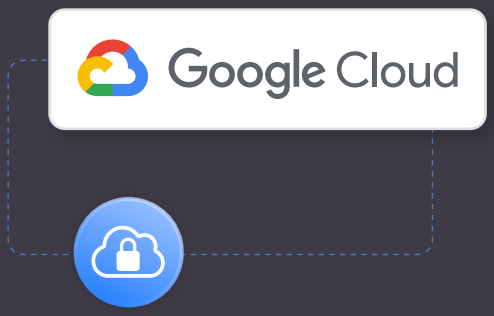
ConductorOne

# Everything You Want to Know about GCP Access Control



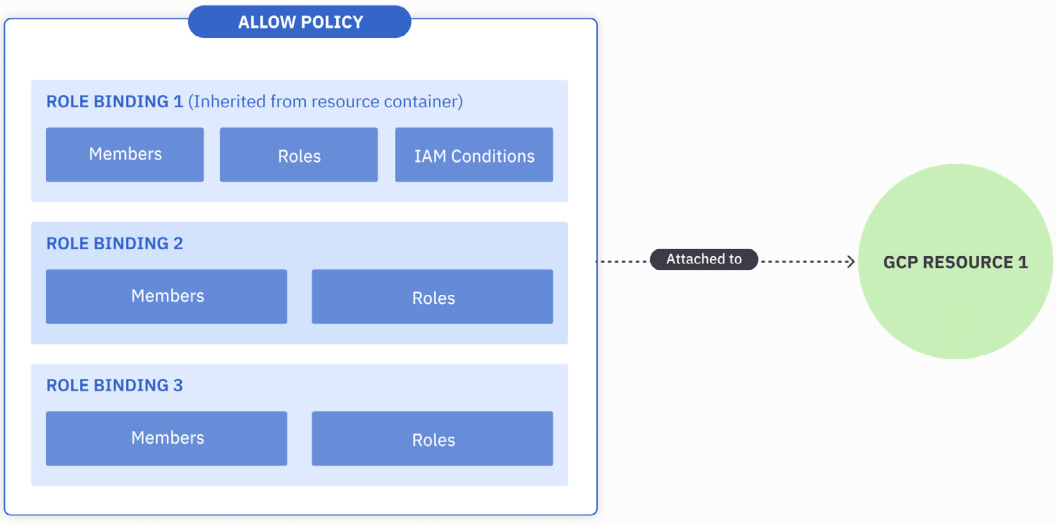

In an era where cloud computing powers most of the products and services on the web, secure and efficient management of access to cloud resources is crucial. For organizations using the Google Cloud Platform (GCP), mastering the art of access control is not just a best practice; it's an essential component of maintaining the integrity, security, and compliance of their cloud environments.

GCP has a detailed access control model. It allows you to manage permissions and policies for your resources across hierarchies, enabling you to easily set up and manage access for your team members. In this article, you'll learn what the GCP access control model is all about, how it works, what its pros and cons are, and what the best practices are for setting up your organization's access control strategy.

## Understanding GCP Access Control

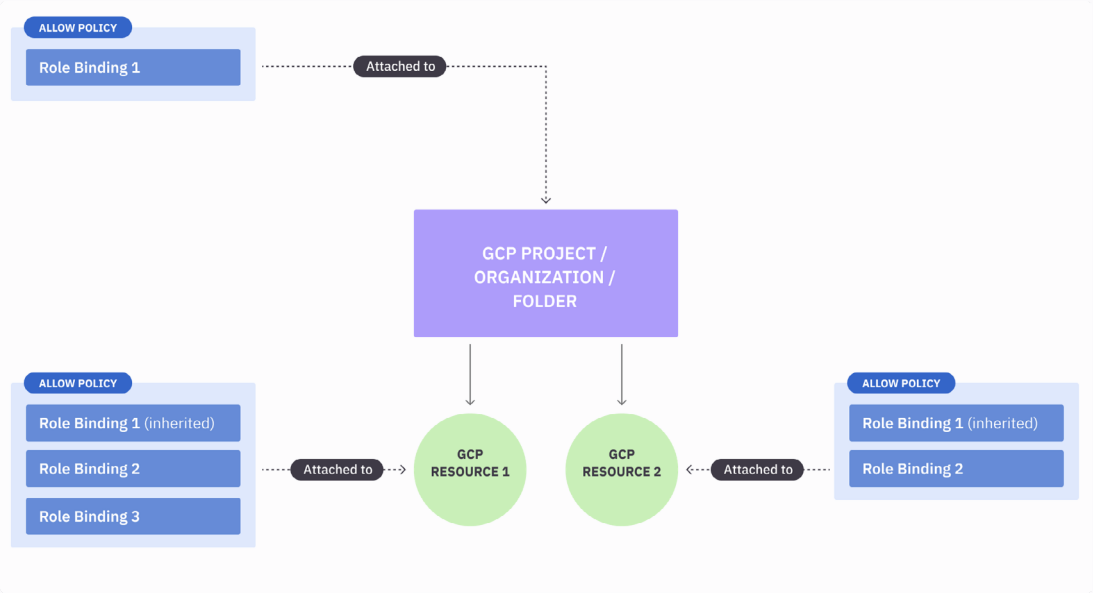
Before getting into how to manage access in GCP, it helps to be aware of a few key concepts used in GCP access control.

