Handling Authentication and Authorization

Developing Applications for GCP

CLOUD IAM, FIREBASE SDK

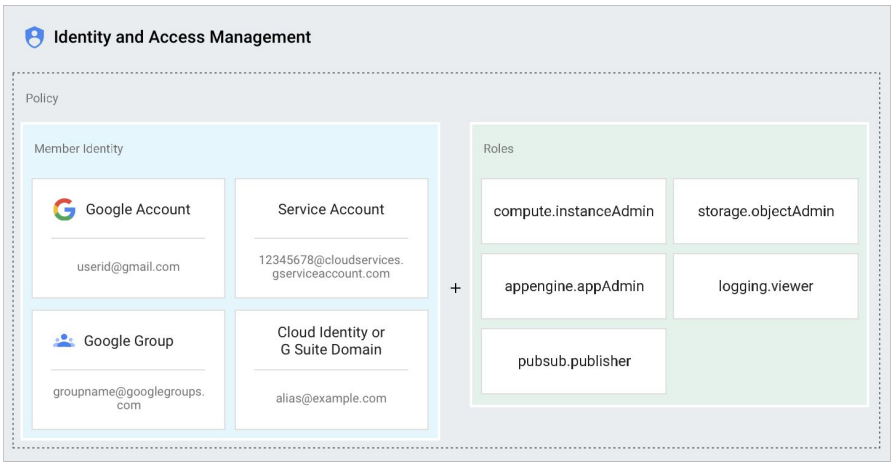
ADDING USER AUTHENTICATION TO YOUR APPLICATION

Google Cloud

Version 2.0
Last modified: 2017-09-25

Explain how to use Cloud IAM roles and service accounts

Cloud Identity and Access Management



Google Cloud

Cloud IAM lets you manage access control by defining who (members) has what access (role) for which resource. You can grant more granular access to GCP resources using the security principle of least privilege—only grant necessary access to resources.

For more information, see: https://cloud.google.com/iam/docs/overview

Specify *who* has access with IAM members

Types of IAM members:

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

Google Cloud

You can specify who has access to your resources with IAM members. Members can be of the following types: Google account, service account, Google group, G Suite domain, or Cloud Identity domain.

A Google cloud account represents a developer, administrator, or person who interacts with Google Cloud Platform. An email address that is associated with a Google account, such as a gmail.com address, can be an identity. For more information on Google account creation, see: https://accounts.google.com/signup

A service account belongs to an application instead of to an individual end user.

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. Google groups are a convenient way to apply an access policy to a collection of users. Google groups don't have login credentials, so you cannot use them to establish identity to make a request to access a resource. For more information on Google groups, see: https://groups.google.com/forum/#!overview

A G Suite domain represents a virtual group of all the members in an organization. Suite customers can associate their email accounts with an Internet domain name. When you do this, each email account takes the form username@yourdomain.com. You can specify an identity by using any Internet domain name that is associated with a G Suite account. Like groups, domains cannot be used to establish identity, but they enable convenient permission management.

A Cloud identity domain represents a virtual group of all members in an organization, but doesn’t provide access to G Suite applications and features.

Specify *what* resources the members have access to

- Grant access to users for specific GCP resources
- Resources include:
- GCP projects
- Compute Engine instances
- Cloud Storage buckets
- Pub/Sub topics

Google Cloud

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

Specify what *operations* are allowed on resources

Permissions are represented with the following syntax:

```
<service>.<resource>.<verb>
```

Examples:

```
<pubsub>.<subscriptions>.<consume>
```

```
<storage>.<objects>.<list>
```

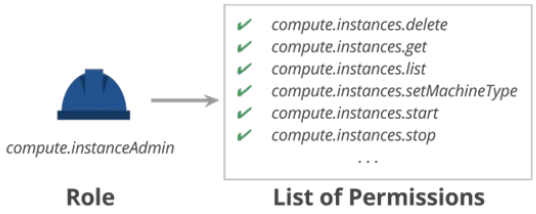
```
<compute>.<disktypes>.<list>
```


Google Cloud

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 Cloud Platform 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.

