# 02 Cloud IAM Roles - KirkYagami🖼️🕵️

## What is IAM - Identity and Access Management?

### Identity / Principal / Member

In GCP, who can be considered as identity or principals:

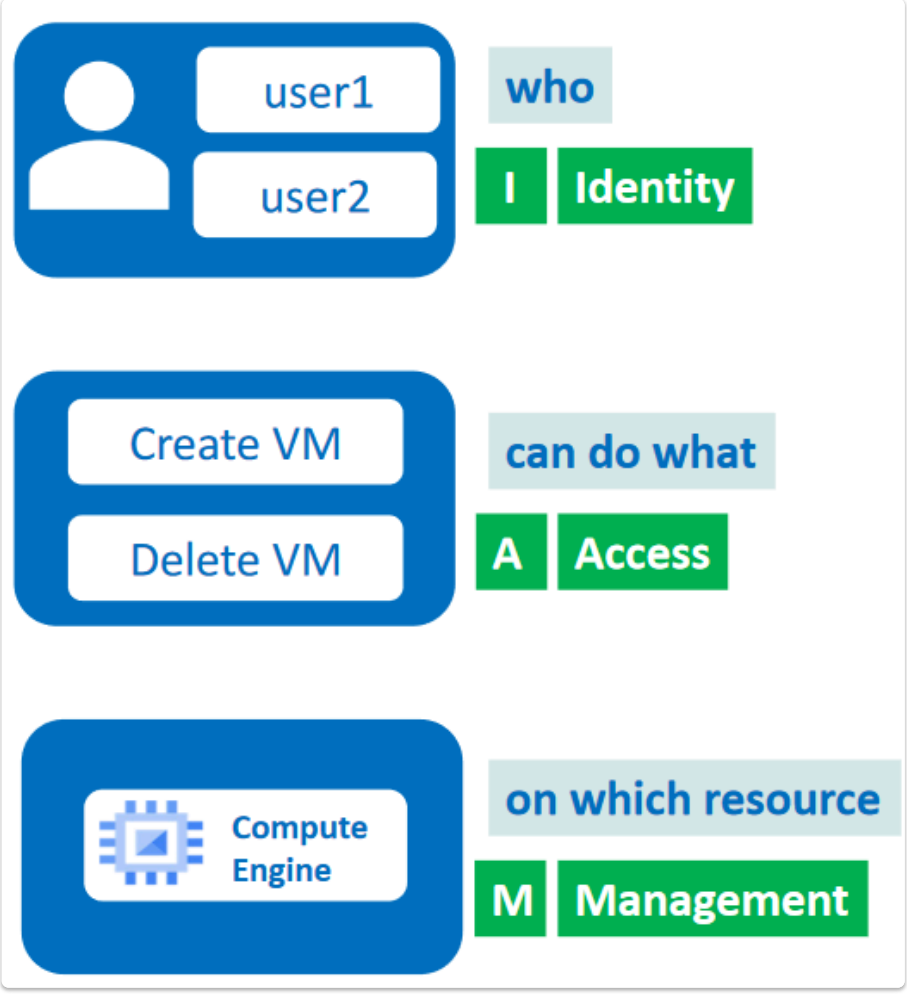
- Google Accounts
- Service Accounts
- Google Groups
- Google Workspace Accounts
- Cloud Identity Domain

### Access: Roles with Permissions

- **Role:** Compute Admin, Storage Admin
- **Permission:** compute.instance.create
- **Example:** Compute Admin (Role) can create, update, delete VM Instances (Permissions)

### Management: Manage access to resources

- **GCP Resources:**
  - Compute Engine
  - Google Kubernetes Engine
  - Cloud Run
  - Cloud SQL
  - GCP services

