IAM Custom Roles

GSP190

Overview

Cloud IAM provides the right tools to manage resource permissions with minimum fuss and high automation. You don't directly grant users permissions. Instead, you grant them roles, which bundle one or more permissions. This allows you to map job functions within your company to groups and roles. Users get access only to what they need to get the job done, and admins can easily grant default permissions to entire groups of users.

There are two kinds of roles in Cloud IAM:

- Predefined Roles
- Custom Roles

Predefined roles are created and maintained by Google. Their permissions are automatically updated as necessary, such as when new features or services are added to Google Cloud.

Custom roles are user-defined, and allow you to bundle one or more supported permissions to meet your specific needs. Custom roles are not maintained by Google; when new permissions, features, or services are added to Google Cloud, your custom roles will not be updated automatically.You create a custom role by combining one or more of the available Cloud IAM permissions. Permissions allow users to perform specific actions on Google Cloud resources.

What you'll do

- Create, update, delete, undelete custom roles

Prerequisites

- Familiarity with IAM Roles

Setup

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This Qwiklabs hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

What you need

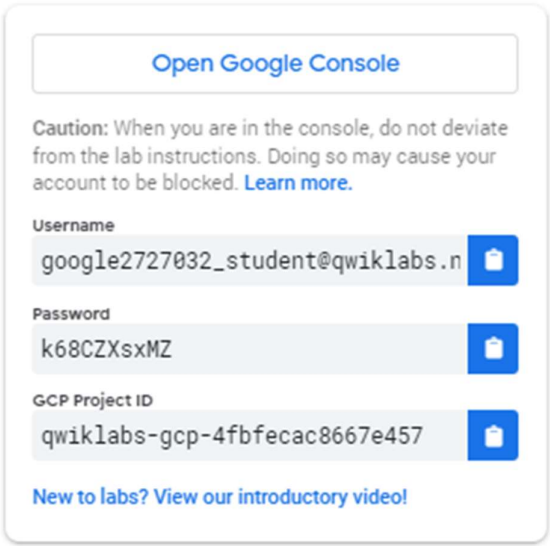
To complete this lab, you need:

- Access to a standard internet browser (Chrome browser recommended).
- Time to complete the lab.
  Note: If you already have your own personal Google Cloud account or project, do not use it for this lab.

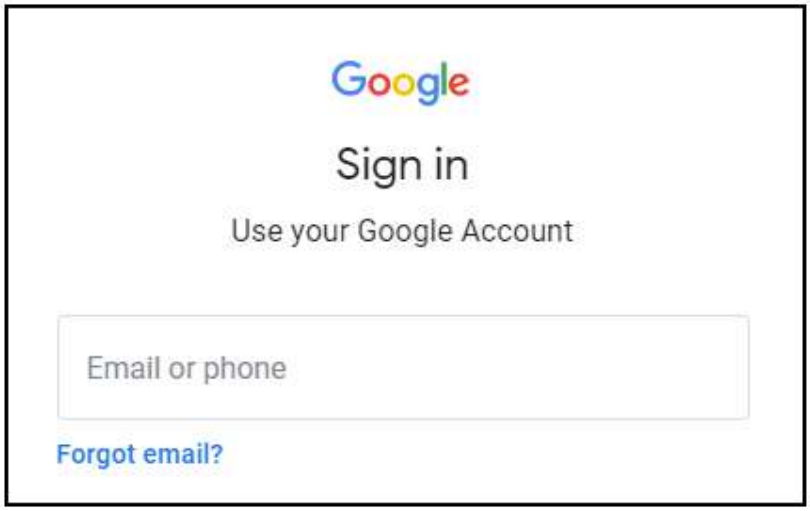
Note: If you are using a Pixelbook, open an Incognito window to run this lab.

How to start your lab and sign in to the Google Cloud Console

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.

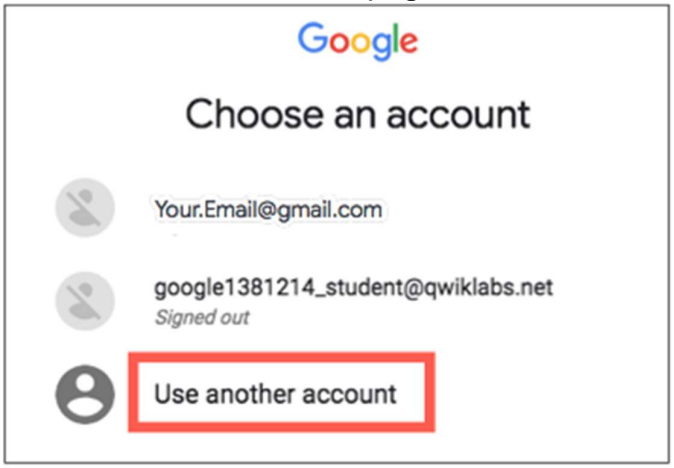


2. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.