Here's a quick overview of the GCP access model for a resource:



Each resource in GCP has an associated allow policy that contains role bindings. Each role binding is a set of members, roles, and conditions. You'll learn more about these in the coming sections. Granting someone access to a GCP resource essentially means creating a role binding for them in the resource's allow policy that provides them with the necessary role to access the resource appropriately, optionally under a condition if needed.

GCP resources constitute a hierarchy through projects, folders, and organizations.
Role bindings set at a higher level in the resource hierarchy get inherited by each resource below it:




## Principals

The entities that can access GCP resources are known as principals. A principal can be a Google account, a service account, a Google Group, a Google Workspace account, or a Cloud Identity domain.

Each of these entities has a unique identifier, usually an email address. They can also be assigned roles, and you can bind them to resources using Identity and Access Management (IAM) policies.

## Permissions and Roles

Permissions in GCP are the granular actions that dictate what someone can do with a resource. These actions range from read-only operations, like viewing a resource, to more powerful actions, like creating or deleting it.

Roles in GCP define a set of permissions that can be assigned to principals. These specify what actions can be performed on which resources. GCP provides a spectrum of predefined roles, each tailored to a specific set of responsibilities, making it easier to grant appropriate access without the need to create roles from scratch. GCP also provides a few basic roles that allow general access to most resources. However, if neither of these two fits your purpose, you can always create custom roles.

## IAM Policies

IAM policies act as the rulebook for your access control strategy. These policies determine which principals have what level of access to which resources. When you grant a principal access to a role for a resource, the role binding is stored in the resource's IAM policy, also known as its allow policy.

> *"Additionally, you can also create deny policies to deny permissions for certain principals in the resource. These are different from allow policies, and each GCP resource can have up to five deny policies."*

IAM policies provide the framework for implementing the principle of least privilege, ensuring that individuals and services are granted only the permissions they require to perform their designated tasks.

IAM policies can be directly attached to resources, and each resource can have only one policy. Here's what the JSON representation of an IAM policy looks like:

```
{
  "bindings": [
    {
      "members": [
        "user:kumar@example.com"
      ],
      "role": "roles/resourceManager.
projectCreator"
    }
  ],
  "etag": "some value here",
  "version": 1
}
```

This policy grants the user with the email address **`kumar@example.com`** the ability to create projects. Once you attach this policy to a GCP organization or folder, the user will be able to create projects under that organization or folder.

## GCP Resource Hierarchy

GCP resources can be organized using containers, such as organizations, folders, and projects. An organization is usually attached to a Google Workspace and a Cloud Identity domain, and an organization can have multiple folders. Folders are containers for projects to allow you to build a hierarchical access model with access inheritance across a large number of projects easily. Finally, projects are the lowest level of resource containers, and they help you provision resources for a single business initiative (such as an application or a system) in one place.

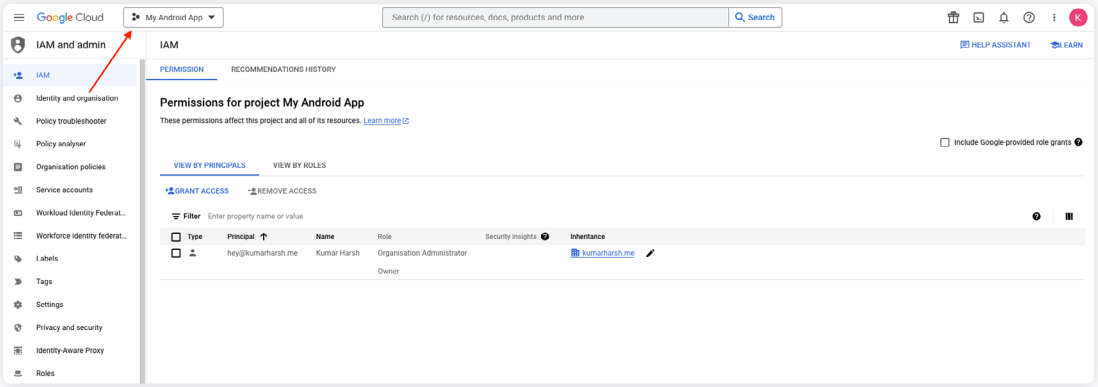
GCP access policies are propagated across the hierarchy created using the resource containers listed earlier. This inheritance allows you to simplify access management, and you can also replicate your organization's internal hierarchy to better manage GCP access.

