ILM 및 객체 수명 주기

StorageGRID software

NetApp
July 23, 2026

목차

ILM 및 객체 수명 주기

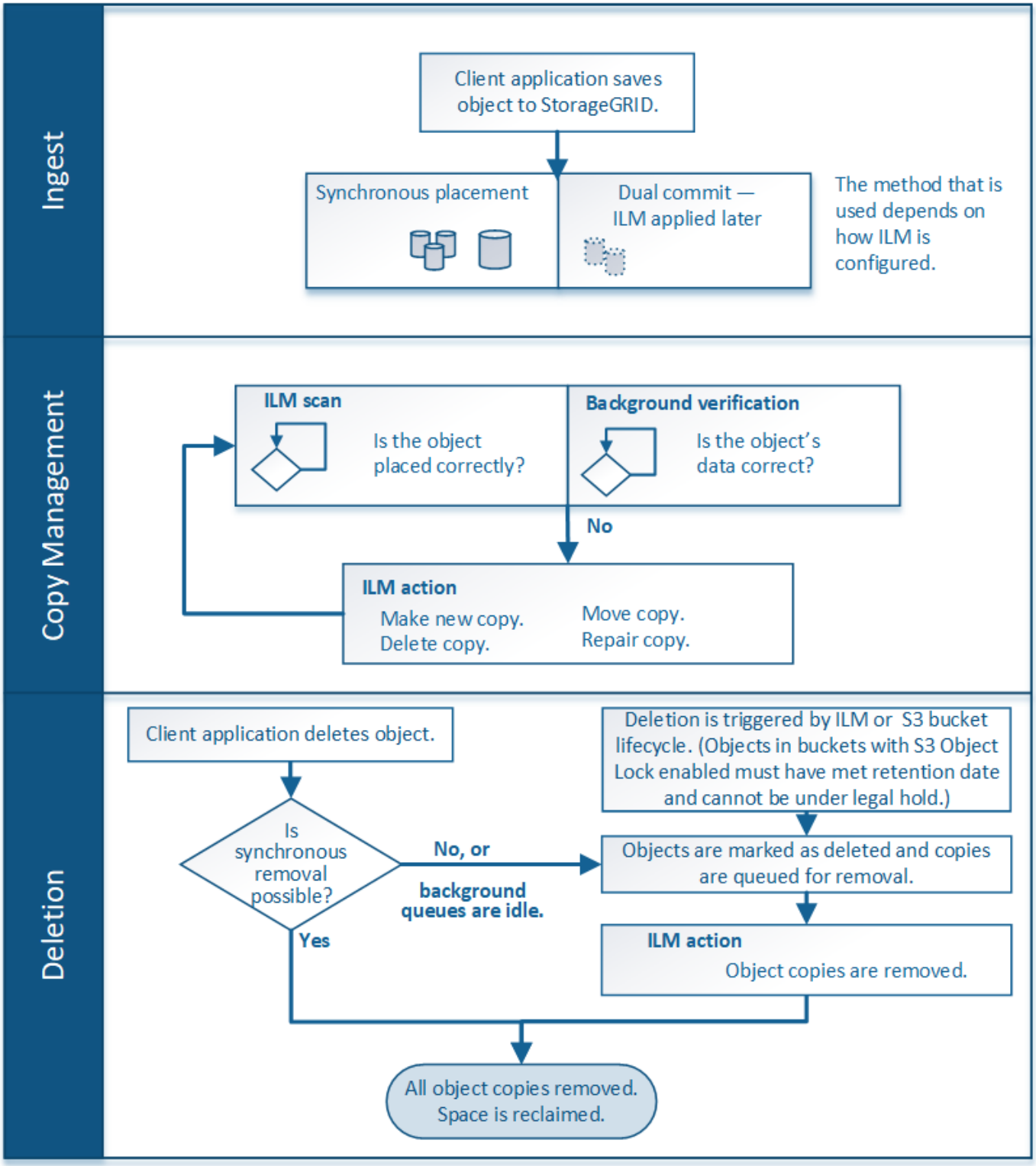
ILM이 StorageGRID에서 오브젝트 수명 전반에 걸쳐 작동하는 방식

StorageGRID가 객체의 수명 주기 각 단계에서 ILM을 사용하여 객체를 관리하는 방식을 이해하면 보다 효과적인 정책을 설계하는 데 도움이 될 수 있습니다.

- 데이터 수집: 데이터 수집은 S3 클라이언트 애플리케이션이 StorageGRID 시스템에 객체를 저장하기 위해 연결을 설정할 때 시작되며, StorageGRID가 클라이언트에 "수집 성공" 메시지를 반환할 때 완료됩니다. 객체 데이터는 수집 중에 ILM 요구 사항 지정 방식에 따라 ILM 명령을 즉시 적용하는 동기식 배치 방식 또는 임시 복사본을 생성하고 나중에 ILM을 적용하는 이중 커밋 방식으로 보호됩니다.
- 복사본 관리: ILM의 배치 지침에 지정된 수량 및 유형의 객체 복사본을 생성한 후, StorageGRID는 객체 위치를 관리하고 객체 손실을 방지합니다.
  - **ILM** 스캔 및 평가: StorageGRID는 그리드에 저장된 객체 목록을 지속적으로 스캔하여 현재 복사본이 ILM 요구 사항을 충족하는지 확인합니다. 객체 복사본의 유형, 개수 또는 위치가 변경되어야 하는 경우 StorageGRID는 필요에 따라 복사본을 생성, 삭제 또는 이동합니다.
  - 백그라운드 검증: StorageGRID는 객체 데이터의 무결성을 확인하기 위해 지속적으로 백그라운드 검증을 수행합니다. 문제가 발견되면 StorageGRID는 현재 ILM 요구 사항을 충족하는 위치에 새 객체 복사본 또는 대체 이레이저 코딩 객체 조각을 자동으로 생성합니다. "객체 무결성 확인"을 참조하십시오.
- 객체 삭제: 객체 관리는 StorageGRID 시스템에서 모든 복사본이 제거되면 종료됩니다. 객체는 클라이언트의 삭제 요청, ILM에 의한 삭제 또는 S3 버킷 수명 주기 만료로 인해 삭제될 수 있습니다.

> ⓘ S3 객체 잠금이 활성화된 버킷의 객체는 법적 보존 조치가 적용 중이거나 보존 기한이 지정되었지만 아직 도래하지 않은 경우 삭제할 수 없습니다.

이 다이어그램은 ILM이 객체의 수명 주기 전반에 걸쳐 어떻게 작동하는지 요약하여 보여줍니다.



오브젝트 수집 방법

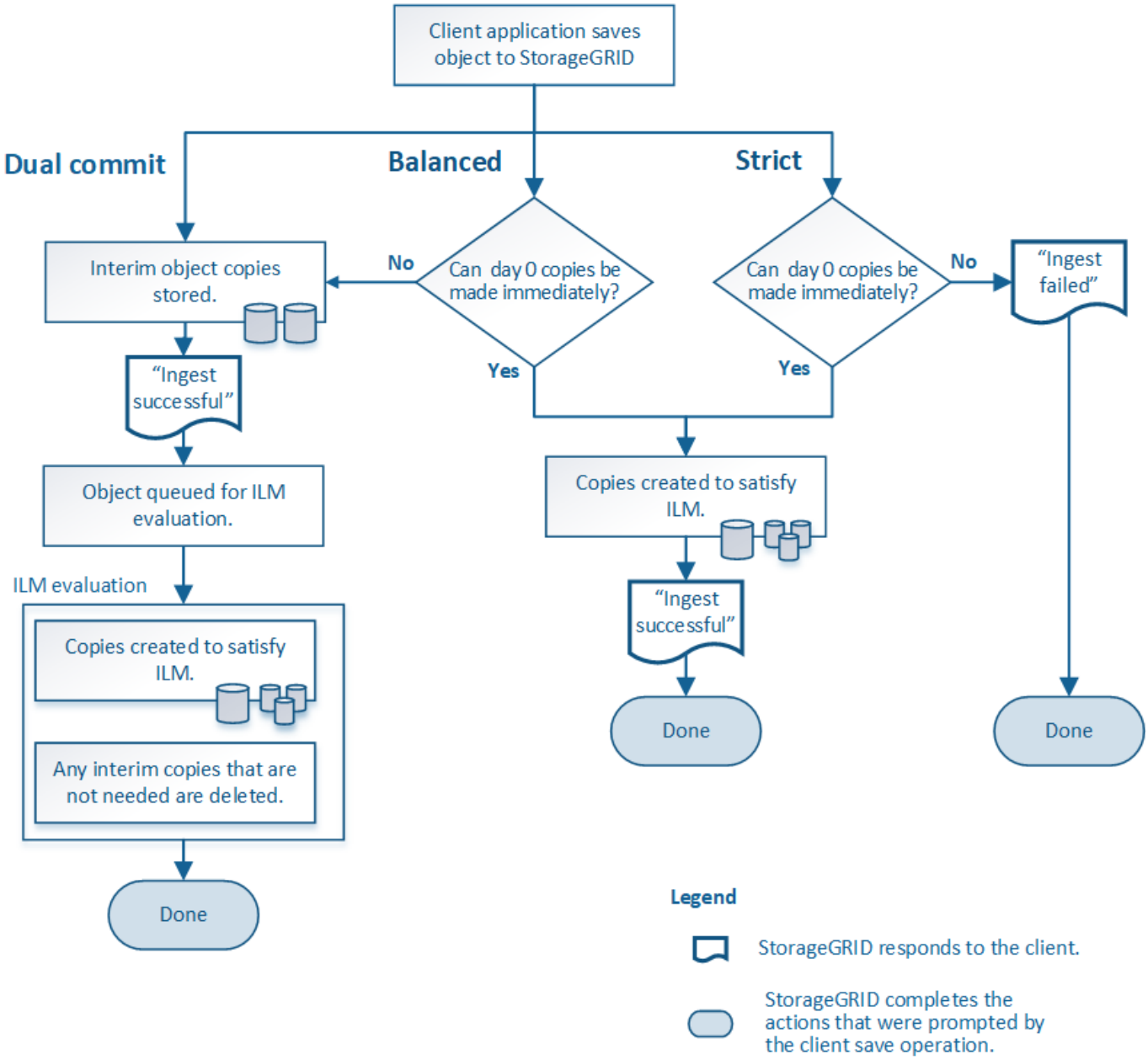
StorageGRID의 ILM 수집 옵션

ILM 규칙을 생성할 때 수집 시 객체를 보호하는 세 가지 옵션(이중 커밋, 엄격, 균형) 중 하나를 지정합니다.

