Lesson 3 Lesson-End Project

**Managing Multiple Namespaces with Respective Roles**

**Project agenda:** To manage multiple namespaces with respective roles

**Description:** Your project requires two namespaces to configure as given below:

**Namespace Username Access**

simplilearn user1 edit

user2 view

cka user1 view

user4 admin

**Tools required:** kubeadm, kubectl, kubelet, and docker

**Prerequisites:** kubeadm, kubectl, kubelet, and docker should be installed

**Expected deliverables:** A Kubernetes cluster with high availability enabled

Steps to be followed:

1. Creating a Kubernetes cluster
2. Connecting to a cluster and then creating a Simplilearn namespace with user1 and user2
3. Connecting to a cluster and then creating a CKA namespace with user1 and user4

**Step** **1**: **Creating a Kubernetes cluster**

|  |
| --- |
| **Note:** Refer Demo 1 of Lesson 2 to create a Kubernetes cluster |

**Step 2: Connecting to a cluster and then creating a Simplilearn namespace with user1 and user2**

1. Connect to the Kubernetes cluster and use **KUBECONFIG** for the storage settings in the home directory:

**KUBECONFIG=~/.kube/config**

**kubectl get node**

A screenshot of a computer

Description automatically generated

1. Create the Simplilearn namespace using the following command:

**kubectl create namespace simplilearn**

**kubectl get namespaces**

A screenshot of a computer

Description automatically generated

1. Create **user1** and **user2** using the following code:

**vi user.yaml**

**apiVersion: v1**

**kind: ServiceAccount**

**metadata:**

**name: user1**

**namespace: simplilearn**

**---**

**apiVersion: v1**

**kind: ServiceAccount**

**metadata:**

**name: user2**

**namespace: simplilearn**

Graphical user interface, text, application

Description automatically generated

1. Create the service accounts, and check their status using the following commands:

**kubectl apply -f user.yaml**

**kubectl get sa -n simplilearn**

**A screenshot of a computer

Description automatically generated**

1. To create roles for user1 and user2, add the following code to the **role.yaml** file:

**vi role.yaml**

**kind: Role**

**apiVersion: rbac.authorization.k8s.io/v1**

**metadata:**

**namespace: simplilearn**

**name: user1-role**

**rules:**

**- apiGroups: ["", "extensions", "apps"]**

**resources: ["\*"]**

**verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]**

**---**

**kind: Role**

**apiVersion: rbac.authorization.k8s.io/v1**

**metadata:**

**namespace: simplilearn**

**name: user2-role**

**rules:**

**- apiGroups: ["", "extensions", "apps"]**

**resources: ["\*"]**

**verbs: ["get", "list", "watch"]**

**Text

Description automatically generated**

1. Create the roles, and check their status using the following commands:

**kubectl apply -f role.yaml**

**kubectl get role -n simplilearn**

A screenshot of a computer

Description automatically generated

1. Once the role is created, use the commands given below to bind the service account and role to assign permissions:

**vi rolebinding.yaml**

**kind: RoleBinding**

**apiVersion: rbac.authorization.k8s.io/v1**

**metadata:**

**name: user1-binding**

**namespace: simplilearn**

**subjects:**

**- kind: User**

**name: user1**

**apiGroup: ""**

**roleRef:**

**kind: Role**

**name: user1-role**

**apiGroup: ""**

**---**

**kind: RoleBinding**

**apiVersion: rbac.authorization.k8s.io/v1**

**metadata:**

**name: user2-binding**

**namespace: simplilearn**

**subjects:**

**- kind: User**

**name: user2**

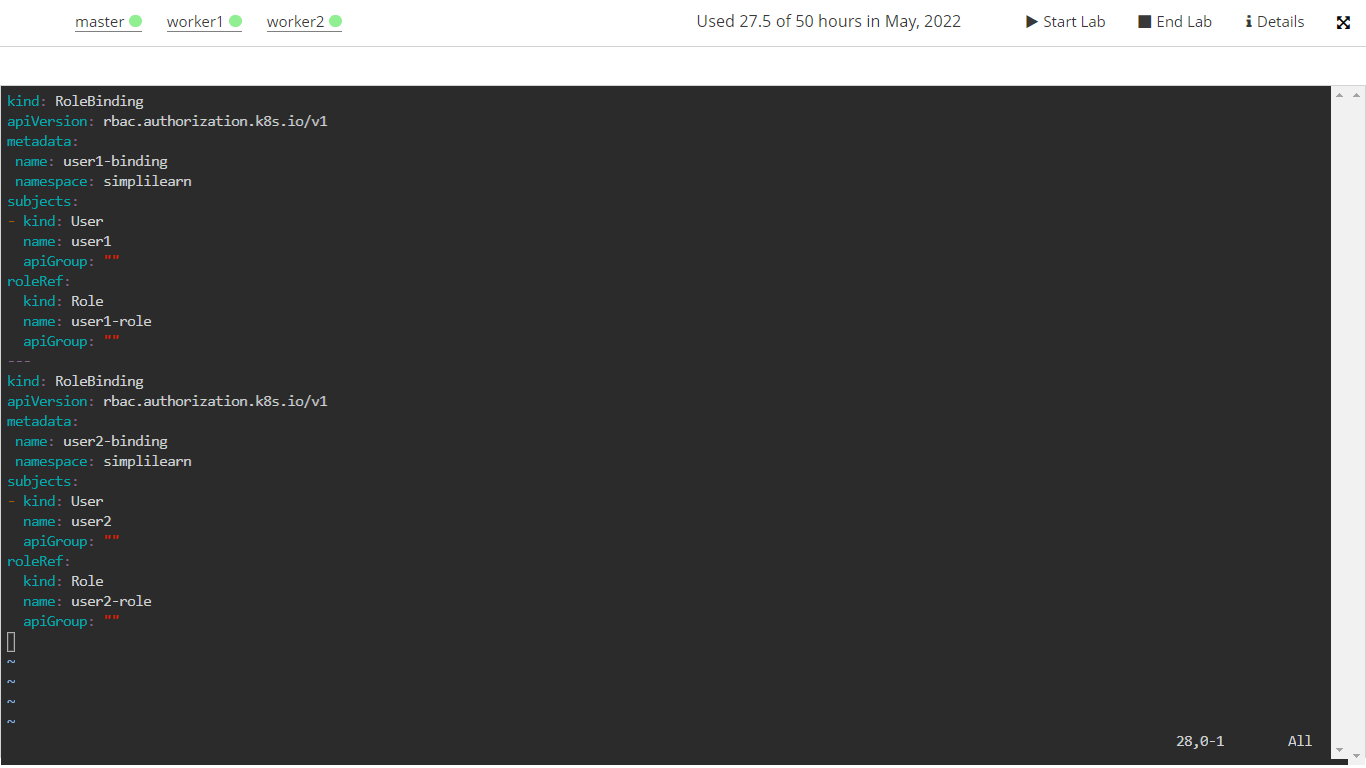
**apiGroup: ""**

**roleRef:**

**kind: Role**

**name: user2-role**

**apiGroup: ""**



1. Create **rolebindings**, and check their status using the following commands:

**kubectl apply -f rolebinding.yaml**

**kubectl get rolebinding -n simplilearn**

**A screenshot of a computer

Description automatically generated**

**Step 3: Connecting to a cluster and then creating a CKA namespace with user1 and user4**

1. Connect to the Kubernetes cluster, and use **KUBECONFIG** for the storage settings in the home directory:

**KUBECONFIG=~/.kube/config**

**kubectl get node**

A screenshot of a computer

Description automatically generated

1. Create the CKA namespace using the following command:

**kubectl create namespace cka**

**kubectl get namespaces**

Text

Description automatically generated

1. Create user1 and user4 using the following code:

