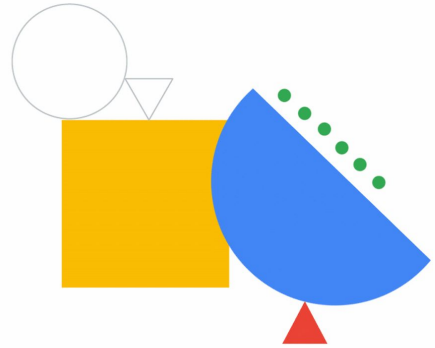


# Service Identity and Authentication



In this module, we discuss the fundamentals of service identity, and how you can authenticate your Cloud Run service with other Google Cloud services.



# Agenda




- 01 Service account and identity
- 02 Resource hierarchy
- 03 Principle of least privilege
- 04 Secrets and environment variables
- 05 Lab: Implementing Least Privilege IAM Policy Bindings in Cloud Run
- 06 Quiz

Here is the list of topics included in this module.

You'll learn about service accounts, and how they provide an identity for your Cloud Run service. We'll discuss the resource hierarchy in Google Cloud, and the principle of least privilege.

You'll also learn how you can use secrets and environment variable with your Cloud Run service and complete a lab.

We'll end the module with a short quiz on the topics that were discussed.



## 01 Service account and identity

---

# Agenda



Google Cloud is *a collection of APIs* that let you create virtual resources.

You can invoke these APIs from your Cloud Run application code with client libraries.

In this topic, we examine how Cloud Run services use service accounts to invoke Google Cloud APIs.

## Example Google Cloud APIs

Google Cloud APIs  
and an example operation

**Cloud Build**  
Submit a build.

**Cloud Run**  
Create a service.

**Compute Engine**  
Start a VM.

**Artifact Registry**  
Push a container  
image.

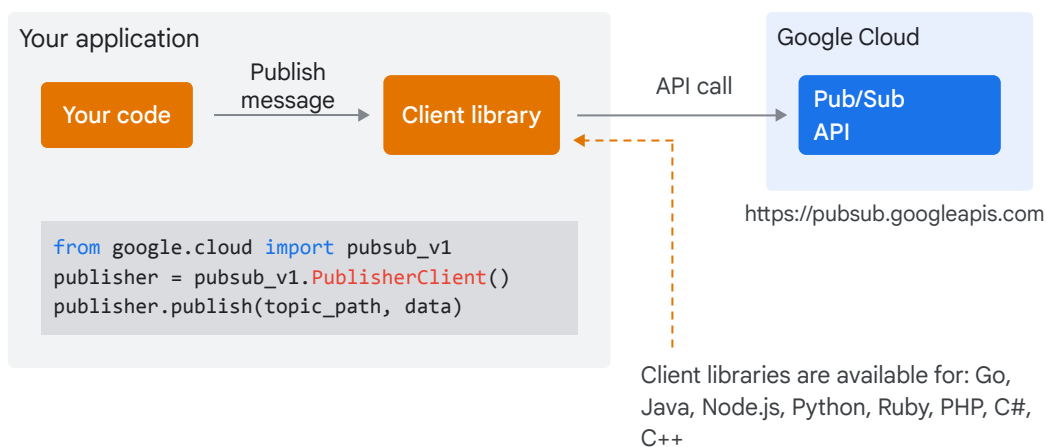
**Cloud SQL**  
Create an instance.

**Cloud Storage**  
Create a bucket.

Here are some examples of Google Cloud products with an example API call.

To interact with these service APIs, you can use the gcloud CLI, the web console, a client library, or use any other process that calls the API, like Terraform.

## An example API call from an application



Google Cloud

Here's another example API call from an application using a client library.

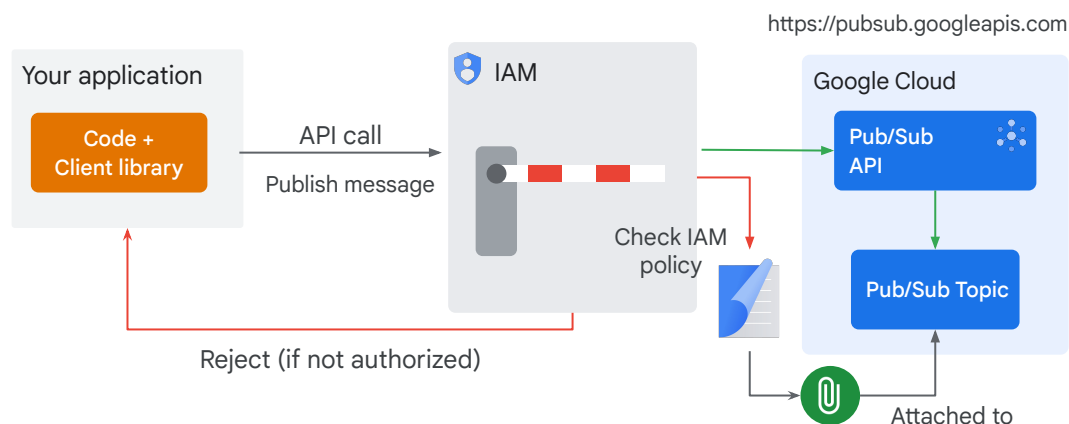
Pub/Sub is a message broker service on Google Cloud that you can use to send messages between services asynchronously.

To publish a message to Pub/Sub from a Python application, you would use the Pub/Sub client library as shown in the sample code. The client library handles the API call to [pubsub.googleapis.com](https://pubsub.googleapis.com).

For most Google Cloud services, there are client libraries available for Go, Java, Node.js, Python, Ruby, PHP, C# and C++ programming languages.