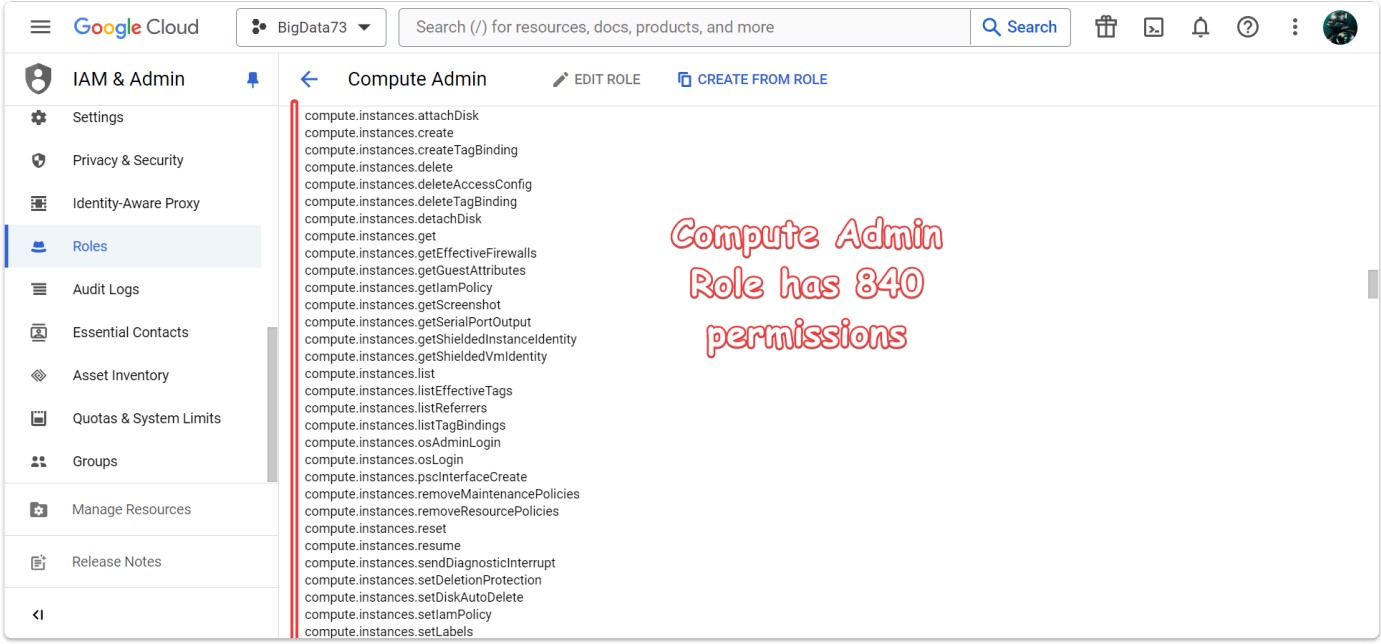



## What are IAM Permissions?

- IAM Permission: Permissions are operations that are allowed on a resource.
  - Example:
    - `compute.instances.create`
    - `compute.instances.list`
    - `compute.instances.start`
    - `compute.instances.stop`

## What are IAM Roles?

- IAM Roles: A role is a collection of Permissions.
  - Example:
    - Compute Admin
    - Storage Admin
  - When we associate / grant a role to a Principal (User), we are granting all permissions that role contains to that user.