사용자의 선택에 따라 StorageGRID는 임시 복사본을 만들고 나중에 ILM 평가를 위해 객체를 대기열에 추가하거나, 동기식 배치를 사용하여 ILM 요구 사항을 충족하기 위해 즉시 복사본을 만듭니다.

데이터 수집 옵션 흐름도

이 순서도는 세 가지 수집 옵션 각각을 사용하는 ILM 규칙에 의해 객체가 일치될 때 발생하는 상황을 보여줍니다.



이중 커밋

이중 커밋 옵션을 선택하면 StorageGRID가 즉시 서로 다른 두 개의 스토리지 노드에 객체의 임시 복사본을 생성하고 클라이언트에 "수집 성공" 메시지를 반환합니다. 객체는 ILM 평가를 위해 대기열에 추가되며, 규칙의 배치 지침을 충족하는 복사본은 나중에 생성됩니다. 이중 커밋 직후 ILM 정책을 즉시 처리할 수 없는 경우 사이트 손실 보호가 완료되는 데 시간이 걸릴 수 있습니다.

다음 두 경우 중 하나에 해당하면 이중 커밋 옵션을 사용하십시오.

- 멀티사이트 ILM 규칙을 사용하고 있으며 클라이언트 수집 지연 시간이 주요 고려 사항입니다. 이중 커밋을 사용하는 경우, 그리드가 ILM 요구 사항을 충족하지 못할 경우 이중 커밋 복사본을 생성하고 제거하는 추가 작업을 수행할 수 있도록 해야 합니다. 구체적으로 다음과 같습니다.
  - ILM 백로그를 방지하려면 그리드의 부하가 충분히 낮아야 합니다.
  - 그리드에는 충분한 하드웨어 리소스(IOPS, CPU, 메모리, 네트워크 대역폭 등)가 있어야 합니다.
- 다중 사이트 ILM 규칙을 사용하고 있으며 사이트 간 WAN 연결의 지연 시간이 길거나 대역폭이 제한된 경우가 많습니다. 이 시나리오에서는 이중 커밋 옵션을 사용하면 클라이언트 시간 초과를 방지하는 데 도움이 될 수 있습니다. 이중 커밋 옵션을 선택하기 전에 실제 워크로드를 사용하여 클라이언트 애플리케이션을 테스트해야 합니다.

균형(기본값)

균형 옵션을 선택하면 StorageGRID는 데이터 수집 시 동기식 배치를 사용하여 규칙의 배치 지침에 지정된 모든 복사본을 즉시 생성합니다. 엄격 옵션과 달리 StorageGRID가 모든 복사본을 즉시 생성할 수 없는 경우 이중 커밋을 사용합니다. ILM 정책에서 여러 사이트에 배치를 사용하고 즉각적인 사이트 손실 보호가 불가능한 경우 **ILM** 배치 불가능 경고가 발생합니다.

데이터 보호, 그리드 성능 및 데이터 수집 성공률을 최적으로 조합하려면 균형 옵션을 사용하십시오. 균형 옵션은 ILM 규칙 생성 마법사의 기본 옵션입니다.

엄격한

'엄격' 옵션을 선택하면 StorageGRID는 수집 시 동기 배치를 사용하여 규칙의 배치 지침에 지정된 모든 객체 복사본을 즉시 생성합니다. 필요한 스토리지 위치를 일시적으로 사용할 수 없는 등 StorageGRID가 모든 복사본을 생성할 수 없는 경우 수집이 실패합니다. 클라이언트는 작업을 다시 시도해야 합니다.

ILM 규칙에 명시된 위치에만 객체를 즉시 저장해야 하는 운영 또는 규제 요건이 있는 경우 Strict 옵션을 사용하십시오. 예를 들어, 규제 요건을 충족하기 위해 Strict 옵션과 Location Constraint 고급 필터를 사용하여 특정 데이터 센터에 객체가 저장되지 않도록 해야 할 수 있습니다.

"예시 5: 엄격한 수집 동작에 대한 ILM 규칙 및 정책"을 참조하십시오.

StorageGRID의 ILM 수집 옵션의 장점, 단점 및 제한 사항

데이터 수집 시 데이터 보호를 위한 세 가지 옵션(균형, 엄격 또는 이중 커밋) 각각의 장단점을 이해하면 ILM 규칙에 어떤 옵션을 선택할지 결정하는 데 도움이 될 수 있습니다.

수집 옵션에 대한 개요는 "수집 옵션"을 참조하십시오.

균형형과 엄격형 옵션의 장점

수집 중에 임시 복사본을 생성하는 Dual commit과 비교했을 때, 두 가지 동기식 배치 옵션은 다음과 같은 이점을 제공할 수 있습니다.

- 향상된 데이터 보안: 객체 데이터는 ILM 규칙의 배치 지침에 따라 즉시 보호되며, 이 지침은 둘 이상의 스토리지 위치 장애를 포함한 다양한 장애 상황에 대비하도록 구성할 수 있습니다. 이중 커밋은 단일 로컬 복사본 손실에 대해서만 보호할 수 있습니다.
- 더욱 효율적인 그리드 운영: 각 객체는 수집되는 즉시 한 번만 처리됩니다. StorageGRID 시스템은 중간 복사본을 추적하거나 삭제할 필요가 없으므로 처리 부하가 줄어들고 데이터베이스 공간 사용량도 감소합니다.

- **(균형) 권장**: 균형 옵션은 최적의 ILM 효율성을 제공합니다. 엄격한 수집 동작이 필요하거나 그리드가 이중 커밋 사용 기준을 모두 충족하는 경우가 아니면 균형 옵션을 사용하는 것이 좋습니다.
- **(Strict) 객체 위치 확정**: Strict 옵션을 선택하면 객체가 ILM 규칙의 배치 지침에 따라 즉시 저장됩니다.

균형형과 엄격형 옵션의 단점

듀얼 커밋과 비교했을 때, 밸런스드 및 스트릭트 옵션에는 몇 가지 단점이 있습니다.

- 클라이언트 수집 시간 증가: 클라이언트 수집 지연 시간이 길어질 수 있습니다. 균형(Balanced) 또는 엄격(Strict) 옵션을 사용하는 경우, 모든 이레이저 코딩된 조각 또는 복제본이 생성 및 저장될 때까지 "수집 성공" 메시지가 클라이언트로 반환되지 않습니다. 하지만 객체 데이터는 최종 저장 위치에 훨씬 빠르게 도달할 가능성이 높습니다.
- **(엄격) 수집 실패율 증가**: '엄격' 옵션을 선택하면 StorageGRID ILM 규칙에 지정된 모든 복사본을 즉시 생성할 수 없는 경우 수집이 실패합니다. 필수 스토리지 위치가 일시적으로 오프라인 상태이거나 네트워크 문제로 인해 사이트 간 객체 복사가 지연되는 경우 수집 실패율이 높아질 수 있습니다.
- **(엄격한) S3** 멀티파트 업로드 배치는 일부 상황에서 예상과 다를 수 있습니다: 엄격한 모드에서는 객체가 ILM 규칙에 따라 배치되거나 수집이 실패할 것으로 예상됩니다. 그러나 S3 멀티파트 업로드의 경우, ILM은 객체가 수집될 때 각 파트에 대해 평가되고, 멀티파트 업로드가 완료되면 객체 전체에 대해 평가됩니다. 다음과 같은 상황에서는 예상과 다른 배치가 발생할 수 있습니다:
  - **S3** 멀티파트 업로드 진행 중 **ILM**이 변경되는 경우: 각 파트는 해당 파트가 수집될 당시 활성화된 규칙에 따라 배치되므로, 멀티파트 업로드가 완료될 때 객체의 일부 파트가 현재 ILM 요구 사항을 충족하지 못할 수 있습니다. 이러한 경우 객체 수집은 실패하지 않습니다. 대신, 올바르게 배치되지 않은 파트는 ILM 재평가를 위해 대기열에 추가되고 나중에 올바른 위치로 이동됩니다.
  - **ILM** 규칙이 크기를 기준으로 필터링하는 경우: 파트에 대한 ILM을 평가할 때 StorageGRID는 객체 전체의 크기가 아닌 파트의 크기를 기준으로 필터링합니다. 즉, 객체의 일부는 객체 전체에 대한 ILM 요구 사항을 충족하지 않는 위치에 저장될 수 있습니다. 예를 들어, 규칙에서 10GB 이상의 모든 객체는 DC1에 저장하고 10GB 미만의 모든 객체는 DC2에 저장하도록 지정한 경우, 10개 파트로 구성된 멀티파트 업로드의 각 1GB 파트는 수집 시 DC2에 저장됩니다. 객체에 대한 ILM이 평가되면 객체의 모든 파트가 DC1으로 이동됩니다.
- **(엄격 모드)** 객체 태그 또는 메타데이터가 업데이트되어 새로 필요한 배치 위치를 지정할 수 없는 경우에도 수집이 실패하지 않습니다.: 엄격 모드에서는 객체가 ILM 규칙에 따라 배치되거나 수집이 실패해야 합니다. 그러나 그리드에 이미 저장된 객체의 메타데이터 또는 태그를 업데이트하면 해당 객체는 다시 수집되지 않습니다. 즉, 업데이트로 인해 발생하는 객체 배치 변경 사항이 즉시 적용되지 않습니다. 배치 변경은 일반적인 백그라운드 ILM 프로세스에 의해 ILM이 다시 평가될 때 이루어집니다. 필요한 배치 변경을 수행할 수 없는 경우(예: 새로 필요한 위치를 사용할 수 없는 경우), 업데이트된 객체는 배치 변경이 가능해질 때까지 현재 배치 위치를 유지합니다.

