

Kubernetes RBAC Cheat Sheet

RBAC isn't broken — it just answers one narrow question. Learn the question and the four objects, and a denied request stops being a reason to hand out cluster-admin.

THE ONLY QUESTION RBAC ANSWERS

Subject — who

User, Group, or ServiceAccount



Verb — what action

get, list, watch, create, delete...



Resource — what object

Pods, secrets, deployments...

The four objects

Role

Defines permissions inside a single namespace.

ONE NAMESPACE

ClusterRole

Defines permissions cluster-wide, or a reusable set to bind per namespace.

CLUSTER-WIDE

RoleBinding

Grants a Role or a ClusterRole to a subject, inside one namespace.

ONE NAMESPACE

ClusterRoleBinding

Grants a ClusterRole to a subject across the entire cluster.

CLUSTER-WIDE

Roles and ClusterRoles **define**. Bindings **hand out**. A Role that is never bound does nothing — it's a template on a shelf.

Which pairing gives what scope

DEFINITION	BINDING	RESULT
Role	RoleBinding	One namespace. Smallest blast radius.
ClusterRole	RoleBinding	Still one namespace — write once, bind per namespace.
ClusterRole	ClusterRoleBinding	Every namespace. Needs a written reason.
Role	ClusterRoleBinding	Not valid — a ClusterRoleBinding only references a ClusterRole.

Role vs ClusterRole — decision flow

- Does the app only touch **one namespace**? Yes → **Role + RoleBinding**
- Same permissions needed in **several namespaces**? Yes → **ClusterRole + RoleBinding each**
- Needs **cluster-scoped** resources (nodes, namespaces) or truly every namespace? Yes → **ClusterRole + ClusterRoleBinding**
- Can't write a **one-line reason** for why this subject needs cluster scope? **Then it shouldn't be cluster-wide.**

A Role + RoleBinding, line by line

```
kind: Role
metadata:
  namespace: billing
  name: pod-reader
rules:
- apiGroups: [""]          # core group
  resources: ["pods"]      # what
  verbs: ["get","list","watch"] # which actions
---
kind: RoleBinding
metadata:
  name: read-pods
  namespace: billing
subjects:
- kind: ServiceAccount      # who
  name: billing-reader
  namespace: billing
roleRef:
  kind: Role                # which Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

Result: **billing-reader** can read pods, only in **billing**. That's the whole blast radius.

Common verbs

READ

`get` `list` `watch`

CHANGE

`create` `update` `patch`

REMOVE

`delete` `deletecollection`

HANDLE WITH CARE

`*` `escalate` `bind` `impersonate`

list and **watch** on secrets return the secret contents — treat them like **get**.

Where teams get it wrong

1 • ClusterRoleBinding by reflex

Grants the permission in every namespace, even if the app touches one.

Fix: bind the ClusterRole with a plain RoleBinding to stay namespace-scoped.

2 • Wildcards

`verbs: ["*"]` or `resources: ["*"]` means anything it matches — delete included.

Fix: list the exact verbs and resources the app uses.

3 • The default service account

Pods with no service account use default. A broad binding on it is inherited by every such pod.

Fix: give each workload its own service account.

Keep next to your terminal

```
# Can this identity do it?
kubectl auth can-i list pods -n billing --as=system:serviceaccount:billing:billing-reader
kubectl auth can-i --list -n billing --as=system:serviceaccount:billing:billing-reader
# Who has what?
kubectl get rolebindings,clusterrolebindings -A
kubectl create role pod-reader --verb=get,list,watch --resource=pods -n billing
kubectl create rolebinding read-pods --role=pod-reader --serviceaccount=billing:billing-reader -n billing
```

When a request is denied

Don't reach for a broader Role. Ask which of **subject**, **verb**, **resource** is missing — and grant exactly that.

The habit

Every new ClusterRoleBinding gets a one-line reason: why this subject, why this scope.