aws

개발자 가이드

Amazon Glacier

API 버전 2012-06-01

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

Amazon Glacier: 개발자 가이드

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

Amazon 상표 및 트레이드 드레스는 Amazon 외 제품 또는 서비스와 함께 사용되어서는 안되며, 고객에게 혼동을 일으키거나 Amazon 브랜드 이미지를 떨어뜨리고 폄하하는 방식으로 이용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon과 제휴 관계이거나 관련이 있거나 후원 관계와 관계없이 해당 소유자의 자산입니다.

Table of Contents

..... x
Amazon Glacier란 무엇인가요? 1
현재 Amazon Glacier를 사용하세요? 1
데이터 모델 3
볼트 3
아카이브 4
작업 4
알림 구성 5
지원되는 작업 6
볼트 작업 6
아카이브 작업 6
작업 7
Amazon Glacier 액세스 7
리전 및 엔드포인트 7
시작하기 8
1단계: 시작하기 전에 9
설정 AWS 계정 9
적절한 AWS SDK 다운로드 10
2단계: 볼트 생성 11
3단계: 아카이브를 볼트에 업로드 12
Java를 사용하여 아카이브 업로드 13
.NET을 사용하여 아카이브 업로드 19
4단계: 아카이브를 볼트에서 다운로드 20
Java를 사용하여 아카이브 다운로드 21
.NET을 사용하여 아카이브 다운로드 23
5단계: 볼트에서 아카이브 삭제 25
관련 섹션 26
Java를 사용하여 아카이브 삭제 26
.NET을 사용하여 아카이브 삭제 27
를 사용하여 아카이브 삭제 AWS CLI 28
6단계: 볼트 삭제 31
추가 정보 32
볼트 작업 33
Amazon Glacier 내의 볼트 작업 34

볼트 생성 및 삭제 34
볼트 메타데이터 가져오기 34
볼트 인벤토리 다운로드 34
볼트 알림 구성 35
볼트 생성 35
Java를 사용하여 볼트 생성 36
.NET을 사용하여 볼트 생성 39
REST를 사용하여 볼트 생성 44
콘솔을 사용하여 볼트 생성 44
AWS CLI를 사용하여 볼트 생성 44
볼트 메타데이터 검색 45
Java를 사용하여 볼트 메타데이터 검색 46
.NET을 사용하여 볼트 메타데이터 검색 48
REST를 사용하여 볼트 메타데이터 검색 51
AWS CLI를 사용하여 볼트 메타데이터 검색 51
볼트 인벤토리 다운로드 52
인벤토리 정보 54
Java를 사용하여 볼트 인벤토리 다운로드 54
.NET을 사용하여 볼트 인벤토리 다운로드 62
REST를 사용하여 볼트 인벤토리 다운로드 69
를 사용하여 볼트 인벤토리 다운로드 AWS CLI 70
볼트 알림 구성 72
일반 개념 73
Java를 사용하여 볼트 알림 구성 75
.NET를 사용하여 볼트 알림 구성 77
REST API를 사용하여 볼트 알림 구성 80
콘솔을 사용하여 볼트 알림 구성 80
CLI를 사용하여 볼트 알림 구성 82
볼트 삭제 84
Java를 사용하여 볼트 삭제 85
.NET을 사용하여 볼트 삭제 86
REST를 사용하여 볼트 삭제 87
콘솔을 사용하여 빈 볼트 삭제 87
를 사용하여 볼트 삭제 AWS CLI 88
볼트의 태그 지정 92
Amazon Glacier 콘솔을 사용하여 볼트에 태그 지정 92

AWS CLI를 사용하여 볼트에 태그를 지정 94
Amazon Glacier API를 사용하여 볼트에 태그 지정 95
관련 섹션 95
볼트 잠금 95
볼트 잠금 개요 95
API를 사용하여 볼트 잠금 97
CLI를 사용하여 볼트 잠금 98
콘솔을 사용하여 볼트 잠금 99
아카이브 작업 102
아카이브 작업 103
아카이브 업로드 103
아카이브 찾기 103
아카이브 다운로드 103
아카이브 삭제 104
아카이브 업데이트 104
클라이언트 측 아카이브 메타데이터 유지 104
아카이브 업로드 104
아카이브를 업로드하기 위한 옵션 105
아카이브의 단일 작업 업로드 106
대용량 아카이브를 여러 부분으로 나누어 업로드 116
아카이브 다운로드 133
아카이브 가져오기 134
Java를 사용하는 아카이브 다운로드 138
.NET을 사용하여 아카이브 다운로드 154
Python을 사용하여 대용량 아카이브 다운로드 170
REST를 사용하여 아카이브 다운로드 178
를 사용하여 아카이브 다운로드 AWS CLI 178
아카이브 삭제 182
Java를 사용하여 아카이브 삭제 183
.NET을 사용하여 아카이브 삭제 185
REST를 사용하여 아카이브 삭제 188
를 사용하여 아카이브 삭제 AWS CLI 188
AWS SDKs 사용 191
AWS Java 및 .NET용 SDK 라이브러리 191
로우레벨 API가 무엇입니까? 191
하이레벨 API가 무엇입니까? 192

하이레벨 API와 로우레벨 API의 사용 시기 192
AWS SDKs 작업 192
사용 AWS SDK for Java 194
로우레벨 API 사용 194
하이레벨 API 사용 195
Eclipse를 사용하여 Java 예 실행 196
엔드포인트 설정 196
AWS SDK for .NET 사용 197
로우레벨 API 사용 198
하이레벨 API 사용 199
.NET 예제 실행 199
엔드포인트 설정 200
코드 예제 201
기본 사항 202
Amazon Glacier 시작 202
작업 204
시나리오 269
파일을 보관하고, 알림을 받고 작업을 시작 269
아카이브 콘텐츠 가져오기 및 아카이브 삭제 275
보안 281
데이터 보호 281
데이터 암호화 282
키 관리 283
인터네트워크 트래픽 개인 정보 283
자격 증명 및 액세스 관리 283
대상 284
ID를 통한 인증 284
정책을 사용하여 액세스 관리 286
Amazon Glacier가 IAM에서 작동하는 방식 287
ID 기반 정책 예시 293
리소스 기반 정책 예제 299
문제 해결 301
Amazon Glacier API 권한 참조 303
로깅 및 모니터링 311
규정 준수 확인 312
복원성 314

인프라 보안 315
VPC 엔드포인트 315
데이터 검색 정책 316
Amazon Glacier 데이터 검색 정책 선택 316
프리 티어만 정책 317
최대 검색 속도 정책 317
검색 제한 없음 정책 318
Amazon Glacier 콘솔을 사용하여 데이터 검색 정책 설정 318
Amazon Glacier API를 사용하여 데이터 검색 정책 설정 319
Amazon Glacier REST API를 사용하여 데이터 검색 정책 설정 319
AWS SDKs를 사용하여 데이터 검색 정책 설정 319
리소스 태그 지정 320
태그 지정 관련 기본 사항 320
태그 제한 321
태그 지정을 사용하여 비용 추적 321
태그 지정을 이용한 액세스 제어 관리 321
관련 섹션 322
를 사용한 감사 로깅 AWS CloudTrail 323
CloudTrail의 Amazon Glacier 정보 323
Amazon Glacier 로그 파일 항목 이해 324
API 참조 328
공통 요청 헤더 329
공통 응답 헤더 332
요청에 서명하기 332
서명 계산 예시 333
스트리밍 작업을 위한 서명 계산 335
체크섬 계산 337
트리-해시 예제 1: 아카이브의 단일 요청 업로드 338
트리-해시 예제 2: 아카이브의 멀티파트 업로드 339
파일의 트리-해시 계산 340
데이터 다운로드 시 체크섬 수신 350
오류 응답 352
예제 1: 존재하지 않는 작업 ID를 사용한 작업 요청 설명 355
예제 2: 요청 파라미터에 잘못된 값이 포함된 작업 목록 조회 요청 356
볼트 작업 357
볼트 잠금 중단 358

볼트에 태그 추가 ... 361
볼트 만들기 ... 364
볼트 잠금 완료 ... 367
볼트 삭제 ... 370
볼트 액세스 정책 삭제 ... 373
볼트 알림 삭제 ... 375
볼트 설명 ... 378
볼트 액세스 정책 가져오기 ... 382
볼트 잠금 가져오기 ... 385
볼트 알림 가져오기 ... 390
볼트 잠금 시작 ... 393
볼트의 태그 목록 조회 ... 397
볼트 목록 조회 ... 400
볼트에서 태그 삭제 ... 407
볼트 액세스 정책 설정 ... 410
볼트 알림 구성 설정 ... 413
아카이브 작업 ... 417
아카이브 삭제 ... 418
아카이브 업로드 ... 420
멀티파트 업로드 작업 ... 426
멀티파트 업로드 중단 ... 426
멀티파트 업로드 완료 ... 429
멀티파트 업로드 시작 ... 433
부분 목록 조회 ... 438
멀티파트 업로드 목록 조회 ... 445
파트 업로드 ... 452
작업 ... 458
작업 설명 ... 458
작업 출력 가져오기 ... 468
작업 시작 ... 478
작업 목록 조회 ... 489
작업에서 사용되는 데이터 형식 ... 499
CSVInput ... 500
CSVOutput ... 501
암호화 ... 502
GlacierJobDescription ... 503

권한 부여 507
피부여자 507
InputSerialization 508
InventoryRetrievalJobInput 509
jobParameters 510
OutputLocation 513
OutputSerialization 514
S3Location 514
SelectParameters 516
데이터 가져오기 작업 517
데이터 가져오기 정책 가져오기 517
프로비저닝된 용량 나열 521
프로비저닝된 용량 구매 524
데이터 가져오기 정책 설정 527
문서 기록 533
이전 업데이트 534
AWS 용어집 536

이 페이지는 볼트와 2012년부터 원래 REST API를 사용하는 Amazon Glacier 서비스의 기존 고객만 사용할 수 있습니다.

아카이브 스토리지 솔루션을 찾고 있다면 Amazon S3의 Amazon Glacier 스토리지 클래스, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive를 사용하는 것이 좋습니다. 이러한 스토리지 옵션에 대한 자세한 내용은 Amazon Glacier 스토리지 클래스를 참조하세요.

Amazon Glacier(기존 독립 실행형 볼트 기반 서비스)는 더 이상 신규 고객을 받지 않습니다. Amazon Glacier는 데이터를 볼트에 저장하고 Amazon S3 및 Amazon S3 Glacier 스토리지 클래스와 구별되는 자체 API를 갖춘 독립 실행형 서비스입니다. 기존 데이터는 Amazon Glacier에서 무기한으로 안전하게 보관되며 액세스 가능합니다. 마이그레이션은 필요하지 않습니다. 저비용 장기 아카이브 스토리지의 경우는 S3 버킷 기반 API, 전체 가용성, 저렴한 비용 및 서비스 통합을 통해 우수한 고객 경험을 제공하는 Amazon S3 Glacier 스토리지 클래스를 AWS 권장합니다. S3 APIs AWS 리전 AWS 향상된 기능을 원하는 경우 Amazon Glacier 볼트에서 Amazon S3 Glacier 스토리지 클래스로 데이터를 전송하기 위한AWS 솔루션 지침을 사용하여 Amazon S3 Glacier 스토리지 클래스로 마이그레이션하는 것이 좋습니다.

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

Amazon Glacier란 무엇인가요?

현재 Amazon Glacier 서비스를 사용 중이고 더 자세한 정보를 알고 싶다면 이 안내서에서 필요한 정보를 찾을 수 있습니다. Amazon Glacier는 볼트를 이용한 데이터 보관 및 장기 백업을 위한 안전하고 안정적이며 저렴한 서비스입니다. Amazon Glacier 서비스 요금에 대한 자세한 내용은 Amazon Glacier 요금을 참조하세요.

주제

- 현재 Amazon Glacier를 사용하세요?
- Amazon Glacier 데이터 모델
- Amazon Glacier에서 지원되는 작업
- Amazon Glacier 액세스

현재 Amazon Glacier를 사용하세요?

> **Note**
>
> 이 섹션은 Amazon Glacier 서비스에 관한 것입니다. 현재 Amazon Glacier 스토리지 클래스(S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval 및 S3 Glacier Deep Archive)를 사용하고 있다면 Amazon S3 사용 설명서의 객체 아카이빙을 위한 스토리지 클래스를 참조하세요.

현재 Amazon Glacier 서비스를 사용 중이고 더 자세한 정보를 알고 싶다면 먼저 다음 섹션을 읽을 것을 권장합니다.

- Amazon Glacier란?: 이 섹션의 나머지 부분에서는 기본 데이터 모델, 이 모델이 지원하는 작업, 이 서비스와 상호 작용하는 데 사용할 수 있는 AWS SDK를 설명합니다.
- 시작하기: Amazon Glacier 시작하기 섹션에서는 볼트를 생성하거나, 아카이브를 업로드하거나, 아카이브 다운로드 작업을 생성하거나, 작업 출력을 검색하거나, 아카이브를 삭제하는 프로세스에 대해서 살펴봅니다.

> **Important**
>
> Amazon Glacier는 콘솔을 제공합니다. 그러나 업로드, 다운로드 또는 삭제와 같은 아카이브 작업을 수행하려면 AWS Command Line Interface (AWS CLI) 또는 쓰기 코드를 사용해야 합니다. 콘솔은 아카이브 작업을 지원하지 않습니다. 예를 들어 사진, 비디오 및 기타 문서와 같은 데이터를 업로드하려면 REST API를 직접 AWS CLI 사용하거나 AWS SDKs를 사용하여 를 사용하거나 코드를 작성하여 요청해야 합니다.
> 를 설치하려면 단원을 AWS CLI참조하십시오AWS Command Line Interface. Amazon Glacier를 AWS CLI와 함께 사용하는 방법에 대한 자세한 내용은 Amazon Glacier에 대한 AWS CLI 참조를 참조하세요. 를 사용하여 Amazon Glacier AWS CLI 에 아카이브를 업로드하는 예제는 에서 Amazon Glacier 사용을 참조하세요 AWS Command Line Interface.

시작하기 섹션 이후 사용자는 Amazon Glacier 작업에 대한 자세한 내용 학습을 원할 것입니다. 다음 섹션에서는 REST API와 Java 및 Microsoft .NET용 AWS SDKs를 사용하여 Amazon Glacier 작업에 대한 자세한 정보를 제공합니다.

- Amazon Glacier에서 AWS SDKs 사용

  이 섹션에서는이 가이드의 다양한 코드 예제에 사용되는 AWS SDKs에 대한 개요를 제공합니다. 이 섹션을 살펴보면 다음 섹션을 이해하는 데 도움이 될 것입니다. 여기에는 이러한 SDK가 제공하는 하이레벨 및 로우레벨 API의 개요와 이러한 API의 사용 시기, 그리고 본 안내서에서 제공하는 코드 예제를 실행하기 위한 공통 단계 등이 언급되어 있습니다.

- Amazon Glacier에서의 볼트 작업

  이 섹션에서는 볼트를 생성하거나, 볼트 메타데이터를 검색하거나, 볼트 인벤토리를 검색하는 작업을 사용하거나, 볼트 알림을 구성하는 등 다양한 볼트 작업의 세부 정보를 제공합니다. Amazon Glacier 콘솔을 사용하는 것 외에도 다양한 볼트 작업에 AWS SDKs를 사용할 수 있습니다. 이 섹션에서는 API를 설명하고 AWS SDK for Java 및를 사용하여 작업 샘플을 제공합니다 AWS SDK for .NET.

- Amazon Glacier의 아카이브 작업

  이번 섹션에서는 아카이브를 단일 요청으로 업로드하거나 멀티파트 업로드 작업을 통해 대용량 아카이브를 여러 파트로 나누어 업로드하는 등 아카이브 작업 세부사항을 제공합니다. 또한 아카이브를 비동기식으로 다운로드하는 작업을 생성하는 방법에 대해서도 설명합니다. 이 섹션에서는 AWS SDK for Java 와 AWS SDK for .NET을 사용하여 예시를 제공합니다.

- Amazon Glacier를 위한 API 참조

Amazon Glacier는 RESTful 서비스입니다. 이 섹션에서는 구문을 비롯한 모든 작업의 요청 및 응답 예제를 포함하여 REST 작업에 대해서 설명합니다. AWS SDK 라이브러리는이 API를 래핑하여 프로그래밍 작업을 간소화합니다.

Amazon Glacier 데이터 모델

Amazon Glacier 데이터 모델 핵심 구성 요소에는 볼트와 아카이브가 포함됩니다. Amazon Glacier는 REST 기반 웹 서비스입니다. REST 관점에서 보면 볼트와 아카이브가 리소스에 해당합니다. 그 밖에 작업과 알림 구성 리소스도 Amazon Glacier 데이터 모델에 포함됩니다. 이러한 리소스들은 주요 리소스를 보완하는 역할을 합니다.

주제

- 볼트
- 아카이브
- 작업
- 알림 구성

볼트

Amazon Glacier에서 볼트는 아카이브 보관용 컨테이너입니다. 볼트는 Amazon S3 버킷과 유사합니다. 볼트를 생성할 때 이름을 지정하고 볼트를 AWS 리전 생성할을 선택합니다.

각 볼트 리소스에는 고유한 주소가 있습니다. 일반적인 형식은 다음과 같습니다.

```
https://region-specific-endpoint/account-id/vaults/vault-name
```

예를 들어 미국 서부(오레곤) 리전에 ID가 111122223333인 Vault(`examplevault`)를 생성한다고 가정합시다. 이 볼트는 다음 URI를 사용하여 주소를 지정할 수 있습니다.

```
https://glacier.us-west-2.amazonaws.com/111122223333/vaults/examplevault
```

URI의 다양한 구성 요소가 의미하는 바는 다음과 같습니다.

- `glacier.us-west-2.amazonaws.com`은 미국 서부(오레곤) 리전을 식별합니다.
- `111122223333`는 볼트를 소유한 AWS 계정 ID입니다.

- `vaults`는 AWS 계정이 소유한 볼트 집합을 말합니다.
- `examplevault`는 볼트 집합에서 특정 볼트를 구분하는 식별자입니다.

는 지원되는 모든에서 볼트를 생성할 AWS 계정 수 있습니다 AWS 리전. 지원되는 목록은 단원을 AWS 리전참조하십시오Amazon Glacier 액세스. 계정에서 사용하는 볼트 이름은 하나의 리전 내에서 고유해야 합니다. 는 서로 다른 리전에서 동일한 이름의 볼트를 생성할 AWS 계정 수 있습니다.

볼트에 저장할 수 있는 아카이브의 수는 제한이 없습니다. 비즈니스 또는 애플리케이션 요건에 따라 아카이브를 단일 볼트 또는 여러 볼트에 저장할 수 있습니다.

Amazon Glacier는 다양한 볼트 작업을 지원합니다. 볼트 작업은 리전에 따라 다릅니다. 예를 들어 볼트는 특정 리전에 생성됩니다. 볼트 목록을 요청하면 특정에서 요청 AWS 리전하고 결과 목록에는 해당 리전에서 생성된 볼트만 포함됩니다.

아카이브

아카이브는 사진, 동영상, 문서 등 모든 데이터가 될 수 있습니다. 아카이브는 Amazon S3 객체와 비슷하며 Amazon Glacier 스토리지의 기본 단위입니다. 각 아카이브는 고유 ID가 있으며 선택 사항으로 설명을 추가할 수 있습니다. 이 선택적 설명은 아카이브의 업로드 중에만 지정할 수 있습니다. Amazon Glacier는 아카이브가 저장되는에서 고유한 ID를 아카이브 AWS 리전 에 할당합니다.

각 아카이브마다 고유의 주소가 있습니다. 일반적인 형식은 다음과 같습니다.

```
https://region-specific-endpoint/account-id/vaults/vault-name/archives/archive-id
```

다음은 미국 서부(오레곤) 리전의 계정 111122223333의 `examplevault` 볼트에 저장된 아카이브 URI의 예시입니다.

```
https://glacier.us-west-2.amazonaws.com/111122223333/vaults/
examplevault/archives/NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-
TjhqG6eGoOY9Z8i1_AUyUsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchi
```

볼트에 저장할 수 있는 아카이브의 수는 제한이 없습니다.

작업

Amazon Glacier 작업은 아카이브를 검색하거나 볼트의 인벤토리를 가져올 수 있습니다.

Amazon Glacier에서 아카이브 및 볼트 인벤토리(아카이브 목록) 검색은 비동기식 작업이기 때문에 우선 작업을 시작하고 Amazon Glacier가 작업을 완료하면 작업 출력을 다운로드합니다.

> **Note**
>
> Amazon Glacier는 콜드 스토리지 데이터 보관 솔루션을 제공합니다. 실시간 데이터 가져오기를 지원하는 스토리지 솔루션이 필요하다면 Amazon S3를 사용하는 것이 좋습니다. 더 자세한 내용은 Amazon Simple Storage Service(S3)를 참조하세요.

볼트 인벤토리 작업을 시작하려면 볼트 이름을 입력합니다. 아카이브 검색 작업에는 볼트 이름과 아카이브 ID가 모두 필요합니다. 또한 작업을 식별할 수 있도록 옵션으로 작업 설명을 입력할 수도 있습니다.

아카이브 검색과 볼트 인벤토리 작업은 볼트와 연결됩니다. 볼트는 언제든지 다수의 작업이 진행 중일 수 있습니다. 작업 요청(작업 시작)을 전송하면 Amazon Glacier는 작업을 추적할 수 있는 작업 ID를 반환합니다. 각 작업은 다음과 같은 URI 형식으로 고유하게 식별됩니다.

```
https://region-specific-endpoint/account-id/vaults/vault-name/jobs/job-id
```

다음은 미국 서부(오리건) 리전의 계정 111122223333의 `examplevault` 볼트와 관련된 작업 예시입니다.

```
https://glacier.us-west-2.amazonaws.com/111122223333/vaults/examplevault/jobs/
HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
```

Amazon Glacier는 작업마다 작업 유형, 설명, 생성 날짜, 완료 날짜, 작업 상태 같은 정보를 유지합니다. 사용자는 특정 작업에 대한 정보를 가져오거나, 혹은 볼트와 연결된 모든 작업 목록을 가져올 수도 있습니다. Amazon Glacier가 반환하는 작업 목록에는 진행 중이거나 최근 완료된 작업이 모두 포함됩니다.

알림 구성

작업을 실행하는데 시간이 걸리기 때문에 Amazon Glacier는 작업이 완료되면 사용자가 알 수 있도록 알림 메커니즘을 지원합니다. 작업이 완료되면 Amazon Simple Notification Service(Amazon SNS) 토픽에 알림을 전송하도록 볼트를 구성할 수 있습니다. 알림 구성에서 Amazon SNS 토픽을 볼트마다 하나씩 지정할 수 있습니다.

Amazon Glacier는 알림 구성을 JSON 문서로 저장합니다. 다음은 볼트 알림 구성의 예제입니다.

```
{
```

```
    "Topic": "arn:aws:sns:us-west-2:111122223333:mytopic",
    "Events": ["ArchiveRetrievalCompleted", "InventoryRetrievalCompleted"]
}
```

알림 구성은 볼트와 연결되며, 각 볼트마다 하나씩 연결할 수 있습니다. 알림 구성 리소스는 각각 다음과 같은 URI 형식으로 고유하게 식별됩니다.

```
https://region-specific-endpoint/account-id/vaults/vault-name/notification-configuration
```

Amazon Glacier는 알림 구성을 설정하거나, 가져오거나, 삭제하는 작업을 지원합니다. 알림 구성을 삭제하면 볼트에서 데이터 검색 작업이 완료되어도 알림 메시지가 전송되지 않습니다.

Amazon Glacier에서 지원되는 작업

Amazon Glacier는 볼트 및 아카이브(Amazon Glacier 데이터 모델 참조)를 사용할 수 있도록 일련의 작업을 지원합니다. 지원되는 작업들 중에서 다음 작업들만 비동기식으로 실행됩니다.

- 아카이브 가져오기
- 볼트 인벤토리(아카이브 목록) 가져오기

이러한 작업을 실행하려면 먼저 작업을 시작한 다음, 작업 출력을 다운로드해야 합니다. 다음 섹션에서는 Amazon Glacier 작업을 요약합니다.

볼트 작업

Amazon Glacier는 볼트를 생성 및 삭제할 수 있는 작업을 제공합니다. 사용자는 AWS 리전에 속한 특정 볼트나 모든 볼트에 대한 설명을 가져올 수 있습니다. 볼트 설명에는 생성 날짜, 볼트 내 아카이브 수, 볼트 내 모든 아카이브에서 사용하는 총 크기(바이트), Amazon Glacier의 볼트 인벤토리 생성 날짜 등의 정보가 제공됩니다. Amazon Glacier는 볼트에서 알림 구성을 설정, 검색 및 삭제하는 작업도 제공합니다. 자세한 내용은 Amazon Glacier에서의 볼트 작업 단원을 참조하십시오.

아카이브 작업

Amazon Glacier는 아카이브를 업로드하고 삭제할 수 있는 작업을 제공합니다. 하지만 기존 아카이브를 업데이트할 수는 없습니다. 이때는 기존 아카이브를 삭제한 후 새로운 아카이브를 업로드해야 합니다. 아카이브를 업로드할 때마다 Amazon Glacier가 새로운 아카이브 ID를 생성합니다. 자세한 내용은 Amazon Glacier의 아카이브 작업 단원을 참조하십시오.

작업

Amazon Glacier 작업을 시작하여 아카이브에서 검색을 수행하거나 볼트의 인벤토리를 가져올 수 있습니다.

Amazon Glacier 작업의 유형은 다음과 같습니다.

- `archive-retrieval`: 아카이브를 검색합니다.

  자세한 내용은 Amazon Glacier에서 아카이브 다운로드 단원을 참조하십시오.
- `inventory-retrieval`: 볼트 인벤토리를 작성합니다.

  자세한 내용은 Amazon Glacier에서 볼트 인벤토리 다운로드 단원을 참조하십시오.

Amazon Glacier 액세스

Amazon Glacier는 전송 프로토콜로 HTTP와 HTTPS를 사용하고 메시지 직렬화 형식으로 JavaScript Object Notation(JSON)을 사용하는 RESTful 웹 서비스입니다. 사용자 애플리케이션 코드로 Amazon Glacier 웹 서비스 API에 직접 요청할 수 있습니다. REST API를 직접 사용하는 경우 요청에 서명하고 이를 인증하기 위해 필요한 코드를 작성해야 합니다. API에 대한 자세한 내용은 Amazon Glacier를 위한 API 참조단원을 참조하세요.

또는 Amazon Glacier REST API 호출을 래핑하는 AWS SDKs를 사용하여 애플리케이션 개발을 간소화할 수 있습니다. 자격 증명을 입력하면 이러한 라이브러리에서 인증 및 서명 요청을 처리합니다. AWS SDK 사용에 대한 자세한 내용은 Amazon Glacier에서 AWS SDKs 사용 섹션을 참조하세요.

Amazon Glacier는 콘솔도 제공합니다. 그러나 모든 아카이브 및 작업 작업에서는 REST API를 직접 사용하거나 AWS SDK 래퍼 라이브러리를 사용하여 코드를 작성하고 요청해야 합니다. Amazon Glacier 콘솔에 액세스하려면 Amazon Glacier 콘솔로 이동하세요.

리전 및 엔드포인트

특정에서 볼트를 생성합니다 AWS 리전. 항상 특정 AWS 리전에 속한 엔드포인트로 Amazon Glacier 요청을 보냅니다. Amazon Glacier에서 지원하는 AWS 리전 목록은 AWS 일반 참조의 Amazon Glacier 엔드포인트 및 할당량을 참조하세요.

Amazon Glacier 시작하기

볼트와 아카이브를 사용하여 Amazon Glacier(Amazon Glacier)를 시작할 수 있습니다. 볼트란 아카이브를 저장하는 컨테이너를 말하며, 아카이브는 사진, 동영상 또는 문서 등 볼트에 저장되는 객체를 말합니다. 아카이브는 Amazon Glacier에서 스토리지의 기본 단위입니다. 이번 시작하기 연습에서는 볼트 및 아카이브에서 기본적인 Amazon Glacier 작업을 탐색하기 위한 지침을 제공합니다. 이 리소스에 대한 자세한 내용은 Amazon Glacier 데이터 모델 섹션을 참조하세요.

시작하기 연습에서는 볼트를 생성하고 아카이브를 업로드 및 다운로드한 후 아카이브와 볼트를 다시 삭제하는 연습을 합니다. 이 모든 작업을 프로그래밍 방식으로도 할 수 있지만 이 연습에서는 Amazon Glacier 관리 콘솔을 사용하여 볼트를 생성하고 삭제합니다. 아카이브 업로드 및 다운로드를 위해이 시작하기 섹션에서는 AWS SDK for Java 및에 대한 상위 수준 API를 사용합니다 AWS SDK for .NET. 하이레벨 API는 Amazon Glacier 작업 시 매우 간편한 프로그래밍 환경을 지원합니다. AWS SDKsAmazon Glacier에서 AWS SDKs 사용.

> **Important**
>
> Amazon Glacier는 콘솔을 제공합니다. 그러나 업로드, 다운로드 또는 삭제와 같은 아카이브 작업을 수행하려면 AWS Command Line Interface (CLI) 또는 쓰기 코드를 사용해야 합니다. 콘솔은 아카이브 작업을 지원하지 않습니다. 예를 들어 사진, 비디오 및 기타 문서와 같은 데이터를 업로드하려면 AWS CLI 를 사용하거나 REST API를 직접 사용하거나 AWS SDKs를 사용하여 요청 코드를 작성해야 합니다.
>
> 를 설치하려면 단원을 AWS CLI참조하십시오AWS Command Line Interface. 에서 Amazon Glacier를 사용하는 방법에 대한 자세한 내용은 Amazon Glacier 참조를 AWS CLI참조하세요. AWS CLI Amazon Glacier 를 사용하여 Amazon Glacier AWS CLI 에 아카이브를 업로드하는 예제는 에서 Amazon Glacier 사용을 참조하세요 AWS Command Line Interface.

이번 시작하기 연습에서는 아카이브를 업로드하거나, 다운로드할 때 사용할 수 있도록 Java 및 C# 코드 예제를 제공합니다. 또한 시작하기 연습의 마지막 섹션에서는 Amazon Glacier를 사용하는 개발자 경험에 대해 더욱 자세하게 알아볼 수 있는 몇 가지 단계를 제공합니다.

주제

- 1단계: Amazon Glacier를 시작하기 전에
- 2단계: Amazon Glacier에서 볼트 생성
- 3단계: 아카이브를 Amazon Glacier의 볼트에 업로드

- 4단계: 아카이브를 Amazon Glacier의 볼트에서 다운로드
- 5단계: Amazon Glacier 볼트에서 아카이브 삭제
- 6단계: Amazon Glacier에서 볼트 삭제
- 추가 정보

1단계: Amazon Glacier를 시작하기 전에

이 연습을 시작하려면 먼저 AWS 계정 (아직 없는 경우)에 가입한 다음 AWS SDKs 중 하나를 다운로드해야 합니다. 지침은 다음 섹션을 참조하세요.

주제

- AWS 계정 및 관리자 설정
- 적절한 AWS SDK 다운로드

AWS 계정 및 관리자 설정

아직 등록하지 않은 경우에 가입 AWS 계정 하고 계정에서 관리자 사용자를 생성해야 합니다.

설정을 완료하려면 다음 토픽의 지침을 따르세요.

설정 AWS 계정 및 관리자 사용자 생성

에 가입 AWS

Amazon Web Services(AWS)에 가입하면 AWS 계정 가 Amazon Glacier를 AWS포함한의 모든 서비스에 자동으로 등록됩니다. 사용자에게는 사용한 서비스에 대해서만 요금이 청구됩니다. Amazon Glacier 사용 요금에 대한 자세한 정보는 Amazon Glacier 요금 페이지를 참조하세요.

가 이미 있는 경우 로 AWS 계정건너뜁니다적절한 AWS SDK 다운로드. 이 없는 경우 AWS 계정다음 절차에 따라 생성합니다.

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

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

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

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

다음 옵션 중 하나를 선택하여 관리 사용자를 생성합니다.

관리자를 관리하는 방법 한 가지 선택	목적	By	다른 방법
IAM Identity Center에서 (권장)	단기 보안 인증 정보를 사용하여 AWS에 액세스합니다. 이는 보안 모범 사례와 일치합니다. 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.	AWS IAM Identity Center 사용 설명서의 시작하기 지침을 따릅니다.	AWS Command Line Interface 사용 설명서에서 사용하도록 AWS CLI 를 구성 AWS IAM Identity Center하여 프로그래밍 방식 액세스를 구성합니다.
IAM에서 (권장되지 않음)	장기 보안 인증 정보를 사용하여 AWS에 액세스합니다.	IAM 사용 설명서의 비상 액세스를 위한 IAM 사용자 생성에 나와 있는 지침을 따르세요.	IAM 사용 설명서에 나온 IAM 사용자의 액세스 키 관리를 수행하여 프로그래밍 방식의 액세스를 구성합니다.

적절한 AWS SDK 다운로드

시작하기 연습을 시도하려면 사용할 프로그래밍 언어를 결정한 다음 개발 플랫폼에 적합한 AWS SDK를 다운로드해야 합니다.

여기에서는 Java와 C# 언어로 작성된 예제를 제공합니다.

다운로드 AWS SDK for Java

이 개발자 안내서의 Java 예제를 테스트하려면 AWS SDK for Java가 필요합니다. 다운로드 옵션은 다음과 같습니다.

- Eclipse를 사용하는 경우 업데이트 사이트 http://aws.amazon.com/eclipse/ AWS Toolkit for Eclipse 사용하여를 다운로드하고 설치할 수 있습니다. 자세한 내용은 AWS Toolkit for Eclipse 단원을 참조하십시오.
- 다른 IDE를 사용하여 애플리케이션을 개발하는 경우에는 AWS SDK for Java를 다운로드합니다.

다운로드 AWS SDK for .NET

이 개발자 안내서의 C# 예제를 테스트하려면 AWS SDK for .NET이 필요합니다. 다운로드 옵션은 다음과 같습니다.

- Visual Studio를 사용하는 경우 AWS SDK for .NET 및를 모두 설치할 수 있습니다 AWS Toolkit for Visual Studio. 도구 키트는 개발에 사용할 수 있는 AWS Explorer for Visual Studio 및 프로젝트 템플릿을 제공합니다. 를 다운로드하려면 http://aws.amazon.com/sdkfornet AWS SDK for .NET이동합니다. 기본적으로 설치 스크립트는 AWS SDK와를 모두 설치합니다 AWS Toolkit for Visual Studio. 툴킷에 대한 자세한 내용은 AWS Toolkit for Visual Studio 사용 설명서를 참조하세요.
- 다른 IDE를 사용하여 애플리케이션을 개발하는 경우 이전 단계에서 제공된 링크를 사용하여 AWS SDK for .NET만 설치할 수 있습니다.

2단계: Amazon Glacier에서 볼트 생성

볼트는 아카이브를 저장하기 위한 컨테이너입니다. 첫 번째 단계는 지원되는 중 하나에 볼트를 생성하는 것입니다 AWS 리전. Amazon Glacier에서 AWS 리전 지원하는 목록은 AWS 일반 참조의 Amazon Glacier 엔드포인트 및 할당량을 참조하세요.

볼트는 프로그래밍 방식으로, 혹은 Amazon Glacier 콘솔을 사용하여 생성할 수 있습니다. 여기에서는 콘솔을 사용하여 볼트를 생성합니다.

볼트를 생성하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/glacier/home Amazon Glacier 콘솔을 엽니다.

2. 왼쪽 탐색 창에서 볼트를 선택합니다.
3. 볼트 생성을 선택합니다.

   볼트 생성 페이지가 열립니다.
4. 리전 선택의 리전 선택기 AWS 리전 에서를 선택합니다. 볼트는 사용자가 선택한 리전에 위치합니다.
5. 볼트 이름에는 볼트를 위한 이름을 입력합니다.

   다음은 볼트의 이름을 지정할 때 필요한 요건입니다.

   - 볼트 이름은 볼트가 AWS 리전 생성되는 AWS 계정 및 내에서 고유해야 합니다.
   - 볼트 이름은 반드시 1~255자 이내로 작성해야 합니다.
   - 볼트 이름에는 a~z, A~Z, 0~9, _(밑줄), -(하이픈), .(마침표)만 사용할 수 있습니다.
6. 작업 완료 시 볼트에 대한 알림을 켜거나 끄려면 이벤트 알림에서 다음 설정 중 하나를 선택합니다.

   - 알림 끄기: 지정된 작업이 완료되면 알림이 꺼지고 알림을 Amazon Simple Notification Service(SNS)로 보내지 않습니다.

   - 알림 켜기: 지정된 작업이 완료되면 알림이 켜지고 제공된 Amazon SNS 토픽으로 알림이 전송됩니다.

     알림 켜기를 선택한 경우 Amazon Glacier 콘솔을 사용하여 볼트 알림 구성을 참조하세요.
7. AWS 리전 및 볼트 이름이 올바르면 볼트 생성을 선택합니다.

이제 Amazon Glacier 콘솔의 볼트 페이지에 새 볼트가 나열됩니다.

3단계: 아카이브를 Amazon Glacier의 볼트에 업로드

이번 단계에서는 샘플 아카이브를 이전 단계에서 생성한 볼트에 업로드합니다(2단계: Amazon Glacier에서 볼트 생성 섹션 참조). 사용하는 개발 플랫폼에 따라 이번 섹션 끝에 있는 링크 중 하나를 클릭합니다.

> **Important**
>
> 업로드, 다운로드, 삭제 등 아카이브 작업에서는 AWS Command Line Interface (CLI)를 사용하거나 코드를 작성해야 합니다. 콘솔은 아카이브 작업을 지원하지 않습니다. 예를 들어 사진, 비

디오 및 기타 문서와 같은 데이터를 업로드하려면 AWS CLI 를 사용하거나 REST API를 직접 사용하거나 AWS SDKs를 사용하여 요청 코드를 작성해야 합니다.
를 설치하려면 단원을 AWS CLI참조하십시오AWS Command Line Interface. 에서 Amazon Glacier를 사용하는 방법에 대한 자세한 내용은 Amazon Glacier 참조 AWS CLI를 참조하세요. AWS CLI Amazon Glacier 를 사용하여 Amazon Glacier AWS CLI 에 아카이브를 업로드하는 예제는 에서 Amazon Glacier 사용을 참조하세요 AWS Command Line Interface.

아카이브란 사진, 동영상, 문서 등 볼트에 저장되는 모든 객체를 말합니다. 아카이브는 Amazon Glacier에서 스토리지의 기본 단위입니다. 아카이브는 단일 요청으로 업로드할 수 있습니다. 대용량 아카이브일 경우에는 Amazon Glacier가 아카이브를 여러 파트로 나누어 업로드할 수 있도록 멀티파트 업로드 API 작업을 제공합니다.

이 섹션에서는 단일 요청으로 샘플 아카이브를 업로드합니다. 또한 크기가 작은 파일을 지정합니다. 파일 용량이 커지면 멀티파트 업로드가 적합합니다. 자세한 내용은 대용량 아카이브를 여러 파트로 나누어 업로드(멀티파트 업로드) 단원을 참조하십시오.

주제

- 를 사용하여 Amazon Glacier의 볼트에 아카이브 업로드 AWS SDK for Java
- 를 사용하여 Amazon Glacier의 볼트에 아카이브 업로드 AWS SDK for .NET

를 사용하여 Amazon Glacier의 볼트에 아카이브 업로드 AWS SDK for Java

다음 Java 코드 예제에서는의 상위 수준 API AWS SDK for Java 를 사용하여 샘플 아카이브를 볼트에 업로드합니다. 코드 예제에서 다음 사항에 유의합니다.

- 코드 예제는 `AmazonGlacierClient` 클래스 인스턴스를 생성합니다.
- 이 예시는 AWS SDK for Java의 하이레벨 API에서 `ArchiveTransferManager` 클래스의 `upload` API 작업을 사용합니다.
- 이 예시는 미국 서부(오레곤) 리전(`us-west-2`)을 사용합니다.

이 예제의 실행 방법에 대한 단계별 지침은 Eclipse를 사용하여 Amazon Glacier의 Java 예 실행 섹션을 참조하세요. 반드시 아래와 같이 업로드할 아카이브 파일 이름을 사용해 코드를 업데이트해야 합니다.

> **Note**
>
> Amazon Glacier는 모든 아카이브 인벤토리를 볼트에 저장합니다. 하지만 다음 예제에서 아카이브를 업로드하더라도 볼트 인벤토리가 업데이트되기 전까지는 관리 콘솔의 볼트에 업로드된 아카이브가 표시되지 않습니다. 볼트 인벤토리의 업데이트는 일반적으로 1일 1회 실행됩니다.

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.UploadArchiveRequest;
import software.amazon.awssdk.services.glacier.model.UploadArchiveResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.io.File;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UploadArchive {

    static final int ONE_MB = 1024 * 1024;

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

```
        final String usage = """

                Usage:   <strPath> <vaultName>\s

                Where:
                   strPath - The path to the archive to upload (for example, C:\\AWS
\\test.pdf).
                   vaultName - The name of the vault.
                """;

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

        String strPath = args[0];
        String vaultName = args[1];
        File myFile = new File(strPath);
        Path path = Paths.get(strPath);
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String archiveId = uploadContent(glacier, path, vaultName, myFile);
        System.out.println("The ID of the archived item is " + archiveId);
        glacier.close();
    }

    public static String uploadContent(GlacierClient glacier, Path path, String
 vaultName, File myFile) {
        // Get an SHA-256 tree hash value.
        String checkVal = computeSHA256(myFile);
        try {
            UploadArchiveRequest uploadRequest = UploadArchiveRequest.builder()
                    .vaultName(vaultName)
                    .checksum(checkVal)
                    .build();

            UploadArchiveResponse res = glacier.uploadArchive(uploadRequest, path);
            return res.archiveId();

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

```
        }
        return "";
    }

    private static String computeSHA256(File inputFile) {
        try {
            byte[] treeHash = computeSHA256TreeHash(inputFile);
            System.out.printf("SHA-256 tree hash = %s\n", toHex(treeHash));
            return toHex(treeHash);

        } catch (IOException ioe) {
            System.err.format("Exception when reading from file %s: %s", inputFile,
ioe.getMessage());
            System.exit(-1);

        } catch (NoSuchAlgorithmException nsae) {
            System.err.format("Cannot locate MessageDigest algorithm for SHA-256:
%s", nsae.getMessage());
            System.exit(-1);
        }
        return "";
    }

    public static byte[] computeSHA256TreeHash(File inputFile) throws IOException,
            NoSuchAlgorithmException {

        byte[][] chunkSHA256Hashes = getChunkSHA256Hashes(inputFile);
        return computeSHA256TreeHash(chunkSHA256Hashes);
    }

    /**
     * Computes an SHA256 checksum for each 1 MB chunk of the input file. This
     * includes the checksum for the last chunk, even if it's smaller than 1 MB.
     */
    public static byte[][] getChunkSHA256Hashes(File file) throws IOException,
            NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        long numChunks = file.length() / ONE_MB;
        if (file.length() % ONE_MB > 0) {
            numChunks++;
        }

        if (numChunks == 0) {
```

```
            return new byte[][] { md.digest() };
        }

        byte[][] chunkSHA256Hashes = new byte[(int) numChunks][];
        FileInputStream fileStream = null;

        try {
            fileStream = new FileInputStream(file);
            byte[] buff = new byte[ONE_MB];

            int bytesRead;
            int idx = 0;

            while ((bytesRead = fileStream.read(buff, 0, ONE_MB)) > 0) {
                md.reset();
                md.update(buff, 0, bytesRead);
                chunkSHA256Hashes[idx++] = md.digest();
            }

            return chunkSHA256Hashes;

        } finally {
            if (fileStream != null) {
                try {
                    fileStream.close();
                } catch (IOException ioe) {
                    System.err.printf("Exception while closing %s.\n %s",
file.getName(),
                            ioe.getMessage());
                }
            }
        }
    }

    /**
     * Computes the SHA-256 tree hash for the passed array of 1 MB chunk
     * checksums.
     */
    public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
            throws NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        byte[][] prevLvlHashes = chunkSHA256Hashes;
        while (prevLvlHashes.length > 1) {
```

```
            int len = prevLvlHashes.length / 2;
            if (prevLvlHashes.length % 2 != 0) {
                len++;
            }

            byte[][] currLvlHashes = new byte[len][];
            int j = 0;
            for (int i = 0; i < prevLvlHashes.length; i = i + 2, j++) {

                // If there are at least two elements remaining.
                if (prevLvlHashes.length - i > 1) {

                    // Calculate a digest of the concatenated nodes.
                    md.reset();
                    md.update(prevLvlHashes[i]);
                    md.update(prevLvlHashes[i + 1]);
                    currLvlHashes[j] = md.digest();

                } else { // Take care of the remaining odd chunk
                    currLvlHashes[j] = prevLvlHashes[i];
                }
            }

            prevLvlHashes = currLvlHashes;
        }

        return prevLvlHashes[0];
    }

    /**
     * Returns the hexadecimal representation of the input byte array
     */
    public static String toHex(byte[] data) {
        StringBuilder sb = new StringBuilder(data.length * 2);
        for (byte datum : data) {
            String hex = Integer.toHexString(datum & 0xFF);

            if (hex.length() == 1) {
                // Append leading zero.
                sb.append("0");
            }
            sb.append(hex);
        }
        return sb.toString().toLowerCase();
```

```
    }
}
```

- API 세부 정보는AWS SDK for Java 2.x API 참조의 UploadArchive를 참조하세요.

를 사용하여 Amazon Glacier의 볼트에 아카이브 업로드 AWS SDK for .NET

다음 C# 코드 예제에서는의 상위 수준 API AWS SDK for .NET 를 사용하여 샘플 아카이브를 볼트에 업로드합니다. 코드 예제에서 다음 사항에 유의합니다.

- 이 예는 지정된 Amazon Glacier 리전 엔드포인트에 대한 `ArchiveTransferManager` 클래스 인스턴스를 생성합니다.
- 이 코드 예는 미국 서부(오리건) 리전(`us-west-2`)을 사용합니다.
- 이 코드 예는 `ArchiveTransferManager` 클래스의 `Upload` API 작업을 사용하여 아카이브를 업로드합니다. 아카이브 크기가 작은 경우에는 이 작업으로 아카이브를 직접 Amazon Glacier에 업로드합니다. 아카이브 크기가 큰 경우에는 이 작업에서 Amazon Glacier의 멀티파트 업로드 API 작업을 사용해 업로드를 여러 파트로 분할하기 때문에 데이터를 Amazon Glacier에 스트리밍하는 도중 오류가 발생하더라도 더욱 쉽게 복구할 수 있습니다.

다음 예제의 실행을 위한 단계별 지침은 코드 예제 실행 섹션을 참조하세요. 반드시 아래와 같이 볼트 이름과 업로드할 아카이브 파일 이름을 사용해 코드를 업데이트해야 합니다.

> **Note**
>
> Amazon Glacier는 모든 아카이브 인벤토리를 볼트에 저장합니다. 다음 예시에서 아카이브를 업로드하더라도 볼트 인벤토리가 업데이트되기 전까지는 관리 콘솔의 볼트에 아카이브가 나타나지 않습니다. 볼트 인벤토리의 업데이트는 일반적으로 1일 1회 실행됩니다.

Example-의 상위 수준 API를 사용하여 아카이브 업로드 AWS SDK for .NET

```
using System;
using Amazon.Glacier;
using Amazon.Glacier.Transfer;
using Amazon.Runtime;
```

```
namespace glacier.amazon.com.docsamples
{
    class ArchiveUploadHighLevel_GettingStarted
    {
        static string vaultName = "examplevault";
        static string archiveToUpload = "*** Provide file name (with full path) to
 upload ***";

        public static void Main(string[] args)
        {
            try
            {
                var manager = new
 ArchiveTransferManager(Amazon.RegionEndpoint.USWest2);
                // Upload an archive.
                string archiveId = manager.Upload(vaultName, "getting started archive
 test", archiveToUpload).ArchiveId;
                Console.WriteLine("Copy and save the following Archive ID for the next
 step.");
                Console.WriteLine("Archive ID: {0}", archiveId);
                Console.WriteLine("To continue, press Enter");
                Console.ReadKey();
            }
            catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
            catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
            catch (Exception e) { Console.WriteLine(e.Message); }
            Console.WriteLine("To continue, press Enter");
            Console.ReadKey();
        }
    }
}
```

4단계: 아카이브를 Amazon Glacier의 볼트에서 다운로드

이 단계에서는 이전에 3단계: 아카이브를 Amazon Glacier의 볼트에 업로드에 업로드한 샘플 아카이브를 다운로드합니다.

> **Important**
>
> Amazon Glacier는 콘솔을 제공합니다. 그러나 업로드, 다운로드 또는 삭제와 같은 아카이브 작업을 수행하려면 AWS Command Line Interface (CLI) 또는 쓰기 코드를 사용해야 합니다. 콘솔

은 아카이브 작업을 지원하지 않습니다. 예를 들어 사진, 비디오 및 기타 문서와 같은 데이터를 업로드하려면 AWS CLI 를 사용하거나 REST API를 직접 사용하거나 AWS SDKs를 사용하여 요청 코드를 작성해야 합니다.
를 설치하려면 단원을 AWS CLI참조하십시오AWS Command Line Interface. 에서 Amazon Glacier를 사용하는 방법에 대한 자세한 내용은 Amazon Glacier 참조 AWS CLI를 참조하세요. AWS CLI Amazon Glacier 를 사용하여 Amazon Glacier AWS CLI 에 아카이브를 업로드하는 예제는 에서 Amazon Glacier 사용을 참조하세요 AWS Command Line Interface.

일반적으로 Amazon Glacier에서 데이터를 가져오는 작업은 2단계 프로세스로 구성됩니다.

1. 검색 작업을 시작합니다.
2. 작업이 완료되면 데이터 바이트를 다운로드합니다.

아카이브를 Amazon Glacier에서 가져오려면 먼저 작업을 시작합니다. 이후 작업이 끝나면 데이터를 다운로드합니다. 아카이브 가져오기에 대한 자세한 내용은 Amazon Glacier 아카이브 가져오기 섹션을 참조하세요.

요청의 액세스 시간은 신속, 표준 또는 벌크 중 사용자가 선택하는 검색 옵션에 따라서 결정됩니다. 매우 큰 아카이브(250MB+)를 제외한 모든 경우, 신속 검색을 사용하여 액세스된 데이터는 일반적으로 1~5분 안에 사용할 수 있습니다. 표준 옵션으로 아카이브를 검색할 때는 일반적으로 작업을 마치는 데 3~5시간이 걸립니다. 벌크 가져오기는 보통 5~12시간 안에 완료됩니다. 여러 가져오기 옵션에 대한 자세한 내용은 Amazon Glacier FAQ를 참조하세요. 데이터 가져오기 요금에 대한 자세한 내용은 Amazon Glacier 요금 페이지를 참조하세요.

다음 토픽의 코드 예시는 작업을 시작하여 끝날 때까지 기다린 후 아카이브의 데이터를 다운로드합니다.

주제

- 를 사용하여 Amazon Glacier의 볼트에서 아카이브 다운로드 AWS SDK for Java
- 를 사용하여 Amazon Glacier의 볼트에서 아카이브 다운로드 AWS SDK for .NET

를 사용하여 Amazon Glacier의 볼트에서 아카이브 다운로드 AWS SDK for Java

다음 Java 코드 예제에서는의 상위 수준 API AWS SDK for Java 를 사용하여 이전 단계에서 업로드한 아카이브를 다운로드합니다. 코드 예제에서 다음 사항에 유의합니다.

- 코드 예제는 `AmazonGlacierClient` 클래스 인스턴스를 생성합니다.
- 이 코드는 미국 서부(오레곤) 리전(`us-west-2`)을 사용하여 2단계: Amazon Glacier에서 볼트 생성에서 볼트를 만든 위치와 일치하도록 합니다.
- 이 예시는 AWS SDK for Java의 하이레벨 API에서 `ArchiveTransferManager` 클래스의 `download` API 작업을 사용합니다. 이 예시에서는 Amazon Simple Notification Service(SNS) 토픽 및 해당 토픽을 구독하는 Amazon Simple Queue Service(Amazon SQS) 대기열을 생성합니다. 의 지침에 따라 AWS Identity and Access Management (IAM) 관리자 사용자를 생성한 경우 1단계: Amazon Glacier를 시작하기 전에사용자는 Amazon SNS 주제 및 Amazon SQS 대기열을 생성하고 사용하는 데 필요한 IAM 권한을 가집니다.

이 예제의 실행 방법에 대한 단계별 지침은 Eclipse를 사용하여 Amazon Glacier의 Java 예 실행 섹션을 참조하세요. 반드시 3단계: 아카이브를 Amazon Glacier의 볼트에 업로드에서 업로드한 파일의 아카이브 ID를 사용하여 아래와 같이 코드를 업데이트해야 합니다.

Example-를 사용하여 아카이브 다운로드 AWS SDK for Java

```
import java.io.File;
import java.io.IOException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.transfer.ArchiveTransferManager;
import com.amazonaws.services.sns.AmazonSNSClient;
import com.amazonaws.services.sqs.AmazonSQSClient;

public class AmazonGlacierDownloadArchive_GettingStarted {
    public static String vaultName = "examplevault";
    public static String archiveId = "*** provide archive ID ***";
    public static String downloadFilePath  = "*** provide location to download archive
 ***";

    public static AmazonGlacierClient glacierClient;
    public static AmazonSQSClient sqsClient;
    public static AmazonSNSClient snsClient;

    public static void main(String[] args) throws IOException {


     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();
```

```
        glacierClient = new AmazonGlacierClient(credentials);
        sqsClient = new AmazonSQSClient(credentials);
        snsClient = new AmazonSNSClient(credentials);

        glacierClient.setEndpoint("glacier.us-west-2.amazonaws.com");
        sqsClient.setEndpoint("sqs.us-west-2.amazonaws.com");
        snsClient.setEndpoint("sns.us-west-2.amazonaws.com");

        try {
            ArchiveTransferManager atm = new ArchiveTransferManager(glacierClient,
 sqsClient, snsClient);

            atm.download(vaultName, archiveId, new File(downloadFilePath));

        } catch (Exception e)
        {
            System.err.println(e);
        }
    }
}
```

를 사용하여 Amazon Glacier의 볼트에서 아카이브 다운로드 AWS SDK for .NET

다음 C# 코드 예제에서는의 상위 수준 API AWS SDK for .NET 를 사용하여 이전에에 업로드한 아카이브를 다운로드합니다를 사용하여 Amazon Glacier의 볼트에 아카이브 업로드 AWS SDK for .NET. 코드 예제에서 다음 사항에 유의합니다.

- 이 예는 지정된 Amazon Glacier 리전 엔드포인트에 대한 `ArchiveTransferManager` 클래스 인스턴스를 생성합니다.
- 이 코드 예시는 미국 서부(오리건) 리전(`us-west-2`)을 사용하여 이전에 2단계: Amazon Glacier에서 볼트 생성에서 볼트를 생성한 위치와 일치하도록 합니다.
- 이 예시에서는 `ArchiveTransferManager` 클래스의 `Download` API 작업을 사용하여 사용자의 아카이브를 다운로드합니다. 이 예시에서는 Amazon Simple Notification Service(SNS) 토픽 및 해당 토픽을 구독하는 Amazon Simple Queue Service(Amazon SQS) 대기열을 생성합니다. 의 지침에 따라 AWS Identity and Access Management (IAM) 관리자 사용자를 생성한 경우 1단계: Amazon Glacier를 시작하기 전에사용자는 Amazon SNS 주제 및 Amazon SQS 대기열을 생성하고 사용하는 데 필요한 IAM 권한을 가집니다.

- 이제 이 예제에서는 아카이브 가져오기 작업을 시작하고, 그 아카이브 대기열을 사용할 수 있도록 폴링합니다. 아카이브를 사용할 수 있게 되면 다운로드가 시작됩니다. 검색 시간에 대한 자세한 내용은 아카이브 검색 옵션 섹션을 참조하세요.

이 예제의 실행 방법에 대한 단계별 지침은 코드 예제 실행 섹션을 참조하세요. 반드시 3단계: 아카이브를 Amazon Glacier의 볼트에 업로드에서 업로드한 파일의 아카이브 ID를 사용하여 아래와 같이 코드를 업데이트해야 합니다.

Example-의 상위 수준 API를 사용하여 아카이브 다운로드 AWS SDK for .NET

```
using System;
using Amazon.Glacier;
using Amazon.Glacier.Transfer;
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
    class ArchiveDownloadHighLevel_GettingStarted
    {
        static string vaultName = "examplevault";
        static string archiveId = "*** Provide archive ID ***";
        static string downloadFilePath = "*** Provide the file name and path to where
 to store the download ***";

        public static void Main(string[] args)
        {
            try
            {
                var manager = new
 ArchiveTransferManager(Amazon.RegionEndpoint.USWest2);

                var options = new DownloadOptions();
                options.StreamTransferProgress +=
 ArchiveDownloadHighLevel_GettingStarted.progress;
                // Download an archive.
                Console.WriteLine("Intiating the archive retrieval job and then polling
 SQS queue for the archive to be available.");
                Console.WriteLine("Once the archive is available, downloading will
 begin.");
                manager.Download(vaultName, archiveId, downloadFilePath, options);
                Console.WriteLine("To continue, press Enter");
                Console.ReadKey();
```

```
            }
            catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
            catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
            catch (Exception e) { Console.WriteLine(e.Message); }
            Console.WriteLine("To continue, press Enter");
            Console.ReadKey();
        }

        static int currentPercentage = -1;
        static void progress(object sender, StreamTransferProgressArgs args)
        {
            if (args.PercentDone != currentPercentage)
            {
                currentPercentage = args.PercentDone;
                Console.WriteLine("Downloaded {0}%", args.PercentDone);
            }
        }
    }
}
```

5단계: Amazon Glacier 볼트에서 아카이브 삭제

이 단계에서는 3단계: 아카이브를 Amazon Glacier의 볼트에 업로드에 업로드한 샘플 아카이브를 삭제합니다.

> **Important**
>
> Amazon Glacier 콘솔을 사용하여 아카이브를 삭제할 수는 없습니다. 업로드, 다운로드 또는 삭제와 같은 아카이브 작업을 수행하려면 AWS Command Line Interface (CLI) 또는 쓰기 코드를 사용해야 합니다. 사진, 동영상 및 기타 문서와 같은 데이터를 업로드하려면 AWS CLI 를 사용하거나 REST API를 직접 사용하거나 AWS SDKs를 사용하여 요청 코드를 작성해야 합니다. 를 설치하려면 단원을 AWS CLI참조하십시오AWS Command Line Interface. 에서 Amazon Glacier를 사용하는 방법에 대한 자세한 내용은 Amazon Glacier 참조 AWS CLI를 참조하세요. AWS CLI Amazon Glacier 를 사용하여 Amazon Glacier AWS CLI 에 아카이브를 업로드하는 예제는 에서 Amazon Glacier 사용을 참조하세요 AWS Command Line Interface.

다음 SDK 중 하나 또는 AWS CLI를 수행하여 샘플 아카이브를 삭제합니다.

- 를 사용하여 Amazon Glacier의 볼트에서 아카이브 삭제 AWS SDK for Java

- 를 사용하여 Amazon Glacier의 볼트에서 아카이브 삭제 AWS SDK for .NET
- 를 사용하여 Amazon Glacier에서 아카이브 삭제 AWS CLI

관련 섹션

- 3단계: 아카이브를 Amazon Glacier의 볼트에 업로드
- Amazon Glacier에서 아카이브 삭제

를 사용하여 Amazon Glacier의 볼트에서 아카이브 삭제 AWS SDK for Java

다음 코드 예제에서는를 사용하여 아카이브 AWS SDK for Java 를 삭제합니다. 코드에서 다음 사항에 유의하세요.

- `DeleteArchiveRequest` 객체는 아카이브가 위치하는 볼트 이름과 아카이브 ID를 포함하여 삭제 요청에 대해서 설명합니다.
- `deleteArchive` API 작업은 Amazon Glacier에 아카이브 삭제 요청을 보냅니다.
- 이 예시에서는 미국 서부(오리건) 리전(`us-west-2`)을 사용합니다.

이 예제의 실행 방법에 대한 단계별 지침은 Eclipse를 사용하여 Amazon Glacier의 Java 예 실행 섹션을 참조하세요. 반드시 3단계: 아카이브를 Amazon Glacier의 볼트에 업로드에서 업로드한 파일의 아카이브 ID를 사용하여 아래와 같이 코드를 업데이트해야 합니다.

Example-를 사용하여 아카이브 삭제 AWS SDK for Java

```
import java.io.IOException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.model.DeleteArchiveRequest;

public class AmazonGlacierDeleteArchive_GettingStarted {

    public static String vaultName = "examplevault";
    public static String archiveId = "*** provide archive ID***";
    public static AmazonGlacierClient client;

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

```
    ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        client = new AmazonGlacierClient(credentials);
        client.setEndpoint("https://glacier.us-west-2.amazonaws.com/");

        try {

            // Delete the archive.
            client.deleteArchive(new DeleteArchiveRequest()
                .withVaultName(vaultName)
                .withArchiveId(archiveId));

            System.out.println("Deleted archive successfully.");

        } catch (Exception e) {
            System.err.println("Archive not deleted.");
            System.err.println(e);
        }
    }
}
```

를 사용하여 Amazon Glacier의 볼트에서 아카이브 삭제 AWS SDK for .NET

다음 C# 코드 예제에서는의 상위 수준 API AWS SDK for .NET 를 사용하여 이전 단계에서 업로드한 아카이브를 삭제합니다. 코드 예제에서 다음 사항에 유의합니다.

- 이 예는 지정된 Amazon Glacier 리전 엔드포인트에 대한 `ArchiveTransferManager` 클래스 인스턴스를 생성합니다.
- 이 코드 예는 미국 서부(오리건) 리전(`us-west-2`)을 사용합니다.
- 이 예시는 AWS SDK for .NET의 하이레벨 API의 일부로 제공되는 `ArchiveTransferManager` 클래스의 `Delete` API 작업을 사용합니다.

이 예제의 실행 방법에 대한 단계별 지침은 코드 예제 실행 섹션을 참조하세요. 반드시 3단계: 아카이브를 Amazon Glacier의 볼트에 업로드에서 업로드한 파일의 아카이브 ID를 사용하여 아래와 같이 코드를 업데이트해야 합니다.

Example-의 상위 수준 API를 사용하여 아카이브 삭제 AWS SDK for .NET

```
using System;
```

```
using Amazon.Glacier;
using Amazon.Glacier.Transfer;
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
  class ArchiveDeleteHighLevel_GettingStarted
  {
    static string vaultName = "examplevault";
    static string archiveId = "*** Provide archive ID ***";

    public static void Main(string[] args)
    {
      try
      {
        var manager = new ArchiveTransferManager(Amazon.RegionEndpoint.USWest2);
        manager.DeleteArchive(vaultName, archiveId);
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }
  }
}
```

를 사용하여 Amazon Glacier에서 아카이브 삭제 AWS CLI

AWS Command Line Interface ()를 사용하여 Amazon Glacier에서 아카이브를 삭제할 수 있습니다 AWS CLI.

주제

- (사전 조건) 설정 AWS CLI
- 예:를 사용하여 아카이브 삭제 AWS CLI

(사전 조건) 설정 AWS CLI

1. AWS CLI를 다운로드하고 구성합니다. 관련 지침은 AWS Command Line Interface 사용 설명서에서 다음 토픽을 참조하세요.

설치 AWS Command Line Interface

구성 AWS Command Line Interface

2. 명령 프롬프트에 다음 명령을 입력하여 AWS CLI 설정을 확인합니다. 이러한 명령은 명시적으로 자격 증명을 제공하지 않으므로 기본 프로파일의 자격 증명이 사용됩니다.

   - help 명령을 사용해 보십시오.

     ```
     aws help
     ```

   - 구성된 계정의 Amazon Glacier 볼트 목록을 가져오려면 `list-vaults` 명령을 사용합니다. *123456789012*을 AWS 계정 ID로 바꿉니다.

     ```
     aws glacier list-vaults --account-id 123456789012
     ```

   - 에 대한 현재 구성 데이터를 보려면 `aws configure list` 명령을 AWS CLI사용합니다.

     ```
     aws configure list
     ```

예:를 사용하여 아카이브 삭제 AWS CLI

1. `initiate-job` 명령을 사용하여 인벤토리 검색 작업을 시작합니다. `initiate-job` 명령에 대한 자세한 내용은 작업 시작을 참조하세요.

   ```
   aws glacier initiate-job --vault-name awsexamplevault --account-id 111122223333 --
   job-parameters "{\"Type\": \"inventory-retrieval\"}"
   ```

   예상 결과:

   ```
   {
       "location": "/111122223333/vaults/awsexamplevault/jobs/*** jobid ***",
       "jobId": "*** jobid ***"
   }
   ```

2. `describe-job` 명령을 사용하여 이전 검색 작업의 상태를 확인합니다. `describe-job` 명령에 대한 자세한 내용은 작업 설명을 참조하세요.

```
aws glacier describe-job --vault-name awsexamplevault --account-id 111122223333 --
job-id *** jobid ***
```

예상 결과:

```
{
    "InventoryRetrievalParameters": {
        "Format": "JSON"
    },
    "VaultARN": "*** vault arn ***",
    "Completed": false,
    "JobId": "*** jobid ***",
    "Action": "InventoryRetrieval",
    "CreationDate": "*** job creation date ***",
    "StatusCode": "InProgress"
}
```

3. 작업이 완료될 때까지 기다립니다.

   작업 출력을 다운로드할 수 있을 때까지 기다려야 합니다. 볼트에서 알림 구성을 설정하거나 작업을 시작할 때 Amazon Simple Notification Service(Amazon SNS) 토픽을 지정했다면 Amazon Glacier가 작업 완료 후 해당 토픽에 메시지를 보냅니다.

   볼트의 특정 이벤트에 대해 알림 구성을 설정할 수 있습니다. 자세한 내용은 Amazon Glacier의 볼트 알림 구성 단원을 참조하십시오. Amazon Glacier는 특정 이벤트가 발생할 때마다 지정된 Amazon SNS 토픽으로 메시지를 보냅니다.

4. 작업이 완료되면 `get-job-output` 명령을 사용하여 검색 작업을 파일 `output.json`로 다운로드합니다. `get-job-output` 명령에 대한 자세한 내용은 작업 출력 얻기를 참조하세요.

```
aws glacier get-job-output --vault-name awsexamplevault --account-id 111122223333
 --job-id *** jobid *** output.json
```

이 명령은 다음 필드가 있는 파일을 생성합니다.

```
{
"VaultARN":"arn:aws:glacier:region:111122223333:vaults/awsexamplevault",
"InventoryDate":"*** job completion date ***",
"ArchiveList":[{
{"ArchiveId":"*** archiveid ***",
```

```
"ArchiveDescription":"*** archive description (if set) ***",
"CreationDate":"*** archive creation date ***",
"Size":"*** archive size (in bytes) ***",
"SHA256TreeHash":"*** archive hash ***"
}],
"ArchiveId": 123456789

}
```

5. `delete-archive` 명령을 사용하여 볼트가 비워질 때까지 볼트에서 각 아카이브를 삭제합니다.

```
aws glacier delete-archive --vault-name awsexamplevault --account-id 111122223333
 --archive-id="*** archiveid ***"
```

`delete-archive` 명령에 대한 자세한 내용은 아카이브 삭제를 참조하세요.

6단계: Amazon Glacier에서 볼트 삭제

볼트는 아카이브를 저장하기 위한 컨테이너입니다. Amazon Glacier 볼트를 삭제하려면 먼저 Amazon Glacier에서 계산된 마지막 인벤토리를 기준으로 볼트에 있는 모든 기존 아카이브를 삭제해야 합니다.

볼트는 프로그래밍 방식으로, 혹은 Amazon Glacier 콘솔을 사용하여 삭제할 수 있습니다. 볼트를 프로그래밍 방식으로 삭제하는 방법에 대한 자세한 내용은 Amazon Glacier에서 볼트 삭제 섹션을 참조하세요.

> **Important**
>
> 최근 24시간 이내에 아카이브를 볼트에 업로드하거나 볼트에서 아카이브를 삭제하는 경우 최신 정보가 반영되도록 마지막 볼트 인벤토리가 업데이트될 때까지 기다려야 합니다. Amazon Glacier는 각 볼트마다 24시간을 주기로 인벤토리를 준비합니다.

빈 볼트를 삭제하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/glacier/home Amazon Glacier 콘솔을 엽니다.
2. 리전 선택 메뉴에서 삭제할 볼트 AWS 리전 의를 선택합니다.

   시작하기 연습에서 예시 볼트는 미국 서부(오레곤) 리전에 속합니다.

3. 삭제를 원하는 빈 볼트 옆의 옵션 버튼을 선택합니다. 볼트가 비어 있지 않으면 볼트를 삭제하기 전에 모든 아카이브를 삭제해야 합니다. 자세한 내용은 Amazon Glacier에서 아카이브 삭제 단원을 참조하십시오.

> **Important**
>
> 볼트 삭제는 실행 취소할 수 없습니다.

4. 삭제를 선택합니다.
5. 볼트 삭제 대화 상자가 나타납니다. 삭제를 선택합니다.

비어 있지 않은 볼트를 삭제하려면

1. 비어 있지 않은 볼트를 삭제하는 경우 볼트를 삭제하기 전에 먼저 기존 아카이브를 모두 삭제해야 합니다. REST API, AWS SDK for Java, AWS SDK for .NET 또는를 사용하여 아카이브 삭제 요청을 하는 코드를 작성하여이 작업을 수행할 수 있습니다 AWS CLI. 아카이브 삭제에 대한 자세한 내용은 5단계: Amazon Glacier 볼트에서 아카이브 삭제 섹션을 참조하세요.
2. 볼트가 비워지면 이전 절차에서 단계에 따라 빈 볼트를 삭제합니다.

추가 정보

이제 시작하기 연습을 마쳤습니다. 다음 섹션을 참조하여 Amazon Glacier에 대해 좀 더 자세히 알아보세요.

- Amazon Glacier에서의 볼트 작업
- Amazon Glacier의 아카이브 작업

Amazon Glacier에서의 볼트 작업

볼트는 아카이브를 저장할 수 있는 컨테이너입니다. 볼트를 생성하려면 볼트 이름과 볼트를 생성하기 원하는 AWS 리전을 지정합니다. Amazon Glacier에서 지원하는 AWS 리전 목록은 AWS 일반 참조의 Amazon Glacier 엔드포인트 및 할당량을 참조하세요.

볼트에 저장할 수 있는 아카이브의 수는 제한이 없습니다.

> **Important**
>
> Amazon Glacier는 콘솔을 제공합니다. 그러나 업로드, 다운로드, 삭제 등 아카이브 작업에서는 AWS Command Line Interface(AWS CLI)를 사용하거나 코드를 작성해야 합니다. 콘솔은 아카이브 작업을 지원하지 않습니다. 예를 들어 사진, 동영상 및 기타 문서 같은 데이터를 업로드하려면 반드시 AWS CLI를 사용하거나, 요청 코드를 작성해야 하는데, 요청 코드는 REST API를 직접 사용하거나 AWS SDK를 사용해서 작성할 수 있습니다.
>
> AWS CLI를 설치하려면 AWS Command Line Interface 섹션을 참조하세요. Amazon Glacier를 AWS CLI와 함께 사용하는 방법에 대한 자세한 내용은 Amazon Glacier에 대한 AWS CLI 참조를 참조하세요. AWS CLI를 사용하여 Amazon Glacier에 아카이브를 업로드하는 방법의 예는 AWS Command Line Interface와 함께 Amazon Glacier 사용을 참조하세요.

주제

- Amazon Glacier 내의 볼트 작업
- Amazon Glacier에 볼트 생성
- Amazon Glacier에서 볼트 메타데이터 검색
- Amazon Glacier에서 볼트 인벤토리 다운로드
- Amazon Glacier의 볼트 알림 구성
- Amazon Glacier에서 볼트 삭제
- Amazon Glacier 볼트의 태그 지정
- Amazon Glacier 볼트 잠금

Amazon Glacier 내의 볼트 작업

Amazon Glacier는 다양한 볼트 작업을 지원합니다. 볼트 작업은 특정 AWS 리전에만 적용됩니다. 즉, 사용자가 볼트를 생성하면 볼트는 특정 AWS 리전에 생성됩니다. 볼트를 나열할 때 Amazon Glacier는 요청에서 지정한 AWS 리전의 볼트 목록을 반환합니다.

볼트 생성 및 삭제

AWS 계정은 AWS 리전당 최대 1,000개의 볼트를 생성할 수 있습니다. Amazon Glacier에서 지원하는 AWS 리전 목록은 AWS 일반 참조의 Amazon Glacier 엔드포인트 및 할당량을 참조하세요.

볼트는 Amazon Glacier에서 마지막으로 계산된 인벤토리를 기준으로 저장된 아카이브가 볼트에 없고, 마지막 인벤토리 이후 쓰기 작업이 없었던 경우에 한해 삭제할 수 있습니다.

> **Note**
>
> Amazon Glacier는 각 볼트마다 24시간을 주기로 인벤토리를 준비합니다. 인벤토리에 최신 정보가 반영되지 않을 수도 있기 때문에 Amazon Glacier는 마지막 볼트 인벤토리 이후 쓰기 작업의 유무를 검사하면서 볼트가 실제로 비어있는지 확인합니다.

자세한 내용은 Amazon Glacier에 볼트 생성 및 Amazon Glacier에서 볼트 삭제 섹션을 참조하세요.

볼트 메타데이터 가져오기

볼트 생성 날짜, 볼트의 아카이브 수, 볼트 내 모든 아카이브의 총 크기 등 볼트에 대한 정보를 검색할 수 있습니다. Amazon Glacier는 계정 내 원하는 AWS 리전에서 특정 볼트나 모든 볼트에 대해 이 정보를 검색할 수 있도록 API 직접 호출을 제공합니다. 자세한 내용은 Amazon Glacier에서 볼트 메타데이터 검색 섹션을 참조하세요.

볼트 인벤토리 다운로드

볼트 인벤토리란 볼트에 저장된 아카이브 목록을 말합니다. 목록에 있는 각 아카이브에 대해 인벤토리는 아카이브 ID, 생성 날짜, 크기와 같은 아카이브 정보를 제공합니다. Amazon Glacier는 아카이브가 볼트에 처음 업데이트된 날짜부터 하루에 한 번씩 볼트 인벤토리를 업데이트합니다. 다운로드하기 위해서는 볼트 인벤토리가 있어야 합니다.

볼트 인벤토리 다운로드는 비동기식 작업입니다. 따라서 먼저 인벤토리 다운로드 작업을 시작해야 합니다. 작업 요청이 수신되면 Amazon Glacier는 인벤토리를 다운로드할 수 있도록 준비합니다. 작업이 완료되면 인벤토리 데이터를 다운로드할 수 있습니다.

작업이 비동기식으로 이루어지기 때문에 Amazon Simple Notification Service(SNS) 알림을 사용하여 작업이 완료되면 알 수 있습니다. 각 작업 요청마다 Amazon SNS 토픽을 지정하거나 특정 볼트 이벤트가 발생할 경우 알림 메시지를 전송하도록 볼트를 구성할 수 있습니다.

Amazon Glacier는 각 볼트마다 24시간을 주기로 인벤토리를 준비합니다. 마지막 인벤토리 이후 볼트에 대한 아카이브 추가 또는 삭제가 없는 경우에는 인벤토리 데이터가 업데이트되지 않습니다.

볼트 인벤토리 작업이 시작되면 Amazon Glacier가 마지막으로 작성된 인벤토리, 즉 실시간 데이터가 아닌 특정 시점 스냅샷을 반환합니다. 아카이브를 업로드할 때마다 볼트 인벤토리를 가져오는 것이 불필요하다고 생각할 수도 있습니다. 하지만 Amazon Glacier에 업로드하는 아카이브와 연결된 메타데이터를 포함하는 클라이언트 측에서 데이터베이스를 관리한다고 가정해 봅시다. 이 경우, 사용자는 볼트 인벤토리가 사용자의 데이터베이스 정보와 실제 볼트 인벤토리를 서로 조정하는 데 얼마나 유용한지 알게 될 것입니다.

볼트 인벤토리 가져오기에 대한 자세한 내용은 Amazon Glacier에서 볼트 인벤토리 다운로드 섹션을 참조하세요.

볼트 알림 구성

볼트 또는 볼트 인벤토리의 아카이브를 검색하는 등 Amazon Glacier에서 무언가를 검색하는 작업은 2단계로 구성됩니다. 먼저 작업을 시작합니다. 작업이 완료되면 작업 출력을 다운로드할 수 있습니다. Amazon Glacier 알림을 사용하면 작업 완료 시 알 수 있습니다. Amazon Glacier는 사용자가 제공한 Amazon Simple Notification Service(Amazon SNS) 토픽에 알림 메시지를 보냅니다.

볼트에 대한 알림을 구성하여 볼트 이벤트와 이벤트 발생 시 알림을 받을 Amazon SNS 토픽을 식별할 수 있습니다. 볼트 이벤트가 발생하면 Amazon Glacier가 지정된 Amazon SNS 토픽으로 알림 메시지를 전송합니다. 자세한 내용은 Amazon Glacier의 볼트 알림 구성 섹션을 참조하세요.

Amazon Glacier에 볼트 생성

볼트를 생성하면 계정의 볼트 집합에 볼트가 추가됩니다. AWS 계정은 AWS 리전당 최대 1,000개까지 볼트를 생성할 수 있습니다. Amazon Glacier(Amazon Glacier)에서 지원하는 AWS 리전의 목록은 AWS 일반 참조의 리전 및 엔드포인트를 참조하세요.

볼트를 생성할 때는 볼트 이름을 입력해야 합니다. 다음은 볼트 이름을 지정할 때 필요한 요건입니다.

- 이름에 포함되는 문자 길이는 1~255자입니다.
- 유효한 문자는 a~z, A~Z, 0~9, '_'(밑줄), '-'(하이픈), '.'(마침표)입니다.

볼트 이름은 볼트를 생성하는 계정 및 AWS 리전에서 고유해야 합니다. 즉, 한 계정에서 서로 다른 AWS 리전에 동일한 이름의 볼트를 만들 수 있지만, 동일한 AWS 리전에서는 만들 수 없습니다.

주제

- Amazon Glacier에서 AWS SDK for Java를 사용하여 볼트 생성
- 를 사용하여 Amazon Glacier에서 볼트 생성 AWS SDK for .NET
- Amazon Glacier에서 REST API를 사용하여 볼트 생성
- Amazon Glacier 콘솔을 사용하여 볼트 생성
- Amazon Glacier에서 AWS Command Line Interface를 사용하여 볼트 생성

Amazon Glacier에서 AWS SDK for Java를 사용하여 볼트 생성

로우레벨 API는 볼트를 생성 및 삭제하거나, 볼트 설명을 가져오거나, 특정 AWS 리전에서 생성된 볼트 목록을 가져오는 등 모든 볼트 작업에 사용할 수 있는 메서드를 제공합니다. 다음은 AWS SDK for Java를 사용하여 볼트를 생성하는 단계입니다.

1. `AmazonGlacierClient` 클래스(클라이언트)의 인스턴스를 만듭니다.

   볼트 생성을 원하는 AWS 리전을 지정합니다. 이 클라이언트를 사용하여 실행하는 모든 작업이 해당 AWS 리전에 적용됩니다.
2. `CreateVaultRequest` 클래스 인스턴스를 생성하여 요청 정보를 입력합니다.

   Amazon Glacier(Amazon Glacier)는 볼트 이름과 사용자의 계정 ID를 요구합니다. 계정 ID를 입력하지 않는 경우에는 요청 서명을 위해 입력하는 자격 증명과 연결되어 있는 계정 ID를 사용합니다. 자세한 내용은 Amazon Glacier AWS SDK for Java 에서 사용 섹션을 참조하세요.
3. 요청 객체를 파라미터로 입력하여 `createVault` 메서드를 실행합니다.

   Amazon Glacier가 반환하는 응답은 `CreateVaultResult` 객체에서 사용할 수 있습니다.

다음은 위에서 설명한 단계를 나타내는 Java 코드 조각입니다. 이 조각을 실행하면 `us-west-2` 리전에 볼트가 생성됩니다. 여기에서 출력되는 `Location`은 계정 ID, AWS 리전 및 볼트 이름이 포함된 볼트의 상대적 URI입니다.

```
AmazonGlacierClient client = new AmazonGlacierClient(credentials);
client.setEndpoint("https://glacier.us-west-2.amazonaws.com");

CreateVaultRequest request = new CreateVaultRequest()
    .withVaultName("*** provide vault name ***");
CreateVaultResult result = client.createVault(request);

System.out.println("Created vault successfully: " + result.getLocation());
```

> **Note**
>
> 기본 REST API에 대한 자세한 내용은 볼트 만들기(PUT vault) 섹션을 참조하세요.

예제: AWS SDK for Java를 사용하여 볼트 생성

다음의 Java 코드 예시는 `us-west-2` 리전(AWS 리전에 대한 더 자세한 내용은 Amazon Glacier 액세스 섹션 참조)에 볼트를 생성합니다. 또한 이 코드 예시는 볼트 정보를 검색하거나, 동일한 AWS 리전에 속하는 모든 볼트를 나열하거나, 생성된 볼트를 삭제하기도 합니다.

다음 예제의 실행을 위한 단계별 지침은 Eclipse를 사용하여 Amazon Glacier의 Java 예 실행 섹션을 참조하세요.

Example

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

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.model.CreateVaultRequest;
import com.amazonaws.services.glacier.model.CreateVaultResult;
import com.amazonaws.services.glacier.model.DeleteVaultRequest;
import com.amazonaws.services.glacier.model.DescribeVaultOutput;
import com.amazonaws.services.glacier.model.DescribeVaultRequest;
import com.amazonaws.services.glacier.model.DescribeVaultResult;
import com.amazonaws.services.glacier.model.ListVaultsRequest;
import com.amazonaws.services.glacier.model.ListVaultsResult;


public class AmazonGlacierVaultOperations {
```

```
    public static AmazonGlacierClient client;

    public static void main(String[] args) throws IOException {

     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        client = new AmazonGlacierClient(credentials);
        client.setEndpoint("https://glacier.us-east-1.amazonaws.com/");

        String vaultName = "examplevaultfordelete";

        try {
            createVault(client, vaultName);
            describeVault(client, vaultName);
            listVaults(client);
            deleteVault(client, vaultName);

        } catch (Exception e) {
            System.err.println("Vault operation failed." + e.getMessage());
        }
    }

    private static void createVault(AmazonGlacierClient client, String vaultName) {
        CreateVaultRequest createVaultRequest = new CreateVaultRequest()
            .withVaultName(vaultName);
        CreateVaultResult createVaultResult = client.createVault(createVaultRequest);

        System.out.println("Created vault successfully: " +
createVaultResult.getLocation());
    }

    private static void describeVault(AmazonGlacierClient client, String vaultName) {
        DescribeVaultRequest describeVaultRequest = new DescribeVaultRequest()
            .withVaultName(vaultName);
        DescribeVaultResult describeVaultResult  =
client.describeVault(describeVaultRequest);

        System.out.println("Describing the vault: " + vaultName);
        System.out.print(
                "CreationDate: " + describeVaultResult.getCreationDate() +
                "\nLastInventoryDate: " + describeVaultResult.getLastInventoryDate() +
                "\nNumberOfArchives: " + describeVaultResult.getNumberOfArchives() +
                "\nSizeInBytes: " + describeVaultResult.getSizeInBytes() +
                "\nVaultARN: " + describeVaultResult.getVaultARN() +
```

```
                "\nVaultName: " + describeVaultResult.getVaultName());
    }

    private static void listVaults(AmazonGlacierClient client) {
        ListVaultsRequest listVaultsRequest = new ListVaultsRequest();
        ListVaultsResult listVaultsResult = client.listVaults(listVaultsRequest);

        List<DescribeVaultOutput> vaultList = listVaultsResult.getVaultList();
        System.out.println("\nDescribing all vaults (vault list):");
        for (DescribeVaultOutput vault : vaultList) {
            System.out.println(
                    "\nCreationDate: " + vault.getCreationDate() +
                    "\nLastInventoryDate: " + vault.getLastInventoryDate() +
                    "\nNumberOfArchives: " + vault.getNumberOfArchives() +
                    "\nSizeInBytes: " + vault.getSizeInBytes() +
                    "\nVaultARN: " + vault.getVaultARN() +
                    "\nVaultName: " + vault.getVaultName());
        }
    }

    private static void deleteVault(AmazonGlacierClient client, String vaultName) {
        DeleteVaultRequest request = new DeleteVaultRequest()
            .withVaultName(vaultName);
        client.deleteVault(request);
        System.out.println("Deleted vault: " + vaultName);
    }

}
```

를 사용하여 Amazon Glacier에서 볼트 생성 AWS SDK for .NET

Amazon SDK에서 .NET용으로 제공하는 하이레벨 및 로우레벨 API는 모두 볼트를 생성하는 메서드를 제공합니다.

주제

- 의 상위 수준 API를 사용하여 볼트 생성 AWS SDK for .NET
- 의 하위 수준 API를 사용하여 볼트 생성 AWS SDK for .NET

의 상위 수준 API를 사용하여 볼트 생성 AWS SDK for .NET

하이레벨 API의 `ArchiveTransferManager` 클래스는 AWS 리전에 볼트를 생성하는 데 사용할 수 있는 `CreateVault` 방법을 제공합니다.

예:의 상위 수준 API를 사용한 볼트 작업 AWS SDK for .NET

다음의 C# 코드 예시는 미국 서부(오레곤)에서 볼트를 생성 및 삭제합니다. 볼트를 생성할 수 AWS 리전 있는 목록은 단원을 참조하십시오Amazon Glacier 액세스.

다음 예제의 실행을 위한 단계별 지침은 코드 예제 실행 섹션을 참조하세요. 아래와 같이 볼트 이름을 사용해 코드를 업데이트해야 합니다.

Example

```
using System;
using Amazon.Glacier;
using Amazon.Glacier.Transfer;
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
  class VaultCreateDescribeListVaultsDeleteHighLevel
  {
    static string vaultName = "*** Provide vault name ***";

    public static void Main(string[] args)
    {
      try
      {
          var manager = new ArchiveTransferManager(Amazon.RegionEndpoint.USWest2);
          manager.CreateVault(vaultName);
          Console.WriteLine("Vault created. To delete the vault, press Enter");
          Console.ReadKey();
          manager.DeleteVault(vaultName);
          Console.WriteLine("\nVault deleted. To continue, press Enter");
          Console.ReadKey();
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }
```

```
    }
}
```

의 하위 수준 API를 사용하여 볼트 생성 AWS SDK for .NET

하위 수준 API는 볼트 생성 및 삭제, 볼트 설명 가져오기, 특정에서 생성된 볼트 목록 가져오기 등 모든 볼트 작업에 대한 메서드를 제공합니다 AWS 리전. 다음은 AWS SDK for .NET를 사용하여 볼트를 생성하는 단계입니다.

1. `AmazonGlacierClient` 클래스(클라이언트)의 인스턴스를 만듭니다.

   볼트를 AWS 리전 생성할를 지정해야 합니다. 이 클라이언트를 사용하여 실행하는 모든 작업이 해당 AWS 리전에 적용됩니다.
2. `CreateVaultRequest` 클래스 인스턴스를 생성하여 요청 정보를 입력합니다.

   Amazon Glacier(Amazon Glacier)는 볼트 이름과 사용자의 계정 ID를 요구합니다. 계정 ID를 입력하지 않는 경우에는 요청 서명을 위해 입력하는 자격 증명과 연결되어 있는 계정 ID로 간주합니다. 자세한 내용은 Amazon Glacier와 함께 AWS SDK for .NET 사용 단원을 참조하십시오.
3. 요청 객체를 파라미터로 입력하여 `CreateVault` 메서드를 실행합니다.

   Amazon Glacier가 반환하는 응답은 `CreateVaultResponse` 객체에서 사용할 수 있습니다.

예:의 하위 수준 API를 사용한 볼트 작업 AWS SDK for .NET

다음은 앞선 단계를 설명하는 C# 예제입니다. 이 예시는 미국 서부(오레곤)에 볼트를 생성합니다. 또한 코드 예제는 볼트 정보를 검색 AWS 리전하고 동일한의 모든 볼트를 나열한 다음 생성된 볼트를 삭제합니다. `Location` 인쇄된는 계정 ID AWS 리전, 및 볼트 이름이 포함된 볼트의 상대적 URI입니다.

> **Note**
>
> 기본 REST API에 대한 자세한 내용은 볼트 만들기(PUT vault) 섹션을 참조하세요.

다음 예제의 실행을 위한 단계별 지침은 코드 예제 실행 섹션을 참조하세요. 아래와 같이 볼트 이름을 사용해 코드를 업데이트해야 합니다.

Example

```
using System;
```

```
using Amazon.Glacier;
using Amazon.Glacier.Model;
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
  class VaultCreateDescribeListVaultsDelete
  {
    static string vaultName = "*** Provide vault name ***";
    static AmazonGlacierClient client;

    public static void Main(string[] args)
    {
       try
      {
         using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2))
        {
          Console.WriteLine("Creating a vault.");
          CreateAVault();
          DescribeVault();
          GetVaultsList();
          Console.WriteLine("\nVault created. Now press Enter to delete the vault...");
          Console.ReadKey();
          DeleteVault();
        }
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }

    static void CreateAVault()
    {
      CreateVaultRequest request = new CreateVaultRequest()
      {
        VaultName = vaultName
      };
      CreateVaultResponse response = client.CreateVault(request);
      Console.WriteLine("Vault created: {0}\n", response.Location);
    }

    static void DescribeVault()
```

```
    {
      DescribeVaultRequest describeVaultRequest = new DescribeVaultRequest()
      {
        VaultName = vaultName
      };

      DescribeVaultResponse describeVaultResponse =
client.DescribeVault(describeVaultRequest);
      Console.WriteLine("\nVault description...");
      Console.WriteLine(
        "\nVaultName: " + describeVaultResponse.VaultName +
        "\nVaultARN: " + describeVaultResponse.VaultARN +
        "\nVaultCreationDate: " + describeVaultResponse.CreationDate +
        "\nNumberOfArchives: " + describeVaultResponse.NumberOfArchives +
        "\nSizeInBytes: " + describeVaultResponse.SizeInBytes +
        "\nLastInventoryDate: " + describeVaultResponse.LastInventoryDate
        );
    }

    static void GetVaultsList()
    {
      string lastMarker = null;
      Console.WriteLine("\n List of vaults in your account in the specific
region ...");
      do
      {
        ListVaultsRequest request = new ListVaultsRequest()
        {
          Marker = lastMarker
        };
        ListVaultsResponse response = client.ListVaults(request);

        foreach (DescribeVaultOutput output in response.VaultList)
        {
          Console.WriteLine("Vault Name: {0} \tCreation Date: {1} \t #of archives:
{2}",
                            output.VaultName, output.CreationDate,
output.NumberOfArchives);
        }
        lastMarker = response.Marker;
      } while (lastMarker != null);
    }

    static void DeleteVault()
```

```
    {
      DeleteVaultRequest request = new DeleteVaultRequest()
      {
        VaultName = vaultName
      };
      DeleteVaultResponse response = client.DeleteVault(request);
    }
  }
}
```

Amazon Glacier에서 REST API를 사용하여 볼트 생성

REST API를 사용하여 볼트를 생성하려면 볼트 만들기(PUT vault) 섹션을 참조하세요.

Amazon Glacier 콘솔을 사용하여 볼트 생성

Amazon Glacier(Amazon Glacier) 콘솔을 사용하여 볼트를 생성하려면 시작하기 자습서의 2단계: Amazon Glacier에서 볼트 생성 섹션을 참조하세요.

Amazon Glacier에서 AWS Command Line Interface를 사용하여 볼트 생성

다음 단계에 따라 AWS Command Line Interface(AWS CLI)를 사용하여 Amazon Glacier(Amazon Glacier)에 볼트를 생성합니다.

주제

- (전제 조건) AWS CLI 설정
- 예제: AWS CLI를 사용하여 볼트 생성

(전제 조건) AWS CLI 설정

1. AWS CLI를 다운로드하고 구성합니다. 관련 지침은 AWS Command Line Interface 사용 설명서에서 다음 토픽을 참조하세요.

   AWS Command Line Interface 설치

   AWS Command Line Interface 구성
2. 명령 프롬프트에 다음 명령을 입력하여 사용자의 AWS CLI 설정을 확인합니다. 이러한 명령은 명시적으로 자격 증명을 제공하지 않으므로 기본 프로파일의 자격 증명이 사용됩니다.

   - help 명령을 사용해 보십시오.

```
aws help
```

- 구성된 계정의 Amazon Glacier 볼트 목록을 가져오려면 `list-vaults` 명령을 사용합니다. *123456789012*를 사용자의 AWS 계정 ID로 바꿉니다.

```
aws glacier list-vaults --account-id 123456789012
```

- AWS CLI를 위한 현재 구성 데이터를 확인하려면 `aws configure list` 명령을 사용합니다.

```
aws configure list
```

예제: AWS CLI를 사용하여 볼트 생성

1. `create-vault` 명령을 사용하여 *111122223333* 계정에 *awsexamplevault*라는 이름의 볼트를 생성합니다.

```
aws glacier create-vault --vault-name awsexamplevault --account-id 111122223333
```

예상 결과:

```
{
    "location": "/111122223333/vaults/awsexamplevault"
}
```

2. `describe-vault` 명령을 사용하여 생성을 확인합니다.

```
aws glacier describe-vault --vault-name awsexamplevault --account-id 111122223333
```

Amazon Glacier에서 볼트 메타데이터 검색

볼트 생성 날짜, 볼트의 아카이브 수, 볼트 내 모든 아카이브의 총 크기 등 볼트에 대한 정보를 검색할 수 있습니다. Amazon Glacier(Amazon Glacier)는 특정 볼트 또는 계정의 특정 AWS 리전에 있는 모든 볼트에 대해이 정보를 검색할 수 있는 API 호출을 제공합니다.

볼트 목록을 검색하는 경우 Amazon Glacier가 볼트 이름의 ASCII 값을 기준으로 정렬된 목록을 반환합니다. 목록에 포함되는 볼트 수는 최대 1,000개입니다. 응답은 항상 확인하여 목록을 계속 이어가는 마커가 있는지 살펴야 합니다. 항목이 더 이상 없는 경우에는 marker 필드가 `null` 값을 갖습니다. 응답으로 반환되는 볼트 수는 옵션으로 제한할 수 있습니다. 응답으로 반환할 볼트가 더 있는 경우에는 결과에 페이지가 매겨집니다. 이후 다음 볼트를 가져오려면 요청을 추가로 전송해야 합니다.

주제

- 를 사용하여 Amazon Glacier에서 볼트 메타데이터 검색 AWS SDK for Java
- 를 사용하여 Amazon Glacier에서 볼트 메타데이터 검색 AWS SDK for .NET
- REST API를 사용하여 볼트 메타데이터 검색
- AWS Command Line Interface를 사용하여 Amazon Glacier에서 볼트 메타데이터 검색

를 사용하여 Amazon Glacier에서 볼트 메타데이터 검색 AWS SDK for Java

주제

- 볼트 메타데이터 검색
- 리전에 속하는 모든 볼트의 메타데이터 검색
- 예제: Amazon SDK for Java를 사용하여 볼트 메타데이터 검색

볼트 메타데이터 검색

특정 볼트 또는 특정 AWS 리전의 모든 볼트에 대한 메타데이터를 검색할 수 있습니다. 다음은 Amazon SDK for Java의 로우레벨 API를 사용해 특정 볼트용 볼트 메타데이터를 검색하는 단계입니다.

1. `AmazonGlacierClient` 클래스(클라이언트)의 인스턴스를 만듭니다.

   볼트가 있는 AWS 리전을 지정해야 합니다. 이 클라이언트를 사용하여 수행하는 모든 작업은 해당 AWS 리전에 적용됩니다.
2. `DescribeVaultRequest` 클래스 인스턴스를 생성하여 요청 정보를 입력합니다.

   Amazon Glacier(Amazon Glacier)는 볼트 이름과 사용자의 계정 ID를 요구합니다. 계정 ID를 입력하지 않는 경우에는 요청 서명을 위해 입력하는 자격 증명과 연결되어 있는 계정 ID로 간주합니다. 자세한 내용은 Amazon Glacier AWS SDK for Java 에서 사용 단원을 참조하십시오.
3. 요청 객체를 파라미터로 입력하여 `describeVault` 메서드를 실행합니다.

Amazon Glacier가 반환하는 볼트 메타데이터 정보는 `DescribeVaultResult` 객체에서 사용할 수 있습니다.

다음은 위에서 설명한 단계를 나타내는 Java 코드 조각입니다.

```
DescribeVaultRequest request = new DescribeVaultRequest()
 .withVaultName("*** provide vault name***");

DescribeVaultResult result = client.describeVault(request);

System.out.print(
        "\nCreationDate: " + result.getCreationDate() +
        "\nLastInventoryDate: " + result.getLastInventoryDate() +
        "\nNumberOfArchives: " + result.getNumberOfArchives() +
        "\nSizeInBytes: " + result.getSizeInBytes() +
        "\nVaultARN: " + result.getVaultARN() +
        "\nVaultName: " + result.getVaultName());
```

> **Note**
>
> 기본 REST API에 대한 자세한 내용은 볼트 설명(GET vault) 섹션을 참조하세요.

리전에 속하는 모든 볼트의 메타데이터 검색

`listVaults` 메서드를 사용하여 특정 AWS 리전의 모든 볼트에 대한 메타데이터를 검색할 수도 있습니다.

다음은 `us-west-2` 리전에 속하는 볼트 목록을 가져오는 Java 코드 조각입니다. 요청에 따라 응답으로 반환되는 볼트 수는 5개로 제한됩니다. 그런 다음 코드 조각은 일련의 `listVaults` 호출을 수행하여 AWS 리전에서 전체 볼트 목록을 검색합니다.

