Filter
Plan cluster security
Authenticate and authorize
Authenticate to the GKE API
Authenticate to Google Cloud APIs from GKE
About RBAC and IAM
Best practices for RBAC
About service accounts in GKE
Authenticate to the Kubernetes API server
Use external identity providers to authenticate to GKE clusters
Authorize actions in clusters using GKE RBAC
Manage permissions for groups using Google Groups with RBAC
Authorize access to Google Cloud resources using IAM policies
Manage node SSH access without using SSH keys
Enable access and view cluster resources by namespace

Send feedback

Authorize actions in clusters using role-based access control

AUTOPILOT STANDARD

This page shows you how to authorize actions on resources in your Google Kubernetes Engine (GKE) clusters using the built-in role-based access control (RBAC) mechanism in Kubernetes.

RBAC is a core security feature in Kubernetes that lets you create fine-grained permissions to manage what actions users and workloads can perform on resources in your clusters. As a platform administrator, you create RBAC *roles* and bind those roles to *subjects*, which are authenticated users such as service accounts or Groups. Kubernetes RBAC is enabled by default.

Before you begin

Before you start, make sure you have performed the following tasks:

Kubernetes RBAC to control access to your GKE cluster:

- IAM is not specific to Kubernetes; it provides identity management for multiple Google Cloud products, and operates primarily at the level of the Google Cloud project.
- Kubernetes RBAC is a core component of Kubernetes and lets you create and grant roles (sets of permissions) for any object or type of object *within* the cluster.
- To authorize an action, GKE checks for an RBAC policy first. If there isn't an RBAC policy, GKE checks for IAM permissions.

In GKE, IAM and Kubernetes RBAC are integrated to authorize users to perform actions if they have sufficient permissions according to *either* tool. This is an important part of bootstrapping a GKE cluster, since by default Google Cloud users do not have any Kubernetes RBAC RoleBindings.

To authorize users using Google Cloud accounts, the client must be correctly configured to authenticate using those accounts first. For example, if you are using `kubectl`, you must configure the `kubectl` command to authenticate to Google Cloud before running any commands that require authorization.

Note: Many failures that appear to be due to authorization are actually caused because the cluster is unable to *authenticate* the client. For example, there are special requirements for authenticating from Compute Engine instances, which are described in Cluster access for kubectl.

In almost all cases, Kubernetes RBAC can be used instead of IAM. GKE users require at minimum, the `container.clusters.get` IAM permission in the project that contains the cluster. This permission is included in the `container.clusterViewer` role, and in other more highly privileged roles. The `container.clusters.get` permission is required for users to *authenticate* to the clusters in the project, but does not *authorize* them to perform any actions inside those clusters. Authorization may then be provided by either IAM or Kubernetes RBAC.

Define and assign permissions

★ **Note:** RBAC roles are *additive*. There are no **deny** rules. When structuring your RBAC roles, think of each role as *granting* access to resources.

Define permissions using Roles or ClusterRoles

You define permissions within a Role or ClusterRole object. A Role defines access to resources within a single Namespace, while a ClusterRole defines access to resources in the entire cluster.

Roles and ClusterRoles have the same syntax. Each has a `rules` section, where you define the resources the rule applies to and allowed operations for the Role. For example, the following Role grants read access (`get`, `watch`, and `list`) to all pods in the `accounting` Namespace:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: accounting
  name: pod-reader
rules:
- apiGroups: [""] # "" indicates the core API group
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

Refer to the Role and ClusterRole API documentation for a full list of allowed fields.

★ **Note:** You cannot create a Role that defines permissions unless you already have the permissions defined in the Role. If you have been granted the `cluster-admin` IAM role, this is sufficient.

`Role` vs. `ClusterRole`

Because permissions granted by a ClusterRole apply across the entire cluster, you can use ClusterRoles to control access to different kinds of resources than you can with Roles. These include:

IAM service account	`User`	Automatically generated IAM service account email address
Google Group address on a verified domain	`Group`	Email address of a Google Workspace Group that is a member of the `gke-security-groups` group. For instructions to set up Google Groups for RBAC, refer to Configure Google Groups for RBAC.

