StorageGRID의 S3 REST API 구현 방식

StorageGRID software

NetApp
July 23, 2026

목차

StorageGRID의 S3 REST API 구현 방식

StorageGRID S3 REST API가 충돌하는 클라이언트 요청을 해결하는 방법

두 클라이언트가 동일한 키에 쓰기를 시도하는 등 클라이언트 요청이 충돌하는 경우, "최신 요청 우선" 원칙에 따라 해결됩니다.

"최신 성공" 평가 시점은 S3 클라이언트가 작업을 시작한 시점이 아니라 StorageGRID 시스템이 특정 요청을 완료한 시점을 기준으로 합니다.

StorageGRID S3 REST API가 가용성과 일관성의 균형을 유지하는 방법

일관성은 객체의 가용성과 다양한 스토리지 노드 및 사이트 간의 객체 일관성 사이의 균형을 제공합니다. 애플리케이션 요구 사항에 따라 일관성 수준을 변경할 수 있습니다.

기본적으로 StorageGRID는 새로 생성된 객체에 대해 쓰기 후 읽기 일관성을 보장합니다. PUT 작업이 성공적으로 완료된 후 발생하는 모든 GET 요청은 새로 기록된 데이터를 읽을 수 있습니다. 기존 객체 덮어쓰기, 메타데이터 업데이트 및 삭제는 최종 일관성을 보장합니다.

객체 작업을 다른 일관성 수준으로 수행하려면 다음을 수행할 수 있습니다.

- 각 버킷에 대한 일관성을 지정합니다.
- 각 API 작업에 대한 일관성을 지정합니다.
- 다음 작업 중 하나를 수행하여 그리드 전체의 기본 일관성 설정을 변경하십시오.
  - 그리드 관리자에서 구성 > 시스템 > 저장소 설정 > *기본 버킷 일관성*으로 이동합니다.
  - .

> 그리드 전체 일관성 설정 변경은 해당 설정이 변경된 후에 생성된 버킷에만 적용됩니다. 변경 사항에 대한 자세한 내용을 확인하려면 `/var/local/log`에 있는 감사 로그를 참조하십시오 (**consistencyLevel** 검색).

일관성 값

일관성은 StorageGRID가 객체를 추적하는 데 사용하는 메타데이터가 노드 간에 분산되는 방식에 영향을 미칩니다. 일관성은 클라이언트 요청에 대한 객체 가용성에 영향을 미칩니다.

버킷 또는 API 작업에 대한 일관성 수준을 다음 값 중 하나로 설정할 수 있습니다.

- 모두: 모든 노드가 객체 메타데이터를 즉시 수신해야 하며, 그렇지 않으면 요청이 실패합니다.
- **Strong-global**: 모든 사이트에서 모든 클라이언트 요청에 대해 쓰기 후 읽기 일관성을 보장합니다. 쿼럼 시맨틱이 구성된 경우 다음과 같은 동작이 적용됩니다.

- 그리드에 사이트가 세 개 이상 있는 경우 클라이언트 요청에 대한 사이트 장애 허용 기능을 제공합니다. 사이트가 두 개인 그리드에는 사이트 장애 허용 기능이 없습니다.
- 다음 S3 작업은 사이트 중 하나라도 다운되면 성공하지 못합니다.
  - DeleteBucketEncryption
  - PutBucketBranch
  - PutBucketEncryption
  - PutBucketVersioning
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

필요한 경우 "강력한 전역 일관성을 위해 StorageGRID 쿼럼 시맨틱 구성".

- **Strong-site**: 객체 메타데이터가 사이트 내 다른 노드로 즉시 배포됩니다. 사이트 내 모든 클라이언트 요청에 대해 쓰기 후 읽기 일관성을 보장합니다.
- **Read-after-new-write**: 새 객체에 대해서는 쓰기 후 읽기 일관성을 제공하고, 객체 업데이트에 대해서는 최종 일관성을 제공합니다. 높은 가용성과 데이터 보호를 보장합니다. 대부분의 경우에 권장됩니다.
- 사용 가능: 새 객체 생성 및 객체 업데이트 모두에 대해 최종 일관성을 제공합니다. S3 버킷의 경우, 필요한 경우에만 사용하십시오(예: 거의 읽지 않는 로그 값이 포함된 버킷 또는 존재하지 않는 키에 대한 HEAD 또는 GET 작업). S3 FabricPool 버킷에서는 지원되지 않습니다.

"새로운 쓰기 후 읽기" 및 "사용 가능" 일관성 사용

HEAD 또는 GET 작업에서 "새로 쓰기 후 읽기" 일관성을 사용하는 경우 StorageGRID는 다음과 같이 여러 단계에 걸쳐 조회를 수행합니다.

- 먼저 낮은 일관성을 사용하여 객체를 찾습니다.
- 해당 조회가 실패하면 강력한 전역 설정에 대한 동작과 동일한 일관성 값에 도달할 때까지 다음 일관성 값에서 조회를 반복합니다.

HEAD 또는 GET 작업에서 "새로 쓰기 후 읽기" 일관성 방식을 사용하지만 객체가 존재하지 않는 경우, 객체 조회는 항상 strong-global 방식과 동일한 일관성 상태로 진행됩니다. 이 일관성 방식은 각 사이트에서 객체 메타데이터의 여러 복사본이 있어야 하므로, 동일 사이트의 스토리지 노드 두 개 이상을 사용할 수 없는 경우 500 내부 서버 오류가 많이 발생할 수 있습니다.

