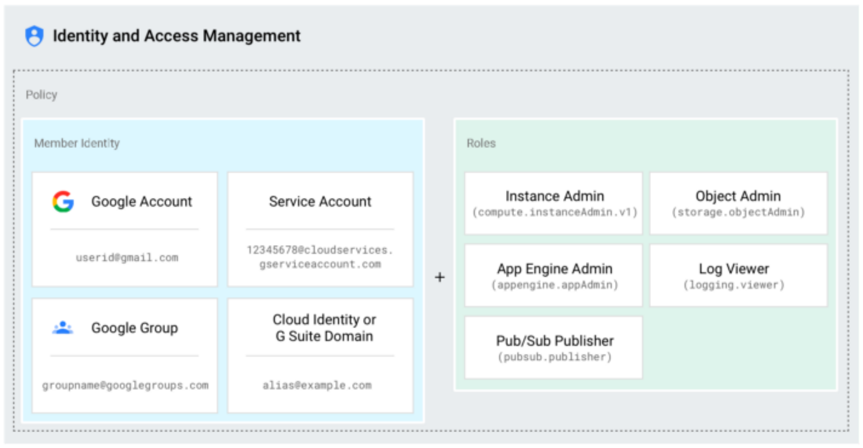
GCP Certification Series: 5.2 Managing service accounts

Prashanta Paudel
Nov 14, 2018 · 26 min read

Overview

This page describes the basic concepts of Cloud Identity and Access Management.

Google Cloud Platform (GCP) offers Cloud IAM, which lets you manage access control by defining ***who* (identity) has *what access* (role) for *which* resource.**



https://cloud.google.com/iam/docs/overview

With Cloud IAM you can grant granular access to specific GCP resources and prevent unwanted access to other resources. Cloud IAM lets you adopt the security principle of least privilege, so you grant only the necessary access to your resources.

Concepts related to identity

In Cloud IAM, you grant access to **members**. Members can be of the following types:

- Google account
- Service account
- Google group
- G Suite domain
- Cloud Identity domain

Google account

A Google account represents a developer, an administrator, or any other person who interacts with GCP. Any email address that is associated with a Google account can be an identity, including gmail.com or other domains. New users can sign up for a Google account by going to the Google account signup page.

Service account

A service account is an account that belongs to your application instead of to an individual end user. When you run code that is hosted on GCP, you specify the account that the code should run as. You can create as many service accounts as needed to represent the different logical components of your application. For more information about using a service account in your application, see Getting started with authentication.

Google group

A Google group is a named collection of Google accounts and service accounts. Every group has a unique email address that is associated with the group. You can find the email address that is associated with a Google group by clicking **About** on the homepage of any Google group. For more information about Google groups, see the Google groups homepage.

Google groups are a convenient way to apply an access policy to a collection of users. You can grant and change access controls for a whole group at once instead of granting or changing access controls one-at-a-time for individual users or service accounts. You can also easily add members to and remove members from a Google group instead of updating a Cloud IAM policy to add or remove users.

Note that Google groups don't have login credentials, and you cannot use Google groups to establish identity to make a request to access a resource.

G Suite domain

G Suite domain represents a virtual group of all the Google accounts that have been created in an organization's G Suite account. G Suite domains represent your organization's Internet domain name (such as *example.com*), and when you add a user to your G Suite domain, a new Google account is created for the user inside this virtual group (such as *username@example.com*).

Like Google groups, G Suite domains cannot be used to establish identity, but they enable convenient permission management.

Cloud Identity domain

A Cloud Identity domain is like a G Suite domain because it represents a virtual group of all Google accounts in an organization. However, Cloud Identity domain users don't have access to G Suite applications and features. For more information, see About Cloud Identity.

allAuthenticatedUsers

This is a special identifier that represents anyone who is authenticated with a Google account or a service account. Users who are not authenticated, such as anonymous visitors, are not included.

allUsers

This is a special identifier that represents anyone who is on the internet, including authenticated and unauthenticated users. Note that some GCP APIs require authentication of any user accessing the service, and in those cases, allUsers will only imply authorization for all authenticated users.

Concepts related to access management

When an authenticated member attempts to make a request, Cloud IAM makes an authorization decision about whether the member is allowed to perform the operation on a resource.

This section describes the entities and concepts involved in the authorization process.

Resource

You can grant access to users for a GCP resource. Some examples of resources are projects,Compute Engine instances, and Cloud Storage buckets.

Some services such as Cloud Pub/Sub and Compute Engine support granting Cloud IAM permissions at a granularity finer than the project-level. For example, you can grant the `pubsub.subscriber` role to a user for a particular Cloud Pub/Sub topic or you can grant the `compute.instanceAdmin` role to a user for a specific Compute Engine instance.

In other cases, you can grant Cloud IAM permissions at the project level. The permissions are then inherited by all resources within that project. For example, to grant access to a Cloud Storage bucket, you must grant the access to the project that contains the bucket. For information on what roles can be granted on which resources, see Understanding Roles.

Permissions

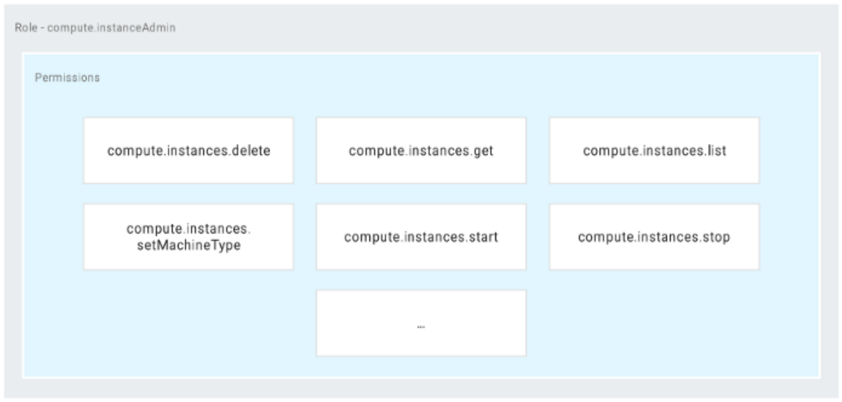
Permissions determine what operations are allowed on a resource. In the Cloud IAM world, permissions are represented in the form of `<service>.<resource>.<verb>`, for example `pubsub.subscriptions.consume`.