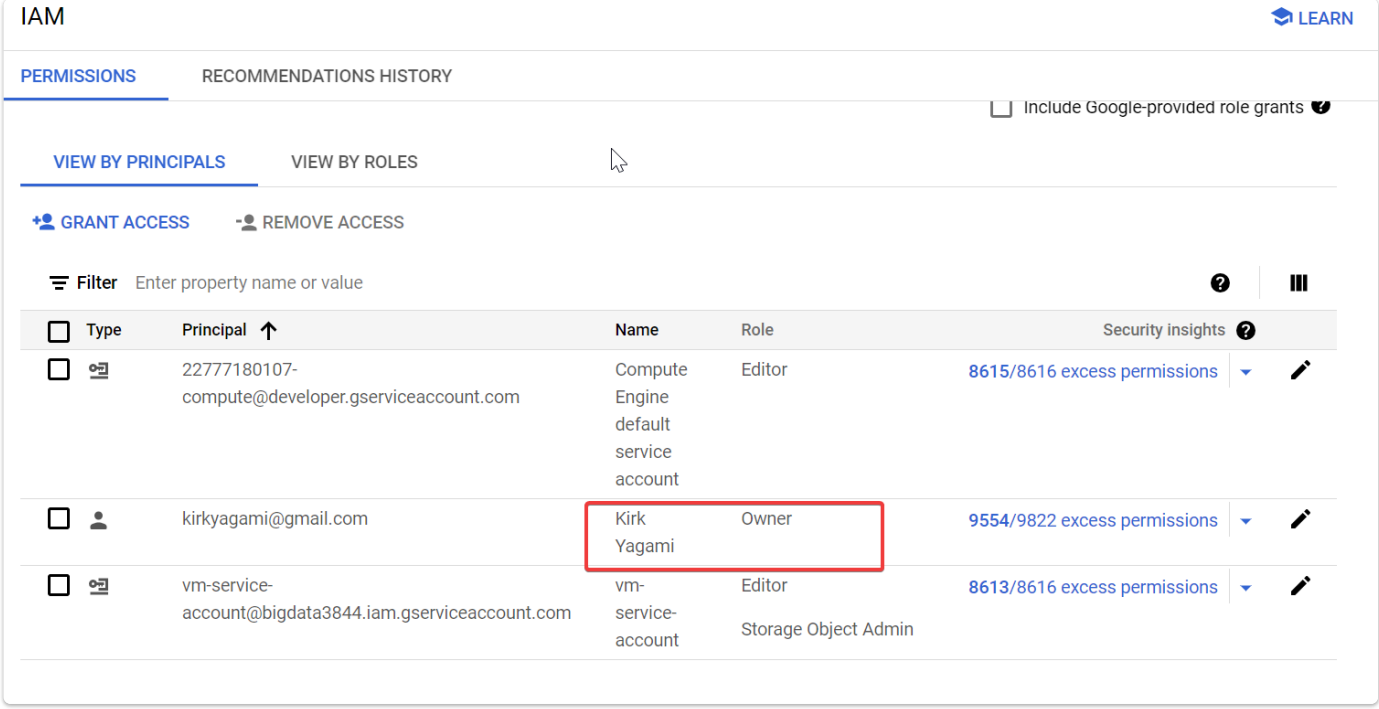

# IAM Role Types

- Basic Roles (Primitive Roles)
- Predefined Roles
- Custom Roles

## Basic Roles (Primitive Roles)

- OWNER: Full access
  - Example: When we created a Google Cloud account, the email ID we used will have this OWNER role assigned.
- EDITOR: Edit + View access across Google Cloud services
- VIEWER: View only or Read-Only access across Google Cloud services

Important Note: Assigning these basic roles to multiple users is not recommended. In short, NOT RECOMMENDED FOR PRODUCTION USE.




## Predefined Roles

- Pre-created by Google and ready to use
- Provides fine-grained access control
- Example Roles:
  - Compute Admin
  - Compute Viewer
  - Compute Network Admin
  - Compute Network Viewer
- Each role serves a different objective:
  - Compute Admin: Full access to Compute Engine
  - Compute Viewer: Read-only access to Compute Engine

| Title | Used in | Status |
|---|---|---|
| Compute Admin | Compute Engine | Enabled |
| Compute Future Reservation Admin | Compute Engine | Enabled |
| Compute Future Reservation User | Compute Engine | Enabled |
| Compute Future Reservation Viewer | Compute Engine | Enabled |
| Compute Image User | Compute Engine | Enabled |
| Compute Instance Admin (beta) | Compute Engine | Enabled |
| Compute Instance Admin (v1) | Compute Engine | Enabled |
| Compute Load Balancer Admin | Compute Engine | Enabled |
| Compute Load Balancer Services User | Compute Engine | Enabled |
| Compute Network Admin | Compute Engine | Enabled |
| Compute Network User | Compute Engine | Enabled |
| Compute Network Viewer | Compute Engine | Enabled |

## Custom Roles

- We can create a new role by assigning desired permissions to it.

