This page provides an overview of Cloud Identity and Access Management (Cloud IAM) and its use with controlling access to buckets and objects in Cloud Storage. To learn how to set and manage Cloud IAM permissions for Cloud Storage buckets and projects, see Using Cloud IAM Permissions (/storage/docs/access-control/using-iam-permissions). To learn about other ways of controlling access to buckets and objects, see Overview of Access Control (/storage/docs/access-control/).

This page focuses on aspects of Cloud IAM that are relevant specifically to Cloud Storage. For a detailed discussion of Cloud IAM and its features generally, see Cloud Identity and Access Management (/iam/). In particular, see the Managing Cloud IAM Policies (/iam/docs/granting-changing-revoking-access) section.

Cloud Identity and Access Management (Cloud IAM) allows you to control who has access to the *resources* in your Google Cloud project. Resources include Cloud Storage buckets and objects stored within buckets, as well as other Google Cloud entities such as Compute Engine instances (/compute/).

The set of access rules you apply to a resource is called a Cloud IAM *policy*. A Cloud IAM policy applied to your project defines the actions that users can take on all objects or buckets within your project. A Cloud IAM policy applied to a single bucket defines the actions that users can take on that specific bucket and objects within it.

For example, you can create a Cloud IAM policy for one of your buckets that gives one user administrative control of that bucket. Meanwhile, you can add another user to your project-wide Cloud IAM policy that gives that user the ability to view objects in any bucket of your project.

Members are the "who" of Cloud IAM. Members can be individual users, groups, domains, or even the public as a whole. Members are assigned *roles*, which grant members the ability to perform actions in Cloud Storage as well as Google Cloud more generally. Each role is a collection of one or more *permissions*. Permissions are the basic units of Cloud IAM: each permission allows you to perform a certain action.

For example, the `storage.objects.create` permission allows you to create objects. This permission is found in roles such as `roles/storage.objectCreator`, where it is the only permission, and `roles/storage.objectAdmin`, where many permissions are bundled together. If you give the `roles/storage.objectCreator` role to a member for a specific bucket, they can only create objects in that bucket. If you give another member the `roles/storage.objectAdmin` role for the bucket, they can

do additional tasks, such as deleting objects, but still only within that bucket. If these two users are assigned these roles and no others, they won't have knowledge of any other buckets in your project. If you give a third member the `roles/storage.objectAdmin` role, but do so for your project and not just a single bucket, they have access to any object in any bucket of your project.

Using Cloud IAM with Cloud Storage makes it easy to limit a member's permissions without having to modify each bucket or object permission individually.

ɲ: For fine-grained access control of individual objects, Cloud Storage uses ACLs (/storage/docs/access-control/list ACLs do not appear in the hierarchy of Cloud IAM policies. When evaluating who has access to one of your objects, ɔu check the ACLs (/storage/docs/access-control/create-manage-lists#retrieving-acls) for the object, in addition to ng your project- and bucket-level Cloud IAM policies.

Permissions allow members to perform specific actions on buckets or objects in Cloud Storage. For example, the `storage.buckets.list` permission allows a member to list the buckets in your project. You don't directly give members permissions; instead, you assign roles (#roles), which have one or more permissions bundled within them.

For a reference list of the Cloud IAM permissions that apply to Cloud Storage, see Cloud IAM Permissions for Cloud Storage (/storage/docs/access-control/iam-permissions).

Roles are a bundle of one or more permissions (#permissions). For example, `roles/storage.objectViewer` contains the permissions `storage.objects.get` and `storage.objects.list`. You assign roles to members, which allows them to perform actions on the buckets and objects in your project.

For a reference list of the Cloud IAM roles that apply to Cloud Storage, see Cloud IAM Roles for Cloud Storage (/storage/docs/access-control/iam-roles).

Granting roles at the bucket level does not affect any existing roles that you granted at the project level, and vice versa. Thus, you can use these two levels of granularity to customize your

permissions. For example, say you want to give a user permission to read objects in any bucket but create objects only in one specific bucket. To achieve this, give the user the `roles/storage.objectViewer` role at the project level, thus allowing the user to read any object stored in any bucket in your project, and give the user `roles/storage.objectCreator` role at the bucket level for a specific bucket, thus allowing the user to create objects only in that bucket.

Some roles can be used at both the project level and the bucket level. When used at the project level, the permissions they contain apply to all buckets and objects in the project. When used at the bucket level, the permissions only apply to a specific bucket and the objects within it. Examples of such roles are `roles/storage.admin`, `roles/storage.objectViewer`, and `roles/storage.objectCreator`.

