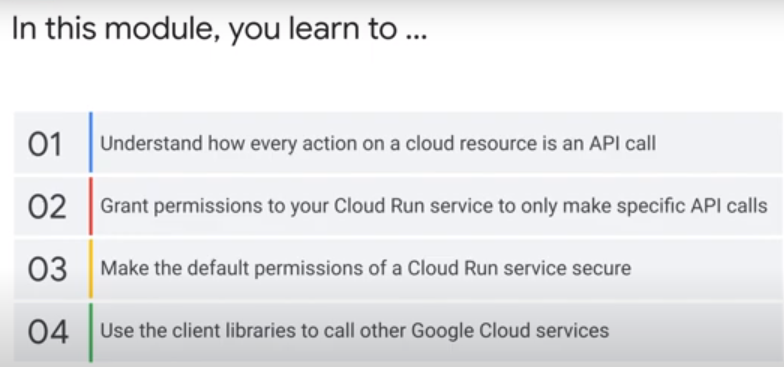
Application Development with Cloud Run

### Service Identity and Authentication

#### Introduction

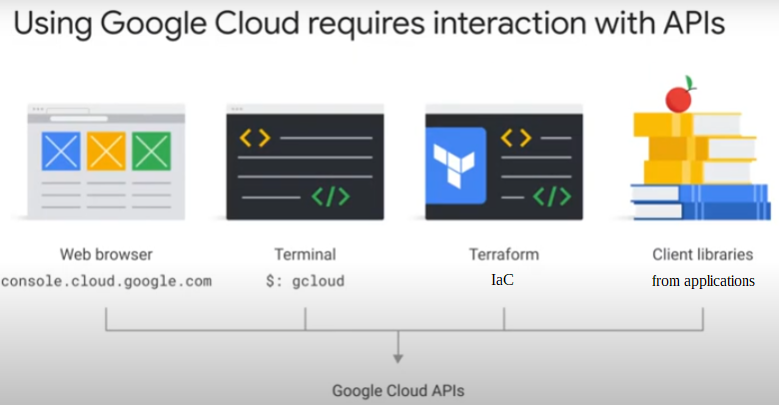


* Broad default permissions can compromise entire Google Cloud Project
* protect using IAM



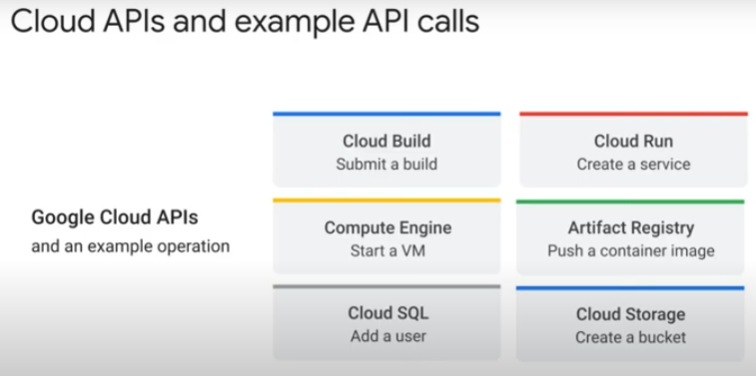
2. applications use official client libraries to call other services

### Introduction to IAM

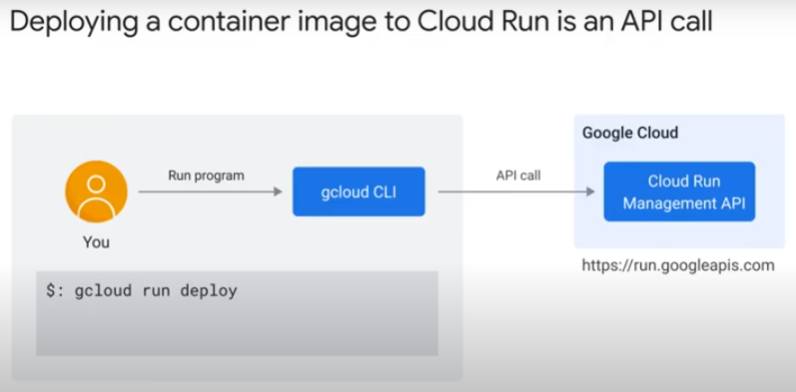


* Collection of APIs that allow you to create and manage virtual resources
  + e.g. VMs, Cloud Run services, load balancers, PubSub topic, database table on Cloud SQL server.
* 4 ways to interact with APIs as shown in diagram.

