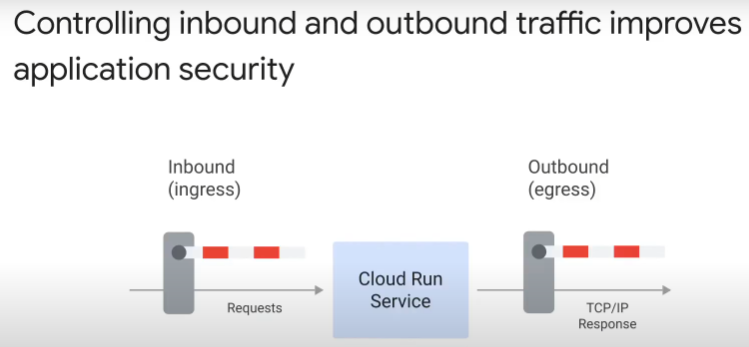
Application Development with Cloud Run

### Controlling Inbound and Outbound Requests

#### Introduction



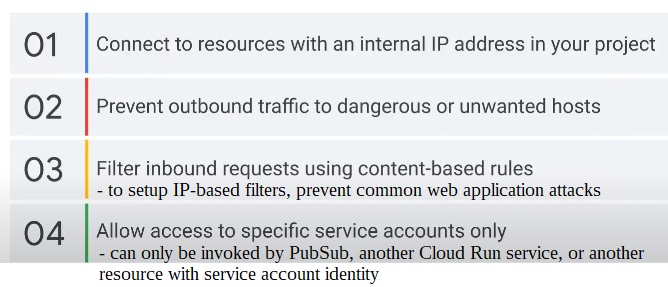
All applications

* receive incoming inbound ingress requests
* deliver outgoing outbound egress responses

Fine control over network traffic improves security

|  |  |
| --- | --- |
| Inbound | Outbound |
| restrict inbound to certain geographical (IP) address | prevent outbound connections to unknown hosts |
| restrict inbound to certain service accounts only | outbound to resources with internal IPs within VPC only (not a default) |
| disable default run.app domain | want Cloud Run services in project to communicate with each other only |

In this module, you will learn to:



#### Google Cloud Armor

https://cloud.google.com/armor/docs/cloud-armor-overview

=> Use Layer 7 content-based rules to filter inbound requests

Two routes to Cloud Run service:

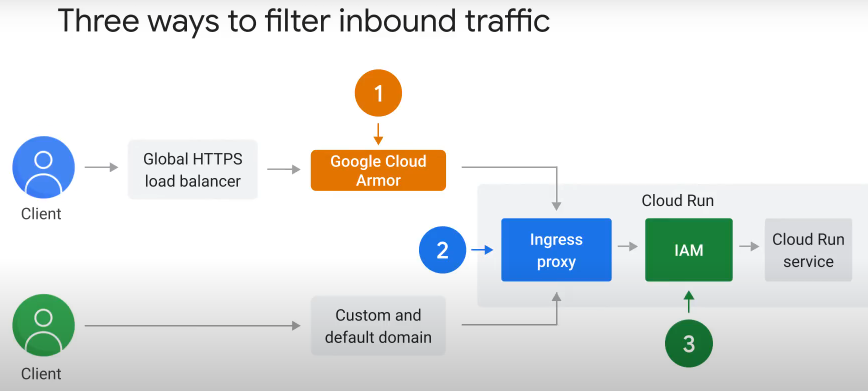
1. global HTTPS load balancer
2. default run.app domain or custom domain

Three sites to filter incoming traffic to a Cloud Run service:

1. Google Cloud Armor, integrated with global HTTPS load balancer
2. Ingress proxy of Cloud Run service
3. IAM

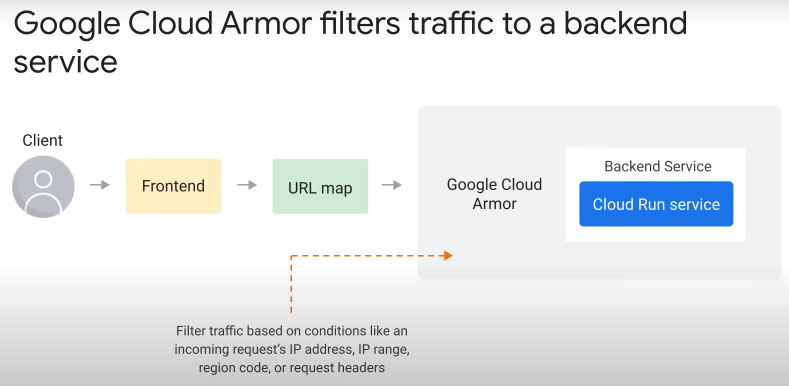
|  |  |  |
| --- | --- | --- |
| Cloud Armor | Ingress proxy | IAM |
| set IP allow/deny rules  filter based on IP geolocation  complex custom rules | filter based on source type  choice between 3 different source types  coarse, not granular, filter | based on identity of sender |

Only charged if request reaches Cloud Run service. Do not pay if request rejected by any of the filters.

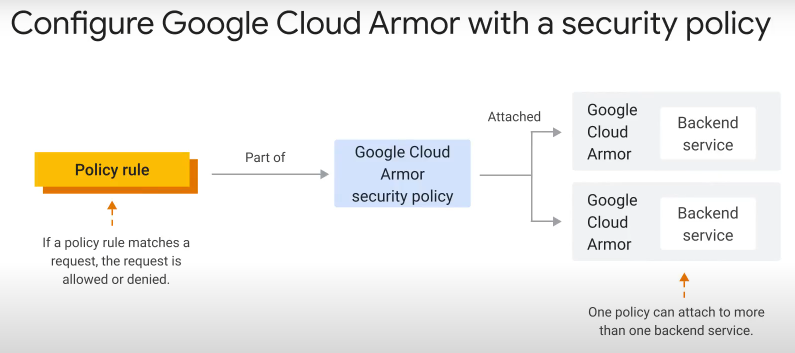


Cloud Armor is enabled at level of backend service:

* Global HTTPS load balancer is made up of 3 parts: Frontend + URL map + **Backend services**
* Cloud Run service is a backend
* Cloud Armor is enabled at level of backend service
* traffic to global HTTPS load balancer comes from internet
* filters incoming traffic with content-based rules. If match conditions based on IP address/range, geo region code, request headers => action taken.



Google Cloud Armor security policy:



Cloud Armor is configured using a security policy

* made up of many policy rules
* each rule defines action (allow/deny/rate-limit/redirect) to apply if simple/custom conditions (e.g. URL path, request method, request header) are matched.
* there are preconfigured Web Application Firewall (WAF) rules to mitigate OWASP top 10 risks (https://owasp.org/www-project-top-ten/)
  + e.g.: (1) cross-site scripting, (2)SQL injection attacks, (3) local file inclusion attacks, (4) remote file inclusion attacks, (5) remote code execution attacks.
  + signatures compiled from open-source industry standards
  + each signature is an attack detection rule
* can configure security policy using “named IP address list” – list of IP addresses and ranges
* Same security policy can be attached to >1 backend service.
* Security policy enforced at edge of Google Cloud, at points of presence (PoPs) – prevents unwanted traffic from consuning resources or entering VPC networks.

#### Ingress Proxy of Cloud Run

https://cloud.google.com/run/docs/securing/ingress

- focus: internal traffic, not from internet

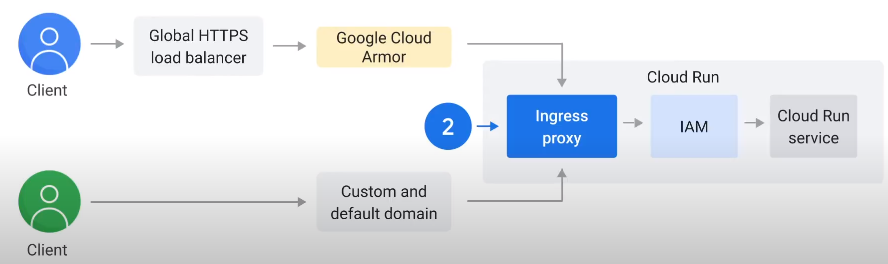
internet traffic going through global HTTPS load balancer

