트 콘텐츠를 루트 도메인 버킷에 업로드합니다.

버킷용 정적 웹 사이트 호스팅을 사용 설정할 때 인덱스 문서의 이름(예: **index.html**)을 입력합니다. 버킷용 정적 웹 사이트 호스팅을 사용 설정한 후 인덱스 문서 이름이 있는 HTML 파일을 버킷에 업로드합니다.

인덱스 문서 구성

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 저장합니다.

   인덱스 문서 파일 이름은 정적 웹 사이트 호스팅 대화 상자에 입력한 인덱스 문서 이름과 정확히 일치해야 합니다. 인덱스 문서 이름은 대/소문자를 구분합니다. 예를 들어 정적 웹 사이트 호스팅 대화 상자에서 인덱스 문서 이름에 `index.html`을 입력하는 경우, 인덱스 문서 파일은 `index.html`이 아니라 `Index.html`이어야 합니다.

3. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
4. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
5. 버킷 목록에서 정적 웹 사이트를 호스팅하는 데 사용할 버킷의 이름을 선택합니다.
6. 버킷에 정적 웹 사이트 호스팅을 사용 설정하고 인덱스 문서의 정확한 이름(예: `index.html`)을 입력합니다. 자세한 내용은 웹 사이트 호스팅 사용 설정 섹션을 참조하세요.

   정적 웹 사이트 호스팅을 사용 설정한 후 6단계로 이동합니다.

7. 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 객체 업로드 섹션을 참조하세요.

8. (선택 사항) 버킷에 다른 웹 사이트 콘텐츠를 업로드합니다.

7단계: 오류 문서 업로드

버킷용 정적 웹 사이트 호스팅을 사용 설정할 때 오류 문서의 이름(예: **`404.html`**)을 입력합니다. 버킷용 정적 웹 사이트 호스팅을 사용 설정한 후 오류 문서 이름이 있는 HTML 파일을 버킷에 업로드합니다.

오류 문서 구성

1. 오류 문서를 생성합니다(예: `404.html`).
2. 오류 문서 파일을 로컬에 저장합니다.

   오류 문서 이름은 대/소문자를 구분하며 정적 웹 사이트 호스팅을 사용하도록 설정할 때 입력한 이름과 정확히 일치해야 합니다. 예를 들어 정적 웹 사이트 호스팅 대화 상자에서 오류 문서 이름에 `404.html`을 입력하는 경우, 오류 문서 파일은 `404.html`이어야 합니다.
3. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
4. 왼쪽 탐색 창에서 범용 버킷을 선택합니다.
5. 버킷 목록에서 정적 웹 사이트를 호스팅하는 데 사용할 버킷의 이름을 선택합니다.
6. 버킷에 정적 웹 사이트 호스팅을 사용 설정하고 오류 문서의 정확한 이름(예: `404.html`)을 입력합니다. 자세한 내용은 웹 사이트 호스팅 사용 설정 및 사용자 지정 오류 문서 구성 단원을 참조하세요.

   정적 웹 사이트 호스팅을 사용 설정한 후 6단계로 이동합니다.
7. 버킷에 오류 문서를 업로드하려면 다음 중 하나를 수행합니다.

   - 오류 문서 파일을 콘솔 버킷 목록으로 끌어다 놓습니다.
   - 업로드를 선택하고 프롬프트의 메시지에 따라 인덱스 파일을 선택하고 업로드합니다.

   단계별 지침은 객체 업로드 섹션을 참조하세요.

8단계: S3 퍼블릭 액세스 차단 설정 편집

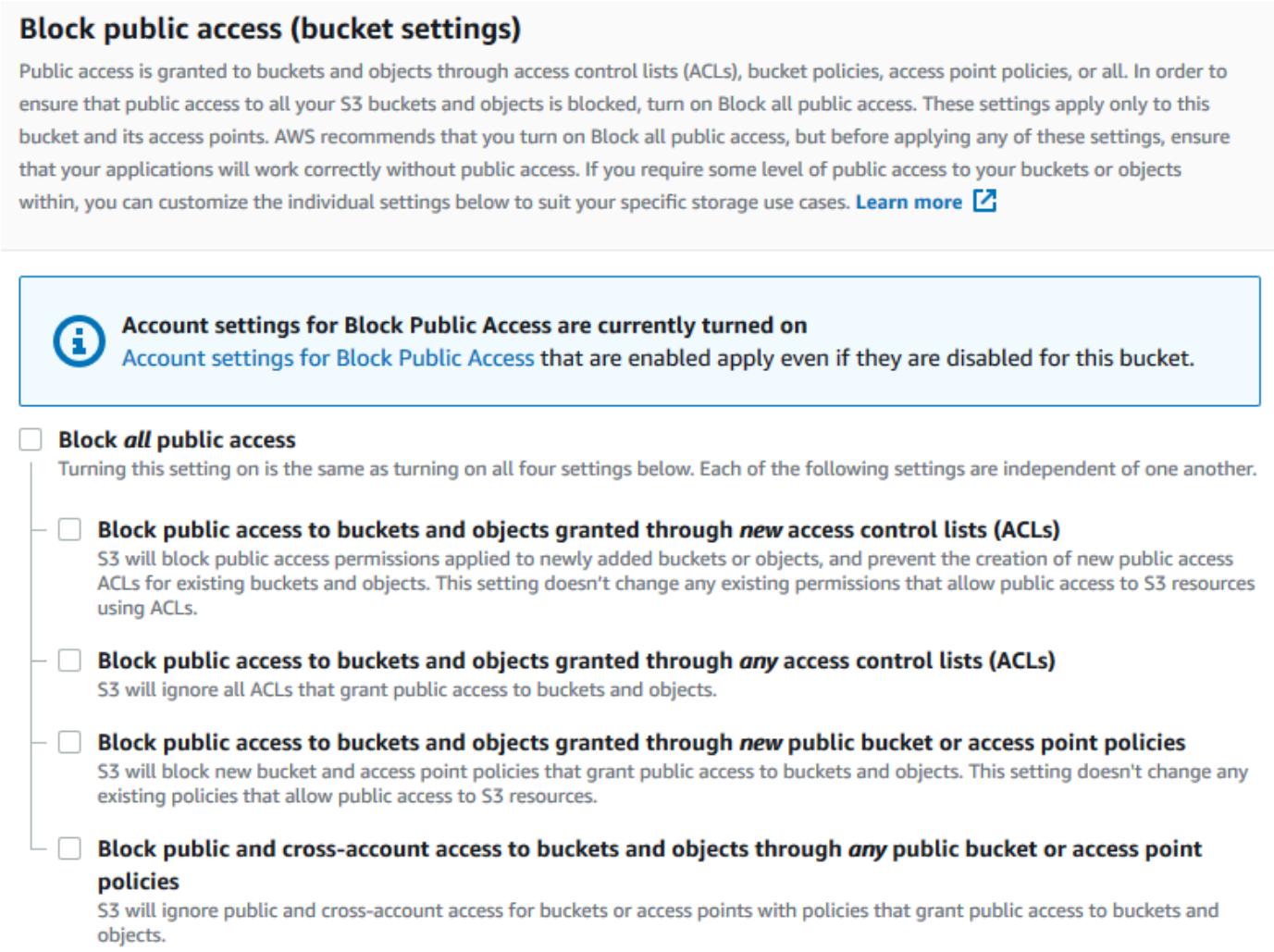
이 예제에서는 퍼블릭 액세스를 허용하도록 도메인 버킷(`example.com`)에 대한 퍼블릭 액세스 차단 설정을 편집합니다.

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**
>
> 이러한 단계를 완료하기 전에 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단을 검토하여 퍼블릭 액세스 허용과 관련된 위험을 이해하고 이에 동의하는지 확인합니다. 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. 퍼블릭 액세스 차단(버킷 설정)(Block public access (bucket settings))에서 편집(Edit)을 선택합니다.
5. 모든 퍼블릭 액세스 차단(Block all public access)을 선택 취소하고 변경 사항 저장(Save changes)을 선택합니다.



Amazon S3은 버킷에 대한 퍼블릭 액세스 차단 설정을 끕니다. 정적 퍼블릭 웹 사이트를 생성하려면 버킷 정책을 추가하기 전에 계정에 대한 퍼블릭 액세스 차단 설정을 편집해야 할 수도 있습니

다. 계정에 대한 퍼블릭 액세스 차단 설정이 현재 켜져 있는 경우 퍼블릭 액세스 차단(버킷 설정) 아래에 메모가 표시됩니다.

9단계: 버킷 정책 연결

이 예제에서는 버킷 정책을 도메인 버킷(`example.com`)에 연결하여 퍼블릭 읽기 액세스를 허용합니다. 버킷 정책의 *`Bucket-Name`*을 도메인 버킷의 이름으로 바꿉니다(예: `example.com`).

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 버킷에 있는 파일을 보호하려면 어떻게 해야 하나요?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. 버킷 정책(Bucket Policy)에서 편집(Edit)을 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:::Bucket-Name/*"
```

