Course: Securing and Integrating Components of your Application

Module 1: Handling Authentication and Authorization

Introduction

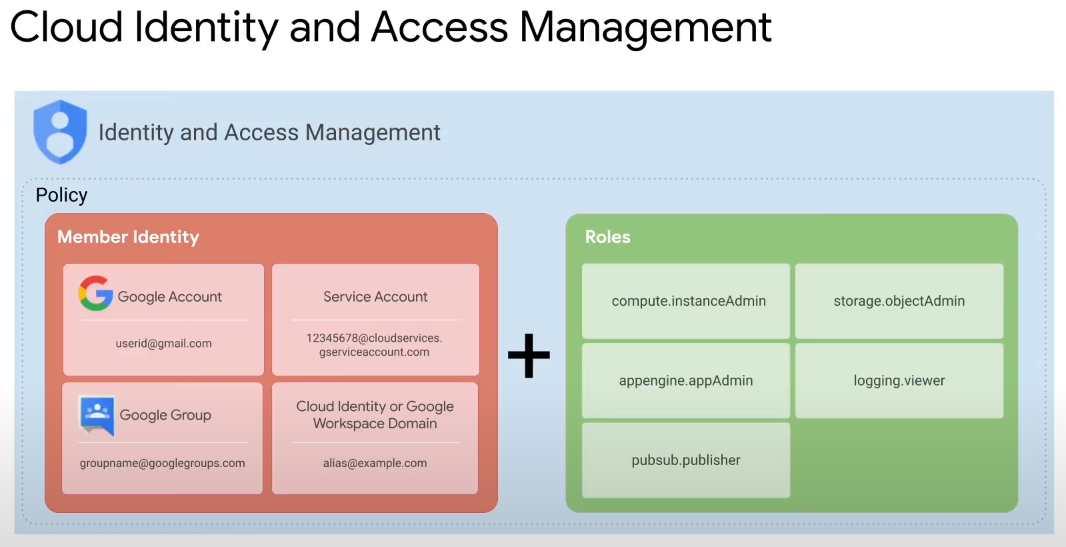
Design application architecture with components (micro-services) that respond to business events

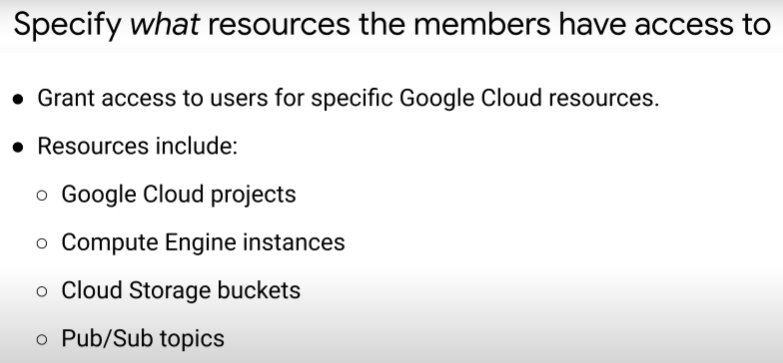
* want: loosely coupled, highly scalable
* The point-to-point communications can be brittle links, prevent scaling in response to high demand
* use Cloud Pub Sub, Cloud Functions and Cloud Endpoints to integrate components
* use Firebase to authenticate and authorize users. Firebase SDK to implement Federated Identity Management.
* use Identity-Aware Proxy to control access to applications
* Apply security principle of least privilege: minimum access needed to perform task.

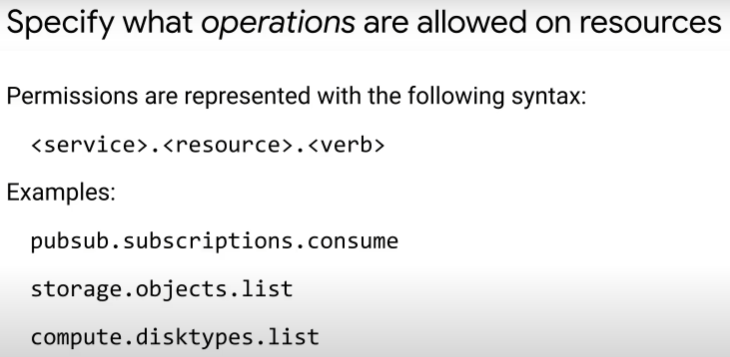
IAM Concepts

Who (members) >> has what kind of access (roles, operations allowed to perform) >> to what resources.

