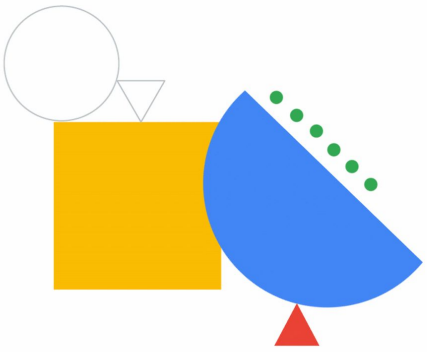
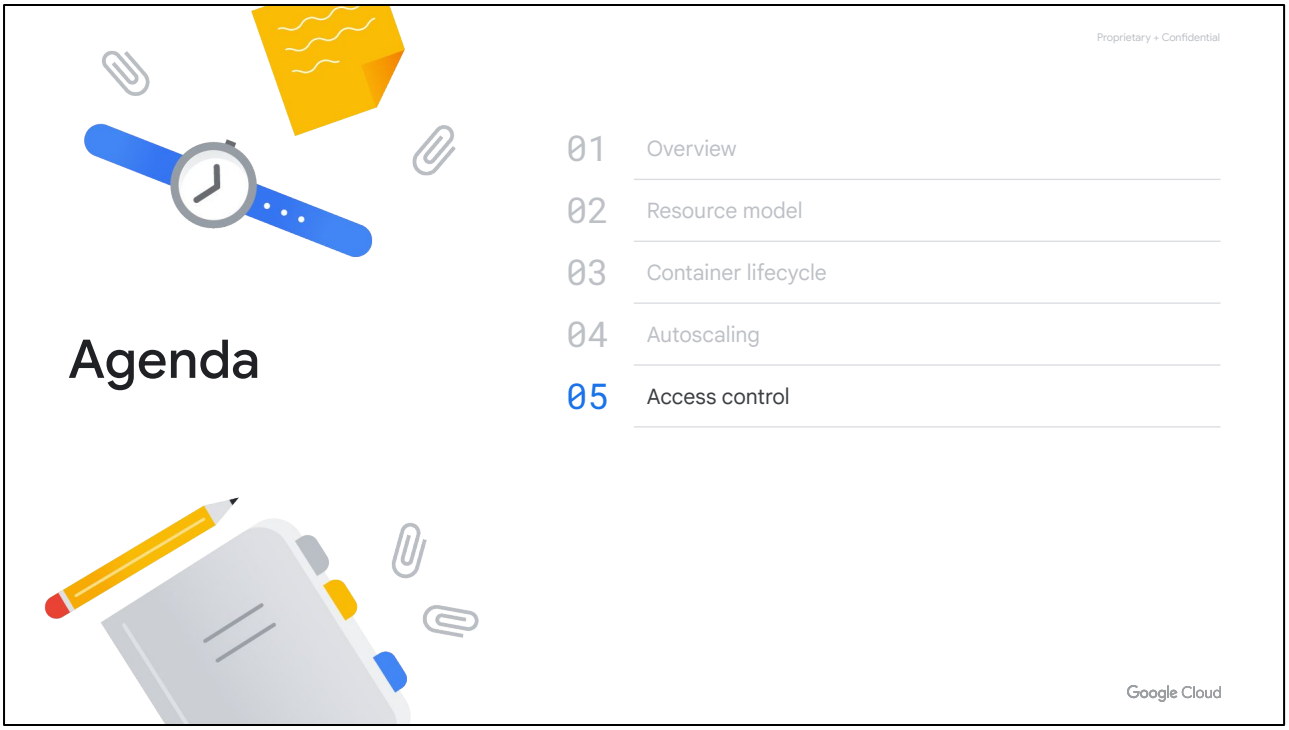
Google Cloud