```
            ]
        }
    ]
}
```

5. `Resource`를 버킷 이름으로 업데이트합니다.

   앞의 버킷 정책 예제에서 *`Bucket-Name`*은 버킷 이름의 자리 표시자입니다. 자체 버킷에 이 버킷 정책을 사용하려면 자체 버킷 이름과 일치하도록 이 이름을 업데이트해야 합니다.

6. [변경 사항 저장(Save changes)]을 선택합니다.

   버킷 정책이 성공적으로 추가되었음을 나타내는 메시지가 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 하나요?를 참조하십시오.

   오류 메시지가 나타나고 버킷 정책을 저장할 수 없는 경우 계정 및 버킷의 퍼블릭 액세스 차단 설정에서 버킷에 대한 퍼블릭 액세스를 허용하는지 확인합니다.

다음 단계에서는 웹 사이트 엔드포인트를 파악하고 도메인 엔드포인트를 테스트할 수 있습니다.

10단계: 도메인 엔드포인트 테스트

퍼블릭 웹 사이트를 호스팅하도록 도메인 버킷을 구성한 후 엔드포인트를 테스트할 수 있습니다. 자세한 내용은 웹 사이트 엔드포인트 섹션을 참조하세요. 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 CloudFront를 사용하여 Amazon S3에 호스팅된 정적 웹 사이트를 제공하려면 어떻게 해야 하나요? 및 최종 사용자와 CloudFront 간의 통신에 HTTPS 요구를 참조하십시오.

1. 버킷에서 버킷의 이름을 선택합니다.
2. [속성(Properties)]을 선택합니다.

3. 페이지 하단의 정적 웹 사이트 호스팅(Static website hosting)에서 버킷 웹 사이트 엔드포인트(Bucket website endpoint)를 선택합니다.

   인덱스 문서가 별도의 브라우저 창에서 열립니다.

다음 단계에서는 Amazon Route 53을 이용해 고객이 사용자 지정 URL을 모두 사용하여 해당 사이트를 탐색할 수 있도록 합니다.

11단계: 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 및 하위 도메인에 대한 별칭 레코드 추가

루트 도메인(**`example.com`**)에 대한 별칭 레코드 추가

1. https://console.aws.amazon.com/route53/에서 Route 53 콘솔을 엽니다.

   > **Note**
   >
   > Route 53을 아직 사용하지 않은 경우 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(레코드 세트 생성)를 선택합니다.
5. 마법사로 전환을 선택합니다.

   > **Note**
   >
   > 빠른 생성을 사용하여 별칭 레코드를 생성하려면 트래픽을 S3 버킷으로 라우팅하도록 Route 53 구성을 참조하십시오.

6. Simple routing(단순 라우팅)을 선택하고 다음을 선택합니다.
7. Define simple record(단순 레코드 정의)를 선택합니다.

8. [레코드 이름(Record name)]에서 기본값(호스팅 영역과 도메인의 이름)을 그대로 사용합니다.
9. 값/트래픽 라우팅 대상에서 Alias to S3 website endpoint(S3 웹 사이트 엔드포인트에 대한 별칭)를 선택합니다.
10. 리전을 선택합니다.
11. S3 버킷을 선택합니다.

    버킷 이름은 이름 상자에 나타나는 이름과 일치해야 합니다. S3 버킷 선택 목록에서 버킷 이름은 버킷이 생성된 리전의 Amazon S3 웹 사이트 엔드포인트와 함께 나타납니다(예: `s3-website-us-west-1.amazonaws.com (example.com)`).

    다음과 같은 경우 S3 버킷 선택이 버킷을 나열합니다.

    - 버킷을 정적 웹 사이트로 구성한 경우
    - 버킷 이름이 생성 중인 레코드의 이름과 동일한 경우
    - 현재 AWS 계정에서 버킷을 생성한 경우.

    버킷이 S3 버킷 선택 목록에 나타나지 않으면 버킷이 생성된 리전의 Amazon S3 웹 사이트 엔드포인트를 입력합니다(예: **`s3-website-us-west-2.amazonaws.com`**). Amazon S3 웹 사이트 엔드포인트의 전체 목록은 Amazon S3 웹 사이트 엔드포인트를 참조하세요. 별칭 대상에 대한 자세한 내용은 Amazon Route 53 개발자 안내서의 값/트래픽 라우팅 대상을 참조하세요.
12. 레코드 유형(Record type)에서 A - IPv4 주소 및 일부 AWS 리소스로 트래픽 라우팅(A - Routes traffic to an IPv4 address and some AWS resources)을 선택합니다.
13. 대상 상태 평가에서 아니요를 선택합니다.
14. Define simple record(단순 레코드 정의)를 선택합니다.

하위 도메인(**`www.example.com`**)에 별칭 레코드 추가

1. 레코드 구성에서 단순 레코드 정의를 선택합니다.
2. 하위 도메인의 레코드 이름에 www를 입력합니다.
3. 값/트래픽 라우팅 대상에서 Alias to S3 website endpoint(S3 웹 사이트 엔드포인트에 대한 별칭)를 선택합니다.
4. 리전을 선택합니다.
5. S3 버킷을 선택합니다(예: `s3-website-us-west-2.amazonaws.com (www.example.com)`).

버킷이 S3 버킷 선택 목록에 나타나지 않으면 버킷이 생성된 리전의 Amazon S3 웹 사이트 엔드포인트를 입력합니다(예: **s3-website-us-west-2.amazonaws.com**). Amazon S3 웹 사이트 엔드포인트의 전체 목록은 Amazon S3 웹 사이트 엔드포인트를 참조하세요. 별칭 대상에 대한 자세한 내용은 Amazon Route 53 개발자 안내서의 값/트래픽 라우팅 대상을 참조하세요.

6. 레코드 유형(Record type)에서 A - IPv4 주소 및 일부 AWS 리소스로 트래픽 라우팅(A - Routes traffic to an IPv4 address and some AWS resources)을 선택합니다.
7. 대상 상태 평가에서 아니요를 선택합니다.
8. Define simple record(단순 레코드 정의)를 선택합니다.
9. 레코드 구성 페이지에서 레코드 생성을 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.

루트 도메인 및 하위 도메인에 대한 별칭 레코드 추가(이전 Route 53 콘솔)

루트 도메인(**example.com**)에 대한 별칭 레코드 추가

Route 53 콘솔이 새롭게 디자인되었습니다. Route 53 콘솔에서 일시적으로 이전 콘솔을 사용할 수 있습니다. 이전 Route 53 콘솔로 작업하도록 선택한 경우 아래 절차를 따릅니다.

1. https://console.aws.amazon.com/route53/에서 Route 53 콘솔을 엽니다.

   > **Note**
   >
   > Route 53을 아직 사용하지 않은 경우 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 주소(A – IPv4 address)를 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. 별칭 대상(Alias Target) 목록에서 버킷 이름 뒤에는 버킷이 생성된 리전의 Amazon S3 웹 사이트 엔드포인트가 옵니다(예: `example.com (s3-website-us-west-2.amazonaws.com)`). 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 [별칭 대상(Alias Target)] 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 값/트래픽 라우팅 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 레코드 추가

1. 루트 도메인(example.com)에 대한 호스팅 영역에서 레코드 세트 생성을 선택합니다.
2. 다음 값을 지정합니다.

   이름

   하위 도메인에 대해 www를 상자에 입력합니다.

   유형

   A - IPv4 주소(A – IPv4 address)를 선택합니다.

   별칭

   예를 선택합니다.

   별칭 대상

   목록의 S3 웹 사이트 엔드포인트(S3 website endpoints) 섹션에서 이름(Name) 필드에 표시되는 동일한 버킷 이름을 선택합니다(예: www.example.com (s3-website-us-west-2.amazonaws.com)).

   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.

12단계: 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.com으로 리디렉션합니다. example.com 버킷의 인덱스 문서가 표시됩니다.

웹 사이트 또는 리디렉션 링크가 작동하지 않으면 다음 방법을 시도할 수 있습니다.

- 캐시 지우기 - 웹 브라우저의 캐시를 지웁니다.
- 이름 서버 확인 - 캐시를 지운 후에도 웹 페이지 및 리디렉션 링크가 작동하지 않으면 도메인의 이름 서버와 호스팅 영역의 이름 서버를 비교합니다. 이름 서버가 일치하지 않는 경우 호스팅 영역 아래에 나열된 이름과 일치하도록 도메인 이름 서버를 업데이트해야 할 수 있습니다. 자세한 내용은 도메인의 글루 레코드 및 이름 서버 추가 또는 변경을 참조하십시오.

루트 도메인과 하위 도메인을 성공적으로 테스트한 후에는 Amazon CloudFront 배포를 설정하여 웹 사이트의 성능을 개선하고 웹 사이트 트래픽을 검토하는 데 사용할 수 있는 로그를 제공할 수 있습니다. 자세한 내용은 Amazon CloudFront로 웹사이트 속도 향상 단원을 참조하십시오.

Amazon CloudFront로 웹사이트 속도 향상

Amazon CloudFront를 사용하여 Amazon S3 웹 사이트의 성능을 향상시킬 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.

Route 53 없이 CloudFront 사용

이 페이지의 자습서에서는 Route 53을 사용하여 CloudFront 배포를 가리킵니다. 그러나 Route 53을 사용하지 않고 CloudFront를 사용하여 Amazon S3 버킷에 호스팅된 콘텐츠를 제공하려면 Amazon CloudFront 자습서, Amazon S3용 동적 콘텐츠 배포 설정을 참조하십시오. CloudFront를 사용하여 Amazon S3 버킷에 호스팅된 콘텐츠를 서비스할 때는 어떤 버킷 이름이라도 사용할 수 있으며 HTTP와 HTTPS가 모두 지원됩니다.

AWS CloudFormation 템플릿으로 설정 자동화

AWS CloudFormation 템플릿을 사용하여 웹 사이트에 제공할 CloudFront 배포를 생성하는 안전한 정적 웹 사이트를 구성하는 방법에 대한 자세한 내용은 Amazon CloudFront 개발자 안내서의 안전한 정적 웹 사이트 시작하기를 참조하십시오.

주제

- 1단계: CloudFront 배포 생성
- 2단계: 도메인 및 하위 도메인용 레코드 세트 업데이트
- (선택 사항) 3단계: 로그 파일 확인

1단계: CloudFront 배포 생성

먼저 CloudFront 배포를 생성합니다. 이를 통해 전 세계 각지의 데이터 센터에서 웹 사이트를 사용할 수 있게 됩니다.

Amazon S3 오리진으로 배포 생성

1. 에서 CloudFront 콘솔을 엽니다https://console.aws.amazon.com/cloudfront/v4/home
2. 배포 생성(Create Distribution)을 선택합니다.
3. [배포 생성(Create Distribution)] 페이지의 [오리진 설정(Origin Settings)] 섹션에서 [오리진 도메인 이름(Origin Domain Name)]에 해당 버킷에 대한Amazon S3 웹 사이트 엔드포인트를 입력합니다(예: **`example.com.s3-website.us-west-1.amazonaws.com`**).

   CloudFront가 사용자 대신 오리진 ID(Origin ID)를 채웁니다.
4. Default Cache Behavior Settings(기본 캐시 동작 설정)에서는 기본값으로 설정되어 있는 값을 유지합니다.

   [뷰어 프로토콜 정책(Viewer Protocol Policy)]의 기본 설정을 사용하면 정적 웹 사이트에 HTTPS를 사용할 수 있습니다. 이 구성 옵션에 대한 자세한 내용은 Amazon CloudFront 개발자 안내서의 웹 배포의 생성 또는 업데이트 시 지정하는 값을 참조하십시오.
5. 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. [대체 도메인 이름(CNAME)(Alternate Domain Names (CNAMEs))]을 루트 도메인과 `www` 하위 도메인으로 설정합니다. 이 자습서에서는 `example.com` 및 `www.example.com`입니다.

> **Important**
>
> 이 단계를 수행하기 전에 대체 도메인 이름 사용과 관련된 요구 사항, 특히 유효한 SSL/TLS 인증서가 필요한지 확인하십시오.

c. SSL 인증서(SSL Certificate)에서 사용자 정의 SSL 인증서(Custom SSL Certificate) (example.com)를 선택하고, 도메인 및 하위 도메인 이름을 포함하는 사용자 지정 인증서를 선택합니다.

   자세한 내용은 Amazon CloudFront 개발자 안내서의 SSL 인증서를 참조하십시오.

d. 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 단원을 참조하십시오.

e. 로깅을 설정으로 설정합니다.

> **Important**
>
> 배포를 생성 또는 업데이트하고 CloudFront 로깅을 사용 설정하면 CloudFront는 버킷 액세스 제어 목록(ACL)을 업데이트하여 버킷에 로그를 쓸 수 있는 `FULL_CONTROL` 권한을 `awslogsdelivery` 계정에 부여합니다. 자세한 내용은 Amazon CloudFront 개발자 안내서의 표준 로깅 구성 및 로그 파일 액세스에 필요한 권한을 참조하세요. 로그를 저장하는 버킷이 S3 객체 소유권에 대해 버킷 소유자 적용 설정을 사용하여 ACL을 비활성화하면 CloudFront에서 버킷에 로그를 쓸 수 없습니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지 단원을 참조하십시오.

f. Bucket for Logs(로그용 버킷)에서 앞서 생성한 로깅 버킷을 선택합니다.

   로깅 버킷 구성에 대한 자세한 내용은 (선택 사항) 웹 트래픽 로깅 단원을 참조하십시오.

g. CloudFront 배포에 대한 트래픽에 의해 생성된 로그를 폴더에 저장하려면 로그 접두사(Log Prefix)에 폴더 이름을 입력합니다.

h. 기타 모든 설정은 기본값을 유지합니다.

6. [Create Distribution]을 선택합니다.
7. 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.

8. CloudFront 콘솔에 표시되는 도메인 이름의 값을 기록합니다(예: `dj4p1rv6mvubz.cloudfront.net`).
9. CloudFront 배포가 이루어지고 있는지 확인하려면, 웹 브라우저에서 배포의 도메인 이름을 입력합니다.

