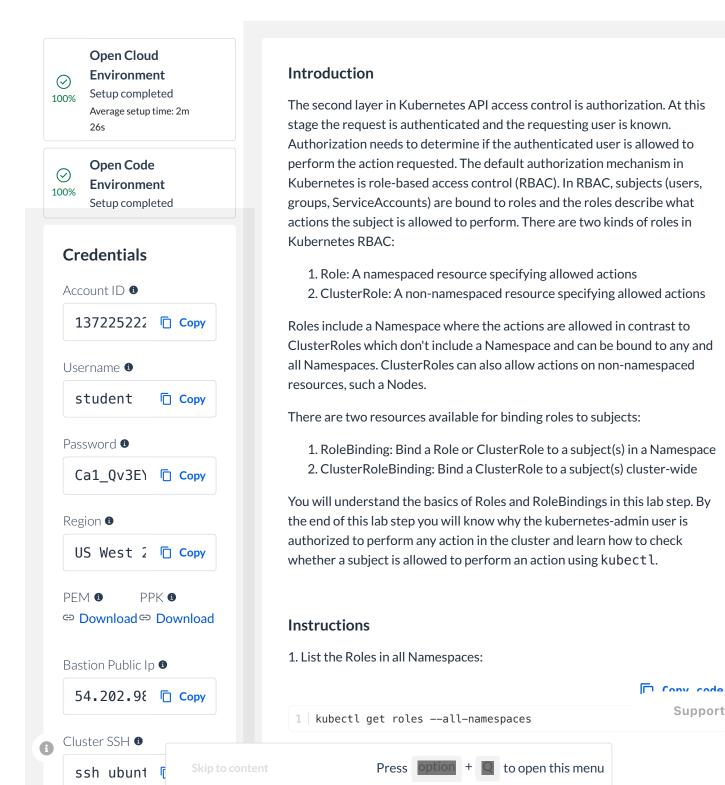




Training Library / Understand Kubernetes API Access Control Mechanisms

Understanding Kubernetes Authorization

56m 18s left













命







Lab Steps

- Connecting to the Kubernetes Cluster
- Understanding Kubernetes Authentication
- **Understanding** 3 **Kubernetes Authorization**
- **Understanding** Kubernetes **Admission Control**
- Need help? Contact our support team

kube-system kube-proxy

The Roles all have an associated Namespace in the first column. It is a best practice to follow the principle of least privilege and ensure subjects don't have more access than they need. The roles in the list are related to specific applications/controllers and provide the least access required to perform their responsibilities.

2. View the Role object for the kube-proxy Role in the kube-system Namespace:



1 | kubectl get -n kube-system role kube-proxy -o yaml

The rules map contains the allowed actions:



Rules follow the same structure as the example above. The example allows read (get) access to ConfigMaps (configmaps) named kube-proxy. The verbs declare which HTTP verbs are allowed for requests. The Kubernetes API is organized into groups and the apiGroups list indicates which API group(s) the rule applies to. The core API group which includes the most commonly used resources, including ConfigMaps, is denoted by an empty string ("").

The rules for ClusterRoles follow the same structure.

3. List all of the cluster roles:



1 kubectl get clusterroles

NAME admin calico-cni-plugin calico-kube-controllers alico-node







Press option + Q to open this menu

















4. Show the cluster-admin ClusterRole resource YAML:

```
Copy code
```

1 | kubectl get clusterrole cluster-admin -o yaml

```
rules:
  apiGroups:
  _ '*'
  resources:
  _ '*'
  verbs:
  _ '*'
  nonResourceURLs:
  _ '*'
  verbs:
   '*'
```

The rules key allows all actions (verbs) on all resources. Access to any nonresource URLs (nonResourceURLs) is also allowed providing full access to the cluster. You can also get the same information from kubectl describe clusterrole cluster-admin.

5. Describe the cluster-admin ClusterRoleBinding to understand how the cluster-admin ClusterRole is bound to users:



kubectl get clusterrolebinding cluster-admin -o yaml

```
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
  apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:masters
```

The **RoleRef** map specifies the **name** of the ClusterRole that is being bound, and the **subjects** map lists all the subjects (users, groups, or service accounts) that are bound to the ClusterRole. In this case, the ClusterRole is bound to a Group named system:masters. Because identities are managed outside of Kubernetes, you cannot use kubectl to show details of users or groups. However, recall that the client certificate used in the kubeconfig identifies the user as kubernetes-admin and the group as system:masters. Because the









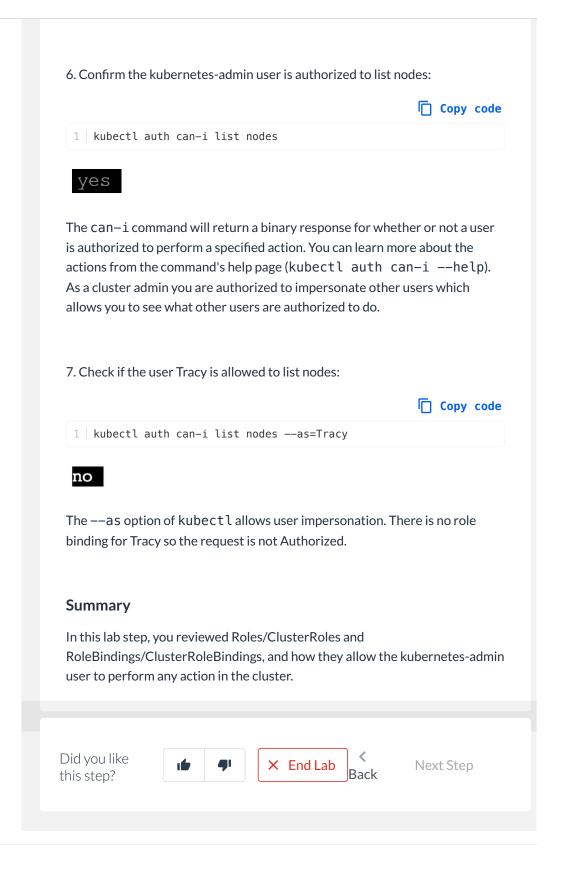


































About QA

About Circus Street

COMMUNITY

Join Discord Channel

HELP

Help Center

Copyright $\hbox{@\,}2024$ Cloud Academy Inc. All rights reserved.

Terms and Conditions

Privacy Policy

Sitemap

System Status

Manage your cookies