Permissions usually, but not always, correspond 1:1 with REST methods. That is, each GCP service has an associated set of permissions for each REST method that it exposes. The caller of that method needs those permissions to call that method. For example, the caller of `Publisher.Publish()` needs the `pubsub.topics.publish` permission.

You don't assign permissions to users directly. Instead, you assign them a **Role which** contains one or more permissions.

Roles

A role is a collection of permissions. You cannot assign a permission to the user directly; instead, you grant them a role. When you grant a role to a user, you grant them all the permissions that the role contains.



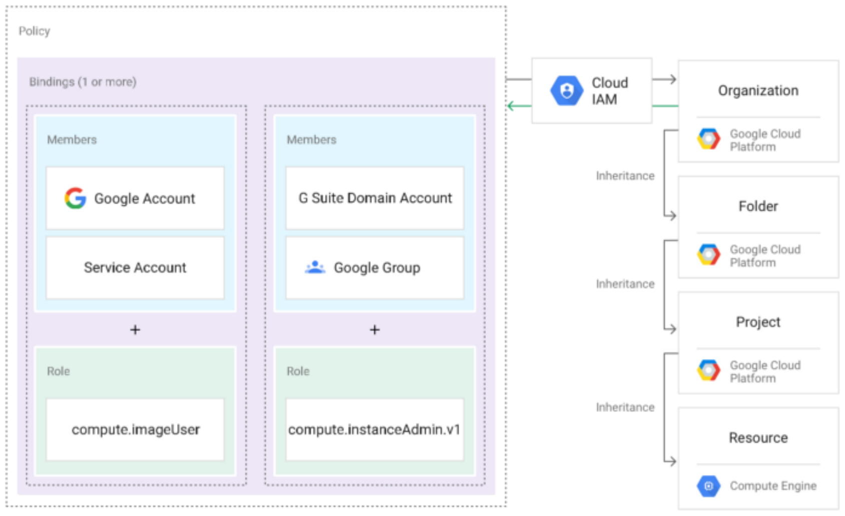
There are three kinds of roles in Cloud IAM:

- **Primitive roles**: The roles historically available in the Google Cloud Platform Console will continue to work. These are the **Owner**, **Editor**, and **Viewer** roles.
- **Predefined roles**: Predefined roles are the Cloud IAM roles that give finer-grained access control than the primitive roles. For example, the predefined role **Pub/Sub Publisher**(roles/pubsub.publisher) provides access to *only* publish messages to a Cloud Pub/Sub topic.
- **Custom roles**: Roles that you create to tailor permissions to the needs of your organization when predefined roles don't meet your needs.

To learn how to assign a role to the user, see Granting, Changing, and Revoking Access. For information about the available fine-grained Cloud IAM predefined roles, see Understanding Roles. For information about custom roles, see Understanding Custom Roles and Creating and Managing Custom Roles.

IAM Policy

You can grant roles to users by creating a *Cloud IAM policy*, which is a collection of statements that define who has what type of access. A policy is attached to a resource and is used to enforce access control whenever that resource is accessed.



A Cloud IAM policy is represented by the IAM `Policy` object. An IAM `Policy` object consists of a list of bindings. A `Binding` binds a list of `members` to a `role` .

`role` is the role you want to assign to the member. The `role` is specified in the form of `roles/<name of the role>` . For example, `roles/storage.objectAdmin` , `roles/storage.objectCreator` , and `roles/storage.objectViewer` .

`members` contains a list of one or more identities as described in the Concepts related to identity section above. Each member type is identified with a prefix, such as a Google account (`user:`), service account (`serviceAccount:`), Google group (`group:`), or a G Suite or Cloud identity domain (`domain:`). In the example snippet below, the `storage.objectAdmin` role is assigned to the following members using the appropriate prefix: `user:alice@example.com` , `serviceAccount:my-`

`other-app@appspot.gserviceaccount.com` , `group:admins@example.com` , and `domain:google.com` . The `objectViewer` role is assigned to `user:bob@example.com` .

The following code snippet shows the structure of a Cloud IAM policy.

```
{
  "bindings": [
   {
     "role": "roles/storage.objectAdmin",
     "members": [
       "user:alice@example.com",
       "serviceAccount:my-other-
app@appspot.gserviceaccount.com",
       "group:admins@example.com",
       "domain:google.com" ]
   },
   {
     "role": "roles/storage.objectViewer",
     "members": ["user:bob@example.com"]
   }
   ]
}
```

Cloud IAM and Policy APIs

Cloud IAM provides a set of methods that you can use to create and manage access control policies on GCP resources. These methods are exposed by the services that support Cloud IAM. For example, the Cloud IAM methods are exposed by the Resource Manager, Cloud Pub/Sub, and Cloud Genomics APIs, just to name a few.

The Cloud IAM methods are:

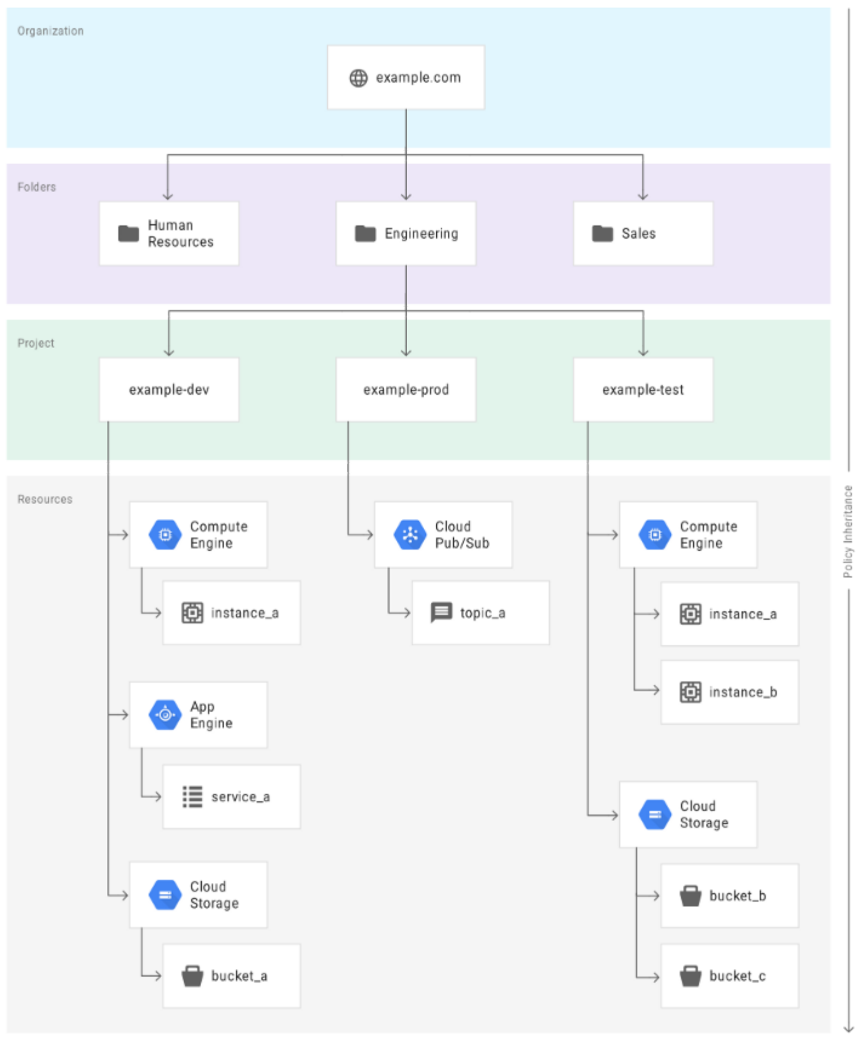
- `setIamPolicy()` : Allows you to set policies on your resources.
- `getIamPolicy()` : Allows you to get a policy that was previously set.
- `testIamPermissions()` : Allows you to test whether the caller has the specified permissions for a resource.

