**Lesson 3 Demo 8**

**RBAC Using Namespace**

**Objective:** To implement a Role-based access control (RBAC) using namespace

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

**Prerequisites:** A Kubernetes cluster must be set up (follow steps of Lesson 2 Demo 1)

Steps to be followed:

1. Creating a namespace
2. Generating an RSA private key and certificate requests
3. Creating a role
4. Creating a rolebinding
5. Setting credentials to the user
6. Copying the config file to the client machine
7. Verifying the roles

**Step 1: Creating a namespace**

1. Create a namespace by using the following command:

**kubectl create namespace role**

**Text

Description automatically generated**

1. Create a directory **role**.

**mkdir role**

**cd role**

**Graphical user interface

Description automatically generated**

**Step 2: Generating an RSA private key and certificate requests**

1. To generate an RSA private key, run the following command:

**sudo openssl genrsa -out user3.key 2048**

**A screenshot of a computer

Description automatically generated with medium confidence**

1. Use the following command to generate certificate requests:

**sudo openssl req -new -key user3.key -out user3.csr**

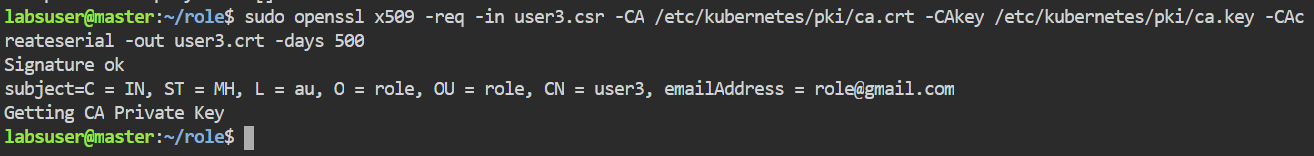
**Text

Description automatically generated**

* Organization Name: namespace
* Common Name: user3

1. Run the following command to link an identity to a private key using a digital signature.

**sudo openssl x509 -req -in user3.csr -CA /etc/kubernetes/pki/ca.crt -CAkey /etc/kubernetes/pki/ca.key -CAcreateserial -out user3.crt -days 500**

****

**Step 3: Creating role**

1. To create a role, add the following code to the **role.yaml** file.

**kind: Role**

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

**metadata:**

**namespace: role**

**name: user3-role**

**rules:**

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

**resources: ["deployments", "pods", "services"]**

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

**Text

Description automatically generated**

1. Create a role by using the following command:

**kubectl create -f role.yaml**

**kubectl get roles -n role**

Text

Description automatically generated

**Step 4: Creating a rolebinding**

1. To create a **rolebinding**, add the following code to the **rolebinding.yaml** file.

**kind: RoleBinding**

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

**metadata:**

**name: role-test**

**namespace: role**

**subjects:**

**- kind: User**

**name: user3**

**apiGroup: ""**

**roleRef:**

**kind: Role**

**name: user3-role**

**apiGroup: ""**

**Text

Description automatically generated**

1. Create **rolebinding** by using the following command:

**kubectl create -f rolebinding.yaml**

**kubectl get rolebinding -n role**

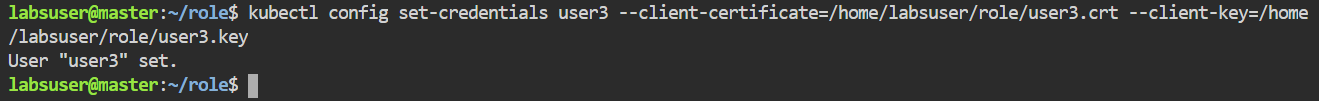
Text

Description automatically generated

**Step 5: Setting credentials to the user**

1. Set credentials to user3.

**kubectl config set-credentials user3 --client-certificate=/home/labsuser/role/user3.crt --client-key=/home/labsuser/role/user3.key**

****

1. Set context to user3.

**kubectl config set-context user3-context --cluster=kubernetes --namespace=role --user=user3**



1. Run the following command to display current contexts:

**kubectl config get-contexts**

Text

Description automatically generated

**Step 6: Copying the config file to the client machine**

1. Copy the config file from the master node in the home directory.

**cd ..**

**cat .kube/config**

**Text

Description automatically generated**

1. Paste the copied config file into the client machine.

**vi myconf**

**Text

Description automatically generated**

**Text

Description automatically generated**

1. Copy the **crt** and **key** files from the master node to the client node in the **/role** directory.

Graphical user interface, text, application

Description automatically generated

**mkdir role**

**cd role**

**vi user3.crt**

**A screenshot of a computer

Description automatically generated with medium confidence**

**vi user3.key**

**Text

Description automatically generated**

**Step 7: Verifying roles**

1. Locate the home directory.

**cd ..**

1. Run the following commands to verify roles we have generated:

**kubectl get pods --kubeconfig=myconf**

**kubectl create deployment test --image=docker.io/httpd -n role --kubeconfig=myconf**

**kubectl get pods --kubeconfig=myconf**

**kubectl get deployment --kubeconfig=myconf**

**A screenshot of a computer

Description automatically generated with medium confidence**

**Text

Description automatically generated**

The worker node can create, update, remove, and list pods, services, and deployments after using the master config settings, as seen in the above screenshots.