Protecting Amazon S3 Against Object Deletion

December 2016

Notices

Contents

Abstract

This whitepaper describes how to configure Amazon Simple Storage Service (S3) to recover from accidental deletion of objects. The document describes Amazon S3 advanced features that enable versioning, delete markers, and lifecycle rules to recover from deletion of objects in an Amazon S3 bucket.

Introduction

Amazon Simple Storage Service (S3) provides 99.999999999% of durability for objects stored within a given region. This durability enables a high-level of protection against data loss in the event of hardware failure or data corruption. Although Amazon S3 is highly resilient and reliable, it does not natively protect against accidental data loss or malicious data destruction caused by authorized users. However, it is possible to leverage advanced S3 features, such as versioning, to provide protection from these types of events.

This document describes a specific configuration that can increase protection and recoverability of objects within S3 against accidental destruction, such as human error or software bugs.

Amazon S3 Advanced Features

To effectively and efficiently use the advanced capabilities of Amazon S3 to meet your custom protection needs, it's important to understand some of the unique S3 concepts and features that you can use to provide protection of objects and object management.

Amazon S3 Bucket Object Versioning

Object versioning is a means of keeping multiple variants of an object in the same Amazon S3 bucket. Versioning provides the ability to recover from both unintended user actions and application failures. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. For example, Figure 1 shows one bucket having two objects with the same key, but different version IDs, such as photo.gif (version 111111) and photo.gif (version 121212).

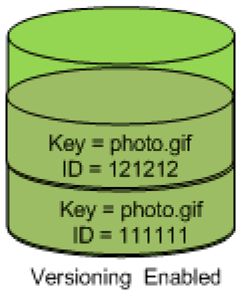


Figure 1: Example of two objects with same key

For more information, see
http://docs.aws.amazon.com/AmazonS3/latest/dev/Versioning.html.

Delete Markers

A *delete marker* is a placeholder (marker) for a versioned object named in a simple DELETE request. Because the object is in a versioning-enabled bucket, the object is not deleted. The delete marker makes Amazon S3 behave as if the object has been deleted. Objects that are deleted in this way or that are overwritten are considered non-current.

A delete marker has a key name (or key) and version ID like any other object. However, a delete marker differs from other objects in the following ways:

- It does not have data associated with it.
- It is not associated with an access control list (ACL) value.
- It does not retrieve anything from a GET request because it has no data; you get a 404 error.
- The only operation you can use on a delete marker is DELETE, and only the bucket owner can issue such a request.

Delete markers accrue a nominal charge for storage in Amazon S3. The storage size of a delete marker is equal to the size of the key name of the delete marker. A key name is a sequence of Unicode characters. The UTF-8 encoding adds from 1 to 4 bytes of storage to your bucket for each character in the name. For more information about key names, see Object Keys. For information about deleting a delete marker, see Removing Delete Markers.

For more information about delete markers in general, see
http://docs.aws.amazon.com/AmazonS3/latest/dev/DeleteMarker.html.

Lifecycle Policies

With Amazon S3, you can set lifecycle policies that define what happens with objects over their expected lifetime. Lifecycle configuration enables you to simplify the lifecycle management of your objects, such as automated transition of less-frequently accessed objects to low-cost storage alternatives, or enabling scheduled deletions.

You manage an object's lifecycle by completing a lifecycle configuration. The configuration defines how Amazon S3 manages objects during their lifetime. You can configure as many as 1000 lifecycle rules per bucket. Lifecycle policies can be applied to either current or non-current versions of objects. For the example provided in this whitepaper, we configure a lifecycle policy on non-current objects to limit how long deleted objects will be kept. This balances storage costs and the period of time accidentally deleted objects will be recoverable for.

For more information, see http://docs.aws.amazon.com/AmazonS3/latest/dev/object-lifecycle-mgmt.html.

Server Access Logging

To track requests for access to your bucket, you can enable access logging. Each access log record provides details about a single access request, such as the requester, bucket name, request time, request action, response status, and error code, if any. Access log information can be useful in security and access audits. It can also help you learn about your customer base and understand your Amazon S3 bill.

For more information see http://docs.aws.amazon.com/AmazonS3/latest/dev/ServerLogs.html.

Creating a "Recycle Bin" for S3 (Leveraging Versioning & Lifecycle Rules)

It is possible to combine both object versioning and lifecycle rules together to create a time-limited protection solution against accidental deletion. This approach can be compared to the "Trash" or "Recycle Bin" concept in modern desktop operating systems.