You can find the API reference topics for these methods in the documentation for each service that supports Cloud IAM.

Policy hierarchy

GCP resources are organized hierarchically, where the organization node is the root node in the hierarchy, the projects are the children of the organization, and the other resources are the descendants of projects. Each resource has exactly one parent. See the Resource Manager Resource Hierarchy topic for more information.

The following diagram is an example of a GCP resource hierarchy:



You can set a Cloud IAM policy at any level in the resource hierarchy: the organization level, the folder level, the project level, or the resource level. Resources inherit the policies of the parent resource. If you set a policy at the organization level, it is automatically inherited by all its children projects, and if you set a policy at the project level, it's inherited by all its child resources. The effective policy for a resource is the union of the policy set at that resource and the policy inherited from higher up in the hierarchy.

This policy inheritance is transitive; in other words, resources inherit policies from the project, which inherit policies from folders, which inherit policies from the organization. Therefore, the organization-level policies also apply at the resource level.

For example in the diagram above, topic_a is a Cloud Pub/Sub resource that lives under the project example-prod. If you grant the Editor role to micah@gmail.com for example-prod and grant the Publisher role to song@gmail.com for topic_a, you effectively grant the Editor role for topic_a to micah@gmail.com and the Publisher role to song@gmail.com.

The Cloud IAM policy hierarchy follows the same path as the GCP resource hierarchy. If you change the resource hierarchy, the policy hierarchy changes as well. For example, moving a project into an organization will update the project's Cloud IAM policy to inherit from the organization's Cloud IAM policy.

Child policies cannot restrict access granted at a higher level. For example, if you grant the Editor role to a user for a project, and grant the Viewer role to the same user for a child resource, then the user still has the Editor role grant for the child resource.

Cloud IAM support for GCP services

With Cloud IAM, every API method across all GCP services is checked to make sure that the account making the API request has the appropriate permission to use the resource.

Prior to Cloud IAM, you could only grant Owner, Editor, or Viewer roles. These roles give very broad access on a project and did not allow fine-grained separation of duties. GCP services now offer additional predefined roles that give finer-grained access control than the Owner, Editor, and Viewer roles. For example, Compute Engine offers roles such as *Instance Admin* and *Network Admin* and App Engine offers roles such as *App Admin* and *Service Admin*. These predefined roles are available in addition to the legacy Owner, Editor, and Viewer roles.

If you need even more control over permissions, consider creating a Custom Role.

The following products offer Cloud IAM predefined roles:

- Google Cloud Platform project
- GCP Organization
- Compute Engine
- Cloud Source Repositories
- App Engine
- Cloud Storage
- BigQuery
- Cloud Bigtable
- IAM for Cloud SQL
- Stackdriver Debugger
- Cloud Deployment Manager
- Cloud Genomics
- Cloud Key Management Service
- Cloud Pub/Sub
- Cloud Machine Learning Engine
- Cloud Spanner
- Stackdriver Logging
- Cloud IAM for Stackdriver Monitoring
- Cloud Dataflow
- Cloud IAM for Cloud Datastore
- Cloud IAM for Cloud Dataproc
- Cloud IAM for Google Kubernetes Engine

- Cloud IAM for Cloud DNS
- Cloud IAM for Stackdriver Trace
- Cloud IAM for Cloud Billing API
- Cloud IAM for Service Management

For a complete list of predefined roles, see Understanding Roles.

You can grant users certain roles to access resources at a granularity *finer than the project level*. For example, you can create a Cloud IAM access control policy that grants a user the *Subscriber* role for a particular Cloud Pub/Sub topic. For information on what roles can be granted on which resources, see Understanding Roles.

++++++++++++++++++++++++++++++++++++++
+++

What are Service accounts?

A service account is a special Google account that belongs to your application or a virtual machine(VM), instead of an individual end user.

Your application uses the service account to call the Google API of a service so that the users aren't directly involved.

For example, a Compute Engine VM may run as a service account, and that account can be given permissions to access the resources it needs. This way the service account is the identity of the service, and the service account's permissions control which resources the service can access.

A service account is identified by its email address, which is unique to the account.

Service account keys

Each service account is associated with a key pair, which is managed by Google Cloud Platform (GCP). It is used for service-to-service authentication within GCP. These keys are rotated automatically by Google and are used for signing for a maximum of two weeks.

You may optionally create one or more external key pairs for use from outside GCP (for example, for use with Application Default Credentials). When you create a new key pair, you download the private key (which is not retained by Google). With external keys, you are responsible for the security of the private key and other management operations such as key rotation. External keys can be managed by the IAM API, `gcloud` command-line tool, or the Service Accounts page in the Google Cloud Platform Console. You can create up to 10 service account keys per service account to facilitate key rotation.