### When do we create a Custom Role?

- When there is no predefined role satisfying our requirement, we can create a custom role.

### Example:

- We can create a custom role which will have permissions to stop and start a VM Instance.

| Type | Title | Used in |
|---|---|---|
| | Custom Compute Instance Delete Role 101 | Custom |
| | Custom Compute Instance Reset Role 101 | Custom |
| | custom-start-stop | Custom |

| | |
|---|---|
| ID | projects/gcplearn9/roles/CustomRole487 |
| Role launch stage | Alpha |

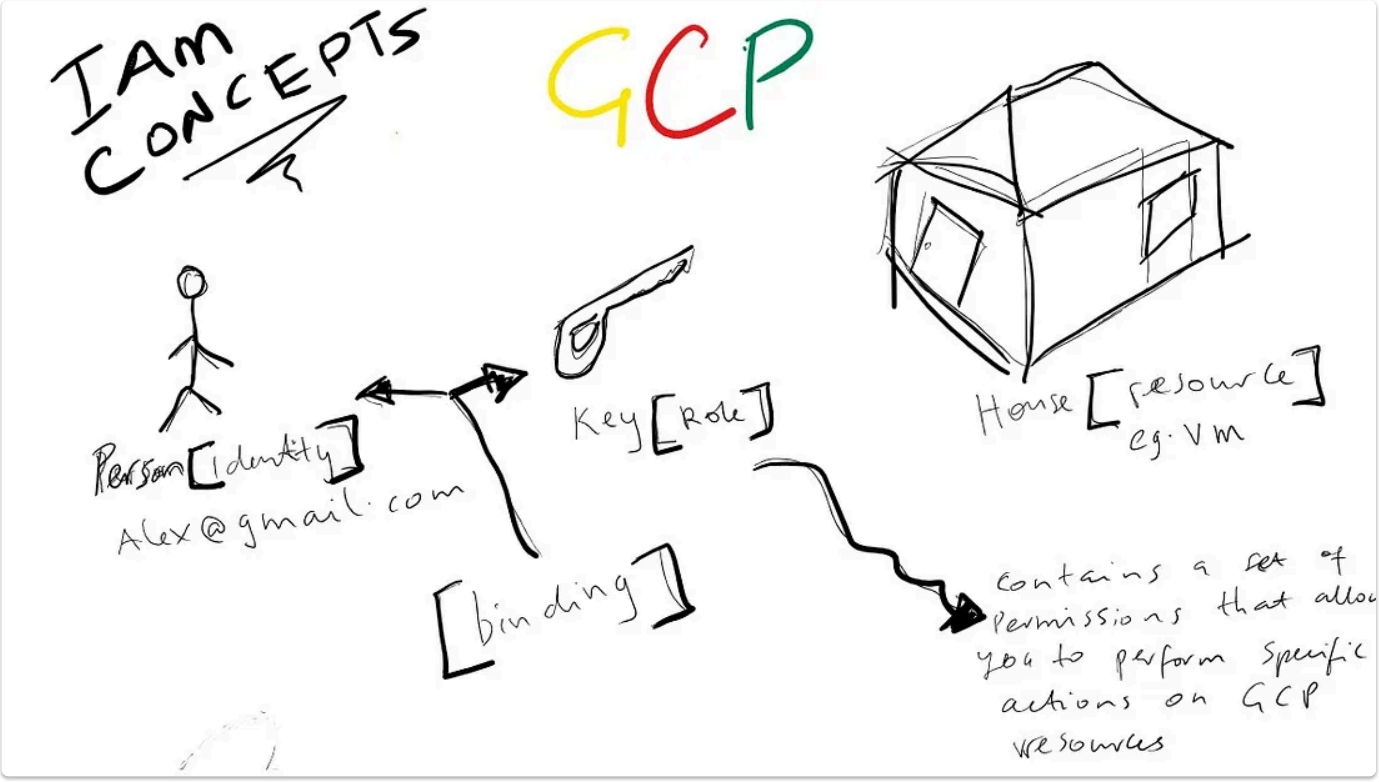
**Description**

Created on: 2024-04-15

**2 assigned permissions**

compute.instances.start
compute.instances.stop

https://towardsdatascience.com/google-cloud-iam-with-stick-figures-cd5ce19c142b




IAM concepts illustration — image by author

Explaining new concepts with analogies and examples can be very helpful. In this article, I will try to explain GCP's IAM (Identity and Access Management) concepts using doodles. GCP's IAM isn't particularly difficult, nevertheless this was fun to do and I hope my not so sophisticated doodles help you to understand IAM concepts a little more.

Before we begin, here is a map for each key concept we will cover so that you can follow along with the analogy.

- **Person**: In GCP, this is equivalent to a `principal` or `identity`. This could be an actual user account (alex@example.com), a service account or other entities. I will use 'principal' and 'identity' interchangeably.
- **Key**: Is equivalent to a `role` in GCP. For instance, alex@example.com could have a Cloud Storage Bucket admin role.