<https://cloud.google.com/apis/docs/cloud-client-libraries>

## Publishing a message requires authorization

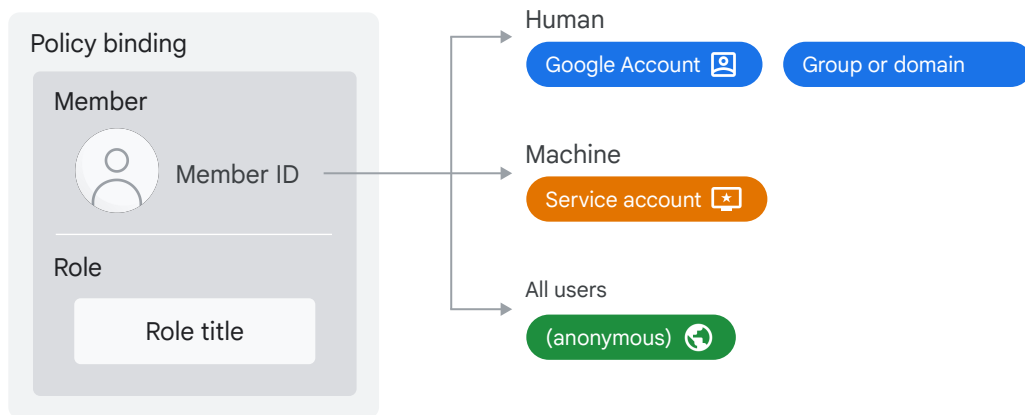


In this example, the application code uses the Pub/Sub client library to send an API call to `pubsub.googleapis.com`. IAM will inspect the request and identify your application by the credentials in the API request.

Now that IAM has the identity, it will need to find what operations are allowed for the identity to perform on the Pub/Sub topic.

IAM does this by checking policy bindings in an IAM policy that you attach to the Pub/Sub topic.

# Policy binding



A policy binding binds one or more members (identities) to a single role. A role contains a set of permissions that allows the member identity to perform specific actions on Google Cloud resources. For example, the Pub/Sub Publisher role includes the `pubsub.topics.publish` permission that provides access to publish messages to a topic.

IAM supports the following types of identities:

- **Human identities**  
Your Google account is a human identity which you use to sign in to Google Cloud. Your Google Account can also be part of a group or a domain.
- **Service account**  
Used by machines or applications. Examples of machines with a service identity are a virtual machine, a Cloud Run service, a Cloud Function, or other services.
- **All users**  
A special identifier to allow everyone or allow public access to a service on Google Cloud.

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

# Service account

- A service account is a special type of identity that is meant to be used by machines.
- A service account does not have a password and can't sign in using a browser.



Service account

Just like your Google Account is your identity from the point of view of IAM, a Cloud Run service also has its own identity, which is called a service account.

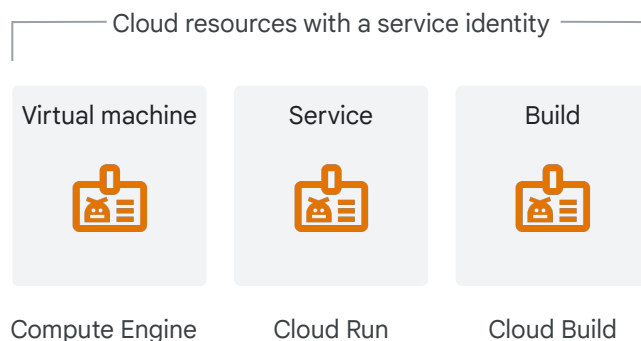
A service account is a special type of account used by machines, applications, or services. It's identified by its email address, which is unique to the account.

Service accounts differ from user accounts in a few ways:

- Service accounts do not have passwords, and cannot sign in by using browsers or cookies.
- You can let other users or service accounts act on behalf of a service account.
- Service accounts are not members of your Google Workspace domain, unlike user accounts, although you can add them to groups.



## Service account usage



Service accounts are meant to be used by machines.

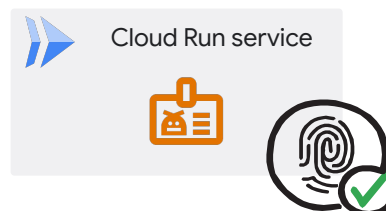
If you run code somewhere, for example on a virtual machine, in a Cloud Run service, or as part of a build in Cloud Build, you have access to a built-in service account.

If you use one of the client libraries to connect to Google Cloud APIs, the libraries will automatically use this built-in service account for authentication.

You can always replace the service account with your own user-managed service account, which is recommended.

## Service accounts in Cloud Run

- Every Cloud Run service or job is linked to a service account, known as the service identity.
- Use a per-service account for each service.
- Grant minimal, selective permissions to each service account.



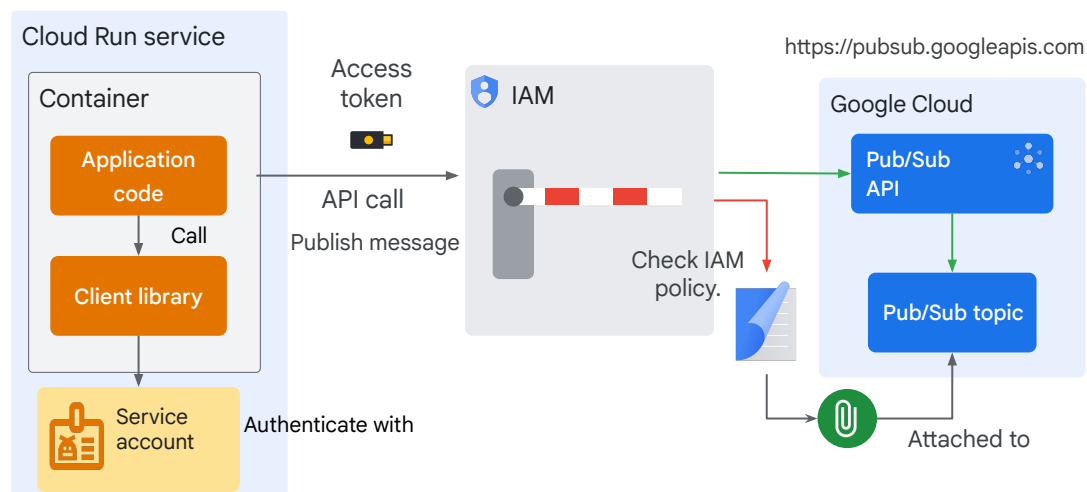
Every Cloud Run service or job is linked to a service account, that is also known as the “service identity.”

By default, Cloud Run services or jobs run as the [default Compute Engine service account](#) with the Editor role.

It’s recommended to use a user-managed service account with the most minimal set of permissions required for the service to perform its functions.

A best practice is to use a service account for each service identity, and grant selective permissions to the account.

# Service identity on Cloud Run



Google Cloud

In practice, a container with your application runs as part of a Cloud Run service.