Examples of Google Cloud services and an example of an operation which is an API call

* use any of the 4 ways to make API calls

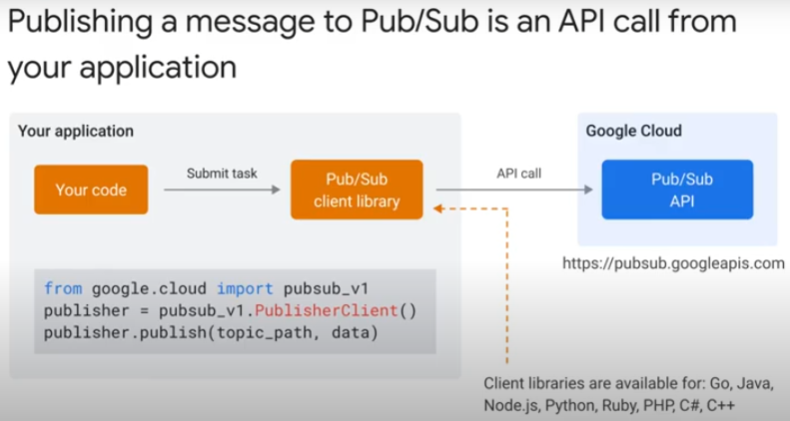
Example 1: Execute “gcloud run deploy” in CLI terminal:



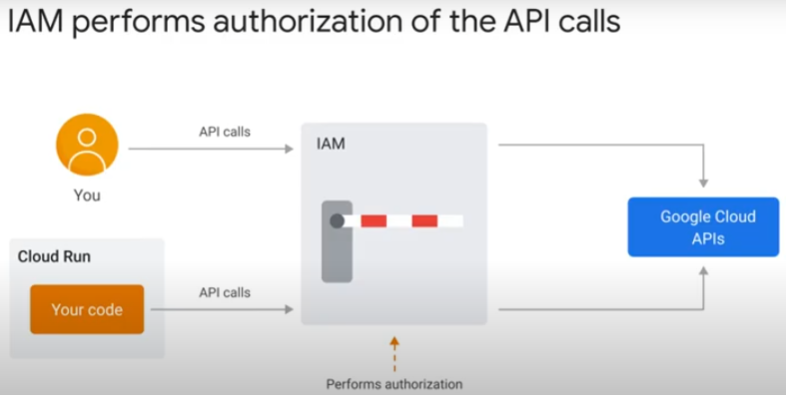
* command to deploy a container

1. First, Cloud shell invokes gcloud program.
2. Gcloud parses command and makes API call to run.googleapis.com

Example 2: Publish message to PubSub in a client Python application



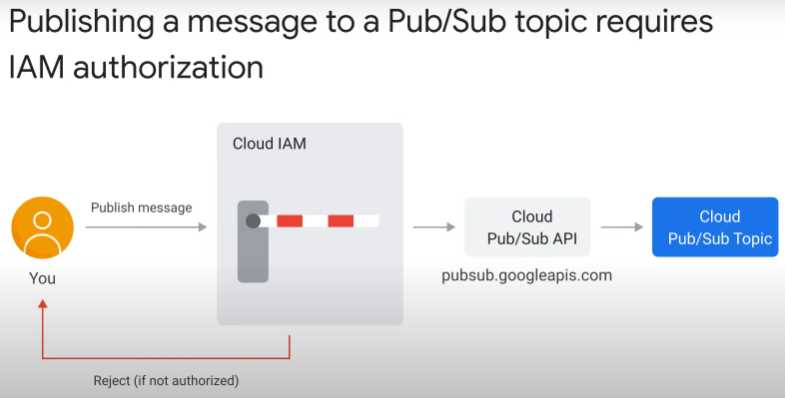
* PubSub is a message broker
* use it to asynchronously send messages between services
* use PubSub client library from your application to publish a message
* client library sends API call to pubsub.googleapis.com
* there are Node.js, Java, Go, Python, Ruby, PHP, C#, C++ client libraries for most Google Cloud services



