This page provides an overview of access control lists (ACLs). To learn how to set and manage ACLs, read Create and Manage Access Control Lists (/storage/docs/access-control/create-manage-lists). To learn about other ways of controlling access to buckets and objects, read Overview of Access Control (/storage/docs/access-control/index).

In most cases, Cloud Identity and Access Management (Cloud IAM) (/storage/docs/access-control/iam) is the recommended method for controlling access to your resources. Cloud IAM and ACLs work in tandem to grant access to your buckets and objects: a user only needs permission from either Cloud IAM or an ACL to access a bucket or object.

You most likely want to use ACLs if you need to customize access to individual objects within a bucket, since Cloud IAM permissions apply to all objects within a bucket. However, you should still use Cloud IAM for any access that is common to all objects in a bucket, because this reduces the amount of micro-managing you have to do.

ɪn: Permissions can be granted either by ACLs or Cloud IAM policies. In general, permissions granted by Cloud IAM p appear in ACLs, and permissions granted by ACLs do not appear in Cloud IAM policies. The only exception is for ACL d directly on a bucket and certain bucket-level Cloud IAM policies, as described in Cloud IAM relation to ACLs age/docs/access-control/iam#acls).

An access control list (ACL) is a mechanism you can use to define who has access to your buckets and objects, as well as what level of access they have. In Cloud Storage, you apply ACLs to individual buckets and objects. Each ACL consists of one or more *entries*. An entry gives a specific user (or group) the ability to perform specific actions. Each entry consists of two pieces of information:

- A **permission**, which defines *what* actions can be performed (for example, read or write).
- A **scope** (sometimes referred to as a *grantee*), which defines *who* can perform the specified actions (for example, a specific user or group of users).

As an example, suppose you have a bucket that you want anyone to be able to access objects from, but you also want your collaborator to be able to add or remove objects from the bucket. In this case, your ACL would consist of two entries:

- In one entry, you would give `READER` permission to a scope of `allUsers`.
- In the other entry, you would give `WRITER` permission to the scope of your collaborator (there are several ways to specify this person, such as by their email).

The maximum number of ACL entries you can create for a bucket or object is 100. When the entry scope is a group or domain, it counts as one ACL entry regardless of how many users are in the group or domain.

When a user requests access to a bucket or object, the Cloud Storage system reads the bucket or object ACL and determines whether to allow or reject the access request. If the ACL grants the user permission for the requested operation, the request is allowed. If the ACL does not grant the user permission for the requested operation, the request fails and a `403 Forbidden` error is returned.

Note that while ACLs can be used to manage most actions involving buckets and objects, the ability to *create* a bucket comes from having the appropriate project permission (/storage/docs/access-control/iam).

Permissions describe *what* can be done to a given object or bucket.

Cloud Storage lets you assign the following concentric (#concentric) permissions for your buckets and objects, as shown in the following table:

	Buckets	Objects
READER	Allows a user to list a bucket's contents. Also allows a user to read bucket metadata, excluding ACLs.	Allows a user to download an object's data.
WRITER	Allows a user to list, create, overwrite, and delete objects in a bucket [1] (#note1).	N/A. You cannot apply this permission to objects.
OWNER	Gives a user **READER** and **WRITER** permissions on the bucket. Also allows a user to read and write bucket metadata, including ACLs.	Gives a user **READER** access. It also allows a user to read and write object metadata, including ACLs.

	Buckets	Objects
Default	Buckets have the predefined project-private (#predefined-project-private) ACL applied when they are created. Buckets are always owned by the **project-owners** group.	Objects have the predefined project-private (#predefined-project-private) ACL applied when they are uploaded. Objects are always owned by the original requester who uploaded the object.

[1] The following bucket metadata properties cannot be changed: `acl` (/storage/docs/json_api/v1/buckets#acl), `cors` (/storage/docs/json_api/v1/buckets#cors), `defaultObjectAcl` (/storage/docs/json_api/v1/buckets#defaultObjectAcl), `lifecycle` (/storage/docs/json_api/v1/buckets#lifecycle), `logging` (/storage/docs/json_api/v1/buckets#logging), `versioning` (/storage/docs/json_api/v1/buckets#versioning), and `website` (/storage/docs/json_api/v1/buckets#website).