Fundamentals of Cloud Run

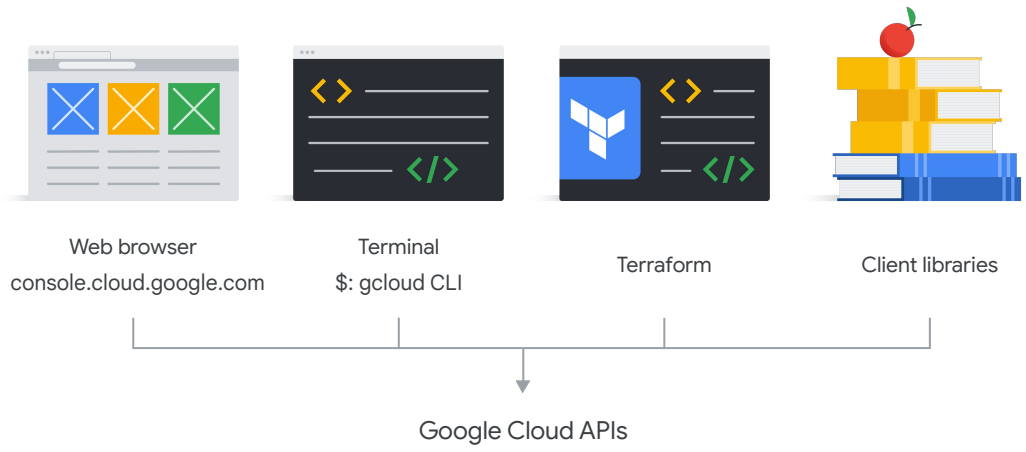


In this module, we discuss the fundamentals of developing and running applications on the Cloud Run platform.



In this topic, we discuss how you can control access to your Cloud Run services and jobs.

Google Cloud uses APIs



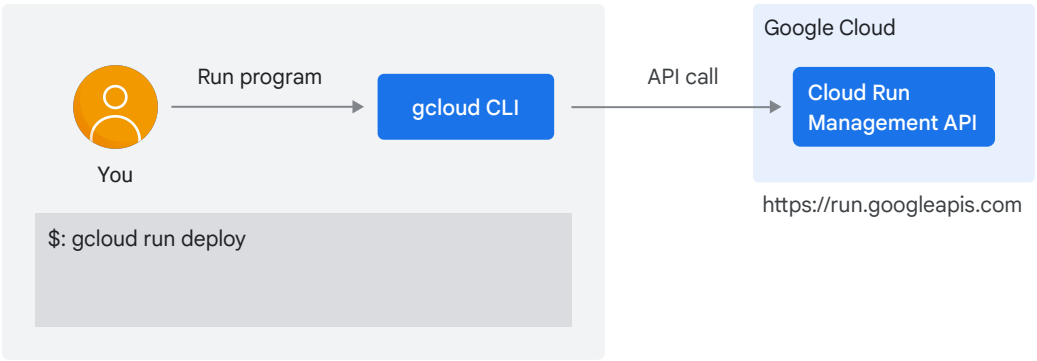
One way to look at Google Cloud is that it's a collection of APIs that lets you create and manage virtual resources, like virtual machines, Cloud Run services, load balancers, or a Pub/Sub topic and a database table on a Cloud SQL database server.

To manage your infrastructure, you interact with these APIs with:

- The web console from a browser.
- The gcloud CLI.
- Terraform (a third party application that lets you practice infrastructure as code).
- Client libraries from your application code.

The key thing to realize is that Google Cloud is *a collection of APIs* that let you create virtual resources.