=> will be filtered by Google Cloud Armor

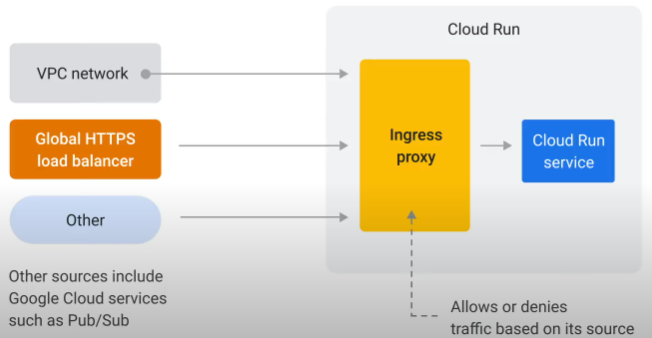
by default, at network level, internet traffic can also reach Cloud Run service on run.app URL & custom URL

=> configure Ingress proxy to filter traffic

=> not filtered by Cloud Armor



Request origins and ingress settings:



Ingress proxy filters requests by origin. Three source types:

1. from resources within VPC network

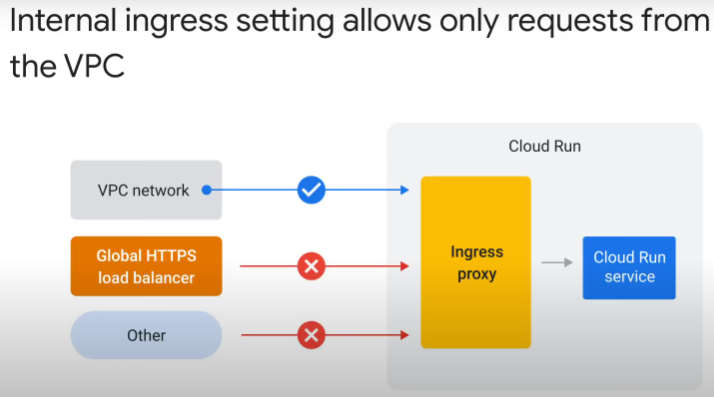
* include those within VPC service control perimeter

1. via global HTTP(S) load balancer
2. all others, including Google Cloud services e.g. PubSub

Three ingress settings:

1. internal
2. internal and load balancing
3. all (default, most permissive)

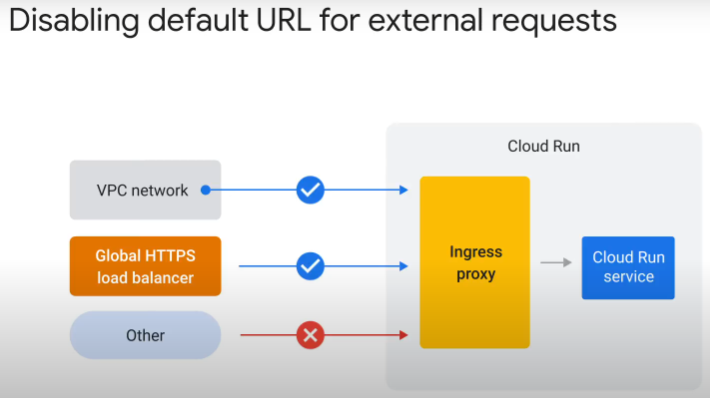
1. Internal only (most restrictive setting):



