# Access Control

## Overview

- By default, when you create a project on GCP, you are the only user who has access to the project or its resources.
- Two types of access control in GKE:
    - Cloud IAM
    - Kubernetes RBAC

## Cloud IAM

- Manages GCP resources, including clusters and types of objects.
- There is no mechanism for granting permissions for specific Kubernetes objects within Cloud IAM.
- A cloud IAM role grants privileges across all clusters in the project, or all clusters in child projects.
- Good choice if other GCP components are used.

## Kubernetes RBAC

- A Role can be scoped to a specific object (or type of objects), and defines which actions (called verbs) the Role grants in relation to that object.
- A RoleBinding is also a Kubernetes object, and grants Roles to users. In a GKE user can be any of:
  - GCP user
  - GCP service account
  - Kubernetes service account
  - G Suite user
  - G Suite Google Group (beta)

## Defining and assigning permissions

- **ClusterRole/Role** object: defines resource types and operations that can be assigned to a user or group of users in a cluster/namespace.
- **ClusterRoleBinding/RoleBinding** object: assigns the corresponding ClusterRole/Role to a user or group of users.
- Roles are purely additive (no "deny" rules).

## Example (Role)

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

- `ClusterRole` has the same syntax, except that it is not namespaced

## Example (RoleBinding)

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

## Service Accounts

- Humans authenticate through User Accounts.
- Service Accounts are needed when processes inside of containers need to access the K8s API server.
- Also needed for other external tools (Web UI Dashboard, Helm).

```
kind: ServiceAccount
apiVersion: v1
metadata:
name: my-service-account
```

## Using service accounts

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: use-service-account-pod
spec:
# Set the service account containers in this pod
# use when they make requests to the API server
serviceAccountName: my-service-account
containers:
  - name: container-service-account
    image: nginx:latest
```

- Service accounts are on a namespace level

## Examples

- Kubernetes dashboard
- For installing Jenkins (through Helm)

```
kubectl create clusterrolebinding cluster-admin-binding \
--clusterrole=cluster-admin --user=USER
```

```
kubectl create serviceaccount tiller --namespace kube-system
```

```
kubectl create clusterrolebinding tiller-admin-binding \
--clusterrole=clusteradmin --serviceaccount=kube-system:tille
```

## Security

## Security

- Many layers to protect:
    - Container image
    - Container runtime
    - Cluster network
    - Access to the cluster API server.
- Kubernetes master components use by default a public IP address.

## Node security

- GKE nodes use Google's Container-Optimized OS which has some useful features:
    - Locked-down firewall.
    - Read-only filesystem where possible.
    - Limited user accounts and disabled root login.

## Network security

- Limit pod-to-pod communication via **network policies.**
- Filtering load balanced traffic:
    - Set `loadBalancerSourceRanges` in a `Service` definition to provide a list of whitelisted CIDR ranges.
- Setting up pod security policies.