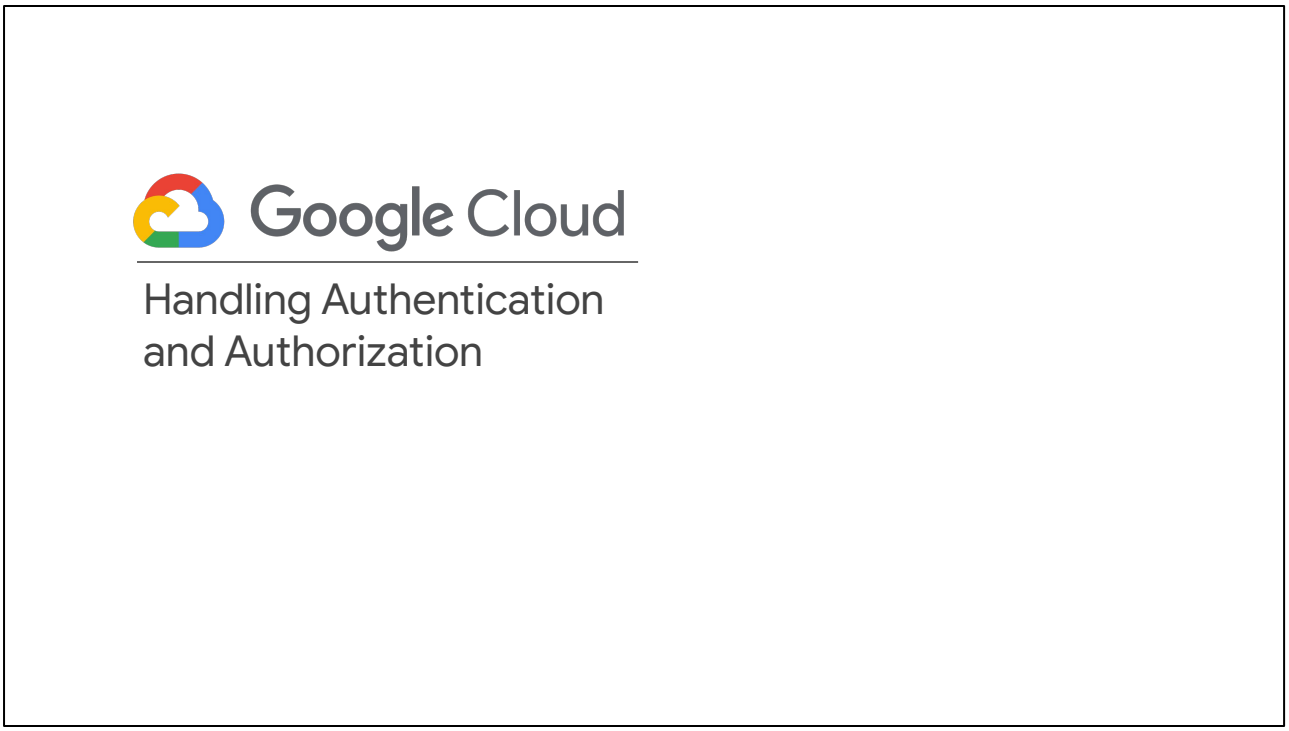


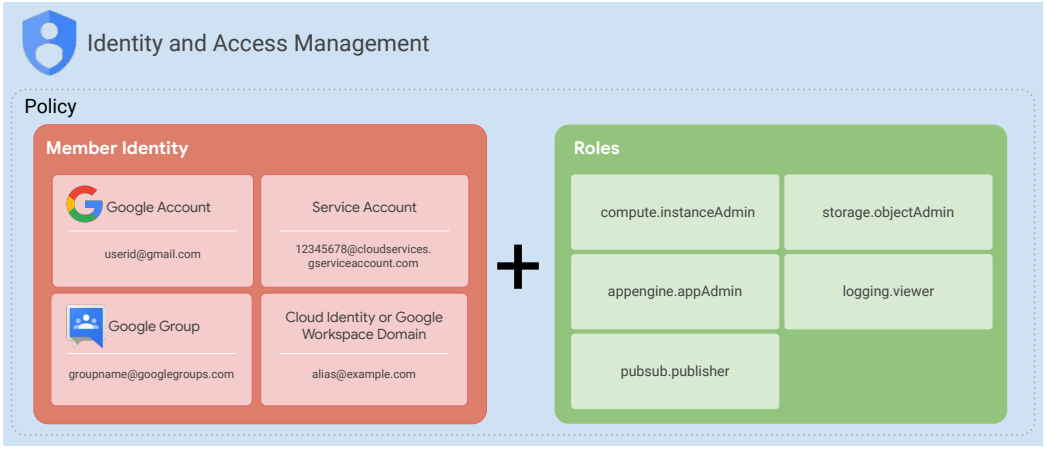
It can be somewhat daunting to implement user authentication and authorization from scratch in your application. With Cloud Identity and Access Management, or Cloud IAM for short, and Identity Platform authentication, you are now free from the burden of building such a system.

In the module, Handling Authentication and Authorization, you will learn how to create IAM members, specify what resources they have access to, and what operations they can perform on these resources.

You will learn how to use service accounts to allow your application to authenticate and authorize itself to invoke Google Cloud APIs. We'll discuss how you can use Identity-Aware Proxy to control access to your application.

The Firebase SDK makes it really easy to implement federated identity management. You'll learn how to use the Firebase SDK to validate a user against credentials that are stored in Identity Platform, Google Cloud's enterprise-grade identity and access management platform.

Cloud Identity and Access Management



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

Specify *who* has access with IAM members

Types of IAM members:

- Google account
- Service account
- Google group
- Google Workspace domain
- Cloud Identity domain

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**, **Google Workspace domain**, or **Cloud Identity domain**.

Specify *who* has access with IAM members

Types of IAM members:

- Google account
- Service account
- Google group
- Google Workspace domain
- Cloud Identity domain