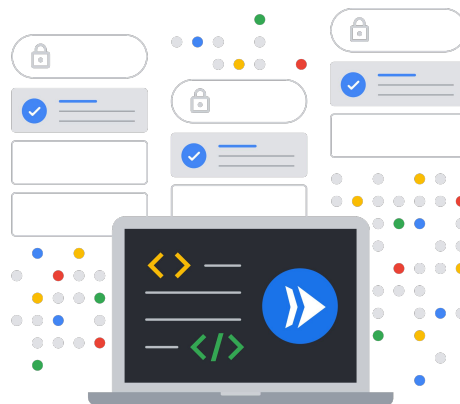
If you use a client library in your application code to publish a message to a Pub/Sub topic, the library automatically acquires appropriate tokens to authenticate your code's requests using the service's runtime service account. When accessing most Google APIs, OAuth 2.0 access tokens are used.

The access token is used to call the Pub/Sub API.

IAM verifies the access token, and uses the identity in the access token to check if there is a policy binding with the required roles to publish a message to the attached Pub/Sub topic.

## Remember

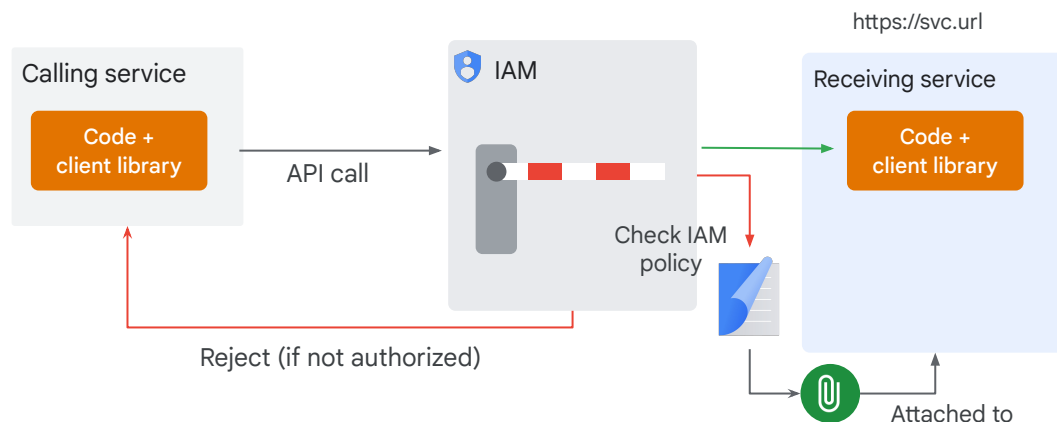
- 1 An IAM policy contains a list of policy bindings that bind members to a role.
- 2 A service account is a type of member identity that is used by machines, applications, or services.
- 3 Every Cloud Run service or job is linked to a service account.
- 4 Use a user-managed service account for each Cloud Run service with a minimum set of permissions.



### In summary:

- An IAM policy contains a list of policy bindings that bind members to a role. To authorize API and service calls, IAM reads the IAM policy that is attached to a resource.
- A service account is a type of member identity that is used by machines, applications, or services.
- Every Cloud Run service or job is linked to a service account.
- Use a user-managed service account for each Cloud Run service with a minimum set of permissions.

## Service to service communication



Google Cloud

If your application architecture uses multiple Cloud Run services, these services probably need to communicate with each other. This communication can be either asynchronous or synchronous. Many of these services might be private and therefore require credentials for access.

For asynchronous communication, you can use various Google Cloud services such as Cloud Tasks, Pub/Sub, Cloud Scheduler, or Eventarc.

For synchronous communication, your service calls another service's endpoint URL directly over HTTP. In this case, it's a best practice to use IAM and an individual service identity for the calling service. The service account is granted the minimum set of permissions required.

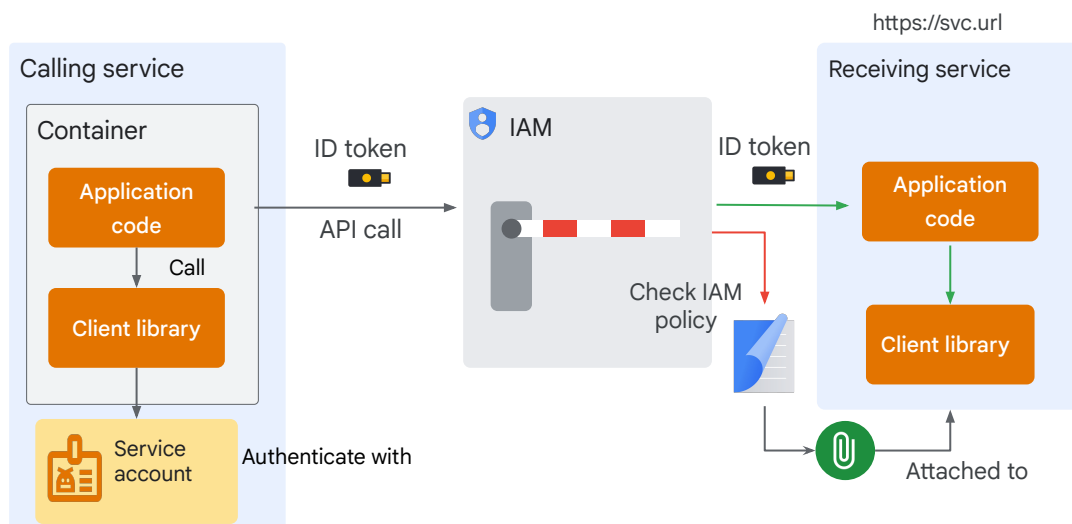
To set up a service account, you configure the receiving service to accept requests from the calling service by making the calling service's service account a *principal* on the receiving service. Then you grant that service account the *Cloud Run Invoker* (*roles/run.invoker*) role.

You can do this in the Google Cloud console, with the gcloud CLI, or with Terraform. Here's the gcloud CLI command:

```
gcloud run services add-iam-policy-binding RECEIVING_SERVICE
```

*--member='serviceAccount:CALLING\_SERVICE\_IDENTITY' --role='roles/run.invoker'*

## Service to service communication



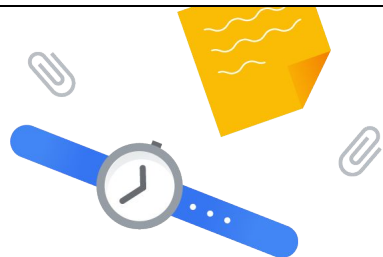
Google Cloud

The request made by the calling service must present proof of this identity in the form of a Google-signed [Open ID Connect](#) token.