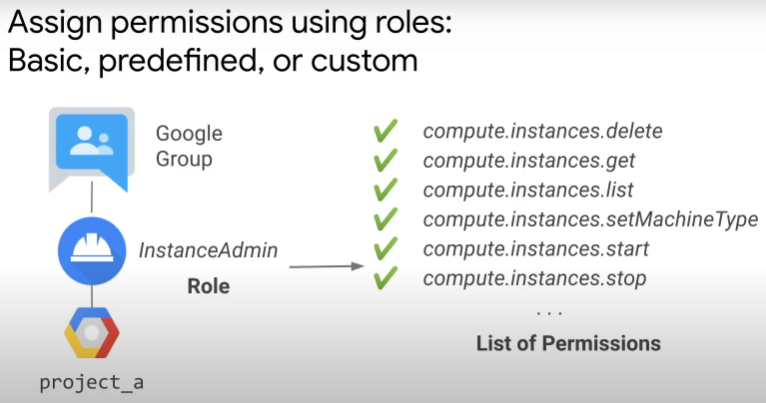
* validate users against credentials stored in Google Cloud IAM platform



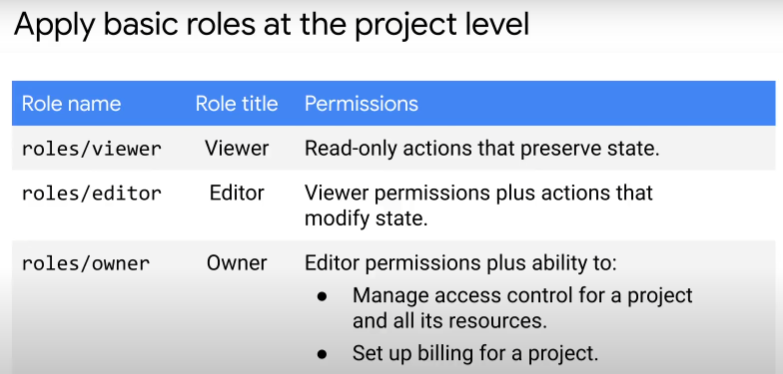
* Who >> types of members
  + Google Account
    - email account as identity
    - usually a person interacting with Google Cloud, e.g. dev, administrator
  + Service Account
    - used by applications to authenticate themselves and invoke Google Cloud APIs
  + Google group
    - a collection of Google Accounts and service accounts
    - identified by group’s unique email address
    - easy way to apply access policy to a group of users
    - does not have login credentials to establish identity and make request to access resource.
  + Google Workspace domain
    - a virtual group for work
    - the Google Workspace account can be associated with a domain name
      * each user identified with an email account yourName@domain.com
    - convenient permission management
    - cannot be used for login to establish identity
  + Cloud Identity domain
    - for members of an organisation
    - each domain user has email [yourName@domain.com](mailto:yourName@domain.com)
    - like Google Workspace domain, but without the Workspace applications and features
* to What resources
  + examples:
* has What kind of access
  + permissions determine what operations allowed on a resource



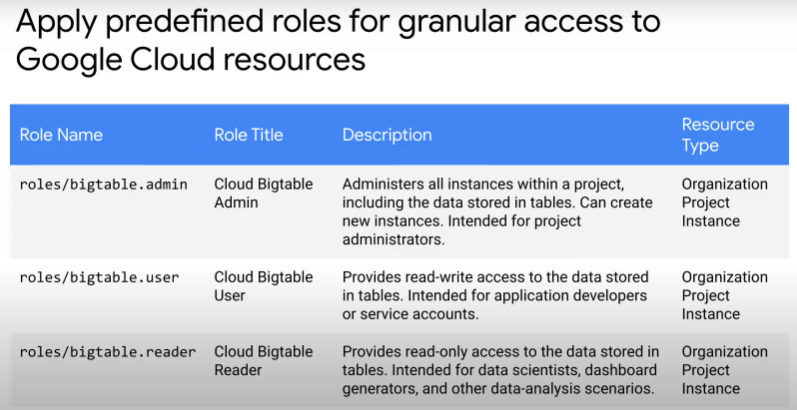
* + - permission usually correspond 1:1 to a REST method
    - caller of API >> REST method >> must have required permissions to use Google Cloud Service
  + a role is a collection of permissions
    - you grant a role to a user/group/service account; cannot assign permission directly
    - 3 kind of roles: basic, predefined and custom