A Google cloud account represents a developer, administrator, or person who interacts with Google Cloud. An email address that is associated with a Google account, such as a gmail.com address, can be an identity.

Specify *who* has access with IAM members

Types of IAM members:

- Google account
- Service account
- Google group
- Google Workspace domain
- Cloud Identity domain

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

Specify *who* has access with IAM members

Types of IAM members:

- Google account
- Service account
- Google group
- Google Workspace domain
- Cloud Identity domain

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.

Specify *who* has access with IAM members

Types of IAM members:

- Google account
- Service account
- Google group
- Google Workspace domain
- Cloud Identity domain

A Google Workspace domain represents a virtual group of all the members in an organization. Google Workspace 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 Google Workspace account. Like groups, domains cannot be used to establish identity, but they enable convenient permission management.

Specify *who* has access with IAM members

Types of IAM members:

- Google account
- Service account
- Google group
- Google Workspace domain
- Cloud Identity domain

A Cloud Identity domain represents a virtual group of all members in an organization, but doesn't provide access to Google Workspace applications and features.

Specify *what* resources the members have access to

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

You can grant access to users for a Google Cloud 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>
```

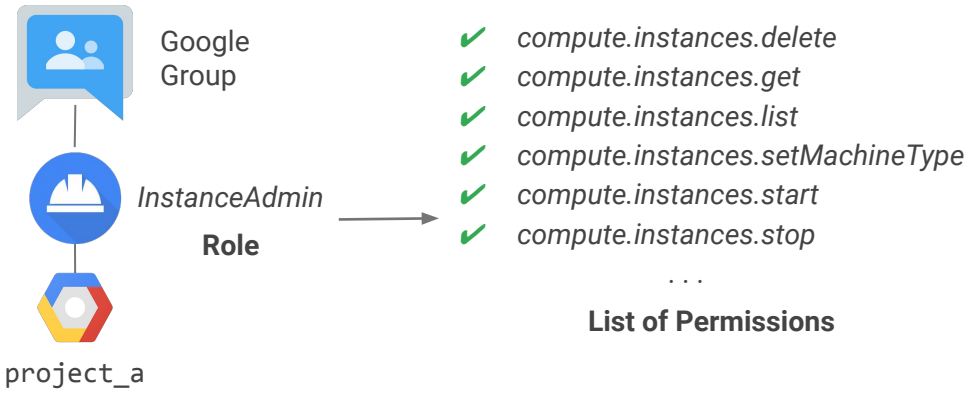
Examples:

