## Roles and Role Bindings

CRUD - Create, Read(one item & all items), Update, Delete

K8s Roles - ["create", "list", "get", "update", "delete"]

K8s Cmd -> k8s role

Create/run - create

get - list

Describe - get

Edit - update

Delete - delete

k create ns test

# Role and RB - Imperative - Kubectl commands

kubectl create role developer --verb=get,list,watch --resource=pods -n test

kubectl create rolebinding developer-binding-swami --role=developer --user=swami -n test

k get roles -n test

# Pod - Will work

k get pods -n test --as swami

# Nodes - Will NOT work

k get nodes -n test --as swami

## Cluster Role & Role Binding

cat ~/.kube/config

k describe clusterrole cluster-admin

Cluster Roles are non-namespaced

You can use a ClusterRole to:

1. define permissions on namespaced resources and be granted within individual namespace(s)
2. define permissions on namespaced resources and be granted across all namespaces
3. define permissions on cluster-scoped resources - Nodes, pv

k get clusterroles

k get clusterrolebindings

k get clusterrole cluster-admin -o yaml

# Imperative - Kubectl commands

kubectl create clusterrole nodes-viewer --verb=get,list,watch --resource=nodes

kubectl create clusterrolebinding nodes-viewer-binding-swami --clusterrole=nodes-viewer --user=swami

k get nodes --as swami

## RBAC with ServiceAccount:

Steps to be followed:

1. Set up the simplilearn namespace and Service Accounts

2. Check API access as ServiceAccounts in simplilearn namespace

**Step 1: Set up the simplilearn namespace and users**

Create the **simplilearn** namespace and verify its creation using the following commands:

**kubectl create namespace simplilearn**

**kubectl get namespaces**

Define two service accounts, **user1** and **user2**, for the **simplilearn** namespace. Save the following configurations in a file called **user.yaml**:

**vi user.yaml**

apiVersion: v1

kind: ServiceAccount

metadata:

name: user1

namespace: simplilearn

---

apiVersion: v1

kind: ServiceAccount

metadata:

name: user2

namespace: simplilearn

Create the service accounts and validate their creation using the following commands:

**kubectl apply -f user.yaml**

**kubectl get sa -n simplilearn**

Utilise the provided code to establish roles for **user1** and **user2**, then save the configuration in a file named **role.yaml.**

**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"]

Apply the roles and verify their status with the following commands:

**kubectl apply -f role.yaml**

**kubectl get role -n simplilearn**

After creating the roles, bind the service account to the role using the provided code. Save this configuration in a file named **rolebinding.yaml**:

**vi rolebinding.yaml**

kind: RoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: user1-binding

namespace: simplilearn

subjects:

- kind: ServiceAccount

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: ServiceAccount

name: user2

apiGroup: ""

roleRef:

kind: Role

name: user2-role

apiGroup: ""

Apply the role bindings and verify their status with these commands:

**kubectl apply -f rolebinding.yaml**

**kubectl get rolebinding -n simplilearn**

**Step 3:** Check API access as ServiceAccounts in simplilearn namespace

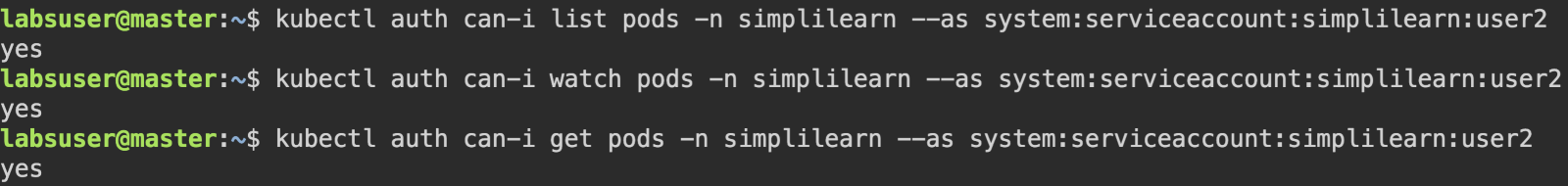
kubectl provides the auth can-i subcommand for quickly querying the API authorization layer.

3.1 Check API access (list, get, watch) for serviceaccount provided in RoleBinding

**kubectl auth can-i list pods -n simplilearn --as system:serviceaccount:simplilearn:user2**   
yes

**kubectl auth can-i watch pods -n simplilearn --as system:serviceaccount:simplilearn:user2**   
yes

**kubectl auth can-i get pods -n simplilearn --as system:serviceaccount:simplilearn:user2**   
yes



3.2 Check API access (other than list, get, watch) for serviceaccount, which are NOT provided in RoleBinding

**kubectl auth can-i delete pods -n simplilearn --as system:serviceaccount:simplilearn:user2**   
no

**kubectl auth can-i create pods -n simplilearn --as system:serviceaccount:simplilearn:user2**   
no

**kubectl auth can-i update pods -n simplilearn --as system:serviceaccount:simplilearn:user2**   
no

**kubectl auth can-i create deployment -n simplilearn --as system:serviceaccount:simplilearn:user2**   
no