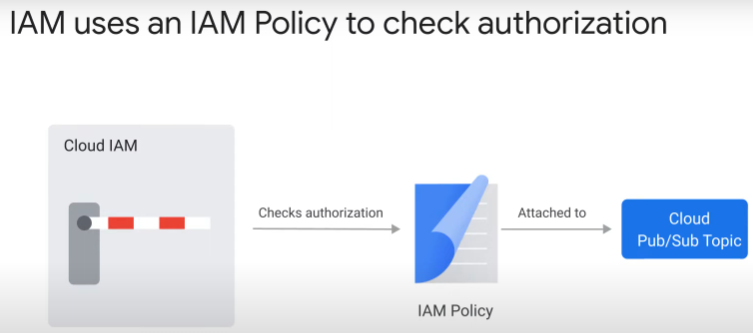
* IAM checks **all** API calls
  + IAM verifies identity of caller and checks if it has permission/authorization to make that API call
  + if fails, rejects API call

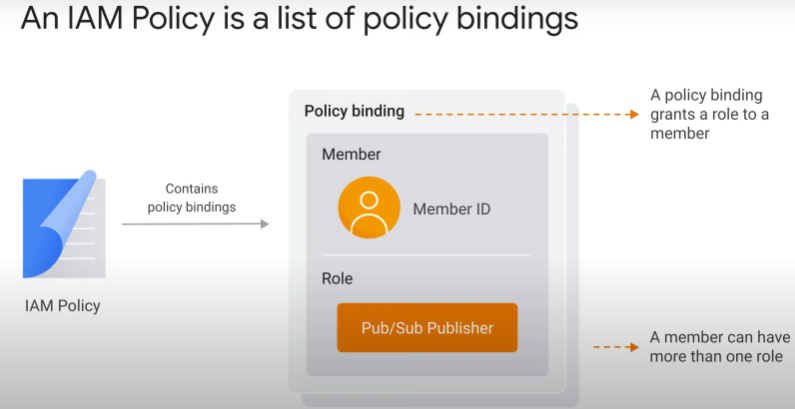
### IAM Policy

IAM -> IAM Policy attached to resource -> policy binding -> binds identity to role (permissions) -> each permission : 1 API call

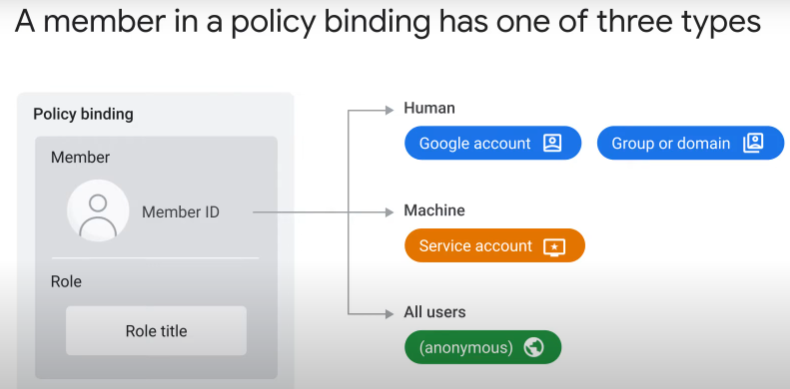


How IAM authorizes API call:

* use example of you using Gcloud to publish message to PubSub topic
* Gcloud tries to send API call to pubsub.googleapis.com
* IAM inspects HTTP request
  + identifies user by credentials in API request
  + checks IAM policy attached to resource (PubSub topic)
    - authorized or not?
* An IAM policy is a list of policy bindings
  + a policy binding binds an identity to a role
  + if have policy binding, IAM lets the API call through
    - e.g. if Tom have PubSub.Publisher role, Tom can publish messages to PubSub topic the IAM policy is attached to.
  + An identity can have multiple policy bindings
    - means, you can grant more than one role to an identity



Possible types of identities in a policy binding:



1. Human

* an individual
  + e,g, one installs Gcloud SDK, initializes it, logs in using Google account, and Gcloud performs API call
* a group or domain
  + useful for setting up access policies for project
    - add/remove individuals from group
  + can use federated identity management with an existing identity system, e.g. Microsoft Azure Active Directory

