**Container Orchestration using Kubernetes.**

**PROJECT II – Backing up the Etcd Cluster Data**

**BY**

**JAYAMURGAN SEKAR**

**Master of DevOps Engineer**

**SIMPLI LEARN**

**Backing up the Etcd Cluster Data .**

Course-end Project 2

DESCRIPTION

**Project Agenda:** To take the backup of an etcd cluster in a file

**Description:**

As an infrastructure admin, you need to take the backup of an etcd in a file called /tmp/myback. Make sure to have a namespace called cep-project2 with a network policy configured in such a way that all the Pods in the same namespace should access each other. Any other Pods from the non cep-project2 should not access the Pods. Configure a Kubernetes client on worker node 3 in such a way that user4 should have only view access to cep-project2. Update the master with the latest version of the Kubernetes.

**Tools Required:** kubeadm, kubectl, kubelet, and Docker

**Expected Deliverables:**

* Creating and verifying the namespaces
* Generating a certificate and private key in the worker node
* Upgrading the Kubernetes cluster with the latest version
* A Kubernetes cluster with high availability enabled

**Task Activities:**

1. Backing up the etcd cluster data

2. Creating and verifying the namespaces

3. Generating a certificate and private key in the worker node

4. Upgrading the Kubernetes cluster with the latest version

**Project Reference:**

Task 1: To back up the etcd cluster data, refer to the lesson 3; demo 9

Task 2: To create and verify the namespaces, refer to the lesson 3; demo 8

Task 3: To generate a certificate and private key in the worker node, refer to the lesson 3; demo 8

Task 4: To upgrade the Kubernetes cluster with the latest version, refer to the lesson 3; demo 11.

**Task 1: Backing up the etcd cluster data**

1.1Use the following command to install the etcd-client:

**sudo apt install etcd-client**

Text

Description automatically generated

1.2 List all the pods of the kube-system namespace

**kubectl get pods -n kube-system**

Text

Description automatically generated

1.3 Describe the etcd pod of the kube-system namespace and copy the IP address of the    --advertise-client-url flag:

**kubectl describe pods <etcd-pod-name> -n kube-system**

Text

Description automatically generated

1.4 Export the advertise-client-url to advertise\_url

1.4 Export the advertise-client-url to advertise\_url

**export advertise\_url=<<advertise-client-url>>**

**echo $advertise\_url**

**Note:** Replace <<advertise-client-url>> with the advertise-client-url copied from Step 1.3

Text

Description automatically generated

1.5 Use the following command to save the etcd backup:

**sudo ETCDCTL\_API=3 etcdctl \**

**--endpoints $advertise\_url \**

**--cacert /etc/kubernetes/pki/etcd/ca.crt \**

**--key /etc/kubernetes/pki/etcd/server.key \**

**--cert /etc/kubernetes/pki/etcd/server.crt snapshot save myback**

1.6 Use the following command to check the newly created **myback** file:

**ls**

Graphical user interface, text

Description automatically generated

**Task 2: Creating and verifying the namespaces**

* 1. Create a namespace by using the following command:

**kubectl create namespace cep-project2**

* 1. Create a directory **cep-project2**.

**mkdir cep-project2**

**cd cep-project2**

Text

Description automatically generated

**Task 3: Generating a certificate and private key in the worker node**

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

**sudo openssl genrsa -out user4.key 2048**

Graphical user interface, text

Description automatically generated

* 1. Use the following command to generate certificate requests:

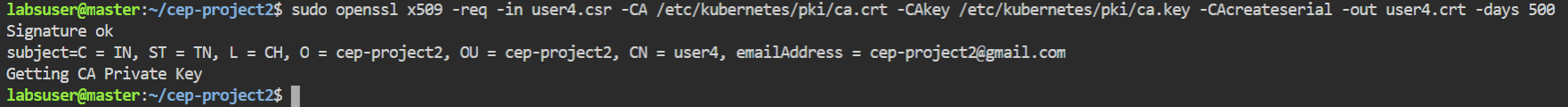
**sudo openssl req -new -key user4.key -out user4.csr**

Text

Description automatically generated

* Organization Name: namespace
* Common Name: user4
  1. Run the following command to link an identity to a private key using a digital signature.