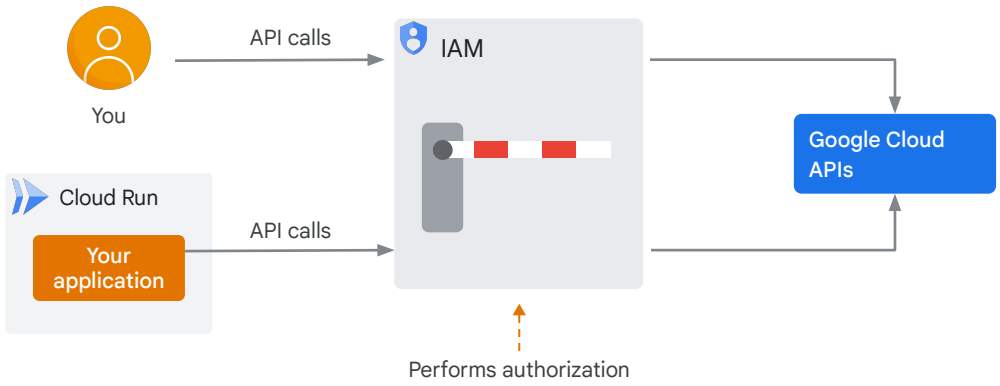
Deploying a container image is an API call



As an example, deploying a container image to Cloud Run is an API call.

When you run the *gcloud run deploy* command on your local machine using the gcloud CLI, an API call is made to the Cloud Run service on run.googleapis.com to deploy the container image.

IAM authorizes API calls



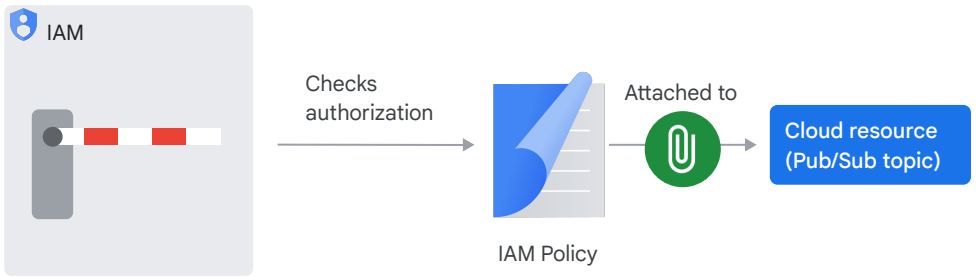
Identity and Access Management (IAM) is a Google Cloud service that lets you create and manage permissions for Google Cloud resources.

IAM verifies the identity of the caller, and checks if it has permissions to perform the API call. If that check fails, it rejects the call.

It's important to note that IAM works the same way, regardless if you're deploying a new revision of a Cloud Run application (API call), or if your application code publishes a message to Pub/Sub using Google Cloud APIs.

In both examples, IAM performs the authorization of the API calls.

IAM uses policies

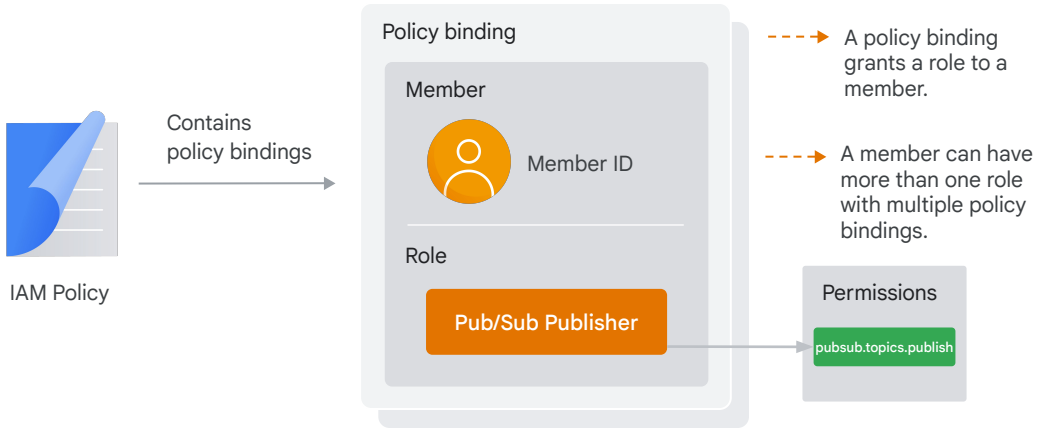


To check if you're authorized to perform certain actions on Google Cloud resources, IAM uses policies.

