Wie StorageGRID die S3 REST API implementiert

StorageGRID software

NetApp
July 23, 2026

Inhalt

Wie StorageGRID die S3 REST API implementiert

Wie StorageGRID S3 REST API widersprüchliche Clientanfragen löst

Widersprüchliche Clientanfragen, wie zum Beispiel das Schreiben zweier Clients auf denselben Schlüssel, werden nach dem Prinzip „latest-wins“ gelöst.

Der Zeitpunkt für die Auswertung „latest-wins“ richtet sich danach, wann das StorageGRID System eine bestimmte Anfrage abschließt, und nicht danach, wann S3 Clients eine Operation starten.

Wie StorageGRID S3 REST API Verfügbarkeit und Konsistenz in Einklang bringt

Konsistenz sorgt für ein Gleichgewicht zwischen der Verfügbarkeit der Objekte und der Konsistenz dieser Objekte über verschiedene Storage Nodes und Standorte hinweg. Die Konsistenz kann entsprechend den Anforderungen Ihrer Anwendung geändert werden.

Standardmäßig garantiert StorageGRID Lesekonsistenz nach dem Schreiben für neu erstellte Objekte. Jeder GET nach einem erfolgreich abgeschlossenen PUT kann die neu geschriebenen Daten lesen. Das Überschreiben bestehender Objekte, Metadatenaktualisierungen und Löschvorgänge sind schließlich konsistent.

Wenn Sie Objektoperationen mit einer anderen Konsistenz durchführen möchten, können Sie:

- Eine Konsistenz für jeder Bucket angeben.
- Eine Konsistenz für jede API-Operation angeben.
- Die standardmäßige netzweite Konsistenz kann durch Ausführen einer der folgenden Aufgaben geändert werden:
  - Im Grid Manager zu **Konfiguration** > **System** > **Speichereinstellungen** > **Standardmäßige Bucket-Konsistenz** navigieren.
  - .

> Eine Änderung der netzweiten Konsistenz gilt nur für Buckets, die nach der Änderung der Einstellung erstellt wurden. Die Details einer Änderung sind im Revisionsprotokoll unter `/var/local/log` zu finden (Suche nach **consistencyLevel**).

Konsistenzwerte

Die Konsistenz beeinflusst, wie die Metadaten, die StorageGRID zur Objektverfolgung verwendet, zwischen den Knoten verteilt werden. Die Konsistenz beeinflusst die Verfügbarkeit von Objekten für Clientanfragen.

Sie können die Konsistenz für einen Bucket oder eine API-Operation auf einen der folgenden Werte festlegen:

- **Alle**: Alle Knoten erhalten die Objektmetadaten sofort oder die Anfrage schlägt fehl.
- **Stark-global**: Gewährleistet Lese-nach-Schreib-Konsistenz für alle Clientanfragen an allen Standorten. Bei konfigurierter Quorum-Semantik gelten folgende Verhaltensweisen:

- Ermöglicht die Standortausfalltoleranz für Clientanfragen bei Grids mit drei oder mehr Standorten. Grids mit nur zwei Standorten bieten keine Standortausfalltoleranz.
- Die folgenden S3-Operationen sind bei einem Standortausfall nicht erfolgreich:
  - DeleteBucketEncryption
  - PutBucketBranch
  - PutBucketEncryption
  - PutBucketVersioning
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

Falls erforderlich kann "StorageGRID Quorumsemantik für starke globale Konsistenz konfigurieren" verwendet werden.

- **Strong-site**: Objektmetadaten werden sofort an andere Knoten am Standort verteilt. Garantiert Lese-nach-Schreib-Konsistenz für alle Clientanfragen innerhalb eines Standorts.
- **Read-after-new-write**: Gewährleistet Lesekonsistenz nach dem Schreiben für neue Objekte und letztendliche Konsistenz für Objektaktualisierungen. Bietet hohe Verfügbarkeit und Garantien für den Datenschutz. Empfohlen für die meisten Fälle.
- **Verfügbar**: Gewährleistet letztendliche Konsistenz sowohl für neue Objekte als auch für Objektaktualisierungen. Für S3-Buckets nur bei Bedarf verwenden (zum Beispiel für einen Bucket, der Log-Werte enthält, die selten gelesen werden, oder für HEAD- oder GET-Operationen auf Schlüsseln, die nicht existieren). Nicht unterstützt für S3 FabricPool Buckets.

Die Konsistenzoptionen „Read-after-new-write“ und „Available“ verwenden

Wenn eine HEAD- oder GET-Operation die Konsistenz "Read-after-new-write" verwendet, führt StorageGRID die Suche in mehreren Schritten durch, wie folgt:

- Zuerst wird das Objekt mit niedriger Konsistenz gesucht.
- Schlägt diese Suche fehl, wird die Suche beim nächsten Konsistenzwert wiederholt, bis eine Konsistenz erreicht ist, die dem Verhalten für strong-global entspricht.