```
AmazonGlacierClient client;
client.setEndpoint("https://glacier.us-west-2.amazonaws.com/");

String marker = null;
do {
    ListVaultsRequest request = new ListVaultsRequest()
```

```
            .withLimit("5")
            .withMarker(marker);
        ListVaultsResult listVaultsResult = client.listVaults(request);

        List<DescribeVaultOutput> vaultList = listVaultsResult.getVaultList();
        marker = listVaultsResult.getMarker();
        for (DescribeVaultOutput vault : vaultList) {
            System.out.println(
                    "\nCreationDate: " + vault.getCreationDate() +
                    "\nLastInventoryDate: " + vault.getLastInventoryDate() +
                    "\nNumberOfArchives: " + vault.getNumberOfArchives() +
                    "\nSizeInBytes: " + vault.getSizeInBytes() +
                    "\nVaultARN: " + vault.getVaultARN() +
                    "\nVaultName: " + vault.getVaultName());
        }
    } while (marker != null);
```

위의 코드 조각에서 요청 시에 `Limit`값을 지정하지 않으면 Amazon Glacier가 Amazon Glacier API의 설정에 따라 최대 10개까지 볼트를 반환합니다. 목록을 조회할 볼트가 더 있는 경우에는 응답 본문의 `marker` 필드에 새로운 요청과 함께 목록이 계속되는 지점에 볼트의 Amazon 리소스 이름(ARN)이 추가됩니다. 그렇지 않으면 `marker` 필드는 null 값을 갖습니다.

참고로 목록의 각 볼트마다 반환되는 정보는 특정 볼트에서 `describeVault` 메서드를 호출하여 가져오는 정보와 동일합니다.

> **Note**
>
> `listVaults` 메서드는 기본 REST API를 호출합니다(볼트 목록 조회(GET vaults) 참조).

예제: Amazon SDK for Java를 사용하여 볼트 메타데이터 검색

유효 코드 예제를 보려면 예제: AWS SDK for Java를 사용하여 볼트 생성 섹션을 참조하세요. Java 코드 예제는 볼트를 생성한 후 볼트 메타데이터를 가져옵니다.

를 사용하여 Amazon Glacier에서 볼트 메타데이터 검색 AWS SDK for .NET

주제

- 볼트 메타데이터 검색

- 리전에 속하는 모든 볼트의 메타데이터 검색
- 예:의 하위 수준 API를 사용하여 볼트 메타데이터 검색 AWS SDK for .NET

볼트 메타데이터 검색

특정 볼트 또는 특정 AWS 리전의 모든 볼트에 대한 메타데이터를 검색할 수 있습니다. 다음은 AWS SDK for .NET의 로우레벨 API를 사용해 특정 볼트의 메타데이터를 가져오는 단계입니다.

1. `AmazonGlacierClient` 클래스(클라이언트)의 인스턴스를 만듭니다.

   볼트가 있는 AWS 리전을 지정해야 합니다. 이 클라이언트를 사용하여 수행하는 모든 작업은 해당 AWS 리전에 적용됩니다.
2. `DescribeVaultRequest` 클래스 인스턴스를 생성하여 요청 정보를 입력합니다.

   Amazon Glacier(Amazon Glacier)는 볼트 이름과 사용자의 계정 ID를 요구합니다. 계정 ID를 입력하지 않는 경우에는 요청 서명을 위해 입력하는 자격 증명과 연결되어 있는 계정 ID로 간주합니다. 자세한 내용은 Amazon Glacier와 함께 AWS SDK for .NET 사용 단원을 참조하십시오.
3. 요청 객체를 파라미터로 입력하여 `DescribeVault` 메서드를 실행합니다.

   Amazon Glacier가 반환하는 볼트 메타데이터 정보는 `DescribeVaultResult` 객체에서 사용할 수 있습니다.

다음은 위에서 설명한 단계를 나타내는 C# 코드 조각입니다. 이 코드 조각은 미국 서부(오레곤) 리전에 속하는 기존 볼트에 대한 메타데이터 정보를 검색합니다.

```
AmazonGlacierClient client;
client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2);

DescribeVaultRequest describeVaultRequest = new DescribeVaultRequest()
{
  VaultName = "*** Provide vault name ***"
};
DescribeVaultResponse describeVaultResponse =
 client.DescribeVault(describeVaultRequest);
Console.WriteLine("\nVault description...");
Console.WriteLine(
   "\nVaultName: " + describeVaultResponse.VaultName +
   "\nVaultARN: " + describeVaultResponse.VaultARN +
   "\nVaultCreationDate: " + describeVaultResponse.CreationDate +
```

```
    "\nNumberOfArchives: " + describeVaultResponse.NumberOfArchives +
    "\nSizeInBytes: " + describeVaultResponse.SizeInBytes +
    "\nLastInventoryDate: " + describeVaultResponse.LastInventoryDate
    );
```

> **Note**
>
> 기본 REST API에 대한 자세한 내용은 볼트 설명(GET vault) 섹션을 참조하세요.

리전에 속하는 모든 볼트의 메타데이터 검색

`ListVaults` 메서드를 사용하여 특정 AWS 리전의 모든 볼트에 대한 메타데이터를 검색할 수도 있습니다.

다음은 미국 서부(오레곤)의 볼트 목록을 검색하는 C# 코드 조각입니다. 요청에 따라 응답으로 반환되는 볼트 수는 5개로 제한됩니다. 그런 다음 코드 조각은 일련의 `ListVaults` 호출을 수행하여 AWS 리전에서 전체 볼트 목록을 검색합니다.

```
AmazonGlacierClient client;
client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2);
string lastMarker = null;
Console.WriteLine("\n List of vaults in your account in the specific AWS Region ...");
do
{
  ListVaultsRequest request = new ListVaultsRequest()
  {
    Limit = 5,
    Marker = lastMarker
  };
  ListVaultsResponse response = client.ListVaults(request);

  foreach (DescribeVaultOutput output in response.VaultList)
  {
    Console.WriteLine("Vault Name: {0} \tCreation Date: {1} \t #of archives: {2}",
                      output.VaultName, output.CreationDate, output.NumberOfArchives);
  }
  lastMarker = response.Marker;
} while (lastMarker != null);
```

위의 코드 조각에서 요청 시에 `Limit`값을 지정하지 않으면 Amazon Glacier가 Amazon Glacier API의 설정에 따라 최대 10개까지 볼트를 반환합니다.

참고로 목록의 각 볼트마다 반환되는 정보는 특정 볼트에서 `DescribeVault` 메서드를 호출하여 가져오는 정보와 동일합니다.

> **Note**
>
> `ListVaults` 메서드는 기본 REST API를 호출합니다(볼트 목록 조회(GET vaults) 참조).

예:의 하위 수준 API를 사용하여 볼트 메타데이터 검색 AWS SDK for .NET

유효 코드 예제를 보려면 예:의 하위 수준 API를 사용한 볼트 작업 AWS SDK for .NET 섹션을 참조하세요. C# 코드 예제는 볼트를 생성한 후 볼트 메타데이터를 가져옵니다.

REST API를 사용하여 볼트 메타데이터 검색

REST API를 사용하여 볼트 목록을 조회하려면 볼트 목록 조회(GET vaults) 섹션을 참조하세요. 하나의 볼트에 대해 설명하려면 볼트 설명(GET vault) 섹션을 참조하세요.

AWS Command Line Interface를 사용하여 Amazon Glacier에서 볼트 메타데이터 검색

이 예시에서는 AWS Command Line Interface(AWS CLI)를 사용하여 Amazon Glacier(Amazon Glacier)에서 볼트 정보 및 메타데이터를 검색하는 방법을 보여줍니다.

주제

- (전제 조건) AWS CLI 설정
- 예제: AWS CLI를 사용하여 볼트 메타데이터 검색

(전제 조건) AWS CLI 설정

1. AWS CLI를 다운로드하고 구성합니다. 관련 지침은 AWS Command Line Interface 사용 설명서에서 다음 토픽을 참조하세요.

   AWS Command Line Interface 설치

   AWS Command Line Interface 구성

2. 명령 프롬프트에 다음 명령을 입력하여 사용자의 AWS CLI 설정을 확인합니다. 이러한 명령은 명시적으로 자격 증명을 제공하지 않으므로 기본 프로파일의 자격 증명이 사용됩니다.

   - help 명령을 사용해 보십시오.

     ```
     aws help
     ```

   - 구성된 계정의 Amazon Glacier 볼트 목록을 가져오려면 `list-vaults` 명령을 사용합니다. *123456789012*를 사용자의 AWS 계정 ID로 바꿉니다.

     ```
     aws glacier list-vaults --account-id 123456789012
     ```

   - AWS CLI를 위한 현재 구성데이터를 보려면 `aws configure list` 명령을 사용합니다.

     ```
     aws configure list
     ```

예제: AWS CLI를 사용하여 볼트 메타데이터 검색

- `describe-vault` 명령을 사용하여 *111122223333* 계정에 있는 *awsexamplevault*라는 이름의 볼트를 설명하세요.

  ```
  aws glacier describe-vault --vault-name awsexamplevault --account-id 111122223333
  ```

Amazon Glacier에서 볼트 인벤토리 다운로드

첫 번째 아카이브를 볼트에 업로드하면 Amazon Glacier(Amazon Glacier)가 자동으로 볼트 인벤토리를 생성한 후 대략 하루에 한 번씩 업데이트합니다. Amazon Glacier가 첫 번째 인벤토리를 생성한 후 해당 인벤토리를 가져올 수 있을 때까지 일반적으로 반나절에서 하루가 걸립니다. Amazon Glacier에서 볼트 인벤토리를 가져오려면 다음 2단계 프로세스를 따릅니다.

1. 작업 시작(POST jobs) 작업을 사용하여 인벤토리 가져오기 작업을 시작합니다.

   > **Important**
   >
   > 데이터 가져오기 정책은 `PolicyEnforcedException` 예외에 따라 가져오기 작업 시작 요청이 오류로 중단되는 원인이 될 수 있습니다. 데이터 가져오기 정책에 대한 자세한 내용은

Amazon Glacier 데이터 검색 정책 섹션을 참조하세요. `PolicyEnforcedException` 예외에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

2. 작업이 완료된 후 작업 출력 가져오기(GET output) 작업을 사용하여 바이트를 다운로드합니다.

예를 들어 아카이브 또는 볼트 인벤토리를 가져오려면 가져오기 작업을 먼저 시작해야 합니다. 이때 작업 요청은 비동기식으로 실행됩니다. 검색 작업을 시작하면 Amazon Glacier가 작업을 생성한 후 응답으로 작업 ID를 반환합니다. Amazon Glacier가 가져오기 작업을 마치면 사용자는 작업 출력, 아카이브 바이트 또는 볼트 인벤토리 데이터를 다운로드할 수 있습니다.

작업 출력을 가져오려면 먼저 작업을 완료해야 합니다. 작업 상태는 다음 옵션을 사용해 확인할 수 있습니다.

- 작업 완료 알림 대기: 작업이 완료되면 Amazon Glacier가 알림 메시지를 게시할 수 있도록 Amazon Simple Notification Service(Amazon SNS) 토픽을 지정할 수 있습니다. Amazon SNS 토픽을 지정하는 방법은 다음과 같습니다.
  - 작업을 기준으로 Amazon SNS 토픽을 지정합니다.

    작업을 시작할 때 선택적으로 Amazon SNS 토픽을 지정할 수 있습니다.
  - 볼트에 알림 구성을 설정합니다.

    볼트의 특정 이벤트에 대해 알림 구성을 설정할 수 있습니다(Amazon Glacier의 볼트 알림 구성 섹션 참조). Amazon Glacier는 특정 이벤트가 발생할 때마다 지정된 SNS 토픽에 메시지를 보냅니다.

  볼트에서 알림 구성을 설정하고, 작업을 시작할 때 Amazon SNS 토픽까지 지정했다면, Amazon Glacier는 두 토픽 모두에 작업 완료 메시지를 전송합니다.

  SNS 토픽을 이메일을 통해 알려주도록 혹은 애플리케이션이 폴링할 수 있는 Amazon Simple Queue Service(Amazon SQS)에 메시지를 저장하도록 구성할 수 있습니다. 메시지가 대기열에 표시되면 작업이 성공적으로 완료되었는지 먼저 확인한 후 작업 출력을 다운로드합니다.
- 명시적인 작업 정보 요청: Amazon Glacier는 작업 정보를 폴링할 수 있도록 작업 설명(작업 설명(GET JobID))도 제공합니다. 주기적으로 이 요청을 전송하여 작업 정보를 가져올 수 있습니다. 그러나 Amazon SNS의 알림 메시지의 사용을 권장합니다.

> **Note**
>
> SNS 알림을 통해 가져오는 정보는 작업 설명을 호출하여 가져오는 정보와 동일합니다.

주제

- 인벤토리 정보
- 를 사용하여 Amazon Glacier에서 볼트 인벤토리 다운로드 AWS SDK for Java
- 를 사용하여 Amazon Glacier에서 볼트 인벤토리 다운로드 AWS SDK for .NET
- REST API를 사용하여 볼트 인벤토리 다운로드
- 를 사용하여 Amazon Glacier에서 볼트 인벤토리 다운로드 AWS Command Line Interface

인벤토리 정보

Amazon Glacier는 아카이브를 볼트에 처음 업데이트한 날짜부터 최소한 하루에 한 번씩 볼트 인벤토리를 업데이트합니다. 마지막 인벤토리 이후 볼트에 대한 아카이브 추가 또는 삭제가 없는 경우에는 인벤토리 데이터가 업데이트되지 않습니다. 볼트 인벤토리 작업이 시작되면 Amazon Glacier는 마지막으로 생성한 인벤토리, 즉 실시간 데이터가 아닌 특정 시점 스냅샷을 반환합니다. Amazon Glacier가 첫 번째 볼트 인벤토리를 생성한 후 해당 인벤토리를 가져올 수 있을 때까지 일반적으로 반나절에서 하루가 걸립니다.

아카이브를 업로드할 때마다 볼트 인벤토리를 가져오는 것이 불필요하다고 생각할 수도 있습니다. 그러나 Amazon Glacier에 업로드하는 아카이브에 대한 메타데이터를 연결하여 클라이언트 측에서 데이터베이스를 관리한다고 가정해보겠습니다. 볼트 인벤토리가 필요에 따라 데이터베이스 정보와 실제 볼트 인벤토리를 서로 조정하는 데 얼마나 유용한지 알게 될 것입니다. 아카이브 생성 날짜를 기준으로 필터링하거나 할당량을 설정하여 가져오는 인벤토리 항목 수를 제한할 수 있습니다. 인벤토리 가져오기 제한에 대한 자세한 내용은 범위가 지정된 인벤토리 가져오기 섹션을 참조하세요.

인벤토리는 쉼표로 분리된 값(CSV) 또는 JSON의 두 가지 형식으로 반환될 수 있습니다. 이 형식은 인벤토리 작업을 시작할 때 옵션으로 지정할 수 있습니다. 기본 형식은 JSON입니다. 인벤토리 작업 출력에서 반환되는 데이터 필드에 대한 자세한 내용은 Get Job Output API의 응답 본문 섹션을 참조하세요.

를 사용하여 Amazon Glacier에서 볼트 인벤토리 다운로드 AWS SDK for Java

다음은 AWS SDK for Java의 로우레벨 API를 사용해 볼트 인벤토리를 가져오는 단계입니다. 하이레벨 API는 볼트 인벤토리를 가져오도록 지원하지 않습니다.

1. `AmazonGlacierClient` 클래스(클라이언트)의 인스턴스를 만듭니다.

   볼트가 있는 AWS 리전을 지정해야 합니다. 이 클라이언트를 사용하여 수행하는 모든 작업은 해당 AWS 리전에 적용됩니다.

2. `initiateJob` 메서드를 실행하여 인벤토리 가져오기 작업을 시작합니다.

   `InitiateJobRequest` 객체에 작업 정보를 제공하여 `initiateJob`을 실행합니다.

   > **Note**
   >
   > 볼트의 인벤토리가 완료되지 않은 경우 오류가 반환된다는 점에 유의하세요. Amazon Glacier(Amazon Glacier)는 각 볼트마다 24시간을 주기로 인벤토리를 준비합니다.

   Amazon Glacier는 응답으로 작업 ID를 반환합니다. 이 응답은 `InitiateJobResult` 클래스 인스턴스에서 사용할 수 있습니다.

```
InitiateJobRequest initJobRequest = new InitiateJobRequest()
    .withVaultName("*** provide vault name ***")
    .withJobParameters(
            new JobParameters()
                .withType("inventory-retrieval")
                .withSNSTopic("*** provide SNS topic ARN ****")
      );

InitiateJobResult initJobResult = client.initiateJob(initJobRequest);
String jobId = initJobResult.getJobId();
```

3. 작업이 완료될 때까지 기다립니다.

   작업 출력을 다운로드할 수 있을 때까지 기다려야 합니다. 볼트에서 알림 구성을 설정하거나 작업을 시작할 때 Amazon Simple Notification Service(Amazon SNS) 토픽을 지정했다면 Amazon Glacier가 작업 완료 후 해당 토픽에 메시지를 보냅니다.

   `describeJob` 메서드를 직접 호출하여 Amazon Glacier를 폴링함으로써 작업 완료 상태를 확인하는 방법도 있습니다. 하지만 Amazon SNS 토픽을 사용해 식별하는 방법을 권장합니다. 다음 섹션에서 제공하는 코드 예시는 Amazon Glacier가 메시지를 게시할 수 있도록 Amazon SNS를 사용합니다.

4. `getJobOutput` 메서드를 실행하여 작업 출력(볼트 인벤토리 데이터)을 다운로드합니다.

   `GetJobOutputRequest` 클래스 인스턴스를 생성하여 계정 ID, 작업 ID 및 볼트 이름을 입력합니다. 계정 ID를 입력하지 않는 경우에는 요청 서명을 위해 입력하는 자격 증명과 연결되어 있는 계정 ID를 사용합니다. 자세한 내용은 Amazon Glacier AWS SDK for Java 에서 사용 단원을 참조하십시오.

   Amazon Glacier가 반환하는 출력은 `GetJobOutputResult` 객체에서 사용할 수 있습니다.

```
GetJobOutputRequest jobOutputRequest = new GetJobOutputRequest()
        .withVaultName("*** provide vault name ***")
        .withJobId("*** provide job ID ***");
GetJobOutputResult jobOutputResult = client.getJobOutput(jobOutputRequest);
// jobOutputResult.getBody(); provides the output stream.
```

> **Note**
>
> 작업과 관련된 기본 REST API에 대한 자세한 내용은 작업 섹션을 참조하세요.

예: Amazon SDK for Java를 사용하여 볼트 인벤토리 검색

다음은 볼트를 지정하여 볼트 인벤토리를 가져오는 Java 코드 예제입니다.

이 예에서는 다음과 같은 작업을 수행합니다.

- Amazon Simple Notification Service(Amazon SNS) 토픽을 생성합니다.

  Amazon Glacier는 작업 완료 후 해당 토픽에 알림을 전송합니다.
- Amazon Simple Queue Service(Amazon SQS) 대기열을 생성합니다.

  아래 예시는 Amazon SNS 토픽이 메시지를 대기열에 게시할 수 있도록 여기에서 해당 대기열에 정책을 연결합니다.
- 지정된 아카이브의 다운로드 작업을 시작합니다.

  작업 요청에서 Amazon Glacier가 작업 완료 후 알림 메시지를 해당 토픽에 게시할 수 있도록 생성된 Amazon SNS 토픽이 지정됩니다.

- 작업 ID가 포함된 메시지가 있는지 Amazon SQS 대기열을 확인합니다.

  메시지가 있으면 JSON 구문을 분석하여 작업이 성공적으로 완료되었는지 확인합니다. 성공적으로 완료되었으면 아카이브를 다운로드합니다.

- Amazon SNS 토픽과 생성된 Amazon SQS 대기열을 삭제하여 정리합니다.

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.core.JsonParseException;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;

import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Principal;
import com.amazonaws.auth.policy.Resource;
import com.amazonaws.auth.policy.Statement;
import com.amazonaws.auth.policy.Statement.Effect;
import com.amazonaws.auth.policy.actions.SQSActions;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.model.GetJobOutputRequest;
import com.amazonaws.services.glacier.model.GetJobOutputResult;
import com.amazonaws.services.glacier.model.InitiateJobRequest;
import com.amazonaws.services.glacier.model.InitiateJobResult;
import com.amazonaws.services.glacier.model.JobParameters;
import com.amazonaws.services.sns.AmazonSNSClient;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.CreateTopicResult;
import com.amazonaws.services.sns.model.DeleteTopicRequest;
import com.amazonaws.services.sns.model.SubscribeRequest;
import com.amazonaws.services.sns.model.SubscribeResult;
import com.amazonaws.services.sns.model.UnsubscribeRequest;
```

```
import com.amazonaws.services.sqs.AmazonSQSClient;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.CreateQueueResult;
import com.amazonaws.services.sqs.model.DeleteQueueRequest;
import com.amazonaws.services.sqs.model.GetQueueAttributesRequest;
import com.amazonaws.services.sqs.model.GetQueueAttributesResult;
import com.amazonaws.services.sqs.model.Message;
import com.amazonaws.services.sqs.model.ReceiveMessageRequest;
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;


public class AmazonGlacierDownloadInventoryWithSQSPolling {

    public static String vaultName = "*** provide vault name ***";
    public static String snsTopicName = "*** provide topic name ***";
    public static String sqsQueueName = "*** provide queue name ***";
    public static String sqsQueueARN;
    public static String sqsQueueURL;
    public static String snsTopicARN;
    public static String snsSubscriptionARN;
    public static String fileName = "*** provide file name ***";
    public static String region = "*** region ***";
    public static long sleepTime = 600;
    public static AmazonGlacierClient client;
    public static AmazonSQSClient sqsClient;
    public static AmazonSNSClient snsClient;

    public static void main(String[] args) throws IOException {

     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        client = new AmazonGlacierClient(credentials);
        client.setEndpoint("https://glacier." + region + ".amazonaws.com");
        sqsClient = new AmazonSQSClient(credentials);
        sqsClient.setEndpoint("https://sqs." + region + ".amazonaws.com");
        snsClient = new AmazonSNSClient(credentials);
        snsClient.setEndpoint("https://sns." + region + ".amazonaws.com");

        try {
            setupSQS();

            setupSNS();

            String jobId = initiateJobRequest();
```

```
            System.out.println("Jobid = " + jobId);

            Boolean success = waitForJobToComplete(jobId, sqsQueueURL);
            if (!success) { throw new Exception("Job did not complete
successfully."); }

            downloadJobOutput(jobId);

            cleanUp();

        } catch (Exception e) {
            System.err.println("Inventory retrieval failed.");
            System.err.println(e);
        }
    }

    private static void setupSQS() {
        CreateQueueRequest request = new CreateQueueRequest()
            .withQueueName(sqsQueueName);
        CreateQueueResult result = sqsClient.createQueue(request);
        sqsQueueURL = result.getQueueUrl();

        GetQueueAttributesRequest qRequest = new GetQueueAttributesRequest()
            .withQueueUrl(sqsQueueURL)
            .withAttributeNames("QueueArn");

        GetQueueAttributesResult qResult = sqsClient.getQueueAttributes(qRequest);
        sqsQueueARN = qResult.getAttributes().get("QueueArn");

        Policy sqsPolicy =
            new Policy().withStatements(
                    new Statement(Effect.Allow)
                    .withPrincipals(Principal.AllUsers)
                    .withActions(SQSActions.SendMessage)
                    .withResources(new Resource(sqsQueueARN)));
        Map<String, String> queueAttributes = new HashMap<String, String>();
        queueAttributes.put("Policy", sqsPolicy.toJson());
        sqsClient.setQueueAttributes(new SetQueueAttributesRequest(sqsQueueURL,
queueAttributes));

    }
    private static void setupSNS() {
        CreateTopicRequest request = new CreateTopicRequest()
            .withName(snsTopicName);
```

```
        CreateTopicResult result = snsClient.createTopic(request);
        snsTopicARN = result.getTopicArn();

        SubscribeRequest request2 = new SubscribeRequest()
            .withTopicArn(snsTopicARN)
            .withEndpoint(sqsQueueARN)
            .withProtocol("sqs");
        SubscribeResult result2 = snsClient.subscribe(request2);

        snsSubscriptionARN = result2.getSubscriptionArn();
    }
    private static String initiateJobRequest() {

        JobParameters jobParameters = new JobParameters()
            .withType("inventory-retrieval")
            .withSNSTopic(snsTopicARN);

        InitiateJobRequest request = new InitiateJobRequest()
            .withVaultName(vaultName)
            .withJobParameters(jobParameters);

        InitiateJobResult response = client.initiateJob(request);

        return response.getJobId();
    }

    private static Boolean waitForJobToComplete(String jobId, String sqsQueueUrl)
throws InterruptedException, JsonParseException, IOException {

        Boolean messageFound = false;
        Boolean jobSuccessful = false;
        ObjectMapper mapper = new ObjectMapper();
        JsonFactory factory = mapper.getFactory();

        while (!messageFound) {
            List<Message> msgs = sqsClient.receiveMessage(
               new
ReceiveMessageRequest(sqsQueueUrl).withMaxNumberOfMessages(10)).getMessages();

            if (msgs.size() > 0) {
                for (Message m : msgs) {
                    JsonParser jpMessage = factory.createJsonParser(m.getBody());
                    JsonNode jobMessageNode = mapper.readTree(jpMessage);
                    String jobMessage = jobMessageNode.get("Message").textValue();
```

```
                    JsonParser jpDesc = factory.createJsonParser(jobMessage);
                    JsonNode jobDescNode = mapper.readTree(jpDesc);
                    String retrievedJobId = jobDescNode.get("JobId").textValue();
                    String statusCode = jobDescNode.get("StatusCode").textValue();
                    if (retrievedJobId.equals(jobId)) {
                        messageFound = true;
                        if (statusCode.equals("Succeeded")) {
                            jobSuccessful = true;
                        }
                    }
                }

            } else {
              Thread.sleep(sleepTime * 1000);
            }
          }
        return (messageFound && jobSuccessful);
    }

    private static void downloadJobOutput(String jobId) throws IOException {

        GetJobOutputRequest getJobOutputRequest = new GetJobOutputRequest()
            .withVaultName(vaultName)
            .withJobId(jobId);
        GetJobOutputResult getJobOutputResult =
client.getJobOutput(getJobOutputRequest);

        FileWriter fstream = new FileWriter(fileName);
        BufferedWriter out = new BufferedWriter(fstream);
        BufferedReader in = new BufferedReader(new
InputStreamReader(getJobOutputResult.getBody()));
        String inputLine;
        try {
            while ((inputLine = in.readLine()) != null) {
                out.write(inputLine);
            }
        }catch(IOException e) {
            throw new AmazonClientException("Unable to save archive", e);
        }finally{
            try {in.close();}  catch (Exception e) {}
            try {out.close();}  catch (Exception e) {}
        }
        System.out.println("Retrieved inventory to " + fileName);
```

```
    }

    private static void cleanUp() {
        snsClient.unsubscribe(new UnsubscribeRequest(snsSubscriptionARN));
        snsClient.deleteTopic(new DeleteTopicRequest(snsTopicARN));
        sqsClient.deleteQueue(new DeleteQueueRequest(sqsQueueURL));
    }
}
```

를 사용하여 Amazon Glacier에서 볼트 인벤토리 다운로드 AWS SDK for .NET

다음은 AWS SDK for .NET의 로우레벨 API를 사용해 볼트 인벤토리를 가져오는 단계입니다. 하이레벨 API는 볼트 인벤토리를 가져오도록 지원하지 않습니다.

1. `AmazonGlacierClient` 클래스(클라이언트)의 인스턴스를 만듭니다.

   볼트가 있는 AWS 리전을 지정해야 합니다. 이 클라이언트를 사용하여 수행하는 모든 작업은 해당 AWS 리전에 적용됩니다.
2. `InitiateJob` 메서드를 실행하여 인벤토리 가져오기 작업을 시작합니다.

   사용자는 `InitiateJobRequest` 객체에 작업 정보를 제공합니다. Amazon Glacier(Amazon Glacier)는 응답으로 작업 ID를 반환합니다. 이 응답은 `InitiateJobResponse` 클래스 인스턴스에서 사용할 수 있습니다.

```
AmazonGlacierClient client;
client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2);

InitiateJobRequest initJobRequest = new InitiateJobRequest()
{
  VaultName = vaultName,
  JobParameters = new JobParameters()
  {
    Type = "inventory-retrieval",
    SNSTopic = "*** Provide Amazon SNS topic arn ***",
  }
};
InitiateJobResponse initJobResponse = client.InitiateJob(initJobRequest);
string jobId = initJobResponse.JobId;
```

3. 작업이 완료될 때까지 기다립니다.

   작업 출력을 다운로드할 수 있을 때까지 기다려야 합니다. 볼트에서 Amazon Simple Notification Service(Amazon SNS) 토픽을 식별할 수 있도록 알림 구성을 설정했거나 작업을 시작할 때 Amazon SNS 토픽을 지정했다면 Amazon Glacier가 작업 완료 후 해당 토픽에 메시지를 보냅니다. 다음 섹션에서 제공하는 코드 예시는 Amazon Glacier가 메시지를 게시할 수 있도록 Amazon SNS를 사용합니다.

   `DescribeJob` 메서드를 직접 호출하여 Amazon Glacier를 폴링함으로써 작업 완료 상태를 확인하는 방법도 있습니다. 그러나 Amazon SNS 토픽의 알림을 사용해 식별하는 방법을 권장합니다.

4. `GetJobOutput` 메서드를 실행하여 작업 출력(볼트 인벤토리 데이터)을 다운로드합니다.

   `GetJobOutputRequest` 클래스 인스턴스를 생성하여 계정 ID, 볼트 이름 및 작업 ID 정보를 입력합니다. 계정 ID를 입력하지 않는 경우에는 요청 서명을 위해 입력하는 자격 증명과 연결되어 있는 계정 ID로 간주합니다. 자세한 내용은 Amazon Glacier와 함께 AWS SDK for .NET 사용 단원을 참조하십시오.

   Amazon Glacier가 반환하는 출력은 `GetJobOutputResponse` 객체에서 사용할 수 있습니다.

```
GetJobOutputRequest getJobOutputRequest = new GetJobOutputRequest()
{
  JobId = jobId,
  VaultName = vaultName
};

GetJobOutputResponse getJobOutputResponse =
 client.GetJobOutput(getJobOutputRequest);
using (Stream webStream = getJobOutputResponse.Body)
{
   using (Stream fileToSave = File.OpenWrite(fileName))
   {
     CopyStream(webStream, fileToSave);
   }
}
```

> **Note**
>
> 작업과 관련된 기본 REST API에 대한 자세한 내용은 작업 섹션을 참조하세요.

예:의 하위 수준 API를 사용하여 볼트 인벤토리 검색 AWS SDK for .NET

다음은 볼트를 지정하여 볼트 인벤토리를 가져오는 C# 코드 예제입니다.

이 예에서는 다음과 같은 작업을 수행합니다.

- Amazon SNS 토픽을 설정합니다.

  Amazon Glacier는 작업 완료 후 해당 토픽에 알림을 전송합니다.
- Amazon SQS 대기열을 생성합니다.

  아래 예시는 Amazon SNS 토픽이 메시지를 게시할 수 있도록 여기에서 해당 대기열에 정책을 연결합니다.
- 지정된 아카이브의 다운로드 작업을 시작합니다.

  아래 예시는 Amazon Glacier가 작업 완료 후 메시지를 전송할 수 있도록 작업 요청에서 Amazon SNS 토픽을 지정합니다.
- Amazon SQS 대기열의 메시지를 주기적으로 확인합니다.

  메시지가 있으면 JSON 구문을 분석하여 작업이 성공적으로 완료되었는지 확인합니다. 성공적으로 완료되었으면 아카이브를 다운로드합니다. 아래 코드 예제에서는 JSON.NET 라이브러리(JSON.NET 참조)를 사용하여 JSON 구문을 분석합니다.
- 생성된 Amazon SNS 토픽과 Amazon SQS 대기열을 삭제하여 정리합니다.

Example

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading;
using Amazon.Glacier;
using Amazon.Glacier.Model;
using Amazon.Glacier.Transfer;
using Amazon.Runtime;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;
using Amazon.SQS;
using Amazon.SQS.Model;
using Newtonsoft.Json;
```

```
namespace glacier.amazon.com.docsamples
{
  class VaultInventoryJobLowLevelUsingSNSSQS
  {
    static string topicArn;
    static string queueUrl;
    static string queueArn;
    static string vaultName = "*** Provide vault name ***";
    static string fileName  = "*** Provide file name and path where to store inventory
 ***";
    static AmazonSimpleNotificationServiceClient snsClient;
    static AmazonSQSClient sqsClient;
    const string SQS_POLICY =
        "{" +
        "    \"Version\" : \"2012-10-17\",&TCX5-2025-waiver;" +
        "    \"Statement\" : [" +
        "        {" +
        "            \"Sid\" : \"sns-rule\"," +
        "            \"Effect\" : \"Allow\"," +
        "            \"Principal\" : {\"AWS\" : \"arn:aws:iam::123456789012:root\" },"
 +
        "            \"Action\"    : \"sqs:SendMessage\"," +
        "            \"Resource\"  : \"{QuernArn}\"," +
        "            \"Condition\" : {" +
        "                \"ArnLike\" : {" +
        "                    \"aws:SourceArn\" : \"{TopicArn}\"" +
        "                }" +
        "            }" +
        "        }" +
        "    ]" +
        "}";

    public static void Main(string[] args)
    {
      AmazonGlacierClient client;
      try
      {
        using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2))
        {
            Console.WriteLine("Setup SNS topic and SQS queue.");
            SetupTopicAndQueue();
            Console.WriteLine("To continue, press Enter"); Console.ReadKey();

            Console.WriteLine("Retrieve Inventory List");
```

```
            GetVaultInventory(client);
        }
        Console.WriteLine("Operations successful.");
        Console.WriteLine("To continue, press Enter"); Console.ReadKey();
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      finally
      {
       // Delete SNS topic and SQS queue.
       snsClient.DeleteTopic(new DeleteTopicRequest() { TopicArn = topicArn });
       sqsClient.DeleteQueue(new DeleteQueueRequest() { QueueUrl = queueUrl });
      }
    }

    static void SetupTopicAndQueue()
    {
      long ticks = DateTime.Now.Ticks;

      // Setup SNS topic.
      snsClient = new
AmazonSimpleNotificationServiceClient(Amazon.RegionEndpoint.USWest2);
      sqsClient = new AmazonSQSClient(Amazon.RegionEndpoint.USWest2);

      topicArn = snsClient.CreateTopic(new CreateTopicRequest { Name =
"GlacierDownload-" + ticks }).TopicArn;
      Console.Write("topicArn: "); Console.WriteLine(topicArn);

      CreateQueueRequest createQueueRequest =  new CreateQueueRequest();
      createQueueRequest.QueueName = "GlacierDownload-" + ticks;
      CreateQueueResponse createQueueResponse =
sqsClient.CreateQueue(createQueueRequest);
      queueUrl = createQueueResponse.QueueUrl;
      Console.Write("QueueURL: "); Console.WriteLine(queueUrl);

      GetQueueAttributesRequest getQueueAttributesRequest = new
GetQueueAttributesRequest();
      getQueueAttributesRequest.AttributeNames = new List<string> { "QueueArn" };
      getQueueAttributesRequest.QueueUrl = queueUrl;
      GetQueueAttributesResponse response =
sqsClient.GetQueueAttributes(getQueueAttributesRequest);
      queueArn = response.QueueARN;
      Console.Write("QueueArn: ");Console.WriteLine(queueArn);
```

```
      // Setup the Amazon SNS topic to publish to the SQS queue.
      snsClient.Subscribe(new SubscribeRequest()
      {
        Protocol = "sqs",
        Endpoint = queueArn,
        TopicArn = topicArn
      });

      // Add the policy to the queue so SNS can send messages to the queue.
      var policy = SQS_POLICY.Replace("{TopicArn}", topicArn).Replace("{QuernArn}",
queueArn);

      sqsClient.SetQueueAttributes(new SetQueueAttributesRequest()
      {
          QueueUrl = queueUrl,
          Attributes = new Dictionary<string, string>
          {
              { QueueAttributeName.Policy, policy }
          }
      });

   }

   static void GetVaultInventory(AmazonGlacierClient client)
   {
     // Initiate job.
     InitiateJobRequest initJobRequest = new InitiateJobRequest()
     {
       VaultName = vaultName,
       JobParameters = new JobParameters()
       {
         Type = "inventory-retrieval",
         Description = "This job is to download a vault inventory.",
         SNSTopic = topicArn,
       }
     };

     InitiateJobResponse initJobResponse = client.InitiateJob(initJobRequest);
     string jobId = initJobResponse.JobId;

     // Check queue for a message and if job completed successfully, download
inventory.
     ProcessQueue(jobId, client);
```

```
    }

    private static void ProcessQueue(string jobId, AmazonGlacierClient client)
    {
      ReceiveMessageRequest receiveMessageRequest = new ReceiveMessageRequest()
{ QueueUrl = queueUrl, MaxNumberOfMessages = 1 };
      bool jobDone = false;
      while (!jobDone)
      {
        Console.WriteLine("Poll SQS queue");
        ReceiveMessageResponse receiveMessageResponse =
sqsClient.ReceiveMessage(receiveMessageRequest);
        if (receiveMessageResponse.Messages.Count == 0)
        {
          Thread.Sleep(10000 * 60);
          continue;
        }
        Console.WriteLine("Got message");
        Message message = receiveMessageResponse.Messages[0];
        Dictionary<string, string> outerLayer =
JsonConvert.DeserializeObject<Dictionary<string, string>>(message.Body);
        Dictionary<string, object> fields =
JsonConvert.DeserializeObject<Dictionary<string, object>>(outerLayer["Message"]);
        string statusCode = fields["StatusCode"] as string;

        if (string.Equals(statusCode, GlacierUtils.JOB_STATUS_SUCCEEDED,
StringComparison.InvariantCultureIgnoreCase))
        {
          Console.WriteLine("Downloading job output");
          DownloadOutput(jobId, client); // Save job output to the specified file
location.
        }
        else if (string.Equals(statusCode, GlacierUtils.JOB_STATUS_FAILED,
StringComparison.InvariantCultureIgnoreCase))
          Console.WriteLine("Job failed... cannot download the inventory.");

        jobDone = true;
        sqsClient.DeleteMessage(new DeleteMessageRequest() { QueueUrl = queueUrl,
ReceiptHandle = message.ReceiptHandle });
      }
    }

    private static void DownloadOutput(string jobId, AmazonGlacierClient client)
    {
```

```
      GetJobOutputRequest getJobOutputRequest = new GetJobOutputRequest()
      {
        JobId = jobId,
        VaultName = vaultName
      };

      GetJobOutputResponse getJobOutputResponse =
 client.GetJobOutput(getJobOutputRequest);
      using (Stream webStream = getJobOutputResponse.Body)
      {
        using (Stream fileToSave = File.OpenWrite(fileName))
        {
          CopyStream(webStream, fileToSave);
        }
      }
    }

    public static void CopyStream(Stream input, Stream output)
    {
      byte[] buffer = new byte[65536];
      int length;
      while ((length = input.Read(buffer, 0, buffer.Length)) > 0)
      {
        output.Write(buffer, 0, length);
      }
    }
  }
}
```

REST API를 사용하여 볼트 인벤토리 다운로드

REST API를 사용하여 볼트 인벤토리를 다운로드하려면

볼트 인벤토리 다운로드는 2단계 프로세스입니다.

1. `inventory-retrieval` 유형의 작업을 시작합니다. 자세한 내용은 작업 시작(POST jobs) 단원을 참조하십시오.
2. 작업이 완료되면 인벤토리 데이터를 다운로드합니다. 자세한 내용은 작업 출력 가져오기(GET output) 단원을 참조하십시오.

를 사용하여 Amazon Glacier에서 볼트 인벤토리 다운로드 AWS Command Line Interface

다음 단계에 따라 AWS Command Line Interface (AWS CLI)를 사용하여 Amazon Glacier(Amazon Glacier)의 볼트 인벤토리를 다운로드합니다.

주제

- (사전 조건) 설정 AWS CLI
- 예:를 사용하여 볼트 인벤토리 다운로드 AWS CLI

(사전 조건) 설정 AWS CLI

1. AWS CLI를 다운로드하고 구성합니다. 관련 지침은 AWS Command Line Interface 사용 설명서에서 다음 토픽을 참조하세요.

   설치 AWS Command Line Interface

   구성 AWS Command Line Interface
2. 명령 프롬프트에 다음 명령을 입력하여 AWS CLI 설정을 확인합니다. 이러한 명령은 명시적으로 자격 증명을 제공하지 않으므로 기본 프로파일의 자격 증명이 사용됩니다.

   - help 명령을 사용해 보십시오.

     ```
     aws help
     ```

   - 구성된 계정의 Amazon Glacier 볼트 목록을 가져오려면 `list-vaults` 명령을 사용합니다. *123456789012*을 AWS 계정 ID로 바꿉니다.

     ```
     aws glacier list-vaults --account-id 123456789012
     ```

   - 에 대한 현재 구성 데이터를 보려면 `aws configure list` 명령을 AWS CLI사용합니다.

     ```
     aws configure list
     ```

예:를 사용하여 볼트 인벤토리 다운로드 AWS CLI

1. `initiate-job` 명령을 사용하여 인벤토리 검색 작업을 시작합니다.

```
aws glacier initiate-job --vault-name awsexamplevault --account-id 111122223333 --
job-parameters='{"Type": "inventory-retrieval"}'
```

예상 결과:

```
{
    "location": "/111122223333/vaults/awsexamplevault/jobs/*** jobid ***",
    "jobId": "*** jobid ***"
}
```

2. describe-job 명령을 사용하여 이전 검색 작업의 상태를 확인합니다.

```
aws glacier describe-job --vault-name awsexamplevault --account-id 111122223333 --
job-id *** jobid ***
```

예상 결과:

```
{
    "InventoryRetrievalParameters": {
        "Format": "JSON"
    },
    "VaultARN": "*** vault arn ***",
    "Completed": false,
    "JobId": "*** jobid ***",
    "Action": "InventoryRetrieval",
    "CreationDate": "*** job creation date ***",
    "StatusCode": "InProgress"
}
```

3. 작업이 완료될 때까지 기다립니다.

   작업 출력을 다운로드할 수 있을 때까지 기다려야 합니다. 작업 ID는 Amazon Glacier가 작업을 완료한 후 최소 24시간 동안 만료되지 않습니다. 볼트에서 알림 구성을 설정하거나 작업을 시작할 때 Amazon Simple Notification Service(Amazon SNS) 토픽을 지정했다면 Amazon Glacier가 작업 완료 후 해당 토픽에 메시지를 보냅니다.

볼트의 특정 이벤트에 대해 알림 구성을 설정할 수 있습니다. 자세한 내용은 Amazon Glacier의 볼트 알림 구성 단원을 참조하십시오. Amazon Glacier는 특정 이벤트가 발생할 때마다 지정된 SNS 토픽에 메시지를 보냅니다.

4. 완료되면 `get-job-output` 명령을 사용하여 검색 작업을 `output.json` 파일로 다운로드합니다.

```
aws glacier get-job-output --vault-name awsexamplevault --account-id 111122223333
 --job-id *** jobid *** output.json
```

이 명령은 다음 필드가 있는 파일을 생성합니다.

```
{
"VaultARN":"arn:aws:glacier:region:111122223333:vaults/awsexamplevault",
"InventoryDate":"*** job completion date ***",
"ArchiveList":[
{"ArchiveId":"*** archiveid ***",
"ArchiveDescription":"*** archive description (if set) ***",
"CreationDate":"*** archive creation date ***",
"Size":"*** archive size (in bytes) ***",
"SHA256TreeHash":"*** archive hash ***"
}
{"ArchiveId":
...
]}
```

Amazon Glacier의 볼트 알림 구성

볼트 또는 볼트 인벤토리의 아카이브를 검색하는 등 Amazon Glacier에서 무언가를 검색하는 작업은 2단계로 구성됩니다.

1. 검색 작업을 시작합니다.
2. 작업이 완료되면 작업 출력을 다운로드합니다.

작업 완료 시 메시지가 Amazon Simple Notification Service(Amazon SNS)의 토픽에 전송되도록 볼트의 알림 구성을 설정할 수 있습니다.

주제

- Amazon Glacier의 볼트 알림 구성: 일반적인 개념
- 를 사용하여 Amazon Glacier에서 볼트 알림 구성 AWS SDK for Java
- 를 사용하여 Amazon Glacier에서 볼트 알림 구성 AWS SDK for .NET
- REST API를 사용하여 Amazon Glacier의 볼트 알림 구성
- Amazon Glacier 콘솔을 사용하여 볼트 알림 구성
- 를 사용하여 볼트 알림 구성 AWS Command Line Interface

Amazon Glacier의 볼트 알림 구성: 일반적인 개념

Amazon Glacier의 검색 작업 요청은 비동기식으로 실행됩니다. 따라서 출력을 얻으려면 Amazon Glacier가 작업을 완료할 때까지 기다려야 합니다. Amazon Glacier를 주기적으로 폴링하여 작업 상태를 알 수도 있지만 가장 좋은 방법은 아닙니다. Amazon Glacier는 알림 기능도 지원하기 때문입니다. 작업이 완료되면 작업은 Amazon Simple Notification Service(Amazon SNS) 토픽에 메시지를 게시할 수 있습니다. 이 기능을 사용하기 위해서는 먼저 볼트의 알림 구성을 설정해야 합니다. 알림을 구성할 때는 이벤트 한개 이상과 이벤트 발생 시 Amazon Glacier가 메시지를 전송할 Amazon SNS 토픽을 지정합니다.

Amazon Glacier는 특히 볼트의 알림 구성에 추가할 수 있는 작업 완료 (`ArchiveRetrievalCompleted`, `InventoryRetrievalCompleted`)와 관련된 이벤트를 정의합니다. 특정 작업이 완료되면 Amazon Glacier가 알림 메시지를 SNS 토픽에 게시합니다.

알림 구성은 다음 예제와 같이 JSON 문서입니다.

```
{
   "SNSTopic": "arn:aws:sns:us-west-2:012345678901:mytopic",
   "Events": ["ArchiveRetrievalCompleted", "InventoryRetrievalCompleted"]
}
```

단, 각 볼트에 대해 1개의 Amazon SNS 토픽만 구성할 수 있습니다.

> **Note**
>
> 알림 구성을 볼트에 추가하면 알림 구성에서 지정한 이벤트가 발생할 때마다 Amazon Glacier가 알림 메시지를 전송합니다. 또한 작업 시작을 요청할 때마다 옵션으로 Amazon SNS 토픽을

지정할 수 있습니다. 볼트에서 알림 구성을 설정하고, 작업 시작을 요청할 때 Amazon SNS 토픽까지 지정하는 경우에는 Amazon Glacier는 두 알림 메시지를 모두 전송합니다.

Amazon Glacier가 전송하는 작업 완료 메시지에는 작업 유형(`InventoryRetrieval`, `ArchiveRetrieval`), 작업 완료 상태, SNS 토픽 이름, 작업 상태 코드, 볼트 ARN이 포함됩니다. 다음은 `InventoryRetrieval` 작업 완료 후 Amazon Glacier가 SNS 토픽에 전송한 알림 메시지 예시입니다.

```
{
 "Action": "InventoryRetrieval",
 "ArchiveId": null,
 "ArchiveSizeInBytes": null,
 "Completed": true,
 "CompletionDate": "2012-06-12T22:20:40.790Z",
 "CreationDate": "2012-06-12T22:20:36.814Z",
 "InventorySizeInBytes":11693,
 "JobDescription": "my retrieval job",
 "JobId":"HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID",
 "SHA256TreeHash":null,
 "SNSTopic": "arn:aws:sns:us-west-2:012345678901:mytopic",
 "StatusCode":"Succeeded",
 "StatusMessage": "Succeeded",
 "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault"
}
```

`Completed` 필드가 true인 경우에는 `StatusCode`에서 작업의 성공적인 완료 여부까지 확인해야 합니다.

> **Note**
>
> 볼트가 알림 메시지를 Amazon SNS 토픽에 게시할 수 있어야 합니다. 기본적으로 Amazon SNS 토픽 소유자만 메시지를 토픽에 게시할 수 있습니다. 그러나 Amazon SNS 주제와 볼트를 서로 다른에서 소유 AWS 계정한 경우 볼트의 게시를 수락하도록 Amazon SNS 주제를 구성해야 합니다. Amazon SNS 토픽 정책은 Amazon SNS 콘솔에서 구성할 수 있습니다.

Amazon SNS에 대한 자세한 내용은 Amazon SNS 시작하기를 참조하세요.

를 사용하여 Amazon Glacier에서 볼트 알림 구성 AWS SDK for Java

다음은 AWS SDK for Java의 로우레벨 API를 사용해 볼트 알림을 구성하는 단계입니다.

1. `AmazonGlacierClient` 클래스(클라이언트)의 인스턴스를 만듭니다.

   볼트가 있는 AWS 리전을 지정해야 합니다. 이 클라이언트를 사용하여 수행하는 모든 작업은 해당 AWS 리전에 적용됩니다.

2. `SetVaultNotificationsRequest` 클래스 인스턴스를 생성하여 알림 구성 정보를 입력합니다.

   볼트 이름과 알림 구성 정보, 그리고 계정 ID를 입력해야 합니다. 알림 구성을 지정할 때 기존 Amazon SNS 토픽의 Amazon 리소스 이름(ARN)과 알림 메시지를 원하는 한 개 이상의 이벤트를 입력합니다. 지원되는 이벤트 목록은 볼트 알림 구성 설정(PUT notification-configuration) 섹션을 참조하세요.

3. 요청 객체를 파라미터로 입력하여 `setVaultNotifications` 메서드를 실행합니다.

다음은 위에서 설명한 단계를 나타내는 Java 코드 조각입니다. 이 코드 조각은 볼트의 알림 구성을 설정합니다. `ArchiveRetrievalCompleted` 이벤트 또는 `InventoryRetrievalCompleted` 이벤트가 발생할 경우 구성에 따라 Amazon Glacier(Amazon Glacier)가 지정된 Amazon SNS 토픽에 알림 메시지를 전송하도록 요청합니다.

```
SetVaultNotificationsRequest request = new SetVaultNotificationsRequest()
        .withAccountId("-")
        .withVaultName("*** provide vault name ***")
        .withVaultNotificationConfig(
                new VaultNotificationConfig()
                .withSNSTopic("*** provide SNS topic ARN ***")
                .withEvents("ArchiveRetrievalCompleted", "InventoryRetrievalCompleted")
         );
client.setVaultNotifications(request);
```

> **Note**
>
> 기본 REST API에 대한 자세한 내용은 볼트 작업 섹션을 참조하세요.

예:를 사용하여 볼트에서 알림 구성 설정 AWS SDK for Java

다음은 볼트의 알림 구성을 설정했다가 삭제한 후 다시 복원하는 Java 코드 예제입니다. 다음 예제의 실행을 위한 단계별 지침은 Amazon Glacier AWS SDK for Java 에서 사용 섹션을 참조하세요.

Example

```
import java.io.IOException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.model.DeleteVaultNotificationsRequest;
import com.amazonaws.services.glacier.model.GetVaultNotificationsRequest;
import com.amazonaws.services.glacier.model.GetVaultNotificationsResult;
import com.amazonaws.services.glacier.model.SetVaultNotificationsRequest;
import com.amazonaws.services.glacier.model.VaultNotificationConfig;


public class AmazonGlacierVaultNotifications {

    public static AmazonGlacierClient client;
    public static String vaultName = "*** provide vault name ****";
    public static String snsTopicARN = "*** provide sns topic ARN ***";

    public static void main(String[] args) throws IOException {

     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        client = new AmazonGlacierClient(credentials);
        client.setEndpoint("https://glacier.us-east-1.amazonaws.com/");

        try {

            System.out.println("Adding notification configuration to the vault.");
            setVaultNotifications();
            getVaultNotifications();
            deleteVaultNotifications();

        } catch (Exception e) {
            System.err.println("Vault operations failed." + e.getMessage());
        }
    }
```

```
    private static void setVaultNotifications() {
        VaultNotificationConfig config = new VaultNotificationConfig()
            .withSNSTopic(snsTopicARN)
            .withEvents("ArchiveRetrievalCompleted", "InventoryRetrievalCompleted");

        SetVaultNotificationsRequest request = new SetVaultNotificationsRequest()
                .withVaultName(vaultName)
                .withVaultNotificationConfig(config);

        client.setVaultNotifications(request);
        System.out.println("Notification configured for vault: " + vaultName);
    }

    private static void getVaultNotifications() {
        VaultNotificationConfig notificationConfig = null;
        GetVaultNotificationsRequest request = new GetVaultNotificationsRequest()
                .withVaultName(vaultName);
        GetVaultNotificationsResult result = client.getVaultNotifications(request);
        notificationConfig = result.getVaultNotificationConfig();

        System.out.println("Notifications configuration for vault: "
                + vaultName);
        System.out.println("Topic: " + notificationConfig.getSNSTopic());
        System.out.println("Events: " + notificationConfig.getEvents());
    }

    private static void deleteVaultNotifications() {
            DeleteVaultNotificationsRequest request = new
 DeleteVaultNotificationsRequest()
                .withVaultName(vaultName);
            client.deleteVaultNotifications(request);
            System.out.println("Notifications configuration deleted for vault: " +
 vaultName);
    }
}
```

를 사용하여 Amazon Glacier에서 볼트 알림 구성 AWS SDK for .NET

다음은 AWS SDK for .NET의 로우레벨 API를 사용해 볼트 알림을 구성하는 단계입니다.

1. AmazonGlacierClient 클래스(클라이언트)의 인스턴스를 만듭니다.

볼트가 있는 AWS 리전을 지정해야 합니다. 이 클라이언트를 사용하여 수행하는 모든 작업은 해당 AWS 리전에 적용됩니다.

2. `SetVaultNotificationsRequest` 클래스 인스턴스를 생성하여 알림 구성 정보를 입력합니다.

   볼트 이름과 알림 구성 정보, 그리고 계정 ID를 입력해야 합니다. 계정 ID를 입력하지 않는 경우에는 요청 서명을 위해 입력하는 자격 증명과 연결되어 있는 계정 ID로 간주합니다. 자세한 내용은 Amazon Glacier와 함께 AWS SDK for .NET 사용 단원을 참조하십시오.

   알림 구성을 지정할 때, 기존 Amazon SNS 토픽의 Amazon 리소스 이름(ARN)과 알림 메시지 수신을 원하는 한 개 이상의 이벤트를 입력합니다. 지원되는 이벤트 목록은 볼트 알림 구성 설정(PUT notification-configuration) 섹션을 참조하세요.
3. 요청 객체를 파라미터로 입력하여 `SetVaultNotifications` 메서드를 실행합니다.
4. 볼트 알림 구성을 설정한 후에는 `GetVaultNotifications` 메서드를 호출하여 구성 정보를 가져오거나, 혹은 클라이언트에서 제공하는 `DeleteVaultNotifications` 메서드를 호출하여 제거할 수 있습니다.

예:를 사용하여 볼트에서 알림 구성 설정 AWS SDK for .NET

다음 C# 코드 예제에서는 이전 단계를 설명합니다. 예제에서는 미국 서부(오리건) 리전에 속한 볼트('`examplevault`')의 알림 구성을 설정하고 구성을 검색한 후 삭제합니다. `ArchiveRetrievalCompleted` 이벤트 또는 `InventoryRetrievalCompleted` 이벤트가 발생할 경우 구성에 따라 Amazon Glacier(Amazon Glacier)가 지정된 Amazon SNS 토픽에 알림 메시지를 전송하도록 요청합니다.

> **Note**
>
> 기본 REST API에 대한 자세한 내용은 볼트 작업 섹션을 참조하세요.

다음 예제를 실행하기 위한 단계별 지침은 코드 예제 실행 섹션을 참조하세요. 예시와 같이 코드를 업데이트한 후 기존 볼트 이름과 Amazon SNS 토픽을 입력해야 합니다.

Example

```
using System;
using System.Collections.Generic;
using Amazon.Glacier;
using Amazon.Glacier.Model;
```

```
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
  class VaultNotificationSetGetDelete
  {
    static string vaultName   = "examplevault";
    static string snsTopicARN = "*** Provide Amazon SNS topic ARN ***";

    static IAmazonGlacier client;

    public static void Main(string[] args)
    {
      try
      {
        using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2))
        {
          Console.WriteLine("Adding notification configuration to the vault.");
          SetVaultNotificationConfig();
          GetVaultNotificationConfig();
          Console.WriteLine("To delete vault notification configuration, press Enter");
          Console.ReadKey();
          DeleteVaultNotificationConfig();
        }
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }

    static void SetVaultNotificationConfig()
    {

      SetVaultNotificationsRequest request = new SetVaultNotificationsRequest()
      {
        VaultName = vaultName,
        VaultNotificationConfig = new VaultNotificationConfig()
        {
          Events   = new List<string>() { "ArchiveRetrievalCompleted",
 "InventoryRetrievalCompleted" },
          SNSTopic = snsTopicARN
        }
```

```
      };
      SetVaultNotificationsResponse response = client.SetVaultNotifications(request);
    }

    static void GetVaultNotificationConfig()
    {
      GetVaultNotificationsRequest request = new GetVaultNotificationsRequest()
      {
        VaultName = vaultName,
        AccountId = "-"
      };
      GetVaultNotificationsResponse response = client.GetVaultNotifications(request);
      Console.WriteLine("SNS Topic ARN: {0}",
  response.VaultNotificationConfig.SNSTopic);
      foreach (string s in response.VaultNotificationConfig.Events)
        Console.WriteLine("Event : {0}", s);
    }

    static void DeleteVaultNotificationConfig()
    {
      DeleteVaultNotificationsRequest request = new DeleteVaultNotificationsRequest()
      {
        VaultName = vaultName
      };
      DeleteVaultNotificationsResponse response =
  client.DeleteVaultNotifications(request);
    }
  }
}
```

REST API를 사용하여 Amazon Glacier의 볼트 알림 구성

REST API를 사용하여 볼트 알림을 구성하는 방법에 대한 자세한 내용은 볼트 알림 구성 설정(PUT notification-configuration) 섹션을 참조하세요. 그 밖에 볼트 알림을 가져오거나(볼트 알림 가져오기(GET notification-configuration)), 혹은 볼트 알림을 삭제할 수도 있습니다(볼트 알림 삭제(DELETE notification-configuration)).

Amazon Glacier 콘솔을 사용하여 볼트 알림 구성

이 섹션에서는 Amazon Glacier 콘솔을 사용하여 볼트 알림을 구성하는 방법을 설명합니다. 알림을 구성할 때, Amazon Simple Notification Service(Amazon SNS) 토픽에 알림을 보내는 작업 완료 이벤트를 지정합니다. 볼트 알림을 구성하는 것 외에도 작업을 시작할 때 알림 메시지를 게시할 토픽을 지정

할 수도 있습니다. 특정 이벤트에 대해 알림 메시지를 게시하도록 볼트를 구성하고, 동시에 작업 시작 요청에서 알림 메시지를 구성할 경우에는 두 알림 메시지가 모두 전송됩니다.

볼트 알림을 구성하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/glacier/home Amazon Glacier 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 볼트를 선택합니다.
3. 볼트 리스트에서 볼트를 선택합니다.
4. 알림 섹션에서 편집을 선택합니다.
5. 이벤트 알림 페이지에서 알림 켜기를 선택합니다.
6. 알림 섹션에서 다음 Amazon Simple Notification Service(SNS) 옵션 중 하나를 선택한 다음 해당 단계를 따르세요.

Amazon SNS 옵션	작업
새 SNS 토픽 생성	1. 새 SNS 토픽 생성을 선택합니다. 2. 토픽 이름에 새 토픽의 이름을 입력합니다. 이름은 최대 256자까지 지정할 수 있습니다. 혼합 문자와 하이픈(-), 밑줄(_)을 사용할 수 있습니다. 주제 이름은 계정 및 내에서 고유해야 합니다 AWS 리전. 3. (선택 사항) SMS 메시지를 사용하여 토픽을 구독하려면 표시 이름에 이름을 입력하세요. 각 표시 이름은 최대 100자까지 가능합니다.
기존 SNS 토픽 선택	1. 기존 SNS 토픽 선택을 선택합니다. 2. SNS 토픽 지정의 항목에서 다음 옵션 중 하나를 선택합니다. • 사용자의 SNS 토픽 중에서 선택

Amazon SNS 옵션	작업
	SNS 토픽 드롭다운 목록이 나타납니다. 드롭다운 목록에서 기존 토픽을 선택합니다. • SNS 토픽 ARN 입력 Amazon SNS 토픽 ARN 텍스트 상자가 나타납니다. 사용자의 SNS 토픽에 Amazon 리소스 이름(ARN)을 입력합니다. SNS 토픽 ARN은 다음과 같은 형식을 가집니다. `arn:aws:sns:` *`region`*`:`*`account-id`*`:`*`topic-name`* Amazon SNS 콘솔에서 SNS 토픽 ARN을 찾을 수 있습니다.

7. 이벤트의 항목에서 알림을 보내려는 이벤트를 하나 혹은 두개 전부 선택합니다.

   - 아카이브 검색 작업이 완료된 경우에만 알림을 보내려면 아카이브 검색 작업완료를 선택합니다.
   - 볼트 인벤토리 작업이 완료된 경우에만 알림을 보내려면 볼트 인벤토리 검색 작업 완료를 선택합니다.

를 사용하여 볼트 알림 구성 AWS Command Line Interface

본 섹션에서는 AWS Command Line Interface를 사용하여 볼트 알림을 구성하는 방법을 설명합니다. 알림을 구성할 때는 알림 메시지를 Amazon Simple Notification Service(SNS) 토픽으로 트리거하는 작업 완료 이벤트를 지정합니다. 볼트 알림을 구성하는 것 외에도 작업을 시작할 때 알림 메시지를 게시할 주제를 지정할 수도 있습니다. 특정 이벤트에 대해 알림 메시지를 게시하도록 볼트를 구성하고, 동시에 작업 시작 요청에서 알림 메시지를 지정할 경우에는 두 알림 메시지가 모두 전송됩니다.

다음 단계에 따라 AWS CLI를 사용하여 볼트 알림을 구성하세요.

주제

- (사전 조건) 설정 AWS CLI
- 예:를 사용하여 볼트 알림 구성 AWS CLI

(사전 조건) 설정 AWS CLI

1. AWS CLI를 다운로드하고 구성합니다. 관련 지침은 AWS Command Line Interface 사용 설명서에서 다음 토픽을 참조하세요.

   설치 AWS Command Line Interface

   구성 AWS Command Line Interface
2. 명령 프롬프트에 다음 명령을 입력하여 AWS CLI 설정을 확인합니다. 이러한 명령은 명시적으로 자격 증명을 제공하지 않으므로 기본 프로파일의 자격 증명이 사용됩니다.

   - help 명령을 사용해 보십시오.

     ```
     aws help
     ```

   - 구성된 계정의 Amazon Glacier 볼트 목록을 가져오려면 `list-vaults` 명령을 사용합니다. *123456789012*을 AWS 계정 ID로 바꿉니다.

     ```
     aws glacier list-vaults --account-id 123456789012
     ```

   - 에 대한 현재 구성 데이터를 보려면 `aws configure list` 명령을 AWS CLI사용합니다.

     ```
     aws configure list
     ```

예:를 사용하여 볼트 알림 구성 AWS CLI

1. `set-vault-notifications` 명령을 사용하여 볼트에 특정 이벤트가 발생할 때 전송될 알림을 구성할 수 있습니다. 기본적으로는 어떤 알림도 전송되지 않습니다.

   ```
   aws glacier set-vault-notifications --vault-name examplevault --account-
   id 111122223333 --vault-notification-config file://notificationconfig.json
   ```

2. 알림 구성은 다음 예제와 같이 JSON 문서입니다.

   ```
   {
   ```

```
    "SNSTopic": "arn:aws:sns:us-west-2:012345678901:mytopic",
    "Events": ["ArchiveRetrievalCompleted", "InventoryRetrievalCompleted"]
}
```

Amazon Glacier에서 Amazon SNS 토픽을 사용하는 방법에 대한 자세한 내용은 Amazon Glacier의 볼트 알림 구성: 일반적인 개념을 참조하세요.

Amazon SNS에 대한 자세한 내용은 Amazon SNS 시작하기를 참조하세요.

Amazon Glacier에서 볼트 삭제

Amazon Glacier(Amazon Glacier)는 마지막으로 계산된 인벤토리를 기준으로 볼트에 아카이브가 없고, 마지막 인벤토리 이후 볼트에 쓰기 작업이 없었던 경우에 한해 볼트를 삭제합니다. 아카이브 삭제에 대한 자세한 내용은 Amazon Glacier에서 아카이브 삭제 섹션을 참조하세요. 볼트 인벤토리 다운로드에 대한 자세한 내용은 Amazon Glacier에서 볼트 인벤토리 다운로드 섹션을 참조하세요.

> **Note**
>
> Amazon Glacier는 각 볼트마다 24시간을 주기로 인벤토리를 준비합니다. 인벤토리에 최신 정보가 반영되지 않을 수도 있기 때문에 Amazon Glacier는 마지막 볼트 인벤토리 이후 쓰기 작업의 유무를 검사하여 볼트가 실제로 비어있는지 확인합니다.

> **Note**
>
> 볼트 아카이브의 자동 삭제는 Amazon S3 Glacier에서 볼트 아카이브의 자동 삭제를 참조하세요.

주제

- AWS SDK for Java를 사용하여 Amazon Glacier에서 볼트 삭제
- AWS SDK for .NET를 사용하여 Amazon Glacier에서 볼트 삭제
- REST API를 사용하여 Amazon Glacier에서 볼트 삭제
- Amazon Glacier 콘솔을 사용하여 빈 볼트 삭제
- 를 사용하여 Amazon Glacier에서 볼트 삭제 AWS Command Line Interface

AWS SDK for Java를 사용하여 Amazon Glacier에서 볼트 삭제

다음은 AWS SDK for Java의 로우레벨 API를 사용해 볼트를 삭제하는 단계입니다.

1. `AmazonGlacierClient` 클래스(클라이언트)의 인스턴스를 만듭니다.

   볼트를 삭제할 AWS 리전을 지정해야 합니다. 이 클라이언트를 사용하여 실행하는 모든 작업이 해당 AWS 리전에 적용됩니다.

2. `DeleteVaultRequest` 클래스 인스턴스를 생성하여 요청 정보를 입력합니다.

   볼트 이름과 계정 ID를 입력해야 합니다. 계정 ID를 입력하지 않는 경우에는 요청 서명을 위해 입력하는 자격 증명과 연결되어 있는 계정 ID로 간주합니다. 자세한 내용은 Amazon Glacier AWS SDK for Java 에서 사용 섹션을 참조하세요.

3. 요청 객체를 파라미터로 입력하여 `deleteVault` 메서드를 실행합니다.

   Amazon Glacier(Amazon Glacier)는 볼트가 비어 있는 경우에만 해당 볼트를 삭제합니다. 자세한 내용은 볼트 삭제(DELETE vault) 섹션을 참조하세요.

다음은 위에서 설명한 단계를 나타내는 Java 코드 조각입니다.

```
try {
    DeleteVaultRequest request = new DeleteVaultRequest()
        .withVaultName("*** provide vault name ***");

    client.deleteVault(request);
    System.out.println("Deleted vault: " + vaultName);
} catch (Exception e) {
    System.err.println(e.getMessage());
}
```

> **Note**
>
> 기본 REST API에 대한 자세한 내용은 볼트 삭제(DELETE vault) 섹션을 참조하세요.

예제: AWS SDK for Java를 사용하여 볼트 삭제

유효 코드 예제를 보려면 예제: AWS SDK for Java를 사용하여 볼트 생성 섹션을 참조하세요. Java 코드 예제는 볼트 생성 및 삭제를 포함하여 기본적인 볼트 작업을 나타냅니다.

AWS SDK for .NET를 사용하여 Amazon Glacier에서 볼트 삭제

Amazon SDK에서 제공하는 .NET용 하이레벨과 로우레벨 API 모두는 볼트를 삭제하는 방법을 제공합니다.

주제

- AWS SDK for .NET의 하이레벨 API를 사용하여 볼트 삭제
- AWS SDK for .NET의 로우레벨 API를 사용하여 볼트 삭제

AWS SDK for .NET의 하이레벨 API를 사용하여 볼트 삭제

하이레벨 API의 `ArchiveTransferManager` 클래스는 볼트를 삭제하는 데 사용할 수 있는 `DeleteVault` 메서드를 제공합니다.

예제: AWS SDK for .NET의 하이레벨 API를 사용하여 볼트 삭제

유효 코드 예제를 보려면 예:의 상위 수준 API를 사용한 볼트 작업 AWS SDK for .NET 섹션을 참조하세요. C# 코드 예제는 볼트 생성 및 삭제를 포함하여 기본적인 볼트 작업을 나타냅니다.

AWS SDK for .NET의 로우레벨 API를 사용하여 볼트 삭제

다음은 AWS SDK for .NET를 사용하여 볼트를 삭제하는 단계입니다.

1. `AmazonGlacierClient` 클래스(클라이언트)의 인스턴스를 만듭니다.

   볼트를 삭제할 AWS 리전을 지정해야 합니다. 이 클라이언트를 사용하여 실행하는 모든 작업이 해당 AWS 리전에 적용됩니다.

2. `DeleteVaultRequest` 클래스 인스턴스를 생성하여 요청 정보를 입력합니다.

   볼트 이름과 계정 ID를 입력해야 합니다. 계정 ID를 입력하지 않는 경우에는 요청 서명을 위해 입력하는 자격 증명과 연결되어 있는 계정 ID로 간주합니다. 자세한 내용은 Amazon Glacier와 함께 AWS SDK for .NET 사용 섹션을 참조하세요.

3. 요청 객체를 파라미터로 입력하여 `DeleteVault` 메서드를 실행합니다.

   Amazon Glacier(Amazon Glacier)는 볼트가 비어 있는 경우에만 해당 볼트를 삭제합니다. 자세한 내용은 볼트 삭제(DELETE vault) 섹션을 참조하세요.

다음은 위에서 설명한 단계를 나타내는 C# 코드 조각입니다. 이 코드 조각은 기본 AWS 리전 내의 볼트에 대한 메타데이터 정보를 검색합니다.

```
AmazonGlacier client;
client = new AmazonGlacierClient(Amazon.RegionEndpoint.USEast1);

DeleteVaultRequest request = new DeleteVaultRequest()
{
  VaultName = "*** provide vault name ***"
};

DeleteVaultResponse response = client.DeleteVault(request);
```

> **Note**
>
> 기본 REST API에 대한 자세한 내용은 볼트 삭제(DELETE vault) 섹션을 참조하세요.

예제: AWS SDK for .NET의 로우레벨 API를 사용하여 볼트 삭제

유효 코드 예제를 보려면 예:의 하위 수준 API를 사용한 볼트 작업 AWS SDK for .NET 섹션을 참조하세요. C# 코드 예제는 볼트 생성 및 삭제를 포함하여 기본적인 볼트 작업을 나타냅니다.

REST API를 사용하여 Amazon Glacier에서 볼트 삭제

REST API를 사용하여 볼트를 삭제하는 방법에 대한 자세한 내용은 볼트 삭제(DELETE vault) 섹션을 참조하세요.

Amazon Glacier 콘솔을 사용하여 빈 볼트 삭제

> **Note**
>
> 볼트를 삭제하기 전에 먼저 기존 아카이브와 볼트를 모두 삭제해야 합니다. REST API, , 또는 AWS Command Line Interface ()를 사용하여 아카이브 삭제 요청을 하는 코드를 작성하여이

작업을 수행할 수 AWS SDK for Java AWS SDK for .NET있습니다AWS CLI. 아카이브 삭제에 대한 자세한 내용은 5단계: Amazon Glacier 볼트에서 아카이브 삭제 섹션을 참조하세요.

볼트를 비운 후 다음 단계에 따라 삭제할 수 있습니다.

Amazon Glacier 콘솔을 사용하여 빈 볼트 삭제

1. 에 로그인 AWS Management Console 하고 Amazon Glacier 콘솔에서 Amazon Glacier 콘솔을 엽니다.
2. 리전 선택에서 볼트 AWS 리전 가 있는를 선택합니다.
3. 왼쪽 탐색 창에서 볼트를 선택합니다.
4. 볼트 목록에서 삭제를 원하는 볼트 이름 옆의 옵션을 선택한 후 페이지 상단에서 삭제를 선택합니다.
5. 볼트 삭제 대화 상자에서 삭제를 선택하여 볼트를 삭제할 것인지 확인합니다.

   > **Important**
   >
   > 볼트 삭제는 실행 취소할 수 없습니다.

6. 볼트를 삭제했는지 확인하려면 볼트 목록을 열고 삭제한 버킷의 이름을 입력합니다. 볼트를 찾을 수 없다면 성공적으로 삭제된 것입니다.

를 사용하여 Amazon Glacier에서 볼트 삭제 AWS Command Line Interface

AWS Command Line Interface (AWS CLI)를 사용하여 Amazon Glacier(Amazon Glacier)에서 빈 볼트 및 비어 있지 않은 볼트를 삭제할 수 있습니다.

주제

- (사전 조건) 설정 AWS CLI
- 예:를 사용하여 빈 볼트 삭제 AWS CLI
- 예:를 사용하여 비어 있지 않은 볼트 삭제 AWS CLI

(사전 조건) 설정 AWS CLI

1. AWS CLI를 다운로드하고 구성합니다. 관련 지침은 AWS Command Line Interface 사용 설명서에서 다음 토픽을 참조하세요.

   설치 AWS Command Line Interface

   구성 AWS Command Line Interface
2. 명령 프롬프트에 다음 명령을 입력하여 AWS CLI 설정을 확인합니다. 이러한 명령은 명시적으로 자격 증명을 제공하지 않으므로 기본 프로파일의 자격 증명이 사용됩니다.

   - help 명령을 사용해 보십시오.

     ```
     aws help
     ```

   - 구성된 계정의 Amazon Glacier 볼트 목록을 가져오려면 `list-vaults` 명령을 사용합니다. *123456789012*을 AWS 계정 ID로 바꿉니다.

     ```
     aws glacier list-vaults --account-id 123456789012
     ```

   - 에 대한 현재 구성 데이터를 보려면 `aws configure list` 명령을 AWS CLI사용합니다.

     ```
     aws configure list
     ```

예:를 사용하여 빈 볼트 삭제 AWS CLI

- `delete-vault` 명령을 사용하여 아카이브가 없는 볼트를 삭제합니다.

  - ```
    aws glacier delete-vault --vault-name awsexamplevault --account-id 111122223333
    ```

예:를 사용하여 비어 있지 않은 볼트 삭제 AWS CLI

Amazon Glacier에서는 마지막으로 계산된 인벤토리를 기준으로 볼트에 아카이브가 없고, 마지막 인벤토리 이후 볼트에 대한 쓰기 작업이 없었던 경우에 한해 해당 볼트를 삭제합니다. 비어 있지 않은 볼트 삭제는 볼트의 인벤토리 보고서에서 아카이브 ID 검색, 각 아카이브 삭제, 볼트 삭제의 3단계로 진행됩니다.

1. `initiate-job` 명령을 사용하여 인벤토리 검색 작업을 시작합니다.
    ```

```
aws glacier initiate-job --vault-name awsexamplevault --account-id 111122223333 --
job-parameters '{"Type": "inventory-retrieval"}'
```

예상 결과:

```
{
    "location": "/111122223333/vaults/awsexamplevault/jobs/*** jobid ***",
    "jobId": "*** jobid ***"
}
```

2. describe-job 명령을 사용하여 이전 검색 작업의 상태를 확인합니다.

```
aws glacier describe-job --vault-name awsexamplevault --account-id 111122223333 --
job-id *** jobid ***
```

예상 결과:

```
{
    "InventoryRetrievalParameters": {
        "Format": "JSON"
    },
    "VaultARN": "*** vault arn ***",
    "Completed": false,
    "JobId": "*** jobid ***",
    "Action": "InventoryRetrieval",
    "CreationDate": "*** job creation date ***",
    "StatusCode": "InProgress"
}
```

3. 작업이 완료될 때까지 기다립니다.

   작업 출력을 다운로드할 수 있을 때까지 기다려야 합니다. 볼트에서 알림 구성을 설정하거나 작업을 시작할 때 Amazon Simple Notification Service(Amazon SNS) 토픽을 지정했다면 Amazon Glacier가 작업 완료 후 해당 토픽에 메시지를 보냅니다.

볼트의 특정 이벤트에 대해 알림 구성을 설정할 수 있습니다. 자세한 내용은 Amazon Glacier의 볼트 알림 구성 단원을 참조하십시오. Amazon Glacier는 특정 이벤트가 발생할 때마다 지정된 SNS 토픽에 메시지를 보냅니다.

4. 완료되면 `get-job-output` 명령을 사용하여 검색 작업을 `output.json` 파일로 다운로드합니다.

```
aws glacier get-job-output --vault-name awsexamplevault --account-id 111122223333
 --job-id *** jobid *** output.json
```

이 명령은 다음 필드가 있는 파일을 생성합니다.

```
{
"VaultARN":"arn:aws:glacier:region:111122223333:vaults/awsexamplevault",
"InventoryDate":"*** job completion date ***",
"ArchiveList":[
{"ArchiveId":"*** archiveid ***",
"ArchiveDescription":*** archive description (if set) ***,
"CreationDate":"*** archive creation date ***",
"Size":"*** archive size (in bytes) ***",
"SHA256TreeHash":"*** archive hash ***"
}
{"ArchiveId":
...
]}
```

5. `delete-archive` 명령을 사용하여 볼트가 비워질 때까지 볼트에서 각 아카이브를 삭제합니다.

```
aws glacier delete-archive --vault-name awsexamplevault --account-id 111122223333
 --archive-id "*** archiveid ***"
```

> **Note**
>
> 아카이브 ID가 하이픈이나 다른 특수 문자로 시작하는 경우, 아카이브 ID를 따옴표로 묶어야 이 명령을 실행할 수 있습니다.

6. `initiate-job` 명령을 사용하여 새 인벤토리 검색 작업을 시작합니다.

```
aws glacier initiate-job --vault-name awsexamplevault --account-id 111122223333 --
job-parameters '{"Type": "inventory-retrieval"}'
```

7. 완료되면 `delete-vault` 명령을 사용하여 아카이브가 없는 볼트를 삭제합니다.

```
aws glacier delete-vault --vault-name awsexamplevault --account-id 111122223333
```

Amazon Glacier 볼트의 태그 지정

사용자 고유의 메타데이터를 태그 형태로 Amazon Glacier 볼트에 할당할 수 있습니다. 태그는 볼트를 나타내도록 정의하는 키-값 페어입니다. 태그 제한을 포함하여 태그에 대한 기본적인 내용은 Amazon Glacier 리소스에 태그 지정 섹션을 참조하세요.

다음 주제에서는 태그를 볼트에 추가하거나, 목록을 조회하거나, 제거하는 방법에 대해서 설명합니다.

주제

- Amazon Glacier 콘솔을 사용하여 볼트에 태그 지정
- AWS CLI를 사용하여 볼트에 태그를 지정
- Amazon Glacier API를 사용하여 볼트에 태그 지정
- 관련 섹션

Amazon Glacier 콘솔을 사용하여 볼트에 태그 지정

Amazon Glacier 콘솔에서 다음 절차에 설명된 대로 태그를 추가하거나, 나열하거나, 제거할 수 있습니다.

볼트에 대한 태그를 확인하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/glacier/home에서 Amazon Glacier 콘솔을 엽니다.
2. 리전 선택의 리전 선택기에서 AWS 리전을 선택합니다.
3. 왼쪽 탐색 창에서 볼트를 선택합니다.
4. 볼트 리스트에서 볼트를 선택합니다.
5. 볼트 속성 탭을 선택합니다. 태그 섹션으로 스크롤하여 볼트와 관련된 태그를 확인합니다.

태그를 볼트에 추가하려면

하나의 볼트에 최대 50개의 태그를 연결할 수 있습니다. 각 볼트에 연결된 태그에는 반드시 고유한 태그 키가 있어야 합니다.

태그 제한에 대한 자세한 내용은 Amazon Glacier 리소스 태그 지정을 참조하세요.

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/glacier/home에서 Amazon Glacier 콘솔을 엽니다.
2. 리전 선택의 리전 선택기에서 AWS 리전을 선택합니다.
3. 왼쪽 탐색 창에서 볼트를 선택합니다.
4. 볼트 목록에서 태그를 추가하려는 볼트의 이름을 선택합니다.
5. 볼트 속성 탭을 선택합니다.
6. 태그 섹션에서 추가를 선택합니다. 태그 추가 페이지가 나타납니다.
7. 태그 추가 페이지의 키 필드에서 태그 키를 지정한 다음, 값 필드에서 선택적으로 태그 값을 지정합니다.
8. 변경 사항 저장을 선택합니다.

태그를 편집하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/glacier/home에서 Amazon Glacier 콘솔을 엽니다.
2. 리전 선택의 리전 선택기에서 AWS 리전을 선택합니다.
3. 왼쪽 탐색 창에서 볼트를 선택합니다.
4. 볼트 목록에서 볼트 이름을 선택합니다.
5. 볼트 속성 탭을 선택한 다음 아래로 스크롤하여 태그 섹션으로 이동합니다.
6. 태그에서 변경을 원하는 태그 옆의 확인란을 선택한 다음 편집을 선택합니다. 그러면 태그 편집 페이지가 나타납니다.
7. 키 필드에서 태그 키를 업데이트하고 선택적으로 값 필드에서 태그 값을 업데이트합니다.
8. 변경 사항 저장을 선택합니다.

볼트에서 태그를 제거하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/glacier/home에서 Amazon Glacier 콘솔을 엽니다.

2. 리전 선택의 리전 선택기에서 AWS 리전을 선택합니다.
3. 왼쪽 탐색 창에서 볼트를 선택합니다.
4. 볼트 목록에서 태그를 제거하기 원하는 볼트의 이름을 선택합니다.
5. 볼트 속성 탭을 선택합니다. 아래로 스크롤하여 태그 섹션을 찾습니다.
6. 태그에서 제거를 원하는 태그 옆의 확인란을 선택한 다음 삭제를 선택합니다.
7. 태그 삭제 대화 상자가 열립니다. 선택한 태그를 삭제하겠다는 의사를 확인하고 삭제를 선택합니다.

AWS CLI를 사용하여 볼트에 태그를 지정

다음 단계에 따라 AWS Command Line Interface(AWS CLI)를 사용해 태그를 추가, 나열 또는 제거할 수 있습니다.

각 태그는 키와 값으로 구성됩니다. 볼트 1개에 추가할 수 있는 태그는 최대 50개입니다.

1. 볼트에 태그를 추가하려면 `add-tags-to-vault` 명령을 사용하세요.

```
aws glacier add-tags-to-vault --vault-name examplevault --account-id 111122223333
 --tags id=1234,date=2020
```

   이 볼트 작업에 대한 자세한 내용은 볼트에 태그 추가를 참조하세요.

2. 볼트에 연결된 모든 태그를 나열하려면 `list-tags-for-vault` 명령을 사용하세요.

```
aws glacier list-tags-for-vault --vault-name examplevault --account-id 111122223333
```

   이 볼트 작업에 대한 자세한 내용은 볼트의 태그 나열을 참조하세요.

3. 볼트에 연결된 태그 세트에서 하나 이상의 태그를 제거하려면 `remove-tags-from-vault` 명령을 사용하세요.

```
aws glacier remove-tags-from-vault --vault-name examplevault --account-
id 111122223333 --tag-keys date
```

   이 볼트 작업에 대한 자세한 내용은 볼트에서 태그 제거를 참조하세요.

Amazon Glacier API를 사용하여 볼트에 태그 지정

Amazon Glacier API를 사용하여 태그를 추가, 나열 및 제거할 수 있습니다. 예제는 다음 설명서를 참조하세요.

볼트에 태그 추가(POST tags add)

지정된 볼트에 대한 태그를 추가 또는 업데이트합니다.

볼트의 태그 목록 조회(GET tags)

지정된 볼트에 대한 태그 목록을 조회합니다.

볼트에서 태그 삭제(POST tags remove)

지정된 볼트에 대한 태그를 제거합니다.

관련 섹션

- Amazon Glacier 리소스에 태그 지정

Amazon Glacier 볼트 잠금

다음 항목에서는 Amazon Glacier에서 볼트를 잠그는 방법과 볼트 잠금 정책을 사용하는 방법을 설명합니다.

주제

- 볼트 잠금 개요
- Amazon Glacier API를 사용하여 볼트 잠금
- 를 사용하여 볼트 잠금 AWS Command Line Interface
- Amazon Glacier 콘솔을 사용하여 볼트 잠금

볼트 잠금 개요

Amazon Glacier 볼트 잠금을 사용하여 각 Amazon Glacier 볼트마다 볼트 잠금 정책을 사용해 규정 준수 제어 항목을 쉽게 배포하고 적용할 수 있습니다. 볼트 잠금 정책에서는 "write once read many"(WORM) 같은 제어 항목을 지정하여 앞으로 편집하지 못하도록 정책을 잠글 수 있습니다.

> **Important**
>
> 볼트 잠금 정책을 잠근 후에는 정책을 더 이상 변경하거나 삭제할 수 없습니다.

Amazon Glacier는 볼트 잠금 정책에 설정된 제어 항목을 적용하여 규정 준수 목표를 달성합니다. 예를 들어 볼트 잠금 정책을 사용하여 데이터 보존을 적용할 수 있습니다. AWS Identity and Access Management (IAM) 정책 언어를 사용하여 볼트 잠금 정책에 다양한 규정 준수 제어를 배포할 수 있습니다. 볼트 잠금 정책에 대한 자세한 내용은 볼트 잠금 정책 섹션을 참조하세요.

볼트 잠금 정책은 볼트 액세스 정책과 다릅니다. 두 정책 모두 볼트에 대한 액세스 제어 항목을 관리하지만 볼트 잠금 정책은 향후 변경할 수 없도록 볼트를 잠근다는 점에서 규정 준수 제어 항목을 강력하게 적용할 수 있습니다. 볼트 잠금 정책을 사용하여 일반적으로 데이터 액세스에 대한 엄격한 제어가 필요한 규제 및 규정 준수 제어 항목을 배포할 수 있습니다.

> **Important**
>
> 먼저 볼트를 만들고 볼트 잠금 정책을 완성한 다음 아카이브를 볼트에 업로드하여 정책을 적용하는 것을 권장합니다.

이와 대조적으로 볼트 액세스 정책은 규정 준수와 관계 없지만 일시적으로 액세스 제어를 구현하는 데 사용되기 때문에 자주 변경될 수 있습니다. 볼트 잠금 정책과 볼트 액세스 정책은 함께 사용할 수 있습니다. 예를 들어 볼트 잠금 정책(삭제 거부)에서 시간에 따른 데이터 보존 규칙을 구현한 후 지정된 타사 또는 비즈니스 파트너에게 (읽기 허용) 볼트 액세스 정책에서 읽기 권한을 부여할 수 있습니다.

볼트 잠금은 다음과 같이 2단계로 구성됩니다.

1. 볼트 잠금 정책을 볼트에 연결하여 잠금을 시작합니다. 그러면 잠금 상태가 진행 중으로 설정되고 잠금 ID를 반환합니다. 정책이 진행 중 상태일 때는 잠금 ID가 만료되기 전 24시간 동안 볼트 잠금 정책이 유효합니다. 볼트가 진행 중 상태를 벗어나지 않도록 하려면 24시간 이내에 볼트 잠금 프로세스를 완료해야 합니다. 그렇지 않으면 볼트 잠금 정책이 삭제됩니다.
2. 잠금 ID를 사용하여 잠금 프로세스를 마칩니다. 볼트 잠금 정책이 예상대로 작동하지 않으면 볼트 잠금 프로세스를 중단한 후 처음부터 다시 시작할 수 있습니다. Amazon Glacier API를 사용하여 볼트를 잠그는 방법에 대한 자세한 내용은 Amazon Glacier API를 사용하여 볼트 잠금 섹션을 참조하세요.

Amazon Glacier API를 사용하여 볼트 잠금

Amazon Glacier API로 볼트를 잠그려면 우선 배포하려는 제어 항목을 지정하는 볼트 잠금 정책으로 볼트 잠금 시작(POST lock-policy)을 직접 호출해야 합니다. `Initiate Vault Lock` 작업은 정책을 사용자의 볼트에 연결하고, 볼트 잠금을 진행 중 상태로 전환하고, 고유한 잠금 ID를 반환합니다. 볼트 잠금 상태가 진행 중 상태이면, `Initiate Vault Lock` 직접 호출에서 반환된 잠금 ID와 함께 볼트 잠금 완료(POST lockId)을 직접 호출하여 이후 24시간 이내에 잠금을 완료합니다.

> **Important**
>
> - 먼저 볼트를 만들고 볼트 잠금 정책을 완성한 다음 아카이브를 볼트에 업로드하여 정책이 적용되도록 하는 것을 권장합니다.
> - 일단 볼트 잠금 정책이 잠기면 더 이상 변경하거나 삭제할 수 없습니다.

진행 중 상태로 전환된 이후 24시간 이내에 볼트 잠금 프로세스를 마치지 않으면 볼트의 진행 중 상태가 자동으로 종료되고 볼트 잠금 정책은 제거됩니다. `Initiate Vault Lock`을 다시 직접 호출하여 새로운 볼트 잠금 정책을 설치하고 진행 중 상태로 전환할 수 있습니다.

진행 중 상태에서는 볼트를 잠그기 전에 볼트 잠금 정책을 테스트할 수 있습니다. 볼트 잠금 정책은 볼트 잠금 중단(DELETE lock-policy)을 직접 호출하여 정책을 제거하는 경우만 아니면 진행 중 상태에서도 마치 볼트가 잠긴 것처럼 완전히 효력을 발휘합니다. 정책을 세부적으로 조정하려면 `Abort Vault Lock` 및 `Initiate Vault Lock` 조합을 필요한 만큼 반복하여 볼트 잠금 정책 변경 사항을 확인할 수 있습니다.

볼트 잠금 정책을 확인한 후에는 가장 최근 잠금 ID와 함께 볼트 잠금 완료(POST lockId)을 직접 호출하여 볼트 잠금 프로세스를 마칩니다. 그러면 볼트가 잠금 상태로 바뀌며, 이후로는 볼트 잠금 정책을 변경할 수 없고 `Abort Vault Lock`을 직접 호출하여 제거할 수도 없습니다.

관련 섹션

- 볼트 잠금 정책
- 볼트 잠금 중단(DELETE lock-policy)
- 볼트 잠금 완료(POST lockId)
- 볼트 잠금 가져오기(GET lock-policy)
- 볼트 잠금 시작(POST lock-policy)

를 사용하여 볼트 잠금 AWS Command Line Interface

를 사용하여 볼트를 잠글 수 있습니다 AWS Command Line Interface. 그러면 지정된 볼트에 볼트 잠금 정책이 설치되고 잠금 ID가 반환됩니다. 볼트 잠금 프로세스는 반드시 24시간 이내에 완료되어야 합니다. 그렇지 않으면 볼트 잠금 정책이 볼트에서 제거됩니다.

(사전 조건) 설정 AWS CLI

1. AWS CLI를 다운로드하고 구성합니다. 관련 지침은 AWS Command Line Interface 사용 설명서에서 다음 토픽을 참조하세요.

   설치 AWS Command Line Interface

   구성 AWS Command Line Interface

2. 명령 프롬프트에 다음 명령을 입력하여 AWS CLI 설정을 확인합니다. 이러한 명령은 명시적으로 자격 증명을 제공하지 않으므로 기본 프로파일의 자격 증명이 사용됩니다.

   - help 명령을 사용해 보십시오.

     ```
     aws help
     ```

   - 구성된 계정의 Amazon Glacier 볼트 목록을 가져오려면 `list-vaults` 명령을 사용합니다. *123456789012*을 AWS 계정 ID로 바꿉니다.

     ```
     aws glacier list-vaults --account-id 123456789012
     ```

   - 에 대한 현재 구성 데이터를 보려면 `aws configure list` 명령을 AWS CLI사용합니다.

     ```
     aws configure list
     ```

1. `initiate-vault-lock`을 사용하여 볼트 잠금 정책을 설치하고 볼트 잠금의 잠금 상태를 `InProgress`으로 설정합니다.

   ```
   aws glacier initiate-vault-lock --vault-name examplevault --account-id 111122223333
    --policy file://lockconfig.json
   ```

2. 잠금 구성은 다음 예시와 같이 JSON 문서입니다. 이 명령을 사용하기 전에 *VAULT_ARN*과 *Principal*을 사용 사례에 적합한 값으로 바꿉니다.

잠그려는 볼트의 ARN을 찾으려면 `list-vaults` 명령을 사용합니다.

```
{"Policy":"{\"Version\":\"2012-10-17\",       \"Statement\":[{\"Sid
\":\"Define-vault-lock\",\"Effect\":\"Deny\",\"Principal\":{\"AWS\":
\"arn:aws:iam::111122223333:root\"},\"Action\":\"glacier:DeleteArchive
\",\"Resource\":\"VAULT_ARN\",\"Condition\":{\"NumericLessThanEquals\":
{\"glacier:ArchiveAgeinDays\":\"365\"}}}]}"}
```

3. 볼트 잠금을 시작하고 나서 반환된 `lockId`를 확인하세요.

```
{
    "lockId": "LOCK_ID"
}
```

볼트 잠금을 완료하려면 24시간 이내에 `complete-vault-lock`을 실행해야 합니다. 그렇지 않으면 볼트 잠금 정책이 볼트에서 제거됩니다.

```
aws glacier complete-vault-lock --vault-name examplevault --account-id 111122223333 --
lock-id LOCK_ID
```

관련 섹션

- AWS CLI 명령 참조의 initiate-vault-lock
- AWS CLI 명령 참조의 list-vaults
- AWS CLI 명령 참조의 complete-vault-lock
- 볼트 잠금 정책
- 볼트 잠금 중단(DELETE lock-policy)
- 볼트 잠금 완료(POST lockId)
- 볼트 잠금 가져오기(GET lock-policy)
- 볼트 잠금 시작(POST lock-policy)

Amazon Glacier 콘솔을 사용하여 볼트 잠금

Amazon Glacier 볼트 잠금을 사용하여 각 Amazon Glacier 볼트마다 볼트 잠금 정책을 사용해 규정 준수 제어 항목을 쉽게 배포하고 적용할 수 있습니다. Amazon Glacier 볼트 잠금에 대한 자세한 내용은 볼트 잠금 정책으로 Amazon Glacier 액세스 제어를 참조하세요.

> **Important**
>
> - 먼저 볼트를 만들고 볼트 잠금 정책을 완성한 다음 아카이브를 볼트에 업로드하여 정책이 적용되도록 하는 것을 권장합니다.
> - 볼트 잠금 정책이 잠기고 나면 더 이상 변경하거나 삭제할 수 없습니다.

Amazon Glacier 콘솔을 사용하여 사용자의 볼트에서 볼트 잠금 정책 시작

볼트 잠금 정책을 사용자의 볼트에 연결하여 잠금을 시작합니다. 그러면 잠금 상태가 진행 중으로 설정되고 잠금 ID를 반환합니다. 정책이 진행 중 상태일 때는 잠금 ID가 만료될 때까지 24시간 동안 볼트 잠금 정책이 유효합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/glacier/home Amazon Glacier 콘솔을 엽니다.
2. 리전 선택의 리전 선택기 AWS 리전 에서를 선택합니다.
3. 왼쪽 탐색 창에서 볼트를 선택합니다.
4. 볼트 페이지에서 볼트 생성을 선택합니다.
5. 볼트를 생성합니다.

   > **Important**
   >
   > 먼저 볼트를 만들고 볼트 잠금 정책을 완료한 다음 아카이브를 볼트에 업로드하여 정책이 적용되도록 하는 것을 권장합니다.

6. 볼트 목록에서 새로운 볼트를 선택합니다.
7. 볼트 정책 탭을 선택합니다.
8. 볼트 잠금 정책 섹션에서 볼트 잠금 정책 시작을 선택합니다.
9. 볼트 잠금 정책 시작 페이지에서 볼트 잠금 정책의 레코드 보존 제어를 표준 텍스트 상자에 텍스트 형식으로 지정합니다.

   > **Note**
   >
   > 볼트 잠금 정책에서 레코드 보존 제어 항목을 텍스트 형식으로 지정하고, `Initiate Vault Lock` API 작업을 직접 호출하거나 Amazon Glacier 콘솔의 대화형 UI를 통해 볼트

잠금을 시작할 수 있습니다. 볼트 잠금 정책 형식에 대한 자세한 내용은 Amazon Glacier 볼트 잠금 정책 예시를 참조하세요.

10. 변경 사항 저장을 선택합니다.
11. 볼트 잠금 ID 레코드 대화 상자에서 사용자의 잠금 ID를 복사하여 안전한 장소에 저장합니다.

> **Important**
>
> 볼트 잠금 정책이 시작되면 24시간 이내에 정책을 확인하고 잠금 프로세스를 완료해야 합니다. 잠금 프로세스를 완료하려면 반드시 잠금 ID를 제공해야 합니다. 24시간 이내에 잠금 ID를 제공하지 않으면 잠금 ID가 만료되고 진행 중인 정책이 삭제됩니다.

12. 잠금 ID를 안전한 곳에 저장한 후 닫기를 선택합니다.
13. 24시간 이내에 볼트 잠금 정책을 테스트합니다. 정책이 의도대로 작동한다면 볼트 잠금 정책 완료를 선택하세요.
14. 볼트 잠금 완료 대화 상자에서 확인란을 선택하여 볼트 잠금 정책 프로세스 완료는 되돌릴 수 없음을 확인합니다.
15. 제공받은 잠금 ID를 텍스트 상자에 입력합니다.
16. 볼트 잠금 완료를 선택합니다.

Amazon Glacier의 아카이브 작업

아카이브란 사진, 동영상, 문서 등 볼트에 저장되는 모든 객체를 말합니다. Amazon Glacier(Amazon Glacier)의 기본 스토리지 단위입니다. 각 아카이브는 고유 ID가 있으며 선택 사항으로 설명을 추가할 수 있습니다. 아카이브를 업로드하면 Amazon Glacier가 아카이브 ID가 포함된 응답을 반환합니다. 이 아카이브 ID는 아카이브가 저장된 AWS 리전에서 고유합니다. 다음은 아카이브 ID의 예제입니다.

```
TJgHcrOSfAkV6hdPqOATYfp_0ZaxL1pIBOc02iZ0gDPMr2ig-
nhwd_PafstsdIf6HSrjHnP-3p6LCJClYytFT_CBhT9CwNxbRaM5MetS3I-
GqwxI3Y8QtgbJbhEQPs0mJ3KExample
```

아카이브 ID의 길이는 138바이트입니다. 또한 아카이브를 업로드할 때 옵션으로 설명까지 입력할 수 있습니다. 단, ID를 사용해 아카이브를 가져올 수 있지만 설명으로는 그렇게 하지 못합니다.

> **Important**
>
> Amazon Glacier는 관리 콘솔을 제공합니다. 이 콘솔에서는 볼트를 생성하거나 삭제할 수 있습니다. 하지만 Amazon Glacier와의 다른 상호작용에서는 AWS Command Line Interface (CLI)를 사용하거나 코드를 작성해야 합니다. 예를 들어 사진, 비디오 및 기타 문서와 같은 데이터를 업로드하려면 AWS CLI 를 사용하거나 REST API를 직접 사용하거나 Amazon SDKs를 사용하여 요청 코드를 작성해야 합니다. 에서 Amazon Glacier를 사용하는 AWS CLI방법에 대한 자세한 내용은 AWS CLI Amazon Glacier 참조를 참조하세요. 를 설치하려면 로 AWS CLI이동합니다AWS Command Line Interface.