Tip: Open the tabs in separate windows, side-by-side.

If you see the **Choose an account** page, click **Use Another Account**.



3. In the **Sign in** page, paste the username that you copied from the Connection Details panel. Then copy and paste the password.

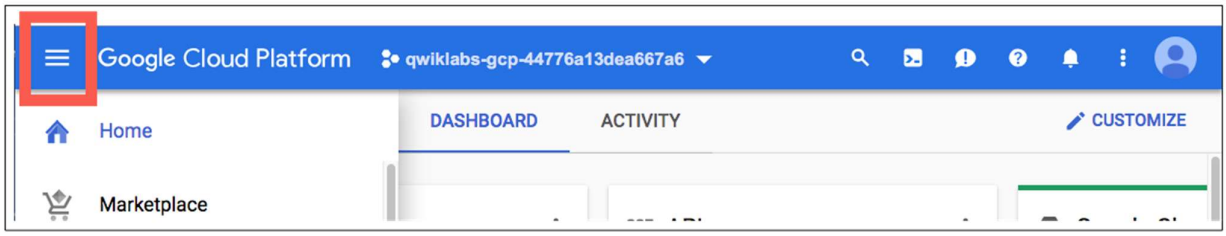
   Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

4. Click through the subsequent pages:

   - Accept the terms and conditions.
   - Do not add recovery options or two-factor authentication (because this is a temporary account).
   - Do not sign up for free trials.

After a few moments, the Cloud Console opens in this tab.

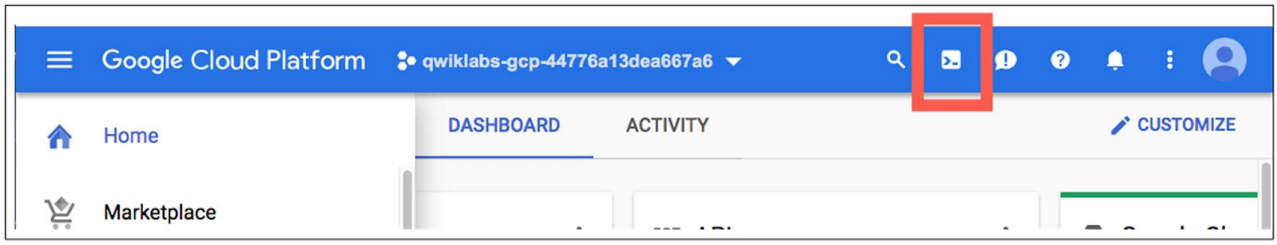
Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



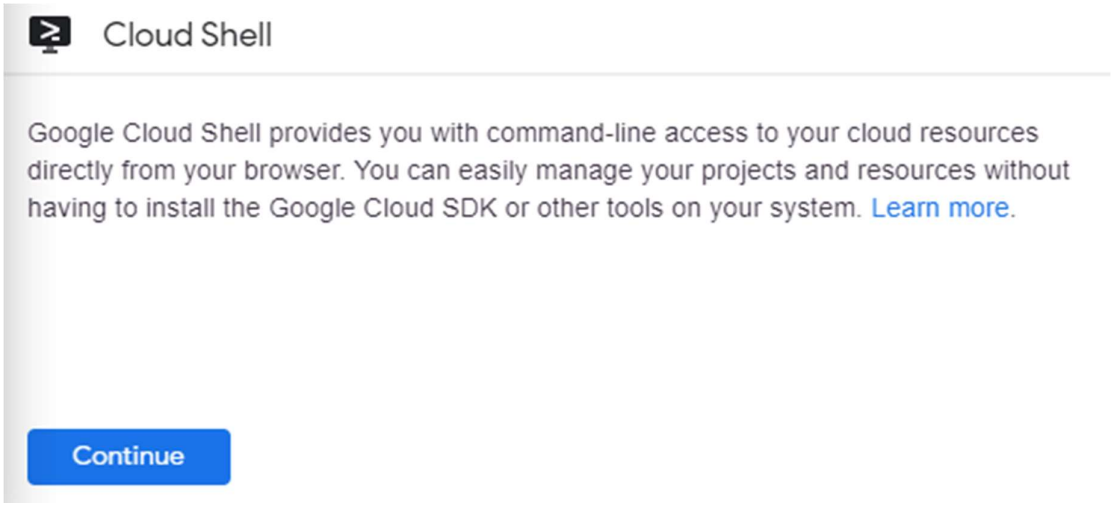
Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