For example, to publish a message to a Pub/Sub topic, IAM checks an IAM policy that is attached to the Pub/Sub topic.

If that policy has a binding that allows you to publish the message, IAM lets the call through.

IAM policy



An IAM Policy is a list of *policy bindings*. A policy binding binds a member (identity) to a single role.

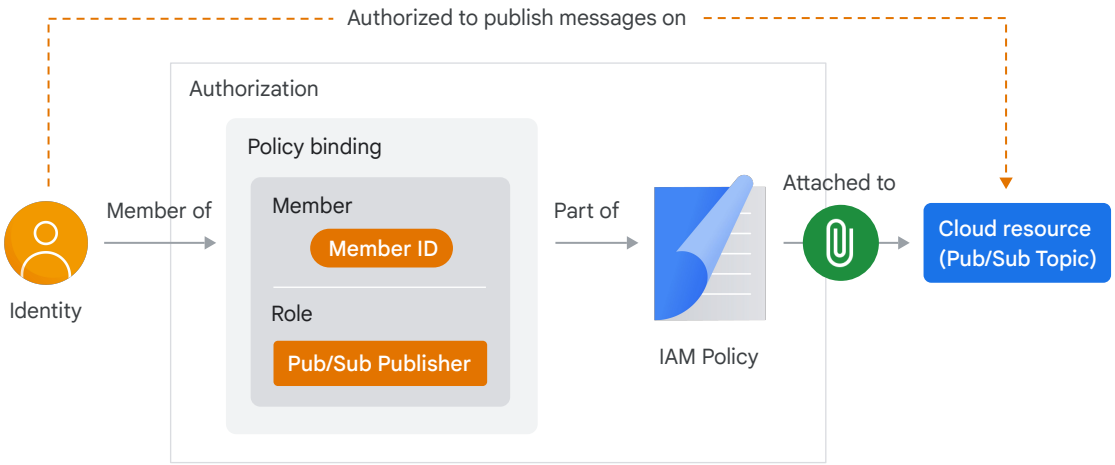
In the example, if you have the role "Pub/Sub Publisher", you're allowed to send messages to the Pub/Sub topic that the IAM policy is attached to.

A member can have multiple policy bindings in an IAM policy enabling that member to have more than one role.

A role contains a set of permissions that allows the member identity to perform specific actions on Google Cloud resources. In the example, the Pub/Sub Publisher role includes the pubsub.topics.publish permission that provides access to publish messages to a topic.

For more information on roles and permissions view the Cloud IAM documentation.

IAM authorization



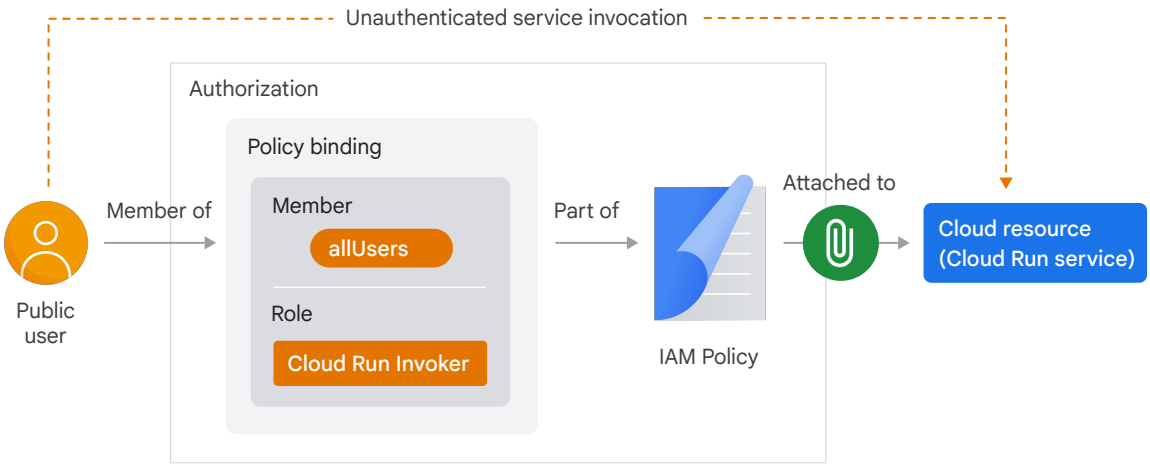
Now, let's bring everything together.