OpenID Connect is an identity protocol based on OAuth 2.0 that enables identity verification of a client based on the authentication performed by an authorization server. It's also used to obtain basic profile information about the client.

One way to acquire this ID token is to use the Google authentication client libraries in your calling service's application code.

In the receiving service, your application code can use Google's authentication libraries to parse the request, and extract and verify information from the ID token.



01 Service account and identity

---

02 Resource hierarchy

---

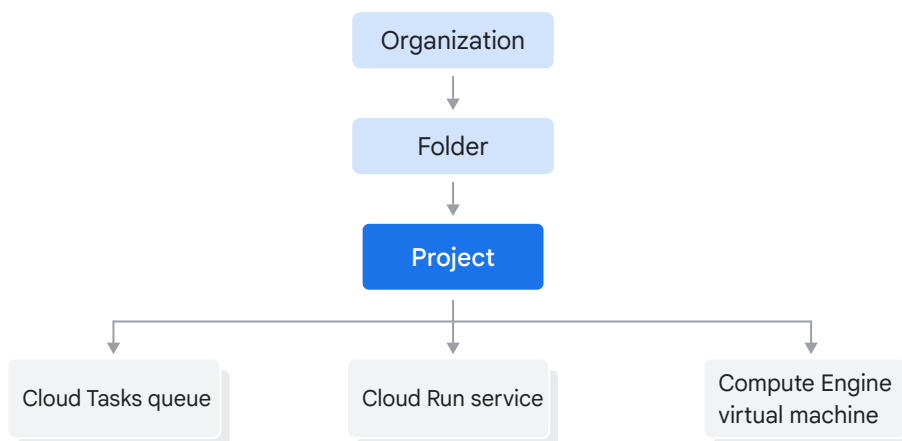
## Agenda



Next, let's look at how you can use Google Cloud resource hierarchy to simplify permissions.



# Resource hierarchy



Google Cloud

Google Cloud resources are organized hierarchically. A Google Cloud project is your primary means to organize resources, where every resource needs to be in a project.

A cloud resource can be, for example, a Cloud Tasks queue, a Cloud Run service, or a Compute Engine virtual machine.

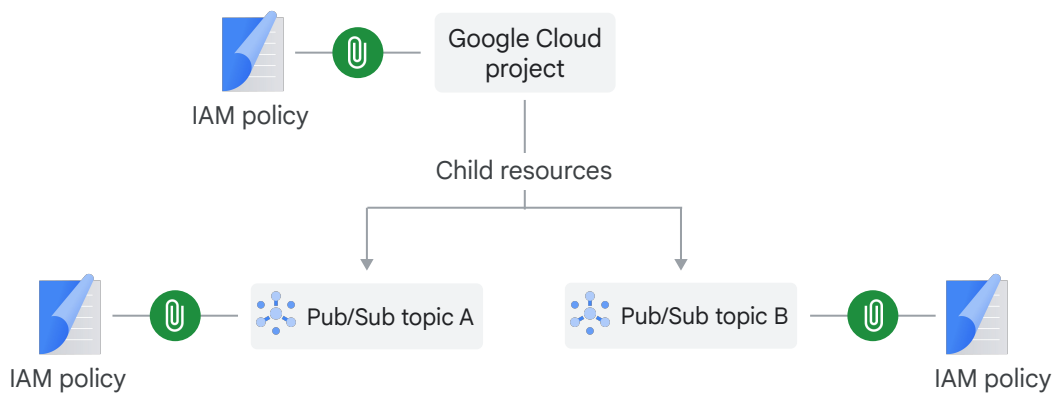
The organization resource is the root node of the resource hierarchy. It provides central visibility and control over every other resource that belongs to the organization.

An additional and optional grouping mechanism under the organization node is the folder. Folders can be mapped to departments, teams, or business units within an organization.

The project is the base-level entity for organizing resources. It's required to create resources, use Cloud APIs and services, manage permissions, and enable billing among other things. You can organize projects into folders.

Each resource has exactly one parent (except for the top organization node which has none).

## Resource permissions

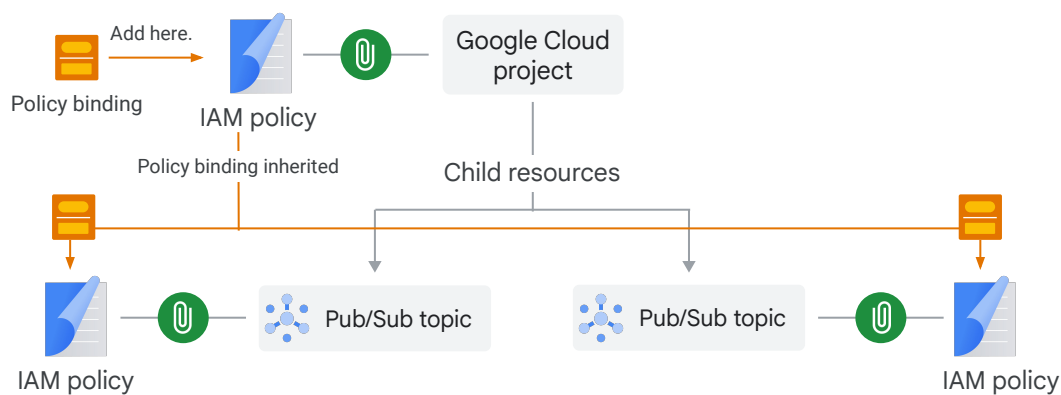


Every resource in the hierarchy has an IAM Policy, and you can grant permissions on it using policy bindings.

A policy binding binds an identity to a role and grants permissions to that identity on that resource.

If you think back to the example of publishing a message to a Pub/Sub topic: what was the role you needed? "Pub/Sub Publisher".