The following RoleBinding grants the `pod-reader` Role to a user, a Kubernetes service account, an IAM service account, and a Google Group:

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: pod-reader-binding
  namespace: accounting
subjects:
# Google Cloud user account
- kind: User
  name: janedoe@example.com
# Kubernetes service account
- kind: ServiceAccount
  name: johndoe
# IAM service account
- kind: User
  name: test-account@test-project.iam.gserviceaccount.com
# Google Group
- kind: Group
  name: accounting-group@example.com
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

Verify API access using `kubectl`

`kubectl` provides the `auth can-i` subcommand for quickly querying the API authorization layer. As a platform administrator, you might need to impersonate users to determine what actions they can perform. You can use the `auth can-i` and pass an additional `--as` flag.

denied due to lack of sufficient permissions, the API server logs an `RBAC DENY` error, along with additional information such as the user's implicit and explicit group membership. If you are using Google Groups for RBAC, `google groups` appears in the log message.

Limitations

The following sections describe interactions that might not seem obvious when working with Kubernetes RBAC and IAM.

Default discovery roles

Clusters are created with a set of default ClusterRoles and ClusterRoleBindings. Requests made with valid credentials are placed in the `system:authenticated` group, whereas all other requests fall into `system:unauthenticated`.

The `system:basic-user` ClusterRole lets users make `SelfSubjectAccessReviews` to test their permissions in the cluster. The `system:discovery` role lets users read discovery APIs, which can reveal information about `CustomResourceDefinitions` added to the cluster.

Anonymous users (`system:unauthenticated`) receive the `system:public-info-viewer` ClusterRole instead, which grants read-only access to `/healthz` and `/version` APIs.

To see the API endpoints allowed by the `system:discovery` ClusterRole, run the following command:

```
kubectl get clusterroles system:discovery -o yaml
```

Forbidden error for service accounts on Google Cloud VM instances

The following error can occur when the VM instance does not have the `userinfo-email` scope:

To create a new role binding that uses the service account's unique ID for an existing VM, perform the following steps:

1. Identify the service account's unique ID:

```
gcloud iam service-accounts describe SERVICE_ACCOUNT_EMA
```

   For example, the following output displays the `uniqueId` for the `my-iam-account@somedomain.com` service account:

```
displayName: Some Domain IAM service account
email: my-iam-account@somedomain.com
etag: BwWWja0YfJA
name: projects/project-name/serviceAccounts/my-iam-accou
oauth2ClientId: '123456789012345678901'
projectId: project-name
uniqueId: '123456789012345678901'
```

2. Create a role binding using the `uniqueId` of the service account:

```
kubectl create clusterrolebinding CLUSTERROLEBINDING_NAM
    --clusterrole cluster-admin \
    --user UNIQUE_ID
```

Permission to create or update roles and role bindings

In Kubernetes, you can only create or update a role or a role binding with specific permissions if you meet the following conditions:

- Create or update a role: You must already have the same permissions that you want to grant to the role. Alternatively, you must have authorization to perform the `escalate` verb on the role.
- Create or update a role binding: You must already have the same permissions that are granted in the role being bound, with the same scope as the role binding. Alternatively, you must have

To mitigate this limitation, grant the caller the permissions in the role using RBAC instead of IAM.

You can alternatively use either RBAC or IAM to grant the caller the `escalate` verb, the `bind` verb, or both. However, GKE does not recommend this approach, because the caller can then grant *any* permission to any role.

What's next

- Learn how to create IAM policies.
- Learn how to configure Google Groups for RBAC.

Was this helpful?

Send feedback

Why Google	Products and pricing	Solutions	Resources	Engage
Choosing Google Cloud	Google Cloud pricing	Infrastructure modernization	Google Cloud Affiliate Program	Contact sales
Trust and security	Google Workspace pricing	Databases	Google Cloud documentation	Find a Partner
Modern Infrastructure Cloud	See all products	Application modernization	Google Cloud quickstarts	Become a Partner
Multicloud		Smart analytics	Google Cloud Marketplace	Events
Global infrastructure		Artificial Intelligence	Learn about cloud computing	Podcasts
Customers and		Security		Developer Center
		Productivity &		Press Corner