주제

- Amazon Glacier에서 아카이브 작업
- 클라이언트 측 아카이브 메타데이터 유지
- Amazon Glacier에 아카이브 업로드
- Amazon Glacier에서 아카이브 다운로드
- Amazon Glacier에서 아카이브 삭제

Amazon Glacier에서 아카이브 작업

Amazon Glacier는 업로드, 다운로드, 삭제 등 기본적인 아카이브 작업을 지원합니다. 아카이브 다운로드는 비동기식 작업입니다.

Amazon Glacier에 아카이브 업로드

아카이브는 단일 작업으로, 혹은 여러 파트로 나누어 업로드할 수 있습니다. 아카이브를 여러 파트로 나누어 업로드할 때 사용하는 API 호출을 멀티파트 업로드라고 부릅니다. 자세한 내용은 Amazon Glacier에 아카이브 업로드 단원을 참조하십시오.

> **Important**
>
> Amazon Glacier는 관리 콘솔을 제공합니다. 이 콘솔에서는 볼트를 생성하거나 삭제할 수 있습니다. 그러나 Amazon Glacier와의 다른 모든 상호 작용에서는 AWS Command Line Interface (CLI)를 사용하거나 코드를 작성해야 합니다. 예를 들어 사진, 비디오 및 기타 문서와 같은 데이터를 업로드하려면 AWS CLI 를 사용하거나 REST API를 직접 사용하거나 Amazon SDKs를 사용하여 요청 코드를 작성해야 합니다. 에서 Amazon Glacier를 사용하는 AWS CLI방법에 대한 자세한 내용은 AWS CLI Amazon Glacier 참조를 참조하세요. 를 설치하려면 로 AWS CLI이동합니다AWS Command Line Interface.

Amazon Glacier에서 아카이브 ID 찾기

아카이브 ID는 해당 아카이브가 포함된 볼트에서 볼트 인벤토리를 다운로드하여 가져올 수 있습니다. 볼트 인벤토리 다운로드에 대한 자세한 내용은 Amazon Glacier에서 볼트 인벤토리 다운로드 섹션을 참조하세요.

Amazon Glacier에서 아카이브 다운로드

아카이브 다운로드는 비동기식 작업입니다. 따라서 먼저 특정 아카이브를 다운로드하는 작업부터 시작해야 합니다. 작업 요청이 수신되면 Amazon Glacier는 아카이브를 다운로드할 수 있도록 준비합니다. 작업이 완료되면 아카이브 데이터를 다운로드할 수 있습니다. 작업이 비동기식으로 이루어지기 때문에 작업 완료 시 Amazon Simple Notification Service(Amazon SNS) 토픽에 알림 메시지를 전송하도록 Amazon Glacier에 요청할 수 있습니다. 각 작업 요청마다 SNS 토픽을 지정하거나, 혹은 특정 이벤트가 발생할 경우 알림 메시지를 전송할 수 있도록 볼트를 구성할 수 있습니다. 아카이브 다운로드에 대한 자세한 내용은 Amazon Glacier에서 아카이브 다운로드 섹션을 참조하세요.

Amazon Glacier에서 아카이브 삭제

Amazon Glacier는 한 번에 하나의 아카이브를 삭제할 수 있는 API 직접 호출을 제공합니다. 자세한 내용은 Amazon Glacier에서 아카이브 삭제 단원을 참조하십시오.

Amazon Glacier에서 아카이브 업데이트

이미 업로드한 아카이브는 내용이나 설명을 업데이트할 수 없습니다. 아카이브 내용이나 설명을 업데이트하려면 먼저 해당 아카이브를 삭제한 후 다른 아카이브를 업로드하는 것이 유일한 방법입니다. 아카이브를 업로드할 때마다 Amazon Glacier가 고유한 아카이브 ID를 반환합니다.

클라이언트 측 아카이브 메타데이터 유지

선택적인 아카이브 설명을 제외하고 Amazon Glacier는 아카이브에 대해 추가적인 메타데이터를 지원하지 않습니다. 아카이브를 업로드할 때 Amazon Glacier가 ID를 할당하지만 불분명한 문자열이기 때문에 아카이브의 의미를 추측할 수 없습니다. 아카이브 메타데이터는 클라이언트 측에서 유지할 수 있습니다. 메타데이터에는 아카이브 이름을 비롯해 아카이브에 대한 유의미한 정보가 포함됩니다.

> **Note**
>
> Amazon Simple Storage Service(Amazon S3) 고객이라면 누구나 객체를 버킷에 업로드할 때 `MyDocument.txt` 또는 `SomePhoto.jpg` 같은 객체 키를 객체에 할당할 수 있습니다. Amazon Glacier에서는 업로드하는 아카이브에 객체 키를 할당할 수 없습니다.

클라이언트 측 아카이브 메타데이터를 유지할 때는 Amazon Glacier가 아카이브 ID와 아카이브 업로드 중 사용자가 입력한 설명을 포함하는 볼트 인벤토리를 유지한다는 점도 알고 있어야 합니다. 간혹 볼트 인벤토리를 다운로드하여 아카이브 메타데이터로 유지하고 있는 클라이언트 측 데이터베이스의 문제를 조정하는 데 사용할 수 있습니다. 그러나 Amazon Glacier는 대략 하루에 한 번씩 볼트 인벤토리를 작성합니다. 볼트 인벤토리를 요청하면 Amazon Glacier가 마지막으로 준비한 인벤토리, 즉 특정 시점의 스냅샷을 반환합니다.

Amazon Glacier에 아카이브 업로드

Amazon Glacier(Amazon Glacier)는 볼트 생성 및 삭제에 사용할 수 있는 관리 콘솔을 제공합니다. 하지만 관리 콘솔을 사용해서 아카이브를 Amazon Glacier에 업로드할 수는 없습니다. 사진, 동영상 및 기

타 문서와 같은 데이터를 업로드하려면 REST API를 직접 AWS CLI 사용하거나 Amazon SDKs를 사용하여를 사용하거나 코드를 작성하여 요청해야 합니다.

에서 Amazon Glacier를 사용하는 AWS CLI방법에 대한 자세한 내용은 AWS CLI Amazon Glacier 참조를 참조하세요. 를 설치하려면 로 AWS CLI이동합니다AWS Command Line Interface. 다음의 업로드 토픽에서는 Amazon SDK for Java, Amazon SDK for .NET 및 REST API를 사용하여 아카이브를 Amazon Glacier에 업로드하는 방법을 설명합니다.

주제

- Amazon Glacier에 아카이브를 업로드하기 위한 옵션
- 아카이브의 단일 작업 업로드
- 대용량 아카이브를 여러 파트로 나누어 업로드(멀티파트 업로드)

Amazon Glacier에 아카이브를 업로드하기 위한 옵션

Amazon Glacier는 업로드하는 데이터의 크기에 따라 다음과 같은 옵션을 제공합니다.

- 아카이브의 단일 작업 업로드: 크기가 1바이트에서 최대 4GB인 아카이브는 단일 작업으로 업로드할 수 있습니다. 하지만 Amazon Glacier 고객이 100MB가 넘는 대용량 아카이브를 업로드할 때는 멀티파트 업로드 기능을 사용할 것을 권장합니다. 자세한 내용은 아카이브의 단일 작업 업로드 단원을 참조하십시오.
- 아카이브의 멀티파트 업로드: 최대 40,000GB(10,000*4GB)의 대용량 아카이브는 멀티파트 업로드 API를 사용하여 업로드할 수 있습니다.

  멀티파트 업로드 API 호출은 대용량 아카이브의 업로드 경험을 개선할 목적으로 설계되었습니다. 여기에서는 아카이브를 여러 파트로 나누어 업로드할 수 있습니다. 이러한 파트들은 순서에 상관없이 각각 병렬 방식으로 업로드됩니다. 멀티파트 업로드가 오류로 중단되더라도 전체 아카이브가 아니라 중단된 파트만 다시 업로드하면 됩니다. 멀티파트 업로드는 크기가 1바이트에서 약 40,000GB에 이르는 아카이브에 사용할 수 있습니다. 자세한 내용은 대용량 아카이브를 여러 파트로 나누어 업로드(멀티파트 업로드) 단원을 참조하십시오.

> **Important**
>
> Amazon Glacier 볼트 인벤토리는 하루 한 번 업데이트됩니다. 따라서 아카이브를 업로드한다고 해서 새로운 아카이브가 볼트에 바로 추가되어 콘솔이나 다운로드한 볼트 인벤토리 목록에 표시되지는 않습니다. 먼저 볼트 인벤토리가 업데이트되어야 합니다.

AWS Snowball Edge 서비스 사용

AWS Snowball Edge 는 인터넷을 우회하여 Amazon 소유 디바이스를 AWS 사용하여 대량의 데이터를 송수신하는 속도를 높입니다. 자세한 내용은 AWS Snowball Edge 세부 정보 페이지를 참조하세요.

기존 데이터를 Amazon Glacier(Amazon Glacier)에 업로드하려면 AWS Snowball Edge 디바이스 유형 중 하나를 사용해서 데이터를 Amazon S3로 가져온 다음, 수명 주기 규칙을 사용하여 아카이브용 Amazon Glacier 스토리지 클래스로 이동하는 것을 고려해 볼 수 있습니다. Amazon S3 객체를 Amazon Glacier 스토리지 클래스로 전환하면, Amazon S3가 내부적으로 Amazon Glacier를 사용하여 보다 저렴한 비용으로 내구성 있는 보관이 가능합니다. 객체가 Amazon Glacier에 저장되어 있지만 이들은 여전히 사용자가 Amazon S3에서 관리하는 Amazon S3 객체이며, 사용자가 Amazon Glacier를 통해 직접 액세스할 수는 없습니다.

Amazon S3 수명 주기 구성 및 Amazon Glacier 스토리지 클래스로의 객체 전환에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 객체 수명 주기 관리 및 객체 전환을 참조하세요.

아카이브의 단일 작업 업로드

Amazon Glacier에 아카이브 업로드 섹션에서 설명한 것과 같이 더 작은 크기의 아카이브도 단일 작업으로 업로드할 수 있습니다. 하지만 Amazon Glacier(Amazon Glacier) 고객이 100MB가 넘는 대용량 아카이브를 업로드할 때는 멀티파트 업로드 기능을 사용하는 것이 권장됩니다.

주제

- 를 사용하여 아카이브를 단일 작업으로 업로드 AWS Command Line Interface
- 를 사용하여 아카이브를 단일 작업으로 업로드 AWS SDK for Java
- Amazon Glacier에서를 사용하여 아카이브를 단일 작업 AWS SDK for .NET 으로 업로드
- REST API를 사용하여 아카이브의 단일 작업 업로드

를 사용하여 아카이브를 단일 작업으로 업로드 AWS Command Line Interface

AWS Command Line Interface ()를 사용하여 Amazon Glacier(Amazon Glacier)에 아카이브를 업로드할 수 있습니다AWS CLI.

주제

- (사전 조건) 설정 AWS CLI
- 예:를 사용하여 아카이브 업로드 AWS CLI

(사전 조건) 설정 AWS CLI

1. AWS CLI를 다운로드하고 구성합니다. 관련 지침은 AWS Command Line Interface 사용 설명서에서 다음 토픽을 참조하세요.

   설치 AWS Command Line Interface

   구성 AWS Command Line Interface
2. 명령 프롬프트에 다음 명령을 입력하여 AWS CLI 설정을 확인합니다. 이러한 명령은 명시적으로 자격 증명을 제공하지 않으므로 기본 프로파일의 자격 증명이 사용됩니다.

   - help 명령을 사용해 보십시오.

     ```
     aws help
     ```

   - 구성된 계정의 Amazon Glacier 볼트 목록을 가져오려면 `list-vaults` 명령을 사용합니다. *123456789012*을 AWS 계정 ID로 바꿉니다.

     ```
     aws glacier list-vaults --account-id 123456789012
     ```

   - 에 대한 현재 구성 데이터를 보려면 `aws configure list` 명령을 AWS CLI사용합니다.

     ```
     aws configure list
     ```

예:를 사용하여 아카이브 업로드 AWS CLI

아카이브를 업로드하려면 반드시 볼트를 생성해야 합니다. 볼트 생성에 대한 자세한 내용은 Amazon Glacier에 볼트 생성 섹션을 참조하세요.

1. `upload-archive` 명령을 사용하여 기존 볼트에 아카이브를 추가합니다. 아래 예시에서는 `vault name` 및 `account ID`를 바꿉니다. body 파라미터에서 업로드하려는 파일의 경로를 지정합니다.

   ```
   aws glacier upload-archive --vault-name awsexamplevault --account-id 123456789012
    --body archive.zip
   ```
2. 예상 결과:

   ```
   {
   ```

```
    "archiveId": "kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--zM_mw6k76ZFGEIWQX-
ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw",
    "checksum": "969fb39823836d81f0cc028195fcdbcbbe76cdde932d4646fa7de5f21e18aa67",
    "location": "/123456789012/vaults/awsexamplevault/archives/
kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--zM_mw6k76ZFGEIWQX-ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw"
}
```

완료되면 명령이 아카이브 ID, 체크섬 및 Amazon Glacier 내의 위치를 출력합니다. upload-archive 명령에 대한 자세한 내용은 AWS CLI 명령 참조의 upload-archive를 참조하세요.

를 사용하여 아카이브를 단일 작업으로 업로드 AWS SDK for Java

Amazon SDK for Java에서 제공하는 하이레벨 및 로우레벨 API 둘 모두는 아카이브를 업로드하는 방법을 제공합니다.

주제

- 의 상위 수준 API를 사용하여 아카이브 업로드 AWS SDK for Java
- 의 하위 수준 API를 사용하여 단일 작업으로 아카이브 업로드 AWS SDK for Java

의 상위 수준 API를 사용하여 아카이브 업로드 AWS SDK for Java

하이레벨 API의 `ArchiveTransferManager` 클래스는 아카이브를 볼트에 업로드하는 데 사용할 수 있는 `upload` 메서드를 제공합니다.

> **Note**
>
> `upload` 메서드는 작든 크든 상관없이 모든 아카이브를 업로드하는 데 사용됩니다. 이 메서드는 업로드하는 아카이브 크기에 따라 아카이브를 단일 작업으로 업로드할지, 혹은 멀티파트 업로드 API를 사용해 아카이브를 여러 파트로 나누어 업로드할지 결정합니다.

예:의 상위 수준 API를 사용하여 아카이브 업로드 AWS SDK for Java

다음은 미국 서부(오레곤) 리전(`us-west-2`)의 볼트(`examplevault`)에 아카이브를 업로드하는 Java 코드 예시입니다. 지원되는 AWS 리전 및 엔드포인트 목록은 섹션을 참조하세요Amazon Glacier 액세스.

이 예제의 실행 방법에 대한 단계별 지침은 Eclipse를 사용하여 Amazon Glacier의 Java 예 실행 섹션을 참조하세요. 아래와 같이 업로드할 볼트 이름과 파일 이름을 사용해 코드를 업데이트해야 합니다.

Example

```
import java.io.File;
import java.io.IOException;
import java.util.Date;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.transfer.ArchiveTransferManager;
import com.amazonaws.services.glacier.transfer.UploadResult;


public class ArchiveUploadHighLevel {
    public static String vaultName = "*** provide vault name ***";
    public static String archiveToUpload = "*** provide name of file to upload ***";

    public static AmazonGlacierClient client;

    public static void main(String[] args) throws IOException {


     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        client = new AmazonGlacierClient(credentials);
        client.setEndpoint("https://glacier.us-west-2.amazonaws.com/");

        try {
            ArchiveTransferManager atm = new ArchiveTransferManager(client,
 credentials);

            UploadResult result = atm.upload(vaultName, "my archive " + (new Date()),
 new File(archiveToUpload));
            System.out.println("Archive ID: " + result.getArchiveId());

        } catch (Exception e)
        {
            System.err.println(e);
        }
    }
}
```

의 하위 수준 API를 사용하여 단일 작업으로 아카이브 업로드 AWS SDK for Java

로우레벨 API는 모든 아카이브 작업에 필요한 메서드를 제공합니다. 다음은 AWS SDK for Java를 사용하여 아카이브를 업로드하는 단계입니다.

1. `AmazonGlacierClient` 클래스(클라이언트)의 인스턴스를 만듭니다.

   아카이브를 업로드할 AWS 리전을 지정해야 합니다. 이 클라이언트를 사용하여 수행하는 모든 작업은 해당 AWS 리전에 적용됩니다.
2. `UploadArchiveRequest` 클래스 인스턴스를 생성하여 요청 정보를 입력합니다.

   업로드할 데이터 외에도 페이로드의 체크섬(SHA256 트리-해시), 볼트 이름, 데이터의 내용 길이, 계정 ID를 입력해야 합니다.

   계정 ID를 입력하지 않는 경우에는 요청 서명을 위해 입력하는 자격 증명과 연결되어 있는 계정 ID로 간주합니다. 자세한 내용은 Amazon Glacier AWS SDK for Java 에서 사용 단원을 참조하십시오.
3. 요청 객체를 파라미터로 입력하여 `uploadArchive` 메서드를 실행합니다.

   그러면 Amazon Glacier(Amazon Glacier)가 응답으로 새롭게 업로드된 아카이브의 ID를 반환합니다.

다음은 위에서 설명한 단계를 나타내는 Java 코드 조각입니다.

```
AmazonGlacierClient client;

UploadArchiveRequest request = new UploadArchiveRequest()
    .withVaultName("*** provide vault name ***")
    .withChecksum(checksum)
    .withBody(new ByteArrayInputStream(body))
    .withContentLength((long)body.length);

UploadArchiveResult uploadArchiveResult = client.uploadArchive(request);

System.out.println("Location (includes ArchiveID): " +
 uploadArchiveResult.getLocation());
```

예:의 하위 수준 API를 사용하여 단일 작업으로 아카이브 업로드 AWS SDK for Java

다음 Java 코드 예제에서는 AWS SDK for Java 를 사용하여 아카이브를 볼트()에 업로드합니다`examplevault`. 이 예제의 실행 방법에 대한 단계별 지침은 Eclipse를 사용하여 Amazon Glacier의

Java 예 실행 섹션을 참조하세요. 아래와 같이 업로드할 볼트 이름과 파일 이름을 사용해 코드를 업데이트해야 합니다.

```
import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.TreeHashGenerator;
import com.amazonaws.services.glacier.model.UploadArchiveRequest;
import com.amazonaws.services.glacier.model.UploadArchiveResult;
public class ArchiveUploadLowLevel {

    public static String vaultName = "*** provide vault name ****";
    public static String archiveFilePath = "*** provide to file upload ****";
    public static AmazonGlacierClient client;

    public static void main(String[] args) throws IOException {

     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        client = new AmazonGlacierClient(credentials);
        client.setEndpoint("https://glacier.us-east-1.amazonaws.com/");

        try {
            // First open file and read.
            File file = new File(archiveFilePath);
            InputStream is = new FileInputStream(file);
            byte[] body = new byte[(int) file.length()];
            is.read(body);

            // Send request.
            UploadArchiveRequest request = new UploadArchiveRequest()
                .withVaultName(vaultName)
                .withChecksum(TreeHashGenerator.calculateTreeHash(new
 File(archiveFilePath)))
                .withBody(new ByteArrayInputStream(body))
                .withContentLength((long)body.length);

            UploadArchiveResult uploadArchiveResult = client.uploadArchive(request);
```

```
            System.out.println("ArchiveID: " + uploadArchiveResult.getArchiveId());

        } catch (Exception e)
        {
            System.err.println("Archive not uploaded.");
            System.err.println(e);
        }
    }
}
```

Amazon Glacier에서를 사용하여 아카이브를 단일 작업 AWS SDK for .NET 으로 업로드

.NET용 Amazon SDK에서 제공하는 하이레벨 및 로우레벨 API 둘 모두는 단일 작업으로 아카이브를 업로드하는 방법을 제공합니다.

주제

- 의 상위 수준 API를 사용하여 아카이브 업로드 AWS SDK for .NET
- 의 하위 수준 API를 사용하여 단일 작업으로 아카이브 업로드 AWS SDK for .NET

의 상위 수준 API를 사용하여 아카이브 업로드 AWS SDK for .NET

하이레벨 API의 `ArchiveTransferManager` 클래스는 아카이브를 볼트에 업로드하는 데 사용할 수 있는 `Upload` 메서드를 제공합니다.

> **Note**
>
> `Upload` 메서드는 작든 크든 상관없이 모든 파일을 업로드하는 데 사용됩니다. 이 메서드는 업로드하는 파일 크기에 따라 아카이브를 단일 작업으로 업로드할지, 혹은 멀티파트 업로드 API를 사용해 파일을 여러 파트로 나누어 업로드할지 결정합니다.

예:의 상위 수준 API를 사용하여 아카이브 업로드 AWS SDK for .NET

다음은 미국 서부(오레곤) 리전의 볼트(`examplevault`)에 아카이브를 업로드하는 C# 코드 예시입니다.

이 예제의 실행 방법에 대한 단계별 지침은 코드 예제 실행 섹션을 참조하세요. 아래와 같이 업로드할 파일 이름을 사용해 코드를 업데이트해야 합니다.

Example

```
using System;
using Amazon.Glacier;
using Amazon.Glacier.Transfer;
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
  class ArchiveUploadHighLevel
  {
    static string vaultName = "examplevault";
    static string archiveToUpload = "*** Provide file name (with full path) to upload
 ***";

    public static void Main(string[] args)
    {
       try
      {
         var manager = new ArchiveTransferManager(Amazon.RegionEndpoint.USWest2);
         // Upload an archive.
         string archiveId = manager.Upload(vaultName, "upload archive test",
 archiveToUpload).ArchiveId;
         Console.WriteLine("Archive ID: (Copy and save this ID for use in other
 examples.) : {0}", archiveId);
         Console.WriteLine("To continue, press Enter");
         Console.ReadKey();
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }
  }
}
```

의 하위 수준 API를 사용하여 단일 작업으로 아카이브 업로드 AWS SDK for .NET

로우레벨 API는 모든 아카이브 작업에 필요한 메서드를 제공합니다. 다음은 AWS SDK for .NET를 사용하여 아카이브를 업로드하는 단계입니다.

1. `AmazonGlacierClient` 클래스(클라이언트)의 인스턴스를 만듭니다.

   아카이브를 업로드할 AWS 리전을 지정해야 합니다. 이 클라이언트를 사용하여 수행하는 모든 작업은 해당 AWS 리전에 적용됩니다.
2. `UploadArchiveRequest` 클래스 인스턴스를 생성하여 요청 정보를 입력합니다.

   업로드할 데이터 외에도 페이로드의 체크섬(SHA256 트리-해시), 볼트 이름, 계정 ID 등을 입력해야 합니다.

   계정 ID를 입력하지 않는 경우에는 요청 서명을 위해 입력하는 자격 증명과 연결되어 있는 계정 ID로 간주합니다. 자세한 내용은 Amazon Glacier와 함께 AWS SDK for .NET 사용 단원을 참조하십시오.
3. 요청 객체를 파라미터로 입력하여 `UploadArchive` 메서드를 실행합니다.

   Amazon Glacier가 응답으로 새롭게 업로드된 아카이브 ID를 반환합니다.

예:의 하위 수준 API를 사용하여 단일 작업으로 아카이브 업로드 AWS SDK for .NET

다음 C# 코드 예제에서는 이전 단계를 설명합니다. 이 예제에서는 AWS SDK for .NET 를 사용하여 아카이브를 볼트()에 업로드합니다`examplevault`.

> **Note**
>
> 아카이브를 단일 작업으로 업로드하는 기본 REST API에 대한 자세한 내용은 아카이브 업로드 (POST archive) 섹션을 참조하세요.

이 예제의 실행 방법에 대한 단계별 지침은 코드 예제 실행 섹션을 참조하세요. 아래와 같이 업로드할 파일 이름을 사용해 코드를 업데이트해야 합니다.

Example

```
using System;
using System.IO;
using Amazon.Glacier;
using Amazon.Glacier.Model;
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
```

```
 class ArchiveUploadSingleOpLowLevel
 {
   static string vaultName       = "examplevault";
   static string archiveToUpload = "*** Provide file name (with full path) to upload
***";

   public static void Main(string[] args)
   {
     AmazonGlacierClient client;
     try
     {
        using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2))
       {
         Console.WriteLine("Uploading an archive.");
         string archiveId = UploadAnArchive(client);
         Console.WriteLine("Archive ID: {0}", archiveId);
       }
     }
     catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
     catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
     catch (Exception e) { Console.WriteLine(e.Message); }
     Console.WriteLine("To continue, press Enter");
     Console.ReadKey();
   }

   static string UploadAnArchive(AmazonGlacierClient client)
   {
     using (FileStream fileStream = new FileStream(archiveToUpload, FileMode.Open,
FileAccess.Read))
     {
       string treeHash = TreeHashGenerator.CalculateTreeHash(fileStream);
       UploadArchiveRequest request = new UploadArchiveRequest()
       {
         VaultName = vaultName,
         Body = fileStream,
         Checksum = treeHash
       };
       UploadArchiveResponse response = client.UploadArchive(request);
       string archiveID = response.ArchiveId;
       return archiveID;
     }
   }
 }
```

```
}
```

REST API를 사용하여 아카이브의 단일 작업 업로드

아카이브 업로드 API 호출을 통해 아카이브를 단일 작업으로 업로드할 수 있습니다. 자세한 내용은 아카이브 업로드(POST archive) 섹션을 참조하세요.

대용량 아카이브를 여러 파트로 나누어 업로드(멀티파트 업로드)

주제

- 멀티파트 업로드 프로세스
- 주요 사양
- 를 사용하여 대용량 아카이브 업로드 AWS CLI
- Amazon SDK for Java를 사용하여 대용량 아카이브를 파트로 나누어 업로드
- 를 사용하여 대용량 아카이브 업로드 AWS SDK for .NET
- REST API를 사용하는 대용량 아카이브의 멀티파트 업로드

멀티파트 업로드 프로세스

Amazon Glacier에 아카이브 업로드에서 설명한 바와 같이 Amazon Glacier(Amazon Glacier) 고객이 100메비바이트(MiB)가 넘는 대용량 아카이브를 업로드할 때는 멀티파트 업로드 기능을 사용하는 것을 권장합니다.

1. 멀티파트 업로드 시작

   멀티파트 업로드 시작 요청을 전송하면 Amazon Glacier가 멀티파트 업로드를 위한 고유 식별자인 업로드 멀티파트 업로드 ID를 반환합니다. 이후로 멀티파트 업로드 작업을 이어가려면 이 ID가 필요합니다. 이 ID는 Amazon Glacier가 작업을 완료한 후 최소 24시간 동안 만료되지 않습니다.

   멀티파트 업로드를 시작하는 요청에서 파트 크기(바이트 수)를 지정합니다. 마지막 파트를 제외하고 업로드하는 파트는 각각 여기에서 지정하는 크기를 따라야 합니다.

   > **Note**
   >
   > 멀티파트 업로드를 사용할 때 전체 아카이브 크기를 알 필요는 없습니다. 즉, 아카이브 업로드를 시작할 때 아카이브 크기를 모르더라도 멀티파트 업로드를 사용할 수 있습니다. 멀티파트 업로드를 시작할 때 파트 크기만 결정하면 됩니다.

멀티파트 업로드 요청을 시작할 때 선택적으로 아카이브 설명을 입력할 수도 있습니다.

2. 파트 업로드

   각 파트의 업로드 요청마다 1단계에서 얻은 멀티파트 업로드 ID를 추가해야 합니다. 요청에서 최종 아카이브의 파트 위치를 식별할 수 있도록 내용 범위를 바이트로 지정해야 합니다. Amazon Glacier는 추후에 콘텐츠 범위 정보를 사용하여 아카이브를 적절한 순서로 조립합니다. 업로드하는 파트마다 내용 범위를 입력하면 최종 아카이브 어셈블리에서 파트 위치를 결정하기 때문에 어떤 순서로든 파트를 업로드할 수 있습니다. 또한 병렬 방식으로 파트를 업로드하는 것도 가능합니다. 이전에 업로드한 파트와 동일한 내용 범위로 새로운 파트를 업로드할 경우에는 이전에 업로드한 파트를 덮어쓰게 됩니다.

3. 멀티파트 업로드 완료(또는 중단)

   아카이브 파트를 모두 업로드한 후에는 완료 작업을 시작합니다. 반드시 요청에 업로드 ID를 지정해야 합니다. Amazon Glacier는 사용자가 제공한 콘텐츠 범위를 바탕으로 오름차순으로 각 부분을 연결하여 아카이브를 생성합니다. 멀티파트 업로드 완료 요청에 대한 Amazon Glacier 응답에는 새로 생성된 아카이브의 아카이브 ID가 포함됩니다. 멀티파트 업로드 시작 요청에서 선택적으로 아카이브 설명을 입력하였다면 Amazon Glacier가 어셈블링된 아카이브에 설명을 연결합니다. 멀티파트 업로드가 성공적으로 완료된 후에는 멀티파트 업로드 ID를 참조할 수 없습니다. 이 말은 멀티파트 업로드 ID로 연결된 파트에 액세스할 수 없다는 것을 의미합니다.

   멀티파트 업로드를 중단한 경우 해당 멀티파트 업로드 ID로는 더 이상 파트를 업로드할 수 없습니다. 중단된 멀티파트 업로드로 연결된 파트가 차지하는 스토리지는 모두 비워집니다. 파트 업로드가 진행 중일 때 진행 중인 파트 업로드는 성공적으로 완료되거나, 혹은 오류로 멈출 수도 있습니다.

추가 멀티파트 업로드 작업

Amazon Glacier(Amazon Glacier)는 다음과 같은 멀티파트 업로드 API 직접 호출을 추가적으로 제공합니다.

- 파트 나열: 이 작업을 사용하면 특정 멀티파트 업로드의 파트를 나열할 수 있습니다. 그러면 멀티파트 업로드로 업로드한 파트의 정보가 반환됩니다. 파트 나열 요청이 있을 때마다 Amazon Glacier는 최대 1,000개까지 파트 정보를 반환합니다. 멀티파트 업로드에 대해 목록을 조회할 파트가 더 있는 경우에는 결과에 페이지 번호가 매겨지고, 응답으로 목록을 계속 이어가는 마커가 반환됩니다. 이후 파트를 가져오려면 추가 요청을 보내야 합니다. 반환된 부분 목록에는 업로드가 완료되지 않은 부분은 포함되지 않습니다.

- 멀티파트 업로드 나열: 이 작업을 사용하여 진행 중인 멀티파트 업로드의 목록을 확인할 수 있습니다. 진행 중인 멀티파트 업로드는 시작했지만 아직 완료 또는 중단하지 않은 업로드입니다. 멀티파트 업로드 목록 조회 요청이 있을 때마다 Amazon Glacier는 최대 1,000개까지 멀티파트 업로드를 반환합니다. 목록을 조회할 멀티파트 업로드가 더 있는 경우에는 결과에 페이지 번호가 매겨지고, 응답으로 목록을 계속 이어가는 마커가 반환됩니다. 나머지 멀티파트 업로드를 가져오려면 추가 요청을 보내야 합니다.

주요 사양

다음 표에 멀티파트 업로드의 주요 사양이 나와 있습니다.

항목	사양
최대 아카이브 크기	10,000x4기비바이트(GiB)
업로드당 최대 파트 개수	10,000
파트 크기	1MiB에서 4GiB까지, 마지막 파트는 1MiB 미만일 수도 있습니다. 크기 값은 바이트 단위로 지정합니다. 파트 크기는 2의 제곱이 곱해진 메비바이트(1024키비바이트[KiB]) 값이어야 합니다. 예를 들어 1048576(1MiB), 2097152(2MiB), 4194304(4MiB), 8388608(8MiB)등과 같습니다.
파트 목록 조회 요청에 대해 반환되는 최대 파트 개수	1,000
멀티파트 업로드 나열 요청에서 반환되는 최대 멀티파트 업로드 개수	1,000

를 사용하여 대용량 아카이브 업로드 AWS CLI

AWS Command Line Interface ()를 사용하여 Amazon Glacier(Amazon Glacier)에 아카이브를 업로드할 수 있습니다AWS CLI. 대규모 아카이브의 업로드 환경을 개선하기 위해, Amazon Glacier는 멀티파트 업로드를 지원하는 여러 API 작업을 제공합니다. 이 API 작업을 사용하여 아카이브를 파트로 나누어 업로드할 수 있습니다. 이러한 파트들은 순서에 상관없이 각각 병렬 방식으로 업로드됩니다. 멀티파

트 업로드에 실패하더라도 전체 아카이브가 아니라 실패한 파트만 다시 업로드하면 됩니다. 멀티파트 업로드는 크기가 1바이트에서 약 40,000기비바이트(GiB)에 이르는 아카이브에 사용할 수 있습니다.

Amazon Glacier 멀티파트 업로드에 대한 자세한 내용은 대용량 아카이브를 여러 파트로 나누어 업로드(멀티파트 업로드) 섹션을 참조하세요.

주제

- (사전 조건) 설정 AWS CLI
- (요구 사항) Python 설치
- (요구 사항) Amazon Glacier 볼트 생성
- 예:를 사용하여 대용량 아카이브를 파트로 업로드 AWS CLI

(사전 조건) 설정 AWS CLI

1. AWS CLI를 다운로드하고 구성합니다. 관련 지침은 AWS Command Line Interface 사용 설명서에서 다음 토픽을 참조하세요.

   설치 AWS Command Line Interface

   구성 AWS Command Line Interface
2. 명령 프롬프트에 다음 명령을 입력하여 AWS CLI 설정을 확인합니다. 이러한 명령은 명시적으로 자격 증명을 제공하지 않으므로 기본 프로파일의 자격 증명이 사용됩니다.

   - help 명령을 사용해 보십시오.

     ```
     aws help
     ```

   - 구성된 계정의 Amazon Glacier 볼트 목록을 가져오려면 `list-vaults` 명령을 사용합니다. *123456789012*을 AWS 계정 ID로 바꿉니다.

     ```
     aws glacier list-vaults --account-id 123456789012
     ```

   - 에 대한 현재 구성 데이터를 보려면 `aws configure list` 명령을 AWS CLI사용합니다.

     ```
     aws configure list
     ```

(요구 사항) Python 설치

멀티파트 업로드를 완료하려면 업로드 중인 아카이브의 SHA256 트리 해시를 반드시 계산해야 합니다. 이 작업은 업로드를 원하는 파일의 SHA256 트리 해시를 계산하는 것과 다릅니다. 업로드 중인 아카이브의 SHA256 트리 해시를 계산할 때 Java, C#(.NET와 함께) 또는 Python을 사용할 수 있습니다. 이 예시에서는 Python을 사용합니다. Java 또는 C#을 사용하는 방법 지침은 체크섬 계산 섹션을 참조하세요.

Python을 설치하는 방법에 대한 자세한 내용은 Boto3 개발자 가이드의 Python 설치 또는 업데이트를 참조하세요.

(요구 사항) Amazon Glacier 볼트 생성

다음 예시를 사용하려면 적어도 하나 이상의 Amazon Glacier 볼트가 생성되어 있어야 합니다. 볼트 생성에 대한 자세한 내용은 Amazon Glacier에 볼트 생성 섹션을 참조하세요.

예:를 사용하여 대용량 아카이브를 파트로 업로드 AWS CLI

이 예시에서는 파일을 생성하고 멀티파트 업로드 API 작업을 사용하여 이 파일을 여러 파트로 나누어 Amazon Glacier에 업로드합니다.

> **Important**
>
> 이 과정을 시작하기 전에 요구 사항 단계를 모두 수행했는지 확인합니다. 아카이브를 업로드하려면 반드시 볼트가 생성되어 있고 AWS CLI 가 구성되어 있어야 하며 Java, C#, 또는 Python을 사용하여 SHA256 트리 해시를 계산할 준비가 되어있어야 합니다.

다음 절차에서는 `initiate-multipart-upload`, `upload-multipart-part`및 `complete-multipart-upload` AWS CLI 명령을 사용합니다.

이 명령에 대한 자세한 내용은 AWS CLI 명령 참조의 initiate-multipart-upload, upload-multipart-part, complete-multipart-upload 섹션을 참조하세요.

1. initiate-multipart-upload 명령을 사용하여 멀티파트 업로드 리소스를 생성합니다. 요청에서 파트 크기를 바이트 수로 지정합니다. 마지막 파트를 제외하고 사용자가 업로드한 파트는 여기에서 지정하는 크기를 따라야 합니다. 멀티파트 업로드를 시작할 때 전체 아카이브 크기를 알 필요는 없습니다. 하지만 최종 단계에서 업로드를 완료할 때는 각 파트의 총 크기가 바이트로 필요합니다.

다음 명령에서 `--vault-name` 및 `--account-ID` 파라미터의 값을 사용자의 정보로 바꿉니다. 이 명령은 파트 크기가 파일당 1메비바이트(MiB)(1024 x 1024바이트)인 아카이브를 업로드하도록 지정합니다. 필요한 경우 이 `--part-size` 파라미터 값을 바꾸세요.

```
aws glacier initiate-multipart-upload --vault-name awsexamplevault --part-
size 1048576 --account-id 123456789012
```

예상 결과:

```
{
"location": "/123456789012/vaults/awsexamplevault/multipart-uploads/uploadId",
"uploadId": "uploadId"
}
```

완료되면 이 명령은 멀티파트 업로드 리소스의 업로드 ID와 Amazon Glacier 내의 위치를 출력합니다. 이후 단계에서 이 업로드 ID를 사용합니다.

2. 이 예시에서는 다음 명령을 사용하여 4.4MiB 파일을 만들고 1MiB 청크로 분할한 다음 각 청크를 업로드할 수 있습니다. 자체 파일을 업로드하려면 데이터를 청크로 분할하고 각 파트를 업로드하는 비슷한 절차를 따를 수 있습니다.

   Linux 또는 macOS

   다음 명령은 Linux 또는 macOS에서 `file_to_upload`로 이름이 지정된 4.4MiB 파일을 만듭니다.

   ```
   mkfile -n 9000b file_to_upload
   ```

   Windows

   다음 명령은 Windows에서 `file_to_upload`로 이름이 지정된 4.4MiB 파일을 만듭니다.

   ```
   fsutil file createnew file_to_upload 4608000
   ```

3. 다음으로 이 파일을 1MiB 청크로 분할합니다.

   ```
   split -b 1048576 file_to_upload chunk
   ```

이제 다음과 같은 다섯 개의 청크가 있습니다. 처음 네 개는 1MiB이고 마지막 한 개는 약 400키비바이트(KiB)입니다.

```
chunkaa
chunkab
chunkac
chunkad
chunkae
```

4. upload-multipart-part 명령을 사용하여 아카이브의 파트를 업로드합니다. 아카이브를 파트를 어떤 순서로든 업로드할 수 있습니다. 또한 병렬 방식으로 파트를 업로드할 수도 있습니다. 멀티파트 업로드에서 업로드할 수 있는 파트 수는 최대 10,000개입니다.

   다음 명령에서 `--vault-name`, `--account-ID`, 및 `--upload-id` 파라미터 값을 바꿉니다. 업로드 ID는 `initiate-multipart-upload` 명령 출력으로 제공된 ID와 일치해야 합니다. `--range` 파라미터는 크기가 1MiB(1024 x 1024바이트)인 파트를 업로드하도록 지정합니다. 이 크기는 `initiate-multipart-upload` 명령에서 지정한 크기와 일치해야 합니다. 필요한 경우 크기 값을 조정하세요. `--body` 파라미터는 업로드하는 파트의 이름을 지정합니다.

   ```
   aws glacier upload-multipart-part --body chunkaa --range='bytes 0-1048575/*' --
   vault-name awsexamplevault --account-id 123456789012 --upload-id upload_ID
   ```

   성공하면 명령에서 업로드된 파트의 체크섬이 포함된 출력이 생성됩니다.

5. `upload-multipart-part` 명령을 다시 실행하여 멀티파트 업로드의 나머지 파트를 업로드합니다. 업로드하는 부분과 일치하도록 각 명령의 `--range` 및 `--body` 매개변수 값을 업데이트합니다.

   ```
   aws glacier upload-multipart-part --body chunkab --range='bytes 1048576-2097151/*'
    --vault-name awsexamplevault --account-id 123456789012 --upload-id upload_ID
   ```

   ```
   aws glacier upload-multipart-part --body chunkac --range='bytes 2097152-3145727/*'
    --vault-name awsexamplevault --account-id 123456789012 --upload-id upload_ID
   ```

   ```
   aws glacier upload-multipart-part --body chunkad --range='bytes 3145728-4194303/*'
    --vault-name awsexamplevault --account-id 123456789012 --upload-id upload_ID
   ```

```
aws glacier upload-multipart-part --body chunkae --range='bytes 4194304-4607999/*'
 --vault-name awsexamplevault --account-id 123456789012 --upload-id upload_ID
```

> **Note**
>
> 업로드의 마지막 부분이 1MiB 미만이므로 최종 명령의 `--range` 파라미터 값은 그보다 더 작습니다. 성공하면 각 명령에서 업로드된 각 부분에 대한 체크섬이 포함된 출력을 생성합니다.

6. 다음으로 아카이브를 조합하고 업로드를 완료합니다. 아카이브의 전체 크기와 SHA256 트리 해시를 반드시 포함해야 합니다.

   아카이브의 SHA256 트리 해시를 계산하기 위해서 Java, C# 또는 Python을 사용할 수 있습니다. 이 예시에서는 Python을 사용합니다. Java 또는 C#을 사용하는 방법 지침은 체크섬 계산 섹션을 참조하세요.

   Python 파일 `checksum.py`를 생성하고 다음 코드를 추가합니다. 필요한 경우 원본 파일의 이름을 바꾸세요.

   ```
   from botocore.utils import calculate_tree_hash

   checksum = calculate_tree_hash(open('file_to_upload', 'rb'))
   print(checksum)
   ```

7. `checksum.py`를 실행하여 SHA256 트리 해시를 계산합니다. 다음 해시는 출력과 일치하지 않을 수 있습니다.

   ```
   $ python3 checksum.py
   $ 3d760edb291bfc9d90d35809243de092aea4c47b308290ad12d084f69988ae0c
   ```

8. complete-multipart-upload 명령을 사용하여 아카이브 업로드를 완료합니다. `--vault-name`, `--account-ID`, `--upload-ID`, 및 `--checksum` 파라미터의 값을 바꿉니다. `--archive` 파라미터 값은 아카이브의 전체 크기를 바이트로 지정합니다. 이 값은 반드시 업로드한 개별 파트 크기의 총합이 되어야 합니다. 필요한 경우 이 값을 바꾸세요.

   ```
   aws glacier complete-multipart-upload --archive-size 4608000 --vault-
   name awsexamplevault --account-id 123456789012 --upload-id upload_ID --
   checksum checksum
   ```

완료되면 명령이 아카이브의 ID, 체크섬 및 Amazon Glacier 내의 위치를 출력합니다.

Amazon SDK for Java를 사용하여 대용량 아카이브를 파트로 나누어 업로드

Amazon SDK for Java에서 제공하는 하이레벨 및 로우레벨 API 둘 모두는 대규모 아카이브를 업로드하는 방법을 제공합니다(Amazon Glacier에 아카이브 업로드 섹션 참조).

- 하이레벨 API는 모든 크기의 아카이브를 업로드하는 데 사용할 수 있는 방법을 제공합니다. 이 방법은 업로드하는 파일에 따라 아카이브를 단일 작업으로 업로드하거나, 혹은 Amazon Glacier(Amazon Glacier)에서 지원되는 멀티파트 업로드 기능을 사용해 아카이브를 여러 파트로 나누어 업로드합니다.
- 로우레벨 API는 기본 REST 구현에 가깝게 매핑합니다. 따라서 더욱 작은 크기의 아카이브를 단일 작업으로 업로드하는 방법을 제공하거나, 혹은 대용량 아카이브의 경우 멀티파트 업로드를 지원하는 방법 그룹을 제공합니다. 이 섹션에서는 로우레벨 API를 사용하는 대용량 아카이브의 멀티파트 업로드에 대해서 설명합니다.

하이레벨 및 로우레벨 API에 대한 자세한 내용은 Amazon Glacier AWS SDK for Java 에서 사용 섹션을 참조하세요.

주제

- 의 상위 수준 API를 사용하여 대용량 아카이브를 파트로 업로드 AWS SDK for Java
- 의 하위 수준 API를 사용하여 대용량 아카이브를 파트로 업로드 AWS SDK for Java

의 상위 수준 API를 사용하여 대용량 아카이브를 파트로 업로드 AWS SDK for Java

작든 크든 상관없이 하이레벨 API를 사용하여 아카이브를 업로드하는 방법은 동일합니다. 다만 하이레벨 API 방법은 아카이브 크기에 따라 아카이브를 단일 작업으로 업로드할지 아니면 Amazon Glacier에서 제공하는 멀티파트 업로드 API를 사용할지 결정합니다. 자세한 내용은 의 상위 수준 API를 사용하여 아카이브 업로드 AWS SDK for Java 단원을 참조하십시오.

의 하위 수준 API를 사용하여 대용량 아카이브를 파트로 업로드 AWS SDK for Java

업로드를 세분화하여 제어할 때는 로우레벨 API를 사용하여 요청을 구성하고 응답을 처리할 수 있습니다. 다음은 AWS SDK for Java을 사용하여 대용량 아카이브를 여러 파트로 나누어 업로드하는 단계입니다.

1. `AmazonGlacierClient` 클래스(클라이언트)의 인스턴스를 만듭니다.

   아카이브를 저장할 AWS 리전을 지정해야 합니다. 이 클라이언트를 사용하여 수행하는 모든 작업은 해당 AWS 리전에 적용됩니다.
2. `initiateMultipartUpload` 메서드를 호출하여 멀티파트 업로드를 시작합니다.

   아카이브를 업로드할 볼트 이름과 아카이브를 여러 파트로 나누어 업로드하는 데 사용할 파트 크기, 그리고 설명(옵션)을 입력해야 합니다. `InitiateMultipartUploadRequest` 클래스의 인스턴스를 만들어 이 정보를 제공합니다. 그러면 Amazon Glacier가 응답으로 업로드 ID를 반환합니다.
3. `uploadMultipartPart` 메서드를 호출하여 각 파트를 업로드합니다.

   업로드하는 파트마다 볼트 이름, 현재 파트에서 업로드할 최종 아카이브의 바이트 범위, 파트 데이터의 체크섬, 그리고 업로드 ID를 입력해야 합니다.
4. `completeMultipartUpload` 메서드를 호출하여 멀티파트 업로드를 마칩니다.

   이때는 업로드 ID, 전체 아카이브의 체크섬, 아카이브 크기(업로드한 모든 파트의 총 크기), 그리고 볼트 이름을 입력해야 합니다. Amazon Glacier는 업로드된 파트부터 아카이브를 구성하여 아카이브 ID를 반환합니다.

예:를 사용하여 대용량 아카이브를 파트에 업로드 AWS SDK for Java

다음 Java 코드 예제에서는 AWS SDK for Java 를 사용하여 아카이브를 볼트()에 업로드합니다`examplevault`. 이 예제의 실행 방법에 대한 단계별 지침은 Eclipse를 사용하여 Amazon Glacier의 Java 예 실행 섹션을 참조하세요. 아래와 같이 업로드할 파일 이름을 사용해 코드를 업데이트해야 합니다.

> **Note**
>
> 아래 예제는 파트 크기가 1MB~1GB일 때 유효합니다. 하지만 Amazon Glacier는 최대 4GB 크기의 파트를 지원합니다.

Example

```
import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
```

```
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.Date;
import java.util.LinkedList;
import java.util.List;

import com.amazonaws.AmazonClientException;
import com.amazonaws.AmazonServiceException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.TreeHashGenerator;
import com.amazonaws.services.glacier.model.CompleteMultipartUploadRequest;
import com.amazonaws.services.glacier.model.CompleteMultipartUploadResult;
import com.amazonaws.services.glacier.model.InitiateMultipartUploadRequest;
import com.amazonaws.services.glacier.model.InitiateMultipartUploadResult;
import com.amazonaws.services.glacier.model.UploadMultipartPartRequest;
import com.amazonaws.services.glacier.model.UploadMultipartPartResult;
import com.amazonaws.util.BinaryUtils;

public class ArchiveMPU {

    public static String vaultName = "examplevault";
    // This example works for part sizes up to 1 GB.
    public static String partSize = "1048576"; // 1 MB.
    public static String archiveFilePath = "*** provide archive file path ***";
    public static AmazonGlacierClient client;

    public static void main(String[] args) throws IOException {

     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        client = new AmazonGlacierClient(credentials);
        client.setEndpoint("https://glacier.us-west-2.amazonaws.com/");

        try {
            System.out.println("Uploading an archive.");
            String uploadId = initiateMultipartUpload();
            String checksum = uploadParts(uploadId);
            String archiveId = CompleteMultiPartUpload(uploadId, checksum);
            System.out.println("Completed an archive. ArchiveId: " + archiveId);

        } catch (Exception e) {
            System.err.println(e);
        }
```

```
    }

    private static String initiateMultipartUpload() {
        // Initiate
        InitiateMultipartUploadRequest request = new InitiateMultipartUploadRequest()
            .withVaultName(vaultName)
            .withArchiveDescription("my archive " + (new Date()))
            .withPartSize(partSize);

        InitiateMultipartUploadResult result = client.initiateMultipartUpload(request);

        System.out.println("ArchiveID: " + result.getUploadId());
        return result.getUploadId();
    }

    private static String uploadParts(String uploadId) throws AmazonServiceException,
NoSuchAlgorithmException, AmazonClientException, IOException {

        int filePosition = 0;
        long currentPosition = 0;
        byte[] buffer = new byte[Integer.valueOf(partSize)];
        List<byte[]> binaryChecksums = new LinkedList<byte[]>();

        File file = new File(archiveFilePath);
        FileInputStream fileToUpload = new FileInputStream(file);
        String contentRange;
        int read = 0;
        while (currentPosition < file.length())
        {
            read = fileToUpload.read(buffer, filePosition, buffer.length);
            if (read == -1) { break; }
            byte[] bytesRead = Arrays.copyOf(buffer, read);

            contentRange = String.format("bytes %s-%s/*", currentPosition,
currentPosition + read - 1);
            String checksum = TreeHashGenerator.calculateTreeHash(new
ByteArrayInputStream(bytesRead));
            byte[] binaryChecksum = BinaryUtils.fromHex(checksum);
            binaryChecksums.add(binaryChecksum);
            System.out.println(contentRange);

            //Upload part.
            UploadMultipartPartRequest partRequest = new UploadMultipartPartRequest()
```

```
            .withVaultName(vaultName)
            .withBody(new ByteArrayInputStream(bytesRead))
            .withChecksum(checksum)
            .withRange(contentRange)
            .withUploadId(uploadId);

            UploadMultipartPartResult partResult =
 client.uploadMultipartPart(partRequest);
            System.out.println("Part uploaded, checksum: " + partResult.getChecksum());

            currentPosition = currentPosition + read;
        }
        fileToUpload.close();
        String checksum = TreeHashGenerator.calculateTreeHash(binaryChecksums);
        return checksum;
    }

    private static String CompleteMultiPartUpload(String uploadId, String checksum)
 throws NoSuchAlgorithmException, IOException {

        File file = new File(archiveFilePath);

        CompleteMultipartUploadRequest compRequest = new
 CompleteMultipartUploadRequest()
            .withVaultName(vaultName)
            .withUploadId(uploadId)
            .withChecksum(checksum)
            .withArchiveSize(String.valueOf(file.length()));

        CompleteMultipartUploadResult compResult =
 client.completeMultipartUpload(compRequest);
        return compResult.getLocation();
    }
}
```

를 사용하여 대용량 아카이브 업로드 AWS SDK for .NET

.NET용 Amazon SDK에서 제공하는 하이레벨 및 로우레벨 API 둘 모두는 대규모 아카이브를 분할하여 업로드하는 방법을 제공합니다(Amazon Glacier에 아카이브 업로드 섹션 참조).

- 하이레벨 API는 모든 크기의 아카이브를 업로드하는 데 사용할 수 있는 방법을 제공합니다. 이 방법은 업로드하는 파일에 따라 아카이브를 단일 작업으로 업로드하거나, 혹은 Amazon Glacier(Amazon

Glacier)에서 지원되는 멀티파트 업로드 기능을 사용해 아카이브를 여러 파트로 나누어 업로드합니다.

- 로우레벨 API는 기본 REST 구현에 가깝게 매핑합니다. 따라서 더욱 작은 크기의 아카이브를 단일 작업으로 업로드하는 방법을 제공하거나, 혹은 대용량 아카이브의 경우 멀티파트 업로드를 지원하는 방법 그룹을 제공합니다. 이 섹션에서는 로우레벨 API를 사용하는 대용량 아카이브의 멀티파트 업로드에 대해서 설명합니다.

하이레벨 및 로우레벨 API에 대한 자세한 내용은 Amazon Glacier와 함께 AWS SDK for .NET 사용 섹션을 참조하세요.

주제

- 의 상위 수준 API를 사용하여 대용량 아카이브를 파트로 업로드 AWS SDK for .NET
- 의 하위 수준 API를 사용하여 대용량 아카이브를 파트로 업로드 AWS SDK for .NET

의 상위 수준 API를 사용하여 대용량 아카이브를 파트로 업로드 AWS SDK for .NET

작든 크든 상관없이 하이레벨 API를 사용하여 아카이브를 업로드하는 방법은 동일합니다. 다만 하이레벨 API 방법은 아카이브 크기에 따라 아카이브를 단일 작업으로 업로드할지 아니면 Amazon Glacier에서 제공하는 멀티파트 업로드 API를 사용할지 결정합니다. 자세한 내용은 의 상위 수준 API를 사용하여 아카이브 업로드 AWS SDK for .NET 단원을 참조하십시오.

의 하위 수준 API를 사용하여 대용량 아카이브를 파트로 업로드 AWS SDK for .NET

업로드를 세분화하여 제어할 때는 로우레벨 API를 사용하여 요청을 구성하고 응답을 처리할 수 있습니다. 다음은 AWS SDK for .NET을 사용하여 대용량 아카이브를 여러 파트로 나누어 업로드하는 단계입니다.

1. `AmazonGlacierClient` 클래스(클라이언트)의 인스턴스를 만듭니다.

   아카이브를 저장할 AWS 리전을 지정해야 합니다. 이 클라이언트를 사용하여 수행하는 모든 작업은 해당 AWS 리전에 적용됩니다.
2. `InitiateMultipartUpload` 메서드를 호출하여 멀티파트 업로드를 시작합니다.

   아카이브를 업로드할 볼트 이름과 아카이브를 여러 파트로 나누어 업로드하는 데 사용할 파트 크기, 그리고 설명(옵션)을 입력해야 합니다. `InitiateMultipartUploadRequest` 클래스의 인스턴스를 만들어 이 정보를 제공합니다. 그러면 Amazon Glacier가 응답으로 업로드 ID를 반환합니다.
3. `UploadMultipartPart` 메서드를 호출하여 각 파트를 업로드합니다.

업로드하는 파트마다 볼트 이름, 현재 파트에서 업로드할 최종 아카이브의 바이트 범위, 파트 데이터의 체크섬, 그리고 업로드 ID를 입력해야 합니다.

4. `CompleteMultipartUpload` 메서드를 호출하여 멀티파트 업로드를 마칩니다.

   업로드 ID, 전체 아카이브의 체크섬, 아카이브 크기(업로드한 모든 파트의 총 크기), 그리고 볼트 이름을 제공해야 합니다. Amazon Glacier는 업로드된 파트부터 아카이브를 구성하여 아카이브 ID를 반환합니다.

예: .NET용 Amazon SDK를 사용하여 대용량 아카이브를 파트로 나누어 업로드

다음 C# 코드 예제에서는 AWS SDK for .NET 를 사용하여 아카이브를 볼트()에 업로드합니다`examplevault`. 이 예제의 실행 방법에 대한 단계별 지침은 코드 예제 실행 섹션을 참조하세요. 아래와 같이 업로드할 파일 이름을 사용해 코드를 업데이트해야 합니다.

Example

```
using System;
using System.Collections.Generic;
using System.IO;
using Amazon.Glacier;
using Amazon.Glacier.Model;
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
  class ArchiveUploadMPU
  {
    static string vaultName       = "examplevault";
    static string archiveToUpload = "*** Provide file name (with full path) to upload
 ***";
    static long partSize          = 4194304; // 4 MB.

    public static void Main(string[] args)
    {
      AmazonGlacierClient client;
      List<string> partChecksumList = new List<string>();
      try
      {
         using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2))
        {
          Console.WriteLine("Uploading an archive.");
```

```
          string uploadId  = InitiateMultipartUpload(client);
          partChecksumList = UploadParts(uploadId, client);
          string archiveId = CompleteMPU(uploadId, client, partChecksumList);
          Console.WriteLine("Archive ID: {0}", archiveId);
        }
        Console.WriteLine("Operations successful. To continue, press Enter");
        Console.ReadKey();
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }

    static string InitiateMultipartUpload(AmazonGlacierClient client)
    {
      InitiateMultipartUploadRequest initiateMPUrequest = new
InitiateMultipartUploadRequest()
      {

        VaultName = vaultName,
        PartSize = partSize,
        ArchiveDescription = "Test doc uploaded using MPU."
      };

      InitiateMultipartUploadResponse initiateMPUresponse =
client.InitiateMultipartUpload(initiateMPUrequest);

      return initiateMPUresponse.UploadId;
    }

    static List<string> UploadParts(string uploadID, AmazonGlacierClient client)
    {
      List<string> partChecksumList = new List<string>();
      long currentPosition = 0;
      var buffer = new byte[Convert.ToInt32(partSize)];

      long fileLength = new FileInfo(archiveToUpload).Length;
      using (FileStream fileToUpload = new FileStream(archiveToUpload, FileMode.Open,
FileAccess.Read))
      {
        while (fileToUpload.Position < fileLength)
        {
```

```
            Stream uploadPartStream = GlacierUtils.CreatePartStream(fileToUpload,
 partSize);
            string checksum = TreeHashGenerator.CalculateTreeHash(uploadPartStream);
            partChecksumList.Add(checksum);
            // Upload part.
            UploadMultipartPartRequest uploadMPUrequest = new
 UploadMultipartPartRequest()
            {

              VaultName = vaultName,
              Body = uploadPartStream,
              Checksum = checksum,
              UploadId = uploadID
            };
            uploadMPUrequest.SetRange(currentPosition, currentPosition +
 uploadPartStream.Length - 1);
            client.UploadMultipartPart(uploadMPUrequest);

            currentPosition = currentPosition + uploadPartStream.Length;
          }
        }
        return partChecksumList;
      }

      static string CompleteMPU(string uploadID, AmazonGlacierClient client, List<string>
 partChecksumList)
      {
        long fileLength = new FileInfo(archiveToUpload).Length;
        CompleteMultipartUploadRequest completeMPUrequest = new
 CompleteMultipartUploadRequest()
        {
          UploadId = uploadID,
          ArchiveSize = fileLength.ToString(),
          Checksum = TreeHashGenerator.CalculateTreeHash(partChecksumList),
          VaultName = vaultName
        };

        CompleteMultipartUploadResponse completeMPUresponse =
 client.CompleteMultipartUpload(completeMPUrequest);
        return completeMPUresponse.ArchiveId;
      }
    }
}
```

REST API를 사용하는 대용량 아카이브의 멀티파트 업로드

대용량 아카이브를 여러 파트로 나누어 업로드(멀티파트 업로드)에서 설명한 것과 같이 멀티파트 업로드는 아카이브를 여러 파트로 나누어 업로드하는 작업을 말하며 관련 작업을 수행합니다. 이러한 작업들에 대한 자세한 내용은 API 참조 항목을 참조하세요.

- 멀티파트 업로드 시작(POST multipart-uploads)
- 파트 업로드(PUT uploadID)
- 멀티파트 업로드 완료(POST uploadID)
- 멀티파트 업로드 중단(DELETE uploadID)
- 파트 목록 조회(GET uploadID)
- 멀티파트 업로드 목록 조회(GET multipart-uploads)

Amazon Glacier에서 아카이브 다운로드

Amazon Glacier는 볼트 생성 및 삭제에 사용할 수 있는 관리 콘솔을 제공합니다. 하지만 관리 콘솔을 이용하여 Amazon Glacier에서 아카이브를 다운로드할 수는 없습니다. 사진, 비디오 및 기타 문서와 같은 데이터를 다운로드하려면 REST API를 직접 사용하거나 AWS SDKs를 사용하여 AWS Command Line Interface (AWS CLI)를 사용하거나 코드를 작성하여 요청해야 합니다.

에서 Amazon Glacier를 사용하는 방법에 대한 자세한 내용은 Amazon Glacier 참조를 AWS CLI참조하세요. AWS CLI Amazon Glacier 를 설치하려면 단원을 AWS CLI참조하십시오AWS Command Line Interface. 다음 주제에서는 AWS SDK for Java AWS SDK for .NET, 및 Amazon Glacier REST API를 사용하여 Amazon Glacier에 아카이브를 다운로드하는 방법을 설명합니다.

주제

- Amazon Glacier 아카이브 가져오기
- 를 사용하여 Amazon Glacier에서 아카이브 다운로드 AWS SDK for Java
- 를 사용하여 Amazon Glacier에서 아카이브 다운로드 AWS SDK for .NET
- Python을 통한 병렬 처리를 사용하여 대용량 아카이브 다운로드
- REST API를 사용하여 아카이브 다운로드
- 를 사용하여 Amazon Glacier에서 아카이브 다운로드 AWS CLI

Amazon Glacier 아카이브 가져오기

Amazon Glacier에서 아카이브를 가져오는 것은 비동기식 작업이기 때문에 먼저 작업을 시작하고 작업이 완료된 후에 출력을 다운로드합니다. 아카이브 가져오기 작업을 시작하려면 작업 시작(POST jobs) REST API 작업 또는 AWS CLI, 또는 AWS SDKs의 해당 작업을 사용합니다.

주제

- 아카이브 검색 옵션
- 범위가 지정된 아카이브 가져오기

Amazon Glacier에서 아카이브를 가져오는 작업은 2단계 프로세스로 구성됩니다. 다음은 이 프로세스의 개요입니다.

아카이브를 가져오려면

1. 아카이브 가져오기 작업을 시작합니다.

   a. 원하는 아카이브의 ID를 가져옵니다. 아카이브 ID는 볼트 인벤토리에서 가져올 수 있습니다. REST API, AWS CLI또는 AWS SDKs. 자세한 내용은 Amazon Glacier에서 볼트 인벤토리 다운로드 단원을 참조하십시오.

   b. 작업 시작(POST jobs) 작업을 사용하여 Amazon Glacier에 후속 다운로드를 위해서 아카이브 전체 또는 일부분을 준비하도록 요청하는 작업을 시작합니다.

   작업을 시작하면 Amazon Glacier는 응답으로 작업 ID를 반환하고 비동기식으로 작업을 실행합니다. (2단계에서 설명된 것과 같이 작업이 끝날 때까지는 작업 출력을 다운로드할 수 없습니다.)

   > **Important**
   >
   > 표준 검색에 한해서 데이터 검색 정책은 `PolicyEnforcedException` 예외에 따라 `Initiate Job` 요청이 실패하는 원인이 될 수 있습니다. 데이터 가져오기 정책에 대한 자세한 내용은 Amazon Glacier 데이터 검색 정책 섹션을 참조하세요. `PolicyEnforcedException` 예외에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

   필요한 경우 Amazon Glacier에 저장된 대용량 데이터 세그먼트를 복원할 수 있습니다. Amazon Glacier 스토리지 클래스에서 데이터를 복원하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 객체 아카이빙을 위한 스토리지 클래스를 참조하세요.

2. 작업이 완료된 후 작업 출력 가져오기(GET output) 작업을 사용하여 바이트를 다운로드합니다.

   바이트 전체를 다운로드하거나, 혹은 바이트 범위를 지정하여 작업 출력의 일부만 다운로드할 수 있습니다. 출력 용량이 클수록 네트워크 장애 같은 다운로드 오류가 발생할 경우를 대비해 청크 단위로 다운로드하는 것이 효과적입니다. 단일 요청으로 작업을 출력하다가 네트워크 장애가 발생하면 출력을 처음부터 다시 다운로드해야 합니다. 하지만 청크 단위로 출력을 다운로드할 경우에는 장애가 발생하더라도 전체가 아닌 일부 출력만 다시 다운로드하면 됩니다.

Amazon Glacier에서는 출력을 다운로드하려면 작업을 먼저 마쳐야 합니다. 작업 완료 후 최소 24시간까지는 작업이 만료되지 않습니다. 이 말은 작업 완료 후 24시간까지는 출력을 다운로드할 수 있다는 것을 의미합니다. 복원은 작업 완료 후 24시간이 지나면 언제든지 만료될 수 있습니다. 작업 완료 여부를 알고 싶다면 다음 옵션 중 한 가지를 사용하여 상태를 확인합니다.

- 작업 완료 알림 대기: Amazon Glacier가 작업 완료 후에도 알림 메시지를 게시할 수 있도록 Amazon Simple Notification Service(Amazon SNS) 토픽을 지정합니다. Amazon Glacier는 작업이 완료된 후에만 알림을 보냅니다.

  Amazon SNS 토픽은 작업을 시작할 때 지정할 수 있습니다. 작업 요청 시 Amazon SNS 토픽을 지정하는 방법 외에 아카이브 가져오기 이벤트에 대한 알림이 볼트에 설정되어 있는 경우에도 Amazon Glacier가 알림 메시지를 해당하는 SNS 토픽에 게시합니다. 자세한 내용은 Amazon Glacier의 볼트 알림 구성 단원을 참조하십시오.
- 명시적인 작업 정보 요청: Amazon Glacier `Describe Job` API 작업(작업 설명(GET JobID))을 사용하여 작업 정보를 주기적으로 폴링할 수도 있습니다. 하지만 Amazon SNS 알림 메시지의 사용을 권장합니다.

> **Note**
>
> Amazon SNS 알림을 사용하여 가져오는 정보는 `Describe Job` API 작업을 직접 호출하여 가져오는 정보와 동일합니다.

아카이브 검색 옵션

아카이브 검색 작업을 시작할 때 액세스 시간과 비용 요건을 기준으로 다음 중 한 가지 검색 옵션을 지정할 수 있습니다. 검색 요금에 대한 자세한 내용은 Amazon Glacier 요금을 참조하세요.

- 신속: 아카이브의 복원을 위한 임시 긴급 요청이 필요한 경우 신속 검색을 사용하면 S3 Glacier Flexible 검색 스토리지 클래스 또는 S3 Intelligent-Tiering 아카이브 액세스 티어에 저장된 데이터에 빠르게 액세스할 수 있습니다. 매우 큰 아카이브(250MB 이상)를 제외한 모든 경우, 신속 검색을 사용하여 액세스된 데이터는 일반적으로 1~5분 안에 사용할 수 있습니다. 프로비저닝된 용량을 통해 필요할 때 신속 검색에 대한 검색 용량이 보장됩니다. 자세한 내용은 프로비저닝된 용량 단원을 참조하십시오.
- 표준: 표준 검색을 사용하면 몇 시간 내에 아카이브에 액세스할 수 있습니다. 표준 검색은 보통 3~5시간 안에 완료됩니다. 검색 요청 시 검색 옵션을 지정하지 않을 경우 스탠다드가 기본 옵션이 됩니다.
- 대량: 대량 검색은 Amazon Glacier에서 가장 저렴한 검색 옵션으로 대용량 데이터, 심지어는 페타바이트 규모까지도 저렴한 비용으로 하루만에 검색할 수 있습니다. 대량 검색은 보통 5~12시간 안에 완료됩니다.

다음 테이블에는 아카이브 검색 옵션이 요약되어 있습니다. 요금에 대한 자세한 정보는 Amazon Glacier 요금을 참조하세요.

`Expedited`, `Standard`또는 `Bulk` 검색을 수행하려면 RestoreObject REST API 작업 요청의 `Tier` 요청 요소를 원하는 옵션 또는 AWS Command Line Interface (AWS CLI) 또는 AWS SDKs의 동등한 옵션으로 설정합니다. 프로비저닝된 용량을 구매하였다면, 사용자의 프로비저닝된 용량을 통해 모든 신속 검색이 자동으로 수행됩니다.

프로비저닝된 용량

프로비저닝된 용량으로 필요시에 신속 검색을 위한 검색 용량을 보장합니다. 각 용량 단위로 초당 최대 150메가바이트(MBps)의 검색 처리량이 제공되고 매 5분마다 긴급 검색을 최소 3회 수행할 수 있습니다.

워크로드에 몇 분 내로 데이터의 서브셋에 대한 신뢰성 높고 예측 가능한 액세스가 필요한 경우 프로비저닝된 검색 용량을 구매를 권장합니다. 프로비저닝된 검색 용량이 없더라도 비정상적으로 수요가 높지 않은 경우를 제외하면 일반적으로 신속 검색은 허용됩니다. 하지만 모든 상황에서 신속 검색에 액세스해야 하는 경우 프로비저닝된 검색 용량을 구매해야 합니다.

프로비저닝된 용량 구매

Amazon Glacier 콘솔, 프로비저닝된 용량 구매(POST provisioned-capacity) REST API 작업, AWS SDKs 또는를 사용하여 프로비저닝된 용량 단위를 구매할 수 있습니다 AWS CLI. 프로비저닝된 용량의 요금에 대한 자세한 내용은 Amazon Glacier 요금을 참조하세요.

프로비저닝된 용량 단위는 구매한 날짜 및 시간으로부터 1개월간 지속됩니다.

시작 날짜가 31일인 경우에는 만료 날짜는 다음 달 말일이 됩니다. 예를 들어 시작 날짜가 8월 31일이라면 만료 날짜는 9월 30일입니다. 시작 날짜가 1월 31일이라면 만료 날짜는 2월 28일입니다.

Amazon Glacier 콘솔을 사용하여 프로비저닝된 용량 구매

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/glacier/home Amazon Glacier 콘솔을 엽니다.
2. 왼쪽의 탐색 창에서 데이터 검색 설정을 선택합니다.
3. 프로비저닝된 용량 단위(PCU)에서 PCU 구매를 선택합니다. PCU 구매 대화 상자가 나타납니다.
4. 프로비저닝된 용량을 구매하려는 경우 구매 확인 상자에 **`confirm`**을 입력합니다.
5. PCU 구매를 선택합니다.

범위가 지정된 아카이브 가져오기

Amazon Glacier에서 아카이브를 검색할 때는 선택사항으로 검색할 아카이브의 범위 및 부분을 지정할 수 있습니다. 기본적으로는 아카이브 전체를 가져오도록 되어 있습니다. 바이트 범위 지정은 다음과 같은 경우에 유용합니다.

- 데이터 다운로드 관리: Amazon Glacier는 검색 요청 완료 후 24시간 동안 검색한 데이터를 다운로드할 수 있도록 합니다. 따라서 다운로드 기간 내에 다운로드 일정을 관리할 수 있도록 아카이브에서 일부 구간만 가져올 수도 있습니다.
- 대용량 아카이브의 타게팅된 부분만 검색: 예를 들어, 이전에 다수의 파일을 집계하여 단일 아카이브로 업로드하였지만 지금 그 파일 중 일부만 검색하려고 하는 경우입니다. 이때는 검색 요청 한 번으로 원하는 파일이 저장된 아카이브 범위를 지정할 수 있습니다. 그렇지 않으면 가져오기 요청을 여러 차례 시작하면서 매번 1개 이상의 파일 범위를 지정해야 합니다.

범위 가져오기를 사용하여 작업을 시작할 때는 메가바이트로 정렬된 범위를 입력해야 합니다. 다시 말해서 바이트 범위는 0부터 시작하거나(아카이브의 시작 부분) 혹은 1MB 간격으로(1MB, 2MB, 3MB 등) 시작할 수 있습니다.

범위의 끝은 아카이브의 끝이거나, 범위 시작보다 1MB 단위로 큰 모든 범위일 수 있습니다. 또한 데이터를 다운로드할 때(검색 작업 완료 후) 체크섬 값까지 원하는 경우에는 작업 시작 시 요청한 범위가 트리-해시로 정렬되어야 합니다. 체크섬은 전송 중에 데이터가 손상되지 않았는지 확인할 수 있는 방법입니다. 메가바이트 정렬 및 트리-해시 정렬에 대한 자세한 내용은 데이터 다운로드 시 체크섬 수신 섹션을 참조하세요.

를 사용하여 Amazon Glacier에서 아카이브 다운로드 AWS SDK for Java

Amazon SDK for Java에서 제공하는 하이레벨 및 로우레벨 API 둘 모두는 아카이브를 다운로드하는 방법을 제공합니다.

주제

- 의 상위 수준 API를 사용하여 아카이브 다운로드 AWS SDK for Java
- 의 하위 수준 API를 사용하여 아카이브 다운로드 AWS SDK for Java

의 상위 수준 API를 사용하여 아카이브 다운로드 AWS SDK for Java

하이레벨 API의 `ArchiveTransferManager` 클래스는 아카이브를 다운로드하는 데 사용할 수 있는 `download` 메서드를 제공합니다.

> **Important**
>
> `ArchiveTransferManager` 클래스는 Amazon Simple Notification Service(SNS) 토픽 및 해당 토픽을 구독하는 Amazon Simple Queue Service(Amazon SQS) 대기열을 생성합니다. 그런 다음 아카이브 가져오기 작업을 시작하고 아카이브를 사용할 수 있도록 대기열을 폴링합니다. 아카이브를 사용할 수 있게 되면 다운로드가 시작됩니다. 검색 시간에 대한 자세한 내용은 아카이브 검색 옵션 섹션을 참조하세요.

예:의 상위 수준 API를 사용하여 아카이브 다운로드 AWS SDK for Java

다음은 미국 서부(오레곤) 리전(`us-west-2`)의 볼트(`examplevault`)에서 아카이브를 다운로드하는 Java 코드 예시입니다.

이번 샘플 실행에 대한 단계별 지침은 Eclipse를 사용하여 Amazon Glacier의 Java 예 실행 섹션을 참조하세요. 코드는 기존 아카이브 ID와 다운로드한 아카이브를 저장할 로컬 파일 경로를 사용해 아래와 같이 업데이트해야 합니다.

Example

```
import java.io.File;
import java.io.IOException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.transfer.ArchiveTransferManager;
```

```
import com.amazonaws.services.sns.AmazonSNSClient;
import com.amazonaws.services.sqs.AmazonSQSClient;


public class ArchiveDownloadHighLevel {
    public static String vaultName = "examplevault";
    public static String archiveId = "*** provide archive ID ***";
    public static String downloadFilePath  = "*** provide location to download archive
 ***";

    public static AmazonGlacierClient glacierClient;
    public static AmazonSQSClient sqsClient;
    public static AmazonSNSClient snsClient;

    public static void main(String[] args) throws IOException {

     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        glacierClient = new AmazonGlacierClient(credentials);

        sqsClient = new AmazonSQSClient(credentials);
        snsClient = new AmazonSNSClient(credentials);
        glacierClient.setEndpoint("glacier.us-west-2.amazonaws.com");
        sqsClient.setEndpoint("sqs.us-west-2.amazonaws.com");
        snsClient.setEndpoint("sns.us-west-2.amazonaws.com");

        try {
            ArchiveTransferManager atm = new ArchiveTransferManager(glacierClient,
 sqsClient, snsClient);

            atm.download(vaultName, archiveId, new File(downloadFilePath));
            System.out.println("Downloaded file to " + downloadFilePath);

        } catch (Exception e)
        {
            System.err.println(e);
        }
    }
}
```

의 하위 수준 API를 사용하여 아카이브 다운로드 AWS SDK for Java

다음은 AWS SDK for Java 로우레벨 API를 사용해 볼트 인벤토리를 가져오는 단계입니다.

1. AmazonGlacierClient 클래스(클라이언트)의 인스턴스를 만듭니다.

   아카이브를 다운로드할 AWS 리전을 지정해야 합니다. 이 클라이언트를 사용하여 수행하는 모든 작업은 해당 AWS 리전에 적용됩니다.

2. initiateJob 메서드를 실행하여 archive-retrieval 작업을 시작합니다.

   InitiateJobRequest 클래스 인스턴스를 생성하여 다운로드를 원하는 아카이브의 ID, Amazon Glacier(Amazon Glacier)가 작업 완료 메시지를 게시할 Amazon SNS 토픽(옵션) 등 작업 정보를 입력합니다. Amazon Glacier는 응답으로 작업 ID를 반환합니다. 이 응답은 InitiateJobResult 클래스 인스턴스에서 사용할 수 있습니다.

```
JobParameters jobParameters = new JobParameters()
    .withArchiveId("*** provide an archive id ***")
    .withDescription("archive retrieval")
    .withRetrievalByteRange("*** provide a retrieval range***") // optional
    .withType("archive-retrieval");

InitiateJobResult initiateJobResult = client.initiateJob(new InitiateJobRequest()
    .withJobParameters(jobParameters)
    .withVaultName(vaultName));

String jobId = initiateJobResult.getJobId();
```

   선택적으로 바이트 범위를 지정하여 Amazon Glacier가 아카이브의 일부만 준비하도록 요청할 수 있습니다. 예를 들어 다음 문을 요청에 추가하면 Amazon Glacier가 아카이브 중 1MB~2MB만 준비하도록 요청하여 이전 요청을 업데이트할 수 있습니다.

```
int ONE_MEG = 1048576;
String retrievalByteRange = String.format("%s-%s", ONE_MEG, 2*ONE_MEG -1);

JobParameters jobParameters = new JobParameters()
    .withType("archive-retrieval")
    .withArchiveId(archiveId)
    .withRetrievalByteRange(retrievalByteRange)
    .withSNSTopic(snsTopicARN);

InitiateJobResult initiateJobResult = client.initiateJob(new InitiateJobRequest()
    .withJobParameters(jobParameters)
    .withVaultName(vaultName));
```

```
String jobId = initiateJobResult.getJobId();
```

3. 작업이 완료될 때까지 기다립니다.

   작업 출력을 다운로드할 수 있을 때까지 기다려야 합니다. 볼트에서 Amazon Simple Notification Service(Amazon SNS) 토픽을 식별할 수 있도록 알림 구성을 설정했거나 작업을 시작할 때 Amazon SNS 토픽을 지정했다면 Amazon Glacier가 작업 완료 후 해당 토픽에 메시지를 보냅니다.

   `describeJob` 메서드를 직접 호출하여 Amazon Glacier를 폴링함으로써 작업 완료 상태를 확인하는 방법도 있습니다. 하지만 Amazon SNS 토픽의 알람을 사용해 식별하는 방법을 권장합니다.

4. `getJobOutput` 메서드를 실행하여 작업 출력(아카이브 데이터)을 다운로드합니다.

   `GetJobOutputRequest` 클래스 인스턴스를 생성하여 작업 ID, 볼트 이름 등 요청 정보를 입력합니다. Amazon Glacier가 반환하는 출력은 `GetJobOutputResult` 객체에서 사용할 수 있습니다.

```
GetJobOutputRequest jobOutputRequest = new GetJobOutputRequest()
        .withJobId("*** provide a job ID ***")
        .withVaultName("*** provide a vault name ****");
GetJobOutputResult jobOutputResult = client.getJobOutput(jobOutputRequest);

// jobOutputResult.getBody() // Provides the input stream.
```

   위의 코드 조각은 전체 작업 출력을 다운로드합니다. 옵션으로 `GetJobOutputRequest`에서 바이트 범위를 지정하여 출력 일부만 가져오거나 전체 출력을 더 작은 청크 단위로 나누어 다운로드할 수 있습니다.

```
GetJobOutputRequest jobOutputRequest = new GetJobOutputRequest()
        .withJobId("*** provide a job ID ***")
        .withRange("bytes=0-1048575")   // Download only the first 1 MB of the
 output.
        .withVaultName("*** provide a vault name ****");
```

   특정 조건이 충족된다면 Amazon Glacier가 `GetJobOutput` 직접 호출에 대한 응답으로 다운로드한 데이터 일부의 체크섬을 반환합니다. 자세한 내용은 데이터 다운로드 시 체크섬 수신 단원을 참조하십시오.

다운로드에 오류가 없는지 확인하려면 클라이언트 측 체크섬을 계산한 뒤 Amazon Glacier가 응답으로 보내는 체크섬과 비교합니다.

아카이브 가져오기 작업에서 옵션으로 범위를 지정한 경우에는 작업 설명에 가져오는 범위의 체크섬도 포함됩니다(SHA256TreeHash). 이 값은 나중에 다운로드하는 전체 바이트 범위의 정확성을 검증하는 데도 사용할 수 있습니다. 예를 들어 트리-해시로 정렬된 아카이브 범위를 가져오는 작업을 시작하면서 `GetJobOutput` 요청이 각각 체크섬을 반환할 수 있도록 청크 단위로 출력을 다운로드하는 경우에는 클라이언트 측에서 다운로드하는 개별 데이터의 체크섬을 계산하고 나서 트리 해시를 계산할 수 있습니다. Amazon Glacier가 작업 설명 요청에 대한 응답으로 반환하는 체크섬과 비교하여 사용자가 다운로드한 전체 바이트 범위가 Amazon Glacier에 저장된 바이트 범위와 일치하는지 확인할 수 있습니다.

사용 가능한 예제는 예제 2:의 하위 수준 API를 사용하여 아카이브 검색 AWS SDK for Java- 청크 단위로 출력 다운로드 섹션을 참조하세요.

예제 1:의 하위 수준 API를 사용하여 아카이브 검색 AWS SDK for Java

다음은 지정된 볼트에서 아카이브를 다운로드하는 Java 코드 예제입니다. 작업을 마치면 `getJobOutput` 호출 한 번으로 전체 출력을 다운로드합니다. 출력을 청크 단위로 다운로드하는 예제는 예제 2:의 하위 수준 API를 사용하여 아카이브 검색 AWS SDK for Java- 청크 단위로 출력 다운로드 섹션을 참조하세요.

이 예에서는 다음과 같은 작업을 수행합니다.

- Amazon Simple Notification Service(Amazon SNS) 토픽을 생성합니다.

  Amazon Glacier는 작업 완료 후 해당 토픽에 알림을 전송합니다.
- Amazon Simple Queue Service(Amazon SQS) 대기열을 생성합니다.

  아래 예시는 Amazon SNS 토픽이 메시지를 대기열에 게시할 수 있도록 여기에서 해당 대기열에 정책을 연결합니다.
- 지정된 아카이브의 다운로드 작업을 시작합니다.

  작업 요청에서 Amazon Glacier가 작업 완료 후 알림 메시지를 해당 토픽에 게시할 수 있도록 생성된 Amazon SNS 토픽이 지정됩니다.
- 작업 ID가 포함된 메시지가 있는지 Amazon SQS 대기열을 주기적으로 확인합니다.

메시지가 있으면 JSON 구문을 분석하여 작업이 성공적으로 완료되었는지 확인합니다. 성공적으로 완료되었으면 아카이브를 다운로드합니다.

- Amazon SNS 토픽과 생성된 Amazon SQS 대기열을 삭제하여 정리합니다.

```
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.codehaus.jackson.JsonFactory;
import org.codehaus.jackson.JsonNode;
import org.codehaus.jackson.JsonParseException;
import org.codehaus.jackson.JsonParser;
import org.codehaus.jackson.map.ObjectMapper;

import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Principal;
import com.amazonaws.auth.policy.Resource;
import com.amazonaws.auth.policy.Statement;
import com.amazonaws.auth.policy.Statement.Effect;
import com.amazonaws.auth.policy.actions.SQSActions;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.model.GetJobOutputRequest;
import com.amazonaws.services.glacier.model.GetJobOutputResult;
import com.amazonaws.services.glacier.model.InitiateJobRequest;
import com.amazonaws.services.glacier.model.InitiateJobResult;
import com.amazonaws.services.glacier.model.JobParameters;
import com.amazonaws.services.sns.AmazonSNSClient;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.CreateTopicResult;
```

```
import com.amazonaws.services.sns.model.DeleteTopicRequest;
import com.amazonaws.services.sns.model.SubscribeRequest;
import com.amazonaws.services.sns.model.SubscribeResult;
import com.amazonaws.services.sns.model.UnsubscribeRequest;
import com.amazonaws.services.sqs.AmazonSQSClient;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.CreateQueueResult;
import com.amazonaws.services.sqs.model.DeleteQueueRequest;
import com.amazonaws.services.sqs.model.GetQueueAttributesRequest;
import com.amazonaws.services.sqs.model.GetQueueAttributesResult;
import com.amazonaws.services.sqs.model.Message;
import com.amazonaws.services.sqs.model.ReceiveMessageRequest;
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;


public class AmazonGlacierDownloadArchiveWithSQSPolling {

    public static String archiveId = "*** provide archive ID ****";
    public static String vaultName = "*** provide vault name ***";
    public static String snsTopicName = "*** provide topic name ***";
    public static String sqsQueueName = "*** provide queue name ***";
    public static String sqsQueueARN;
    public static String sqsQueueURL;
    public static String snsTopicARN;
    public static String snsSubscriptionARN;
    public static String fileName = "*** provide file name ***";
    public static String region = "*** region ***";
    public static long sleepTime = 600;
    public static AmazonGlacierClient client;
    public static AmazonSQSClient sqsClient;
    public static AmazonSNSClient snsClient;

    public static void main(String[] args) throws IOException {

     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        client = new AmazonGlacierClient(credentials);
        client.setEndpoint("https://glacier." + region + ".amazonaws.com");
        sqsClient = new AmazonSQSClient(credentials);
        sqsClient.setEndpoint("https://sqs." + region + ".amazonaws.com");
        snsClient = new AmazonSNSClient(credentials);
        snsClient.setEndpoint("https://sns." + region + ".amazonaws.com");

        try {
```

```
            setupSQS();

            setupSNS();

            String jobId = initiateJobRequest();
            System.out.println("Jobid = " + jobId);

            Boolean success = waitForJobToComplete(jobId, sqsQueueURL);
            if (!success) { throw new Exception("Job did not complete
successfully."); }

            downloadJobOutput(jobId);

            cleanUp();

        } catch (Exception e) {
            System.err.println("Archive retrieval failed.");
            System.err.println(e);
        }
    }

    private static void setupSQS() {
        CreateQueueRequest request = new CreateQueueRequest()
            .withQueueName(sqsQueueName);
        CreateQueueResult result = sqsClient.createQueue(request);
        sqsQueueURL = result.getQueueUrl();

        GetQueueAttributesRequest qRequest = new GetQueueAttributesRequest()
            .withQueueUrl(sqsQueueURL)
            .withAttributeNames("QueueArn");

        GetQueueAttributesResult qResult = sqsClient.getQueueAttributes(qRequest);
        sqsQueueARN = qResult.getAttributes().get("QueueArn");

        Policy sqsPolicy =
            new Policy().withStatements(
                    new Statement(Effect.Allow)
                    .withPrincipals(Principal.AllUsers)
                    .withActions(SQSActions.SendMessage)
                    .withResources(new Resource(sqsQueueARN)));
        Map<String, String> queueAttributes = new HashMap<String, String>();
        queueAttributes.put("Policy", sqsPolicy.toJson());
        sqsClient.setQueueAttributes(new SetQueueAttributesRequest(sqsQueueURL,
queueAttributes));
```

```
    }
    private static void setupSNS() {
        CreateTopicRequest request = new CreateTopicRequest()
            .withName(snsTopicName);
        CreateTopicResult result = snsClient.createTopic(request);
        snsTopicARN = result.getTopicArn();

        SubscribeRequest request2 = new SubscribeRequest()
            .withTopicArn(snsTopicARN)
            .withEndpoint(sqsQueueARN)
            .withProtocol("sqs");
        SubscribeResult result2 = snsClient.subscribe(request2);

        snsSubscriptionARN = result2.getSubscriptionArn();
    }
    private static String initiateJobRequest() {

        JobParameters jobParameters = new JobParameters()
            .withType("archive-retrieval")
            .withArchiveId(archiveId)
            .withSNSTopic(snsTopicARN);

        InitiateJobRequest request = new InitiateJobRequest()
            .withVaultName(vaultName)
            .withJobParameters(jobParameters);

        InitiateJobResult response = client.initiateJob(request);

        return response.getJobId();
    }

    private static Boolean waitForJobToComplete(String jobId, String sqsQueueUrl)
throws InterruptedException, JsonParseException, IOException {

        Boolean messageFound = false;
        Boolean jobSuccessful = false;
        ObjectMapper mapper = new ObjectMapper();
        JsonFactory factory = mapper.getJsonFactory();

        while (!messageFound) {
            List<Message> msgs = sqsClient.receiveMessage(
               new
ReceiveMessageRequest(sqsQueueUrl).withMaxNumberOfMessages(10)).getMessages();
```

```
            if (msgs.size() > 0) {
                for (Message m : msgs) {
                    JsonParser jpMessage = factory.createJsonParser(m.getBody());
                    JsonNode jobMessageNode = mapper.readTree(jpMessage);
                    String jobMessage = jobMessageNode.get("Message").getTextValue();

                    JsonParser jpDesc = factory.createJsonParser(jobMessage);
                    JsonNode jobDescNode = mapper.readTree(jpDesc);
                    String retrievedJobId = jobDescNode.get("JobId").getTextValue();
                    String statusCode = jobDescNode.get("StatusCode").getTextValue();
                    if (retrievedJobId.equals(jobId)) {
                        messageFound = true;
                        if (statusCode.equals("Succeeded")) {
                            jobSuccessful = true;
                        }
                    }
                }

            } else {
              Thread.sleep(sleepTime * 1000);
            }
          }
        return (messageFound && jobSuccessful);
    }

    private static void downloadJobOutput(String jobId) throws IOException {

        GetJobOutputRequest getJobOutputRequest = new GetJobOutputRequest()
            .withVaultName(vaultName)
            .withJobId(jobId);
        GetJobOutputResult getJobOutputResult =
client.getJobOutput(getJobOutputRequest);

        InputStream input = new BufferedInputStream(getJobOutputResult.getBody());
        OutputStream output = null;
        try {
            output = new BufferedOutputStream(new FileOutputStream(fileName));

            byte[] buffer = new byte[1024 * 1024];

            int bytesRead = 0;
            do {
                bytesRead = input.read(buffer);
```

```
                if (bytesRead <= 0) break;
                output.write(buffer, 0, bytesRead);
            } while (bytesRead > 0);
        } catch (IOException e) {
            throw new AmazonClientException("Unable to save archive", e);
        } finally {
            try {input.close();}  catch (Exception e) {}
            try {output.close();} catch (Exception e) {}
        }
        System.out.println("Retrieved archive to " + fileName);
    }

    private static void cleanUp() {
        snsClient.unsubscribe(new UnsubscribeRequest(snsSubscriptionARN));
        snsClient.deleteTopic(new DeleteTopicRequest(snsTopicARN));
        sqsClient.deleteQueue(new DeleteQueueRequest(sqsQueueURL));
    }
}
```

예제 2:의 하위 수준 API를 사용하여 아카이브 검색 AWS SDK for Java- 청크 단위로 출력 다운로드

다음은 Amazon Glacier에서 아카이브를 검색하는 Java 코드 예입니다. 이 코드 예제는 `GetJobOutputRequest` 객체에서 바이트 범위를 지정하여 작업 출력을 청크 단위로 다운로드합니다.

```
import java.io.BufferedInputStream;
import java.io.ByteArrayInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.core.JsonParseException;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;

import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Principal;
import com.amazonaws.auth.policy.Resource;
```

```
import com.amazonaws.auth.policy.Statement;
import com.amazonaws.auth.policy.Statement.Effect;
import com.amazonaws.auth.policy.actions.SQSActions;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.TreeHashGenerator;
import com.amazonaws.services.glacier.model.GetJobOutputRequest;
import com.amazonaws.services.glacier.model.GetJobOutputResult;
import com.amazonaws.services.glacier.model.InitiateJobRequest;
import com.amazonaws.services.glacier.model.InitiateJobResult;
import com.amazonaws.services.glacier.model.JobParameters;
import com.amazonaws.services.sns.AmazonSNSClient;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.CreateTopicResult;
import com.amazonaws.services.sns.model.DeleteTopicRequest;
import com.amazonaws.services.sns.model.SubscribeRequest;
import com.amazonaws.services.sns.model.SubscribeResult;
import com.amazonaws.services.sns.model.UnsubscribeRequest;
import com.amazonaws.services.sqs.AmazonSQSClient;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.CreateQueueResult;
import com.amazonaws.services.sqs.model.DeleteQueueRequest;
import com.amazonaws.services.sqs.model.GetQueueAttributesRequest;
import com.amazonaws.services.sqs.model.GetQueueAttributesResult;
import com.amazonaws.services.sqs.model.Message;
import com.amazonaws.services.sqs.model.ReceiveMessageRequest;
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;


public class ArchiveDownloadLowLevelWithRange {

    public static String vaultName = "*** provide vault name ***";
    public static String archiveId = "*** provide archive id ***";
    public static String snsTopicName = "glacier-temp-sns-topic";
    public static String sqsQueueName = "glacier-temp-sqs-queue";
    public static long downloadChunkSize = 4194304; // 4 MB
    public static String sqsQueueARN;
    public static String sqsQueueURL;
    public static String snsTopicARN;
    public static String snsSubscriptionARN;
    public static String fileName = "*** provide file name to save archive to ***";
    public static String region   = "*** region ***";
    public static long sleepTime  = 600;
```

```
    public static AmazonGlacierClient client;
    public static AmazonSQSClient sqsClient;
    public static AmazonSNSClient snsClient;

    public static void main(String[] args) throws IOException {

     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        client = new AmazonGlacierClient(credentials);
        client.setEndpoint("https://glacier." + region + ".amazonaws.com");
        sqsClient = new AmazonSQSClient(credentials);
        sqsClient.setEndpoint("https://sqs." + region + ".amazonaws.com");
        snsClient = new AmazonSNSClient(credentials);
        snsClient.setEndpoint("https://sns." + region + ".amazonaws.com");

        try {
            setupSQS();

            setupSNS();

            String jobId = initiateJobRequest();
            System.out.println("Jobid = " + jobId);

            long archiveSizeInBytes = waitForJobToComplete(jobId, sqsQueueURL);
            if (archiveSizeInBytes==-1) { throw new Exception("Job did not complete
successfully."); }

            downloadJobOutput(jobId, archiveSizeInBytes);

            cleanUp();

        } catch (Exception e) {
            System.err.println("Archive retrieval failed.");
            System.err.println(e);
        }
    }

    private static void setupSQS() {
        CreateQueueRequest request = new CreateQueueRequest()
            .withQueueName(sqsQueueName);
        CreateQueueResult result = sqsClient.createQueue(request);
        sqsQueueURL = result.getQueueUrl();

        GetQueueAttributesRequest qRequest = new GetQueueAttributesRequest()
```

```
            .withQueueUrl(sqsQueueURL)
            .withAttributeNames("QueueArn");

        GetQueueAttributesResult qResult = sqsClient.getQueueAttributes(qRequest);
        sqsQueueARN = qResult.getAttributes().get("QueueArn");

        Policy sqsPolicy =
            new Policy().withStatements(
                    new Statement(Effect.Allow)
                    .withPrincipals(Principal.AllUsers)
                    .withActions(SQSActions.SendMessage)
                    .withResources(new Resource(sqsQueueARN)));
        Map<String, String> queueAttributes = new HashMap<String, String>();
        queueAttributes.put("Policy", sqsPolicy.toJson());
        sqsClient.setQueueAttributes(new SetQueueAttributesRequest(sqsQueueURL,
queueAttributes));

    }
    private static void setupSNS() {
        CreateTopicRequest request = new CreateTopicRequest()
            .withName(snsTopicName);
        CreateTopicResult result = snsClient.createTopic(request);
        snsTopicARN = result.getTopicArn();

        SubscribeRequest request2 = new SubscribeRequest()
            .withTopicArn(snsTopicARN)
            .withEndpoint(sqsQueueARN)
            .withProtocol("sqs");
        SubscribeResult result2 = snsClient.subscribe(request2);

        snsSubscriptionARN = result2.getSubscriptionArn();
    }
    private static String initiateJobRequest() {

        JobParameters jobParameters = new JobParameters()
            .withType("archive-retrieval")
            .withArchiveId(archiveId)
            .withSNSTopic(snsTopicARN);

        InitiateJobRequest request = new InitiateJobRequest()
            .withVaultName(vaultName)
            .withJobParameters(jobParameters);

        InitiateJobResult response = client.initiateJob(request);
```

```
        return response.getJobId();
    }

    private static long waitForJobToComplete(String jobId, String sqsQueueUrl) throws
InterruptedException, JsonParseException, IOException {

        Boolean messageFound = false;
        Boolean jobSuccessful = false;
        long archiveSizeInBytes = -1;
        ObjectMapper mapper = new ObjectMapper();
        JsonFactory factory = mapper.getFactory();

        while (!messageFound) {
            List<Message> msgs = sqsClient.receiveMessage(
               new
ReceiveMessageRequest(sqsQueueUrl).withMaxNumberOfMessages(10)).getMessages();

            if (msgs.size() > 0) {
                for (Message m : msgs) {
                    JsonParser jpMessage = factory.createJsonParser(m.getBody());
                    JsonNode jobMessageNode = mapper.readTree(jpMessage);
                    String jobMessage = jobMessageNode.get("Message").textValue();

                    JsonParser jpDesc = factory.createJsonParser(jobMessage);
                    JsonNode jobDescNode = mapper.readTree(jpDesc);
                    String retrievedJobId = jobDescNode.get("JobId").textValue();
                    String statusCode = jobDescNode.get("StatusCode").textValue();
                    archiveSizeInBytes =
jobDescNode.get("ArchiveSizeInBytes").longValue();
                    if (retrievedJobId.equals(jobId)) {
                        messageFound = true;
                        if (statusCode.equals("Succeeded")) {
                            jobSuccessful = true;
                        }
                    }
                }

            } else {
              Thread.sleep(sleepTime * 1000);
            }
          }
        return (messageFound && jobSuccessful) ? archiveSizeInBytes : -1;
    }
```

```
   private static void downloadJobOutput(String jobId, long archiveSizeInBytes) throws
IOException {

       if (archiveSizeInBytes < 0) {
           System.err.println("Nothing to download.");
           return;
       }

       System.out.println("archiveSizeInBytes: " + archiveSizeInBytes);
       FileOutputStream fstream = new FileOutputStream(fileName);
       long startRange = 0;
       long endRange = (downloadChunkSize > archiveSizeInBytes) ? archiveSizeInBytes
-1 : downloadChunkSize - 1;

       do {

           GetJobOutputRequest getJobOutputRequest = new GetJobOutputRequest()
               .withVaultName(vaultName)
               .withRange("bytes=" + startRange + "-" + endRange)
               .withJobId(jobId);
           GetJobOutputResult getJobOutputResult =
client.getJobOutput(getJobOutputRequest);

           BufferedInputStream is = new
BufferedInputStream(getJobOutputResult.getBody());
           byte[] buffer = new byte[(int)(endRange - startRange + 1)];

           System.out.println("Checksum received: " +
getJobOutputResult.getChecksum());
           System.out.println("Content range " +
getJobOutputResult.getContentRange());


           int totalRead = 0;
           while (totalRead < buffer.length) {
               int bytesRemaining = buffer.length - totalRead;
               int read = is.read(buffer, totalRead, bytesRemaining);
               if (read > 0) {
                   totalRead = totalRead + read;
               } else {
                   break;
               }
```

```
            }
            System.out.println("Calculated checksum: " +
 TreeHashGenerator.calculateTreeHash(new ByteArrayInputStream(buffer)));
            System.out.println("read = " + totalRead);
            fstream.write(buffer);

            startRange = startRange + (long)totalRead;
            endRange = ((endRange + downloadChunkSize) >  archiveSizeInBytes) ?
 archiveSizeInBytes : (endRange + downloadChunkSize);
            is.close();
        } while (endRange <= archiveSizeInBytes  && startRange < archiveSizeInBytes);

        fstream.close();
        System.out.println("Retrieved file to " + fileName);

    }

    private static void cleanUp() {
        snsClient.unsubscribe(new UnsubscribeRequest(snsSubscriptionARN));
        snsClient.deleteTopic(new DeleteTopicRequest(snsTopicARN));
        sqsClient.deleteQueue(new DeleteQueueRequest(sqsQueueURL));
    }
}
```

를 사용하여 Amazon Glacier에서 아카이브 다운로드 AWS SDK for .NET

Amazon SDK for .NET에서 제공하는 하이레벨 및 로우레벨 API 둘 모두는 아카이브를 다운로드하는 방법을 제공합니다.