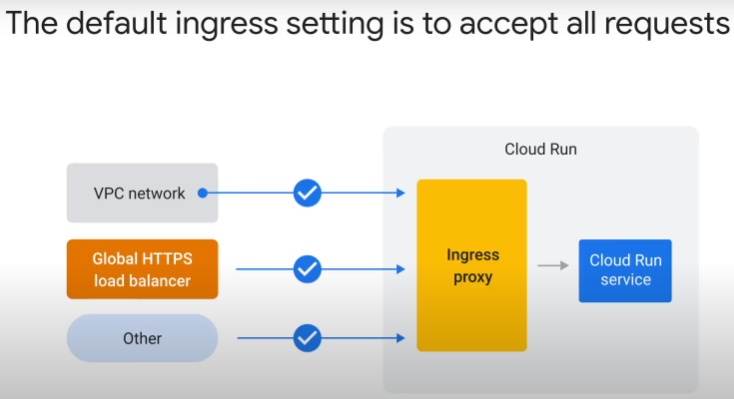
* only allow requests from
  + resources in same VPC network, same project as Cloud Run service
    - e.g. Compute Engine VMs, containers on GKE
  + resources within VPC Service Controls perimeter
  + internal HTTP(S) load balancer
    - including requests from shared VPC networks routed through this internal HTTP(S) load balancer
  + following Google Cloud services **if** they are in same project or VPC Service Controls perimeter:
    - PubSub
    - Cloud Tasks
    - Workflows
    - Eventarc
    - Cloud Scheduler
      * they access Cloud Run service at run.app URL or external domain name, but requests stay within the Google network
      * if these services are **NOT** in same project or allowed by VPC Service Controls perimeter => requests denied
* requests from other Cloud Run services denied, unless you use **serverless VPC access connector**.
  + by default, Cloud Run containers outside VPC network

2. Internal and load-balancing:

* allow 2 source types
  + those allowed by internal setting
  + requests through global HTTP(S) load balancer



* main purpose is to
  1. accept requests from internet through external load balancer / make Cloud Run service available to requests coming through external load balancer
     + subject requests to features of the load balancer:
       - Google Cloud Armor
       - Identity-Aware Proxy
       - Cloud CDN
  2. deny requests from internet directly to run.app default URL
* no option to allow traffic from external HTTP(S) load balancer only.
  + MUST include traffic from internal sources.

3. allow all requests

* including internet traffic to run.app and custom domain
* the default

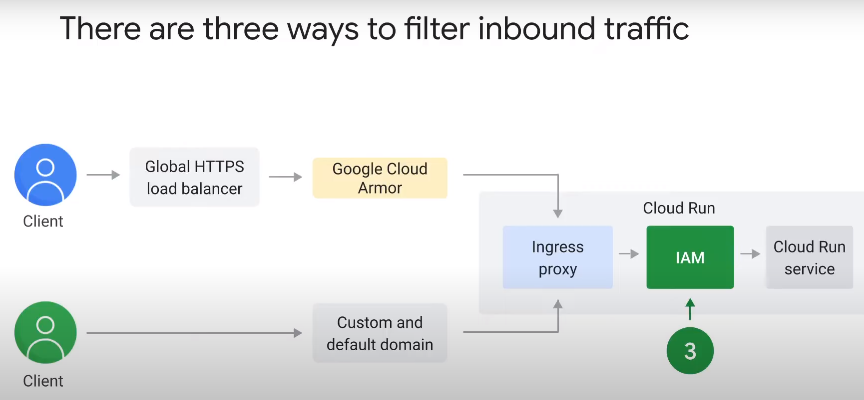
#### IAM

Filters traffic based on identity of sender

3rd component that filters inbound requests to Cloud Run service

* after Cloud Armor and Ingress settings

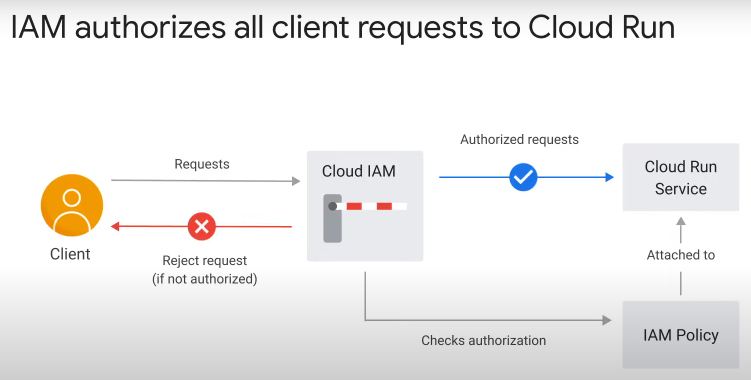
Every incoming request to Cloud Run service must pass through IAM