# Policy binding inheritance



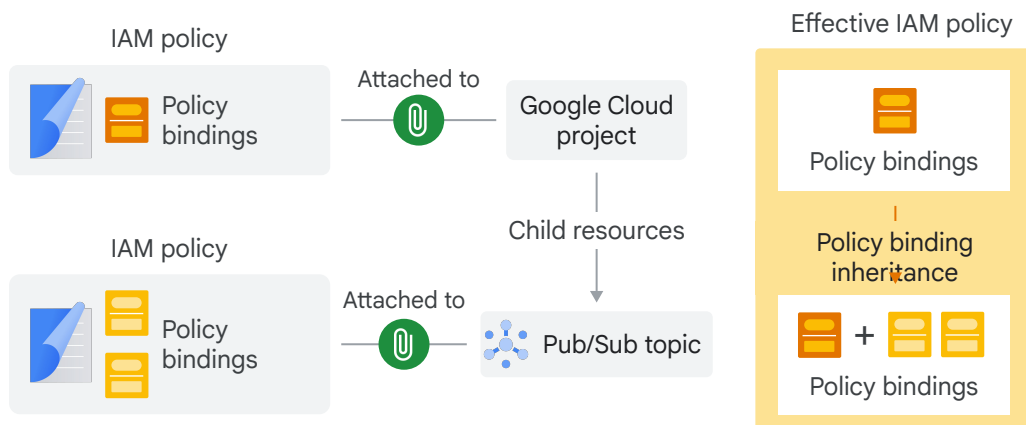
How useful would it be to add that policy binding to a project, instead of to an individual Pub/Sub topic?

Well, that is very useful, because if you add a policy binding to a higher level resource, it's inherited by lower level resources.

In this example, the lower level resources such as the Pub/Sub topics inherit policy bindings from their parent resource, the Google Cloud project.

This is useful when you need to grant permission to publish messages on *all topics in a project*, as opposed to just to a single topic.

# Effective IAM policy



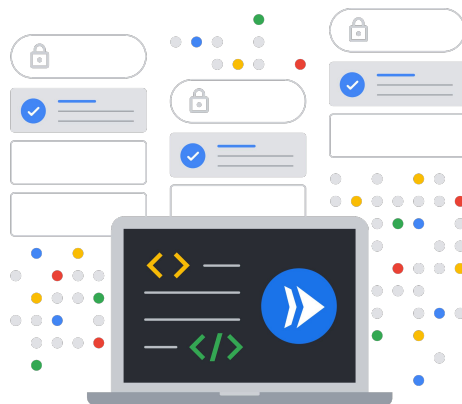
Here's another way to look at policy binding inheritance.

When IAM evaluates access to a resource, it evaluates policy bindings from the parent resource (and their parent) too.

The effective IAM policy on the resource includes those bindings that are granted to a parent.

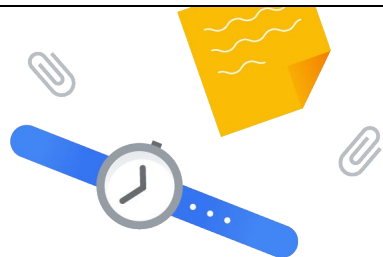
## Remember

- 1 Google Cloud resources are organized into a hierarchy.
- 2 The project is the base-level entity for organizing resources.
- 3 Resources inherit policy bindings from all their ancestors.



In summary:

- Google Cloud resources are organized into a hierarchy, with the *project* as the base-level entity for organizing resources.
- Resources inherit policy bindings from their parent and all their ancestors.



01 Service account and identity

---

02 Resource hierarchy

---

03 Principle of least privilege

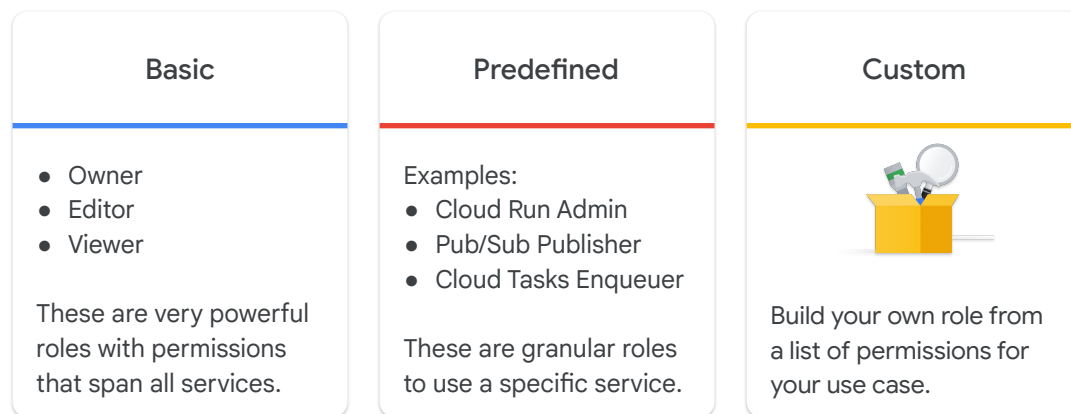
---

## Agenda



As we continue the discussion on roles and permissions in Google Cloud, let's talk about the principle of least privilege.

## Types of IAM roles



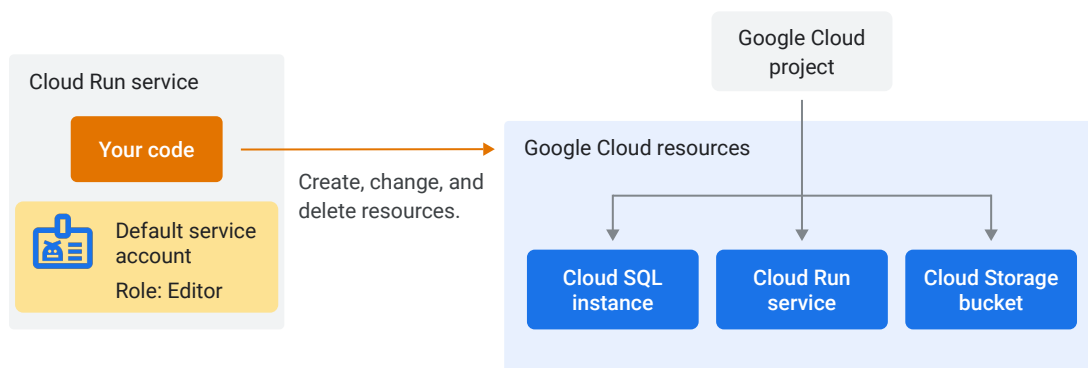
There are three types of roles in IAM:

Basic roles include the Owner, Editor, and Viewer roles. These roles include many permissions across all Google Cloud services. Do not grant these roles by default in production environments. Instead, grant the most limited predefined or custom roles that meet your needs.