   웹 사이트가 표시되는 경우 CloudFront 배포가 작동하는 것입니다. 웹 사이트의 Amazon Route 53에 사용자 지정 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 새 CloudFront 배포를 가리키도록 Route 53의 별칭 레코드를 업데이트합니다.

CloudFront 배포를 가리키도록 별칭 레코드 업데이트

1. https://console.aws.amazon.com/route53/에서 Route 53 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Hosted Zones(호스팅 영역)를 선택합니다.
3. Hosted Zones(호스팅 영역) 페이지에서 하위 도메인을 위해 생성한 호스팅 영역(예: `www.example.com`)을 선택합니다.
4. 레코드에서 하위 도메인에 대해 만든 A 레코드를 선택합니다.
5. 레코드 세부 정보에서 레코드 편집을 선택합니다.
6. 다음으로 트래픽 라우팅에서 CloudFront 배포로 별칭을 선택합니다.
7. 배포 선택에서 CloudFront 배포를 선택합니다.
8. 저장을 선택합니다.
9. 루트 도메인에 대한 A 레코드를 CloudFront 배포로 리디렉션하려면 루트 도메인에 대해 이 절차(예: `example.com`)를 반복합니다.

   레코드 세트를 업데이트한 내용은 2~48시간 내에 적용됩니다.
10. 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`을 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 사용 설정할 때 선택한 버킷과 폴더에 저장됩니다. CloudFront는 해당 요청 이후 24시간 이내에 로그 버킷에 로그를 기록합니다.

웹 사이트의 로그 파일 보기

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 CloudFront에서 작성한 `.gzip` 파일을 다운로드합니다.

실습용으로만 웹 사이트를 생성한 경우에는 요금이 발생하지 않도록 할당한 리소스를 삭제할 수 있습니다. 그렇게 하려면 예제 리소스 정리 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

- 2단계: Route 53 호스팅 영역 삭제
- 3단계: 로깅 사용 중지 및 S3 버킷 삭제

1단계: Amazon CloudFront 배포 삭제

Amazon CloudFront 배포를 삭제하려면 먼저 사용 중지해야 합니다. 사용 중지된 배포가 작동하지 않아 요금이 발생하지 않습니다. 언제든지 사용 중지된 배포를 사용 설정할 수 있습니다. 사용 중지된 배포는 삭제 후 사용할 수 없습니다.

CloudFront 배포를 사용하지 않도록 설정하고 삭제

1. 에서 CloudFront 콘솔을 엽니다https://console.aws.amazon.com/cloudfront/v4/home
2. 사용 중지하려는 배포를 선택한 후 사용 중지를 선택합니다.
3. 확인 메시지가 표시되면 예, 사용 중지를 선택합니다.
4. 사용 중지된 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 세트 삭제

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 영역 삭제

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 사용 중지 및 S3 버킷 삭제

S3 버킷을 삭제하기 전에 버킷에 대한 로깅이 사용 중지되어 있는지 확인하십시오. 그렇지 않으면 삭제해도 AWS이(가) 해당 버킷에 계속 로그를 기록합니다.

버킷에 대한 로깅 사용 중지

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. 사용(Enabled) 확인란의 선택을 취소합니다.
5. 저장을 선택합니다.

이제 버킷을 삭제할 수 있습니다. 자세한 내용은 범용 버킷 삭제 단원을 참조하십시오.

S3 범용 버킷에서 AWS Amplify Hosting에 정적 웹 사이트 배포

AWS Amplify Hosting을 사용하여 S3에 저장된 정적 웹 사이트 콘텐츠를 호스팅하는 것이 좋습니다. Amplify Hosting은 Amazon CloudFront로 구동되는 전 세계적으로 사용 가능한 콘텐츠 전송 네트워크(CDN)에 웹 사이트를 쉽게 배포할 수 있도록 하는 완전 관리형 서비스이므로 광범위한 설정을 하지 않아도 안전한 정적 웹 사이트 호스팅이 가능합니다. AWS Amplify Hosting을 사용하면 범용 버킷 내에서 객체의 위치를 선택하고, 관리형 CDN에 콘텐츠를 배포하고, 어디서나 웹 사이트에 액세스할 수 있는 퍼블릭 HTTPS URL을 생성할 수 있습니다. Amplify Hosting을 사용하여 정적 웹 사이트를 배포하면 다음과 같은 이점 및 기능을 확보할 수 있습니다.

- Amazon CloudFront로 구동되는 AWS 콘텐츠 전송 네트워크(CDN)에 배포 - CloudFront는 정적 및 동적 웹 콘텐츠를 사용자에게 배포하는 속도를 높이는 웹 서비스입니다. CloudFront는 엣지 로케이션이라고 하는 데이터 센터의 전 세계 네트워크를 통해 콘텐츠를 제공합니다. CloudFront를 통해 서비스하는 콘텐츠를 사용자가 요청하면 지연 시간이 가장 낮은 엣지 로케이션으로 요청이 라우팅되므로 가능한 최고의 성능, 향상된 신뢰성 및 가용성으로 콘텐츠가 제공됩니다. 자세한 내용은 Amazon CloudFront 개발자 안내서의 CloudFront에서 콘텐츠를 제공하는 방법을 참조하세요.
- HTTPS 지원 - 웹 사이트와 사용자의 웹 브라우저 간에 안전한 통신 및 데이터 전송을 제공합니다.

- 사용자 지정 도메인 - Amazon Route 53과 같은 도메인 등록 기관에서 구매한 사용자 지정 URL에 웹 사이트를 쉽게 연결합니다.
- 사용자 지정 SSL 인증서 - 사용자 지정 도메인을 설정할 때 Amplify가 제공하는 기본 관리형 인증서를 사용하거나 직접 선택한 타사 인증 기관에서 구매한 자체 사용자 지정 인증서를 사용할 수 있습니다.
- 기본 제공 지표 및 CloudWatch 모니터링 - 웹 사이트의 트래픽, 오류, 데이터 전송 및 지연 시간을 모니터링합니다.
- 암호 보호 - Amplify 콘솔에서 사용자 이름과 암호 요구 사항을 설정하여 웹 사이트에 대한 액세스를 제한합니다.
- 리디렉션 및 재작성 - Amplify 콘솔에서 리디렉션 및 재작성 규칙을 생성하여 웹 서버가 한 URL에서 다른 URL로 탐색 경로를 변경할 수 있도록 합니다.

Amazon S3 범용 버킷에서 Amplify Hosting으로 애플리케이션을 배포하면 AWS 요금은 Amplify의 요금 모델을 기반으로 합니다. 자세한 내용은 AWS Amplify요금을 참조하세요.

> **Important**
>
> Amazon S3를 사용할 수 있는 모든 AWS 리전에서 Amplify Hosting을 사용할 수 있는 것은 아닙니다. 정적 웹 사이트를 Amplify Hosting에 배포하려면 웹 사이트가 저장된 Amazon S3 범용 버킷이 Amplify를 사용할 수 있는 리전에 있어야 합니다. Amplify를 사용할 수 있는 리전 목록은 Amazon Web Services 일반 참조의 Amplify 엔드포인트를 참조하세요.

Amazon S3 콘솔, Amplify 콘솔, AWS CLI 또는 AWS SDK에서 배포 프로세스를 시작할 수 있습니다. 자신의 계정에 위치한 범용 버킷에서만 Amplify에 배포할 수 있습니다. Amplify는 교차 계정 버킷 액세스를 지원하지 않습니다.

다음 지침에 따라 Amazon S3 콘솔을 시작으로 Amazon S3 범용 버킷의 정적 웹 사이트를 Amplify Hosting에 배포합니다.

S3 콘솔에서 Amplify에 정적 웹 사이트 배포

Amazon S3 콘솔에서 정적 웹 사이트 배포

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 버킷(Buckets)을 선택합니다.

3. 버킷 목록에서 Amplify Hosting에 배포하려는 웹 사이트가 포함된 범용 버킷을 선택합니다.
4. 속성(Properties) 탭을 선택합니다.
5. 정적 웹 사이트 호스팅에서 Amplify 앱 생성을 선택합니다. 이 단계에서 배포 프로세스는 Amplify 콘솔로 이동합니다.
6. S3로 배포 페이지에서 다음 단계를 수행합니다.

   a. API 이름에 앱 또는 웹 사이트의 이름을 입력합니다.
   b. 브랜치 이름에 앱의 백엔드 이름을 입력합니다.
   c. 호스팅할 객체의 S3 위치에 범용 버킷의 디렉터리 경로를 입력하거나 S3 찾아보기를 선택한 다음, 해당 항목을 찾아서 선택합니다.
7. 저장 및 배포를 선택합니다.

> **Note**
>
> Amplify에서 호스팅되는 범용 버킷에서 정적 웹 사이트의 객체를 업데이트하는 경우 변경 사항이 적용되도록 애플리케이션을 Amplify Hosting에 재배포해야 합니다. Amplify Hosting은 버킷의 변경 사항을 자동으로 감지하지 않습니다. 자세한 내용은 AWS Amplify Hosting 사용 설명서의 S3 버킷에서 Amplify로 배포된 정적 웹 사이트 업데이트를 참조하세요.

Amplify 콘솔에서 직접 시작하려면 AWS Amplify Hosting 사용 설명서의 Amplify 콘솔을 사용하여 S3에서 정적 웹 사이트 배포를 참조하세요.

AWS SDK를 사용하려면 AWS Amplify Hosting 사용 설명서의 AWS SDK를 사용하여 S3에서 정적 웹 사이트를 배포하기 위한 버킷 정책 생성을 참조하세요.

할당량

AWS 계정에는 각 AWS 서비스에 대한 기본 할당량(이전에는 제한이라고 함)이 있습니다. Amazon S3 할당량에는 범용 버킷, 디렉터리 버킷, 액세스 포인트 등의 수가 포함됩니다. 일부 할당량의 경우 증가를 요청할 수 있지만, 모든 할당량을 늘릴 수 있지는 않습니다. 이 증가는 즉시 허용되지 않으므로 증가가 적용되려면 이틀 정도 걸릴 수 있습니다.

Amazon S3 할당량 및 기본값 목록은 AWS 일반 참조의 Amazon S3 할당량을 참조하세요.

할당량 증가

할당량 증가 요청

다음 옵션 중 하나를 사용하여 할당량 증가를 요청할 수 있습니다.

- AWS Management Console에서: Service Quotas 콘솔을 엽니다. 탐색 창에서 AWS 서비스를 선택합니다. Amazon S3와 할당량을 차례로 선택한 후 지침에 따라 할당량 증가를 요청합니다. 지침은 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하십시오.
- AWS CLI에서: request-service-quota-increase AWS CLI 명령을 사용합니다. 지침은 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하십시오.

레퍼런스

Amazon S3 콘솔 외에도 Amazon S3 REST API, AWS SDK 또는 AWS Command Line Interface(AWS CLI)를 사용하여 Amazon S3와 프로그래밍 방식으로 작업할 수 있습니다.

- Amazon Simple Storage Service API 참조
- Amazon Simple Storage Service API 참조의 코드 예제
- AWS Command Line Interface 사용 설명서의 AWS Command Line Interface
- AWS SDK

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

문서 기록

- 현재 API 버전: 2006-03-01

다음 표에서는 Amazon Simple Storage Service API 참조 및 Amazon S3 사용 설명서의 각 릴리스에서 변경된 중요한 사항에 대해 설명합니다. 이 설명서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

변경 사항	설명	날짜
이제 Amazon S3 Express One Zone에서 비용 할당 및 속성 기반 액세스 제어를 위한 태그 지원	고성능 S3 스토리 클래스인 Amazon S3 Express One Zone에서 이제 비용 할당 및 속성 기반 액세스 제어(ABAC)를 위한 태그를 지원합니다. AWS 결제 및 비용 관리를 사용하여 S3 디렉터리 버킷에 태그를 추가하여 AWS 비용을 추적하고 정리할 수 있습니다. 또한 ABAC 지원을 통해 태그를 사용하여 디렉터리 버킷에 대한 액세스 제어를 확장할 수 있습니다. S3 Express One Zone은 스토리 클래스를 사용할 수 있는 모든 AWS 리전에서 디렉터리 버킷의 태그를 지원합니다. AWS Management Console, S3 REST API, AWS CLI 또는 AWS SDK를 사용하여 태그 지정을 시작할 수 있습니다. 태그를 사용하여 비용 할당 또는 ABAC를 간소화하는 방법에 대한 자세한 내용은 Using tags with S3 directory buckets을 참조하세요.	2025년 7월 2일

이제 액세스 포인트에서 FSx for OpenZFS 볼륨에 연결 지원	Amazon FSx 콘솔, AWS CLI 또는 API를 사용하여 액세스 포인트를 생성하여 FSx for OpenZFS 볼륨에 연결할 수 있습니다. 연결되면 S3 객체 API를 사용하여 파일 데이터에 액세스할 수 있습니다. 데이터는 Amazon FSx 파일 시스템에 계속 상주하며 기존 워크로드에서 직접 액세스할 수 있습니다. 자세한 내용은 Managing access to shared datasets with access points를 참조하세요.	2025년 6월 25일
이제 S3 Tables에서 테이블에 대한 새로운 압축 전략 지원	S3 Tables은 이제 다양한 데이터 액세스 패턴을 기반으로 쿼리 성능을 최적화하는 여러 압축 전략을 지원합니다. 새로운 전략에는 Sort 및 Z-order가 포함됩니다. 자세한 내용은 Compaction Strategies for tables을 참조하세요.	2025년 6월 18일
S3 Tables에 S3 Tables에 대한 새로운 AWS 관리형 정책 추가	S3 Tables에 `AmazonS3TablesLakeFormationServiceRole` 라는 새로운 AWS 관리형 정책을 추가했습니다. 이 정책은 Lake Formation 서비스 역할이 Amazon S3 테이블 및 테이블 버킷에 액세스할 수 있는 권한을 부여합니다. 자세한 내용은 S3 Tables에 대한 AWS 관리형 정책을 참조하세요.	2025년 5월 19일

S3 Tables에서 이제 고객 관리형 키를 사용하는 SSE-KMS 지원	이제 S3 Tables는 고객 관리형 키를 사용하는 AWS Key Management Service(AWS KMS) 키(SSE-KMS)로 서버 측 암호화를 지원합니다. 테이블 버킷 및 테이블 수준에서 SSE-KMS 암호화를 적용할 수 있습니다. 자세한 내용은 테이블 버킷에서 AWS KMS 키(SSE-KMS)로 서버 측 암호화 사용을 참조하세요.	2025년 4월 16일
AWS 로컬 영역에서 디렉터리 버킷의 액세스 포인트 사용 가능	디렉터리 버킷은 이제 Amazon S3의 공유 데이터세트에 대한 대규모 데이터 액세스 관리를 간소화하는 액세스 포인트를 지원합니다. 이 새로운 기능을 사용하면 버킷당 수백 개의 액세스 포인트를 생성할 수 있으며, 각 액세스 포인트에는 각 애플리케이션에 대해 사용자 지정된 고유한 이름과 권한이 있습니다. 자세한 내용은 Managing access to shared datasets in directory buckets with access points를 참조하세요.	2025년 3월 31일