Amazon S3와 유사한 일관성 보장이 필요한 경우가 아니라면, HEAD 및 GET 작업에서 일관성 설정을 "Available"로 지정하여 이러한 오류를 방지할 수 있습니다. HEAD 또는 GET 작업에서 "Available" 일관성을 사용하는 경우, StorageGRID는 최종 일관성만 제공합니다. 실패한 작업에 대해 일관성 수준을 높여 재시도하지 않으므로 객체 메타데이터의 여러 복사본을 사용할 수 있어야 하는 요구 사항이 없습니다.

API 작업에 대한 일관성 지정

개별 API 작업에 대한 일관성 설정을 하려면 해당 작업에서 일관성 값이 지원되어야 하며, 요청 헤더에 일관성을 지정해야 합니다. 이 예제는 GetObject 작업에 대해 일관성을 "Strong-site"로 설정합니다.

```
GET /bucket/object HTTP/1.1
Date: date
Authorization: authorization name
Host: host
Consistency-Control: strong-site
```

PutObject 및 GetObject 작업 모두에 동일한 일관성을 유지해야 합니다.

버킷에 대한 일관성 지정

버킷의 일관성을 설정하려면 StorageGRID "PUT 버킷 일관성" 요청을 사용할 수 있습니다. 또는 테넌트 관리자에서 "버킷의 일관성 변경" 할 수 있습니다.

버킷의 일관성을 설정할 때 다음 사항에 유의하십시오.

- 버킷의 일관성 설정은 버킷의 객체 또는 버킷 구성에 대해 수행되는 S3 작업에 사용되는 일관성을 결정합니다. 이는 버킷 자체에 대한 작업에는 영향을 미치지 않습니다.
- 개별 API 작업에 대한 일관성 설정이 버킷의 일관성 설정보다 우선합니다.
- 일반적으로 버킷은 기본 일관성 방식인 "새 쓰기 후 읽기"를 사용해야 합니다. 요청이 제대로 작동하지 않으면 가능하면 애플리케이션 클라이언트의 동작을 변경하십시오. 또는 각 API 요청에 대해 일관성 방식을 지정하도록 클라이언트를 구성하십시오. 버킷 수준에서 일관성 방식을 설정하는 것은 최후의 수단으로만 사용하십시오.

일관성과 **ILM** 규칙이 상호 작용하여 데이터 보호에 영향을 미치는 방식

일관성 설정과 ILM 규칙 모두 객체 보호 방식에 영향을 미칩니다. 이러한 설정은 서로 상호 작용할 수 있습니다.

예를 들어, 객체 저장 시 사용되는 일관성 수준은 객체 메타데이터의 초기 배치에 영향을 미치며, ILM 규칙에 대해 선택된 수집 동작은 객체 복사본의 초기 배치에 영향을 미칩니다. StorageGRID는 클라이언트 요청을 처리하기 위해 객체의 메타데이터와 데이터 모두에 액세스해야 하므로, 일관성 수준과 수집 동작에 대해 동일한 수준의 보호 방식을 선택하면 초기 데이터 보호가 강화되고 시스템 응답이 더욱 예측 가능해질 수 있습니다.

다음 "수집 옵션"은(는) ILM 규칙에 사용할 수 있습니다.

이중 커밋
: StorageGRID는 즉시 객체의 임시 복사본을 생성하고 클라이언트에 성공을 반환합니다. ILM 규칙에 지정된 복사본은 가능한 경우 생성됩니다.

엄격한
: ILM 규칙에 명시된 모든 사본이 생성되어야만 클라이언트에 성공이 반환됩니다.

균형 잡힌
: StorageGRID 데이터 수집 시 ILM 규칙에 지정된 모든 복사본을 생성하려고 시도합니다. 이것이 불가능한 경우, 임시 복사본을 생성하고 클라이언트에 성공을 반환합니다. ILM 규칙에 지정된 복사본은 가능한 경우 생성됩니다.

일관성 규칙과 **ILM** 규칙이 상호 작용하는 방식의 예시

다음과 같은 ILM 규칙과 일관성 조건을 갖는 3개 사이트 그리드가 있다고 가정해 보겠습니다.

- **ILM** 규칙: 객체 복사본 3개를 생성합니다. 하나는 로컬 사이트에, 나머지 두 개는 각 원격 사이트에 생성합니다. 엄격한 수집 동작을 사용합니다.
- 일관성: Strong-global(오브젝트 메타데이터가 여러 사이트에 즉시 배포됨).

클라이언트가 그리드에 객체를 저장하면 StorageGRID는 객체의 세 가지 복사본을 모두 만들고 메타데이터를 여러 사이트에 배포한 후 클라이언트에 성공 응답을 반환합니다.

객체는 메시지 수집 성공 시점에 손실로부터 완벽하게 보호됩니다. 예를 들어, 수집 직후 로컬 사이트에 장애가 발생하더라도 객체 데이터와 객체 메타데이터의 복사본이 원격 사이트에 여전히 남아 있습니다. 따라서 다른 사이트에서 객체를 완벽하게 복구할 수 있습니다.