균형 및 엄격 옵션 사용 시 객체 배치 제한 사항

균형 또는 엄격 옵션은 다음과 같은 배치 지침이 포함된 ILM 규칙에는 사용할 수 없습니다.

- 0일차에 클라우드 스토리지 풀에 배치됩니다.
- 규칙의 참조 시간이 사용자 정의 생성 시간으로 설정된 경우 클라우드 스토리지 풀에 배치됩니다.

이러한 제한 사항은 StorageGRID가 클라우드 스토리지 풀에 동기적으로 복사본을 만들 수 없으며, 사용자가 정의한 생성 시간이 현재 시간으로 해석될 수 있기 때문에 발생합니다.

ILM 규칙과 일관성이 데이터 보호에 미치는 영향

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

예를 들어, ILM 규칙에 대해 선택된 수집 동작은 객체 복사본의 초기 배치에 영향을 미치고, 객체 저장 시 사용되는

일관성 수준은 객체 메타데이터의 초기 배치에 영향을 미칩니다. StorageGRID는 클라이언트 요청을 처리하기 위해 객체의 데이터와 메타데이터 모두에 대한 액세스가 필요하므로, 일관성 수준과 수집 동작에 대해 동일한 수준의 보호 방식을 선택하면 초기 데이터 보호가 강화되고 시스템 응답이 더욱 예측 가능해집니다.

다음은 StorageGRID에서 사용 가능한 일관성 값에 대한 간략한 요약입니다.

- 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.

> 일관성 값을 선택하기 전에, "일관성에 대한 전체 설명을 읽어보세요.". 기본값을 변경하기 전에 이점과 한계를 이해해야 합니다.

일관성과 **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.

관련 정보

"예시 5: 엄격한 수집 동작에 대한 ILM 규칙 및 정책"

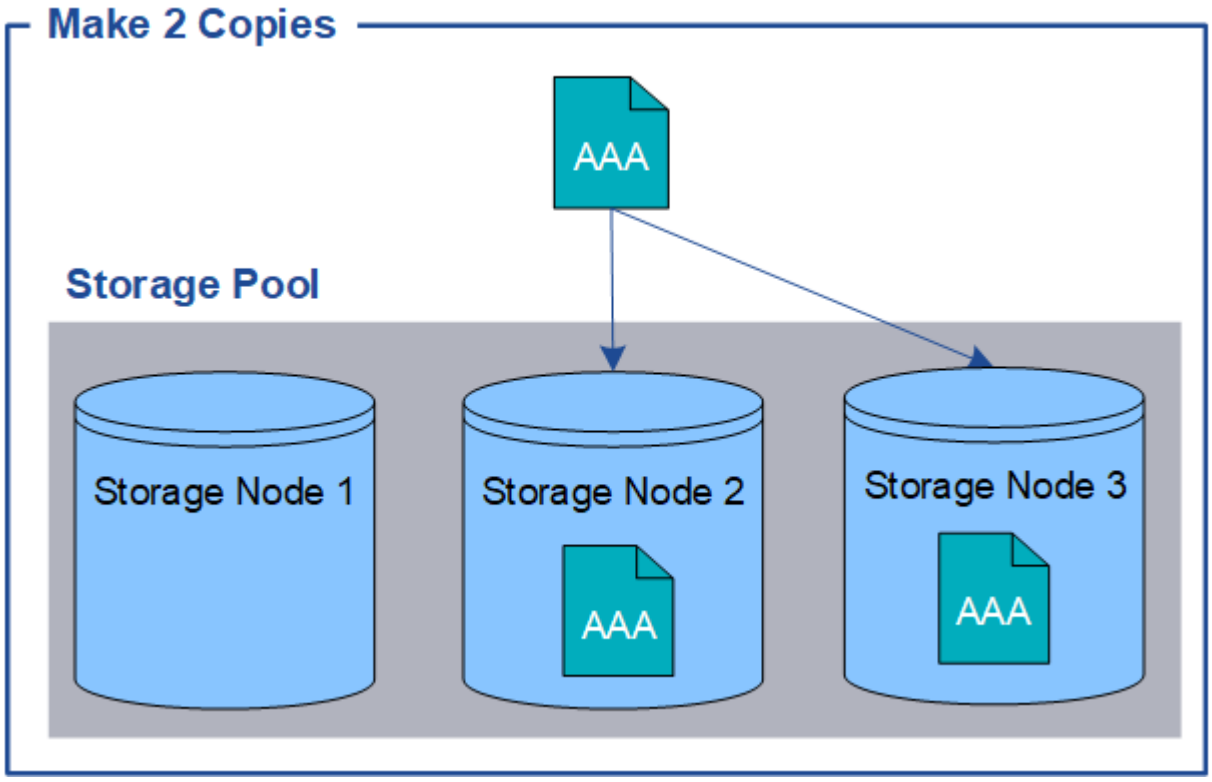
객체 저장 방식(복제 또는 소거 코딩)

StorageGRID의 복제에 대해 알아보십시오

복제는 StorageGRID가 객체 데이터를 저장하는 두 가지 방법 중 하나입니다(다른 하나는 이레이저 코딩). 객체가 복제를 사용하는 ILM 규칙과 일치하면 시스템은 객체 데이터의 정확한 복사본을 생성하여 스토리지 노드에 저장합니다.

복제본을 생성하도록 ILM 규칙을 구성할 때 생성할 복제본의 수, 복제본의 위치, 그리고 각 위치에 복제본을 보관할 기간을 지정합니다.

다음 예에서 ILM 규칙은 각 객체의 복제본 두 개를 세 개의 스토리지 노드가 포함된 스토리지 풀에 배치하도록 지정합니다.



StorageGRID가 이 규칙에 따라 객체를 찾으면 해당 객체의 복사본을 두 개 생성하여 스토리지 풀의 서로 다른 스토리지

노드에 각 복사본을 배치합니다. 두 개의 복사본은 사용 가능한 세 개의 스토리지 노드 중 임의의 두 개에 배치될 수 있습니다. 이 경우 규칙에 따라 객체 복사본이 스토리지 노드 2와 3에 배치되었습니다. 두 개의 복사본이 있으므로 스토리지 풀의 노드 중 하나에 장애가 발생하더라도 객체를 검색할 수 있습니다.

StorageGRID는 특정 스토리지 노드에 객체의 복제본을 하나만 저장할 수 있습니다. 그리드에 스토리지 노드가 세 개 있고 4개 복사본 ILM 규칙을 생성하면 스토리지 노드당 하나씩, 총 세 개의 복사본만 생성됩니다. ILM 규칙을 완전히 적용할 수 없음을 나타내기 위해 **ILM** 배치 불가 경고가 트리거됩니다.

관련 정보

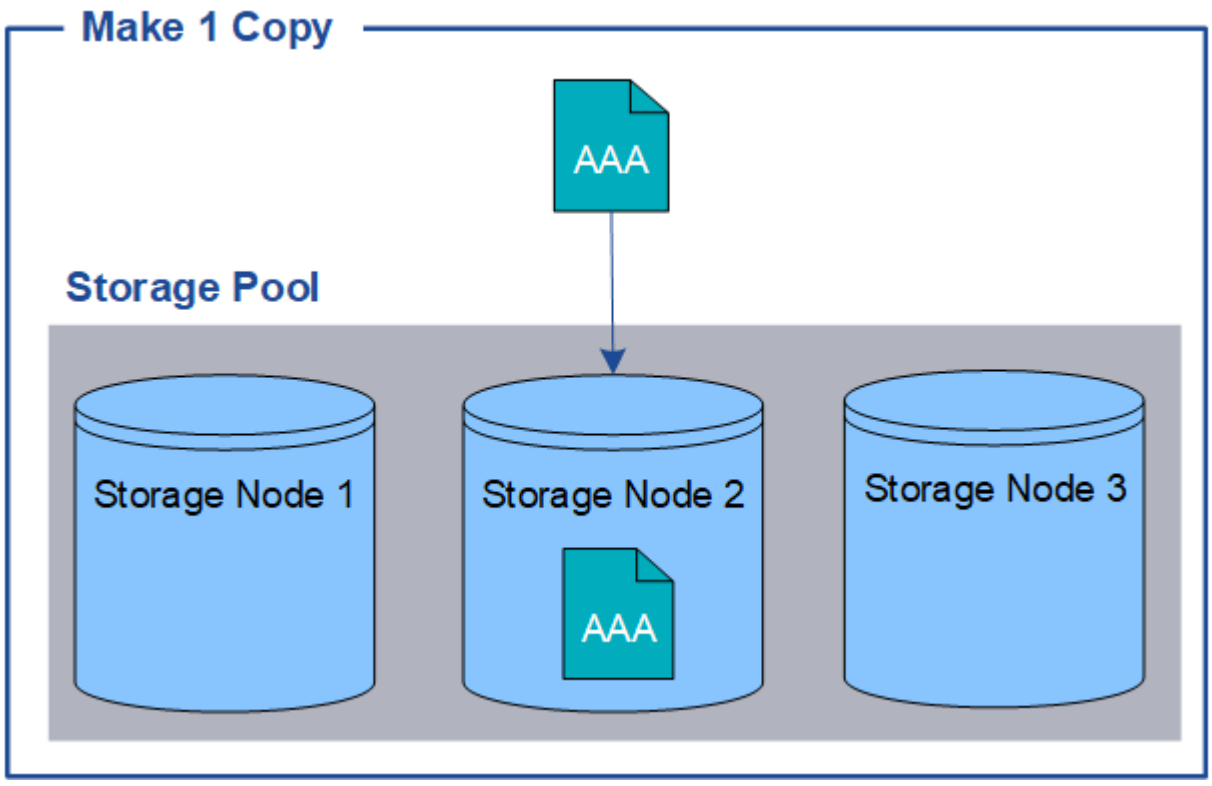
- "이레이저 코딩이란 무엇입니까"
- "스토리지 풀이란 무엇인가"
- "복제 및 이레이저 코딩을 사용하여 사이트 손실 방지 기능 활성화"