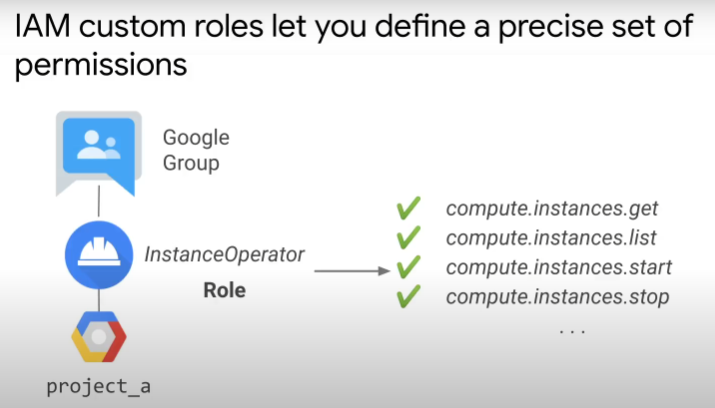
* + Basic roles:
    - viewer >> editor >> owner
    - hierachy: organisation is the root node >> folder >> project >> resource
      * every resource only 1 parent
    - can be applied at project level using Cloud Console, API, gcloud command line tool
    - owner role at lower resource level (e.g. pubsub topic) does not grant owner role for entire parent project
    - owner role at organisation level cannot change organisation’s metadata, but can modify all projects under that organisation.



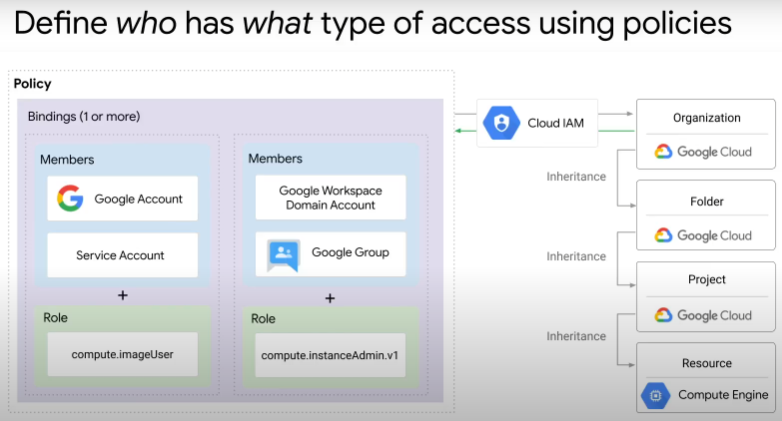
* + Predefined roles:
    - for granular access to specific resource, and prevent unwanted access to other resources
    - same user can be granted multiple roles



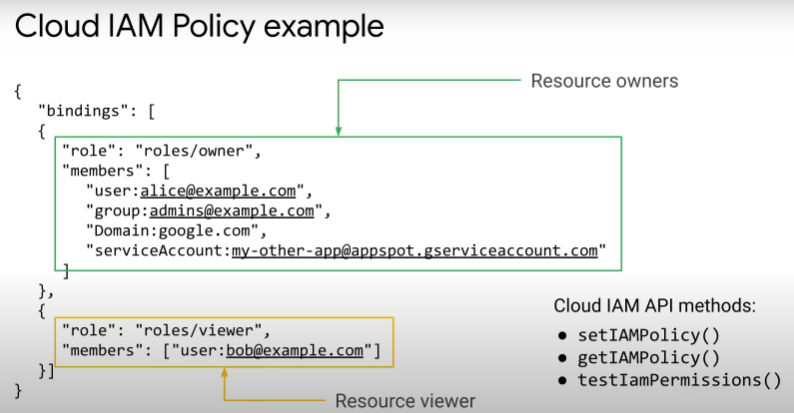
* + Custom roles:
    - fine-grain, custom set of permissions
    - follow least privileged model: only have minimal permissions needed to do job
    - need to actively manage permissions that make up custom roles. May have additions, changes to scope, deprecation etc.
    - Can only be used at organisation and project level. Not at folder level.