```
pubsub.subscriptions.consume

storage.objects.list

compute.disktypes.list
```

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 Google Cloud 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: Basic, predefined, or custom



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: basic, predefined and custom.

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

Apply basic 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 basic roles can be applied at the project level using the Cloud 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.

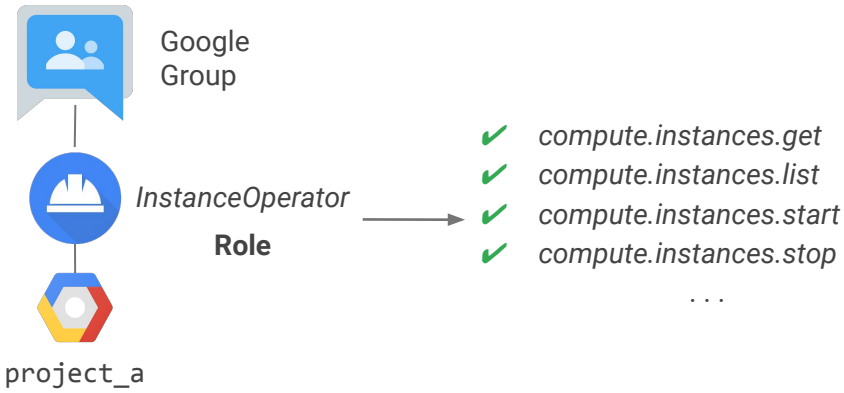
Apply predefined roles for granular access to Google Cloud resources

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

Cloud IAM provides predefined roles that give granular access to specific Google Cloud 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.

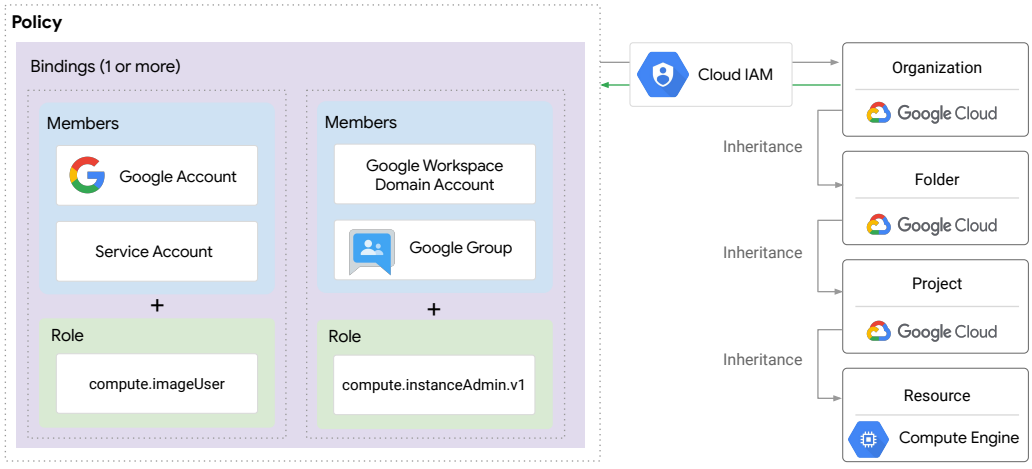
IAM custom roles let you define a precise set of permissions



What if you need something even finer-grained? That's what custom roles permit. A lot of companies use a "least-privilege" model, in which each person in your organization has the minimal amount of privilege needed to do his or her job. So, for example, maybe I want to define an "instanceOperator" role, to allow some users to stop and start Compute Engine virtual machines but not reconfigure them. Custom roles allow me to do that.

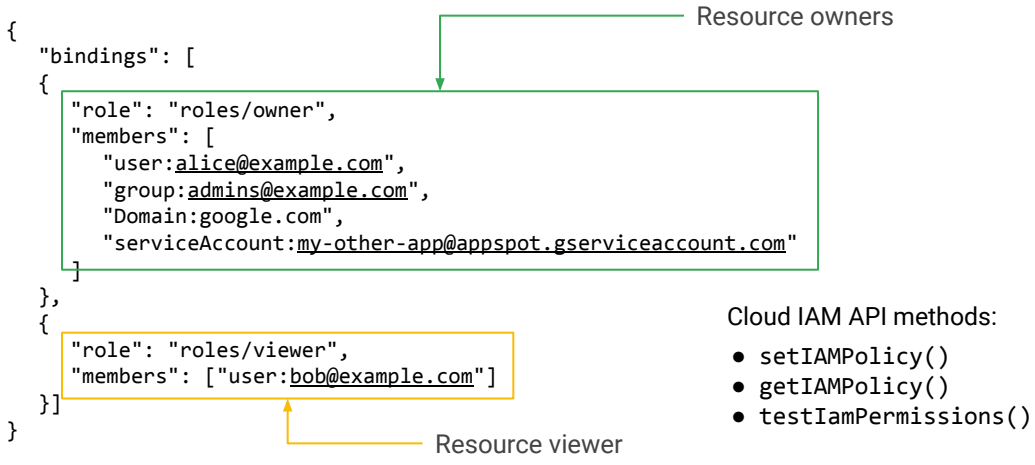
A couple of cautions about custom roles. First, if you decide to use custom roles, you'll need to manage the permissions that make them up. Some companies decide they'd rather stick with the predefined roles. Second, custom roles can only be used at the project or organization levels. They can't be used at the folder level.

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



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

Cloud IAM Policy example



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 Google Cloud APIs.
- Enable authentication and authorization: you can assign specific IAM roles to a service account.

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 Google Cloud

External keys:

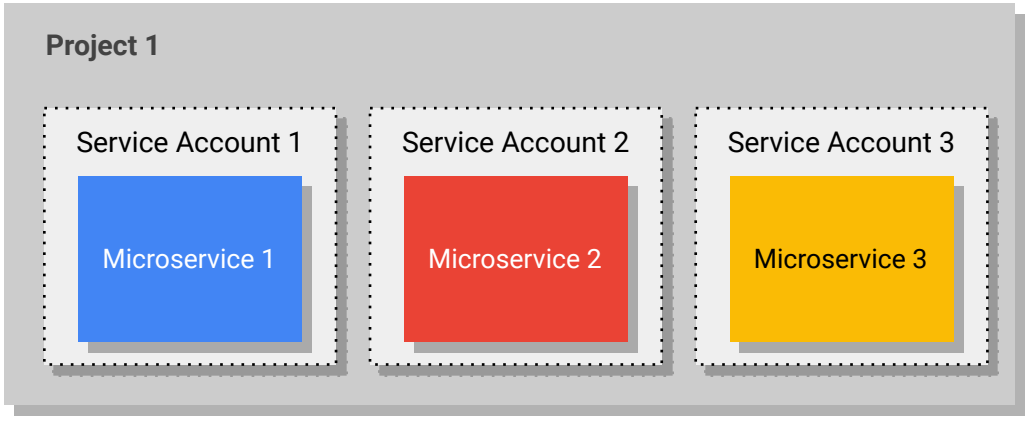
- Can be created for use from outside Google Cloud.