Types of service accounts

User-managed service accounts

When you create a new Cloud project using GCP Console and if Compute Engine API is enabled for your project, a Compute Engine Service account is created for you by default. It is identifiable using the email:

`PROJECT_NUMBER-compute@developer.gserviceaccount.com`

If your project contains an App Engine application, the default App Engine service account is created in your project by default. It is identifiable using the email:

`PROJECT_ID@appspot.gserviceaccount.com`

If you create a service account in your project, you'll name the service account and it will be assigned an email with the following format:

`SERVICE_ACCOUNT_NAME@PROJECT_ID.iam.gserviceaccount.com`

You can create up to 100 service accounts per project (including the default Compute Engine service account and the App Engine service account) using the IAM API, the GCP console, or the `gcloud` command-line tool. These default service accounts and the service accounts you explicitly create are the user-managed service accounts.

Caution: The exact behavior of when the default service accounts are created and how they show up in your project might change in the future since they are designed to be used by Compute Engine and App Engine, so it is recommended that you don't rely on the existence of these default accounts for your use. It is recommended that you create additional service accounts explicitly using the IAM API, GCP Console or the `gcloud` command-line tool for your long-term use.

Google-managed service accounts

In addition to the user-managed service accounts, you might see some additional service accounts in your project's IAM policy or in GCP Console. These service accounts are created and owned by Google. These accounts represent different Google services and each account is automatically granted IAM roles to access your GCP project.

Google APIs service account

An example of a Google-managed service account is a Google API service account identifiable using the email:

```
PROJECT_NUMBER@cloudservices.gserviceaccount.com
```

This service account is designed specifically to run internal Google processes on your behalf and is not listed in the **Service Accounts** section of GCP Console. By default, the account is automatically granted the project editor role on the project and is listed in the **IAM** section of GCP Console. This service account is deleted only when the project is deleted. Google services rely on the account having access to your project, so you should not remove or change the service account's role on your project.

Service account permissions

In addition to being an identity, a service account is a resource which has IAM policies attached to it. These policies determine who can use the service account.

For instance, Alice can have the editor role on a service account and Bob can have viewer role on a service account. This is just like granting roles for any other GCP resource.

The default Compute Engine and App Engine service accounts are granted editor roles on the project when they are created so that the code executing in your App or VM instance has the necessary permissions. In this case, the service accounts are identities that are granted the editor role for a resource (project).

If you want to allow your automation to access a Cloud Storage bucket, you grant the service account (that your automation uses) the permissions to read the Cloud Storage bucket. In this case, the service account is the identity that you are granting permissions for another resource (the Cloud Storage bucket).

The Service Account User role

You can grant the `iam.serviceAccountUser` role at the project level for all service accounts in the project, or at the service account level.

- Granting the `iam.serviceAccountUser` role to a user for a project gives the user access to all service accounts in the project, including service accounts that may be created in the future.
- Granting the `iam.serviceAccountUser` role to a user for a specific service account gives a user access to the service account.

If you grant a user the `compute.instanceAdmin` role with the `iam.serviceAccountUser` role, they can create and manage Compute Engine instances that use a service account.

After you grant IAM roles to service accounts, you can assign the service account to one or more new virtual machine instances. For instructions on how to do this, see Setting up a new instance to run as a service account.

Users who are serviceAccountUsers can use the service account to indirectly access all the resources to which the service account has access. For example, a user who is a serviceAccountUser can start an instance using the service account. They can then use the instance to access anything the service account identity has access to. However, the serviceAccountUser role doesn’t allow a user to directly use the service account’s roles. Therefore, be cautious when granting the `iam.serviceAccountUser` role to a user.

Service accounts represent your service-level security. The security of the service is determined by the people who have IAM roles to manage and use the service accounts, and people who hold private external keys for those service accounts. Best practices to ensure security include the following:

- Use the IAM API to audit the service accounts, the keys, and the policies on those service accounts.
- If your service accounts don't need external keys, delete them.
- If users don't need permission to manage or use service accounts, then remove them from the IAM Policy.

To learn more about best practices, see Understanding service accounts.

The Service Account Token Creator role

This role enables impersonation of service accounts to create OAuth2 access tokens, sign blobs, or sign JWTs.

The Service Account Actor role

This role has been deprecated. If you need to run operations as the service account, use the Service Account User role. To effectively provide the same permissions as Service Account Actor, you should also grant Service Account Token Creator.

Access scopes

Access scopes are the legacy method of specifying permissions for your VM. Before the existence of IAM roles, access scopes were the only mechanism for granting permissions to service accounts. Although they are not the primary way of granting permissions now, you must still set access scopes when configuring an instance to run as a service account. For information on access scopes seeGoogle Compute Engine documentation

Short-Lived Service Account Credentials

You can create short-lived credentials that allow you to assume the identity of a GCP service account. These credentials can be used to authenticate calls to Google Cloud Platform APIs or other non-Google APIs.

The most common use case for these credentials is to temporarily delegate access to GCP resources across different projects, organizations, or accounts. For example, instead of providing an external caller with the permanent credentials of a highly-privileged service account, temporary emergency access can be granted instead. Alternatively, a designated service account with restricted permissions can be impersonated by an external caller without requiring a more highly-privileged service account's credentials.

For more information, see Creating Short-Lived Service Account Credentials.

Application Default Credentials

Application default credentials are a mechanism to make it easy to use service accounts when operating inside and outside GCP, as well as across multiple GCP projects. The most common use case is testing code on a local machine, and then moving to a development project in GCP, and then moving to a production project in GCP. Using Application Default Credentials ensures that the service account works seamlessly; that is, that it uses a locally-stored service account key when testing on your local machine but uses the project's default Compute Engine service account when running on Compute Engine. For more information, see Application Default Credentials.

Creating and Managing Service Accounts

When you create a new Cloud project, Google Cloud Platform (GCP) automatically creates one Compute Engine service account and one App Engine service account under that project. You can create up to 98 additional service accounts for your project to control access to your resources.

Required permissions

To allow a user to manage service accounts, grant one of the following roles:

- *Service Account User* (`roles/iam.serviceAccountUser`): Grants permissions to get, list, or impersonate a service account.
- *Service Account Admin* (`roles/iam.serviceAccountAdmin`): Includes *Service Account User*permissions and also grants permissions to create, update, delete, and set or get the Cloud IAM policy on a service account.