Then there are predefined roles, which provide granular access to a specific resource and are managed by Google Cloud.

And finally there are custom roles, which provide granular access according to a user-specified list of permissions.

## Default service account in Cloud Run



Google Cloud

As previously mentioned, if you deploy a Cloud Run service and do not specify a service account, a default service account is used.

The default service account used is the Compute Engine service account which has the broad *Editor* role on the project.

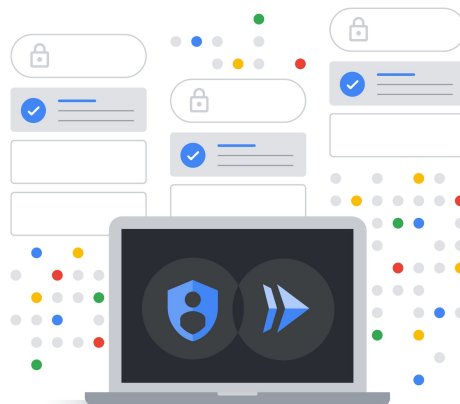
Because of policy binding inheritance, the default service account has read and write permissions on most resources in your project.

While convenient, it's an inherent security risk as resources can be created, modified, or deleted with this service account.



# Use the principle of least privilege

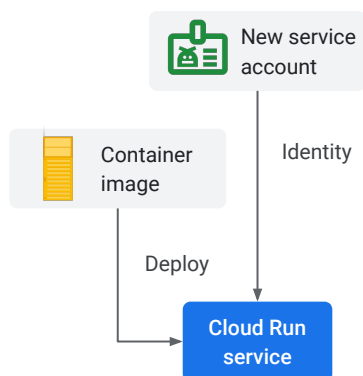
- 1 Create a new service account for a Cloud Run service.
- 2 Configure the service account as the Cloud Run service's identity.
- 3 Add policy bindings for the identity with predefined or custom roles to resources that your service needs to access.



To mitigate this security risk, you should:

1. Create a new service account for a Cloud Run service.
2. Configure the service account as the Cloud Run service's identity.
3. Add policy bindings for this identity with predefined or custom roles on resources that your service needs to access.

## Create a new service account

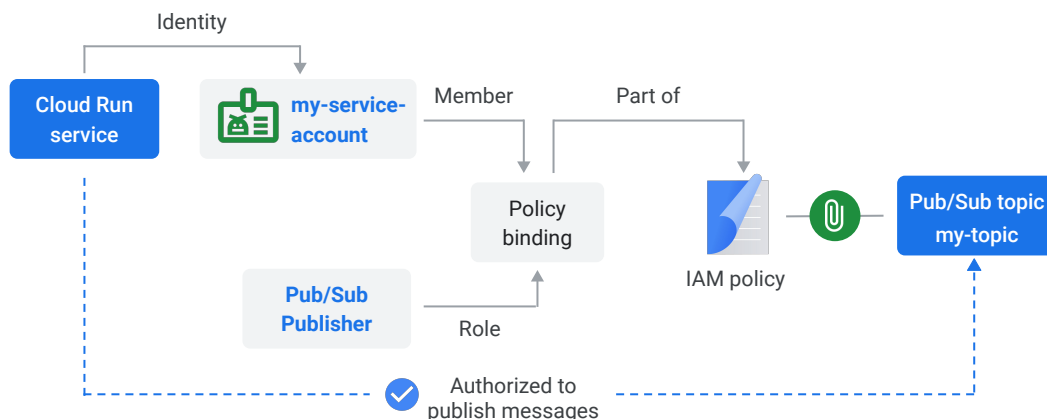


The first step is to create a new user-managed service account for a Cloud Run service, and set it as the service identity of the service.

You can create a service account in the Google Cloud console or with the `gcloud` CLI.

You can set the service account for a Cloud Run service when you create or update a service, and when you deploy a new service revision. This can be done in the Google Cloud console, the `gcloud` CLI, a YAML file, or with Terraform.

## Add policy bindings with predefined roles



```
gcloud pubsub topics add-iam-policy-binding my-topic \
--member="my-service-account-email" --role="roles/pubsub.publisher"
```

By default, this service account does not have any permissions. You've just created it, and it doesn't appear in any policy binding.

If you call any Google Cloud API from code that runs as part of the Cloud Run service, the call will be rejected by IAM.

To grant the service account permissions, you add a policy binding with the appropriate role for the service account member to the IAM policy that is attached to the required resource.

For example, to enable your Cloud Run service to publish a message to a Pub/Sub topic, add a policy binding with the role *Pub/Sub Publisher* to the IAM policy that's attached to the topic.

## Remember

- 1 A Cloud Run service has access to a default service account.
- 2 Google client libraries use the default service account to call Google Cloud APIs.
- 3 The default service account has the Editor basic role, which has broad permissions across all services.
- 4 In production environments, you must replace the default service account with a user-managed per-service account with predefined roles.



### In summary:

- A Cloud Run service has access to a default service account.
- Google client libraries use the default service account to call Google Cloud APIs.
- The default service account has the Editor basic role, which has broad permissions across all services.
- In production environments, you must replace the default service account with a user-managed per-service account with predefined roles.



# Agenda



- 01 Service account and identity
- 02 Resource hierarchy
- 03 Principle of least privilege
- 04** Secrets and environment variables

In this topic, we discuss how you can use secrets and environment variables with your Cloud Run services.

# Environment variables in Cloud Run

## Service

```
gcloud run deploy my-service --image \
my-container-image-url \
--update-env-vars F00=bar,BAZ=boo
```

## Job

```
gcloud run jobs create my-job --image \
my-container-image-url \
--update-env-vars F00=bar,BAZ=boo
```

Cloud Run environment variables are:

- Set as key-value pairs.
- Injected into your application container.
- Accessed by your application code at runtime.
- Set when you create or update a service or job, or deploy a new revision of your service.
- Set, updated, or removed in the Google Cloud console, with the gcloud CLI, YAML file, or Terraform.