- **House**: This is equivalent to a `resource` in GCP. This could be a compute engine instance or BigQuery.

Using the image above as a guide, let's imagine the stick figure user (could be any GCP `identity` ) tries to gain access to a house (could be any `resource` like a Compute

Engine instance, in GCP). Some verifications would need to be carried out to ensure our user has the right ***permissions*** to access to that particular property. The user will need to be granted the right permissions to proceed.

> *In IAM, permission to access a resource isn't granted directly to the end user. Instead, permissions are grouped into roles, and roles are granted to authenticated principals.*

## So what is a Role?

> *A role is a collection of permissions. Permissions determine what operations are allowed on a resource. When you grant a `role` to a `principal`, you grant all the permissions that the role contains.*

## What are Permissions?

> *Permissions determine what operations are allowed on a resource. For example, 'invoking' a Cloud Function is an operation that can be performed on a Cloud Function. In the IAM world, permissions are represented in the form of `service.resource.verb`, for example, `pubsub.subscriptions.consume`.*

Continuing with our example from the preceding image, the *key* in the illustration will represent a `_role_` and this could have the following fictitious permissions assigned to it: `living.house.openFrontDoor` and `living.house.openKitchenDoor` and `living.house.openBasement`

Allowed permissions on a key (aka role) — image by author

Now in GCP, this is what a mapping of roles and permissions looks like conceptually:

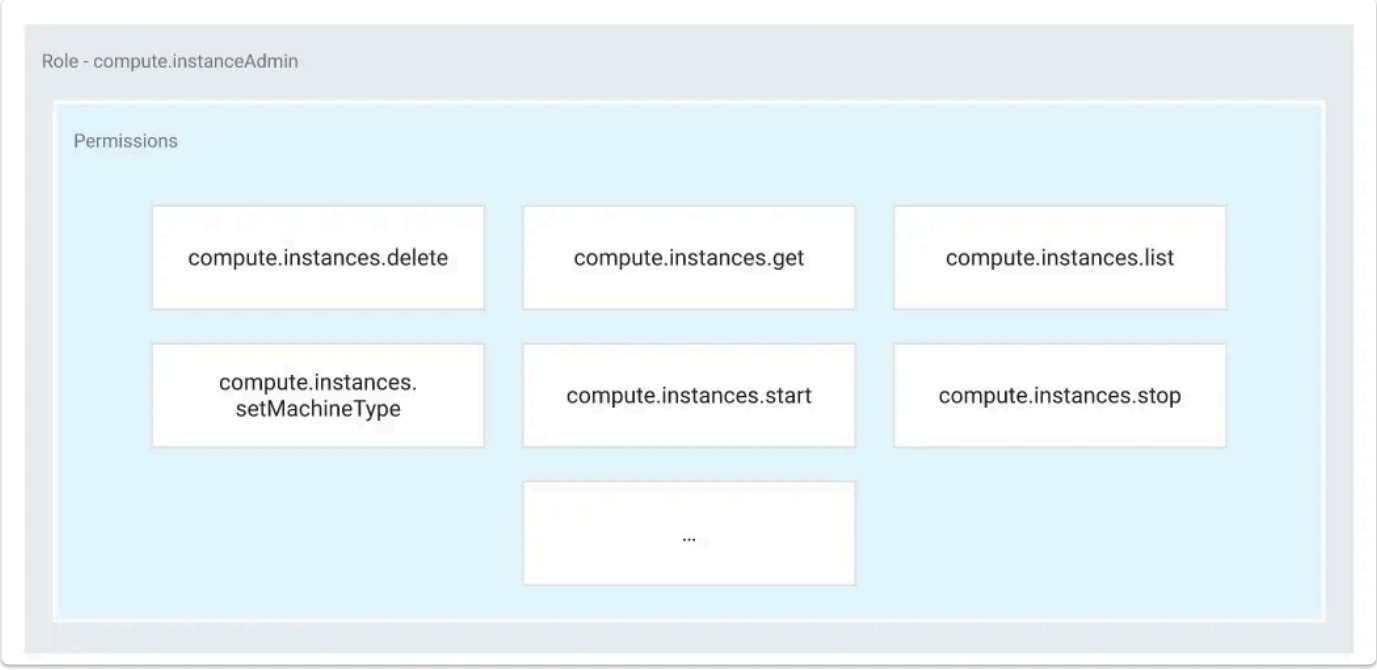



Image Source — [Google Cloud Platform]

Now that we have permissions assigned to the key, the next step in the process is to assign the key to our stick figure user. We also need to let the house master know what the user is allowed to do in the property. To do that, we need some sort of policy outlining authorized users and clearly defining what they are allowed to do.

To achieve this in GCP, each resource is assigned something called an ***IAM policy***.

## IAM Policy

> *An IAM policy defines and enforces what roles are granted to which identities. This policy is attached to a resource. If we had 20 Compute Engine instances, they will each have one IAM policy. Importantly, if you assign the policy to a GCP project, the user gains the specified roles across the project.*