동일한 ILM 규칙과 강력한 사이트 일관성을 사용하는 경우, 클라이언트는 객체 데이터가 원격 사이트로 복제된 후 객체 메타데이터가 배포되기 전에 성공 메시지를 수신할 수 있습니다. 이 경우 객체 메타데이터의 보호 수준이 객체 데이터의 보호 수준과 일치하지 않습니다. 수집 직후 로컬 사이트가 손실되면 객체 메타데이터도 손실됩니다. 객체를 검색할 수 없습니다.

일관성과 ILM 규칙 간의 상호 관계는 복잡할 수 있습니다. 도움이 필요하시면 NetApp에 문의하십시오.

StorageGRID에서 객체 버전 관리를 사용하는 방법

버킷에 버전 관리 기능을 설정하면 각 객체의 여러 버전을 보존할 수 있습니다. 버킷에 버전 관리를 활성화하면 객체가 실수로 삭제되는 것을 방지하고 이전 버전의 객체를 검색 및 복원할 수 있습니다.

StorageGRID 시스템은 대부분의 기능을 지원하는 버전 관리 기능을 구현하지만, 몇 가지 제한 사항이 있습니다. StorageGRID는 각 객체에 대해 최대 10,000개의 버전을 지원합니다.

객체 버전 관리는 StorageGRID 정보 라이프사이클 관리(ILM) 또는 S3 버킷 수명 주기 구성과 결합할 수 있습니다. 각 버킷에 대해 버전 관리를 명시적으로 활성화해야 합니다. 버킷에 버전 관리가 활성화되면 해당 버킷에 추가되는 모든 객체에 StorageGRID 시스템에서 생성된 버전 ID가 할당됩니다.

다단계 인증(MFA)을 사용하는 경우 삭제는 지원되지 않습니다.

> ⓘ 버전 관리는 StorageGRID 버전 10.3 이상으로 생성된 버킷에서만 활성화할 수 있습니다.

ILM 및 버전 관리

ILM 정책은 객체의 각 버전에 적용됩니다. ILM 스캔 프로세스는 모든 객체를 지속적으로 스캔하고 현재 ILM 정책에 따라 재평가합니다. ILM 정책을 변경하면 이전에 수집된 모든 객체에 적용됩니다. 버전 관리가 활성화된 경우 이전에 수집된 버전에도 적용됩니다. ILM 스캔은 새로운 ILM 변경 사항을 이전에 수집된 객체에 적용합니다.

버전 관리가 활성화된 버킷의 S3 객체의 경우, 버전 관리 지원을 통해 "비현재 시간"을 참조 시간으로 사용하는 ILM 규칙을 생성할 수 있습니다("ILM 규칙 생성 마법사의 1단계"의 "이 규칙을 이전 객체 버전에만 적용하시겠습니까?"라는 질문에 *예*를 선택). 객체가 업데이트되면 이전 버전은 비현재 버전이 됩니다. "비현재 시간" 필터를 사용하면 이전 버전 객체의 스토리지 사용량을 줄이는 정책을 만들 수 있습니다.

> ⓘ 멀티파트 업로드 작업을 사용하여 객체의 새 버전을 업로드할 때, 원본 버전의 비현재 시간은 멀티파트 업로드가 완료된 시간이 아니라 새 버전에 대한 멀티파트 업로드가 생성된 시간을 반영합니다. 드물지만 원본 버전의 비현재 시간이 현재 버전의 시간보다 몇 시간 또는 며칠 전일 수 있습니다.

관련 정보

- "S3 버전 관리 객체가 삭제되는 방법"
- "S3 버전 관리 객체에 대한 ILM 규칙 및 정책(예제 4)".

S3 REST API를 사용하여 StorageGRID S3 Object Lock을 구성합니다

StorageGRID 시스템에서 전역 S3 객체 잠금 설정이 활성화된 경우, S3 객체 잠금이 활성화된 버킷을 생성할 수 있습니다. 각 버킷에 대한 기본 보존 기간 또는 각 객체 버전에 대한 보존 기간 설정을 지정할 수 있습니다.

버킷에 S3 Object Lock을 활성화하는 방법

StorageGRID 시스템에 대해 전역 S3 객체 잠금 설정이 활성화된 경우, 각 버킷을 생성할 때 선택적으로 S3 Object Lock을 활성화할 수 있습니다.

S3 객체 잠금은 버킷 생성 시에만 활성화할 수 있는 영구 설정입니다. 버킷 생성 후에는 S3 객체 잠금을 추가하거나 비활성화할 수 없습니다.

버킷에 S3 Object Lock을 활성화하려면 다음 방법 중 하나를 사용하십시오.

- 테넌트 관리자를 사용하여 버킷을 생성합니다. "S3 버킷 생성"을 참조하십시오.
- CreateBucket 요청과 `x-amz-bucket-object-lock-enabled` 요청 헤더를 사용하여 버킷을 생성합니다. "버킷에 대한 작업"을 참조하십시오.

S3 객체 잠금에는 버킷 버전 관리가 필요하며, 이는 버킷 생성 시 자동으로 활성화됩니다. 버킷의 버전 관리를 일시 중지할 수 없습니다. "객체 버전 관리"을 참조하십시오.