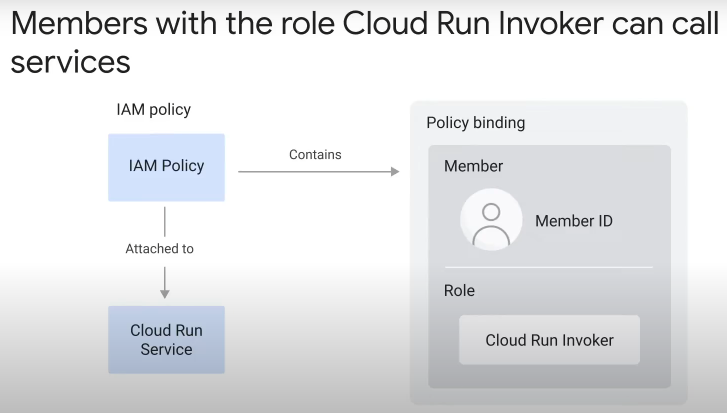
Configure IAM with IAM policy that is attached to the resource (Cloud Run service)

Two use cases:

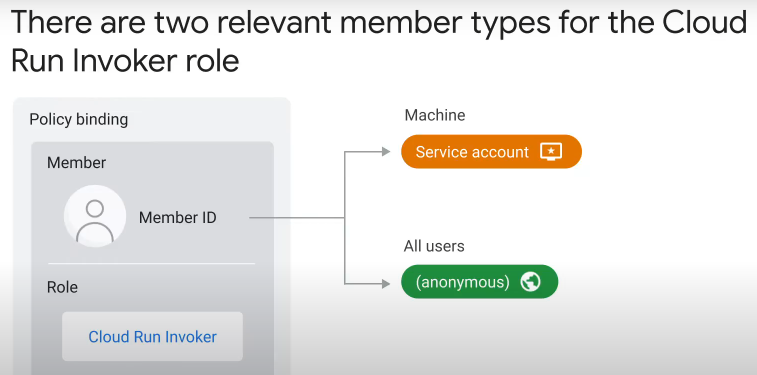
1. allow unauthenticated access – Cloud Run service publicly accessible without login required
2. restrict access to specific identities (usually service accounts) only



IAM policy:



* contains a list of policy bindings
* a policy binding binds an identity to a role
  + a role is a group of permissions for action on the resource
  + e.g. role (Cloud Run Invoker) – resource (Cloud Run service)



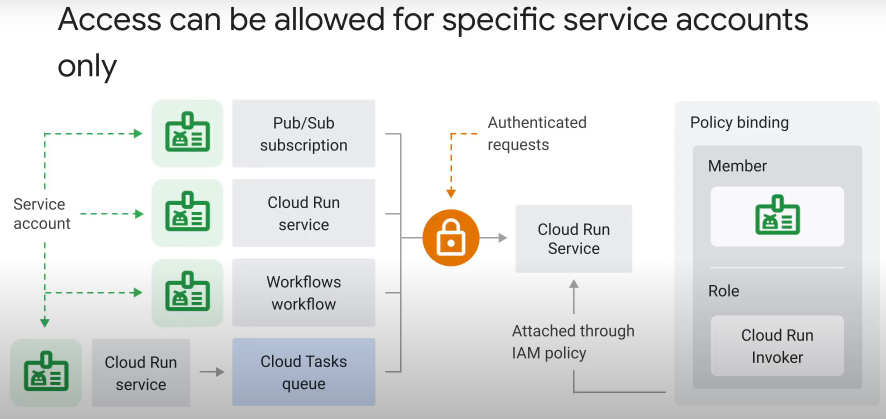
2 types of identity relevant to policy binding for Cloud Run Invoker role:

* service account / robot identity – all services in Google Cloud that need to call another API has one
* “all users” – effectively disabling IAM, allowing all unauthenticated requests through

3rd identity type: human identity.

