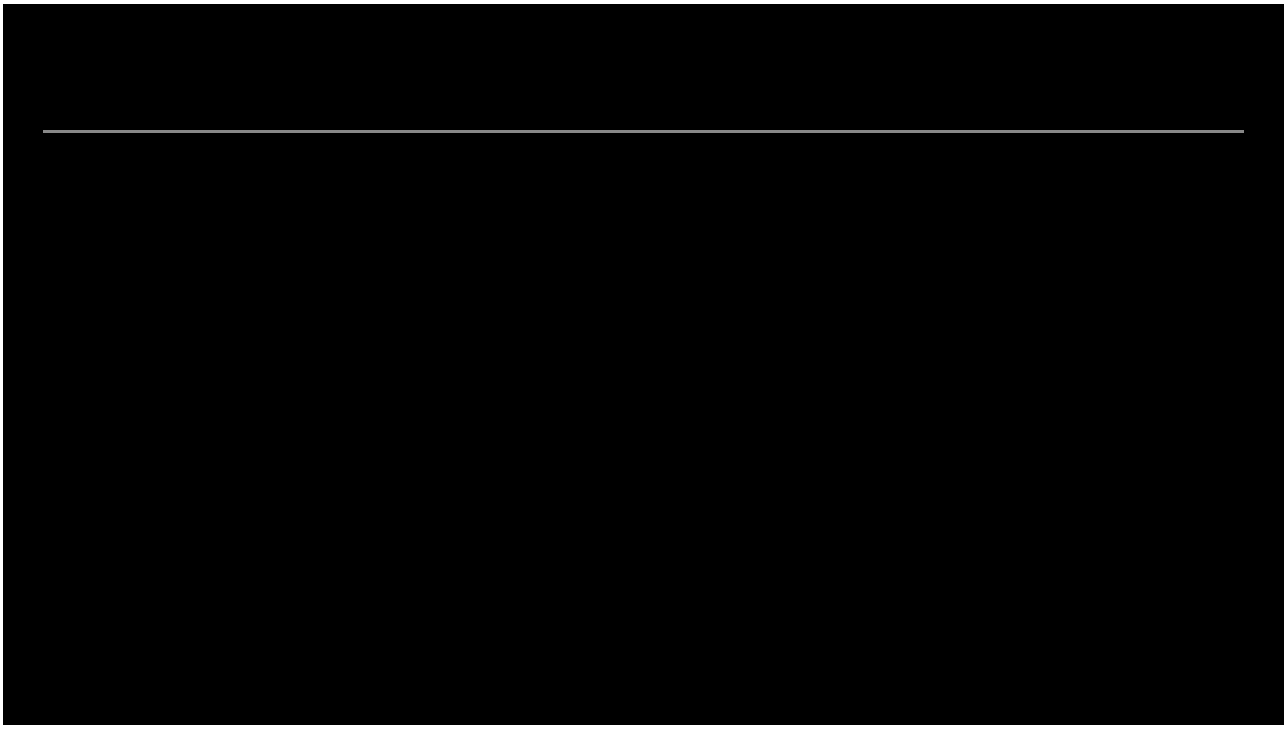
Service accounts overview

Archived: 2026-04-05 23:38:57 UTC



This page explains what service accounts are and describes important considerations for managing your service accounts at each stage of their lifecycle.

What are service accounts?

A service account is a special kind of account typically used by an application or compute workload, such as a Compute Engine instance, rather than a person. A service account is identified by its email address, which is unique to the account.

Applications use service accounts to make authorized API calls by authenticating as either the service account itself, or as Google Workspace or Cloud Identity users through domain-wide delegation. When an application authenticates as a service account, it has access to all resources that the service account has permission to access.

The most common way to let an application authenticate as a service account is to attach a service account to the resource running the application. For example, you can attach a service account to a Compute Engine instance so that applications running on that instance can authenticate as the service account. Then, you can grant the service account IAM roles to let the service account—and, by extension, applications on the instance—access Google Cloud resources.

There are other ways to let applications authenticate as service accounts besides attaching a service account. For example, you could set up Workload Identity Federation to allow external workloads to authenticate as service

accounts, or create a service account key and use it in any environment to obtain OAuth 2.0 access tokens.

To learn more about service account authentication for applications, see Overview of identities for workloads.

Principals, such as users and other service accounts, can also authenticate as service accounts. For more information, see Service account impersonation on this page.

Types of service accounts

In Google Cloud, there are several different types of service accounts:

- **User-managed service accounts**: Service accounts that you create and manage. These service accounts are often used as identities for workloads.
- **Default service accounts**: User-managed service accounts that are created automatically when you enable certain Google Cloud services. You are responsible for managing these service accounts.
- **Service agents**: Service accounts that are created and managed by Google Cloud, and that allow services to access resources on your behalf.

To learn more about the different types of service accounts, see Types of service accounts.

Service account credentials

Applications and principals authenticate as a service account by doing one of the following:

- Obtaining short-lived credentials. In many cases, such as attached service accounts and commands using the gcloud CLI `--impersonate-service-account` flag, these credentials are obtained automatically—you don't need to create or manage them yourself.
- Using a service account key to sign a JSON Web Token (JWT) and exchanging it for an access token. Because service account keys are a security risk if not managed correctly, you should choose a more secure alternative to service account keys whenever possible.