Wenn eine HEAD- oder GET-Operation die Konsistenz „Read-after-new-write“ verwendet, das Objekt aber nicht existiert, erreicht die Objektsuche immer eine Konsistenz, die dem Verhalten bei strong-global entspricht. Da diese Konsistenz voraussetzt, dass an jedem Standort mehrere Kopien der Objektmetadaten verfügbar sind, kann eine hohe Anzahl von 500 Internal Server Errors auftreten, wenn zwei oder mehr Storage Nodes am selben Standort nicht verfügbar sind.

Sofern Sie keine Konsistenzgarantien wie bei Amazon S3 benötigen, können Sie diese Fehler bei HEAD- und GET-Operationen vermeiden, indem Sie die Konsistenz auf „Verfügbar“ setzen. Wenn eine HEAD- oder GET-Operation die Konsistenz „Verfügbar“ verwendet, bietet StorageGRID lediglich letztendliche Konsistenz. Es wird keine fehlgeschlagene Operation mit erhöhter Konsistenz wiederholt, sodass keine mehreren Kopien der Objektmetadaten erforderlich sind.

Konsistenz für API-Operationen angeben

Um die Konsistenz für eine einzelne API-Operation festzulegen, müssen die Konsistenzwerte für die Operation unterstützt werden und die Konsistenz im Anfrageheader angegeben werden. In diesem Beispiel wird die Konsistenz für eine GetObject-Operation auf „Strong-site“ gesetzt.

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

Für die PutObject- und GetObject-Operationen muss die gleiche Konsistenz verwendet werden.

Konsistenz für Bucket angeben

Um die Konsistenz für einen Bucket festzulegen, kann die StorageGRID "PUT Bucket Konsistenz" Anfrage verwendet werden. Alternativ ist dies auch "die Konsistenz eines Buckets ändern" aus dem Tenant Manager möglich.

Beim Festlegen der Konsistenz für einen Bucket ist Folgendes zu beachten:

- Durch die Festlegung der Konsistenz für einen Bucket wird bestimmt, welche Konsistenz für S3-Operationen verwendet wird, die an den Objekten im Bucket oder an der Bucket-Konfiguration durchgeführt werden. Dies wirkt sich nicht auf Operationen am Bucket selbst aus.
- Die Konsistenz für eine einzelne API-Operation hat Vorrang vor der Konsistenz für den Bucket.
- Buckets sollten im Allgemeinen die Standardkonsistenz „Read-after-new-write“ verwenden. Falls Anfragen nicht korrekt funktionieren, sollte nach Möglichkeit das Verhalten des Anwendungsclients geändert werden. Alternativ kann der Client so konfiguriert werden, dass er die Konsistenz für jede API-Anfrage einzeln festlegt. Die Konsistenz auf Bucket-Ebene sollte nur als letzte Möglichkeit festgelegt werden.

Wie Konsistenz- und ILM-Regeln interagieren und den Datenschutz beeinflussen

Sowohl die Wahl der Konsistenz als auch die ILM-Regel beeinflussen, wie Objekte geschützt werden. Diese Einstellungen können miteinander interagieren.

Die beim Speichern eines Objekts verwendete Konsistenz beeinflusst beispielsweise die anfängliche Platzierung der Objektmetadaten, während das für die ILM-Regel ausgewählte Aufnahmeverhalten die anfängliche Platzierung von Objektkopien bestimmt. Da StorageGRID sowohl auf die Metadaten als auch auf die Daten eines Objekts zugreifen muss, um Clientanfragen zu erfüllen, kann die Auswahl übereinstimmender Schutzstufen für Konsistenz und Aufnahmeverhalten einen besseren anfänglichen Datenschutz und vorhersehbarere Systemreaktionen bieten.

Die folgenden "Aufnahmeoptionen" stehen für ILM-Regeln zur Verfügung:

Dual Commit

StorageGRID erstellt umgehend temporäre Kopien des Objekts und gibt dem Client eine Erfolgsmeldung zurück. Kopien, die in der ILM-Regel angegeben sind, werden nach Möglichkeit erstellt.

Strikt

Alle in der ILM-Regel angegebenen Kopien müssen erstellt werden, bevor dem Client eine Erfolgsmeldung zurückgegeben wird.

Ausgewogen

StorageGRID versucht, beim Datenimport alle in der ILM-Regel festgelegten Kopien zu erstellen. Ist dies nicht möglich, werden Zwischenkopien erstellt und dem Client eine Erfolgsmeldung zurückgegeben. Die in

der ILM-Regel festgelegten Kopien werden, sofern möglich, erstellt.

Beispiel dafür, wie Konsistenz- und ILM-Regel miteinander interagieren können

Angenommen, Sie verfügen über ein Drei-Standort-Grid mit der folgenden ILM-Regel und der folgenden Konsistenz:

- **ILM-Regel**: Drei Objektkopien werden erstellt, eine am lokalen Standort und je eine an jedem Remote-Standort. Das strikte Aufnahmeverhalten wird verwendet.
- **Konsistenz**: Stark-global (Objektmetadaten werden sofort an mehrere Standorte verteilt).

Wenn ein Client ein Objekt im Grid speichert, erstellt StorageGRID alle drei Objektkopien und verteilt Metadaten an mehrere Standorte, bevor dem Client eine Erfolgsmeldung zurückgegeben wird.

