

Backing up the Etcd Cluster Data

Course-end Project 2

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

Task 1. Backing up the Etcd Cluster Data

Step1:we need to install etcd-client

```
labsuser@master:~/cluster-admin1$ etcdctl
sudo snap install etcd
sudo apt install etcd-client
sudo chmod a+r -R /etc/kubernetes/pki
```

Step 2: backup data

```
labsuser@master:~/cluster-admin1$ ETCDCTL_API=3 etcdctl
--endpoints=https://172.31.31.207:2379 \
--cert=/etc/kubernetes/pki/etcd/server.crt \
--key=/etc/kubernetes/pki/etcd/server.key \
--cacert=/etc/kubernetes/pki/etcd/ca.crt \
snapshot save /tmp/myback
2024-02-12 00:49:04.001225 I | clientv3: opened snapshot stream; downloading
2024-02-12 00:49:04.430909 I | clientv3: completed snapshot read; closing
Snapshot saved at /tmp/myback
```

Step3 .Giving a permission

```
sudo chmod a+r -R /etc/kubernetes/pki
```

```
labsuser@master:~/cluster-admin1$ ETCDCTL_API=3 etcdctl --endpoints=https://172.31.31.207:2379 \
--cert=/etc/kubernetes/pki/etcd/server.crt \
--key=/etc/kubernetes/pki/etcd/server.key \
--cacert=/etc/kubernetes/pki/etcd/ca.crt \
snapshot save /tmp/myback
2024-02-12 00:49:04.001225 I | clientv3: opened snapshot stream; downloading
2024-02-12 00:49:04.430909 I | clientv3: completed snapshot read; closing
Snapshot saved at /tmp/myback
labsuser@master:~/cluster-admin1$
```

```
labsuser@master:~$ ls /tmp
myback
myback.db.part
```

```
Labsuser@master:~$ ls /tmp
dcv-pcsd-0
myback
myback.db.part
```

Step 5.Lets check a size

ls -lta

Verifying the snapshot

```
labsuser@master:~$ ETCDCTL_API=3 etcdctl --endpoints=https://172.31.31.207:2379 \
--cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key \
--cacert=/etc/kubernetes/pki/etcd/ca.crt snapshot status /tmp/myback
8e0e676c, 451960, 2058, 24 MB
```

```
Labsuser@master:~$ ETCDCTL_API=3 etcdctl --endpoints=https://172.31.31.207:2379 --cert=/etc/kubernetes/pki/etcd/ca.crt snapshot status /tmp/myback
8e0e676c, 451960, 2058, 24 MB
```

Task 2. Creating and verifying the namespaces

To create a namespace called cep-project2 with a network policy that allows all Pods in the same namespace to access one another, you can use Kubernetes YAML manifests. Below are the YAML manifests for creating the namespace and the network policy:

```
labsuser@master:~$ kubectl apply -f cep-project2-namespace.yaml  
namespace/cep-project2 unchanged
```

```
# cep-project2-namespace.yaml  
apiVersion: v1  
kind: Namespace  
metadata:  
  name: cep-project2
```

```
# cep-project2-network-policy.yaml  
apiVersion: networking.k8s.io/v1  
kind: NetworkPolicy  
metadata:  
  name: allow-intra-namespace  
  namespace: cep-project2  
spec:  
  podSelector: {}  
  policyTypes:  
    - Ingress  
    - Egress
```

```
kubectl apply -f cep-project2-namespace.yaml  
kubectl apply -f cep-project2-network-policy.yaml
```

```
labsuser@master:~$  
kubectl apply -f cep-project2-network-policy.yaml  
networkpolicy.networking.k8s.io/allow-intra-namespace created
```

To verify the creation of the namespace, you can run:

```
labsuser@master:~$ kubectl get namespaces
```

NAME	STATUS	AGE
alpha	Active	8h
cep-project2	Active	43m
default	Active	28d
ingress-nginx	Active	14d
kube-node-lease	Active	28d
kube-public	Active	28d
kube-system	Active	28d
kubernetes-dashboard	Active	28d
quotaz	Active	32h
simplilearn	Active	32h
test	Active	14d
vote	Active	16

```
labsuser@master:~$ kubectl get namespaces
NAME          STATUS   AGE
alpha          Active   8h
cep-project2  Active   43m
default        Active   28d
ingress-nginx Active   14d
kube-node-lease Active  28d
kube-public    Active   28d
kube-system    Active   28d
kubernetes-dashboard Active 28d
quotaz         Active   32h
simplilearn   Active   32h
test           Active   14d
vote           Active   16d
labsuser@master:~$
```

this will list all the namespaces in your Kubernetes cluster, and you should see cep-project2 among them.

