This document provides common data sharing and collaboration scenarios. It includes how to configure your project and bucket Identity and Access Management (IAM) policies (/storage/docs/access-control/iam) and your object Access Control Lists (ACLs) (/storage/docs/access-control/lists) to implement the scenarios.

In this scenario, a company's marketing analyst wants to use Cloud Storage to back up confidential revenue forecasts and sales projection data. The data must be accessible only by the marketing analyst. The company's IT department oversees and manages the company's Cloud Storage account. Their primary management responsibilities include creating and sharing buckets so that various departments throughout the company have access to Cloud Storage.

To meet the confidentiality and privacy needs of the marketing analyst, the bucket and object permissions must allow the IT staff to maintain the bucket in which the spreadsheets are stored, but also ensure that the IT staff cannot view/download the data that is stored in the bucket. To accomplish this, you create a bucket named `finance-marketing` and grant the following *roles* (/storage/docs/access-control/iam-roles) for the listed *resources* to the specified *members*:

Role	Resource	Member	Description
`roles/storage.legacyBucketOwner`	The bucket `finance-marketing`	IT staff	Giving the IT staff the `roles/storage.legacyBucketOwner` role for the bucket allows them to perform common bucket management tasks, such as deleting objects and changing the IAM policy on the bucket. It also allows the IT staff to list the contents of the `finance-marketing` bucket, but not view or download any of the contents.
`roles/storage.objectCreator`	The bucket `finance-marketing`	Marketing analyst	Giving the marketing analyst the `roles/storage.objectCreator` role for the bucket allows her to upload objects to the bucket. When she does so, she becomes the owner of the uploaded object, which she can then view, update, and delete.

With these roles, nobody except the marketing analyst can view/download the objects that she adds to the bucket. She can, however, grant other users access by changing the object's ACLs (/storage/docs/access-control/create-manage-lists#set-an-acl). The IT staff can still list the contents of the `finance-marketing` bucket, and they can delete and overwrite the files that are stored in the bucket should the need arise.

n: Because the IT staff has permission to manage the bucket, it is possible for them to grant themselves or others ssion to view objects in the bucket. To mitigate this, you can view audit logs (/logging/docs/audit/) in order to track es made to bucket and project permissions.

Your actions

You should take the following actions to implement the scenario:

1. Create a bucket named `finance-marketing`. For step-by-step instructions, see Creating a bucket (/storage/docs/creating-buckets).
2. Give each IT staff member the `roles/storage.legacyBucketOwner` role for the bucket. For step-by-step instructions, see Adding a member to a bucket-level policy (/storage/docs/access-control/using-iam-permissions#bucket-add).

IT staff actions

The IT staff should take the following actions to implement the scenario:

1. Give the marketing analyst the `roles/storage.objectCreator` for the bucket. For step-by-step instructions, see Adding a member to a bucket-level policy (/storage/docs/access-control/using-iam-permissions#bucket-add).

In this scenario, a construction company works with several architectural design firms that deliver building plans for various projects. The construction company wants to set up a drop box for the vendor firms so they can upload architectural plans at various project milestones. The drop box must ensure the privacy of the construction company's clients, which means the drop box cannot allow the vendors to see each other's work. To accomplish this, you create a separate bucket for each architectural firm and grant the following *roles* (/storage/docs/access-control/iam-roles) for the listed *resources* to the specified *members*:

Role	Resource	Member	Description

Role	Resource	Member	Description
`roles/owner`	Overall project	Construction company manager	Giving the construction company manager the `roles/owner` role at the project level enables her to create buckets for each vendor.
`roles/storage.objectViewer`	Overall project	Construction company manager	The `roles/storage.objectViewer` allows the construction company manager to download the objects that the vendors are uploading.
`roles/storage.legacyBucketOwner`	Each vendor bucket	Construction company manager	The `roles/storage.legacyBucketOwner` role allows the construction company manager to list the contents of each bucket as well as delete objects at the end of each project milestone.
`roles/storage.objectAdmin`	Each vendor bucket	The vendor associated with the bucket	Giving each vendor the `roles/storage.objectAdmin` for their own bucket gives them complete control over the objects in their bucket, including the ability to upload objects, list objects in the bucket, and control who has access to each object. It does not allow them to change or view metadata such as roles on the bucket as a whole, nor does it allow them to list or view other buckets in the project, thus preserving privacy between vendors.

Your actions

You should take the following actions to implement the scenario:

1. Give the construction company manager the `roles/owner` role for the project as well as the `roles/storage.objectViewer` role for the project. For step-by-step instructions, see Adding a member to a project-level policy (/storage/docs/access-control/using-iam-permissions#project-add).

Construction company manager actions

The construction company manager should take the following actions to implement the scenario:

1. Create a separate bucket for each vendor. For step-by-step instructions, see Creating a bucket (/storage/docs/creating-buckets).

   Since the construction manager has the `roles/owner` role, she automatically gets the `roles/storage.legacyBucketOwner` role for each bucket she creates.

2. Give each vendor the `roles/storage.objectAdmin` for their respective bucket. For step-by-step instructions, see Adding a member to a bucket-level policy (/storage/docs/access-control/using-iam-permissions#bucket-add).