StorageGRID의 단일 복사본 복제 위험

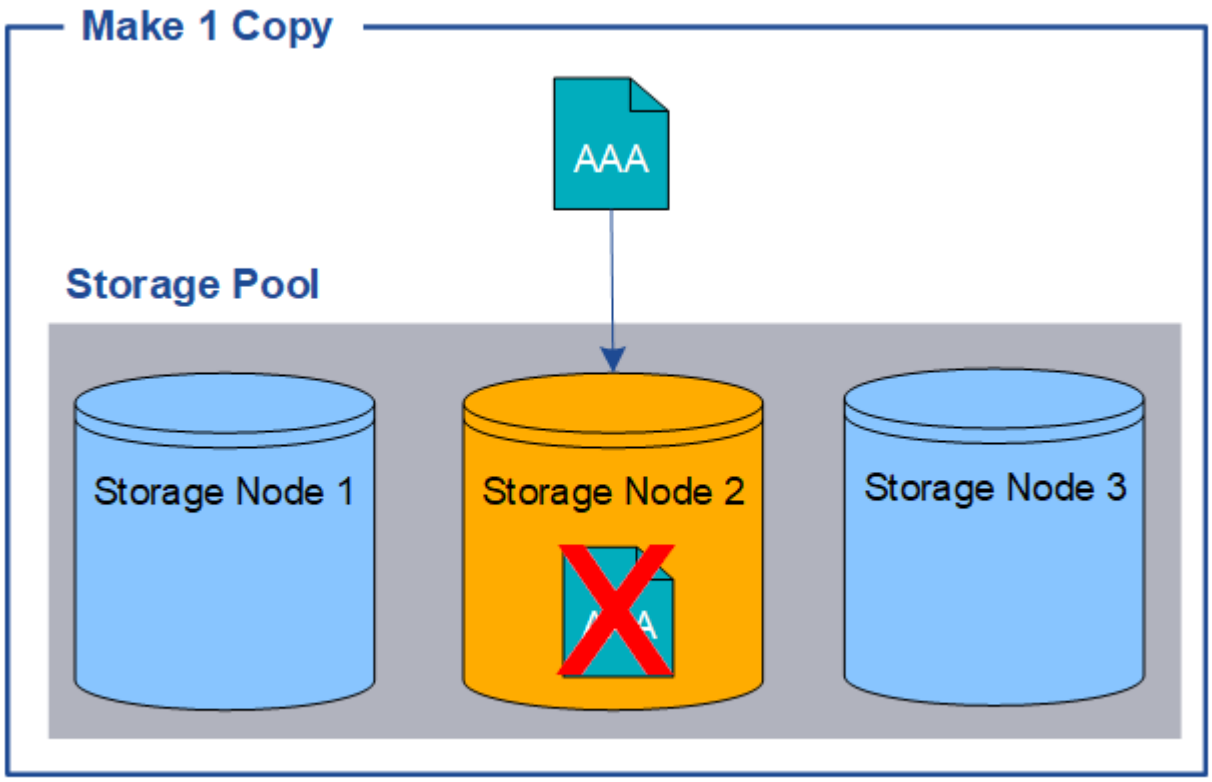
복제본을 생성하는 ILM 규칙을 만들 때는 배치 지침에서 항상 모든 기간에 대해 최소 두 개의 복제본을 지정해야 합니다.

특정 기간 동안 복제본을 하나만 생성하는 ILM 규칙을 사용하지 마십시오. 객체의 복제본이 하나만 존재하는 경우, 스토리지 노드에 장애가 발생하거나 심각한 오류가 발생하면 해당 객체가 손실됩니다. 또한 업그레이드와 같은 유지 관리 절차 중에 해당 객체에 대한 액세스가 일시적으로 차단될 수 있습니다.

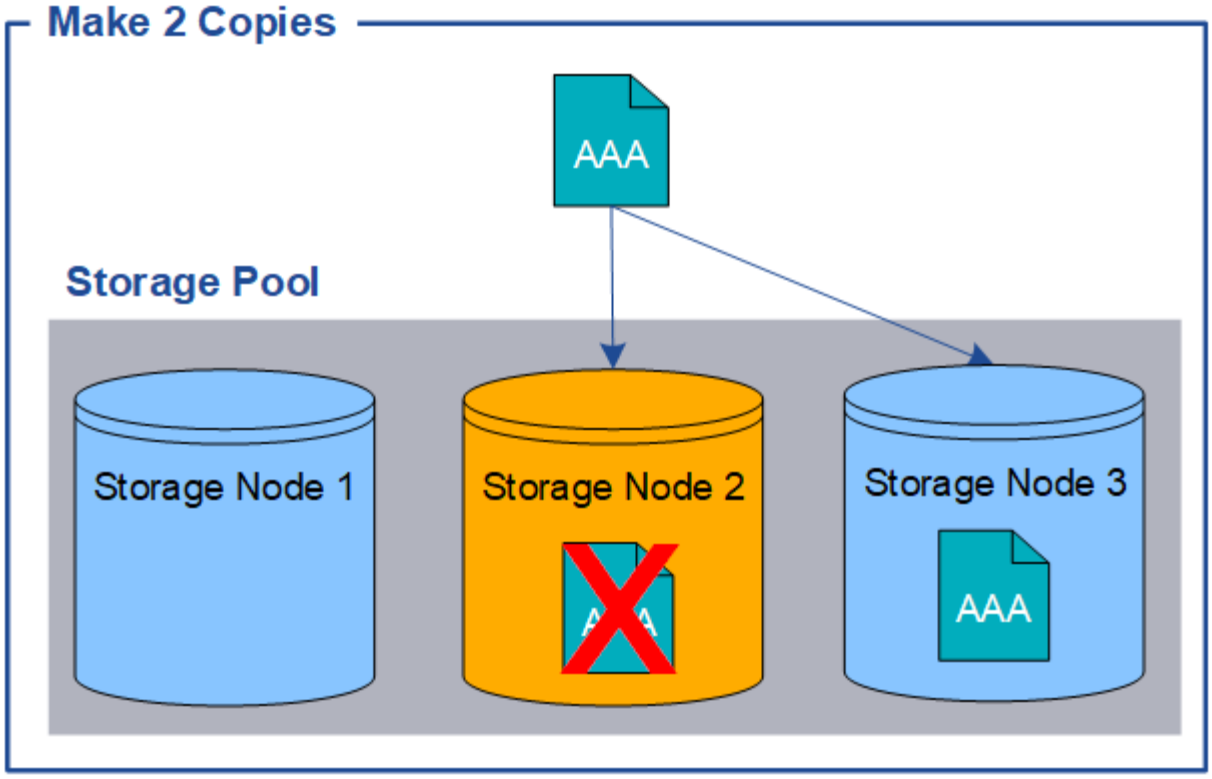
다음 예에서 'Make 1 Copy' ILM 규칙은 객체의 복제본 하나를 3개의 스토리지 노드가 포함된 스토리지 풀에 배치하도록 지정합니다. 이 규칙과 일치하는 객체가 수집되면 StorageGRID는 하나의 스토리지 노드에만 단일 복사본을 배치합니다.



ILM 규칙에 따라 객체의 복제본이 하나만 생성되는 경우, 스토리지 노드를 사용할 수 없으면 해당 객체에 접근할 수 없게 됩니다. 이 예시에서는 스토리지 노드 2가 오프라인 상태일 때(예: 업그레이드 또는 기타 유지 관리 작업 중) 객체 AAA에 대한 접근이 일시적으로 차단됩니다. 스토리지 노드 2에 장애가 발생하면 객체 AAA를 완전히 잃게 됩니다.



객체 데이터 손실을 방지하려면 복제를 통해 보호하려는 모든 객체의 복사본을 최소 두 개 이상 만들어야 합니다. 복사본이 두 개 이상 있으면 스토리지 노드 중 하나에 장애가 발생하거나 오프라인 상태가 되더라도 해당 객체에 계속 액세스할 수 있습니다.



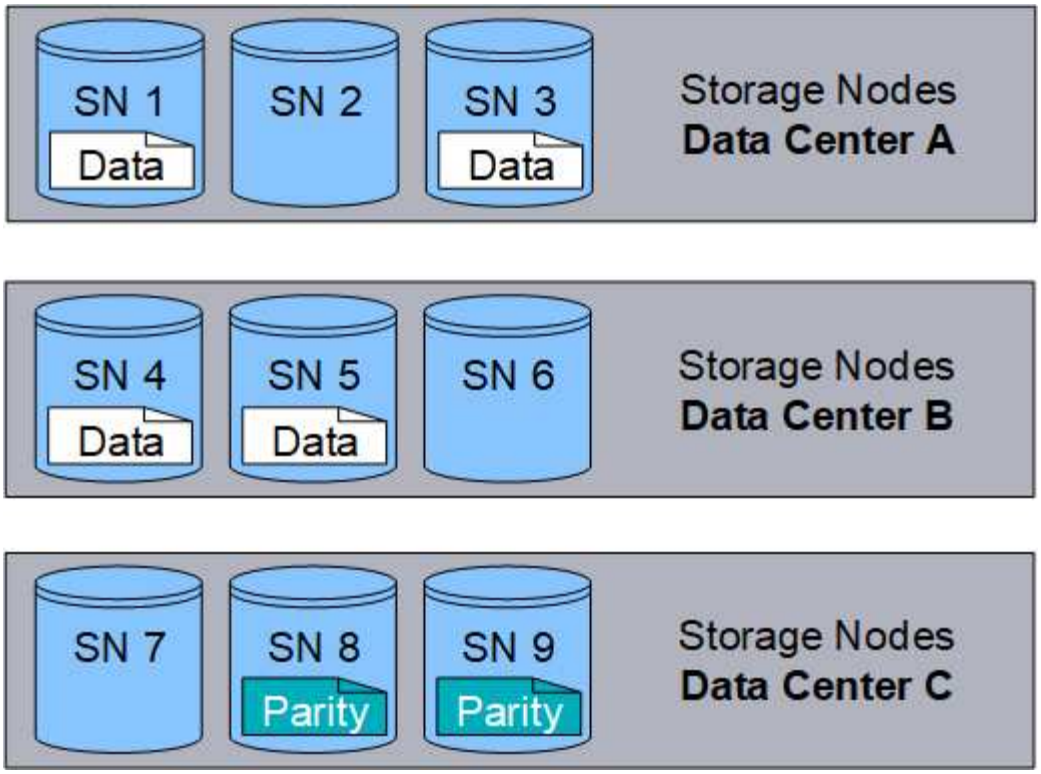
StorageGRID의 삭제 코딩에 대해 알아보십시오

이레이저 코딩은 StorageGRID에서 객체 데이터를 저장하는 두 가지 방법 중 하나입니다(다른 하나는 복제). 객체가 이레이저 코딩을 사용하는 ILM 규칙과 일치하면 해당 객체는 데이터 조각으로 분할되고, 추가 패리티 조각이 계산되며, 각 조각은 서로 다른 스토리지 노드에 저장됩니다.