버킷의 기본 보존 설정

버킷에 대해 S3 Object Lock이 활성화된 경우, 선택적으로 버킷에 대한 기본 보존을 활성화하고 기본 보존 모드 및 기본 보존 기간을 지정할 수 있습니다.

기본 보존 모드

- 규정 준수 모드에서:
  - 해당 객체는 지정된 보존 기간이 만료될 때까지 삭제할 수 없습니다.
  - 객체의 보존 기간은 늘릴 수는 있지만 줄일 수는 없습니다.
  - 해당 날짜에 도달할 때까지 객체의 보존 기한을 제거할 수 없습니다.
- 거버넌스 모드에서:
  - `s3:BypassGovernanceRetention` 권한이 있는 사용자는 `x-amz-bypass-governance-retention: true` 요청 헤더를 사용하여 보존 설정을 우회할 수 있습니다.
  - 이러한 사용자는 보존 기한이 도래하기 전에 개체 버전을 삭제할 수 있습니다.
  - 이러한 사용자는 개체의 보존 기간을 늘리거나 줄이거나 없앨 수 있습니다.

기본 보존 기간

각 버킷에는 연 또는 일 단위로 지정된 기본 보존 기간이 있을 수 있습니다.

버킷의 기본 보존 기간을 설정하는 방법

버킷의 기본 보존 기간을 설정하려면 다음 방법 중 하나를 사용하십시오.

- 테넌트 관리자에서 버킷 설정을 관리합니다. "S3 버킷 생성" 및 "S3 객체 잠금 기본 보존 기간 업데이트"을 참조하십시오.
- 버킷에 대한 PutObjectLockConfiguration 요청을 실행하여 기본 모드와 기본 일수 또는 연수를 지정합니다.

PutObjectLockConfiguration

PutObjectLockConfiguration 요청을 사용하면 S3 객체 잠금이 활성화된 버킷의 기본 보존 모드와 기본 보존 기간을 설정하고 수정할 수 있습니다. 또한 이전에 구성된 기본 보존 설정을 제거할 수도 있습니다.

새로운 객체 버전이 버킷에 수집될 때, `x-amz-object-lock-mode` 및 `x-amz-object-lock-retain-until-date`이(가) 지정되지 않은 경우 기본 보존 모드가 적용됩니다. `x-amz-object-lock-retain-until-date`이(가) 지정되지 않으면 기본 보존 기간을 사용하여 보존 종료일을 계산합니다.

객체 버전이 수집된 후 기본 보존 기간이 수정되더라도 객체 버전의 보존 만료일은 그대로 유지되며 새 기본 보존 기간을 사용하여 재계산되지 않습니다.

이 작업을 완료하려면 `s3:PutBucketObjectLockConfiguration` 권한이 있거나 루트 계정이어야 합니다.

The `Content-MD5` 요청 헤더는 PUT 요청에 반드시 지정해야 합니다.

요청 예시

이 예제는 버킷에 대해 S3 Object Lock을 활성화하고 기본 보존 모드를 COMPLIANCE로, 기본 보존 기간을 6년으로 설정합니다.

```
PUT /bucket?object-lock HTTP/1.1
Accept-Encoding: identity
Content-Length: 308
Host: host
Content-MD5: request header
User-Agent: s3sign/1.0.0 requests/2.24.0 python/3.8.2
X-Amz-Date: date
X-Amz-Content-SHA256: authorization-string
Authorization: authorization-string

<ObjectLockConfiguration>
    <ObjectLockEnabled>Enabled</ObjectLockEnabled>
    <Rule>
        <DefaultRetention>
            <Mode>COMPLIANCE</Mode>
            <Years>6</Years>
        </DefaultRetention>
    </Rule>
</ObjectLockConfiguration>
```

버킷의 기본 보존 기간을 확인하는 방법

버킷에 S3 객체 잠금이 활성화되어 있는지 여부와 기본 보존 모드 및 보존 기간을 확인하려면 다음 방법 중 하나를 사용하십시오.

- 테넌트 관리자에서 버킷을 확인합니다. "S3 버킷 보기"을 참조하십시오.
- GetObjectLockConfiguration 요청을 실행합니다.

GetObjectLockConfiguration

GetObjectLockConfiguration 요청을 사용하면 버킷에 대해 S3 객체 잠금이 활성화되어 있는지 확인할 수 있으며, 활성화되어 있는 경우 버킷에 대해 기본 보존 모드 및 보존 기간이 구성되어 있는지 확인할 수 있습니다.

새로운 객체 버전이 버킷에 수집될 때, `x-amz-object-lock-mode`이 지정되지 않으면 기본 보존 모드가 적용됩니다. `x-amz-object-lock-retain-until-date`이 지정되지 않은 경우 기본 보존 기간이 보존 만료일 계산에 사용됩니다.

이 작업을 완료하려면 `s3:GetBucketObjectLockConfiguration` 권한이 있거나 루트 계정이어야 합니다.