* Policy
  + attached to a resource
    - controls who has what kind of access to resources

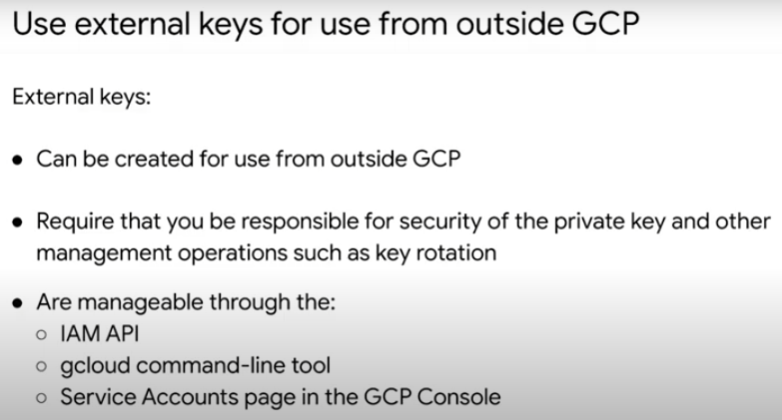


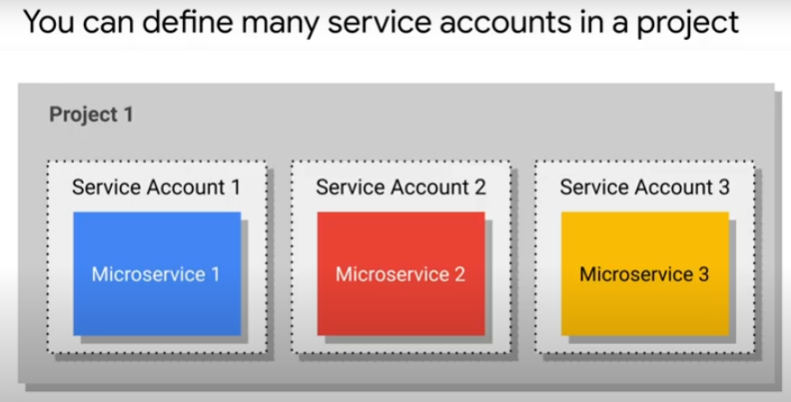
* + Cloud IAM policy is represented by a policy object
    - made up of a list of bindings
    - each binding ties a list of members to a role
    - API methods for IAM include to:
      * set IAM policy: set policies on resources
      * get IAM policy: retrieve info
      * test IAM permissions: test if caller of API has specific permissions for this resource