You cannot grant discrete permissions for reading or writing ACLs or other metadata. To allow someone to read and you must grant them **OWNER** permission.

In this page, we generally refer to the permissions as `READER`, `WRITER`, and `OWNER`, which are how they are specified in the JSON API (/storage/docs/json_api/) and the Google Cloud Console (https://console.cloud.google.com/). If you are using the XML API (/storage/docs/xml-api/overview), the equivalent permissions are `READ`, `WRITE`, and `FULL_CONTROL`, respectively. And, when you use OAuth 2.0 authentication (/storage/docs/authentication#oauth) to authenticate tools and applications (grant permission to them) to access Google Cloud Storage API on your behalf, access is restricted by *OAuth scope* `devstorage.read_only`, `devstorage.read_write`, and `devstorage.full_control`. The following table summarizes the permissions terminology you commonly encounter:

JSON API	XML API	OAuth2 Scope
`READER`	`READ`	`https://www.googleapis.com/auth/devstorage.read_only`
`WRITER`	`WRITE`	`https://www.googleapis.com/auth/devstorage.read_write`
`OWNER`	`FULL_CONTROL`	`https://www.googleapis.com/auth/devstorage.full_control`

Scopes specify *who* it is that has a given permission.

An ACL consists of one or more entries, where each entry grants permissions to a scope. You can specify an ACL scope using any of the following entities:

Scope ("grantee")	Entity Type(s)	Example
Google account email address	User	collaborator@gmail.com
Google group email address	Group	work-group@googlegroups.com
Convenience values for projects	Project	`owners-123456789012`
G Suite domain	Domain	[USERNAME]@[YOUR_DOMAIN].com
Cloud Identity domain	Domain	[USERNAME]@[YOUR_DOMAIN].com
Special identifier for all Google account holders	User	`allAuthenticatedUsers`
Special identifier for all users	User	`allUsers`

- **Google account email address**:

  Every user who has a Google account must have a unique email address associated with that account. You can specify a scope by using any email address that is associated with a Google account, such as a gmail.com address.

  Cloud Storage remembers email addresses as they are provided in ACLs until the entries are removed or overwritten. If a user changes email addresses, you should update ACL entries to reflect these changes.

- **Google group email address**:

  Every Google group has a unique email address that is associated with the group. For example, the Cloud Storage Announce (https://groups.google.com/forum/#!aboutgroup/gs-announce) group has the following email address: gs-announce@googlegroups.com. You can find the email address that is associated with a Google group by clicking **About** on the homepage of every Google group. For more information about Google groups, see the Google groups homepage (http://groups.google.com/).

  Like Google account email addresses, Cloud Storage remembers group email addresses as they are provided in ACLs until the entries are removed or overwritten. You do not need to worry about updating Google Group email addresses, because Google Group email addresses are permanent and unlikely to change.

- **Convenience values for projects**:

  The convenience values `owners-<project-number>`, `editors-<project-number>`, and `viewers-<project-number>` represent the lists of owners, editors, and viewers of the project whose project number (/storage/docs/projects) is <project-number>.

- **G Suite or Cloud Identity**:

G Suite (https://gsuite.google.com/) and Cloud Identity (https://support.google.com/cloudidentity/answer/7319251) customers can associate their email accounts with an Internet domain name. When you do this, each email account takes the form [USERNAME]@[YOUR_DOMAIN].com. You can specify a scope by using any Internet domain name that is associated with G Suite or Cloud Identity.

- **Special identifier for all Google account holders**:

  This special scope identifier represents anyone who is authenticated with a Google account. The special scope identifier for all Google account holders is `allAuthenticatedUsers`.

- **Special identifier for all users**:

  This special scope identifier represents anyone who is on the Internet, with or without a Google account. The special scope identifier for all users is `allUsers`.