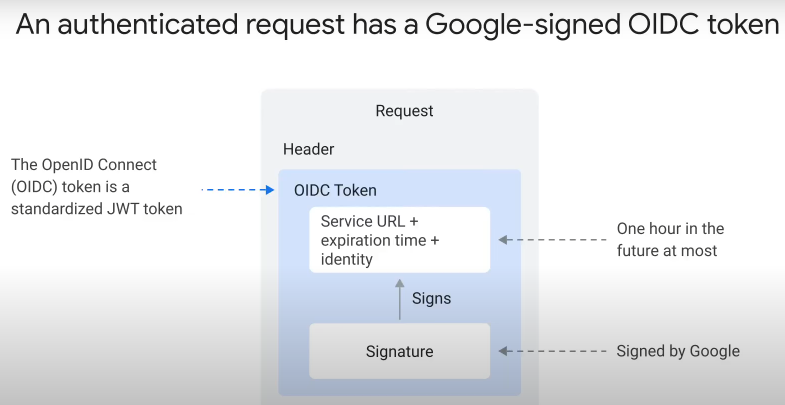
* Not common in policy binding for Cloud Run Invoker role
* IAM not meant to enable user logins
* if human user wants to call IAM-protected Cloud Run service, generate basic/bearer/digest/ Oauth2/JWT token first

Use cases of IAM with Cloud Run service:



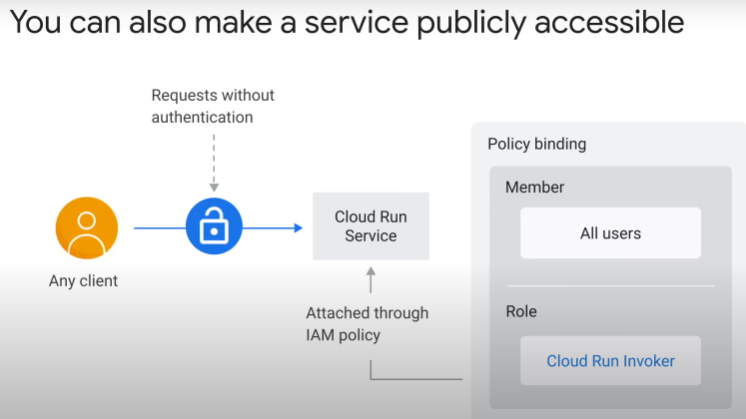
* use IAM to allow specific service accounts to access Cloud Run service on an “as needed” basis
* some Google Cloud services like PubSub and Workflows have built-in support to call IAM-protected services.
  + Configure their service identity to authenticate requests

What is an authenticated request?



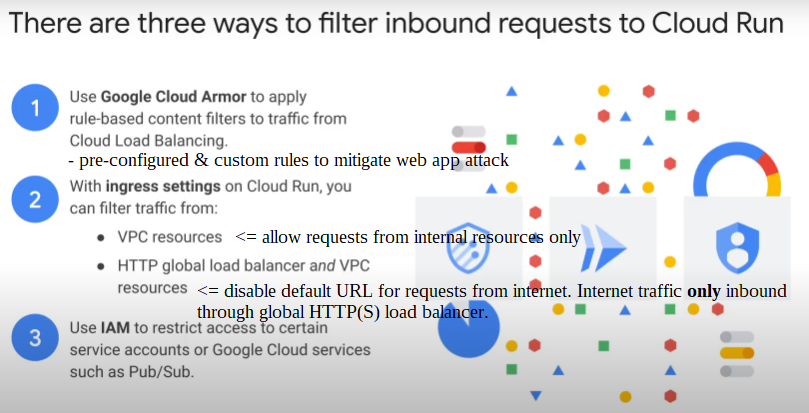
* Request has header with an identity token
  + identity token is an OpenID Connect (OIDC) token
  + OIDC token is a standardized Jason Web Token (JWT)
  + token contains audience, destination service URL, an expiration time < 1hr in the future, caller ID (service account ID)
  + cryptographic signature signed by Google proves that token has not been tampered with
* in your application, use one of the client libraries to send authenticated requests with an OIDC token
* Google managed services, e.g. PubSub, automatically add OIDC tokens

