To support common use cases like setting a Time to Live (TTL) for objects, retaining noncurrent versions of objects, or "downgrading" storage classes of objects to help manage costs, Cloud Storage offers the Object Lifecycle Management feature. This page describes the feature as well as the options available when using it. To learn how to enable Object Lifecycle Management, and for examples of lifecycle policies, see Managing Lifecycles (/storage/docs/managing-lifecycles).

You can assign a lifecycle management configuration to a bucket. The configuration contains a set of rules which apply to current and future objects in the bucket. When an object meets the criteria of one of the rules, Cloud Storage automatically performs a specified action on the object. Here are some example use cases:

- Downgrade the storage class of objects older than 365 days to Coldline Storage.
- Delete objects created before January 1, 2013.
- Keep only the 3 most recent versions of each object in a bucket with versioning enabled.

Each lifecycle management configuration contains a set of rules. When defining a rule, you can specify any set of conditions for any action. If you specify multiple conditions in a rule, an object has to match *all* of the conditions for the action to be taken. If you specify multiple rules that contain the same action, the action is taken when an object matches the condition(s) in *any* of the rules. Each rule should contain only one action.

If multiple rules have their conditions satisfied simultaneously for a single object, Cloud Storage performs the action associated with only one of the rules, based on the following considerations:

- The `Delete` action takes precedence over any `SetStorageClass` action.

- The `SetStorageClass` action that switches the object to the storage class with the lowest at-rest storage pricing takes precedence.

Once an action occurs, the object is re-evaluated before any additional actions are taken.

So, for example, if you have one rule that deletes an object and another rule that changes the object's storage class, but both rules use the exact same condition, the delete action always occurs when the condition is met. If you have one rule that changes the object's class to Nearline Storage and another rule that changes the object's class to Coldline Storage, but both rules use the exact same condition, the object's class always changes to Coldline Storage when the condition is met.

The following actions are supported for a lifecycle rule:

- **Delete**: Delete objects.

Warning: Once an object is deleted, it cannot be undeleted. Take care in setting up your lifecycle rules so that you do not cause more data to be deleted than you intend. You should test your lifecycle rules on development data before applying to production and observe the expiration time metadata (#expirationtime) to ensure your rule has the effect you intend.

Exception: In buckets with object versioning (/storage/docs/object-versioning) enabled, deleting the live version of an object creates a noncurrent version, while deleting a noncurrent version deletes that version permanently.

- **SetStorageClass**: Change the storage class (/storage/docs/storage-classes) of objects. This action supports the following storage class transitions:

Original storage class	New storage class
Durable Reduced Availability (DRA) Storage	Nearline Storage Coldline Storage Archive Storage Multi-Regional Storage/Regional Storage[1]

Original storage class	New storage class
Multi-Regional Storage	Nearline Storage Coldline Storage Archive Storage
Regional Storage	Nearline Storage Coldline Storage Archive Storage
Standard Storage	Nearline Storage Coldline Storage Archive Storage
Nearline Storage	Coldline Storage Archive Storage
Coldline Storage	Archive Storage

[1] For buckets in a region (/storage/docs/locations#key-concepts), the new storage class cannot be Multi-Regional Storage. For buckets in a multi-region or dual-region, the new storage class cannot be Regional Storage.

★ **Note:** Each **`SetStorageClass`** action counts as a Class A operation charge (/storage/pricing#operations-pricing). If the destination storage class is Archive Storage, charges are determined by the rate for Archive Storage. Otherwise, charges are determined by the rate for the original storage class of the object. For example, changing 1,000 objects from Standard Storage to Coldline Storage counts as 1,000 Class A operations and is billed at the Class A operations rate for Standard Storage.

The following conditions are supported for a lifecycle rule:

- **Age**: This condition is satisfied when an object reaches the specified age (in days). Age is measured from the object's creation time. For example, if an object's creation time is 2019/01/10 10:00 UTC and the **`Age`** condition is 10 days, then the condition is satisfied for the object on and after 2019/01/20 10:00 UTC. This is true even if the object becomes noncurrent through object versioning sometime after its creation.