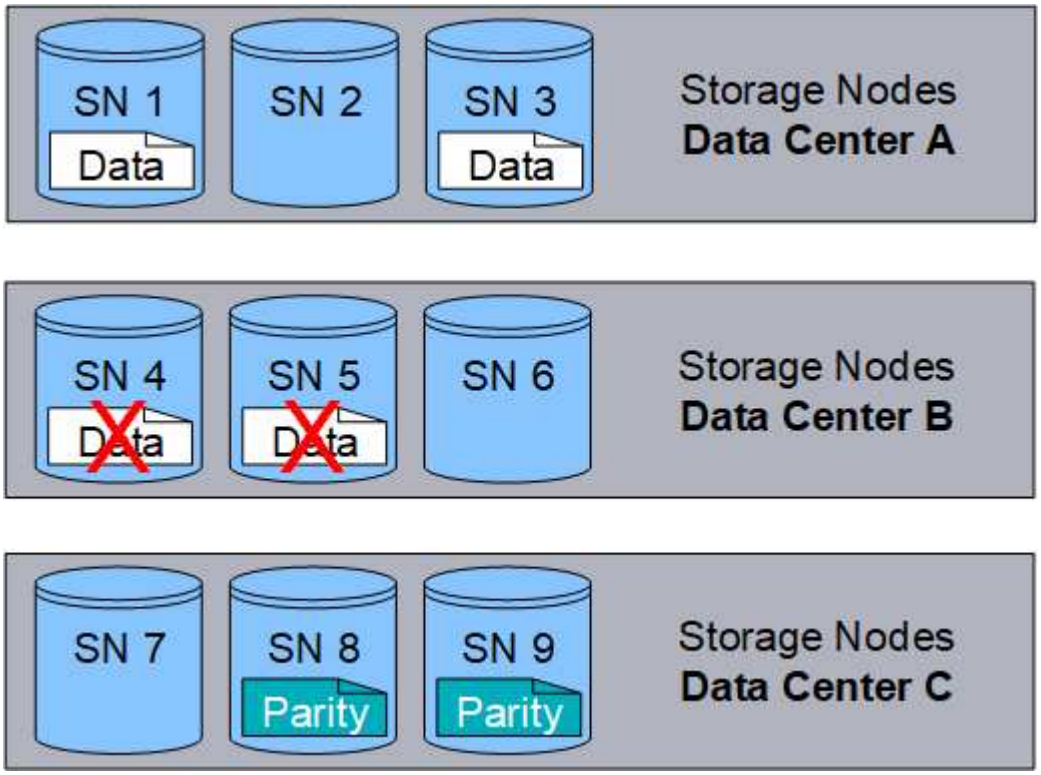
객체에 접근할 때, 저장된 조각들을 사용하여 객체가 재조립됩니다. 데이터 조각이나 패리티 조각이 손상되거나 손실될 경우, 이레이저 코딩 알고리즘은 남아 있는 데이터 및 패리티 조각의 일부를 사용하여 해당 조각을 복원할 수 있습니다.

ILM 규칙을 생성하면 StorageGRID가 해당 규칙을 지원하는 이레이저 코딩 프로파일을 생성합니다. 이레이저 코딩 프로파일 목록을 보거나 "이레이저 코딩 프로파일 이름 바꾸기", 또는 "현재 어떤 ILM 규칙에서도 사용되지 않는 이레이저 코딩 프로파일을 비활성화합니다."할 수 있습니다.

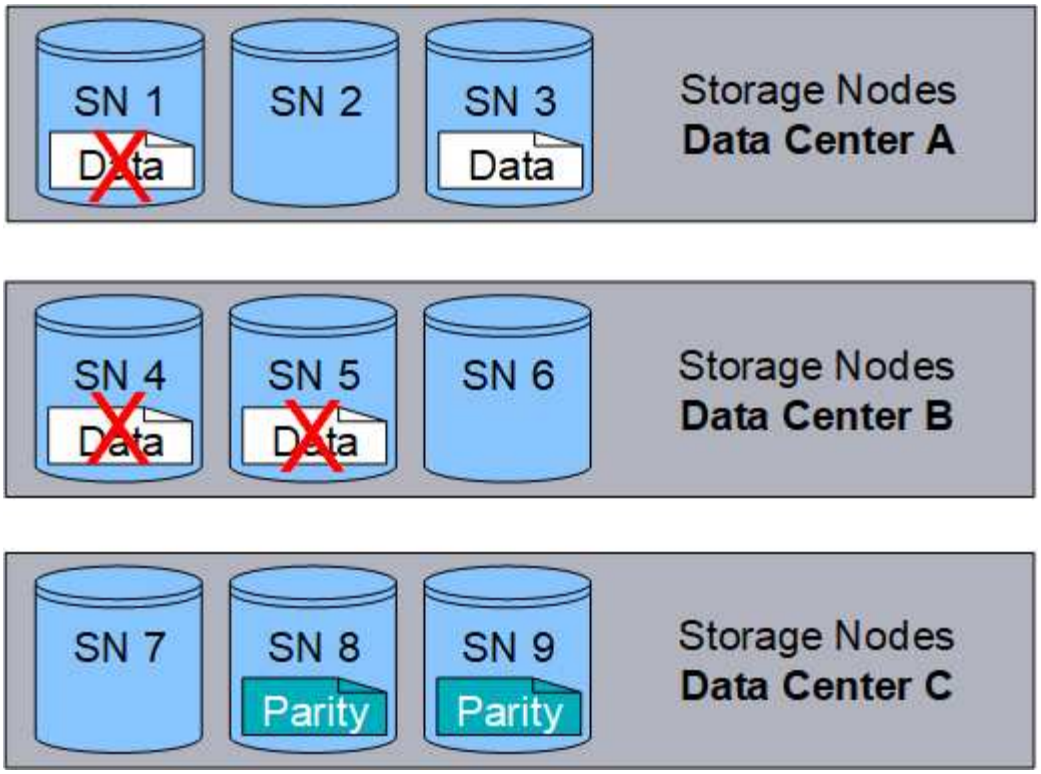
다음 예시는 객체 데이터에 소거 부호화 알고리즘을 적용하는 방법을 보여줍니다. 이 예시에서 ILM 규칙은 4+2 소거 부호화 방식을 사용합니다. 각 객체는 네 개의 동일한 데이터 조각으로 분할되고, 객체 데이터로부터 두 개의 패리티 조각이 계산됩니다. 이렇게 생성된 여섯 개의 조각은 각각 세 개의 데이터 센터 사이트에 분산된 서로 다른 노드에 저장되어 노드 장애나 사이트 손실 시에도 데이터를 보호합니다.



4+2 소거 부호화 방식은 다양한 방식으로 구성할 수 있습니다. 예를 들어, 6개의 스토리지 노드를 포함하는 단일 사이트 스토리지 풀을 구성할 수 있습니다. 또는 "사이트 손실 보호"각 사이트에 3개의 스토리지 노드가 있는 3개의 사이트로 구성된 스토리지 풀을 사용할 수도 있습니다. 객체는 6개의 조각(데이터 또는 패리티) 중 4개 이상이 사용 가능한 상태라면 복구할 수 있습니다. 최대 2개의 조각이 손실되더라도 객체 데이터는 손실되지 않습니다. 전체 사이트가 손실되더라도 나머지 모든 조각에 접근 가능한 한 객체를 복구하거나 복원할 수 있습니다.



스토리지 노드가 두 개 이상 손실되면 해당 객체를 복구할 수 없습니다.



관련 정보

- "복제란 무엇입니까"
- "스토리지 풀이란 무엇인가"
- "소거 부호화 방식이란 무엇인가요?"
- "이레이저 코딩 프로파일 이름 바꾸기"
- "이레이저 코딩 프로파일 비활성화"

StorageGRID의 삭제 코딩 체계에 대해 알아보십시오

소거 부호화 방식은 각 객체에 대해 생성되는 데이터 조각의 수와 패리티 조각의 수를 제어합니다.

ILM 규칙을 생성하거나 편집할 때 사용 가능한 소거 코딩 체계를 선택합니다. StorageGRID는 사용하려는 스토리지 풀을 구성하는 스토리지 노드 및 사이트 수에 따라 소거 코딩 체계를 자동으로 생성합니다.

데이터 보호

StorageGRID 시스템은 Reed-Solomon 소거 부호화 알고리즘을 사용합니다. 이 알고리즘은 객체를 `k`데이터 조각으로 나누고 `m`패리티 조각을 계산합니다.

다음과 같이 데이터 보호를 제공하기 위해 `k + m = n` 조각들이 여러 `n` Storage Node에 걸쳐 분산됩니다.

