To use Cloud Storage effectively, you should understand some of the concepts on which it is built. This page provides an overview of key terms and concepts that apply to Cloud Storage.

For an introduction to using Cloud Storage, see the Quickstart Using the Console (/storage/docs/quickstart-console).

All data in Cloud Storage belongs inside a project. A project consists of a set of users, a set of APIs, and billing, authentication, and monitoring settings for those APIs. You can have one project or multiple projects.

Buckets are the basic containers that hold your data. Everything that you store in Cloud Storage must be contained in a bucket. You can use buckets to organize your data and control access to your data, but unlike directories and folders, you cannot nest buckets. Because there are limits to bucket creation and deletion (/storage/quotas), you should design your storage applications to favor intensive object operations and relatively few buckets operations.

When you create a bucket (/storage/docs/creating-buckets), you specify a globally-unique name, a geographic location (/storage/docs/locations) where the bucket and its contents are stored, and a default storage class (/storage/docs/storage-classes). The default storage class you choose applies to objects added to the bucket that don't have a storage class specified explicitly.

After you create a bucket, you can still change its default storage class (/storage/docs/changing-default-storage-class), to any class supported in the bucket's location; however, you can only change the bucket name and location by deleting and re-creating the bucket (/storage/docs/moving-buckets).

Bucket names have more restrictions than object names and must be globally unique, because every bucket resides in a single Cloud Storage namespace. Also, bucket names can be used with a `CNAME` redirect, which means they need to conform to DNS naming conventions. For more information, see the bucket naming guidelines (/storage/docs/naming#requirements).

Bucket labels are key:value metadata pairs that allow you to group your buckets along with other Google Cloud resources such as virtual machine instances (/compute/docs/instances/) and persistent disks (/compute/docs/disks/#pdspecs). For example, you can use labels to create a `team` key that has values `alpha`, `beta`, and `delta`, and apply the `team:alpha`, `team:beta`, and `team:delta` labels to different buckets in order to indicate which team is associated with those buckets.

You can apply multiple labels to each bucket (/storage/docs/using-bucket-labels#add-label), with a maximum of 64 labels per bucket.

- Keys and values cannot be longer than 63 characters each.
- Keys and values can only contain lowercase letters, numeric characters, underscores, and dashes. International characters are allowed.
- Label keys must start with a lowercase letter and international characters are allowed.
- Label keys cannot be empty.

For a general example of using labels to organize your resources in billing, see Billing Export to BigQuery Query Examples (/billing/docs/how-to/bq-examples).

Objects are the individual pieces of data that you store in Cloud Storage. There is no limit on the number of objects that you can create in a bucket.

Objects have two components: *object data* and *object metadata* (/storage/docs/metadata). Object data is typically a file that you want to store in Cloud Storage. Object metadata is a collection of name-value pairs that describe various object qualities.

An object's name is treated as a piece of object metadata in Cloud Storage. Object names can contain any combination of Unicode characters (UTF-8 encoded) and must be less than 1024 bytes in length.

A common character to include in object names is a slash (/). By using slashes, you can make objects appear as though they're stored in a hierarchical structure. For example, you could name one object `/europe/france/paris.jpg` and another object `/europe/france/cannes.jpg`. When you list these objects, they appear to be in a hierarchical directory structure based on location; however, Cloud Storage sees the objects as independent with no hierarchical relationship whatsoever.

For more information, including how to rename an object, see the object naming guidelines (/storage/docs/naming#objectnames).

An object in Cloud Storage can have different *versions*: by default, when you overwrite an object, Cloud Storage deletes the old version and replaces it with a new version. When you enable Object Versioning (/storage/docs/object-versioning) on your bucket, older versions remain in your bucket when an overwrite or deletion occurs.

Each object version is uniquely identified by its *generation number*, found in the object's metadata. When Object Versioning has created an older version of an object, you can use the generation number to refer to the older version (/storage/docs/using-object-versioning#copy). This allows you to restore an overwritten object in your bucket, or permanently delete older object versions that you no longer need. Generation numbers are also used when you include preconditions in your requests (/storage/docs/generations-preconditions#_Preconditions).

A resource is an entity within Google Cloud. Each project, bucket, and object in Google Cloud is a resource, as are things such as Compute Engine instances (/compute/docs/instances/).

The use of resource names within Cloud Storage is currently limited to Pub/Sub Notifications for Cloud Storage
age/docs/pubsub-notifications) and Identity and Access Management (/storage/docs/access-control/iam).

Each resource has a unique name that identifies it, much like a filename. Buckets have a resource name in the form of `projects/_/buckets/[BUCKET_NAME]`, where `[BUCKET_NAME]` is the ID of the bucket. Objects have a resource name in the form of `projects/_/buckets/[BUCKET_NAME]/objects/[OBJECT_NAME]`, where `[OBJECT_NAME]` is the ID of the object.

A `#[NUMBER]` appended to the end of the resource name indicates a specific generation of the object. `#0` is a special identifier for the most recent version of an object. `#0` is useful to add when the name of the object ends in a string that would otherwise be interpreted as a generation number.

Data that is *geo-redundant* is stored redundantly in at least two separate geographic places separated by at least 100 miles. Objects stored in multi-regions and dual-regions (/storage/docs/bucket-locations#location-mr) are geo-redundant, regardless of their storage class.

Geo-redundancy occurs asynchronously, but all Cloud Storage data is redundant within at least one geographic place as soon as you upload it.

Geo-redundancy ensures maximum availability of your data, even in the event of large-scale disruptions, such as natural disasters. For dual-regions, geo-redundancy is achieved using two specific regions. For multi-regions, geo-redundancy is achieved using any combination of data centers within the specified multi-region, which may include data centers that are not explicitly listed as available regions.

An object's data component is completely opaque to Cloud Storage. It is just a chunk of data to Cloud Storage.

Objects are immutable, which means that an uploaded object cannot change throughout its storage lifetime. An object's storage lifetime is the time between successful object creation (upload) and successful object deletion. In practice, this means that you cannot make incremental changes to objects, such as append operations or truncate operations. However, it is possible to overwrite objects that are stored in Cloud Storage, and doing so happens atomically — until the new upload completes the old version of the object will be served to readers, and after the upload completes the new version of the object will be served to readers. So a single overwrite operation simply marks the end of one immutable object's lifetime and the beginning of a new immutable object's lifetime.

There is no limit to how frequently you can create or update different objects in a bucket. However, a single particular object can only be updated or overwritten up to once per second. For example, if you have an object `bar` in bucket `foo`, then you should only upload a new copy of `foo/bar` about once per second. Updating the same object more frequently than once per second may result in **`429 Too Many Requests`** errors.

You should retry failed requests using truncated exponential backoff (/storage/docs/exponential-backoff).

Cloud Storage uses a flat namespace to store objects. However, some tools (for example, Google Cloud Console (https://console.cloud.google.com/) and gsutil (/storage/docs/gsutil)) can work with objects as if they are stored in a virtual hierarchy, as a convenience.

There is only one Cloud Storage namespace, which means every bucket must have a unique name across the entire Cloud Storage namespace. Object names must be unique only within a given bucket.