Once enabled, object versioning will keep copies of historic objects (both deleted and replaced). Once configured, only delete requests that include both the object's key name and version ID can be used to permanently delete historic objects. Additionally, to reduce ongoing consumption, a lifecycle rule can be created to delete non-current object versions after a grace period defined by the bucket owner.

The length of the lifecycle rule on the non-current versions of objects should be well thought through to balance between the incremental cost of keeping data in S3 (versions) versus the impact of data loss in the event data is deleted. Versioned objects are priced the same as standard objects.

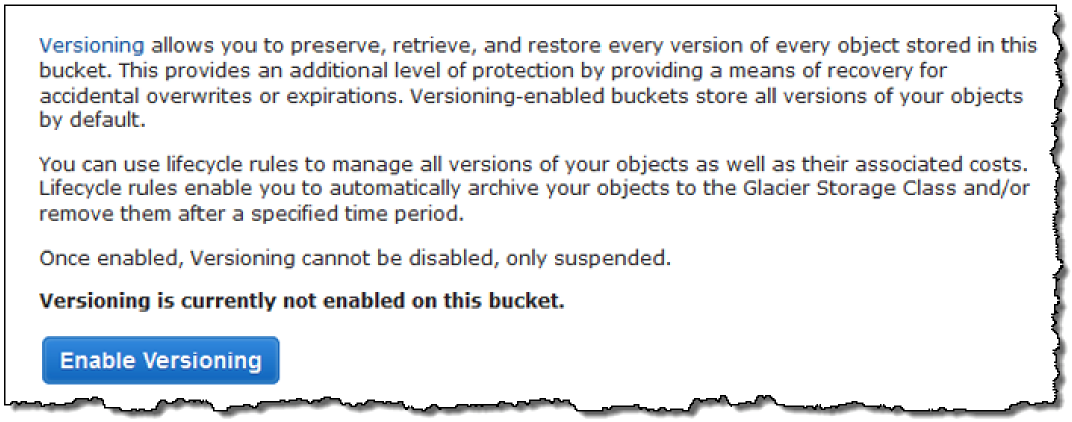
It is important to consider the expected rate of object churn when forecasting costs and ultimately choosing a grace period for the lifecycle rule. In environments where objects are frequently deleted or frequently replaced (overwritten), the amount of versioned objects stored will be high. Conversely, in environments where objects are rarely replaced or deleted, the amount of versioned objects will be lower, and therefore extending the grace period will have a negligible impact.

Creating a Recycle Bin with a Seven-Day Retention

The steps that follow walk you through an example of how to create an efficient "Recycle Bin" that only keeps deleted and overwritten objects (non-current versions) for seven days after deletion. This configuration allows the Bucket owner seven days to discover and restore any accidentally deleted objects within a given bucket.

Enable versioning on a bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, click the details icon on the left of the bucket name and then click **Properties** to display bucket properties.
3. In the **Properties** pane, click **Versioning** and then click **Enable Versioning**.



Versioning allows you to preserve, retrieve, and restore every version of every object stored in this bucket. This provides an additional level of protection by providing a means of recovery for accidental overwrites or expirations. Versioning-enabled buckets store all versions of your objects by default.

You can use lifecycle rules to manage all versions of your objects as well as their associated costs. Lifecycle rules enable you to automatically archive your objects to the Glacier Storage Class and/or remove them after a specified time period.