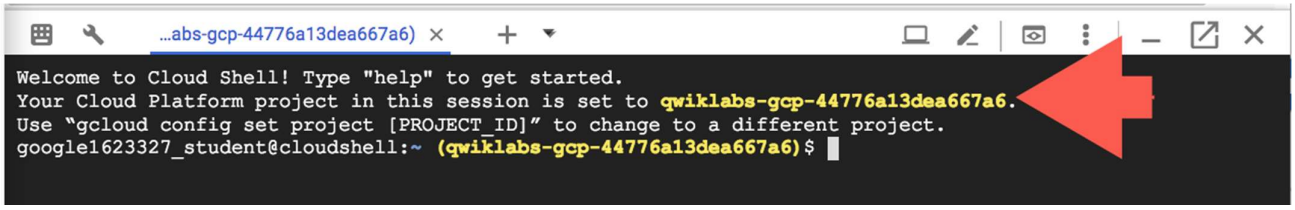
In the Cloud Console, in the top right toolbar, click the **Activate Cloud Shell** button.



Click **Continue**.



It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT_ID*. For example:



`gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

You can list the active account name with this command:

```
gcloud auth list
```

(Output)

```
Credentialed accounts:
 - <myaccount>@<mydomain>.com (active)
```

(Example output)

```
Credentialed accounts:
 - google1623327_student@qwiklabs.net
```

You can list the project ID with this command:

```
gcloud config list project
```

(Output)

```
[core]
project = <project_ID>
```

(Example output)

```
[core]
project = qwiklabs-gcp-44776a13dea667a6
```

For full documentation of `gcloud` see the gcloud command-line tool overview.

Understanding IAM Custom Roles

Cloud IAM also provides the ability to create customized Cloud IAM roles. You can create a custom Cloud IAM role with one or more permissions and then grant that custom role to users. Cloud IAM provides a UI and API for creating and managing custom roles.

Key Point: Custom roles enable you to enforce the principle of least privilege, ensuring that the user and service accounts in your organization have only the permissions essential to performing their intended functions.

Note: You can create a custom role at the organization level and at the project level. However, you cannot create custom roles at the folder level.
You create a custom role by combining one or more of the available Cloud IAM permissions. Permissions allow users to perform specific actions on Google Cloud resources.

In the Cloud IAM world, permissions are represented in the form:

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

For example, the `compute.instances.list` permission allows a user to list the Compute Engine instances they own, while `compute.instances.stop` allows a user to stop a VM.

Permissions usually, but not always, correspond 1:1 with REST methods. That is, each Google Cloud service has an associated permission for each REST method that it has. To call a method, the caller needs that permission. For example, the caller of `topic.publish()` needs the `pubsub.topics.publish` permission.

Custom roles can only be used to grant permissions in policies for the same project or organization that owns the roles or resources under them. You cannot grant custom roles from one project or organization on a resource owned by a different project or organization.

Required permissions and roles

To create a custom role, a caller must have the `iam.roles.create` permission.

Users who are not owners, including organization administrators, must be assigned either the Organization Role Administrator role (roles/iam.organizationRoleAdmin) or the IAM Role Administrator role (roles/iam.roleAdmin). The IAM Security Reviewer role (roles/iam.securityReviewer) enables the ability to view custom roles but not administer them.

The custom roles user interface is in the Cloud Console under IAM Roles. It is only available to users who have permissions to create or manage custom roles. By default, only project owners can create new roles. Project owners can control access to this feature by granting IAM Role Administrator role to others on the same project; for organizations, only Organization Administrators can grant the Organization Role, Administrator role.

Preparing to create a custom role

Before you create a custom role, you might want to know:

- What permissions can be applied to a resource
- What roles are grantable on a resource
- What a role's metadata is

Viewing the available permissions for a resource

Before you create a custom role, you might want to know what permissions can be applied to a resource. You can get all permissions that can be applied to a resource, and the resources below that in the hierarchy, using the gcloud command-line tool, the Cloud Console, or the IAM API. For example, you can get all permissions that you can apply on an organization and on projects in that organization.