To verify the creation of the network policy, you can run:

This command will list all the network policies in the cep-project2 namespace, and you should see allow-intra-namespace among them.

```
labsuser@master:~$ kubectl get networkpolicies -n cep-project2
NAME      POD-SELECTOR  AGE
```

```
allow-intra-namespace <none> 3m28s
```

```
labsuser@master:~$ kubectl get networkpolicies -n cep-project2
NAME          POD-SELECTOR   AGE
allow-intra-namespace <none> 3m28s
labsuser@master:~$
```

Task 3: Making 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

Now creating a NetworkPolicy to restrict access between Pods in the cep-project2 namespace and then configure RBAC (Role-Based Access Control) to provide limited access to the cep-project2 namespace for user4. Below are the steps to accomplish this:

1. Create a NetworkPolicy for the cep-project2 namespace to allow Pods within the namespace to communicate with each other:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-same-namespace
  namespace: cep-project2
spec:
  podSelector: {}
  policyTypes:
  - IntraNamespace
```

```
kubectl apply -f network-policy.yaml
```

Create a ClusterRole that grants view access to resources in the cep-project2 namespace:

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: view-cep-project2
rules:
- apiGroups: [""]
  resources: ["pods", "services", "configmaps", "secrets"]
  resourceNames: ["] # empty string means all resources
  verbs: ["get", "list", "watch"]
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: view-cep-project2-binding
subjects:
- kind: User
  name: user4
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: view-cep-project2
  apiGroup: rbac.authorization.k8s.io
```

```
labsuser@master:~$ vi cep-project2-namespace.yaml
labsuser@master:~$ kubectl apply -f cep-project2-namespace.yaml
Warning: resource namespaces/cep-project2 is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required by kubectl apply. kubectl apply should only be used on resources created declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be patched automatically.
namespace/cep-project2 configured
labsuser@master:~$ vi cep-project2-network-policy.yaml
labsuser@master:~$ kubectl apply -f cep-project2-namespace.yaml
namespace/cep-project2 unchanged
labsuser@master:~$ 
kubectl apply -f cep-project2-network-policy.yaml
networkpolicy.networking.k8s.io/allow-intra-namespace created
labsuser@master:~$ kubectl get namespaces
NAME          STATUS   AGE
alpha          Active   8h
ceph-project2 Active   43m
default        Active   28d
ingress-nginx  Active   14d
kube-node-lease Active   28d
kube-public    Active   28d
kube-system    Active   28d
kubernetes-dashboard Active   28d
quotaz         Active   32h
simpleilearn   Active   32h
test           Active   14d
vote           Active   16d
labsuser@master:~$ kubectl get networkpolicies -n cep-project2
NAME          POD-SELECTOR   AGE
allow-intra-namespace <none>       3m28s
labsuser@master:~$ ■
```

After applying these configurations, you can verify the NetworkPolicy and the RBAC settings using the following commands:

```
kubectl get networkpolicy -n cep-project2
```

```
labsuser@master:~$ kubectl get networkpolicy -n cep-project2
NAME          POD-SELECTOR   AGE
allow-intra-namespace <none>    23h
```

```
kubectl get clusterrole view-cep-project2
```

```
labsuser@master:~$ kubectl get clusterrole view-cep-project2
NAME      CREATED AT
view-cep-project2  2024-02-13T01:02:47Z
```

```
kubectl get clusterrolebinding view-cep-project2-binding
```

```
labsuser@master:~$ kubectl get clusterrolebinding view-cep-project2-binding
NAME        ROLE          AGE
view-cep-project2-binding  ClusterRole/view-cep-project2  84s
```

Verify NetworkPolicy:

verifying whether the NetworkPolicy is applied correctly and is restricting access between Pods in the cep-project2 namespace and Pods outside the namespace. To do this, create two test Pods, one within the cep-project2 namespace and another outside the namespace, and try to establish connections between them.

```
# Creating test Pod within cep-project2 namespace
```

```
kubectl run test-pod -n cep-project2 --image=busybox --restart=Never -- sleep 3600
```

```
# Creating test Pod outside cep-project2 namespace
```

```
kubectl run test-pod-outside -n default --image=busybox --restart=Never -- sleep 3600
```

```
# Try to establish a connection from test-pod to test-pod-outside
```

```
kubectl exec -n cep-project2 -it test-pod -- wget -qO- http://test-pod-outside.default
```

```
view-cep-project2-binding  ClusterRole/view-cep-project2  845
labsuser@master:~$ kubectl run test-pod -n cep-project2 --image=busybox --restart=Never -- sleep 3600
pod/test-pod created
labsuser@master:~$ kubectl run test-pod-outside -n default --image=busybox --restart=Never -- sleep 3600
pod/test-pod-outside created
labsuser@master:~$ kubectl exec -n cep-project2 -it test-pod -- wget -qO- http://test-pod-outside.default
wget: bad address 'test-pod-outside.default'
command terminated with exit code 1
```