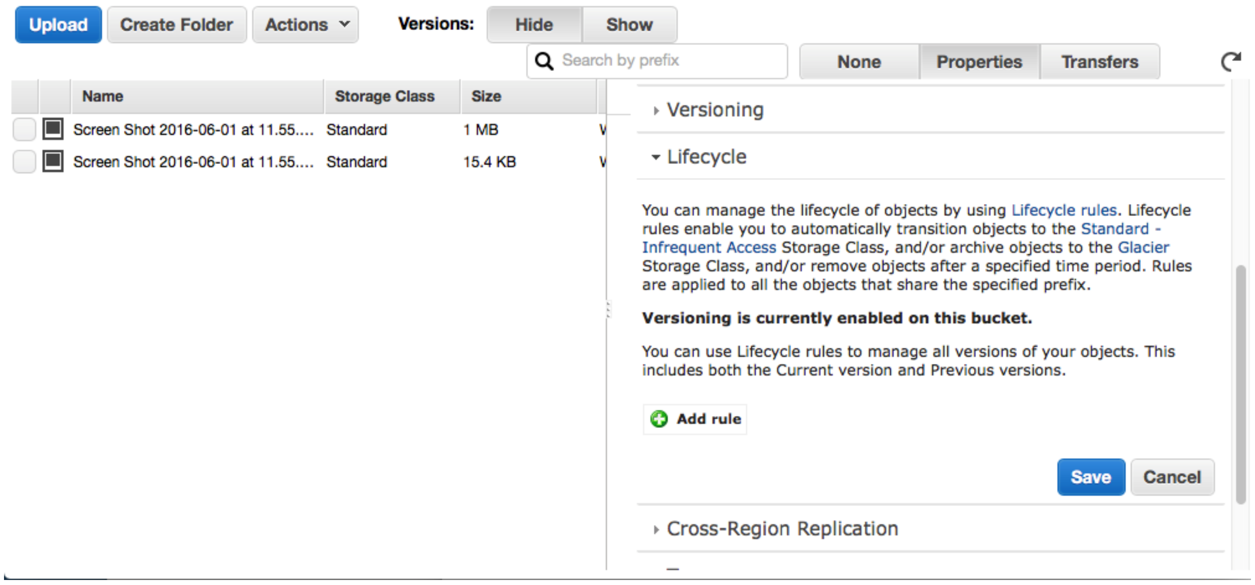
Once enabled, Versioning cannot be disabled, only suspended.

Versioning is currently not enabled on this bucket.

Create a lifecycle rule to delete versioned objects after grace period

1. In the **Buckets** list, click the details icon on the left of the bucket name and then click **Properties** to display bucket properties.

2. Expand **Lifecycle** and click **Add Rule**.



Name	Storage Class	Size
Screen Shot 2016-06-01 at 11.55....	Standard	1 MB
Screen Shot 2016-06-01 at 11.55....	Standard	15.4 KB

You can manage the lifecycle of objects by using Lifecycle rules. Lifecycle rules enable you to automatically transition objects to the Standard - Infrequent Access Storage Class, and/or archive objects to the Glacier Storage Class, and/or remove objects after a specified time period. Rules are applied to all the objects that share the specified prefix.

Versioning is currently enabled on this bucket.

You can use Lifecycle rules to manage all versions of your objects. This includes both the Current version and Previous versions.

3. In **Step 1: Choose Rule Target**, select **Whole Bucket**.

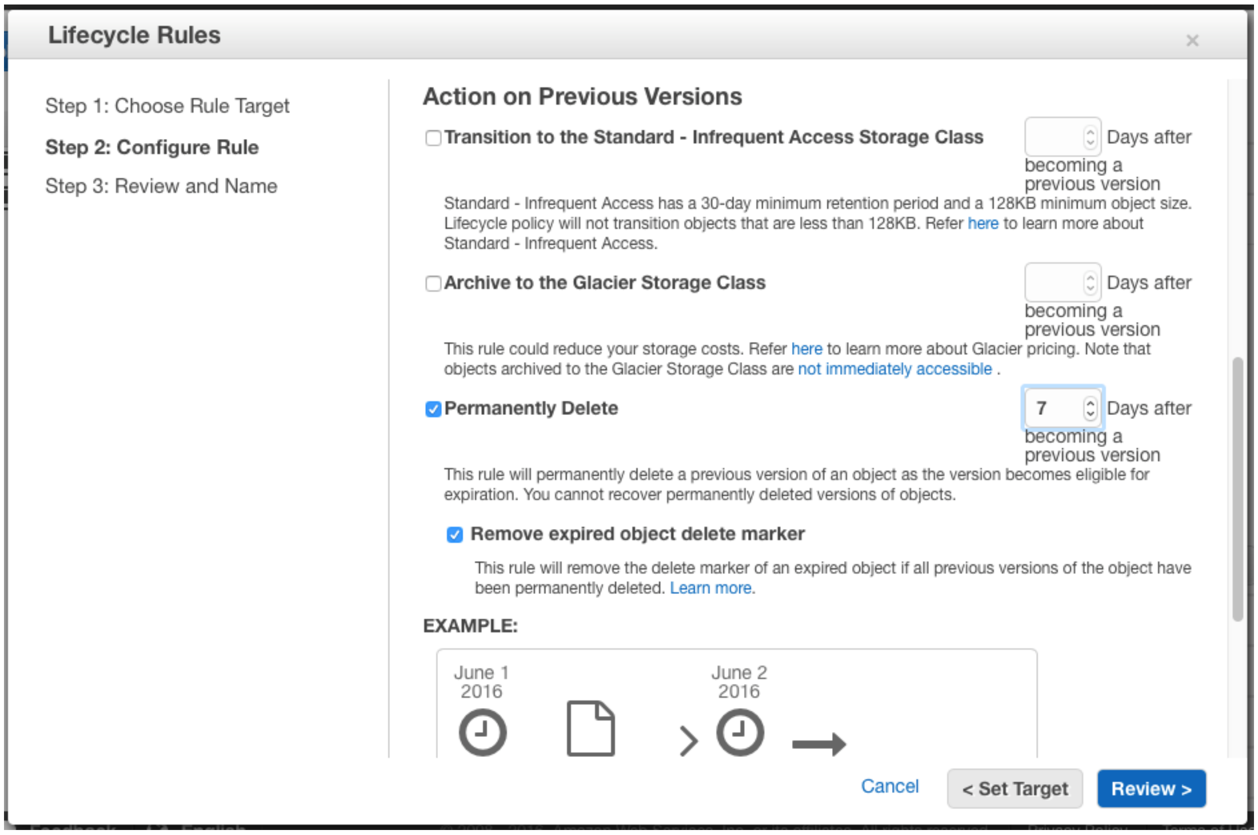
4. In **Step 2: Configure Rule**, scroll down to **Action on Previous Versions**.