Assign permissions using roles



There are two kinds of IAM roles:

- Primitive
- Predefined

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. With Cloud IAM, a Cloud API method requires the identity making the API request to have the appropriate permissions to use the resource. You can grant permissions by granting roles to a user, a group, or a service account.

There are three kinds of roles: primitive and predefined.

https://cloud.google.com/iam/docs/understanding-roles

Apply primitive roles at the project level

Role Name	Role Title	Permissions
`roles/viewer`	Viewer	Read-only actions that preserve state.
`roles/editor`	Editor	Viewer permissions plus actions that modify state.
`roles/owner`	Owner	Editor permissions plus ability to: ● Manage access control for a project and all its resources. ● Set up billing for a project.

The primitive roles can be applied at the project level using Cloud Platform Console, the API and the gcloud command-line tool.

Note: Granting the owner role at a resource level (such as a pubsub topic) does not grant the owner role on the parent project. The owner role does not contain any permission for the Organization resource. Granting the owner role at the organization level does not allow you to update the organization's metadata, but it allows you to modify projects under that organization.

Apply predefined roles for granular access to GCP resources

Predefined roles give granular access to specific GCP resources.
You can grant multiple roles to the same user.

Role Name	Role Title	Description	Resource Type
roles/bigtable.admin	Cloud Bigtable Admin	Administers all instances within a project, including the data stored in tables. Can create new instances. Intended for project administrators.	Organization Project Instance
roles/bigtable.user	Cloud Bigtable User	Provides read-write access to the data stored in tables. Intended for application developers or service accounts.	Organization Project Instance
roles/bigtable.reader	Cloud Bigtable Reader	Provides read-only access to the data stored in tables. Intended for data scientists, dashboard generators, and other data-analysis scenarios.	Organization Project Instance

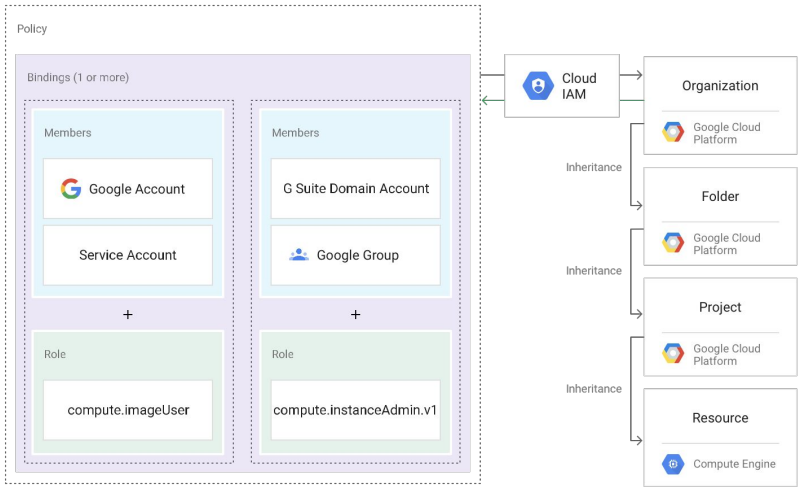
Google Cloud

Cloud IAM provides predefined roles that give granular access to specific Google Cloud Platform resources and prevent unwanted access to other resources.

You can grant multiple roles to the same user. For example, the same user can have Network Admin and Log Viewer roles on a project and also have a Publisher role for a Pub/Sub topic within that project.