이제 Amazon SageMaker Lakehouse와의 S3 테이블 버킷 통합 정식 출시	S3 테이블 버킷을 Amazon SageMaker Lakehouse와 통합하여 Amazon Athena, Amazon Redshift, QuickSight와 같은 AWS 분석 서비스의 테이블에 액세스할 수 있습니다. Amazon SageMaker Lakehouse는 Amazon S3 데이터 레이크와 Amazon Redshift 데이터 웨어하우스의 모든 데이터를 통합하여 단일 데이터 사본에 기반하는 분석, 머신 러닝(ML), 생성 AI 애플리케이션을 구축할 수 있도록 지원합니다. 통합은 AWS Glue Data Catalog를 테이블 리소스로 채우고 이러한 리소스에 대한 액세스를 AWS Lake Formation과 페더레이션합니다. 통합을 사용하면 Lake Formation을 통해 세분화된 액세스 제어를 사용해서 추가 보안을 제공할 수 있습니다. 통합에 대한 자세한 내용은 Using Amazon S3 Tables with AWS analytics services를 참조하세요. 미리 보기 릴리스와의 통합을 설정한 경우 현재 통합을 계속 사용할 수 있습니다. 그러나 업데이트된 통합 프로세스는 성능 개선을 제공하므로 마이그레이션하는 것이 좋습니다. 업데이트된 통합으로 마이그레이션하려면 Migrating to the updated integration process를 참조하세요.	2025년 3월 13일

Amazon S3 콘솔에 S3 Tables 테이블 만들기 및 테이블 쿼리 지원 추가	S3 Tables는 이제 Amazon Athena를 사용하여 Amazon S3 콘솔에서 직접 테이블 만들기 및 테이블 쿼리 작업을 지원합니다. 이 새로운 기능을 사용하면 이제 Amazon S3 콘솔에서 몇 단계만으로 테이블을 만들고, 데이터로 채우고, 쿼리할 수 있습니다. 자세한 내용은 Creating an Amazon S3 table 및 Querying Amazon S3 tables with Athena를 참조하세요.	2025년 3월 13일
S3 Tables에서 테이블 생성 시 스키마 추가 지원 제공	이제 S3 Tables `CreateTable` API 작업을 사용하여 선택적 메타데이터 플래그를 추가하여 스키마가 있는 테이블을 생성할 수 있습니다. 자세한 내용은 Creating tables를 참조하세요.	2025년 1월 30일
S3 Tables에서 테이블 생성 시 스키마 추가 지원 제공	이제 S3 Tables `CreateTable` API 작업을 사용하여 선택적 메타데이터 플래그를 추가하여 스키마가 있는 테이블을 생성할 수 있습니다. 자세한 내용은 Creating tables를 참조하세요.	2025년 1월 30일

이제 S3 Metadata 정식 출시	이제 Amazon S3 메타데이터를 정식으로 사용할 수 있습니다. S3 메타데이터를 사용하면 거의 실시간으로 업데이트되는 쿼리 가능한 자동 메타데이터를 사용하여 S3 데이터를 쉽게 검색하고 이해할 수 있습니다. S3 메타데이터를 사용하면 비즈니스 분석, 인공 지능 및 기계 학습(AI/ML) 모델 훈련 등을 위해 S3 데이터를 큐레이션, 식별 및 사용할 수 있습니다. S3 메타데이터는 객체의 크기 및 소스 등의 시스템 정의 정보를 포함하는 객체 메타데이터와 제품 SKU, 트랜잭션 ID 또는 콘텐츠 등급 등의 정보가 포함된 태그와 같은 사용자 지정 메타데이터를 지원합니다. 자세한 내용은 Accelerating data discovery with S3 Metadata를 참조하세요.	2025년 1월 27일
AWS 관리형 정책 - 새 정책	S3 Tables에 새 AWS 관리형 정책 2개가 추가되었습니다.	2024년 12월 6일

S3 Tables	Amazon S3 Tables은 쿼리 성능을 개선하고, 테이블의 스토리지 비용을 절감하고, 데이터 레이크 운영을 대규모로 간소화하는 기능을 통해 분석 워크로드에 최적화된 S3 스토리지를 제공합니다. S3 Tables은 Apache Iceberg 테이블을 하위 리소스로 저장하기 위해 특별히 빌드된 테이블 버킷이라는 새로운 버킷 유형을 도입합니다. 테이블 버킷은 S3 범용 버킷의 자체 관리형 테이블에 비해 초당 트랜잭션(TPS)이 더 높고 쿼리 처리량이 더 우수합니다. 테이블 버킷을 Athena, Amazon Redshift, QuickSight 등과 같은 AWS 분석 서비스와 자동으로 통합할 수 있습니다. 자세한 내용은 Working with Amazon S3 Tables and table buckets을 참조하세요.	2024년 12월 3일

S3 Metadata 미리 보기	Amazon S3 메타데이터를 사용하면 거의 실시간으로 업데이트되는 쿼리 가능한 자동 메타데이터를 사용하여 S3 데이터를 쉽게 검색하고 이해할 수 있습니다. S3 메타데이터를 사용하면 비즈니스 분석, 인공 지능 및 기계 학습(AI/ML) 모델 훈련 등을 위해 S3 데이터를 큐레이션, 식별 및 사용할 수 있습니다. S3 메타데이터는 객체의 크기 및 소스 등의 시스템 정의 정보를 포함하는 객체 메타데이터와 제품 SKU, 트랜잭션 ID 또는 콘텐츠 등급 등의 정보가 포함된 태그와 같은 사용자 지정 메타데이터를 지원합니다. 자세한 내용은 Accelerating data discovery with S3 Metadata를 참조하세요.	2024년 12월 3일
Storage Browser for S3	Storage Browser for S3는 웹 애플리케이션에 추가하여 최종 사용자에게 S3에 저장된 데이터에 대한 간단한 인터페이스를 제공할 수 있는 오픈 소스 구성 요소입니다. 자세한 내용은 Storage Browser for S3를 참조하세요.	2024년 12월 1일
새로운 Amazon S3 체크섬 알고리즘 및 향상된 체크섬 무결성 기능	Amazon S3는 CRC-64NVME 체크섬 알고리즘을 추가하고 체크섬 무결성 기능을 개선했습니다. 자세한 내용은 Checking object integrity in Amazon S3를 참조하세요.	2024년 12월 1일

데이터 레지던시 워크로드	전용 로컬 영역에서 S3 디렉터리 버킷을 생성하여 데이터 레지던시 및 격리 사용 사례에 대한 데이터를 저장할 수 있습니다. 자세한 내용은 Data residency workloads를 참조하세요.	2024년 12월 1일
조건부 쓰기를 강제 적용하기 위한 새 조건 키	Amazon S3는 버킷 정책에서 사용할 새 조건 키 `s3:if-match` 및 `s3:if-none-match`를 추가하여 클라이언트가 `If-None-Match` 또는 `If-Match` HTTP 헤더를 사용하도록 강제합니다. 자세한 내용은 Enforce conditional writes on Amazon S3 buckets을 참조하세요.	2024년 11월 25일
객체가 변경되었는지 확인하기 위한 조건부 쓰기의 새 HTTP 헤더	Amazon S3는 일부 API 작업에 대해 객체를 쓰기 전에 `If-Match` HTTP 헤더를 추가하여 객체의 엔터티 태그(ETag)를 확인합니다. 이 헤더를 사용하여 Amazon S3는 제공된 ETag 값을 S3에 있는 객체의 ETag 값과 비교합니다. 이때 ETag 값이 일치하지 않으면 작업이 실패합니다. 자세한 내용은 조건부 쓰기를 통해 객체 덮어쓰기를 방지하는 방법을 참조하세요.	2024년 11월 25일

S3 Access Grants 및 Amazon Redshift 통합	이제 Amazon Redshift 고객은 S3 Access Grants를 사용하여 S3 데이터에 대한 권한의 규모를 조정하고 관리할 수 있습니다. 이를 통해 Amazon Redshift 고객은 IAM 사용자 및 그룹뿐만 아니라 AWS IAM Identity Center을 사용하여 기업 ID에 대한 S3 권한의 규모를 조정할 수 있습니다. 자세한 내용은 Amazon S3 Access Grants 및 Amazon Redshift 통합을 참조하세요.	2024년 11월 15일
이제 AWS Organizations 멤버 계정이 실수로 잠긴 Amazon S3 버킷에 다시 액세스할 수 있습니다.	이제 AWS Organizations 멤버 계정은 AWS Identity and Access Management(IAM)를 통한 간단한 프로세스를 사용하여 실수로 잠긴 Amazon S3 버킷에 다시 액세스할 수 있습니다. 자세한 내용은 AWS Identity and Access Management 사용 설명서의 AWS Organizations 멤버 계정에서 권한 있는 작업 수행을 참조하세요.	2024년 11월 14일
AWS Organizations의 새로운 권한 부여 정책 유형인 리소스 제어 정책(RCP)도 Amazon S3 버킷에 사용할 수 있습니다.	AWS Organizations에서 관리되는 새로운 권한 부여 정책인 리소스 제어 정책(RCP)을 사용하여 전체 조직 내의 Amazon S3 버킷에 사용 가능한 최대 권한을 설정할 수 있습니다. 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCP)을 참조하세요.	2024년 11월 13일

Amazon S3는 이제 버킷 할당량을 최대 1,000개의 버킷까지 늘릴 수 있도록 자동으로 승인	Amazon S3는 이제 버킷 할당량을 최대 1,000개의 버킷까지 늘릴 수 있도록 자동으로 승인합니다. 버킷 사용량을 보거나 증가를 요청하려면 Service Quotas 콘솔을 방문하세요.	2024년 9월 30일
Amazon S3 수명 주기 구성을 위한 새로운 기본 최소 객체 크기 전환 동작	Amazon S3는 이제 128KB 미만의 객체가 스토리지 클래스로 전환되지 않도록 하는 기본 동작을 S3 수명 주기 구성에 적용합니다. 이 동작을 재정의하는 방법을 알아보려면 Allowing objects smaller than 128 KB to be transitioned를 참조하세요.	2024년 9월 24일
이제 디렉터리 버킷이 고객 관리형 키를 사용하여 SSE-KMS를 지원합니다.	이제 디렉터리 버킷은 고객 관리형 키를 사용하는 AWS Key Management Service(AWS KMS) 키(SSE-KMS)로 서버 측 암호화를 지원합니다. 디렉터리 버킷에서 데이터의 보안을 암호화하고 관리할 수 있는 더 많은 옵션이 있습니다. 자세한 내용은 디렉터리 버킷의 데이터 보호 및 암호화를 참조하세요.	2024년 9월 17일

S3 Access Grants는 호출자의 액세스 권한 부여를 나열하는 API 작업 지원	IAM ID 및 IAM Identity Center 기업 디렉터리 ID는 이제 `ListCallerAccessGrants` API를 사용하여 S3 Access Grants에서 정의한 대로 액세스할 수 있는 모든 Amazon S3 버킷, 접두사 및 객체를 나열할 수 있습니다. 이 API를 사용하면 특정 AWS 계정 내에서 IAM 또는 기업 디렉터리 ID가 S3 Access Grants를 통해 액세스할 수 있는 모든 S3 데이터를 검색할 수 있습니다. 자세한 내용은 호출자의 액세스 권한 부여 목록을 참조하세요.	2024년 9월 5일

동일 계정 요청에 대한 액세스 거부 오류 메시지 개선	이제 Amazon S3는 동일한 AWS 계정 계정 내의 리소스에 대한 요청에 대한 액세스 거부 (HTTP `403 Forbidden`) 오류에 추가 컨텍스트를 포함합니다. 이 새로운 컨텍스트에는 액세스를 거부한 정책 유형, 거부 이유, 리소스에 대한 액세스를 요청한 AWS Identity and Access Management(IAM) 사용자 또는 역할에 대한 정보가 포함됩니다. 이 컨텍스트는 액세스 문제를 해결하고, 액세스 거부 오류의 근본 원인을 파악하고, 관련 정책을 업데이트하여 잘못된 액세스 제어를 수정하는 데 도움이 됩니다. 이 추가 컨텍스트는 AWS CloudTrail 로그에서도 사용할 수 있습니다. 동일 계정 요청에 대한 향상된 액세스 거부 오류 메시지는 이제 AWS GovCloud (US) Regions 및 중국 리전을 포함한 모든 AWS 리전에서 사용할 수 있습니다. 자세한 내용은 Amazon S3의 액세스 거부(403 금지) 오류 문제 해결을 참조하세요.	2024년 8월 21일

Amazon S3에서 PutObject 및 CompleteMultipartUpload 에 대한 조건부 쓰기 사용 지원	업로드 시 조건부 쓰기 작업을 사용하여 객체를 생성하기 전에 버킷에 객체가 있는지 확인할 수 있습니다. 이렇게 하면 기존 데이터의 덮어쓰기를 방지할 수 있습니다. 조건부 쓰기는 버킷에 이미 같은 키 이름을 가진 기존 객체가 없는지 확인합니다. 자세한 내용은 조건부 요청을 참조하세요.	2024년 8월 20일
Amazon S3는 더 이상 여러 HTTP 오류 코드에 대해 요금을 부과하지 않습니다.	Amazon S3는 고객이 시작하지 않은 무단 요청이 무료로 제공되도록 변경되었습니다. 자세한 내용은 Amazon S3 오류 응답에 대한 청구를 참조하세요.	2024년 8월 19일
신규 고객은 더 이상 Amazon S3 Select를 사용할 수 없음	신규 고객은 더 이상 Amazon S3 Select를 사용할 수 없습니다. Amazon S3 Select 기존 고객은 평소처럼 이 기능을 계속 사용할 수 있습니다. 자세히 알아보기	2024년 7월 25일
Amazon S3 인벤토리, s3:InventoryAccessibleOptionalFields 조건 키 지원	Amazon S3 인벤토리가 s3:InventoryAccessibleOptionalFields 조건 키를 지원하여 사용자가 보고서에 선택적 메타데이터 필드를 포함할 수 있는지를 제어합니다. 자세한 내용은 S3 인벤토리 보고서 구성 생성 제어를 참조하세요.	2024년 2월 20일

S3 on Outposts에 대한 IPv6 지원	이제 S3 on Outposts 듀얼 스택 엔드포인트를 통해 IPv6를 사용하여 S3 on Outposts 버킷에 액세스할 수 있습니다. S3 on Outposts에 대한 IPv6 지원을 사용하면 IPv6 네트워크를 통해 S3 on Outposts 버킷과 컨트롤 플레인 리소스를 관리할 수 있습니다.	2024년 1월 16일
새로운 고성능, 단일 영역 Amazon S3 스토리지 클래스 - S3 Express One Zone	S3 Express One Zone은 지연 시간에 가장 민감한 애플리케이션에 대해 일관되게 10밀리초 미만의 데이터 액세스를 제공하도록 특별히 설계된 고성능 단일 영역 Amazon S3 스토리지 클래스입니다. 자세한 내용은 S3 Express One Zone을 참조하세요.	2023년 11월 28일
Mountpoint for Amazon S3에 S3 Express One Zone에 대한 지원 추가	이제 Mountpoint를 사용하여 S3 Express One Zone 디렉터리 버킷을 마운트할 수 있습니다.	2023년 11월 28일
Lambda 간접 호출 스키마 버전	Amazon S3 배치 작업에 디렉터리 버킷에서 작동하는 배치 작업 작업과 함께 사용할 수 있는 새로운 Lambda 호출 스키마 버전을 도입했습니다. 자세한 내용은 디렉터리 버킷에 Lambda 및 Amazon S3 배치 작업 사용을 참조하세요.	2023년 11월 28일