Run the following to get the list of permissions available for your project.

```
gcloud iam list-testable-permissions
//cloudresourcemanager.googleapis.com/projects/$DEVSHELL_PROJECT_ID
```

(Output)

```
name: appengine.applications.create
stage: GA
---
name: appengine.applications.get
stage: GA
---
name: appengine.applications.update
stage: GA
---
name: appengine.instances.delete
 stage: GA
---
name: appengine.instances.get
stage: GA
---
name: appengine.instances.list
stage: GA
---
customRolesSupportLevel: TESTING
name: appengine.memcache.addKey
stage: BETA
---
customRolesSupportLevel: TESTING
name: appengine.memcache.flush
stage: BETA
---
```

Getting the role metadata

Before you create a custom role, you might want to get the metadata for both predefined and custom roles. Role metadata includes the role ID and permissions contained in the role. You can view the metadata using the Cloud Console or the IAM API.

To view the role metadata, use command below, replacing `[ROLE_NAME]` with the role. For example: `roles/viewer` or `roles/editor`

```
gcloud iam roles describe [ROLE_NAME]
```

Example Outptut (for roles/viewer):

```
description: Read access to all custom roles in the project.
etag: AA==
includedPermissions:
- iam.roles.get
- iam.roles.list
- resourcemanager.projects.get
- resourcemanager.projects.getIamPolicy
...
...
name: roles/iam.roleViewer
stage: GA
title: Role Viewer
```

Viewing the grantable roles on resources

Use the `gcloud iam list-grantable-roles` command to return a list of all roles that can be applied to a given resource.

Execute the following `gcloud` command to list grantable roles from your project:

```
gcloud iam list-grantable-roles
//cloudresourcemanager.googleapis.com/projects/$DEVSHELL_PROJECT_ID
```

Your output will look something like this:

```
---
description: Full management of App Engine apps (but not storage).
name: roles/appengine.appAdmin
title: App Engine Admin
---
description: Ability to view App Engine app status.
name: roles/appengine.appViewer
title: App Engine Viewer
---
description: Ability to view App Engine app status and deployed source code.
name: roles/appengine.codeViewer
title: App Engine Code Viewer
---
...
...
```

Creating a custom role

To create a custom role, a caller must possess `iam.roles.create` permission. By default, the owner of a project or an organization has this permission and can create and manage custom roles.

Users who are not owners, including organization admins, must be assigned either the Organization Role Administrator role, or the IAM Role Administrator role.

Use the `gcloud iam roles create` command to create new custom roles. You can use this command in two ways:

- By providing a YAML file that contains the role definition
- By using flags to specify the role definition When creating a custom role, you must specify whether it applies to the organization level or project level by using the `--organization [ORGANIZATION_ID]` or `--project [PROJECT_ID]` flags. Each example below creates a custom role at the project level.
  In the exercises to follow you'll create custom roles at the project level.

To create a custom role using a YAML file

Create a YAML file that contains the definition for your custom role. The file must be structured in the following way:

```
title: [ROLE_TITLE]
description: [ROLE_DESCRIPTION]
stage: [LAUNCH_STAGE]
includedPermissions:
- [PERMISSION_1]
- [PERMISSION_2]
```

Each of the placeholder values is described below:

- `[ROLE_TITLE]` is a friendly title for the role, such as **Role Viewer**.
- `[ROLE_DESCRIPTION]` is a short description about the role, such as **My custom role description**.
- `[LAUNCH_STAGE]` indicates the stage of a role in the launch lifecycle, such as ALPHA, BETA, or GA.
- `includedPermissions` specifies the list of one or more permissions to include in the custom role, such as **iam.roles.get**.
  Time to get started! Create your role definition YAML file by running:

```
nano role-definition.yaml
```

Add this custom role definition to the YAML file:

```
title: "Role Editor"
description: "Edit access for App Versions"
stage: "ALPHA"
includedPermissions:
- appengine.versions.create
- appengine.versions.delete
```

Then save and close the file.