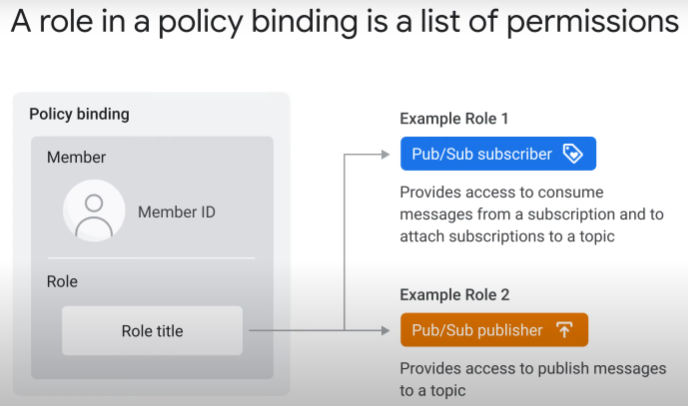
2. Service account

* by machines on Google Cloud
  + e.g. VMs, Cloud Run services, Cloud Function, PubSub.

3. All users (annoymous)

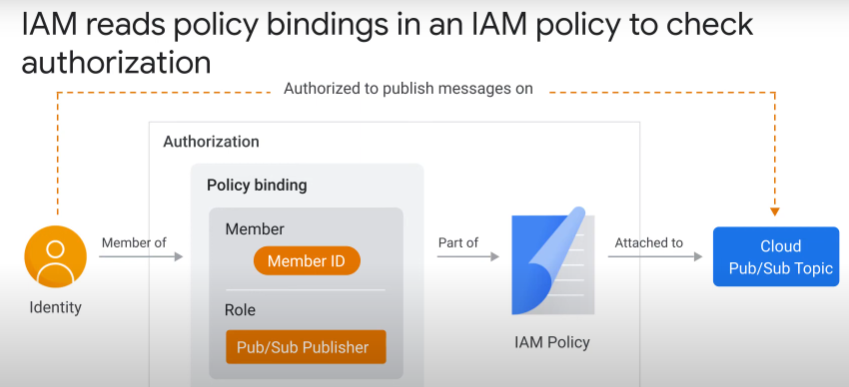
* e.g. to allow upload to Cloud bucket, HTTP request to Cloud Run service

Role in policy binding is a list of permissions:



* each permission usually corresponds to an API call

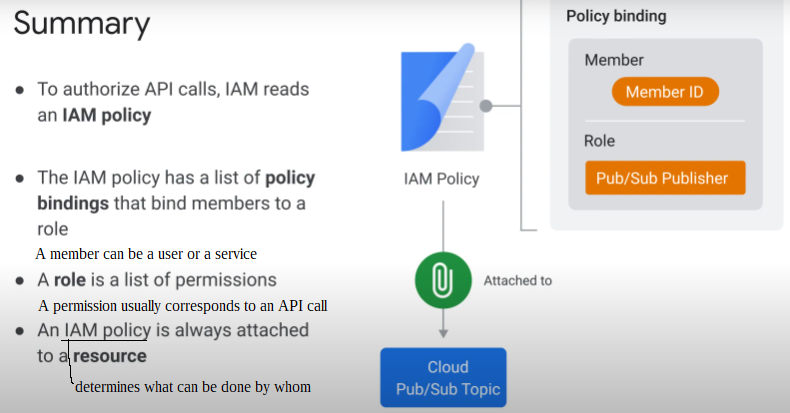
Overall:

The parts:

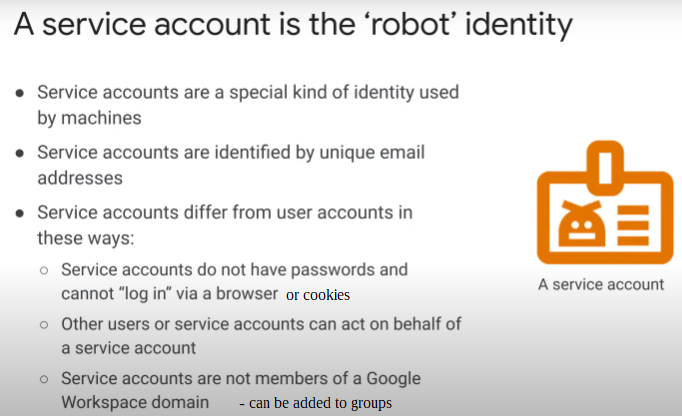
* IAM policy attached to a resource (Cloud PubSub topic)
* binding that is part of this IAM policy
  + has role “pubsub.Publisher”
  + identity (you) are a member of this binding