Das Objekt ist zum Zeitpunkt der erfolgreichen Datenerfassung vollständig vor Verlust geschützt. Wenn beispielsweise der lokale Standort kurz nach der Datenerfassung ausfällt, sind Kopien sowohl der Objektdaten als auch der Objektmetadaten weiterhin an den Remote-Standorten vorhanden. Das Objekt ist von den anderen Standorten vollständig abrufbar.

Wenn stattdessen dieselbe ILM-Regel und die starke Standortkonsistenz verwendet werden, kann der Client eine Erfolgsmeldung erhalten, nachdem die Objektdaten auf die Remote-Standorte repliziert wurden, aber bevor die Objektmetadaten dort verteilt sind. In diesem Fall entspricht der Schutzgrad der Objektmetadaten nicht dem Schutzgrad der Objektdaten. Geht der lokale Standort kurz nach der Aufnahme verloren, gehen die Objektmetadaten verloren. Das Objekt kann nicht abgerufen werden.

Die Wechselbeziehung zwischen Konsistenz- und ILM-Regeln kann komplex sein. Wenden Sie sich an NetApp, falls Sie Unterstützung benötigen.

Wie StorageGRID die Objektversionierung nutzt

Sie können den Versionsstatus eines Buckets festlegen, wenn Sie mehrere Versionen jedes Objekts aufbewahren möchten. Die Aktivierung der Versionierung für einen Bucket kann vor versehentlichem Löschen von Objekten schützen und ermöglicht das Abrufen und Wiederherstellen früherer Versionen eines Objekts.

Das StorageGRID System implementiert Versionierung mit Unterstützung für die meisten Funktionen, jedoch mit einigen Einschränkungen. StorageGRID unterstützt bis zu 10.000 Versionen jedes Objekts.

Die Objektversionierung kann mit StorageGRID Information Lifecycle Management (ILM) oder mit der S3 Bucket-Lifecycle-Konfiguration kombiniert werden. Die Versionierung muss für jeden Bucket explizit aktiviert werden. Wenn die Versionierung für einen Bucket aktiviert ist, erhält jedes dem Bucket hinzugefügte Objekt eine Versions-ID, die vom StorageGRID System generiert wird.

Die Verwendung von MFA (Multi-Faktor Authentifizierung) Delete wird nicht unterstützt.

> Die Versionsverwaltung kann nur für Buckets aktiviert werden, die mit StorageGRID Version 10.3 oder höher erstellt wurden.

ILM und Versionierung

ILM-Richtlinien werden auf jede Version eines Objekts angewendet. Ein ILM-Scanprozess durchsucht kontinuierlich alle Objekte und bewertet sie anhand der aktuellen ILM-Richtlinie neu. Alle Änderungen an ILM-

Richtlinien werden auf alle zuvor erfassten Objekte angewendet. Dies schließt zuvor erfasste Versionen ein, wenn die Versionierung aktiviert ist. Die ILM-Überprüfung wendet neue ILM-Änderungen auf zuvor erfasste Objekte an.

Für S3-Objekte in versioning-fähigen Buckets ermöglicht die Versionierungsunterstützung die Erstellung von ILM-Regeln, die „Nicht-aktuelle Zeit“ als Referenzzeit verwenden (bei der Frage „Diese Regel nur auf ältere Objektversionen anwenden?“ in "Schritt 1 des Assistenten zum Erstellen einer ILM Regel" **Ja** auswählen). Wenn ein Objekt aktualisiert wird, werden die vorherigen Versionen zu nicht-aktuellen Versionen. Mit einem Filter für „Nicht-aktuelle Zeit“ lassen sich Richtlinien erstellen, die den Speicherbedarf vorheriger Objektversionen reduzieren.

> Wenn Sie eine neue Version eines Objekts mit einer Multipart-Upload-Operation hochladen, entspricht der nicht aktuelle Zeitpunkt der ursprünglichen Version dem Zeitpunkt, zu dem der Multipart-Upload für die neue Version erstellt wurde, nicht dem Zeitpunkt, zu dem der Multipart-Upload abgeschlossen wurde. In einigen Fällen kann der nicht aktuelle Zeitpunkt der ursprünglichen Version Stunden oder Tage vor dem Zeitpunkt der aktuellen Version liegen.

Verwandte Informationen

- "Wie versionierte S3-Objekte gelöscht werden"
- "ILM-Regeln und Richtlinien für versionierte S3-Objekte (Beispiel 4)".

Verwenden Sie die S3 REST-API, um StorageGRID S3 Object Lock zu konfigurieren.

Wenn die globale S3 Object Lock-Einstellung für Ihr StorageGRID System aktiviert ist, können Buckets mit aktivierter S3 Object Lock-Funktion erstellt werden. Für jeden Bucket kann eine Standardaufbewahrung oder für jede Objektversion eine Aufbewahrungseinstellung angegeben werden.

Aktivierung der S3 Object Lock für einen Bucket

Wenn die globale S3 Object Lock-Einstellung für Ihr StorageGRID System aktiviert ist, kann die S3 Object Lock-Option beim Erstellen jedes Buckets optional aktiviert werden.