**vi cka-sa.yaml**

**apiVersion: v1**

**kind: ServiceAccount**

**metadata:**

**name: user1**

**namespace: cka**

**---**

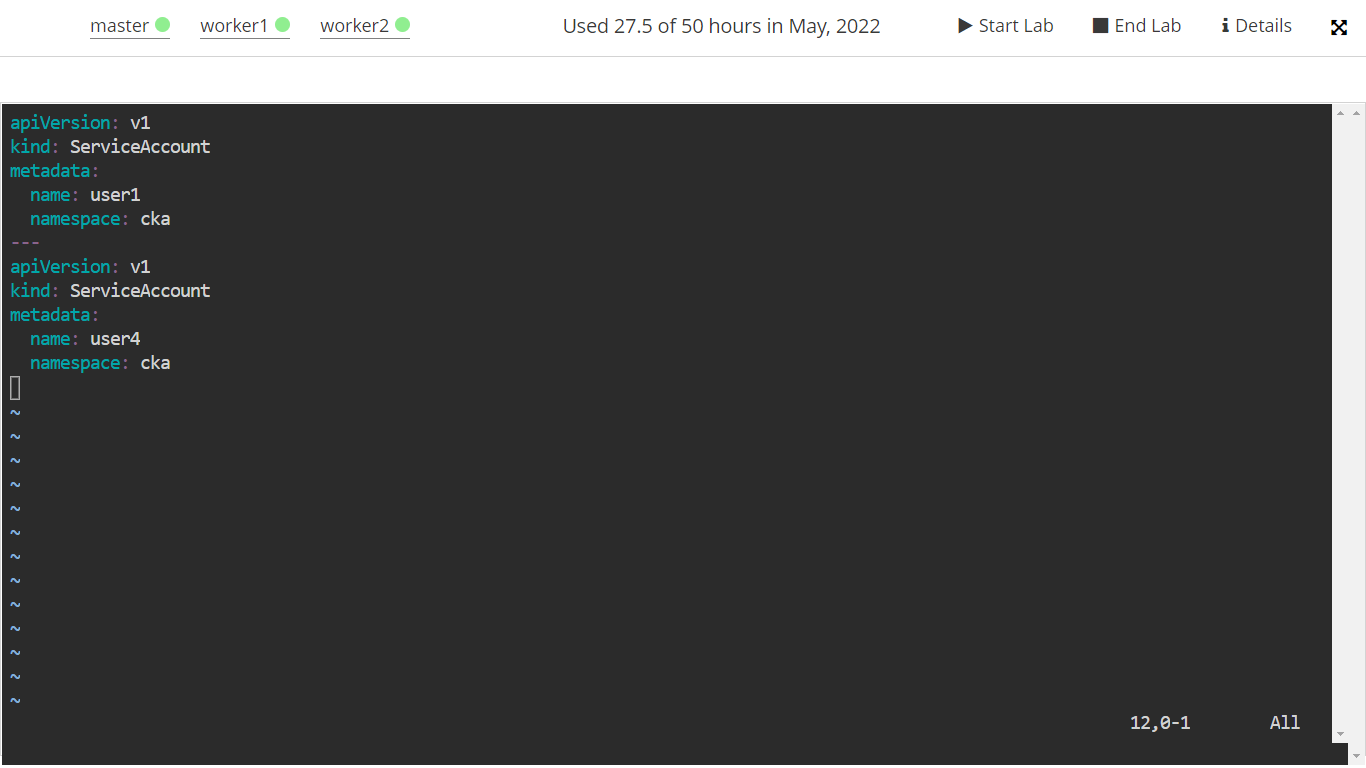
**apiVersion: v1**

**kind: ServiceAccount**

**metadata:**

**name: user4**

**namespace: cka**

****

1. Create the service accounts, and check their status using the following commands:

**kubectl apply -f cka-sa.yaml**

**kubectl get sa -n cka**

**A screenshot of a computer

Description automatically generated**

1. To create the roles for **user1** and **user4**, add the following code to the **cka-role.yaml** file:

**vi cka-role.yaml**

**kind: Role**

**apiVersion: rbac.authorization.k8s.io/v1**

**metadata:**

**namespace: cka**

**name: user1-role**

**rules:**

**- apiGroups: ["", "extensions", "apps"]**

**resources: ["\*"]**

**verbs: ["get", "list", "watch"]**

**---**

**kind: Role**

**apiVersion: rbac.authorization.k8s.io/v1**

**metadata:**

**namespace: cka**

**name: user4-role**

**rules:**

**- apiGroups: ["", "extensions", "apps"]**

**resources: ["\*"]**

**verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]**

**Text

Description automatically generated**

1. Create the roles, and check their status using the following commands:

**kubectl apply -f cka-role.yaml**

**kubectl get roles -n cka**

**A screenshot of a computer

Description automatically generated**

1. Once the roles are created, use the commands given below to bind the service account and role to assign permissions:

**vi cka-binding.yaml**

**kind: RoleBinding**

**apiVersion: rbac.authorization.k8s.io/v1**

**metadata:**

**name: user1-binding**

**namespace: cka**

**subjects:**

**- kind: User**

**name: user1**

**apiGroup: ""**

**roleRef:**

**kind: Role**

**name: user1-role**

**apiGroup: ""**

**---**

**kind: RoleBinding**

**apiVersion: rbac.authorization.k8s.io/v1**

**metadata:**

**name: user4-binding**

**namespace: cka**

**subjects:**

**- kind: User**

**name: user4**

**apiGroup: ""**

**roleRef:**

**kind: Role**

**name: user4-role**

**apiGroup: ""**

**Text

Description automatically generated**

1. Create **rolebindings**, and check their status using the following commands:

**kubectl apply -f cka-binding.yaml**

**kubectl get rolebinding -n cka**

**A screenshot of a computer

Description automatically generated**