When specifying ACLs in Cloud Storage, you do not need to list multiple scopes to grant multiple permissions. Cloud Storage uses concentric permissions, so when you grant `WRITER` permission, you also grant `READER` permission, and if you grant `OWNER` permission, you also grant `READER` and `WRITER` permission.

When specifying an ACL using the Google Cloud Console, JSON API, or gsutil, you can specify multiple scopes for the same entry. The most permissive permission is the access granted to the scope. For example, if you provide two entries for a user, one with `READER` permission and one with `WRITER` permission on a bucket, the user will have `WRITER` permission on the bucket.

In the XML API, it is not possible to provide two ACL entries with the same scope. For example, granting a user `READ` permission and `WRITE` permission on a bucket results in an error. Instead, grant the user `WRITE` permission, which also grants the user `READ` permission.

A predefined or "canned" ACL is an alias for a set of specific ACL entries that you can use to quickly apply many ACL entries at once to a bucket or object.

ɪn: By applying a predefined ACL to an existing bucket or object, you completely replace the existing bucket or object ıe predefined ACL. This change might cause you to lose access to the bucket or object ACL in some cases. For exam e in the project owners group but are not the owner of an object with **`projectPrivate`** ACL, then after you apply th

ined ACL **publicRead** to the object, you lose **OWNER** permission and thus no longer can access the object ACL. If this
, you can use the Cloud IAM (/storage/docs/access-control/iam#acls) role **storage.objectAdmin** so that you hav
sion necessary to update the object's ACL and correct the change.

The table below lists predefined ACLs and shows which ACL entries are applied for each predefined ACL. When using the table below, note that:

- The project owners group has ownership of buckets in the project, and the user that creates an object has ownership of that object. If an object was created by an anonymous user, then the project owners group has ownership of the object.
- In the table, the JSON API descriptions of permissions, `OWNER`, `WRITER`, and `READER`, are used. The equivalent XML API scopes are `FULL_CONTROL`, `WRITE`, and `READ`.

JSON API	XML API/gsutil	Description
`private`	`private`	Gives the bucket or object owner **OWNER** permission for a bucket or object.
`bucketOwnerRead`	`bucket-owner-read`	Gives the object owner **OWNER** permission, and gives the bucket owner **READER** permission. **This is used only with objects**.
`bucketOwnerFullControl`	`bucket-owner-full-control`	Gives the object and bucket owners **OWNER** permission. **This is used only with objects**.
`projectPrivate`	`project-private`	Gives permission to the project team based on their roles. Anyone who is part of the team has **READER** permission. Project owners and project editors have **OWNER** permission. This is the default ACL for newly created buckets. This is also the default ACL for newly created objects unless the default object ACL (#defaultobjects) for that bucket has been changed.
`authenticatedRead`	`authenticated-read`	Gives the bucket or object owner **OWNER** permission, and gives all authenticated Google account holders **READER** permission.
`publicRead`	`public-read`	Gives the bucket or object owner **OWNER** permission, and gives all users, both authenticated and anonymous, **READER** permission. When you apply this to an object, anyone on the Internet can read the object without authenticating. When you apply this to a bucket, anyone on the Internet can list objects without authenticating. * See the note at the end of the table regarding caching.

JSON API	XML API/gsutil	Description
`publicReadWrite`	`public-read-write`	Gives the bucket owner **`OWNER`** permission, and gives all users, both authenticated and anonymous, **`READER`** and **`WRITER`** permission. This ACL applies only to buckets. When you apply this to a bucket, anyone on the Internet can list, create, overwrite and delete objects without authenticating. ★ **Important**: Setting a bucket to **`publicReadWrite`** allows anyone on the Internet to upload anything to your bucket. You are responsible for this content. * See the note at the end of the table regarding caching.

* By default, publicly readable objects are served with a `Cache-Control` header that allows the objects to be cached for 3600 seconds. If you need to ensure that updates become visible immediately, you should set the `Cache-Control` metadata (/storage/docs/viewing-editing-metadata) for the objects to `Cache-Control:private, max-age=0, no-transform`.