S3 Object Lock ist eine permanente Einstellung, die nur beim Erstellen eines Buckets aktiviert werden kann. S3 Object Lock kann nach der Erstellung eines Buckets nicht hinzugefügt oder deaktiviert werden.

Um die S3-Objektsperre für einen Bucket zu aktivieren, kann eine der folgenden Methoden verwendet werden:

- Der Bucket wird mithilfe des Tenant Manager erstellt. Siehe "S3 Bucket erstellen".
- Der Bucket wird mit einer CreateBucket-Anfrage und dem `x-amz-bucket-object-lock-enabled` Anfrageheader erstellt. Siehe "Operationen an Buckets".

S3 Object Lock erfordert die Bucket-Versionierung, die beim Erstellen des Buckets automatisch aktiviert wird. Die Versionierung des Buckets kann nicht deaktiviert werden. Siehe "Objektversionierung".

Standardmäßige Aufbewahrungseinstellungen für einen Bucket

Wenn S3 Object Lock für einen Bucket aktiviert ist, kann optional die Standardaufbewahrung für den Bucket aktiviert und ein Standardaufbewahrungsmodus sowie eine Standardaufbewahrungsdauer festgelegt werden.

Standard-Aufbewahrungsmodus

- Im COMPLIANCE Modus:
  - Das Objekt kann erst gelöscht werden, wenn sein Aufbewahrungsdatum erreicht ist.
  - Das Aufbewahrungsdatum des Objekts kann erhöht, aber nicht verringert werden.
  - Das Aufbewahrungsdatum des Objekts kann erst nach Erreichen dieses Datums entfernt werden.
- Im GOVERNANCE Modus:
  - Benutzer mit der `s3:BypassGovernanceRetention` Berechtigung können den `x-amz-bypass-governance-retention: true` Anfrageheader verwenden, um die Aufbewahrungseinstellungen zu umgehen.
  - Diese Benutzer können eine Objektversion löschen, bevor deren Aufbewahrungsdatum erreicht ist.
  - Diese Benutzer können das Aufbewahrungsdatum eines Objekts erhöhen, verringern oder entfernen.

Standardmäßige Aufbewahrungsdauer

Für jeden Bucket kann eine Standard-Aufbewahrungsfrist in Jahren oder Tagen festgelegt werden.

Festlegen der Standardaufbewahrungsdauer für einen Bucket

Um die Standardaufbewahrungsdauer für einen Bucket festzulegen, kann eine der folgenden Methoden verwendet werden:

- Bucket-Einstellungen können im Tenant Manager verwaltet werden. Siehe "Einen S3-Bucket erstellen" und "Standardaufbewahrung für S3 Object Lock aktualisieren".
- Eine PutObjectLockConfiguration-Anfrage für den Bucket ausgeben, um den Standardmodus und die Standardanzahl von Tagen oder Jahren festzulegen.

PutObjectLockConfiguration

Die PutObjectLockConfiguration-Anfrage ermöglicht das Festlegen und Ändern des Standardaufbewahrungsmodus und der Standardaufbewahrungsdauer für einen Bucket mit aktiviertem S3 Object Lock. Auch das Entfernen zuvor konfigurierter Standardaufbewahrungseinstellungen ist möglich.

Wenn neue Objektversionen in den Bucket aufgenommen werden, wird der Standardaufbewahrungsmodus angewendet, wenn `x-amz-object-lock-mode` und `x-amz-object-lock-retain-until-date` nicht angegeben sind. Die Standardaufbewahrungsdauer wird zur Berechnung des Aufbewahrungsdatums verwendet, wenn `x-amz-object-lock-retain-until-date` nicht angegeben ist.

Wird die Standardaufbewahrungsfrist nach der Aufnahme einer Objektversion geändert, bleibt das Aufbewahrungsdatum der Objektversion gleich und wird nicht anhand der neuen Standardaufbewahrungsfrist neu berechnet.

Sie müssen über die `s3:PutBucketObjectLockConfiguration`-Berechtigung verfügen oder Root-Benutzer sein, um diesen Vorgang abzuschließen.

Der `Content-MD5`Anfrageheader muss in der PUT-Anfrage angegeben werden.

Anfragebeispiel

Dieses Beispiel aktiviert die S3 Object Lock für einen Bucket und legt den Standard-Aufbewahrungsmodus auf COMPLIANCE und die Standard-Aufbewahrungsdauer auf 6 Jahre fest.

```
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>
```

So wird die Standardaufbewahrungsdauer für einen Bucket ermittelt

${post_edited_translations.segment}

- Der Bucket ist im Tenant Manager sichtbar. Weitere Informationen unter "${post_edited_translations.segment}".
- Eine GetObjectLockConfiguration-Anfrage ausgeben.

GetObjectLockConfiguration

Die GetObjectLockConfiguration-Anfrage ermöglicht die Feststellung, ob S3 Object Lock für einen Bucket aktiviert ist und, falls dies der Fall ist, ob ein Standard-Aufbewahrungsmodus und eine Standard-Aufbewahrungsdauer für den Bucket konfiguriert sind.