- 개체를 검색하거나 복구하려면 `k` 조각이 필요합니다.
- 객체는 최대 `m`개의 손실되거나 손상된 조각을 견딜 수 있습니다. `m`의 값이 높을수록 오류 허용치가 높아집니다.

스토리지 풀 내에서 가장 높은 노드 또는 볼륨 장애 허용 범위를 가진 삭제 코딩 방식이 최상의 데이터 보호를

제공합니다.

스토리지 오버헤드

이레이저 코딩 방식의 스토리지 오버헤드는 패리티 조각 수 (m)를 데이터 조각 수 (k)로 나누어 계산합니다. 스토리지 오버헤드를 사용하여 이레이저 코딩된 각 객체에 필요한 디스크 공간을 계산할 수 있습니다.

*disk space = object size + (object size * storage overhead)*

예를 들어, 10MB 크기의 객체를 4+2 방식(스토리지 오버헤드 50%)으로 저장하면 해당 객체는 15MB의 그리드 스토리지를 사용합니다. 동일한 10MB 객체를 6+2 방식(스토리지 오버헤드 33%)으로 저장하면 해당 객체는 약 13.3MB의 그리드 스토리지를 사용합니다.

요구 사항을 충족하는 소거 부호화 방식 중 `k+m`의 총값이 가장 낮은 방식을 선택하십시오. 조각 수가 적은 소거 부호화 방식은 다음과 같은 이유로 계산 효율성이 더 높습니다.

- 객체당 생성 및 배포(또는 검색)되는 조각의 수가 줄어듭니다.
- 조각 크기가 더 크기 때문에 더 나은 성능을 보입니다.
- 추가해야 하는 노드 수를 줄일 수 있습니다."저장 용량이 더 필요할 때 확장 가능"

스토리지 풀에 대한 지침

이레이저 코딩 복사본을 생성하는 규칙에 사용할 스토리지 풀을 선택할 때는 다음 스토리지 풀 지침을 따르십시오.

- 스토리지 풀에는 세 개 이상의 사이트 또는 정확히 하나의 사이트가 포함되어야 합니다.

  > 스토리지 풀에 두 개의 사이트가 포함된 경우 이레이저 코딩을 사용할 수 없습니다.

  - 세 개 이상의 사이트를 포함하는 스토리지 풀에 대한 소거 코딩 방식
  - 단일 사이트 스토리지 풀을 위한 소거 코딩 방식
- 모든 사이트 사이트가 포함된 스토리지 풀을 사용하지 마십시오.
- 스토리지 풀에는 객체 데이터를 저장할 수 있는 Storage Nodes가 최소 *k+m* +1 개 이상 포함되어야 합니다.

  > 스토리지 노드는 설치 중에 객체 데이터와 메타데이터("결합형" 스토리지 노드), 객체 메타데이터만, 또는 객체 데이터만 포함하도록 구성할 수 있습니다. 자세한 내용은 "스토리지 노드의 유형"을 참조하십시오.

  최소 스토리지 노드 수는 `k+m`입니다. 하지만 필수 스토리지 노드를 일시적으로 사용할 수 없는 경우 데이터 수집 실패 또는 ILM 백로그를 방지하기 위해 스토리지 노드를 하나 이상 추가로 확보하는 것이 좋습니다.

세 개 이상의 사이트를 포함하는 스토리지 풀에 대한 소거 코딩 방식

다음 표는 세 개 이상의 사이트를 포함하는 스토리지 풀에 대해 StorageGRID에서 현재 지원하는 소거 코딩 방식을 설명합니다. 이러한 모든 방식은 사이트 손실 보호 기능을 제공합니다. 하나의 사이트에 손실이 발생하더라도 해당 객체는 계속 액세스할 수 있습니다.

사이트 손실 보호 기능을 제공하는 이레이저 코딩 방식의 경우, 각 사이트에 최소 3개의 스토리지 노드가 필요하므로 스토리지 풀에 권장되는 스토리지 노드 수가 초과됩니다. *k+m* +1

소거 부호화 방식 (*k+m*)	최소 배포 사이트 수	사이트별 권장 스토리지 노드 수	권장 총 스토리지 노드 수	사이트 손실 보호?	스토리지 오버헤드
4+2	3	3	9	예	50%
6+2	4	3	12	예	33%
8+2	5	3	15	예	25%
6+3	3	4	12	예	50%
9+3	4	4	16	예	33%
2+1	3	3	9	예	50%
4+1	5	3	15	예	25%
6+1	7	3	21	예	17%
7+5	3	5	15	예	71%

StorageGRID는 사이트당 최소 3개의 스토리지 노드가 필요합니다. 7+5 구성 방식을 사용하려면 각 사이트에 최소 4개의 스토리지 노드가 필요합니다. 사이트당 5개의 스토리지 노드를 사용하는 것이 좋습니다.

사이트 보호 기능을 제공하는 소거 코딩 방식을 선택할 때는 다음 요소들의 상대적 중요도를 균형 있게 고려해야 합니다.

- 조각 개수: 일반적으로 전체 조각 개수가 적을수록 성능과 확장 유연성이 향상됩니다.
- 내결함성: 패리티 세그먼트 수가 많을수록(즉, `m`값이 높을수록) 내결함성이 향상됩니다.
- 네트워크 트래픽: 장애 복구 시, 더 많은 조각(즉, `k+m`의 합계가 더 높은)을 사용하는 방식은 더 많은 네트워크 트래픽을 생성합니다.
- 스토리지 오버헤드: 오버헤드가 높은 스키마는 객체당 더 많은 스토리지 공간을 필요로 합니다.

예를 들어, 4+2 방식과 6+3 방식(둘 다 스토리지 오버헤드가 50%) 중에서 선택할 때, 추가적인 내결함성이 필요하다면 6+3 방식을 선택하십시오. 네트워크 리소스가 제한적이라면 4+2 방식을 선택하십시오. 다른 모든 조건이 동일하다면, 전체 조각 수가 더 적은 4+2 방식을 선택하십시오.

어떤 방식을 사용해야 할지 확실하지 않은 경우 4+2 또는 6+3을 선택하거나 기술 지원에 문의하십시오.

단일 사이트 스토리지 풀을 위한 소거 코딩 방식

단일 사이트 스토리지 풀은 해당 사이트에 충분한 스토리지 노드가 있는 경우 3개 이상의 사이트에 대해 정의된 모든 이레이저 코딩 방식을 지원합니다.

최소 스토리지 노드 수는 `k+m`이지만, `k+m +1`개의 스토리지 노드가 포함된 스토리지 풀을 사용하는 것이 좋습니다. 예를 들어, 2+1 이레이저 코딩 방식에는 최소 3개의 스토리지 노드가 포함된 스토리지 풀이 필요하지만, 4개의 스토리지

노드를 사용하는 것이 권장됩니다.

소거 부호화 방식(*k*+*m*)	최소 스토리지 노드 수	권장 스토리지 노드 수	스토리지 오버헤드
4+2	6	7	50%
6+2	8	9	33%
8+2	10	11	25%
6+3	9	10	50%
9+3	12	13	33%
2+1	3	4	50%
4+1	5	6	25%
6+1	7	8	17%
7+5	12	13	71%

StorageGRID의 삭제 코딩에 대한 장점, 단점 및 요구 사항

객체 데이터 손실을 방지하기 위해 복제 또는 소거 코딩 중 어떤 방법을 사용할지 결정하기 전에 소거 코딩의 장점, 단점 및 요구 사항을 이해해야 합니다.

이레이저 코딩의 장점

복제 방식과 비교했을 때, 이레이저 코딩은 향상된 신뢰성, 가용성 및 스토리지 효율성을 제공합니다.

- 신뢰성: 데이터 손실 없이 견딜 수 있는 동시 장애 발생 횟수로 측정합니다.
  - 복제: 동일한 객체의 복사본이 여러 노드와 사이트에 걸쳐 저장됩니다.
  - 이레이저 코딩: 객체를 데이터와 패리티 조각으로 인코딩하여 여러 노드와 사이트에 분산시킵니다. 이러한 분산은 사이트 및 노드 장애로부터 객체를 보호합니다.
- 가용성: 스토리지 노드에 장애가 발생하거나 접근할 수 없게 된 경우에도 객체에 접근할 수 있는 능력. 복제 방식과 비교했을 때, 이레이저 코딩은 비슷한 스토리지 비용으로 향상된 가용성을 제공합니다.
- 저장 효율성: 유사한 가용성과 안정성을 유지할 경우, 이레이저 코딩된 객체는 복제된 객체보다 디스크 공간을 적게 사용합니다. 예를 들어, 10MB 크기의 객체를 두 사이트에 복제하면 20MB의 디스크 공간(두 개의 복사본)이 사용되는 반면, 6+3 이레이저 코딩 방식을 사용하여 세 사이트에 걸쳐 이레이저 코딩된 객체는 15MB의 디스크 공간만 사용합니다.

> 소거 부호화된 객체의 디스크 공간은 객체 크기와 스토리지 오버헤드의 합으로 계산됩니다. 스토리지 오버헤드 비율은 패리티 조각 수를 데이터 조각 수로 나눈 값입니다.

소거 코딩의 단점

복제 방식과 비교했을 때, 이레이저 코딩은 다음과 같은 단점이 있습니다.