- Require that you be responsible for security of the private key and other management operations such as key rotation.
- Are manageable through the:
  - Cloud IAM API
  - gcloud command-line tool
  - Service Accounts page in the Cloud Console

You can create external keys to be used from outside Google Cloud. 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 Cloud IAM API, the gcloud tool, and the Cloud 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



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.

Steps for using a service account in your application

1. Create the service account via the Cloud 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.

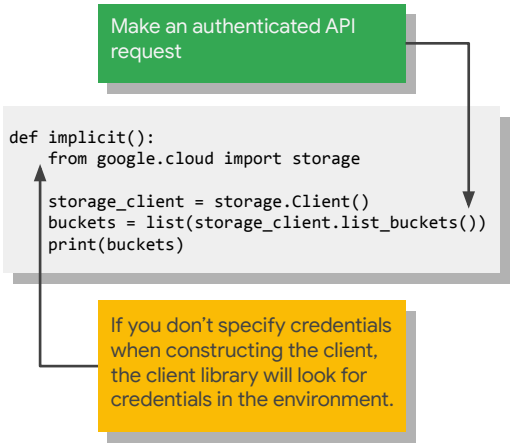
To use a service account in your application, first create the service account via the console. You then generate and download your credentials file, and then you can set an environment variable with the path to your downloaded credentials. The example shows commands to set the environment variable in 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:

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



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

Google Cloud 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, Google Kubernetes Engine (GKE), 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, GKE, 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]

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 accounts keys.
- Use Cloud Audit Logs and export logs to Cloud Storage.
- Set organization-level IAM policies.
- Grant roles to a Google group when possible.

Remember the following best practices when using Cloud IAM.

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 accounts keys.
- Use Cloud Audit Logs and export logs to Cloud Storage.
- Set organization-level IAM policies.
- Grant roles to a Google group when possible.

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.

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 accounts keys.
- Use Cloud Audit Logs and export logs to Cloud Storage.
- Set organization-level IAM policies.
- Grant roles to a Google group when possible.

Rotate service account keys regularly and implement processes to