요청 예시

```
GET /bucket?object-lock HTTP/1.1
Host: host
Accept-Encoding: identity
User-Agent: aws-cli/1.18.106 Python/3.8.2 Linux/4.4.0-18362-Microsoft
botocore/1.17.29
x-amz-date: date
x-amz-content-sha256: authorization-string
Authorization: authorization-string
```

응답 예시

```
HTTP/1.1 200 OK
x-amz-id-2:
iVmcB7OXXJRkRH1FiVq1151/T24gRfpwpuZrEG11Bb9ImOMAAe98oxSpXlknabA0LTvBYJpSIX
k=
x-amz-request-id: B34E94CACB2CEF6D
Date: Fri, 04 Sep 2020 22:47:09 GMT
Transfer-Encoding: chunked
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<ObjectLockConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <ObjectLockEnabled>Enabled</ObjectLockEnabled>
    <Rule>
        <DefaultRetention>
            <Mode>COMPLIANCE</Mode>
            <Years>6</Years>
        </DefaultRetention>
    </Rule>
</ObjectLockConfiguration>
```

객체에 대한 보존 설정을 지정하는 방법

S3 객체 잠금이 활성화된 버킷에는 S3 객체 잠금 보존 설정이 있는 객체와 없는 객체가 혼합되어 포함될 수 있습니다.

객체 수준의 보존 설정은 S3 REST API를 사용하여 지정합니다. 객체에 대한 보존 설정은 버킷의 기본 보존 설정을 재정의합니다.

각 객체에 대해 다음과 같은 설정을 지정할 수 있습니다.

- 보존 모드: COMPLIANCE 또는 GOVERNANCE 중 하나.
- 보존 기한: 개체 버전을 StorageGRID에서 얼마나 오래 보존해야 하는지를 지정하는 날짜입니다.
    - 규정 준수 모드에서 보존 기한이 미래 날짜로 설정된 경우, 해당 개체를 검색할 수는 있지만 수정하거나 삭제할 수는 없습니다. 보존 기한은 늘릴 수 있지만, 줄이거나 삭제할 수는 없습니다.

    - 거버넌스 모드에서 특별 권한을 가진 사용자는 보존 기간 설정을 무시할 수 있습니다. 보존 기간이 만료되기 전에 개체 버전을 삭제할 수 있습니다. 또한 보존 기간을 늘리거나 줄이거나 아예 제거할 수도 있습니다.

- 법적 보류: 객체 버전에 법적 보류를 적용하면 해당 객체가 즉시 잠깁니다. 예를 들어, 조사 또는 법적 분쟁과 관련된 객체에 법적 보류를 적용해야 할 수 있습니다. 법적 보류에는 만료일이 없으며 명시적으로 해제될 때까지 유지됩니다.

  객체에 대한 법적 보존 설정은 보존 모드 및 보존 만료일과 무관합니다. 객체 버전이 법적 보존 상태에 있는 경우 누구도 해당 버전을 삭제할 수 없습니다.

버킷에 객체 버전을 추가할 때 S3 Object Lock 설정을 지정하려면 "PutObject", "CopyObject", 또는 "CreateMultipartUpload" 요청을 실행하십시오.

다음을 사용할 수 있습니다.

- `x-amz-object-lock-mode`이는 COMPLIANCE 또는 GOVERNANCE일 수 있습니다(대소문자 구분).

  > 을(를) 지정하는 경우 `x-amz-object-lock-mode`, 도 지정해야 합니다 `x-amz-object-lock-retain-until-date`.

- `x-amz-object-lock-retain-until-date`
    - 보존 기한 값은 반드시 형식이어야 합니다 `2020-08-10T21:46:00Z`. 소수점 이하 초 단위까지 허용되지만, 소수점 이하 3자리까지만 보존됩니다(밀리초 정밀도). 다른 ISO 8601 형식은 허용되지 않습니다.
    - 보존 기한은 미래 날짜로 설정해야 합니다.
- `x-amz-object-lock-legal-hold`

  법적 보류(Legal Hold)가 켜짐(대소문자 구분)인 경우 해당 객체는 법적 보류 상태가 됩니다. 법적 보류가 꺼짐(Off)인 경우 법적 보류가 적용되지 않습니다. 그 외의 값은 400 Bad Request(InvalidArgument) 오류를 발생시킵니다.

이러한 요청 헤더를 사용하는 경우 다음 제한 사항에 유의하십시오.

- `Content-MD5` 요청 헤더는 PutObject 요청에 `x-amz-object-lock-*` 요청 헤더가 있는 경우 필수입니다. `Content-MD5`은 CopyObject 또는 CreateMultipartUpload에는 필요하지 않습니다.
- 버킷에 S3 객체 잠금이 활성화되어 있지 않고 `x-amz-object-lock-*`요청 헤더가 있는 경우 400 잘못된 요청(InvalidRequest) 오류가 반환됩니다.
- PutObject 요청은 AWS 동작과 일치하도록 `x-amz-storage-class: REDUCED_REDUNDANCY`을 사용할 수 있습니다. 그러나 S3 객체 잠금이 활성화된 버킷에 객체를 수집할 경우 StorageGRID는 항상 이중 커밋 수집을 수행합니다.
- 이후의 GET 또는 HeadObject 버전 응답에는 구성되어 있고 요청 발신자에게 올바른 `s3:Get*` 권한이 있는 경우 헤더 `x-amz-object-lock-mode`, `x-amz-object-lock-retain-until-date` 및 `x-amz-object-lock-legal-hold`이 포함됩니다.