- 소거 코딩 방식에 따라 스토리지 노드 및 사이트 수를 늘리는 것이 좋습니다. 반면 객체 데이터를 복제하는 경우에는 복사본 하나당 스토리지 노드 하나만 필요합니다. "세 개 이상의 사이트를 포함하는 스토리지 풀에 대한 소거 코딩 방식" 및 "단일 사이트 스토리지 풀을 위한 소거 코딩 방식"을 참조하십시오.
- 스토리지 확장 비용 및 복잡성 증가. 복제를 사용하는 배포를 확장하려면 객체 복사본이 생성되는 모든 위치에 스토리지 용량을 추가해야 합니다. 이레이저 코딩을 사용하는 배포를 확장하려면 사용 중인 이레이저 코딩 방식과 기존 스토리지 노드의 사용률을 모두 고려해야 합니다. 예를 들어 기존 노드가 100% 꽉 찰 때까지 기다리면 최소 `k+m` 스토리지 노드를 추가해야 하지만, 기존 노드가 70% 찼을 때 확장하면 사이트당 노드를 두 개만 추가해도 사용 가능한 스토리지 용량을 최대화할 수 있습니다. 자세한 내용은 "삭제 코딩 오브젝트의 스토리지 용량 추가"을 참조하십시오.
- 지리적으로 분산된 사이트에서 소거 부호화를 사용할 경우 검색 지연 시간이 증가합니다. 소거 부호화되어 원격 사이트에 분산된 객체의 조각들을 WAN 연결을 통해 검색하는 데 걸리는 시간은 복제되어 로컬(클라이언트가 연결하는 동일한 사이트)에 있는 객체를 검색하는 데 걸리는 시간보다 더 오래 걸립니다.
- 지리적으로 분산된 사이트에서 이레이저 코딩을 사용하는 경우, 검색 및 복구를 위한 WAN 네트워크 트래픽 사용량이 증가합니다. 특히 자주 검색되는 객체나 WAN 네트워크 연결을 통한 객체 복구의 경우 더욱 그렇습니다.
- 사이트 간 이레이저 코딩을 사용하는 경우, 사이트 간 네트워크 지연 시간이 증가함에 따라 최대 객체 처리량이 급격히 감소합니다. 이러한 감소는 TCP 네트워크 처리량 저하로 인해 발생하며, 이는 StorageGRID 시스템이 객체 조각을 저장하고 검색하는 속도에 영향을 미칩니다.
- 컴퓨팅 자원 사용량 증가.

삭제 코딩을 사용해야 하는 경우

다음 요구 사항에 대해서는 소거 코딩을 사용하십시오.

- 크기가 1MB보다 큰 객체.

> 이레이저 코딩은 1MB보다 큰 객체에 가장 적합합니다. 매우 작은 이레이저 코딩된 조각을 관리하는 데 드는 오버헤드를 피하기 위해 200KB보다 작은 객체에는 이레이저 코딩을 사용하지 마십시오.

- 자주 검색하지 않는 콘텐츠를 위한 장기 또는 콜드 스토리지.
- 높은 데이터 가용성과 신뢰성.
- 사이트 전체 또는 노드 장애로부터 보호합니다.
- 스토리지 효율성.
- 여러 복제본 대신 단일 이레이저 코딩 복사본만으로 효율적인 데이터 보호가 필요한 단일 사이트 배포 환경.
- 사이트 간 지연 시간이 100ms 미만인 다중 사이트 배포 환경.

StorageGRID에서 객체 보존을 결정하는 방법

StorageGRID는 그리드 관리자와 개별 테넌트 사용자 모두에게 객체를 저장할 기간을 지정할 수 있는 옵션을 제공합니다. 일반적으로 테넌트 사용자가 제공하는 보존 지침은 그리드 관리자가 제공하는 보존 지침보다 우선합니다.

테넌트 사용자가 객체 보존을 제어하는 방법

테넌트 사용자는 이러한 방법을 사용하여 개체가 StorageGRID에 저장되는 기간을 제어할 수 있습니다.

- 그리드에 대해 전역 S3 객체 잠금 설정이 활성화된 경우 S3 테넌트 사용자는 S3 객체 잠금이 활성화된 버킷을 생성한 다음 각 버킷에 대해 *기본 보존 기간*을 선택할 수 있습니다.
- 그리드에 대해 전역 S3 객체 잠금 설정이 활성화된 경우, S3 테넌트 사용자는 S3 객체 잠금이 활성화된 버킷을 생성한 다음 S3 REST API를 사용하여 해당 버킷에 추가된 각 객체 버전에 대한 retain-until-date 및 법적 보존 설정을 지정할 수 있습니다.
  - 법적 보존 조치가 적용된 객체 버전은 어떤 방법으로도 삭제할 수 없습니다.
  - 객체 버전의 retain-until-date에 도달하기 전까지는 어떤 방법으로도 해당 버전을 삭제할 수 없습니다.
  - S3 객체 잠금이 활성화된 버킷의 객체는 ILM에 의해 "영구적으로" 보존됩니다. 그러나 보존 기한이 만료된 후에는 클라이언트 요청이나 버킷 수명 주기 만료로 객체 버전이 삭제될 수 있습니다. "S3 Object Lock으로 객체 관리"을 참조하십시오.
- S3 테넌트 사용자는 만료 작업을 지정하는 수명 주기 구성을 버킷에 추가할 수 있습니다. 버킷 수명 주기가 있는 경우, 클라이언트가 객체를 먼저 삭제하지 않는 한 StorageGRID는 만료 작업에 지정된 날짜 또는 일수가 충족될 때까지 객체를 저장합니다. "S3 라이프사이클 구성 생성"을 참조하십시오.
- S3 클라이언트는 객체 삭제 요청을 보낼 수 있습니다. StorageGRID는 객체 삭제 또는 유지 여부를 결정할 때 S3 버킷 수명 주기 또는 ILM보다 클라이언트 삭제 요청을 항상 우선시합니다.

그리드 관리자가 객체 보존을 제어하는 방법

그리드 관리자는 다음 방법을 사용하여 객체 보존을 제어할 수 있습니다.

- 각 테넌트에 대한 S3 Object Lock 최대 보존 기간을 설정합니다. 그러면 테넌트 사용자는 각 버킷에 대한 기본 보존 기간을 설정할 수 있습니다. 최대 보존 기간은 해당 버킷에 새로 추가되는 모든 객체(객체의 보존 만료일)에도 적용됩니다.
- ILM 배치 지침을 생성하여 객체의 저장 기간을 제어합니다. 객체가 ILM 규칙과 일치하면 StorageGRID는 해당 ILM 규칙에 지정된 마지막 기간이 경과할 때까지 객체를 저장합니다. 배치 지침에 "영구"를 지정하면 객체는 무기한으로 보존됩니다.
- 객체 보존 기간을 누가 제어하든 관계없이 ILM 설정은 객체 복사본의 유형(복제본 또는 이레이저 코딩본)과 복사본의 위치(스토리지 노드 또는 클라우드 스토리지 풀)를 제어합니다.

S3 버킷 수명 주기와 ILM의 상호 작용 방식

S3 버킷 수명 주기가 구성된 경우, 수명 주기 필터와 일치하는 객체에 대해서는 수명 주기 만료 작업이 ILM 정책을 재정의합니다. 결과적으로, 객체를 그리드에 배치하라는 ILM 지침이 만료된 후에도 객체가 그리드에 유지될 수 있습니다.

객체 보존의 예

S3 객체 잠금, 버킷 수명 주기 설정, 클라이언트 삭제 요청 및 ILM 간의 상호 작용을 더 잘 이해하려면 다음 예제를 살펴보십시오.

예시 **1: S3** 버킷 수명 주기는 **ILM**보다 객체를 더 오래 보관합니다

ILM

1년(365일) 동안 두 개의 복사본을 보관합니다.

버킷 수명 주기

2년(730일) 후 개체 만료

결과

StorageGRID는 객체를 730일 동안 보관합니다. StorageGRID는 버킷 수명 주기 설정을 사용하여 객체를 삭제할지 유지할지 결정합니다.

> ⓘ 버킷 수명 주기에서 ILM에서 지정한 기간보다 더 오래 객체를 보관하도록 지정된 경우, StorageGRID는 저장할 복사본의 수와 유형을 결정할 때 ILM 배치 지침을 계속 사용합니다. 이 예에서는 객체의 복사본 두 개가 366일부터 730일까지 StorageGRID에 계속 저장됩니다.

예시 **2: S3** 버킷 수명 주기가 **ILM**보다 먼저 객체를 만료시킴

ILM

사본 두 부를 2년(730일) 동안 보관합니다.

버킷 수명 주기

1년(365일) 후 개체 만료

결과

StorageGRID는 365일 후 해당 객체의 두 복사본을 모두 삭제합니다.

예시 **3:** 클라이언트 삭제가 버킷 수명 주기 및 **ILM**을 재정의합니다

ILM

스토리지 노드에 두 개의 복사본을 "영구적으로" 저장합니다.

버킷 수명 주기

2년(730일) 후 개체 만료

클라이언트 삭제 요청

400일째 되는 날 발행됨

결과

StorageGRID는 클라이언트의 삭제 요청에 따라 400일째 되는 날 해당 객체의 두 복사본을 모두 삭제합니다.

예시 **4: S3 Object Lock**이 클라이언트 삭제 요청을 재정의함

S3 Object Lock

해당 객체 버전의 보존 기한은 2026년 3월 31일입니다. 법적 보존 조치는 적용되지 않습니다.

규정 준수 **ILM** 규칙

스토리지 노드에 두 개의 복사본을 "영구적으로" 저장합니다.

클라이언트 삭제 요청

2024년 3월 31일 발행

결과

StorageGRID 보존 기한이 아직 2년 남았으므로 객체 버전을 삭제하지 않습니다.

StorageGRID에서 객체를 삭제하는 방법

StorageGRID는 클라이언트 요청에 직접 응답하여 객체를 삭제하거나, S3 버킷 수명 주기 만료 또는 ILM 정책 요구 사항에 따라 자동으로 객체를 삭제할 수 있습니다. 객체를 삭제하는 다양한 방법과 StorageGRID가 삭제 요청을 처리하는 방식을 이해하면 객체를 더욱 효과적으로 관리할 수 있습니다.

StorageGRID는 객체를 삭제하기 위해 다음 두 가지 방법 중 하나를 사용할 수 있습니다.

- 동기식 삭제: StorageGRID가 클라이언트의 삭제 요청을 수신하면 모든 객체 복사본을 즉시 제거합니다. 복사본이 제거된 후 클라이언트에게 삭제가 성공했음을 알립니다.
- 객체는 삭제 대기열에 추가됩니다. StorageGRID가 삭제 요청을 수신하면 해당 객체는 삭제 대기열에 추가되고, 클라이언트에 삭제가 성공적으로 완료되었다는 알림이 즉시 전송됩니다. 객체 복사본은 백그라운드 ILM 처리 과정을 통해 나중에 제거됩니다.