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 accounts keys.
- Use Cloud Audit Logs and export logs to Cloud Storage.
- Set organization-level IAM policies.
- Grant roles to a Google group when possible.

manage user-managed service account keys.

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 accounts keys.
- Use Cloud Audit Logs and export logs to Cloud Storage.
- Set organization-level IAM policies.
- Grant roles to a Google group when possible.

Don’t check in service account keys to source code.

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 accounts keys.
- **Use Cloud Audit Logs and export logs to Cloud Storage.**
- Set organization-level IAM policies.
- Grant roles to a Google group when possible.

Use **Cloud Audit Logs** to regularly audit changes to your IAM policy and export logs to Cloud Storage.

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 accounts keys.
- Use Cloud Audit Logs and export logs to Cloud Storage.
- Set organization-level IAM policies.
- Grant roles to a Google group when possible.

S**et organization-level IAM policies** to grant access to all projects in your organization and

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 accounts keys.
- Use Cloud Audit Logs and export logs to Cloud Storage.
- Set organization-level IAM policies.
- Grant roles to a Google group when possible.

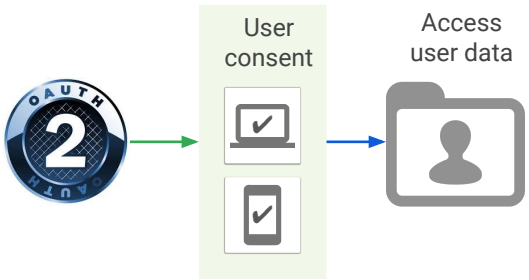
grant roles to a Google group instead of to individual users when possible.

For a detailed list of best practices, see:
https://cloud.google.com/iam/docs/using-iam-securely

Use 0Auth 2.0 to access resources on behalf of a user

Use cases include:

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



It is generally best to use a service account for authentication to a Google Cloud API. In some cases, you might want to access a resource on behalf of a user. Use cases where you might want to access resources on behalf of a user include:

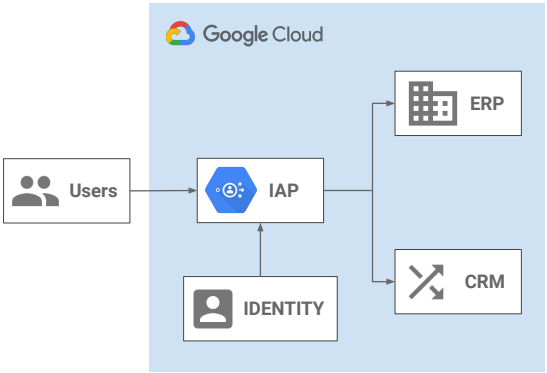
- Your application needing access to BigQuery datasets that belong to your application users, and
- Your application needing to authenticate as a user to create projects on their behalf.

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.