디렉터리 버킷의 가져오기 작업	Amazon S3에 가져오기 작업을 도입했습니다. 가져오기는 범용 버킷에서 디렉터리 버킷으로 객체를 복사하는 Amazon S3 배치 작업 건을 생성하는 간소화된 방법입니다. 자세한 내용은 디렉터리 버킷으로 객체 가져오기을 참조하세요.	2023년 11월 28일
S3 Access Grants로 S3 액세스 관리	Amazon S3 Access Grants를 사용하면 Azure AD와 같은 기업 디렉터리의 디렉터리 ID 외에도 AWS Identity and Access Management(IAM) 보안 주체에 대한 대규모 데이터 권한을 관리할 수 있습니다. 이제 S3 권한을 최소 권한으로 적용하고 비즈니스 요구 사항에 따라 해당 권한을 쉽게 확장할 수 있습니다. 자세한 내용은 S3 Access Grants를 통한 액세스 관리를 참조하세요.	2023년 11월 26일
Mountpoint for Amazon S3에 캐시 기능 추가	이제 Mountpoint를 사용하여 반복적으로 액세스하는 데이터에 대한 캐시를 구성할 수 있습니다.	2023년 11월 22일

Amazon S3 배치 작업 매니페스트 생성 향상	이제 작업을 생성할 때 지정한 객체 필터 기준에 따라 자동으로 매니페스트를 생성하도록 Amazon S3 배치 작업에 지시할 수 있습니다. 이 옵션은 Amazon S3 콘솔에서 생성하는 배치 복제 작업 또는 AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용하여 생성하는 작업 유형에 사용할 수 있습니다. 자세한 내용은 Amazon S3 배치 작업 건 생성을 참조하세요.	2023년 11월 22일
기존 Amazon S3 버킷에 객체 잠금 구성 추가 가능	이제 기존 Amazon S3 버킷에 객체 잠금을 활성화할 수 있습니다. 새 버킷 또는 기존 버킷에 법적 보존 및 보존 기간을 설정할 수 있습니다. 자세한 내용은 객체 잠금 사용을 참조하세요.	2023년 11월 20일
접두사에 대한 S3 Storage Lens 요청 지표	S3 Storage Lens에서 Amazon S3 버킷의 접두사에 대한 요청 지표를 도입했습니다. 자세한 내용은 지표 범주를 참조하십시오.	2023년 11월 17일
Amazon S3 스토리지 렌즈 그룹	S3 스토리지 렌즈는 객체 메타데이터를 기반으로 객체에 대한 사용자 지정 필터인 스토리지 렌즈 그룹을 도입했습니다. 자세한 내용은 Amazon S3 스토리지 렌즈 사용을 참조하십시오.	2023년 11월 15일

새 IAM 정책	S3 on Outposts에는 네트워크 리소스를 관리하는 데 도움이 되는 서비스 연결 역할인 `AWSServiceRoleForS3OnOutposts` 가 도입되었습니다. 자세한 내용은 S3 on Outposts에 서비스 연결 역할 사용을 참조하십시오.	2023년 10월 3일
Amazon S3에서 삭제 마커의 `Last-Modified` 시간 제공	Amazon S3는 S3 `Head` 및 `Get` API 작업의 응답 헤더에 삭제 마커의 `Last-Modified` 시간을 제공합니다. 자세한 내용은 삭제 마커를 통한 작업을 참조하십시오.	2023년 9월 27일
AWS 관리형 정책에 대한 Amazon S3 업데이트	Amazon S3에서 `AmazonS3ReadOnlyAccess` 에 `s3:Describe*` 권한이 추가되었습니다. 자세한 내용은 Amazon S3용 AWS 관리형 정책을 참조하십시오.	2023년 8월 11일
S3 배치 작업을 통해 이루어진 스탠다드 복원 요청의 시작 시간 개선	S3 배치 작업을 통해 이루어진 복원 요청의 스탠다드 검색을 이제 몇 분 내에 시작할 수 있습니다. 자세한 내용은 아카이브 검색 옵션을 참조하십시오.	2023년 8월 9일
Amazon S3 버킷을 로컬 파일 시스템에 마운트하는 데 사용되는 높은 처리량 성능의 클라이언트인 Mountpoint가 추가됨	Mountpoint를 사용하면 애플리케이션에서 파일 작업을 통해 Amazon S3에 저장된 객체에 액세스할 수 있으므로 애플리케이션이 파일 인터페이스를 통해 Amazon S3의 탄력적 스토리지 및 처리량을 이용할 수 있습니다.	2023년 8월 9일

AWS Key Management Service 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)	AWS Key Management Service(AWS KMS) 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)는 Amazon S3에 객체를 업로드할 때 객체에 두 계층의 암호화를 적용합니다. 자세한 내용은 AWS KMS 키를 사용한 이중 계층 서버 측 암호화 사용을 참조하십시오.	2023년 6월 13일
Amazon S3는 S3 퍼블릭 액세스 차단을 활성화하고 모든 새 버킷에 대해 S3 액세스 제어 목록(ACL)을 비활성화합니다.	이제 Amazon S3가 S3 퍼블릭 액세스 차단을 활성화하고 모든 AWS 리전에 있는 모든 새 S3 버킷에 대해 S3 액세스 제어 목록(ACL)을 비활성화합니다. 자세한 내용은 Amazon S3 스토리지에 대한 퍼블릭 액세스 차단 및 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지를 참조하십시오.	2023년 4월 27일
S3 복제 작업 실패 지표	S3 복제 실패를 모니터링하기 위해 Amazon S3에 새로운 Amazon CloudWatch 지표가 추가됩니다. 자세한 내용은 복제 지표로 진행률 모니터링을 참조하십시오.	2023년 4월 5일
프라이빗 DNS	AWS PrivateLink for Amazon S3에서 이제 프라이빗 DNS를 지원합니다. 자세한 내용은 프라이빗 DNS를 참조하십시오.	2023년 3월 14일

Amazon S3 콘솔에서 크로스 계정 액세스 포인트 지원	Amazon S3는 이제 Amazon S3 콘솔을 사용한 크로스 계정 액세스 포인트 생성을 지원합니다. 자세한 내용은 액세스 포인트 생성을 참조하십시오.	2023년 3월 14일
Amazon S3 on Outposts에서 Outposts에 대한 S3 복제 지원	로컬 S3 복제를 사용하여 단일 Outposts 대상 버킷 또는 여러 대상 버킷에 객체를 복제할 수 있습니다. 대상 버킷은 다른 AWS Outposts에 있을 수도 있고 소스 버킷과 동일한 리전 내에 있을 수도 있습니다. 자세한 내용은 S3 on Outposts에 대한 객체 복제를 참조하십시오.	2023년 3월 14일
Amazon S3 객체 Lambda 액세스 포인트 별칭	객체 Lambda 액세스 포인트를 생성할 때 Amazon S3는 객체 Lambda 액세스 포인트에 대한 고유한 별칭을 자동으로 생성합니다. 액세스 포인트 데이터 영역 작업 요청에 Amazon S3 버킷 이름이나 객체 Lambda 액세스 포인트 Amazon 리소스 이름(ARN) 대신 이 별칭을 사용할 수 있습니다. 자세한 내용은 객체 Lambda 액세스 포인트에 버킷 스타일 별칭을 사용하는 방법을 참조하십시오.	2023년 3월 14일
Amazon S3 다중 리전 액세스 포인트 크로스 계정 지원	Amazon S3는 이제 Amazon S3 콘솔을 사용한 크로스 계정 다중 리전 액세스 포인트 생성을 지원합니다. 자세한 내용은 다중 리전 액세스 포인트 생성을 참조하십시오.	2023년 3월 14일

크로스 계정 액세스 포인트	Amazon S3는 크로스 계정 액세스 포인트 생성을 지원합니다. AWS Command Line Interface(AWS CLI) 또는 REST API `CreateAccessPoint` 작업을 사용하여 크로스 계정 액세스 포인트를 만들 수 있습니다. 자세한 내용은 액세스 포인트 생성을 참조하십시오.	2022년 11월 30일
Amazon S3가 Amazon S3 다중 리전 액세스 포인트에 장애 조치 제어를 지원함	Amazon S3에 다중 리전 액세스 포인트에 대한 장애 조치 제어가 출시되었습니다. 이러한 제어를 통해 Amazon S3 다중 리전 액세스 포인트를 통해 라우팅되는 S3 데이터 액세스 요청 트래픽을 몇 분 내에 대체 AWS 리전으로 전환하여 가용성이 높은 애플리케이션을 테스트하고 빌드할 수 있습니다. 자세한 내용은 Amazon S3 Multi-Region Access Point failover controls(Amazon S3 다중 리전 액세스 포인트 장애 조치 제어)를 참조하십시오.	2022년 11월 28일
34개의 새로운 지표로 Amazon S3 스토리지 렌즈에서 조직 전반의 가시성을 높임	S3 스토리지 렌즈에 추가로 34개의 지표가 출시되어 더욱 심층적인 비용 최적화 기회를 찾아내고 데이터 보호 모범 사례를 식별하며 애플리케이션 워크플로의 성능을 개선합니다. 자세한 내용은 S3 스토리지 렌즈 metrics(S3 스토리지 렌즈 지표)를 참조하십시오.	2022년 11월 17일

Amazon S3가 S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive에 더 높은 복원 요청 속도를 지원함	Amazon S3가 S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive 스토리지 클래스에 대해 AWS 계정별로 초당 최대 1,000개의 트랜잭션 속도로 복원 요청을 지원합니다.	2022년 11월 15일
Amazon S3 on Outposts가 추가 S3 수명 주기 작업 및 필터를 지원함	S3 on Outposts가 용량 관리를 최적화할 수 있도록 추가 S3 수명 주기 규칙을 지원합니다. 객체가 오래되거나 더 최신 버전으로 교체되면 객체를 만료시킬 수 있습니다. 전체 버킷에 대해 또는 접두사, 객체 태그 또는 객체 크기로 필터링하여 버킷 객체의 하위 집합에 대해 수명 주기 규칙을 만들 수 있습니다. 자세한 내용은 Creating and managing a lifecycle configuration(수명 주기 구성 생성 및 관리)을 참조하십시오.	2022년 11월 2일
SSE-C 객체에 S3 복제를 지원함	고객 제공 키를 사용하는 서버 측 암호화로 생성된 객체는 복제할 수 있습니다. 암호화된 객체 복제에 대한 자세한 내용은 서버 측 암호화(SSE-C, SSE-S3, SSE-KMS)를 사용하여 생성된 객체 복제를 참조하십시오.	2022년 10월 24일

Amazon S3 on Outposts에서 액세스 포인트 별칭 지원	S3 on Outposts를 통해 Outposts 버킷의 객체에 액세스하려면 액세스 포인트를 사용해야 합니다. 버킷에 대한 액세스 포인트를 생성할 때마다 S3 on Outposts는 자동으로 액세스 포인트 별칭을 생성합니다. 모든 데이터 플레인 작업에 액세스 포인트 ARN 대신 이 액세스 포인트 별칭을 사용할 수 있습니다. 자세한 내용은 S3 on Outposts 버킷 액세스 포인트에 버킷 스타일 별칭 사용을 참조하십시오.	2022년 10월 21일
S3 객체 Lambda에서 `HeadObject`, `ListObjects` 및 `ListObjectsV2` 작업 지원	사용자 지정 코드를 사용하여 행을 필터링하고 동적으로 이미지 크기를 조정하며 기밀 데이터를 교정하는 등 표준 S3 `GET`, `LIST` 또는 `HEAD` 요청에서 반환한 데이터를 수정할 수 있습니다. 자세한 내용은 S3 객체 Lambda를 사용하여 객체 변환을 참고하십시오.	2022년 10월 4일

Amazon S3 on Outposts에서 S3 버전 관리 지원	S3 버전 관리를 활성화하면 동일 버킷 내에 여러 개의 개별 객체 복제본을 저장합니다. S3 버전 관리를 사용하여 Outposts 버킷에 저장된 모든 버전의 객체를 모두 보존, 검색 및 복원할 수 있습니다. S3 버전 관리는 의도치 않은 사용자 작업 및 애플리케이션 장애로부터 복구하는 데 도움이 됩니다. 자세한 내용은 S3 on Outposts 버킷의 S3 버전 관리에 대한 관리를 참조하십시오.	2022년 9월 21일
Amazon S3용 AWS Backup	AWS Backup은 Amazon S3 데이터를 보호하기 위해 중앙 백업 정책을 정의하는 데 사용할 수 있는 완전관리형 정책 기반 서비스입니다. 자세한 내용은 Amazon S3에 AWS Backup 사용을 참조하십시오.	2022년 2월 18일
S3 Batch Replication을 사용하여 기존 객체 복제	S3 배치 복제를 사용하면 복제 구성이 적용되기 전에 존재했던 객체를 복제할 수 있습니다. 기존 객체 복제는 배치 작업을 통해 수행됩니다. S3 배치 복제는 Amazon S3 버킷에서 새 객체를 지속적으로 자동으로 복사하는 라이브 복제와 다릅니다. 자세한 내용은 S3 배치 복제를 사용한 기존 객체 복제를 참조하십시오.	2022년 2월 8일

S3 Glacier Flexible Retrieval의 이름 변경	Glacier 스토리지 클래스의 이름이 S3 Glacier Flexible Retrieval로 변경되었습니다. 이 변경 사항은 API에 영향을 미치지 않습니다.	2021년 11월 30일
ACL을 사용 중지하는 새로운 S3 객체 소유권 설정	객체 소유권에 대해 버킷 소유자 적 설정을 적용하여 버킷과 버킷에 있는 객체에 대한 ACL을 비활성화하고 버킷에 있는 모든 객체의 소유권을 얻을 수 있습니다. 버킷 소유자 적용 설정은 Amazon S3 저장된 데이터에 대한 액세스 관리를 단순화합니다. 자세한 내용은 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지를 참조하십시오.	2021년 11월 30일
새로운 S3 Intelligent-Tiering 스토리지 클래스	S3 Intelligent-Tiering Archive Instant Access는 S3 Intelligent-Tiering 아래의 추가 스토리지 클래스입니다. 자세한 내용은 S3 Intelligent-Tiering 작동 방식을 참조하십시오.	2021년 11월 30일
새로운 S3 Glacier Instant Retrieval 스토리지 클래스	이제 S3 Glacier Instant Retrieval 스토리지 클래스에 객체를 배치할 수 있습니다. 이 스토리지 클래스에 대한 자세한 내용은 Amazon S3 스토리지 클래스 사용을 참조하십시오.	2021년 11월 30일

AWS Backup for Amazon S3 평가판	AWS Backup은 Amazon S3 데이터를 보호하기 위해 중앙 백업 정책을 정의하는 데 사용할 수 있는 완전관리형 정책 기반 서비스입니다. 자세한 내용은 Amazon S3에 AWS Backup 사용을 참조하십시오.	2021년 11월 30일
Amazon S3용 AWS Identity and Access Management Access Analyzer	IAM Access Analyzer는 정책 확인은 실행하여 IAM 정책 문법 및 모범 사례에 대해 정책을 검증합니다. IAM 액세스 분석기를 사용한 정책 검증에 대한 자세한 내용은 IAM 사용 설명서의 IAM 액세스 분석기 정책 검증을 참조하십시오.	2021년 11월 30일
새로운 이벤트 유형	Amazon S3 이벤트 알림에 추가된 새로운 이벤트 유형은 Amazon S3 이벤트 알림을 참조하십시오.	2021년 11월 29일
버킷에서 Amazon EventBridge 사용	Amazon S3 버킷에서 EventBridge를 사용 설정하여 Amazon EventBridge로 이벤트를 보낼 수 있습니다. EventBridge 사용을 참조하십시오.	2021년 11월 29일
새로운 S3 수명 주기 필터	객체 크기를 기반으로 수명 주기 규칙을 생성하거나, 최신이 아닌 객체 버전을 몇 개 유지할지 지정할 수 있습니다. 자세한 내용은 S3 수명 주기 구성의 예제를 참조하십시오.	2021년 11월 23일