Some roles can only be applied at one level. For example, you can only apply the `Viewer` role at the project level, while you can only apply the `roles/storage.legacyObjectOwner` role at the bucket level.

n: Setting Cloud IAM policies at project and bucket levels potentially allow users access to objects that they did not h
usly. Carefully evaluate your policies before setting them.

When you view the Cloud IAM policy for a project, you do not see the Cloud IAM policies that exist for individual bucke
ew the Cloud IAM policy for an individual bucket using the Google Cloud Console (/storage/docs/cloud-console), you
oject-level permissions that apply to that bucket; however, other Cloud Storage tools, such as gsutil (/storage/docs/g
e Client Libraries (/storage/docs/reference/libraries) only return the policy of the bucket, and do not include informa
ed from the project-level policy. Be sure to consult both project and bucket policies when evaluating access levels for
ers.

Legacy Bucket Cloud IAM roles work in tandem with bucket ACLs (/storage/docs/access-control/lists#permissions): when you add or remove a Legacy Bucket role, the ACLs associated with the bucket reflect your changes. Similarly, changing a bucket-specific ACL updates the corresponding Legacy Bucket role for the bucket.

Legacy Bucket role	Equivalent ACL
`roles/storage.legacyBucketReader`	Bucket Reader
`roles/storage.legacyBucketWriter`	Bucket Writer
`roles/storage.legacyBucketOwner`	Bucket Owner

All other bucket-level Cloud IAM roles, and all project-level Cloud IAM roles, work independently from ACLs. For example, if you give a user the **Storage Object Viewer** role, the ACLs remain unchanged. This means you can use bucket-level Cloud IAM roles to grant broad access to all objects within a bucket and use the fine-grained object ACLs (/storage/docs/access-control/lists#permissions) to customize access to specific objects within the bucket.

You can use Cloud IAM to grant members the permission needed to change ACLs on objects. The following `storage.buckets` permissions together allow users to work with bucket ACLs and default object ACLs: `.get`, `.getIamPolicy`, `.setIamPolicy`, and `.update`.

Similarly, the following `storage.objects` permissions together allow users to work with object ACLs: `.get`, `.getIamPolicy`, `.setIamPolicy`, and `.update`.

While Cloud IAM has many predefined roles that cover common use cases, you may wish to define your own roles which contain bundles of permissions that you specify. To support this, Cloud IAM offers custom roles (/iam/docs/creating-custom-roles).

There are a number of different types of members. For example, Google accounts and Google groups represent two general types, while `allAuthenticatedUsers` and `allUsers` are two specialized types. For a list of typical member types in Cloud IAM, see Concepts related to identity (/iam/docs/overview#concepts_related_identity). In addition to the types listed there, Cloud IAM supports the following member types, which can be applied specifically to your Cloud Storage bucket Cloud IAM policies:

- `projectOwner:[PROJECT_ID]`
- `projectEditor:[PROJECT_ID]`
- `projectViewer:[PROJECT_ID]`

Where `[PROJECT_ID]` is the ID of a specific project.

When you grant one of the above member types a role, all members with the specified permission for the specified project get the role you select. For example, say you want to give all members who have

the `Viewer` role for the project `my-example-project` the `roles/storage.objectCreator` role for one of your buckets. To do so, give the member `projectViewer:my-example-project` the `roles/storage.objectCreator` role for that bucket.

ature is in a pre-release state and might change or have limited support. For more information, see the product launc (/products/#product-launch-stages).

Cloud IAM Conditions (/iam/docs/conditions-overview) allows you to set conditions that control how permissions are granted to members. Cloud Storage supports the following types of condition attributes:

- **`resource.name`** (/iam/docs/conditions-attribute-reference#resourcename_attribute)**:** Grant access to buckets and objects that have the specified prefix. You can also use **`resource.type`** (/iam/docs/conditions-attribute-reference#resourcetype_attribute) to grant access to either buckets or objects, but doing so is mostly redundant with using `resource.name`.

- **Date/time** (/iam/docs/conditions-attribute-reference#date-time)**:** Set an expiration date for the permission.

These conditional expressions are logic statements that use a subset of the Common Expression Language (https://github.com/google/cel-spec) (CEL). You specify conditions in the role bindings of a bucket's Cloud IAM policy.

You can also use double quotes in your conditional expressions, but they should be properly escaped when you are u l commands and HTTP/REST requests to update a conditional policy. For example, **`request.time < tamp(\"2019-01-01T00:00:00Z\")`**.