개체에 허용되는 최소 및 최대 보존 기간을 제한하려면 `s3:object-lock-remaining-retention-days` 정책 조건 키를 사용할 수 있습니다.

객체의 보존 설정을 업데이트하는 방법

기존 개체 버전의 법적 보존 또는 보존 기간 설정을 업데이트해야 하는 경우 다음 개체 하위 리소스 작업을 수행할 수 있습니다.

- `PutObjectLegalHold`

  새로운 법적 보류 값이 ON으로 설정되면 해당 개체는 법적 보류 상태가 됩니다. 법적 보류 값이 OFF로 설정되면 법적 보류가 해제됩니다.

- `PutObjectRetention`
  - 모드 값은 COMPLIANCE 또는 GOVERNANCE일 수 있습니다(대소문자 구분).
  - 보존 기한 값은 반드시 형식이어야 합니다 `2020-08-10T21:46:00Z`. 소수점 이하 초 단위까지 허용되지만, 소수점 이하 3자리까지만 보존됩니다(밀리초 정밀도). 다른 ISO 8601 형식은 허용되지 않습니다.
  - 객체 버전에 기존 보존 날짜가 있는 경우 해당 날짜를 늘릴 수만 있습니다. 새 값은 미래 날짜여야 합니다.

거버넌스 모드 사용 방법

`s3:BypassGovernanceRetention`권한이 있는 사용자는 `GOVERNANCE` 모드를 사용하는 객체의 활성 보존 설정을 무시할 수 있습니다. 모든 `DELETE` 또는 `PutObjectRetention` 작업에는 `x-amz-bypass-governance-retention:true`요청 헤더가 포함되어야 합니다. 이러한 사용자는 다음과 같은 추가 작업을 수행할 수 있습니다.

- DeleteObject 또는 DeleteObjects 작업을 수행하여 보존 기간이 경과하기 전에 개체 버전을 삭제합니다.

  법적 보존 조치가 적용된 개체는 삭제할 수 없습니다. 법적 보존 조치를 해제해야 합니다.

- PutObjectRetention 작업을 수행하여 객체의 보존 기간이 만료되기 전에 객체 버전의 모드를 GOVERNANCE에서 COMPLIANCE로 변경합니다.

  규정 준수 모드에서 거버넌스 모드로 변경하는 것은 절대 허용되지 않습니다.

- PutObjectRetention 작업을 수행하여 객체 버전의 보존 기간을 늘리거나, 줄이거나, 제거합니다.

관련 정보

- "S3 Object Lock으로 객체 관리"
- "S3 객체 잠금을 사용하여 객체 보존"
- "Amazon Simple Storage Service 사용자 가이드: 객체 잠금"

StorageGRID의 S3 라이프사이클 구성에 대해 알아보십시오

StorageGRID 시스템에서 특정 객체가 삭제되는 시점을 제어하기 위해 S3 수명 주기 구성을 생성할 수 있습니다.

이 섹션의 간단한 예제는 S3 수명 주기 구성을 통해 특정 S3 버킷에서 특정 객체가 삭제(만료)되는 시점을 제어하는 방법을 보여줍니다. 이 섹션의 예제는 설명 목적으로만 제공됩니다. S3 수명 주기 구성 생성에 대한 자세한 내용은

"Amazon Simple Storage Service 사용자 가이드: 객체 수명 주기 관리"을 참조하십시오. StorageGRID는 만료 작업만 지원하며 전환 작업은 지원하지 않습니다.

라이프사이클 구성이란 무엇인가요?

수명 주기 구성은 특정 S3 버킷의 객체에 적용되는 규칙 집합입니다. 각 규칙은 어떤 객체가 영향을 받는지, 그리고 해당 객체가 언제 만료되는지(특정 날짜 또는 일정 기간 후)를 지정합니다.

StorageGRID는 라이프사이클 구성에서 최대 1,000개의 라이프사이클 규칙을 지원합니다. 각 규칙에는 다음 XML 요소가 포함될 수 있습니다.

- 만료: 지정된 날짜가 되거나 객체가 수집된 시점으로부터 지정된 일수가 경과하면 객체를 삭제합니다.
- NoncurrentVersionExpiration: 개체가 비활성 상태가 된 시점부터 지정된 일수가 경과하면 개체를 삭제합니다.
- 필터(접두사, 태그)
- 상태
- ID

각 객체는 S3 버킷 수명 주기 또는 ILM 정책의 보존 설정을 따릅니다. S3 버킷 수명 주기가 구성된 경우, 버킷 수명 주기 필터와 일치하는 객체에 대해서는 수명 주기 만료 작업이 ILM 정책을 재정의합니다. 버킷 수명 주기 필터와 일치하지 않는 객체는 ILM 정책의 보존 설정을 사용합니다. 객체가 버킷 수명 주기 필터와 일치하고 만료 작업이 명시적으로 지정되지 않은 경우, ILM 정책의 보존 설정은 사용되지 않으며 객체 버전은 영구적으로 보존되는 것으로 간주됩니다. "S3 버킷 수명 주기 및 ILM 정책에 대한 우선순위 예시"을 참조하십시오.