5. Select the **Permanently Delete** option and enter the number of days that fit your “grace period” requirement. In this example, it’s 7 days.

6. Select the **Remove expired object delete markers** option.

Important Note: Step 6 will remove any delete markers that are associated with objects that no longer exist as a current or previous version within the S3 bucket. This is a very important step in environments that have high object churn because not removing markers can create LIST degradation over time.

7. Click the **Review** button.



Lifecycle Rules

Step 1: Choose Rule Target
Step 2: Configure Rule
Step 3: Review and Name

Action on Previous Versions

☐ Transition to the Standard - Infrequent Access Storage Class — Days after becoming a previous version

Standard - Infrequent Access has a 30-day minimum retention period and a 128KB minimum object size. Lifecycle policy will not transition objects that are less than 128KB. Refer here to learn more about Standard - Infrequent Access.

☐ Archive to the Glacier Storage Class — Days after becoming a previous version

This rule could reduce your storage costs. Refer here to learn more about Glacier pricing. Note that objects archived to the Glacier Storage Class are not immediately accessible .

☑ Permanently Delete — 7 Days after becoming a previous version

This rule will permanently delete a previous version of an object as the version becomes eligible for expiration. You cannot recover permanently deleted versions of objects.

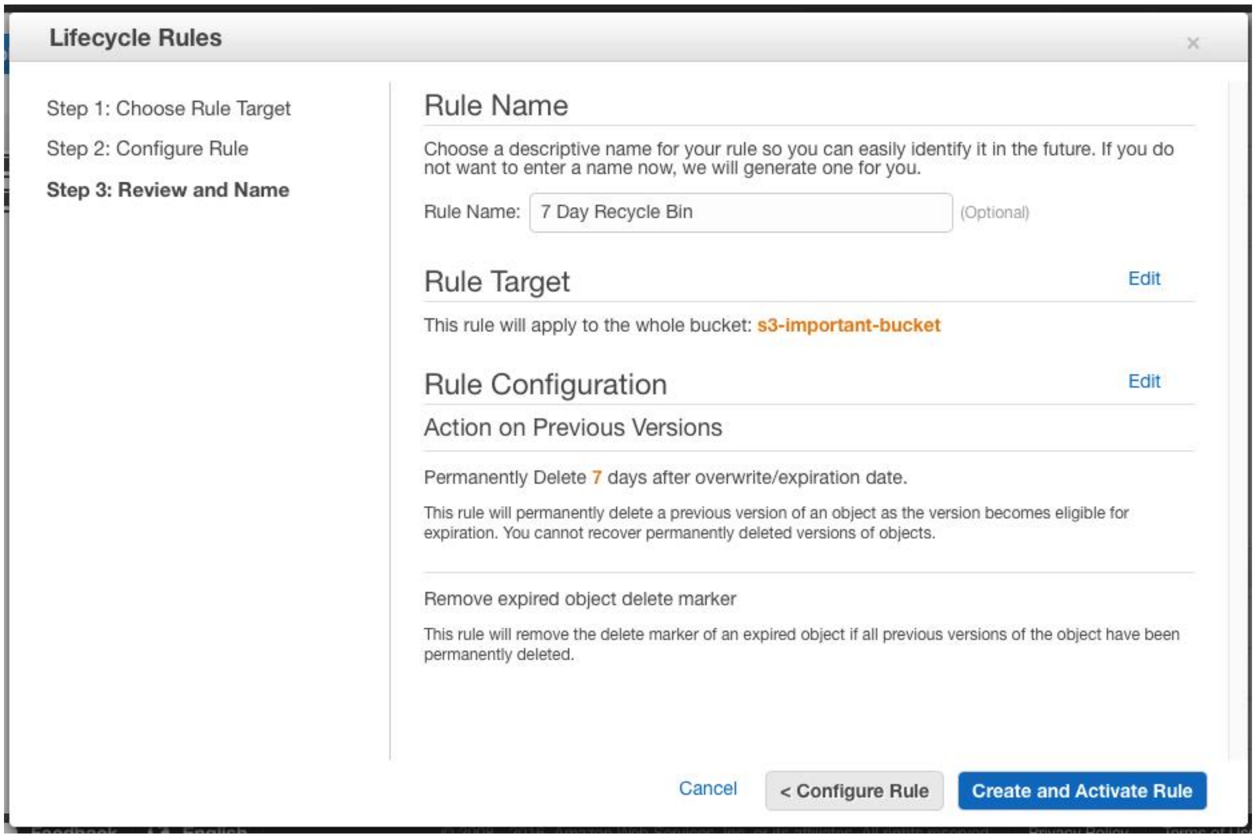
☑ Remove expired object delete marker

This rule will remove the delete marker of an expired object if all previous versions of the object have been permanently deleted. Learn more.

EXAMPLE:

8. In the Rule Name box, enter a rule name that is descriptive. In this example the name is “7 Day Recycle Bin.”

9. Review the summary of the Rule and click **Create and Activate Rule** to finish.



Rule Name

Choose a descriptive name for your rule so you can easily identify it in the future. If you do not want to enter a name now, we will generate one for you.

Rule Name: 7 Day Recycle Bin (Optional)

Rule Target

This rule will apply to the whole bucket: s3-important-bucket

Rule Configuration

Action on Previous Versions

Permanently Delete 7 days after overwrite/expiration date.

This rule will permanently delete a previous version of an object as the version becomes eligible for expiration. You cannot recover permanently deleted versions of objects.

Remove expired object delete marker

This rule will remove the delete marker of an expired object if all previous versions of the object have been permanently deleted.

Limits of Protection