As an example, roles for Cloud Bigtable are provided. For more information on predefined roles, see:
https://cloud.google.com/iam/docs/understanding-roles#predefined_roles

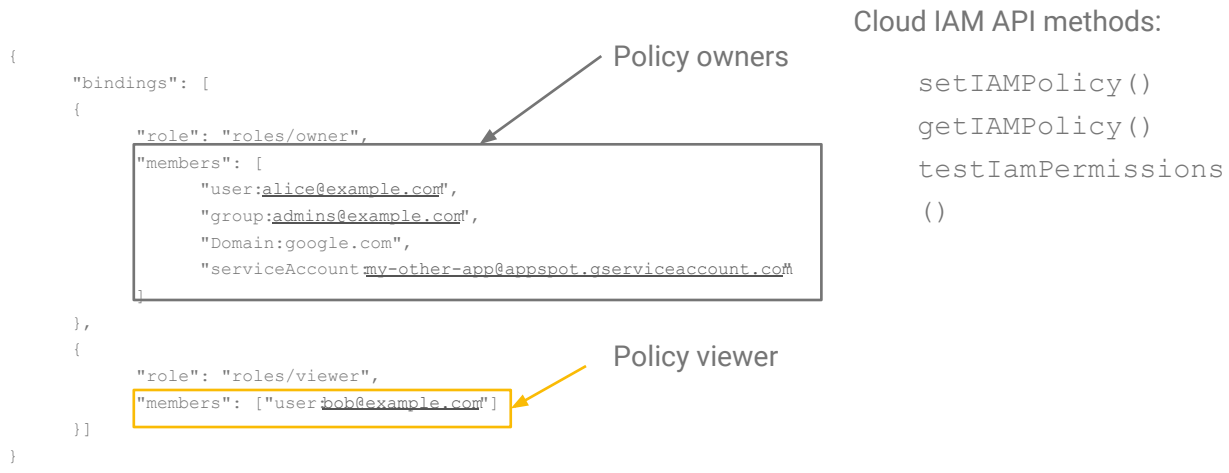
Define *who* has *what* type of access using policies



Google Cloud

A policy is attached to a resource and is used to enforce access control whenever that resource is accessed.

Cloud IAM Policy example



Google Cloud

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

The Cloud IAM API methods available 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.

Cloud Platform 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 children of projects. Each resource has exactly one parent.

Use service accounts to authenticate your applications when invoking Google APIs

Service accounts:

- Belong to your application or VM.
- Are used by your application to call the Google API or service so users aren't directly involved.
- Are identified by their unique email addresses.
- Are associated with a key pair.
- Can have up to 10 keys associated with them to facilitate key rotation (done daily by Google).
- Are supported by all GCP APIs.
- Enable authentication and authorization: you can assign specific IAM roles to a service account.

Google Cloud

Service accounts belong to your application or VM instance instead of to an individual user. They are used by your application to call the Google API or service so that users aren't directly involved in the authentication process. Each service account is identified by its unique email address. Service accounts are associated with a key pair and can have up to 10 key pairs associated with them, to facilitate key rotation, which is managed by Google and done daily. Service accounts enable authentication and authorization because you can assign specific IAM roles to a service account.

For more information, see:
Service Accounts: https://cloud.google.com/iam/docs/service-accounts
Grant a Service Account an IAM Role:
https://cloud.google.com/compute/docs/access/create-enable-service-accounts-for-instances#createanewserviceaccount

Use external keys for use from outside GCP

External keys:

- Can be created for use from outside GCP.
- Require that you be responsible for security of the private key and other management operations such as key rotation.
- Are manageable through the:
  - IAM API.
  - gcloud command-line tool.
  - Service Accounts page in the GCP Console.