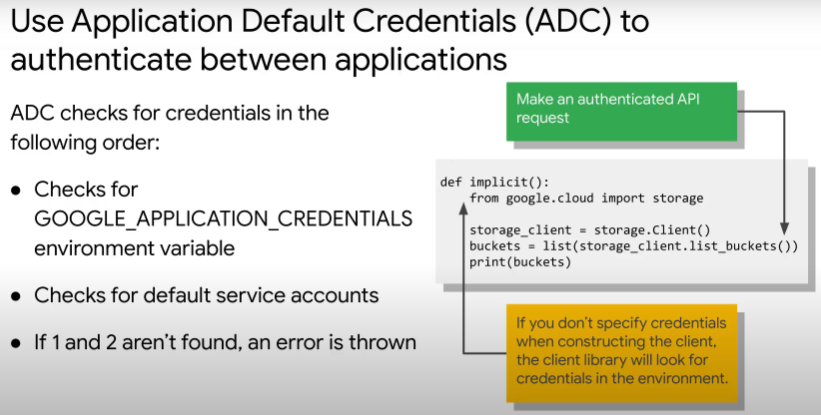
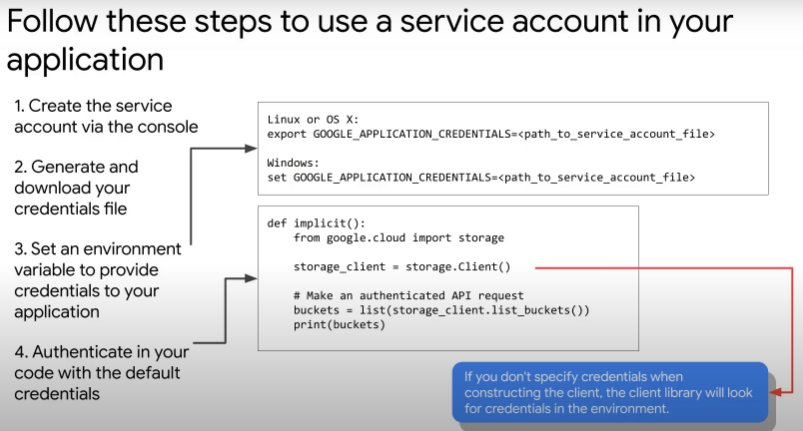


IAM Best Practices





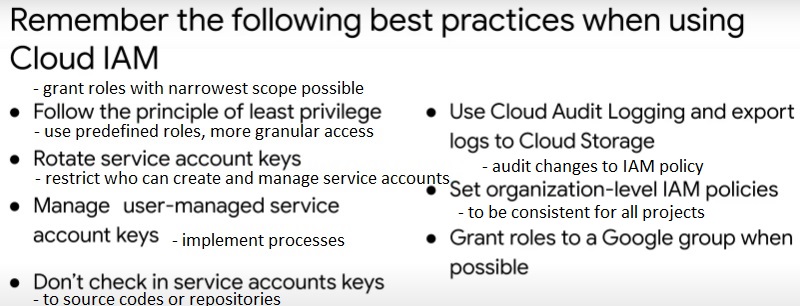
* To call APIs that do not access private user data, option to use API key (token).
  + Useful for clients e.g. browser and mobile applications, that do not have back-end server
  + used to track API requests under a project, for billing and quota
* A potential architecture design:
  + within single cloud project
  + let each microservice have its own service account
  + specify which other services a service account can call
  + achieve granular control



Credentials for

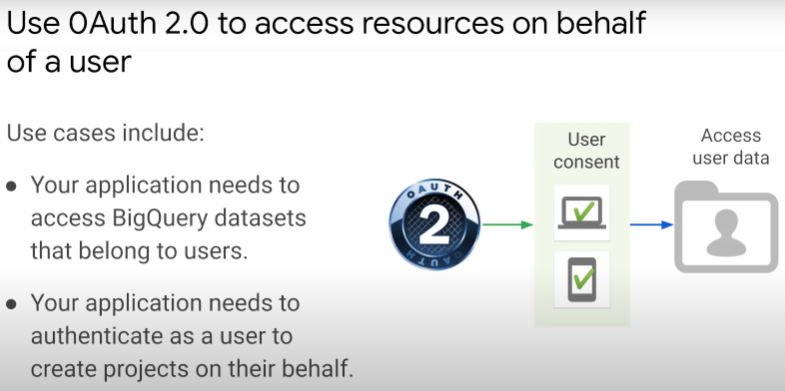
* quota and billing
* authorize access to APIs, resources and features.

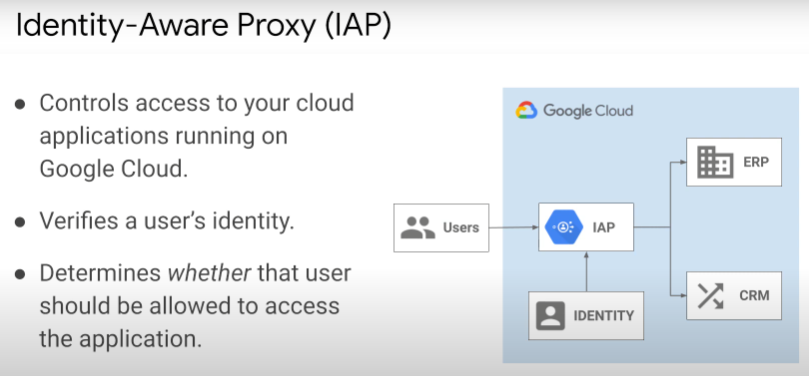
GCP client libraries use ADC to find credentials for your application.