The example in this whitepaper is designed to provide recoverability in the event accidental console-based deletion or API simple deletions. The configuration however, does not protect against a few scenarios that are out of scope, including:

- Deletion of a whole S3 bucket using the root AWS account that owns the bucket. If this level of protection is required, then we recommend using S3 cross-region replication to a separately managed AWS account. For more information, see Cross Region Replication Walkthrough for Different Accounts.
- Automated or manual deletion that includes versioned objects (by referencing version ID) in the deletion operation. It is recommended that version ID not be used in API-based automated tasks to protect against this scenario.

Logging for Analysis of Data Loss

Enabling access logging on a given S3 bucket gives you the ability to capture detailed transaction information on individual delete operations. This additional information can be helpful when you analyze how accidental deletions occur. Logs can be stored within a given bucket or sent to an alternative bucket. The transaction information included within a log is time, requester, remote IP address, and user-agent.

For more information, see
http://docs.aws.amazon.com/AmazonS3/latest/dev/ServerLogs.html.

Enhancing Long-Term Protection

This whitepaper provides just one example of how to create additional short-term protection for data that resides in Amazon S3. If long-term retention is required, more detailed configurations can be used that leverage S3 Infrequent Access or Amazon Glacier as the location for versioned objects. If you use these services, your capacity costs can be reduced. However, you should consider transition costs, restore costs, and retrieval times, as well. For more information, see http://aws.amazon.com/s3/pricing/.

To provide additional protection against unauthorized access, multi-factor authentication (MFA) Delete can be enabled. MFA Delete requires secondary authentication to take place before objects can be permanently deleted from an Amazon S3 bucket. For more information, see
http://docs.aws.amazon.com/AmazonS3/latest/dev/UsingMFADelete.html.

Conclusion

The configuration defined in this whitepaper can greatly assist in the protection of objects in an Amazon S3 bucket that are overwritten or deleted. In the event that accidental deletion takes place, the ability to restore objects will still be available for seven days after the simple delete operations occur. Considering that the configuration discussed in this whitepaper will only have an incremental additional storage cost, it is a good fit for most data sets that include business-critical data that must be protected against accidental deletion.

Contributors

The following individual contributed to this document:

- Peter Levett, AWS Solutions Architect, Storage