Public unauthenticated access to Cloud Run service:



* bind “all users” to “Cloud Run Invoker” role in policy binding
* attach IAM policy to resource (Cloud Run service)
* will allow public unauthenticated access to Cloud Run service
* If using Cloud console > Cloud Run, can set “allow public invocation”

Summary:



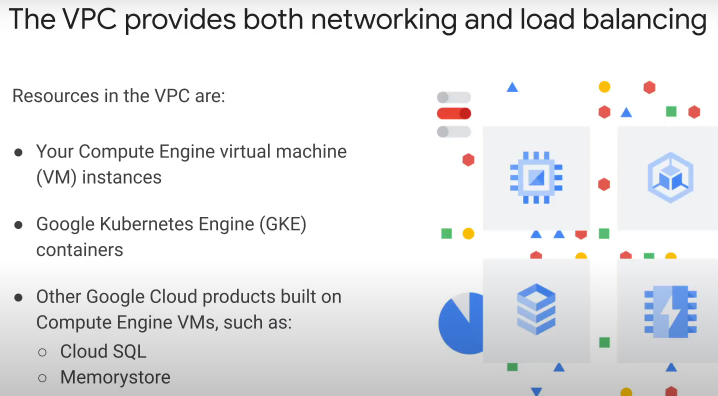
#### Severless VPC Access Connector

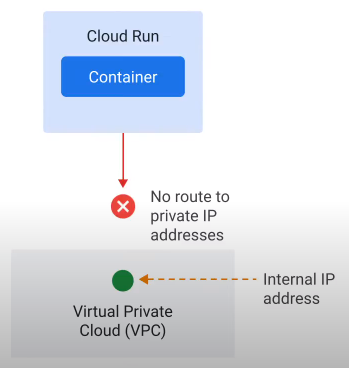
How to filter outbound traffic

* prevent outbound connections to unwanted or dangerous hosts
* restrict egress to VPC only
* use VPC Access Connector

VPC :

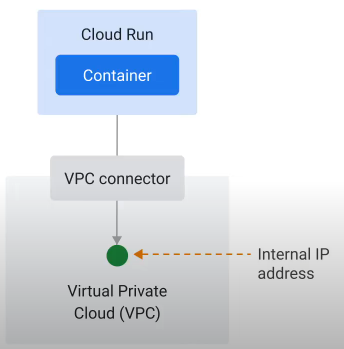
* provides networking and load balancing
* anything that have internal IP address is part of VPC
* example of resources in diagram below



Cloud Run containers are not part of any VPC:

* by default, cannot connect with any resource that only have internal IP address

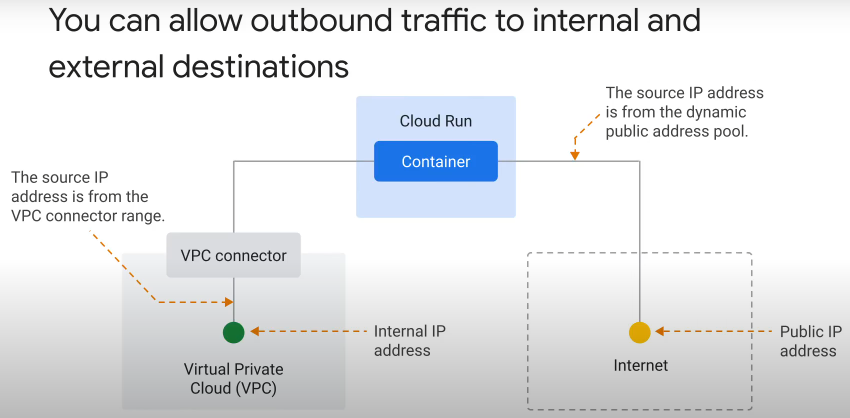
Create **VPC Access Connector** to give Cloud Run service access to private internal IP addresses in a VPC:

* VPC Access Connector is a resource in the VPC
* acts as router to forward traffic from a Cloud Run service
* one service – one connector
* not serverless. Always running, like VMs.
  + Hence, you are charged as long as connector enabled. Even when not connected to a Cloud Run service, or not forwarding traffic.