Cloud IAM primitive roles also contain permissions to manage service accounts. However, we recommend granting one of the predefined roles above to prevent unnecessary access to other GCP resources.

See the Service Account Roles topic for more information about roles related to service accounts.

Creating a service account

Creating a service account is similar to adding a member to your project, but the service account belongs to your applications rather than an individual end user.

In the examples below, **[SA-NAME]** is the name of the service account. This is a unique identifier; it will appear in the service account's email address, and you'll use it to update the service account with other APIs. It cannot be changed once created. **[SA-DISPLAY-NAME]** is a friendly name for the service account. **[PROJECT-ID]** is the ID of your Google Cloud Platform project.

To create a service account, at minimum the user must be granted the *Service Account Admin* role (`roles/iam.serviceAccountAdmin`) or the *Editor* primitive role (`roles/editor`).

1. Open the **Service Accounts** page in the GCP Console.
2. OPEN THE SERVICE ACCOUNTS PAGE
3. Click **Select a project**.
4. Select your project and click **Open**.

5. Click **Create Service Account**.
6. Enter a service account name, select a role you wish to grant to the service account, and then click **Save**.

After you create a service account, grant one or more roles to the service account so that it can act on your behalf.

Listing service accounts

When listing service accounts, you can specify parameters to limit the number of service accounts to include in the response. You can then use `ListServiceAccountsResponse.next_page_token` in a subsequent request to list the remaining service accounts.

Use this method to audit service accounts and keys, or to build custom tools for managing service accounts.

To list service accounts, at a minimum the user must be granted the *Service Account User* role (`roles/iam.serviceAccountUser`) or the *Viewer* primitive role (`roles/viewer`).

1. Open the **Service Accounts** page in the GCP Console.
2. OPEN THE SERVICE ACCOUNTS PAGE
3. Click **Select a project**.
4. Select your project and click **Open**. All service accounts are listed in the Service Accounts page.

Renaming a service account

The display name of a service account is commonly used to capture additional information about the service account, such as the purpose of the service account or a contact person for the account.

To rename a service account, at a minimum the user must be granted the *Service Account Admin* role (`roles/iam.serviceAccountAdmin`) or the *Editor* primitive role (`roles/editor`).

1. Open the **Service Accounts** page in the GCP Console.

2. OPEN THE SERVICE ACCOUNTS PAGE
3. Click **Select a project**.
4. Select your project and click **Open**.
5. Look for the service account you wish to rename, click the **More** more_vert button in that row, and then click **Edit**.
6. Enter the new name and click **Save**.

Deleting a service account

When you delete a service account, applications will no longer have access to Google Cloud Platform resources through that service account. If you delete the default App Engine and Compute Engine service accounts, the instances will no longer have access to resources in the project.

Delete with caution; make sure your critical applications no longer use a service account before deleting it. Additionally, role bindings for a deleted service account are not immediately removed; they are automatically purged from the system after a maximum of 60 days.

To delete a service account, at a minimum the user must be granted the *Service Account Admin* role (`roles/iam.serviceAccountAdmin`) or the *Editor* primitive role (`roles/editor`).

1. Open the **Service Accounts** page in the GCP Console.
2. OPEN THE SERVICE ACCOUNTS PAGE
3. Click **Select a project**.
4. Select your project and click **Open**.
5. Select the service account(s) you wish to delete, and then click **Delete.**

After deleting a service account, avoid creating a new service account with the same name. This can result in unexpected behavior. For more information, see Deleting and recreating service accounts

——-

Understanding Service Accounts

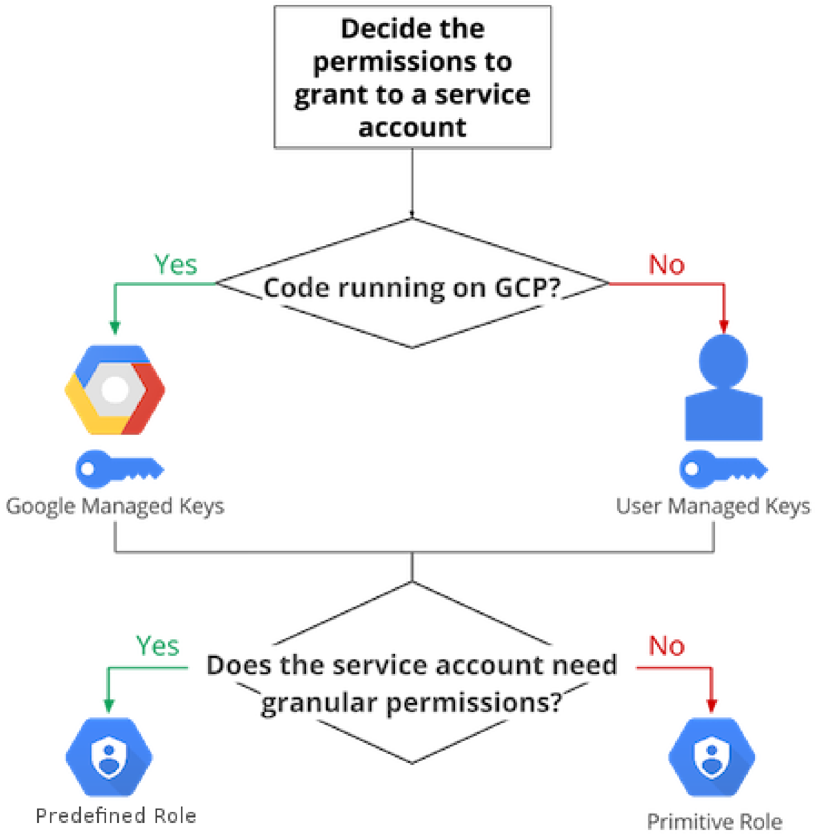
Background

A service account is a special type of Google account that belongs to your application or a virtual machine (VM), instead of to an individual end user. Your application assumes the identity of the service account to call Google APIs so that the users aren't directly involved. A service account can have zero or more pairs of service account keys, which are used to authenticate to Google.

Once you decide that you need a service account, you can ask yourself the following questions to understand how you're going to use the service account:

- What resources can the service account access?
- What permissions does the service account need?
- Where will the code that assumes the identity of the service account be running: on Google Cloud Platform or on-premises?