Environment variables are key-value pairs that can be used by your service's application code to control functionality.

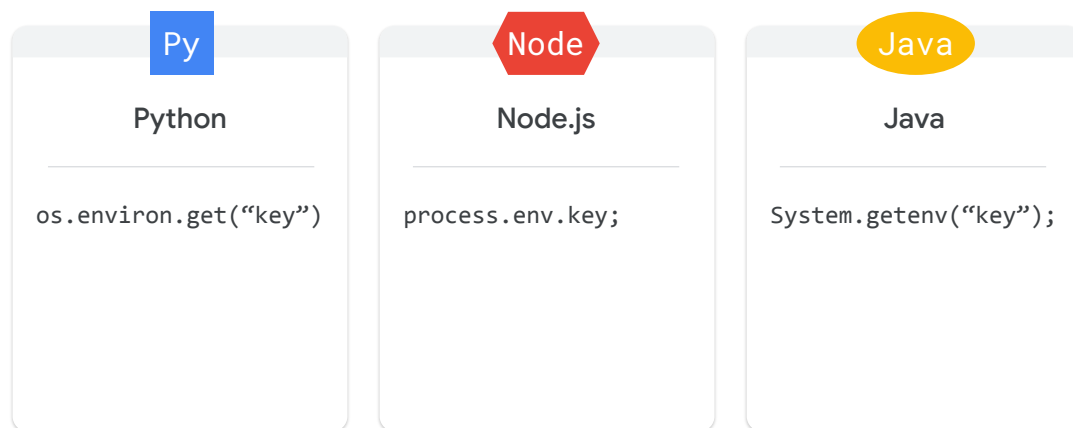
When you set environment variables in Cloud Run, they are injected into your application container and are made accessible to your code.

There are certain reserved environment variables that cannot be set. A list of these variables is documented in the [container runtime contract](#).

You can set environment variables when creating or updating a service or job, or when deploying a new service revision.

You can set default environment variables in the container with the ENV statement in a Dockerfile. An environment variable set with the same name on a Cloud Run service or job overrides the value set in the default variable.

## Accessing environment variables



To access environment variables in your application code, you use the appropriate functions in library modules that are available for your programming language.

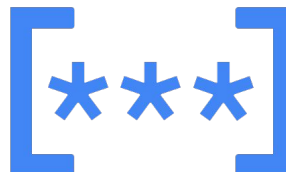
For example, to access environment variables in your code that is written in Python, use the `environ.get` function in the `os` module.

For Node.js, use `process.env`, and for Java use `System.getenv()`.

# Using secrets in Cloud Run

A secret is an object that contains:

- A collection of metadata that includes:
  - Replication locations
  - Labels
  - Permissions
- Secret versions that store the secret data



Secret Manager:

- A service that lets you store, manage, and access secrets.

Your service or job might need to access downstream services that require sensitive configuration such as API keys, passwords, or other information.

For Cloud Run services, it's recommended to store this type of sensitive information in a secret that is created in Secret Manager.

Secret Manager is a Google Cloud service that lets you store, manage, and access secrets.

A secret is an object that contains a collection of metadata like replication locations, labels, permissions, and other information; and secret versions.

A secret version stores the actual secret data such as an API key or password as a text string or binary blob.



# Accessing secrets

To access a secret from your service:

- Make the secret available to the service as:
  - A file by mounting the secret as a volume
  - An environment variable
- Deploy or update your service with the specified secrets.

```
# secret mounted as a volume

gcloud run deploy my-service --image \
my-container-image --update-secrets= \
SECRET_FILE_PATH=my_secret:VERSION

# secret passed as an environment variable

gcloud run deploy my-service --image \
my-container-image --update-secrets= \
ENV_VAR_NAME=my_secret:VERSION
```

Google Cloud

You can make a secret accessible to your service or job running in Cloud Run in either of two ways:

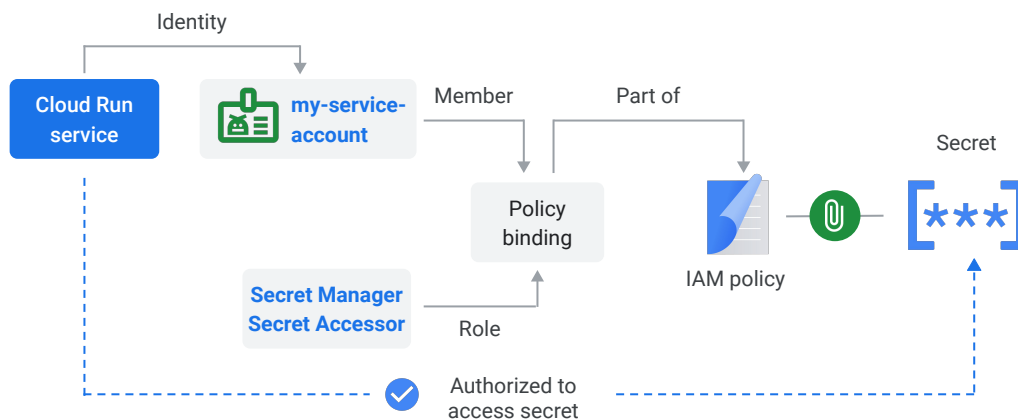
- Mount the secret as a volume, which makes the secret available to the container as a file. Reading a volume always fetches the secret value from Secret Manager, so it can be used with the *latest* version.
- Pass a secret to your Cloud Run service as an environment variable. Environment variables are resolved at instance startup time, so if you use this method, it's recommended that you pin the secret to a particular version rather than using *latest*.

You can make a secret accessible to your service when deploying it to Cloud Run. You can update existing secrets by deploying a new revision or updating the service.

You can do this in the Google Cloud console, with the gcloud CLI, or using a YAML file.

Any configuration change such as updating secrets leads to the creation of a new service revision. Subsequent revisions will also automatically get this configuration setting.

## Allowing access to secrets



```
gcloud secrets add-iam-policy-binding my-secret-id \
--member="my-service-account-email" --role="roles/secretmanager.secretAccessor"
```

To allow a Cloud Run service to access a secret, you must grant the *Secret Manager Secret Accessor* role to the Cloud Run service account.