# Managing Access in GCP

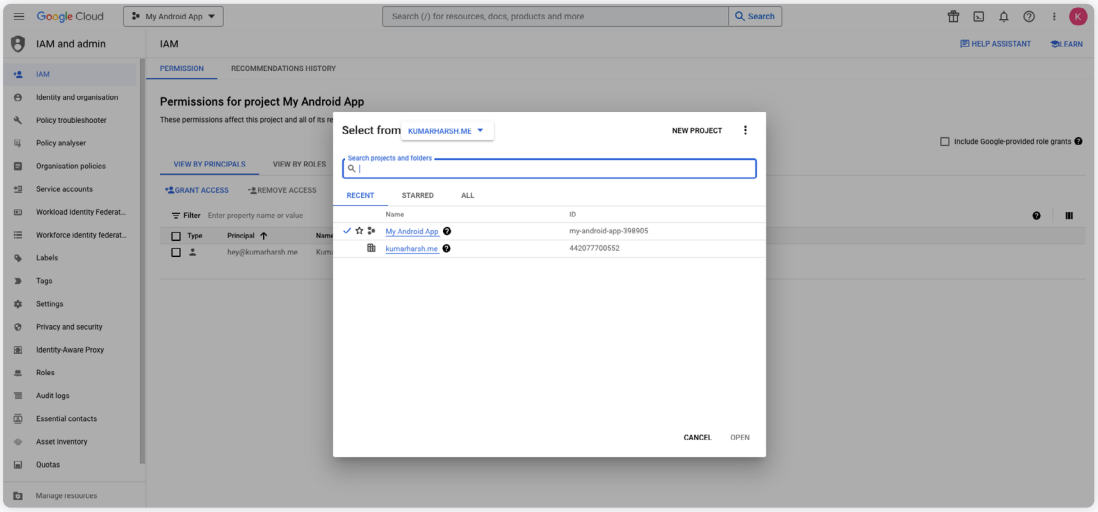
Now that you've been introduced to the main concepts in GCP access control, it's time to learn how to manage permissions in the GCP platform.

# Granting and Revoking Permissions in GCP

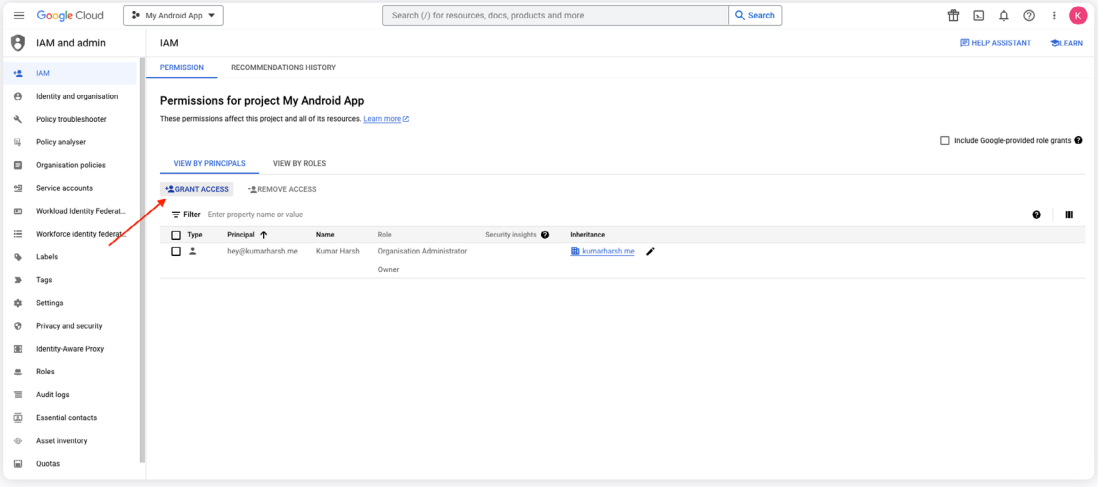
Granting and revoking permissions is a very straightforward task in GCP. You can start by navigating to the IAM and admin page in your GCP account. GCP will most probably autoselect a project for you. Make sure to select the appropriate project by clicking the projects drop-down at the top left of the window:



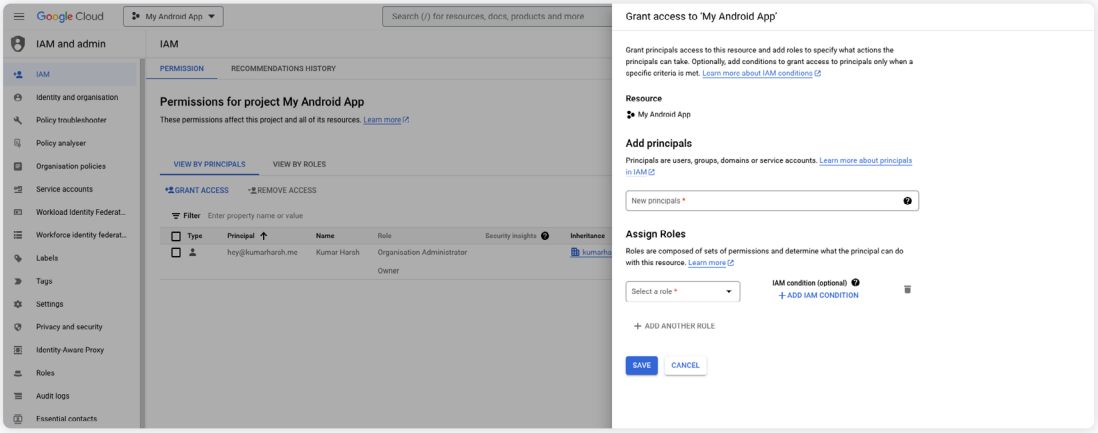

In the dialog that opens, choose the appropriate project. You can also create a new project if you don't have one in your account:




Once you have selected your project, you can now grant access to principals.
To start off, click the **GRANT ACCESS** button on the **IAM** page:

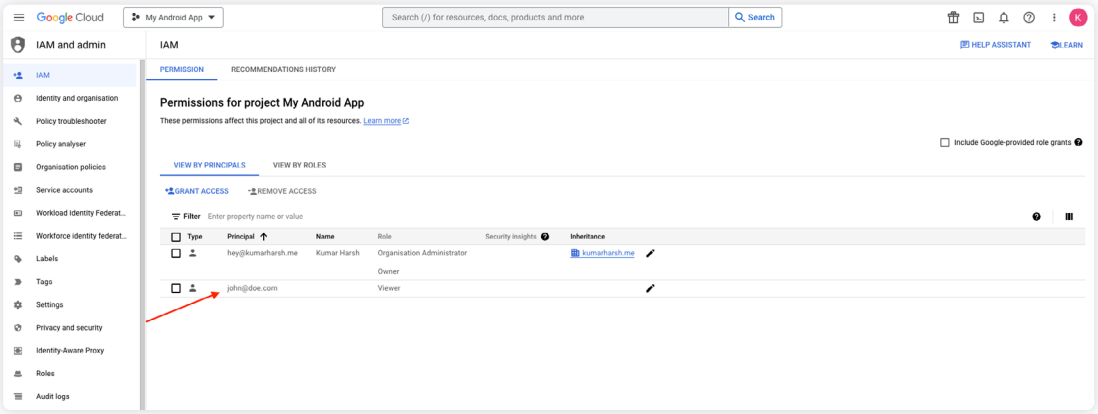


On the side pane that appears, you can choose the specifics of the new permission binding:



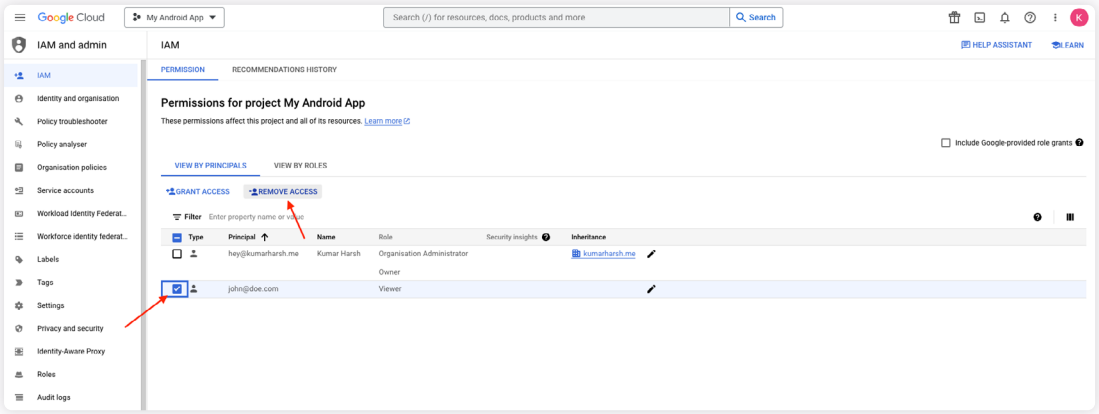

This pane allows you to first choose a principal that will receive access and then assign that principal a role to define what they will be able to do within the chosen resource (in this case, the **My Android App** project).

Once you click the **SAVE** button, a new entry in the permissions table is created:

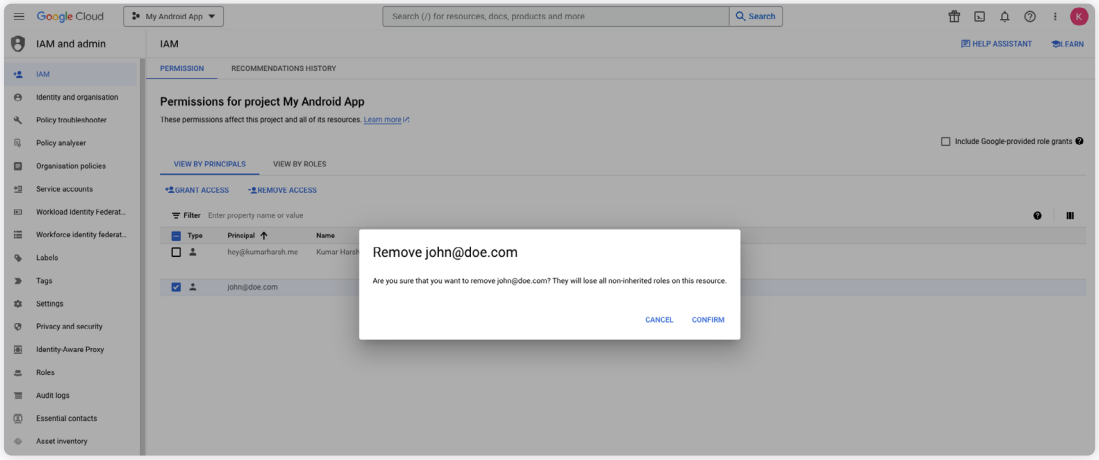



Similarly, you can assign IAM roles to users, groups, and service accounts. It's recommended that you use Google Groups to allow easy updating of the users associated with a role without having to manually follow this process every time. Groups can also be associated logically with a business function, such as development or DevOps; hence, it makes logical sense to grant the entire team access to the necessary resources rather than do it for each member individually. Furthermore, if an employee leaves or joins the team, they will be added to or removed from the Google Group as well, which will automatically handle the granting or revoking of their access to GCP resources.