결과적으로, ILM 규칙의 배치 지침이 해당 객체에 여전히 적용되더라도 객체가 그리드에서 제거될 수 있습니다. 또는, 객체에 대한 ILM 배치 지침의 유효 기간이 만료된 후에도 객체가 그리드에 남아 있을 수 있습니다. 자세한 내용은 "ILM이 객체의 수명 주기 전반에 걸쳐 작동하는 방식"을 참조하십시오.

> 버킷 수명 주기 구성은 S3 Object Lock이 활성화된 버킷에서 사용할 수 있지만, 버킷 수명 주기 구성은 레거시 규정 준수 버킷에서는 지원되지 않습니다.

StorageGRID는 수명 주기 구성을 관리하기 위해 다음 버킷 작업의 사용을 지원합니다.

- DeleteBucketLifecycle
- GetBucketLifecycleConfiguration
- PutBucketLifecycleConfiguration

라이프사이클 구성 생성

라이프사이클 구성을 만드는 첫 번째 단계로, 하나 이상의 규칙이 포함된 JSON 파일을 만듭니다. 예를 들어, 이 JSON 파일에는 다음과 같이 세 가지 규칙이 포함되어 있습니다.

1. 규칙 1은 접두사 `category1`/와 일치하고 `key2`값이 `tag2`인 객체에만 적용됩니다. `Expiration`매개변수는 필터와 일치하는 객체가 2020년 8월 22일 자정에 만료됨을 지정합니다.
2. 규칙 2는 접두사 `category2`/와 일치하는 객체에만 적용됩니다. 해당 `Expiration` 매개변수는 필터와 일치하는 객체가 수집된 후 100일 후에 만료되도록 지정합니다.

일수를 지정하는 규칙은 객체가 수집된 날짜를 기준으로 합니다. 현재 날짜가 수집 날짜에 지정된 일수를 더한 날짜를 초과하면 수명 주기 구성이 적용되는 즉시 일부 객체가 버킷에서 제거될 수 있습니다.

3. 규칙 3은 접두사 `category3/`와 일치하는 객체에만 적용됩니다. `Expiration` 매개변수는 일치하는 객체의 비현재 버전이 비현재 버전이 된 후 50일 후에 만료되도록 지정합니다.

```
{
    "Rules": [
        {
            "ID": "rule1",
            "Filter": {
                "And": {
                    "Prefix": "category1/",
                    "Tags": [
                        {
                            "Key": "key2",
                            "Value": "tag2"
                        }
                    ]
                }
            },
            "Expiration": {
                "Date": "2020-08-22T00:00:00Z"
            },
            "Status": "Enabled"
        },
        {
            "ID": "rule2",
            "Filter": {
                "Prefix": "category2/"
            },
            "Expiration": {
                "Days": 100
            },
            "Status": "Enabled"
        },
        {
            "ID": "rule3",
            "Filter": {
                "Prefix": "category3/"
            },
            "NoncurrentVersionExpiration": {
                "NoncurrentDays": 50
            },
            "Status": "Enabled"
        }
    ]
}
```

버킷에 수명 주기 구성 적용

라이프사이클 구성 파일을 생성한 후 PutBucketLifecycleConfiguration 요청을 실행하여 해당 파일을 버킷에 적용합니다.

이 요청은 예제 파일의 수명 주기 구성을 `testbucket`라는 이름의 버킷에 있는 객체에 적용합니다.

```
aws s3api --endpoint-url <StorageGRID endpoint> put-bucket-lifecycle-
configuration
--bucket testbucket --lifecycle-configuration file://bktjson.json
```

버킷에 수명 주기 구성이 성공적으로 적용되었는지 확인하려면 GetBucketLifecycleConfiguration 요청을 실행하십시오. 예를 들면 다음과 같습니다.

```
aws s3api --endpoint-url <StorageGRID endpoint> get-bucket-lifecycle-
configuration
 --bucket testbucket
```

성공적인 응답에는 방금 적용한 라이프사이클 구성이 나열됩니다.

버킷 수명 주기 만료가 객체에 적용되는지 확인합니다.

PutObject, HeadObject 또는 GetObject 요청을 실행할 때 수명 주기 구성의 만료 규칙이 특정 개체에 적용되는지 여부를 확인할 수 있습니다. 규칙이 적용되는 경우 응답에는 `Expiration` 개체가 만료되는 시점과 일치하는 만료 규칙을 나타내는 매개변수가 포함됩니다.

> 버킷 수명 주기가 ILM을 재정의하므로 `expiry-date` 표시되는 날짜는 객체가 실제로 삭제되는 날짜입니다. 자세한 내용은 "오브젝트 보존 기간 결정 방법"을 참조하십시오.