Use the following flowchart to figure out the responses to the above questions:


Decide the permissions to grant to a service account
Code running on GCP?
Yes
No
Google Managed Keys
User Managed Keys
Does the service account need granular permissions?
Yes
No
Predefined Role
Primitive Role

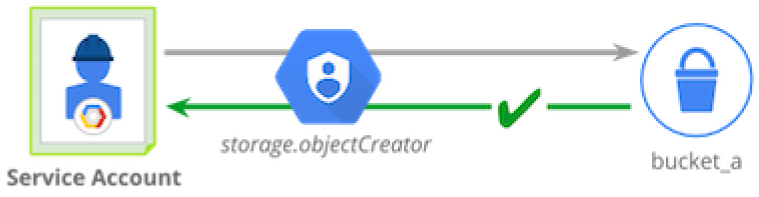
One of the features of IAM service accounts is that you can treat it both as a resource and as an identity.

When treating the service account as an identity, you can grant a role to a service account, enabling it to access a resource (such as a project).

When treating a service account as a resource, you can grant permission to a user to access that service account. You can grant the Owner, Editor, Viewer, or Service Account User role to a user to access the service account.

Granting access to service accounts

Granting access to a service account to access a resource is similar to granting access to any other identity. For example, if you have an application running on Google Compute Engine and you want the application to *only* have access to create objects in Google Cloud Storage. You can create a service account for the application and grant it the Storage Object Creator role. The following diagram illustrates this example:



Learn about Granting roles to service accounts.

Acting as a service account

Let’s say that you have a long-running job that your employees have permissions to start. You don’t want that job to be terminated when the employee who last started that job leaves the company.

The way you would solve this problem is by creating a service account to start and stop the job. You can do this using the following steps:

1. Create a service account.
2. Grant the Service Account User (iam.serviceAccountUser) role for the service account to your employees who need permission to start the job. In this scenario, the service is the resource.
3. Grant the Compute Instance Admin (roles/compute.instanceAdmin.v1) role to the same employees.
4. Now, employees can create Compute Engine instances that run that service account, connect to them, and use the service account to start the job. For example:

-
  ```
  gcloud compute instances create my-instance --scopes=cloud-platform \
  --service-account=my-service-account@test9q.iam.gserviceaccount.com \
  --zone=us-central1-a
  ```

For more information, see The serviceAccountUser role.

Note: Users with the Service Account User and Compute Instance Admin roles can indirectly access all resources the service account has access to, as well as create, modify, and delete Compute Engine instances. Therefore, be cautious when granting these roles to users.

Migrating data to Google Cloud Platform

Let’s say that you have some data processing that happens on another cloud provider and you want to transfer the processed data to Google Cloud Platform. You can use a service account from the virtual machines on the external cloud to push the data to Google Cloud Platform. To do this, you must create and download a service account key when you create the service account and then use that key from the external process to call the Cloud Platform APIs.

Keeping 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.

Deleting and recreating service accounts

It is possible to delete a service account and then create a new service account with the same name. If you reuse the name of a deleted service account, it may result in unexpected behavior.

When you delete a service account, its role bindings are not immediately deleted. 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 Cloud 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.

To avoid confusion, we suggest using unique service account names. If this is not possible, you can grant a role to the new service account by:

1. Explicitly removing any bindings granting that role to the old service account.
2. Re-granting those roles to the new service account.

You must remove the role bindings first before re-adding them. Simply granting the role again will silently fail by granting the role to the old, deleted service account.

Granting minimum permissions to service accounts

You should only grant the service account the minimum set of permissions required to achieve their goal. Learn about Granting roles to a service account for specific resources.

When granting permissions to users to access a service account, keep in mind that the user can access all the resources for which the service account has permissions. Therefore it's important to configure permissions of your service accounts carefully; that is, be strict about who on your team can act as a service account.

Users with IAM roles to update the App Engine and Compute Engine instances (such as App Engine Deployer or Compute Instance Admin) can effectively run code as the service accounts used to run these instances, and indirectly gain access to all the resources for which the service accounts have access. Similarly, SSH access to a Compute Engine instance may also provide the ability to execute code as that instance.

Managing service account keys

There are two types of service account keys:

- **GCP-managed keys**. These keys are used by Cloud Platform services such as App Engine and Compute Engine. They cannot be downloaded, and are automatically rotated and used for signing for a maximum of two weeks. The rotation process is probabilistic; usage of the new key will gradually ramp up and down over the key's lifetime. We recommend caching the public key set for a service account for at most 24 hours to ensure that you always have access to the current key set.

- **User-managed keys**. These keys are created, downloadable, and managed by users. They expire 10 years from creation.

For user-managed keys, you need to make sure that you have processes in place to address key management requirements such as:

- Key storage
- Key distribution
- Key revocation
- Key rotation
- Protecting the keys from unauthorized users
- Key recovery

Anyone who has access to the keys will be able to access resources through the service account. Always discourage developers from checking keys into the source code or leaving them in Downloads directory.

To enhance the security of keys, follow the guidance below:

- Use the IAM service account API to automatically rotate your service account keys. You can rotate a key by creating a new key, switching applications to use the new key and then deleting the old key. Use the `serviceAccount.keys.create()` and `serviceAccount.keys.delete()` methods together to automate the rotation. The GCP-managed keys are rotated approximately once a week.
- Use the `serviceAccount.keys.list()` method to audit service accounts and keys.

Using service accounts with Compute Engine

Compute Engine instances need to run as service accounts to have access to other Cloud Platform resources. To make sure that your

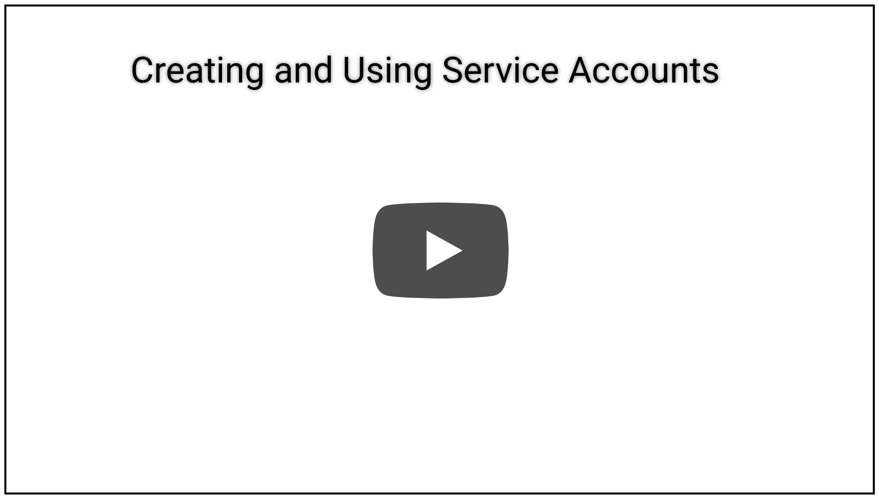
Compute Engine instances are secure, consider the following:

- You can create VMs in the same project with different service accounts. To change the service account of a VM after it's created, use the `instances.setServiceAccount` method.
- You can grant IAM roles to service accounts to define what they can access. In many cases, you won't need to rely on scopes anymore. This gives you the advantage of being able to modify permissions of a VM's service account without recreating the instance.
- Since instances depend on their service accounts to have access to Cloud Platform resources, avoid deleting service accounts when they are still used by running instances. If you delete the service accounts, the instances may start failing their operations.

Best practices

- Restrict who can act as service accounts. Users who are Service Account Users for a service account can indirectly access all the resources the service account has access to. Therefore, be cautious when granting the serviceAccountUser role to a user.
- Grant the service account only the minimum set of permissions required to achieve their goal. Learn about Granting roles to a service account for specific resources.
- Create service accounts for each service with only the permissions required for that service.
- Use the display name of a service account to keep track of the service accounts. When you create a service account, populate its display name with the purpose of the service account.
- Define a naming convention for your service accounts.
- Implement processes to automate the rotation of user-managed service account keys.
- Take advantage of the IAM service account API to implement key rotation.

- Audit service accounts and keys using either the `serviceAccount.keys.list()` method or the Logs Viewer page in the console.
- Do not delete service accounts that are in use by running instances on Google App Engine or Google Compute Engine.



++++++++++++++++++++++++++++++++++++
++

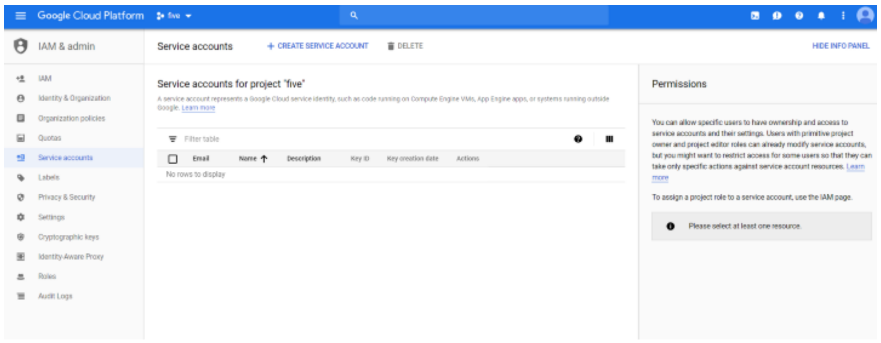
Managing service accounts with limited scopes

Giving access to the service account is a very crucial task as wrong access can lead to various security threats.

best practice is to give any that access that is required to fulfill the job that needs to be done.

let us see how to do it.

First go to IAM &services tab, then to Service account



Service accounts for project "five"

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. Learn more

Email	Name ↑	Description	Key ID	Key creation date	Actions
No rows to display					

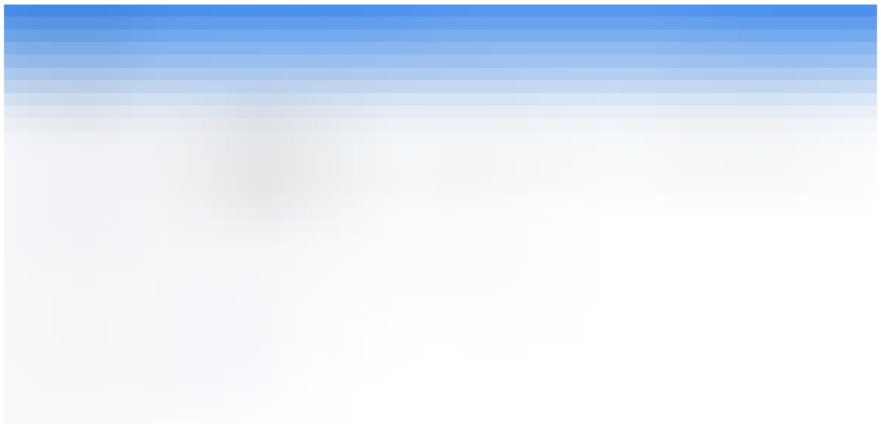
You can allow specific users to have ownership and access to service accounts and their settings. Users with primitive project owner and project editor roles can already modify service accounts, but you might want to restrict access for some users so that they can take only specific actions against service account resources. Learn more

To assign a project role to a service account, use the IAM page.

Please select at least one resource.

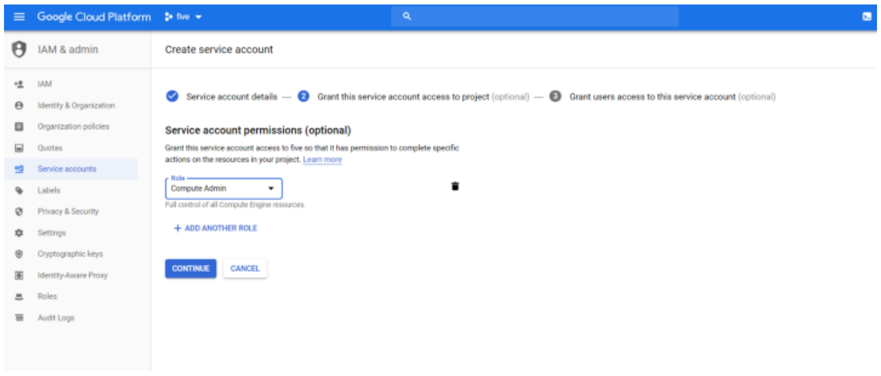
Service account

Now click on create a service account



create service account

Then you will see the permissions page, this is a place where you can limit the access for the service account.