Execute the following `gcloud` command:

```
gcloud iam roles create editor --project $DEVSHELL_PROJECT_ID \
--file role-definition.yaml
```

If the role was created successfully, the following response is returned:

```
Created role [editor].
description: Edit access for App Versions
etag: BwVs4O4E3e4=
includedPermissions:
- appengine.versions.create
- appengine.versions.delete
name: projects/[PROJECT_ID]/roles/editor
stage: ALPHA
title: Role Editor
```

Click *Check my progress* to verify the objective.

Create a custom role using flags

Now you'll use the flag method to create a new custom role. The flags take a similar form to the YAML file, so you'll recognize how the command is being built.

Execute the following `gcloud` command to create a new role using flags:

```
gcloud iam roles create viewer --project $DEVSHELL_PROJECT_ID \
--title "Role Viewer" --description "Custom role description." \
--permissions compute.instances.get,compute.instances.list --stage ALPHA
```

sample output:

```
Created role [viewer].
description: Custom role description.
etag: BwVs4PYHqYI=
includedPermissions:
- compute.instances.get
- compute.instances.list
name: projects/[PROJECT_ID]/roles/viewer
stage: ALPHA
title: Role Viewer
```

Click *Check my progress* to verify the objective.

Listing the custom roles

Execute the following `gcloud` command to list custom roles, specifying either project-level or organization-level custom roles:

```
gcloud iam roles list --project $DEVSHELL_PROJECT_ID
```

Example Output:

```
---
description: Edit access for App Versions
etag: BwVxLgrnawQ=
name: projects/[PROJECT_ID]/roles/editor
title: Role Editor
---
description: Custom role description.
etag: BwVxLg18IQg=
name: projects/[PROJECT_ID]/roles/viewer
title: Role Viewer
```

To list deleted roles, you can also specify the `--show-deleted` flag.

Execute the following `gcloud` command to list predefined roles:

```
gcloud iam roles list
```

Editing an existing custom role

A common pattern for updating a resource's metadata, such as a custom role, is to read its current state, update the data locally, and then send the modified data for writing. This pattern could cause a conflict if two or more independent processes attempt the sequence simultaneously.

For example, if two owners for a project try to make conflicting changes to a role at the same time, some changes could fail.

Cloud IAM solves this problem using an `etag` property in custom roles. This property is used to verify if the custom role has changed since the last request. When you make a request to Cloud IAM with an etag value, Cloud IAM compares the etag value in the request with the existing etag value associated with the custom role. It writes the change only if the etag values match.

Use the `gcloud iam roles update` command to update custom roles. You can use this command in two ways:

- By providing a YAML file that contains the updated role definition
- By using flags to specify the updated role definition
  When updating a custom role, you must specify whether it applies to the organization level or project level by using the `--organization [ORGANIZATION_ID]` or `--project [PROJECT_ID]` flags. Each example below creates a custom role at the project level.

The `describe` command returns the role's definition and includes an etag value that uniquely identifies the current version of the role. The etag value should be provided in the updated role definition to ensure that any concurrent role changes are not overwritten.

To update a custom role using a YAML file

Get the current definition for the role by executing the following `gcloud` command, replacing `[ROLE_ID]` with **editor**.

```
gcloud iam roles describe [ROLE_ID] --project $DEVSHELL_PROJECT_ID
```

The `describe` command returns the following output:

```
description: [ROLE_DESCRIPTION]
etag: [ETAG_VALUE]
includedPermissions:
```

```
- [PERMISSION_1]
- [PERMISSION_2]
name: [ROLE_ID]
stage: [LAUNCH_STAGE]
title: [ROLE_TITLE]
```

You'll create a new YAML file with the outputted value. Copy the output from this command.

Create a `new-role-definition.yaml` file with your editor.

```
nano new-role-definition.yaml
```

Paste in the outputted value from the last command and add these two permissions under `includedPermissions`:

```
- storage.buckets.get
- storage.buckets.list
```

Your YAML file will look like this when you're finished:

```
description: Edit access for App Versions
etag: BwVxIAbRq_I=
includedPermissions:
- appengine.versions.create
- appengine.versions.delete
- storage.buckets.get
- storage.buckets.list
name: projects/[PROJECT_ID]/roles/editor
stage: ALPHA
title: Role Editor
```

Save and close the file.