Wenn neue Objektversionen in den Bucket aufgenommen werden, wird der Standardaufbewahrungsmodus angewendet, wenn `x-amz-object-lock-mode`nicht angegeben ist. Die Standardaufbewahrungsdauer wird zur Berechnung des Aufbewahrungsdatums verwendet, wenn `x-amz-object-lock-retain-until-date`nicht angegeben ist.

Sie müssen über die `s3:GetBucketObjectLockConfiguration`Berechtigung verfügen oder Kontoinhaber (root) sein, um diesen Vorgang abzuschließen.

Anfragebeispiel

```
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
```

Antwortbeispiel

```
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>
```

Festlegen von Aufbewahrungseinstellungen für ein Objekt

${post_edited_translations.segment}

Die Aufbewahrungseinstellungen auf Objektebene werden über die S3 REST API festgelegt. Die Aufbewahrungseinstellungen für ein Objekt überschreiben alle Standard-Aufbewahrungseinstellungen für den Bucket.

Für jedes Objekt lassen sich die folgenden Einstellungen angeben:

- ${post_edited_translations.segment}
- **Retain-until-date**: Ein Datum, das angibt, wie lange die Objektversion von StorageGRID aufbewahrt werden muss.
  - Im COMPLIANCE Modus kann das Objekt abgerufen werden, aber es kann nicht geändert oder

gelöscht werden, wenn das Aufbewahrungsdatum in der Zukunft liegt. Das Aufbewahrungsdatum kann verlängert werden, aber dieses Datum kann nicht verkürzt oder entfernt werden.

  - Im GOVERNANCE-Modus können Benutzer mit einer speziellen Berechtigung die Einstellung für das Aufbewahrungsdatum umgehen. Sie können eine Objektversion löschen, bevor deren Aufbewahrungszeitraum abgelaufen ist. Sie können das Aufbewahrungsdatum auch heraufsetzen, herabsetzen oder sogar entfernen.

- **Legal Hold**: Das Anwenden eines Legal Hold auf eine Objektversion sperrt dieses Objekt sofort. Beispielsweise kann es erforderlich sein, einen Legal Hold für ein Objekt einzurichten, das im Zusammenhang mit einer Untersuchung oder einem Rechtsstreit steht. Ein Legal Hold hat kein Ablaufdatum, sondern bleibt bestehen, bis er explizit entfernt wird.

  Die Legal-Hold-Einstellung für ein Objekt ist unabhängig vom Aufbewahrungsmodus und dem Retain-until-Date. Wenn eine Objektversion unter einem Legal Hold steht, kann niemand diese Version löschen.

Um beim Hinzufügen einer Objektversion zu einem Bucket S3 Object Lock Einstellungen anzugeben, muss eine "PutObject"-, "CopyObject"- oder "CreateMultipartUpload"-Anforderung gesendet werden.

${post_edited_translations.segment}

- `x-amz-object-lock-mode`, was COMPLIANCE oder GOVERNANCE sein kann (Groß-/Kleinschreibung).

  > Wenn Sie `x-amz-object-lock-mode` angeben, müssen Sie auch `x-amz-object-lock-retain-until-date` angeben.

- `x-amz-object-lock-retain-until-date`
  - Der Wert für die Aufbewahrungsfrist muss im Format `2020-08-10T21:46:00Z` angegeben werden. Sekundenbruchteile sind zulässig, jedoch werden nur 3 Dezimalstellen beibehalten (Millisekundenpräzision). Andere ISO 8601 Formate sind nicht zulässig.
  - Das Aufbewahrungsdatum muss in der Zukunft liegen.
- `x-amz-object-lock-legal-hold`

  Wenn die rechtliche Aufbewahrung aktiviert ist (Groß-/Kleinschreibung beachten), wird das Objekt unter eine rechtliche Aufbewahrung gestellt. Wenn die rechtliche Aufbewahrung deaktiviert ist, wird keine rechtliche Aufbewahrung angewendet. Jeder andere Wert führt zu einem 400 Bad Request (InvalidArgument) Fehler.

Wenn Sie einen dieser Anfrage-Header verwenden, gelten folgende Einschränkungen:

- Der `Content-MD5` Anfrageheader ist erforderlich, wenn ein `x-amz-object-lock-*` Anfrageheader in der PutObject Anfrage vorhanden ist. `Content-MD5` ist nicht erforderlich für CopyObject oder CreateMultipartUpload.
- Wenn für den Bucket die S3 Object Lock-Funktion nicht aktiviert ist und ein `x-amz-object-lock-*` Request-Header vorhanden ist, wird ein 400 Bad Request (InvalidRequest)-Fehler zurückgegeben.
- Die PutObject-Anfrage unterstützt die Verwendung von `x-amz-storage-class: REDUCED_REDUNDANCY`, um das Verhalten von AWS nachzubilden. Wenn jedoch ein Objekt in einen Bucket mit aktiviertem S3 Object Lock aufgenommen wird, führt StorageGRID immer eine Dual-Commit-Aufnahme durch.
- Eine nachfolgende GET- oder HeadObject-Versionsantwort enthält die Header `x-amz-object-lock-`

`mode`, `x-amz-object-lock-retain-until-date` und `x-amz-object-lock-legal-hold`, sofern diese konfiguriert sind und der Absender der Anfrage über die entsprechenden `s3:Get*` Berechtigungen verfügt.