[OAuth 2.0: https://developers.google.com/identity/protocols/OAuth2
Authenticating as an End User:
https://cloud.google.com/docs/authentication/end-user]

Identity-Aware Proxy (IAP)

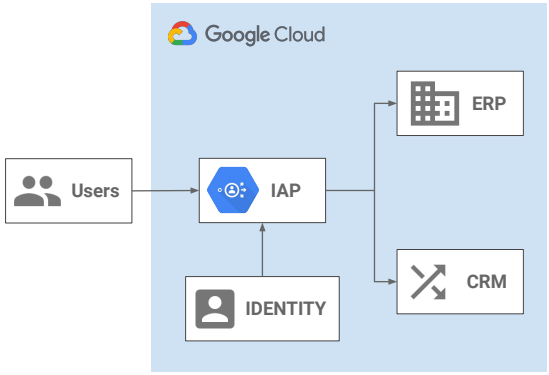
- Controls access to your cloud applications running on Google Cloud.
- Verifies a user's identity.
- Determines *whether* that user should be allowed to access the application.



Identity-Aware Proxy, or IAP, ...

Identity-Aware Proxy (IAP)

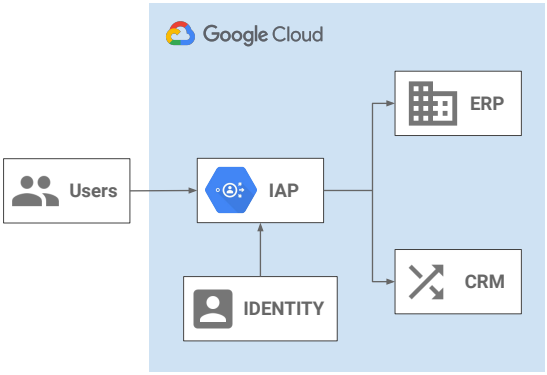
- Controls access to your cloud applications running on Google Cloud.
- Verifies a user's identity.
- Determines *whether* that user should be allowed to access the application.



controls access to your cloud applications running on Google Cloud. IAP ...

Identity-Aware Proxy (IAP)

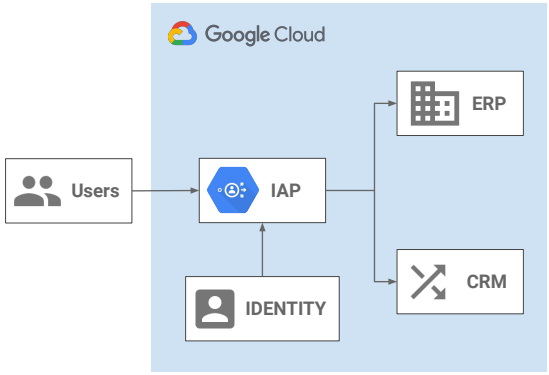
- Controls access to your cloud applications running on Google Cloud.
- Verifies a user's identity.
- Determines *whether* that user should be allowed to access the application.



verifies a user's identity and

Identity-Aware Proxy (IAP)

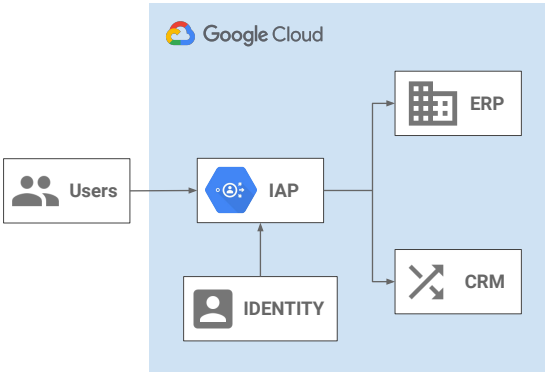
- Controls access to your cloud applications running on Google Cloud.
- Verifies a user's identity.
- Determines *whether* that user should be allowed to access the application.



determines whether that user should be allowed to access the application.

Identity-Aware Proxy (IAP)

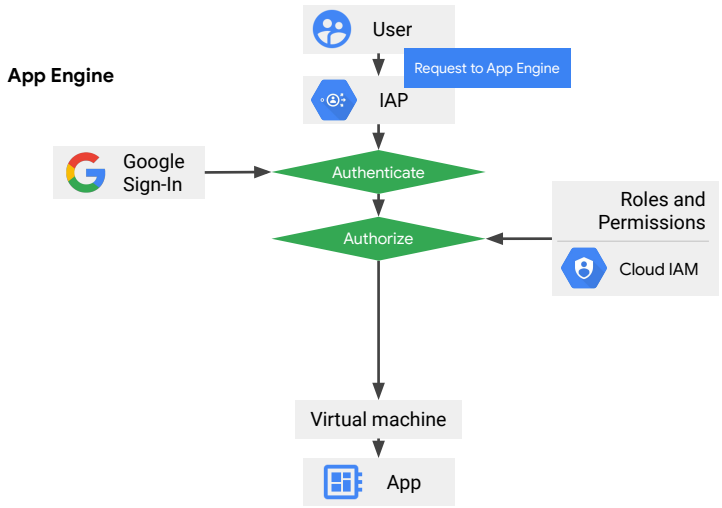
- Controls access to your cloud applications running on Google Cloud.
- Verifies a user's identity.
- Determines *whether* that user should be allowed to access the application.



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.

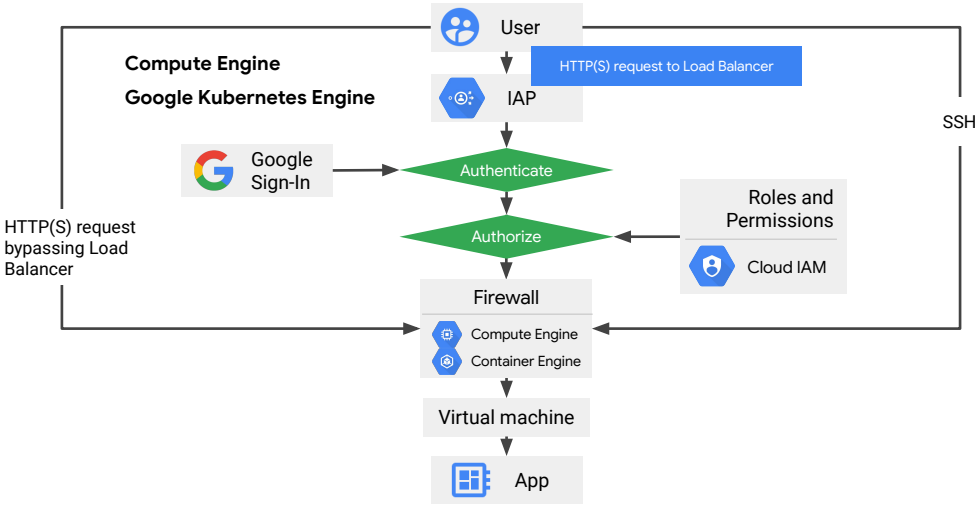
[https://cloud.google.com/iap/docs/]

How IAP works



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

How IAP works



… as well as in **Compute Engine** and GKE.

[https://cloud.google.com/iap/docs/concepts-overview]

Follow these precautions when using 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.

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

Follow these precautions when using 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.

Configure your firewall and load balancer to protect against traffic that doesn't come from the serving infrastructure; and