주제

- 의 상위 수준 API를 사용하여 아카이브 다운로드 AWS SDK for .NET
- 의 하위 수준 API를 사용하여 아카이브 다운로드 AWS SDK for .NET

의 상위 수준 API를 사용하여 아카이브 다운로드 AWS SDK for .NET

하이레벨 API의 `ArchiveTransferManager` 클래스는 아카이브를 다운로드하는 데 사용할 수 있는 `Download` 메서드를 제공합니다.

> **Important**
>
> `ArchiveTransferManager` 클래스는 Amazon Simple Notification Service(SNS) 토픽 및 해당 토픽을 구독하는 Amazon Simple Queue Service(Amazon SQS) 대기열을 생성합니다. 그런 다음 아카이브 가져오기 작업을 시작하고 아카이브를 사용할 수 있도록 대기열을 폴링합니다. 아카이브를 사용할 수 있게 되면 다운로드가 시작됩니다. 검색 시간에 대한 자세한 내용은 아카이브 검색 옵션 섹션을 참조하세요.

예:의 상위 수준 API를 사용하여 아카이브 다운로드 AWS SDK for .NET

다음은 미국 서부(오레곤) 리전의 볼트(`examplevault`)에서 아카이브를 다운로드하는 C# 코드 예시입니다.

이 예제의 실행 방법에 대한 단계별 지침은 코드 예제 실행 섹션을 참조하세요. 코드는 기존 아카이브 ID와 다운로드한 아카이브를 저장할 로컬 파일 경로를 사용해 아래와 같이 업데이트해야 합니다.

```
using System;
using Amazon.Glacier;
using Amazon.Glacier.Transfer;
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
  class ArchiveDownloadHighLevel
  {
    static string vaultName        = "examplevault";
    static string archiveId        = "*** Provide archive ID ***";
    static string downloadFilePath = "*** Provide the file name and path to where to
 store the download ***";

    public static void Main(string[] args)
    {
      try
      {
        var manager = new ArchiveTransferManager(Amazon.RegionEndpoint.USWest2);

        var options = new DownloadOptions();
        options.StreamTransferProgress += ArchiveDownloadHighLevel.progress;
        // Download an archive.
        Console.WriteLine("Intiating the archive retrieval job and then polling SQS
 queue for the archive to be available.");
```

```
        Console.WriteLine("Once the archive is available, downloading will begin.");
        manager.Download(vaultName, archiveId, downloadFilePath, options);
        Console.WriteLine("To continue, press Enter");
        Console.ReadKey();
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }

    static int currentPercentage = -1;
    static void progress(object sender, StreamTransferProgressArgs args)
    {
      if (args.PercentDone != currentPercentage)
      {
        currentPercentage = args.PercentDone;
        Console.WriteLine("Downloaded {0}%", args.PercentDone);
      }
    }
  }
}
```

의 하위 수준 API를 사용하여 아카이브 다운로드 AWS SDK for .NET

다음은 AWS SDK for .NET의 로우레벨 API를 사용해 Amazon Glacier(Amazon Glacier) 아카이브를 다운로드하는 단계입니다.

1. `AmazonGlacierClient` 클래스(클라이언트)의 인스턴스를 만듭니다.

   아카이브를 다운로드할 AWS 리전을 지정해야 합니다. 이 클라이언트를 사용하여 수행하는 모든 작업은 해당 AWS 리전에 적용됩니다.
2. `InitiateJob` 메서드를 실행하여 `archive-retrieval` 작업을 시작합니다.

   `InitiateJobRequest` 클래스 인스턴스를 생성하여 다운로드를 원하는 아카이브의 ID, Amazon Glacier가 작업 완료 메시지를 게시할 Amazon SNS 토픽(옵션) 등 작업 정보를 제공합니다. Amazon Glacier는 응답으로 작업 ID를 반환합니다. 이 응답은 `InitiateJobResponse` 클래스 인스턴스에서 사용할 수 있습니다.

   ```
   AmazonGlacierClient client;
   client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2);
   ```

```
InitiateJobRequest initJobRequest = new InitiateJobRequest()
{
  VaultName = vaultName,
  JobParameters = new JobParameters()
  {
    Type      = "archive-retrieval",
    ArchiveId = "*** Provide archive id ***",
    SNSTopic  = "*** Provide Amazon SNS topic ARN ***",
  }
};

InitiateJobResponse initJobResponse = client.InitiateJob(initJobRequest);
string jobId = initJobResponse.JobId;
```

다음 요청과 같이 선택적으로 바이트 범위를 지정하여 Amazon Glacier에 아카이브의 일부만 준비하도록 요청할 수 있습니다. 그러면 Amazon Glacier는 요청이 지정한 바에 따라 아카이브에서 1~2MB의 데이터만 준비합니다.

```
AmazonGlacierClient client;
client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2);


InitiateJobRequest initJobRequest = new InitiateJobRequest()
{
  VaultName = vaultName,
  JobParameters = new JobParameters()
  {
    Type      = "archive-retrieval",
    ArchiveId = "*** Provide archive id ***",
    SNSTopic  = "*** Provide Amazon SNS topic ARN ***",
  }
};
// Specify byte range.
int ONE_MEG = 1048576;
initJobRequest.JobParameters.RetrievalByteRange = string.Format("{0}-{1}", ONE_MEG, 2
 * ONE_MEG -1);

InitiateJobResponse initJobResponse = client.InitiateJob(initJobRequest);
string jobId = initJobResponse.JobId;
```

3. 작업이 완료될 때까지 기다립니다.

작업 출력을 다운로드할 수 있을 때까지 기다려야 합니다. 볼트에서 Amazon Simple Notification Service(Amazon SNS) 토픽을 식별할 수 있도록 알림 구성을 설정했거나 작업을 시작할 때 Amazon SNS 토픽을 지정했다면 Amazon Glacier가 작업 완료 후 해당 토픽에 메시지를 보냅니다. 다음 섹션에서 제공하는 코드 예시는 Amazon Glacier가 메시지를 게시할 수 있도록 Amazon SNS를 사용합니다.

`DescribeJob` 메서드를 직접 호출하여 Amazon Glacier를 폴링함으로써 작업 완료 상태를 확인하는 방법도 있습니다. 하지만 Amazon SNS 토픽의 알림을 사용해 식별하는 방법이 권장됩니다.

4. `GetJobOutput` 메서드를 실행하여 작업 출력(아카이브 데이터)을 다운로드합니다.

   `GetJobOutputRequest` 클래스 인스턴스를 생성하여 작업 ID, 볼트 이름 등 요청 정보를 입력합니다. Amazon Glacier가 반환하는 출력은 `GetJobOutputResponse` 객체에서 사용할 수 있습니다.

```
GetJobOutputRequest getJobOutputRequest = new GetJobOutputRequest()
{
  JobId = jobId,
  VaultName = vaultName
};

GetJobOutputResponse getJobOutputResponse = client.GetJobOutput(getJobOutputRequest);
using (Stream webStream = getJobOutputResponse.Body)
{
  using (Stream fileToSave = File.OpenWrite(fileName))
  {
     CopyStream(webStream, fileToSave);
  }
}
```

   위의 코드 조각은 전체 작업 출력을 다운로드합니다. 옵션으로 `GetJobOutputRequest`에서 바이트 범위를 지정하여 출력 일부만 가져오거나 전체 출력을 더 작은 청크 단위로 나누어 다운로드할 수 있습니다.

```
GetJobOutputRequest getJobOutputRequest = new GetJobOutputRequest()
{
  JobId = jobId,
  VaultName = vaultName
};
getJobOutputRequest.SetRange(0, 1048575); // Download only the first 1 MB chunk of
 the output.
```

특정 조건이 충족된다면 Amazon Glacier가 `GetJobOutput` 직접 호출에 대한 응답으로 다운로드한 데이터 일부의 체크섬을 반환합니다. 자세한 내용은 데이터 다운로드 시 체크섬 수신 단원을 참조하십시오.

다운로드에 오류가 없는지 확인하려면 클라이언트 측 체크섬을 계산한 뒤 Amazon Glacier가 응답으로 보내는 체크섬과 비교합니다.

옵션으로 범위를 지정하여 아카이브를 가져오는 작업에서는 작업 설명을 다운로드할 때 가져오는 범위에 대한 체크섬(SHA256 트리-해시)도 포함됩니다. 이 체크섬 값은 나중에 다운로드하는 전체 바이트 범위의 정확성을 검증하는 데 사용할 수 있습니다. 예를 들어 트리-해시로 정렬된 아카이브 범위를 가져오는 작업을 시작하면서 `GetJobOutput` 요청이 각각 체크섬을 반환할 수 있도록 청크 단위로 출력을 다운로드하는 경우에는 클라이언트 측에서 다운로드하는 개별 데이터의 체크섬을 계산하고 나서 트리 해시를 계산할 수 있습니다. Amazon Glacier가 작업 설명 요청에 대한 응답으로 반환하는 체크섬과 비교하여 사용자가 다운로드한 전체 바이트 범위가 Amazon Glacier에 저장된 바이트 범위와 일치하는지 확인할 수 있습니다.

사용 가능한 예제는 예제 2:의 하위 수준 API를 사용하여 아카이브 검색 AWS SDK for .NET- 청크 단위로 출력 다운로드 섹션을 참조하세요.

예제 1:의 하위 수준 API를 사용하여 아카이브 검색 AWS SDK for .NET

다음은 지정된 볼트에서 아카이브를 다운로드하는 C# 코드 예제입니다. 작업을 마치면 `GetJobOutput` 호출 한 번으로 전체 출력을 다운로드합니다. 출력을 청크 단위로 다운로드하는 예제는 예제 2:의 하위 수준 API를 사용하여 아카이브 검색 AWS SDK for .NET- 청크 단위로 출력 다운로드 섹션을 참조하세요.

이 예에서는 다음과 같은 작업을 수행합니다.

- Amazon Simple Notification Service(Amazon SNS) 토픽 설정

  Amazon Glacier는 작업 완료 후 해당 토픽에 알림을 전송합니다.
- Amazon Simple Queue Service(Amazon SQS) 대기열을 설정합니다.

  아래 예시는 Amazon SNS 토픽이 메시지를 게시할 수 있도록 대기열에 정책을 연결합니다.
- 지정된 아카이브의 다운로드 작업을 시작합니다.

  아래 예시는 Amazon Glacier가 작업 완료 후 메시지를 전송할 수 있도록 작업 요청에서 Amazon SNS 토픽을 지정합니다.

- Amazon SQS 대기열의 메시지를 주기적으로 확인합니다.

  메시지가 있으면 JSON 구문을 분석하여 작업이 성공적으로 완료되었는지 확인합니다. 성공적으로 완료되었으면 아카이브를 다운로드합니다. 아래 코드 예제에서는 JSON.NET 라이브러리(JSON.NET 참조)를 사용하여 JSON 구문을 분석합니다.

- Amazon SNS 토픽과 토픽이 생성한 Amazon SQS 대기열을 삭제하여 정리합니다.

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading;
using Amazon.Glacier;
using Amazon.Glacier.Model;
using Amazon.Runtime;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;
using Amazon.SQS;
using Amazon.SQS.Model;
using Newtonsoft.Json;

namespace glacier.amazon.com.docsamples
{
  class ArchiveDownloadLowLevelUsingSNSSQS
  {
    static string topicArn;
    static string queueUrl;
    static string queueArn;
    static string vaultName = "*** Provide vault name ***";
    static string archiveID = "*** Provide archive ID ***";
    static string fileName  = "*** Provide the file name and path to where to store
 downloaded archive ***";
    static AmazonSimpleNotificationServiceClient snsClient;
    static AmazonSQSClient sqsClient;
    const string SQS_POLICY =
        "{" +
        "    \"Version\" : \"2012-10-17\",&TCX5-2025-waiver;" +
        "    \"Statement\" : [" +
        "        {" +
        "            \"Sid\" : \"sns-rule\"," +
        "            \"Effect\" : \"Allow\"," +
        "            \"Principal\" : {\"Service\" : \"sns.amazonaws.com\" }," +
        "            \"Action\"    : \"sqs:SendMessage\"," +
```

```
        "            \"Resource\"  : \"{QueueArn}\"," +
        "            \"Condition\" : {" +
        "                \"ArnLike\" : {" +
        "                    \"aws:SourceArn\" : \"{TopicArn}\"" +
        "                }" +
        "            }" +
        "        }" +
        "    ]" +
        "}";

    public static void Main(string[] args)
    {
      AmazonGlacierClient client;
      try
      {
        using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2))
        {
          Console.WriteLine("Setup SNS topic and SQS queue.");
          SetupTopicAndQueue();
          Console.WriteLine("To continue, press Enter"); Console.ReadKey();
          Console.WriteLine("Retrieving...");
          RetrieveArchive(client);
        }
        Console.WriteLine("Operations successful. To continue, press Enter");
        Console.ReadKey();
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      finally
      {
        // Delete SNS topic and SQS queue.
        snsClient.DeleteTopic(new DeleteTopicRequest() { TopicArn = topicArn });
        sqsClient.DeleteQueue(new DeleteQueueRequest() { QueueUrl = queueUrl });
      }
    }

    static void SetupTopicAndQueue()
    {
      snsClient = new
AmazonSimpleNotificationServiceClient(Amazon.RegionEndpoint.USWest2);
      sqsClient = new AmazonSQSClient(Amazon.RegionEndpoint.USWest2);

      long ticks = DateTime.Now.Ticks;
```

```
      topicArn = snsClient.CreateTopic(new CreateTopicRequest { Name =
"GlacierDownload-" + ticks }).TopicArn;
      Console.Write("topicArn: "); Console.WriteLine(topicArn);

      CreateQueueRequest createQueueRequest = new CreateQueueRequest();
      createQueueRequest.QueueName = "GlacierDownload-" + ticks;
      CreateQueueResponse createQueueResponse =
sqsClient.CreateQueue(createQueueRequest);
      queueUrl = createQueueResponse.QueueUrl;
      Console.Write("QueueURL: "); Console.WriteLine(queueUrl);

      GetQueueAttributesRequest getQueueAttributesRequest = new
GetQueueAttributesRequest();
      getQueueAttributesRequest.AttributeNames = new List<string> { "QueueArn" };
      getQueueAttributesRequest.QueueUrl = queueUrl;
      GetQueueAttributesResponse response =
sqsClient.GetQueueAttributes(getQueueAttributesRequest);
      queueArn = response.QueueARN;
      Console.Write("QueueArn: "); Console.WriteLine(queueArn);

      // Setup the Amazon SNS topic to publish to the SQS queue.
      snsClient.Subscribe(new SubscribeRequest()
      {
        Protocol = "sqs",
        Endpoint = queueArn,
        TopicArn = topicArn
      });

      // Add policy to the queue so SNS can send messages to the queue.
      var policy = SQS_POLICY.Replace("{TopicArn}", topicArn).Replace("{QueueArn}",
queueArn);

      sqsClient.SetQueueAttributes(new SetQueueAttributesRequest()
      {
          QueueUrl = queueUrl,
          Attributes = new Dictionary<string, string>
          {
              { QueueAttributeName.Policy, policy }
          }
      });
   }

   static void RetrieveArchive(AmazonGlacierClient client)
   {
```

```
      // Initiate job.
      InitiateJobRequest initJobRequest = new InitiateJobRequest()
      {
        VaultName = vaultName,
        JobParameters = new JobParameters()
        {
          Type = "archive-retrieval",
          ArchiveId = archiveID,
          Description = "This job is to download archive.",
          SNSTopic = topicArn,
        }
      };
      InitiateJobResponse initJobResponse = client.InitiateJob(initJobRequest);
      string jobId = initJobResponse.JobId;

      // Check queue for a message and if job completed successfully, download archive.
      ProcessQueue(jobId, client);
    }

    private static void ProcessQueue(string jobId, AmazonGlacierClient client)
    {
      ReceiveMessageRequest receiveMessageRequest = new ReceiveMessageRequest()
{ QueueUrl = queueUrl, MaxNumberOfMessages = 1 };
      bool jobDone = false;
      while (!jobDone)
      {
        Console.WriteLine("Poll SQS queue");
        ReceiveMessageResponse receiveMessageResponse =
sqsClient.ReceiveMessage(receiveMessageRequest);
        if (receiveMessageResponse.Messages.Count == 0)
        {
          Thread.Sleep(10000 * 60);
          continue;
        }
        Console.WriteLine("Got message");
        Message message = receiveMessageResponse.Messages[0];
        Dictionary<string, string> outerLayer =
JsonConvert.DeserializeObject<Dictionary<string, string>>(message.Body);
        Dictionary<string, object> fields =
JsonConvert.DeserializeObject<Dictionary<string, object>>(outerLayer["Message"]);
        string statusCode = fields["StatusCode"] as string;

        if (string.Equals(statusCode, GlacierUtils.JOB_STATUS_SUCCEEDED,
StringComparison.InvariantCultureIgnoreCase))
```

```
        {
          Console.WriteLine("Downloading job output");
          DownloadOutput(jobId, client); // Save job output to the specified file
location.
        }
        else if (string.Equals(statusCode, GlacierUtils.JOB_STATUS_FAILED,
StringComparison.InvariantCultureIgnoreCase))
          Console.WriteLine("Job failed... cannot download the archive.");

        jobDone = true;
        sqsClient.DeleteMessage(new DeleteMessageRequest() { QueueUrl = queueUrl,
ReceiptHandle = message.ReceiptHandle });
      }
    }

    private static void DownloadOutput(string jobId, AmazonGlacierClient client)
    {
      GetJobOutputRequest getJobOutputRequest = new GetJobOutputRequest()
      {
        JobId = jobId,
        VaultName = vaultName
      };

      GetJobOutputResponse getJobOutputResponse =
client.GetJobOutput(getJobOutputRequest);
      using (Stream webStream = getJobOutputResponse.Body)
      {
          using (Stream fileToSave = File.OpenWrite(fileName))
          {
              CopyStream(webStream, fileToSave);
          }
      }
    }

    public static void CopyStream(Stream input, Stream output)
    {
      byte[] buffer = new byte[65536];
      int length;
      while ((length = input.Read(buffer, 0, buffer.Length)) > 0)
      {
        output.Write(buffer, 0, length);
      }
    }
  }
```

```
}
```

예제 2:의 하위 수준 API를 사용하여 아카이브 검색 AWS SDK for .NET- 청크 단위로 출력 다운로드

다음은 Amazon Glacier에서 아카이브를 검색하는 C# 코드 예입니다. 이 코드 예제는 `GetJobOutputRequest` 객체에서 바이트 범위를 지정하여 작업 출력을 청크 단위로 다운로드합니다.

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading;
using Amazon.Glacier;
using Amazon.Glacier.Model;
using Amazon.Glacier.Transfer;
using Amazon.Runtime;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;
using Amazon.SQS;
using Amazon.SQS.Model;
using Newtonsoft.Json;
using System.Collections.Specialized;

namespace glacier.amazon.com.docsamples
{
  class ArchiveDownloadLowLevelUsingSQLSNSOutputUsingRange
  {
    static string topicArn;
    static string queueUrl;
    static string queueArn;
    static string vaultName = "*** Provide vault name ***";
    static string archiveId = "*** Provide archive ID ***";
    static string fileName  = "*** Provide the file name and path to where to store
 downloaded archive ***";
    static AmazonSimpleNotificationServiceClient snsClient;
    static AmazonSQSClient sqsClient;
    const string SQS_POLICY =
        "{" +
        "    \"Version\" : \"2012-10-17\",&TCX5-2025-waiver;" +
        "    \"Statement\" : [" +
        "        {" +
        "            \"Sid\" : \"sns-rule\"," +
        "            \"Effect\" : \"Allow\"," +
```

```
        "            \"Principal\" : {\"AWS\" : \"arn:aws:iam::123456789012:root\" },"
+
        "            \"Action\"  : \"sqs:SendMessage\"," +
        "            \"Resource\"  : \"{QuernArn}\"," +
        "            \"Condition\" : {" +
        "                \"ArnLike\" : {" +
        "                    \"aws:SourceArn\" : \"{TopicArn}\"" +
        "                }" +
        "            }" +
        "        }" +
        "    ]" +
        "}";

    public static void Main(string[] args)
    {
      AmazonGlacierClient client;

      try
      {
          using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2))
          {
              Console.WriteLine("Setup SNS topic and SQS queue.");
              SetupTopicAndQueue();
              Console.WriteLine("To continue, press Enter"); Console.ReadKey();

              Console.WriteLine("Download archive");
              DownloadAnArchive(archiveId, client);
        }
        Console.WriteLine("Operations successful. To continue, press Enter");
        Console.ReadKey();
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      finally
      {
        // Delete SNS topic and SQS queue.
        snsClient.DeleteTopic(new DeleteTopicRequest() { TopicArn = topicArn });
        sqsClient.DeleteQueue(new DeleteQueueRequest() { QueueUrl = queueUrl });
      }
    }

       static void SetupTopicAndQueue()
    {
```

```
      long ticks = DateTime.Now.Ticks;

      // Setup SNS topic.
      snsClient = new
AmazonSimpleNotificationServiceClient(Amazon.RegionEndpoint.USWest2);
      sqsClient = new AmazonSQSClient(Amazon.RegionEndpoint.USWest2);

      topicArn = snsClient.CreateTopic(new CreateTopicRequest { Name =
"GlacierDownload-" + ticks }).TopicArn;
      Console.Write("topicArn: "); Console.WriteLine(topicArn);

      CreateQueueRequest createQueueRequest = new CreateQueueRequest();
      createQueueRequest.QueueName = "GlacierDownload-" + ticks;
      CreateQueueResponse createQueueResponse =
sqsClient.CreateQueue(createQueueRequest);
      queueUrl = createQueueResponse.QueueUrl;
      Console.Write("QueueURL: "); Console.WriteLine(queueUrl);

      GetQueueAttributesRequest getQueueAttributesRequest = new
GetQueueAttributesRequest();
      getQueueAttributesRequest.AttributeNames = new List<string> { "QueueArn" };
      getQueueAttributesRequest.QueueUrl = queueUrl;
      GetQueueAttributesResponse response =
sqsClient.GetQueueAttributes(getQueueAttributesRequest);
      queueArn = response.QueueARN;
      Console.Write("QueueArn: "); Console.WriteLine(queueArn);

      // Setup the Amazon SNS topic to publish to the SQS queue.
      snsClient.Subscribe(new SubscribeRequest()
      {
        Protocol = "sqs",
        Endpoint = queueArn,
        TopicArn = topicArn
      });

      // Add the policy to the queue so SNS can send messages to the queue.
      var policy = SQS_POLICY.Replace("{TopicArn}", topicArn).Replace("{QuernArn}",
queueArn);

      sqsClient.SetQueueAttributes(new SetQueueAttributesRequest()
      {
          QueueUrl = queueUrl,
          Attributes = new Dictionary<string, string>
          {
```

```
            { QueueAttributeName.Policy, policy }
        }
    });
  }

  static void DownloadAnArchive(string archiveId, AmazonGlacierClient client)
  {
    // Initiate job.
    InitiateJobRequest initJobRequest = new InitiateJobRequest()
    {

      VaultName = vaultName,
      JobParameters = new JobParameters()
      {
        Type = "archive-retrieval",
        ArchiveId = archiveId,
        Description = "This job is to download the archive.",
        SNSTopic = topicArn,
      }
    };
    InitiateJobResponse initJobResponse = client.InitiateJob(initJobRequest);
    string jobId = initJobResponse.JobId;

    // Check queue for a message and if job completed successfully, download archive.
    ProcessQueue(jobId, client);
  }

  private static void ProcessQueue(string jobId, AmazonGlacierClient client)
  {
      var receiveMessageRequest = new ReceiveMessageRequest() { QueueUrl = queueUrl,
MaxNumberOfMessages = 1 };
      bool jobDone = false;
      while (!jobDone)
      {
          Console.WriteLine("Poll SQS queue");
          ReceiveMessageResponse receiveMessageResponse =
sqsClient.ReceiveMessage(receiveMessageRequest);
          if (receiveMessageResponse.Messages.Count == 0)
          {
              Thread.Sleep(10000 * 60);
              continue;
          }
          Console.WriteLine("Got message");
          Message message = receiveMessageResponse.Messages[0];
```

```
            Dictionary<string, string> outerLayer =
JsonConvert.DeserializeObject<Dictionary<string, string>>(message.Body);
            Dictionary<string, object> fields =
JsonConvert.DeserializeObject<Dictionary<string, object>>(outerLayer["Message"]);
            string statusCode = fields["StatusCode"] as string;
            if (string.Equals(statusCode, GlacierUtils.JOB_STATUS_SUCCEEDED,
StringComparison.InvariantCultureIgnoreCase))
            {
                long archiveSize = Convert.ToInt64(fields["ArchiveSizeInBytes"]);
                Console.WriteLine("Downloading job output");
                DownloadOutput(jobId, archiveSize, client); // This where we save job
output to the specified file location.
            }
            else if (string.Equals(statusCode, GlacierUtils.JOB_STATUS_FAILED,
StringComparison.InvariantCultureIgnoreCase))
                Console.WriteLine("Job failed... cannot download the archive.");
            jobDone = true;
            sqsClient.DeleteMessage(new DeleteMessageRequest() { QueueUrl = queueUrl,
ReceiptHandle = message.ReceiptHandle });
        }
    }

    private static void DownloadOutput(string jobId, long archiveSize,
AmazonGlacierClient client)
    {
      long partSize = 4 * (long)Math.Pow(2, 20);  // 4 MB.
      using (Stream fileToSave = new FileStream(fileName, FileMode.Create,
FileAccess.Write))
      {

        long currentPosition = 0;
        do
        {
          GetJobOutputRequest getJobOutputRequest = new GetJobOutputRequest()
          {
            JobId = jobId,
            VaultName = vaultName
          };

          long endPosition = currentPosition + partSize - 1;
          if (endPosition > archiveSize)
            endPosition = archiveSize;

          getJobOutputRequest.SetRange(currentPosition, endPosition);
```

```
          GetJobOutputResponse getJobOutputResponse =
  client.GetJobOutput(getJobOutputRequest);

          using (Stream webStream = getJobOutputResponse.Body)
          {
            CopyStream(webStream, fileToSave);
          }
          currentPosition += partSize;
        } while (currentPosition < archiveSize);
      }
    }

    public static void CopyStream(Stream input, Stream output)
    {
      byte[] buffer = new byte[65536];
      int length;
      while ((length = input.Read(buffer, 0, buffer.Length)) > 0)
      {
        output.Write(buffer, 0, length);
      }
    }
  }
}
```

Python을 통한 병렬 처리를 사용하여 대용량 아카이브 다운로드

이 주제에서는 Python을 통한 병렬 처리를 사용하여 Amazon S3 Glacier(S3 Glacier)에서 대용량 아카이브를 다운로드하는 방법을 설명합니다. 이 접근 방식을 사용하면 아카이브를 독립적으로 처리할 수 있는 작은 조각으로 나누어 원하는 크기의 아카이브를 안정적으로 다운로드할 수 있습니다.

개요

이 예제에 제공된 Python 스크립트는 다음 작업을 수행합니다.

1. 알림에 필요한 AWS 리소스(Amazon SNS 주제 및 Amazon SQS 대기열)를 설정합니다.
2. Amazon Glacier에서 아카이브 가져오기 작업을 시작합니다.
3. Amazon SQS 대기열에서 작업 완료 알림을 모니터링합니다.
4. 대용량 아카이브를 관리 가능한 청크로 분할합니다.
5. 여러 작업자 스레드를 사용하여 청크를 병렬로 다운로드합니다.
6. 나중에 다시 조립할 수 있도록 각 청크를 디스크에 저장합니다.

사전 조건

시작하기 전에, 다음 사항을 확인해야 합니다.

- Python 3.6 이상 설치
- AWS SDK for Python(Boto3) 설치됨
- AWS Amazon Glacier, Amazon SNS 및 Amazon SQS에 대한 적절한 권한으로 구성된 자격 증명
- 다운로드한 아카이브 청크를 저장하기에 충분한 디스크 공간

예: Python을 통한 병렬 처리를 사용하여 아카이브 다운로드

다음 Python 스크립트는 병렬 처리를 사용하여 Amazon Glacier에서 대용량 아카이브를 다운로드하는 방법을 보여줍니다.

```
import boto3
import time
import json
import jmespath
import re
import concurrent.futures
import os

output_file_path = "output_directory_path"
vault_name = "vault_name"

chunk_size = 1000000000 #1gb - size of chunks for parallel download.
notify_queue_name = 'GlacierJobCompleteNotifyQueue' # SQS queue for Glacier recall
 notification
chunk_download_queue_name='GlacierChunkReadyNotifyQueue' # SQS queue for chunks
sns_topic_name = 'GlacierRecallJobCompleted' # the SNS topic to be notified when
 Glacier archive is restored.
chunk_queue_visibility_timeout = 7200 # 2 hours - this may need to be adjusted.
region = 'us-east-1'
archive_id = "archive_id_to_restore"
retrieve_archive = True # set to false if you do not want to restore from Glacier -
 useful for restarting or parallel processing of the chunk queue.
workers = 12 # the number of parallel worker threads for downloading chunks.

def setup_queues_and_topic():
    sqs = boto3.client('sqs')
```

```
    sns = boto3.client('sns')

    # Create the SNS topic
    topic_response = sns.create_topic(
        Name=sns_topic_name
    )
    topic_arn = topic_response['TopicArn']
    print("Creating the SNS topic " + topic_arn)

    # Create the notification queue
    notify_queue_response = sqs.create_queue(
        QueueName=notify_queue_name,
        Attributes={
            'VisibilityTimeout': '300',  # 5 minutes
            'ReceiveMessageWaitTimeSeconds': '20'  # Enable long polling
        }
    )
    notify_queue_url = notify_queue_response['QueueUrl']
    print("Creating the archive-retrieval notification queue " + notify_queue_url)

    # Create the chunk download queue
    chunk_queue_response = sqs.create_queue(
        QueueName=chunk_download_queue_name,
        Attributes={
            'VisibilityTimeout': str(chunk_queue_visibility_timeout),  # 5 minutes
            'ReceiveMessageWaitTimeSeconds': '0'
        }
    )
    chunk_queue_url = chunk_queue_response['QueueUrl']

    print("Creating the chunk ready notification queue " + chunk_queue_url)


   # Get the ARN for the notification queue
    notify_queue_attributes = sqs.get_queue_attributes(
        QueueUrl=notify_queue_url,
        AttributeNames=['QueueArn']
    )
    notify_queue_arn = notify_queue_attributes['Attributes']['QueueArn']

    # Set up the SNS topic policy on the notification queue
    queue_policy = {
        "Version": "2012-10-17",
        "Statement": [{
```

```
            "Sid": "allow-sns-messages",
            "Effect": "Allow",
            "Principal": {"AWS": "*"},
            "Action": "SQS:SendMessage",
            "Resource": notify_queue_arn,
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": topic_arn
                }
            }
        }]
    }

    # Set the queue policy
    sqs.set_queue_attributes(
        QueueUrl=notify_queue_url,
        Attributes={
            'Policy': json.dumps(queue_policy)
        }
    )

    # Subscribe the notification queue to the SNS topic
    sns.subscribe(
        TopicArn=topic_arn,
        Protocol='sqs',
        Endpoint=notify_queue_arn
    )

    return {
        'topic_arn': topic_arn,
        'notify_queue_url': notify_queue_url,
        'chunk_queue_url': chunk_queue_url
    }


def split_and_send_chunks(archive_size, job_id,chunk_queue_url):
    ranges = []
    current = 0
    chunk_number = 0

    while current < archive_size:
        chunk_number += 1
        next_range = min(current + chunk_size - 1, archive_size - 1)
        ranges.append((current, next_range, chunk_number))
```

```
        current = next_range + 1

    # Send messages to SQS queue
    for start, end, chunk_number in ranges:
        body = {"start": start, "end": end, "job_id": job_id, "chunk_number":
 chunk_number}
        body = json.dumps(body)
        print("Sending SQS message for range:" + str(body))
        response = sqs.send_message(
            QueueUrl=chunk_queue_url,
            MessageBody=str(body)
        )

def GetJobOutputChunks(job_id, byterange, chunk_number):
    glacier = boto3.client('glacier')
    response = glacier.get_job_output(
        vaultName=vault_name,
        jobId=job_id,
        range=byterange,

    )

    with open(os.path.join(output_file_path,str(chunk_number)+".chunk"), 'wb') as
 output_file:
        output_file.write(response['body'].read())

    return response

def ReceiveArchiveReadyMessages(notify_queue_url,chunk_queue_url):

    response = sqs.receive_message(
        QueueUrl=notify_queue_url,
        AttributeNames=['All'],
        MaxNumberOfMessages=1,
        WaitTimeSeconds=20,
        MessageAttributeNames=['Message']
    )
    print("Polling archive retrieval job ready queue...")
    # Checking that there is a Messages key before proceeding. No 'Messages' key likely
 means the queue is empty

    if 'Messages' in response:
        print("Received a message from the archive retrieval job queue")
        jsonresponse = response
```

```
        # Loading the string into JSON and checking that ArchiveSizeInBytes key is
 present before continuing.
        jsonresponse=json.loads(jsonresponse['Messages'][0]['Body'])
        jsonresponse=json.loads(jsonresponse['Message'])
        if 'ArchiveSizeInBytes' in jsonresponse:
            receipt_handle = response['Messages'][0]['ReceiptHandle']
            if jsonresponse['ArchiveSizeInBytes']:
                archive_size = jsonresponse['ArchiveSizeInBytes']

                print(f'Received message: {response}')
                if archive_size > chunk_size:
                    split_and_send_chunks(archive_size,
 jsonresponse['JobId'],chunk_queue_url)

                    sqs.delete_message(
                    QueueUrl=notify_queue_url,
                    ReceiptHandle=receipt_handle)

            else:
                print("No ArchiveSizeInBytes value found in message")
                print(response)

    else:
        print('No messages available in the queue at this time.')

    time.sleep(1)

def ReceiveArchiveChunkMessages(chunk_queue_url):
    response = sqs.receive_message(
        QueueUrl=chunk_queue_url,
        AttributeNames=['All'],
        MaxNumberOfMessages=1,
        WaitTimeSeconds=0,
        MessageAttributeNames=['Message']
    )
    print("Polling archive chunk queue...")
    print(response)
    # Checking that there is a Messages key before proceeding. No 'Messages' key likely
 means the queue is empty
    if 'Messages' in response:
        jsonresponse = response
        # Loading the string into JSON and checking that ArchiveSizeInBytes key is
 present before continuing.
        jsonresponse=json.loads(jsonresponse['Messages'][0]['Body'])
```

```
        if 'job_id' in jsonresponse: #checking that there is a job id before continuing
            job_id = jsonresponse['job_id']
            byterange = "bytes="+str(jsonresponse['start']) + '-' +
 str(jsonresponse['end'])
            chunk_number = jsonresponse['chunk_number']
            receipt_handle = response['Messages'][0]['ReceiptHandle']
            if jsonresponse['job_id']:
                print(f'Received message: {response}')
                GetJobOutputChunks(job_id,byterange,chunk_number)
                sqs.delete_message(
                QueueUrl=chunk_queue_url,
                ReceiptHandle=receipt_handle)
    else:
        print('No messages available in the chunk queue at this time.')

def initiate_archive_retrieval(archive_id, topic_arn):
    glacier = boto3.client('glacier')

    job_parameters = {
        "Type": "archive-retrieval",
        "ArchiveId": archive_id,
        "Description": "Archive retrieval job",
        "SNSTopic": topic_arn,
        "Tier": "Bulk"  # You can change this to "Standard" or "Expedited" based on
 your needs
    }

    try:
        response = glacier.initiate_job(
            vaultName=vault_name,
            jobParameters=job_parameters
        )

        print("Archive retrieval job initiated:")
        print(f"Job ID: {response['jobId']}")
        print(f"Job parameters: {job_parameters}")
        print(f"Complete response: {json.dumps(response, indent=2)}")

        return response['jobId']

    except Exception as e:
        print(f"Error initiating archive retrieval job: {str(e)}")
        raise
```

```
def run_async_tasks(chunk_queue_url, workers):
    max_workers = workers  # Set the desired maximum number of concurrent tasks
    with concurrent.futures.ThreadPoolExecutor(max_workers=max_workers) as executor:
        for _ in range(max_workers):
            executor.submit(ReceiveArchiveChunkMessages, chunk_queue_url)

# One time setup of the necessary queues and topics.
queue_and_topic_atts = setup_queues_and_topic()

topic_arn = queue_and_topic_atts['topic_arn']
notify_queue_url = queue_and_topic_atts['notify_queue_url']
chunk_queue_url = queue_and_topic_atts['chunk_queue_url']

if retrieve_archive:
    print("Retrieving the defined archive... The topic arn we will notify when
 recalling the archive is: "+topic_arn)
    job_id = initiate_archive_retrieval(archive_id, topic_arn)
else:
    print("Retrieve archive is false, polling queues and downloading only.")

while True:
   ReceiveArchiveReadyMessages(notify_queue_url,chunk_queue_url)
   run_async_tasks(chunk_queue_url,workers)
```

스크립트 사용

이 스크립트를 사용하려면 다음 단계를 따르세요.

1. 스크립트의 자리 표시자 값을 특정 정보로 바꿉니다.
   - *output_file_path*: 청크 파일이 저장될 디렉터리
   - *vault_name*: S3 Glacier 볼트의 이름
   - *notify_queue_name*: 작업 알림 대기열의 이름
   - *chunk_download_queue_name*: 청크 다운로드 대기열의 이름
   - *sns_topic_name*: SNS 토픽의 이름
   - 볼트가 위치한 *region*: AWS region
   - *archive_id*: 검색할 아카이브의 ID
2. 스크립트를 실행합니다.

```
python download_large_archive.py
```

3. 모든 청크를 다운로드한 후 다음과 같은 명령을 사용하여 청크를 단일 파일로 결합할 수 있습니다.

```
cat /path/to/chunks/*.chunk > complete_archive.file
```

중요 고려 사항

이 스크립트를 사용할 때는 다음 사항에 유의해야 합니다.

- 선택한 검색 계층에 따라 S3 Glacier에서 아카이브 검색을 완료하는 데 몇 시간이 걸릴 수 있습니다.
- 스크립트는 무기한 실행되어 대기열을 지속적으로 폴링합니다. 특정 요구 사항에 따라 종료 조건을 추가할 수 있습니다.
- 아카이브의 모든 청크를 저장할 디스크 공간이 충분한지 확인합니다.
- 스크립트가 중단되면 `retrieve_archive=False`를 사용해 다시 시작하여 새 검색 작업을 시작하지 않고 청크를 계속 다운로드할 수 있습니다.
- 네트워크 대역폭 및 시스템 리소스에 따라 *`chunk_size`* 및 *`workers`* 파라미터를 조정합니다.
- Amazon S3 검색, Amazon SNS 및 Amazon SQS 사용량에는 표준 AWS 요금이 적용됩니다.

REST API를 사용하여 아카이브 다운로드

REST API를 사용하여 아카이브 다운로드

아카이브를 다운로드하는 작업은 2단계 프로세스로 구성됩니다.

1. `archive-retrieval` 유형의 작업을 시작합니다. 자세한 내용은 작업 시작(POST jobs) 단원을 참조하십시오.
2. 작업이 완료된 후 아카이브 데이터를 다운로드합니다. 자세한 내용은 작업 출력 가져오기(GET output) 단원을 참조하십시오.

를 사용하여 Amazon Glacier에서 아카이브 다운로드 AWS CLI

AWS Command Line Interface ()를 사용하여 Amazon Glacier(Amazon Glacier)에서 아카이브를 다운로드할 수 있습니다AWS CLI.

주제

- (사전 조건) 설정 AWS CLI
- 예:를 사용하여 아카이브 다운로드 AWS CLI

(사전 조건) 설정 AWS CLI

1. AWS CLI를 다운로드하고 구성합니다. 관련 지침은 AWS Command Line Interface 사용 설명서에서 다음 토픽을 참조하세요.

   설치 AWS Command Line Interface

   구성 AWS Command Line Interface
2. 명령 프롬프트에 다음 명령을 입력하여 AWS CLI 설정을 확인합니다. 이러한 명령은 명시적으로 자격 증명을 제공하지 않으므로 기본 프로파일의 자격 증명이 사용됩니다.

   - help 명령을 사용해 보십시오.

     ```
     aws help
     ```

   - 구성된 계정의 Amazon Glacier 볼트 목록을 가져오려면 `list-vaults` 명령을 사용합니다. *123456789012*을 AWS 계정 ID로 바꿉니다.

     ```
     aws glacier list-vaults --account-id 123456789012
     ```

   - 에 대한 현재 구성 데이터를 보려면 `aws configure list` 명령을 AWS CLI사용합니다.

     ```
     aws configure list
     ```

예:를 사용하여 아카이브 다운로드 AWS CLI

> **Note**
>
> 아카이브를 다운로드하려면 반드시 아카이브 ID를 알아야 합니다. 1~4단계를 따라 아카이브 ID를 검색합니다. 다운로드하려는 아카이브 ID를 이미 알고 있는 경우 5단계로 건너뛰세요.

1. `initiate-job` 명령을 사용하여 인벤토리 검색 작업을 시작합니다. 인벤토리 보고서에는 사용자의 아카이브 ID가 나열됩니다.

```
aws glacier initiate-job --vault-name awsexamplevault --account-id 111122223333 --
job-parameters="{\"Type\":\"inventory-retrieval\"}"
```

예상 결과:

```
{
    "location": "/111122223333/vaults/awsexamplevault/jobs/*** jobid ***",
    "jobId": "*** jobid ***"
}
```

2. `describe-job` 명령을 사용하여 이전 작업의 명령의 상태를 확인합니다.

```
aws glacier describe-job --vault-name awsexamplevault --account-id 111122223333 --
job-id *** jobid ***
```

예상 결과:

```
{
    "InventoryRetrievalParameters": {
        "Format": "JSON"
    },
    "VaultARN": "*** vault arn ***",
    "Completed": false,
    "JobId": "*** jobid ***",
    "Action": "InventoryRetrieval",
    "CreationDate": "*** job creation date ***",
    "StatusCode": "InProgress"
}
```

3. 작업이 완료될 때까지 기다립니다.

   작업 출력을 다운로드할 수 있을 때까지 기다려야 합니다. 볼트에서 알림 구성을 설정하거나 작업을 시작할 때 Amazon Simple Notification Service(Amazon SNS) 토픽을 지정했다면 Amazon Glacier가 작업 완료 후 해당 토픽에 메시지를 보냅니다.

볼트의 특정 이벤트에 대해 알림 구성을 설정할 수 있습니다. 자세한 내용은 Amazon Glacier의 볼트 알림 구성 단원을 참조하십시오. Amazon Glacier는 특정 이벤트가 발생할 때마다 지정된 SNS 토픽에 메시지를 보냅니다.

4. 완료되면 `get-job-output` 명령을 사용하여 검색 작업을 `output.json` 파일로 다운로드합니다. 이 파일에는 사용자의 아카이브 ID가 포함될 것입니다.

```
aws glacier get-job-output --vault-name awsexamplevault --account-id 111122223333
 --job-id *** jobid *** output.json
```

이 명령은 다음 필드가 있는 파일을 생성합니다.

```
{
"VaultARN":"arn:aws:glacier:region:111122223333:vaults/awsexamplevault",
"InventoryDate":"*** job completion date ***",
"ArchiveList":[
{"ArchiveId":"*** archiveid ***",
"ArchiveDescription":*** archive description (if set) ***,
"CreationDate":"*** archive creation date ***",
"Size":"*** archive size (in bytes) ***",
"SHA256TreeHash":"*** archive hash ***"
}
{"ArchiveId":
...
]}
```

5. `initiate-job` 명령을 사용하여 볼트에서 각 아카이브를 검색하는 프로세스를 시작합니다. 아래 `archive-retrieval`과 같이 작업 파라미터를 지정해야 합니다.

```
aws glacier initiate-job --vault-name awsexamplevault --account-id 111122223333
 --job-parameters="{\"Type\":\"archive-retrieval\",\"ArchiveId\":\"*** archiveId
 ***\"}"
```

6. `archive-retrieval` 작업이 완료될 때까지 기다립니다. `describe-job` 명령을 사용하여 이전 명령의 상태를 확인합니다.

```
aws glacier describe-job --vault-name awsexamplevault --account-id 111122223333 --
job-id *** jobid ***
```

7. 위 작업이 완료되면 `get-job-output` 명령을 사용하여 아카이브를 다운로드합니다.

```
aws glacier get-job-output --vault-name awsexamplevault --account-id 111122223333
 --job-id *** jobid *** output_file_name
```

Amazon Glacier에서 아카이브 삭제

Amazon Glacier(Amazon Glacier) 관리 콘솔을 사용하여 아카이브를 삭제할 수 없습니다. 아카이브를 삭제하려면 AWS Command Line Interface (CLI) 또는 쓰기 코드를 사용하여 REST API 직접 또는 AWS SDK for Java 및 .NET 래퍼 라이브러리를 사용하여 삭제 요청을 해야 합니다. 다음 주제에서는 AWS SDK for Java 및 .NET 래퍼 라이브러리, REST API 및를 사용하는 방법을 설명합니다 AWS CLI.

주제

- AWS SDK for Java를 사용하여 Amazon Glacier에서 아카이브 삭제
- 를 사용하여 Amazon Glacier에서 아카이브 삭제 AWS SDK for .NET
- REST API를 사용하여 Amazon Glacier 아카이브 삭제
- 를 사용하여 Amazon Glacier에서 아카이브 삭제 AWS Command Line Interface

아카이브는 볼트에서 한 번에 하나씩 삭제할 수 있습니다. 아카이브를 삭제하려면 삭제 요청 시 아카이브 ID를 입력해야 합니다. 아카이브 ID는 해당 아카이브가 포함된 볼트에서 볼트 인벤토리를 다운로드하여 가져올 수 있습니다. 볼트 인벤토리 다운로드에 대한 자세한 내용은 Amazon Glacier에서 볼트 인벤토리 다운로드 섹션을 참조하세요.

아카이브를 삭제한 후에도 삭제한 아카이브를 가져오는 작업을 시작할 수 있도록 요청하는 것은 가능하지만 아카이브 가져오기 작업이 실행되지는 않습니다.

아카이브를 삭제할 때 아카이브 ID에 대해 진행 중인 아카이브 가져오기는 다음 시나리오에 따라 성공할 수도, 혹은 성공하지 않을 수도 있습니다.

- Amazon Glacier가 아카이브 삭제 요청을 수신할 때, 아카이브 가져오기 작업이 활성화되어 데이터 다운로드를 준비하고 있다면 아카이브 가져오기 작업이 중단될 수도 있습니다.
- Amazon Glacier가 아카이브 삭제 요청을 수신할 때 아카이브 가져오기 작업이 아카이브 다운로드를 성공적으로 준비했다면 출력을 다운로드할 수 있습니다.

아카이브 가져오기에 대한 자세한 내용은 Amazon Glacier에서 아카이브 다운로드 섹션을 참조하세요.

이 작업은 멱등성을 갖습니다. 이미 삭제된 아카이브를 삭제하려고 해도 오류가 발생하지는 않습니다.

아카이브를 삭제한 직후에 볼트 인벤토리를 다운로드할 경우 삭제된 아카이브가 목록에 포함될 수도 있습니다. Amazon Glacier는 하루에 한 번만 볼트 인벤토리를 준비하기 때문입니다.

> **Note**
>
> 볼트 아카이브의 자동 삭제는 Amazon S3 Glacier에서 볼트 아카이브의 자동 삭제를 참조하세요.

AWS SDK for Java를 사용하여 Amazon Glacier에서 아카이브 삭제

다음은 AWS SDK for Java의 로우레벨 API를 사용해 아카이브를 삭제하는 단계입니다.

1. `AmazonGlacierClient` 클래스(클라이언트)의 인스턴스를 만듭니다.

   삭제를 원하는 아카이브가 저장된 AWS 리전을 지정합니다. 이 클라이언트를 사용하여 실행하는 모든 작업이 해당 AWS 리전에 적용됩니다.
2. `DeleteArchiveRequest` 클래스 인스턴스를 생성하여 요청 정보를 입력합니다.

   아카이브 ID와 볼트 이름, 그리고 계정 ID를 입력해야 합니다. 계정 ID를 입력하지 않는 경우에는 요청 서명을 위해 입력하는 자격 증명과 연결되어 있는 계정 ID로 간주합니다. 자세한 내용은 Amazon Glacier AWS SDK for Java 에서 사용 섹션을 참조하세요.
3. 요청 객체를 파라미터로 입력하여 `deleteArchive` 메서드를 실행합니다.

다음은 위에서 설명한 단계를 나타내는 Java 코드 조각입니다.

```
AmazonGlacierClient client;

DeleteArchiveRequest request = new DeleteArchiveRequest()
    .withVaultName("*** provide a vault name ***")
    .withArchiveId("*** provide an archive ID ***");

client.deleteArchive(request);
```

> **Note**
>
> 기본 REST API에 대한 자세한 내용은 아카이브 삭제(DELETE archive) 섹션을 참조하세요.

예: AWS SDK for Java를 사용하여 아카이브 삭제

다음은 AWS SDK for Java를 이용해 아카이브를 삭제하는 Java 코드 예제입니다. 이 예제의 실행 방법에 대한 단계별 지침은 Eclipse를 사용하여 Amazon Glacier의 Java 예 실행 섹션을 참조하세요. 아래와 같이 볼트 이름과 삭제를 원하는 아카이브의 아카이브 ID를 사용해 코드를 업데이트해야 합니다.

Example

```
import java.io.IOException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.model.DeleteArchiveRequest;

public class ArchiveDelete {

    public static String vaultName = "*** provide vault name ****";
    public static String archiveId = "*** provide archive ID***";
    public static AmazonGlacierClient client;

    public static void main(String[] args) throws IOException {

     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        client = new AmazonGlacierClient(credentials);
        client.setEndpoint("https://glacier.us-east-1.amazonaws.com/");

        try {

            // Delete the archive.
            client.deleteArchive(new DeleteArchiveRequest()
                .withVaultName(vaultName)
                .withArchiveId(archiveId));

            System.out.println("Deleted archive successfully.");

        } catch (Exception e) {
            System.err.println("Archive not deleted.");
            System.err.println(e);
        }
    }
}
```

를 사용하여 Amazon Glacier에서 아카이브 삭제 AWS SDK for .NET

.NET용 Amazon SDK에서 제공하는 하이레벨 및 로우레벨 API는 모두 아카이브를 삭제하는 방법을 제공합니다.

주제

- 의 상위 수준 API를 사용하여 아카이브 삭제 AWS SDK for .NET
- 하위 수준 API를 사용하여 아카이브 삭제 AWS SDK for .NET

의 상위 수준 API를 사용하여 아카이브 삭제 AWS SDK for .NET

하이레벨 API의 `ArchiveTransferManager` 클래스는 아카이브를 삭제하는 데 사용할 수 있는 `DeleteArchive` 메서드를 제공합니다.

예:의 상위 수준 API를 사용하여 아카이브 삭제 AWS SDK for .NET

다음 C# 코드 예제에서는의 상위 수준 API AWS SDK for .NET 를 사용하여 아카이브를 삭제합니다. 이 예제의 실행 방법에 대한 단계별 지침은 코드 예제 실행 섹션을 참조하세요. 아래와 같이 삭제를 원하는 아카이브의 아카이브 ID를 사용해 코드를 업데이트합니다.

Example

```
using System;
using Amazon.Glacier;
using Amazon.Glacier.Transfer;
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
  class ArchiveDeleteHighLevel
  {
    static string vaultName = "examplevault";
    static string archiveId = "*** Provide archive ID ***";

    public static void Main(string[] args)
    {
      try
      {
        var manager = new ArchiveTransferManager(Amazon.RegionEndpoint.USWest2);
        manager.DeleteArchive(vaultName, archiveId);
```

```
        Console.ReadKey();
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }
  }
}
```

하위 수준 API를 사용하여 아카이브 삭제 AWS SDK for .NET

다음은 AWS SDK for .NET를 사용하여 삭제하는 단계입니다.

1. `AmazonGlacierClient` 클래스(클라이언트)의 인스턴스를 만듭니다.

   삭제하려는 아카이브가 저장되는 AWS 리전을 지정해야 합니다. 이 클라이언트를 사용하여 수행하는 모든 작업은 해당 AWS 리전에 적용됩니다.

2. `DeleteArchiveRequest` 클래스 인스턴스를 생성하여 요청 정보를 입력합니다.

   아카이브 ID와 볼트 이름, 그리고 계정 ID를 입력해야 합니다. 계정 ID를 입력하지 않는 경우에는 요청 서명을 위해 입력하는 자격 증명과 연결되어 있는 계정 ID로 간주합니다. 자세한 내용은 Amazon Glacier에서 AWS SDKs 사용 단원을 참조하십시오.

3. 요청 객체를 파라미터로 입력하여 `DeleteArchive` 메서드를 실행합니다.

예:의 하위 수준 API를 사용하여 아카이브 삭제 AWS SDK for .NET

다음은 앞선 단계를 설명하는 C# 예제입니다. 이 예제에서는의 하위 수준 API AWS SDK for .NET 를 사용하여 아카이브를 삭제합니다.

> **Note**
>
> 기본 REST API에 대한 자세한 내용은 아카이브 삭제(DELETE archive) 섹션을 참조하세요.

이 예제의 실행 방법에 대한 단계별 지침은 코드 예제 실행 섹션을 참조하세요. 아래와 같이 삭제를 원하는 아카이브의 아카이브 ID를 사용해 코드를 업데이트합니다.

Example

```
using System;
using Amazon.Glacier;
using Amazon.Glacier.Model;
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
  class ArchiveDeleteLowLevel
  {
    static string vaultName = "examplevault";
    static string archiveId = "*** Provide archive ID ***";

    public static void Main(string[] args)
    {
      AmazonGlacierClient client;
      try
      {
        using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2))
        {
          Console.WriteLine("Deleting the archive");
          DeleteAnArchive(client);
        }
        Console.WriteLine("Operations successful. To continue, press Enter");
        Console.ReadKey();
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }

    static void DeleteAnArchive(AmazonGlacierClient client)
    {
      DeleteArchiveRequest request = new DeleteArchiveRequest()
      {
        VaultName = vaultName,
        ArchiveId = archiveId
      };
      DeleteArchiveResponse response = client.DeleteArchive(request);
    }
  }
```

```
}
```

REST API를 사용하여 Amazon Glacier 아카이브 삭제

아카이브는 아카이브 삭제 API를 사용하여 삭제할 수 있습니다.

- 아카이브 삭제 API에 대한 자세한 내용은 아카이브 삭제(DELETE archive) 섹션을 참조하세요.
- REST API 사용에 대한 자세한 내용은 Amazon Glacier를 위한 API 참조 섹션을 참조하세요.

를 사용하여 Amazon Glacier에서 아카이브 삭제 AWS Command Line Interface

AWS Command Line Interface ()를 사용하여 Amazon Glacier(Amazon Glacier)에서 아카이브를 삭제할 수 있습니다AWS CLI.

주제

- (사전 조건) 설정 AWS CLI
- 예:를 사용하여 아카이브 삭제 AWS CLI

(사전 조건) 설정 AWS CLI

1. AWS CLI를 다운로드하고 구성합니다. 관련 지침은 AWS Command Line Interface 사용 설명서에서 다음 토픽을 참조하세요.

   설치 AWS Command Line Interface

   구성 AWS Command Line Interface
2. 명령 프롬프트에 다음 명령을 입력하여 AWS CLI 설정을 확인합니다. 이러한 명령은 명시적으로 자격 증명을 제공하지 않으므로 기본 프로파일의 자격 증명이 사용됩니다.

   - help 명령을 사용해 보십시오.

     ```
     aws help
     ```

   - 구성된 계정의 Amazon Glacier 볼트 목록을 가져오려면 `list-vaults` 명령을 사용합니다. *123456789012*을 AWS 계정 ID로 바꿉니다.

```
aws glacier list-vaults --account-id 123456789012
```

- 에 대한 현재 구성 데이터를 보려면 `aws configure list` 명령을 AWS CLI사용합니다.

```
aws configure list
```

예:를 사용하여 아카이브 삭제 AWS CLI

1. initiate-job 명령을 사용하여 인벤토리 검색 작업을 시작합니다.

```
aws glacier initiate-job --vault-name awsexamplevault --account-id 111122223333 --
job-parameters="{\"Type\":\"inventory-retrieval\"}"
```

예상 결과:

```
{
    "location": "/111122223333/vaults/awsexamplevault/jobs/*** jobid ***",
    "jobId": "*** jobid ***"
}
```

2. describe-job 명령을 사용하여 이전 검색 작업의 상태를 확인합니다.

```
aws glacier describe-job --vault-name awsexamplevault --account-id 111122223333 --
job-id *** jobid ***
```

예상 결과:

```
{
    "InventoryRetrievalParameters": {
        "Format": "JSON"
    },
    "VaultARN": "*** vault arn ***",
    "Completed": false,
    "JobId": "*** jobid ***",
    "Action": "InventoryRetrieval",
    "CreationDate": "*** job creation date ***",
```

```
    "StatusCode": "InProgress"
}
```

3. 작업이 완료될 때까지 기다립니다.

   작업 출력을 다운로드할 수 있을 때까지 기다려야 합니다. 볼트에서 알림 구성을 설정하거나 작업을 시작할 때 Amazon Simple Notification Service(Amazon SNS) 토픽을 지정했다면 Amazon Glacier가 작업 완료 후 해당 토픽에 메시지를 보냅니다.

   볼트의 특정 이벤트에 대해 알림 구성을 설정할 수 있습니다. 자세한 내용은 Amazon Glacier의 볼트 알림 구성 단원을 참조하십시오. Amazon Glacier는 특정 이벤트가 발생할 때마다 지정된 SNS 토픽에 메시지를 보냅니다.

4. 완료되면 get-job-output 명령을 사용하여 검색 작업을 `output.json` 파일로 다운로드합니다.

```
aws glacier get-job-output --vault-name awsexamplevault --account-id 111122223333
 --job-id *** jobid *** output.json
```

   이 명령은 다음 필드가 있는 파일을 생성합니다.

```
{
"VaultARN":"arn:aws:glacier:region:111122223333:vaults/awsexamplevault",
"InventoryDate":"*** job completion date ***",
"ArchiveList":[
{"ArchiveId":"*** archiveid ***",
"ArchiveDescription":*** archive description (if set) ***,
"CreationDate":"*** archive creation date ***",
"Size":"*** archive size (in bytes) ***",
"SHA256TreeHash":"*** archive hash ***"
}
{"ArchiveId":
...
]}
```

5. `delete-archive` 명령을 사용하여 볼트가 비워질 때까지 볼트에서 각 아카이브를 삭제합니다.

```
aws glacier delete-archive --vault-name awsexamplevault --account-id 111122223333
 --archive-id *** archiveid ***
```

Amazon Glacier에서 AWS SDKs 사용

AWS 는 Amazon Glacier용 애플리케이션을 개발할 수 있는 SDKs를 제공합니다. SDK 라이브러리가 기본적인 Amazon Glacier API를 래핑하기 때문에 프로그래밍 작업이 단순화됩니다. 예를 들어 각 요청을 Amazon Glacier에 전송할 때마다 반드시 요청 인증을 위한 서명을 추가해야 합니다. SDK 라이브러리를 사용하는 경우 코드에 AWS 보안 자격 증명만 제공해야 하며 라이브러리는 필요한 서명을 계산하여 Amazon Glacier로 전송된 요청에 포함합니다. AWS SDKs는 기본 REST API에 매핑되는 라이브러리를 제공하고 요청을 쉽게 구성하고 응답을 처리하는 데 사용할 수 있는 객체를 제공합니다.

주제

- AWS Java 및 .NET용 SDK 라이브러리
- AWS SDK에서 Amazon Glacier 사용
- Amazon Glacier AWS SDK for Java 에서 사용
- Amazon Glacier와 함께 AWS SDK for .NET 사용

AWS Command Line Interface (AWS CLI)는 Amazon Glacier를 AWS 서비스포함하여를 관리하는 통합 도구입니다. 다운로드에 대한 자세한 내용은 단원을 AWS CLI참조하십시오AWS Command Line Interface. Amazon Glacier CLI 명령 목록은 AWS CLI 명령 참조를 참조하세요.

AWS Java 및 .NET용 SDK 라이브러리

Java 및 .NET용 AWS SDKs는 상위 수준 및 하위 수준 래퍼 라이브러리를 제공합니다.

이 개발자 안내서 AWS SDK for .NET 전체에서 AWS SDK for Java 및를 사용하여 Amazon Glacier 작업의 예를 찾을 수 있습니다.

로우레벨 API가 무엇입니까?

로우레벨 래퍼 라이브러리는 Amazon Glacier에서 지원되는 기본 REST API(Amazon Glacier를 위한 API 참조)와 긴밀하게 매핑됩니다. 각 Amazon Glacier REST 작업마다 로우레벨 API가 해당하는 메서드, 요청 정보를 입력할 요청 객체, Amazon Glacier 응답을 처리할 응답 객체를 제공합니다. 로우레벨 래퍼 라이브러리는 기본 Amazon Glacier 작업을 가장 완벽하게 구현한 것입니다.

SDK 라이브러리에 대한 자세한 내용은 Amazon Glacier AWS SDK for Java 에서 사용 및 Amazon Glacier와 함께 AWS SDK for .NET 사용 섹션을 참조하세요.

하이레벨 API가 무엇입니까?

이 라이브러리는 애플리케이션 개발을 더욱 간소화할 목적으로 몇 가지 작업에 하이레벨 추상화를 제공합니다. 예제:

- 아카이브 업로드: 로우레벨 API를 사용하여 아카이브를 업로드하려면 파일 이름과 아카이브를 저장할 볼트 이름 외에도 페이로드 체크섬(SHA-256 트리해시)을 입력해야 합니다. 하지만 하이레벨 API는 자동으로 체크섬을 계산합니다.
- 아카이브 또는 볼트 인벤토리 다운로드: 로우레벨 API를 사용하여 아카이브를 다운로드하려면 먼저 작업을 시작하고 작업이 끝날 때까지 기다린 후에 작업 출력을 가져옵니다. 작업 완료 시 Amazon Glacier에서 알림을 전송하도록 하려면 추가로 코드를 작성하여 Amazon Simple Notification Service(Amazon SNS) 토픽을 설정해야 합니다. 또한 작업 완료 메시지가 토픽에 게시되었는지 확인할 수 있는 폴링 메커니즘도 필요합니다. 하지만 하이레벨 API는 아카이브를 다운로드하는 메서드를 제공하여 이 모든 단계를 관리합니다. 사용자는 아카이브 ID와 다운로드된 데이터를 저장할 폴더 경로만 지정하면 됩니다.

SDK 라이브러리에 대한 자세한 내용은 Amazon Glacier AWS SDK for Java 에서 사용 및 Amazon Glacier와 함께 AWS SDK for .NET 사용 섹션을 참조하세요.

하이레벨 API와 로우레벨 API의 사용 시기

일반적으로 하이레벨 API가 작업을 수행하는 데 필요한 메서드를 제공하는 경우에는 간편성이 뛰어나기 때문에 하이레벨 API를 사용하는 것이 좋습니다. 하지만 하이레벨 API가 필요한 기능을 제공하지 않을 때는 로우레벨 API를 사용할 수 있습니다. 또한 로우레벨 API는 장애 발생 시 재시도 로직 등 작업을 세분화하여 제어할 수 있습니다. 예를 들어 아카이브를 업로드할 때 하이레벨 API는 파일 크기에 따라 아카이브를 단일 작업으로 업로드할지, 아니면 멀티파트 업로드 API를 사용할지 결정합니다. 또한 업로드 장애 시 재시도 로직을 기본적으로 지원합니다. 하지만 로우레벨 API를 사용할 수 있는 경우에는 애플리케이션이 이러한 사용 결정을 세분화하여 제어할 필요도 있습니다.

AWS SDK에서 Amazon Glacier 사용

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

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

Amazon Glacier에 대한 구체적인 예는 AWS SDKs를 사용하는 Amazon Glacier의 코드 예제 섹션을 참조하세요.

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

Amazon Glacier AWS SDK for Java 에서 사용

는에 설명된 대로 Amazon Glacier(Amazon Glacier)에 대한 상위 수준 및 하위 수준 APIs를 모두 AWS SDK for Java 제공합니다Amazon Glacier에서 AWS SDKs 사용. 다운로드에 대한 자세한 내용은 Java 용 Amazon SDK를 AWS SDK for Java참조하세요.

> **Note**
>
> 는 Amazon Glacier에 액세스하기 위한 스레드 세이프 클라이언트를 AWS SDK for Java 제공합니다. 모범 사례로서 애플리케이션에서 클라이언트 하나를 생성한 후 스레드 간에 재사용해야 합니다.

주제

- 로우레벨 API 사용
- 하이레벨 API 사용
- Eclipse를 사용하여 Amazon Glacier의 Java 예 실행
- 엔드포인트 설정

로우레벨 API 사용

로우레벨 `AmazonGlacierClient` 클래스는 Amazon Glacier의 기본 REST 작업을 매핑하는 데 필요한 모든 메서드를 제공합니다(Amazon Glacier를 위한 API 참조). 이러한 메서드를 직접 호출할 때는 먼저 해당하는 요청 객체를 생성한 후 메서드에서 작업에 대한 Amazon Glacier의 응답을 반환할 수 있도록 응답 객체를 제공해야 합니다.

예를 들어 `AmazonGlacierClient` 클래스는 볼트를 생성할 수 있는 `createVault` 메서드를 제공합니다. 이 메서드는 기본적인 볼트 생성 REST 작업에 매핑됩니다(볼트 만들기(PUT vault) 참조). 이 메서드를 사용하려면 먼저 다음 Java 코드 조각과 같이 Amazon Glacier 응답을 수신할 `CreateVaultResult` 객체 인스턴스를 생성해야 합니다.

```
AmazonGlacierClient client = new AmazonGlacierClient(credentials);
client.setEndpoint("https://glacier.us-west-2.amazonaws.com/");

CreateVaultRequest request = new CreateVaultRequest()
    .withAccountId("-")
```

```
    .withVaultName(vaultName);
CreateVaultResult result = client.createVault(createVaultRequest);
```

이 안내서에서 언급하는 로우레벨 샘플은 모두 이러한 패턴을 사용합니다.

> **Note**
>
> 요청을 생성할 때는 선행하는 코드 세그먼트가 `AccountID`를 지정합니다. 그러나를 사용하는 경우 요청`AccountId`의 AWS SDK for Java는 선택 사항이므로이 가이드의 모든 하위 수준 예제는이 값을 설정하지 않습니다. `AccountId`는 AWS 계정 ID입니다. 이 값은 요청에 서명하는 데 사용되는 자격 증명과 연결된 AWS 계정 ID와 일치해야 합니다. AWS 계정 ID 또는 선택적으로 '-'를 지정할 수 있습니다.이 경우 Amazon Glacier는 요청에 서명하는 데 사용되는 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 지정하는 경우에는 ID에 하이픈('-')을 추가하지 않습니다. 를 사용할 때 계정 ID를 제공하지 AWS SDK for Java않으면 라이브러리는 계정 ID를 '-'로 설정합니다.

하이레벨 API 사용

애플리케이션 개발을 더욱 간소화하기 위해는 하위 수준 API의 일부 메서드에 대해 상위 수준 추상화를 구현하는 `ArchiveTransferManager` 클래스를 AWS SDK for Java 제공합니다. 그 밖에 `upload`, `download` 메서드 등 아카이브 작업에 유용한 메서드도 있습니다.

예를 들어 다음 Java 코드 조각은 하이레벨 메서드인 `upload`를 사용하여 아카이브를 업로드합니다.

```
String vaultName = "examplevault";
String archiveToUpload = "c:/folder/exampleArchive.zip";

ArchiveTransferManager atm = new ArchiveTransferManager(client, credentials);
String archiveId = atm.upload(vaultName, "Tax 2012 documents", new
 File(archiveToUpload)).getArchiveId();
```

수행하는 모든 작업은 `ArchiveTransferManager` 객체를 생성할 때 지정한 AWS 리전에 적용됩니다. AWS 리전을 지정하지 않으면가 기본 AWS 리전`us-east-1`으로 AWS SDK for Java 설정됩니다.

이 안내서에서 언급하는 하이레벨 예제는 모두 이러한 패턴을 사용합니다.

> **Note**
>
> 하이레벨 `ArchiveTransferManager` 클래스는 `AmazonGlacierClient` 인스턴스 또는 `AWSCredentials` 인스턴스로 작성할 수 있습니다.

Eclipse를 사용하여 Amazon Glacier의 Java 예 실행

Java 코드 예제를 시작하는 가장 쉬운 방법은 최신 AWS Toolkit for Eclipse를 설치하는 것입니다. 최신 toolkit의 설치 및 업데이트 방법에 대한 자세한 내용은 http://aws.amazon.com/eclipse에서 확인할 수 있습니다. 다음은 이 섹션에서 제공하는 Java 코드 예제를 작성 및 테스트할 수 있는 작업에 대한 설명입니다.

Java 코드 생성 예제의 일반적인 프로세스

1	Java용 Amazon SDK의 자격 증명 제공 AWS SDK for Java 항목에 설명된 대로 AWS 자격 증명에 대한 기본 자격 증명 프로파일을 생성합니다. AWS
2	Eclipse에서 새 AWS Java 프로젝트를 생성합니다. 프로젝트는 AWS SDK for Java로 사전 구성되어 있습니다.
3	읽고 있는 섹션에서 코드를 프로젝트로 복사합니다.
4	필요한 데이터를 모두 제공하여 코드를 업데이트합니다. 예를 들어, 파일을 업로드하는 경우 파일 경로 및 버킷 이름을 제공합니다.
5	코드를 실행합니다. AWS Management Console을 사용하여 객체가 생성되었는지 확인합니다. 에 대한 자세한 내용은 http://aws.amazon.com/console/ AWS Management Console참조하십시오.

엔드포인트 설정

기본적으로는 엔드포인트를 AWS SDK for Java 사용합니다`https://glacier.us-east-1.amazonaws.com`. 다음 Java 코드 조각과 같이 엔드포인트를 명시적으로 설정할 수 있습니다.

다음은 로우레벨 API에서 엔드포인트를 미국 서부(오리건) 리전(`us-west-2`)으로 설정하는 방법을 나타낸 코드 조각입니다.

Example

```
client = new AmazonGlacierClient(credentials);
client.setEndpoint("glacier.us-west-2.amazonaws.com");
```

다음은 하이레벨 API에서 엔드포인트를 미국 서부(오리건) 리전으로 설정하는 방법을 나타낸 코드 조각입니다.

```
glacierClient = new AmazonGlacierClient(credentials);
sqsClient = new AmazonSQSClient(credentials);
snsClient = new AmazonSNSClient(credentials);

glacierClient.setEndpoint("glacier.us-west-2.amazonaws.com");
sqsClient.setEndpoint("sqs.us-west-2.amazonaws.com");
snsClient.setEndpoint("sns.us-west-2.amazonaws.com");

ArchiveTransferManager atm = new ArchiveTransferManager(glacierClient, sqsClient,
 snsClient);
```

지원되는 AWS 리전 및 엔드포인트 목록은 섹션을 참조하세요Amazon Glacier 액세스.

Amazon Glacier와 함께 AWS SDK for .NET 사용

`AWSSDK.dll`에서 AWS SDK for .NET API를 사용할 수 있습니다. AWS SDK for .NET 다운로드에 대한 자세한 내용은 샘플 코드 라이브러리를 참조하세요. Amazon Glacier에서 AWS SDKs 사용에서 설명한 것과 같이 AWS SDK for .NET은 하이레벨 API와 로우레벨 API를 모두 제공합니다.

> **Note**
>
> 로우레벨 API와 하이레벨 API는 Amazon Glacier에 액세스할 수 있도록 스레드 세이프 클라이언트를 제공합니다. 모범 사례로서 애플리케이션에서 클라이언트 하나를 생성한 후 스레드 간에 재사용해야 합니다.

주제

- 로우레벨 API 사용
- 하이레벨 API 사용

- 코드 예제 실행
- 엔드포인트 설정

로우레벨 API 사용

로우레벨 AmazonGlacierClient 클래스는 Amazon Glacier(Amazon Glacier)의 기본 REST 작업을 매핑하는 데 필요한 모든 메서드를 제공합니다(Amazon Glacier를 위한 API 참조). 이러한 메서드를 직접 호출할 때는 먼저 해당하는 요청 객체를 생성한 후 메서드에서 작업에 대한 Amazon Glacier의 응답을 반환할 수 있도록 응답 객체를 제공해야 합니다.

예를 들어 AmazonGlacierClient 클래스는 볼트를 생성할 수 있는 CreateVault 메서드를 제공합니다. 이 메서드는 기본적인 볼트 생성 REST 작업에 매핑됩니다(볼트 만들기(PUT vault) 참조). 이 메서드를 사용하기 위해서는 먼저 다음 C# 코드 조각과 같이 CreateVaultRequest 및 CreateVaultResponse 클래스의 인스턴스를 생성하여 요청 정보를 제공하고 Amazon Glacier 응답을 수신해야 합니다.

```
AmazonGlacierClient client;
client = new AmazonGlacierClient(Amazon.RegionEndpoint.USEast1);

CreateVaultRequest request = new CreateVaultRequest()
{
  AccountId = "-",
  VaultName = "*** Provide vault name ***"
};

CreateVaultResponse response = client.CreateVault(request);
```

이 안내서에서 언급하는 모든 로우레벨 샘플은 이러한 패턴을 사용합니다.

> **Note**
>
> 요청을 생성할 때는 선행하는 코드 세그먼트가 AccountId를 지정합니다. 하지만 AWS SDK for .NET을 사용할 때는 요청에서 AccountId가 옵션이기 때문에 이 안내서의 로우레벨 예제에서는 모두 이 값을 설정하지 않습니다. AccountId는 AWS 계정 ID입니다. 이 값은 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID와 일치해야 합니다. AWS 계정 ID 또는 선택적으로 하나의 하이픈('-')을 지정할 수 있습니다. 이 경우에 Amazon Glacier는 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 지정하는 경우에는 ID에 하이픈

('-')을 추가하지 않습니다. AWS SDK for .NET을 사용하면서 계정 ID를 입력하지 않으면 라이브러리가 계정 ID를 '-'으로 설정합니다.

하이레벨 API 사용

AWS SDK for .NET은 애플리케이션 배포를 더욱 간소화할 목적으로 로우레벨 API의 일부 메서드에 하이레벨 추상화를 구현할 수 있는 `ArchiveTransferManager` 클래스를 제공합니다. 그 밖에 `Upload`, `Download` 등 아카이브 작업에 유용한 메서드도 있습니다.

예를 들어 다음 C# 코드 조각은 하이레벨 메서드인 `Upload`를 사용하여 아카이브를 업로드합니다.

```
string vaultName = "examplevault";
string archiveToUpload = "c:\folder\exampleArchive.zip";

var manager = new ArchiveTransferManager(Amazon.RegionEndpoint.USEast1);
string archiveId = manager.Upload(vaultName, "archive description",
 archiveToUpload).ArchiveId;
```

단, 실행하는 모든 작업은 `ArchiveTransferManager` 객체를 생성할 때 지정한 AWS 리전에 적용됩니다. 이 안내서에서 언급하는 하이레벨 예제는 모두 이러한 패턴을 사용합니다.

> **Note**
>
> 하이레벨 `ArchiveTransferManager` 클래스는 여전히 로우레벨 `AmazonGlacierClient` 클라이언트가 필요하기 때문에 사용자가 명시적으로 전달하거나 `ArchiveTransferManager`가 클라이언트를 생성합니다.

코드 예제 실행

.NET 코드 예제를 시작하는 가장 쉬운 방법은 AWS SDK for .NET를 설치하는 것입니다. 자세한 내용은 Amazon SDK for .NET를 참조하세요.

다음은 이 안내서에서 제공되는 코드 예제의 테스트 단계를 간략히 나타낸 절차입니다.

.NET 코드 생성 예제의 일반 프로세스(Visual Studio 사용)

1	Amazon SDK for .NET 토픽 AWS 자격 증명 구성에 설명된 대로 사용자의 AWS 자격 증명에 대한 자격 증명 프로필을 생성합니다.
2	AWS 빈 프로젝트 템플릿을 사용하여 새 Visual Studio 프로젝트를 만듭니다.
3	프로젝트 파일 `Program.cs` 의 코드를 읽고 있는 섹션의 코드로 대체합니다.
4	코드를 실행합니다. AWS Management Console을 사용하여 객체가 생성되었는지 확인합니다. AWS Management Console에 대한 자세한 내용은 https://aws.amazon.com/console/을 참조하세요.

엔드포인트 설정

기본적으로 AWS SDK for .NET은 엔드포인트를 미국 서부(오레곤) 리전(`https://glacier.us-west-2.amazonaws.com`)으로 설정합니다. 다음 C# 코드 조각과 같이 엔드포인트를 다른 AWS 리전으로 설정하는 방법도 있습니다.

다음은 로우레벨 API에서 엔드포인트를 미국 서부(오레곤) 리전(`us-west-2`)으로 설정하는 방법을 나타낸 코드 조각입니다.

Example

```
AmazonGlacierClient client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2);
```

다음은 하이레벨 API에서 엔드포인트를 미국 서부(오레곤) 리전으로 설정하는 방법을 나타낸 코드 조각입니다.

```
var manager = new ArchiveTransferManager(Amazon.RegionEndpoint.USWest2);
```

현재 지원되는 AWS 리전 및 엔드포인트 목록은 Amazon Glacier 액세스 섹션을 참조하세요.

AWS SDKs를 사용하는 Amazon Glacier의 코드 예제

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

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

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

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

코드 예시

- AWS SDKs 사용하는 Amazon Glacier의 기본 예제
  - Hello Amazon Glacier
  - AWS SDKs 사용하는 Amazon Glacier에 대한 작업
    - AWS SDK 또는 CLI와 AddTagsToVault 함께 사용
    - AWS SDK 또는 CLI와 CreateVault 함께 사용
    - AWS SDK 또는 CLI와 DeleteArchive 함께 사용
    - AWS SDK 또는 CLI와 DeleteVault 함께 사용
    - AWS SDK 또는 CLI와 DeleteVaultNotifications 함께 사용
    - AWS SDK 또는 CLI와 DescribeJob 함께 사용
    - AWS SDK 또는 CLI와 DescribeVault 함께 사용
    - AWS SDK 또는 CLI와 GetJobOutput 함께 사용
    - AWS SDK 또는 CLI와 GetVaultNotifications 함께 사용
    - AWS SDK 또는 CLI와 InitiateJob 함께 사용
    - AWS SDK 또는 CLI와 ListJobs 함께 사용
    - AWS SDK 또는 CLI와 ListTagsForVault 함께 사용
    - AWS SDK 또는 CLI와 ListVaults 함께 사용
    - AWS SDK 또는 CLI와 SetVaultNotifications 함께 사용
    - AWS SDK 또는 CLI와 UploadArchive 함께 사용

    - AWS SDK 또는 CLI와 UploadMultipartPart 함께 사용
- AWS SDKs를 사용한 Amazon Glacier 시나리오
  - 파일을 Amazon Glacier에 아카이브하고, 알림을 받고, AWS SDK를 사용하여 작업을 시작합니다.
  - AWS SDK를 사용하여 Amazon Glacier 아카이브 콘텐츠 가져오기 및 아카이브 삭제

AWS SDKs 사용하는 Amazon Glacier의 기본 예제

다음 코드 예제에서는 AWS SDK와 함께 Amazon Glacier의 기본 기능을 사용하는 방법을 보여줍니다.

예제

- Hello Amazon Glacier
- AWS SDKs 사용하는 Amazon Glacier에 대한 작업
  - AWS SDK 또는 CLI와 AddTagsToVault 함께 사용
  - AWS SDK 또는 CLI와 CreateVault 함께 사용
  - AWS SDK 또는 CLI와 DeleteArchive 함께 사용
  - AWS SDK 또는 CLI와 DeleteVault 함께 사용
  - AWS SDK 또는 CLI와 DeleteVaultNotifications 함께 사용
  - AWS SDK 또는 CLI와 DescribeJob 함께 사용
  - AWS SDK 또는 CLI와 DescribeVault 함께 사용
  - AWS SDK 또는 CLI와 GetJobOutput 함께 사용
  - AWS SDK 또는 CLI와 GetVaultNotifications 함께 사용
  - AWS SDK 또는 CLI와 InitiateJob 함께 사용
  - AWS SDK 또는 CLI와 ListJobs 함께 사용
  - AWS SDK 또는 CLI와 ListTagsForVault 함께 사용
  - AWS SDK 또는 CLI와 ListVaults 함께 사용
  - AWS SDK 또는 CLI와 SetVaultNotifications 함께 사용
  - AWS SDK 또는 CLI와 UploadArchive 함께 사용
  - AWS SDK 또는 CLI와 UploadMultipartPart 함께 사용

Hello Amazon Glacier

다음 코드 예제에서는 Amazon Glacier 사용을 시작하는 방법을 보여줍니다.

.NET

SDK for .NET

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

```
using Amazon.Glacier;
using Amazon.Glacier.Model;

namespace GlacierActions;

public static class HelloGlacier
{
    static async Task Main()
    {
        var glacierService = new AmazonGlacierClient();

        Console.WriteLine("Hello Amazon Glacier!");
        Console.WriteLine("Let's list your Glacier vaults:");

        // You can use await and any of the async methods to get a response.
        // Let's get the vaults using a paginator.
        var glacierVaultPaginator = glacierService.Paginators.ListVaults(
            new ListVaultsRequest { AccountId = "-" });

        await foreach (var vault in glacierVaultPaginator.VaultList)
        {
            Console.WriteLine($"{vault.CreationDate}:{vault.VaultName}, ARN:
{vault.VaultARN}");
        }
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ListVaults를 참조하세요.

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

AWS SDKs 사용하는 Amazon Glacier에 대한 작업

다음 코드 예제에서는 AWS SDKs를 사용하여 개별 Amazon Glacier 작업을 수행하는 방법을 보여줍니다. 각 예제에는 GitHub에 대한 링크가 포함되어 있습니다. 여기에서 코드 설정 및 실행에 대한 지침을 찾을 수 있습니다.

Amazon Glacier API를 직접적으로 호출하는 이들 발췌문은 컨텍스트에서 실행되어야 하는 더 큰 프로그램에서 발췌한 코드입니다. AWS SDKs를 사용한 Amazon Glacier 시나리오에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 Amazon Glacier API 참조를 참조하세요.

예제

- AWS SDK 또는 CLI와 AddTagsToVault 함께 사용
- AWS SDK 또는 CLI와 CreateVault 함께 사용
- AWS SDK 또는 CLI와 DeleteArchive 함께 사용
- AWS SDK 또는 CLI와 DeleteVault 함께 사용
- AWS SDK 또는 CLI와 DeleteVaultNotifications 함께 사용
- AWS SDK 또는 CLI와 DescribeJob 함께 사용
- AWS SDK 또는 CLI와 DescribeVault 함께 사용
- AWS SDK 또는 CLI와 GetJobOutput 함께 사용
- AWS SDK 또는 CLI와 GetVaultNotifications 함께 사용
- AWS SDK 또는 CLI와 InitiateJob 함께 사용
- AWS SDK 또는 CLI와 ListJobs 함께 사용
- AWS SDK 또는 CLI와 ListTagsForVault 함께 사용
- AWS SDK 또는 CLI와 ListVaults 함께 사용
- AWS SDK 또는 CLI와 SetVaultNotifications 함께 사용
- AWS SDK 또는 CLI와 UploadArchive 함께 사용
- AWS SDK 또는 CLI와 UploadMultipartPart 함께 사용

AWS SDK 또는 CLI와 **AddTagsToVault** 함께 사용

다음 코드 예시는 AddTagsToVault의 사용 방법을 보여 줍니다.

.NET

SDK for .NET

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

```
   /// <summary>
   /// Add tags to the items in an Amazon S3 Glacier vault.
   /// </summary>
   /// <param name="vaultName">The name of the vault to add tags to.</param>
   /// <param name="key">The name of the object to tag.</param>
   /// <param name="value">The tag value to add.</param>
   /// <returns>A Boolean value indicating the success of the action.</returns>
   public async Task<bool> AddTagsToVaultAsync(string vaultName, string key,
string value)
   {
       var request = new AddTagsToVaultRequest
       {
           Tags = new Dictionary<string, string>
               {
                   { key, value },
               },
           AccountId = "-",
           VaultName = vaultName,
       };

       var response = await _glacierService.AddTagsToVaultAsync(request);
       return response.HttpStatusCode == HttpStatusCode.NoContent;
   }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 AddTagsToVault를 참조하세요.

CLI

AWS CLI

다음 명령은 이름이 지정된 `my-vault`에 두 개의 태그를 추가합니다.

```
aws glacier add-tags-to-vault --account-id - --vault-name my-vault --
tags id=1234,date=july2015
```

Amazon Glacier에서는 작업을 수행할 때 계정 ID 인수가 필요하지만 하이픈을 사용하여 사용 중인 계정을 지정할 수 있습니다.

- API 세부 정보는 AWS CLI 명령 참조의 AddTagsToVault 섹션을 참조하세요.

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

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

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

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 파일을 보관하고, 알림을 받고 작업을 시작

.NET

SDK for .NET

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

```
/// <summary>
/// Create an Amazon S3 Glacier vault.
```

```
    /// </summary>
    /// <param name="vaultName">The name of the vault to create.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateVaultAsync(string vaultName)
    {
        var request = new CreateVaultRequest
        {
            // Setting the AccountId to "-" means that
            // the account associated with the current
            // account will be used.
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await _glacierService.CreateVaultAsync(request);

        Console.WriteLine($"Created {vaultName} at: {response.Location}");

        return response.HttpStatusCode == HttpStatusCode.Created;
    }
```

- API에 대한 세부 정보는 AWS SDK for .NET API 참조의 CreateVault를 참조하세요.

CLI

AWS CLI

다음 명령은 `my-vault`라는 새 볼트를 생성합니다.

```
aws glacier create-vault --vault-name my-vault --account-id -
```

Amazon Glacier에서는 작업을 수행할 때 계정 ID 인수가 필요하지만 하이픈을 사용하여 사용 중인 계정을 지정할 수 있습니다.

- API 세부 정보는 AWS CLI 명령 참조의 CreateVault를 참조하세요.

Java

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.CreateVaultRequest;
import software.amazon.awssdk.services.glacier.model.CreateVaultResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateVault {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <vaultName>

                Where:
                   vaultName - The name of the vault to create.

                """;

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

        String vaultName = args[0];
```

```
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createGlacierVault(glacier, vaultName);
        glacier.close();
    }

    public static void createGlacierVault(GlacierClient glacier, String
 vaultName) {
        try {
            CreateVaultRequest vaultRequest = CreateVaultRequest.builder()
                    .vaultName(vaultName)
                    .build();

            CreateVaultResponse createVaultResult =
 glacier.createVault(vaultRequest);
            System.out.println("The URI of the new vault is " +
 createVaultResult.location());

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

- API 세부 정보는AWS SDK for Java 2.x API 참조의 CreateVault를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

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

클라이언트를 생성합니다.

```
const { GlacierClient } = require("@aws-sdk/client-glacier");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const glacierClient = new GlacierClient({ region: REGION });
export { glacierClient };
```

볼트를 생성합니다.

```
// Load the SDK for JavaScript
import { CreateVaultCommand } from "@aws-sdk/client-glacier";
import { glacierClient } from "./libs/glacierClient.js";

// Set the parameters
const vaultname = "VAULT_NAME"; // VAULT_NAME
const params = { vaultName: vaultname };

const run = async () => {
  try {
    const data = await glacierClient.send(new CreateVaultCommand(params));
    console.log("Success, vault created!");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error");
  }
};
run();
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 CreateVault를 참조하십시오.

SDK for JavaScript (v2)

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

```
// Load the SDK for JavaScript
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create a new service object
var glacier = new AWS.Glacier({ apiVersion: "2012-06-01" });
// Call Glacier to create the vault
glacier.createVault({ vaultName: "YOUR_VAULT_NAME" }, function (err) {
  if (!err) {
    console.log("Created vault!");
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 CreateVault를 참조하세요.

PowerShell

Tools for PowerShell V4

예제 1: 사용자 계정용 새 볼트를 생성합니다. -AccountId 파라미터에 값이 제공되지 않았으므로 cmdlet은 현재 계정을 나타내는 기본값인 '-'를 사용합니다.

```
New-GLCVault -VaultName myvault
```

출력:

```
/01234567812/vaults/myvault
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateVault를 참조하세요.

Tools for PowerShell V5

예제 1: 사용자 계정용 새 볼트를 생성합니다. -AccountId 파라미터에 값이 제공되지 않았으므로 cmdlet은 현재 계정을 나타내는 기본값인 '-'를 사용합니다.

```
New-GLCVault -VaultName myvault
```

출력:

```
/01234567812/vaults/myvault
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 CreateVault를 참조하세요.

Python

SDK for Python(Boto3)

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

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    def create_vault(self, vault_name):
        """
        Creates a vault.

        :param vault_name: The name to give the vault.
        :return: The newly created vault.
        """
        try:
            vault = self.glacier_resource.create_vault(vaultName=vault_name)
            logger.info("Created vault %s.", vault_name)
        except ClientError:
            logger.exception("Couldn't create vault %s.", vault_name)
            raise
        else:
            return vault
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 CreateVault를 참조하세요.

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

AWS SDK 또는 CLI와 **DeleteArchive** 함께 사용

다음 코드 예시는 DeleteArchive의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 아카이브 콘텐츠 가져오기 및 아카이브 삭제

CLI

AWS CLI

볼트에서 아카이브를 삭제하는 방법

다음 delete-archive 예시에서는 example_vault에서 지정된 아카이브를 제거합니다.

```
aws glacier delete-archive \
    --account-id 111122223333  \
    --vault-name example_vault \
    --archive-id Sc0u9ZP8yaWkmh-XGlIvAVprtLhaLCGnNwNl5I5x9HqPIkX5mjc0DrId3Ln-
Gi_k2HzmlIDZUz117KSdVMdMXLuFWi9PJUitxWO73edQ43eTlMWkH0pd9zVSAuV_XXZBVhKhyGhJ7w
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteArchive를 참조하세요.

Java

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DeleteArchiveRequest;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteArchive {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <vaultName> <accountId> <archiveId>

                Where:
                   vaultName - The name of the vault that contains the archive to
 delete.
                   accountId - The account ID value.
                   archiveId - The archive ID value.
                """;

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

```
        String vaultName = args[0];
        String accountId = args[1];
        String archiveId = args[2];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deleteGlacierArchive(glacier, vaultName, accountId, archiveId);
        glacier.close();
    }

    public static void deleteGlacierArchive(GlacierClient glacier, String
 vaultName, String accountId,
            String archiveId) {
        try {
            DeleteArchiveRequest delArcRequest = DeleteArchiveRequest.builder()
                    .vaultName(vaultName)
                    .accountId(accountId)
                    .archiveId(archiveId)
                    .build();

            glacier.deleteArchive(delArcRequest);
            System.out.println("The archive was deleted.");

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

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

Python

SDK for Python (Boto3)

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

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def delete_archive(archive):
        """
        Deletes an archive from a vault.

        :param archive: The archive to delete.
        """
        try:
            archive.delete()
            logger.info(
                "Deleted archive %s from vault %s.", archive.id,
 archive.vault_name
            )
        except ClientError:
            logger.exception("Couldn't delete archive %s.", archive.id)
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DeleteArchive를 참조하세요.

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

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

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

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 아카이브 콘텐츠 가져오기 및 아카이브 삭제

CLI

AWS CLI

다음 명령은 my-vault라는 볼트를 삭제합니다.

```
aws glacier delete-vault --vault-name my-vault --account-id -
```

이 명령은 출력을 생성하지 않습니다. Amazon Glacier에서는 작업을 수행할 때 계정 ID 인수가 필요하지만 하이픈을 사용하여 사용 중인 계정을 지정할 수 있습니다.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteVault를 참조하세요.

Java

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DeleteVaultRequest;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteVault {
```

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

        final String usage = """

                Usage:    <vaultName>

                Where:
                   vaultName - The name of the vault to delete.\s
                """;

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

        String vaultName = args[0];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deleteGlacierVault(glacier, vaultName);
        glacier.close();
    }

    public static void deleteGlacierVault(GlacierClient glacier, String
 vaultName) {
        try {
            DeleteVaultRequest delVaultRequest = DeleteVaultRequest.builder()
                    .vaultName(vaultName)
                    .build();

            glacier.deleteVault(delVaultRequest);
            System.out.println("The vault was deleted!");

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

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

Python

SDK for Python (Boto3)

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

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def delete_vault(vault):
        """
        Deletes a vault.

        :param vault: The vault to delete.
        """
        try:
            vault.delete()
            logger.info("Deleted vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't delete vault %s.", vault.name)
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DeleteVault를 참조하세요.

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

AWS SDK 또는 CLI와 **DeleteVaultNotifications** 함께 사용

다음 코드 예시는 DeleteVaultNotifications의 사용 방법을 보여 줍니다.

CLI

AWS CLI

저장소에 대한 SNS 알림 제거

다음 delete-vault-notifications 예시에서는 지정된 볼트에 대해 Amazon Simple Notification Service(Amazon SNS)에서 전송한 알림을 제거합니다.

```
aws glacier delete-vault-notifications \
    --account-id 111122223333 \
    --vault-name example_vault
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteVaultNotifications를 참조하세요.

Python

SDK for Python(Boto3)

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

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource
```

```
    @staticmethod
    def stop_notifications(notification):
        """
        Stops notifications to the configured Amazon SNS topic.

        :param notification: The notification configuration to remove.
        """
        try:
            notification.delete()
            logger.info("Notifications stopped.")
        except ClientError:
            logger.exception("Couldn't stop notifications.")
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DeleteVaultNotifications를 참조하세요.

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

AWS SDK 또는 CLI와 **DescribeJob** 함께 사용

다음 코드 예시는 DescribeJob의 사용 방법을 보여 줍니다.

CLI

AWS CLI

다음 명령은 my-vault라는 저장소의 인벤토리 검색 작업에 대한 정보를 검색합니다.

```
aws glacier describe-job --account-id - --vault-name my-
vault --job-id zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61ds04CnMW
```

출력:

```
{
```

```
    "InventoryRetrievalParameters": {
        "Format": "JSON"
    },
    "VaultARN": "arn:aws:glacier:us-west-2:0123456789012:vaults/my-vault",
    "Completed": false,
    "JobId": "zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW",
    "Action": "InventoryRetrieval",
    "CreationDate": "2015-07-17T20:23:41.616Z",
    "StatusCode": "InProgress"
}
```

작업 ID는 `aws glacier initiate-job` 및 `aws glacier list-jobs`의 출력에서 찾을 수 있습니다. Amazon Glacier에서는 작업을 수행할 때 계정 ID 인수가 필요하지만 하이픈을 사용하여 사용 중인 계정을 지정할 수 있습니다.

- API 세부 정보는 AWS CLI 명령 참조의 DescribeJob을 참조하세요.

PowerShell

Tools for PowerShell V4

예제 1: 지정된 작업의 세부 정보를 반환합니다. 작업이 성공적으로 완료되면 Read-GCJobOutput cmdlet을 사용하여 작업 콘텐츠(아카이브 또는 인벤토리 목록)를 로컬 파일 시스템으로 가져올 수 있습니다.

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

출력:

```
Action                       : ArchiveRetrieval
ArchiveId                    : o9O9j...X-TpIhQJw
ArchiveSHA256TreeHash        : 79f3ea754c02f58...dc57bf4395b
ArchiveSizeInBytes           : 38034480
Completed                    : False
CompletionDate               : 1/1/0001 12:00:00 AM
CreationDate                 : 12/13/2018 11:00:14 AM
InventoryRetrievalParameters :
InventorySizeInBytes         : 0
JobDescription               :
JobId                        : op1x...JSbthM
JobOutputPath                :
```

```
OutputLocation                :
RetrievalByteRange            : 0-38034479
SelectParameters              :
SHA256TreeHash                : 79f3ea754c02f58...dc57bf4395b
SNSTopic                      :
StatusCode                    : InProgress
StatusMessage                 :
Tier                          : Standard
VaultARN                      : arn:aws:glacier:us-west-2:012345678912:vaults/test
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeJob을 참조하세요.

Tools for PowerShell V5

예제 1: 지정된 작업의 세부 정보를 반환합니다. 작업이 성공적으로 완료되면 Read-GCJobOutput cmdlet을 사용하여 작업 콘텐츠(아카이브 또는 인벤토리 목록)를 로컬 파일 시스템으로 가져올 수 있습니다.

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

출력:

```
Action                        : ArchiveRetrieval
ArchiveId                     : o9O9j...X-TpIhQJw
ArchiveSHA256TreeHash         : 79f3ea754c02f58...dc57bf4395b
ArchiveSizeInBytes            : 38034480
Completed                     : False
CompletionDate                : 1/1/0001 12:00:00 AM
CreationDate                  : 12/13/2018 11:00:14 AM
InventoryRetrievalParameters :
InventorySizeInBytes          : 0
JobDescription                :
JobId                         : op1x...JSbthM
JobOutputPath                 :
OutputLocation                :
RetrievalByteRange            : 0-38034479
SelectParameters              :
SHA256TreeHash                : 79f3ea754c02f58...dc57bf4395b
SNSTopic                      :
StatusCode                    : InProgress
StatusMessage                 :
Tier                          : Standard
VaultARN                      : arn:aws:glacier:us-west-2:012345678912:vaults/test
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 DescribeJob을 참조하세요.

Python

SDK for Python(Boto3)

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

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def get_job_status(job):
        """
        Gets the status of a job.