Revoking permissions is quite simple. Click on the checkbox on the left end of the permission that you wish to delete, and click on the **REMOVE ACCESS** button above the table:



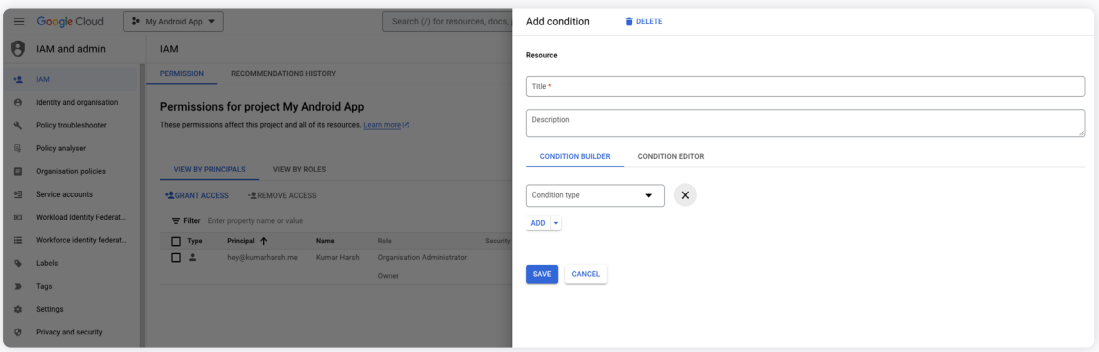

Click **CONFIRM** on the dialog box that appears, and the chosen permission will be removed from the resource:




## Implementing IAM Conditions and Fine-Grained Permissions Across Resource Hierarchy

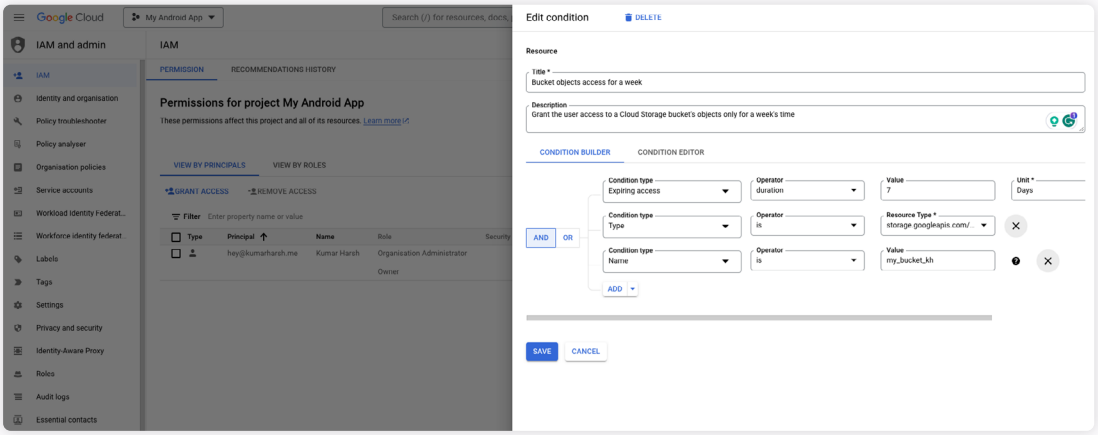
When granting access to a resource, you can set up IAM conditions, as mentioned earlier, o allow conditional access to the chosen resource. You can set both attribute-based access rules and time-based access rules for each permission that you grant.

When you click on the **ADD IAM CONDITION** button in the side pane that opens up when granting access, another side pane pops up that looks like this:



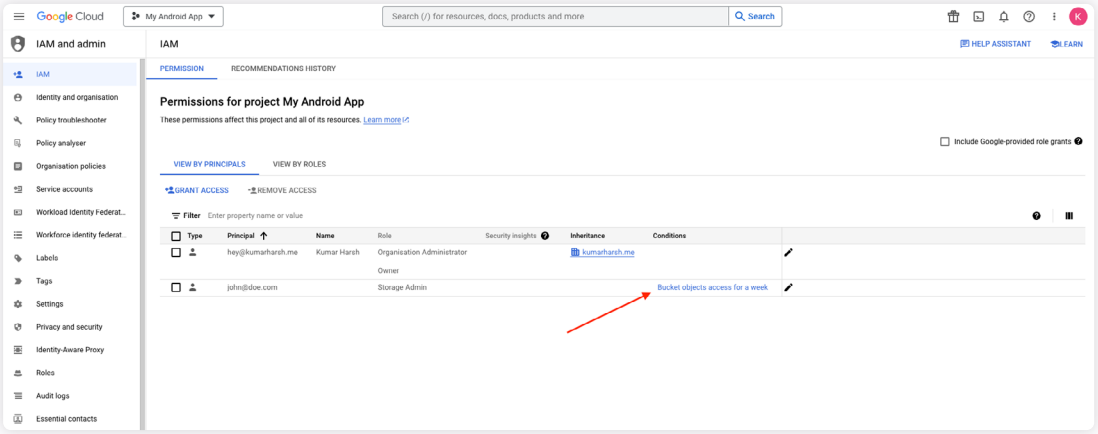
You can set a title and description for your condition that will help you understand its purpose when you refer back to it later. You can use either the condition builder or the condition editor to create the condition.

For instance, if you wanted to only grant **`john@doe.com`** to the Cloud Storage objects in a particular bucket named **`my_bucket_kh`** under the chosen project for a week, the condition builder would look like the following:



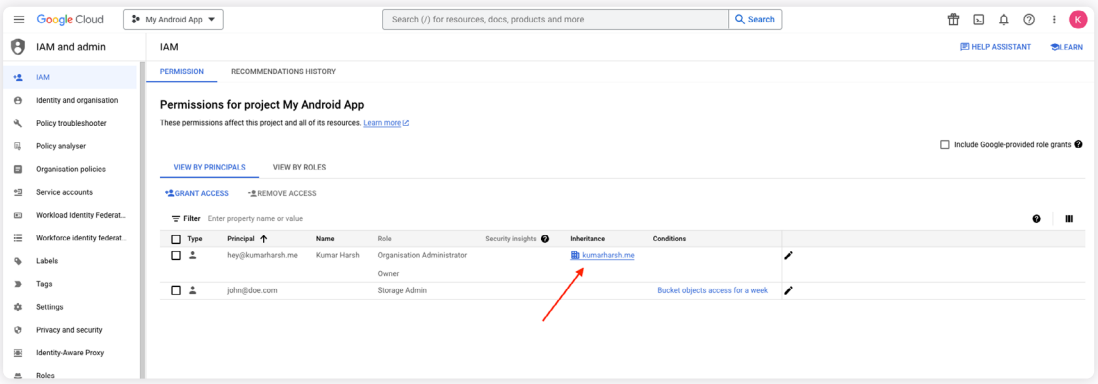

You can now choose the **Storage Admin** role for the permission, but the user will still be able to only access the resource based on the condition you attached to the permission.

You'll notice that the condition is mentioned in the permissions list as well:



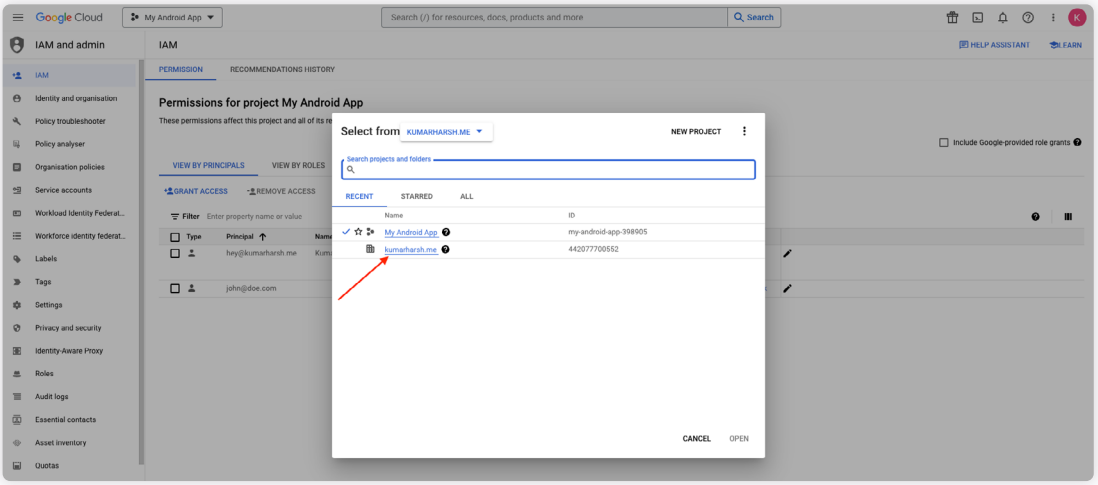

You can use conditions to set up fine-grained access control easily in GCP.