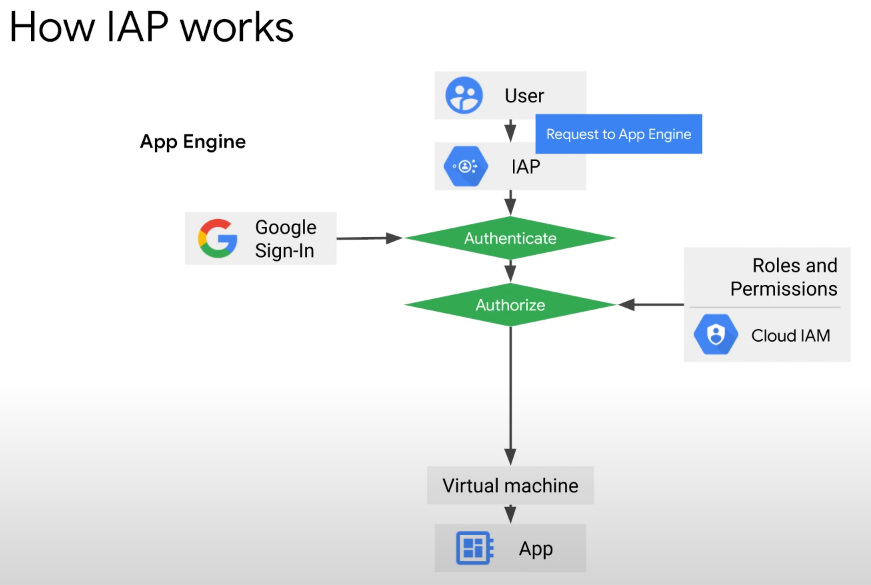
* Default service account referred above is for services like Compute Engine, GKE, App Engine, Cloud Functions.