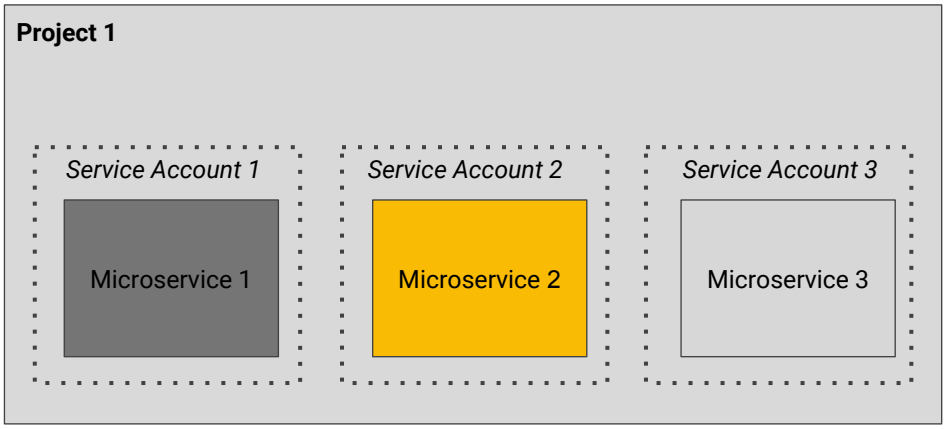
Google Cloud

You can create external keys to be used from outside GCP. They require that you manage the security of the private key and perform management operations such as key rotation. External keys are manageable through the IAM API, gcloud, and the GCP console.

You can optionally use API keys to call certain APIs that don't need to access private user data. API keys are useful in clients such as browser and mobile applications that don't have a backend server. The API key is used to track API requests associated with your project for quota and billing.

For more information on API keys, see:
https://cloud.google.com/docs/authentication/api-keys

You can define many service accounts in a project



Google Cloud

You can define many non-default service accounts in a project. The diagram shows a use case to manage many microservices in a single Cloud Project by letting each microservice be represented by its own service account. Granularity is employed by specifying which service accounts are allowed to call which other services.

Follow these steps to use a service account in your application

1. Create the service account via the console.
2. Generate and download your credentials file.
3. Set an environment variable to provide credentials to your application.
4. Authenticate in your code with the default credentials.

Linux or OS X:

```
export GOOGLE_APPLICATION_CREDENTIALS=<path_to_service_account_file>
```

Windows:

```
set GOOGLE_APPLICATION_CREDENTIALS=<path_to_service_account_file>
```

```
def implicit():
    from google.cloud import storage

    storage_client = storage.Client()

    # Make an authenticated API request
    buckets = list(storage_client.list_buckets())
    print(buckets)
```

If you don't specify credentials when constructing the client, the client library will look for credentials in the environment.

Google Cloud

To use a service account in your application, first create the service account via the console. You then generate and download your credentials file, and then you can set an environment variable with the path to your downloaded credentials. The example shows commands to set the environment variable in both Linux/OS X and Windows. You can then make an authenticated API request in your code. The example code is in Python and shows that if you don't explicitly specify credentials when constructing a client, the client library will look for credentials in the environment.

For more information, see:
https://cloud.google.com/docs/authentication/getting-started

Use Application Default Credentials (ADC) to authenticate between applications

ADC checks for credentials in the following order:

1. Checks for `GOOGLE_APPLICATION_CREDENTIALS` environment variable.
2. Checks for default service accounts.
3. If 1 and 2 aren't found, an error is thrown.

```
def implicit():

    from google.cloud import storage

    storage_client = storage.Client()

    buckets = list(storage_client.list_buckets())

    print(buckets)
```

Make an authenticated API request.

If you don't specify credentials when constructing the client, the client library will look for credentials in the environment.

Google Cloud

Google uses credentials to identify your application for quota and billing and to authorize access to GCP APIs, resources, and features.

GCP client libraries use Application Default Credentials (ADC) to find your application's credentials. Credentials are checked for in the following order: ADC will check for the GOOGLE_APPLICATION_CREDENTIALS environment variable. If the environment variable is set, ADC will use the service account file that the variable points to. If the environment variable isn't set, ADC will use the default service account that Compute Engine, Container Engine, App Engine, and Cloud Functions provide (assuming that your application runs on those services). If neither the environment variable nor the default service accounts can be found, an error will occur.

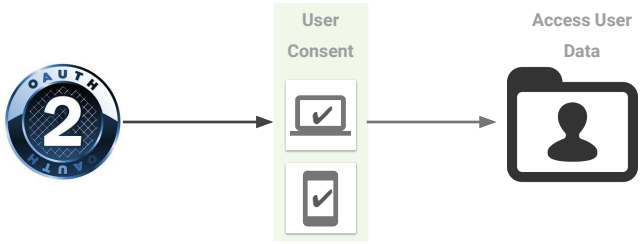
The example demonstrates using Python to implicitly find the credentials—assuming that either the environment variable is set or the application is running on Compute Engine, Container Engine, App Engine, or Cloud Functions.