An IAM policy is always attached to a resource. In this example, you're authorized to publish messages to the topic because:

1. You're a member of a policy binding.
2. The binding has the role Pub/Sub Publisher.
3. The binding is part of an IAM Policy, which is attached to the topic.

Another way to say this is that the policy binding in the IAM policy *grants* you the permission to publish messages to a specific topic.

Making a Cloud Run service public




```
gcloud run services add-iam-policy-binding my-service --member="allUsers"
--role="roles/run.invoker"
```


As previously mentioned, IAM is about managing access to Google APIs to create and managing cloud resources. You can also use IAM for access control to invoke Cloud Run services, which also works well for service-to-service communication.

By default, only users or identities with the owner or editor roles on a project can create, update, delete, or invoke Cloud Run services and jobs.
Also, project owners and identities with the Cloud Run Admin role (*roles/run.admin*), can modify IAM policies on the project or individual Cloud Run service or job.

To make a Cloud Run service publicly accessible, you can allow unauthenticated invocations to the service. You do this by assigning the IAM *Cloud Run Invoker* role to the *allUsers* member type on the service. To configure authentication by assigning this role, you must have the required permission which is included in the project *Owner* and *Cloud Run Admin* roles.

You can also use the *–allow-unauthenticated* option with the *gcloud run deploy* command when you deploy your service.
You can make a service publicly accessible using the Google Cloud console, the gcloud CLI, a YAML configuration file, or Terraform.

Controlling access to services and jobs

To control access to individual Cloud Run services or jobs:

- Add a principal to a service or job with the desired role.
- Remove a principal from a role for a service or job.

```
gcloud run services
add-iam-policy-binding my-service
--member=MEMBER_TYPE --role=role
```

```
gcloud run jobs
remove-iam-policy-binding my-job
--member=MEMBER_TYPE --role=role
```

To control access to all Cloud Run services or jobs in a project, use project-level IAM.

```
gcloud projects
add-iam-policy-binding
my-project-id --member=MEMBER_TYPE
--role=role
```