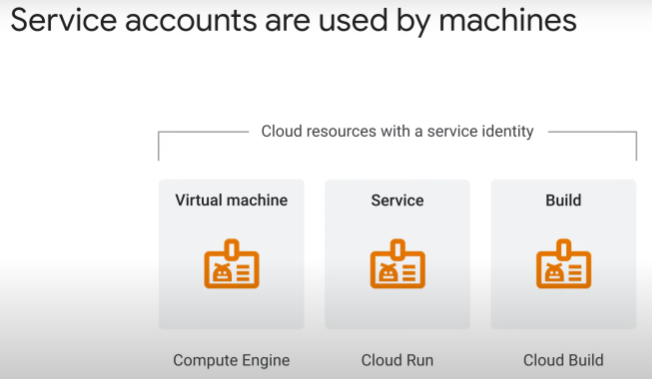
=> IAM policy grants permission for identity to do action.

Summary:

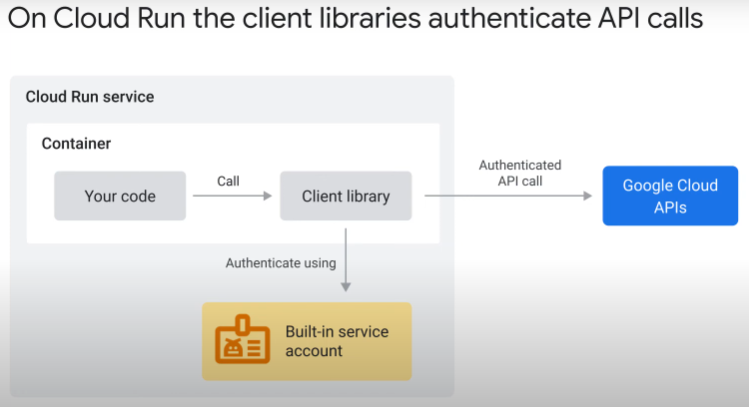


#### Service Identity



* If run code on some Cloud resources, you have to use a built-in service account.
  + It is related to a service identity.
  + if an application use client library to connect to Google Cloud API, the client library will use a similar service account to authenticate.
* can replace default service account with custom user-managed service account.

Example for application code running on Cloud Run:

* container with application runs on Cloud Run Worker.
  + The worker has a service account built-in. Its service identity.
* If use client library to publish message to PubSub queue, client library retrieves an access token from an authentication server > proves the identity of the service account.
* IAM
  + verifies the access token
  + checks the policy binding to see if the verified identity has correct role bound.
  + If have permission, calls the PubSub API.

#### Resource Hierachy

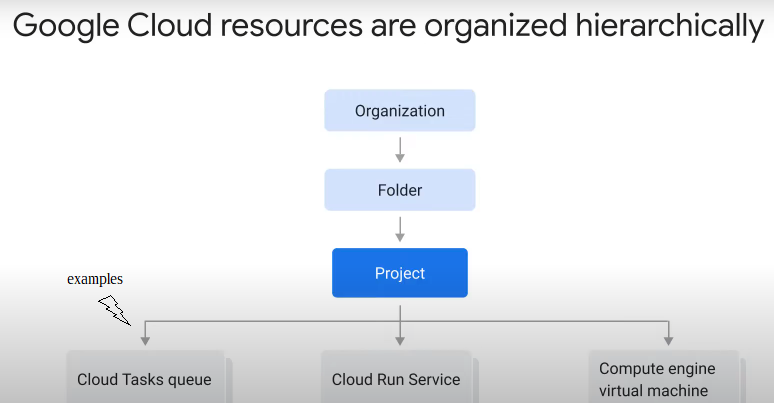
Resource hierachy can simplify permissions.

Google Cloud resources are organised hierachically

* have parent-child relationship
  + each resource has exactly one (1) parent, except for the top organization node (none).
* inheritance of permission

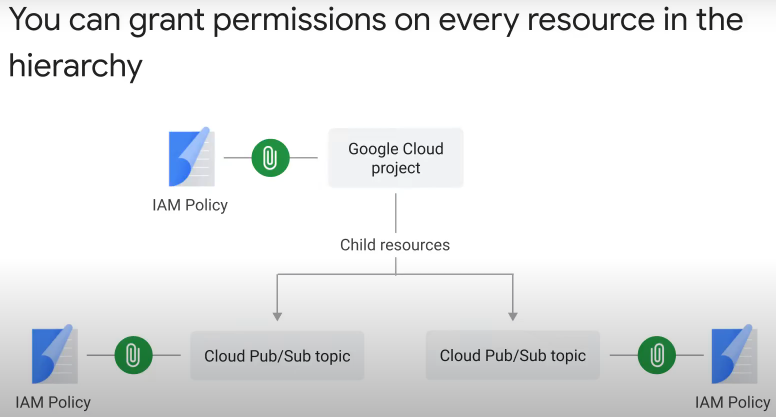
A GCP **Project** is the main way to organise resources.