Sie können den `s3:object-lock-remaining-retention-days` Richtlinienbedingungsschlüssel verwenden, um die minimalen und maximalen zulässigen Aufbewahrungsfristen für Ihre Objekte zu begrenzen.

Wie die Aufbewahrungseinstellungen für ein Objekt aktualisiert werden

Wenn Sie die Einstellungen für die rechtliche Aufbewahrung oder die Aufbewahrungsfristen einer bestehenden Objektversion aktualisieren müssen, können Sie die folgenden Objekt-Subressourcenoperationen durchführen:

- `PutObjectLegalHold`

  Wenn der neue Wert für den Legal Hold ON ist, wird das Objekt unter Legal Hold gesetzt. Wenn der Wert für den Legal Hold OFF ist, wird der Legal Hold aufgehoben.

- `PutObjectRetention`
  - Der Moduswert kann COMPLIANCE oder GOVERNANCE sein (Groß-/Kleinschreibung).
  - Der Wert für die Aufbewahrungsfrist muss im Format `2020-08-10T21:46:00Z` angegeben werden. Sekundenbruchteile sind zulässig, jedoch werden nur 3 Dezimalstellen beibehalten (Millisekundenpräzision). Andere ISO 8601 Formate sind nicht zulässig.
  - Hat eine Objektversion bereits ein Gültigkeitsdatum (Retain-Until-Datum), kann dieses nur erhöht werden. Der neue Wert muss in der Zukunft liegen.

Verwendung des GOVERNANCE Modus

Benutzer, die über die `s3:BypassGovernanceRetention` Berechtigung verfügen, können die aktiven Aufbewahrungseinstellungen eines Objekts, das den GOVERNANCE-Modus verwendet, umgehen. Jede DELETE- oder PutObjectRetention Operation muss den `x-amz-bypass-governance-retention:true` Anfrageheader enthalten. Diese Benutzer können folgende zusätzliche Operationen ausführen:

- Führen Sie DeleteObject oder DeleteObjects Operationen durch, um eine Objektversion zu löschen, bevor deren Aufbewahrungsfrist abgelaufen ist.

  Objekte, die einem Legal Hold unterliegen, können nicht gelöscht werden. Legal Hold muss deaktiviert sein.

- Durchführen von PutObjectRetention-Operationen, die den Modus einer Objektversion von GOVERNANCE in COMPLIANCE ändern, bevor der Aufbewahrungszeitraum des Objekts abgelaufen ist.

  Ein Wechsel des Modus von COMPLIANCE zu GOVERNANCE ist niemals zulässig.

- Sie können PutObjectRetention-Operationen durchführen, um den Aufbewahrungszeitraum einer Objektversion zu verlängern, zu verkürzen oder zu entfernen.

Verwandte Informationen

- "Objekte mit S3 Object Lock verwalten"
- "S3 Object Lock zur Aufbewahrung von Objekten verwenden"
- "Amazon Simple Storage Service Benutzerhandbuch: Objekte sperren"

Erfahren Sie mehr über die S3-Lebenszykluskonfiguration für StorageGRID

Sie können eine S3 Lifecycle Konfiguration erstellen, um zu steuern, wann bestimmte Objekte aus dem StorageGRID System gelöscht werden.

Das einfache Beispiel in diesem Abschnitt veranschaulicht, wie eine S3-Lifecycle-Konfiguration steuern kann, wann bestimmte Objekte aus bestimmten S3-Buckets gelöscht (abgelaufen) werden. Das Beispiel in diesem Abschnitt dient nur zur Veranschaulichung. Vollständige Informationen zur Erstellung von S3-Lifecycle-Konfigurationen finden sich "Amazon Simple Storage Service Benutzerhandbuch: Objektlebenszyklusverwaltung". Es ist zu beachten, dass StorageGRID nur Expiration-Aktionen unterstützt und keine Transition-Aktionen.

Was eine Lebenszykluskonfiguration ist

Eine Lebenszykluskonfiguration ist ein Satz von Regeln, die auf die Objekte in bestimmten S3-Buckets angewendet werden. Jede Regel legt fest, welche Objekte betroffen sind und wann diese Objekte ablaufen (an einem bestimmten Datum oder nach einer bestimmten Anzahl von Tagen).

