IAM

Typical identity management in the cloud

- You have **resources** in the cloud (examples - a virtual server, a database etc)
- You have **identities (human and non-human)** that need to access those resources and perform actions
  - For example: launch (stop, start or terminate) a virtual server
- How do you **identify users** in the cloud?
  - How do you configure resources they can access?
  - How can you configure what actions to allow?
- In GCP: *Identity and Access Management (Cloud IAM)* provides this service

Cloud IAM

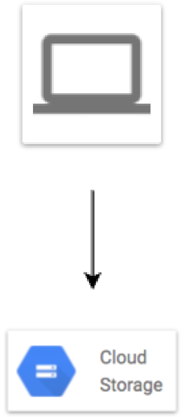
Cloud Identity and Access Management (IAM)

- **Authentication** (is it the right user?) and
- **Authorization** (do they have the right access?)
- **Identities** can be
  - A GCP User (Google Account or Externally Authenticated User)
  - A Group of GCP Users
  - An Application running in GCP
  - An Application running in your data center
  - Unauthenticated users
- Provides very **granular** control
  - Limit a single user:
    - to perform single action
    - on a specific cloud resource
    - from a specific IP address
    - during a specific time window

Cloud IAM

Cloud IAM Example

- I want to provide access to manage a specific cloud storage bucket to a colleague of mine:
  - Important Generic Concepts:
    - **Member**: My colleague
    - **Resource**: Specific cloud storage bucket
    - **Action**: Upload/Delete Objects
  - In Google Cloud IAM:
    - **Roles:** A set of permissions (to perform specific actions on specific resources)
      - **Roles do NOT know about members**. It is all about permissions!
    - How do you assign permissions to a member?
      - **Policy**: You assign (or **bind**) a role to a member
- 1: **Choose a Role** with right permissions (Ex: Storage Object Admin)
- 2: **Create Policy** binding member (your friend) with role (permissions)
- IAM in AWS is very different from GCP (Forget AWS IAM & Start FRESH!)
  - Example: Role in AWS is NOT the same as Role in GCP



IAM - Roles

- **Roles are Permissions:**
  - Perform some set of actions on some set of resources
- Three Types:
  - **Basic Roles (or Primitive roles)** - Owner/Editor/Viewer
    - **Viewer(roles.viewer)** - Read-only actions
    - **Editor(roles.editor)** - Viewer + Edit actions
    - **Owner(roles.owner)** - Editor + Manage Roles and Permissions + Billing
    - EARLIEST VERSION: Created before IAM
    - NOT RECOMMENDED: **Don't use in production**
  - **Predefined Roles** - Fine grained roles predefined and managed by Google
    - Different roles for different purposes
    - **Examples:** Storage Admin, Storage Object Admin, Storage Object Viewer, Storage Object Creator
  - **Custom Roles** - When predefined roles are NOT sufficient, you can create your own custom roles

Cloud IAM

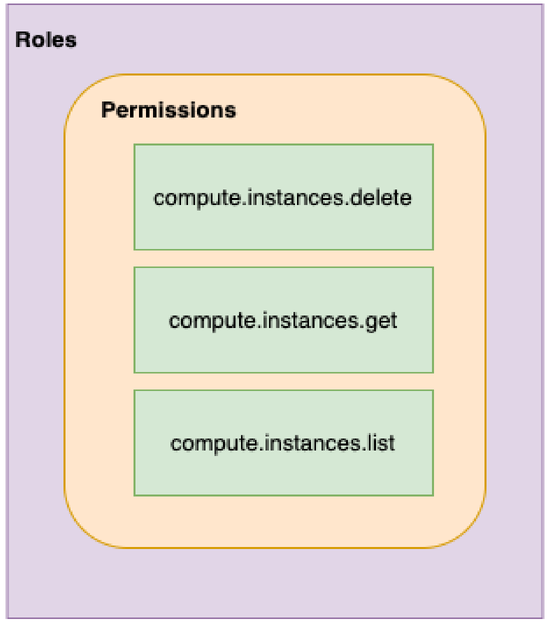
IAM - Predefined Roles - Example Permissions

- Important **Cloud Storage Roles**:
  - **Storage Admin (roles/storage.admin)**
    - storage.buckets.*
    - storage.objects.*
  - **Storage Object Admin (roles/storage.objectAdmin)**
    - storage.objects.*
  - **Storage Object Creator (roles/storage.objectCreator)**
    - storage.objects.create
  - **Storage Object Viewer (roles/storage.objectViewer)**
    - storage.objects.get
    - storage.objects.list
- All four roles have these permissions:
  - resourcemanager.projects.get
  - resourcemanager.projects.list

