## Introduction

Role-based access control is an important component when it comes to managing a Kubernetes cluster securely. The more users and automated processes there are that need to interface with the Kubernetes API, the more important controlling access becomes. In this lab, you will have the opportunity to practice your skills with the Kubernetes RBAC system by implementing your own RBAC permissions to appropriately limit user access.

## Solution

Log in to the lab server using the credentials provided:

ssh cloud\_user@<PUBLIC\_IP\_ADDRESS>

Note: When copying and pasting code into Vim from the lab guide, first enter :set paste (and then i to enter insert mode) to avoid adding unnecessary spaces and hashes.

### Create a Role for the dev User

1. Test access by attempting to list pods as the dev user:

kubectl get pods -n beebox-mobile --kubeconfig dev-k8s-config

We'll get an error message.

1. Create a role spec file:

vi pod-reader-role.yml

1. Add the following to the file:

apiVersion: rbac.authorization.k8s.io/v1

kind: Role

metadata:

namespace: beebox-mobile

name: pod-reader

rules:

- apiGroups: [""]

resources: ["pods", "pods/log"]

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

1. Save and exit the file by pressing Escape followed by :wq.
2. Create the role:

kubectl apply -f pod-reader-role.yml

### Bind the Role to the dev User and Verify Your Setup Works

1. Create the RoleBinding spec file:

vi pod-reader-rolebinding.yml

1. Add the following to the file:

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

name: pod-reader

namespace: beebox-mobile

subjects:

- kind: User

name: dev

apiGroup: rbac.authorization.k8s.io

roleRef:

kind: Role

name: pod-reader

apiGroup: rbac.authorization.k8s.io

1. Save and exit the file by pressing Escape followed by :wq.
2. Create the RoleBinding:

kubectl apply -f pod-reader-rolebinding.yml

1. Test access again to verify you can successfully list pods:

kubectl get pods -n beebox-mobile --kubeconfig dev-k8s-config

This time, we should see a list of pods (there's just one).

1. Verify the dev user can read pod logs:

kubectl logs beebox-auth -n beebox-mobile --kubeconfig dev-k8s-config

We'll get an Auth processing... message.

1. Verify the dev user cannot make changes by attempting to delete a pod:

kubectl delete pod beebox-auth -n beebox-mobile --kubeconfig dev-k8s-config

We'll get an error, which is what we want.

## Conclusion

Congratulations on successfully completing this hands-on lab!