- **CreatedBefore**: This condition is satisfied when an object is created before midnight of the specified date in UTC.
- **IsLive**: This condition is typically only used in conjunction with object versioning (/storage/docs/object-versioning). When set to `false`, this condition is satisfied for any noncurrent version of an object. When set to `true`, this condition is satisfied for the live version of an object. If you don't use object versioning, all your objects are considered live and match when **IsLive** is `true`.
- **MatchesStorageClass**: This condition is satisfied when an object in the bucket is stored as the specified storage class. You can use the following values for `MatchesStorageClass`: `STANDARD`, `NEARLINE`, `COLDLINE`, `ARCHIVE`, `MULTI_REGIONAL`, `REGIONAL`, and `DURABLE_REDUCED_AVAILABILITY`.

Caution: Do not use **`MULTI_REGIONAL`** in **`MatchesStorageClass`** if the bucket is located in a region (/storage/docs/locations#key-concepts), and do not use **`REGIONAL`** in **`MatchesStorageClass`** if the bucket is located in a multi-region or dual-region. Doing so results in an error when you attempt to set a lifecycle management configuration (/storage/docs/managing-lifecycles#enable).

Generally, if you intend to use the `MatchesStorageClass` condition on Standard Storage objects, you should also include the following:

- If the bucket is in a region (/storage/docs/locations#key-concepts), include `REGIONAL` and `DURABLE_REDUCED_AVAILABILITY` in the condition.
- If the bucket is in a multi-region or dual-region, include `MULTI_REGIONAL` and `DURABLE_REDUCED_AVAILABILITY` in the condition.

Including these additional classes ensures the lifecycle rule covers older objects in your buckets which may be set to legacy storage classes.

- **NumberOfNewerVersions**: This condition is typically only used in conjunction with object versioning (/storage/docs/object-versioning). If the value of this condition is set to *N*, an object version satisfies the condition when there are at least *N* versions (including the live version) newer than it. For a live object version, the number of newer versions is considered to be 0. For the most recent noncurrent version, the number of newer versions is 1 (or 0 if there is no live object version), and so on.

★ **Important:** When specifying this condition in a .json configuration file, you must use `numNewerVersions` instead of `NumberOfNewerVersions`.

All conditions are optional, but at least one condition is required. If you attempt to set an invalid lifecycle configuration, such as by using an action or condition that does not exist, you receive a `400 Bad request` error response, and any existing lifecycle configuration remains in place.

See Managing Object Lifecycles (/storage/docs/managing-lifecycles) for examples of using lifecycle configurations. For the general format of a lifecycle configuration file, see the bucket resource representation for JSON (/storage/docs/json_api/v1/buckets#resource-representations) or the lifecycle configuration format for XML (/storage/docs/xml-api/put-bucket-lifecycle#request_body_elements).

- Cloud Storage regularly inspects all the objects in a bucket for which Object Lifecycle Management is configured and performs all actions applicable according to the bucket's rules. Cloud Storage performs an action asynchronously, so there can be a lag between when the conditions are satisfied and when the action is taken.

  For example, if an object meets the conditions for deletion, the object might not be deleted right away. Therefore, you see the object until the lifecycle action is executed on the object. Applicable charges (/storage/pricing) still apply while the object exists, with one exception: at-rest storage costs are waived if the object is subject to a `Delete` action because of a rule that has only an `age` condition.

! **Caution:** Because there may be a lag between when conditions are satisfied and when the action is taken, your applications should not rely on lifecycle actions occurring within a certain amount of time after a lifecycle condition is met.

- Updates to your lifecycle configuration may take up to 24 hours to go into effect. This means that when you change your lifecycle configuration, Object Lifecycle Management may still perform actions based on the old configuration for up to 24 hours.

  For example, if you change an `Age` condition from 10 days to 20 days, an object that is 11 days old could be deleted by Object Lifecycle Management up to 24 hours later, due to the

criteria of the old configuration.