* every resource must be part of a project



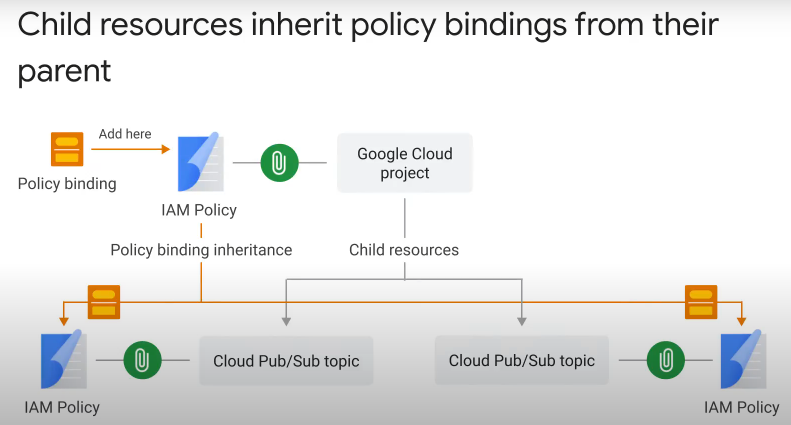
* If you use Google Workspace or Cloud Identity (<https://cloud.google.com/identity/>),
  + your projects can be organised into folders (departments)
  + at the very top, only one (1) organisation node.

Every resource is attached to an IAM policy:



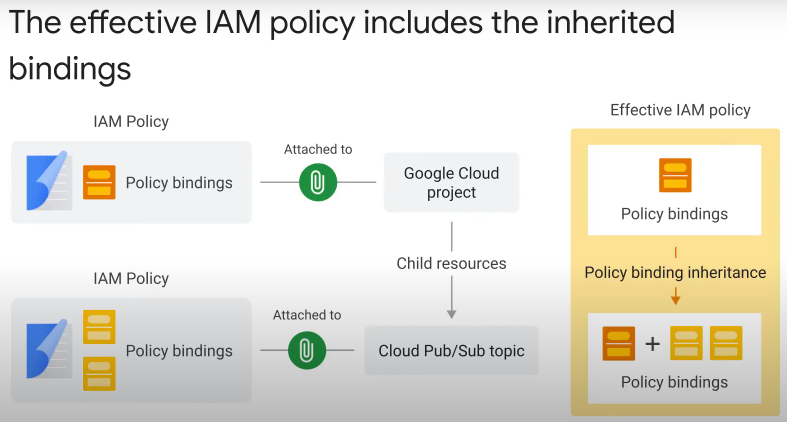
* the IAM policy
  + grant permissions for actions on resource using policy bindings
  + policy binding binds an identity to role(s)
  + a role is a list of permissions on the attached resource
  + each permission usually corresponds to an API

Child resource inherits policy binidngs of parents:



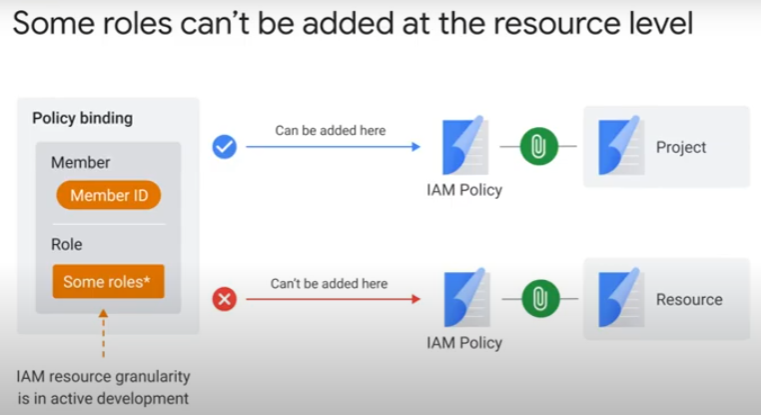
* example: “pubsub.Publisher” role attached at project level instead of a specific PubSub topic
  + all lower-level resources will inherit policy bindings from all parents/ancestors
  + grant permissions to publish messages to all topics in project. Not just 1 topic.
* Another use case: a role that needs to create new child resources > bind IAM policy at project level > inhirited by all child resources