        :param job: The job to query.
        :return: The current status of the job.
        """
        try:
            job.load()
            logger.info(
                "Job %s is performing action %s and has status %s.",
                job.id,
                job.action,
                job.status_code,
            )
        except ClientError:
            logger.exception("Couldn't get status for job %s.", job.id)
            raise
        else:
```

```
            return job.status_code
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DescribeJob을 참조하세요.

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

AWS SDK 또는 CLI와 **DescribeVault** 함께 사용

다음 코드 예시는 DescribeVault의 사용 방법을 보여 줍니다.

.NET

SDK for .NET

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

```
/// <summary>
/// Describe an Amazon S3 Glacier vault.
/// </summary>
/// <param name="vaultName">The name of the vault to describe.</param>
/// <returns>The Amazon Resource Name (ARN) of the vault.</returns>
public async Task<string> DescribeVaultAsync(string vaultName)
{
    var request = new DescribeVaultRequest
    {
        AccountId = "-",
        VaultName = vaultName,
    };

    var response = await _glacierService.DescribeVaultAsync(request);

    // Display the information about the vault.
    Console.WriteLine($"{response.VaultName}\tARN: {response.VaultARN}");
```

```
        Console.WriteLine($"Created on: {response.CreationDate}\tNumber
of Archives: {response.NumberOfArchives}\tSize (in bytes):
{response.SizeInBytes}");
        if (response.LastInventoryDate != DateTime.MinValue)
        {
            Console.WriteLine($"Last inventory: {response.LastInventoryDate}");
        }

        return response.VaultARN;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DescribeVault를 참조하세요.

CLI

AWS CLI

다음 명령은 `my-vault`라는 볼트에 대한 데이터를 검색합니다.

```
aws glacier describe-vault --vault-name my-vault --account-id -
```

Amazon Glacier에서는 작업을 수행할 때 계정 ID 인수가 필요하지만 하이픈을 사용하여 사용 중인 계정을 지정할 수 있습니다.

- API 세부 정보는 AWS CLI 명령 참조의 DescribeVault 섹션을 참조하세요.

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

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

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

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 아카이브 콘텐츠 가져오기 및 아카이브 삭제

CLI

AWS CLI

다음 명령은 볼트 인벤토리 작업의 출력을 output.json라는 현재 디렉터리의 파일에 저장합니다.

```
aws glacier get-job-output --account-id - --vault-name my-
vault --job-id zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW output.json
```

job-id는 aws glacier list-jobs의 출력에서 확인할 수 있습니다. 참고로 출력 파일 이름은 옵션 이름이 접두사로 붙지 않는 위치 인수입니다. Amazon Glacier에서는 작업을 수행할 때 계정 ID 인수가 필요하지만 하이픈을 사용하여 사용 중인 계정을 지정할 수 있습니다.

출력:

```
{
    "status": 200,
    "acceptRanges": "bytes",
    "contentType": "application/json"
}
```

output.json:

```
{"VaultARN":"arn:aws:glacier:us-west-2:0123456789012:vaults/
my-vault","InventoryDate":"2015-04-07T00:26:18Z","ArchiveList":
[{"ArchiveId":"kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--zM_mw6k76ZFGEIWQX-
ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw","ArchiveDescription":"multipart
 upload
 test","CreationDate":"2015-04-06T22:24:34Z","Size":3145728,"SHA256TreeHash":"9628195fcdb
```

- API 세부 정보는 AWS CLI 명령 참조의 GetJobOutput을 참조하세요.

PowerShell

Tools for PowerShell V4

예제 1: 지정된 작업에서 가져오도록 예약된 아카이브 콘텐츠를 다운로드하고 콘텐츠를 디스크의 파일에 저장합니다. 다운로드 시 체크섬(있는 경우)이 검증됩니다. 원하는 경우 **-Select '*'**를 지정하여 체크섬을 포함한 전체 응답을 반환할 수 있습니다.

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:
\temp\blue.bin"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetJobOutput을 참조하세요.

Tools for PowerShell V5

예제 1: 지정된 작업에서 가져오도록 예약된 아카이브 콘텐츠를 다운로드하고 콘텐츠를 디스크의 파일에 저장합니다. 다운로드 시 체크섬(있는 경우)이 검증됩니다. 원하는 경우 **-Select '*'**를 지정하여 체크섬을 포함한 전체 응답을 반환할 수 있습니다.

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:
\temp\blue.bin"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 GetJobOutput을 참조하세요.

Python

SDK for Python(Boto3)

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

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
```

```
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def get_job_output(job):
        """
        Gets the output of a job, such as a vault inventory or the contents of an
        archive.

        :param job: The job to get output from.
        :return: The job output, in bytes.
        """
        try:
            response = job.get_output()
            out_bytes = response["body"].read()
            logger.info("Read %s bytes from job %s.", len(out_bytes), job.id)
            if "archiveDescription" in response:
                logger.info(
                    "These bytes are described as '%s'",
response["archiveDescription"]
                )
        except ClientError:
            logger.exception("Couldn't get output for job %s.", job.id)
            raise
        else:
            return out_bytes
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 GetJobOutput를 참조하세요.

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

AWS SDK 또는 CLI와 **GetVaultNotifications** 함께 사용

다음 코드 예시는 GetVaultNotifications의 사용 방법을 보여 줍니다.

CLI

AWS CLI

다음 명령은 my-vault라는 저장소의 알림 구성 설명을 가져옵니다.

```
aws glacier get-vault-notifications --account-id - --vault-name my-vault
```

출력:

```
{
    "vaultNotificationConfig": {
        "Events": [
            "InventoryRetrievalCompleted",
            "ArchiveRetrievalCompleted"
        ],
        "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-vault"
    }
}
```

볼트에 대한 알림이 구성되지 않은 경우에는 오류가 반환됩니다. Amazon Glacier에서는 작업을 수행할 때 계정 ID 인수가 필요하지만 하이픈을 사용하여 사용 중인 계정을 지정할 수 있습니다.

- API 세부 정보는 AWS CLI 명령 참조의 GetVaultNotifications를 참조하세요.

Python

SDK for Python(Boto3)

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

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
```

```
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def get_notification(vault):
        """
        Gets the currently notification configuration for a vault.

        :param vault: The vault to query.
        :return: The notification configuration for the specified vault.
        """
        try:
            notification = vault.Notification()
            logger.info(
                "Vault %s notifies %s on %s events.",
                vault.name,
                notification.sns_topic,
                notification.events,
            )
        except ClientError:
            logger.exception("Couldn't get notification data for %s.",
vault.name)
            raise
        else:
            return notification
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 GetVaultNotifications를 참조하세요.

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

AWS SDK 또는 CLI와 **InitiateJob** 함께 사용

다음 코드 예시는 InitiateJob의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 파일을 보관하고, 알림을 받고 작업을 시작

.NET

SDK for .NET

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

저장소에서 아카이브를 가져옵니다. 이 예제에서는 ArchiveTransferManager 클래스를 사용합니다. API 세부 정보는 ArchiveTransferManager를 참조하세요.

```
    /// <summary>
    /// Download an archive from an Amazon S3 Glacier vault using the Archive
    /// Transfer Manager.
    /// </summary>
    /// <param name="vaultName">The name of the vault containing the object.</
param>
    /// <param name="archiveId">The Id of the archive to download.</param>
    /// <param name="localFilePath">The local directory where the file will
    /// be stored after download.</param>
    /// <returns>Async Task.</returns>
    public async Task<bool> DownloadArchiveWithArchiveManagerAsync(string
 vaultName, string archiveId, string localFilePath)
    {
        try
        {
            var manager = new ArchiveTransferManager(_glacierService);

            var options = new DownloadOptions
            {
                StreamTransferProgress = Progress!,
            };

            // Download an archive.
            Console.WriteLine("Initiating the archive retrieval job and then
 polling SQS queue for the archive to be available.");
            Console.WriteLine("When the archive is available, downloading will
 begin.");
```

```
            await manager.DownloadAsync(vaultName, archiveId, localFilePath,
 options);

            return true;
        }
        catch (AmazonGlacierException ex)
        {
            Console.WriteLine(ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Event handler to track the progress of the Archive Transfer Manager.
    /// </summary>
    /// <param name="sender">The object that raised the event.</param>
    /// <param name="args">The argument values from the object that raised the
    /// event.</param>
    static void Progress(object sender, StreamTransferProgressArgs args)
    {
        if (args.PercentDone != _currentPercentage)
        {
            _currentPercentage = args.PercentDone;
            Console.WriteLine($"Downloaded {_currentPercentage}%");
        }
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 InitiateJob을 참조하세요.

CLI

AWS CLI

다음 명령은 `my-vault` 저장소의 인벤토리를 가져오는 작업을 시작합니다.

```
aws glacier initiate-job --account-id - --vault-name my-vault --job-parameters
 '{"Type": "inventory-retrieval"}'
```

출력:

```
{
    "location": "/0123456789012/vaults/my-vault/jobs/
zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW",
    "jobId": "zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW"
}
```

Amazon Glacier에서는 작업을 수행할 때 계정 ID 인수가 필요하지만 하이픈을 사용하여 사용 중인 계정을 지정할 수 있습니다.

다음 명령은 `my-vault` 저장소에서 아카이브를 가져오는 작업을 시작합니다.

```
aws glacier initiate-job --account-id - --vault-name my-vault --job-
parameters file://job-archive-retrieval.json
```

`job-archive-retrieval.json`은 작업 유형, 아카이브 ID 및 일부 선택적 파라미터를 지정하는 로컬 폴더의 JSON 파일입니다.

```
{
  "Type": "archive-retrieval",
  "ArchiveId": "kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--zM_mw6k76ZFGEIWQX-
ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw",
  "Description": "Retrieve archive on 2015-07-17",
  "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-topic"
}
```

아카이브 ID는 `aws glacier upload-archive` 및 `aws glacier get-job-output` 출력에 표시됩니다.

출력:

```
{
    "location": "/011685312445/vaults/mwunderl/jobs/l7IL5-
EkXyEY9Ws95fClzIbk2O5uLYaFdAYOi-
azsX_Z8V6NH4yERHzars8wTKYQMX6nBDI9cMNHzyZJO59-8N9aHWav",
    "jobId": "l7IL5-EkXy2O5uLYaFdAYOiEY9Ws95fClzIbk-
azsX_Z8V6NH4yERHzars8wTKYQMX6nBDI9cMNHzyZJO59-8N9aHWav"
}
```

작업 파라미터 형식에 대한 자세한 내용은 Amazon Glacier API 참조의 작업 시작 섹션을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 InitiateJob을 참조하세요.

Java

SDK for Java 2.x

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

저장소 인벤토리를 가져옵니다.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.JobParameters;
import software.amazon.awssdk.services.glacier.model.InitiateJobResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import software.amazon.awssdk.services.glacier.model.InitiateJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobResponse;
import software.amazon.awssdk.services.glacier.model.GetJobOutputRequest;
import software.amazon.awssdk.services.glacier.model.GetJobOutputResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
```

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

        final String usage = """

                Usage:    <vaultName> <accountId> <path>

                Where:
                   vaultName - The name of the vault.
                   accountId - The account ID value.
                   path - The path where the file is written to.
                """;

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

        String vaultName = args[0];
        String accountId = args[1];
        String path = args[2];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String jobNum = createJob(glacier, vaultName, accountId);
        checkJob(glacier, jobNum, vaultName, accountId, path);
        glacier.close();
    }

    public static String createJob(GlacierClient glacier, String vaultName,
 String accountId) {
        try {
            JobParameters job = JobParameters.builder()
                    .type("inventory-retrieval")
                    .build();

            InitiateJobRequest initJob = InitiateJobRequest.builder()
                    .jobParameters(job)
                    .accountId(accountId)
                    .vaultName(vaultName)
                    .build();

            InitiateJobResponse response = glacier.initiateJob(initJob);
```

```
            System.out.println("The job ID is: " + response.jobId());
            System.out.println("The relative URI path of the job is: " +
response.location());
            return response.jobId();

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

        }
        return "";
    }

    // Poll S3 Glacier = Polling a Job may take 4-6 hours according to the
    // Documentation.
    public static void checkJob(GlacierClient glacier, String jobId, String name,
String account, String path) {
        try {
            boolean finished = false;
            String jobStatus;
            int yy = 0;

            while (!finished) {
                DescribeJobRequest jobRequest = DescribeJobRequest.builder()
                        .jobId(jobId)
                        .accountId(account)
                        .vaultName(name)
                        .build();

                DescribeJobResponse response = glacier.describeJob(jobRequest);
                jobStatus = response.statusCodeAsString();

                if (jobStatus.compareTo("Succeeded") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + jobStatus);
                    Thread.sleep(1000);
                }
                yy++;
            }

            System.out.println("Job has Succeeded");
            GetJobOutputRequest jobOutputRequest = GetJobOutputRequest.builder()
                    .jobId(jobId)
```

```
                    .vaultName(name)
                    .accountId(account)
                    .build();

            ResponseBytes<GetJobOutputResponse> objectBytes =
 glacier.getJobOutputAsBytes(jobOutputRequest);
            // Write the data to a local file.
            byte[] data = objectBytes.asByteArray();
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from a Glacier
 vault");
            os.close();

        } catch (GlacierException | InterruptedException | IOException e) {
            System.out.println(e.getMessage());
            System.exit(1);

        }
    }
}
```

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

PowerShell

Tools for PowerShell V4

예제 1: 사용자가 소유한 지정된 볼트에서 아카이브를 가져오는 작업을 시작합니다. Get-GLCJob cmdlet을 사용하여 작업 상태를 확인할 수 있습니다. 작업이 성공적으로 완료되면 Read-GCJobOutput cmdlet을 사용하여 아카이브 콘텐츠를 로컬 파일 시스템으로 가져올 수 있습니다.

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription
 "archive retrieval" -ArchiveId "o9O9j...TX-TpIhQJw"
```

출력:

```
JobId            JobOutputPath Location
-----            ------------- --------
```

```
op1x...JSbthM                  /012345678912/vaults/test/jobs/
op1xe...I4HqCHkSJSbthM
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 InitiateJob을 참조하세요.

Tools for PowerShell V5

예제 1: 사용자가 소유한 지정된 볼트에서 아카이브를 가져오는 작업을 시작합니다. Get-GLCJob cmdlet을 사용하여 작업 상태를 확인할 수 있습니다. 작업이 성공적으로 완료되면 Read-GCJobOutput cmdlet을 사용하여 아카이브 콘텐츠를 로컬 파일 시스템으로 가져올 수 있습니다.

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription
 "archive retrieval" -ArchiveId "o9O9j...TX-TpIhQJw"
```

출력:

```
JobId            JobOutputPath Location
-----            ------------- --------
op1x...JSbthM                  /012345678912/vaults/test/jobs/
op1xe...I4HqCHkSJSbthM
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 InitiateJob을 참조하세요.

Python

SDK for Python(Boto3)

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

저장소 인벤토리를 가져옵니다.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
```

```
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def initiate_inventory_retrieval(vault):
        """
        Initiates an inventory retrieval job. The inventory describes the
 contents
        of the vault. Standard retrievals typically complete within 3–5 hours.
        When the job completes, you can get the inventory by calling
 get_output().

        :param vault: The vault to inventory.
        :return: The inventory retrieval job.
        """
        try:
            job = vault.initiate_inventory_retrieval()
            logger.info("Started %s job with ID %s.", job.action, job.id)
        except ClientError:
            logger.exception("Couldn't start job on vault %s.", vault.name)
            raise
        else:
            return job
```

저장소에서 아카이브를 가져옵니다.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def initiate_archive_retrieval(archive):
```

```
        """
        Initiates an archive retrieval job. Standard retrievals typically
complete
        within 3–5 hours. When the job completes, you can get the archive
contents
        by calling get_output().

        :param archive: The archive to retrieve.
        :return: The archive retrieval job.
        """
        try:
            job = archive.initiate_archive_retrieval()
            logger.info("Started %s job with ID %s.", job.action, job.id)
        except ClientError:
            logger.exception("Couldn't start job on archive %s.", archive.id)
            raise
        else:
            return job
```

- API 세부 정보는 AWS SDK for Python(Boto3) API 참조의 InitiateJob을 참조하세요.

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

AWS SDK 또는 CLI와 **`ListJobs`** 함께 사용

다음 코드 예시는 `ListJobs`의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 파일을 보관하고, 알림을 받고 작업을 시작
- 아카이브 콘텐츠 가져오기 및 아카이브 삭제

.NET

SDK for .NET

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

```
    /// <summary>
    /// List Amazon S3 Glacier jobs.
    /// </summary>
    /// <param name="vaultName">The name of the vault to list jobs for.</param>
    /// <returns>A list of Amazon S3 Glacier jobs.</returns>
    public async Task<List<GlacierJobDescription>> ListJobsAsync(string
vaultName)
    {
        var request = new ListJobsRequest
        {
            // Using a hyphen "-" for the Account Id will
            // cause the SDK to use the Account Id associated
            // with the current account.
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await _glacierService.ListJobsAsync(request);

        return response.JobList;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ListJobs를 참조하세요.

CLI

AWS CLI

다음 명령은 `my-vault`라는 볼트에 대해 진행 중인 작업과 최근에 완료된 작업을 나열합니다.

```
aws glacier list-jobs --account-id - --vault-name my-vault
```

출력:

```
{
    "JobList": [
        {
            "VaultARN": "arn:aws:glacier:us-west-2:0123456789012:vaults/my-
vault",
            "RetrievalByteRange": "0-3145727",
            "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-vault",
            "Completed": false,
            "SHA256TreeHash":
 "9628195fcdbcbbe76cdde932d4646fa7de5f219fb39823836d81f0cc0e18aa67",
            "JobId": "l7IL5-EkXyEY9Ws95fClzIbk2O5uLYaFdAYOi-
azsX_Z8V6NH4yERHzars8wTKYQMX6nBDI9cMNHzyZJO59-8N9aHWav",
            "ArchiveId": "kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--
zM_mw6k76ZFGEIWQX-ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw",
            "JobDescription": "Retrieve archive on 2015-07-17",
            "ArchiveSizeInBytes": 3145728,
            "Action": "ArchiveRetrieval",
            "ArchiveSHA256TreeHash":
 "9628195fcdbcbbe76cdde932d4646fa7de5f219fb39823836d81f0cc0e18aa67",
            "CreationDate": "2015-07-17T21:16:13.840Z",
            "StatusCode": "InProgress"
        },
        {
            "InventoryRetrievalParameters": {
                "Format": "JSON"
            },
            "VaultARN": "arn:aws:glacier:us-west-2:0123456789012:vaults/my-
vault",
            "Completed": false,
            "JobId": "zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW",
            "Action": "InventoryRetrieval",
            "CreationDate": "2015-07-17T20:23:41.616Z",
            "StatusCode": ""InProgress""
        }
    ]
}
```

Amazon Glacier에서는 작업을 수행할 때 계정 ID 인수가 필요하지만 하이픈을 사용하여 사용 중인 계정을 지정할 수 있습니다.

- API 세부 정보는 AWS CLI 명령 참조의 ListJobs를 참조하세요.

Python

SDK for Python(Boto3)

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

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def list_jobs(vault, job_type):
        """
        Lists jobs by type for the specified vault.

        :param vault: The vault to query.
        :param job_type: The type of job to list.
        :return: The list of jobs of the requested type.
        """
        job_list = []
        try:
            if job_type == "all":
                jobs = vault.jobs.all()
            elif job_type == "in_progress":
                jobs = vault.jobs_in_progress.all()
            elif job_type == "completed":
                jobs = vault.completed_jobs.all()
```

```
            elif job_type == "succeeded":
                jobs = vault.succeeded_jobs.all()
            elif job_type == "failed":
                jobs = vault.failed_jobs.all()
            else:
                jobs = []
                logger.warning("%s isn't a type of job I can get.", job_type)
            for job in jobs:
                job_list.append(job)
                logger.info("Got %s %s job %s.", job_type, job.action, job.id)
        except ClientError:
            logger.exception("Couldn't get %s jobs from %s.", job_type,
vault.name)
            raise
        else:
            return job_list
```

- API 세부 정보는 AWS SDK for Python(Boto3) API 참조의 ListJobs를 참조하세요.

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

AWS SDK 또는 CLI와 `ListTagsForVault` 함께 사용

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

.NET

SDK for .NET

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

```
    /// <summary>
    /// List tags for an Amazon S3 Glacier vault.
```

```
    /// </summary>
    /// <param name="vaultName">The name of the vault to list tags for.</param>
    /// <returns>A dictionary listing the tags attached to each object in the
    /// vault and its tags.</returns>
    public async Task<Dictionary<string, string>> ListTagsForVaultAsync(string
 vaultName)
    {
        var request = new ListTagsForVaultRequest
        {
            // Using a hyphen "-" for the Account Id will
            // cause the SDK to use the Account Id associated
            // with the default user.
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await _glacierService.ListTagsForVaultAsync(request);

        return response.Tags;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ListTagsForVault를 참조하세요.

CLI

AWS CLI

다음 명령은 `my-vault`라는 볼트에 적용된 태그를 나열합니다.

```
aws glacier list-tags-for-vault --account-id - --vault-name my-vault
```

출력:

```
{
    "Tags": {
        "date": "july2015",
        "id": "1234"
    }
}
```

Amazon Glacier에서는 작업을 수행할 때 계정 ID 인수가 필요하지만 하이픈을 사용하여 사용 중인 계정을 지정할 수 있습니다.

- API 세부 정보는 AWS CLI 명령 참조의 ListTagsForVault 섹션을 참조하세요.

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

AWS SDK 또는 CLI와 `ListVaults` 함께 사용

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

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 파일을 보관하고, 알림을 받고 작업을 시작

.NET

SDK for .NET

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

```
/// <summary>
/// List the Amazon S3 Glacier vaults associated with the current account.
/// </summary>
/// <returns>A list containing information about each vault.</returns>
public async Task<List<DescribeVaultOutput>> ListVaultsAsync()
{
    var glacierVaultPaginator = _glacierService.Paginators.ListVaults(
        new ListVaultsRequest { AccountId = "-" });
    var vaultList = new List<DescribeVaultOutput>();

    await foreach (var vault in glacierVaultPaginator.VaultList)
    {
```

```
            vaultList.Add(vault);
        }

        return vaultList;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ListVaults를 참조하세요.

CLI

AWS CLI

다음 명령은 기본 계정 및 리전 내 볼트를 나열합니다.

```
aws glacier list-vaults --account-id -
```

출력:

```
{
    "VaultList": [
        {
            "SizeInBytes": 3178496,
            "VaultARN": "arn:aws:glacier:us-west-2:0123456789012:vaults/my-
vault",
            "LastInventoryDate": "2015-04-07T00:26:19.028Z",
            "VaultName": "my-vault",
            "NumberOfArchives": 1,
            "CreationDate": "2015-04-06T21:23:45.708Z"
        }
    ]
}
```

Amazon Glacier에서는 작업을 수행할 때 계정 ID 인수가 필요하지만 하이픈을 사용하여 사용 중인 계정을 지정할 수 있습니다.

- API 세부 정보는 AWS CLI 명령 참조의 ListVaults를 참조하세요.

Java

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.model.ListVaultsRequest;
import software.amazon.awssdk.services.glacier.model.ListVaultsResponse;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DescribeVaultOutput;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.util.List;

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

        listAllVault(glacier);
        glacier.close();
    }

    public static void listAllVault(GlacierClient glacier) {
        boolean listComplete = false;
        String newMarker = null;
        int totalVaults = 0;
        System.out.println("Your Amazon Glacier vaults:");
```

```
        try {
            while (!listComplete) {
                ListVaultsResponse response = null;
                if (newMarker != null) {
                    ListVaultsRequest request = ListVaultsRequest.builder()
                            .marker(newMarker)
                            .build();

                    response = glacier.listVaults(request);
                } else {
                    ListVaultsRequest request = ListVaultsRequest.builder()
                            .build();
                    response = glacier.listVaults(request);
                }

                List<DescribeVaultOutput> vaultList = response.vaultList();
                for (DescribeVaultOutput v : vaultList) {
                    totalVaults += 1;
                    System.out.println("* " + v.vaultName());
                }

                // Check for further results.
                newMarker = response.marker();
                if (newMarker == null) {
                    listComplete = true;
                }
            }

            if (totalVaults == 0) {
                System.out.println("No vaults found.");
            }

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

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

Python

SDK for Python (Boto3)

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

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    def list_vaults(self):
        """
        Lists vaults for the current account.
        """
        try:
            for vault in self.glacier_resource.vaults.all():
                logger.info("Got vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't list vaults.")
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ListVaults를 참조하세요.

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

AWS SDK 또는 CLI와 **SetVaultNotifications** 함께 사용

다음 코드 예시는 SetVaultNotifications의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 파일을 보관하고, 알림을 받고 작업을 시작

CLI

AWS CLI

다음 명령은 my-vault라는 볼트에 대한 SNS 알림을 구성합니다.

```
aws glacier set-vault-notifications --account-id - --vault-name my-vault --vault-notification-config file://notificationconfig.json
```

notificationconfig.json은 게시할 SNS 주제와 이벤트를 지정하는 현재 폴더의 JSON 파일입니다.

```
{
  "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-vault",
  "Events": ["ArchiveRetrievalCompleted", "InventoryRetrievalCompleted"]
}
```

Amazon Glacier에서는 작업을 수행할 때 계정 ID 인수가 필요하지만 하이픈을 사용하여 사용 중인 계정을 지정할 수 있습니다.

- API 세부 정보는 AWS CLI 명령 참조의 SetVaultNotifications를 참조하세요.

Python

SDK for Python(Boto3)

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

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    def set_notifications(self, vault, sns_topic_arn):
        """
        Sets an Amazon Simple Notification Service (Amazon SNS) topic as a target
        for notifications. Amazon S3 Glacier publishes messages to this topic for
        the configured list of events.

        :param vault: The vault to set up to publish notifications.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of the topic that
                              receives notifications.
        :return: Data about the new notification configuration.
        """
        try:
            notification = self.glacier_resource.Notification("-", vault.name)
            notification.set(
                vaultNotificationConfig={
                    "SNSTopic": sns_topic_arn,
                    "Events": [
                        "ArchiveRetrievalCompleted",
                        "InventoryRetrievalCompleted",
                    ],
                }
            )
            logger.info(
                "Notifications will be sent to %s for events %s from %s.",
                notification.sns_topic,
                notification.events,
                notification.vault_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't set notifications to %s on %s.", sns_topic_arn,
 vault.name
            )
```

```
            raise
        else:
            return notification
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 SetVaultNotifications를 참조하세요.

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

AWS SDK 또는 CLI와 **UploadArchive** 함께 사용

다음 코드 예시는 UploadArchive의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 파일을 보관하고, 알림을 받고 작업을 시작

.NET

SDK for .NET

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

```
    /// <summary>
    /// Upload an object to an Amazon S3 Glacier vault.
    /// </summary>
    /// <param name="vaultName">The name of the Amazon S3 Glacier vault to upload
    /// the archive to.</param>
    /// <param name="archiveFilePath">The file path of the archive to upload to
the vault.</param>
```

```
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<string> UploadArchiveWithArchiveManager(string vaultName,
 string archiveFilePath)
    {
        try
        {
            var manager = new ArchiveTransferManager(_glacierService);

            // Upload an archive.
            var response = await manager.UploadAsync(vaultName, "upload archive
 test", archiveFilePath);
            return response.ArchiveId;
        }
        catch (AmazonGlacierException ex)
        {
            Console.WriteLine(ex.Message);
            return string.Empty;
        }
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 UploadArchive를 참조하세요.

CLI

AWS CLI

다음 명령은 `archive.zip`이라는 현재 폴더의 아카이브를 `my-vault`라는 볼트에 업로드합니다.

```
aws glacier upload-archive --account-id - --vault-name my-vault --
body archive.zip
```

출력:

```
{
    "archiveId": "kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--
zM_mw6k76ZFGEIWQX-ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw",
    "checksum":
 "969fb39823836d81f0cc028195fcdbcbbe76cdde932d4646fa7de5f21e18aa67",
```

```
    "location": "/0123456789012/vaults/my-vault/archives/
kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--zM_mw6k76ZFGEIWQX-
ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw"
}
```

Amazon Glacier에서는 작업을 수행할 때 계정 ID 인수가 필요하지만 하이픈을 사용하여 사용 중인 계정을 지정할 수 있습니다.

업로드된 아카이브를 검색하려면 aws glacier initiate-job 명령을 사용하여 검색 작업을 시작하세요.

- API 세부 정보는 AWS CLI 명령 참조의 UploadArchive를 참조하세요.

Java

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.UploadArchiveRequest;
import software.amazon.awssdk.services.glacier.model.UploadArchiveResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.io.File;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class UploadArchive {

    static final int ONE_MB = 1024 * 1024;

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

                Usage:   <strPath> <vaultName>\s

                Where:
                   strPath - The path to the archive to upload (for example, C:\
\AWS\\test.pdf).
                   vaultName - The name of the vault.
                """;

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

        String strPath = args[0];
        String vaultName = args[1];
        File myFile = new File(strPath);
        Path path = Paths.get(strPath);
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String archiveId = uploadContent(glacier, path, vaultName, myFile);
        System.out.println("The ID of the archived item is " + archiveId);
        glacier.close();
    }

    public static String uploadContent(GlacierClient glacier, Path path, String
 vaultName, File myFile) {
        // Get an SHA-256 tree hash value.
        String checkVal = computeSHA256(myFile);
        try {
            UploadArchiveRequest uploadRequest = UploadArchiveRequest.builder()
                    .vaultName(vaultName)
```

```
                    .checksum(checkVal)
                    .build();

            UploadArchiveResponse res = glacier.uploadArchive(uploadRequest,
path);
            return res.archiveId();

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    private static String computeSHA256(File inputFile) {
        try {
            byte[] treeHash = computeSHA256TreeHash(inputFile);
            System.out.printf("SHA-256 tree hash = %s\n", toHex(treeHash));
            return toHex(treeHash);

        } catch (IOException ioe) {
            System.err.format("Exception when reading from file %s: %s",
inputFile, ioe.getMessage());
            System.exit(-1);

        } catch (NoSuchAlgorithmException nsae) {
            System.err.format("Cannot locate MessageDigest algorithm for SHA-256:
%s", nsae.getMessage());
            System.exit(-1);
        }
        return "";
    }

    public static byte[] computeSHA256TreeHash(File inputFile) throws
IOException,
            NoSuchAlgorithmException {

        byte[][] chunkSHA256Hashes = getChunkSHA256Hashes(inputFile);
        return computeSHA256TreeHash(chunkSHA256Hashes);
    }

    /**
     * Computes an SHA256 checksum for each 1 MB chunk of the input file. This
     * includes the checksum for the last chunk, even if it's smaller than 1 MB.
```

```
     */
    public static byte[][] getChunkSHA256Hashes(File file) throws IOException,
            NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        long numChunks = file.length() / ONE_MB;
        if (file.length() % ONE_MB > 0) {
            numChunks++;
        }

        if (numChunks == 0) {
            return new byte[][] { md.digest() };
        }

        byte[][] chunkSHA256Hashes = new byte[(int) numChunks][];
        FileInputStream fileStream = null;

        try {
            fileStream = new FileInputStream(file);
            byte[] buff = new byte[ONE_MB];

            int bytesRead;
            int idx = 0;

            while ((bytesRead = fileStream.read(buff, 0, ONE_MB)) > 0) {
                md.reset();
                md.update(buff, 0, bytesRead);
                chunkSHA256Hashes[idx++] = md.digest();
            }

            return chunkSHA256Hashes;

        } finally {
            if (fileStream != null) {
                try {
                    fileStream.close();
                } catch (IOException ioe) {
                    System.err.printf("Exception while closing %s.\n %s",
file.getName(),
                            ioe.getMessage());
                }
            }
        }
    }
```

```
/**
 * Computes the SHA-256 tree hash for the passed array of 1 MB chunk
 * checksums.
 */
public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
        throws NoSuchAlgorithmException {

    MessageDigest md = MessageDigest.getInstance("SHA-256");
    byte[][] prevLvlHashes = chunkSHA256Hashes;
    while (prevLvlHashes.length > 1) {
        int len = prevLvlHashes.length / 2;
        if (prevLvlHashes.length % 2 != 0) {
            len++;
        }

        byte[][] currLvlHashes = new byte[len][];
        int j = 0;
        for (int i = 0; i < prevLvlHashes.length; i = i + 2, j++) {

            // If there are at least two elements remaining.
            if (prevLvlHashes.length - i > 1) {

                // Calculate a digest of the concatenated nodes.
                md.reset();
                md.update(prevLvlHashes[i]);
                md.update(prevLvlHashes[i + 1]);
                currLvlHashes[j] = md.digest();

            } else { // Take care of the remaining odd chunk
                currLvlHashes[j] = prevLvlHashes[i];
            }
        }

        prevLvlHashes = currLvlHashes;
    }

    return prevLvlHashes[0];
}

/**
 * Returns the hexadecimal representation of the input byte array
 */
public static String toHex(byte[] data) {
```

```
        StringBuilder sb = new StringBuilder(data.length * 2);
        for (byte datum : data) {
            String hex = Integer.toHexString(datum & 0xFF);

            if (hex.length() == 1) {
                // Append leading zero.
                sb.append("0");
            }
            sb.append(hex);
        }
        return sb.toString().toLowerCase();
    }
}
```

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

JavaScript

SDK for JavaScript (v3)

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

클라이언트를 생성합니다.

```
const { GlacierClient } = require("@aws-sdk/client-glacier");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const glacierClient = new GlacierClient({ region: REGION });
export { glacierClient };
```

아카이브를 업로드합니다.

```
// Load the SDK for JavaScript
import { UploadArchiveCommand } from "@aws-sdk/client-glacier";
```

```
import { glacierClient } from "./libs/glacierClient.js";

// Set the parameters
const vaultname = "VAULT_NAME"; // VAULT_NAME

// Create a new service object and buffer
const buffer = new Buffer.alloc(2.5 * 1024 * 1024); // 2.5MB buffer
const params = { vaultName: vaultname, body: buffer };

const run = async () => {
  try {
    const data = await glacierClient.send(new UploadArchiveCommand(params));
    console.log("Archive ID", data.archiveId);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error uploading archive!", err);
  }
};
run();
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 UploadArchive를 참조하십시오.

SDK for JavaScript (v2)

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

```
// Load the SDK for JavaScript
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create a new service object and buffer
var glacier = new AWS.Glacier({ apiVersion: "2012-06-01" });
buffer = Buffer.alloc(2.5 * 1024 * 1024); // 2.5MB buffer

var params = { vaultName: "YOUR_VAULT_NAME", body: buffer };
```

```
// Call Glacier to upload the archive.
glacier.uploadArchive(params, function (err, data) {
  if (err) {
    console.log("Error uploading archive!", err);
  } else {
    console.log("Archive ID", data.archiveId);
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 UploadArchive를 참조하세요.

PowerShell

Tools for PowerShell V4

예제 1: 단일 파일을 지정된 볼트에 업로드하여 아카이브 ID와 계산된 체크섬을 반환합니다.

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

출력:

```
FilePath                  ArchiveId             Checksum
--------                  ---------             --------
C:\temp\blue.bin          o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
```

예제 2: 폴더 계층 구조의 콘텐츠를 사용자 계정의 지정된 볼트에 업로드합니다. 업로드된 각 파일에 대해 cmdlet은 파일 이름, 해당 아카이브 ID, 아카이브의 계산된 체크섬을 내보냅니다.

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

출력:

```
FilePath                  ArchiveId             Checksum
--------                  ---------             --------
C:\temp\blue.bin          o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
C:\temp\green.bin         qXAfOdSG...czo729UHXrw d50a1...9184b9
C:\temp\lum.bin           39aNifP3...q9nb8nZkFIg 28886...5c3e27
```

```
C:\temp\red.bin             vp7E6rU_...Ejk_HhjAxKA e05f7...4e34f5
C:\temp\Folder1\file1.txt   _eRINlip...5Sxy7dD2BaA d0d2a...c8a3ba
C:\temp\Folder2\file2.iso   -Ix3jlmu...iXiDh-XfOPA 7469e...3e86f1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UploadArchive를 참조하세요.

Tools for PowerShell V5

예제 1: 단일 파일을 지정된 볼트에 업로드하여 아카이브 ID와 계산된 체크섬을 반환합니다.

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

출력:

```
FilePath                    ArchiveId             Checksum
--------                    ---------             --------
C:\temp\blue.bin            o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
```

예제 2: 폴더 계층 구조의 콘텐츠를 사용자 계정의 지정된 볼트에 업로드합니다. 업로드된 각 파일에 대해 cmdlet은 파일 이름, 해당 아카이브 ID, 아카이브의 계산된 체크섬을 내보냅니다.

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

출력:

```
FilePath                    ArchiveId             Checksum
--------                    ---------             --------
C:\temp\blue.bin            o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
C:\temp\green.bin           qXAfOdSG...czo729UHXrw d50a1...9184b9
C:\temp\lum.bin             39aNifP3...q9nb8nZkFIg 28886...5c3e27
C:\temp\red.bin             vp7E6rU_...Ejk_HhjAxKA e05f7...4e34f5
C:\temp\Folder1\file1.txt   _eRINlip...5Sxy7dD2BaA d0d2a...c8a3ba
C:\temp\Folder2\file2.iso   -Ix3jlmu...iXiDh-XfOPA 7469e...3e86f1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 UploadArchive를 참조하세요.

Python

SDK for Python(Boto3)

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

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def upload_archive(vault, archive_description, archive_file):
        """
        Uploads an archive to a vault.

        :param vault: The vault where the archive is put.
        :param archive_description: A description of the archive.
        :param archive_file: The archive file to put in the vault.
        :return: The uploaded archive.
        """
        try:
            archive = vault.upload_archive(
                archiveDescription=archive_description, body=archive_file
            )
            logger.info(
                "Uploaded %s with ID %s to vault %s.",
                archive_description,
                archive.id,
                vault.name,
            )
        except ClientError:
            logger.exception(
```

```
                "Couldn't upload %s to %s.", archive_description, vault.name
            )
            raise
        else:
            return archive
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 UploadArchive를 참조하세요.

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

AWS SDK 또는 CLI와 `UploadMultipartPart` 함께 사용

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

CLI

AWS CLI

다음 명령은 아카이브의 첫 번째 1MiB(1024 x 1024바이트) 부분을 업로드합니다.

```
aws glacier upload-multipart-part --body part1 --range 'bytes
 0-1048575/*' --account-id - --vault-name my-vault --upload-
id 19gaRezEXAMPLES6Ry5YYdqthHOC_kGRCT03L9yetr220UmPtBYKk-
OssZtLqyFu7sY1_lR7vgFuJV6NtcV5zpsJ
```

Amazon Glacier에서는 작업을 수행할 때 계정 ID 인수가 필요하지만 하이픈을 사용하여 사용 중인 계정을 지정할 수 있습니다.

본문 파라미터는 로컬 파일 시스템의 부분 파일 경로를 사용합니다. 범위 파라미터는 완성된 아카이브에서 부분이 차지하는 바이트를 나타내는 HTTP 콘텐츠 범위를 사용합니다. 업로드 ID는 `aws glacier initiate-multipart-upload` 명령으로 반환되며 `aws glacier list-multipart-uploads`를 사용하여 가져올 수도 있습니다.

AWS CLI를 사용하여 Amazon Glacier에 멀티파트 업로드하는 방법에 대한 자세한 내용은 AWS CLI 사용 설명서의 Amazon Glacier 사용을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 UploadMultipartPart를 참조하세요.

JavaScript

SDK for JavaScript(v2)

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

버퍼 객체의 1메가바이트 청크 멀티파트 업로드를 생성합니다.

```
// Create a new service object and some supporting variables
var glacier = new AWS.Glacier({ apiVersion: "2012-06-01" }),
  vaultName = "YOUR_VAULT_NAME",
  buffer = new Buffer(2.5 * 1024 * 1024), // 2.5MB buffer
  partSize = 1024 * 1024, // 1MB chunks,
  numPartsLeft = Math.ceil(buffer.length / partSize),
  startTime = new Date(),
  params = { vaultName: vaultName, partSize: partSize.toString() };

// Compute the complete SHA-256 tree hash so we can pass it
// to completeMultipartUpload request at the end
var treeHash = glacier.computeChecksums(buffer).treeHash;

// Initiate the multipart upload
console.log("Initiating upload to", vaultName);
// Call Glacier to initiate the upload.
glacier.initiateMultipartUpload(params, function (mpErr, multipart) {
  if (mpErr) {
    console.log("Error!", mpErr.stack);
    return;
  }
  console.log("Got upload ID", multipart.uploadId);

  // Grab each partSize chunk and upload it as a part
  for (var i = 0; i < buffer.length; i += partSize) {
    var end = Math.min(i + partSize, buffer.length),
      partParams = {
        vaultName: vaultName,
        uploadId: multipart.uploadId,
        range: "bytes " + i + "-" + (end - 1) + "/*",
        body: buffer.slice(i, end),
```

```
      };

    // Send a single part
    console.log("Uploading part", i, "=", partParams.range);
    glacier.uploadMultipartPart(partParams, function (multiErr, mData) {
      if (multiErr) return;
      console.log("Completed part", this.request.params.range);
      if (--numPartsLeft > 0) return; // complete only when all parts uploaded

      var doneParams = {
        vaultName: vaultName,
        uploadId: multipart.uploadId,
        archiveSize: buffer.length.toString(),
        checksum: treeHash, // the computed tree hash
      };

      console.log("Completing upload...");
      glacier.completeMultipartUpload(doneParams, function (err, data) {
        if (err) {
          console.log("An error occurred while uploading the archive");
          console.log(err);
        } else {
          var delta = (new Date() - startTime) / 1000;
          console.log("Completed upload in", delta, "seconds");
          console.log("Archive ID:", data.archiveId);
          console.log("Checksum:  ", data.checksum);
        }
      });
    });
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 UploadMultipartPart를 참조하세요.

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

AWS SDKs를 사용한 Amazon Glacier 시나리오

다음 코드 예제에서는 Amazon Glacier AWS SDKs에서 일반적인 시나리오를 구현하는 방법을 보여줍니다. 이러한 시나리오에서는 Amazon Glacier 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여줍니다. 각 시나리오에는 전체 소스 코드에 대한 링크가 포함되어 있습니다. 여기에서 코드를 설정 및 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시나리오는 컨텍스트에 맞는 서비스 작업을 이해하는 데 도움이 되도록 중급 수준의 경험을 대상으로 합니다.

예시

- 파일을 Amazon Glacier에 아카이브하고, 알림을 받고, AWS SDK를 사용하여 작업을 시작합니다.
- AWS SDK를 사용하여 Amazon Glacier 아카이브 콘텐츠 가져오기 및 아카이브 삭제

파일을 Amazon Glacier에 아카이브하고, 알림을 받고, AWS SDK를 사용하여 작업을 시작합니다.

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- Amazon Glacier 볼트를 생성합니다.
- Amazon SNS 토픽에 알림을 게시하도록 볼트를 구성합니다.
- 볼트에 아카이브 파일을 업로드합니다.
- 아카이브 가져오기 작업을 시작합니다.

Python

SDK for Python(Boto3)

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

Amazon Glacier 작업을 래핑하는 클래스를 만듭니다.

```
import argparse
import logging
import os
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    def create_vault(self, vault_name):
        """
        Creates a vault.

        :param vault_name: The name to give the vault.
        :return: The newly created vault.
        """
        try:
            vault = self.glacier_resource.create_vault(vaultName=vault_name)
            logger.info("Created vault %s.", vault_name)
        except ClientError:
            logger.exception("Couldn't create vault %s.", vault_name)
            raise
        else:
            return vault


    def list_vaults(self):
        """
        Lists vaults for the current account.
        """
        try:
            for vault in self.glacier_resource.vaults.all():
                logger.info("Got vault %s.", vault.name)
```

```
        except ClientError:
            logger.exception("Couldn't list vaults.")
            raise


    @staticmethod
    def upload_archive(vault, archive_description, archive_file):
        """
        Uploads an archive to a vault.

        :param vault: The vault where the archive is put.
        :param archive_description: A description of the archive.
        :param archive_file: The archive file to put in the vault.
        :return: The uploaded archive.
        """
        try:
            archive = vault.upload_archive(
                archiveDescription=archive_description, body=archive_file
            )
            logger.info(
                "Uploaded %s with ID %s to vault %s.",
                archive_description,
                archive.id,
                vault.name,
            )
        except ClientError:
            logger.exception(
                "Couldn't upload %s to %s.", archive_description, vault.name
            )
            raise
        else:
            return archive


    @staticmethod
    def initiate_archive_retrieval(archive):
        """
        Initiates an archive retrieval job. Standard retrievals typically
complete
        within 3–5 hours. When the job completes, you can get the archive
contents
        by calling get_output().

        :param archive: The archive to retrieve.
```

```
        :return: The archive retrieval job.
        """
        try:
            job = archive.initiate_archive_retrieval()
            logger.info("Started %s job with ID %s.", job.action, job.id)
        except ClientError:
            logger.exception("Couldn't start job on archive %s.", archive.id)
            raise
        else:
            return job


    @staticmethod
    def list_jobs(vault, job_type):
        """
        Lists jobs by type for the specified vault.

        :param vault: The vault to query.
        :param job_type: The type of job to list.
        :return: The list of jobs of the requested type.
        """
        job_list = []
        try:
            if job_type == "all":
                jobs = vault.jobs.all()
            elif job_type == "in_progress":
                jobs = vault.jobs_in_progress.all()
            elif job_type == "completed":
                jobs = vault.completed_jobs.all()
            elif job_type == "succeeded":
                jobs = vault.succeeded_jobs.all()
            elif job_type == "failed":
                jobs = vault.failed_jobs.all()
            else:
                jobs = []
                logger.warning("%s isn't a type of job I can get.", job_type)
            for job in jobs:
                job_list.append(job)
                logger.info("Got %s %s job %s.", job_type, job.action, job.id)
        except ClientError:
            logger.exception("Couldn't get %s jobs from %s.", job_type,
vault.name)
            raise
        else:
```

```
            return job_list


    def set_notifications(self, vault, sns_topic_arn):
        """
        Sets an Amazon Simple Notification Service (Amazon SNS) topic as a target
        for notifications. Amazon S3 Glacier publishes messages to this topic for
        the configured list of events.

        :param vault: The vault to set up to publish notifications.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of the topic that
                              receives notifications.
        :return: Data about the new notification configuration.
        """
        try:
            notification = self.glacier_resource.Notification("-", vault.name)
            notification.set(
                vaultNotificationConfig={
                    "SNSTopic": sns_topic_arn,
                    "Events": [
                        "ArchiveRetrievalCompleted",
                        "InventoryRetrievalCompleted",
                    ],
                }
            )
            logger.info(
                "Notifications will be sent to %s for events %s from %s.",
                notification.sns_topic,
                notification.events,
                notification.vault_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't set notifications to %s on %s.", sns_topic_arn,
vault.name
            )
            raise
        else:
            return notification
```

래퍼 클래스의 함수를 직접 호출하여 볼트를 만들고 파일을 업로드한 다음, 볼트를 구성하여 알림을 게시하고 아카이브 가져오기 작업을 시작합니다.

```
def upload_demo(glacier, vault_name, topic_arn):
    """
    Shows how to:
    * Create a vault.
    * Configure the vault to publish notifications to an Amazon SNS topic.
    * Upload an archive.
    * Start a job to retrieve the archive.

    :param glacier: A Boto3 Amazon S3 Glacier resource.
    :param vault_name: The name of the vault to create.
    :param topic_arn: The ARN of an Amazon SNS topic that receives notification
 of
                      Amazon S3 Glacier events.
    """
    print(f"\nCreating vault {vault_name}.")
    vault = glacier.create_vault(vault_name)
    print("\nList of vaults in your account:")
    glacier.list_vaults()
    print(f"\nUploading glacier_basics.py to {vault.name}.")
    with open("glacier_basics.py", "rb") as upload_file:
        archive = glacier.upload_archive(vault, "glacier_basics.py", upload_file)
    print(
        "\nStarting an archive retrieval request to get the file back from the "
        "vault."
    )
    glacier.initiate_archive_retrieval(archive)
    print("\nListing in progress jobs:")
    glacier.list_jobs(vault, "in_progress")
    print(
        "\nBecause Amazon S3 Glacier is intended for infrequent retrieval, an "
        "archive request with Standard retrieval typically completes within 3-5 "
        "hours."
    )
    if topic_arn:
        notification = glacier.set_notifications(vault, topic_arn)
        print(
            f"\nVault {vault.name} is configured to notify the "
            f"{notification.sns_topic} topic when {notification.events} "
            f"events occur. You can subscribe to this topic to receive "
            f"a message when the archive retrieval completes.\n"
```

```
        )
    else:
        print(
            f"\nVault {vault.name} is not configured to notify an Amazon SNS
topic "
            f"when the archive retrieval completes so wait a few hours."
        )
    print("\nRetrieve your job output by running this script with the --retrieve
flag.")
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하세요.
  - CreateVault
  - InitiateJob
  - ListJobs
  - ListVaults
  - SetVaultNotifications
  - UploadArchive

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

AWS SDK를 사용하여 Amazon Glacier 아카이브 콘텐츠 가져오기 및 아카이브 삭제

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- Amazon Glacier 볼트에 대한 작업을 나열하고 작업 상태를 확인합니다.
- 완료된 아카이브 가져오기 작업의 출력을 가져옵니다.
- 아카이브를 삭제합니다.
- 볼트를 삭제합니다.

Python

SDK for Python(Boto3)

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

Amazon Glacier 작업을 래핑하는 클래스를 만듭니다.

```
import argparse
import logging
import os
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def list_jobs(vault, job_type):
        """
        Lists jobs by type for the specified vault.

        :param vault: The vault to query.
        :param job_type: The type of job to list.
        :return: The list of jobs of the requested type.
        """
        job_list = []
        try:
            if job_type == "all":
```

```
                jobs = vault.jobs.all()
            elif job_type == "in_progress":
                jobs = vault.jobs_in_progress.all()
            elif job_type == "completed":
                jobs = vault.completed_jobs.all()
            elif job_type == "succeeded":
                jobs = vault.succeeded_jobs.all()
            elif job_type == "failed":
                jobs = vault.failed_jobs.all()
            else:
                jobs = []
                logger.warning("%s isn't a type of job I can get.", job_type)
            for job in jobs:
                job_list.append(job)
                logger.info("Got %s %s job %s.", job_type, job.action, job.id)
        except ClientError:
            logger.exception("Couldn't get %s jobs from %s.", job_type,
vault.name)
            raise
        else:
            return job_list


    @staticmethod
    def get_job_output(job):
        """
        Gets the output of a job, such as a vault inventory or the contents of an
        archive.

        :param job: The job to get output from.
        :return: The job output, in bytes.
        """
        try:
            response = job.get_output()
            out_bytes = response["body"].read()
            logger.info("Read %s bytes from job %s.", len(out_bytes), job.id)
            if "archiveDescription" in response:
                logger.info(
                    "These bytes are described as '%s'",
response["archiveDescription"]
                )
        except ClientError:
            logger.exception("Couldn't get output for job %s.", job.id)
            raise
```

```
        else:
            return out_bytes


    @staticmethod
    def delete_archive(archive):
        """
        Deletes an archive from a vault.

        :param archive: The archive to delete.
        """
        try:
            archive.delete()
            logger.info(
                "Deleted archive %s from vault %s.", archive.id,
 archive.vault_name
            )
        except ClientError:
            logger.exception("Couldn't delete archive %s.", archive.id)
            raise


    @staticmethod
    def delete_vault(vault):
        """
        Deletes a vault.

        :param vault: The vault to delete.
        """
        try:
            vault.delete()
            logger.info("Deleted vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't delete vault %s.", vault.name)
            raise
```

래퍼 클래스의 함수를 직접 호출하여 완료된 작업에서 아카이브 콘텐츠를 가져온 다음 아카이브를 삭제합니다.

```
def retrieve_demo(glacier, vault_name):
    """
```

```
    Shows how to:
    * List jobs for a vault and get job status.
    * Get the output of a completed archive retrieval job.
    * Delete an archive.
    * Delete a vault.

    :param glacier: A Boto3 Amazon S3 Glacier resource.
    :param vault_name: The name of the vault to query for jobs.
    """
    vault = glacier.glacier_resource.Vault("-", vault_name)
    try:
        vault.load()
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceNotFoundException":
            print(
                f"\nVault {vault_name} doesn't exist. You must first run this
script "
                f"with the --upload flag to create the vault."
            )
            return
        else:
            raise

    print(f"\nGetting completed jobs for {vault.name}.")
    jobs = glacier.list_jobs(vault, "completed")
    if not jobs:
        print("\nNo completed jobs found. Give it some time and try again
later.")
        return

    retrieval_job = None
    for job in jobs:
        if job.action == "ArchiveRetrieval" and job.status_code == "Succeeded":
            retrieval_job = job
            break
    if retrieval_job is None:
        print(
            "\nNo ArchiveRetrieval jobs found. Give it some time and try again "
            "later."
        )
        return

    print(f"\nGetting output from job {retrieval_job.id}.")
    archive_bytes = glacier.get_job_output(retrieval_job)
```

```
    archive_str = archive_bytes.decode("utf-8")
    print("\nGot archive data. Printing the first 10 lines.")
    print(os.linesep.join(archive_str.split(os.linesep)[:10]))

    print(f"\nDeleting the archive from {vault.name}.")
    archive = glacier.glacier_resource.Archive(
        "-", vault.name, retrieval_job.archive_id
    )
    glacier.delete_archive(archive)

    print(f"\nDeleting {vault.name}.")
    glacier.delete_vault(vault)
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하세요.
  - DeleteArchive
  - DeleteVault
  - GetJobOutput
  - ListJobs

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

Amazon Glacier의 보안

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

보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이를 클라우드의 보안과 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 는에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다 AWS 클라우드. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 서드 파티 감사자는 정기적으로 AWS 규정 준수 프로그램의 일환으로 보안 효과를 테스트하고 검증합니다. Amazon Glacier(Amazon Glacier)에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 제공 범위 내AWS 서비스를 참조하세요.
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 데이터의 민감도, 조직의 요건 및 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

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

주제

- Amazon Glacier에서 데이터 보호
- Amazon Glacier의 ID 및 액세스 관리(IAM)
- Amazon Glacier의 로깅 및 모니터링
- Amazon Glacier에 대한 규정 준수 확인
- Amazon Glacier의 복원성
- Amazon Glacier의 인프라 보안

Amazon Glacier에서 데이터 보호

Amazon Glacier(Amazon Glacier)는 데이터 보관 및 장기 백업을 위한 내구성이 매우 뛰어난 클라우드 스토리지를 제공합니다. Amazon Glacier는 99.999999999%의 내구성을 제공하도록 설계되었으며 엄격한 규제 요구 사항을 충족하도록 포괄적인 보안 및 규정 준수 기능을 제공합니다. Amazon Glacier는

여러 AWS 가용 영역(AZ)과 각 AZ내의 여러 디바이스에 데이터를 중복 저장합니다. Amazon Glacier는 내구성을 높이기 위해 업로드 성공을 확인하기 전 여러 AZ에 데이터를 동기식으로 저장합니다.

AWS 글로벌 클라우드 인프라에 대한 자세한 내용은 글로벌 인프라를 참조하세요.

데이터 보호를 위해 AWS 계정 보안 인증 정보를 보호하고 개별 사용자나 그룹, 역할 계정에 직무를 수행하는 데 필요한 권한만 부여하는 것을 권장합니다.

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 참조하세요.

주제

- 데이터 암호화
- 키 관리
- 인터네트워크 트래픽 개인 정보

데이터 암호화

데이터 보호란 전송 중(Amazon Glacier가 데이터를 내보내거나 받는 경우)과 저장 시(AWS 데이터 센터에 데이터가 저장된 동안)에 데이터를 보호하는 것을 말합니다. SSL(Secure Sockets Layer) 또는 클라이언트 측 암호화를 사용하여 Amazon Glacier로 직접 업로드되는 전송 중 데이터를 보호할 수 있습니다.

Amazon S3를 통해 Amazon Glacier에 액세스할 수도 있습니다. Amazon S3는 보관 목적으로 객체를 Amazon Glacier 스토리지 클래스로 전송할 수 있도록 Amazon S3 버킷에 대한 수명 주기 구성을 지원합니다. 수명 주기 정책을 통한 Amazon S3 및 Amazon Glacier 사이의 전송 중 데이터는 SSL을 사용하여 암호화됩니다.

Amazon Glacier 저장 데이터는 256비트 Advanced Encryption Standard(AES-256)를 사용하여 서버 측에서 자동으로 암호화되며, AWS가 관리하는 키를 사용합니다. 자체 키를 관리하려는 경우 Amazon Glacier에 데이터를 저장하기 전에 클라이언트 측 암호화를 사용할 수도 있습니다. Amazon S3 기본 암호화 설정 방법에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 Amazon S3 기본 암호화를 참조하세요.

키 관리

서버 측 암호화는 저장 데이터 암호화를 처리합니다. 즉, Amazon Glacier는 데이터 센터에 데이터를 쓰면서 암호화하고 해당 데이터에 사용자가 액세스할 때 자동으로 복호화합니다. 요청을 인증하기만 하면 액세스 권한을 갖게 되며, 데이터의 암호화 여부와 관계없이 액세스 방식에는 차이가 없습니다.

Amazon Glacier 저장 데이터는 AES-256을 사용하여 서버 측에서 자동으로 암호화되며 AWS가 관리하는 키를 사용합니다. 또한 추가 보안 조치로 AWS는 주기적으로 바뀌는 루트 키를 사용하여 키 자체를 암호화합니다.

인터네트워크 트래픽 개인 정보

네트워크를 통한 Amazon Glacier 액세스는 AWS에서 게시한 API를 통해 이루어집니다. 클라이언트가 Transport Layer Security(TLS) 1.2를 지원해야 합니다. TLS 1.3 이상을 권장합니다. 클라이언트는 DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Diffie-Hellman Ephemeral)와 같은 PFS(전달 완전 보안)가 포함된 암호 제품군도 지원해야 합니다. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다. 또한, 반드시 액세스 키 ID와 IAM 보안 주체와 관련된 비밀 액세스 키를 사용하여 요청에 서명하거나 AWS Security Token Service(AWS STS)를 사용하여 요청에 서명할 수 있는 임시 보안 인증 정보를 생성할 수 있습니다.

VPC 엔드포인트

Virtual Private Cloud(VPC) 엔드포인트를 사용하면 인터넷 게이트웨이, NAT 장치, VPN 연결 또는 AWS Direct Connect 연결 요청 없이 지원되는 AWS 서비스 및 AWS PrivateLink로 구동되는 VPC 엔드포인트 서비스에 VPC를 비공개로 연결할 수 있습니다. Amazon Glacier는 VPC 엔드포인트를 직접 지원하지는 않지만 Amazon S3로 통합된 스토리지 계층으로 Amazon Glacier에 액세스하는 경우 Amazon Simple Storage Service(Amazon S3) VPC 엔드포인트를 활용할 수 있습니다.

Amazon S3 수명 주기 구성 및 Amazon Glacier 스토리지 클래스로의 객체 전환에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 객체 수명 주기 관리 및 객체 전환을 참조하세요. VPC 엔드포인트에 대한 자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 참조하세요.

Amazon Glacier의 ID 및 액세스 관리(IAM)

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 AWS 서비스 되는 입니다. IAM 관리자는 어떤 사용자가 Amazon Glacier 리소스를

사용할 수 있도록 인증(로그인)되고 권한이 부여(권한 있음)될 수 있는지 제어합니다. IAM은 추가 비용 없이 사용할 수 AWS 서비스 있는 입니다.

주제

- 대상
- ID를 통한 인증
- 정책을 사용하여 액세스 관리
- Amazon Glacier가 IAM에서 작동하는 방식
- Amazon Glacier의 ID 기반 정책 예제
- Amazon Glacier용 리소스 기반 정책 예제
- Amazon Glacier ID 및 액세스 문제 해결
- API 권한 준거

대상

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

- 서비스 사용자 - 기능에 액세스할 수 없는 경우 관리자에게 권한 요청(참조Amazon Glacier ID 및 액세스 문제 해결)
- 서비스 관리자 - 사용자 액세스 결정 및 권한 요청 제출(Amazon Glacier가 IAM에서 작동하는 방식 참조)
- IAM 관리자 - 액세스를 관리하기 위한 정책 작성(Amazon Glacier의 ID 기반 정책 예제 참조)

ID를 통한 인증

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

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

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

AWS 계정 루트 사용자

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

페더레이션 ID

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

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

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center를 추천합니다. 자세한 정보는 AWS IAM Identity Center 사용 설명서의 What is IAM Identity Center?를 참조하세요.

IAM 사용자 및 그룹

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

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

IAM 역할

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

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

정책을 사용하여 액세스 관리

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

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

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

ID 기반 정책

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

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

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 예를 들어 IAM 역할 신뢰 정책 및 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다.

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

기타 정책 유형

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

- 권한 경계 - ID 기반 정책에서 IAM 엔터티에 부여할 수 있는 최대 권한을 설정합니다. 자세한 정보는 IAM 사용 설명서의 IAM 엔터티의 권한 범위를 참조하세요.
- 서비스 제어 정책(SCP) - AWS Organizations내 조직 또는 조직 단위에 대한 최대 권한을 지정합니다. 자세한 내용은AWS Organizations 사용 설명서의 서비스 제어 정책을 참조하세요.
- 리소스 제어 정책(RCP) – 계정의 리소스에 사용할 수 있는 최대 권한을 설정합니다. 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCP)을 참조하세요.
- 세션 정책 – 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 자세한 내용은 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

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

Amazon Glacier가 IAM에서 작동하는 방식

IAM을 사용하여 Amazon Glacier에 대한 액세스를 관리하기 전에, Amazon Glacier와 함께 사용할 수 있는 IAM 기능을 알아보세요.

Amazon Glacier에서 사용할 수 있는 IAM 기능

IAM 특성	Amazon Glacier 지원
자격 증명 기반 정책	예
리소스 기반 정책	예
정책 작업	예
정책 리소스	예
정책 조건 키(서비스별)	예
ACL	아니요
ABAC(정책 내 태그)	아니요

IAM 특성	Amazon Glacier 지원
임시 보안 인증	예
위탁자 권한	아니요
서비스 역할	아니요
서비스 연결 역할	아니요

Amazon Glacier 및 기타 AWS 서비스에서 대부분의 IAM 기능을 사용하는 방법을 개괄적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

Amazon Glacier의 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. JSON 정책에서 사용할 수 있는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

Amazon Glacier의 ID 기반 정책 예시

Amazon Glacier ID 기반 정책 예제를 보려면 Amazon Glacier의 ID 기반 정책 예제 섹션을 참조하세요.

Amazon Glacier 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 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는가 포함될 수 있습니다 AWS 서비스.

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

Amazon Glacier 서비스는 볼트에 연결되는 볼트 정책이라는 리소스 기반 정책의 한 가지 유형만 지원합니다. 이 정책은 볼트에서 작업을 수행할 수 있는 보안 주체를 정의합니다.

Amazon Glacier 볼트 정책을 사용하여 다음과 같이 권한을 관리할 수 있습니다.

- 다수의 개별 사용자 정책 대신 단일 볼트 정책을 사용하여 계정의 사용자 권한을 관리하세요.
- 교차 계정 권한을 관리하여 IAM 역할 사용을 대체하세요.

Amazon Glacier 내 리소스 기반 정책 예시

Amazon Glacier의 리소스 기반 정책 예시를 보려면 Amazon Glacier용 리소스 기반 정책 예제 섹션을 참조하세요.

Amazon Glacier의 정책 작업

정책 작업 지원: 예

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

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

GlacierGlacier의 작업 목록을 보려면 서비스 인증 참조의 Amazon Glacier가 정의한 작업을 참조하세요.

Amazon Glacier의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
glacier
```

단일 문에서 여러 작업을 지정하려면 쉼표로 구분합니다.

```
"Action": [
            "glacier:CreateVault",
```

```
            "glacier:DescribeVault",
            "glacier:ListVaults"
        ]
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, Describe라는 단어로 시작하는 모든 작업을 지정하려면 다음 작업을 포함합니다.

```
"Action": "glacier:GetVault*"
```

Amazon Glacier ID 기반 정책 예제를 보려면 Amazon Glacier의 ID 기반 정책 예제 섹션을 참조하세요.

Amazon Glacier의 정책 리소스

정책 리소스 지원: 예

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

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

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

Amazon Glacier의 리소스 유형 및 해당 ARN의 목록을 보려면 서비스 권한 부여 참조의 Amazon Glacier가 정의한 리소스를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 Amazon Glacier가 정의한 작업을 참조하세요.

Amazon Glacier에서 기본 리소스는 볼트입니다. Amazon Glacier는 볼트 수준에서만 정책을 지원합니다. 즉, IAM 정책에서 지정하는 Resource 값은 특정 볼트 또는 특정 AWS 리전의 볼트 집합일 수 있습니다. Amazon Glacier는 아카이브 수준의 권한을 지원하지 않습니다.

따라서 모든 Amazon Glacier 작업에서 Resource는 사용자가 권한을 부여하기 원하는 볼트를 지정하게 됩니다. 이러한 리소스들은 다음 표에 나온 것처럼 고유한 Amazon 리소스 이름(ARN)이 연결되어 있습니다. 그리고 ARN에서 같은 접두사를 사용하는 볼트 이름과 일치하도록 와일드카드 문자(*)를 사용할 수 있습니다.

Amazon Glacier는 Amazon Glacier 리소스를 처리하기 위한 작업 세트를 제공합니다. 가능한 작업에 대한 자세한 내용은 Amazon Glacier를 위한 API 참조 섹션을 참조하세요.

일부 Amazon Glacier API 작업은 여러 리소스를 지원합니다. 예를 들어 `glacier:AddTagsToVault`는 examplevault1과 examplevault2에 액세스하므로 보안 주체는 두 리소스에 모두 액세스할 수 있는 권한을 반드시 가져야 합니다. 단일 문에서 여러 리소스를 지정하려면 ARN을 쉼표로 구분합니다.

```
"Resource": [
""arn:aws:glacier:us-west-2:123456789012:vaults/examplevault1",",
""arn:aws:glacier:us-west-2:123456789012:vaults/examplevault2","
]
```

Amazon Glacier에 사용되는 정책 조건 키

서비스별 정책 조건 키 지원: 예

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

`Condition` 요소는 정의된 기준에 따라 문이 실행되는 시기를 지정합니다. 같음(equals) 또는 미만(less than)과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

Amazon Glacier 조건 키 목록을 보려면 서비스 인증 참조의 Amazon Glacier용 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 Amazon Glacier가 정의한 작업을 참조하세요.

Glacier 전용 조건 키 사용 예시는 볼트 잠금 정책 섹션을 참조하세요.

Amazon Glacier의 ACL

ACL 지원: 아니요

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 Glacier를 사용한 ABAC

ABAC 지원(정책의 태그): 아니요

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

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

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

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

Amazon Glacier에서 임시 자격 증명 사용

임시 자격 증명 지원: 예

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

Amazon Glacier를 위한 서비스 간 위탁자 권한

전달 액세스 세션(FAS) 지원: 아니요

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

Amazon Glacier의 서비스 역할

서비스 역할 지원: 아니요

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 서비스 AWS에 권한을 위임할 역할 생성을 참조하세요.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 Amazon Glacier 기능이 중단될 수 있습니다. Amazon Glacier가 지침을 제공하는 경우에만 서비스 역할을 편집합니다.

Amazon Glacier를 위한 서비스 연결 역할

서비스 연결 역할 지원: 아니요

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

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

Amazon Glacier의 ID 기반 정책 예제

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

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

각 리소스 유형에 대한 ARN 형식을 포함하여 Amazon Glacier가 정의한 작업 및 리소스 유형에 대한 자세한 내용은 서비스 인증 참조의 Amazon Glacier를 위한 작업, 리소스, 조건 키를 참조하세요.

다음은 `us-west-2` AWS 리전의 모든 볼트를 식별하는 Amazon 리소스 이름(ARN)을 사용하여 리소스에 대한 세 가지 Amazon Glacier 볼트 관련 작업(,`glacier:CreateVault` `glacier:DescribeVault` 및)에 대한 권한을 부여하는 정책 예제입니다. `glacier:ListVaults` ARNs AWS 리소스를 고유하게 식별합니다. Amazon Glacier에서 사용하는 ARN에 대한 자세한 내용은 Amazon Glacier의 정책 리소스 섹션을 참조하세요.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
         "glacier:CreateVault",
         "glacier:DescribeVault",
         "glacier:ListVaults"
         ],
         "Resource": "arn:aws:glacier:us-west-2:123456789012:vaults/*"
      }
   ]
}
```

위 정책은 us-west-2 리전에서 볼트를 생성하거나, 목록을 조회하거나, 가져오는 권한을 부여합니다. ARN 끝에 있는 와일드카드 문자(*)는 이 문이 모든 볼트 이름과 일치할 수 있다는 것을 의미합니다.

> **Important**
>
> glacier:CreateVault 작업을 사용하여 볼트 생성 권한을 부여할 때는 볼트를 생성할 때까지 볼트 이름을 모르기 때문에 와일드카드 문자(*)를 반드시 지정해야 합니다.

주제

- 정책 모범 사례
- Amazon Glacier 콘솔 사용
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용
- 고객 관리형 정책 예

정책 모범 사례

ID 기반 정책에 따라 해당 계정에서 누가 Amazon Glacier 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 따르세요.

- AWS 관리형 정책을 시작하고 최소 권한으로 전환 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 에서 사용할 수

있습니다 AWS 계정. 사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS 직무에 대한 관리형 정책을 참조하세요.

- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. AWS 서비스와 같은 특정를 통해 사용되는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다 CloudFormation. 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을 확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer에서 정책 검증을 참조하세요.
- 다중 인증(MFA) 필요 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 AWS 계정켭니다. API 작업을 직접적으로 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.

Amazon Glacier 콘솔 사용

Amazon Glacier 콘솔에 액세스하려면 최소한의 권한 세트가 있어야 합니다. 이러한 권한은 AWS 계정에서 Amazon Glacier 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용해야 합니다. 최소 필수 권한보다 더 제한적인 ID 기반 정책을 생성하는 경우, 콘솔이 해당 정책에 연결된 엔티티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

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

Amazon Glacier 콘솔은 Amazon Glacier 볼트를 생성하고 관리하기 위한 통합 환경을 제공합니다. 다음의 예시와 같이 최소한 사용자가 생성한 IAM 자격증명은 Amazon Glacier 콘솔을 보는 `glacier:ListVaults` 작업에 대한 권한을 반드시 부여받아야 합니다.

JSON

```
{
      "Version":"2012-10-17",
      "Statement": [
         {
            "Action": [
            "glacier:ListVaults"
            ],
            "Effect": "Allow",
            "Resource": "*"
         }
      ]
   }
```

AWS 는에서 생성하고 관리하는 독립 실행형 IAM 정책을 제공하여 많은 일반적인 사용 사례를 처리합니다 AWS. 관리형 정책은 필요한 권한을 사용자가 조사할 필요가 없도록 일반 사용 사례에 필요한 권한을 부여합니다. 자세한 내용은 IAM 사용자 안내서의 AWS 관리형 정책을 참조하세요.

계정의 사용자에게 연결할 수 있는 다음 AWS 관리형 정책은 Amazon Glacier에 고유합니다.

- AmazonGlacierReadOnlyAccess –를 통해 Amazon Glacier에 대한 읽기 전용 액세스 권한을 부여합니다 AWS Management Console.
- AmazonGlacierFullAccess –를 통해 Amazon Glacier에 대한 전체 액세스 권한을 부여합니다 AWS Management Console.

Amazon Glacier API 작업 및 리소스에 대한 권한을 허용하는 사용자 지정 IAM 정책을 생성할 수도 있습니다. Amazon Glacier 볼트를 위해 생성한 사용자 지정 IAM 역할에 이러한 사용자 지정 정책을 연결할 수 있습니다.

다음 섹션에서 설명한 두 Amazon Glacier AWS 관리형 정책 모두에 대한 권한을 부여합니다`glacier:ListVaults`.

자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하세요.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

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

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

고객 관리형 정책 예

이 섹션에서는 다양한 Amazon Glacier 작업에 대한 권한을 부여하는 사용자 정책 예시를 제공합니다. 이러한 정책은 Amazon Glacier REST API, Amazon SDKs AWS CLI, 또는 해당하는 경우 Amazon Glacier 관리 콘솔을 사용할 때 작동합니다.

> **Note**
>
> 모든 예에서는 미국 서부(오리건) 리전(`us-west-2`)을 사용하며 가상의 계정 ID를 포함합니다.

예제

- 예제 1: 사용자가 볼트에서 아카이브를 다운로드할 수 있도록 허용
- 예제 2: 사용자가 볼트를 생성한 후 알림을 구성할 수 있도록 허용
- 예제 3: 사용자가 아카이브를 특정 볼트로 업로드하도록 허용
- 예제 4: 사용자에게 특정 볼트에 대한 모든 권한 허용

예제 1: 사용자가 볼트에서 아카이브를 다운로드할 수 있도록 허용

아카이브를 다운로드하려면 먼저 아카이브를 가져오는 작업부터 시작합니다. 가져오기 작업을 먼저 마쳐야만 데이터를 다운로드할 수 있습니다. 아래 정책 예제는 작업(사용자가 볼트에서 아카이브 또는 볼트 인벤토리를 가져올 수 있는 작업)을 시작할 수 있는 `glacier:InitiateJob` 권한과 가져온 데이터를 다운로드할 수 있는 `glacier:GetJobOutput` 권한을 부여합니다. 또한 사용자가 작업 상태를 가져올 수 있는 `glacier:DescribeJob` 권한도 부여합니다. 자세한 내용은 작업 시작(POST jobs) 단원을 참조하십시오.

이 정책은 이름이 `examplevault`인 볼트와 관련하여 위와 같은 권한을 부여합니다. 볼트 ARN은 Amazon Glacier 콘솔에서 혹은 볼트 설명(GET vault) 또는 볼트 목록 조회(GET vaults) API 작업을 직접 호출하여 프로그래밍 방식으로 가져올 수 있습니다.

예제 2: 사용자가 볼트를 생성한 후 알림을 구성할 수 있도록 허용

다음 정책 예시는 `Resource` 요소에 지정된 것과 같이 볼트를 us-west-2 리전에 생성하고 알림을 구성할 수 있는 권한을 부여합니다. 알림 작업에 대한 자세한 내용은 Amazon Glacier의 볼트 알림 구성 섹션을 참조하세요. 또한이 정책은 AWS 리전의 볼트를 나열하고 특정 볼트 설명을 가져올 수 있는 권한을 부여합니다.

> **Important**
>
> `glacier:CreateVault` 작업을 사용하여 볼트 생성 권한을 부여할 때는 볼트를 생성할 때까지 볼트 이름을 모르기 때문에 `Resource` 값에 와일드카드 문자(*)를 반드시 지정해야 합니다.

예제 3: 사용자가 아카이브를 특정 볼트로 업로드하도록 허용

다음 정책 예시는 아카이브를 us-west-2 리전에 속한 특정 볼트로 업로드할 수 있는 권한을 부여합니다. 이 권한이 부여된 사용자는 아카이브 업로드(POST archive) API 작업을 사용하여 한 번에 아카이브 전체를 업로드하거나, 혹은 멀티파트 업로드 시작(POST multipart-uploads) API 작업을 사용하여 여러 파트로 나누어 업로드할 수 있습니다.

예제 4: 사용자에게 특정 볼트에 대한 모든 권한 허용

다음 정책 예시는 이름이 `examplevault`인 볼트의 모든 Amazon Glacier 작업에 권한을 부여합니다.

Amazon Glacier용 리소스 기반 정책 예제

Amazon Glacier 볼트에는 각각 볼트 액세스 정책 1개와 볼트 잠금 정책 1개를 연결할 수 있습니다. Amazon Glacier 볼트 액세스 정책은 볼트에 대한 권한을 관리할 때 사용할 수 있는 리소스 기반 정책입니다. 볼트 잠금 정책은 잠글 수 있는 볼트 액세스 정책입니다. 볼트 잠금 정책으로 고정한 후에는 정책을 변경할 수 없습니다. 볼트 잠금 정책은 규정 준수 제어 항목을 적용할 때 사용됩니다.

주제

- 볼트 액세스 정책
- 볼트 잠금 정책

볼트 액세스 정책

Amazon Glacier 볼트 액세스 정책은 볼트에 대한 권한을 관리할 때 사용할 수 있는 리소스 기반 정책입니다.

각 볼트마다 권한을 관리할 수 있는 볼트 액세스 정책은 한 가지만 생성할 수 있습니다. 언제든 볼트 액세스 정책에서 권한을 수정할 수 있습니다. 또한 Amazon Glacier는 각 볼트에 대해 볼트 잠금 정책을 지원합니다. 이 정책에 따르면 볼트를 잠근 후에는 변경할 수 없습니다. 볼트 잠금 정책의 사용에 대한 자세한 내용은 볼트 잠금 정책 섹션을 참조하세요.

예제

- 예시 1: 특정 Amazon Glacier 작업용 교차 계정 권한 부여
- 예제 2: MFA 삭제 작업에 대한 교차 계정 권한 부여

예시 1: 특정 Amazon Glacier 작업용 교차 계정 권한 부여

다음 정책 예시에서는 이름이 `examplevault`인 볼트의 Amazon Glacier 작업 세트를 위한 교차 계정 권한을 두 AWS 계정 에 부여합니다.

> **Note**
>
> 볼트가 속하는 계정에게는 볼트와 관련된 모든 비용이 청구됩니다. 허용되는 외부 계정에서 실행한 요청, 데이터 전송 및 가져오기 비용 역시 볼트가 속한 계정에게 청구됩니다.

예제 2: MFA 삭제 작업에 대한 교차 계정 권한 부여

다중 인증(MFA)을 사용하여 Amazon Glacier 리소스를 보호할 수 있습니다. MFA에서 보안 계층을 추가하려면 사용자가 유효한 MFA 코드를 입력하여 MFA 디바이스에 대한 물리적인 소유권을 입증해야 합니다. MFA 액세스 구성에 대한 자세한 정보는 IAM 사용 설명서의 MFA 보호 API 액세스 구성을 참조하세요.

예제 정책은 요청이 MFA 디바이스로 인증된 경우 임시 자격 증명이 AWS 계정 있는에 examplevault라는 볼트에서 아카이브를 삭제할 수 있는 권한을 부여합니다. 이 정책은 `aws:MultiFactorAuthPresent` 조건 키를 사용하여 이러한 추가 요건을 지정합니다. 자세한 내용은 IAM 사용 설명서의 조건에 사용 가능한 키를 참조하세요.

볼트 잠금 정책

Amazon Glacier(Amazon Glacier) 볼트는 각각 리소스 기반 볼트 액세스 정책 한 개와 볼트 잠금 정책 한 개를 연결할 수 있습니다. 볼트 잠금 정책은 잠글 수 있는 볼트 액세스 정책입니다. 볼트 잠금 정책을 사용하면 규제 및 규정 준수 요구 사항을 적용할 수 있습니다. Amazon Glacier가 제공하는 볼트 잠금 정책을 관리하기 위한 API 작업 세트는 Amazon Glacier API를 사용하여 볼트 잠금 섹션을 참조하세요.

볼트 잠금 정책의 예를 들자면, 아카이브를 삭제하기 전에 1년간 저장해야 한다고 가정하겠습니다. 이러한 요건을 만족하려면 먼저 아카이브를 1년간 저장할 때까지 사용자가 아카이브를 삭제하지 못하도록 볼트 잠금 정책을 생성해야 합니다. 생성된 정책은 고정하기 전에 테스트할 수 있습니다. 일단 고정된 정책은 변경할 수 없습니다. 잠금 프로세스에 대한 자세한 내용은 볼트 잠금 정책 섹션을 참조하세

요. 그 밖에 변경 가능한 사용자 권한을 관리하려면 볼트 액세스 정책을 사용할 수 있습니다(볼트 액세스 정책 참조).

Amazon Glacier API, Amazon SDKs AWS CLI또는 Amazon Glacier 콘솔을 사용하여 볼트 잠금 정책을 생성하고 관리할 수 있습니다. 볼트 리소스 기반 정책에서 허용되는 Amazon Glacier 작업 목록은 API 권한 준거 섹션을 참조하세요.

예제

- 예제 1: 365일이 지나지 않은 아카이브에 대한 삭제 권한 거부
- 예제 2: 태그를 기반으로 한 삭제 권한 거부

예제 1: 365일이 지나지 않은 아카이브에 대한 삭제 권한 거부

예를 들어 규제 요건에 따라 아카이브를 삭제하기 전에 최대 1년까지 저장해야 한다고 가정하겠습니다. 이러한 요건은 다음 볼트 잠금 정책을 구현하여 만족할 수 있습니다. 이 정책에 따라 삭제하려는 아카이브의 저장 기간이 1년 미만인 경우에는 examplevault 볼트에 대한 `glacier:DeleteArchive` 작업이 거부됩니다. 이 정책은 Amazon Glacier별 조건 키인 `ArchiveAgeInDays`를 사용하여 1년 보존 요건을 적용합니다.

예제 2: 태그를 기반으로 한 삭제 권한 거부

저장 기간이 1년 미만인 아카이브는 삭제할 수 있다는 시간 기반 저장 규칙이 있다고 가정하겠습니다. 또한 법적 조사 기간에는 아카이브를 삭제 또는 변경할 수 없도록 법적 보존을 무기한 설정해야 한다고 가정하겠습니다. 이 경우 법적 보존이 볼트 잠금 정책에서 지정한 시간 기반 저장 규칙보다 우선합니다.

다음 정책 예제에는 이러한 두 가지 정책을 적용할 수 있도록 2개의 문이 있습니다.

- 첫 번째 문은 볼트를 잠가서 모든 사용자의 삭제 권한을 거부합니다. 이 잠금에서는 `LegalHold` 태그를 사용합니다.
- 두 번째 문은 아카이브의 저장 기간이 365일 미만일 때 삭제 권한을 부여합니다. 하지만 아카이브의 저장 기간이 365일 미만이라고 해도 첫 번째 문의 조건에 부합될 경우 아무도 삭제할 수 없습니다.

Amazon Glacier ID 및 액세스 문제 해결

다음 정보를 사용하여 Amazon Glacier 및 IAM으로 작업할 때 발생할 수 있는 일반적인 문제를 진단하고 수정할 수 있습니다.

주제

- Amazon Glacier 에서 작업을 수행할 권한이 없음
- iam:PassRole을 수행하도록 인증되지 않음
- 내 외부의 사람이 내 Amazon Glacier 리소스 AWS 계정 에 액세스하도록 허용하고 싶습니다.

Amazon Glacier 에서 작업을 수행할 권한이 없음

작업을 수행할 권한이 없다는 오류가 표시되면 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 glacier:*GetWidget* 권한이 없을 때 발생합니다.

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

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

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

iam:PassRole을 수행하도록 인증되지 않음

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

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

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

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우, Mary가 iam:PassRole작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

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

내 외부의 사람이 내 Amazon Glacier 리소스 AWS 계정 에 액세스하도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세한 내용은 다음을 참조하세요.

- Amazon Glacier에서 이러한 기능을 지원하는지 알아보려면 Amazon Glacier가 IAM에서 작동하는 방식 섹션을 참조하세요.
- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유한 다른의 IAM 사용자에게 액세스 권한 제공을 참조 AWS 계정 하세요.
- 타사에 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사가 AWS 계정 소유한에 대한 액세스 권한 제공을 AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 교차 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 교차 계정 리소스 액세스를 참조하세요.

API 권한 준거

Amazon Glacier가 IAM에서 작동하는 방식을 설정하여 IAM 자격 증명(자격 증명 기반 정책) 또는 리소스(리소스 기반 정책)에 연결할 수 있는 권한 정책을 생성할 때는 아래 표를 참조할 수 있습니다. 는 각 Amazon Glacier API 작업, 작업을 수행할 권한을 부여할 수 있는 해당 작업, 권한을 부여할 수 있는 리소스가 포함되어 있습니다. Amazon Glacier AWS

정책의 `Action` 요소에서 작업을 지정하고, 정책의 `Resource` 요소에서 리소스 값을 지정합니다. 또한 IAM 정책 언어 `Condition` 요소를 사용하여 정책이 시행되는 시점을 지정할 수 있습니다.

작업을 지정하려면 `glacier:` 접두사 다음에 API 작업 이름을 사용합니다(예: `glacier:CreateVault`). 대부분 Amazon Glacier 작업에서 `Resource`는 권한을 부여하고 싶은 볼트에 해당합니다. 볼트 ARN을 사용하여 `Resource` 값으로 볼트를 지정합니다. 조건을 표시하려면 미리 정의된 조건 키를 사용합니다. 자세한 내용은 Amazon Glacier 내의 리소스 기반 정책 단원을 참조하십시오.

다음 표에는 자격 증명 기반 정책과 리소스 기반 정책에서 사용할 수 있는 작업이 나열되어 있습니다.

> **Note**
>
> 일부 작업은 자격 증명 기반 정책에서만 사용할 수 있습니다. 이러한 작업은 첫 번째 열에서 API 작업 이름 뒤에 빨간색 별표(*)로 표시됩니다.

Amazon Glacier API 및 작업에 필요한 권한

멀티파트 업로드 중단(DELETE uploadID)

필요한 권한(API 작업): `glacier:AbortMultipartUpload`

리소스: `arn:aws:glacier:region:account-id:vaults/vault-name`, `arn:aws:glacier:region:account-id:vaults/example*`, `arn:aws:glacier:region:account-id:vaults/*`

Amazon Glacier 조건 키:

볼트 잠금 중단(DELETE lock-policy)

필요한 권한(API 작업): `glacier:AbortVaultLock`

리소스:

Amazon Glacier 조건 키:

볼트에 태그 추가(POST tags add)

필요한 권한(API 작업):`glacier:AddTagsToVault`

리소스: `arn:aws:glacier:region:account-id:vaults/vault-name`, `arn:aws:glacier:region:account-id:vaults/example*`, `arn:aws:glacier:region:account-id:vaults/*`

Amazon Glacier 조건 키: `glacier:ResourceTag/TagKey`

멀티파트 업로드 완료(POST uploadID)

필요한 권한(API 작업):`glacier:CompleteMultipartUpload`

리소스: `arn:aws:glacier:region:account-id:vaults/vault-name`, `arn:aws:glacier:region:account-id:vaults/example*`, `arn:aws:glacier:region:account-id:vaults/*`

Amazon Glacier 조건 키: `glacier:ResourceTag/`*`TagKey`*

볼트 잠금 완료(POST lockId)

필요한 권한(API 작업):`glacier:CompleteVaultLock`

리소스:

Amazon Glacier 조건 키: `glacier:ResourceTag/`*`TagKey`*

볼트 만들기(PUT vault) *

필요한 권한(API 작업):`glacier:CreateVault`

리소스:

Amazon Glacier 조건 키:

아카이브 삭제(DELETE archive)

필요한 권한(API 작업):`glacier:DeleteArchive`

리소스: `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/vault-name`, `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/example*`, `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/*`

Amazon Glacier 조건 키: `glacier:ArchiveAgeInDays`, `glacier:ResourceTag/`*`TagKey`*

볼트 삭제(DELETE vault)

필요한 권한(API 작업):`glacier:DeleteVault`

리소스: `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/vault-name`, `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/example*`, `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/*`

Amazon Glacier 조건 키: `glacier:ResourceTag/`*`TagKey`*

볼트 액세스 정책 삭제(DELETE access-policy)

필요한 권한(API 작업):`glacier:DeleteVaultAccessPolicy`

리소스: `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/vault-name`, `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/example*`, `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/*`

Amazon Glacier 조건 키: `glacier:ResourceTag/`*`TagKey`*

볼트 알림 삭제(DELETE notification-configuration)

필요한 권한(API 작업):glacier:DeleteVaultNotifications

리소스: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier 조건 키: glacier:ResourceTag/*TagKey*

작업 설명(GET JobID)

필요한 권한(API 작업):glacier:DescribeJob

리소스: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier 조건 키:

볼트 설명(GET vault)

필요한 권한(API 작업):glacier:DescribeVault

리소스: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier 조건 키:

데이터 가져오기 정책 가져오기(GET policy) *

필요한 권한(API 작업):glacier:GetDataRetrievalPolicy

리소스: arn:aws:glacier:*region*:*account-id*:policies/retrieval-limit-policy

Amazon Glacier 조건 키:

작업 출력 가져오기(GET output)

필요한 권한(API 작업):glacier:GetJobOutput

리소스: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier 조건 키:

볼트 액세스 정책 가져오기(GET access-policy)

필요한 권한(API 작업):glacier:GetVaultAccessPolicy

리소스: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier 조건 키:

볼트 잠금 가져오기(GET lock-policy)

필요한 권한(API 작업):glacier:GetVaultLock

리소스: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier 조건 키:

볼트 알림 가져오기(GET notification-configuration)

필요한 권한(API 작업):glacier:GetVaultNotifications

리소스: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier 조건 키:

작업 시작(POST jobs)

필요한 권한(API 작업):glacier:InitiateJob

리소스: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier 조건 키: glacier:ArchiveAgeInDays, glacier:ResourceTag/*TagKey*

멀티파트 업로드 시작(POST multipart-uploads)

필요한 권한(API 작업):glacier:InitiateMultipartUpload

리소스: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier 조건 키: glacier:ResourceTag/*TagKey*

볼트 잠금 시작(POST lock-policy)

필요한 권한(API 작업):glacier:InitiateVaultLock

리소스:

Amazon Glacier 조건 키: glacier:ResourceTag/*TagKey*

작업 목록 조회(GET jobs)

필요한 권한(API 작업):glacier:ListJobs

리소스: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier 조건 키:

멀티파트 업로드 목록 조회(GET multipart-uploads)

필요한 권한(API 작업):glacier:ListMultipartUploads

리소스: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier 조건 키:

파트 목록 조회(GET uploadID)

필요한 권한(API 작업):glacier:ListParts

리소스: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier 조건 키:

볼트의 태그 목록 조회(GET tags)

필요한 권한(API 작업):glacier:ListTagsForVault

리소스: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier 조건 키:

볼트 목록 조회(GET vaults)

필요한 권한(API 작업):glacier:ListVaults

리소스:

Amazon Glacier 조건 키:

볼트에서 태그 삭제(POST tags remove)

필요한 권한(API 작업):glacier:RemoveTagsFromVault

리소스: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier 조건 키: glacier:ResourceTag/*TagKey*

데이터 가져오기 정책 설정(PUT policy) *

필요한 권한(API 작업):glacier:SetDataRetrievalPolicy

리소스:arn:aws:glacier:*region*:*account-id*:policies/retrieval-limit-policy

Amazon Glacier 조건 키:

볼트 액세스 정책 설정(PUT access-policy)

필요한 권한(API 작업):glacier:SetVaultAccessPolicy

리소스: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier 조건 키: glacier:ResourceTag/*TagKey*

볼트 알림 구성 설정(PUT notification-configuration)

필요한 권한(API 작업):glacier:SetVaultNotifications

리소스: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier 조건 키: glacier:ResourceTag/*TagKey*

아카이브 업로드(POST archive)

필요한 권한(API 작업):glacier:UploadArchive

리소스: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier 조건 키: glacier:ResourceTag/*TagKey*

파트 업로드(PUT uploadID)

필요한 권한(API 작업):glacier:UploadMultipartPart

리소스: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier 조건 키: glacier:ResourceTag/*TagKey*

Amazon Glacier의 로깅 및 모니터링

모니터링은 Amazon Glacier(Amazon Glacier)와 AWS 솔루션의 신뢰성, 가용성 및 성능을 유지하는 데 중요한 부분입니다. AWS 솔루션의 모든 부분에서 모니터링 데이터를 수집하여 실패가 발생한 경우 실패의 원인을 보다 쉽게 식별하고 디버그할 수 있도록 해야 합니다. AWS는 Amazon Glacier 리소스를 모니터링하고 잠재적인 인시던트에 대응하기 위한 다음과 같은 도구를 제공합니다.

Amazon CloudWatch 경보

Amazon S3를 통해 Amazon Glacier를 사용하는 경우 Amazon CloudWatch 경보를 사용하여 지정한 기간 동안 단일 지표를 감시할 수 있습니다. 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 사용하여 지표 모니터링 섹션을 참조하세요.

AWS CloudTrail 로그

CloudTrail은 사용자, 역할 또는 Amazon Glacier의 AWS 서비스가 수행한 작업 기록을 제공합니다. CloudTrail은 Amazon Glacier 콘솔의 직접 호출 및 Amazon Glacier API로의 코드 호출을 포함하여 Amazon Glacier에 대한 모든 API 직접 호출을 이벤트로 캡처합니다. 자세한 내용은 를 사용하여 Amazon Glacier API 호출 로깅 AWS CloudTrail 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.

자세한 내용은 지원 사용 설명서의 AWS Trusted Advisor 섹션을 참조하세요.

Amazon Glacier에 대한 규정 준수 확인

서드 파티 감사자는 Amazon Glacier(Amazon 3 Glacier)의 보안 및 규정 준수를 복수의 AWS 규정 준수 프로그램의 일환으로 평가하며, 여기에는 다음 내용이 포함됩니다.

- 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 Artifact 사용 설명서의 AWS 아티팩트를 참조하세요.

AWS 규정 준수 프로그램에 대한 자세한 내용은 AWS 규정 준수 프로그램을 참조하세요.

Amazon Glacier 사용 시 귀하의 규정 준수 책임은 데이터의 민감도, 조직의 규정 준수 목표, 관련 법률과 규정에 따라 결정됩니다. Amazon Glacier 사용 시 HIPAA, PCI, FedRAMP와 같은 표준의 규정을 준수해야 하는 경우 AWS에서는 도움이 되도록 다음과 같은 리소스를 제공합니다.

- Amazon Glacier 볼트 잠금은 각 Amazon Glacier 볼트마다 볼트 잠금 정책을 사용해 규정 준수 제어 항목을 쉽게 배포하고 적용할 수 있는 기능입니다. 볼트 잠금 정책에서는 'write once, read many'(WORM) 같은 제어 항목을 지정하여 앞으로 편집하지 못하도록 정책을 잠글 수 있습니다. 일단 잠긴 정책은 더 이상 변경할 수 없습니다. 볼트 잠금 정책은 SEC17a-4 및 HIPAA와 같은 규제 프레임워크를 준수하는 데 도움이 됩니다.
- 보안 및 규정 준수 빠른 시작 안내서에서는 아키텍처의 고려 사항과 보안 및 규정 준수에 중점을 둔 AWS 기본 환경을 배포하기 위한 단계에 대해 설명합니다.
- HIPAA 보안 및 규정 준수를 위한 설계에서는 기업에서 AWS를 사용하여 HIPAA 요구 사항을 충족하는 방법을 설명합니다.
- AWS Well-Architected Tool(AWS WA Tool)은 AWS의 모범 사례를 사용해 아키텍처를 검토 및 측정할 수 있도록 일관적인 프로세스를 제공하는 클라우드의 서비스입니다. AWS WA Tool은 워크로드의 신뢰성 및 보안, 효율성, 경제성을 높일 수 있는 권장 사항을 제공합니다.
- AWS 규정 준수 리소스는 귀사의 산업 및 위치에 적용될 수 있는 몇몇 종류별 워크북 및 안내서를 제공합니다.

- AWS Config으로 리소스 구성이 내부 관행, 업계 지침 및 규정을 준수하는 정도를 평가할 수 있습니다.
- AWS Security Hub는 보안 업계 표준 및 모범 사례 준수 여부를 확인하는 데 도움이 되는 AWS 내 보안 상태에 대한 포괄적인 관점을 제공합니다.

Amazon Glacier의 복원성

AWS 글로벌 인프라는 리전 및 가용 영역을 중심으로 구축됩니다. AWS 리전에서는 물리적으로 분리되고 격리된 다수의 가용 영역을 제공하며 이러한 가용 영역은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크에 연결되어 있습니다. 이러한 가용 영역은 애플리케이션과 데이터베이스를 설계하고 운영하는 효과적인 방법을 제공합니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다. Amazon Glacier는 최소 세 개의 가용 영역에 걸친 여러 디바이스에 데이터를 중복 저장합니다. Amazon Glacier는 내구성을 높이기 위해 업로드 성공을 확인하기 전 여러 AZ에 데이터를 동기식으로 저장합니다.

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

Amazon Glacier의 인프라 보안

관리형 서비스인 Amazon Glacier(Amazon Glacier)는 Amazon Web Services: 보안 프로세스의 개요에 설명된 AWS 글로벌 네트워크 보안 절차로 보호됩니다.

네트워크를 통한 Amazon Glacier 액세스는 AWS에서 게시한 API를 통해 이루어집니다. 클라이언트가 Transport Layer Security(TLS) 1.2를 지원해야 합니다. TLS 1.3 이상을 권장합니다. 클라이언트는 DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Diffie-Hellman Ephemeral)와 같은 PFS(전달 완전 보안)가 포함된 암호 제품군도 지원해야 합니다. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다. 또한 반드시 액세스 키 ID와 IAM 보안 주체와 관련된 비밀 액세스 키를 사용하여 요청에 서명하거나 AWS Security Token Service(AWS STS)를 사용하여 요청에 서명할 수 있는 임시 보안 인증 정보를 생성하여야 합니다.

VPC 엔드포인트

Virtual Private Cloud(VPC) 엔드포인트를 사용하면 인터넷 게이트웨이, NAT 장치, VPN 연결 또는 AWS Direct Connect 연결 요청 없이 지원되는 AWS 서비스 및 AWS PrivateLink로 구동되는 VPC 엔드포인트 서비스에 VPC를 비공개로 연결할 수 있습니다. Amazon Glacier는 VPC 엔드포인트를 직접 지원하지는 않지만 Amazon S3로 통합된 스토리지 계층으로 Amazon Glacier에 액세스하는 경우 Amazon S3 VPC 엔드포인트를 활용할 수 있습니다.

Amazon S3 수명 주기 구성 및 Amazon Glacier 스토리지 클래스로의 객체 전환에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 객체 수명 주기 관리 및 객체 전환을 참조하세요. VPC 엔드포인트에 대한 자세한 내용은Amazon VPC 사용 설명서의 VPC 엔드포인트를 참조하세요.

Amazon Glacier 데이터 검색 정책

Amazon Glacier 데이터 검색 정책을 사용하면 데이터 검색 할당량을 쉽게 설정하고 각에서의 데이터 검색 활동을 관리할 수 AWS 계정 있습니다 AWS 리전. Amazon Glacier 데이터 검색 요금에 대한 자세한 내용은 Amazon Glacier 요금을 참조하세요.

> **Important**
>
> 데이터 검색 정책은 표준 검색에만 적용되며 Amazon Glacier에 직접적으로 보낸 검색 요청을 관리합니다.
>
> Amazon Glacier 스토리지 클래스에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 객체 아카이빙을 위한 스토리지 클래스 및 객체 전환을 참조하세요.

주제

- Amazon Glacier 데이터 검색 정책 선택
- Amazon Glacier 콘솔을 사용하여 데이터 검색 정책 설정
- Amazon Glacier API를 사용하여 데이터 검색 정책 설정

Amazon Glacier 데이터 검색 정책 선택

검색 제한 없음, 프리 티어만 및 최대 검색 속도의 세 가지 Amazon Glacier 데이터 검색 정책 유형 중에 선택할 수 있습니다.