Cloud IAM

IAM - Most Important Concepts - A Review

- **Member** : Who?
- **Roles** : Permissions (What Actions? What Resources?)
- **Policy** : Assign Permissions to Members
  - Map Roles (What?) , Members (Who?) and Conditions (Which Resources?, When?, From Where?)
  - Remember: Permissions are NOT directly assigned to Member
    - Permissions are represented by a Role
    - Member gets permissions through Role!
- A Role can have multiple permissions
- You can assign multiple roles to a Member



IAM policy

- Roles are assigned to users through **IAM Policy** documents
- Represented by a **policy object**
  - Policy object has list of bindings
  - A binding, binds a role to list of members
- Member type is identified by **prefix**:
  - Example: user, serviceaccount, group or domain

Cloud IAM

IAM policy - Example

```
{
    "bindings": [
        {
            "role": "roles/storage.objectAdmin",
             "members": [
                 "user:you@in28minutes.com",
                 "serviceAccount:myAppName@appspot.gserviceaccount.com",
                 "group:administrators@in28minutes.com",
                 "domain:google.com"
             ]
        },
        {
            "role": "roles/storage.objectViewer",
            "members": [
                "user:you@in28minutes.com"
            ],
            "condition": {
                "title": "Limited time access",
                "description": "Only upto Feb 2022",
                "expression": "request.time < timestamp('2022-02-01T00:00:00.000Z')",
            }
        }
    ]
}
```

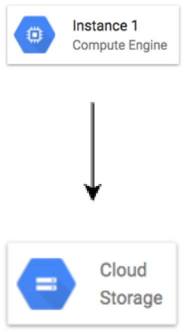
Playing With IAM

- **gcloud**: Playing with IAM
  - gcloud **compute project-info** describe - Describe current project
  - gcloud **auth login** - Access the Cloud Platform with Google user credentials
  - gcloud **auth revoke** - Revoke access credentials for an account
  - gcloud **auth list** - List active accounts
  - gcloud **projects**
    - gcloud projects **add-iam-policy-binding** - Add IAM policy binding
    - gcloud projects **get-iam-policy** - Get IAM policy for a project
    - gcloud projects **remove-iam-policy-binding** - Remove IAM policy binding
    - gcloud projects **set-iam-policy** - Set the IAM policy
    - gcloud projects **delete** - Delete a project
  - gcloud **iam**
    - gcloud iam **roles describe** - Describe an IAM role
    - gcloud iam **roles create** - create an iam role(--project, --permissions, --stage)
    - gcloud iam **roles copy** - Copy IAM Roles

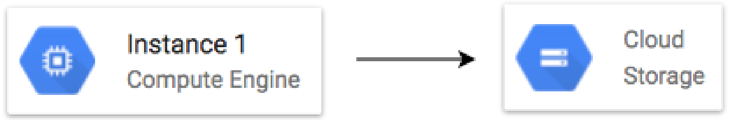
Cloud Shell

Service Accounts

- Scenario: An Application on a VM needs access to cloud storage
  - You DONT want to use personal credentials to allow access
- (RECOMMENDED) Use **Service Accounts**
  - Identified by an email address (Ex: id-compute@developer.gserviceaccount.com)
  - Does NOT have password
    - Has a **private/public RSA key-pairs**
    - Can't login via browsers or cookies
- Service account types:
  - **Default service account** - Automatically created when some services are used
    - (NOT RECOMMENDED) Has **Editor role** by default
  - **User Managed** - User created
    - (RECOMMENDED) Provides fine grained access control
  - **Google-managed service** accounts - Created and managed by Google
    - Used by GCP to perform operations on user's behalf
    - In general, we DO NOT need to worry about them



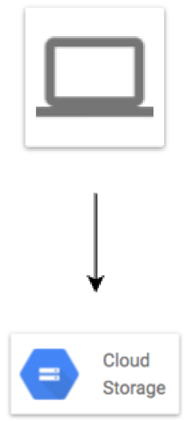
Use case 1 : VM <-> Cloud Storage



- **1:** Create a Service Account Role with the right permissions
- **2:** Assign Service Account role to VM instance
- **Uses Google Cloud-managed keys:**
  - Key generation and use are automatically handled by IAM when we assign a service account to the instance
  - Automatically rotated
  - No need to store credentials in config files
- **Do NOT delete** service accounts used by running instances:
  - Applications running on those instances will lose access!