Follow these precautions when using IAP

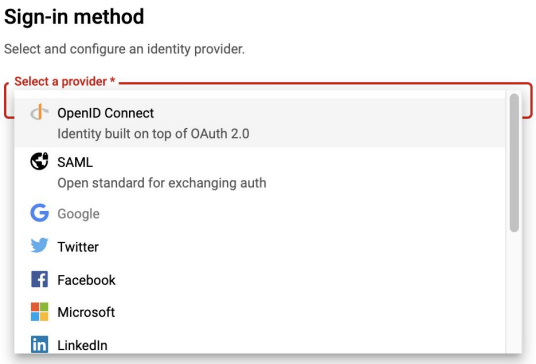
- Configure your firewall and load balancer to protect against traffic that doesn't come from the serving infrastructure.
- Use signed headers or the App Engine standard environment Users API.

Use signed headers or the App Engine standard environment Users API.

[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]

Identity Platform provides authentication as a service

- Provides federated login that integrates with many common providers.
- Used to provide sign-up and sign-in for your end users' applications.



Identity Platform is a customer identity and access management platform for adding identity and access management to applications. It provides authentication as a service that developers access via a set of SDKs.

Identity Platform can be used to support authentication methods like SAML, OpenID Connect, email and password, and phone, as well as a broad range of identity providers like Google, Apple, and Twitter.

Differences between Identity Platform and Firebase Authentication

- Identify Platform offers additional capabilities for enterprise customers.
- Upgrading to Identity Platform does not present any issues for your apps which will continue to work with existing Firebase services.
- Both support a collection of Client and Admin SDKs.

Identity Platform and Firebase Authentication offer similar functionality. Both allow you to easily sign users in to your apps by providing backend services, SDKs, and UI libraries.

However, Identity Platform offers additional capabilities designed for enterprise customers, such as OpenID Connect and SAML authentication, multi-tenancy support, Identity-Aware Proxy integration, and a 99.95% uptime SLA.

Differences between Identity Platform and Firebase Authentication

- Identify Platform offers additional capabilities for enterprise customers.
- Upgrading to Identity Platform does not present any issues for your apps which will continue to work with existing Firebase services.
- Both support a collection of Client and Admin SDKs.

Identity Platform is fully compatible with both Google Cloud and Firebase products. If you are currently using Firebase Authentication and you upgrade to Identity Platform, your app will continue to work with existing Firebase services.