To learn more about service account authentication, see Service account credentials.

Service account impersonation

When an authenticated principal, such as a user or another service account, authenticates as a service account to gain the service account's permissions, it's called *impersonating* the service account. Impersonating a service account lets an authenticated principal access whatever the service account can access. Only authenticated principals with the appropriate permissions can impersonate service accounts.

Impersonation is useful when you want to change a user's permissions without changing your Identity and Access Management (IAM) policies. For example, you can use impersonation to temporarily grant a user elevated access, or to test whether a specific set of permissions is sufficient for a task. You can also use impersonation to locally develop applications that can only run as a service account, or to authenticate applications that run outside of Google Cloud.

To learn more about service account impersonation, see Service account impersonation.

Service accounts and Google Workspace domains

Service accounts do **not** belong to your Google Workspace domain, unlike user accounts. If you share Google Workspace assets, like documents or events, with your entire Google Workspace domain, they are not shared with service accounts.

Service accounts can't own Google Workspace assets. They can, however, create assets in a shared drive in the domain. Assets created in a shared drive are owned and managed by your organization.

Service account permissions

Service accounts are principals. This means that you can grant service accounts access to Google Cloud resources. For example, you could grant a service account the Compute Admin role (`roles/compute.admin`) on a project. Then, the service account would be able to manage Compute Engine resources in that project.

However, service accounts are also resources. This means that you can give other principals permission to access the service account. For example, you could grant a user the Service Account User role (`roles/iam.serviceAccountUser`) on a service account to let the user attach that service account to resources. Or, you could grant a user the Service Account Admin role (`roles/iam.serviceAccountAdmin`) to let the user do things like view, edit, disable, and delete the service account.

The following sections discuss how to manage service accounts as principals and as resources.

Service accounts as principals

Because service accounts are principals, you can manage access for service accounts just like you would for any other principal. For example, if you want to let your application's service account access objects in a Cloud Storage bucket, you can grant the service account the Storage Object Viewer role (`roles/storage.objectViewer`) on the bucket. Or, if you want to ensure that an application's service account can't change a project's IAM policies, you could use a deny policy to prevent the service account from using the `resourcemanager.projects.setIamPolicy` permission on that project. To learn more about managing access for principals, see Access in Google Cloud.

You can manage access for individual service accounts or for all service accounts in a specific project, folder, or organization. When managing access for service accounts, use the following principal identifiers to refer to the service accounts:

Principal type	Principal identifier

An individual service account	`serviceAccount:`*`SA_EMAIL_ADDRESS`* **Example:** `serviceAccount:my-service-account@my-project.iam.gserviceaccount.com`
All service accounts in a project	`principalSet://cloudresourcemanager.googleapis.com/projects/`*`PROJECT_NUMBER`*`/type/ServiceAccount` **Example:** `principalSet://cloudresourcemanager.googleapis.com/projects/123456789012/type/ServiceAccount`
All service accounts in all projects in a folder	`principalSet://cloudresourcemanager.googleapis.com/folders/`*`FOLDER_NUMBER`*`/type/ServiceAccount` **Example:** `principalSet://cloudresourcemanager.googleapis.com/folders/123456789012/type/ServiceAccount`
All service accounts in all projects in an organization	`principalSet://cloudresourcemanager.googleapis.com/organizations/`*`ORGANIZATION_NUMBER`*`/type/ServiceAccount` **Example:** `principalSet://cloudresourcemanager.googleapis.com/organizations/123456789012/type/ServiceAccount`

If service accounts in a specific project, folder, or organization share requirements, then use service account principal sets to grant them roles instead of using custom groups.

As with all types of principals, you should only give the service account the minimum set of permissions required to achieve its goal.

To learn how to grant roles to principals, including service accounts, see Manage access to projects, folders, and organizations.

Service accounts as resources

Service accounts are also resources that can have their own allow policies. As a result, you can let other principals access a service account by granting them a role on the service account, or on one of the service account's parent resources. For example, to let a user impersonate a service account, you could grant the user the Service Account Token Creator role (`roles/iam.serviceAccountTokenCreator`) on the service account.

When granting a role that allows a user to impersonate a service account, keep in mind that the user can access all the resources that the service account can access. Use caution when letting users impersonate highly privileged

service accounts, such as the Compute Engine and App Engine default service accounts.

For more information on the roles that you can grant to principals on service accounts, see Service account permissions.

To learn how to grant a principal a role on a service account, see Manage access to service accounts.

Service account lifecycle

As you manage your projects, you'll likely create, manage, and delete many different service accounts. This section describes key considerations for managing your service accounts at the various stages of their lifecycle.

Where to create service accounts

Each service account is located in a project. After you create a service account, you cannot move it to a different project.

There are a few ways to organize your service accounts into projects:

- **Create service accounts and resources in the same project.**

  This approach makes it easier to get started with service accounts. However, it can be difficult to keep track of your service accounts when they are spread across many projects.