Amazon CloudWatch에 Amazon S3 스토리지 렌즈 지표 게시	S3 스토리지 렌즈 사용량 및 활동 지표를 Amazon CloudWatch에 게시하여 CloudWatch 대시보드에서 운영 상태에 대한 통합 보기를 생성할 수 있습니다. 또한 경보 및 트리거된 작업, 지표 수학, 이상 감지와 같은 CloudWatch 기능을 사용하여 S3 스토리지 렌즈 지표를 모니터링하고 조치를 취할 수 있습니다. 또한 CloudWatch API를 사용하면 서드 파티 공급자를 포함한 애플리케이션이 S3 스토리지 렌즈 지표에 액세스할 수 있습니다. 자세한 내용은 CloudWatch에서 S3 스토리지 렌즈 지표 모니터링을 참조하십시오.	2021년 11월 22일
다중 리전 액세스 포인트	다중 리전 액세스 포인트를 사용하여 애플리케이션이 여러 AWS 리전에 있는 Amazon S3 버킷의 요청을 이행하는 데 사용할 수 있는 글로벌 엔드포인트를 생성할 수 있습니다. 이 다중 리전 액세스 포인트를 사용하여 지연 시간이 가장 낮은 버킷으로 데이터 경로를 지정할 수 있습니다. 다중 리전 액세스 포인트 및 해당 사용 방법에 대한 자세한 내용은 Amazon S3의 다중 리전 액세스 포인트를 참조하십시오.	2021년 9월 2일

Amazon S3 on Outposts를 사용하여 애플리케이션에 직접 로컬 액세스 추가	AWS Outposts 가상 프라이빗 클라우드(VPC) 외부에서 애플리케이션을 실행하고 S3 on Outposts 데이터에 액세스합니다. 온프레미스 네크워크에서 직접 S3 on Outposts 객체에 액세스할 수도 있습니다. 고객 소유 IP(CoIP) 주소를 사용하여 S3 on Outposts 엔드포인트를 구성하고 온프레미스 네트워크에서 로컬 게이트웨이를 생성하여 객체에 액세스하는 방법에 대한 자세한 내용은 VPC 전용 액세스 포인트를 사용해 Amazon S3 on Outposts에 액세스를 참조하십시오.	2021년 7월 29일
Amazon S3 액세스 포인트 별칭	액세스 포인트를 생성하면 Amazon S3가 자동으로 데이터 액세스에 버킷 이름 대신 사용할 수 있는 별칭을 생성합니다. 모든 액세스 포인트 데이터 영역 작업에 Amazon 리소스 이름(ARN) 대신 이 액세스 포인트 별칭을 사용할 수 있습니다. 자세한 내용은 액세스 포인트에 버킷 스타일 별칭 사용을 참조하십시오.	2021년 7월 26일

Amazon S3 Inventory 및 S3 배치 작업에서 S3 버킷 키 상태 지원	Amazon S3 인벤토리 및 배치 작업은 S3 버킷 키를 사용하여 기존 객체를 식별하고 복사하는 작업을 지원합니다. S3 버킷 키는 기존 객체에 대한 서버 측 암호화 비용 절감 효과를 가속화합니다. 자세한 내용은 Amazon S3 인벤토리 및 Batch Operations 객체 복사를 참조하십시오.	2021년 6월 3일
Amazon S3 스토리지 렌즈 지표 계정 스냅샷	S3 스토리지 렌즈 계정 스냅샷은 기본 대시보드의 지표를 요약하여 S3 콘솔 홈(버킷) 페이지에 총 스토리지, 객체 수 및 평균 객체 크기를 표시합니다. 자세한 내용은 S3 스토리지 렌즈 지표 계정 스냅샷을 참조하십시오.	2021년 5월 5일
Amazon S3 on Outposts 엔드포인트 지원 증가	S3 on Outposts는 이제 Outpos당 최대 100개의 엔드포인트를 지원합니다. 자세한 내용은 S3 on Outposts 네트워크 제한을 참조하십시오.	2021년 4월 29일
Amazon CloudWatch Events에서 Amazon S3 on Outposts 이벤트 알림	CloudWatch Events를 사용하여 S3 on Outposts API 이벤트를 캡처하고 지원되는 모든 CloudWatch 대상을 통해 알림을 받는 규칙을 생성할 수 있습니다. 자세한 내용은 CloudWatch Events를 사용하여 S3 on Outposts 이벤트 알림 수신을 참조하십시오.	2021년 4월 19일

S3 객체 Lambda	S3 객체 Lambda를 통해 자체 코드를 Amazon S3 GET 요청에 추가하여 애플리케이션으로 데이터가 반환될 때 데이터를 수정 및 처리할 수 있습니다. 사용자 지정 코드를 사용하여 행을 필터링하고 동적으로 이미지 크기를 조정하며 기밀 데이터를 교정하는 등 표준 S3 GET 요청에서 반환한 데이터를 수정할 수 있습니다. 자세한 내용은 객체 변환을 참조하십시오.	2021년 3월 18일
AWS PrivateLink	Amazon S3용 AWS PrivateLink를 사용하면 인터넷을 통해 연결하지 않고 가상 프라이빗 클라우드(VPC)의 인터페이스 엔드포인트를 사용하여 S3에 직접 연결할 수 있습니다. 인터페이스 엔드포인트는 온프레미스 또는 다른 AWS 리전에 있는 애플리케이션에서 직접 액세스할 수 있습니다. 자세한 내용은 Amazon S3용 AWS PrivateLink를 참조하십시오.	2021년 2월 2일
다음을 사용하여 Amazon S3 on Outposts 용량 관리AWS CloudTrail	S3 on Outposts 관리 이벤트는 CloudTrail 로그를 통해 사용할 수 있습니다. 자세한 내용은 CloudTrail을 사용하여 S3 on Outposts 용량 관리를 참조하십시오.	2020년 12월 21일

강력한 일관성	Amazon S3는 모든 AWS 리전의 Amazon S3 버킷에 있는 객체의 PUT 및 DELETE 요청에 대해 강력한 쓰기 후 읽기 일관성을 제공합니다. 또한 Amazon S3 Select, Amazon S3 액세스 제어 목록, Amazon S3 객체 태그, 객체 메타데이터(예: HEAD 객체)에 대한 읽기 작업은 매우 일관적입니다. 자세한 내용은 Amazon S3 데이터 일관성 모델을 참조하십시오.	2020년 12월 1일
Amazon S3 복제본 수정 동기화	Amazon S3 복제본 수정 동기화는 태그, ACL 및 객체 잠금 설정과 같은 객체 메타데이터를 소스 객체와 복제본 간에 동기화된 상태로 유지합니다. 이 기능을 사용 설정하면 Amazon S3가 소스 객체 또는 복제본 복사본에 대한 메타데이터 변경 사항을 복제합니다. 자세한 내용은 복제본 수정 동기화를 사용하여 메타데이터 변경 복제를 참조하십시오.	2020년 12월 1일

Amazon S3 버킷 키	Amazon S3 버킷 키는 AWS Key Management Service(SSE-KMS)를 사용하여 Amazon S3 서버 측 암호화 비용을 절감합니다. 이 새로운 서버 측 암호화용 버킷 수준 키는 Amazon S3에서 AWS KMS(으)로 가는 요청 트래픽을 줄여 AWS KMS 요청 비용을 최대 99% 줄일 수 있습니다. 자세한 내용은 S3 버킷 키를 사용하여 SSE-KMS 비용 절감을 참조하십시오.	2020년 12월 1일
Amazon S3 스토리지 렌즈	S3 스토리지 렌즈는 지표를 집계하고 Amazon S3 콘솔 버킷 페이지의 계정 스냅샷 섹션에 이 정보를 표시합니다. S3 스토리지 렌즈는 또한 인사이트와 추세를 시각화하고, 이상치에 플래그를 지정하고, 스토리지 비용 최적화와 데이터 보호 모범 사례 적용을 위한 권장 사항을 수신하는 데 사용할 수 있는 대화형 대시보드를 제공합니다. 대시보드에 있는 드릴다운 옵션을 통해 조직, 계정, AWS 리전, 스토리지 클래스, 버킷, 접두사 또는 스토리지 렌즈 그룹 수준에서 인사이트를 생성하거나 시각화할 수 있습니다. 또한 CSV 또는 Parquet 형식의 일일 지표 내보내기를 S3 버킷으로 전송할 수 있습니다. 자세한 내용은 S3 스토리지 렌즈를 사용한 스토리지 활동 및 사용량 평가를 참조하십시오.	2020년 11월 18일

AWS X-Ray를 사용하여 S3 요청 추적	Amazon S3가 X-Ray와 통합되어 추적 컨텍스트를 전파하고 업스트림 및 다운스트림 노드와 함께 하나의 요청 체인을 제공합니다. 자세한 내용은 X-Ray를 사용하여 요청 추적을 참조하십시오.	2020년 11월 16일
S3 복제 지표	S3 복제 지표는 복제 구성의 복제 규칙에 대한 세부 지표를 제공합니다. 자세한 내용은 복제 지표 및 Amazon S3 이벤트 알림을 참조하십시오.	2020년 11월 9일
S3 Intelligent-Tiering Archive Access 및 Deep Archive Access	S3 Intelligent-Tiering Archive Access 및 Deep Archive Access는 S3 Intelligent-Tiering의 추가 스토리지 계층입니다. 자세한 내용은 자주 액세스하는 객체와 자주 액세스하지 않는 객체를 자동으로 최적화하는 스토리지 클래스를 참조하십시오.	2020년 11월 9일
삭제 마커 복제	삭제 마커 복제를 사용하면 복제 규칙의 대상 버킷에 삭제 마커가 복사됩니다. 자세한 내용은 삭제 마커 복제 사용을 참조하십시오.	2020년 11월 9일
S3 객체 소유권	객체 소유권은 버킷에 업로드되는 새 객체의 소유권을 제어하는 데 사용할 수 있는 S3 버킷 설정입니다. 자세한 내용은 S3 객체 소유권 사용을 참조하십시오.	2020년 10월 2일

Amazon S3 on Outposts	Amazon S3 on Outposts를 사용하면 AWS Outposts 리소스에서 S3 버킷을 생성하고 로컬 데이터 액세스, 로컬 데이터 처리 및 데이터 레지던시가 필요한 애플리케이션을 위해 온프레미스에서 객체를 저장하고 검색할 수 있습니다. AWS Management Console, AWS CLI, AWS SDK 또는 REST API를 통해 S3 on Outposts를 사용할 수 있습니다. 자세한 내용은 Amazon S3 on Outposts 사용을 참조하십시오.	2020년 9월 30일
버킷 소유자 조건	Amazon S3 버킷 소유자 조건을 사용하여 S3 작업에 사용하는 버킷이 의도한 AWS 계정에 속하는지 확인할 수 있습니다. 자세한 내용은 버킷 소유자 조건을 참조하십시오.	2020년 9월 11일
S3 배치 작업에서 객체 잠금 보존 지원	이제 S3 객체 잠금을 적용한 배치 작업을 사용하여 보존 설정을 한 번에 여러 Amazon S3 객체에 추가할 수 있습니다. 자세한 내용은 S3 배치 작업을 사용하여 S3 객체 잠금 보관 날짜 설정을 참조하십시오.	2020년 5월 4일
S3 배치 작업에서 객체 잠금 법적 보존 지원	이제 S3 객체 잠금을 적용한 배치 작업을 사용하여 법적 보존을 한 번에 여러 Amazon S3 객체에 추가할 수 있습니다. 자세한 내용은 S3 객체 잠금 법적 보존 설정에 S3 배치 작업 사용을 참조하십시오.	2020년 5월 4일

S3 배치 작업에 대한 작업 태그	S3 배치 작업에 태그를 추가하여 해당 작업을 제어하고 레이블을 지정할 수 있습니다. 자세한 내용은 S3 배치 작업 작업 태그를 참조하십시오.	2020년 3월 16일
Amazon S3 액세스 포인트	Amazon S3 액세스 포인트는 S3의 공유 데이터 세트에 대한 대규모 데이터 액세스 관리를 간소화합니다. 액세스 포인트는 S3 객체 작업을 수행하는 데 사용할 수 있는 버킷에 연결된 네트워크 엔드포인트입니다. 자세한 내용을 알아보려면 Amazon S3 액세스 지점을 사용한 데이터 액세스 관리를 참조하십시오.	2019년 12월 2일
Access Analyzer for Amazon S3	Access Analyzer for Amazon S3은 인터넷상의 모든 사용자 또는 조직 외부의 계정을 포함한 다른 AWS 계정에 대한 액세스를 허용하도록 구성된 S3 버킷에 대한 알림을 제공합니다. 자세한 내용은 Access Analyzer for Amazon S3 사용을 참조하십시오.	2019년 12월 2일
S3 Replication Time Control(S3 RTC)	S3 Replication Time Control(S3 RTC)은 Amazon S3에 업로드하는 대부분의 객체를 몇 초 만에 복제하고 이러한 객체의 99.99%를 15분 내에 복제합니다. 자세한 내용은 S3 Replication Time Control(S3 RTC)을 사용하여 객체 복제를 참조하십시오.	2019년 11월 20일

동일 리전 복제	동일 리전 복제(SRR)를 사용하여 동일한 AWS 리전의 Amazon S3 버킷에서 객체를 복사할 수 있습니다. 교차 지역 복제(CRR) 및 동일 지역 복제에 대한 자세한 내용은 복제를 참조하십시오.	2019년 9월 18일
교차 리전 복제가 S3 객체 잠금 지원	교차 리전 복제가 이제 객체 잠금을 지원합니다. 자세한 내용은 Amazon S3는 무엇을 복제합니까?를 참조하십시오.	2019년 5월 28일
S3 배치 작업	S3 배치 작업을 사용하면 Amazon S3 객체에 대해 대규모 배치 작업을 수행할 수 있습니다. S3 배치 작업은 지정한 객체 목록에 대해 단일 작업을 실행할 수 있습니다. 단일 작업으로 엑사바이트 규모의 데이터가 포함된 수십억 개의 객체에서 지정된 작업을 수행할 수 있습니다. 자세한 내용은 S3 배치 작업 수행을 참조하십시오.	2019년 4월 30일
아시아 태평양(홍콩) 리전	이제 아시아 태평양(홍콩) 리전에서 Amazon S3을 사용할 수 있습니다. Amazon S3 리전 및 엔드포인트에 대한 자세한 내용은 AWS 일반 참조의 리전 및 엔드포인트를 참조하십시오.	2019년 4월 24일
서버 액세스 로그에 새 필드가 추가됨	Amazon S3은 서버 액세스 로그에 TLS(전송 계층 보안) 버전인 새 필드를 추가했습니다. 자세한 내용은 서버 액세스 로그 형식을 참조하세요.	2019년 3월 28일

새 아카이브 스토리지 클래스	Amazon S3은 이제 거의 액세스하지 않는 객체를 저장하기 위한 새로운 아카이브 스토리지 클래스인 S3 Glacier Deep Archive(DEEP_ARCHIVE)를 제공합니다. 자세한 내용은 스토리지 클래스를 참조하세요.	2019년 3월 27일
서버 액세스 로그에 새 필드가 추가됨	Amazon S3은 서버 액세스 로그에 호스트 ID, 서명 버전, 암호 모음, 인증 유형 및 호스트 헤더와 같은 새 필드를 추가했습니다. 자세한 내용은 서버 액세스 로그 형식을 참조하십시오.	2019년 3월 5일
Parquet 형식 Amazon S3 인벤토리 파일 지원	이제 Amazon S3은 인벤토리 출력 파일에 대해 Apache ORC(Optimized Row Columnar) 및 쉼표로 구분된 값(CSV) 파일 형식 이외에 Apache Parquet(Parquet) 형식도 지원합니다. 자세한 내용은 인벤토리를 참조하십시오.	2018년 12월 4일
S3 객체 잠금	Amazon S3이 이제 Amazon S3 객체에 대한 Write Once Read Many 보호를 제공하는 객체 잠금 기능을 제공합니다. 자세한 내용은 객체 잠금을 참조하십시오.	2018년 11월 26일

복원 속도 업그레이드	Amazon S3 복원 속도 업그레이드를 사용하면 복원이 진행되는 동안 S3 Glacier Flexible Retrieval 스토리지 클래스에서 더 빠른 속도로 복원 속도를 변경할 수 있습니다. 자세한 내용은 보관된 객체의 복원을 참조하십시오.	2018년 11월 26일
복원 이벤트 알림	Amazon S3 이벤트 알림은 이제 S3 Glacier Flexible Retrieval 스토리지 클래스에서 객체를 복원할 때 초기화 및 완료 이벤트를 지원합니다. 자세한 내용은 이벤트 알림을 참조하십시오.	2018년 11월 26일