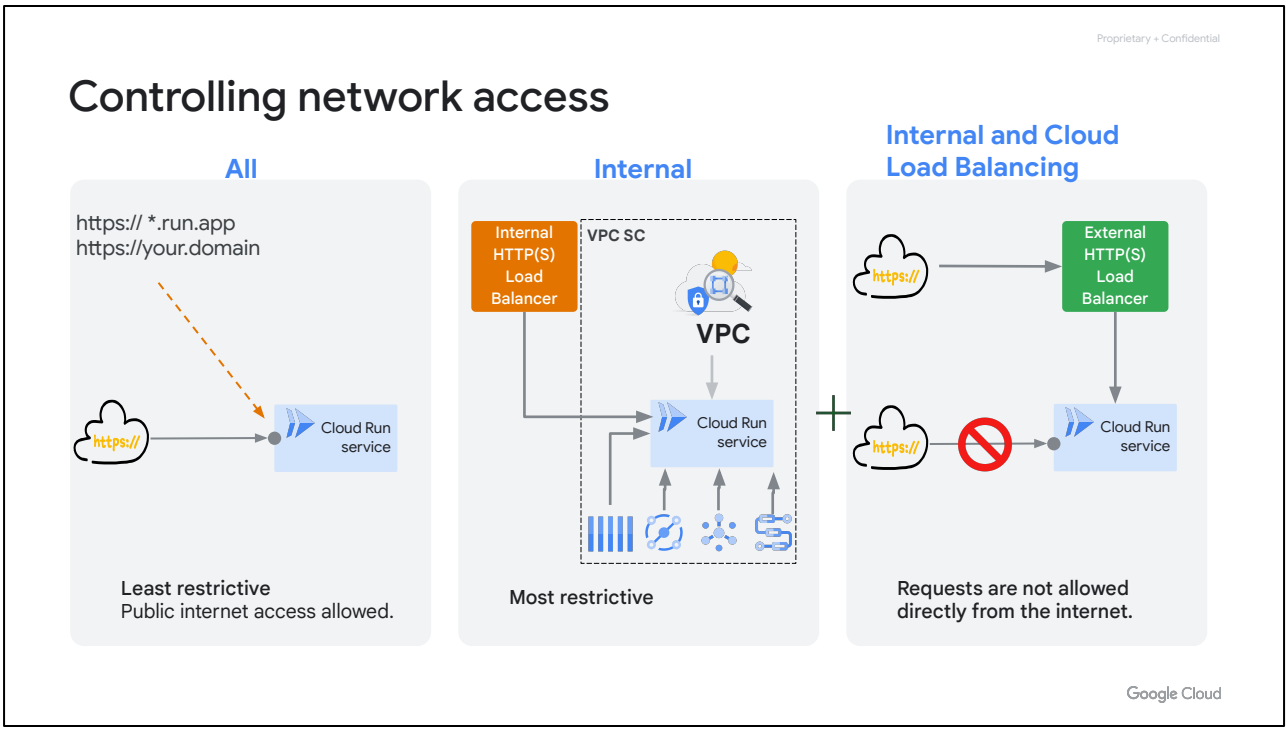

You can add individual users or principals (identities) to a Cloud Run service or job with the desired roles and permissions in the Google Cloud console, or with the *gcloud run [services | jobs] add-iam-policy-binding* command.

For example, to invoke a Cloud Run service with a service account, you can grant the invoke permission to the member account with this gcloud CLI command:
`gcloud run services add-iam-policy-binding my-service --member=serviceAccount:sa_email --role=roles/run.invoker`

To remove principals from a role for a service or job, use the console or the *gcloud run [services | jobs] remove-iam-policy-binding* command.

To grant access to principals on all services and jobs in a project, you can use project-level IAM with the *gcloud projects add-iam-policy-binding* command.

For more details, refer to the Cloud Run IAM roles documentation.



In addition to the IAM authentication methods discussed previously, network ingress settings is another way to manage access to a service.
These methods are independent of each other, but for a layered approach to managing access, you can use both.

To control network access to your service, Cloud Run provides three ingress settings at the service level:

- **All** - This is the least restrictive default setting, that allows all requests including those sent directly from the internet to the default run.app URL or custom domain of your service.

- **Internal** - This is the most restrictive setting, that only allows requests from:
  - Internal HTTP(S) load balancer
  - Resources that are allowed by any VPC Service Controls perimeter that contains your Cloud Run service.
    VPC Service Controls is a Google Cloud feature that lets you set up a secure perimeter to guard against data exfiltration. Both the default run.app URL and custom domains are subject to VPC Service Controls.
  - VPC networks in the same project or VPC Service Controls perimeter as your Cloud Run service.
  - Google Cloud services: Cloud Tasks, Eventarc, Pub/Sub, and Workflows if they are in the same project or VPC SC perimeter as your

    - Cloud Run service. Requests from these sources stay within the Google network, even if they access your service at the run.app URL.

Requests from other sources, including the internet, cannot reach your service at the run.app URL or custom domains.

- **Internal and Cloud Load Balancing** - allows requests from:
    - Resources that are allowed by the more restrictive *Internal* setting.
    - External HTTP(S) load balancer, but not directly from the internet.