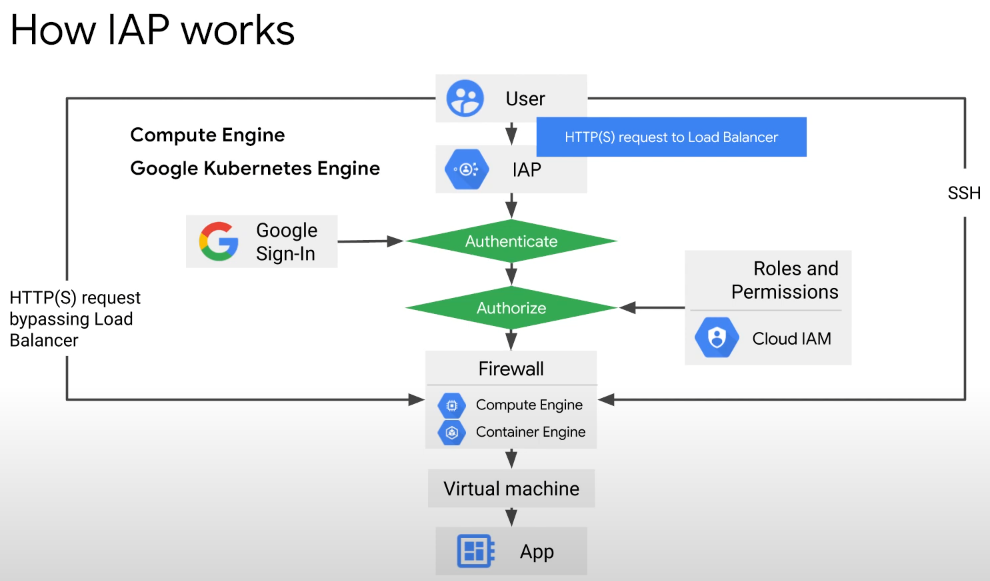
OAuth2.0, IAP & Firebase Authentication

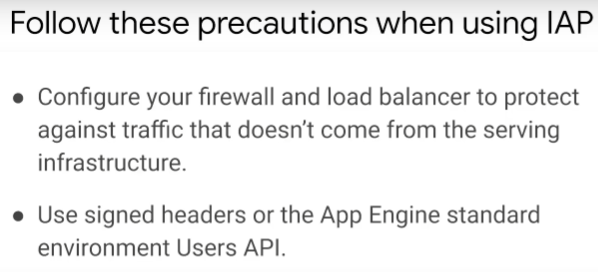
Client application uses service account for authentication to Google Cloud API

* for example, when want to access protected resource on behalf of user of that application
* use OAuth2.0 for better security
* process:
  1. application requests access to resource
  2. resource returns an authorization code .. prompt user for consent
  3. if user consents, application requests for credentials from an authorization server
  4. application receives credentials (Access Token) and uses them to access resources

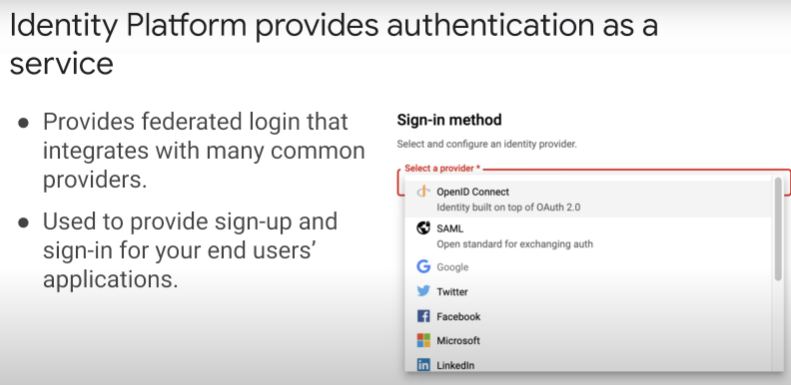
IAP

* a proxy that controls access to protected applications and Cloud resources
* verifies identity of users and groups and only those with correct Cloud IAM roles can access resources
* all access traffic are by HTTPS
* this is an access control model at the application level
* it does not rely on network firewalls or VPN
* granting access via IAP subject users to fine-grained access controls implemented by Google Cloud product.
* IAP performs authentication and authorization checks whenever a user tries to access
* Works with Compute Engine, GKE, AppEngine



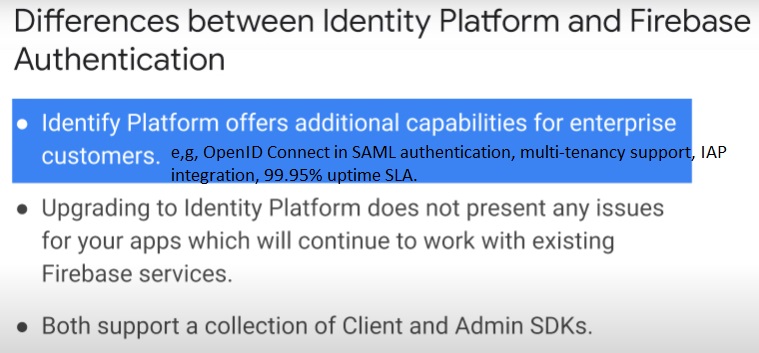


**Identity Platform**

* customer identity
* access management platform
* provides authentication as a service
* developers use SDKs.
* Can be used to support authentication methods such as SAML, OpenID Connect, email and password, phone.
* Works with identity providers like Apply, Facebook, Twitter.

Similarities between Identity Platform and Firebase

* both sign users into applications
* provide back-end services, SDKs and UI libraries.



Authentication overview documentation

<https://cloud.google.com/docs/authentication/>

Identity Platform website

https://cloud.google.com/identity-platform

Lab: App Dev - Adding User Authentication to your Application: Java

<https://www.cloudskillsboost.google/course_sessions/1924225/labs/345050>