- **Centralize service accounts in separate projects.**

  This approach puts all of the service accounts for your organization in a small number of projects, which can make the service accounts easier to manage. However, it requires extra setup if you attach service accounts to resources in other projects, which allows those resources to use the service account as their identity.

  When a service account is in one project, and it accesses a resource in another project, you usually must enable the API for that resource in both projects. For example, if you have a service account in the project `my-service-accounts` and a Cloud SQL instance in the project `my-application`, you must enable the Cloud SQL API in both `my-service-accounts` and `my-application`.

The number of service accounts that you can have in each project depends on your project. To view the quota for a project, view your project's quotas in the Google Cloud console and search for **Service Account Count**.

To learn how to create a service account, see Create service accounts.

Prevent the creation of service accounts

To better control where service accounts are created, you might want to prevent service account creation in some projects in your organization.

You can prevent the creation of service accounts by enforcing the `constraints/iam.disableServiceAccountCreation` organization policy constraint in an organization, project, or

folder.

Before you enforce this constraint, consider the following limitations:

- If you enforce this constraint in a project, or in all projects within an organization, then some Google Cloud services cannot create default service accounts. As a result, if the project runs workloads that need to authenticate as a service account, the project might not contain a service account that the workload can use.

  To address this issue, you can enable service account impersonation across projects. When you enable this feature, you can create service accounts in a centralized project, then attach the service accounts to resources in other projects. Workloads running on those resources can use the attached service accounts to authenticate, making the default service accounts unnecessary.

- Some features, such as Workload Identity Federation, require you to create service accounts.

  If you don't use Workload Identity Federation, consider using organization policy constraints to block federation from all identity providers.

Keep track of service accounts

Over time, as you create more and more service accounts, you might lose track of which service account is used for what purpose.

The display name of a service account is a good way to capture additional information about the service account, such as the purpose of the service account or a contact person for the account. For new service accounts, you can populate the display name when creating the service account. For existing service accounts use the `serviceAccounts.update()` method to modify the display name.

Use service accounts with Compute Engine

Compute Engine instances need to run as service accounts to have access to other Google Cloud resources. To help secure your Compute Engine instances, consider the following:

- You can create instances in the same project with different service accounts. To change the service account of an instance after it's created, use the `instances.setServiceAccount` method.
- To set up authorization for attached service accounts, you need to configure access scopes in addition to configuring IAM roles.
- Since instances depend on their service accounts to have access to Google Cloud resources, avoid deleting service accounts when they are still used by running instances.

To learn more about using service accounts with Compute Engine, see Service accounts in the Compute Engine documentation.

Identify unused service accounts

After some time, you might have service accounts in your projects that you no longer use.

Unused service accounts create an unnecessary security risk, so we recommend disabling unused service accounts, then deleting the service accounts when you are sure that you no longer need them. You can use the following methods to identify unused service accounts:

- **Service account insights** tell you which service accounts in your project have not authenticated in the past 90 days.
- **Activity Analyzer** lets you check when a service account or key was last used.

You can also use service account usage metrics to track service account and key usage generally.

If you are an Security Command Center Premium customer, you can use Event Threat Detection to get a notification when a dormant service account triggers an action. Dormant service accounts are service accounts that have been inactive for more than 180 days. After a service account is used, it is no longer dormant.

Delete service accounts

Before deleting a service account, disable the service account to make sure it isn't necessary. Disabled service accounts can be re-enabled if they are still in use.

After you confirm that a service account isn't necessary, you can delete the service account.

Recreate deleted service accounts

It is possible to delete a service account and then create a new service account with the same name.

When you delete a service account, its role bindings are not immediately deleted. Instead, the role bindings list the service account with the prefix `deleted:`. For an example, see Policies with deleted principals.

If you create a new service account with the same name as a recently deleted service account, the old bindings may still exist; however, they **will not apply to the new service account** even though both accounts have the same email address. This behavior occurs because service accounts are given a unique ID within Identity and Access Management (IAM) at creation. Internally, all role bindings are granted using these IDs, not the service account's email address. Therefore, any role bindings that existed for a deleted service account do not apply to a new service account that uses the same email address.

Similarly, if you attach a service account to a resource, then delete the service account and create a new service account with the same name, the new service account **will not** be attached to the resource.

To prevent this unexpected behavior, consider using a new, unique name for every service account. Also, if you accidentally delete a service account, you can try to undelete the service account instead of creating a new service account.

If you cannot undelete the original service account, and you need to create a new service account with the same name and the same roles, you must grant the roles to the new service account. For details, see Policies with deleted principals.

If you also need the new service account to be attached to the same resources as the original service account, do one of the following:

- For Compute Engine instances, you can change the service account that is attached to the instance to replace the original service account with the new service account.
- For all other resources, you must delete the existing resource, then create a new resource of the same type and attach the new service account.

What's next

- Find out how to create service accounts.
- Get best practices for working with service accounts.
- Review best practices for managing service account keys.

Try it for yourself

If you're new to Google Cloud, create an account to evaluate how our products perform in real-world scenarios. New customers also get $300 in free credits to run, test, and deploy workloads.

Get started for free