예를 들어, 이 PutObject 요청은 2020년 6월 22일에 발행되었으며 `testbucket` 버킷에 객체를 배치합니다.

```
aws s3api --endpoint-url <StorageGRID endpoint> put-object
--bucket testbucket --key obj2test2 --body bktjson.json
```

성공 응답은 해당 객체가 100일 후(2020년 10월 1일)에 만료되며 수명 주기 구성의 규칙 2와 일치함을 나타냅니다.

```
{
      *"Expiration": "expiry-date=\"Thu, 01 Oct 2020 09:07:49 GMT\", rule-
id=\"rule2\"",
      "ETag": "\"9762f8a803bc34f5340579d4446076f7\""
}
```

예를 들어, 이 HeadObject 요청은 testbucket 버킷에 있는 동일한 객체의 메타데이터를 가져오는 데 사용되었습니다.

```
aws s3api --endpoint-url <StorageGRID endpoint> head-object
--bucket testbucket --key obj2test2
```

성공 응답에는 객체의 메타데이터가 포함되며 해당 객체가 100일 후에 만료되고 규칙 2를 충족했음을 나타냅니다.

```
{
      "AcceptRanges": "bytes",
      *"Expiration": "expiry-date=\"Thu, 01 Oct 2020 09:07:48 GMT\", rule-
id=\"rule2\"",
      "LastModified": "2020-06-23T09:07:48+00:00",
      "ContentLength": 921,
      "ETag": "\"9762f8a803bc34f5340579d4446076f7\""
      "ContentType": "binary/octet-stream",
      "Metadata": {}
}
```

버전 관리가 활성화된 버킷의 경우 `x-amz-expiration` 응답 헤더는 객체의 현재 버전에만 적용됩니다.

StorageGRID로 S3 REST API 구현 권장 사항

StorageGRID와 함께 사용하기 위해 S3 REST API를 구현할 때 다음 권장 사항을 따라야 합니다.

존재하지 않는 객체에 대한 HEAD 권장 사항

애플리케이션에서 객체가 실제로 존재하지 않을 것으로 예상되는 경로에 객체가 있는지 정기적으로 확인하는 경우 "Available" "일관성"을 사용해야 합니다. 예를 들어, 애플리케이션에서 PUT 작업을 수행하기 전에 해당 위치에 HEAD 요청을 보내는 경우 "Available" 일관성을 사용해야 합니다.

그렇지 않으면 HEAD 작업에서 객체를 찾지 못할 경우, 동일 사이트의 두 개 이상의 스토리지 노드를 사용할 수 없거나 원격 사이트에 연결할 수 없는 경우 500 내부 서버 오류가 많이 발생할 수 있습니다.

"PUT 버킷 일관성" 요청을 사용하여 각 버킷에 대한 "사용 가능" 일관성을 설정하거나 개별 API 작업의 요청 헤더에 일관성을 지정할 수 있습니다.

객체 키에 대한 권장 사항

버킷이 처음 생성된 시점을 기준으로 객체 키 이름 지정에 대한 다음 권장 사항을 따르십시오.

StorageGRID 11.4 이하 버전에서 생성된 버킷

- 객체 키의 처음 네 문자에 임의의 값을 사용하지 마십시오. 이는 이전 AWS의 키 접두사 권장 사항과 상반됩니다. 대신, `image`과 같이 임의적이지 않고 고유하지 않은 접두사를 사용하십시오.
- AWS의 이전 권장 사항에 따라 키 접두사에 임의적이고 고유한 문자를 사용하려면 객체 키 앞에 디렉터리 이름을

접두사로 붙이십시오. 즉, 다음 형식을 사용하십시오.

```
mybucket/mydir/f8e3-image3132.jpg
```

이 형식 대신에:

```
mybucket/f8e3-image3132.jpg
```

StorageGRID 11.4 이상에서 생성된 버킷

성능 최적화를 위해 객체 키 이름을 제한할 필요는 없습니다. 대부분의 경우 객체 키 이름의 처음 네 문자에 임의의 값을 사용할 수 있습니다.

> 단, S3 워크로드가 짧은 시간 후 모든 객체를 지속적으로 삭제하는 경우는 예외입니다. 이러한 사용 사례에서 성능 영향을 최소화하려면 수천 개의 객체마다 키 이름의 앞부분을 날짜와 같은 것으로 변경하는 것이 좋습니다. 예를 들어, S3 클라이언트가 일반적으로 초당 2,000개의 객체를 기록하고 ILM 또는 버킷 수명 주기 정책에 따라 3일 후에 모든 객체가 삭제된다고 가정해 보겠습니다. 성능 영향을 최소화하려면 다음과 같은 패턴을 사용하여 키 이름을 지정할 수 있습니다.
> `/mybucket/mydir/yyyymmddhhmmss-random_UUID.jpg`

"범위 읽기"에 대한 권장 사항

이 "저장된 객체를 압축하는 전역 옵션"옵션이 활성화된 경우, S3 클라이언트 애플리케이션은 반환할 바이트 범위를 지정하는 GetObject 작업을 수행하지 않아야 합니다. 이러한 "범위 읽기" 작업은 StorageGRID가 요청된 바이트에 액세스하기 위해 객체를 사실상 압축 해제해야 하므로 비효율적입니다. GetObject 작업 중 매우 큰 객체에서 작은 바이트 범위를 요청하는 경우는 특히 비효율적입니다. 예를 들어, 50GB로 압축된 객체에서 10MB 범위를 읽는 것은 비효율적입니다.

압축된 객체에서 범위를 읽어오는 경우 클라이언트 요청 시간이 초과될 수 있습니다.

> 객체를 압축해야 하고 클라이언트 애플리케이션에서 범위 읽기를 사용해야 하는 경우 애플리케이션의 읽기 시간 제한을 늘리십시오.