Service account details — 2 Grant this service account access to project (optional) — 3 Grant users access to this service account (optional)

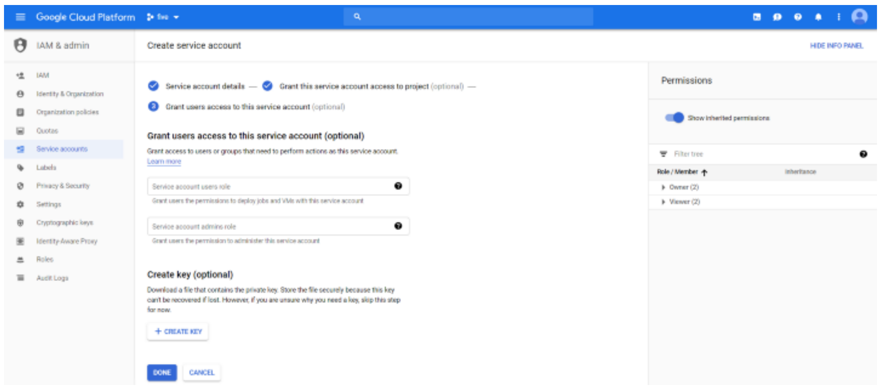
Service account permissions (optional)

Grant this service account access to five so that it has permission to complete specific actions on the resources in your project. Learn more

Role: Compute Admin

Full control of all Compute Engine resources.

Then you will see the page where you can assign this service account to users.



Create service account

Service account details — Grant this service account access to project (optional) — Grant users access to this service account (optional)

Grant users access to this service account (optional)

Grant access to users or groups that need to perform actions as this service account. Learn more

Service account users role
Grant users the permissions to deploy jobs and VMs with this service account

Service account admins role
Grant users the permission to administer this service account

Create key (optional)

Download a file that contains the private key. Store the file securely because this key can't be recovered if lost. However, if you are unsure why you need a key, skip this step for now.

Permissions

Show inherited permissions

Filter tree

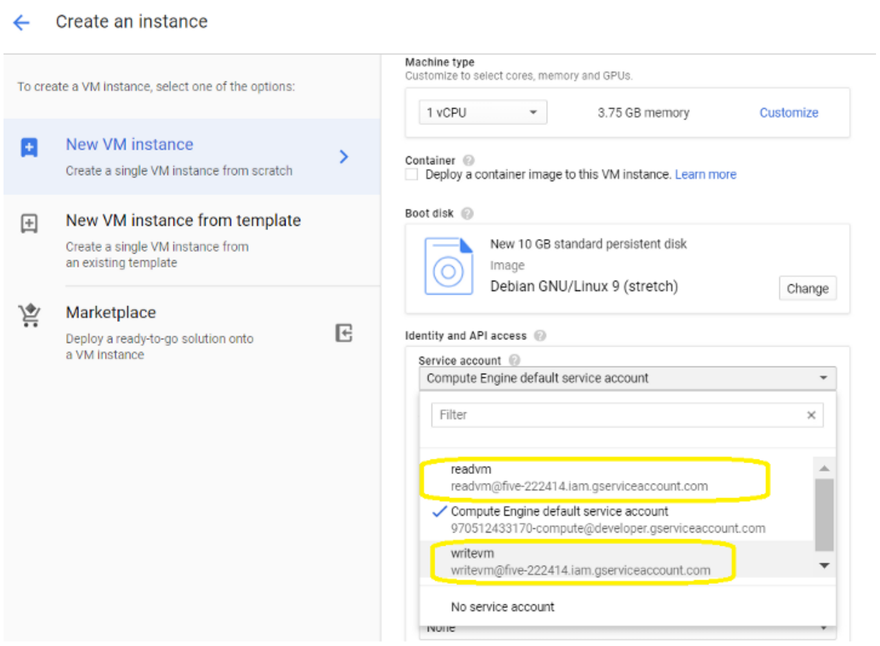
Role / Member	Inheritance
Owner (2)	
Viewer (2)	

After creating the service account we can modify more from the dashboard.

———

Assigning a service account to VM instances

You can easily assign service account to the VM instance while building new VM in IAM section as below. For that, you have to create a service account before making VM.



To create a VM instance, select one of the options:

New VM instance
Create a single VM instance from scratch

New VM instance from template
Create a single VM instance from an existing template

Marketplace
Deploy a ready-to-go solution onto a VM instance

Machine type
Customize to select cores, memory and GPUs.
1 vCPU 3.75 GB memory

Container
Deploy a container image to this VM instance. Learn more

Boot disk
New 10 GB standard persistent disk
Image
Debian GNU/Linux 9 (stretch)

Identity and API access
Service account
Compute Engine default service account

readvm
readvm@five-222414.iam.gserviceaccount.com

✓ Compute Engine default service account
970512433170-compute@developer.gserviceaccount.com

writevm
writevm@five-222414.iam.gserviceaccount.com

No service account

None

——
— -

Granting access to a service account in another project

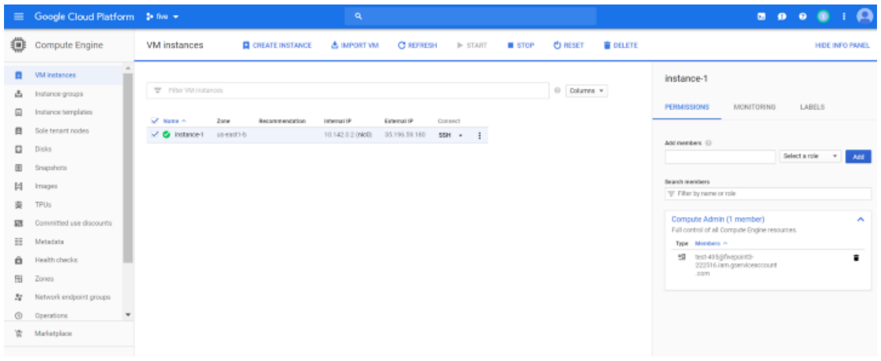
If you have two projects and want to grant service account access to another project then you can use the service account in this case.

First, create a service account like above.

Now go to the second project and then to VM where you want to access.

Select the VM and add service account in permission page at the right-hand side.

Apply



Name	Zone	Recommendation	Internal IP	External IP	Connect
instance-1	us-east1-b		10.142.0.2 (nic0)	35.196.59.180	SSH

instance-1

Compute Admin (1 member)
Full control of all Compute Engine resources.

Type	Members
	test-495@fivepoint3-222516.iam.gserviceaccount.com

Add service account outside the project

——
——— -

In this way, you can perform tasks on the service account.

Google Cloud: IAM and Organization Node