**sudo openssl x509 -req -in user4.csr -CA /etc/kubernetes/pki/ca.crt -CAkey /etc/kubernetes/pki/ca.key -CAcreateserial -out user4.crt -days 500**



3.4: Creating role

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

**kind: Role**

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

**metadata:**

**namespace: cep-project2**

**name: user4-role**

**rules:**

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

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

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

* + 1. Create a role by using the following command:

**kubectl create -f role.yaml**

**kubectl get roles -n cep-project2**

Text

Description automatically generated

3.4.3 Creating a rolebinding

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: cep-project2**

**subjects:**

**- kind: User**

**name: user4**

**apiGroup: ""**

**roleRef:**

**kind: Role**

**name: user4-role**

**apiGroup: ""**

Text

Description automatically generated

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

**kubectl create -f rolebinding.yaml**

**kubectl get rolebinding -n cep-project2**

**Text

Description automatically generated**

* + 1. Setting credentials to the user

Set credentials to user4.

**kubectl config set-credentials user4 --client-certificate=/home/labsuser/cep-project2/user4.crt --client-key=/home/labsuser/cep-project2/user4.key**

****

Set context to user4.

**kubectl config set-context user4-context --cluster=kubernetes --namespace= cep-project2 --user=user4**

****

**sudo chown labsuser /home/labsuser/ cep-project2/user3.crt**

**sudo chown labsuser /home/labsuser/ cep-project2/user3.key**

Run the following command to display current contexts:

**kubectl config get-contexts**

**Graphical user interface, text

Description automatically generated**

**kubectl config use-context user4-context**

3.4.5: Copying the config file to the client machine

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

**cd ..**

**cat .kube/config**

**Text

Description automatically generated**

Paste the copied config file into the client machine.

**vi myconf**

Text

Description automatically generated

Copy the **crt** and **key** files from the master node to the client node in the **/cep-project2** directory.

**Graphical user interface, text

Description automatically generated**

**mkdir cep-project2**

**cd cep-project2**

**vi user4.crt**

**cat user4.crt**

**vi user4.key**

**cat user4.key**

**user this command if permission is denied in master node to change the ownership while copying the key.**

**sudo chmod 777 user3.key**

**Text

Description automatically generated**

**Text

Description automatically generated**

**FINAL OUTPUT IMAGE:**

**Text

Description automatically generated**

**Output has been generated has mentioned in the course-end project guideline pdf**

**Task 4: Upgrading the Kubernetes cluster with the latest version**

**4.1: Determining which version to upgrade**

Check which version to upgrade using the following command:

**sudo apt update**

Text

Description automatically generated

**4.2** **Finding the latest release of Kubernetes**

Find the latest patch release of kubeadm 1.23 using the OS package manager:

**sudo apt-cache madison kubeadm**

A picture containing text, window

Description automatically generated

Find the latest 1.23 version in the list. It should look like 1.23.x-00, where x is the latest patch.

If the kubeadm version is 1.23.5 or lower, we are going to upgrade it to 1.23.6 (x=6)

Find the latest patch release of kubectl 1.23 using the OS package manager:

**sudo apt-cache madison kubectl**

**A picture containing text, window, computer

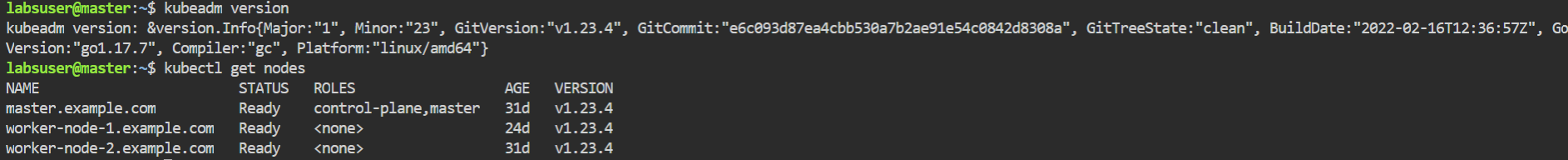
Description automatically generated**

**4.3 Verifying the current version of Kubernetes**

Verify the kubeadm and kubectl versions of the control plane (master):

**kubeadm version**

**kubectl get nodes**

****

**4.4: Upgrading the repositories**

Update and upgrade the repositories using the following commands:

**sudo apt update**

**sudo apt upgrade**

**4.5: Holding the Kubernetes versions**

Use the following commands to hold the kubeadm, kubectl, and kubelet versions:

**sudo apt-mark hold kubeadm**

**sudo apt-mark hold kubelet kubectl**

**4.6: Upgrading the control plane**

Use the following commands to upgrade the control plane (master):

**sudo apt-get install -y kubeadm=1.23.6-00 --allow-change-held-packages**

**sudo apt-get install -y kubelet=1.23.6-00 kubectl=1.23.6-00 --allow-change-held-packages**

**4.7: Verifying the updated version of Kubernetes**

1. Verify the updated version by using the following command:

**kubeadm version**

**kubectl get nodes**

**sudo kubeadm upgrade plan**

A screenshot of a computer

Description automatically generated