Keep these restrictions in mind:

- You must enable uniform bucket-level access (/storage/docs/uniform-bucket-level-access) on a bucket before adding conditions at the bucket level. Although conditions are allowed at the project level, you should migrate all buckets in the project to uniform bucket-level access to prevent Cloud Storage ACLs from overriding project-level Cloud IAM conditions. You can also set an organization policy (/storage/docs/setting-org-policies#uniform-bucket) to enable uniform bucket-level access for all new buckets in your project.
- The `gsutil iam ch` command does not work with policies that contain conditions. To modify a policy that has conditions, use `gsutil iam get` to retrieve the policy for the relevant bucket, edit it locally, and then use `gsutil iam set` to re-apply it to the bucket.
- gsutil must be at version 4.38 or higher to use conditions.
- When using the JSON API to call `getIamPolicy` (/storage/docs/json_api/v1/buckets/getIamPolicy) and `setIamPolicy` (/storage/docs/json_api/v1/buckets/setIamPolicy) on buckets with conditions, you must set the Cloud IAM policy version (/iam/docs/policies#versions) to 3.
- Expired conditions remain in your Cloud IAM policy until you remove them.

Although Cloud IAM permissions cannot be set through the XML API, users granted Cloud IAM permissions can still use the XML API, as well as any other tool for accessing Cloud Storage.

For references of which Cloud IAM permissions allow users to perform actions with different Cloud Storage tools, see Cloud IAM with the Cloud Console (/storage/docs/access-control/iam-console), Cloud IAM with gsutil (/storage/docs/access-control/iam-gsutil), Cloud IAM with JSON (/storage/docs/access-control/iam-json), and Cloud IAM with XML (/storage/docs/access-control/iam-xml).

Cloud IAM, like any other administrative settings, requires active management to be effective. Before you make a bucket or object accessible to other users (/storage/docs/access-control/using-iam-permissions), 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 Cloud IAM settings on buckets and projects, especially if you manage Cloud Storage 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.**

  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 is necessary for a user to accomplish their assigned task. For example, if you want to share a file with someone, grant them the `storage.objectReader` role and not the `storage.admin` role.

- **Avoid granting roles with `setIamPolicy` permission to people you do not know.**

  Granting `setIamPolicy` permission allows a user to change permissions and take control of data. You should use roles with `setIamPolicy` 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` member types 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 certain permissions such as `setIamPolicy`, `update`, `create`, or `delete`.

- **Understand the `Viewer`/`Editor`/`Owner` project roles in Cloud Storage**

  The `Viewer`/`Editor`/`Owner` project-level roles (/storage/docs/projects#permissions) effectively grant the access their names imply within Cloud Storage; however they do so indirectly through additional access that is provided at the bucket and object levels, using member types (#identities) unique to Cloud Storage. While the access is added by default, you can revoke it.

  For example, the `Viewer` role, by itself, only gives a member `storage.buckets.list` permission, but new buckets, by default, grant the `roles/storage.legacyBucketReader` role to all members with the `Viewer` role for the project. This bucket role is what allows a `Viewer` to view a bucket. Additionally, the bucket has a default object ACL of `projectPrivate` (/storage/docs/access-control/lists#predefined-acl), which means objects added to the bucket gain, by default, the `projectPrivate` ACL. This ACL is what allows a `Viewer` to view the object.

  Similarly, the `Editor` and `Owner` project roles have limited Cloud Storage access by themselves, but members assigned these roles gain `roles/storage.legacyBucketOwner` for new buckets, and ownership of objects through the `projectPrivate` ACL.

  Note that because some bucket and object access is not inherent to the `Viewer`/`Editor`/`Owner` project roles, it is possible to revoke access that members might otherwise expect they have.

- **Avoid setting permissions that result in inaccessible buckets and objects.**

A bucket or object becomes inaccessible when there is no one with permission to read it. This can happen for a bucket when all Cloud IAM permissions on the bucket get removed. This can happen for an object when the object's owner leaves a project and there are no bucket or project-level Cloud IAM policies that grant access to objects. In both cases, you can regain access by assigning an appropriate role, such as `roles/storage.admin`, to yourself or another member at the project level. Note that doing so gives access to **all** buckets and objects in the project, so you may want to revoke the role (/storage/docs/access-control/using-iam-permissions#project-remove) once you've reset access to the affected bucket or object.

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

  By default, members with the project-level `Owner` role are the only entities that have the `roles/legacyBucketOwner` role on a newly created bucket. You should have at least two members with the `Owner` role at any given time so that if a team member leaves the group, your buckets can still be managed by the other members.

- Learn how to use Cloud IAM with Cloud Storage (/storage/docs/access-control/using-iam-permissions).
- Review the Cloud IAM reference table for Cloud Storage (/storage/docs/access-control/iam-reference).