For more information, including how to obtain and provide service credentials manually, see: https://cloud.google.com/docs/authentication/production

Use OAuth 2.0 to access resources on behalf of a user

Use cases include:

- Your application needs to access Google BigQuery datasets that belong to users.
- Your application needs to authenticate as a user to create projects on their behalf.



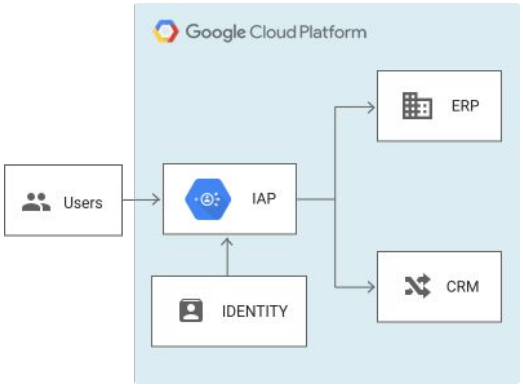
Google Cloud

It is generally best to use a service account for authentication to a GCP API. In some cases, you might want to access resource on behalf of a user. Use cases where you might want to access resources on behalf of a user include your application needing access to Google BigQuery datasets that belong to your application users, and your application needing to authenticate as a user to create projects on their behalf. You can use the OAuth 2.0 protocol to access resources on behalf of a user. Your application will request access to the resources, the user will be prompted for consent, and if consent is provided, the application can request credentials from an authorization server. The application can then use those credentials to access resources on behalf of the user.

For more information, see:
OAuth 2.0: https://developers.google.com/identity/protocols/OAuth2
Authenticating as an End User: https://cloud.google.com/docs/authentication/end-user

Cloud Identity-Aware Proxy (IAP)

- Controls access to your cloud applications running on GCP
- Verifies a user's identity
- Determines *whether* that user should be allowed to access the application



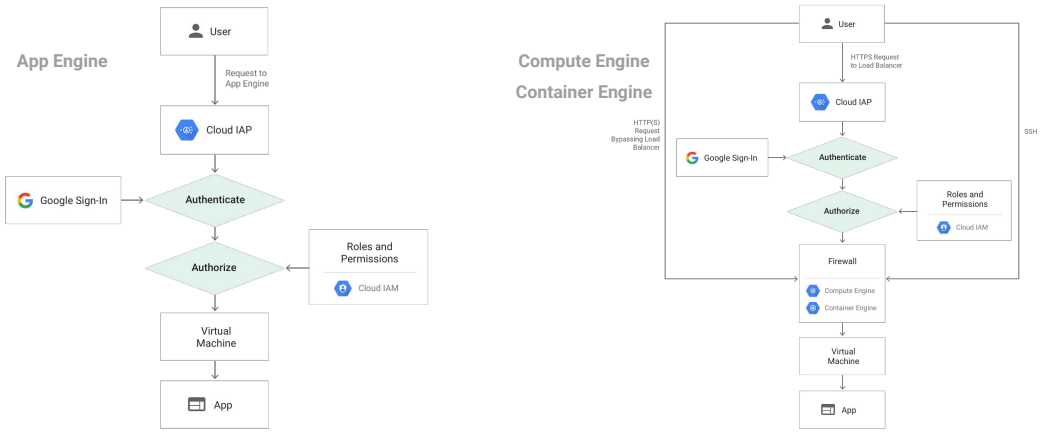
Google Cloud

Cloud Identity-Aware Proxy (IAP) controls access to your cloud applications running on GCP. IAP verifies a user's identity and determines whether that user should be allowed to access the application.

IAP allows you to establish a central authorization layer for applications accessed by HTTPS. IAP lets you adopt an application-level access control model instead of relying on network-level firewalls.