Now you'll use the `update` command to update the role. Execute the following `gcloud` command, replacing `[ROLE_ID]` with **editor**:

```
gcloud iam roles update [ROLE_ID] --project $DEVSHELL_PROJECT_ID \
--file new-role-definition.yaml
```

If the role was updated successfully, the following response is returned:

```
description: Edit access for App Versions
etag: BwVxIBjfN3M=
includedPermissions:
- appengine.versions.create
- appengine.versions.delete
- storage.buckets.get
- storage.buckets.list
name: projects/[PROJECT_ID]/roles/editor
stage: ALPHA
title: Role Editor
```

Click *Check my progress* to verify the objective.

To update a custom role using flags

Each part of a role definition can be updated using a corresponding flag. See the gcloud iam roles update topic for a list of all possible flags.
You can use the following flags to add or remove permissions:

- `--add-permissions`: Adds one or more comma-separated permissions to the role.
- `--remove-permissions`: Removes one or more comma-separated permissions from the role.
  Alternatively, you can simply specify the new permissions using the `--permissions [PERMISSIONS]` flag and providing a comma-separated list of permissions to replace the existing permissions list.

Execute the following `gcloud` command to add permissions to the **viewer** role using flags:

```
gcloud iam roles update viewer --project $DEVSHELL_PROJECT_ID \
--add-permissions storage.buckets.get,storage.buckets.list
```

If the role was updated successfully, the following response is returned:

```
description: Custom role description.
etag: BwVxLi4wTvk=
includedPermissions:
- compute.instances.get
- compute.instances.list
- storage.buckets.get
- storage.buckets.list
name: projects/[PROJECT_ID]/roles/viewer
stage: ALPHA
title: Role Viewer
```

Click *Check my progress* to verify the objective.

Disabling a custom role

When a role is disabled, any policy bindings related to the role are inactivated, meaning that the permissions in the role will not be granted, even if you grant the role to a user.

The easiest way to disable an existing custom role is to use the `--stage` flag and set it to DISABLED.

Execute the following `gcloud` command to disable the **viewer** role:

```
gcloud iam roles update viewer --project $DEVSHELL_PROJECT_ID \
--stage DISABLED
```

If the role was updated successfully, the following response is returned:

```
description: Custom role description.
etag: BwVxLkIYHrQ=
includedPermissions:
- compute.instances.get
- compute.instances.list
- storage.buckets.get
- storage.buckets.list
name: projects/[PROJECT_ID]/roles/viewer
stage: DISABLED
title: Role Viewer
```

Click *Check my progress* to verify the objective.

Deleting a custom role

Use the `gcloud iam roles delete` command to delete a custom role. Once deleted the role is inactive and cannot be used to create new IAM policy bindings.

```
gcloud iam roles delete viewer --project $DEVSHELL_PROJECT_ID
```

Example Output:

```
description: Custom role description.
etag: BwVxLkf_epw=
includedPermissions:
- compute.instances.get
- compute.instances.list
- storage.buckets.get
- storage.buckets.list
name: projects/[PROJECT_ID]/roles/viewer
stage: DISABLED
title: Role Viewer
```

After the role has been deleted, existing bindings remain, but are inactive. The role can be undeleted within 7 days. After 7 days, the role enters a permanent deletion process that lasts 30 days. After 37 days, the Role ID is available to be used again.

Note: If a role is being phased out, change its **role.stage** property to **DEPRECATED**, and set the `deprecation_message` to let users know what alternative roles should be used or where to get more information.

Undeleting a custom role

Within the 7 days window you can undelete a role. Deleted roles are in a **DISABLED** state. You can make it available again by updating the `--stage` flag:

```
gcloud iam roles undelete viewer --project $DEVSHELL_PROJECT_ID
```

Click *Check my progress* to verify the objective.

Congratulations

Finish Your Quest

This self-paced lab is part of the Qwiklabs Security & Identity Fundamentals Quest. A Quest is a series of related labs that form a learning path. Completing this Quest earns you the badge above, to recognize your achievement. You can make your badge (or badges) public and link to them in your online resume or social media account. Enroll in this Quest and get immediate completion credit if you've taken this lab. See other available Qwiklabs Quests.

Take Your Next Lab

Continue your Quest with Getting Started with Cloud KMS or try one of these:

- Service Accounts and Roles: Fundamentals
- VPC Network Peering

Next Steps / Learn More

- Read more about Cloud Identity and Access Management: https://cloud.google.com/iam/

Google Cloud Training & Certification

...helps you make the most of Google Cloud technologies. Our classes include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. Certifications help you validate and prove your skill and expertise in Google Cloud technologies.
Manual Last Updated September 3, 2020
Lab Last Tested September 3, 2020