검색 제한 없음은 검색에 사용되는 기본 데이터 검색 정책입니다. 검색 제한 없음 정책을 사용하면 검색 할당량이 설정되지 않고 유효한 모든 데이터 검색 요청이 허용됩니다.

프리 티어 전용 정책을 사용하면 일일 AWS 프리 티어 허용량 내에서 검색을 유지할 수 있으며 데이터 검색 비용이 발생하지 않습니다. AWS 프리 티어 허용량보다 많은 데이터를 검색하려면 최대 검색 속도 정책을 사용하여 bytes-per-hour 검색 속도 할당량을 설정할 수 있습니다. 최대 검색 속도 정책을 사용하면 AWS 리전 의 계정 내에서 실행하는 모든 검색 작업의 최대 검색 속도가 사용자가 설정한 시간당 바이트 할당량을 초과하지 않습니다.

프리 티어만 정책과 최대 검색 속도 정책을 사용하면 지정한 검색 할당량을 초과하는 데이터 검색 요청은 허용되지 않습니다. 프리 티어만 정책을 사용하면 Amazon Glacier는 AWS 프리 티어 허용 한도

를 초과하는 검색 요청을 모두 동기식으로 거부합니다. 또한 최대 검색 속도 정책을 사용하면 Amazon Glacier는 진행 중인 작업의 최대 검색 속도가 정책에서 설정한 시간당 바이트 할당량을 초과하는 경우 검색 요청을 거부합니다. 이 두 정책은 데이터 검색 비용을 더욱 쉽게 관리하는 데 유용합니다.

데이터 검색 정책에 대한 유용한 사실 몇 가지를 소개합니다.

- 데이터 검색 정책을 설정하더라도 표준 검색을 사용하여 Amazon Glacier에서 데이터를 검색하는 데는 여전히 3~5시간이 걸립니다.
- 새로운 데이터 검색 정책을 설정하더라도 앞서 허용되어 이미 진행 중인 검색 작업에는 아무런 영향도 미치지 못합니다.
- 데이터 검색 정책으로 인해 검색 작업 요청이 거부된 경우에는 해당하는 검색 작업 또는 요청에 대해 아무런 요금도 청구되지 않습니다.
- 각에 대해 하나의 데이터 검색 정책을 설정할 수 있으며 AWS 리전,이 정책은 계정 AWS 리전 의에서 모든 데이터 검색 활동을 관리합니다. 데이터 검색 비용은 서로 다르 AWS 리전 기 때문에 데이터 검색 정책은 특정에 고유합니다 AWS 리전. 자세한 내용은 Amazon Glacier 요금을 참조하세요.

프리 티어만 정책

데이터 검색 정책을 프리 티어 전용으로 설정하여 데이터 검색 요금이 발생하지 않도록 검색이 항상 AWS 프리 티어 허용량 내에 있도록 할 수 있습니다. 이때 검색 요청이 거부되면 현재 데이터 검색 정책에 따라 요청이 거부되었다는 오류 메시지가 수신됩니다.

리전별 기준에 따라 데이터 검색 정책을 프리 티어만으로 설정할 수 있습니다. 정책을 설정한 후에는 하루에 그 AWS 리전의 할당된 일일 AWS 프리 티어 검색 허용 한도보다 더 많은 데이터를 검색할 수 없습니다. 데이터 검색 요금도 발생하지 않습니다.

데이터 검색 비용이 발생한 후라도 한 달 내에 프리 티어만 정책으로 전환할 수 있습니다. 이런 경우에 프리 티어만 정책은 새로운 검색 요청에만 적용되며 이전의 요청에는 영향이 가지 않습니다. 따라서 이전에 발생한 비용에 대해서는 요금이 청구됩니다.

최대 검색 속도 정책

데이터 검색 정책을 최대 검색 속도로 설정하여 최대 속도가 시간당 바이트인 데이터 검색 할당량을 지정함으로써 최대 검색 속도를 제어할 수 있습니다. 데이터 검색 정책을 최대 검색 속도로 설정하면 진행 중인 작업의 최대 검색 속도가 정책에서 지정한 시간당 바이트 할당량을 초과할 경우 새로운 검색 요청이 거부됩니다. 검색 작업 요청이 거부되면 현재 데이터 검색 정책에 따라 요청이 거부되었다는 오류 메시지가 수신됩니다.

데이터 검색 정책을 최대 검색 속도 정책으로 설정하면 하루에 사용할 수 있는 AWS 프리 티어 허용량에 영향을 미칠 수 있습니다. 예를 들어 최대 검색 속도를 시간당 1MB로 설정한다고 가정하겠습니다. 프리 AWS 티어 정책 요금보다 낮습니다. 일일 AWS 프리 티어 허용량을 잘 활용하려면 먼저 정책을 프리 티어 전용으로 설정한 다음 필요한 경우 나중에 최대 검색 속도 정책으로 전환할 수 있습니다. 검색 허용 한도 계산 방법에 대한 자세한 내용은 Amazon Glacier FAQ를 참조하세요.

검색 제한 없음 정책

데이터 검색 정책을 검색 제한 없음으로 설정하면 유효한 데이터 검색 요청이 모두 허용되며, 사용량에 따라 데이터 검색 비용이 결정됩니다.

Amazon Glacier 콘솔을 사용하여 데이터 검색 정책 설정

Amazon Glacier 콘솔을 사용하여 데이터 검색 정책 생성

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/glacier/home Amazon Glacier 콘솔을 엽니다.
2. 리전 선택 AWS 리전 의 드롭다운 메뉴에서을 선택합니다. 각각에 대한 데이터 검색 정책을 구성할 수 있습니다 AWS 리전.
3. 왼쪽의 탐색 창에서 데이터 검색 설정을 선택합니다.
4. 편집을 선택합니다. 그러면 데이터 검색 정책 편집 페이지가 나타납니다.
5. 데이터 검색 정책에서 정책을 선택합니다.

   데이터 검색 정책은 검색 제한 없음, 프리 티어만, 최대 검색 속도 지정의 세 가지 중에서 하나를 선택할 수 있습니다.

   - 검색 제한 없음을 선택하면 모든 유효한 데이터 검색 요청이 허용됩니다.
   - 프리 티어만 선택하면 AWS 프리 티어를 초과하는 데이터 검색 요청은 허용되지 않습니다.
   - 최대 검색 속도 지정을 선택하면 진행 중인 작업의 최대 검색 속도가 사용자가 지정한 최대 검색 속도를 초과하는 경우 데이터 검색 요청을 거부합니다. 최대 검색 속도 하단의 GB/시간 상자에 시간당 기가바이트(GB)값을 지정해야 합니다. GB/시간 값을 입력하면 콘솔이 예상 비용을 계산합니다.
6. 변경 사항 저장을 선택합니다.

Amazon Glacier API를 사용하여 데이터 검색 정책 설정

Amazon Glacier REST API 또는 AWS SDK를 사용하여 데이터 검색 정책을 확인하고 설정할 수 있습니다.

Amazon Glacier REST API를 사용하여 데이터 검색 정책 설정

Amazon Glacier REST API를 사용하여 데이터 검색 정책을 확인하고 설정할 수 있습니다. 기존 데이터 가져오기 정책은 데이터 가져오기 정책 가져오기(GET policy) 작업으로 확인할 수 있습니다. 데이터 가져오기 정책 설정(PUT policy) 작업을 사용하여 데이터 검색 정책을 설정할 수도 있습니다.

PUT 정책 작업을 사용할 때는 JSON `Strategy` 필드 값을 `BytesPerHour`, `FreeTier` 혹은 `None`로 설정해 데이터 검색 정책을 선택해야 합니다. `BytesPerHour`는 콘솔에서 최대 검색 속도 지정을 선택하는 것과 같고, `FreeTier`은 프리 티어만을 선택하는 것과, `None`은 검색 정책 없음을 선택하는 것과 같습니다.

데이터 검색 정책에서 설정한 최대 검색 속도를 초과하는 데이터 검색 작업을 작업 시작(POST jobs) 작업을 사용하여 시작하면, `Initiate Job` 작업은 작업을 중단하고 예외를 던집니다.

AWS SDKs를 사용하여 데이터 검색 정책 설정

AWS 는 Amazon Glacier용 애플리케이션을 개발할 수 있는 SDKs를 제공합니다. 이 SDK는 기본 REST API로 매핑되는 라이브러리를 비롯해 쉽게 요청을 구성하여 응답을 처리하도록 하는 객체를 제공합니다. 자세한 내용은 Amazon Glacier에서 AWS SDKs 사용 단원을 참조하십시오.

Amazon Glacier 리소스에 태그 지정

태그는 AWS 리소스에 할당하는 레이블입니다. 각 태그는 사용자가 정의하는 키와 값으로 구성됩니다. 사용자가 정의한 태그는 Amazon Glacier(Amazon Glacier) 볼트 리소스에 할당할 수 있습니다. 태그를 사용하면 간단하면서도 효과적으로 AWS 리소스를 관리하고 결제 데이터를 포함하여 데이터를 구성할 수 있습니다.

주제

- 태그 지정 관련 기본 사항
- 태그 제한
- 태그 지정을 사용하여 비용 추적
- 태그 지정을 이용한 액세스 제어 관리
- 관련 섹션

태그 지정 관련 기본 사항

Amazon Glacier 콘솔, AWS Command Line Interface(AWS CLI) 또는 Amazon Glacier API를 사용하여 다음 작업을 완료할 수 있습니다.

- 태그를 볼트에 추가
- 볼트에 대한 태그 목록 조회
- 태그를 볼트에서 제거

태그를 추가하거나, 목록을 조회하거나, 제거하는 방법에 대한 자세한 내용은 Amazon Glacier 볼트의 태그 지정 섹션을 참조하세요.

태그를 사용하여 볼트를 여러 카테고리로 분류할 수 있습니다. 예를 들어 용도, 소유자 또는 환경을 기준으로 볼트 카테고리를 분류할 수 있습니다. 각 태그에 대해 키와 값이 정의되기 때문에 특정 요구를 충족하는 사용자 지정 카테고리 세트를 생성할 수 있습니다. 예를 들어 볼트 소유자 및 용도를 기준으로 볼트를 쉽게 추적할 수 있도록 태그 집합을 정의할 수도 있습니다. 다음은 몇 가지 태그 예입니다.

- 소유자: 이름
- 용도: 동영상 아카이브
- 환경: 프로덕션

태그 제한

기본적인 태그 제한은 다음과 같습니다.

- 리소스(볼트)에 할당되는 최대 태그 수는 50개입니다.
- 태그 키와 값은 대/소문자를 구분합니다.

태그 키 제한은 다음과 같습니다.

- 볼트에 할당되는 태그 집합에서 각 태그 키는 고유해야 합니다. 이미 사용 중인 키를 가진 태그를 추가하면 기존 키-값 페어에 새 태그가 덮어쓰기 됩니다.
- 태그 키는 `aws:`로 시작할 수 없습니다. 이 접두사는 AWS용으로 예약되어 있기 때문입니다. AWS는 이 접두사로 시작되는 태그를 생성할 수 있지만, 사용자는 이를 편집하거나 삭제할 수 없습니다.
- 태그 키는 1~128자의 유니코드 문자이어야 합니다.
- 태그 키에는 유니코드 문자, 숫자, 스페이스, 그리고 `_ . / = + - @` 같은 특수 문자만 사용 가능합니다.

태그 값 제한은 다음과 같습니다.

- 태그 값은 0~128자의 유니코드 문자이어야 합니다.
- 태그 값은 공백 상태로 둘 수 있습니다. 아니면 유니코드 문자, 숫자, 스페이스, 그리고 `_ . / = + - @` 같은 특수 문자만 사용 가능합니다.

태그 지정을 사용하여 비용 추적

태그를 사용하여 AWS 비용을 분류하고 추적할 수 있습니다. 볼트를 포함하여 AWS 리소스에 태그를 적용할 때 AWS 비용 할당 보고서에는 태그별로 집계된 사용 내역 및 비용이 집계됩니다. 비즈니스 범주를 나타내는 태그(예: 비용 센터, 애플리케이션 이름 및 소유자)를 적용하여 여러 서비스의 비용을 정리할 수 있습니다. 자세한 내용은 AWS Billing 사용 설명서의 사용자 지정 결제 보고서에 비용 할당 태그 사용 섹션을 참조하세요.

태그 지정을 이용한 액세스 제어 관리

태그는 액세스 정책 문에서 조건으로 사용할 수 있습니다. 예를 들어 legal hold 태그를 설정한 후 데이터 유지 정책에서 “legal hold 태그 값이 `True`로 설정되는 경우 모든 사용자의 아카이브 삭제를 거부

하도록 명시하는" 조건으로 추가할 수 있습니다. 정상적인 조건에서는 데이터 유지 정책을 배포하여 legal hold 태그를 `False`로 설정할 수 있습니다. 하지만 조사를 지원할 목적으로 데이터를 보존해야 하는 경우에는 태그 값을 `True`로 설정하여 legal hold를 쉽게 활성화한 후 나중에 비슷한 방법으로 legal hold를 비활성화할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 태그를 사용한 액세스 제어를 참조하세요.

관련 섹션

- Amazon Glacier 볼트의 태그 지정

를 사용하여 Amazon Glacier API 호출 로깅 AWS CloudTrail

Amazon Glacier(Amazon Glacier)는 Amazon Glacier에서 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업에 대한 레코드를 제공하는 AWS 서비스와 통합됩니다. CloudTrail은 Amazon Glacier 콘솔의 직접 호출 및 Amazon Glacier API로의 코드 호출을 포함하여 Amazon Glacier에 대한 모든 API 직접 호출을 이벤트로 캡처합니다. 추적을 생성하면 Amazon Glacier 이벤트를 포함한 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 전달할 수 있습니다. 추적을 구성하지 않은 경우에도 이벤트 기록에서 CloudTrail 콘솔의 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 Amazon Glacier에 수행된 요청과 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간, 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

CloudTrail의 Amazon Glacier 정보

CloudTrail은 계정을 생성할 AWS 계정 때에서 활성화됩니다. Amazon Glacier에서 활동이 발생하면 해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. 에서 최근 이벤트를 보고 검색하고 다운로드할 수 있습니다 AWS 계정. 자세한 설명은 CloudTrail 이벤트 기록으로 이벤트 보기를 참조하세요.

Amazon Glacier에 대한 이벤트를 AWS 계정포함하여에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 기본적으로 콘솔에서 추적을 생성하면 추적이 모든 AWS 리전에 적용됩니다. 추적은 AWS 파티션의 모든 AWS 리전에서 이벤트를 로깅하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음을 참조하세요.

- 트레일 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에서 Amazon SNS 알림 구성
- 여러 리전으로부터 CloudTrail 로그 파일 받기 및 여러 계정으로부터 CloudTrail 로그 파일 받기

모든 Amazon Glacier 작업이 CloudTrail에서 로깅되고 Amazon Glacier를 위한 API 참조에서 문서화됩니다. 예를 들어 볼트 만들기(PUT vault), 볼트 삭제(DELETE vault) 및 볼트 목록 조회(GET vaults) 작업을 직접적으로 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 관한 정보가 포함됩니다. ID 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청이 생성될 때 루트 사용자를 이용하였는지 또는 다른 자격 증명을 이용하였는지 여부
- 역할 또는 페더레이션 사용자에 대한 임시 보안 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 요청이 다른 AWS 서비스에서 이루어졌는지 여부입니다.

자세한 설명은 CloudTrail userIdentity 요소를 참조하세요.

Amazon Glacier 로그 파일 항목 이해

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

다음의 예시는 볼트 만들기(PUT vault), 볼트 삭제(DELETE vault), 볼트 목록 조회(GET vaults) 및 볼트 설명(GET vault) 작업을 보여 주는 CloudTrail 로그 항목을 나타냅니다.


```
{
    "Records": [
        {
            "awsRegion": "us-east-1",
            "eventID": "52f8c821-002e-4549-857f-8193a15246fa",
            "eventName": "CreateVault",
            "eventSource": "glacier.amazonaws.com",
            "eventTime": "2014-12-10T19:05:15Z",
            "eventType": "AwsApiCall",
            "eventVersion": "1.02",
            "recipientAccountId": "999999999999",
            "requestID": "HJiLgvfXCY88QJAC6rRoexS9ThvI21Q1Nqukfly02hcUPPo",
            "requestParameters": {
                "accountId": "-",
                "vaultName": "myVaultName"
            },
            "responseElements": {
                "location": "/999999999999/vaults/myVaultName"
            },
            "sourceIPAddress": "127.0.0.1",
```

```
            "userAgent": "aws-sdk-java/1.9.6 Mac_OS_X/10.9.5 Java_HotSpot(TM)_64-
Bit_Server_VM/25.25-b02/1.8.0_25",
            "userIdentity": {
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "accountId": "999999999999",
                "arn": "arn:aws:iam::999999999999:user/myUserName",
                "principalId": "A1B2C3D4E5F6G7EXAMPLE",
                "type": "IAMUser",
                "userName": "myUserName"
            }
        },
        {
            "awsRegion": "us-east-1",
            "eventID": "cdd33060-4758-416a-b7b9-dafd3afcec90",
            "eventName": "DeleteVault",
            "eventSource": "glacier.amazonaws.com",
            "eventTime": "2014-12-10T19:05:15Z",
            "eventType": "AwsApiCall",
            "eventVersion": "1.02",
            "recipientAccountId": "999999999999",
            "requestID": "GGdw-VfhVfLCFwAM6iVUvMQ6-fMwSqSO9FmRd0eRSa_Fc7c",
            "requestParameters": {
                "accountId": "-",
                "vaultName": "myVaultName"
            },
            "responseElements": null,
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "aws-sdk-java/1.9.6 Mac_OS_X/10.9.5 Java_HotSpot(TM)_64-
Bit_Server_VM/25.25-b02/1.8.0_25",
            "userIdentity": {
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "accountId": "999999999999",
                "arn": "arn:aws:iam::999999999999:user/myUserName",
                "principalId": "A1B2C3D4E5F6G7EXAMPLE",
                "type": "IAMUser",
                "userName": "myUserName"
            }
        },
        {
            "awsRegion": "us-east-1",
            "eventID": "355750b4-e8b0-46be-9676-e786b1442470",
            "eventName": "ListVaults",
            "eventSource": "glacier.amazonaws.com",
            "eventTime": "2014-12-10T19:05:15Z",
```

```
            "eventType": "AwsApiCall",
            "eventVersion": "1.02",
            "recipientAccountId": "999999999999",
            "requestID": "yPTs22ghTsWprFivb-2u30FAaDALIZP17t4jM_xL9QJQyVA",
            "requestParameters": {
                "accountId": "-"
            },
            "responseElements": null,
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "aws-sdk-java/1.9.6 Mac_OS_X/10.9.5 Java_HotSpot(TM)_64-
Bit_Server_VM/25.25-b02/1.8.0_25",
            "userIdentity": {
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "accountId": "999999999999",
                "arn": "arn:aws:iam::999999999999:user/myUserName",
                "principalId": "A1B2C3D4E5F6G7EXAMPLE",
                "type": "IAMUser",
                "userName": "myUserName"
            }
        },
        {
            "awsRegion": "us-east-1",
            "eventID": "569e830e-b075-4444-a826-aa8b0acad6c7",
            "eventName": "DescribeVault",
            "eventSource": "glacier.amazonaws.com",
            "eventTime": "2014-12-10T19:05:15Z",
            "eventType": "AwsApiCall",
            "eventVersion": "1.02",
            "recipientAccountId": "999999999999",
            "requestID": "QRt1ZdFLGn0TCm784HmKafBmcB2lVaV81UU3fsOR3PtoIiM",
            "requestParameters": {
                "accountId": "-",
                "vaultName": "myVaultName"
            },
            "responseElements": null,
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "aws-sdk-java/1.9.6 Mac_OS_X/10.9.5 Java_HotSpot(TM)_64-
Bit_Server_VM/25.25-b02/1.8.0_25",
            "userIdentity": {
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "accountId": "999999999999",
                "arn": "arn:aws:iam::999999999999:user/myUserName",
                "principalId": "A1B2C3D4E5F6G7EXAMPLE",
                "type": "IAMUser",
```

```
                "userName": "myUserName"
            }
        }
    ]
}
```

Amazon Glacier를 위한 API 참조

Amazon Glacier는 서비스와 상호 작용할 수 있도록 작업 세트, 특히 RESTful API 직접 호출 세트를 지원합니다.

HTTP 요청을 보낼 수 있는 프로그래밍 라이브러리를 사용하여 사용자의 REST 요청을 Amazon Glacier로 전송할 수 있습니다. REST 요청을 Amazon Glacier로 전송할 때는 요청에 서명하여 모든 요청을 인증해야 합니다. 또한 아카이브를 업로드할 때는 페이로드 체크섬까지 계산하여 요청에 추가해야 합니다. 자세한 내용은 요청에 서명하기 섹션을 참조하세요.

오류가 발생할 경우에는 오류를 처리할 있도록 Amazon Glacier가 오류 응답으로 보내는 정보를 알아야 합니다. 이 섹션에서는 직접 REST API를 호출할 수 있도록 REST 작업은 물론이고 이러한 정보까지 모두 살펴보도록 하겠습니다.

REST API 직접 호출을 직접 사용하거나, 래퍼 라이브러리를 제공하는 Amazon SDK를 사용할 수 있습니다. 이 라이브러리는 각각 전송되는 요청에 서명하고 페이로드 체크섬을 계산합니다. 따라서 Amazon SDK를 사용할 때 코딩 작업이 더욱 간편합니다. 이 개발자 안내서에서는 AWS SDK for Java 및 .NET을 사용하는 기본 Amazon Glacier 작업 예시를 제공합니다. 자세한 내용은 Amazon Glacier에서 AWS SDKs 사용 섹션을 참조하세요.

주제

- 공통 요청 헤더
- 공통 응답 헤더
- 요청에 서명하기
- 체크섬 계산
- 오류 응답
- 볼트 작업
- 아카이브 작업
- 멀티파트 업로드 작업
- 작업
- 작업에서 사용되는 데이터 형식
- 데이터 가져오기 작업

공통 요청 헤더

Amazon Glacier(Amazon Glacier) REST 요청에는 요청에 대한 기본 정보가 있는 헤더가 포함됩니다. 다음 표에서는 모든 Amazon Glacier REST 요청에서 사용할 수 있는 헤더를 설명합니다.

헤더 이름	설명	필수
`Authorization`	요청에 서명하는 데 필요한 헤더입니다. Amazon Glacier에는 서명 버전 4가 필요합니다. 자세한 내용은 요청에 서명하기 섹션을 참조하세요. 유형: 문자열	예
`Content-Length`	요청 본문(헤더 제외) 길이입니다. 유형: 문자열 조건: 아카이브 업로드(POST archive) API에만 필요합니다.	조건부
`Date`	`Authorization` 헤더에 포함되는 서명을 생성할 때 사용할 수 있는 날짜입니다. 서명할 때 `Date` 헤더를 사용해야 하는 경우에는 ISO 8601 기본 형식에 따라 지정해야 합니다. 이때 `x-amz-date` 헤더는 필요하지 않습니다. `x-amz-date` 가 있으면 항상 `Date` 헤더 값보다 우선하기 때문입니다. 서명할 때 Date 헤더가 사용되지 않는 경우 RFC 2616 섹션 3.3에 지정된 전체 날짜 형식 중 하나일 수 있습니다. 예를 들어 다음의 날짜/시간 `Wed, 10 Feb 2017 12:00:00 GMT`는 Amazon Glacier에서 사용할 수 있는 유효한 날짜/시간 헤더입니다. 서명에 `Date` 헤더를 사용할 때는 ISO 8601 기본 `YYYYMMDD'T'HHMMSS'Z'` 형식을 따라야 합니다. 유형: 문자열	조건부

헤더 이름	설명	필수
	조건: Date를 지정하면서 ISO 8601 기본 형식을 따르지 않는 경우에는 x-amz-date 헤더도 포함해야 합니다. ISO 8601 기본 형식에 따라 Date를 지정하는 경우에는 이 헤더만으로도 요청 서명이 가능하므로 x-amz-date 헤더가 필요 없습니다. 자세한 정보는 Amazon Web Services 용어집의 서명 버전 4의 날짜 처리 섹션을 참조하세요.	
Host	이 헤더는 요청을 전송하는 서비스 엔드포인트를 지정합니다. 헤더 값은 'glacier.*region*.amazonaws.com ' 형식을 따라야 하며, 여기에서 *region*은 us-west-2 와 같은 AWS 리전 지정으로 치환됩니다. 유형: 문자열	예
x-amz-content-sha256	아카이브 업로드(POST archive) 또는 파트 업로드(PUT uploadID)로 업로드되는 전체 페이로드에 대해 계산된 SHA256 체크섬입니다. 이 헤더는 x-amz-sha256-tree-hash 헤더와 같지 않지만 일부 작은 페이로드의 경우에는 값이 동일합니다. x-amz-content-sha256 이 필요한 경우에는 x-amz-content-sha256 과 x-amz-sha256-tree-hash 를 모두 지정해야 합니다. 유형: 문자열 조건: API, 아카이브 업로드(POST archive) 및 파트 업로드(PUT uploadID)를 스트리밍하는 데 필요합니다.	조건부

헤더 이름	설명	필수
x-amz-date	Authorization 헤더에서 서명을 생성할 때 사용하는 날짜입니다. 형식은 ISO 8601 기본 형식(YYYYMMDD'T'HHMMSS'Z')을 따라야 합니다. 예를 들어 다음 날짜/시간 20170210T120000Z 는 Amazon Glacier에서 사용할 수 있는 유효한 x-amz-date 입니다. 유형: 문자열 조건: x-amz-date 는 모든 요청에서 옵션이지만 서명 요청에 사용되는 날짜보다 우선할 때 사용됩니다. Date 헤더가 ISO 8601 기본 형식으로 지정되면 x-amz-date 는 필요하지 않습니다. x-amz-date 가 있으면 항상 Date 헤더 값보다 우선하기 때문입니다. 자세한 정보는 Amazon Web Services 용어집의 서명 버전 4의 날짜 처리 섹션을 참조하세요.	조건부
x-amz-glacier-version	사용할 Amazon Glacier API 버전입니다. 현재 버전은 2012-06-01 입니다. 유형: 문자열	예
x-amz-sha256-tree-hash	업로드된 아카이브(아카이브 업로드(POST archive)) 또는 아카이브 파트(파트 업로드(PUT uploadID))에 대해 계산된 SHA256 트리-해시 체크섬입니다. 이 체크섬 계산에 대한 자세한 내용은 체크섬 계산 섹션을 참조하세요. 유형: 문자열 기본값: 없음 조건: 아카이브 업로드(POST archive) 및 파트 업로드(PUT uploadID)에 필요합니다.	조건부

공통 응답 헤더

다음은 대부분 API 응답에 공통적으로 나타나는 응답 헤더를 설명한 표입니다.

이름	설명
`Content-Length`	응답 본문의 길이(바이트)입니다. 유형: 문자열
`Date`	Amazon Glacier(Amazon Glacier)가 응답한 날짜 및 시간입니다(예: `Wed, 10 Feb 2017 12:00:00 GMT`). 날짜 형식은 RFC 2616 섹션 3.3에 지정된 전체 날짜 형식 중 하나여야 합니다. 반환되는 `Date`는 다른 날짜와 약간 다를 수도 있습니다. 예를 들어 아카이브 업로드(POST archive) 요청에서 반환되는 날짜는 아카이브 목적으로 볼트의 인벤토리 목록에 표시되는 날짜와 일치하지 않는 경우도 있습니다. 유형: 문자열
`x-amzn-RequestId`	사용자의 요청을 고유하게 식별하기 위해 Amazon Glacier에서 생성된 값입니다. Amazon Glacier에 문제가 있을 경우 AWS가 이 값을 사용하여 문제를 해결할 수 있습니다. 따라서 이 값을 기록해두는 것이 좋습니다. 유형: 문자열
`x-amz-sha256-tree-hash`	아카이브 또는 인벤토리 본문의 SHA256 트리-해시 체크섬입니다. 이 체크섬 계산에 대한 자세한 내용은 체크섬 계산 섹션을 참조하세요. 유형: 문자열

요청에 서명하기

Amazon Glacier는 사용자가 요청에 서명하여, 보낸 모든 요청을 인증하도록 요구합니다. 요청에 서명하려면 암호화 해시 함수를 이용해 디지털 서명을 계산해야 합니다. 암호화 해시는 입력을 근거로 하여 고유 해시 값을 반환하는 함수입니다. 해시 함수에 대한 입력에는 요청 텍스트와 시크릿 액세스 키가 포함됩니다. 해시 함수는 요청에 서명으로 포함하는 해시 값을 반환합니다. 서명은 요청에서 `Authorization` 헤더의 일부입니다.

Amazon Glacier는 요청을 수신한 후 사용자가 요청에 서명할 때 사용한 것과 동일한 해시 함수 및 입력을 사용하여 서명을 재계산합니다. 결과 서명이 요청 서명과 일치할 경우 Amazon Glacier가 요청을 처리합니다. 그렇지 않으면 요청이 거부됩니다.

Amazon Glacier는 AWS 서명 버전 4를 사용한 인증을 지원합니다. 서명을 계산하기 위한 프로세스는 다음 세 작업으로 나뉠 수 있습니다.

- 작업 1: 표준 요청 생성

  HTTP 요청을 정규 형식으로 재배열합니다. 정규 형식을 사용해야 하는 이유는 Amazon Glacier가 서명을 재계산하여 사용자가 전송한 서명과 비교할 때 동일한 정규 형식을 사용하기 때문입니다.
- 작업 2: 서명할 문자열 생성

  암호화 해시 함수에 대한 입력 값 중 하나로 사용할 문자열을 만듭니다. 서명할 문자열이라는 문자열은 해시 알고리즘의 이름, 요청 날짜, 자격 증명 범위 문자열, 이전 작업에서 정규화된 요청을 연결한 것입니다. 자격 증명 범위 문자열 자체는 날짜, AWS 리전 및 서비스 정보의 연결입니다.
- 작업 3: 서명 생성

  서명할 문자열과 파생된 키의 두 입력 문자열을 허용하는 암호화 해시 함수를 사용하여 요청에 대한 서명을 만듭니다. 파생된 키는 시크릿 액세스 키로 시작해서 자격 증명 범위 문자열을 사용하여 일련의 해시 기반 메시지 인증 코드(HMAC)를 만들어 계산됩니다. 단, 이 서명 단계에서 사용하는 해시 함수는 데이터 업로드를 위해 Amazon Glacier API에서 사용하는 트리 해시 알고리즘이 아닙니다.

주제

- 서명 계산 예시
- 스트리밍 작업을 위한 서명 계산

서명 계산 예시

다음 예시에서는 볼트 만들기(PUT vault)에 대해 서명을 생성하는 세부 과정을 안내합니다. 이 예시는 서명 계산 방법을 점검하기 위한 참조로 사용할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS API 요청 서명을 참조하세요.

이 예시에서는 다음과 같이 가정합니다.

- 요청 타임스탬프는 `Fri, 25 May 2012 00:24:53 GMT`입니다.
- 엔드포인트는 미국 동부(버지니아 북부) 리전인 `us-east-1`입니다.

일반 요청 구문(JSON 본문 포함)은 다음과 같습니다.

```
PUT /-/vaults/examplevault HTTP/1.1
Host: glacier.us-east-1.amazonaws.com
Date: Fri, 25 May 2012 00:24:53 GMT
Authorization: SignatureToBeCalculated
x-amz-glacier-version: 2012-06-01
```

작업 1: 표준 요청 생성에서 계산되는 요청의 정규 형식은 다음과 같습니다.

```
PUT
/-/vaults/examplevault

host:glacier.us-east-1.amazonaws.com
x-amz-date:20120525T002453Z
x-amz-glacier-version:2012-06-01

host;x-amz-date;x-amz-glacier-version
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

표준 요청의 마지막 줄은 요청 본문의 해시입니다. 또한 표준 요청에서 비어 있는 세 번째 줄에 주의해야 합니다. 비어 있는 이유는 이 API에 대해 쿼리 파라미터가 없기 때문입니다.

작업 2: 서명할 문자열 생성에서 서명할 문자열은 다음과 같습니다.

```
AWS4-HMAC-SHA256
20120525T002453Z
20120525/us-east-1/glacier/aws4_request
5f1da1a2d0feb614dd03d71e87928b8e449ac87614479332aced3a701f916743
```

서명할 문자열의 첫째 줄은 알고리즘, 둘째 줄은 타임스탬프, 셋째 줄은 자격 증명 범위, 마지막 줄은 작업 1: 표준 요청 생성의 표준 요청에 대한 해시입니다. 자격 증명 범위에서 사용하는 서비스 이름은 `glacier`입니다.

작업 3: 서명 생성에서 파생된 키는 다음과 같이 표현할 수 있습니다.

```
derived key = HMAC(HMAC(HMAC(HMAC("AWS4" + YourSecretAccessKey,"20120525"),"us-
east-1"),"glacier"),"aws4_request")
```

시크릿 액세스 키인 wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY를 사용하는 경우, 계산된 서명은 다음과 같습니다.

```
3ce5b2f2fffac9262b4da9256f8d086b4aaf42eba5f111c21681a65a127b7c2a
```

마지막 단계는 Authorization 헤더를 생성하는 것입니다. 데모용 액세스 키 AKIAIOSFODNN7EXAMPLE의 경우 헤더는 다음과 같습니다(가독성을 높이기 위해 줄 바꿈을 추가함).

```
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-east-1/
glacier/aws4_request,
SignedHeaders=host;x-amz-date;x-amz-glacier-version,
Signature=3ce5b2f2fffac9262b4da9256f8d086b4aaf42eba5f111c21681a65a127b7c2a
```

스트리밍 작업을 위한 서명 계산

아카이브 업로드(POST archive)와 파트 업로드(PUT uploadID)는 요청을 서명하고 전송할 때 x-amz-content-sha256 헤더를 추가해야 하는 스트리밍 작업입니다. 스트리밍 작업의 서명 단계는 스트리밍 헤더를 추가하는 것만 제외하고 다른 작업의 서명 단계와 정확히 일치합니다.

스트리밍 헤더인 x-amz-content-sha256의 계산은 업로드할 전체 내용(페이로드)의 SHA256 해시에 따라 달라집니다. 단, 이 계산은 SHA256 트리-해시(체크섬 계산)와 다릅니다. 일반적인 경우 외에도 페이로드 데이터의 SHA256 해시 값은 페이로드 데이터의 SHA256 트리-해시와 다릅니다.

페이로드 데이터가 바이트 배열로 지정되면 다음 Java 코드 조각을 사용하여 SHA256 해시를 계산할 수 있습니다.

```
public static byte[] computePayloadSHA256Hash2(byte[] payload) throws
 NoSuchAlgorithmException, IOException {
    BufferedInputStream bis =
       new BufferedInputStream(new ByteArrayInputStream(payload));
    MessageDigest messageDigest = MessageDigest.getInstance("SHA-256");
    byte[] buffer = new byte[4096];
    int bytesRead = -1;
    while ( (bytesRead = bis.read(buffer, 0, buffer.length)) != -1 ) {
        messageDigest.update(buffer, 0, bytesRead);
    }
    return messageDigest.digest();
}
```

마찬가지로 C#에서도 다음 코드 조각과 같이 페이로드 데이터의 SHA256 해시를 계산할 수 있습니다.

```
public static byte[] CalculateSHA256Hash(byte[] payload)
{
    SHA256 sha256 = System.Security.Cryptography.SHA256.Create();
    byte[] hash = sha256.ComputeHash(payload);

    return hash;
}
```

스트리밍 API의 서명 계산 예제

다음은 Amazon Glacier의 두 스트리밍 API 중 하나인 아카이브 업로드(POST archive) 작업에 사용할 서명을 생성하기 위한 세부 정보를 설명하는 예시입니다. 이 예시에서는 다음과 같이 가정합니다.

- 요청 타임스탬프는 `Mon, 07 May 2012 00:00:00 GMT`입니다.
- 엔드포인트는 미국 동부(버지니아 북부) 리전인 us-east-1입니다.
- 내용 페이로드는 "Welcome to Amazon Glacier" 문자열입니다.

일반 요청 구문(JSON 본문 포함)은 아래 예제와 같습니다. 예제를 보면 `x-amz-content-sha256` 헤더가 포함되어 있습니다. 또한 `x-amz-sha256-tree-hash`와 `x-amz-content-sha256`의 값이 동일합니다. 하지만 크기가 1MB를 넘는 아카이브를 업로드할 때는 그렇지 않습니다.

```
POST /-/vaults/examplevault HTTP/1.1
Host: glacier.us-east-1.amazonaws.com
Date: Mon, 07 May 2012 00:00:00 GMT
x-amz-archive-description: my archive
x-amz-sha256-tree-hash: SHA256 tree hash
x-amz-content-sha256: SHA256 payload hash
Authorization: SignatureToBeCalculated
x-amz-glacier-version: 2012-06-01
```

작업 1: 표준 요청 생성에서 계산되는 요청의 정규 형식은 아래와 같습니다. 예제를 보면 스트리밍 헤더인 `x-amz-content-sha256`에 값이 포함되어 있습니다. 이 말은 페이로드를 읽고 SHA256 해시를 먼저 계산한 후에 서명을 계산해야 한다는 것을 의미합니다.

```
POST
/-/vaults/examplevault
```

```
host:glacier.us-east-1.amazonaws.com
x-amz-content-sha256:726e392cb4d09924dbad1cc0ba3b00c3643d03d14cb4b823e2f041cff612a628
x-amz-date:20120507T000000Z
x-amz-glacier-version:2012-06-01

host;x-amz-content-sha256;x-amz-date;x-amz-glacier-version
726e392cb4d09924dbad1cc0ba3b00c3643d03d14cb4b823e2f041cff612a628
```

남은 서명 계산은 서명 계산 예시에 간략하게 설명되어 있는 단계를 따릅니다. 시크릿 액세스 키인 wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY과 액세스 키인 AKIAIOSFODNN7EXAMPLE을 사용하는 Authorization 헤더는 아래와 같습니다(가독성을 위해 줄바꿈이 추가됨).

```
Authorization=AWS4-HMAC-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20120507/us-east-1/glacier/aws4_request,
SignedHeaders=host;x-amz-content-sha256;x-amz-date;x-amz-glacier-version,
Signature=b092397439375d59119072764a1e9a144677c43d9906fd98a5742c57a2855de6
```

체크섬 계산

아카이브를 업로드할 때는 x-amz-sha256-tree-hash 및 x-amz-content-sha256 헤더를 모두 포함시켜야 합니다. x-amz-sha256-tree-hash 헤더는 요청 본문에서 페이로드 체크섬을 나타냅니다. 이 주제에서는 x-amz-sha256-tree-hash 헤더의 계산 방법에 대해서 설명하겠습니다. x-amz-content-sha256 헤더는 전체 페이로드의 해시이며, 권한을 부여하는 데 필요합니다. 자세한 내용은 스트리밍 API의 서명 계산 예제 단원을 참조하십시오.

요청 페이로드는 다음과 같습니다.

- 전체 아카이브: 아카이브 업로드 API를 사용하여 단일 요청으로 아카이브를 업로드할 때 요청 본문에서 전체 아카이브를 전송합니다. 이때는 전체 아카이브의 체크섬이 본문에 포함되어야 합니다.
- 아카이브 파트: 멀티파트 업로드 API를 사용하여 아카이브를 여러 파트로 나누어 업로드할 때는 요청 본문에서 아카이브 파트만 전송합니다. 이때는 아카이브 파트의 체크섬이 본문에 포함되어야 합니다. 모든 파트를 업로드한 후 멀티파트 업로드 완료 요청을 전송할 때는 전체 아카이브의 체크섬이 포함되어야 합니다.

페이로드 체크섬은 SHA256 트리-해시입니다. 트리-해시라고 불리는 이유는 체크섬을 계산하는 과정에서 SHA256 해시 값으로 이루어진 트리를 계산하기 때문입니다. 루트에 존재하는 해시 값이 전체 아카이브의 체크섬입니다.

> **Note**
>
> 이 섹션에서는 SHA256 트리-해시를 계산하는 방법에 대해서 다루지만 결과만 동일하다면 어떤 절차를 사용해도 상관없습니다.

SHA256 트리-해시의 계산 방법은 다음과 같습니다.

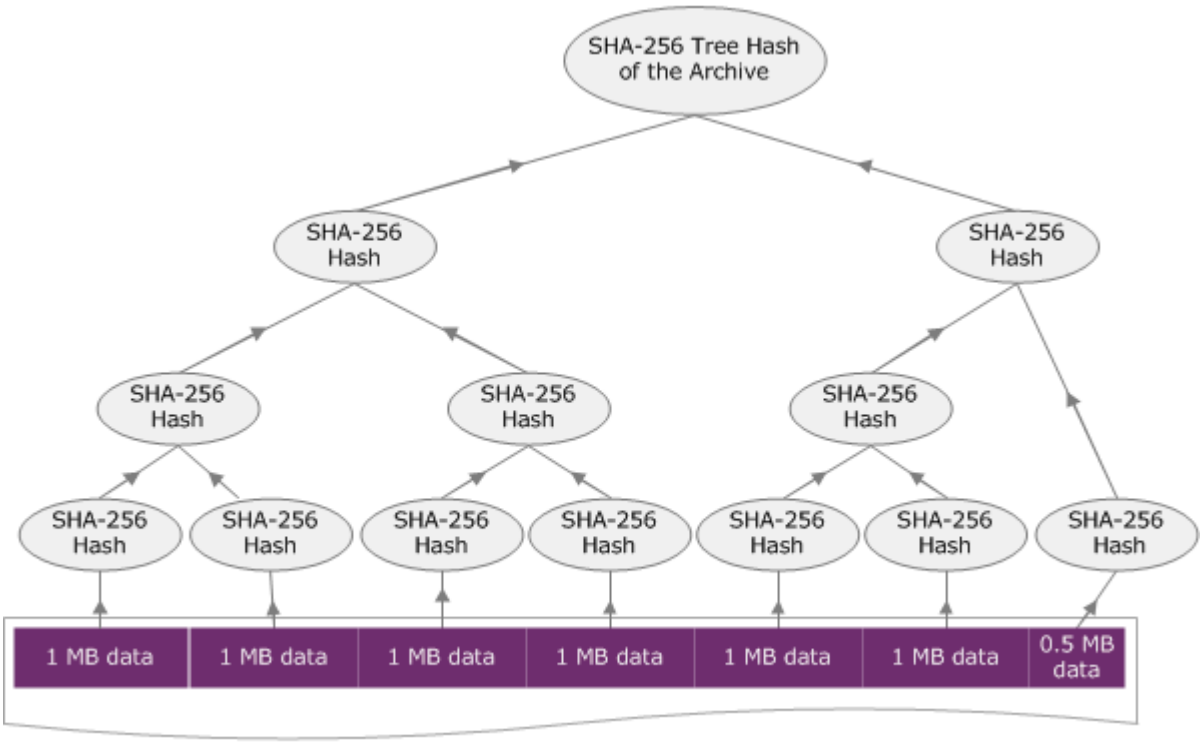
1. 1MB 청크 단위마다 페이로드 데이터의 SHA256 해시를 계산합니다. 마지막 청크 데이터는 1MB보다 작을 수 있습니다. 예를 들어 3.2MB 아카이브를 업로드할 경우 처음 1MB 청크 단위 3개에 대해 각각 SHA256 해시 값을 계산한 후 나머지 0.2MB 데이터의 SHA256 해시 값을 계산합니다. 이러한 해시 값들이 트리의 리프 노드를 형성합니다.
2. 트리에서 다음 계층을 만듭니다.
   a. 이어지는 하위 노드 해시 값 2개를 연결하고 연결된 해시 값의 SHA256 해시를 계산합니다. 이렇게 SHA256 해시를 연결 및 생성하여 하위 노드 2개에 대한 상위 노드를 만듭니다.
   b. 하위 노드가 한 개만 남게 되었을 때는 해당 해시 값을 트리의 다음 레벨로 승격합니다.
3. 최종 트리가 루트가 될 때까지 2단계를 반복합니다. 트리의 루트는 전체 아카이브의 해시 값에 해당하며, 그에 따른 하위 트리의 루트는 멀티파트 업로드에서 각 파트의 해시 값에 해당합니다.

주제

- 트리-해시 예제 1: 아카이브의 단일 요청 업로드
- 트리-해시 예제 2: 아카이브의 멀티파트 업로드
- 파일의 트리-해시 계산
- 데이터 다운로드 시 체크섬 수신

트리-해시 예제 1: 아카이브의 단일 요청 업로드

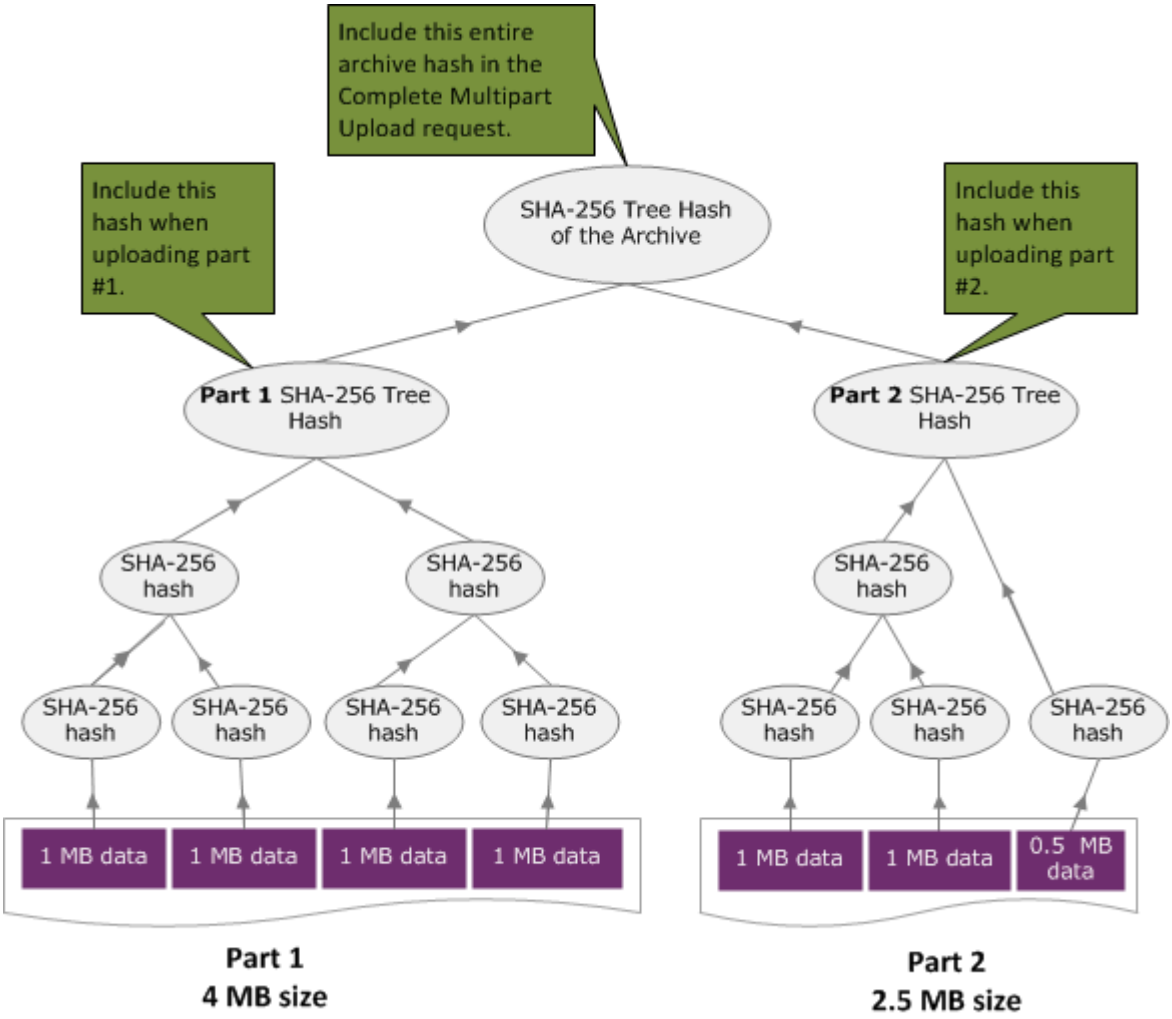
아카이브 업로드 API(아카이브 업로드(POST archive) 참조)를 사용하여 아카이브를 단일 요청으로 업로드할 때는 요청 페이로드에 전체 아카이브가 포함됩니다. 따라서 `x-amz-sha256-tree-hash` 요청 헤더에도 전체 아카이브의 트리-해시가 포함되어야 합니다. 예를 들어 6.5MB 아카이브를 업로드한다고 가정하겠습니다. 다음 다이어그램은 아카이브의 SHA256 해시를 구하는 프로세스입니다. 아카이브를 읽고 1MB 청크씩 SHA256 해시 값을 계산합니다. 나머지 0.5MB 데이터의 해시 값도 계산한 후 이전 절차에서 설명한 대로 트리를 만듭니다.



6.5 MB Archive

트리-해시 예제 2: 아카이브의 멀티파트 업로드

멀티파트 업로드를 사용해 아카이브를 업로드할 경우 트리-해시의 계산 과정도 단일 요청으로 아카이브를 업로드할 때와 동일합니다. 멀티파트 업로드는 각 요청(파트 업로드(PUT uploadID) API 사용)마다 아카이브 파트만 업로드한다는 점만 유일하게 다릅니다. 따라서 각 파트에 해당하는 체크섬만 `x-amz-sha256-tree-hash` 요청 헤더에 입력합니다. 하지만 모든 파트를 업로드한 후에는 `x-amz-sha256-tree-hash` 요청 헤더에 전체 아카이브의 트리-해시를 입력하여 멀티파트 업로드 완료(멀티파트 업로드 완료(POST uploadID) 참조) 요청을 전송해야 합니다.



파일의 트리-해시 계산

여기에서 소개하는 알고리즘은 설명을 목적으로 선택되었습니다. 구현 시나리오에 따라 코드를 알맞게 최적화할 수 있습니다. 예를 들어 Amazon Glacier(Amazon Glacier)에 대해 Amazon SDK를 사용하여 프로그래밍하는 경우에는 트리 해시 계산이 자동으로 이루어지기 때문에 사용자는 파일 참조만 제공하면 됩니다.

Example 1: Java 예제

다음은 Java를 사용하여 파일의 SHA256 트리-해시를 계산하는 방법을 나타낸 예제입니다. 이 예제는 파일 위치를 인수로 입력하여 실행하거나, 혹은 코드에서 `TreeHashExample.computeSHA256TreeHash` 메서드를 직접 사용할 수도 있습니다.

```
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
```

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class TreeHashExample {

static final int ONE_MB = 1024 * 1024;

    /**
     * Compute the Hex representation of the SHA-256 tree hash for the specified
     * File
     *
     * @param args
     *            args[0]: a file to compute a SHA-256 tree hash for
     */
    public static void main(String[] args) {

        if (args.length < 1) {
            System.err.println("Missing required filename argument");
            System.exit(-1);
        }

        File inputFile = new File(args[0]);
        try {

            byte[] treeHash = computeSHA256TreeHash(inputFile);
            System.out.printf("SHA-256 Tree Hash = %s\n", toHex(treeHash));

        } catch (IOException ioe) {
            System.err.format("Exception when reading from file %s: %s", inputFile,
                    ioe.getMessage());
            System.exit(-1);

        } catch (NoSuchAlgorithmException nsae) {
            System.err.format("Cannot locate MessageDigest algorithm for SHA-256: %s",
                    nsae.getMessage());
            System.exit(-1);
        }
    }

    /**
     * Computes the SHA-256 tree hash for the given file
     *
     * @param inputFile
     *            a File to compute the SHA-256 tree hash for
```

```
     * @return a byte[] containing the SHA-256 tree hash
     * @throws IOException
     *             Thrown if there's an issue reading the input file
     * @throws NoSuchAlgorithmException
     */
    public static byte[] computeSHA256TreeHash(File inputFile) throws IOException,
            NoSuchAlgorithmException {

        byte[][] chunkSHA256Hashes = getChunkSHA256Hashes(inputFile);
        return computeSHA256TreeHash(chunkSHA256Hashes);
    }

    /**
     * Computes a SHA256 checksum for each 1 MB chunk of the input file. This
     * includes the checksum for the last chunk even if it is smaller than 1 MB.
     *
     * @param file
     *            A file to compute checksums on
     * @return a byte[][] containing the checksums of each 1 MB chunk
     * @throws IOException
     *             Thrown if there's an IOException when reading the file
     * @throws NoSuchAlgorithmException
     *             Thrown if SHA-256 MessageDigest can't be found
     */
    public static byte[][] getChunkSHA256Hashes(File file) throws IOException,
            NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");

        long numChunks = file.length() / ONE_MB;
        if (file.length() % ONE_MB > 0) {
            numChunks++;
        }

        if (numChunks == 0) {
            return new byte[][] { md.digest() };
        }

        byte[][] chunkSHA256Hashes = new byte[(int) numChunks][];
        FileInputStream fileStream = null;

        try {
            fileStream = new FileInputStream(file);
            byte[] buff = new byte[ONE_MB];
```

```
            int bytesRead;
            int idx = 0;
            int offset = 0;

            while ((bytesRead = fileStream.read(buff, offset, ONE_MB)) > 0) {
                md.reset();
                md.update(buff, 0, bytesRead);
                chunkSHA256Hashes[idx++] = md.digest();
                offset += bytesRead;
            }

            return chunkSHA256Hashes;

        } finally {
            if (fileStream != null) {
                try {
                    fileStream.close();
                } catch (IOException ioe) {
                    System.err.printf("Exception while closing %s.\n %s",
file.getName(),
                            ioe.getMessage());
                }
            }
        }
    }

    /**
     * Computes the SHA-256 tree hash for the passed array of 1 MB chunk
     * checksums.
     *
     * This method uses a pair of arrays to iteratively compute the tree hash
     * level by level. Each iteration takes two adjacent elements from the
     * previous level source array, computes the SHA-256 hash on their
     * concatenated value and places the result in the next level's destination
     * array. At the end of an iteration, the destination array becomes the
     * source array for the next level.
     *
     * @param chunkSHA256Hashes
     *            An array of SHA-256 checksums
     * @return A byte[] containing the SHA-256 tree hash for the input chunks
     * @throws NoSuchAlgorithmException
     *             Thrown if SHA-256 MessageDigest can't be found
     */
```

```
    public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
            throws NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");

        byte[][] prevLvlHashes = chunkSHA256Hashes;

        while (prevLvlHashes.length > 1) {

            int len = prevLvlHashes.length / 2;
            if (prevLvlHashes.length % 2 != 0) {
                len++;
            }

            byte[][] currLvlHashes = new byte[len][];

            int j = 0;
            for (int i = 0; i < prevLvlHashes.length; i = i + 2, j++) {

                // If there are at least two elements remaining
                if (prevLvlHashes.length - i > 1) {

                    // Calculate a digest of the concatenated nodes
                    md.reset();
                    md.update(prevLvlHashes[i]);
                    md.update(prevLvlHashes[i + 1]);
                    currLvlHashes[j] = md.digest();

                } else { // Take care of remaining odd chunk
                    currLvlHashes[j] = prevLvlHashes[i];
                }
            }

            prevLvlHashes = currLvlHashes;
        }

        return prevLvlHashes[0];
    }

    /**
     * Returns the hexadecimal representation of the input byte array
     *
     * @param data
     *            a byte[] to convert to Hex characters
```

```
     * @return A String containing Hex characters
     */
    public static String toHex(byte[] data) {
        StringBuilder sb = new StringBuilder(data.length * 2);

        for (int i = 0; i < data.length; i++) {
            String hex = Integer.toHexString(data[i] & 0xFF);

            if (hex.length() == 1) {
                // Append leading zero.
                sb.append("0");
            }
            sb.append(hex);
        }
        return sb.toString().toLowerCase();
    }
}
```

Example 2: C# .NET 예제

다음은 파일의 SHA256 트리-해시를 계산하는 방법을 나타낸 예제입니다. 이 예제는 파일 위치를 인수로 입력하여 실행할 수 있습니다.

```
using System;
using System.IO;

using System.Security.Cryptography;

namespace ExampleTreeHash
{
    class Program
    {
        static int ONE_MB = 1024 * 1024;

        /**
        * Compute the Hex representation of the SHA-256 tree hash for the
        * specified file
        *
        * @param args
        *            args[0]: a file to compute a SHA-256 tree hash for
        */
        public static void Main(string[] args)
        {
```

```
            if (args.Length < 1)
            {
                Console.WriteLine("Missing required filename argument");
                Environment.Exit(-1);
            }
            FileStream inputFile = File.Open(args[0], FileMode.Open, FileAccess.Read);
            try
            {
                byte[] treeHash = ComputeSHA256TreeHash(inputFile);
                Console.WriteLine("SHA-256 Tree Hash = {0}",
BitConverter.ToString(treeHash).Replace("-", "").ToLower());
                Console.ReadLine();
                Environment.Exit(-1);
            }
            catch (IOException ioe)
            {
                Console.WriteLine("Exception when reading from file {0}: {1}",
                    inputFile, ioe.Message);
                Console.ReadLine();
                Environment.Exit(-1);
            }
            catch (Exception e)
            {
                Console.WriteLine("Cannot locate MessageDigest algorithm for SHA-256:
{0}",
                    e.Message);
                Console.WriteLine(e.GetType());
                Console.ReadLine();
                Environment.Exit(-1);
            }
            Console.ReadLine();
        }


        /**
         * Computes the SHA-256 tree hash for the given file
         *
         * @param inputFile
         *            A file to compute the SHA-256 tree hash for
         * @return a byte[] containing the SHA-256 tree hash
         */
        public static byte[] ComputeSHA256TreeHash(FileStream inputFile)
        {
            byte[][] chunkSHA256Hashes = GetChunkSHA256Hashes(inputFile);
```

```
            return ComputeSHA256TreeHash(chunkSHA256Hashes);
        }


        /**
         * Computes a SHA256 checksum for each 1 MB chunk of the input file. This
         * includes the checksum for the last chunk even if it is smaller than 1 MB.
         *
         * @param file
         *            A file to compute checksums on
         * @return a byte[][] containing the checksums of each 1MB chunk
         */
        public static byte[][] GetChunkSHA256Hashes(FileStream file)
        {
            long numChunks = file.Length / ONE_MB;
            if (file.Length % ONE_MB > 0)
            {
                numChunks++;
            }

            if (numChunks == 0)
            {
                return new byte[][] { CalculateSHA256Hash(null, 0) };
            }
            byte[][] chunkSHA256Hashes = new byte[(int)numChunks][];

            try
            {
                byte[] buff = new byte[ONE_MB];

                int bytesRead;
                int idx = 0;

                while ((bytesRead = file.Read(buff, 0, ONE_MB)) > 0)
                {
                    chunkSHA256Hashes[idx++] = CalculateSHA256Hash(buff, bytesRead);
                }
                return chunkSHA256Hashes;
            }
            finally
            {
                if (file != null)
                {
                    try
```

```
                {
                    file.Close();
                }
                catch (IOException ioe)
                {
                    throw ioe;
                }
            }
        }

    }

    /**
     * Computes the SHA-256 tree hash for the passed array of 1MB chunk
     * checksums.
     *
     * This method uses a pair of arrays to iteratively compute the tree hash
     * level by level. Each iteration takes two adjacent elements from the
     * previous level source array, computes the SHA-256 hash on their
     * concatenated value and places the result in the next level's destination
     * array. At the end of an iteration, the destination array becomes the
     * source array for the next level.
     *
     * @param chunkSHA256Hashes
     *            An array of SHA-256 checksums
     * @return A byte[] containing the SHA-256 tree hash for the input chunks
     */
    public static byte[] ComputeSHA256TreeHash(byte[][] chunkSHA256Hashes)
    {
        byte[][] prevLvlHashes = chunkSHA256Hashes;
        while (prevLvlHashes.GetLength(0) > 1)
        {

            int len = prevLvlHashes.GetLength(0) / 2;
            if (prevLvlHashes.GetLength(0) % 2 != 0)
            {
                len++;
            }

            byte[][] currLvlHashes = new byte[len][];

            int j = 0;
            for (int i = 0; i < prevLvlHashes.GetLength(0); i = i + 2, j++)
            {
```

```
                    // If there are at least two elements remaining
                    if (prevLvlHashes.GetLength(0) - i > 1)
                    {

                        // Calculate a digest of the concatenated nodes
                        byte[] firstPart = prevLvlHashes[i];
                        byte[] secondPart = prevLvlHashes[i + 1];
                        byte[] concatenation = new byte[firstPart.Length +
 secondPart.Length];
                        System.Buffer.BlockCopy(firstPart, 0, concatenation, 0,
 firstPart.Length);
                        System.Buffer.BlockCopy(secondPart, 0, concatenation,
 firstPart.Length, secondPart.Length);

                        currLvlHashes[j] = CalculateSHA256Hash(concatenation,
 concatenation.Length);

                    }
                    else
                    { // Take care of remaining odd chunk
                        currLvlHashes[j] = prevLvlHashes[i];
                    }
                }

                prevLvlHashes = currLvlHashes;
            }

            return prevLvlHashes[0];
        }

        public static byte[] CalculateSHA256Hash(byte[] inputBytes, int count)
        {
            SHA256 sha256 = System.Security.Cryptography.SHA256.Create();
            byte[] hash = sha256.ComputeHash(inputBytes, 0, count);
            return hash;
        }
    }
}
```

데이터 다운로드 시 체크섬 수신

작업 시작 API(작업 시작(POST jobs) 참조)를 사용하여 아카이브를 가져올 때는 옵션으로 아카이브를 가져오는 범위를 지정할 수 있습니다. 마찬가지로 작업 출력 가져오기 API(작업 출력 가져오기(GET output) 참조)를 사용하여 데이터를 다운로드할 때도 옵션으로 다운로드할 데이터 범위를 지정할 수 있습니다. 이러한 범위에는 아카이브 데이터를 가져오거나 다운로드할 때 반드시 알고 있어야 할 두 가지 특성이 있습니다. 가져오기 범위는 아카이브에 대해 메가바이트 정렬로 이루어져야 합니다. 또한 데이터를 다운로드할 때 체크섬 값을 수신하려면 가져오기 범위와 다운로드 범위 모두 트리-해시 정렬로 이루어져야 합니다. 이러한 두 가지 유형의 범위 정렬은 아래와 같이 정의할 수 있습니다.

- 메가바이트 정렬: StartBytes를 1MB로 나눌 수 있고 EndBytes에 1을 더한 값을 1MB로 나눌 수 있거나 지정된 아카이브의 끝(아카이브 바이트 크기에서 1을 뺀 값)과 같을 때만 범위[StartByte, EndBytes]가 메가바이트(1024*1024) 로 정렬됩니다. 작업 시작 API에서 사용하는 범위는 지정하는 경우에 한해 메가바이트 정렬로 이루어져야 합니다.
- 트리-해시 정렬: 범위에서 만들어진 트리-해시의 루트가 전체 아카이브의 트리-해시 노드에 해당할 경우, 그리고 이러한 경우일 때만 아카이브에 대한 범위 [StartBytes, EndBytes]가 트리-해시로 정렬됩니다. 데이터를 다운로드하면서 체크섬 값을 수신하려면 가져오기 범위와 다운로드 범위 모두 트리-해시 정렬로 이루어져야 합니다. 아카이브 트리-해시에 대한 범위 및 범위 관계 예는 트리-해시 예제: 트리-해시로 정렬하여 아카이브 범위 가져오기 섹션을 참조하세요.

  트리-해시로 정렬되는 범위는 메가바이트로도 정렬됩니다. 하지만 메가바이트로 정렬된 범위가 꼭 트리-해시로 정렬될 필요는 없습니다.

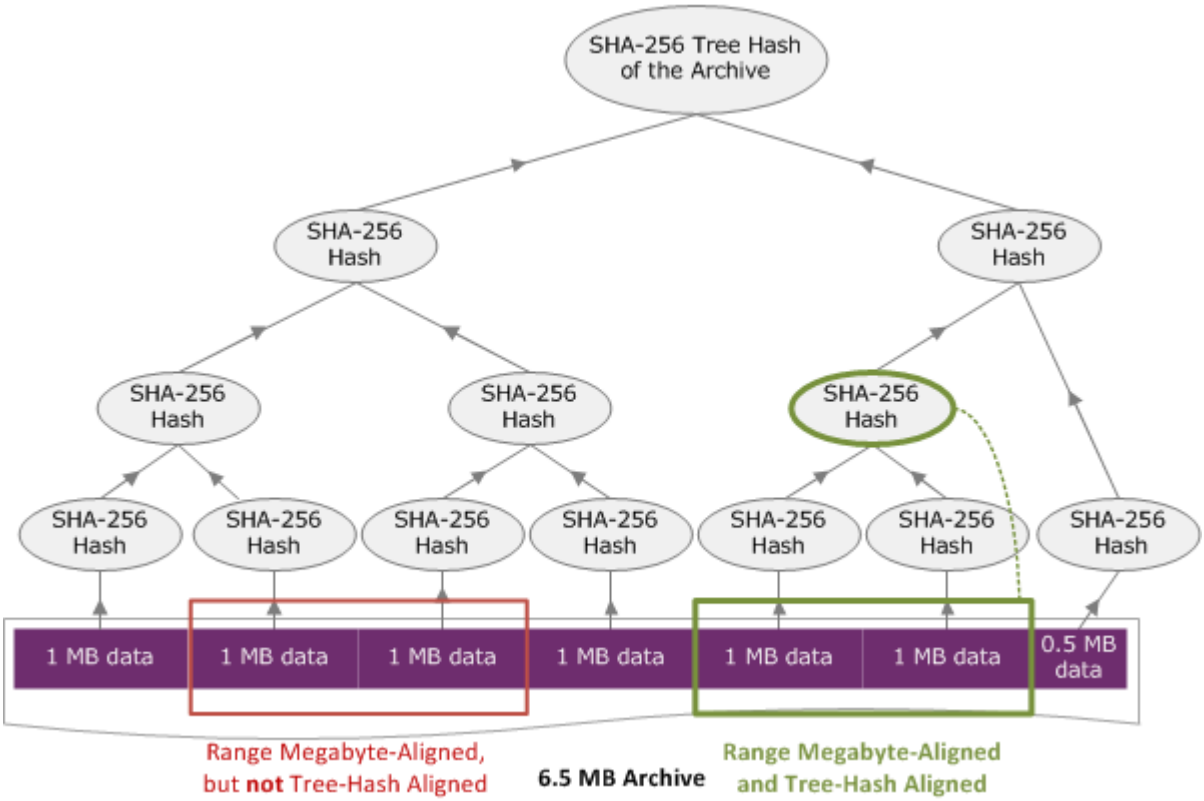
다음은 아카이브 데이터를 다운로드하면서 체크섬 값을 수신하는 경우에 대한 설명입니다.

- 작업 시작 요청에서 가져오기 범위를 지정하지 않고 작업 가져오기 요청에서 전체 아카이브를 다운로드하는 경우
- 작업 시작 요청에서 가져오기 범위를 지정하지 않고 작업 가져오기 요청에서 다운로드할 트리-해시 정렬 범위를 지정하는 경우
- 작업 시작 요청에서 가져올 트리-해시 정렬 범위를 지정하고 작업 가져오기 요청에서 전체 범위를 다운로드하는 경우
- 작업 시작 요청에서 가져올 트리-해시 정렬 범위를 지정하고 작업 가져오기 요청에서 다운로드할 트리-해시 정렬 범위를 지정하는 경우

작업 시작 요청에서 트리-해시로 정렬하지 않고 가져올 범위를 지정하더라도 아카이브 데이터를 가져올 수는 있지만 작업 가져오기 요청에서 데이터를 다운로드할 때 체크섬 값은 반환되지 않습니다.

트리-해시 예제: 트리-해시로 정렬하여 아카이브 범위 가져오기

볼트에 6.5MB의 아카이브가 저장되어 있으며 여기에서 2MB를 가져온다고 가정하겠습니다. 이때는 작업 시작하기 요청에서 2MB 범위의 지정 방식에 따라 데이터 다운로드 시 데이터 체크섬 값의 수신 여부가 결정됩니다. 다음은 6.5MB 아카이브에서 다운로드할 수 있는 두 가지 2MB 범위를 설명한 다이어그램입니다. 두 범위 모두 메가바이트로 정렬되지만 하나만 트리-해시로 정렬됩니다.



트리-해시 정렬 범위의 지정

이 섹션에서는 트리-해시 정렬 범위의 구성 요소를 정확하게 지정할 수 있는 방법에 대해서 설명합니다. 트리-해시 정렬 범위는 아카이브 일부를 다운로드하면서 가져올 데이터 범위와 가져온 데이터에서 다운로드할 범위를 지정할 때 매우 중요합니다. 이 두 범위를 트리-해시로 정렬하는 경우에는 데이터 다운로드 시 체크섬 데이터까지 수신하게 됩니다.

새 트리 해시가 [A, B]에서 만들어지고, 해당 범위의 트리 해시의 루트가 전체 아카이브의 트리 해시 노드에 해당할 경우, 그리고 이러한 경우일 때만 아카이브에 대한 범위 [A, B]가 트리 해시로 정렬됩니다. 트리-해시 예제: 트리-해시로 정렬하여 아카이브 범위 가져오기의 다이어그램에서 이러한 예를 볼 수 있습니다. 이 섹션에서는 트리-해시 정렬을 위한 지정 방법에 대해서 살펴보겠습니다.

이번에는 [P, Q)가 N메가바이트(MB)의 아카이브에 대한 범위 쿼리이고, 여기에서 P와 Q는 1MB의 승수라고 가정합니다. 이때 실제로 포함되는 범위는 [PMB, QMB-1바이트]이지만, 여기에서는 간단명료하게 [P, Q)로 나타냈습니다. 이를 감안하여 다음 내용을 살펴보십시오.

- P가 홀수인 경우 가능한 트리 해시 정렬 범위는 한 가지, 즉 [P, P+1MB)가 유일합니다.
- P가 짝수이고 k가 최대 수인 경우 P는 2k*X로 쓸 수 있고, P로 시작하는 트리 해시 정렬 범위는 최대 k개 입니다. X는 0보다 큰 정수입니다. 결과적으로 트리-해시 정렬 범위는 다음 카테고리에 해당합니다.
  - 각 i에 대해 (0 <= i <= k)이고, $P + 2^i < N$인 경우, $[P, Q + 2^i)$는 트리-해시 정렬 범위에 해당합니다.
  - P = 0은 A = 2[lgN]*0인 특수한 경우입니다.

오류 응답

오류가 발생할 경우 API는 다음 예외 중 한 가지를 반환합니다.

코드	설명	HTTP 상태 코드	Type
`AccessDeniedException`	AWS Identity and Access Management (IAM) 정책에서 허용하지 않는 리소스에 액세스하려는 시도가 있거나 요청 URI에 잘못된 AWS 계정 ID가 사용된 경우 반환됩니다. 자세한 내용은 Amazon Glacier의 ID 및 액세스 관리(IAM) 단원을 참조하십시오.	`403 Forbidden`	클라이언트
`BadRequest`	요청을 처리할 수 없는 경우에 반환됩니다.	`400 Bad Request`	클라이언트
`ExpiredTokenException`	요청에서 사용한 보안 토큰이 만료된 경우에 반환됩니다.	`403 Forbidden`	클라이언트

코드	설명	HTTP 상태 코드	Type
InsufficientCapacityException	신속 요청을 처리할 수 있는 용량이 부족한 경우에 반환됩니다. 이 오류는 신속 가져오기에만 적용되며 표준 또는 벌크 가져오기에는 적용되지 않습니다.	503 Service Unavailable	Server
InvalidParameterValueException	요청 파라미터가 잘못 지정된 경우에 반환됩니다.	400 Bad Request	클라이언트
InvalidSignatureException	요청 서명이 잘못된 경우에 반환됩니다.	403 Forbidden	클라이언트
LimitExceededException	요청 결과가 볼트 제한, 태그 제한 또는 프로비저닝된 용량 제한 중 한 가지를 초과한 경우에 반환됩니다.	400 Bad Request	클라이언트
MissingAuthenticationTokenException	요청에 대한 인증 데이터를 찾을 수 없는 경우에 반환됩니다.	400 Bad Request	클라이언트
MissingParameterValueException	필수 헤더 또는 파라미터가 요청에서 누락된 경우에 반환됩니다.	400 Bad Request	클라이언트
PolicyEnforcedException	가져오기 작업이 현재 데이터 정책의 가져오기 속도 제한을 초과한 경우에 반환됩니다. 데이터 가져오기 정책에 대한 자세한 내용은 Amazon Glacier 데이터 검색 정책 섹션을 참조하세요.	400 Bad Request	클라이언트

코드	설명	HTTP 상태 코드	Type
ResourceNotFoundException	볼트, 업로드 ID, 작업 ID 등 지정된 리소스가 존재하지 않는 경우에 반환됩니다.	404 Not Found	클라이언트
RequestTimeoutException	아카이브를 업로드할 때 Amazon Glacier(Amazon Glacier)가 업로드를 수신하는 동안 시간이 초과되는 경우에 반환됩니다.	408 Request Timeout	클라이언트
SerializationException	요청 본문이 잘못된 경우에 반환됩니다. JSON 페이로드를 포함하는 경우에는 형식이 올바른지 확인해야 합니다.	400 Bad Request	클라이언트
ServiceUnavailableException	서비스가 요청을 완료할 수 없는 경우에 반환됩니다.	500 Internal Server Error	Server
ThrottlingException	Amazon Glacier에 대한 요청 비율을 줄여야 하는 경우에 반환됩니다.	400 Bad Request	클라이언트
UnrecognizedClientException	액세스 키 ID 또는 보안 토큰이 잘못된 경우에 반환됩니다.	400 Bad Request	클라이언트

다양한 Amazon Glacier API가 동일한 예외를 반환하지만 예외 메시지가 다르기 때문에 특정 오류 문제를 해결하는 데 효과적입니다.

Amazon Glacier는 오류 정보를 응답 본문에 반환합니다. 다음은 오류 응답 일부를 나타내는 예제입니다.

예제 1: 존재하지 않는 작업 ID를 사용한 작업 요청 설명

존재하지 않는 작업에 대해 작업 설명(GET JobID) 요청을 전송한다고 가정하겠습니다. 즉, 존재하지 않는 작업 ID를 지정합니다.

```
GET /-/vaults/examplevault/jobs/HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVEXAMPLEbadJobID HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

그러면 Amazon Glacier가 다음과 같은 오류 응답을 반환합니다.

```
HTTP/1.1 404 Not Found
x-amzn-RequestId: AAABaZ9N92Iiyv4N7sru3ABEpSQkuFtmH3NP6aAC51ixfjg
Content-Type: application/json
Content-Length: 185
Date: Wed, 10 Feb 2017 12:00:00 GMT
{
  "code": "ResourceNotFoundException",
  "message": "The job ID was not found: HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVEXAMPLEbadJobID",
  "type": "Client"
  }
```

위치:

코드

일반 예외 중 하나입니다.

유형: 문자열

메시지

오류를 반환하는 API에 따른 일반적인 오류 조건 설명입니다.

유형: 문자열

유형

오류 출처입니다. 이 필드는 Client, Server 또는 Unknown 중 한 가지 값을 가질 수 있습니다.

유형: 문자열.

위의 응답에서 다음 사항에 유의할 필요가 있습니다.

- 오류 응답에서 Amazon Glacier는 상태 코드 값을 4xx와 5xx로 반환합니다. 이 예제에서 상태 코드는 404 Not Found입니다.
- Content-Type 헤더 값인 application/json은 본문의 JSON을 나타냅니다.
- 본문의 JSON은 오류 정보를 제공합니다.

이전 요청에서 잘못된 작업 ID 대신 존재하지 않은 볼트를 지정한다고 가정하겠습니다. 그러면 응답으로 다른 메시지가 반환됩니다.

```
HTTP/1.1 404 Not Found
x-amzn-RequestId: AAABBeC9Zw0rp_5D0L8VfB3FA_WlTupqTKAUehMcPhdgni0
Content-Type: application/json
Content-Length: 154
Date: Wed, 10 Feb 2017 12:00:00 GMT
{
  "code": "ResourceNotFoundException",
  "message": "Vault not found for ARN: arn:aws:glacier:us-west-2:012345678901:vaults/
examplevault",
  "type": "Client"
}
```

예제 2: 요청 파라미터에 잘못된 값이 포함된 작업 목록 조회 요청

이 예제에서는 작업 목록 조회(GET jobs) 요청을 전송하여 특정 statuscode가 있는 볼트 작업을 가져오려고 합니다. 이때 올바른 값인 InProgress, Succeeded 또는 Failed 대신 잘못된 statuscode 값인 finished를 입력합니다.

```
GET /-/vaults/examplevault/jobs?statuscode=finished HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
```

```
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

Amazon Glacier는 관련 메시지와 함께 `InvalidParameterValueException`을 반환합니다.

```
HTTP/1.1 400 Bad Request
x-amzn-RequestId: AAABaZ9N92Iiyv4N7sru3ABEpSQkuFtmH3NP6aAC51ixfjg
Content-Type: application/json
Content-Length: 141
Date: Wed, 10 Feb 2017 12:00:00 GMT
{
  "code": "InvalidParameterValueException",
  "message": "The job status code is not valid: finished",
  "type: "Client"
}
```

볼트 작업

다음은 Amazon Glacier에서 사용할 수 있는 볼트 작업입니다.

주제

- 볼트 잠금 중단(DELETE lock-policy)
- 볼트에 태그 추가(POST tags add)
- 볼트 만들기(PUT vault)
- 볼트 잠금 완료(POST lockId)
- 볼트 삭제(DELETE vault)
- 볼트 액세스 정책 삭제(DELETE access-policy)
- 볼트 알림 삭제(DELETE notification-configuration)
- 볼트 설명(GET vault)
- 볼트 액세스 정책 가져오기(GET access-policy)
- 볼트 잠금 가져오기(GET lock-policy)
- 볼트 알림 가져오기(GET notification-configuration)
- 볼트 잠금 시작(POST lock-policy)
- 볼트의 태그 목록 조회(GET tags)
- 볼트 목록 조회(GET vaults)

- 볼트에서 태그 삭제(POST tags remove)
- 볼트 액세스 정책 설정(PUT access-policy)
- 볼트 알림 구성 설정(PUT notification-configuration)

볼트 잠금 중단(DELETE lock-policy)

설명

이 작업에서는 볼트 잠금이 Locked 상태가 아닐 경우 볼트 잠금 프로세스를 중단합니다. 이 작업을 요청할 때 볼트 잠금이 Locked 상태일 경우에는 AccessDeniedException 오류가 반환됩니다. 볼트 잠금 프로세스를 중단하면 지정된 볼트에서 볼트 잠금 정책도 삭제됩니다.

InProgress를 호출하면 볼트 잠금이 볼트 잠금 시작(POST lock-policy) 상태로 전환됩니다. Locked를 호출하면 볼트 잠금이 볼트 잠금 완료(POST lockId) 상태로 전환됩니다. 볼트 잠금 상태는 볼트 잠금 가져오기(GET lock-policy)을 호출하면 알 수 있습니다. 볼트 잠금 프로세스에 대한 자세한 내용은 Amazon Glacier 볼트 잠금 섹션을 참조하세요. 볼트 잠금 정책에 대한 자세한 내용은 볼트 잠금 정책 섹션을 참조하세요.

이 작업은 멱등성을 갖습니다. 볼트 잠금이 InProgress 상태이거나 볼트에 연결된 정책이 없는 경우에는 이 작업을 여러 번 성공적으로 호출할 수 있습니다.

요청

볼트 잠금 정책을 삭제하려면 HTTP DELETE 요청을 볼트의 lock-policy 하위 리소스 URI로 전송합니다.

구문

```
DELETE /AccountId/vaults/vaultName/lock-policy HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 값은 AWS 계정 ID입니다. 이 값은 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID와 일치해야 합니다. AWS 계정 ID 또는 선택적으로 하나의 '-'(하이픈)을 지정할 수 있

습니다. 이 경우에 Amazon Glacier는 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 지정하는 경우 ID에 하이픈('-')을 포함하지 않습니다.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

이 작업에는 요청 본문이 없습니다.

응답

정책이 성공적으로 삭제되면 Amazon Glacier가 `HTTP 204 No Content` 응답을 반환합니다.

구문

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

이 작업은 응답 본문을 반환하지 않습니다.

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예시

다음은 볼트 잠금 프로세스를 중단하는 방법을 설명한 예시입니다.

요청 예시

이 예시에서 DELETE 요청은 볼트 이름이 **examplevault**인 lock-policy 하위 리소스로 전송됩니다.

```
DELETE /-/vaults/examplevault/lock-policy HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
x-amz-glacier-version: 2012-06-01
```

응답의 예

정책이 성공적으로 삭제되면 Amazon Glacier가 다음 예와 같이 HTTP 204 No Content 응답을 반환합니다.

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

관련 섹션

- 볼트 잠금 완료(POST lockId)
- 볼트 잠금 가져오기(GET lock-policy)
- 볼트 잠금 시작(POST lock-policy)

참고

언어 특정 Amazon SDK 중 하나에서 이 API를 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface

볼트에 태그 추가(POST tags add)

이 작업에서는 지정한 태그를 볼트에 추가합니다. 각 태그는 키와 값으로 구성됩니다. 각 볼트는 최대 50개까지 태그를 가질 수 있습니다. 볼트의 태그 제한을 초과하는 요청일 경우에는 `LimitExceededException` 오류가 반환됩니다.

지정한 키 아래의 볼트에 태그가 이미 있는 경우에는 기존 키 값을 덮어쓰게 됩니다. 태그에 대한 자세한 내용은 Amazon Glacier 리소스에 태그 지정 섹션을 참조하세요.

요청 구문

태그를 볼트에 추가하려면 다음 구문 예제와 같이 HTTP POST 요청을 태그 URI로 전송합니다.

```
POST /AccountId/vaults/vaultName/tags?operation=add HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
Content-Length: Length
x-amz-glacier-version: 2012-06-01

{
   "Tags":
      {
         "string": "string",
         "string": "string"
      }
}
```

> **Note**
>
> `AccountId` 값은 AWS 계정 ID입니다. 이 값은 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID와 일치해야 합니다. AWS 계정 ID 또는 선택적으로 하나의 '-'(하이픈)을 지정할 수 있습니다. 이 경우에 Amazon Glacier는 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 지정하는 경우 ID에 하이픈('-')을 포함하지 않습니다.

요청 파라미터

이름	설명	필수
operation=add	볼트에서 태그 삭제(POST tags remove)와 구분할 목적으로 add 값이 포함된 단일 쿼리 문자열 파라미터 operation 입니다.	예

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

요청 본문에는 다음의 JSON 필드가 포함됩니다.

태그

볼트에 추가할 태그입니다. 각 태그는 키와 값으로 구성됩니다. 값은 빈 문자열일 수도 있습니다.

유형: 문자열 간 맵

길이 제한: 최소 길이 1. 최대 길이 10

필수 항목 여부: 예

응답

작업 요청이 성공하면 서비스가 HTTP 204 No Content 응답을 반환합니다.

구문

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

이 작업은 응답 본문을 반환하지 않습니다.

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예시

요청 예시

다음 예에서는 볼트에 추가할 태그와 함께 HTTP POST 요청을 전송합니다.

```
POST /-/vaults/examplevault/tags?operation=add HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
Content-Length: length
x-amz-glacier-version: 2012-06-01

{
  "Tags":
    {
       "examplekey1": "examplevalue1",
       "examplekey2": "examplevalue2"
    }
}
```

응답의 예

요청이 성공하면 Amazon Glacier가 다음 예제와 같이 `HTTP 204 No Content`를 반환합니다.

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

관련 섹션

- 볼트의 태그 목록 조회(GET tags)

- 볼트에서 태그 삭제(POST tags remove)

참고

언어 특정 Amazon SDK 중 하나에서 이 API를 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface

볼트 만들기(PUT vault)

설명

이 작업은 지정한 이름으로 새로운 볼트를 생성합니다. 볼트 이름은 AWS 계정에 속하는 AWS 리전 내에서 반드시 고유해야 합니다. 계정당 최대 1,000개까지 볼트를 생성할 수 있습니다. 더 많은 볼트 생성을 위한 자세한 정보는 Amazon Glacier 제품 세부 정보 페이지를 참조하세요.

볼트 이름을 지정할 때는 다음 지침을 따라야 합니다.

- 이름에 포함되는 문자 길이는 1~255자입니다.
- 유효한 문자는 A~Z, a~z, 0~9, '-'(하이픈), '_'(밑줄), '.'(마침표)입니다.

이 작업은 멱등성을 갖기 때문에 동일한 요청을 여러 차례 전송할 수 있으며, Amazon Glacier(Amazon Glacier)가 지정한 볼트를 처음 생성한 이후부터는 추가적인 영향을 미치지 않습니다.

요청

구문

볼트 생성을 위해 HTTP PUT 요청을 생성할 볼트의 URI로 전송합니다.

```
PUT /AccountId/vaults/VaultName HTTP/1.1
```

```
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
Content-Length: Length
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 값은 AWS 계정 ID입니다. 이 값은 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID와 일치해야 합니다. AWS 계정 ID 또는 선택적으로 하나의 '-'(하이픈)을 지정할 수 있습니다. 이 경우에 Amazon Glacier는 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 지정하는 경우 ID에 하이픈('-')을 포함하지 않습니다.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

이 작업의 요청 본문은 비어 있어야 합니다(0바이트).

응답

구문

```
HTTP/1.1 201 Created
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Location: Location
```

응답 헤더

성공적인 응답에는 모든 작업에 일반적인 응답 헤더 외에 다음 응답 헤더가 포함됩니다. 일반적인 응답 헤더에 대한 자세한 내용은 공통 응답 헤더 섹션을 참조하세요.

이름	설명
Location	생성된 볼트의 상대적 URI 경로입니다. 유형: 문자열

응답 본문

이 작업은 응답 본문을 반환하지 않습니다.

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예시

요청 예시

다음은 HTTP PUT 요청을 전송하여 examplevault라는 이름의 볼트를 생성하는 예제입니다.

```
PUT /-/vaults/examplevault HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Content-Length: 0
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

Amazon Glacier가 볼트를 생성하고, Location 헤더에 볼트의 상대적 URI 경로를 반환합니다. 계정 ID는 요청에서 계정 ID 또는 하이픈('-') 지정 여부에 상관없이 항상 Location 헤더에 표시됩니다.

```
HTTP/1.1 201 Created
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
Location: /111122223333/vaults/examplevault
```

관련 섹션

- 볼트 목록 조회(GET vaults)
- 볼트 삭제(DELETE vault)
- Amazon Glacier의 ID 및 액세스 관리(IAM)

참고

언어 특정 Amazon SDK 중 하나에서 이 API를 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface

볼트 잠금 완료(POST lockId)

설명

이번 작업에서는 볼트 잠금 상태를 `InProgress`에서 `Locked`로 전환하여 볼트 잠금 프로세스를 완료합니다. 이렇게 완료된 볼트 잠금 정책은 변경할 수 없습니다. `InProgress`을 호출하면 볼트 잠금이 볼트 잠금 시작(POST lock-policy) 상태로 전환됩니다. 볼트 잠금 상태는 볼트 잠금 가져오기(GET lock-policy)을 호출하면 가져올 수 있습니다. 볼트 잠금 프로세스에 대한 자세한 내용은 Amazon Glacier 볼트 잠금 섹션을 참조하세요.

이 작업은 멱등성을 갖습니다. 볼트 잠금 상태가 `Locked`이고, 입력한 잠금 ID가 처음에 볼트를 잠글 때 사용한 잠금 ID와 일치하는 경우에는 이 요청이 항상 성공합니다.

볼트 잠금이 `Locked` 상태라고 해도 요청 시 잘못된 잠금 ID를 입력하면 `AccessDeniedException` 오류가 반환됩니다. 볼트 잠금이 `InProgress` 상태일 때 잘못된 잠금 ID를 입력하면 `InvalidParameter` 오류가 발생합니다.

요청

볼트 잠금 프로세스를 완료하려면 HTTP `POST` 요청을 유효한 잠금 ID와 함께 볼트의 `lock-policy` 하위 리소스 URI로 전송합니다.

구문

```
POST /AccountId/vaults/vaultName/lock-policy/lockId HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
```

```
Authorization: SignatureValue
Content-Length: Length
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 값은 AWS 계정 ID입니다. 이 값은 요청 서명에 사용된 보안 인증 정보와 연결된 AWS 계정 ID와 반드시 일치해야 합니다. AWS 계정 ID 또는 선택적으로 하나의 '-'(하이픈)을 지정할 수 있습니다. 이 경우에 Amazon Glacier는 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 지정하는 경우 ID에 하이픈('-')을 포함하지 않습니다.

lockId 값은 볼트 잠금 시작(POST lock-policy) 요청에서 가져오는 잠금 ID입니다.

요청 파라미터

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

이 작업에는 요청 본문이 없습니다.

응답

작업 요청이 성공하면 서비스가 HTTP 204 No Content 응답을 반환합니다.

구문

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

이 작업은 응답 본문을 반환하지 않습니다.

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예시

요청 예시

다음은 HTTP POST 요청을 잠금 ID와 함께 전송하여 볼트 잠금 프로세스를 완료하는 예제입니다.

```
POST /-/vaults/examplevault/lock-policy/AE863rKkWZU53SLW5be4DUcW HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
Content-Length: length
x-amz-glacier-version: 2012-06-01
```

응답의 예

요청이 성공하면 Amazon Glacier(Amazon Glacier)가 다음 예시와 같이 `HTTP 204 No Content` 응답을 반환합니다.

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

관련 섹션

- 볼트 잠금 중단(DELETE lock-policy)
- 볼트 잠금 가져오기(GET lock-policy)

- 볼트 잠금 시작(POST lock-policy)

참고

언어 특정 Amazon SDK 중 하나에서 이 API를 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface

볼트 삭제(DELETE vault)

설명

이 작업은 볼트를 삭제합니다. Amazon Glacier(Amazon Glacier)는 마지막 인벤토리를 기준으로 볼트에 저장된 아카이브가 없고, 마지막 인벤토리 이후 볼트에 대한 쓰기 작업이 없었던 경우에 한해 볼트를 삭제합니다. 두 조건 중 하나라도 충족되지 않는 경우에는 볼트 삭제가 중단되고(볼트가 삭제되지 않음) Amazon Glacier는 오류를 반환합니다.

볼트에 저장된 아카이브 수 등 볼트 관련 정보를 제공하는 볼트 설명(GET vault) 작업을 사용할 수 있지만, 해당 정보는 Amazon Glacier가 마지막으로 생성한 볼트 인벤토리를 기준으로 합니다.

이 작업은 멱등성을 갖습니다.

> **Note**
>
> 볼트를 삭제하면 해당 볼트에 연결된 볼트 액세스 정책 역시 삭제됩니다. 볼트 액세스 정책에 대한 자세한 내용은 볼트 액세스 정책 섹션을 참조하세요.

요청

볼트를 삭제하려면 DELETE 요청을 볼트의 리소스 URI로 전송합니다.

구문

```
DELETE /AccountId/vaults/VaultName HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
```

```
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> `AccountId` 값은 볼트를 소유한 계정의 AWS 계정 ID입니다. AWS 계정 ID 또는 선택적으로 하나의 '-'(하이픈)을 지정할 수 있습니다. 이 경우에 Amazon Glacier는 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 사용하는 경우 ID에 하이픈('-')을 포함할 수 없습니다.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

이 작업에는 요청 본문이 없습니다.

응답

구문

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

이 작업은 응답 본문을 반환하지 않습니다.

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예시

요청 예시

다음은 이름이 examplevault인 볼트를 삭제하는 예제입니다. 이 예제 요청은 삭제할 리소스(볼트)의 URI에 대한 DELETE 요청입니다.

```
DELETE /-/vaults/examplevault HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

관련 섹션

- 볼트 만들기(PUT vault)
- 볼트 목록 조회(GET vaults)
- 작업 시작(POST jobs)
- Amazon Glacier의 ID 및 액세스 관리(IAM)

참고

언어 특정 Amazon SDK 중 하나에서 이 API를 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface

볼트 액세스 정책 삭제(DELETE access-policy)

설명

이 작업에서는 지정된 볼트에 연결되어 있는 액세스 정책을 삭제합니다. 이 작업은 최종적으로 일관성을 갖습니다. 다시 말해서 Amazon Glacier(Amazon Glacier)가 액세스 정책을 완전히 제거하는 데 약간의 시간이 필요하기 때문에 삭제 요청을 전송한 후에도 짧은 시간 동안 정책이 적용되는 것을 볼 수도 있습니다.

이 작업은 멱등성을 갖습니다. 따라서 볼트에 연결되어 있는 정책이 없더라도 삭제를 여러 차례 호출할 수 있습니다. 볼트 액세스 정책에 대한 자세한 내용은 볼트 액세스 정책 섹션을 참조하세요.

Requests

현재 볼트 액세스 정책을 삭제하려면 HTTP `DELETE` 요청을 볼트의 `access-policy` 하위 리소스 URI로 전송합니다.

구문

```
DELETE /AccountId/vaults/vaultName/access-policy HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> `AccountId` 값은 볼트를 소유한 계정의 AWS 계정 ID입니다. AWS 계정 ID를 지정하거나 선택적으로 단일 '-'(하이픈)를 지정할 수 있습니다.이 경우 Amazon Glacier는 요청에 서명하는 데 사용되는 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 사용하는 경우 ID에 하이픈('-')을 포함할 수 없습니다.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

이 작업에는 요청 본문이 없습니다.

응답

정책이 성공적으로 삭제되면 Amazon Glacier가 응답에서 `204 No Content`를 반환합니다.

구문

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

이 작업은 응답 본문을 반환하지 않습니다.

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예제

다음은 볼트 액세스 정책을 삭제하는 방법을 설명한 예제입니다.

요청 예시

이 예시에서 DELETE 요청은 볼트 이름이 **examplevault**인 `access-policy` 하위 리소스로 전송됩니다.

```
DELETE /-/vaults/examplevault/access-policy HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

```
x-amz-glacier-version: 2012-06-01
```

응답의 예

정책이 성공적으로 삭제되면 Amazon Glacier가 응답에서 다음 예시와 같이 `204 No Content`를 반환합니다.

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

관련 섹션

- 볼트 액세스 정책 가져오기(GET access-policy)
- 볼트 액세스 정책 설정(PUT access-policy)

참고

언어 특정 Amazon SDK 중 하나에서 이 API를 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface

볼트 알림 삭제(DELETE notification-configuration)

설명

이 작업에서는 볼트에 설정된 알림 구성(볼트 알림 구성 설정(PUT notification-configuration))을 삭제합니다. 작업은 최종적으로 일관성을 갖습니다. 즉, Amazon Glacier(Amazon Glacier)가 알림을 완전히 비활성화하는 데 약간의 시간이 필요하며, 삭제 요청을 보낸 후 잠시 동안은 알림을 받을 수도 있습니다.

Requests

볼트의 알림 구성을 삭제하려면 `DELETE` 요청을 볼트의 `notification-configuration` 하위 리소스로 전송합니다.

구문

```
DELETE /AccountId/vaults/VaultName/notification-configuration HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 값은 볼트를 소유한 계정의 AWS 계정 ID입니다. AWS 계정 ID를 지정하거나 선택적으로 단일 '-'(하이픈)를 지정할 수 있습니다.이 경우 Amazon Glacier는 요청에 서명하는 데 사용되는 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 사용하는 경우 ID에 하이픈('-')을 포함할 수 없습니다.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

이 작업에는 요청 본문이 없습니다.

응답

구문

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

이 작업은 응답 본문을 반환하지 않습니다.

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예제

다음은 볼트에 설정된 알림 구성을 제거하는 방법을 설명한 예제입니다.

요청 예시

이 예시에서 DELETE 요청은 볼트 이름이 examplevault인 notification-configuration 하위 리소스로 전송됩니다.

```
DELETE /111122223333/vaults/examplevault/notification-configuration HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

관련 섹션

- 볼트 알림 가져오기(GET notification-configuration)
- 볼트 알림 구성 설정(PUT notification-configuration)
- Amazon Glacier의 ID 및 액세스 관리(IAM)

참고

언어 특정 Amazon SDK 중 하나에서 이 API를 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface

볼트 설명(GET vault)

설명

이 작업에서는 볼트의 Amazon 리소스 이름(ARN), 볼트 생성 날짜, 볼트에 저장된 아카이브 수, 볼트에 저장된 모든 아카이브의 총 크기 등 볼트에 대한 정보를 반환합니다. 아카이브의 수와 총 크기는 Amazon Glacier(Amazon Glacier)가 마지막으로 생성한 볼트 인벤토리를 기준으로 합니다(Amazon Glacier에서의 볼트 작업 참조). Amazon Glacier는 거의 매일 볼트 인벤토리를 생성합니다. 이는 아카이브를 볼트에 추가하거나 삭제한 직후에 볼트 설명 요청을 전송하더라도 응답에 변경 내용이 반영되지 않을 수도 있다는 것을 의미합니다.

요청

볼트 정보를 가져오려면 GET 요청을 특정 볼트 리소스의 URI로 전송합니다.

구문

```
GET /AccountId/vaults/VaultName HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 값은 볼트를 소유한 계정의 AWS 계정 ID입니다. AWS 계정 ID 또는 선택적으로 하나의 '-'(하이픈)을 지정할 수 있습니다. 이 경우에 Amazon Glacier는 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 사용하는 경우 ID에 하이픈('-')을 포함할 수 없습니다.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

이 작업에는 요청 본문이 없습니다.

응답

구문

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: Length

{
  "CreationDate" : String,
  "LastInventoryDate" : String,
  "NumberOfArchives" : Number,
  "SizeInBytes" : Number,
  "VaultARN" : String,
  "VaultName" : String
}
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

JSON 응답 본문에는 다음과 같은 JSON 필드가 포함됩니다.

CreationDate

볼트가 생성된 UTC 날짜입니다.

유형: ISO 8601 날짜 형식의 문자열 표현입니다. 예: `2013-03-20T17:03:43.221Z`

LastInventoryDate

Amazon Glacier가 마지막 볼트 인벤토리를 완료한 UTC 날짜입니다. 볼트 인벤토리의 시작에 대한 자세한 내용은 작업 시작(POST jobs) 섹션을 참조하세요.

유형: ISO 8601 날짜 형식의 문자열 표현입니다(예: `2013-03-20T17:03:43.221Z`).

NumberOfArchives

마지막 볼트 인벤토리를 기준으로 볼트에 저장된 아카이브 수입니다. 인벤토리가 아직 볼트에서 실행되지 않은 경우, 예를 들어 볼트를 방금 생성한 경우 이 필드는 null 값을 반환합니다.

형식: 숫자

SizeInBytes

마지막 인벤토리 날짜를 기준으로 아카이브별 오버헤드를 포함하여 볼트에 저장된 아카이브의 총 크기(바이트)입니다. 인벤토리가 아직 볼트에서 실행되지 않은 경우, 예를 들어 볼트를 방금 생성한 경우 이 필드는 `null` 값을 반환합니다.

형식: 숫자

VaultARN

볼트의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

VaultName

생성 시 지정한 볼트 이름입니다. 볼트 이름은 볼트의 ARN에도 포함됩니다.

유형: 문자열

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예시

요청 예시

다음은 이름이 `examplevault`인 볼트에 대한 정보를 가져오는 방법을 설명한 예제입니다.