S3 Glacier Flexible Retrieval 스토리지 클래스에 직접 PUT	Amazon S3 PUT 작업은 이제 객체를 생성할 때 S3 Glacier Flexible Retrieval을 스토리지 클래스로 지정하는 것을 지원합니다. 이전에는 다른 Amazon S3 스토리지 클래스에서 S3 Glacier Flexible Retrieval 스토리지 클래스로 객체를 전환해야 했습니다. 또한 S3 교차 리전 복제(CRR)를 사용할 때 이제 복제된 객체의 스토리지 클래스로 S3 Glacier Flexible Retrieval을 지정할 수 있습니다. S3 Glacier Flexible Retrieval 스토리지 클래스에 대한 자세한 내용은 스토리지 클래스를 참조하십시오. 복제된 객체에 대한 스토리지 클래스 지정에 대한 자세한 내용은 복제 구성 개요를 참조하십시오. S3 Glacier Flexible Retrieval REST API 변경 사항에 직접 PUT에 대한 자세한 내용은 문서 이력: S3 Glacier Flexible Retrieval에 직접 PUT을 참조하십시오.	2018년 11월 26일
새 스토리지 클래스	Amazon S3이 이제 변경되고 있거나 알 수 없는 액세스 패턴으로 수명이 긴 데이터용으로 설계된 S3 Intelligent-Tiering(`INTELLIGENT_TIERING`)이라는 새로운 스토리지 클래스를 제공합니다. 자세한 내용은 스토리지 클래스를 참조하십시오.	2018년 11월 26일

Amazon S3 퍼블릭 액세스 차단	Amazon S3에 이제 버킷 및 객체에 대한 퍼블릭 액세스를 버킷당 또는 계정 전체 기준으로 차단하는 기능이 포함됩니다. 자세한 내용은 Amazon S3 퍼블릭 액세스 차단 사용을 참조하십시오.	2018년 11월 15일
교차 리전 복제(CRR) 규칙에서 필터링 개선	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 개요를 참조하십시오.	2018년 9월 19일
새로운 Amazon S3 Select 기능	Amazon S3 Select는 이제 Apache Parquet 입력, 중첩 JSON 객체에 대한 쿼리, 두 개의 새로운 Amazon CloudWatch 모니터링 지표(`SelectScannedBytes` 및 `SelectReturnedBytes`)를 지원합니다.	2018년 9월 5일
RSS에서 현재 사용 가능한 업데이트	이제 Amazon S3 사용 설명서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.	2018년 6월 19일

이전 업데이트

다음 표에서는 2018년 6월 19일 이전 Amazon S3 사용 설명서의 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 AWS SDK for .NET 개발자 안내서에서 Amazon Web Services Asynchronous APIs for .NET을 참조하십시오. 코드 예제는 이제 AWS SDK for .NET 버전 3과 호환됩니다. • Java – 클라이언트 빌더 모델을 사용하도록 모든 예제가 업데이트되었습니다. 클라이언트 빌더 모델에 대한 자세한 내용은 서비스 클라이언트 생성을 참조하십시오. • PHP – AWS SDK for PHP 3.0을 사용하도록 모든 예제가 업데이트되었습니다. AWS SDK for PHP 3.0에 대한 자세한 내용은 AWS SDK for PHP 섹션을 참조하십시오. • Ruby – AWS SDK for Ruby 버전 3과 작동하도록 예제 코드가 업데이트되었습니다.	2018년 4월 30일
Amazon S3가 이제 S3 Glacier Flexible Retrieval 및 `ONEZONE_IA` 스토리지 클래스를 Amazon CloudWatch Logs 스토리지 지표에 보고함	이러한 스토리지 지표는 실제 바이트 수를 보고하는 것 이외에 적용 가능한 스토리지 클래스(`ONEZONE_IA` , `STANDARD_IA` 및 S3 Glacier Flexible Retrieval)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 이해 및 관리 섹션을 참조하세요. •	2018년 30월 4일

변경 사항	설명	날짜
	S3 Glacier Flexible Retrieval 스토리지 클래스 객체의 경우 스토리지 지표는 다음 오버헤드를 보고합니다. • 객체당 32KB 오버헤드(S3 Glacier Flexible Retrieval 스토리지 클래스 요금으로 청구됨) • 객체당 8KB 오버헤드(STANDARD 스토리지 클래스 요금으로 청구됨) 자세한 내용은 Amazon S3 수명 주기를 사용하여 객체 전환 섹션을 참조하세요. 스토리지 측정치에 대한 자세한 내용은 Amazon CloudWatch를 사용한 지표 모니터링 단원을 참조하십시오.	
새 스토리지 클래스	Amazon S3가 이제 객체 저장을 위한 새 스토리지 클래스인 STANDARD_IA (IA는 자주 액세스하지 않음을 의미)를 제공합니다. 이 스토리지 클래스는 수명이 길고 액세스 빈도가 낮은 데이터에 최적화되었습니다. 자세한 내용은 Amazon S3 스토리지 클래스 이해 및 관리 섹션을 참조하세요.	2018년 4월 4일
Amazon S3 Select	a>Amazon S3가 이제 SQL 표현식에 기초하여 객체 콘텐츠를 검색하는 기능을 지원합니다. 자세한 내용은 Amazon S3 Select를 사용하여 데이터 쿼리 적용 섹션을 참조하세요.	2018년 4월 4일

변경 사항	설명	날짜
아시아 태평양(오사카-로컬) 리전	Amazon S3을 이제 아시아 태평양 (오사카-로컬) 리전에서 사용할 수 있습니다. Amazon S3 리전 및 엔드포인트에 대한 자세한 내용은 AWS 일반 참조의 리전 및 엔드포인트를 참조하세요. **Important** 아시아 태평양(오사카-로컬) 리전은 반드시 아시아 태평양(도쿄) 리전과 함께 사용해야 합니다. 아시아 태평양(오사카-로컬) 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	Amazon S3을 이제 유럽(파리) 리전에서 사용할 수 있습니다. Amazon S3 리전 및 엔드포인트에 대한 자세한 내용은 AWS 일반 참조의 리전 및 엔드포인트를 참조하세요.	2017년 12월 18일
중국(닝샤) 리전	Amazon S3를 이제 중국(닝샤) 리전에서 사용할 수 있습니다 Amazon S3 리전 및 엔드포인트에 대한 자세한 내용은 AWS 일반 참조의 리전 및 엔드포인트를 참조하세요.	2017년 11월 29일
ORC 형식 Amazon S3 인벤토리 파일 지원	Amazon S3은 이제 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 그리고 Presto, Apache Hive 및 Apache Spark와 같은 기타 도구를 사용하여 표준 SQL을 통해 Amazon S3 인벤토리에 쿼리할 수 있습니다. 자세한 내용은 S3 Inventory를 사용한 데이터 카탈로그화 및 분석 섹션을 참조하세요.	2017년 11월 17일

변경 사항	설명	날짜
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. 객체는 Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) 또는 AWS 관리형 키(SSE-KMS)를 통한 서버 측 암호화를 사용하여 암호화됩니다. 자세한 내용은 Amazon S3 버킷에 대한 기본 서버 측 암호화 동작 설정 섹션을 참조하세요.	2017년 11월 06일
Amazon S3 인벤토리의 암호화 상태	Amazon S3가 이제 Amazon S3 인벤토리의 암호화 상태를 지원하므로 규정 준수 감사 또는 기타 목적을 위해 유휴 시 객체가 암호화되는 방식을 확인할 수 있습니다. 또한 모든 인벤토리 파일이 적절히 암호화되도록 SSE(서버 측 암호화) 또는 SSE-KMS를 사용하여 Amazon S3 인벤토리를 암호화하도록 구성할 수 있습니다. 자세한 내용은 S3 Inventory를 사용한 데이터 카탈로그화 및 분석 섹션을 참조하세요.	2017년 11월 06일
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 섹션을 참조하세요. • 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 암호화(SSE-S3, SSE-KMS, DSSE-KMS, SSE-C)된 객체 복제 섹션을 참조하세요.	2017년 11월 06일
유럽(런던) 리전	Amazon S3을 이제 유럽(런던) 리전에서 사용할 수 있습니다. Amazon S3 리전 및 엔드포인트에 대한 자세한 내용은 AWS 일반 참조의 리전 및 엔드포인트를 참조하세요.	2016년 13월 12일

변경 사항	설명	날짜
캐나다(중부) 리전	Amazon S3을 이제 캐나다(중부) 리전에서 사용할 수 있습니다. Amazon S3 리전 및 엔드포인트에 대한 자세한 내용은 AWS 일반 참조의 리전 및 엔드포인트를 참조하세요.	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. • CRR(교차 리전 복제)을 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 필터를 통해 정보를 표시할 수도 있습니다. 자세한 내용은 태그를 사용하여 스토리지 분류 섹션을 참조하세요.	2016년 11월 29일

변경 사항	설명	날짜
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 섹션을 참조하세요.	2016년 11월 29일
버킷에 대한 CloudWatch 요청 지표	Amazon S3가 이제 버킷에 대한 요청에서 CloudWatch 지표를 지원합니다. 버킷에 대한 이러한 지표를 사용 설정하면, 지표는 1분 간격으로 보고됩니다. 버킷 내 어떤 객체가 이러한 요청 지표를 보고할지 구성할 수도 있습니다. 자세한 내용은 Amazon CloudWatch를 사용한 지표 모니터링 섹션을 참조하세요.	2016년 11월 29일
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 S3 Inventory를 사용한 데이터 카탈로그화 및 분석 섹션을 참조하세요.	2016년 11월 29일
Amazon S3 분석 - 스토리지 클래스 분석	새로운 Amazon S3 분석 – 스토리지 클래스 분석 기능은 데이터 액세스 패턴을 관찰해 자주 액세스하지 않는 STANDARD 스토리지를 STANDARD_IA (IA는 자주 액세스하지 않음을 의미) 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 구성을 개선할 수 있습니다. 또한 이 기능은 S3 버킷으로 내보낼 수 있는 지정된 버킷, 접두사 또는 태그 레벨에서의 스토리지 사용에 대한 상세한 일일 분석도 포함합니다.	2016년 11월 29일

변경 사항	설명	날짜
S3 Glacier에 저장된 객체를 복원할 때 새로운 신속 처리 및 벌크 데이터 검색	Amazon S3가 이제 S3 Glacier에 저장된 객체를 복원할 때 표준 검색 외에 신속 처리 및 벌크 데이터 검색도 지원합니다. 자세한 내용은 아카이브된 객체 복원 섹션을 참조하세요.	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 섹션을 참조하세요.	2016년 11월 21일
미국 동부(오하이오) 리전	Amazon S3을 이제 미국 동부(오하이오) 리전에서 사용할 수 있습니다. Amazon S3 리전 및 엔드포인트에 대한 자세한 내용은 AWS 일반 참조의 리전 및 엔드포인트를 참조하세요.	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 섹션을 참조하세요.	2016년 10월 6일
IPv6 지원	Amazon S3가 이제 IPv6(인터넷 프로토콜 버전 6)을 지원합니다. 듀얼 스택 엔드포인트를 사용하여 IPv6을 통해 Amazon S3에 액세스할 수 있습니다. 자세한 내용은 Amazon S3 API 참조에서 IPv6을 통해 Amazon S3에 요청을 참조하세요.	2016년 8월 11일
아시아 태평양(뭄바이) 리전	Amazon S3를 이제 아시아 태평양(뭄바이) 리전에서 사용할 수 있습니다. Amazon S3 리전 및 엔드포인트에 대한 자세한 내용은 AWS 일반 참조의 리전 및 엔드포인트를 참조하세요.	2016년 6월 27일

변경 사항	설명	날짜
Amazon S3 Transfer Acceleration	Amazon S3 Transfer Acceleration을 사용하면 클라이언트와 S3 버킷 간에 파일을 빠르고 쉽고 안전하게 장거리 전송할 수 있습니다. Transfer Acceleration은 Amazon CloudFront에서 전 세계에 분산된 엣지 로케이션을 활용합니다. 자세한 내용은 Amazon S3 Transfer Acceleration을 사용하여 빠르고 안전한 파일 전송 구성 섹션을 참조하세요.	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 S3에 지시할 수 있습니다. 자세한 내용은 수명 주기 작업을 설명할 요소 섹션을 참조하세요.	2016년 3월 16일

변경 사항	설명	날짜
이제 버킷 수명 주기 구성은 미완료 멀티파트 업로드 중지 작업을 지원합니다	이제 버킷 수명 주기 구성은 시작된 후 지정 일수 내에 완료되지 않은 멀티파트 업로드를 중지하도록 Amazon S3에 지시하는 데 사용할 수 있는 `AbortIncompleteMultipartUpload` 작업을 지원합니다. 멀티파트 업로드가 중지 작업을 할 수 있는 조건을 갖춘 경우, Amazon S3는 업로드된 부분을 제거하고 멀티파트 업로드를 중지합니다. 개념에 대한 설명은 Amazon S3 사용 설명서에서 다음 주제를 참조하십시오. • 멀티파트 업로드 중단 • 수명 주기 작업을 설명할 요소 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 `x-amz-abort-date` 작업을 지정하는 수명 주기 규칙이 있는 경우, 2개의 응답 헤더(`x-amz-abort-rule-id` 및 `AbortIncompleteMultipartUpload`)를 추가로 반환합니다. 이 응답 헤더들은 시작된 멀티파트 업로드가 중지 작업을 할 수 있는 조건이 되는 시점 및 적용 가능한 수명 주기 규칙을 알려줍니다.	2016년 3월 16일

변경 사항	설명	날짜
아시아 태평양(서울) 리전	Amazon S3을 이제 아시아 태평양(서울) 리전에서 사용할 수 있습니다. Amazon S3 리전 및 엔드포인트에 대한 자세한 내용은 AWS 일반 참조의 리전 및 엔드포인트를 참조하십시오.	2016년 1월 6일
새로운 조건 키 및 멀티파트 업로드 변경	IAM 정책이 이제 Amazon S3 `s3:x-amz-storage-class` 조건 키를 지원합니다. 자세한 내용은 조건 키를 사용한 버킷 정책 예시 섹션을 참조하세요. 이제 파트를 업로드하고 업로드를 완료할 목적으로 직접 멀티파트 업로드를 시작할 필요가 없습니다. 자세한 내용은 멀티파트 업로드 API 및 권한 섹션을 참조하세요.	2015년 12월 14일
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 이해 및 관리 섹션을 참조하세요. 이제 수명 주기 구성 기능 업데이트를 사용하여 객체를 STANDARD_IA 스토리지 클래스로 전환할 수 있습니다. 자세한 내용은 객체 수명 주기 관리 섹션을 참조하세요. 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 리전 내 및 리전 간 객체 복제 섹션을 참조하세요.	2015년 9월 16일

변경 사항	설명	날짜