3. If any vendor intends to use the Google Cloud Console, give them a link to their bucket, which has the format:

   `console.cloud.google.com/storage/browser/[BUCKET_NAME]`

   where [BUCKET_NAME] is the name of the vendor's bucket.

Vendor actions

Each vendor should take the following actions to implement the scenario:

1. Upload objects to the assigned bucket. The easiest way to accomplish this is through the Google Cloud Console. Other methods, such as the gsutil command line tool, require additional setup prior to use. For step-by-step instructions, see Uploading an object (/storage/docs/uploading-objects).

★ **Note:** Based on the permissions set in this scenario, vendors are required to sign in using one of the emails associated with the bucket assigned to them.

In this scenario, a client wants to make specific files available to specific individuals through simple browser downloads. You can do this by using the Cloud Storage cookie-based authentication (/storage/docs/access-control/cookie-based-authentication). To use the feature, you grant a user permission to access an object, and then you give the user a special URL to the object. When the user clicks the URL, Cloud Storage prompts them to sign in to their Google account (if they are not already logged in) and the object is downloaded to their computer. The following users will be able to download the object:

- Users who have `READ` or `FULL_CONTROL` permission on the access control list (ACL) (/storage/docs/access-control/lists) of the object itself.
- Users with `storage.objects.get` permission in the Identity and Access Management (IAM) (/storage/docs/access-control/iam) policy for the bucket or project that contains the object.

All other users get a `403 Forbidden (access denied)` error.

You can implement cookie-based authentication in four general steps:

1. **Create a bucket.** For step-by-step instructions, see Creating a bucket (/storage/docs/creating-buckets).

   Assuming you create a bucket in a project you own, you automatically gain permissions that allow you to upload objects to the bucket and change who has access to the bucket.

   ★ **Note:** If you already have a bucket that you want to use, keep in mind that you're serving secure content from the bucket in this scenario. As a best practice, make sure anonymous users don't have bucket or project-level roles that give access to the objects.

2. **Upload the object you want to share.** For step-by-step instructions, see Uploading an object (/storage/docs/uploading-objects).

   When you upload an object you become the owner of the object, which means you can modify the object's ACLs.

3. **Give users access to the object.** You can do this in one of two ways:

   a. Modify the bucket's IAM policy to give each desired user the `roles/storage.objectViewer` role, which applies to all objects in the bucket. For step-by-step instructions, see Adding a member to a bucket-level policy (/storage/docs/access-control/using-iam-permissions#bucket-add).

   b. Modify the object's ACLs to give each desired user `READ` permission on the individual object. For step-by-step instructions, see Setting ACLs (/storage/docs/access-control/create-manage-lists#set-an-acl).

4. **Provide users with a special URL to the object.**

   Authenticated browser downloads access Cloud Storage through a specific URL endpoint (/storage/docs/request-endpoints#cookieauth). Use the following URL, replacing `[VALUES_IN_BRACKETS]` with appropriate values:

   ```
   https://storage.cloud.google.com/[BUCKET_NAME]/[OBJECT_NAME]
   ```

   Since only users with appropriate ACL or IAM permissions can view it, it doesn't matter how you make this URL available. You can send it to them directly, or you can post it on a web page.

In this scenario, you want to make objects available to specific users, such as users invited to try out new software. In addition, you want to invite many users, but you do not want to set permission for

each user individually. At the same time, you don't want to make the objects publicly readable and send invited customers links to access the objects, because there is a risk the links may be sent to users who are not invited.

One way to handle this scenario is through the use of Google Groups (https://groups.google.com/). You can create a group and add only invited users to the group. Then, you can give the group as a whole access to the objects:

Role	Resource	Member	Description
`roles/storage.objectViewer`	Your "whitelisted" bucket	Google Group	Giving the Google Group the `roles/storage.objectViewer` role for the bucket allows any customer who is part of the Google Group to view objects in the bucket. No one outside of the group has access to the objects.

When using groups to manage access to your resources, you should be aware of Group policies and limits ://support.google.com/a/answer/6099642?hl=en) that determine how many members can be in the group. If you n more users than can be added to a group, you can create a service that authenticates users and redirects them to a U by a service account (/iam/docs/service-accounts). For more information, see Signed URLs (query string authentica age/docs/access-control/signed-urls).

1. Create a Google Group and add customers to it. For step-by-step instructions see Create a group (https://support.google.com/groups/answer/2464926?hl=en).
2. Create a bucket. For step-by-step instructions, see Creating a bucket (/storage/docs/creating-buckets).
3. Upload objects to your bucket. For step-by-step instructions, see Uploading an object (/storage/docs/uploading-objects).
4. Give the Google Group access to the objects.
    - You can use the IAM role **storage.objectViewer** to give viewing access to all objects in your bucket. For step-by-step instructions, see Adding a member to a bucket-level policy (/storage/docs/access-control/using-iam-permissions#bucket-add).
    - If you want to only give access to some of the objects in the bucket, set the **Reader** ACL on those individual objects. For step-by-step instructions, see Setting ACLs (/storage/docs/access-control/create-manage-lists#set-an-acl).