```
GET /-/vaults/examplevault HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
Content-Type: application/json
Content-Length: 260

{
  "CreationDate" : "2012-02-20T17:01:45.198Z",
  "LastInventoryDate" : "2012-03-20T17:03:43.221Z",
  "NumberOfArchives" : 192,
  "SizeInBytes" : 78088912,
  "VaultARN" : "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault",
  "VaultName" : "examplevault"
}
```

관련 섹션

- 볼트 만들기(PUT vault)
- 볼트 목록 조회(GET vaults)
- 볼트 삭제(DELETE vault)
- 작업 시작(POST jobs)
- Amazon Glacier의 ID 및 액세스 관리(IAM)

참고

언어 특정 Amazon SDK 중 하나에서 이 API를 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface

볼트 액세스 정책 가져오기(GET access-policy)

설명

이 작업은 볼트에 설정되어 있는 access-policy 하위 리소스를 검색합니다. 하위 리소스 설정에 대한 자세한 내용은 볼트 액세스 정책 설정(PUT access-policy) 섹션을 참조하세요. 볼트에 설정되어 있는 액세스 정책이 없는 경우에는 404 Not found 오류가 반환됩니다. 볼트 액세스 정책에 대한 자세한 내용은 볼트 액세스 정책 섹션을 참조하세요.

Requests

현재 볼트 액세스 정책을 반환하려면 HTTP GET 요청을 볼트의 access-policy 하위 리소스 URI로 전송합니다.

구문

```
GET /AccountId/vaults/vaultName/access-policy HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 값은 볼트를 소유한 계정의 AWS 계정 ID입니다. AWS 계정 ID를 지정하거나 선택적으로 단일 '-'(하이픈)를 지정할 수 있습니다.이 경우 Amazon Glacier는 요청에 서명하는 데 사용되는 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 사용하는 경우 ID에 하이픈('-')을 포함할 수 없습니다.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

이 작업에는 요청 본문이 없습니다.

응답

응답 시, Amazon Glacier(Amazon Glacier)는 응답 본문에서 JSON 형식으로 볼트 액세스 정책을 반환합니다.

구문

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: length

{
  "Policy": "string"
}
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

JSON 응답 본문에는 다음과 같은 JSON 필드가 포함됩니다.

정책

JSON 문자열 형식의 볼트 액세스 정책으로서 "\"를 이스케이프 문자로 사용합니다.

유형: 문자열

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예제

다음은 볼트 액세스 정책을 가져오는 방법을 설명한 예제입니다.

요청 예시

이 예제에서는 GET 요청이 볼트의 access-policy 하위 리소스 URI로 전송됩니다.

```
GET /-/vaults/examplevault/access-policy HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

요청이 성공하면 Amazon Glacier가 응답 본문에서 볼트 액세스 정책을 JSON 문자열 형식으로 반환합니다. 반환된 JSON 문자열은 볼트 액세스 정책 설정(PUT access-policy) 예제와 같이 "\"를 이스케이프 문자로 사용합니다. 하지만 다음 예제에서는 가독성을 위해 반환된 JSON 문자열에서 이스케이프 문자를 제외합니다.

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/json
Content-Length: length

{
  "Policy": "
    {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Sid": "allow-time-based-deletes",
          "Principal": {
            "AWS": "999999999999"
          },
          "Effect": "Allow",
          "Action": "glacier:Delete*",
          "Resource": [
            "arn:aws:glacier:us-west-2:999999999999:vaults/examplevault"
          ],
          "Condition": {
            "DateGreaterThan": {
```

```
              "aws:CurrentTime": "2018-12-31T00:00:00Z"
            }
          }
        }
      ]
    }
  "
}
```

관련 섹션

- 볼트 액세스 정책 삭제(DELETE access-policy)
- 볼트 액세스 정책 설정(PUT access-policy)

참고

언어 특정 Amazon SDK 중 하나에서 이 API를 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface

볼트 잠금 가져오기(GET lock-policy)

설명

이 작업에서는 지정된 볼트에 설정되어 있는 `lock-policy` 하위 리소스에서 다음과 같은 속성을 가져옵니다.

- 볼트에 설정되어 있는 볼트 잠금 정책
- 볼트 잠금 상태(`InProgess` 또는 `Locked`)
- 잠금 ID의 만료 시점. 잠금 ID는 볼트 잠금 프로세스를 마치는 데 사용됩니다.
- 볼트 잠금이 시작되어 `InProgress` 상태로 전환된 시점

볼트 잠금 시작(POST lock-policy)을 호출하면 볼트 잠금이 `InProgress` 상태로 전환됩니다. 볼트 잠금 완료(POST lockId)을 호출하면 볼트 잠금이 `Locked` 상태로 전환됩니다. 또한 볼트 잠금 중단

(DELETE lock-policy)을 직접 호출하여 볼트 잠금 프로세스를 중단시킬 수 있습니다. 볼트 잠금 프로세스에 대한 자세한 내용은 Amazon Glacier 볼트 잠금 섹션을 참조하세요.

볼트에 설정되어 있는 볼트 잠금 정책이 없는 경우에는 `404 Not found` 오류가 반환됩니다. 볼트 잠금 정책에 대한 자세한 내용은 볼트 잠금 정책 섹션을 참조하세요.

Requests

현재 볼트 잠금 정책과 기타 속성을 반환하려면 다음 구문 예제와 같이 HTTP `GET` 요청을 볼트의 `lock-policy` 하위 리소스 URI로 전송합니다.

구문

```
GET /AccountId/vaults/vaultName/lock-policy HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> `AccountId` 값은 볼트를 소유한 계정의 AWS 계정 ID입니다. AWS 계정 ID를 지정하거나 선택적으로 단일 '-'(하이픈)를 지정할 수 있습니다.이 경우 Amazon Glacier는 요청에 서명하는 데 사용되는 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 사용하는 경우 ID에 하이픈('-')을 포함할 수 없습니다.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

이 작업에는 요청 본문이 없습니다.

응답

응답 시, Amazon Glacier(Amazon Glacier)는 응답 본문에서 JSON 형식으로 볼트 액세스 정책을 반환합니다.

구문

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: length

{
  "Policy": "string",
  "State": "string",
  "ExpirationDate": "string",
  "CreationDate":"string"
}
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

JSON 응답 본문에는 다음과 같은 JSON 필드가 포함됩니다.

정책

JSON 문자열 형식의 볼트 잠금 정책으로서 "\"를 이스케이프 문자로 사용합니다.

유형: 문자열

상태

볼트 잠금 상태입니다.

타입: 문자열

유효한 값: `InProgress|Locked`

ExpirationDate

잠금 ID가 만료되는 UTC 날짜와 시간입니다. 볼트 잠금이 null 상태인 경우 이 값은 Locked가 될 수 있습니다.

유형: ISO 8601 날짜 형식의 문자열 표현입니다. 예: 2013-03-20T17:03:43.221Z

CreationDate

볼트 잠금이 InProgress 상태로 전환된 UTC 날짜와 시간입니다.

유형: ISO 8601 날짜 형식의 문자열 표현입니다. 예: 2013-03-20T17:03:43.221Z

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예제

다음은 볼트 잠금 정책을 가져오는 방법을 설명한 예제입니다.

요청 예시

이 예제에서는 GET 요청이 볼트의 lock-policy 하위 리소스 URI로 전송됩니다.

```
GET /-/vaults/examplevault/lock-policy HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

요청이 성공하면 Amazon Glacier가 응답 본문에서 볼트 액세스 정책을 JSON 문자열 형식으로 반환합니다. 반환된 JSON 문자열은 볼트 잠금 시작(POST lock-policy) 요청 예제와 같이 "\"를 이스케이프 문자로 사용합니다. 하지만 다음 예제에서는 가독성을 위해 반환된 JSON 문자열에서 이스케이프 문자를 제외합니다.

```
HTTP/1.1 200 OK
```

```
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/json
Content-Length: length

{
  "Policy": "
    {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Sid": "Define-vault-lock",
          "Principal": {
            "AWS": "arn:aws:iam::999999999999:root"
          },
          "Effect": "Deny",
          "Action": "glacier:DeleteArchive",
          "Resource": [
            "arn:aws:glacier:us-west-2:999999999999:vaults/examplevault"
          ],
          "Condition": {
            "NumericLessThanEquals": {
              "glacier:ArchiveAgeInDays": "365"
            }
          }
        }
      ]
    }
  ",
  "State": "InProgress",
  "ExpirationDate": "exampledate",
  "CreationDate": "exampledate"
}
```

관련 섹션

- 볼트 잠금 중단(DELETE lock-policy)
- 볼트 잠금 완료(POST lockId)

- 볼트 잠금 시작(POST lock-policy)

참고

언어 특정 Amazon SDK 중 하나에서 이 API를 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface

볼트 알림 가져오기(GET notification-configuration)

설명

이 작업에서는 볼트에 설정되어 있는 `notification-configuration` 하위 리소스를 가져옵니다(볼트 알림 구성 설정(PUT notification-configuration) 참조). 볼트에 알림 구성이 설정되어 있지 않으면 `404 Not Found` 오류가 반환됩니다. 볼트 알림에 대한 자세한 내용은 Amazon Glacier의 볼트 알림 구성 섹션을 참조하세요.

Requests

알림 구성 정보를 가져오려면 `GET` 요청을 볼트의 `notification-configuration` 하위 리소스 URI로 전송합니다.

구문

```
GET /AccountId/vaults/VaultName/notification-configuration HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> `AccountId` 값은 볼트를 소유한 계정의 AWS 계정 ID입니다. AWS 계정 ID를 지정하거나 선택적으로 단일 '-'(하이픈)를 지정할 수 있습니다.이 경우 Amazon Glacier는 요청에 서명하는 데 사용되는 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 사용하는 경우 ID에 하이픈('-')을 포함할 수 없습니다.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

이 작업에는 요청 본문이 없습니다.

응답

구문

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: length
{
  "Events": [
    String,
    ...
  ],
  "SNSTopic": String
}
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

JSON 응답 본문에는 다음과 같은 JSON 필드가 포함됩니다.

이벤트

Amazon Glacier(Amazon Glacier)가 지정된 Amazon SNS 토픽으로 알림 메시지를 보내게 되는 하나 이상의 이벤트로 이루어진 목록입니다. 알림 메시지를 게시하도록 볼트를 구성할 수 있는 볼트

이벤트에 대한 자세한 내용은 볼트 알림 구성 설정(PUT notification-configuration) 섹션을 참조하세요.

유형: 배열

SNSTopic

Amazon Simple Notification Service(SNS) 토픽의 Amazon 리소스 이름(ARN)입니다. 자세한 내용은 Amazon Simple Notification Service 시작 가이드의 Amazon SNS 시작하기를 참조하세요.

유형: 문자열

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예제

다음은 볼트에 설정된 알림 구성을 가져오는 방법을 설명한 예제입니다.

요청 예시

이 예에서는 GET 요청이 볼트의 notification-configuration 하위 리소스로 전송됩니다.

```
GET /-/vaults/examplevault/notification-configuration HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

응답이 성공적으로 반환되면 감사 로깅 구성 문서가 응답 본문에 JSON 형식으로 표시됩니다. 이번 예시에서는 구성에 따라 두 가지 이벤트(ArchiveRetrievalCompleted와 InventoryRetrievalCompleted)에 대한 알림 메시지가 Amazon SNS 토픽 arn:aws:sns:us-west-2:012345678901:mytopic으로 전송되는 것을 보여줍니다.

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/json
```

```
Content-Length: 150

{
  "Events": [
    "ArchiveRetrievalCompleted",
    "InventoryRetrievalCompleted"
  ],
  "SNSTopic": "arn:aws:sns:us-west-2:012345678901:mytopic"
}
```

관련 섹션

- 볼트 알림 삭제(DELETE notification-configuration)
- 볼트 알림 구성 설정(PUT notification-configuration)
- Amazon Glacier의 ID 및 액세스 관리(IAM)

참고

언어 특정 Amazon SDK 중 하나에서 이 API를 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface

볼트 잠금 시작(POST lock-policy)

설명

이 작업에서는 다음과 같은 방법으로 볼트 잠금 프로세스를 시작합니다.

- 지정된 볼트에 볼트 잠금 정책을 설치합니다.
- 볼트 잠금 상태를 `InProgress`로 설정합니다.
- 볼트 잠금 프로세스를 마치는 데 사용되는 잠금 ID를 반환합니다.

각 볼트마다 볼트 잠금 정책 1개를 설정할 수 있으며, 정책 크기는 최대 20KB로 제한됩니다. 볼트 잠금 정책에 대한 자세한 내용은 볼트 잠금 정책 섹션을 참조하세요.

볼트 잠금 상태가 `InProgress`로 전환된 후에는 24시간 이내에 볼트 잠금 프로세스를 마쳐야 합니다. 24시간이 지나면 잠금 ID가 만료되고 볼트가 자동으로 `InProgress` 상태를 종료하고 볼트 잠금 정책

이 볼트에서 제거됩니다. 볼트 잠금 완료(POST lockId)을 호출하여 볼트 잠금 프로세스를 마치면 볼트 잠금 상태가 Locked로 설정됩니다.

> **Note**
>
> 볼트 잠금이 Locked 상태로 바뀐 이후에는 동일한 볼트에 대해 새로운 볼트 잠금을 시작할 수 없습니다.

볼트 잠금 중단(DELETE lock-policy)을 직접 호출하여 볼트 잠금 프로세스를 중단할 수 있습니다. 볼트 잠금 상태는 볼트 잠금 가져오기(GET lock-policy)을 호출하면 알 수 있습니다. 볼트 잠금 프로세스에 대한 자세한 내용은 Amazon Glacier 볼트 잠금 섹션을 참조하세요.

볼트 잠금이 InProgress 상태일 때 이 작업을 호출하면 AccessDeniedException 오류가 반환됩니다. 볼트 잠금이 InProgress 상태일 때 새로운 볼트 잠금 정책을 시작하려면 볼트 잠금 중단(DELETE lock-policy)을 먼저 호출해야 합니다.

요청

볼트 잠금 프로세스를 시작하려면 다음 구문 예제와 같이 HTTP POST 요청을 볼트의 lock-policy 하위 리소스 URI로 전송합니다.

구문

```
POST /AccountId/vaults/vaultName/lock-policy HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
Content-Length: Length
x-amz-glacier-version: 2012-06-01

{
  "Policy": "string"
}
```

> **Note**
>
> AccountId 값은 AWS 계정 ID입니다. 이 값은 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID와 일치해야 합니다. AWS 계정 ID 또는 선택적으로 하나의 '-'(하이픈)을 지정할 수 있

습니다. 이 경우에 Amazon Glacier는 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 지정하는 경우 ID에 하이픈('-')을 포함하지 않습니다.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

요청 본문의 JSON에 포함되는 필드는 다음과 같습니다.

정책

JSON 문자열 형식의 볼트 잠금 정책으로서 "\"를 이스케이프 문자로 사용합니다.

유형: 문자열

필수 항목 여부: 예

응답

Amazon Glacier(Amazon Glacier)는 정책이 수락되면 `HTTP 201 Created` 응답을 반환합니다.

구문

```
HTTP/1.1 201 Created
x-amzn-RequestId: x-amzn-RequestId
Date: Date
x-amz-lock-id: lockId
```

응답 헤더

성공적인 응답에는 모든 작업에 일반적인 응답 헤더 외에 다음 응답 헤더가 포함됩니다. 일반적인 응답 헤더에 대한 자세한 내용은 공통 응답 헤더 섹션을 참조하세요.

이름	설명
x-amz-lock-id	볼트 잠금 프로세스를 마치는 데 사용되는 잠금 ID입니다. 유형: 문자열

응답 본문

이 작업은 응답 본문을 반환하지 않습니다.

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예시

요청 예시

다음은 HTTP PUT 요청을 볼트의 lock-policy 하위 리소스 URI로 전송하는 예제입니다. Policy JSON 문자열은 이스케이프 문자로 "\"를 사용합니다.

```
PUT /-/vaults/examplevault/lock-policy HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
Content-Length: length
x-amz-glacier-version: 2012-06-01

{"Policy":"{\"Version\":\"2012-10-17\",       \"Statement\":[{\"Sid\":\"Define-vault-
lock\",\"Effect\":\"Deny\",\"Principal\":{\"AWS\":\"arn:aws:iam::999999999999:root
\"},\"Action\":\"glacier:DeleteArchive\",\"Resource\":\"arn:aws:glacier:us-
west-2:999999999999:vaults/examplevault\",\"Condition\":{\"NumericLessThanEquals\":
{\"glacier:ArchiveAgeinDays\":\"365\"}}}]}"}
```

응답의 예

요청이 성공하면 Amazon Glacier가 다음 예시와 같이 HTTP 201 Created 응답을 반환합니다.

```
HTTP/1.1 201 Created
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
x-amz-lock-id: AE863rKkWZU53SLW5be4DUcW
```

관련 섹션

- 볼트 잠금 중단(DELETE lock-policy)
- 볼트 잠금 완료(POST lockId)
- 볼트 잠금 가져오기(GET lock-policy)

참고

언어 특정 Amazon SDK 중 하나에서 이 API를 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface

볼트의 태그 목록 조회(GET tags)

이 작업에서는 볼트에 연결되어 있는 모든 태그의 목록을 조회합니다. 태그가 없는 경우에는 비어있는 맵을 반환합니다. 태그에 대한 자세한 내용은 Amazon Glacier 리소스에 태그 지정 섹션을 참조하세요.

요청 구문

볼트의 태그 목록을 조회하려면 다음 구문 예제와 같이 HTTP GET 요청을 태그 URI로 전송합니다.

```
GET /AccountId/vaults/vaultName/tags HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> `AccountId` 값은 AWS 계정 ID입니다. 이 값은 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID와 일치해야 합니다. AWS 계정 ID 또는 선택적으로 하나의 '-'(하이픈)을 지정할 수 있습니다. 이 경우에 Amazon Glacier는 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 지정하는 경우 ID에 하이픈('-')을 포함하지 않습니다.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

이 작업에는 요청 본문이 없습니다.

응답

작업이 성공하면 서비스가 HTTP `200 OK` 응답을 다시 전송합니다.

응답 구문

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: Length
{
   "Tags":
      {
         "string" : "string",
         "string" : "string"
      }
}
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

JSON 응답 본문에는 다음과 같은 JSON 필드가 포함됩니다.

태그

볼트에 연결된 태그입니다. 각 태그는 키와 값으로 구성됩니다.

유형: 문자열 간 맵

필수 항목 여부: 예

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예시

예제: 볼트의 태그 목록 조회

다음은 볼트의 태그 목록을 조회하는 예제입니다.

요청 예시

이 예제에서는 GET 요청을 전송하여 지정된 볼트의 태그 목록을 가져옵니다.

```
GET /-/vaults/examplevault/tags HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

요청이 성공하면 Amazon Glacier(Amazon Glacier)가 다음 예시와 같이 볼트의 태그 목록과 함께 `HTTP 200 OK`를 반환합니다.

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
Content-Type: application/json
Content-Length: length

{
   "Tags",
      {
         "examplekey1": "examplevalue1",
         "examplekey2": "examplevalue2"
      }
}
```

관련 섹션

- 볼트에 태그 추가(POST tags add)
- 볼트에서 태그 삭제(POST tags remove)

참고

언어 특정 Amazon SDK 중 하나에서 이 API를 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface

볼트 목록 조회(GET vaults)

설명

이 작업에서는 호출 사용자의 계정에 속한 모든 볼트 목록을 조회합니다. 이때 응답으로 반환되는 목록은 볼트 이름을 기준으로 ASCII 정렬 순서를 따릅니다.

기본적으로 이 작업에서 요청 1건당 반환되는 항목 수는 최대 10개입니다. 목록을 조회할 볼트가 더 있는 경우에는 응답 본문의 `marker` 필드에 새로운 볼트 목록 조회 요청과 함께 목록이 계속되는 지점에 볼트의 Amazon 리소스 이름(ARN)이 추가됩니다. 그렇지 않으면 `marker` 필드는 `null` 값을 갖습니

다. 다음 볼트 목록 조회 요청에서는 Amazon Glacier(Amazon Glacier)가 이전 볼트 목록 조회 요청에 대한 응답으로 반환한 값으로 `marker` 파라미터를 설정합니다. 또한 요청 시 `limit` 파라미터를 지정하여 응답으로 반환되는 볼트 수를 제한할 수도 있습니다.

요청

볼트 목록을 가져오려면 `GET` 요청을 볼트 리소스에 전송합니다.

구문

```
GET /AccountId/vaults HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> `AccountId` 값은 AWS 계정 ID입니다. 이 값은 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID와 일치해야 합니다. AWS 계정 ID 또는 선택적으로 하나의 '-'(하이픈)을 지정할 수 있습니다. 이 경우에 Amazon Glacier는 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 지정하는 경우 ID에 하이픈('-')을 포함하지 않습니다.

요청 파라미터

이 작업은 다음 요청 파라미터를 사용합니다.

이름	설명	필수
`limit`	반환할 볼트의 최대 수입니다. 기본 제한은 10개입니다. 반환되는 볼트 수가 지정한 제한보다 적을 수 있지만 제한을 초과할 수는 없습니다. 유형: 문자열 제약 조건: 최소 정수 값 1. 최대 정수 값 10	아니요

이름	설명	필수
marker	페이지 매김에 사용되는 문자열입니다. marker는 볼트 목록 조회가 시작되는 볼트 ARN을 지정합니다. (marker에서 지정하는 볼트는 반환 목록에 포함되지 않습니다.) 이전 볼트 목록 조회 응답에서 marker 값을 가져옵니다. marker는 이전 볼트 목록 조회 요청에서 시작된 결과에 페이지를 계속해서 매겨야 하는 경우에만 포함해야 합니다. 마커에 대해 빈 값("")을 지정하면 첫 번째 볼트부터 볼트 목록을 반환합니다. 유형: 문자열 제약 조건: 없음	아니요

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

이 작업에는 요청 본문이 없습니다.

응답

구문

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: Length

{
  "Marker": String
  "VaultList": [
   {
    "CreationDate": String,
    "LastInventoryDate": String,
    "NumberOfArchives": Number,
```

```
    "SizeInBytes": Number,
    "VaultARN": String,
    "VaultName": String
   },
   ...
  ]
}
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

JSON 응답 본문에는 다음과 같은 JSON 필드가 포함됩니다.

CreationDate

볼트가 생성된 날짜(UTC)입니다.

유형: 문자열. ISO 8601 날짜 형식의 문자열 표현입니다. 예: `2013-03-20T17:03:43.221Z`

LastInventoryDate

마지막 볼트 인벤토리의 날짜(UTC)입니다. 인벤토리가 아직 볼트에서 실행되지 않은 경우, 예를 들어 볼트를 방금 생성한 경우 이 필드는 null 값을 가질 수 있습니다. 볼트 인벤토리의 시작에 대한 자세한 내용은 작업 시작(POST jobs) 섹션을 참조하세요.

유형: ISO 8601 날짜 형식의 문자열 표현입니다. 예: `2013-03-20T17:03:43.221Z`

마커

결과에 페이지를 계속해서 매기는 지점을 나타내는 `vaultARN`입니다. 다른 볼트 목록 조회 요청에서 `marker`를 사용하여 목록에서 추가 볼트를 가져올 수 있습니다. 볼트가 더 없을 경우 이 값은 `null`입니다.

유형: 문자열

NumberOfArchives

마지막 인벤토리 날짜를 기준으로 볼트의 아카이브 수입니다.

유형: 숫자

SizeInBytes

마지막 인벤토리 날짜를 기준으로 아카이브별 오버헤드를 포함하여 볼트에 저장된 모든 아카이브의 총 크기(바이트)입니다.

유형: 숫자

VaultARN

볼트의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

VaultList

각각 볼트에 대해 설명하는 객체의 배열입니다.

유형: 배열

VaultName

볼트 이름입니다.

유형: 문자열

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예시

예제: 모든 볼트 목록 조회

다음은 볼트 목록을 조회하는 예제입니다. 요청에서 `marker` 파라미터와 `limit` 파라미터를 지정하지 않았기 때문에 최대 10개까지 볼트가 반환됩니다.

요청 예시

```
GET /-/vaults HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
```

```
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

Marker가 null 값을 갖기 때문에 더 이상 목록을 조회할 볼트가 없다는 것을 의미합니다.

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
Content-Type: application/json
Content-Length: 497

{
  "Marker": null,
  "VaultList": [
   {
    "CreationDate": "2012-03-16T22:22:47.214Z",
    "LastInventoryDate": "2012-03-21T22:06:51.218Z",
    "NumberOfArchives": 2,
    "SizeInBytes": 12334,
    "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault1",
    "VaultName": "examplevault1"
   },
   {
    "CreationDate": "2012-03-19T22:06:51.218Z",
    "LastInventoryDate": "2012-03-21T22:06:51.218Z",
    "NumberOfArchives": 0,
    "SizeInBytes": 0,
    "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault2",
    "VaultName": "examplevault2"
   },
   {
    "CreationDate": "2012-03-19T22:06:51.218Z",
    "LastInventoryDate": "2012-03-25T12:14:31.121Z",
    "NumberOfArchives": 0,
    "SizeInBytes": 0,
    "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault3",
    "VaultName": "examplevault3"
   }
  ]
}
```

예제: 부분적 볼트 목록 조회

다음은 marker에서 지정한 볼트부터 볼트 2개를 반환하는 예제입니다.

요청 예시

```
GET /-/vaults?limit=2&marker=arn:aws:glacier:us-west-2:012345678901:vaults/
examplevault1 HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

목록에서 볼트 2개가 반환됩니다. Marker에 볼트 ARN이 포함되어 있으므로 다른 볼트 목록 조회 요청에서 페이지 매김이 계속 됩니다.

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
Content-Type: application/json
Content-Length: 497

{
  "Marker": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault3",
  "VaultList": [
   {
    "CreationDate": "2012-03-16T22:22:47.214Z",
    "LastInventoryDate": "2012-03-21T22:06:51.218Z",
    "NumberOfArchives": 2,
    "SizeInBytes": 12334,
    "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault1",
    "VaultName": "examplevault1"
   },
   {
    "CreationDate": "2012-03-19T22:06:51.218Z",
    "LastInventoryDate": "2012-03-21T22:06:51.218Z",
    "NumberOfArchives": 0,
    "SizeInBytes": 0,
    "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault2",
```

```
    "VaultName": "examplevault2"
   }
  ]
}
```

관련 섹션

- 볼트 만들기(PUT vault)
- 볼트 삭제(DELETE vault)
- 작업 시작(POST jobs)
- Amazon Glacier의 ID 및 액세스 관리(IAM)

참고

언어 특정 Amazon SDK 중 하나에서 이 API를 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface

볼트에서 태그 삭제(POST tags remove)

이 작업에서는 볼트에 연결되어 있는 태그 집합에서 태그를 1개 이상 삭제합니다. 태그에 대한 자세한 내용은 Amazon Glacier 리소스에 태그 지정 섹션을 참조하세요.

이 작업은 멱등성을 갖습니다. 볼트에 연결되어 있는 태그가 없더라도 작업은 성공적으로 실행됩니다.

요청 구문

볼트에서 태그를 삭제하려면 다음 구문 예제와 같이 HTTP POST 요청을 태그 URI로 전송합니다.

```
POST /AccountId/vaults/vaultName/tags?operation=remove HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
Content-Length: Length
x-amz-glacier-version: 2012-06-01
{
```

```
   "TagKeys": [
      "string",
      "string"
   ]
}
```

> **Note**
>
> AccountId 값은 AWS 계정 ID입니다. 이 값은 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID와 일치해야 합니다. AWS 계정 ID 또는 선택적으로 하나의 '-'(하이픈)을 지정할 수 있습니다. 이 경우에 Amazon Glacier는 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 지정하는 경우 ID에 하이픈('-')을 포함하지 않습니다.

요청 파라미터

이름	설명	필수
operation =remove	볼트에 태그 추가(POST tags add)와 구분할 목적으로 remove 값이 추가된 단일 쿼리 문자열 파라미터 operation 입니다.	예

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

요청 본문에는 다음과 같은 JSON 필드가 포함됩니다.

TagKeys

태그 키 목록입니다. 해당하는 태그가 각각 볼트에서 삭제됩니다.

유형: 문자열 배열

길이 제한: 목록의 최소 항목 수 1개. 목록의 최대 항목 수 10개

필수 항목 여부: 예

응답

작업이 성공하면 서비스가 비어있는 HTTP 본문과 함께 HTTP `204 No Content` 응답을 다시 전송합니다.

구문

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

이 작업은 응답 본문을 반환하지 않습니다.

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예시

요청 예시

다음은 HTTP POST 요청을 전송하여 지정한 태그를 삭제하는 예제입니다.

```
POST /-/vaults/examplevault/tags?operation=remove HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
Content-Length: length
x-amz-glacier-version: 2012-06-01
```

```
{
   "TagsKeys": [
      "examplekey1",
      "examplekey2"
   ]
}
```

응답의 예

요청이 성공하면 Amazon Glacier(Amazon Glacier)가 다음 예시와 같이 `HTTP 204 No Content`를 반환합니다.

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

관련 섹션

- 볼트에 태그 추가(POST tags add)
- 볼트의 태그 목록 조회(GET tags)

참고

언어 특정 Amazon SDK 중 하나에서 이 API를 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface

볼트 액세스 정책 설정(PUT access-policy)

설명

이 작업에서는 볼트 액세스 정책을 구성하여 기존 정책을 덮어씁니다. 볼트 액세스 정책을 구성하려면 PUT 요청을 볼트의 `access-policy` 하위 리소스에 전송합니다. 볼트마다 액세스 정책 1개를 설정할 수 있으며, 정책 크기는 최대 20KB로 제한됩니다. 볼트 액세스 정책에 대한 자세한 내용은 볼트 액세스 정책 섹션을 참조하세요.

Requests

구문

볼트 액세스 정책을 설정하려면 다음 구문 예제와 같이 HTTP PUT 요청을 볼트의 access-policy 하위 리소스 URI로 전송합니다.

```
PUT /AccountId/vaults/vaultName/access-policy HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
Content-Length: Length
x-amz-glacier-version: 2012-06-01

{
  "Policy": "string"
}
```

> **Note**
>
> AccountId 값은 볼트를 소유한 계정의 AWS 계정 ID입니다. AWS 계정 ID를 지정하거나 선택적으로 단일 '-'(하이픈)를 지정할 수 있습니다.이 경우 Amazon Glacier는 요청에 서명하는 데 사용되는 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 사용하는 경우 ID에 하이픈('-')을 포함할 수 없습니다.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

요청 본문의 JSON에 포함되는 필드는 다음과 같습니다.

정책

JSON 문자열 형식의 볼트 액세스 정책으로서 "\"를 이스케이프 문자로 사용합니다.

유형: 문자열

필수 항목 여부: 예

응답

정책이 허용되면 Amazon Glacier가 응답으로 `204 No Content`를 반환합니다.

구문

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

이 작업은 응답 본문을 반환하지 않습니다.

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예제

요청 예시

다음은 HTTP PUT 요청을 볼트의 `access-policy` 하위 리소스 URI로 전송하는 예제입니다. `Policy` JSON 문자열은 이스케이프 문자로 "\"를 사용합니다.

```
PUT /-/vaults/examplevault/access-policy HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

```
Content-Length: length
x-amz-glacier-version: 2012-06-01

{"Policy":"{\"Version\":\"2012-10-17\",       \"Statement\":[{\"Sid\":
\"Define-owner-access-rights\",\"Effect\":\"Allow\",\"Principal\":{\"AWS\":
\"arn:aws:iam::999999999999:root\"},\"Action\":\"glacier:DeleteArchive\",\"Resource\":
\"arn:aws:glacier:us-west-2:999999999999:vaults/examplevault\"}]}"}
```

응답의 예

요청이 성공하면 Amazon Glacier(Amazon Glacier)가 다음 예시와 같이 `HTTP 204 No Content`를 반환합니다.

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

관련 섹션

- 볼트 액세스 정책 삭제(DELETE access-policy)
- 볼트 액세스 정책 가져오기(GET access-policy)

참고

언어 특정 Amazon SDK 중 하나에서 이 API를 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface

볼트 알림 구성 설정(PUT notification-configuration)

설명

Amazon Glacier(Amazon Glacier)에서 아카이브 및 볼트 인벤토리 가져오기는 비동기식 작업이기 때문에 먼저 작업을 시작하여 완료될 때까지 기다린 후에 작업 출력을 다운로드해야 합니다. 작업이 완료되었을 때 알림 메시지를 Amazon Simple Notification Service(Amazon SNS) 토픽에 게시하도록 볼트

를 구성할 수 있습니다. 이 작업을 사용하여 볼트에 대한 알림 구성을 설정할 수 있습니다. 자세한 내용은 Amazon Glacier의 볼트 알림 구성 단원을 참조하십시오.

볼트 알림을 구성하려면 PUT 요청을 볼트의 `notification-configuration` 하위 리소스에 전송합니다. 알림 구성은 볼트에 따라 다르기 때문에 볼트 하위 리소스라고 불리기도 합니다. 요청에는 Amazon Simple Notification Service(Amazon SNS) 토픽을 제공하는 JSON 문서와 Amazon Glacier가 알림 메시지를 토픽에 전송할 이벤트가 포함되어야 합니다.

다음과 같은 볼트 이벤트에 대해 알림 메시지를 게시하도록 볼트를 구성할 수 있습니다.

- **`ArchiveRetrievalCompleted`**: 이 이벤트는 아카이브 가져오기로 시작된 작업이 완료되었을 때 발생합니다(작업 시작(POST jobs)). 완료된 작업은 상태는 `Succeeded` 또는 `Failed`가 될 수 있습니다. SNS 토픽으로 전송되는 알림 메시지는 작업 설명(GET JobID)에서 반환되는 출력과 동일합니다.
- **`InventoryRetrievalCompleted`**: 이 이벤트는 인벤토리 가져오기로 시작된 작업이 완료되었을 때 발생합니다(작업 시작(POST jobs)). 완료된 작업은 상태는 `Succeeded` 또는 `Failed`가 될 수 있습니다. SNS 토픽으로 전송되는 알림 메시지는 작업 설명(GET JobID)에서 반환되는 출력과 동일합니다.

Amazon SNS 토픽은 알림 메시지를 해당 토픽 게시할 수 있도록 반드시 볼트에 권한을 부여해야 합니다.

Requests

볼트에 대한 알림 구성을 설정하려면 PUT 요청을 볼트의 `notification-configuration` 하위 리소스 URI로 전송합니다. 구성은 요청 본문에서 지정하며, Amazon SNS 토픽 이름을 비롯해 각 토픽으로 알림 메시지를 트리거하는 이벤트들의 배열이 포함됩니다.

구문

```
PUT /AccountId/vaults/VaultName/notification-configuration HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01

{
   "SNSTopic": String,
```

```
   "Events":[String, ...]
}
```

> **Note**
>
> AccountId 값은 볼트를 소유한 계정의 AWS 계정 ID입니다. AWS 계정 ID를 지정하거나 선택적으로 단일 '-'(하이픈)를 지정할 수 있습니다.이 경우 Amazon Glacier는 요청에 서명하는 데 사용되는 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 사용하는 경우 ID에 하이픈('-')을 포함할 수 없습니다.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

요청 본문의 JSON에 포함되는 필드는 다음과 같습니다.

이벤트

Amazon Glacier가 알림 메시지를 전송하는 하나 이상의 이벤트의 배열입니다.

유효한 값: ArchiveRetrievalCompleted | InventoryRetrievalCompleted

필수 항목 여부: 예

유형: 배열

SNSTopic

Amazon SNS 토픽 ARN입니다. 자세한 내용은 Amazon Simple Notification Service 시작 가이드의 Amazon SNS 시작하기를 참조하세요.

필수 항목 여부: 예

유형: 문자열

응답

알림 구성이 허용되면 Amazon Glacier(Amazon Glacier)가 응답으로 `204 No Content`를 반환합니다.

구문

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

응답 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

응답 본문

이 작업은 응답 본문을 반환하지 않습니다.

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예제

다음은 볼트 알림 구성 방법을 설명한 예제입니다.

요청 예시

다음 요청은 `examplevault` 알림 구성을 설정하여 두 개의 이벤트(`ArchiveRetrievalCompleted`와 `InventoryRetrievalCompleted`) 알림이 Amazon SNS 토픽 `arn:aws:sns:us-west-2:012345678901:mytopic`으로 전송되게 합니다.

```
PUT /-/vaults/examplevault/notification-policy HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
```

```
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2

{
   "Events": ["ArchiveRetrievalCompleted", "InventoryRetrievalCompleted"],
   "SNSTopic": "arn:aws:sns:us-west-2:012345678901:mytopic"
}
```

응답의 예

요청이 성공하면 응답으로 `204 No Content`가 반환됩니다.

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

관련 섹션

- 볼트 알림 가져오기(GET notification-configuration)
- 볼트 알림 삭제(DELETE notification-configuration)
- Amazon Glacier의 ID 및 액세스 관리(IAM)

참고

언어 특정 Amazon SDK 중 하나에서 이 API를 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface

아카이브 작업

다음은 Amazon Glacier에서 사용할 수 있는 아카이브 작업입니다.

주제

- 아카이브 삭제(DELETE archive)

- 아카이브 업로드(POST archive)

아카이브 삭제(DELETE archive)

설명

이 작업에서는 아카이브를 볼트에서 삭제합니다. 아카이브는 볼트에서 한 번에 하나씩 삭제할 수 있습니다. 아카이브를 삭제하려면 삭제 요청 시 아카이브 ID를 입력해야 합니다. 아카이브 ID는 해당 아카이브가 포함된 볼트에서 볼트 인벤토리를 다운로드하여 가져올 수 있습니다. 볼트 인벤토리 다운로드에 대한 자세한 내용은 Amazon Glacier에서 볼트 인벤토리 다운로드 섹션을 참조하세요.

아카이브를 삭제한 후에도 삭제한 아카이브를 가져오는 작업을 시작할 수 있도록 요청하는 것은 가능하지만 아카이브 가져오기 작업이 실행되지는 않습니다.

아카이브를 삭제할 때 아카이브 ID에 대해 진행 중인 아카이브 가져오기는 다음 시나리오에 따라 성공할 수도, 혹은 성공하지 않을 수도 있습니다.

- Amazon Glacier(Amazon Glacier)가 아카이브 삭제 요청을 수신할 때 아카이브 검색 작업이 활성화되어 데이터 다운로드를 준비하고 있다면 아카이브 검색 작업이 중단될 수도 있습니다.
- Amazon Glacier가 아카이브 삭제 요청을 수신할 때 아카이브 검색 작업이 아카이브 다운로드를 성공적으로 준비한 경우에는 출력을 다운로드할 수 있습니다.

아카이브 가져오기에 대한 자세한 내용은 Amazon Glacier에서 아카이브 다운로드 섹션을 참조하세요.

이 작업은 멱등성을 갖습니다. 이미 삭제된 아카이브를 삭제하려고 해도 오류가 발생하지는 않습니다.

요청

아카이브를 삭제하려면 DELETE 요청을 아카이브 리소스 URI로 전송합니다.

구문

```
DELETE /AccountId/vaults/VaultName/archives/ArchiveID HTTP/1.1
Host: glacier.Region.amazonaws.com
x-amz-Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> `AccountId` 값은 볼트를 소유한 계정의 AWS 계정 ID입니다. AWS 계정 ID 또는 선택적으로 하나의 '-'(하이픈)을 지정할 수 있습니다. 이 경우에 Amazon Glacier는 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 사용하는 경우 ID에 하이픈('-')을 포함할 수 없습니다.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

이 작업에는 요청 본문이 없습니다.

응답

구문

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

이 작업은 응답 본문을 반환하지 않습니다.

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예시

다음은 이름이 examplevault인 볼트에서 아카이브를 삭제하는 방법을 나타낸 예제입니다.

요청 예시

삭제할 아카이브의 ID는 archives 하위 리소스로 지정됩니다.

```
DELETE /-/vaults/examplevault/archives/NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-
TjhqG6eGoOY9Z8i1_AUyUsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchi
 HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

요청이 성공하면 Amazon Glacier는 아카이브가 삭제되었음을 나타내기 위해 204 No Content로 응답합니다.

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

관련 섹션

- 멀티파트 업로드 시작(POST multipart-uploads)
- 아카이브 업로드(POST archive)
- Amazon Glacier의 ID 및 액세스 관리(IAM)

아카이브 업로드(POST archive)

설명

이 작업에서는 아카이브를 볼트에 추가합니다. 성공적으로 업로드되면 데이터를 오랜 기간 저장할 수 있습니다. 응답 시, Amazon Glacier(Amazon Glacier)는 응답의 x-amz-archive-id 헤더에 아카이브

ID를 반환합니다. 반환되는 아카이브 ID는 나중에 아카이브에 액세스하는 데 필요하므로 저장해야 합니다.

업로드하는 데이터의 SHA256 트리-해시를 제공해야 합니다. SHA256 트리-해시의 계산에 대한 자세한 내용은 체크섬 계산 섹션을 참조하세요.

> **Note**
>
> SHA256 트리 해시는 API를 사용할 때의 아카이브 업로드(POST archive) 작업에만 필요합니다. AWS CLI를 사용할 때는 필요하지 않습니다.

아카이브를 업로드할 때 선택적으로 최대 1,024개의 인쇄 가능한 ASCII 문자의 아카이브 설명을 지정할 수 있습니다. Amazon Glacier는 아카이브를 가져오거나 볼트 인벤토리를 가져올 때 아카이브 설명을 반환합니다. Amazon Glacier는 어떤 방식으로든 설명을 해석하지 않습니다. 아카이브 설명이 고유할 필요는 없습니다. 또한 설명을 사용하여 아카이브 목록을 가져오거나 정렬할 수도 없습니다.

선택적인 아카이브 설명을 제외하고 Amazon Glacier는 아카이브에 대해 추가적인 메타데이터를 지원하지 않습니다. 아카이브 ID는 불투명한 문자열이기 때문에 아카이브의 의미를 추정할 수 없습니다. 대신에 클라이언트 측에서 아카이브 메타데이터를 유지할 수 있습니다. 자세한 내용은 Amazon Glacier의 아카이브 작업 섹션을 참조하세요.

아카이브는 변경할 수 없습니다. 아카이브를 업로드한 후에는 해당 아카이브나 설명을 편집할 수 없습니다.

요청

아카이브를 업로드하려면 HTTP POST 메서드를 사용하여 아카이브를 저장할 볼트의 `archives` 하위 리소스에 요청을 전송합니다. 이때 요청에는 아카이브 페이로드 크기와 체크섬(SHA256 트리-해시)이 포함되어야 하며, 옵션으로 아카이브 설명을 추가할 수도 있습니다.

구문

```
POST /AccountId/vaults/VaultName/archives
Host: glacier.Region.amazonaws.com
x-amz-glacier-version: 2012-06-01
Date: Date
Authorization: SignatureValue
x-amz-archive-description: Description
x-amz-sha256-tree-hash: SHA256 tree hash
x-amz-content-sha256: SHA256 linear hash
```

```
Content-Length: Length

<Request body.>
```

> **Note**
>
> AccountId 값은 볼트를 소유한 계정의 AWS 계정 ID입니다. AWS 계정 ID 또는 선택적으로 하나의 '-'(하이픈)을 지정할 수 있습니다. 이 경우에 Amazon Glacier는 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 사용하는 경우 ID에 하이픈('-')을 포함할 수 없습니다.

요청 파라미터

이 작업 구현에서는 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더 외에 다음 요청 헤더를 사용합니다. 일반적인 요청 헤더에 대한 자세한 내용은 공통 요청 헤더 섹션을 참조하세요.

이름	설명	필수
Content-Length	객체의 크기(바이트)입니다. 자세한 내용은 http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.13을 참조하세요. 유형: 숫자 기본값: 없음 제약 조건: 없음	예
x-amz-archive-description	옵션으로 업로드하는 아카이브 설명입니다. 평문으로 설명하거나 원하는 식별자를 할당할 수도 있습니다. 아카이브 전체에서 설명이 고유할 필요는 없습니다. 볼트 인벤토리를 가져올 때는(작업 시작(POST jobs) 참조) 응답으로 반환되는 아카이브마다 이 설명이 포함됩니다.	아니요

이름	설명	필수
	유형: 문자열 기본값: 없음 제약 조건: 설명은 1,024자보다 작거나 같아야 합니다. 허용되는 문자는 제어 코드를 제외한 7비트 ASCII로 그 중에서도 특히 ASCII 10진수값 32~126 또는 16진수값 0x20~0x7E입니다.	
`x-amz-content-sha256`	페이로드의 SHA256 체크섬(선형 해시)입니다. 이 값은 `x-amz-sha256-tree-hash` 헤더에서 지정하는 값과 다릅니다. 유형: 문자열 기본값: 없음 제약 조건: 없음	예
`x-amz-sha256-tree-hash`	사용자가 계산하는 페이로드 체크섬, 즉 SHA256 트리-해시입니다. SHA256 트리-해시의 계산에 대한 자세한 내용은 체크섬 계산 섹션을 참조하세요. Amazon Glacier가 다른 페이로드 체크섬을 계산할 경우에는 요청이 거부됩니다. 유형: 문자열 기본값: 없음 제약 조건: 없음	예

요청 본문

요청 본문에는 업로드할 데이터가 포함됩니다.

응답

응답 시, Amazon Glacier는 아카이브를 오랜 기간 저장하면서 아카이브 ID에 대한 URI 경로를 반환합니다.

구문

```
HTTP/1.1 201 Created
x-amzn-RequestId: x-amzn-RequestId
Date: Date
x-amz-sha256-tree-hash: ChecksumComputedByAmazonGlacier
Location: Location
x-amz-archive-id: ArchiveId
```

응답 헤더

성공적인 응답에는 모든 작업에 일반적인 응답 헤더 외에 다음 응답 헤더가 포함됩니다. 일반적인 응답 헤더에 대한 자세한 내용은 공통 응답 헤더 섹션을 참조하세요.

이름	설명
Location	새롭게 추가된 아카이브 리소스의 상대적 URI 경로입니다. 유형: 문자열
x-amz-archive-id	아카이브 ID입니다. 이 값은 Location 헤더에도 포함됩니다. 유형: 문자열
x-amz-sha256-tree-hash	Amazon Glacier에서 계산하는 아카이브 체크섬입니다. 유형: 문자열

응답 본문

이 작업은 응답 본문을 반환하지 않습니다.

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예시

요청 예시

다음은 아카이브를 업로드하는 요청 예제입니다.

```
POST /-/vaults/examplevault/archives HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-sha256-tree-hash:
 beb0fe31a1c7ca8c6c04d574ea906e3f97b31fdca7571defb5b44dca89b5af60
x-amz-content-sha256: 7f2fe580edb35154041fa3d4b41dd6d3adaef0c85d2ff6309f1d4b520eeecda3
Content-Length: 2097152
x-amz-glacier-version: 2012-06-01
Authorization: Authorization=AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-date;x-
amz-glacier-
version,Signature=16b9a9e220a37e32f2e7be196b4ebb87120ca7974038210199ac5982e792cace

<Request body (2097152 bytes).>
```

응답의 예

아래와 같이 성공적인 응답에는 Amazon Glacier가 아카이브에 할당한 ID를 가져올 수 있는 `Location` 헤더가 포함됩니다.

```
HTTP/1.1 201 Created
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
x-amz-sha256-tree-hash:
 beb0fe31a1c7ca8c6c04d574ea906e3f97b31fdca7571defb5b44dca89b5af60
Location: /111122223333/vaults/examplevault/archives/
NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-
TjhqG6eGoOY9Z8i1_AUyUsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchi
x-amz-archive-id: NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-
TjhqG6eGoOY9Z8i1_AUyUsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchi
```

관련 섹션

- Amazon Glacier의 아카이브 작업
- 대용량 아카이브를 여러 파트로 나누어 업로드(멀티파트 업로드)

- 아카이브 삭제(DELETE archive)
- Amazon Glacier의 ID 및 액세스 관리(IAM)

멀티파트 업로드 작업

다음은 Amazon Glacier에서 사용할 수 있는 멀티파트 업로드 작업입니다.

주제

- 멀티파트 업로드 중단(DELETE uploadID)
- 멀티파트 업로드 완료(POST uploadID)
- 멀티파트 업로드 시작(POST multipart-uploads)
- 파트 목록 조회(GET uploadID)
- 멀티파트 업로드 목록 조회(GET multipart-uploads)
- 파트 업로드(PUT uploadID)

멀티파트 업로드 중단(DELETE uploadID)

설명

멀티파트 업로드 작업을 위한 본 명령어는 업로드 ID로 식별된 멀티파트 업로드를 중단합니다.

멀티파트 업로드 중단 요청에 성공한 후에는 업로드 ID를 사용하여 추가적인 파트를 업로드하거나 다른 작업을 수행할 수 없습니다. 완료된 멀티파트 업로드는 중단이 불가능합니다. 그러나 이미 중단된 업로드는 일시적으로 중단이 가능합니다.

이 작업은 멱등성을 갖습니다.

멀티파트 업로드에 대한 자세한 내용은 대용량 아카이브를 여러 파트로 나누어 업로드(멀티파트 업로드) 섹션을 참조하세요.

Requests

멀티파트 업로드를 중단하려면 볼트의 `multipart-uploads` 하위 리소스 URI로 HTTP `DELETE` 요청을 보내고 특정 멀티파트 업로드 ID를 URI의 일부분으로 식별하십시오.

구문

```
DELETE /AccountId/vaults/VaultName/multipart-uploads/uploadID HTTP/1.1
```

```
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 값은 볼트를 소유한 계정의 AWS 계정 ID입니다. AWS 계정 ID를 지정하거나 선택적으로 단일 '-'(하이픈)를 지정할 수 있습니다.이 경우 Amazon Glacier는 요청에 서명하는 데 사용되는 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 사용하는 경우 ID에 하이픈('-')을 포함할 수 없습니다.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

이 작업에는 요청 본문이 없습니다.

응답

구문

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

이 작업은 응답 본문을 반환하지 않습니다.

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예제

요청 예시

다음의 예시에서는 멀티파트 업로드 ID 리소스의 URI로 DELETE 요청이 전송됩니다.

```
DELETE /-/vaults/examplevault/multipart-uploads/
OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-
khxOjyEXAMPLE  HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

관련 섹션

- 멀티파트 업로드 시작(POST multipart-uploads)
- 파트 업로드(PUT uploadID)
- 멀티파트 업로드 완료(POST uploadID)
- 멀티파트 업로드 목록 조회(GET multipart-uploads)
- 파트 목록 조회(GET uploadID)
- 대용량 아카이브를 여러 파트로 나누어 업로드(멀티파트 업로드)
- Amazon Glacier의 ID 및 액세스 관리(IAM)

멀티파트 업로드 완료(POST uploadID)

설명

이 멀티파트 업로드 작업을 직접 호출하여 모든 아카이브 파트의 업로드를 마쳤으며 이제 업로드된 파트에서 Amazon Glacier가 아카이브를 어셈블링할 수 있음을 Amazon Glacier(Amazon Glacier)에 알립니다.

멀티파트 업로드에 대한 자세한 내용은 대용량 아카이브를 여러 파트로 나누어 업로드(멀티파트 업로드) 섹션을 참조하세요.

아카이브를 어셈블링하여 볼트에 저장하고 나면 Amazon Glacier가 새롭게 생성된 아카이브 리소스의 아카이브 ID를 반환합니다. 아카이브를 업로드하고 나서 나중에 아카이브를 가져오려면 반환된 아카이브 ID를 저장해야 합니다.

요청할 때는 업로드한 전체 아카이브에 대해 계산된 SHA256 트리-해시를 포함시켜야 합니다. SHA256 트리-해시의 계산에 대한 자세한 내용은 체크섬 계산 섹션을 참조하세요. 서버 측에서도 Amazon Glacier는 어셈블링된 아카이브의 SHA256 트리-해시를 구성합니다. 값이 일치하면 Amazon Glacier가 아카이브를 볼트에 저장합니다. 그렇지 않으면 오류를 반환하고 작업이 중단됩니다. 파트 목록 조회(GET uploadID) 작업은 특정 멀티파트 업로드에 대해 업로드한 파트 목록을 반환합니다. 여기에는 업로드된 각 파트에서 불량 체크섬 문제를 디버깅하는 데 사용할 수 있는 체크섬 정보가 포함됩니다.

또한 Amazon Glacier는 내용 범위가 누락되지 않았는지 검사합니다. 파트를 업로드할 때는 최종 아카이브 어셈블리에서 각 파트가 위치하는 곳을 찾을 수 있도록 범위 값을 지정합니다. 최종 아카이브를 어셈블링할 때 Amazon Glacier는 누락된 내용 범위의 유무를 검사하며, 누락된 내용 범위가 있는 경우에는 Amazon Glacier가 오류를 반환하고 멀티파트 업로드 완료 작업이 중단됩니다.

멀티파트 업로드 완료 작업은 멱등성 작업입니다. 따라서 첫 번째 멀티파트 업로드를 성공적으로 완료한 후 단기간 내에 해당 작업을 다시 호출하는 경우 작업이 성공적으로 실행되고 동일한 아카이브 ID를 반환합니다. 이는 네트워크 장애로 인해 연결이 중단되거나 500 서버 오류가 발생하는 경우에 유용합니다. 이때는 멀티파트 업로드 완료 요청을 반복하더라도 중복 아카이브를 생성하지 않고 동일한 아카이브 ID가 얻습니다. 단, 멀티파트 업로드 완료 후에는 파트 목록 조회 작업을 호출할 수 없습니다. 따라서 멱등성을 갖는 완료 작업이라고 해도 멀티파트 업로드 목록 조회 응답에 멀티파트 업로드가 표시되지는 않습니다.

요청

멀티파트 업로드를 완료하려면 멀티파트 업로드 시작 요청에 대한 응답으로 HTTP POST 요청을 Amazon Glacier가 생성한 업로드 ID의 URI로 전송합니다. 이는 파트 업로드 시 사용한 URI와 동일합

니다. 이때 필요한 공통 헤더 외에도 전체 아카이브의 SHA256 트리-해시 결과와 아카이브의 총 크기(바이트)를 요청에 포함시켜야 합니다.

구문

```
POST /AccountId/vaults/VaultName/multipart-uploads/uploadID
Host: glacier.Region.amazonaws.com
Date: date
Authorization: SignatureValue
x-amz-sha256-tree-hash: SHA256 tree hash of the archive
x-amz-archive-size: ArchiveSize in bytes
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 값은 볼트를 소유한 계정의 AWS 계정 ID입니다. AWS 계정 ID 또는 선택적으로 하나의 '-'(하이픈)을 지정할 수 있습니다. 이 경우에 Amazon Glacier는 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 사용하는 경우 ID에 하이픈('-')을 포함할 수 없습니다.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더 외에 다음 요청 헤더를 사용합니다. 일반적인 요청 헤더에 대한 자세한 내용은 공통 요청 헤더 섹션을 참조하세요.

이름	설명	필수
x-amz-archive-size	전체 아카이브의 총 크기(바이트)입니다. 이 값은 업로드한 개별 파트 크기의 총합이 되어야 합니다. 유형: 문자열 기본값: 없음	예

이름	설명	필수
	제약 조건: 없음	
x-amz-sha256-tree-hash	전체 아카이브의 SHA256 트리-해시입니다. 즉, 개별 파트의 SHA256 트리-해시로 이루어진 트리-해시입니다. 사용자가 요청에서 지정한 값이 Amazon Glacier에서 계산한 최종 어셈블링된 아카이브의 SHA256 트리–해시와 일치하지 않으면 Amazon Glacier가 오류를 반환하고 요청은 중단됩니다. 유형: 문자열 기본값: 없음 제약 조건: 없음	예

요청 요소

이 작업에서는 요청 요소를 사용하지 않습니다.

응답

Amazon Glacier(Amazon Glacier)는 전체 아카이브의 SHA256 트리-해시를 생성합니다. 해당 값이 요청에서 지정한 전체 아카이브의 SHA256 트리-해시와 일치하는 경우 Amazon Glacier는 아카이브를 볼트에 추가합니다. 또한 응답으로 새롭게 추가된 아카이브 리소스의 URL 경로와 함께 HTTP `Location` 헤더를 반환합니다. 아카이브 크기 또는 요청에서 보낸 SHA256이 일치하지 않으면 Amazon Glacier가 오류를 반환하고 업로드는 미완료 상태를 유지합니다. 하지만 나중에 아카이브를 성공적으로 생성할 수 있는 지점부터 정확한 값으로 멀티파트 업로드 완료 작업을 재시도할 수 있습니다. 멀티파트 업로드가 완료되지 않으면 결국 Amazon Glacier가 업로드 ID를 회수합니다.

구문

```
HTTP/1.1 201 Created
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Location: Location
x-amz-archive-id: ArchiveId
```

응답 헤더

성공적인 응답에는 모든 작업에 일반적인 응답 헤더 외에 다음 응답 헤더가 포함됩니다. 일반적인 응답 헤더에 대한 자세한 내용은 공통 응답 헤더 섹션을 참조하세요.

이름	설명
Location	새롭게 생성된 아카이브의 상대적 URI 경로입니다. 이 URL에는 Amazon Glacier에서 생성된 아카이브 ID가 포함됩니다. 유형: 문자열
x-amz-archive-id	아카이브 ID입니다. 이 값은 Location 헤더에도 포함됩니다. 유형: 문자열

응답 필드

이 작업은 응답 본문을 반환하지 않습니다.

예시

요청 예시

이 예에서는 HTTP POST 요청이 멀티파트 업로드 시작 요청에 의해 반환된 URI로 전송됩니다. 요청에서는 전체 아카이브의 SHA256 트리-해시와 총 아카이브 크기를 모두 지정합니다.

```
POST /-/vaults/examplevault/multipart-uploads/
OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-
khxOjyEXAMPLE HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
z-amz-Date: 20170210T120000Z
x-amz-sha256-tree-hash:1ffc0f54dd5fdd66b62da70d25edacd0
x-amz-archive-size:8388608
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

다음 응답 예에서는 Amazon Glacier가 업로드된 파트에서 아카이브를 성공적으로 생성했음을 보여줍니다. 이 응답에는 완료 경로와 함께 아카이브 ID가 포함됩니다.

```
HTTP/1.1 201 Created
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Location: /111122223333/vaults/examplevault/archives/
NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-
TjhqG6eGoOY9Z8i1_AUyUsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchi
x-amz-archive-id: NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-
TjhqG6eGoOY9Z8i1_AUyUsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchi
```

이제 HTTP 요청을 새롭게 추가된 리소스/아카이브의 URI로 전송할 수 있습니다. 예를 들어 GET 요청을 전송하여 아카이브를 가져올 수 있습니다.

관련 섹션

- 멀티파트 업로드 시작(POST multipart-uploads)
- 파트 업로드(PUT uploadID)
- 멀티파트 업로드 중단(DELETE uploadID)
- 멀티파트 업로드 목록 조회(GET multipart-uploads)
- 파트 목록 조회(GET uploadID)
- 대용량 아카이브를 여러 파트로 나누어 업로드(멀티파트 업로드)
- 아카이브 삭제(DELETE archive)
- Amazon Glacier의 ID 및 액세스 관리(IAM)

멀티파트 업로드 시작(POST multipart-uploads)

설명

이 작업으로 멀티파트 업로드가 시작됩니다(대용량 아카이브를 여러 파트로 나누어 업로드(멀티파트 업로드) 섹션 참조). Amazon Glacier(Amazon Glacier)는 멀티파트 업로드 리소스를 생성하고 응답에서 해당 ID를 반환합니다. 업로드 ID는 이후 멀티파트 업로드 작업에서 사용됩니다.

멀티파트 업로드를 시작할 때는 파트 크기(바이트 수)를 지정합니다. 부품 크기는 2의 제곱이 곱해진 메비바이트(MiB)(1024키비바이트 [KiB]) 값이어야 합니다. 예를 들어 1048576(1MiB), 2097152(2MiB),

4194304(4MiB), 8388608(8MiB) 등과 같습니다. 최소 허용 파트 크기는 1MiB이고, 최대 크기는 4기비바이트(GiB)입니다.

마지막 파트를 제외하고 현재 업로드 ID를 사용해 업로드하는 파트는 모두 크기가 같아야 합니다. 마지막 파트는 크기가 같거나 작을 수 있습니다. 예를 들어 16.2MiB 파일을 업로드한다고 가정하겠습니다. 파트 크기를 4MiB씩 나누어 멀티파트 업로드를 시작한다면 각각 4MiB의 파트 네 개와 0.2MiB의 파트 한 개를 업로드하게 됩니다.

> **Note**
>
> Amazon Glacier에서는 전체 아카이브 크기를 지정할 필요가 없기 때문에 멀티파트 업로드를 시작할 때 아카이브의 크기를 몰라도 상관없습니다.

멀티파트 업로드를 완료한 후 Amazon Glacier는 ID가 참조한 멀티파트 업로드 리소스를 제거합니다. 사용자가 멀티파트 업로드를 취소하면 Amazon Glacier가 멀티파트 업로드 리소스를 제거하며, 24시간 동안 활동이 없어도 리소스가 제거될 수 있습니다. ID는 24시간 이후에도 사용할 수 있지만 애플리케이션은 이러한 사용을 예상하지 못합니다.

Requests

멀티파트 업로드를 시작하려면 HTTP POST 요청을 아카이브를 저장할 볼트의 `multipart-uploads` 하위 리소스 URI로 전송합니다. 이때 요청에는 파트 크기가 포함되어야 하며, 옵션으로 아카이브 설명을 추가할 수도 있습니다.

구문

```
POST /AccountId/vaults/VaultName/multipart-uploads
Host: glacier.us-west-2.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
x-amz-archive-description: ArchiveDescription
x-amz-part-size: PartSize
```

> **Note**
>
> `AccountId` 값은 볼트를 소유한 계정의 AWS 계정 ID입니다. AWS 계정 ID를 지정하거나 선택적으로 단일 '-'(하이픈)를 지정할 수 있습니다.이 경우 Amazon Glacier는 요청에 서명하는

데 사용되는 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 사용하는 경우 ID에 하이픈('-')을 포함할 수 없습니다.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더 외에 다음 요청 헤더를 사용합니다. 일반적인 요청 헤더에 대한 자세한 내용은 공통 요청 헤더 섹션을 참조하세요.

이름	설명	필수
x-amz-part-size	마지막 파트를 제외한 각 파트의 크기(바이트)입니다. 마지막 파트는 이 파트 크기보다 작을 수 있습니다. 유형: 문자열 기본값: None 제약 조건: 부품 크기는 2의 제곱이 곱해진 메비바이트(1024KiB) 값이어야 합니다. 예를 들어 1048576(1MiB), 2097152(2MiB), 4194304(4MiB), 8388608(8MiB) 등과 같습니다. 최소 허용 파트 크기는 1MiB이고, 최대 크기는 4GiB(4096MiB)입니다.	예
x-amz-archive-description	여러 파트로 나누어 업로드하는 아카이브 설명입니다. 평문으로 설명하거나 원하는 고유 식별자를 할당할 수도 있습니다. 볼트 인벤토리를 가져올 때는(작업 시작(POST jobs) 참조) 응답으로 반환되는 아카이브마다 이 설명이 인벤토리에 포함됩니다. 아카이브 설명에서 선행 스페이스는 제거됩니다.	아니요

이름	설명	필수
	유형: 문자열 기본값: None 제약 조건: 설명은 1,024바이트보다 작거나 같아야 합니다. 허용되는 문자는 제어 코드를 제외한 7비트 ASCII 중에서도 특히 ASCII 값 32~126 십진수 또는 0x20~0x7E 16진수입니다.	

요청 본문

이 작업에는 요청 본문이 없습니다.

응답

응답 시, Amazon Glacier는 ID로 식별된 멀티파트 업로드 리소스를 생성한 후 멀티파트 업로드 ID의 상대적 URI 경로를 반환합니다.

구문

```
HTTP/1.1 201 Created
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Location: Location
x-amz-multipart-upload-id: multiPartUploadId
```

응답 헤더

성공적인 응답에는 모든 작업에 일반적인 응답 헤더 외에 다음 응답 헤더가 포함됩니다. 일반적인 응답 헤더에 대한 자세한 내용은 공통 응답 헤더 섹션을 참조하세요.

이름	설명
Location	Amazon Glacier가 생성한 멀티파트 업로드 ID의 상대적 URI 경로입니다. 이 URI 경로를 통해 파트를 업로드하거나 멀티파트 업로드를 완료하는 요청을 전송합니다.

이름	설명
	유형: 문자열
x-amz-multipart-upload-id	멀티파트 업로드 ID입니다. 이 값은 Location 헤더에도 포함됩니다. 유형: 문자열

응답 본문

이 작업은 응답 본문을 반환하지 않습니다.

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예제

요청 예시

다음은 HTTP POST 요청을 이름이 examplevault인 볼트의 multipart-uploads 하위 리소스 URI로 전송하여 멀티파트 업로드를 시작하는 예제입니다. 여기에는 4MiB(4194304바이트)의 파트 크기를 지정하는 헤더와 아카이브 설명(옵션)이 포함되어 있습니다.

```
POST /-/vaults/examplevault/multipart-uploads
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-archive-description: MyArchive-101
x-amz-part-size: 4194304
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

Amazon Glacier는 멀티파트 업로드 리소스를 생성한 후 볼트의 multipart-uploads 하위 리소스에 추가합니다. Location 응답 헤더에는 멀티파트 업로드 ID에 대한 상대적 URI 경로가 포함되어 있습니다.

```
HTTP/1.1 201 Created
```

```
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Location: /111122223333/vaults/examplevault/multipart-uploads/
OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-
khxOjyEXAMPLE
x-amz-multipart-upload-id:
 OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-
khxOjyEXAMPLE
```

개별 파트 업로드에 대한 자세한 내용은 파트 업로드(PUT uploadID) 섹션을 참조하세요.

관련 섹션

- 파트 업로드(PUT uploadID)
- 멀티파트 업로드 완료(POST uploadID)
- 멀티파트 업로드 중단(DELETE uploadID)
- 멀티파트 업로드 목록 조회(GET multipart-uploads)
- 파트 목록 조회(GET uploadID)
- 아카이브 삭제(DELETE archive)
- 대용량 아카이브를 여러 파트로 나누어 업로드(멀티파트 업로드)
- Amazon Glacier의 ID 및 액세스 관리(IAM)

파트 목록 조회(GET uploadID)

설명

이 멀티파트 업로드 작업에서는 업로드 ID로 식별되는 특정 멀티파트 업로드에서 업로드된 아카이브의 파트 목록을 조회합니다. 멀티파트 업로드에 대한 자세한 내용은 대용량 아카이브를 여러 파트로 나누어 업로드(멀티파트 업로드) 섹션을 참조하세요.

이 요청은 멀티파트 업로드를 완료하기 전에 멀티파트 업로드가 진행 중이라면 언제든지 가능합니다. Amazon Glacier는 각 파트 업로드에서 사용자가 지정한 범위를 기준으로 정렬된 파트 목록을 반환합니다. 멀티파트 업로드가 완료된 후에 파트 목록 조회 요청을 보내면 Amazon Glacier(Amazon Glacier)가 오류를 반환합니다.

파트 목록 조회 작업은 페이지 매김을 지원합니다. 응답 본문의 `Marker` 필드는 항상 확인하여 목록을 계속 이어가는 마커가 있는지 살펴야 합니다. 항목이 더 이상 없는 경우에는 `marker` 필드가 `null` 값을 갖습니다. `marker`가 null 값이 아닌 경우 다음 파트 세트를 가져오려면 Amazon Glacier가 이전 파

트 목록 조회 요청에 대한 응답으로 반환한 마커 값으로 설정된 marker 요청 파라미터로 파트 목록 조회 요청을 한 번 더 전송합니다.

또한 요청 시 limit 파라미터를 지정하여 응답으로 반환되는 파트 수를 제한할 수도 있습니다.

Requests

구문

진행 중인 멀티파트 업로드의 파트 목록을 조회하려면 GET 요청을 멀티파트 업로드 ID의 리소스 URI로 전송합니다. 멀티파트 업로드 ID는 멀티파트 업로드를 시작할 때(멀티파트 업로드 시작(POST multipart-uploads)) 반환됩니다. 옵션으로 marker 및 limit 파라미터를 지정할 수 있습니다.

```
GET /AccountId/vaults/VaultName/multipart-uploads/uploadID HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 값은 볼트를 소유한 계정의 AWS 계정 ID입니다. AWS 계정 ID를 지정하거나 선택적으로 단일 '-'(하이픈)를 지정할 수 있습니다.이 경우 Amazon Glacier는 요청에 서명하는 데 사용되는 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 사용하는 경우 ID에 하이픈('-')을 포함할 수 없습니다.

요청 파라미터

이름	설명	필수
limit	반환할 파트의 최대 수입니다. 기본 제한은 50개입니다. 반환되는 파트 수가 지정한 제한보다 적을 수 있지만 제한을 초과할 수는 없습니다. 유형: 문자열 제약 조건: 최소 정수 값 1. 최대 정수 값 50	아니요

이름	설명	필수
marker	페이지 매김에 사용되는 불투명한 문자열입니다. marker는 파트 목록 조회가 시작되는 파트를 지정합니다. 이전 파트 목록 조회 응답에서 marker 값을 가져옵니다. marker는 이전 파트 목록 조회 요청에서 시작된 결과에 페이지를 계속해서 매겨야 하는 경우에만 포함해야 합니다. 유형: 문자열 제약 조건: 없음	아니요

요청 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

요청 본문

이 작업에는 요청 본문이 없습니다.

응답

구문

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: Length

{
    "ArchiveDescription" : String,
    "CreationDate" : String,
    "Marker": String,
    "MultipartUploadId" : String,
    "PartSizeInBytes" : Number,
    "Parts" :
    [ {
      "RangeInBytes" : String,
```

```
      "SHA256TreeHash" : String
      },
      ...
     ],
    "VaultARN" : String
}
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

JSON 응답 본문에는 다음과 같은 JSON 필드가 포함됩니다.

ArchiveDescription

멀티파트 업로드 시작 요청에서 지정한 아카이브 설명입니다. 멀티파트 업로드 시작 작업에서 아카이브 설명을 지정하지 않은 경우 이 필드는 null 값을 갖습니다.

유형: 문자열

CreationDate

멀티파트 업로드가 시작된 UTC 시간입니다.

유형: 문자열. ISO 8601 날짜 형식의 문자열 표현입니다. 예: 2013-03-20T17:03:43.221Z

마커

결과에 페이지를 계속해서 매기는 지점을 나타내는 불투명한 문자열입니다. 새 파트 목록 조회 요청에서 marker를 사용하여 목록의 작업을 추가로 가져옵니다. 파트가 더 없을 경우 이 값은 null입니다.

유형: 문자열

MultipartUploadId

파트가 연결되는 업로드의 ID입니다.

유형: 문자열

PartSizeInBytes

파트 크기(바이트)입니다. 이 크기는 멀티파트 업로드 시작 요청에서 지정한 값과 동일합니다.

형식: 숫자

파트

멀티파트 업로드의 파트 크기 목록입니다. 배열의 각 객체는 RangeBytes 및 sha256-tree-hash 이름/값 페어로 구성됩니다.

유형: 배열

RangeInBytes

파트의 바이트 범위(범위 상한값 포함)입니다.

유형: 문자열

SHA256TreeHash

Amazon Glacier가 파트에 대해 계산한 SHA256 트리-해시 값입니다. 이 필드는 절대 null 값을 가질 수 없습니다.

유형: 문자열

VaultARN

멀티파트 업로드가 시작되는 볼트의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예제

예제: 멀티파트 업로드의 파트 목록 조회

다음은 업로드하는 모든 파트의 목록을 조회하는 예제입니다. 아래 예제는 HTTP GET 요청을 진행 중인 멀티파트 업로드의 특정 멀티파트 업로드 ID URI로 전송하고 최대 1,000개까지 파트를 반환합니다.

요청 예시

```
GET /-/vaults/examplevault/multipart-uploads/
OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-
khxOjyEXAMPLE HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
```

```
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

응답 시, Amazon Glacier가 지정된 멀티파트 업로드 ID와 연결되어 업로드된 파트 목록을 반환합니다. 이 예제에서는 파트가 2개뿐입니다. 반환되는 `Marker` 필드가 `null` 값을 갖기 때문에 멀티파트 업로드에 남아있는 파트가 더 없다는 것을 의미합니다.

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/json
Content-Length: 412

{
    "ArchiveDescription" : "archive description",
    "CreationDate" : "2012-03-20T17:03:43.221Z",
    "Marker": null,
    "MultipartUploadId" :
 "OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-
khxOjyEXAMPLE",
    "PartSizeInBytes" : 4194304,
    "Parts" :
    [ {
      "RangeInBytes" : "0-4194303",
      "SHA256TreeHash" : "01d34dabf7be316472c93b1ef80721f5d4"
      },
      {
      "RangeInBytes" : "4194304-8388607",
      "SHA256TreeHash" : "0195875365afda349fc21c84c099987164"
      }],
    "VaultARN" : "arn:aws:glacier:us-west-2:012345678901:vaults/demo1-vault"
}
```

예제: 멀티파트 업로드의 파트 목록 조회(Marker 및 Limit Request 파라미터 지정)

다음은 페이지 매김을 사용해 결과 수를 제한하는 방법을 나타낸 예제입니다. 아래 예제는 HTTP GET 요청을 진행 중인 멀티파트 업로드의 특정 멀티파트 업로드 ID URI로 전송하고 파트 1개를 반환합니

다. 파트 목록을 시작할 파트에서 첫 번째 marker 파라미터가 지정됩니다. 이전 파트 목록 조회 요청의 응답에서 marker 값을 가져올 수 있습니다. 또한 이 예제에서는 limit 파라미터가 1로 설정되어 파트를 1개 반환합니다. Marker 필드가 null 값이 아니기 때문에 가져올 파트가 적어도 1개 이상이란 것을 의미합니다.

요청 예시

```
GET /-/vaults/examplevault/multipart-uploads/
OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-
khxOjyEXAMPLE?marker=1001&limit=1 HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

응답 시, Amazon Glacier가 지정된 진행 중인 멀티파트 업로드 ID와 연결되어 업로드된 파트 목록을 반환합니다.

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: text/json
Content-Length: 412

{
    "ArchiveDescription" : "archive description 1",
    "CreationDate" : "2012-03-20T17:03:43.221Z",
    "Marker": "MfgsKHVjbQ6EldVl72bn3_n5h2TaGZQUO-Qb3B9j3TITf7WajQ",
    "MultipartUploadId" :
 "OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-
khxOjyEXAMPLE",
    "PartSizeInBytes" : 4194304,
    "Parts" :
    [ {
      "RangeInBytes" : "4194304-8388607",
      "SHA256TreeHash" : "01d34dabf7be316472c93b1ef80721f5d4"
      }],
    "VaultARN" : "arn:aws:glacier:us-west-2:012345678901:vaults/demo1-vault"
```

```
}
```

관련 섹션

- 멀티파트 업로드 시작(POST multipart-uploads)
- 파트 업로드(PUT uploadID)
- 멀티파트 업로드 완료(POST uploadID)
- 멀티파트 업로드 중단(DELETE uploadID)
- 멀티파트 업로드 목록 조회(GET multipart-uploads)
- 대용량 아카이브를 여러 파트로 나누어 업로드(멀티파트 업로드)
- Amazon Glacier의 ID 및 액세스 관리(IAM)

멀티파트 업로드 목록 조회(GET multipart-uploads)

설명

이 멀티파트 업로드 작업에서는 특정 볼트에서 진행 중인 멀티파트 업로드 목록을 표시합니다. 진행 중인 멀티파트 업로드란 멀티파트 업로드 시작(POST multipart-uploads) 요청으로 시작되었지만 아직 완료되거나 중단되지 않은 멀티파트 업로드를 말합니다. 멀티파트 업로드 목록 조회에 대한 응답으로 반환되는 목록은 정해진 순서가 없습니다.

멀티파트 업로드 목록 조회 작업은 페이지 매김을 지원합니다. 기본적으로 이 작업에서 반환되는 멀티파트 업로드 수는 최대 50개입니다. 응답 본문의 `marker` 필드는 항상 확인하여 목록을 계속 이어가는 마커가 있는지 살펴야 합니다. 항목이 더 이상 없는 경우에는 `marker` 필드가 `null` 값을 갖습니다.

`marker`가 null 값이 아닌 경우 다음 멀티파트 업로드 세트를 가져오려면 Amazon Glacier(Amazon Glacier)가 이전 멀티파트 업로드 목록 조회 요청에 대한 응답으로 반환한 마커 값으로 설정된 `marker` 요청 파라미터로 멀티파트 업로드 목록 조회 요청을 한 번 더 전송합니다.

이 작업과 파트 목록 조회(GET uploadID) 작업의 차이점에 대해서 알아둘 필요가 있습니다. 멀티파트 업로드 목록 조회 작업은 볼트 1개의 멀티파트 업로드 목록을 모두 조회합니다. 하지만 파트 목록 조회 작업은 업로드 ID로 식별되는 특정 멀티파트 업로드의 파트를 반환합니다.

멀티파트 업로드에 대한 자세한 내용은 대용량 아카이브를 여러 파트로 나누어 업로드(멀티파트 업로드) 섹션을 참조하세요.

Requests

구문

멀티파트 업로드 목록을 조회하려면 GET 요청을 볼트의 multipart-uploads 하위 리소스 URI로 전송합니다. 옵션으로 marker 및 limit 파라미터를 지정할 수 있습니다.

```
GET /AccountId/vaults/VaultName/multipart-uploads HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 값은 볼트를 소유한 계정의 AWS 계정 ID입니다. AWS 계정 ID를 지정하거나 선택적으로 단일 '-'(하이픈)를 지정할 수 있습니다.이 경우 Amazon Glacier는 요청에 서명하는 데 사용되는 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 사용하는 경우 ID에 하이픈('-')을 포함할 수 없습니다.

요청 파라미터

이름	설명	필수
limit	응답 본문으로 반환되는 최대 업로드 수를 지정합니다. 이 파라미터를 지정하지 않으면 업로드 목록 조회 작업이 최대 50개까지 업로드를 반환합니다. 유형: 문자열 제약 조건: 최소 정수 값 1. 최대 정수 값 50	아니요
marker	페이지 매김에 사용되는 불투명한 문자열입니다. marker는 업로드 목록 조회가 시작되는 업로드를 지정합니다. 이전 업로드 목록 조회 응답에서 marker 값을 가져옵니다. marker는 이전 업로드 목록 조회 요청에서 시작된 결과에 페이지를 계속해서 매겨야 하는 경우에만 포함해야 합니다.	아니요

이름	설명	필수
	유형: 문자열 제약 조건: 없음	

요청 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

요청 본문

이 작업에는 요청 본문이 없습니다.

응답

구문

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: Length

{
  "Marker": String,
  "UploadsList" : [
    {
      "ArchiveDescription": String,
      "CreationDate": String,
      "MultipartUploadId": String,
      "PartSizeInBytes": Number,
      "VaultARN": String
    },
   ...
  ]
}
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

JSON 응답 본문에는 다음과 같은 JSON 필드가 포함됩니다.

ArchiveDescription

멀티파트 업로드 시작 요청에서 지정한 아카이브 설명입니다. 멀티파트 업로드 시작 작업에서 아카이브 설명을 지정하지 않은 경우 이 필드는 null 값을 갖습니다.

유형: 문자열

CreationDate

멀티파트 업로드가 시작된 UTC 시간입니다.

유형: 문자열. ISO 8601 날짜 형식의 문자열 표현입니다. 예: 2013-03-20T17:03:43.221Z

마커

결과에 페이지를 계속해서 매기는 지점을 나타내는 불투명한 문자열입니다. 새 멀티파트 업로드 목록 조회 요청에서 marker를 사용하여 목록의 업로드를 추가로 가져옵니다. 업로드가 더 없을 경우 이 값은 null입니다.

유형: 문자열

PartSizeInBytes

멀티파트 업로드 시작(POST multipart-uploads) 요청에서 지정된 파트 크기입니다. 마지막 파트를 제외하고 업로드되는 파트의 총 크기이기 때문에 이 크기보다 작을 수도 있습니다.

형식: 숫자

MultipartUploadId

멀티파트 업로드 ID입니다.

유형: 문자열

UploadsList

멀티파트 업로드 객체에 대한 메타데이터 목록입니다. 이 목록의 각 항목은 `ArchiveDescription`, `CreationDate`, `MultipartUploadId`, `PartSizeInBytes`, `VaultARN` 등을 포함하여 해당하는 업로드의 이름-값 페어 집합으로 구성됩니다.

유형: 배열

VaultARN

아카이브가 저장되는 볼트의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예제

예제: 모든 멀티파트 업로드 목록 조회

다음은 볼트에서 진행 중인 모든 멀티파트 업로드 목록을 조회하는 예제입니다. 이 예제에서는 HTTP `GET` 요청이 지정된 볼트의 `multipart-uploads` 하위 리소스 URI로 전송됩니다. 요청에서 `marker` 파라미터와 `limit` 파라미터를 지정하지 않았기 때문에 최대 1,000개까지 진행 중인 멀티파트 업로드가 반환됩니다.

요청 예시

```
GET /-/vaults/examplevault/multipart-uploads HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

응답 시, Amazon Glacier는 지정된 볼트에서 진행 중인 모든 멀티파트 업로드의 목록을 반환합니다. `marker` 필드가 `null` 값이므로 더 이상 목록을 조회할 업로드가 없다는 것을 의미합니다.

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/json
Content-Length: 1054

{
  "Marker": null,
  "UploadsList": [
    {
      "ArchiveDescription": "archive 1",
      "CreationDate": "2012-03-19T23:20:59.130Z",
      "MultipartUploadId":
 "xsQdFIRsfJr20CW2AbZBKpRZAFTZSJIMtL2hYf8mvp8dM0m4RUzlaqoEye6g3h3ecqB_zqwB7zLDMeSWhwo65re4C4Ev"
      "PartSizeInBytes": 4194304,
      "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault"
    },
    {
      "ArchiveDescription": "archive 2",
      "CreationDate": "2012-04-01T15:00:00.000Z",
      "MultipartUploadId": "nPyGOnyFcx67qqX7E-0tSGiRi88hHMOwOxR-
_jNyM6RjVMFfV29lFqZ3rNsSaWBugg6OP92pRtufeHdQH7ClIpSF6uJc",
      "PartSizeInBytes": 4194304,
      "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault"
    },
    {
      "ArchiveDescription": "archive 3",
      "CreationDate": "2012-03-20T17:03:43.221Z",
      "MultipartUploadId": "qt-RBst_7yO8gVIonIBsAxr2t-db0pE4s8MNeGjKjGdNpuU-
cdSAcqG62guwV9r5jh5mLyFPzFEitTpNE7iQfHiu1XoV",
      "PartSizeInBytes": 4194304,
      "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault"
    }
  ]
}
```

예제: 부분적 멀티파트 업로드 목록 조회

다음은 페이지 매김을 사용해 결과 수를 제한하는 방법을 나타낸 예제입니다. 이 예제에서는 HTTP GET 요청이 지정된 볼트의 `multipart-uploads` 하위 리소스 URI로 전송됩니다. 또한 `limit` 파라미터가 1로 설정되어 있습니다. 이는 목록에서 반환되는 업로드 수가 1개로 제한된다는 것을 의미하며, `marker` 파라미터는 반환 목록이 시작되는 멀티파트 업로드 ID를 나타냅니다.

요청 예시

```
GET /-/vaults/examplevault/multipart-uploads?
limit=1&marker=xsQdFIRsfJr20CW2AbZBKpRZAFTZSJIMtL2hYf8mvp8dM0m4RUzlaqoEye6g3h3ecqB_zqwB7zLDMeSw
 HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

응답 시, Amazon Glacier(Amazon Glacier)는 지정된 볼트에서 진행 중인 두 개 이하의 멀티파트 업로드 목록을 반환하는데, 지정된 마커부터 시작하여 두 개의 결과를 반환합니다.

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/json
Content-Length: 470

{
  "Marker": "qt-RBst_7yO8gVIonIBsAxr2t-db0pE4s8MNeGjKjGdNpuU-
cdSAcqG62guwV9r5jh5mLyFPzFEitTpNE7iQfHiu1XoV",
  "UploadsList" : [
    {
      "ArchiveDescription": "archive 2",
      "CreationDate": "2012-04-01T15:00:00.000Z",
      "MultipartUploadId": "nPyGOnyFcx67qqX7E-0tSGiRi88hHMOwOxR-
_jNyM6RjVMFfV29lFqZ3rNsSaWBugg6OP92pRtufeHdQH7ClIpSF6uJc",
      "PartSizeInBytes": 4194304,
      "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault"
    }
  ]
}
```

관련 섹션

- 멀티파트 업로드 시작(POST multipart-uploads)

- 파트 업로드(PUT uploadID)
- 멀티파트 업로드 완료(POST uploadID)
- 멀티파트 업로드 중단(DELETE uploadID)
- 파트 목록 조회(GET uploadID)
- 대용량 아카이브를 여러 파트로 나누어 업로드(멀티파트 업로드)
- Amazon Glacier의 ID 및 액세스 관리(IAM)

파트 업로드(PUT uploadID)

설명

이 멀티파트 업로드 작업에서는 아카이브 파트를 업로드합니다. 파트 업로드 요청에서 업로드되는 아카이브 어셈블의 바이트 범위를 지정하기 때문에 아카이브 파트는 순서에 상관없이 업로드할 수 있습니다. 또한 병렬 방식으로 파트를 업로드하는 것도 가능합니다. 멀티파트 업로드에서 업로드할 수 있는 파트 수는 최대 10,000개입니다.

멀티파트 업로드에 대한 자세한 내용은 대용량 아카이브를 여러 파트로 나누어 업로드(멀티파트 업로드) 섹션을 참조하세요.

다음 조건 중 한 가지라도 사실인 경우에는 Amazon Glacier(Amazon Glacier)가 사용자의 파트 업로드 요청을 거부합니다.

- SHA256 트리-해시가 일치하지 않습니다: 파트 데이터가 전송 중 손상되지 않도록 하려면 파트의 SHA256 트리 해시를 계산하여 요청에 추가해야 합니다. Amazon Glacier가 파트 데이터 수신과 함께 SHA256 트리 해시를 계산합니다. 이때 두 해시 값이 일치하지 않으면 작업이 오류로 중단됩니다. SHA256 트리-해시의 계산에 대한 자세한 내용은 체크섬 계산 섹션을 참조하세요.
- SHA256 선형 해시가 일치하지 않습니다: 권한 부여에 필요하기 때문에 업로드된 전체 페이로드의 SHA256 선형 해시를 계산한 후 요청에 추가합니다. SHA256 선형 해시의 계산에 대한 자세한 내용은 체크섬 계산 섹션을 참조하세요.
- 파트 크기가 일치하지 않습니다: 마지막 파트를 제외하고 각 파트의 크기는 해당하는 멀티파트 업로드 시작(POST multipart-uploads) 요청에서 지정한 크기와 반드시 일치해야 합니다. 마지막 파트의 크기는 지정된 크기보다 작거나 같아야 합니다.

> **Note**
>
> 멀티파트 업로드 시작 요청에서 지정한 파트 크기보다 작으면서 동시에 마지막 파트가 아닌 파트를 업로드하는 경우에도 파트 업로드 요청은 성공적으로 실행됩니다. 하지만 이후 멀티파트 업로드 완료 요청이 실행되지 않습니다.

- 범위가 맞지 않습니다: 요청의 바이트 범위 값이 해당하는 시작 요청에서 지정한 파트 크기와 맞지 않습니다. 예를 들어 크기가 4194304바이트(4MB)인 파트를 지정한다면 유효한 파트 범위는 0~4194303바이트(4MB-1)와 4194304(4MB)~8388607바이트(8MB-1)입니다. 하지만 범위 값을 2~6MB로 설정하면 범위와 파트 크기가 맞지 않아 업로드가 오류로 중단되고 맙니다.

이 작업은 멱등성을 갖습니다. 따라서 동일한 파트를 여러 차례 업로드할 경우 가장 최근 요청에서 입력했던 데이터가 이전에 업로드했던 데이터를 덮어씁니다.

요청

이 HTTP PUT 요청을 멀티파트 업로드 시작 요청에서 반환된 업로드 ID의 URI로 전송합니다. Amazon Glacier는 업로드 ID를 사용하여 파트 업로드와 특정 멀티파트 업로드를 연결합니다. 요청할 때는 파트 데이터의 SHA256 트리-해시(`x-amz-SHA256-tree-hash` 헤더), 전체 페이로드의 SHA256 선형 해시(`x-amz-content-sha256` 헤더), 바이트 범위(`Content-Range` 헤더), 그리고 바이트 단위의 파트 길이(`Content-Length` 헤더)가 포함되어야 합니다.

구문

```
PUT /AccountId/vaults/VaultName/multipart-uploads/uploadID HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
Content-Range: ContentRange
Content-Length: PayloadSize
Content-Type: application/octet-stream
x-amz-sha256-tree-hash: Checksum of the part
x-amz-content-sha256: Checksum of the entire payload
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> `AccountId` 값은 볼트를 소유한 계정의 AWS 계정 ID입니다. AWS 계정 ID 또는 선택적으로 하나의 '-'(하이픈)을 지정할 수 있습니다. 이 경우에 Amazon Glacier는 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 사용하는 경우 ID에 하이픈('-')을 포함할 수 없습니다.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더 외에 다음 요청 헤더를 사용합니다. 일반적인 요청 헤더에 대한 자세한 내용은 공통 요청 헤더 섹션을 참조하세요.

이름	설명	필수
`Content-Length`	파트 길이(바이트)를 식별합니다. 유형: 문자열 기본값: 없음 제약 조건: 없음	아니요
`Content-Range`	이 파트에 업로드될 어셈블링된 아카이브의 바이트 범위를 식별합니다. Amazon Glacier는 이 정보를 사용하여 아카이브를 적절한 순서로 어셈블링합니다. 이 헤더의 형식은 RFC 2616을 따릅니다. 헤더 예를 들면 `Content-Range:bytes 0-4194303/*` 과 같습니다. 유형: 문자열 기본값: 없음	예

이름	설명	필수
	제약 조건: 범위는 멀티파트 업로드를 시작할 때 지정한 파트 크기보다 클 수 없습니다.	
x-amz-content-sha256	업로드되는 페이로드의 SHA256 체크섬(선형 해시)입니다. 이 값은 x-amz-sha256-tree-hash 헤더에서 지정하는 값과 다릅니다. 유형: 문자열 기본값: 없음 제약 조건: 없음	예
x-amz-sha256-tree-hash	업로드되는 데이터의 SHA256 트리-해시를 지정합니다. SHA256 트리-해시의 계산에 대한 자세한 내용은 체크섬 계산 섹션을 참조하세요. 유형: 문자열 기본값: 없음 제약 조건: 없음	예

요청 본문

요청 본문에는 업로드할 데이터가 포함됩니다.

응답

파트 업로드가 성공하면 Amazon Glacier가 204 No Content 응답을 반환합니다.

구문

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
x-amz-sha256-tree-hash: ChecksumComputedByAmazonGlacier
```

응답 헤더

성공적인 응답에는 모든 작업에 일반적인 응답 헤더 외에 다음 응답 헤더가 포함됩니다. 일반적인 응답 헤더에 대한 자세한 내용은 공통 응답 헤더 섹션을 참조하세요.

이름	설명
x-amz-sha256-tree-hash	Amazon Glacier가 업로드된 파트에 대해 계산한 SHA256 트리 해시입니다. 유형: 문자열

응답 본문

이 작업은 응답 본문을 반환하지 않습니다.

예시

다음은 4MB 파트를 업로드하는 요청입니다. 이 요청은 아카이브에서 4MB 파트가 첫 번째 파트가 되도록 바이트 범위를 설정합니다.

요청 예시

다음은 HTTP PUT 요청을 전송하여 4MB 파트를 업로드하는 예제입니다. 이 요청은 멀티파트 업로드 시작 요청에서 반환된 업로드 ID의 URI로 전송됩니다. Content-Range 헤더는 아카이브에서 첫 번째 4MB 데이터 파트를 업로드할 파트로 식별합니다.

```
PUT /-/vaults/examplevault/multipart-uploads/
OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-
khxOjyEXAMPLE HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Range:bytes 0-4194303/*
x-amz-sha256-tree-hash:c06f7cd4baacb087002a99a5f48bf953
x-amz-content-sha256:726e392cb4d09924dbad1cc0ba3b00c3643d03d14cb4b823e2f041cff612a628
Content-Length: 4194304
Authorization: Authorization=AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-date;x-
amz-glacier-
version,Signature=16b9a9e220a37e32f2e7be196b4ebb87120ca7974038210199ac5982e792cace
```

다음 파트를 업로드할 때도 절차는 같습니다. 하지만 업로드할 파트의 SHA256 트리-해시를 새롭게 계산해야 합니다. 또한 파트가 최종 어셈블리에서 포함되는 위치를 알 수 있도록 새로운 바이트 범위도 지정해야 합니다. 다음은 동일한 업로드 ID를 사용하여 다른 파트를 업로드하는 요청입니다. 이 요청에서는 이전 요청과 4MB 파트 크기 다음에 이어지는 4MB 아카이브를 지정합니다.

```
PUT /-/vaults/examplevault/multipart-uploads/
OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-
khxOjyEXAMPLE HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Range:bytes 4194304-8388607/*
Content-Length: 4194304
x-amz-sha256-tree-hash:f10e02544d651e2c3ce90a4307427493
x-amz-content-sha256:726e392cb4d09924dbad1cc0ba3b00c3643d03d14cb4b823e2f041cff612a628
x-amz-glacier-version: 2012-06-01
Authorization: Authorization=AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/
us-west-2/glacier/aws4_request, SignedHeaders=host;x-amz-content-sha256;x-amz-date;x-
amz-glacier-version,
 Signature=16b9a9e220a37e32f2e7be196b4ebb87120ca7974038210199ac5982e792cace
```

어떤 순서로든 파트를 업로드할 수 있지만 Amazon Glacier가 각 파트의 범위 사양을 사용하여 어셈블링 순서를 결정합니다.

응답의 예

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
x-amz-sha256-tree-hash: c06f7cd4baacb087002a99a5f48bf953
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

관련 섹션

- 멀티파트 업로드 시작(POST multipart-uploads)
- 파트 업로드(PUT uploadID)
- 멀티파트 업로드 완료(POST uploadID)
- 멀티파트 업로드 중단(DELETE uploadID)
- 멀티파트 업로드 목록 조회(GET multipart-uploads)
- 파트 목록 조회(GET uploadID)

- 대용량 아카이브를 여러 파트로 나누어 업로드(멀티파트 업로드)
- Amazon Glacier의 ID 및 액세스 관리(IAM)

작업

다음은 Amazon Glacier에서 사용할 수 있는 작업입니다.

주제

- 작업 설명(GET JobID)
- 작업 출력 가져오기(GET output)
- 작업 시작(POST jobs)
- 작업 목록 조회(GET jobs)

작업 설명(GET JobID)

설명

이 작업은 작업 시작 날짜, 작업을 시작한 사용자, 작업 상태 코드/메시지, 그리고 Amazon Glacier(Amazon Glacier)가 작업을 완료한 후 알림을 보낼 Amazon Simple Notification Service(Amazon SNS) 토픽 등 이전에 시작한 작업에 대한 정보를 반환합니다. 작업 시작에 대한 자세한 내용은 작업 시작(POST jobs) 섹션을 참조하세요.

> **Note**
>
> 이 작업을 통해 작업 상태를 확인할 수 있습니다. 하지만 Amazon Glacier가 작업 완료 후 토픽에 알림을 보낼 수 있도록 작업 시작 요청에서 Amazon SNS 토픽을 설정 및 지정하는 것을 적극 권장합니다.

작업 ID는 Amazon Glacier가 작업을 완료한 후 최소 24시간 동안 만료되지 않습니다.

Requests

구문

작업에 대한 정보를 가져오려면 HTTP GET 메서드를 사용하여 요청을 원하는 작업까지 전송합니다. 이때 상대 URI 경로는 작업 시작 시 Amazon Glacier가 반환하는 경로와 동일합니다.

```
GET /AccountID/vaults/VaultName/jobs/JobID HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: date
Authorization: signatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 값은 볼트를 소유한 계정의 AWS 계정 ID입니다. AWS 계정 ID를 지정하거나 선택적으로 단일 '-'(하이픈)를 지정할 수 있습니다.이 경우 Amazon Glacier는 요청에 서명하는 데 사용되는 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 사용하는 경우 ID에 하이픈('-')을 포함할 수 없습니다.

> **Note**
>
> 요청에서 JobID를 생략하는 경우에는 응답이 특정 볼트에서 현재 활성화되어 있는 모든 작업의 목록을 반환합니다. 작업 목록 조회에 대한 자세한 내용은 작업 목록 조회(GET jobs) 섹션을 참조하세요.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

이 작업에는 요청 본문이 없습니다.