StorageGRID unterstützt bis zu 1.000 Lebenszyklusregeln in einer Lebenszykluskonfiguration. Jede Regel kann die folgenden XML-Elemente enthalten:

- Ablauf: Ein Objekt wird gelöscht, wenn ein bestimmtes Datum erreicht ist oder eine bestimmte Anzahl von Tagen ab dem Zeitpunkt der Erfassung des Objekts vergangen ist.
- NoncurrentVersionExpiration: Ein Objekt wird gelöscht, wenn eine bestimmte Anzahl von Tagen erreicht ist, beginnend ab dem Zeitpunkt, an dem das Objekt nicht mehr aktuell ist.
- Filter (Präfix, Tag)
- Status
- ID

Jedes Objekt unterliegt den Aufbewahrungseinstellungen entweder eines S3-Bucket-Lebenszyklus oder einer ILM-Richtlinie. Wenn ein S3-Bucket-Lebenszyklus konfiguriert ist, überschreiben die Ablaufaktionen des Lebenszyklus die ILM-Richtlinie für Objekte, die dem Bucket-Lebenszyklusfilter entsprechen. Objekte, die dem Bucket-Lebenszyklusfilter nicht entsprechen, verwenden die Aufbewahrungseinstellungen der ILM-Richtlinie. Wenn ein Objekt einem Bucket-Lebenszyklusfilter entspricht und keine Ablaufaktionen explizit angegeben sind, werden die Aufbewahrungseinstellungen der ILM-Richtlinie nicht verwendet und es gilt, dass Objektversionen unbegrenzt aufbewahrt werden. Siehe "Beispielprioritäten für den S3 Bucket-Lebenszyklus und die ILM-Richtlinie".

Daher kann ein Objekt aus dem Grid entfernt werden, obwohl die Platzierungsanweisungen in einer ILM-Regel weiterhin für das Objekt gelten. Oder ein Objekt kann im Grid verbleiben, selbst nachdem alle ILM-Platzierungsanweisungen für das Objekt abgelaufen sind. Weitere Informationen finden sich unter "Wie ILM während der gesamten Lebensdauer eines Objekts funktioniert".

> Die Bucket-Lebenszykluskonfiguration kann für Buckets verwendet werden, bei denen S3 Object Lock aktiviert ist, aber für ältere Compliant Buckets wird die Bucket-Lebenszykluskonfiguration nicht unterstützt.

StorageGRID unterstützt die Verwendung der folgenden Bucket-Operationen zur Verwaltung von Lebenszykluskonfigurationen:

- DeleteBucketLifecycle
- GetBucketLifecycleConfiguration
- PutBucketLifecycleConfiguration

Lebenszykluskonfiguration erstellen

Als ersten Schritt beim Erstellen einer Lebenszykluskonfiguration wird eine JSON-Datei erstellt, die eine oder mehrere Regeln enthält. Diese JSON-Datei enthält beispielsweise drei Regeln:

1. Regel 1 gilt nur für Objekte, die mit dem Präfix `category1/` übereinstimmen und die einen `key2` Wert von `tag2` haben. Der `Expiration` Parameter gibt an, dass Objekte, die dem Filter entsprechen, am 22. August 2020 um Mitternacht ablaufen.
2. Regel 2 gilt nur für Objekte, die dem Präfix `category2/` entsprechen. Der `Expiration` Parameter legt fest, dass Objekte, die dem Filter entsprechen, 100 Tage nach ihrer Aufnahme ablaufen.

   > Regeln, die eine Anzahl von Tagen angeben, beziehen sich auf den Zeitpunkt der Objektaufnahme. Wenn das aktuelle Datum den Aufnahmezeitpunkt plus die Anzahl der Tage überschreitet, können einige Objekte aus dem Bucket entfernt werden, sobald die Lifecycle-Konfiguration angewendet wird.

3. Regel 3 gilt nur für Objekte, die dem Präfix `category3/` entsprechen. Der `Expiration` Parameter gibt an, dass alle nicht aktuellen Versionen entsprechender Objekte 50 Tage nach dem Übergang in den nicht aktuellen Status ablaufen.

```
{
    "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"
        }
    ]
}
```

Lebenszykluskonfiguration auf Bucket anwenden

Nachdem Sie die Lifecycle-Konfigurationsdatei erstellt haben, wenden Sie diese auf einen Bucket an, indem Sie eine PutBucketLifecycleConfiguration-Anforderung senden.

Diese Anfrage wendet die Lebenszykluskonfiguration aus der Beispieldatei auf Objekte in einem Bucket mit dem Namen `testbucket` an.

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

Um zu validieren, dass eine Lebenszykluskonfiguration erfolgreich auf den Bucket angewendet wurde, kann eine GetBucketLifecycleConfiguration-Anfrage gestellt werden. Beispielsweise:

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

Eine erfolgreiche Antwort listet die soeben angewendete Lebenszykluskonfiguration auf.

Überprüfen, ob der Bucket-Lifecycle-Ablauf auf das Objekt angewendet wird

Es lässt sich feststellen, ob eine Ablaufregel in der Lebenszykluskonfiguration auf ein bestimmtes Objekt zutrifft, wenn eine PutObject-, HeadObject- oder GetObject-Anfrage gestellt wird. Wenn eine Regel zutrifft, enthält die Antwort einen `Expiration` Parameter, der angibt, wann das Objekt abläuft und welche Ablaufregel angewendet wurde.

> Da der Bucket-Lebenszyklus ILM überschreibt, ist das `expiry-date` angezeigte Datum das tatsächliche Löschdatum des Objekts. Weitere Informationen finden Sie unter "Wie die Objektretention bestimmt wird".