## Remember

- 1 Environment variables are set as key-value pairs and made available to your Cloud Run service or job.
- 2 The default value of an environment variable set in the container's Dockerfile can be overridden when a service is deployed.
- 3 Use secrets to store and access sensitive information in a Cloud Run service or job.
- 4 To access a secret, mount it as a volume or provide it as an environment variable for the service or job.



### In summary:

- Environment variables are set as key-value pairs and injected into your application container to make them accessible to your service or job.
- The default value of an environment variable set in the container's Dockerfile can be overridden when a service is deployed.
- Use secrets to store and access sensitive information in a Cloud Run service or job.
- To access a secret, mount it as a volume or provide it as an environment variable for the service or job.



# Agenda



- 01 Service account and identity
- 02 Resource hierarchy
- 03 Principle of least privilege
- 04 Secrets and environment variables
- 05** Lab: Implementing Least Privilege IAM Policy Bindings in Cloud Run

You now complete a lab on implementing least privilege in Cloud Run.

# Lab



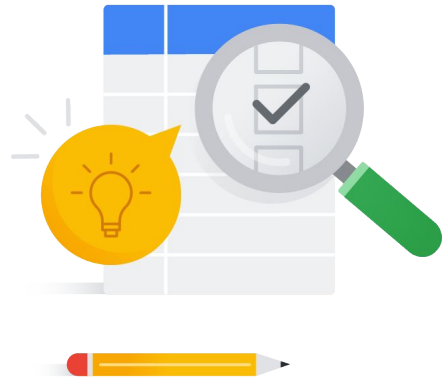
30 min



Individual

## Implementing Least Privilege IAM Policy Bindings in Cloud Run

Implement IAM policy bindings in Cloud Run to adhere to the security principle of least privilege.



This lab teaches you how to implement IAM policy bindings in Cloud Run to adhere to the security principle of least privilege.

## Lab instructions



30 min



Individual



Tasks

- Create and deploy a public service to Cloud Run.
- Update the service to require authentication.
- Invoke the service with authentication credentials.
- Implement the principle of least privilege with service account permissions.

1

Deploy a public service to Cloud Run.

2

Authenticate service requests.

3

Invoke the service.

4

Implement least privilege.



# Agenda



- 01 Service account and identity
- 02 Resource hierarchy
- 03 Principle of least privilege
- 04 Secrets and environment variables
- 05 Lab: Implementing Least Privilege IAM Policy Bindings in Cloud Run
- 06 Quiz**

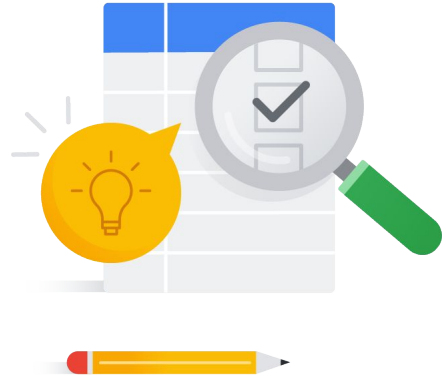
Let's do a short quiz on the topics that were discussed in this module.

## Quiz

🕒 5 min

👥 Group

### Introduction to Containers



Let's do a short quiz on this module.



## Quiz | Question 1

### Question

Every Cloud Run service is linked to a service account by default. What role is given to this service account?

- A. Owner
- B. Editor
- C. Viewer
- D. Billing Admin

## Quiz | Question 2

### Question

What are some characteristics of an IAM policy? Select three.

- A. An IAM policy is attached to a Google Cloud resource.
- B. An IAM policy consists of at most one policy binding that binds a member to one or more roles.
- C. An IAM policy consists of a list of policy bindings that binds members to roles.
- D. A member can only have a single role in an IAM policy.
- E. You can attach only one IAM policy to a resource.

## Quiz | Question 3

### Question

How can you implement the principle of least privilege for a Cloud Run service?

- A. Remove the Editor role on the default service account that is used by Cloud Run.
- B. Use secrets to store data that is needed by the Cloud Run service.  
Create a new service account and configure it as the Cloud Run service's identity.
- C. Grant minimal permissions to the account on the resources that the service needs to access.
- D. Use only Google client libraries to access Google Cloud APIs and services.

## Quiz | Question 4

### Question

Which statements about using environment variables with Cloud Run are correct? Select two.

- A. Once set, the value of an environment variable cannot be updated.
- B. Environment variables are key-value pairs that can be set when deploying a Cloud Run service or job.
- C. Environment variables are injected into your application container and accessed by your code at runtime.
- D. Environment variables set in the container with the ENV statement in a Dockerfile take precedence over variables with the same name set on a Cloud Run service or job.

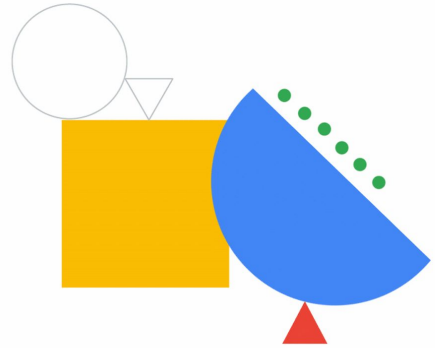
## Quiz | Question 5

### Question

What are two methods of making a secret available to a Cloud Run service?

- A. Provide the secret as an environment variable when deploying the service.
- B. Provide the secret name and value as query parameters in the request to the service.
- C. Mount the secret as a volume so that the service can access the secret from a file.
- D. Secrets cannot be accessed from a Cloud Run service.

## Review: Service Identity and Authentication



Let's review the topics that were discussed in this module.

## In this module, you learned about:

- |    |                                   |
|----|-----------------------------------|
| 01 | Service account and identity      |
| 02 | Resource hierarchy                |
| 03 | Principle of least privilege      |
| 04 | Secrets and environment variables |



In this module, we discussed what is a service account and how you can provide your Cloud Run service with its own identity, by creating and setting a user-managed service account. We discussed the role of the service account when accessing Google Cloud APIs and how they can be used for service to service communication.

We reviewed how resources in Google Cloud are organized hierarchically. You learned about policy bindings, how they are inherited by resources from their parents, and how they are used to allow or deny access to resources.

We also discussed the principle of least privilege that should be followed when granting roles and permissions to members including service accounts.

Finally, we discussed how you can provide your Cloud Run service or job access to environment variables and secrets.