응답

구문

```
HTTP/1.1 201 Created
x-amzn-RequestId: x-amzn-RequestId
```

```
Date: Date
Content-Type: application/json
Content-Length: Length

{
    "Action": "string",
    "ArchiveId": "string",
    "ArchiveSHA256TreeHash": "string",
    "ArchiveSizeInBytes": number,
    "Completed": boolean,
    "CompletionDate": "string",
    "CreationDate": "string",
    "InventoryRetrievalParameters": {
        "EndDate": "string",
        "Format": "string",
        "Limit": "string",
        "Marker": "string",
        "StartDate": "string"
    },
    "InventorySizeInBytes": number,
    "JobDescription": "string",
    "JobId": "string",
    "JobOutputPath": "string",
    "OutputLocation": {
        "S3": {
            "AccessControlList": [
                {
                    "Grantee": {
                        "DisplayName": "string",
                        "EmailAddress": "string",
                        "ID": "string",
                        "Type": "string",
                        "URI": "string"
                    },
                    "Permission": "string"
                }
            ],
            "BucketName": "string",
            "CannedACL": "string",
            "Encryption": {
                "EncryptionType": "string",
                "KMSContext": "string",
                "KMSKeyId": "string"
            },
```

```
            "Prefix": "string",
            "StorageClass": "string",
            "Tagging": {
                "string": "string"
            },
            "UserMetadata": {
                "string": "string"
            }
        }
    },
    "RetrievalByteRange": "string",
    "SelectParameters": {
        "Expression": "string",
        "ExpressionType": "string",
        "InputSerialization": {
            "csv": {
                "Comments": "string",
                "FieldDelimiter": "string",
                "FileHeaderInfo": "string",
                "QuoteCharacter": "string",
                "QuoteEscapeCharacter": "string",
                "RecordDelimiter": "string"
            }
        },
        "OutputSerialization": {
            "csv": {
                "FieldDelimiter": "string",
                "QuoteCharacter": "string",
                "QuoteEscapeCharacter": "string",
                "QuoteFields": "string",
                "RecordDelimiter": "string"
            }
        }
    },
    "SHA256TreeHash": "string",
    "SNSTopic": "string",
    "StatusCode": "string",
    "StatusMessage": "string",
    "Tier": "string",
    "VaultARN": "string"
}
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

응답 본문에는 다음과 같은 JSON 필드가 포함됩니다.

작업

작업 유형입니다. `ArchiveRetrieval`, `InventoryRetrieval` 또는 `Select`입니다.

유형: 문자열

ArchiveId

선택 또는 아카이브 가져오기 작업을 위해 요청된 아카이브 ID입니다. 그렇지 않으면 이 필드는 `null` 값을 갖습니다.

유형: 문자열

ArchiveSHA256TreeHash

아카이브 가져오기 작업에서 전체 아카이브의 SHA256 트리-해시입니다. 인벤토리 가져오기 작업일 때는 이 필드가 `null` 값을 갖습니다.

유형: 문자열

ArchiveSizeInBytes

`ArchiveRetrieval` 작업일 경우 이 값은 다운로드를 요청하는 아카이브의 크기(바이트)입니다. `InventoryRetrieval` 작업일 경우 이 값은 `null`입니다.

형식: 숫자

Completed

작업 상태입니다. 아카이브 또는 인벤토리 가져오기 작업이 완료되면 작업 출력 가져오기(GET output)을 사용하여 작업의 출력을 가져올 수 있습니다.

유형: 부울

CompletionDate

작업 요청이 완료된 협정 세계시(UTC) 시간입니다. 작업이 진행 중일 때는 null 값을 갖습니다.

유형: 문자열

CreationDate

작업이 생성된 UTC 시간입니다.

유형: ISO 8601 날짜 형식의 문자열 표현입니다(예: `2013-03-20T17:03:43.221Z`).

InventoryRetrievalParameters

범위가 지정된 인벤토리 가져오기에 사용되는 입력 파라미터입니다.

유형: InventoryRetrievalJobInput 객체

InventorySizeInBytes

`InventoryRetrieval` 작업일 경우 이 값은 다운로드를 요청하는 인벤토리의 크기(바이트)입니다. `ArchiveRetrieval` 또는 `Select` 작업의 경우 이 값은 `null`입니다.

형식: 숫자

JobDescription

작업을 시작할 때 입력한 작업 설명입니다.

유형: 문자열

JobId

Amazon Glacier에서 작업을 식별하는 ID입니다.

유형: 문자열

JobOutputPath

작업 출력 위치가 포함되어 있습니다.

유형: 문자열

OutputLocation

선택 작업 결과와 오류가 저장되는 위치에 관한 정보가 포함된 객체입니다.

유형: OutputLocation 객체

RetrievalByteRange

아카이브 가져오기 작업에 대해 가져온 바이트 범위이며, 형식은 "*StartByteValue*-*EndByteValue*"를 따릅니다. 아카이브 가져오기에서 범위를 지정하지 않으

면 전체 아카이브를 가져옵니다. 또한 StartByteValue의 값은 0이고, EndByteValue의 값은 아카이브 크기에서 1을 뺀 값입니다. 인벤토리 가져오기 또는 선택 작업일 때는 이 필드가 null 값을 갖습니다.

유형: 문자열

SelectParameters

선택에 사용되는 파라미터에 관한 정보를 포함하는 객체입니다.

유형: SelectParameters 객체

SHA256TreeHash

요청하는 아카이브 범위에 대한 SHA256 트리-해시 값입니다. 아카이브에 대한 작업 시작(POST jobs) 요청에서 트리-해시 정렬 범위를 지정한 경우 이 필드가 값을 반환합니다. 아카이브 범위 가져오기에서 트리-해시 정렬에 대한 자세한 내용은 데이터 다운로드 시 체크섬 수신 섹션을 참조하세요.

전체 아카이브를 가져오는 명확한 경우에는 이 값이 ArchiveSHA256TreeHash 값과 동일합니다.

이 필드는 다음 상황에서 null 값을 갖습니다.

- 아카이브 가져오기 작업에서 트리-해시로 정렬되지 않은 범위를 지정할 때
- 아카이브 작업에서 전체 아카이브와 같은 범위를 지정하고, 작업 상태가 InProgress일 때
- 인벤토리 작업일 때
- 작업을 선택합니다.

유형: 문자열

SNSTopic

알림을 받는 Amazon SNS 토픽입니다.

유형: 문자열

StatusCode

작업 상태를 나타내는 코드입니다.

유효한 값: InProgress | Succeeded | Failed

유형: 문자열

StatusMessage

쉽게 이해할 수 있도록 작업 상태를 설명하는 메시지입니다.

유형: 문자열

계층

선택 또는 아카이브 가져오기에 사용할 데이터 액세스 계층입니다.

유효한 값: `Bulk` | `Expedited` | `Standard`

유형: 문자열

VaultARN

작업이 하위 리소스인 볼트의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예제

다음은 아카이브 가져오기 작업을 요청하는 예제입니다.

요청 예제: 작업 설명 가져오기

```
GET /-/vaults/examplevault/jobs/HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

응답 본문에는 지정한 작업을 설명하는 JSON이 포함됩니다. 이때 인벤토리 가져오기 작업 및 아카이브 가져오기 작업의 경우 JSON 필드는 동일합니다. 하지만 필드가 작업 유형에 적용되지 않을 때는 `null` 값을 갖습니다. 다음은 아카이브 가져오기 작업의 응답 예제입니다. 다음 사항에 유의하세요.

- Action 필드 값은 ArchiveRetrieval입니다.
- ArchiveSizeInBytes 필드는 아카이브 가져오기 작업에서 요청하는 아카이브의 크기를 나타냅니다.
- ArchiveSHA256TreeHash 필드는 전체 아카이브에 대한 SHA256 트리-해시를 나타냅니다.
- RetrievalByteRange 필드는 작업 시작 요청에서 요청하는 범위를 나타냅니다. 이 예제에서는 전체 아카이브를 요청합니다.
- SHA256TreeHash 필드는 작업 시작 요청에서 요청하는 범위에 대한 SHA256 트리-해시를 나타냅니다. 이 예제에서는 ArchiveSHA256TreeHash 필드와 동일하며, 이는 전체 아카이브를 요청했다는 것을 의미합니다.
- InventorySizeInBytes 필드는 아카이브 가져오기 작업에 적용되지 않기 때문에 null 값을 갖습니다.

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/json
Content-Length: 419
{
  "Action": "ArchiveRetrieval",
  "ArchiveId": "NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-
TjhqG6eGoOY9Z8i1_AUyUsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchi
  "ArchiveSizeInBytes": 16777216,
  "ArchiveSHA256TreeHash":
 "beb0fe31a1c7ca8c6c04d574ea906e3f97b31fdca7571defb5b44dca89b5af60",
  "Completed": false,
  "CompletionDate": null,
  "CreationDate": "2012-05-15T17:21:39.339Z",
  "InventorySizeInBytes": null,
  "JobDescription": "My ArchiveRetrieval Job",
  "JobId": "HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID",
  "RetrievalByteRange": "0-16777215",
  "SHA256TreeHash": "beb0fe31a1c7ca8c6c04d574ea906e3f97b31fdca7571defb5b44dca89b5af60",
  "SNSTopic": "arn:aws:sns:us-west-2:012345678901:mytopic",
  "StatusCode": "InProgress",
  "StatusMessage": "Operation in progress.",
  "Tier": "Bulk",
  "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault"
}
```

다음은 인벤토리 가져오기 작업의 응답 예제입니다. 다음 사항에 유의하세요.

- Action 필드 값은 InventoryRetrieval입니다.
- ArchiveSizeInBytes, ArchiveSHA256TreeHash 및 RetrievalByteRange 필드는 인벤토리 가져오기 작업에 적용되지 않기 때문에 null 값을 갖습니다.
- 작업이 여전히 진행 중이고 다운로드할 인벤토리가 완전히 준비되지 않았기 때문에 InventorySizeInBytes 필드 값은 null입니다. 하지만 작업 설명 요청 전에 작업이 완료되었다면 이 필드에서 출력 크기가 반환됩니다.

```
{
   "Action": "InventoryRetrieval",
   "ArchiveId": null,
   "ArchiveSizeInBytes": null,
   "ArchiveSHA256TreeHash": null,
   "Completed": false,
   "CompletionDate": null,
   "CreationDate": "2012-05-15T23:18:13.224Z",
   "InventorySizeInBytes": null,
   "JobDescription": "Inventory Description",
   "JobId": "HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID",
   "RetrievalByteRange": null,
   "SHA256TreeHash": null,
   "SNSTopic": "arn:aws:sns:us-west-2:012345678901:mytopic",
   "StatusCode": "InProgress",
   "StatusMessage": "Operation in progress.",
   "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault"
}
```

다음은 볼트 인벤토리 가져오기의 페이지 매김을 계속하는 데 사용되는 마커를 포함하는 완료된 인벤토리 가져오기 작업의 응답 예제입니다.

```
{
    "Action": "InventoryRetrieval",
    "ArchiveId": null,
    "ArchiveSHA256TreeHash": null,
    "ArchiveSizeInBytes": null,
    "Completed": true,
    "CompletionDate": "2013-12-05T21:51:13.591Z",
    "CreationDate": "2013-12-05T21:51:12.281Z",
```

```
    "InventorySizeInBytes": 777062,
    "JobDescription": null,
    "JobId": "sCC2RZNBF2nildYD_roe0J9bHRdPQUbDRkmTdg-mXi2u3lc49uW6TcEhDF2D9pB2phx-
BN30JaBru7PMyOlfXHdStzu8",
    "NextInventoryRetrievalMarker": null,
    "RetrievalByteRange": null,
    "SHA256TreeHash": null,
    "SNSTopic": null,
    "StatusCode": "Succeeded",
    "StatusMessage": "Succeeded",
    "Tier": "Bulk",
    "VaultARN": "arn:aws:glacier-devo:us-west-2:836579025725:vaults/inventory-
icecube-2",
    "InventoryRetrievalParameters": {
        "StartDate": "2013-11-12T13:43:12Z",
        "EndDate": "2013-11-20T08:12:45Z",
        "Limit": "120000",
        "Format": "JSON",
        "Marker":
 "vyS0t2jHQe5qbcDggIeD50chS1SXwYMrkVKo0KHiTUjEYxBGCqRLKaiySzdN7QXGVVV5XZpNVG67pCZ_uykQXFMLaxOSu
    },
}
```

관련 섹션

- 작업 출력 가져오기(GET output)
- Amazon Glacier의 ID 및 액세스 관리(IAM)

작업 출력 가져오기(GET output)

설명

이 작업에서는 작업 시작(POST jobs)를 사용하여 시작한 작업의 출력을 다운로드합니다. 작업을 시작할 때 지정한 작업 유형에 따라서 출력 내용은 아카이브 또는 볼트 인벤토리가 될 수 있습니다.

모든 작업 출력을 다운로드하거나, 바이트 범위를 지정하여 출력 일부를 다운로드할 수도 있습니다. 아카이브 가져오기 작업과 인벤토리 가져오기 작업 모두 다운로드 크기를 작업 출력 가져오기 응답에서 헤더로 반환되는 크기와 비교하여 맞는지 확인해야 합니다.

아카이브 가져오기 작업일 때는 다운로드 크기가 예상되는 크기와 맞는지도 확인해야 합니다. 출력 일부만 다운로드하는 경우에는 예상되는 크기가 지정한 바이트 범위에 따라 달라집니다. 예를 들어 범위

를 bytes=0-1048575로 지정하는 경우에는 다운로드 크기가 1,048,576바이트이어야 합니다. 전체 아카이브를 다운로드하는 경우 예상 크기는 Amazon Glacier(Amazon Glacier)에 아카이브를 업로드했을 때의 크기입니다. 예상되는 크기는 작업 출력 가져오기 응답에서 헤더로도 반환됩니다.

아카이브 가져오기 작업의 경우, 지정하는 바이트 범위에 따라서 Amazon Glacier가 데이터 부분에 대한 체크섬을 반환합니다. 다운로드한 부분이 정확한 데이터인지 확인하려면 클라이언트에서 체크섬을 계산하여 값이 일치하는지, 그리고 예상한 크기가 맞는지도 함께 확인합니다.

작업 ID는 Amazon Glacier가 작업을 완료한 후 최소 24시간 동안 만료되지 않습니다. 다시 말해서 Amazon Glacier가 작업을 완료한 후 24시간 안에는 언제든지 작업 출력을 다운로드할 수 있습니다.

Requests

구문

작업 출력을 가져오려면 HTTP GET 요청을 특정 작업의 output URI로 전송합니다.

```
GET /AccountId/vaults/VaultName/jobs/JobID/output HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
Range: ByteRangeToRetrieve
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 값은 볼트를 소유한 계정의 AWS 계정 ID입니다. AWS 계정 ID를 지정하거나 선택적으로 단일 '-'(하이픈)를 지정할 수 있습니다.이 경우 Amazon Glacier는 요청에 서명하는 데 사용되는 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 사용하는 경우 ID에 하이픈('-')을 포함할 수 없습니다.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더 외에 다음 요청 헤더를 사용합니다. 일반적인 요청 헤더에 대한 자세한 내용은 공통 요청 헤더 섹션을 참조하세요.

이름	설명	필수
Range	출력에서 가져올 바이트 범위입니다. 예를 들어 첫 번째 1,048,576바이트를 다운로드하려면 범위를 `bytes=0-1048575` 로 지정합니다. 자세한 내용은 Range 헤더 필드 정의 섹션을 참조하세요. 범위는 작업 시작 요청에서 지정하는 범위에 비례합니다. 기본적으로 이 작업에서는 전체 출력을 다운로드합니다. 작업 출력 크기가 큰 경우에는 Range 요청 헤더를 사용하여 출력 중 일부를 가져올 수 있습니다. 이 경우 전체 출력을 작은 바이트 청크로 나누어 다운로드할 수 있습니다. 예를 들어 1GB의 작업 출력을 한 번에 128MB씩 데이터 청크로 나누어 다운로드하여 작업 출력 가져오기 요청을 총 8번 전송한다고 가정하겠습니다. 이때는 다음 프로세스에 따라 작업 출력을 다운로드합니다. 1. Range 헤더를 사용하여 적합한 바이트 범위를 지정하고 128MB 청크씩 출력을 다운로드합니다. 128MB의 데이터가 모두 다운로드되었는지 확인합니다. 2. 응답에는 데이터와 함께 페이로드 체크섬도 포함됩니다. 클라이언트에서 페이로드 체크섬을 계산한 후 응답으로 수신한 체크섬과 비교하여 예상 데이터를 모두 수신했는지 확인합니다. 3. 128MB의 출력 데이터 청크 8개 모두에 대해 1단계와 2단계를 반복하면서 각각 적합한 바이트 범위를 지정합니다. 4. 작업 출력 부분을 모두 다운로드하면 체크섬 값 8개로 이루어진 목록이 생깁니다. 이 값의 트리-해시를 계산하여 전체 출력의 체크섬을 구합니다. 작업 설명(GET JobID) 작업을 사용하여 출력을 제공한 작업에 대한 정보를 가져옵니다. 응답에는 Amazon Glacier에 저장된 전체 아카이브의 체크섬이 포함됩니다. 이 값과 직접 계산한 체크섬을 비교하여 전체 아카이브 내용이 아무런 오류도 없이 다운로드되었는지 확인합니다.	아니요

이름	설명	필수
	유형: 문자열 기본값: None 제약 조건: 없음	

요청 본문

이 작업에는 요청 본문이 없습니다.

응답

구문

모든 작업 데이터를 반환하는 가져오기 요청이 전송되면 작업 출력 응답으로 `200 OK` 응답 코드가 반환됩니다. 내용을 부분적으로 요청할 때, 예를 들어 요청에서 Range 헤더를 지정하면 응답 코드로 `206 Partial Content`가 반환됩니다.

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: ContentType
Content-Length: Length
x-amz-sha256-tree-hash: ChecksumComputedByAmazonGlacier

[Body containing job output.]
```

응답 헤더

헤더	설명
`Content-Range`	Amazon Glacier가 반환하는 바이트 범위입니다. 부분적 출력만 다운로드하는 경우에는 응답에서 Amazon Glacier가 반환한 바이트 범위를 보여줍니다. 예를 들어 `bytes 0-1048575/8388608` 는 8MB에서 첫 번째 1MB를 반환합니다.

헤더	설명
	`Content-Range` 헤더에 대한 자세한 내용은 Content-Range 헤더 필드 정의 섹션을 참조하세요. 유형: 문자열
`Content-Type`	Content-Type은 작업 출력이 아카이브인지 또는 볼트 인벤토리인지에 따라 달라집니다. • 아카이브 데이터일 경우 Content-Type은 `application/octet-stream` 입니다. • 볼트 인벤토리의 경우 작업을 시작할 때 CSV 형식을 요청하였다면 Content-Type은 `text/csv`입니다. 그 밖에는 기본적으로 볼트 인벤토리가 JSON 형식으로 반환되며, 이때 Content-Type은 `application/json` 입니다. 유형: 문자열

헤더	설명
x-amz-sha256-tree-hash	응답으로 반환되는 데이터 체크섬입니다. 이 헤더는 아카이브 가져오기 작업에 대한 출력을 가져올 때만 반환됩니다. 또한 이 헤더는 작업 시작 요청에서 요청한 데이터 범위가 트리-해시로 정렬되고, 작업 출력 가져오기에서 다운로드되는 범위 또한 트리-해시로 정렬될 경우에 표시됩니다. 트리-해시 정렬 범위에 대한 자세한 내용은 데이터 다운로드 시 체크섬 수신 섹션을 참조하세요. 예를 들어 작업 시작 요청에서 가져올 트리-해시 정렬 범위(전체 아카이브 포함)를 지정한 경우에는 다음 조건 하에서 다운로드하는 데이터의 체크섬이 수신됩니다. • 가져온 데이터의 전체 범위를 다운로드합니다. • 크기가 2의 거듭제곱으로 곱한 1MB(1024KB)이고, 요청한 범위의 크기 승수로 시작하고 끝날 수 있도록 가져온 데이터의 바이트 범위를 요청합니다. 예를 들어 가져온 데이터가 3.1MB이고, 1MB로 시작하여 2MB로 끝나도록 반환 범위를 지정한다면 x-amz-sha256-tree-hash 가 응답 헤더로 반환됩니다. • 데이터 끝까지 가도록 가져온 데이터의 반환 범위를 요청하고, 범위 시작은 가져올 범위 크기의 승수입니다. 단, 다음 2의 거듭제곱까지 반올림하되 1MB(1024KB)보다는 작지 않습니다. 예를 들어 가져온 데이터가 3.1MB이고, 2MB로 시작하여 3.1MB(데이터의 끝)로 끝나도록 범위를 지정한다면 x-amz-sha256-tree-hash 가 응답 헤더로 반환됩니다. 유형: 문자열

응답 본문

Amazon Glacier는 작업 출력을 응답 본문에 반환합니다. 이때 출력은 작업 유형에 따라 아카이브 내용 또는 볼트 인벤토리가 될 수 있습니다. 볼트 인벤토리의 경우에는 기본적으로 인벤토리 목록이 아래와 같은 JSON 본문으로 반환됩니다.

```
{
 "VaultARN": String,
 "InventoryDate": String,
 "ArchiveList": [
      {"ArchiveId": String,
       "ArchiveDescription": String,
       "CreationDate": String,
       "Size": Number,
       "SHA256TreeHash": String
      },
      ...
    ]
}
```

볼트 인벤토리 작업을 시작할 때 CSV(쉼표로 구분된 값) 출력 형식을 요청한 경우 볼트 인벤토리가 본문에서 CSV 형식으로 반환됩니다. CSV 형식에는 "ArchiveId", "ArchiveDescription", "CreationDate", "Size", "SHA256TreeHash" 등 열이 5개 있으며, 각 열의 정의는 해당하는 JSON 필드와 같습니다.

> **Note**
>
> 반환된 CSV 형식에서는 전체 필드가 큰따옴표로 묶여서 반환될 수 있습니다. 쉼표 또는 큰따옴표가 포함된 필드는 항상 큰따옴표로 묶여서 반환됩니다. 예를 들어 `my archive description,1`은 `"my archive description,1"`로 반환됩니다. 큰따옴표로 묶여서 반환되는 필드에 포함된 큰따옴표 문자는 앞에 백슬래시 문자와 함께 이스케이프 처리됩니다. 예를 들어 `my archive description,1"2`는 `"my archive description,1\"2"`로, 그리고 `my archive description,1\"2`는 `"my archive description,1\\"2"`로 반환됩니다. 백슬래시 문자는 이스케이프 처리되지 않습니다.

JSON 응답 본문에는 다음과 같은 JSON 필드가 포함됩니다.

ArchiveDescription

아카이브에 대한 설명입니다.

유형: 문자열

ArchiveId

아카이브 ID입니다.

유형: 문자열

ArchiveList

아카이브 메타데이터의 배열입니다. 이 배열을 구성하는 객체는 각각 볼트에 저장된 아카이브 1개의 메타데이터를 나타냅니다.

유형: 배열

CreationDate

아카이브가 생성된 UTC 날짜와 시간입니다.

유형: ISO 8601 날짜 형식의 문자열 표현입니다. 예: `2013-03-20T17:03:43.221Z`

InventoryDate

볼트 변경 이후 마지막으로 작성된 볼트 인벤토리의 UTC 날짜와 시간입니다. Amazon Glacier가 하루에 한 번 볼트 인벤토리를 준비하지만, 마지막 인벤토리 이후 볼트에 아카이브 추가 또는 삭제가 있는 경우에만 인벤토리 날짜가 업데이트됩니다.

유형: ISO 8601 날짜 형식의 문자열 표현입니다. 예: `2013-03-20T17:03:43.221Z`

SHA256TreeHash

아카이브의 트리-해시입니다.

유형: 문자열

Size:

아카이브 크기(바이트)입니다.

형식: 숫자

VaultARN

아카이브 가져오기가 요청된 Amazon 리소스 이름(ARN) 리소스입니다.

유형: 문자열

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예제

다음은 아카이브 가져오기 작업을 요청하는 예제입니다.

예제 1: 출력 다운로드

이 예는 아카이브 가져오기 작업 시작 요청에 대한 응답으로 Amazon Glacier가 준비한 데이터를 가져옵니다.

요청 예시

```
GET /-/vaults/examplevault/jobs/HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID/output
 HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

다음은 아카이브 가져오기 작업의 응답 예제입니다. 예제를 보면 Content-Type 헤더가 application/octet-stream이고, x-amz-sha256-tree-hash 헤더가 응답에 포함되어 있습니다. 이 말은 모든 작업 데이터가 반환된다는 것을 의미합니다.

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
x-amz-sha256-tree-hash:
 beb0fe31a1c7ca8c6c04d574ea906e3f97b31fdca7571defb5b44dca89b5af60
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/octet-stream
Content-Length: 1048576

[Archive data.]
```

다음은 인벤토리 가져오기 작업의 응답 예제입니다. 예제를 보면 Content-Type 헤더가 application/json입니다. 또한 응답에 x-amz-sha256-tree-hash 헤더가 포함되어 있지 않습니다.

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/json
Content-Length: 906

{
 "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault",
 "InventoryDate": "2011-12-12T14:19:01Z",
 "ArchiveList": [
   {
     "ArchiveId": "DMTmICA2n5Tdqq5BV2z7og-
A20xnpAPKt3UXwWxdWsn_D6auTUrW6kwy5Qyj9xd1MCE1mBYvMQ63LWaT8yTMzMaCxB_9VBWrW4Jw4zsvg5kehAPDVKcppU
oA",
     "ArchiveDescription": "my archive1",
     "CreationDate": "2012-05-15T17:19:46.700Z",
     "Size": 2140123,
     "SHA256TreeHash":
 "6b9d4cf8697bd3af6aa1b590a0b27b337da5b18988dbcc619a3e608a554a1e62"
   },
   {
     "ArchiveId": "2lHzwhKhgF2JHyvCS-
ZRuF08IQLuyB4265Hs3AXj9MoAIhz7tbXAvcFeHusgU_hViO1WeCBe0N5lsYYHRyZ7rrmRkNRuYrXUs_sjl2K8ume_7mKO_
uHE1oHqaW9d37pabXrSA",
     "ArchiveDescription": "my archive2",
     "CreationDate": "2012-05-15T17:21:39.339Z",
     "Size": 2140123,
     "SHA256TreeHash":
 "7f2fe580edb35154041fa3d4b41dd6d3adaef0c85d2ff6309f1d4b520eeecda3"
   }
  ]
}
```

예제 2: 부분 출력 다운로드

이 예는 아카이브 가져오기 작업 시작 요청에 대한 응답으로 Amazon Glacier가 준비한 아카이브의 일부만 가져옵니다. 요청에서는 옵션 Range 헤더를 사용하여 첫 번째 1,024바이트만 가져옵니다.

요청 예시

```
GET /-/vaults/examplevault/jobs/HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID/output
 HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
Range: bytes=0-1023
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

다음 성공적인 응답에서는 `206 Partial Content` 응답을 보여줍니다. 이 경우 Amazon Glacier가 반환하는 바이트 범위를 지정하는 `Content-Range` 헤더도 응답에 포함됩니다.

```
HTTP/1.1 206 Partial Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Range: bytes 0-1023/8388608
Content-Type: application/octet-stream
Content-Length: 1024

[Archive data.]
```

관련 섹션

- 작업 설명(GET JobID)
- 작업 시작(POST jobs)
- Amazon Glacier의 ID 및 액세스 관리(IAM)

작업 시작(POST jobs)

이 작업은 다음 유형의 Amazon Glacier(Amazon Glacier) 작업을 시작합니다.

- `archive-retrieval`: 아카이브 가져오기

- `inventory-retrieval`: 볼트 인벤토리 작성

주제

- 아카이브 또는 볼트 인벤토리 가져오기 작업 시작
- Requests
- 응답
- 예제
- 관련 섹션

아카이브 또는 볼트 인벤토리 가져오기 작업 시작

아카이브 또는 볼트 인벤토리 가져오기는 비동기식 작업이기 때문에 먼저 작업을 시작해야 합니다. 일단 시작되면 작업을 취소할 수 없습니다. 가져오기 작업은 2단계 프로세스로 구성됩니다.

1. 작업 시작(POST jobs) 작업을 사용하여 가져오기 작업을 시작합니다.

   > **Important**
   >
   > 데이터 가져오기 정책으로 인해 가져오기 작업 시작 요청이 `PolicyEnforcedException`과 함께 실패할 수 있습니다. 데이터 가져오기 정책에 대한 자세한 내용은 Amazon Glacier 데이터 검색 정책 섹션을 참조하세요. `PolicyEnforcedException` 예외에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

2. 작업이 완료된 후 작업 출력 가져오기(GET output) 작업을 사용하여 바이트를 다운로드합니다.

가져오기 요청은 비동기식으로 실행됩니다. 가져오기 작업을 시작하면 Amazon Glacier가 작업을 생성한 후 응답으로 작업 ID를 반환합니다. Amazon Glacier가 작업을 완료하면 작업 출력(아카이브 또는 인벤토리 데이터)을 가져올 수 있습니다. 작업 출력 가져오기에 대한 자세한 내용은 작업 출력 가져오기(GET output) 작업을 참조하세요.

작업 출력을 가져오려면 먼저 작업을 완료해야 합니다. 작업 완료 시기는 다음과 같은 옵션으로 알 수 있습니다.

- Amazon SNS 알림 사용: 작업이 완료된 후 사용자는 Amazon Glacier가 알림 메시지를 게시할 수 있는 Amazon SNS 토픽을 지정할 수 있습니다. 작업 요청 1건당 SNS 토픽 1개를 지정할 수 있습니다. 알림 메시지는 Amazon Glacier가 작업을 마친 후에만 전송됩니다. 작업 요청 1건당 SNS 토픽을 1개

지정하는 것 외에도 모든 가져오기 작업에서 작업 알림 메시지를 전송할 수 있도록 볼트 알림을 구성할 수 있습니다. 자세한 내용은 볼트 알림 구성 설정(PUT notification-configuration) 단원을 참조하십시오.

- 작업 세부 정보 가져오기: 작업이 진행 중일 때 작업 설명(GET JobID) 요청을 통해 작업 상태 정보를 가져올 수 있습니다. 그러나 Amazon SNS 알림을 사용하여 작업 완료 시점을 확인하는 방법이 더욱 효율적입니다.

> **Note**
>
> 알림 메시지를 통해 가져오는 정보는 작업 설명(GET JobID)을 호출하여 가져오는 정보와 동일합니다.

특정 이벤트에 대해 볼트에 알림 구성을 추가하고 작업 시작을 요청할 때 SNS 토픽을 지정하면 Amazon Glacier는 두 알림 메시지를 모두 전송합니다. 자세한 내용은 볼트 알림 구성 설정(PUT notification-configuration) 단원을 참조하십시오.

볼트 인벤토리

Amazon Glacier는 아카이브를 볼트에 처음 업데이트한 날짜부터 대략 하루에 한 번씩 볼트 인벤토리를 업데이트합니다. 마지막 인벤토리 이후 볼트에 대한 아카이브 추가 또는 삭제가 없는 경우에는 인벤토리 데이터가 업데이트되지 않습니다. 볼트 인벤토리 작업이 시작되면 Amazon Glacier는 마지막으로 생성한 인벤토리, 즉 실시간 데이터가 아닌 특정 시점 스냅샷을 반환합니다.

Amazon Glacier가 볼트를 위한 첫 번째 인벤토리를 생성한 뒤, 인벤토리 가져오기가 가능해지기까지는 일반적으로 반나절에서 하루가 걸립니다.

아카이브를 업로드할 때마다 볼트 인벤토리를 가져오는 것이 불필요하다고 생각할 수도 있습니다. 그러나 Amazon Glacier에 업로드하는 아카이브에 대한 메타데이터를 연결하여 클라이언트 측에서 데이터베이스를 관리한다고 가정해보겠습니다. 볼트 인벤토리가 필요에 따라 데이터베이스 정보와 실제 볼트 인벤토리를 서로 조정하는 데 얼마나 유용한지 알게 될 것입니다. 인벤토리 작업 출력으로 반환되는 데이터 필드에 대한 자세한 내용은 응답 본문 섹션을 참조하세요.

범위가 지정된 인벤토리 가져오기

아카이브 생성 날짜를 기준으로 필터링하거나 제한을 설정하여 가져오는 인벤토리 항목 수를 제한할 수 있습니다.

아카이브 생성 날짜에 따른 필터링

아카이브가 StartDate와 EndDate 사이에서 생성되었을 때 작업 시작 요청에서 두 파라미터 값을 지정하여 인벤토리 항목을 가져올 수 있습니다. 그러면 StartDate 당일 또는 이후 및 EndDate 이전에 생성된 아카이브가 반환됩니다. EndDate 없이 StartDate만 입력하면 StartDate 당일 또는 이후에 생성된 모든 아카이브의 인벤토리를 가져옵니다. 반대로 StartDate 없이 EndDate만 입력하면 EndDate 이전에 생성된 모든 아카이브의 인벤토리를 다시 가져옵니다.

가져오기 1건당 인벤토리 항목 수 제한

작업 시작 요청에서 Limit 파라미터를 설정하여 반환되는 인벤토리 항목 수를 제한할 수 있습니다. 그러면 지정된 Limit까지의 인벤토리 항목이 인벤토리 작업 출력에 포함됩니다. 인벤토리 항목이 더 있는 경우에는 결과에 페이지가 매겨집니다. 작업을 마친 후 작업 설명(GET JobID) 작업을 통해 후속 작업 시작 요청에서 사용하는 마커를 확인할 수 있습니다. 마커는 다음 인벤토리 항목 집합을 가져오는 시작점을 나타냅니다. 이전 작업 설명 출력의 마커로 작업 시작 요청을 반복 생성해 전체 인벤토리를 페이지별로 살펴볼 수 있습니다. null을 반환하는 작업 설명 마커를 얻을 때까지 이를 계속합니다. null은 사용할 수 있는 인벤토리 항목이 더 이상 없음을 나타냅니다.

Limit 파라미터는 날짜 범위 파라미터와 함께 사용할 수 있습니다.

범위가 지정된 아카이브 가져오기

아카이브 가져오기는 아카이브 전체 또는 범위로 시작할 수 있습니다. 범위를 지정하여 아카이브를 가져올 때는 반환할 바이트 범위 또는 전체 아카이브를 지정합니다. 지정하는 범위는 메가바이트(MB)로 정렬되어야 합니다. 다시 말해 범위의 시작 값이 1MB로 나누어져야 하고, 범위의 종료 값 + 1이 1MB로 나누어지거나 아카이브의 끝과 같아야 합니다. 범위가 지정된 아카이브 가져오기가 메가바이트 정렬을 따르지 않으면 400 응답이 반환됩니다. 그 밖에 작업 출력 가져오기(작업 출력 가져오기(GET output))를 사용해 다운로드하는 데이터의 체크섬 값을 가져오려면 범위가 트리-해시로 정렬되어야 합니다. 트리-해시 정렬 범위에 대한 자세한 내용은 데이터 다운로드 시 체크섬 수신 섹션을 참조하세요.

신속, 표준, 벌크 계층

아카이브 검색 작업을 시작할 때 요청 본문의 Tier 필드에서 다음 옵션 중 하나를 지정할 수 있습니다.

- **Expedited**: 신속 계층을 사용하면 아카이브 복원에 대한 긴급 요청이 필요한 경우, 빠르게 데이터에 액세스할 수 있습니다. 가장 큰 아카이브(250MB+)를 제외한 모든 아카이브의 경우 신속 계층을 사용하여 액세스되는 데이터는 일반적으로 1~5분 안에 사용할 수 있습니다.
- **Standard**: 표준 계층을 사용하면 몇 시간 내에 모든 아카이브에 액세스할 수 있습니다. 표준 계층을 사용하여 액세스하는 데이터는 일반적으로 3~5시간 안에 사용할 수 있습니다. 이 옵션은 계층 옵션을 지정하지 않는 작업 요청의 기본 옵션입니다.

- **Bulk**: 벌크 계층은 Amazon Glacier에서 가장 저렴한 계층으로서 페타바이트 단위의 대용량 데이터도 저렴한 비용으로 하루 만에 가져올 수 있습니다. 벌크 계층을 사용하여 액세스하는 데이터는 일반적으로 5~12시간 안에 사용할 수 있습니다.

신속 및 벌크 가져오기에 대한 자세한 내용은 Amazon Glacier 아카이브 가져오기 섹션을 참조하세요.

Requests

작업을 시작하려면 HTTP POST 메서드를 사용하여 작업 요청을 볼트의 jobs 하위 리소스로 전송합니다. 또한 요청의 JSON 문서에서 작업 요청에 대한 세부 정보를 지정합니다. 작업 유형은 Type 필드로 지정됩니다. 필요하다면 SNSTopic 필드를 지정하여 Amazon Glacier가 작업 완료 후 알림 메시지를 게시할 수 있는 Amazon SNS 토픽을 나타낼 수 있습니다.

> **Note**
>
> 토픽이 존재하지 않는 경우, 알림 메시지를 Amazon SNS에 게시하려면 토픽을 직접 생성해야 합니다. Amazon Glacier는 사용자를 위해 토픽을 생성하지 않습니다. 토픽에는 Amazon Glacier 볼트의 게시물을 수신할 권한이 있어야 합니다. Amazon Glacier는 볼트가 토픽을 게시할 권한이 있는지 확인하지 않습니다. 권한이 올바로 구성되어 있지 않으면 작업이 완료되더라도 알림 메시지가 수신되지 않을 수도 있습니다.

구문

다음은 작업 시작을 위한 요청 구문입니다.

```
POST /AccountId/vaults/VaultName/jobs HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01

{
   "jobParameters": {
      "ArchiveId": "string",
      "Description": "string",
      "Format": "string",
      "InventoryRetrievalParameters": {
         "EndDate": "string",
         "Limit": "string",
```

```
         "Marker": "string",
         "StartDate": "string"
      },
      "OutputLocation": {
         "S3": {
            "AccessControlList": [
               {
                  "Grantee": {
                     "DisplayName": "string",
                     "EmailAddress": "string",
                     "ID": "string",
                     "Type": "string",
                     "URI": "string"
                  },
                  "Permission": "string"
               }
            ],
            "BucketName": "string",
            "CannedACL": "string",
            "Encryption": {
               "EncryptionType": "string",
               "KMSContext": "string",
               "KMSKeyId": "string"
            },
            "Prefix": "string",
            "StorageClass": "string",
            "Tagging": {
               "string" : "string"
            },
            "UserMetadata": {
               "string" : "string"
            }
         }
      },
      "RetrievalByteRange": "string",
      "SelectParameters": {
         "Expression": "string",
         "ExpressionType": "string",
         "InputSerialization": {
            "csv": {
               "Comments": "string",
               "FieldDelimiter": "string",
               "FileHeaderInfo": "string",
               "QuoteCharacter": "string",
```

```
                "QuoteEscapeCharacter": "string",
                "RecordDelimiter": "string"
            }
         },
         "OutputSerialization": {
            "csv": {
                "FieldDelimiter": "string",
                "QuoteCharacter": "string",
                "QuoteEscapeCharacter": "string",
                "QuoteFields": "string",
                "RecordDelimiter": "string"
            }
         }
      },
      "SNSTopic": "string",
      "Tier": "string",
      "Type": "string"
   }
}
```

> **Note**
>
> `AccountId` 값은 볼트를 소유한 계정의 AWS 계정 ID입니다. AWS 계정 ID를 지정하거나 선택적으로 단일 '-'(하이픈)를 지정할 수 있습니다.이 경우 Amazon Glacier는 요청에 서명하는 데 사용되는 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 사용하는 경우 ID에 하이픈('-')을 포함할 수 없습니다.

요청 본문

요청은 요청 본문에서 JSON 형식의 다음 데이터를 받습니다.

jobParameters

작업 정보를 지정하기 위한 옵션을 제공합니다.

유형: jobParameters 객체

필수 항목 여부: 예

응답

Amazon Glacier가 작업을 생성합니다. 응답으로 작업 URI를 반환합니다.

구문

```
HTTP/1.1 202 Accepted
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Location: location
x-amz-job-id: jobId
x-amz-job-output-path: jobOutputPath
```

응답 헤더

헤더	설명
`Location`	작업의 상대적 URI 경로입니다. 이 URI 경로를 사용해 작업 상태를 알아볼 수 있습니다. 자세한 내용은 작업 설명(GET JobID) 단원을 참조하십시오. 유형: 문자열 기본값: None
`x-amz-job-id`	작업 ID입니다. 이 값은 `Location` 헤더에도 포함됩니다. 유형: 문자열 기본값: None
`x-amz-job-output-path`	선택 결과가 저장되는 위치의 경로입니다. 유형: 문자열 기본값: None

응답 본문

이 작업은 응답 본문을 반환하지 않습니다.

오류

이 작업에는 모든 Amazon Glacier 작업에 일반적으로 발생할 수 있는 오류 외에 다음 오류도 포함됩니다. Amazon Glacier의 오류에 대한 자세한 내용과 오류 코드 목록은 오류 응답 섹션을 참조하세요.

코드	설명	HTTP 상태 코드	Type
InsufficientCapacityException	신속 요청을 처리할 수 있는 용량이 부족한 경우에 반환됩니다. 이 오류는 신속 가져오기에만 적용되며 표준 또는 벌크 가져오기에는 적용되지 않습니다.	503 Service Unavailable	Server

예제

요청 예제: 아카이브 가져오기 작업 시작

```
POST /-/vaults/examplevault/jobs HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2

{
  "Type": "archive-retrieval",
  "ArchiveId": "NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-
TjhqG6eGoOY9Z8i1_AUyUsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchi
  "Description": "My archive description",
  "SNSTopic": "arn:aws:sns:us-west-2:111111111111:Glacier-ArchiveRetrieval-topic-
Example",
  "Tier" : "Bulk"
}
```

다음은 요청 본문의 RetrievalByteRange 필드를 사용하여 가져올 아카이브 범위를 지정하는 예제입니다.

```
{
  "Type": "archive-retrieval",
  "ArchiveId": "NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-
TjhqG6eGoOY9Z8i1_AUyUsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchi
  "Description": "My archive description",
  "RetrievalByteRange": "2097152-4194303",
  "SNSTopic": "arn:aws:sns:us-west-2:111111111111:Glacier-ArchiveRetrieval-topic-
Example",
  "Tier" : "Bulk"
}
```

응답의 예

```
HTTP/1.1 202 Accepted
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Location: /111122223333/vaults/examplevault/jobs/HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
x-amz-job-id: HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
```

요청 예제: 인벤토리 가져오기 작업 시작

다음은 인벤토리 가져오기 작업을 시작하여 examplevault 볼트에서 아카이브 목록을 가져오는 요청 예제입니다. 요청 본문에서 Format이 CSV로 설정되어 있으므로 인벤토리가 CSV 형식으로 반환된다는 것을 의미합니다.

```
POST /-/vaults/examplevault/jobs HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
Content-Type: application/x-www-form-urlencoded
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2

{
  "Type": "inventory-retrieval",
  "Description": "My inventory job",
  "Format": "CSV",
  "SNSTopic": "arn:aws:sns:us-west-2:111111111111:Glacier-InventoryRetrieval-topic-
Example"
```

```
}
```

응답의 예

```
HTTP/1.1 202 Accepted
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Location: /111122223333/vaults/examplevault/jobs/HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
x-amz-job-id: HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
```

요청 예제: 한도가 설정된 날짜 필터링을 사용하여 인벤토리 가져오기 작업을 시작하고, 다음 인벤토리 항목 페이지를 가져오기 위한 후속 요청을 시작합니다.

다음은 날짜 필터링을 사용하고 한도를 설정하여 볼트 인벤토리 가져오기 작업을 시작하는 요청 예제입니다.

```
{
    "ArchiveId": null,
    "Description": null,
    "Format": "CSV",
    "RetrievalByteRange": null,
    "SNSTopic": null,
    "Type": "inventory-retrieval",
    "InventoryRetrievalParameters": {
        "StartDate": "2013-12-04T21:25:42Z",
        "EndDate": "2013-12-05T21:25:42Z",
        "Limit" : "10000"
    },
}
```

다음은 작업 설명(GET JobID)에서 가져온 마커를 사용하여 다음 인벤토리 항목 페이지를 가져오는 후속 요청의 예제입니다.

```
{
    "ArchiveId": null,
    "Description": null,
    "Format": "CSV",
    "RetrievalByteRange": null,
```

```
    "SNSTopic": null,
    "Type": "inventory-retrieval",
    "InventoryRetrievalParameters": {
        "StartDate": "2013-12-04T21:25:42Z",
        "EndDate": "2013-12-05T21:25:42Z",
        "Limit": "10000",
        "Marker":
 "vyS0t2jHQe5qbcDggIeD50chS1SXwYMrkVKo0KHiTUjEYxBGCqRLKaiySzdN7QXGVVV5XZpNVG67pCZ_uykQXFMLaxOSu
    },
}
```

응답의 예

```
HTTP/1.1 202 Accepted
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Location: /111122223333/vaults/examplevault/jobs/HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
x-amz-job-id: HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
x-amz-job-output-path: test/HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID/
```

관련 섹션

- 작업 설명(GET JobID)
- 작업 출력 가져오기(GET output)
- Amazon Glacier의 ID 및 액세스 관리(IAM)

작업 목록 조회(GET jobs)

설명

이 작업에서는 진행 중인 작업과 최근에 마친 작업을 포함하여 볼트 작업 목록을 조회합니다.

> **Note**
>
> Amazon Glacier(Amazon Glacier)는 최근에 마친 작업을 삭제하기 전에 일정 기간 유지하지만 최종적으로는 완료된 작업을 삭제합니다. 완료된 작업 출력은 가져올 수 있습니다. 완료된 작

업이라고 해도 이후 일정 기간은 유지하기 때문에 작업 완료 알림 메시지를 놓치거나 첫 번째 다운로드 시도가 실패하더라도 작업 출력을 가져올 수 있습니다. 예를 들어 아카이브 가져오기 작업을 시작하여 아카이브를 다운로드한다고 가정하겠습니다. 하지만 작업이 완료되어 아카이브를 다운로드하려고 하지만 네트워크 장애가 발생하고 맙니다. 이러한 시나리오에서도 작업이 존재하는 동안에는 아카이브 다운로드를 재시도할 수 있습니다.

List Jobs 작업은 페이지 매김을 지원합니다. 따라서 항상 응답 Marker 필드를 확인해야 합니다. 목록을 조회할 작업이 더 없으면 Marker 필드가 null로 설정됩니다. 목록을 조회할 작업이 더 있으면 Marker 필드가 null이 아닌 값으로 설정되어 이 값을 목록 페이지 매김에 계속해서 사용할 수 있습니다. 특정 작업에서 시작되는 작업 목록을 반환하려면 marker 요청 파라미터를 이전 List Jobs 요청에서 가져온 해당 작업의 Marker 값으로 설정합니다.

요청에서 limit 파라미터를 지정하면 응답으로 반환되는 최대 작업 수를 제한할 수 있습니다. 기본 제한은 50개입니다. 반환되는 작업 수가 여기에서 설정하는 제한 값보다 적을 수 있지만 제한 값을 초과할 수는 없습니다.

또한 옵션으로 statuscode 파라미터 또는 completed 파라미터를 지정하거나 둘 모두 지정하여 반환되는 작업 목록을 필터링할 수도 있습니다. statuscode 파라미터를 사용하면 InProgress, Succeeded 또는 Failed 상태와 일치하는 작업만 반환하도록 지정할 수 있습니다. completed 파라미터를 사용하면 완료된 작업(true) 또는 완료되지 않은 작업(false)만 반환하도록 지정할 수 있습니다.

Requests

구문

모든 유형의 작업 목록을 반환하려면 GET 요청을 볼트의 jobs 하위 리소스 URI로 전송합니다.

```
GET /AccountId/vaults/VaultName/jobs HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 값은 볼트를 소유한 계정의 AWS 계정 ID입니다. AWS 계정 ID를 지정하거나 선택적으로 단일 '-'(하이픈)를 지정할 수 있습니다.이 경우 Amazon Glacier는 요청에 서명하는

데 사용되는 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 사용하는 경우 ID에 하이픈('-')을 포함할 수 없습니다.

요청 파라미터

이름	설명	필수
completed	반환할 작업 상태입니다. true 또는 false를 지정할 수 있습니다. 유형: Boolean 제약 조건: 없음	아니요
limit	반환할 작업의 최대 수입니다. 기본 제한은 50개입니다. 반환되는 작업 수가 여기에서 지정하는 제한 값보다 적을 수 있지만 제한 값을 초과할 수는 없습니다. 유형: 문자열 제약 조건: 최소 정수 값 1. 최대 정수 값 50	아니요
marker	작업 목록 조회가 시작되는 지점의 작업을 지정할 수 있도록 페이지 매김에 사용되는 불투명한 문자열입니다. 이전 List Jobs 응답에서 marker 값을 가져옵니다. marker는 이전 List Jobs 요청에서 시작된 결과에 페이지를 계속해서 매겨야 하는 경우에만 추가합니다. 유형: 문자열 제약 조건: 없음	아니요
statuscode	반환할 작업 상태의 유형입니다. 유형: 문자열	아니요

이름	설명	필수
	제약 조건: `InProgress` , `Succeeded` 또는 `Failed` 값 중 하나입니다.	

요청 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

요청 본문

이 작업에는 요청 본문이 없습니다.

응답

구문

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Location: Location
Content-Type: application/json
Content-Length: Length

{
    "JobList": [
        {
            "Action": "string",
            "ArchiveId": "string",
            "ArchiveSHA256TreeHash": "string",
            "ArchiveSizeInBytes": number,
            "Completed": boolean,
            "CompletionDate": "string",
            "CreationDate": "string",
            "InventoryRetrievalParameters": {
                "EndDate": "string",
                "Format": "string",
                "Limit": "string",
                "Marker": "string",
                "StartDate": "string"
            },
            "InventorySizeInBytes": number,
```

```
            "JobDescription": "string",
            "JobId": "string",
            "JobOutputPath": "string",
            "OutputLocation": {
                "S3": {
                    "AccessControlList": [
                        {
                            "Grantee": {
                                "DisplayName": "string",
                                "EmailAddress": "string",
                                "ID": "string",
                                "Type": "string",
                                "URI": "string"
                            },
                            "Permission": "string"
                        }
                    ],
                    "BucketName": "string",
                    "CannedACL": "string",
                    "Encryption": {
                        "EncryptionType": "string",
                        "KMSContext": "string",
                        "KMSKeyId": "string"
                    },
                    "Prefix": "string",
                    "StorageClass": "string",
                    "Tagging": {
                        "string": "string"
                    },
                    "UserMetadata": {
                        "string": "string"
                    }
                }
            },
            "RetrievalByteRange": "string",
            "SelectParameters": {
                "Expression": "string",
                "ExpressionType": "string",
                "InputSerialization": {
                    "csv": {
                        "Comments": "string",
                        "FieldDelimiter": "string",
                        "FileHeaderInfo": "string",
                        "QuoteCharacter": "string",
```

```
                    "QuoteEscapeCharacter": "string",
                    "RecordDelimiter": "string"
                }
            },
            "OutputSerialization": {
                "csv": {
                    "FieldDelimiter": "string",
                    "QuoteCharacter": "string",
                    "QuoteEscapeCharacter": "string",
                    "QuoteFields": "string",
                    "RecordDelimiter": "string"
                }
            }
        },
        "SHA256TreeHash": "string",
        "SNSTopic": "string",
        "StatusCode": "string",
        "StatusMessage": "string",
        "Tier": "string",
        "VaultARN": "string"
    }
  ],
  "Marker": "string"
}
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

JSON 응답 본문에는 다음과 같은 JSON 필드가 포함됩니다.

JobList

작업 객체의 목록입니다. 각 작업 객체는 작업을 설명하는 메타데이터를 포함합니다.

유형: GlacierJobDescription 객체 배열

마커

결과에 페이지를 계속해서 매기는 지점을 나타내는 불투명한 문자열입니다. 새로운 `List Jobs` 요청에서 `marker` 값을 사용하여 목록의 작업을 추가로 가져옵니다. 목록을 조회할 작업이 더 없을 경우 이 값은 `null`입니다.

유형: 문자열

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예제

다음은 볼트 작업에 대한 정보를 반환하는 방법을 설명한 예제입니다. 첫 번째 예제에서는 작업 2개로 구성된 목록이, 그리고 두 번째 예제에서는 작업의 하위 집합이 반환됩니다.

예제: 모든 작업 반환

요청 예시

다음은 볼트 작업을 반환하는 GET 요청 예제입니다.

```
GET /-/vaults/examplevault/jobs  HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

다음은 볼트 인벤토리 가져오기에서 마커로 페이지 매김을 계속하는 아카이브 가져오기 작업과 인벤토리 가져오기 작업의 응답 예제입니다. 그 밖에도 응답을 보면 `Marker` 필드가 `null`로 설정되어 더 이상 목록을 조회할 작업이 없다는 것을 알 수 있습니다.

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/json
Content-Length: 1444
```

```
{
  "JobList": [
    {
      "Action": "ArchiveRetrieval",
      "ArchiveId": "BDfaUQul0dVzYwAMr8YSa_6_8abbhZq-
i1oT69g8ByClfJyBgAGBkWl2QbF5os851P7Y7KdZDOHWJIn4rh1ZHaOYD3MgFhK_g0oDPesW34uHQoVGwoIqubf6BgUEfQm
      "ArchiveSizeInBytes": 1048576,
      "ArchiveSHA256TreeHash":
 "25499381569ab2f85e1fd0eb93c5406a178ab77c5933056eb5d6e7d4adda609b",
      "Completed": true,
      "CompletionDate": "2012-05-01T00:00:09.304Z",
      "CreationDate": "2012-05-01T00:00:06.663Z",
      "InventorySizeInBytes": null,
      "JobDescription": null,
      "JobId": "hDe9t9DTHXqFw8sBGpLQQOmIM0-
JrGtu1O_YFKLnzQ64548qJc667BRWTwBLZC76Ygy1jHYruqXkdcAhRsh0hYv4eVRU",
      "RetrievalByteRange": "0-1048575",
      "SHA256TreeHash":
 "25499381569ab2f85e1fd0eb93c5406a178ab77c5933056eb5d6e7d4adda609b",
      "SNSTopic": null,
      "StatusCode": "Succeeded",
      "StatusMessage": "Succeeded",
      "Tier": "Bulk",
      "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault"
    },
    {
      "Action": "InventoryRetrieval",
      "ArchiveId": null,
      "ArchiveSizeInBytes": null,
      "ArchiveSHA256TreeHash": null,
      "Completed": true,
      "CompletionDate": "2013-05-11T00:25:18.831Z",
      "CreationDate": "2013-05-11T00:25:14.981Z",
      "InventorySizeInBytes": 1988,
      "JobDescription": null,
      "JobId":
 "2cvVOnBL36btzyP3pobwIceiaJebM1bx9vZOOUtmNAr0KaVZ4WkWgVjiPldJ73VU7imlm0pnZriBVBebnqaAcirZq_C5"
      "RetrievalByteRange": null,
      "SHA256TreeHash": null,
      "SNSTopic": null,
      "StatusCode": "Succeeded",
      "StatusMessage": "Succeeded",
      "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault"
```

```
      "InventoryRetrievalParameters": {
          "StartDate": "2013-11-12T13:43:12Z",
          "EndDate": "2013-11-20T08:12:45Z",
          "Limit": "120000",
          "Format": "JSON",
          "Marker":
 "vyS0t2jHQe5qbcDggIeD50chS1SXwYMrkVKo0KHiTUjEYxBGCqRLKaiySzdN7QXGVVV5XZpNVG67pCZ_uykQXFMLaxOSu
    }
  ],
  "Marker": null
}
```

예제: 부분적 작업 목록 조회

요청 예시

다음은 작업을 GET 파라미터로 지정하여 반환하는 marker 요청 예제입니다. 여기에서는 limit 파라미터가 2로 설정되어 최대 2개까지 작업을 반환하도록 지정하고 있습니다.

```
GET /-/vaults/examplevault/jobs?marker=HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID&limit=2
 HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

다음은 작업 2개를 반환한 후 작업 목록에 페이지를 계속해서 매길 수 있도록 Marker 필드가 null이 아닌 값으로 설정되어 있는 응답 예제입니다.

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/json
Content-Length: 1744

{
  "JobList": [
    {
```

```
      "Action": "ArchiveRetrieval",
      "ArchiveId": "58-3KpZfcMPUznvMZNPaKyJx9wODCsWTnqcjtx2CjKZ6b-
XgxEuA8yvZOYTPQfd7gWR4GRm2XR08gcnWbLV4VPV_kDWtZJKi0TFhKKVPzwrZnA4-
FXuIBfViYUIVveeiBE51FO4bvg",
      "ArchiveSizeInBytes": 8388608,
      "ArchiveSHA256TreeHash":
 "106086b256ddf0fedf3d9e72f461d5983a2566247ebe7e1949246bc61359b4f4",
      "Completed": true,
      "CompletionDate": "2012-05-01T00:25:20.043Z",
      "CreationDate": "2012-05-01T00:25:16.344Z",
      "InventorySizeInBytes": null,
      "JobDescription": "aaabbbccc",
      "JobId": "s4MvaNHIh6mOa1f8iY4ioG2921SDPihXxh3Kv0FBX-
JbNPctpRvE4c2_BifuhdGLqEhGBNGeB6Ub-JMunR9JoVa8y1hQ",
      "RetrievalByteRange": "0-8388607",
      "SHA256TreeHash":
 "106086b256ddf0fedf3d9e72f461d5983a2566247ebe7e1949246bc61359b4f4",
      "SNSTopic": null,
      "StatusCode": "Succeeded",
      "StatusMessage": "Succeeded",
      "Tier": "Bulk",
      "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault"
    },
    {
      "Action": "ArchiveRetrieval",
      "ArchiveId": "2NVGpf83U6qB9M2u-
Ihh61yoFLRDEoh7YLZWKBn80A2i1xG8uieBwGjAr4RkzOHA0E07ZjtI267R03Z-6Hxd8pyGQkBdciCSH1-
Lw63Kx9qKpZbPCdU0uTW_WAdwF6lR6w8iSyKdvw",
      "ArchiveSizeInBytes": 1048576,
      "ArchiveSHA256TreeHash":
 "3d2ae052b2978727e0c51c0a5e32961c6a56650d1f2e4ceccab6472a5ed4a0",
      "Completed": true,
      "CompletionDate": "2012-05-01T16:59:48.444Z",
      "CreationDate": "2012-05-01T16:59:42.977Z",
      "InventorySizeInBytes": null,
      "JobDescription": "aaabbbccc",
      "JobId":
 "CQ_tf6fOR4jrJCL61Mfk6VM03oY8lmnWK93KK4gLig1UPAbZiN3UV4G_5nq4AfmJHQ_dOMLOX5k8ItFv0wCPN0oaz5dG"
      "RetrievalByteRange": "0-1048575",
      "SHA256TreeHash":
 "3d2ae052b2978727e0c51c0a5e32961c6a56650d1f2e4ceccab6472a5ed4a0",
      "SNSTopic": null,
      "StatusCode": "Succeeded",
      "StatusMessage": "Succeeded",
```

```
      "Tier": "Standard",
      "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault"
    }
  ],
  "Marker":
 "CQ_tf6fOR4jrJCL61Mfk6VM03oY8lmnWK93KK4gLig1UPAbZiN3UV4G_5nq4AfmJHQ_dOMLOX5k8ItFv0wCPN0oaz5dG"
}
```

관련 섹션

- 작업 설명(GET JobID)
- Amazon Glacier의 ID 및 액세스 관리(IAM)

작업에서 사용되는 데이터 형식

다음은 Amazon Glacier에서 작업에 사용되는 데이터 형식입니다.

주제

- CSVInput
- CSVOutput
- 암호화
- GlacierJobDescription
- 권한 부여
- 피부여자
- InputSerialization
- InventoryRetrievalJobInput
- jobParameters
- OutputLocation
- OutputSerialization
- S3Location
- SelectParameters

CSVInput

쉼표로 분리된 값(CSV) 파일에 대한 정보를 포함합니다.

내용

설명

행의 시작에 존재할 경우, 해당 행을 무시해야 함을 나타내는 데 사용되는 단일 문자입니다.

유형: 문자열

필수 항목 여부: 아니요

FieldDelimiter

한 레코드 내에서 개별 필드를 서로 분리하는 데 사용되는 단일 문자입니다. 문자는 32~126 범위의 `\n`, `\r`, 혹은 ASCII 문자여야 합니다. 기본값은 쉼표입니다 (`,`).

유형: 문자열

기본값: /

필수 항목 여부: 아니요

FileHeaderInfo

입력의 첫 번째 줄에 할 일을 설명하는 값입니다.

유형: 문자열

유효한 값: `Use` | `Ignore` | `None`

필수 항목 여부: 아니요

QuoteCharacter

필드 구분 기호가 값의 일부인 이스케이프 문자로 사용되는 단일 문자입니다.

유형: 문자열

필수 항목 여부: 아니요

QuoteEscapeCharacter

이미 이스케이프된 값 내의 따옴표 문자를 이스케이프하는 데 사용되는 단일 문자입니다.

유형: 문자열

필수 항목 여부: 아니요

RecordDelimiter

개별 레코드를 서로 분리하는 데 사용되는 단일 문자입니다.

유형: 문자열

필수 항목 여부: 아니요

추가 정보

- 작업 시작(POST jobs)

CSVOutput

작업 결과가 저장되는 쉼표로 분리된 값(CSV) 형식에 관한 정보를 포함합니다.

내용

FieldDelimiter

한 레코드 내에서 개별 필드를 서로 분리하는 데 사용되는 단일 문자입니다.

유형: 문자열

필수 항목 여부: 아니요

QuoteCharacter

필드 구분 기호가 값의 일부인 이스케이프 문자로 사용되는 단일 문자입니다.

유형: 문자열

필수 항목 여부: 아니요

QuoteEscapeCharacter

이미 이스케이프된 값 내의 따옴표 문자를 이스케이프하는 데 사용되는 단일 문자입니다.

유형: 문자열

필수 항목 여부: 아니요

QuoteFields

모든 출력 필드가 따옴표 안에 포함되어야 하는지를 나타내는 값입니다.

유효한 값: `ALWAYS | ASNEEDED`

유형: 문자열

필수 항목 여부: 아니요

RecordDelimiter

개별 레코드를 서로 분리하는 데 사용되는 단일 문자입니다.

유형: 문자열

필수 항목 여부: 아니요

추가 정보

- 작업 시작(POST jobs)

암호화

작업 결과를 Amazon S3에 저장하는 데 사용되는 암호화 관련 정보를 포함합니다.

내용

암호화

Amazon S3에서 작업 결과를 저장할 때 사용되는 서버 측 암호화 알고리즘입니다. 기본값은 암호화 없음입니다.

유형: 문자열

유효한 값: `aws:kms` | `AES256`

필수 항목 여부: 아니요

KMSContext

선택 사항입니다. 암호화 유형이 `aws:kms`,인 경우 이 값을 사용하여 작업 결과의 암호화 컨텍스트를 지정할 수 있습니다.

유형: 문자열

필수 항목 여부: 아니요

KMSKeyId

개체 암호화에 사용할 AWS Key Management Service(AWS KMS) 키 ID입니다.

유형: 문자열

필수 항목 여부: 아니요

추가 정보

- 작업 시작(POST jobs)

GlacierJobDescription

Amazon Glacier(Amazon Glacier) 작업에 대한 설명을 포함합니다.

내용

작업

작업 유형입니다. `ArchiveRetrieval`, `InventoryRetrieval` 또는 `Select`입니다.

유형: 문자열

ArchiveId

선택 또는 아카이브 가져오기 작업을 위해 요청된 아카이브 ID입니다. 그렇지 않으면 이 필드는 `null` 값을 갖습니다.

유형: 문자열

ArchiveSHA256TreeHash

아카이브 가져오기에서 전체 아카이브의 SHA256 트리-해시입니다. 인벤토리 가져오기 작업일 때는 이 필드가 null 값을 갖습니다.

유형: 문자열

ArchiveSizeInBytes

ArchiveRetrieval 작업일 경우 이 값은 다운로드를 요청하는 아카이브의 크기(바이트)입니다. InventoryRetrieval 작업일 경우 이 값은 null입니다.

유형: 숫자

Completed

작업이 완료된 경우에는 true이며, 그 밖에는 false입니다.

유형: 부울

CompletionDate

작업이 완료된 날짜입니다.

작업 요청이 완료된 협정 세계시(UTC) 시간입니다. 작업이 진행 중일 때는 null 값을 갖습니다.

유형: ISO 8601 날짜 형식의 문자열 표현입니다. 예: 2013-03-20T17:03:43.221Z

CreationDate

작업이 시작된 협정 세계시(UTC) 날짜입니다.

유형: ISO 8601 날짜 형식의 문자열 표현입니다. 예: 2013-03-20T17:03:43.221Z

InventoryRetrievalParameters

범위가 지정된 인벤토리 가져오기에 사용되는 입력 파라미터입니다.

유형: InventoryRetrievalJobInput 객체

InventorySizeInBytes

InventoryRetrieval 작업일 경우 이 값은 다운로드를 요청하는 인벤토리의 크기(바이트)입니다. ArchiveRetrieval 또는 Select 작업의 경우, null 값을 갖습니다.

유형: 숫자

JobDescription

작업을 시작할 때 입력한 작업 설명입니다.

유형: 문자열

JobId

Amazon Glacier에서 작업을 식별하는 ID입니다.

유형: 문자열

JobOutputPath

작업 출력 위치가 포함되어 있습니다.

유형: 문자열

OutputLocation

선택 작업 결과와 오류가 저장되는 위치에 관한 정보가 포함된 객체입니다.

유형: OutputLocation 객체

RetrievalByteRange

아카이브 가져오기 작업에 대해 가져온 바이트 범위이며, 형식은 "*StartByteValue*-*EndByteValue*"를 따릅니다. 아카이브 가져오기에서 범위를 지정하지 않으면 전체 아카이브를 가져옵니다. 이때 StartByteValue의 값은 0이고, EndByteValue의 값은 아카이브 크기에서 1을 뺀 값입니다. 인벤토리 가져오기 작업일 때는 이 필드가 `null` 값을 갖습니다.

유형: 문자열

SelectParameters

선택에 사용되는 파라미터에 관한 정보를 포함하는 객체입니다.

유형: SelectParameters 객체

SHA256TreeHash

요청하는 아카이브 범위에 대한 SHA256 트리-해시 값입니다. 아카이브에 대한 작업 시작(POST jobs) 요청에서 트리-해시 정렬 범위를 지정하였다면 이 필드가 값을 반환합니다. 아카이브 범위 가져오기에서 트리-해시 정렬에 대한 자세한 내용은 데이터 다운로드 시 체크섬 수신 섹션을 참조하세요.

전체 아카이브를 가져오는 특정 경우에는 이 값이 ArchiveSHA256TreeHash 값과 동일합니다.

이 필드는 다음 상황에서 null 값을 갖습니다.

- 아카이브 가져오기 작업에서 트리-해시로 정렬되지 않은 범위를 지정할 때
- 아카이브 작업에서 전체 아카이브와 같은 범위를 지정하고, 작업 상태가 InProgress일 때
- 인벤토리 작업일 때
- 작업을 선택합니다.

유형: 문자열

SNSTopic

작업 시작(작업 시작(POST jobs)) 시 알림을 구성한 경우 작업 완료 또는 중단에 대한 알림 메시지가 전송되는 Amazon SNS 토픽의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

StatusCode

작업 상태를 나타내는 코드입니다.

유효한 값: InProgress | Succeeded | Failed

유형: 문자열

StatusMessage

작업 상태 메시지입니다.

유형: 문자열

Tier

선택 또는 아카이브 가져오기에 사용할 데이터 액세스 계층입니다.

유효한 값: Expedited | Standard | Bulk

유형: 문자열

VaultARN

작업이 하위 리소스인 볼트의 ARN입니다.

유형: 문자열

추가 정보

- 작업 시작(POST jobs)

권한 부여

권한 부여에 대한 정보가 들어 있습니다.

내용

피부여자

피부여자입니다.

유형: 피부여자 객체

필수 항목 여부: 아니요

권한

피부여자에게 부여된 권한입니다.

유형: 문자열

유효한 값: `FULL_CONTROL` | `WRITE` | `WRITE_ACP` | `READ` | `READ_ACP`

필수 항목 여부: 아니요

추가 정보

- 작업 시작(POST jobs)

피부여자

피부여자에 대한 정보가 들어 있습니다.

내용

DisplayName

피부여자의 표시 이름입니다.

유형: 문자열

필수 항목 여부: 아니요

EmailAddress

피부여자의 이메일 주소입니다.

유형: 문자열

필수 항목 여부: 아니요

ID:

피부여자의 정식 사용자 ID입니다.

유형: 문자열

필수 항목 여부: 아니요

유형

피부여자의 유형입니다.

유형: 문자열

유효한 값: `AmazonCustomerByEmail` | `CanonicalUser` | `Group`

필수 항목 여부: 아니요

URI

피부여자 그룹의 URI입니다.

유형: 문자열

필수 항목 여부: 아니요

추가 정보

- 작업 시작(POST jobs)

InputSerialization

아카이브가 직렬화되는 방법을 설명합니다.

내용

CSV

CSV로 인코딩된 객체의 직렬화를 설명하는 객체입니다.

유형: CSVInput 객체

필수 항목 여부: 아니요

추가 정보

- 작업 시작(POST jobs)

InventoryRetrievalJobInput

범위가 지정된 인벤토리 가져오기 작업을 지정하기 위한 옵션을 제공합니다.

내용

EndDate

볼트 인벤토리 가져오기에서 지정하는 날짜(UTC) 범위의 끝이며, 이 날짜 이전에 생성된 아카이브도 포함됩니다.

유효한 값: ISO 8601 날짜 형식(YYYY-MM-DDThh:mm:ssTZD)의 초 단위 문자열 표현입니다. 예: 2013-03-20T17:03:43Z

유형: 문자열입니다. ISO 8601 날짜 형식(YYYY-MM-DDThh:mm:ssTZD)의 초 단위 문자열 표현입니다. 예: 2013-03-20T17:03:43Z

필수 항목 여부: 아니요

형식.

볼트 인벤토리 목록의 출력 형식이며, 볼트 인벤토리 가져오기 작업을 시작할 때 작업 시작(POST jobs) 요청에서 설정합니다.

유효한 값: CSV | JSON

필수 항목 여부: 아니요

유형: 문자열

Limit

볼트 인벤토리 가져오기 요청을 할 때마다 반환될 수 있는 인벤토리 항목의 최대 수를 지정합니다.

유효 값: 1보다 크거나 같은 정수 값

유형: 문자열

필수 항목 여부: 아니요

마커

볼트 인벤토리 가져오기 결과에 페이지를 계속해서 매기는 지점을 나타내는 불투명한 문자열입니다. 인벤토리 항목을 추가로 가져오려면 새로운 `Initiate Job` 요청에서 이 마커를 사용하면 됩니다. 인벤토리 항목이 더 없을 경우 이 값은 null 입니다.

유형: 문자열

필수 항목 여부: 아니요

StartDate

볼트 인벤토리 가져오기에서 지정하는 날짜(UTC) 범위의 시작이며, 이 날짜 당일 또는 이후에 생성된 아카이브도 포함됩니다.

유효한 값: ISO 8601 날짜 형식(`YYYY-MM-DDThh:mm:ssTZD`)의 초 단위 문자열 표현입니다. 예: `2013-03-20T17:03:43Z`

유형: 문자열입니다. ISO 8601 날짜 형식(`YYYY-MM-DDThh:mm:ssTZD`)의 초 단위 문자열 표현입니다. 예: `2013-03-20T17:03:43Z`

필수 항목 여부: 아니요

추가 정보

- 작업 시작(POST jobs)

jobParameters

작업을 정의하기 위한 옵션을 제공합니다.

내용

ArchiveId

원하는 아카이브의 ID입니다. Type 필드가 select 또는 archive-retrieval로 설정된 경우에 이 필드는 필수입니다. 인벤토리 가져오기 작업 요청에 대해 이 필드를 지정하면 오류가 발생합니다.

유효한 값: Amazon Glacier(Amazon Glacier)에 대한 이전 요청에서 얻은 유효한 아카이브 ID이어야 합니다.

유형: 문자열

필수 항목 여부: Type이 select 또는 archive-retrieval로 설정된 경우, 예

설명

작업 설명(옵션)입니다.

유효한 값: 설명은 1,024바이트보다 작거나 같아야 합니다. 허용되는 문자는 제어 코드를 제외한 7비트 ASCII로 그 중에서도 특히 ASCII 값 32~126 십진수 또는 0x20~0x7E 16진수입니다.

유형: 문자열

필수 항목 여부: 아니요

형식.

(선택 사항) 볼트 인벤토리 가져오기 작업을 시작할 때 출력 형식입니다. 인벤토리 작업을 시작하면서 Format 필드를 지정하지 않으면 JSON이 기본 형식으로 사용됩니다.

유효한 값: CSV | JSON

유형: 문자열

필수 항목 여부: 아니요

InventoryRetrievalParameters

범위가 지정된 인벤토리 가져오기에 사용되는 입력 파라미터입니다.

유형: InventoryRetrievalJobInput 객체

필수 항목 여부: 아니요

OutputLocation

선택 작업 결과가 저장되는 위치에 관한 정보가 포함된 객체입니다.

유형: OutputLocation 객체

필수 항목 여부: `select` 작업의 경우, 예.

RetrievalByteRange

`archive-retrieval`에 대해 검색할 바이트 범위는 '*StartByteValue*~*EndByteValue*' 형식이 되어야 합니다. 이 필드를 지정하지 않으면 전체 아카이브를 가져옵니다. 이 필드를 지정할 경우에는 바이트 범위가 메가바이트(1024*1024)로 정렬되어야 합니다. 메가바이트 정렬은 StartByteValue가 1MB로 나누어지고, EndByteValue + 1이 1MB로 나누어지거나 혹은 아카이브 끝이 아카이브 바이트 크기 값에서 1을 뺀 값으로 지정되어야 한다는 것을 의미합니다. RetrievalByteRange가 메가바이트로 정렬되지 않으면 `400` 응답이 반환됩니다.

`inventory-retrieval` 또는 `select` 작업 요청에 대해 이 필드를 지정하면 오류가 발생합니다.

유형: 문자열

필수 항목 여부: 아니요

SelectParameters

선택에 사용되는 파라미터에 관한 정보를 포함하는 객체입니다.

유형: SelectParameters 객체

필수 항목 여부: 아니요

SNSTopic

작업이 완료되고 출력을 다운로드할 수 있을 때 Amazon Glacier가 알림 메시지를 전송하는 Amazon SNS 토픽의 Amazon 리소스 이름(ARN)입니다. 지정된 토픽은 알림 메시지를 구독자에게 게시합니다.

이때 SNS 토픽은 반드시 존재해야 합니다. 존재하지 않는 경우, Amazon Glacier는 자동으로 토픽을 생성하지 않습니다. 또한 SNS 토픽은 작업을 생성한 계정에서 메시지를 토픽에 게시할 수 있는 정책이 필요합니다. SNS 토픽 이름에 대한 자세한 정보는 Amazon Simple Notification Service의 API 참조에서 CreateTopic을 참조하세요.

유형: 문자열

필수 항목 여부: 아니요

계층

선택 또는 아카이브 가져오기 작업에 사용할 계층입니다. 기본값으로 `Standard`가 사용됩니다.

유효한 값: `Expedited` | `Standard` | `Bulk`

유형: 문자열

필수 항목 여부: 아니요

유형

작업 유형입니다. 아카이브에서 선택 쿼리를 수행하거나 아카이브를 가져오거나 볼트의 인벤토리를 가져오는 작업을 시작할 수 있습니다.

유효한 값: `select` | `archive-retrieval` | `inventory-retrieval`

유형: 문자열

필수 항목 여부: 예

추가 정보

- 작업 시작(POST jobs)

OutputLocation

작업 결과와 오류가 저장되는 위치에 관한 정보를 포함합니다.

내용

S3

복원 요청 결과를 수신할 Amazon S3 위치를 설명하는 객체입니다.

유형: S3Location

필수 항목 여부: 예

추가 정보

- 작업 시작(POST jobs)

OutputSerialization

출력이 직렬화되는 방법을 설명합니다.

내용

CSV

쉼표로 분리된 값(CSV)으로 인코딩된 쿼리 결과의 직렬화를 설명하는 객체입니다.

유형: CSVOutput 객체

필수 항목 여부: 아니요

추가 정보

- 작업 시작(POST jobs)

S3Location

작업 결과가 저장되는 Amazon S3 내 위치에 관한 정보를 포함합니다.

내용

AccessControlList

저장된 결과에 대한 액세스를 제어하는 권한의 목록입니다.

유형: 권한 부여 객체 배열

필수 항목 여부: 아니요

BucketName

작업 결과가 저장되는 Amazon S3 버킷의 이름입니다. 버킷은 입력 아카이브 객체를 포함하는 볼트와 동일한 AWS 리전에 있어야 합니다.

유형: 문자열

필수 항목 여부: 예

CannedACL

작업 결과에 적용되는 미리 준비된 액세스 제어 목록(ACL)입니다.

유형: 문자열

유효한 값: `private` | `public-read` | `public-read-write` | `aws-exec-read` | `authenticated-read` | `bucket-owner-read` | `bucket-owner-full-control`

필수 항목 여부: 아니요

암호화

작업 결과를 Amazon S3에 저장하는 데 사용되는 암호화에 관한 정보를 포함하는 객체입니다.

유형: 암호화 객체

필수 항목 여부: 아니요

접두사

이 요청의 결과 앞에 추가되는 접두사입니다. 접두사의 최대 길이는 512바이트입니다.

유형: 문자열

필수 항목 여부: 예

StorageClass

작업 결과 저장에 사용하는 스토리지 클래스입니다.

유형: 문자열

유효한 값: `STANDARD` | `REDUCED_REDUNDANCY` | `STANDARD_IA`

필수 항목 여부: 아니요

태그 지정

작업 결과에 적용되는 태그 집합입니다.

유형: 문자열 간 맵

필수 항목 여부: 아니요

UserMetadata

작업 결과와 함께 Amazon S3에 저장할 메타데이터 맵입니다.

유형: 문자열 간 맵

필수 항목 여부: 아니요

추가 정보

- 작업 시작(POST jobs)

SelectParameters

선택에 사용되는 파라미터에 관한 정보를 포함합니다.

내용

표현식

객체 선택에 사용되는 표현식입니다. 표현식은 128,000자 할당량을 초과하면 안 됩니다.

유형: 문자열

필수 항목 여부: 예

ExpressionType

제공된 표현식의 유형입니다(예: SQL).

유효 값: SQL

유형: 문자열

필수 항목 여부: 예

InputSerialization

선택에서 객체의 직렬화된 형식을 설명합니다.

유형: InputSerialization 객체

필수 항목 여부: 아니요

OutputSerialization

선택 작업의 결과가 직렬화되는 방법을 설명합니다.

필수 항목 여부: 아니요

유형: OutputSerialization 객체

추가 정보

- 작업 시작(POST jobs)

데이터 가져오기 작업

다음은 Amazon Glacier에서 사용할 수 있는 데이터 가져오기 관련 작업입니다.

주제

- 데이터 가져오기 정책 가져오기(GET policy)
- 프로비저닝된 용량 나열(GET provisioned-capacity)
- 프로비저닝된 용량 구매(POST provisioned-capacity)
- 데이터 가져오기 정책 설정(PUT policy)

데이터 가져오기 정책 가져오기(GET policy)

설명

이 작업은 GET 요청 시 지정한 AWS 계정 및 AWS 리전의 현재 데이터 검색 정책을 반환합니다. 데이터 가져오기 정책에 대한 자세한 내용은 Amazon Glacier 데이터 검색 정책 섹션을 참조하세요.

요청

현재 데이터 가져오기 정책을 반환하려면 다음 구문 예제와 같이 HTTP GET 요청을 데이터 가져오기 정책 URI로 전송합니다.

구문

```
GET /AccountId/policies/data-retrieval HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
```

```
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> `AccountId` 값은 AWS 계정 ID입니다. 이 값은 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID와 일치해야 합니다. AWS 계정 ID 또는 선택적으로 하나의 '-'(하이픈)을 지정할 수 있습니다. 이 경우에 Amazon Glacier는 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 지정하는 경우 ID에 하이픈('-')을 포함하지 않습니다.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

이 작업에는 요청 본문이 없습니다.

응답

구문

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: length
{
  "Policy":
    {
      "Rules":[
         {
            "BytesPerHour": Number,
            "Strategy": String
         }
```

```
        ]
    }
}
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

JSON 응답 본문에는 다음과 같은 JSON 필드가 포함됩니다.

BytesPerHour

한 시간이 가져올 수 있는 최대 바이트 수입니다.

이 필드는 Strategy 필드 값이 `BytesPerHour`인 경우에 한해 나타납니다.

형식: 숫자

규칙

정책 규칙입니다. 이 규칙이 목록 형식이기는 하지만 현재는 Strategy 필드와 옵션으로 BytesPerHour 필드로 구성된 규칙 하나만 있습니다.

유형: 배열

Strategy

데이터 가져오기 정책의 유형입니다.

유형: 문자열

유효한 값: `BytesPerHour`|`FreeTier`|None. `BytesPerHour`은 콘솔에서 최대 가져오기 속도를 선택하는 것과 동일합니다. `FreeTier`는 콘솔에서 프리 티어만을 선택하는 것과 동일합니다. None은 콘솔에서 가져오기 정책 없음을 선택하는 것과 동일합니다. 콘솔에서 데이터 가져오기 정책을 선택하는 것에 대한 자세한 내용은 Amazon Glacier 데이터 검색 정책 섹션을 참조하세요.

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예시

다음은 데이터 가져오기 정책을 가져오는 방법을 설명한 예제입니다.

요청 예시

이 예제에서는 GET 요청이 정책의 위치 URI로 전송됩니다.

```
GET /-/policies/data-retrieval HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

응답이 성공적으로 반환되면 데이터 가져오기 정책이 응답 본문에 JSON 형식으로 표시됩니다.

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/json
Content-Length: 85

{
  "Policy":
    {
      "Rules":[
         {
           "BytesPerHour":10737418240,
           "Strategy":"BytesPerHour"
          }
       ]
    }
}
```

관련 섹션

- 데이터 가져오기 정책 설정(PUT policy)

- 작업 시작(POST jobs)

프로비저닝된 용량 나열(GET provisioned-capacity)

이 작업은 지정된 AWS 계정를 위해 프로비저닝된 용량 단위를 나열합니다. 프로비저닝된 용량에 대한 자세한 내용은 아카이브 검색 옵션 섹션을 참조하세요.

프로비저닝된 용량 단위는 구매한 날짜 및 시간, 즉 시작 날짜로부터 1개월 지속됩니다. 단위는 시작 날짜로부터 정확하게 1개월(초 단위) 후인 만료 날짜에 만료됩니다.

시작 날짜가 31일인 경우에는 만료 날짜는 다음 달 말일이 됩니다. 예를 들어 시작 날짜가 8월 31일이라면 만료 날짜는 9월 30일입니다. 시작 날짜가 1월 31일이라면 만료 날짜는 2월 28일입니다. 응답의 예에서 이러한 기능을 볼 수 있습니다.

Request Syntax

계정에 프로비저닝된 가져오기 용량을 나열하려면 다음 구문 예제와 같이 HTTP GET 요청을 프로비저닝된 용량 URI에게 전송합니다.

```
GET /AccountId/provisioned-capacity HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 값은 AWS 계정 ID입니다. 이 값은 요청에 서명하는 데 사용되는 자격 증명과 연결된 AWS 계정 ID와 일치해야 합니다. AWS 계정 ID를 지정하거나 선택적으로 단일 '-'(하이픈)를 지정할 수 있습니다.이 경우 Amazon Glacier는 요청에 서명하는 데 사용되는 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 지정하는 경우 ID에 하이픈('-')을 포함하지 않습니다.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

이 작업에는 요청 본문이 없습니다.

응답

작업이 성공하면 서비스가 HTTP 200 OK 응답을 다시 전송합니다.

Response Syntax

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: Length
{
   "ProvisionedCapacityList":
      {
         "CapacityId" : "string",
         "StartDate" : "string"
         "ExpirationDate" : "string"
      }
}
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

JSON 응답 본문에는 다음과 같은 JSON 필드가 포함됩니다.

CapacityId

프로비저닝된 용량 단위를 식별하는 ID입니다.

유형: 문자열.

StartDate

프로비저닝된 용량 단위를 구매한 날짜(UTC)입니다.

유형: 문자열. ISO 8601 날짜 형식의 문자열 표현입니다. 예: `2013-03-20T17:03:43.221Z`

ExpirationDate

프로비저닝된 용량 단위가 만료되는 날짜(UTC)입니다.

유형: 문자열. ISO 8601 날짜 형식의 문자열 표현입니다. 예: `2013-03-20T17:03:43.221Z`

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예제

다음은 계정에 프로비저닝된 용량 단위를 나열하는 예제입니다.

요청 예시

이 예에서는 GET 요청을 전송하여 지정된 계정에 프로비저닝된 용량 단위 목록을 가져옵니다.

```
GET /123456789012/priority-capacity HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

응답의 예

요청이 성공하면 Amazon Glacier(Amazon Glacier)가 다음 예제와 같이 계정에 프로비저닝된 용량 단위 목록과 함께 `HTTP 200 OK`를 반환합니다.

처음에 나열된 프로비저닝된 용량 단위는 시작 날짜가 2017년 1월 31일이고 만료 날짜가 2017년 2월 28일인 단위의 예입니다. 앞서 설명한 대로, 시작 날짜가 31일인 경우에는 만료 날짜는 다음 달 말일이 됩니다.

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
```

```
Date: Wed, 10 Feb 2017 12:02:00 GMT
Content-Type: application/json
Content-Length: length

{
   "ProvisionedCapacityList",
      {
         "CapacityId": "zSaq7NzHFQDANTfQkDen4V7z",
         "StartDate": "2017-01-31T14:26:33.031Z",
         "ExpirationDate": "2017-02-28T14:26:33.000Z",
      },
      {
         "CapacityId": "yXaq7NzHFQNADTfQkDen4V7z",
         "StartDate": "2016-12-13T20:11:51.095Z"",
         "ExpirationDate": "2017-01-13T20:11:51.000Z" ",
      },
      ...
}
```

관련 섹션

- 프로비저닝된 용량 구매(POST provisioned-capacity)

프로비저닝된 용량 구매(POST provisioned-capacity)

이 작업에서는 AWS 계정에 프로비저닝된 용량 단위를 구매합니다.

프로비저닝된 용량 단위는 구매한 날짜 및 시간, 즉 시작 날짜로부터 1개월 지속됩니다. 단위는 시작 날짜로부터 정확하게 1개월(초 단위) 후인 만료 날짜에 만료됩니다.

시작 날짜가 31일인 경우에는 만료 날짜는 다음 달 말일이 됩니다. 예를 들어 시작 날짜가 8월 31일이라면 만료 날짜는 9월 30일입니다. 시작 날짜가 1월 31일이라면 만료 날짜는 2월 28일입니다.

프로비저닝된 용량을 통해 필요할 때 신속 검색에 대한 검색 용량이 보장됩니다. 각 용량 단위로 5분마다 신속 검색 3회를 수행할 수 있고, 최대 150MB/s의 검색 처리량이 제공됩니다. 프로비저닝된 용량에 대한 자세한 내용은 아카이브 검색 옵션 섹션을 참조하세요.

> **Note**
>
> 프로비저닝된 용량 단위는 1개당 2개로 제한됩니다 AWS 계정.

Requests

에 대해 프로비저닝된 용량 단위를 구매하려면 프로비저닝된 용량 URI로 HTTP POST 요청을 AWS 계정 전송합니다.

구문

```
POST /AccountId/provisioned-capacity HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
Content-Length: Length
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 값은 AWS 계정 ID입니다. 이 값은 요청에 서명하는 데 사용되는 자격 증명과 연결된 AWS 계정 ID와 일치해야 합니다. AWS 계정 ID를 지정하거나 선택적으로 단일 '-'(하이픈)를 지정할 수 있습니다.이 경우 Amazon Glacier는 요청에 서명하는 데 사용되는 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 지정하는 경우 ID에 하이픈('-')을 포함하지 않습니다.

요청 파라미터

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

이 작업에는 요청 본문이 없습니다.

응답

작업 요청이 성공하면 서비스가 HTTP 201 Created 응답을 반환합니다.

구문

```
HTTP/1.1 201 Created
```

```
x-amzn-RequestId: x-amzn-RequestId
Date: Date
x-amz-capacity-id: CapacityId
```

응답 헤더

성공적인 응답에는 모든 작업에 일반적인 응답 헤더 외에 다음 응답 헤더가 포함됩니다. 일반적인 응답 헤더에 대한 자세한 내용은 공통 응답 헤더 섹션을 참조하세요.

이름	설명
x-amz-capacity-id	프로비저닝된 용량 단위를 식별하는 ID입니다. 유형: 문자열

응답 본문

이 작업은 응답 본문을 반환하지 않습니다.

오류

이 작업에는 모든 Amazon Glacier 작업에 일반적으로 발생할 수 있는 오류 외에 다음 오류도 포함됩니다. Amazon Glacier의 오류에 대한 자세한 내용과 오류 코드 목록은 오류 응답 섹션을 참조하세요.

코드	설명	HTTP 상태 코드	Type
LimitExceededException	임의의 요청이 프로비저닝된 용량 단위의 계정 제한을 초과한 경우에 반환됩니다.	400 Bad Request	클라이언트

예제

다음은 계정에 프로비저닝된 용량 단위를 구매하는 예제입니다.

요청 예시

아래 예제에서는 HTTP POST 요청을 전송하여 프로비저닝된 용량 단위를 구매합니다.

```
POST /123456789012/provisioned-capacity HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
Content-Length: length
x-amz-glacier-version: 2012-06-01
```

응답의 예

요청이 성공하면 Amazon Glacier(Amazon Glacier)가 다음 예시와 같이 `HTTP 201 Created` 응답을 반환합니다.

```
HTTP/1.1 201 Created
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
x-amz-capacity-id: zSaq7NzHFQDANTfQkDen4V7z
```

관련 섹션

- 프로비저닝된 용량 나열(GET provisioned-capacity)

데이터 가져오기 정책 설정(PUT policy)

설명

이번 작업에서는 PUT 요청 시 지정한 AWS 리전의 데이터 검색 정책을 설정한 후 규정합니다. 정책은 AWS 계정의 AWS 리전당 하나씩 설정할 수 있습니다. PUT 요청이 성공하면 몇 분 이내에 정책이 규정됩니다.

하지만 정책이 규정되기 전에는 정책 설정 작업이 진행 중인 가져오기 작업에 적용되지 않습니다. 데이터 가져오기 정책에 대한 자세한 내용은 Amazon Glacier 데이터 검색 정책 섹션을 참조하세요.

요청

구문

데이터 가져오기 정책을 설정하려면 다음 구문 예제와 같이 HTTP PUT 요청을 데이터 가져오기 정책 URI로 전송합니다.

```
PUT /AccountId/policies/data-retrieval HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
Content-Length: Length
x-amz-glacier-version: 2012-06-01

{
  "Policy":
    {
      "Rules":[
         {
             "Strategy": String,
             "BytesPerHour": Number
         }
       ]
    }
}
```

> **Note**
>
> AccountId 값은 AWS 계정 ID입니다. 이 값은 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID와 일치해야 합니다. AWS 계정 ID 또는 선택적으로 하나의 '-'(하이픈)을 지정할 수 있습니다. 이 경우에 Amazon Glacier는 요청 서명에 사용된 자격 증명과 연결된 AWS 계정 ID를 사용합니다. 계정 ID를 지정하는 경우 ID에 하이픈('-')을 포함하지 않습니다.

요청 파라미터

이 작업은 요청 파라미터를 사용하지 않습니다.

요청 헤더

이 작업은 모든 작업에 일반적인 요청 헤더만 사용합니다. 일반적인 요청 헤더에 대한 내용은 공통 요청 헤더 섹션을 참조하세요.

요청 본문

요청 본문의 JSON에 포함되는 필드는 다음과 같습니다.

BytesPerHour

한 시간이 가져올 수 있는 최대 바이트 수입니다.

이 필드는 Strategy 필드 값이 BytesPerHour인 경우에만 필요합니다. Strategy 필드가 BytesPerHour로 설정되어 있지 않은 상태에서 이 필드를 설정하면 PUT 작업이 거부됩니다.

유형: 숫자

필수 항목 여부: Strategy 필드가 BytesPerHour로 설정되어 있는 경우 예이고, 그렇지 않으면 아니요입니다.

유효한 값: 최소 정수 값 1. 최대 정수 값 2^63 - 1 포함

Rules

정책 규칙입니다. 이 규칙이 목록 유형이지만 현재는 Strategy 필드와 BytesPerHour 필드(옵션)로 구성된 규칙 하나만 있어야 합니다.

유형: 배열

필수 항목 여부: 예

Strategy

설정할 데이터 가져오기 정책의 유형입니다.

유형: 문자열

필수 항목 여부: 예

유효한 값: BytesPerHour|FreeTier|None. BytesPerHour는 콘솔에서 최대 가져오기 속도를 선택하는 것과 동일합니다. FreeTier는 콘솔에서 프리 티어만을 선택하는 것과 동일합니다. None은 콘솔에서 가져오기 정책 없음을 선택하는 것과 동일합니다. 콘솔에서 데이터 가져오기 정책을 선택하는 것에 대한 자세한 내용은 Amazon Glacier 데이터 검색 정책 섹션을 참조하세요.

응답

구문

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

응답 헤더

이 작업은 대부분의 응답에 일반적인 응답 헤더만 사용합니다. 일반적인 응답 헤더에 대한 내용은 공통 응답 헤더 섹션을 참조하세요.

응답 본문

이 작업은 응답 본문을 반환하지 않습니다.

오류

Amazon Glacier 예외 및 오류 메시지에 대한 자세한 내용은 오류 응답 섹션을 참조하세요.

예시

요청 예시

다음은 Strategy 필드가 `BytesPerHour`로 설정되어 있을 때 HTTP PUT 요청을 전송하는 예제입니다.

```
PUT /-/policies/data-retrieval HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2

{
  "Policy":
    {
      "Rules":[
         {
             "Strategy":"BytesPerHour",
             "BytesPerHour":10737418240
          }
       ]
    }
}
```

다음은 Strategy 필드가 `FreeTier`로 설정되어 있을 때 HTTP PUT 요청을 전송하는 예제입니다.

```
PUT /-/policies/data-retrieval HTTP/1.1
```

```
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2

{
  "Policy":
    {
      "Rules":[
         {
             "Strategy":"FreeTier"
          }
       ]
    }
}
```

다음은 Strategy 필드가 None으로 설정되어 있을 때 HTTP PUT 요청을 전송하는 예제입니다.

```
PUT /-/policies/data-retrieval HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2

{
  "Policy":
    {
      "Rules":[
         {
             "Strategy":"None"
          }
       ]
    }
}
```

응답의 예

요청이 성공하면 Amazon Glacier(Amazon Glacier)가 다음 예시와 같이 정책을 설정한 후 `HTTP 204 No Content`를 반환합니다.

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

관련 섹션

- 데이터 가져오기 정책 가져오기(GET policy)
- 작업 시작(POST jobs)

문서 기록

- 현재 제품 버전: 2012-06-01

다음 표는 2018년 7월 5일 이후로 Amazon Glacier 개발자 안내서의 각 릴리스에서 변경된 중요 사항을 설명합니다. 이 설명서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

변경 사항	설명	날짜
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 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 계정 별로 초당 최대 1,000개의 트랜잭션 속도로 복원 요청을 지원합니다.	2022년 11월 15일
Amazon Glacier 이름 변경	Amazon Glacier의 이름은 Glacier와 Amazon S3의 통합을 더 잘 반영하기 위해 Amazon Glacier로 변경되었습니다.	2018년 11월 20일
RSS에서 현재 사용 가능한 업데이트	이제 RSS 피드를 구독하면 Amazon Glacier 개발자 안내서의 업데이트 알림을 받을 수 있습니다.	2018년 7월 5일

이전 업데이트

다음 표는 2018년 7월 5일 이후로 Amazon Glacier 개발자 안내서의 각 릴리스에서 변경된 중요 사항을 설명합니다.

변경	설명	릴리스 날짜
신속 및 벌크 데이터 가져오기	이제 Amazon Glacier는 표준 검색 외에 신속 및 벌크 데이터 검색도 지원합니다. 자세한 내용은 아카이브 검색 옵션 단원을 참조하십시오.	2016년 11월 21일
볼트 잠금	이제 Amazon Glacier는 볼트 잠금 기능을 지원하여, 각 Amazon Glacier의 볼트마다 볼트 잠금 정책을 사용해 규정 준수 제어 항목을 쉽게 배포하고 적용할 수 있습니다. 자세한 내용은 Amazon Glacier 볼트 잠금 및 볼트 잠금 정책 섹션을 참조하세요.	2015년 7월 8일
볼트의 태그 할당	이제 Amazon Glacier에서 Amazon Glacier 볼트에 태그를 지정하여 리소스 및 비용을 더욱 쉽게 관리할 수 있습니다. 태그는 정의하여 볼트와 연결할 수 있는 레이블이며, 태그를 사용하면 AWS 비용 보고서와 같은 작업에 필터링 기능이 추가됩니다. 자세한 내용은 Amazon Glacier 리소스에 태그 지정 및 Amazon Glacier 볼트의 태그 지정 섹션을 참조하세요.	2015년 6월 22일
볼트 액세스 정책	이제 Amazon Glacier는 볼트 액세스 정책을 사용하여 각 Amazon Glacier 볼트에 대한 액세스를 관리하는 것을 지원합니다. 또한 볼트에서 액세스 정책을 직접 정의하기 때문에 기업 내부의 사용자와 비즈니스 그룹은 물론이고 외부의 비즈니스 파트너에게도 볼트에 대한 액세스 권한을 더욱 쉽게 부여할 수 있습니다. 자세한 내용은 볼트 액세스 정책 단원을 참조하십시오.	2015년 4월 27일
데이터 가져오기 정책 및 감사 로깅	이제 Amazon Glacier는 데이터 가져오기 정책과 감사 로깅을 지원합니다. 데이터 가져오기 정책이 지원되면서 이제는 데이터 가져오기 제한을 쉽게 설정하는 동시에 데이터를 가져오는 데 따른 비용을 간편하게 관리할 수 있습니다. 에서 몇 번의 클릭으로 AWS Managemen	2014년 12월 11일

변경	설명	릴리스 날짜
	t Console 또는 Amazon Glacier API를 사용하여 자체 데이터 검색 한도를 정의할 수 있습니다. 자세한 내용은 Amazon Glacier 데이터 검색 정책 단원을 참조하십시오. 또한 Amazon Glacier는 이제 계정에 대한 Amazon Glacier API 호출을 AWS CloudTrail기록하고 지정한 Amazon S3 버킷에 로그 파일을 전송하는 감사 로깅을 지원합니다. 자세한 내용은 를 사용하여 Amazon Glacier API 호출 로깅 AWS CloudTrail 단원을 참조하십시오.	
Java 샘플 업데이트	본 안내서에서 AWS SDK for Java을 사용하는 Java 코드 예시가 업데이트되었습니다.	2014년 6월 27일
볼트 인벤토리 가져오기 제한	이제 아카이브 생성 날짜를 기준으로 필터링하거나 제한을 설정하여 가져오는 볼트 인벤토리 항목 수를 제한할 수 있습니다. 인벤토리 가져오기 제한에 대한 자세한 내용은 범위가 지정된 인벤토리 가져오기 항목에서 작업 시작(POST jobs) 섹션을 참조하세요.	2013년 12월 31일
오랜 기간이 지난 URL 삭제	코드 예제에서 이전 보안 자격 증명 페이지를 가리키던 URL이 삭제되었습니다.	2013년 7월 26일
범위 가져오기 지원	이제 Amazon Glacier는 아카이브의 특정 범위 가져오기를 지원합니다. Amazon Glacier에 아카이브 전체 또는 후속 다운로드를 위한 아카이브 일부를 준비하도록 요청하는 작업을 시작할 수 있습니다. 아카이브 용량이 큰 경우에는 몇 차례 순차 작업으로 아카이브를 준비하는 것이 비용 효율적인 방법입니다. 자세한 내용은 Amazon Glacier에서 아카이브 다운로드 단원을 참조하십시오.	2012년 11월 13일
새 안내서	이 안내서는Amazon Glacier 개발자 안내서의 최초 릴리스입니다.	2012년 8월 20일

AWS 용어집

최신 AWS 용어는 AWS 용어집 참조서의 AWS 용어집을 참조하십시오.