Verify RBAC:

To verify RBAC settings, you can try accessing resources within the cep-project2 namespace using user4 credentials.

```
# Switch to user4 context (assuming you have kubeconfig with user4 credentials)
kubectl config use-context user4-context

# List Pods within cep-project2 namespace
kubectl get pods -n cep-project2
```

```
labsuser@master:~$ kubectl get pods -n cep-project2
NAME      READY   STATUS    RESTARTS   AGE
test-pod  1/1     Running   0          109s
labsuser@master:~$ █
```

Task 4. Upgrading the Kubernetes cluster with the latest version

Steps to do:

- 1.Upgrade control plane to v1.28.6
- 2.Upgrade worker node to v1.28.6

3.Validate upgrade, by creating a pod

Step 1: Upgrade control plane to v1.28.5

First step is you need to find what is the version

```
labsuser@master:~$ sudo apt update
```

```
sudo apt-cache madison kubeadm
```

```
labsuser@master:~$ sudo apt update
sudo apt-cache madison kubeadm
Hit:1 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:4 https://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Hit:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb InRelease
Get:6 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1366 kB]
Get:7 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [272 kB]
Get:8 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [1412 kB]
Get:9 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [233 kB]
Get:10 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1043 kB]
Get:11 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [235 kB]
Get:12 https://ppa.launchpadcontent.net/mozillateam/ppa/ubuntu jammy InRelease [23.8 kB]
Get:13 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [24.3 kB]
Get:14 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1142 kB]
Get:15 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [211 kB]
Get:16 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [1366 kB]
Get:17 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [224 kB]
Get:18 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [839 kB]
Get:19 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [160 kB]
Get:20 https://ppa.launchpadcontent.net/mozillateam/ppa/ubuntu jammy/main amd64 Packages [34.2 kB]
Fetched 8922 kB in 3s (3440 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
74 packages can be upgraded. Run 'apt list --upgradable' to see them.
  kubeadm | 1.28.6-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb Packages
  kubeadm | 1.28.5-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb Packages
  kubeadm | 1.28.4-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb Packages
  kubeadm | 1.28.3-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb Packages
  kubeadm | 1.28.2-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb Packages
  kubeadm | 1.28.1-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb Packages
  kubeadm | 1.28.0-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb Packages
labsuser@master:~$
```

This is listed all available versions

We are going to select latest version

kubeadm | 1.28.6-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb Packages

Upgrading control plane

1.Upgrade kubeadm:

```
sudo apt-mark unhold kubeadm && \
sudo apt-get update &&
```

```
sudo apt-get install -y kubeadm='1.28.6-1.1' && \
sudo apt-mark hold kubeadm
```

```
labsuser@master:~$ sudo apt-mark unhold kubeadm && \
sudo apt-get update &&
sudo apt-get install -y kubeadm='1.28.6-1.1' && \
sudo apt-mark hold kubeadm
Canceled hold on kubeadm.
Hit:1 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 https://prod-cdn.packages.k8s.io/repositories/1sv:/kubernetes:/core:/stable:/v1.28/deb InRelease
Hit:6 https://ppa.launchpadcontent.net/mozillateam/ppa/ubuntu jammy InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
kubeadm is already the newest version (1.28.6-1.1).
0 upgraded, 0 newly installed, 0 to remove and 73 not upgraded.
kubeadm set on hold.
```

2.Verify that the download works and has the expected version:

```
labsuser@master:~$ kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"28", GitVersion:"v1.28.6",
GitCommit:"be3af46a4654bdf05b4838fe94e95ec8c165660c", GitTreeState:"clean",
BuildDate:"2024-01-17T13:47:00Z", GoVersion:"go1.20.13", Compiler:"gc",
Platform:"linux/amd64"}
```

```
labsuser@master:~$ kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"28", GitVersion:"v1.28.6", GitCommit:"be3af46a4654bdf05b4838fe94e95ec8c165660c", GitTreeState:"clean", BuildDate:"2024-01-17T13:47:00Z", GoVersion:"go1.20.13", Compiler:"gc", Platform:"linux/amd64"}
labsuser@master:~$
```