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 구성 변경을 추적할 수 있습니다. 자세한 내용은 AWS CloudTrail을 사용하여 Amazon S3 API 호출 로깅 섹션을 참조하세요.	2015년 9월 1일
버킷 한도 향상	Amazon S3이 이제 버킷 한도 향상을 지원합니다. 기본적으로 고객은 AWS 계정에서 최대 100개의 버킷을 생성할 수 있습니다. 추가로 버킷이 필요한 고객은 서비스 한도 향상을 제출하여 버킷 한도를 늘릴 수 있습니다. 버킷 한도를 늘리는 방법에 대한 자세한 내용은 AWS 일반 참조의 AWS 서비스 할당량을 참조하십시오. 자세한 내용은 AWS SDK 사용 및 범용 버킷 할당량, 제한 및 규제(을)를 참조하세요.	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 리전 섹션을 참조하세요.	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 섹션을 참조하세요.	2015년 7월 28일

변경 사항	설명	날짜
Amazon CloudWatch 통합	새로운 Amazon CloudWatch 통합을 통해 Amazon S3에 대한 CloudWatch 지표를 통해 Amazon S3 사용량을 모니터링하고 경보를 설정할 수 있습니다. 지원되는 지표에는 Standard 스토리지의 총 바이트 수, Reduced-Redundancy Storage(RRS)의 총 바이트 수, 특정 S3 버킷의 총 객체 수가 포함됩니다. 자세한 내용은 Amazon CloudWatch를 사용한 지표 모니터링 섹션을 참조하세요.	2015년 7월 28일
비어 있지 않은 버킷의 삭제 및 비우기 지원	Amazon S3에서는 이제 비어 있지 않은 버킷의 삭제와 비우기가 지원됩니다. 자세한 내용은 범용 버킷 비우기 섹션을 참조하세요.	2015년 7월 16일
Amazon VPC 종단점의 버킷 정책	Amazon S3이 가상 프라이빗 클라우드(VPC) 엔드포인트에 대한 버킷 정책 지원을 추가했습니다. S3 버킷 정책을 사용하여 특정 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 버킷 정책을 사용하여 VPC 엔드포인트에서 액세스 제어 섹션을 참조하세요.	2015년 4월 29일
이벤트 알림	AWS Lambda 함수에 대해 리소스 기반 권한으로 전환하는 작업을 지원하기 위해 Amazon S3 이벤트 알림이 업데이트되었습니다. 자세한 내용은 Amazon S3 이벤트 알림 섹션을 참조하세요.	2015년 4월 9일
리전 간 복제	Amazon S3이 이제 교차 리전 복제를 지원합니다. 교차 리전 복제는 서로 다른 AWS 리전의 버킷 간 객체를 비동기식으로 자동 복사하는 것을 말합니다. 자세한 내용은 리전 내 및 리전 간 객체 복제 섹션을 참조하세요.	2015년 3월 24일

변경 사항	설명	날짜
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 알림을 참조하십시오.	2014년 11월 13일
고객 제공 암호화 키를 사용한 서버 측 암호화	AWS Key Management Service(AWS KMS) 키(SSE-KMS)를 사용한 서버 측 암호화 Amazon S3에서는 이제 AWS KMS을(를) 사용한 서버 측 암호화가 지원됩니다. 이 기능을 사용하면 AWS KMS을(를) 통해 엔벌로프 키를 관리할 수 있습니다. Amazon S3는 AWS KMS을(를) 호출하여 사용자가 설정한 권한 내에서 엔벌로프 키에 액세스합니다. AWS KMS를 사용한 서버 측 암호화에 대한 자세한 내용은 AWS Key Management Service를 사용한 서버 측 암호화를 사용하여 데이터 보호를 참조하십시오.	2014년 11월 12일
유럽(프랑크푸르트) 리전	Amazon S3를 이제 유럽(프랑크푸르트) 리전에서 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 서버 측 암호화(고객 제공 암호화 키 사용)를 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 사용 설정된 버킷 모두에서 수명 주기를 구성할 수 있습니다. 자세한 내용은 객체 수명 주기 관리 섹션을 참조하세요.	2014년 5월 20일
액세스 제어 주제 개정	Amazon S3 액세스 제어 설명서가 개정되었습니다. 자세한 내용은 Amazon S3의 ID 및 액세스 관리 섹션을 참조하세요.	2014년 4월 15일
서버 액세스 로깅 주제 개정	서버 액세스 로깅 설명서가 개정되었습니다. 자세한 내용은 서버 액세스 로깅을 사용한 요청 로깅 섹션을 참조하세요.	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 정책 언어가 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 Amazon S3의 ID 기반 정책 예시를 참조하십시오.	2013년 4월 3일
요청자 지불에 대한 콘솔 지원	이제 Amazon S3 콘솔을 사용하여 요청자 지불에 대한 버킷을 구성할 수 있습니다. 자세한 내용은 스토리지 전송 및 사용량에 대한 요청자 지불 범용 버킷 사용 섹션을 참조하세요.	2012년 12월 31일

변경 사항	설명	날짜
웹 사이트 호스팅에 대한 루트 도메인 지원	Amazon S3에서는 이제 루트 도메인에서의 정적 웹 사이트 호스팅이 지원됩니다. 웹 사이트 방문자는 웹 주소에 www를 지정하지 않고(예: www.example.com 대신 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 도메인 요청을 프록시할 수 있었습니다. 요청을 프록시하기 위해 웹 서버를 실행하면 추가적인 비용, 운영 부담 및 또 다른 잠재적 장애 지점이 생깁니다. 이제는 www와 루트 도메인 주소 모두에 대해 Amazon S3의 고가용성 및 내구성을 활용할 수 있습니다. 자세한 내용은 Amazon S3를 사용하여 정적 웹 사이트 호스팅 섹션을 참조하세요.	2012년 12월 27일
콘솔 개정	Amazon S3 콘솔이 업데이트되었습니다. 콘솔을 참조하는 설명서 주제도 이에 따라 개정되었습니다.	2012년 12월 14일
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 기간을 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 섹션을 참조하세요.	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 섹션을 참조하세요. 웹 사이트 호스팅에 대한 자세한 내용은 Amazon S3를 사용하여 정적 웹 사이트 호스팅을 참조하십시오.	2012년 10월 4일

변경 사항	설명	날짜
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 빌드하고 Amazon S3 리소스에 대한 교차 도메인 액세스를 선택적으로 허용할 수 있습니다. 자세한 내용은 교차 오리진 리소스 공유(CORS) 사용 섹션을 참조하세요.	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 사용을 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 MFA 필요를 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

변경 사항	설명	날짜
새로운 리전 지원	Amazon S3에서 이제 남아메리카(상파울루) 리전을 지원합니다. 자세한 내용은 Amazon S3 범용 버킷 액세스 섹션을 참조하세요.	2011년 12월 14일
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 Amazon S3 객체 삭제 섹션을 참조하세요.	2011년 12월 7일
새로운 리전 지원	Amazon S3에서 이제 미국 서부(오레곤) 리전을 지원합니다. 자세한 내용은 버킷 및 리전을 참조하세요.	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. • AWS SDK for PHP 및 AWS SDK for Ruby(Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3) 지정 참조)를 사용하는 새로운 서버 측 암호화 섹션	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(를) 사용하여 객체를 업로드할 때도 서버 측 암호화를 요청할 수 있습니다. 데이터 암호화에 대해 자세히 알아보려면 데이터 암호화 사용을 참조하십시오.	2011년 10월 4일

변경 사항	설명	날짜
설명서 업데이트	이 릴리스에는 설명서 버그 수정 사항 외에 다음과 같은 향상된 내용도 포함되어 있습니다. • Amazon S3 API 참조 섹션의 요청에 Ruby 및 PHP 샘플을 추가했습니다. • 미리 서명된 URL을 생성하고 사용하는 방법에 대해 설명하는 섹션을 추가했습니다. 자세한 정보는 미리 서명된 URL을 통해 객체 공유 및 미리 서명된 URL을 통해 객체 공유 섹션을 참조하십시오. • Eclipse 및 Visual Studio용 AWS Explorer를 소개하기 위해 기존 단원을 업데이트했습니다. 자세한 내용은 Amazon S3 API 참조의 AWS SDK를 사용하여 Amazon S3에서 개발을 참조하세요.	2011년 9월 22일
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 Amazon S3 API 참조에서 요청을 참조하세요. 임시 보안 자격 증명에 대한 IAM 지원에 관한 자세한 내용은 IAM 사용 설명서의 임시 보안 자격 증명을 참조하십시오.	2011년 8월 3일

변경 사항	설명	날짜
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, 이동, 이름 변경 섹션을 참조하세요. 멀티파트 업로드 API에 대한 개념적 정보는 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사를 참조하십시오.	2011년 6월 21일
HTTP를 통한 SOAP API 호출 사용 중지	보안을 높이기 위해 HTTP를 통한 SOAP API 호출이 사용 중지되었습니다. 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 IAM 사용자에게 권한을 위임하기 위한 역할 생성을 참조하십시오. 버킷 정책에 보안 주체 지정에 대한 자세한 내용은 버킷 정책의 보안 주체을 참조하십시오.	2011년 6월 6일
새로운 링크	이제 이 서비스의 엔드포인트 정보는 AWS 일반 참조에서 찾을 수 있습니다. 자세한 내용은 https://docs.aws.amazon.com/general/latest/gr/rande.htmlAWS 일반 참조의 리전 및 엔드포인트를 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 S3를 사용하여 정적 웹 사이트 호스팅 섹션을 참조하세요.	2011년 6월 6일
이제 이 서비스의 엔드포인트 정보는 AWS 일반 참조에서 찾을 수 있습니다. 자세한 내용은 https://docs.aws.amazon.com/general/latest/gr/rande.htmlAWS 일반 참조의 리전 및 엔드포인트를 참조하십시오.	2011년 3월 1일	
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 S3를 사용하여 정적 웹 사이트 호스팅 섹션을 참조하세요.	2011년 17월 2일

변경 사항	설명	날짜
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 다운로드 섹션을 참조하세요.	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 5GB의 객체를 업로드할 수 있습니다. 더 큰 객체의 경우 멀티파트 업로드 REST API를 사용하여 객체를 여러 부분으로 나누어 업로드해야 합니다. 자세한 내용은 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 섹션을 참조하세요.	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 Amazon S3에서 멀티파트 업로드를 사용한 객체 업로드 및 복사 섹션을 참조하세요.	2010년 11월 10일
버킷 정책에서의 정식 ID 지원	이제 버킷 정책에서 정식 ID를 지정할 수 있습니다. 자세한 내용은 버킷 정책의 보안 주체 섹션을 참조하세요.	2010년 9월 17일
Amazon S3가 IAM과 함께 작동함	이 서비스는 이제 AWS Identity and Access Management(IAM)와 통합됩니다. 자세한 내용은 IAM 사용 설명서의 IAM으로 작업하는 AWS 서비스 서비스를 참조하십시오.	2010년 9월 2일
알림	Amazon S3 알림 기능을 사용하면 Amazon S3에서 버킷에 대한 주요 이벤트를 감지할 때 Amazon S3에서 Amazon Simple Notification Service(Amazon SNS) 주제에 메시지를 게시하도록 버킷을 구성할 수 있습니다. 자세한 내용은 버킷 이벤트의 알림 설정을 참조하십시오.	2010년 7월 14일

변경 사항	설명	날짜
버킷 정책	버킷 정책은 버킷, 객체 및 객체 집합에 대한 액세스 권한을 설정하기 위해 사용하는 액세스 관리 시스템입니다. 이 기능은 액세스 통제 목록을 보완하며, 많은 경우 이를 대체합니다. 자세한 내용은 Amazon S3의 버킷 정책 섹션을 참조하세요.	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 가상 호스팅을 참조하십시오.	2010년 6월 9일
유럽(아일랜드)에 대한 새로운 엔드포인트	Amazon S3에서는 이제 유럽(아일랜드)에 대한 `http://s3-eu-west-1.amazonaws.com` 엔드포인트를 제공합니다.	2010년 6월 9일
콘솔	이제 AWS Management Console을 통해 Amazon S3를 사용할 수 있습니다. 콘솔의 모든 Amazon S3 기능에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서에서 확인할 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 Storage를 참조하십시오.	2010년 5월 12일
새로운 리전 지원	Amazon S3에서 이제 아시아 태평양(싱가포르) 리전을 지원합니다. 자세한 내용은 Buckets and Regions를 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 사용 설정하면 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 및 버전 관리 사용을 참조하십시오.	2010년 2월 8일

변경 사항	설명	날짜
새로운 리전 지원	Amazon S3에서 이제 미국 서부(캘리포니아 북부) 리전을 지원합니다. 이 리전에 대한 요청의 새로운 엔드포인트는 `s3-us-west-1.amazonaws.com` 입니다. 자세한 내용은 버킷 및 리전을 참조하십시오.	2009년 12월 2일
AWS SDK for .NET	AWS에서는 이제 REST나 SOAP 대신 .NET 언어별 API 작업을 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 Amazon S3 API 참조의 AWS SDK를 사용하여 Amazon S3에서 개발을 참조하세요.	2009년 11월 11일