When buckets are created or objects are uploaded, if you do not explicitly assign an ACL to them, they are given the default ACL. You can change the default ACL given to an object; the process to do so is described in Changing default object ACLs (/storage/docs/access-control/create-manage-lists#defaultobjects). Note that when you change the default ACL, the ACLs of objects that already exist in the bucket or buckets that already exist in the project remain unchanged.

All buckets are owned by the project owners group. Project owners are granted `OWNER` permission automatically to all buckets inside their project. When you create a project, you are automatically added as a project owner.

If you create a bucket with the default bucket ACL—that is, you do not specify a predefined (#predefined-acl) ACL when you create the bucket—your bucket has the predefined `projectPrivate` ACL applied to it. The `projectPrivate` ACL gives additional permissions to project team members based on their roles. These additional permissions are defined as follows:

- **Project Viewers**

  The `projectPrivate` ACL provides project viewers with `READER` access to buckets in a project. All project team members can list objects within buckets. All project team members can also list buckets within a project, independent of bucket ACLs.

- **Project Editors**

  The `projectPrivate` ACL provides project editors with `OWNER` permissions to buckets in a project. Project editors can list a bucket's contents and create, overwrite, or delete objects in a bucket. Project editors can also list, create, and delete buckets, independent of bucket ACLs.

- **Project Owners**

  The `projectPrivate` ACL provides project owners with `OWNER` permissions. Project owners can also perform all tasks that project editors can perform, in addition to administrative tasks such as adding and removing team members or changing billing information.

Project viewers, project editors, and project owners are identified by combining their role with the associated project number. For example, in project `867489160491`, editors are identified as `project-editors-867489160491`. You can find your project number on the homepage of the Google Cloud Console (https://console.cloud.google.com/).

By default, anyone who has `OWNER` permission or `WRITER` permission on a bucket can upload objects into that bucket. When you upload an object, you can provide a predefined (#predefined-acl) ACL or not specify an ACL at all. If you don't specify an ACL, Cloud Storage applies the bucket's default object ACL to the object. Every bucket has a default object ACL, and this ACL is applied to all objects uploaded to that bucket without a predefined ACL or an ACL specified in the request (JSON API only). The initial value for the default object ACL of every bucket is `projectPrivate`.

Based on how objects are uploaded, object ACLs are applied accordingly:

- **Authenticated Uploads**

  If you make an authenticated request to upload an object and do not specify any object ACLs when you upload it, then you are listed as the owner of the object and the predefined `projectPrivate` ACL is applied to the object by default. This means:

  - You (the person who uploaded the object) are listed as the object owner. Object ownership cannot be changed by modifying ACLs. You can change object ownership only by overwriting an object.

- You (the object owner) are granted `OWNER` permission on the object. If you attempt to give less than `OWNER` permission to the owner, Cloud Storage automatically escalates the permission to `OWNER`.
- The project owners and project editors group have `OWNER` permission on the object.
- The project team members group has `READER` permission on the object.

- **Anonymous Uploads**

  If an unauthenticated (anonymous) user uploads an object, which is possible if a bucket grants the `allUsers` group `WRITER` or `OWNER` permission, then the default bucket ACLs are applied to the object as described above.

  Anonymous users cannot specify a predefined ACL during object upload.

ant: If you change the default object ACL for a bucket, the change may take time to propagate, and new objects crea
cket may still get the old default object ACL for a short period of time (see Consistency (/storage/docs/consistency))
o make sure that new objects created in the bucket will get the updated default object ACL, you should wait at least
ds between changing the default object ACL and creating new objects.