- An object lifecycle `Delete` action will not take effect on an object while the object either has an object hold (/storage/docs/bucket-lock#object-holds) placed on it or a retention policy (/storage/docs/bucket-lock#retention-policy) that it has not yet fulfilled. Any `Delete` action that would occur while an object had a hold or retention policy restriction instead occurs after the restrictions no longer apply to the object.
- An object lifecycle `SetStorageClass` action is not affected by the existence of object holds or retention policies.

Object Lifecycle Management does not rewrite an object when changing its storage class. This means that when an object is transitioned to Nearline Storage, Coldline Storage, or Archive Storage using the `SetStorageClass` feature, any subsequent early deletion and associated charges (/storage/pricing#archival-pricing) are based on the original creation time of the object, regardless of when the storage class changed.

For example, say you upload an object as Standard Storage, and 20 days later your lifecycle configuration changes the storage class of the object to Nearline Storage. If you then delete the object immediately, there is a 10-day early deletion charge, since the object existed for 20 days. If you delete the object 10 days after changing its storage class to Nearline Storage, there is no early deletion charge, since the object existed for 30 days.

In comparison, say you upload an object as Standard Storage and 20 days later change the storage class using a rewrite (/storage/docs/changing-storage-classes) (again to Nearline Storage). If you then delete the object immediately afterward, you would incur a full 30-day early deletion charge, since the rewriting time becomes the new creation time. Similarly, if you waited 10 days after the rewrite to delete the object, you would incur a 20-day early deletion charge.

If a `Delete` action is specified for a bucket with the `Age` condition (and no `NumberOfNewerVersions` condition), then some objects may be tagged with expiration time metadata. An object's expiration time indicates the time at which the object becomes (or became) eligible for deletion by Object Lifecycle Management. The expiration time may change as the bucket's lifecycle configuration or retention policy (/storage/docs/bucket-lock) change.

Note that the absence of expiration time metadata does not necessarily mean the object will not be deleted, but rather that not enough information is available to determine when or if it will be deleted. For example, if the object creation time is 2013/01/10 10:00 UTC and the `Age` condition is set to 10 days, then the object expiration time is 2013/01/20 10:00 UTC. However, the expiration time will not be available for the object if:

- The `NumberOfNewerVersions` condition is also specified. In this case, older versions of the object may still be deleted if new versions are added.
- The `CreatedBefore` condition is also specified and set to “2013-01-01”, because the object doesn’t satisfy this condition.
- The object is under a hold (/storage/docs/bucket-lock#object-holds), because Cloud Storage cannot know when the hold will be removed.

You are not charged for storage after the object expiration time even if the object is not deleted immediately. You can continue to access the object before it is deleted and are responsible for other charges (request, network bandwidth). If the expiration time is not available for an object, the object is charged for storage until the time it is deleted.

You can access an object's expiration time in its metadata if it is available. The REST API returns the object's expiration time in the `x-goog-expiration` response header.

When working with expiration times, keep in mind the following:

- If the bucket has a retention policy (/storage/docs/bucket-lock), the expiration time is the later of the Object Lifecycle Management age condition and the time the object satisfies the retention period specified by the retention policy.
- If there are multiple conflicting expiration times applicable for an object due to different lifecycle management rules, then the earliest applicable expiration time is used.

To track the lifecycle management actions that Cloud Storage takes, use one of the following options:

- Use Cloud Storage Access Logs (/storage/docs/access-logs). This feature logs both the action and who performed the action. A value of `GCS Lifecycle Management` in the

`cs_user_agent` field of the log entry indicates the action was taken by Cloud Storage in accordance with a lifecycle configuration.

- Enable Pub/Sub Notifications for Cloud Storage (/storage/docs/pubsub-notifications) for your bucket. This feature sends notifications to a Pub/Sub topic of your choice when specified actions occur. Note that this feature does not record who performed the actions.

- Enable Object Lifecycle Management (/storage/docs/managing-lifecycles#enable).