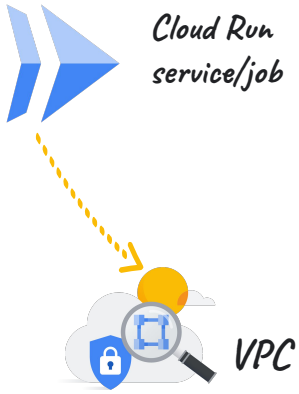
For more details on using VPC service controls with Cloud Run, view the documentation.

Connecting to a VPC network

A VPC network is a virtualized physical network that is implemented in Google's production network.

Use Serverless VPC Access to:

- Connect a Cloud Run service or job to a VPC network.
- Send requests and receive responses to and from the VPC network using internal DNS and internal IP addresses.
- Prevent requests and responses to and from internal resources from going over the internet.



A Virtual Private Cloud (VPC) network is a virtual version of a physical network, implemented inside Google's production network.
It's a global resource that consists of a list of regional virtual subnetworks (subnets) in data centers, all connected by a global wide area network.

To connect a Cloud Run service or job directly to your VPC network to access VM instances, Memorystore instances, and other resources with an internal IP address, use Serverless VPC Access.

With Serverless VPC Access, you can send requests and receive responses to and from your VPC network using internal DNS and internal IP addresses, so that traffic is not exposed to the internet.

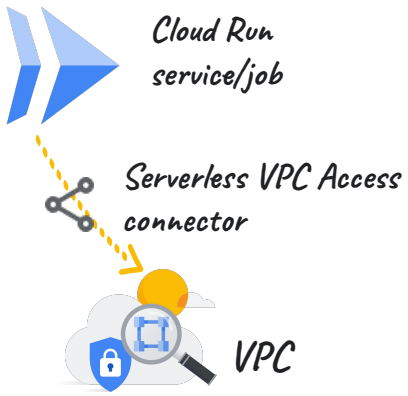
Configuring Serverless VPC Access

To configure Serverless VPC Access:

1. Enable the Serverless VPC Access API.
2. Create a Serverless VPC Access connector in your Google Cloud project.
3. Attach the connector to a VPC network and region.
4. Configure your Cloud Run service or job to use the connector.

gcloud CLI

```
gcloud run deploy my-service --image \
my-image --vpc-connector my-connector
```



A Serverless VPC Access connector is a resource that handles traffic between your Cloud Run service or job and your VPC network.

The region that is configured for the connector must match the region where your service or job is deployed.

Configure the connector with an unused /28 subnet or non-overlapping /28 CIDR range. The subnet or CIDR range must be used exclusively by the connector and no other resources.

You can create a connector in the Google Cloud console, with the Google Cloud CLI, or with Terraform.

After you have created a Serverless VPC Access connector, you must configure your Cloud Run service or job to use the connector. You can do this in the Google Cloud console, with the Google Cloud CLI, YAML file, or Terraform, when you create the service or deploy a new revision. Similarly, you must configure your Cloud Run job to use the connector when the job is created. For an internal Cloud Run service, you should set all egress from the service to use the VPC connector.

You can also restrict your connector's access to the resources in your VPC network with firewall rules.

For more details, view the documentation on connecting to a VPC network.

Remember

1. IAM lets you create and manage roles and permissions to Google Cloud resources.
2. To authorize API calls, IAM uses policies that are attached to resources.
3. To control access to individual Cloud Run services, add principals with desired roles.
4. Configure ingress network settings to make a Cloud Run service public or internal.
5. Use Serverless VPC Access to connect Cloud Run services to internal resources.

In summary:

Identity and Access Management (IAM) is a Google Cloud service that lets you create and manage permissions for Google Cloud resources.
To authorize calls made to Google Cloud APIs, IAM uses policies that are attached to resources.

To control access, you can add individual users or principals (identities) to a Cloud Run service or job with the desired roles and permissions.

To control network access to your service, configure Cloud Run ingress settings at the service level.

To connect a Cloud Run service or job directly to your VPC network to access VM instances, Memorystore instances, and other resources with an internal IP address, use Serverless VPC Access.