Additionally, you may have noticed that the first entry in the permissions list contains a value for the **Inheritance** column:

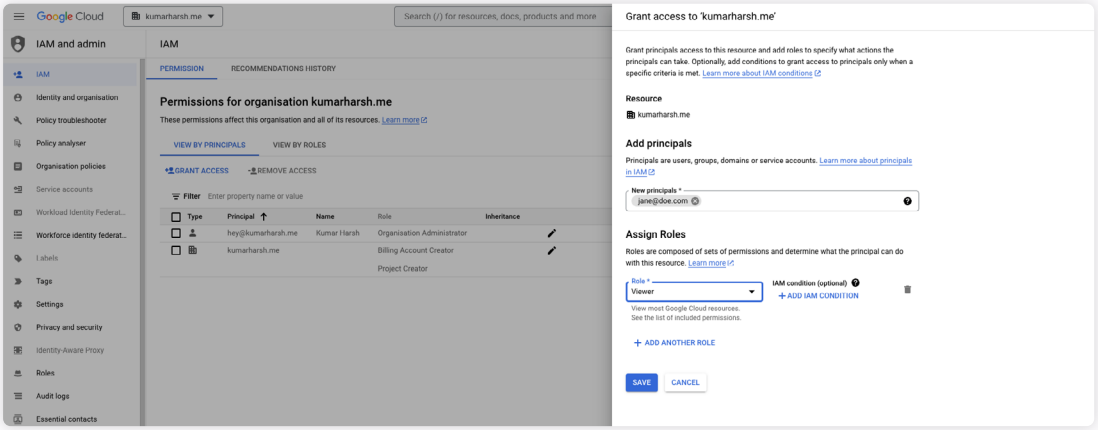


This indicates that this role was inherited from the organization resource and will be auto-added to any projects that are created in the same organization. You can set more roles to be auto-inherited by the projects in an organization by adding them to the organization's permissions list.

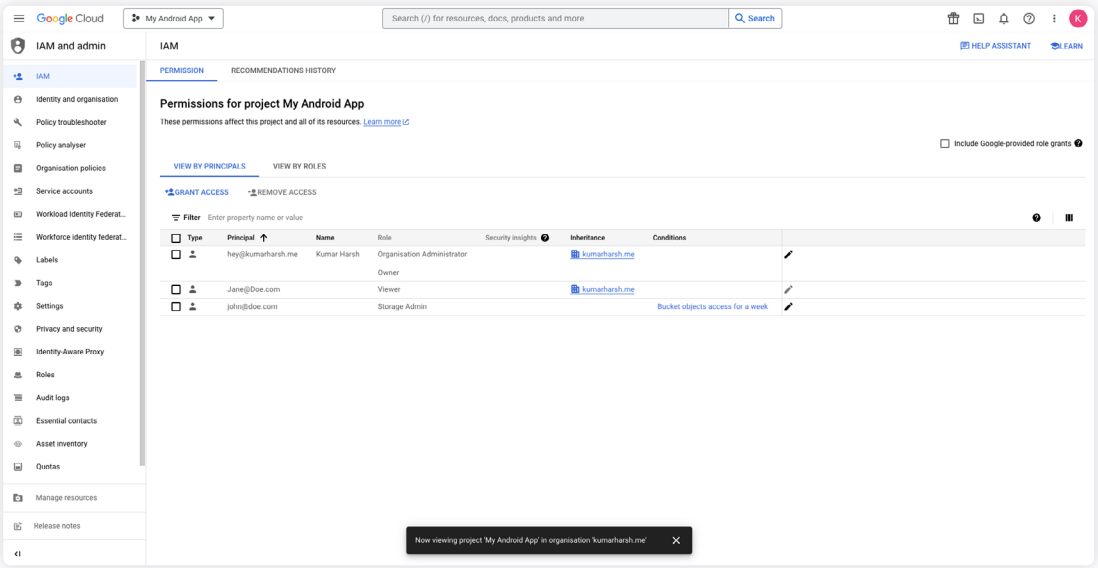
To do that, click on the projects drop-down at the top left of the page and click on your organization:




You will now see a list of permissions that are granted for the organization resource. Grant a new permission here using the **GRANT ACCESS** button:




Once you create this entry, go back to your project's **IAM** page to see the list of permissions:



You'll see that the newly created permission at the org level is automatically inherited to your project. This is how you can propagate access through the resource and containers hierarchy. The same applies to folders as well.

## Managing Access through IAM Policies

As mentioned already, all permission rules are stored as policies attached to resource objects. This means that you can directly create policy objects in either JSON or YAML and attach them to GCP resources instead of having to manually grant access one by one through the UI.

You can use the GCP SDK or the GCP CLI to attach policies directly to GCP resources. Here's a sample policy JSON:

```
{
  "bindings": [
    {
      "members": [
        "user:jane@doe.com"
      ],
      "role": "roles/viewer"
    }
  ],
  "etag": "new policy",
  "version": 1
}
```

You can save the JSON object in a file named, for example, **`policy.json`** , and run the following command to add it to your GCP organization:

```
gcloud organizations set-iam-policy your-
project.domain policy.json
```

This uses the gcloud CLI to attach the policy object directly to the organization resource. You can do the same to any GCP resource, with commands like **`gcloud compute instances set-iam-policy ...`** and **`gcloud project set-iam-policy ...`** .