Use case 2 : On Prem <-> Cloud Storage (Long Lived)

- You CANNOT assign Service Account directly to an On Prem App
- **1:** Create a **Service Account** with right permissions
- **2:** Create a **Service Account User Managed Key**
  - *gcloud iam service-accounts keys create*
  - Download the service account key file
    - Keep it secure (It can be used to impersonate service account)!
- **3:** Make the service account key file accessible to your application
  - Set environment variable GOOGLE_APPLICATION_CREDENTIALS
    - *export GOOGLE_APPLICATION_CREDENTIALS="/PATH_TO_KEY_FILE"*
- **4:** Use **Google Cloud Client Libraries**
  - Google Cloud Client Libraries use a library - Application Default Credentials (ADC)
    - ADC uses the service account key file if env var GOOGLE_APPLICATION_CREDENTIALS exists!



Use case 3 : On Prem <-> Google Cloud APIs (Short Lived)

- **Make calls from outside GCP to Google Cloud APIs** with short lived permissions
  - Few hours or shorter
  - **Less risk** compared to sharing service account keys!
- **Credential Types:**
  - OAuth 2.0 access tokens
  - OpenID Connect ID tokens
  - Self-signed JSON Web Tokens (JWTs)
- **Examples:**
  - When a member needs elevated permissions, he can assume the service account role (Create OAuth 2.0 access token for service account)
  - OpenID Connect ID tokens is recommended for service to service authentications:
    - A service in GCP needs to authenticate itself to a service in other cloud

Service Account Use case Scenarios

Scenario	Solution
Application on a VM wants to talk to a Cloud Storage bucket	Configure the VM to use a Service Account with right permissions
Application on a VM wants to put a message on a Pub Sub Topic	
Configure the VM to use a Service Account with right permissions	
Is Service Account an identity or a resource?	It is both. You can attach roles with Service Account (identity). You can let other members access a SA by granting them a role on the Service Account (resource).
VM instance with default service account in Project A needs to access Cloud Storage bucket in Project B	In project B, add the service account from Project A and assign Storage Object Viewer Permission on the bucket

ACL (Access Control Lists)

- **ACL**: Define **who** has access to your buckets and objects, as well as **what level** of access they have
- **How is this different from IAM?**
  - IAM permissions apply to all objects within a bucket
  - ACLs can be used to customized specific accesses to different objects
- User gets access if he is allowed by either IAM or ACL!
- (Remember) **Use IAM for common permissions** to all objects in a bucket
- (Remember) **Use ACLs** if you need to **customize access to individual objects**

Cloud IAM

Access Control - Overview

In28 Minutes

- How do you control access to objects in a Cloud Storage bucket?
- Two types of access controls:
  - **Uniform** (Recommended) - Uniform bucket level access using IAM
  - **Fine-grained** - Use IAM and ACLs to control access:
    - Both bucket level and individual object level permissions
- Use Uniform access when all users have same level of access across all objects in a bucket
- Fine grained access with ACLs can be used when you need to customize the access at an object level
  - Give a user specific access to edit specific objects in a bucket

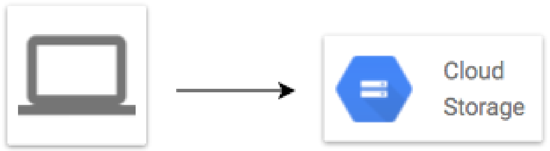
Cloud IAM

Cloud Storage - Signed URL

- You would want to **allow a user limited time access** to your objects:
  - Users do NOT need Google accounts
- Use **Signed URL** functionality
  - A URL that gives **permissions for limited time duration** to perform specific actions
- **To create a Signed URL**:
  - **1:** Create a key (YOUR_KEY) for the Service Account/User with the desired permissions
  - **2:** Create Signed URL with the key:
    - *gsutil signurl -d 10m YOUR_KEY gs://BUCKET_NAME/OBJECT_PATH*

Cloud Storage

Cloud Storage - Static website



- **1:** Create a bucket with the **same name** as website name (Name of bucket should match DNS name of the website)
  - **Verify** that the domain is owned by you
- **2:** Copy the files to the bucket
  - Add index and error html files for better user experience
- **3:** Add member **allUsers** and grant **Storage Object Viewer** option
  - Select **Allow Public Access**

IAM - Scenarios

Scenario	Solution
An Application on a GCE VM needs access to cloud storage	Use a Service Account (Google Cloud-managed keys)
An Application on premises needs access to cloud storage	Use Service Account User Managed Key
Allow a user limited time access to your objects	Signed URL
Customize access to a subset of objects in a bucket	Use ACL (Access Control Lists)
Permission is allowed by IAM but NOT by ACL. Will user be able to access the object?	Yes.