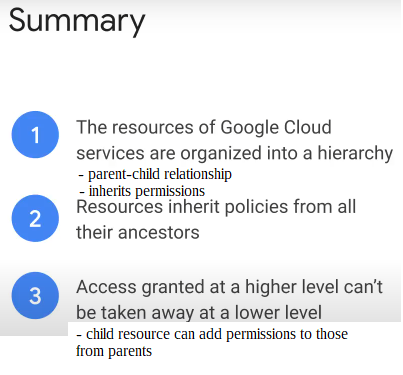
The effective IAM policy for a resource is the sum of its own and all inherited policy bindings:



* IAM evaluates policy bindings for a resource and its parent resources
* cannot ignore/remove permissions granted at parent level.

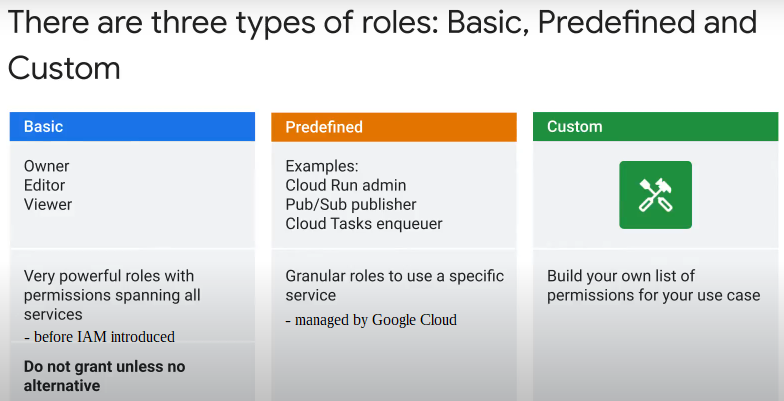
Some roles/permissions can only apply to all resources of that type in a project:

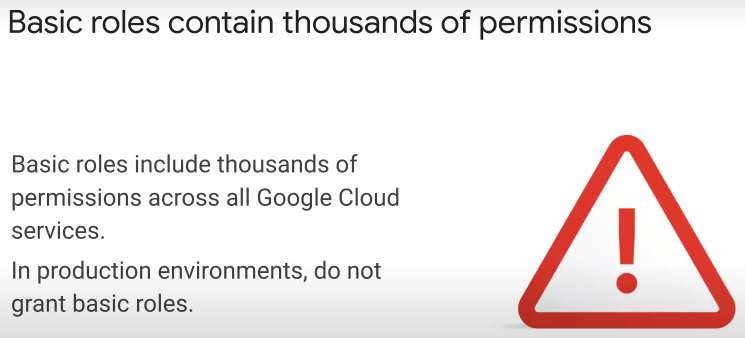
* cannot apply to one specific resource instance only.
* no finer role granularity
* example: Cloud SQL client role. Permits connection to SQL instances, but only to all instances. Cannot choose a particular instance.
* Role granularity is in active development, may have changes



#### Least Privilege Principle

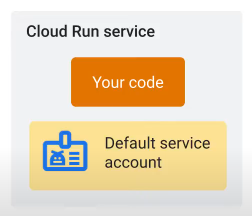
3 types of roles:

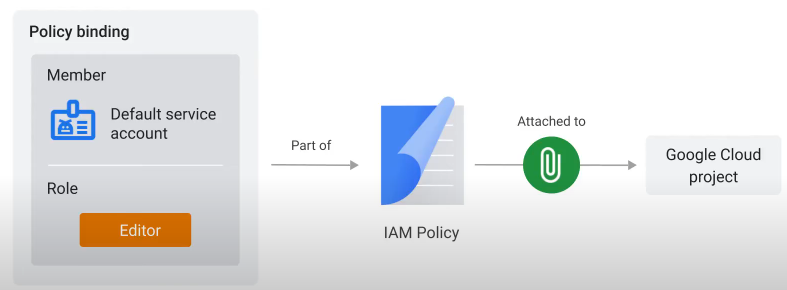


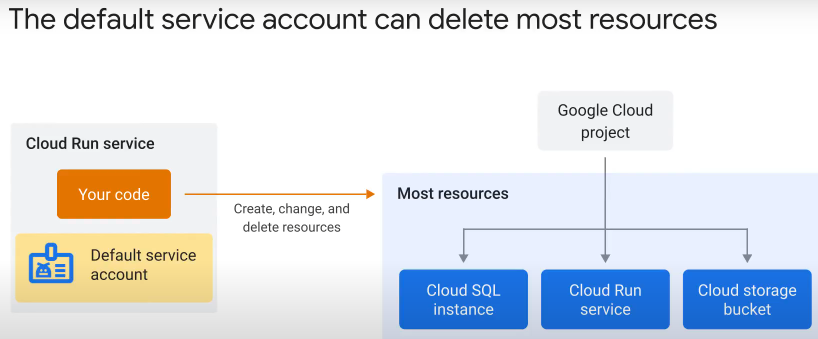


* grant limited pre-defined role, or custom role, that meets need
* any extra unneeded permission granted increases security risk

Example: Cloud Run service uses default service account if you do not specify one

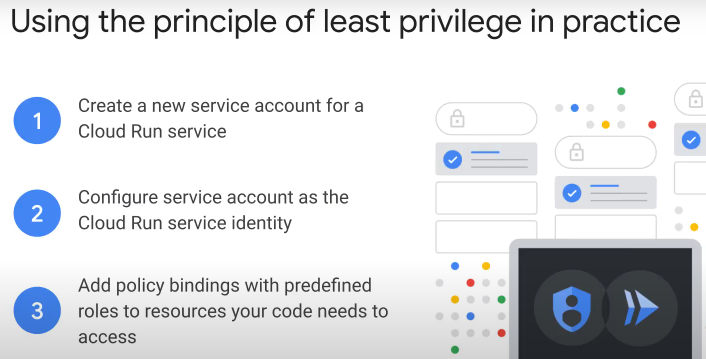
* default service account is inherited from project level
  + comes with policy binding of editor role
    - a basic role with permissions to do a lot





* default service account (editor role) has permissions to modify most resources

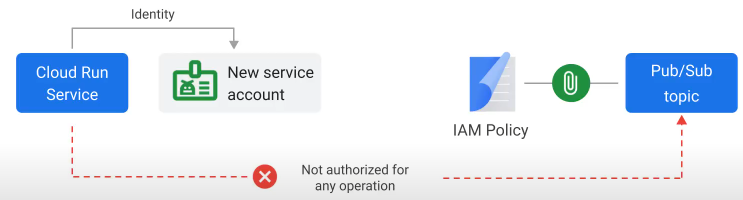
The correct approach:



Steps to take as above:

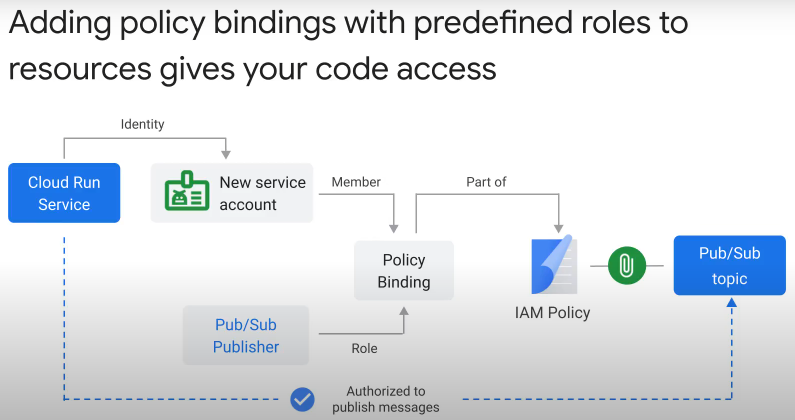
1. Create new service account dedicated for Cloud Run service.

* Set as service identity for that Cloud Run service
* “gcloud iam service-accounts create [name]”
* No policy bindings or permissions yet
  + any API call will be rejected by IAM



2. Then add policy binding

* for role of PubSub.publisher
* to the IAM policy attached to the PubSub topic of interest
  + this will give service account permissions on the PubSub topic
* “gcloud run services add-iam-policy-binding [name] –member [unique email address of service account] –role roles/xxx”
* principle of least privilege – give needed correct permissions only



#### Summary