ACLs, like any other administrative settings, require active management to be effective. Before you make a bucket or object accessible to other users (/storage/docs/access-control/create-manage-lists), be sure you know who you want to share the bucket or object with and what roles you want each of those people to play. Over time, changes in project management, usage patterns, and organizational ownership may require you to modify ACL settings on buckets and objects, especially if you manage buckets and objects in a large organization or for a large group of users. As you evaluate and plan your access control settings, keep the following best practices in mind:

- **Use the principle of least privilege when granting access to your buckets and objects.**

  The principle of least privilege is a security guideline for granting privileges or rights. When you grant access based on the principle of least privilege, you grant the minimum privilege that's necessary for a user to accomplish their assigned task. For example, if you want to share a file with someone, grant them `READER` permission and not `OWNER` permission.

- **Avoid granting `OWNER` permission to people you do not know.**

  Granting `OWNER` permission allows a user to change ACLs and take control of data. You should use the `OWNER` permission only when you want to delegate administrative control over objects

and buckets.

- **Be careful how you grant permissions for anonymous users.**

  The `allUsers` and `allAuthenticatedUsers` scopes should only be used when it is acceptable for anyone on the Internet to read and analyze your data. While these scopes are useful for some applications and scenarios, it is usually not a good idea to grant all users `OWNER` permission.

- **Avoid setting ACLs that result in inaccessible objects.**

  An inaccessible object is an object that cannot be downloaded (read) and can only be deleted. This can happen when the owner of an object leaves a project without granting anyone else `OWNER` or `READER` permission on the object. To avoid this problem, you can use the `bucket-owner-read` or `bucket-owner-full- control` predefined ACLs when you or anyone else uploads objects to your buckets.

- **Be sure you delegate administrative control of your buckets.**

  By default, the project owners group is the only entity that has `OWNER` permission on a bucket when it is created. You should have at least two members in the project owners group at any given time so that if a team member leaves the group, your buckets can still be managed by the other project owners.

- **Be aware of Cloud Storage's interoperable behavior.**

  When using the XML API for interoperable access with other storage services, such as Amazon S3, the signature identifier determines the ACL syntax. For example, if the tool or library you are using makes a request to Cloud Storage to retrieve ACLs and the request uses another storage provider's signature identifier, then Cloud Storage returns an XML document that uses the corresponding storage provider's ACL syntax. If the tool or library you are using makes a request to Cloud Storage to apply ACLs and the request uses another storage provider's signature identifier, then Cloud Storage expects to receive an XML document that uses the corresponding storage provider's ACL syntax.

  For more information about using the XML API for interoperable access with Amazon S3, see Migrating from Amazon S3 to Google Cloud Storage (/storage/docs/migrating).

Cloud Storage helps you adhere to these best practices by enforcing some ACL modification rules, which prevent you from setting ACLs that make data inaccessible:

- **You cannot apply an ACL that specifies a different bucket or object owner.**

  Bucket and object ownership cannot be changed by modifying ACLs. If you apply a new ACL to a bucket or object, be sure that the bucket or object owner remains unchanged in the new ACL.

- **The bucket or object owner always has `OWNER` permission of the bucket or object.**

  The owner of a bucket is the project owners group, and the owner of an object is either the user who uploaded the object, or the project owners group if the object was uploaded by an anonymous user.

  When you apply a new ACL to a bucket or object, Cloud Storage respectively adds `OWNER` permission to the bucket or object owner if you omit the grants. It does *not* grant the project owners group `OWNER` permission for an object (unless the object was created by an anonymous user), so you must explicitly include it.

You cannot apply ACLs that change the *ownership* of a bucket or object (which should not be confused with the `OWNER` permission). Once created in Cloud Storage, bucket and object ownership are permanent. You can, however, effectively change the ownership of objects (but not buckets) by overwriting them. Overwriting is basically a delete operation followed immediately by an upload operation. During an upload operation, the person who is performing the upload becomes the owner of the object. Keep in mind that to overwrite an object, the person performing the overwrite (and is gaining ownership of the object by doing so) must have `WRITER` or `OWNER` permission on the bucket in which the object is being uploaded.

You can disable **all** ownership properties of a bucket and the objects within it by enabling uniform bucket-level acces
age/docs/uniform-bucket-level-access). Doing so also removes all use of ACLs in the bucket, making Cloud IAM polic
age/docs/access-control/iam) the only access control system used by the bucket.