Beispielsweise wurde diese PutObject-Anfrage am 22. Juni 2020 gestellt und platziert ein Objekt im `testbucket` Bucket.

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

Die Erfolgsmeldung zeigt an, dass das Objekt in 100 Tagen (01. Okt 2020) abläuft und dass es mit Regel 2 der Lebenszykluskonfiguration übereinstimmt.

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

Beispielsweise wurde diese HeadObject-Anfrage verwendet, um Metadaten für dasselbe Objekt im testbucket Bucket abzurufen.

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

Die Erfolgsmeldung enthält die Metadaten des Objekts und gibt an, dass das Objekt in 100 Tagen abläuft und dass es Regel 2 entspricht.

```
{
      "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": {}
}
```

Bei Buckets mit aktivierter Versionsverwaltung gilt der `x-amz-expiration` Antwortheader nur für die aktuellen Versionen der Objekte.

Empfehlungen für die Implementierung der S3 REST API mit StorageGRID

Diese Empfehlungen sind bei der Implementierung der S3 REST API zur Verwendung mit StorageGRID zu beachten.

Empfehlungen für HEADs zu nicht existierenden Objekten

Wenn Ihre Anwendung regelmäßig prüft, ob ein Objekt unter einem Pfad existiert, unter dem Sie nicht erwarten, dass das Objekt tatsächlich existiert, sollte die Konsistenz „Verfügbar“ "Konsistenz" verwendet werden. Zum Beispiel ist die Konsistenz „Verfügbar“ zu verwenden, wenn Ihre Anwendung einen HEAD-Befehl an einen Speicherort sendet, bevor per PUT darauf geschrieben wird.

Andernfalls kann es zu einer hohen Anzahl von 500 Internal Server-Fehlern kommen, wenn die HEAD-Operation das Objekt nicht findet und zwei oder mehr Storage Nodes am selben Standort nicht verfügbar sind

oder ein entfernter Standort nicht erreichbar ist.

Sie können die Konsistenz „Verfügbar“ für jeden Bucket mit der "PUT Bucket Konsistenz" Anfrage festlegen oder die Konsistenz im Anfrageheader für eine einzelne API-Operation angeben.

Empfehlungen für Objektschlüssel

Diese Empfehlungen für Objektschlüsselnamen gelten abhängig vom Zeitpunkt der ersten Erstellung des Buckets.

Buckets, die in StorageGRID 11.4 oder früher erstellt wurden

- Verwenden Sie keine zufälligen Werte als die ersten vier Zeichen von Objektschlüsseln. Dies steht im Gegensatz zur früheren AWS-Empfehlung für Schlüsselpräfixe. Stattdessen sollten nicht zufällige, nicht eindeutige Präfixe wie `image` verwendet werden.
- Wenn Sie der früheren AWS-Empfehlung folgen, zufällige und eindeutige Zeichen in den Schlüsselpräfixen zu verwenden, sollte den Objektschlüsseln ein Verzeichnisname vorangestellt werden. Das bedeutet, folgendes Format wird verwendet:

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

  Statt dieses Formats:

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

Buckets, die in StorageGRID 11.4 oder höher erstellt wurden

Die Beschränkung von Objektschlüsselnamen zur Einhaltung von Performance-Best Practices ist nicht erforderlich. In den meisten Fällen können für die ersten vier Zeichen von Objektschlüsselnamen zufällige Werte verwendet werden.

> Eine Ausnahme bildet ein S3-Workload, der kontinuierlich alle Objekte nach kurzer Zeit entfernt. Um die Leistungseinbußen für diesen Anwendungsfall zu minimieren, sollte ein führender Teil des Schlüsselnamens alle paar tausend Objekte mit etwas wie dem Datum variiert werden. Beispielsweise schreibt ein S3-Client typischerweise 2.000 Objekte pro Sekunde und die ILM- oder Bucket-Lifecycle-Richtlinie entfernt alle Objekte nach drei Tagen. Um die Leistungseinbußen zu minimieren, könnte die Benennung der Schlüssel nach folgendem Muster erfolgen: `/mybucket/mydir/yyyymmddhhmmss-random_UUID.jpg`

Empfehlungen für „Range Reads“

Wenn die "Globale Option zum Komprimieren gespeicherter Objekte" Option aktiviert ist, sollten S3-Clientanwendungen GetObject-Operationen vermeiden, die einen Bytebereich zur Rückgabe festlegen. Diese „Bereichslese“-Operationen sind ineffizient, da StorageGRID die Objekte effektiv dekomprimieren muss, um auf die angeforderten Bytes zuzugreifen. GetObject-Operationen, die einen kleinen Bytebereich aus einem sehr großen Objekt anfordern, sind besonders ineffizient; beispielsweise ist es ineffizient, einen 10 MB großen Bereich aus einem 50 GB großen komprimierten Objekt zu lesen.

Wenn Bereiche aus komprimierten Objekten gelesen werden, kann die Clientanfrage eine Zeitüberschreitung aufweisen.

> Wenn Objekte komprimiert werden müssen und die Clientanwendung Bereichslesevorgänge verwenden muss, sollte das Zeitüberschreitung für die Anwendung erhöht werden.