객체를 삭제할 때 StorageGRID는 삭제 성능을 최적화하고, 삭제 백로그 발생 가능성을 최소화하며, 공간을 가장 빠르게 확보하는 방법을 사용합니다.

이 표는 StorageGRID가 각 메서드를 사용하는 시점을 요약한 것입니다.

삭제 수행 방법	사용 시
객체가 삭제 대기열에 추가됩니다.	다음 조건 중 하나라도 충족될 경우: • 자동 객체 삭제는 다음 이벤트 중 하나에 의해 트리거되었습니다. ◦ S3 버킷의 라이프사이클 구성에 설정된 만료일 또는 일수에 도달했습니다. ◦ ILM 규칙에 지정된 마지막 기간이 경과합니다. 참고: S3 객체 잠금이 활성화된 버킷의 객체는 법적 보존 조치가 적용 중이거나 보존 기한이 지정되었지만 아직 도래하지 않은 경우 삭제할 수 없습니다. • S3 클라이언트가 삭제를 요청했고 다음 조건 중 하나 이상이 충족되었습니다. ◦ 예를 들어 개체 위치를 일시적으로 사용할 수 없는 경우와 같이 30초 이내에 복사본을 삭제할 수 없습니다. ◦ 백그라운드 삭제 대기열이 유휴 상태입니다.
객체는 즉시 삭제됩니다 (동기식 삭제)	S3 클라이언트가 삭제 요청을 하고 다음 조건이 모두 충족될 경우: • 모든 복사본은 30초 이내에 삭제할 수 있습니다. • 백그라운드 삭제 대기열에는 처리할 객체가 포함되어 있습니다.

S3 클라이언트가 삭제 요청을 보내면 StorageGRID는 먼저 삭제 큐에 객체를 추가합니다. 그런 다음 동기식 삭제로 전환합니다. 백그라운드 삭제 큐에 처리할 객체가 항상 있도록 함으로써 StorageGRID는 특히 동시 접속 클라이언트 수가 적은 경우 삭제 작업을 더욱 효율적으로 처리할 수 있으며, 클라이언트 삭제 요청이 누적되는 것을 방지할 수 있습니다.

객체 삭제에 소요되는 시간

StorageGRID가 객체를 삭제하는 방식은 시스템 성능에 영향을 미칠 수 있습니다.

- StorageGRID가 동기식 삭제를 수행할 때 StorageGRID가 클라이언트에 결과를 반환하는 데 최대 30초가 걸릴 수 있습니다. 즉, StorageGRID가 객체를 삭제 대기열에 저장할 때보다 실제로 복사본이 더 빠르게 제거되더라도 삭제 속도가 더 느리게 느껴질 수 있습니다.
- 대량 삭제 시 삭제 성능을 면밀히 모니터링하는 경우, 일정 개수의 객체가 삭제된 후 삭제 속도가 느려지는 것처럼 보일 수 있습니다. 이러한 변화는 StorageGRID가 삭제를 위해 객체를 큐에 대기시키는 방식에서 동기식 삭제 방식으로 전환하면서 발생합니다. 삭제 속도가 느려진 것처럼 보이는 것은 객체 복사본이 더 느리게 제거된다는 의미가 아닙니다. 오히려 평균적으로 공간이 더 빠르게 확보되고 있음을 나타냅니다.

대량의 객체를 삭제해야 하고 공간을 신속하게 확보하는 것이 최우선이라면, ILM이나 다른 방법을 사용하는 대신 클라이언트 요청을 통해 객체를 삭제하는 것을 고려하십시오. 일반적으로 StorageGRID가 동기식 삭제를 지원하기 때문에 클라이언트를 통한 삭제가 더 빠르게 공간을 확보합니다.

객체를 삭제한 후 공간이 확보되는 데 필요한 시간은 여러 요인에 따라 달라집니다.

- 객체 복사본이 동기적으로 삭제되는지 아니면 나중에 삭제되도록 대기열에 추가되는지 여부(클라이언트 삭제 요청 시).
- 그리드에 있는 객체 수 또는 객체 복사가 삭제 대기열에 추가될 때(클라이언트 삭제 및 기타 방법 모두에 대해) 그리드 리소스의 가용성과 같은 다른 요소도 영향을 미칩니다.

S3 버전 관리 객체가 삭제되는 방법

S3 버킷에 버전 관리가 활성화된 경우, StorageGRID는 S3 클라이언트, S3 버킷 라이프사이클 만료 또는 ILM 정책 요구 사항에서 오는 삭제 요청에 응답할 때 Amazon S3 동작을 따릅니다.

객체에 버전 관리 기능이 있는 경우, 객체 삭제 요청은 현재 버전의 객체를 삭제하거나 공간을 확보하지 않습니다. 대신, 객체 삭제 요청은 객체의 현재 버전으로 0바이트 삭제 마커를 생성하여 이전 버전의 객체를 "비최신" 상태로 만듭니다. 객체 삭제 마커는 현재 버전이고 비최신 버전이 없을 때 만료된 객체 삭제 마커가 됩니다.

객체가 삭제되지 않았더라도 StorageGRID는 해당 객체의 현재 버전을 더 이상 사용할 수 없는 것처럼 동작합니다. 해당 객체에 대한 요청은 404 NotFound를 반환합니다. 하지만 현재 버전이 아닌 객체 데이터가 삭제되지 않았기 때문에, 현재 버전이 아닌 객체를 지정하는 요청은 성공할 수 있습니다.

클라이언트가 "브랜치 버킷"에서 버전 ID가 있는 객체 버전을 삭제하면 StorageGRID는 해당 객체 버전의 존재를 숨기지만 브랜치 버킷에 삭제 표시를 생성하지 않습니다. 클라이언트가 버전 ID가 없는 객체를 브랜치 버킷에서 삭제하면 StorageGRID는 표준 S3 동작에 따라 브랜치 버킷에 삭제 표시를 생성합니다.

버전 관리되는 개체를 삭제할 때 공간을 확보하거나 삭제 표시를 제거하려면 다음 중 하나를 사용하십시오.

- **S3** 클라이언트 요청: S3 DELETE Object 요청에서 객체 버전 ID를 지정하십시오 (`DELETE /object?versionId=ID`). 이 요청은 지정된 버전의 객체 복사본만 제거하며, 다른 버전은 여전히 공간을 차지한다는 점에 유의하십시오.
- 버킷 수명 주기: 버킷 수명 주기 구성에서 `NoncurrentVersionExpiration` 작업을 사용합니다. 지정된

NoncurrentDays 일수가 경과하면 StorageGRID는 현재 사용되지 않는 객체 버전의 모든 복사본을 영구적으로 삭제합니다. 이러한 객체 버전은 복구할 수 없습니다.

버킷 수명 주기 구성의 `NewerNoncurrentVersions` 작업은 버전이 지정된 S3 버킷에 보존할 비활성 버전의 수를 지정합니다. 비활성 버전이 `NewerNoncurrentVersions`에서 지정한 수보다 많으면 `NoncurrentDays` 값이 경과할 때 StorageGRID가 이전 버전을 삭제합니다. `NewerNoncurrentVersions` 임계값은 ILM에서 제공하는 수명 주기 규칙을 재정의하므로, ILM에서 삭제를 요청하더라도 `NewerNoncurrentVersions` 임계값 내의 버전을 가진 비활성 객체는 보존됩니다.

만료된 객체 삭제 표시를 제거하려면 `Expiration` 작업을 다음 태그 중 하나와 함께 사용하십시오: `ExpiredObjectDeleteMarker`, `Days`, 또는 `Date`.

- **ILM**: "활성 정책 복제" 새 정책에 ILM 규칙 두 개를 추가합니다.
  - 첫 번째 규칙: 객체의 이전 버전을 일치시키기 위한 참조 시간으로 "Noncurrent time"을 사용합니다. "ILM 규칙 생성 마법사의 1단계(세부 정보 입력)"에서 "버전 관리가 활성화된 S3 버킷에서 이 규칙을 이전 객체 버전에만 적용하시겠습니까?"라는 질문에 *예*를 선택합니다.
  - 두 번째 규칙: 수집 시간*을 현재 버전과 일치시키는 데 사용합니다**. "Noncurrent time"** 규칙은 정책에서 *수집 시간 규칙보다 위에 위치해야 합니다.

    만료된 객체 삭제 표시를 제거하려면 현재 삭제 표시와 일치하는 수집 시간 규칙을 사용하십시오. 삭제 표시는 *일*의 *기간*이 경과하고 현재 삭제 표시가 만료된 경우에만 제거됩니다(현재 버전이 아닌 다른 버전이 없는 경우).
- 버킷에서 객체 삭제: 테넌트 관리자를 사용하여 삭제 마커를 포함한 객체를 버킷에서 "모든 객체 버전 삭제" 삭제할 수 있습니다.

버전이 지정된 객체를 삭제하면 StorageGRID는 해당 객체의 현재 버전으로 0바이트 크기의 삭제 마커를 생성합니다. 버전이 지정된 버킷을 삭제하려면 모든 객체와 삭제 마커를 제거해야 합니다.

- StorageGRID 11.7 이하 버전에서 생성된 삭제 마커는 S3 클라이언트 요청을 통해서만 제거할 수 있으며, ILM, 버킷 수명 주기 규칙 또는 버킷 내 객체 삭제 작업으로는 제거되지 않습니다.
- StorageGRID 11.8 이상에서 생성된 버킷의 삭제 마커는 ILM, 버킷 수명 주기 규칙, 버킷 내 객체 삭제 작업 또는 명시적인 S3 클라이언트 삭제를 통해 제거할 수 있습니다.

관련 정보

- "S3 REST API 사용"
- "예제 4: S3 버전 관리 객체에 대한 ILM 규칙 및 정책"