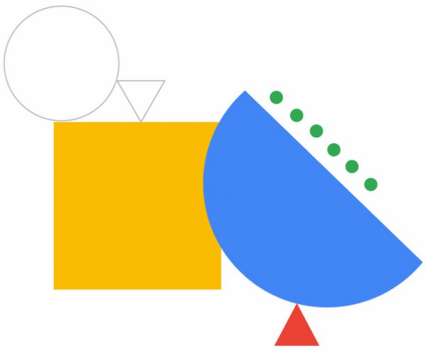
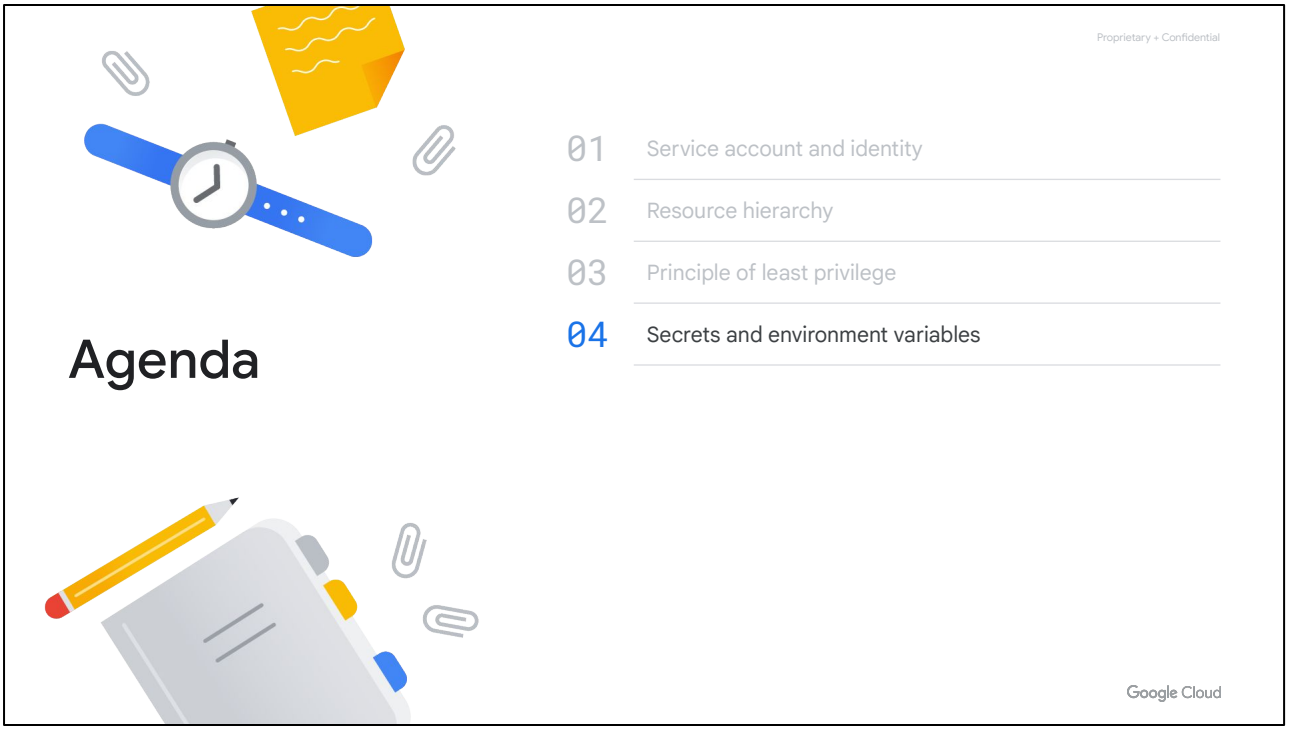
Google Cloud

Service Identity and Authentication



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



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 FOO=bar,BAZ=boo
```

Job

```
gcloud run jobs create my-job --image \
my-container-image-url \
--update-env-vars FOO=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.

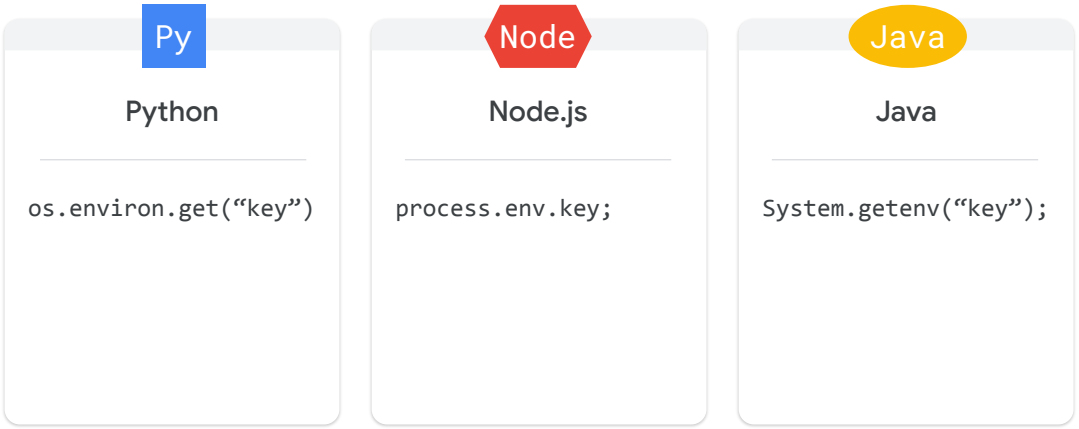
When you set environment variables in Cloud Run, they are injected into your application container and are made accessible to your code.

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().

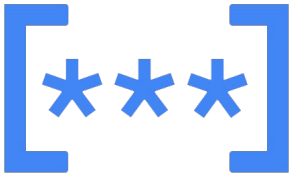
Using secrets in Cloud Run

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 enables you to 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
```


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

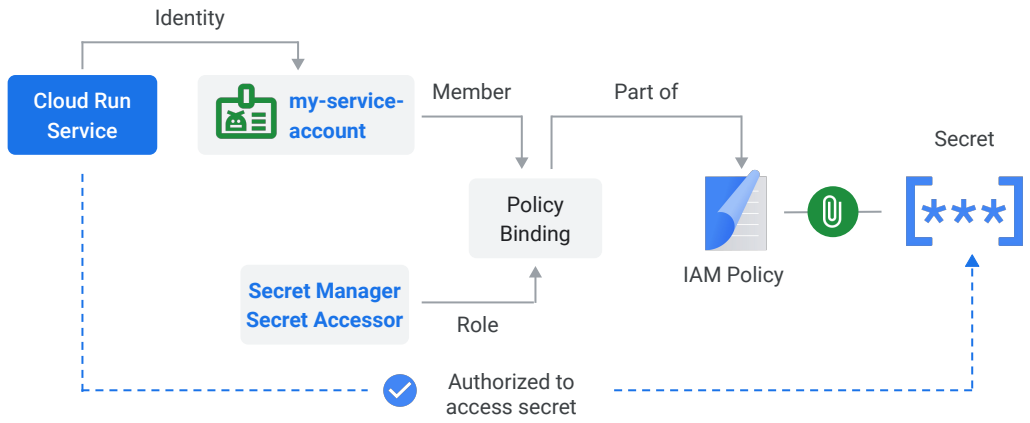
- Mount the secret as a volume, which makes the secret available to the container as a file. Reading a volume always fetches the secret value from Secret Manager, so it can be used with the *latest* version.
- Pass a secret to your Cloud Run service as an environment variable. Environment variables are 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.