You can also use the **`add-iam-policy-binding`** command to directly add a role binding to an existing policy on a resource instead of creating and supplying the complete policy object:

```
gcloud organizations add-iam-policy-binding
your-organization-id --member='user:jane@doe.
com' --role='roles/viewer'
```

Alternatively, you can use this method to set conditions:

```
gcloud organizations add-iam-policy-binding
your-organization-id --member='user:jane@
doe.com' --role='roles/viewer'
--condition='resource.name == \"my_bucket_
kh\"'
```

Finally, you can also set conditions through the **`set-iam-policy`** method by adding it to your policy JSON object:

```
{
  "bindings": [
    {
      "members": [
        "user:jane@doe.com"
      ],
      "role": "roles/viewer",
      "condition": {
        title: "Bucket access only"
        description: "Grant access to the my_
bucket_kh bucket only"
        expression: "resource.name == \"my_
bucket_kh\""
      }
    }
  ],
  "etag": "new policy",
  "version": 2
}
```

## What's Great about GCP Access Control

As you have seen already, GCP implements quite a detailed model of access control. The following are some of the strongest advantages of the GCP access control model:

- **Granularity in defining permissions:** As you've seen already, GCP enables you to grant granular access to resources structured using projects, organizations, and folders, to individual users, groups, service accounts, and even domains. You can control who can access resources, what actions they can perform on those resources, and under what conditions.
- **Easy assignment and management of IAM roles:** As shown earlier, adding and removing principals from resources and roles is quite straightforward. On the **IAM** page of the Google Cloud Console, you can manage the permissions for all your GCP resources quite easily. GCP offers a wide range of predefined roles that cover common use cases. These roles make it easier to assign appropriate permissions without having to create custom roles from scratch.
- **Seamless integration with other GCP services for access control:** With product and service-level roles, you can easily grant principals access to a wide range of GCP services from a single management portal. Additionally, GCP seamlessly integrates with identity providers like Google Workspace and external identity systems, making it easier to manage access and authentication.
- **Access control aligned with resource hierarchy:** GCP has a hierarchical structure, including organizations, folders, projects, and resources. This hierarchy allows you to set access control policies at different levels, simplifying management for large organizations.

## Challenges with GCP Access Control

While the GCP access model is quite robust and easily manageable, it has a few shortcomings:

- **Challenges in maintaining and auditing permissions at scale:** Defining and managing IAM policies can become complex in large organizations or projects with numerous users, groups, and roles. Keeping track of who has access to what resources and at what level of granularity can be challenging, especially since the IAM UI does not provide a high-level view of permissions across projects. You can use the Policy Analyzer to query permissions from across your projects and resources, but it can still be a very cumbersome experience.
- **Potential issues with policy inheritance and propagation:** While the resource hierarchy in GCP is beneficial for access control, it can also become complex in large organizations with many projects, folders, and resources. Ensuring consistent access policies across this hierarchy can be challenging. As organizations create custom roles, there's a risk of permissions overlap or conflicts between roles. This can lead to unexpected or undesirable access scenarios, such as greater permissions than necessary being granted to a principal.
- **Considerations for managing access in multitenant environments:** Managing access control in a multitenant environment often requires you to create a large number of custom roles. Creating and maintaining custom IAM roles for each tenant can become unwieldy, leading to role explosion. You must also establish efficient processes for tenant onboarding and offboarding, including the provisioning and deprovisioning of access and resources to avoid security issues and cost inefficiencies.

## Best Practices for Using GCP Access Control

When working with GCP access control, there are a few tips you can keep in mind to get the most out of it:

- **Implement the principle of least privilege:** Make sure that your roles and policies only provide access to those principals that need it and that you only grant the minimal necessary access. Try to avoid using overly permissive roles like Owner as much as possible. Prefer predefined roles over basic roles wherever possible.
- **Regularly review and audit IAM policies:** Make sure that you keep a close eye on your GCP account access history. Enable Cloud Audit Logs to track and monitor user and system activity in your GCP account. Use Cloud Monitoring and Cloud Logging to set up alerts and monitor for suspicious activities. Regularly review permissions and policies to ensure your team members have only the permissions they need.
- **Utilize conditional bindings:** Make sure to implement IAM Conditions to add context-based restrictions to your IAM policies and permissions. For example, restrict access to resources based on IP addresses, device attributes, or time of day. You could also use conditional bindings to grant temporary access to principals, which gets autodisabled after a set period of time, to be deleted later.
- **Implement multifactor authentication (MFA):** As with most modern systems, enforce the use of MFA for all users to add an extra layer of security to their accounts.

## Conclusion

GCP Access Control is an important aspect of securing your cloud environment and ensuring that your organization's data and resources remain safe and secure. This article explored the key components, benefits, and best practices associated with GCP access control mechanisms.

You saw how the GCP IAM allows for fine-grained control over who can access what resources, emphasizing the principle of least privilege. You also learned about the shortcomings of the GCP access control methods. If you are looking to security your GCP environment with just in time access, self service, and full visibility over permissions and user authorization, make sure to check out ConductorOne.

ConductorOne is on a mission to modernize access controls to reduce standing privilege, boost team productivity, and enforce zero trust with self-service access, just-in-time provisioning, and automated access reviews. For modern companies, this often lives at the intersection of IT and security teams, who are responsible for managing and securing access across their workforce.