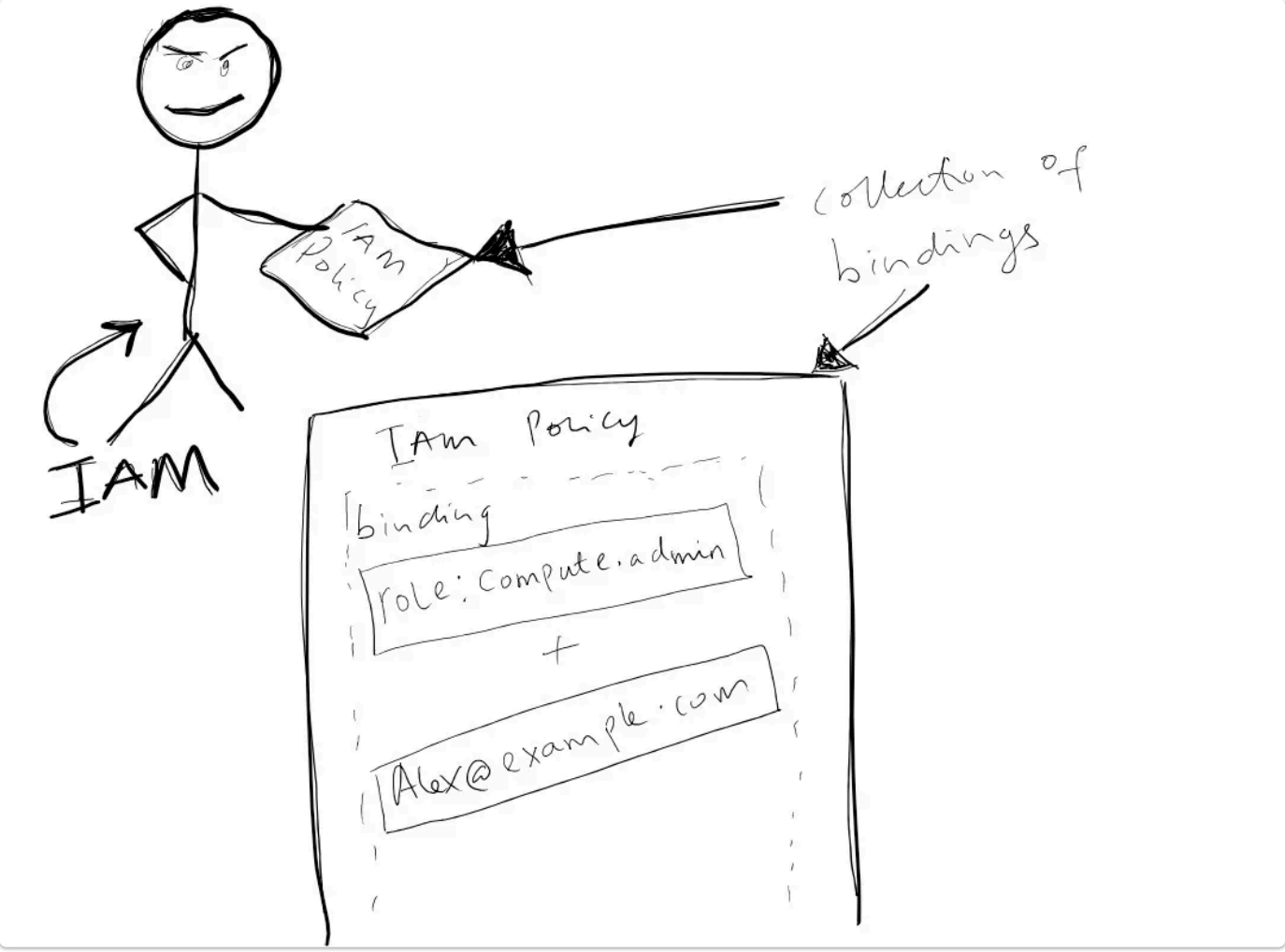
So in practice, an IAM policy is just a collection of bindings. A binding is what ties an identity to a role. Below is what an actual IAM policy looks like in GCP.

```
{
"bindings": [
{
"members": [
"user:alex@example.com"
],
"role": "roles/pubsub.subscriptions.consume"
}
```

],
"etag": "BwUjMhCsNvY=",
"version": 1
}

IAM policies can be in JSON or YAML format. You can read more about IAM policies here.

Finally, imagine we have a house master called 'IAM', standing guard in front of the property in the illustration. When our authenticated user attempts to access this property (aka `resource`), IAM checks the house's policy to determine whether the action the user is trying to perform is permitted.




IAM Policy illustration — Image by author

If the action is permitted according to the predefined policy, the user will be allowed to perform that action on the house (aka `resource`).

## Conclusion

We have looked at a few key concepts of IAM in Google Cloud and I hope the illustrations were helpful in getting a better understanding. IAM lets you grant granular access to specific Google Cloud resources and helps prevent access to other resources. IAM lets you adopt the security principle of least privilege, which states that nobody should have more permissions than they actually need.