For more information, see: https://cloud.google.com/iap/docs/

How Cloud IAP works



Google Cloud

Applications and resources protected by Cloud IAP can only be accessed through the proxy by users and groups with the correct Cloud IAM role. When you grant a user access to an application or resource by Cloud IAP, they're subject to the fine-grained access controls implemented by the product in use without requiring a VPN. Cloud IAP performs authentication and authorization checks when a user tries to access a Cloud IAP-secured resource.

For more information, see: https://cloud.google.com/iap/docs/concepts-overview

Follow these precautions when using Cloud IAP

- Configure your firewall and load balancer to protect against traffic that doesn't come from the serving infrastructure.
- Use signed headers or the App Engine standard environment Users API.

Google Cloud

To ensure the security of your applications, you should take the following precautions when using Cloud IAP:

- Configure your firewall and load balancer to protect against traffic that doesn't come from the serving infrastructure.
- Use signed headers or the App Engine standard environment Users API.

For more information, see:
Cloud IAP Best Practices: https://cloud.google.com/iap/docs/concepts-best-practices
Signed Headers: https://cloud.google.com/iap/docs/signed-headers-howto
Users API: https://cloud.google.com/appengine/docs/standard/#users

Use the Firebase SDK to authenticate application users

1. Get authentication credentials from the user.
2. Pass the credentials to the Firebase Authentication SDK.
3. Firebase backend services verify credentials and return a response to the client.
4. After successful sign-in, you can:
   a. Access the user's basic profile.
   b. Control the user's access to data stored in other Firebase products.
   c. Use the provided authentication token to verify the identity of users in your own backend services.

Google Cloud

Firebase authentication is Google's federated authentication supporting end user sign-in at the client app using third-party credentials, for example, from Google or Facebook.

To sign a user into your app, you first get authentication credentials from the user. These credentials can be the user's email address and password or an OAuth token from a federated identity provider. Then, you pass these credentials to the Firebase Authentication SDK. Firebase's backend services will then verify those credentials and return a response to the client.

After a successful sign-in, you can access the user's basic profile information, and you can control the user's access to data stored in other Firebase products. You can also use the provided authentication token to verify the identity of users in your own backend services.

For more information, see: https://firebase.google.com/docs/auth/#how_does_it_work

Firebase Pricing

	Spark Plan	Flame Plan	Blaze Plan
Price	FREE	$25/month	Pay as you go
Authentication	FREE	FREE	FREE
Phone Auth	10K authentications a month	10K authentications a month	$0.01/verification (US, Canada, India) $0.06/verification (All other countries)

Google Cloud

Authentication, except for phone authentication, is free regardless of your Firebase pricing plan. Phone authentication allows for 10,000 authentications a month on the Spark and Flame plans and is $0.01 per verification in the US, Canada, and India and $0.06 per verification in all other countries on the Blaze plan.

For more information, see: https://firebase.google.com/pricing/

Remember the following best practices when using Cloud IAM

- Follow the principle of least privilege.
- Rotate service account keys.
- Manage user-managed service account keys.
- Don’t check in service account keys.
- Use Cloud Audit Logging and export logs to Cloud Storage.
- Set organization-level IAM policies.
- Grant roles to a Google group when possible.

Google Cloud

Remember the following best practices when using Cloud IAM. For a detailed list of best practices, see: https://cloud.google.com/iam/docs/using-iam-securely

- Follow the principle of least privilege: use predefined roles (more granular access) when possible. Grant predefined roles to identities when possible, so you only give the least amount of access necessary to access your resources. Grant roles at the smallest scope needed. Restrict who can create and manage service accounts in your project.

- Rotate service account keys regularly and implement processes to manage user-managed service account keys. Don’t check in service account keys to source code.

- Use Cloud Audit Logging to regularly audit changes to your IAM policy and export logs to Cloud Storage.

- Set organization-level IAM policies to grant access to all projects in your organization and grant roles to a Google group instead of to individual users when possible.