Steps:

* first, create the VPC Access Connector
* second, configure Cloud Run service to use it

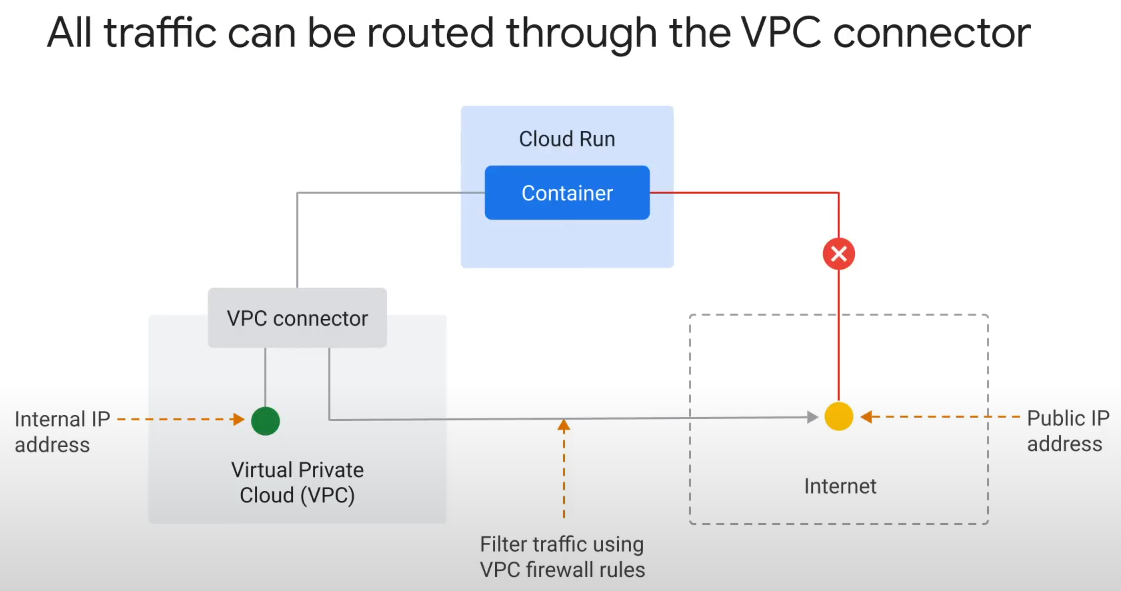
Outbound traffic from Cloud Run service can go to internal (within VPC) or external (internet) destinations when VPC Access Connector is attached:



Source IP for traffic to

* internal IP addresses – from VPC connector range
  + can use VPC features, e.g. firewall, to monitor and control traffic
* external public IP addresses on the internet – from dynamic public address pool
  + share with other Cloud Run services and other customers
  + no control

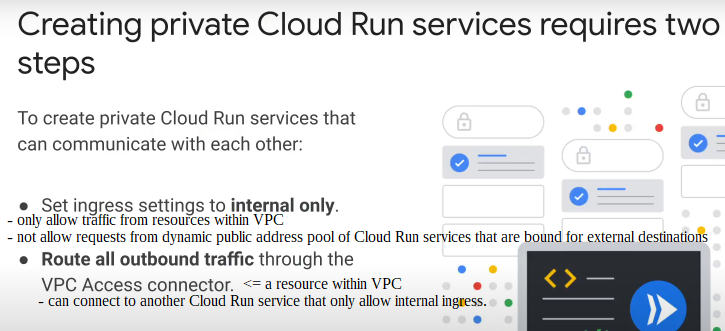
Can route all egress traffic from Cloud Run through VPC only:



To control outbound traffic

* set VPC egress setting
* route all Cloud Run traffic through VPC Access Connector to VPC network
* traffic with external destination will also connect through VPC
  + and be subject to network’s firewall, DNS and routing rules
* firewall can prevent outbound connection to external dangerous or unwanted hosts
* Cloud Network Address Translation (NAT) can use static IP address instead of dynamic IP from public pool

How to connect internal (non-public) Cloud Run services:



* must set ingress and egress settings