Differences between Identity Platform and Firebase Authentication

- Identify Platform offers additional capabilities for enterprise customers.
- Upgrading to Identity Platform does not present any issues for your apps which will continue to work with existing Firebase services.
- Both support a collection of Client and Admin SDKs.

Identity Platform and Firebase Authentication both support a collection of Client and Admin SDKs. To preserve backwards-compatibility, the SDKs occasionally use Firebase branding and naming conventions.

[https://cloud.google.com/identity-platform/docs/product-comparison]

Lab

Adding User Authentication to your Application

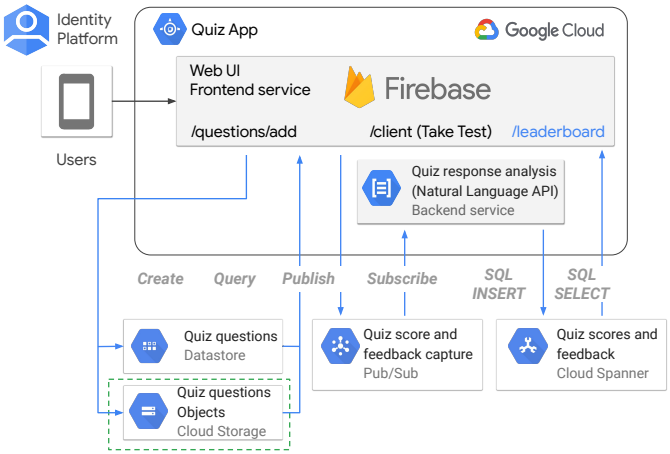
Duration: 45 minutes

In the lab, Adding User Authentication to your Application, you will enhance the online Quiz application to use Identity Platform authentication.

Lab objectives

Add Identity Platform configuration to a client-side web application

Write code to integrate Identity Platform authentication into a client-side web application



Google Cloud

You'll configure Identity Platform authentication to use email address and password credentials and ensure that users register and login before taking a quiz.

Summary

You can control user access to the resources in your application by creating IAM members with the appropriate permissions. Application access to Google Cloud APIs is controlled using service accounts. Your application assumes the identity of the service account to invoke Google Cloud APIs. You can create one or more service accounts to restrict access to different resources in your application.

Identity-Aware Proxy, or IAP for short, enables you to control access to your application. It verifies the user's identity and checks whether that user should be allowed to access the application. End-users simply use an internet-accessible url to access IAP-secured applications. No more VPN!

With Identity Platform, you can add a widely adopted, user-friendly, and customizable authentication service to your web and mobile apps with ease, so you can focus on building your app or service.


Google Cloud