3.Verify the upgrade plan:

```
sudo kubeadm upgrade plan
```

```
absuser@master:~$ sudo kubeadm upgrade plan
```

```
labsuser@master:~$ sudo kubeadm upgrade plan
[upgrade/config] Making sure the configuration is correct:
[upgrade/config] Reading configuration from the cluster...
[upgrade/config] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[preflight] Running pre-flight checks.
[upgrade] Running cluster health checks
[upgrade] Fetching available versions to upgrade to
[upgrade/versions] Cluster version: v1.28.5
[upgrade/versions] kubeadm version: v1.28.6
I0212 06:02:16.598366 37793 version.go:256] remote version is much newer: v1.29.1; falling back to: stable-1.28
[upgrade/versions] Target version: v1.28.6
[upgrade/versions] Latest version in the v1.28 series: v1.28.6

Components that must be upgraded manually after you have upgraded the control plane with 'kubeadm upgrade apply':
COMPONENT      CURRENT      TARGET
kubelet        3 x v1.28.5  v1.28.6

Upgrade to the latest version in the v1.28 series:

COMPONENT      CURRENT      TARGET
kube-apiserver  v1.28.5    v1.28.6
kube-controller-manager  v1.28.5  v1.28.6
kube-scheduler   v1.28.5    v1.28.6
kube-proxy       v1.28.5    v1.28.6
CoreDNS          v1.10.1    v1.10.1
etcd            3.5.9-0   3.5.10-0

You can now apply the upgrade by executing the following command:
  kubeadm upgrade apply v1.28.6
```

The table below shows the current state of component configs as understood by this version of kubeadm. Configs that have a "yes" mark in the "MANUAL UPGRADE REQUIRED" column require manual config upgrade or resetting to kubeadm defaults before a successful upgrade can be performed. The version to manually upgrade to is denoted in the "PREFERRED VERSION" column.

API GROUP	CURRENT VERSION	PREFERRED VERSION	MANUAL UPGRADE REQUIRED
kubeproxy.config.k8s.io	v1alpha1	v1alpha1	no
kubelet.config.k8s.io	v1beta1	v1beta1	no

4. Choose a version to upgrade to, and run the appropriate command.

```
sudo kubeadm upgrade apply v1.28.6
```

```
labsuser@master:~$ sudo kubeadm upgrade apply v1.28.6
```

Once the command finishes we will see:

```
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to get nodes
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

[upgrade/successful] SUCCESS! Your cluster was upgraded to "v1.28.6". Enjoy!

[upgrade/kubelet] Now that your control plane is upgraded, please proceed with upgrading your kubelets if you haven't already done so.
labsuser@master:~$
```

```
labsuser@master:~$ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
master.example.com	Ready	control-plane	28d	v1.28.5
worker-node-1.example.com	Ready	<none>	28d	v1.28.5
worker-node-2.example.com	Ready	<none>	28d	v1.28.5

```
labsuser@master:~$ kubectl get nodes
NAME                  STATUS   ROLES      AGE   VERSION
master.example.com    Ready    control-plane   28d   v1.28.5
worker-node-1.example.com  Ready    <none>     28d   v1.28.5
worker-node-2.example.com  Ready    <none>     28d   v1.28.5
labsuser@master:~$
```

Drain the node

Prepare the node for maintenance by marking it unschedulable and evicting the workloads:

```
labsuser@master:~$ kubectl drain master.example.com --ignore-daemonsets
```

```
labsuser@master:~$ kubectl drain master.example.com --ignore-daemonsets
node/master.example.com cordoned
Warning: ignoring DaemonSet-managed Pods: kube-system/calico-node-7m2ph, kube-system/fluentd-elasticsearch-jw5gg, kube-system/kube-proxy-xdfmt
evicting pod kube-system/coredns-5dd5756b68-hqqn4
evicting pod kube-system/calico-kube-controllers-658d97c59c-gw5vt
pod/calico-kube-controllers-658d97c59c-gw5vt evicted
pod/coredns-5dd5756b68-hqqn4 evicted
node/master.example.com drained
labsuser@master:~$
```

```
labsuser@master:~$ k get pod -n kube-system -o wide | grep coredns
```

```

coredns-5dd5756b68-7n5n2           1/1  Running  19 (69m ago)  10d  172.16.47.171
worker-node-1.example.com <none>      <none>
coredns-5dd5756b68-m9lws          1/1  Running  0       6m34s  172.16.232.227
worker-node-2.example.com <none>      <none>

```

To see which pods running on control plane

```
labsuser@master:~$ k get pods -A -o wide | grep master
```

```

labsuser@master:~$ k get pods -A -o wide | grep master
default      redis-master-84769768c6-5hjbc   1/1  Running   19 (71m ago)  10d  172.16.47.179  worker-node-1.example.com
e>          <none>                         calico-node-7mpfh   1/1  Running   41 (71m ago)  28d  172.31.31.207  master.example.com
e>          <none>                         etcd-master.example.com  1/1  Running   0       16m  172.31.31.207  master.example.com
e>          <none>                         fluentd-elasticsearch-jw5gg  1/1  Running   34 (71m ago)  21d  172.16.204.121  master.example.com
e>          <none>                         kube-apiserver-master.example.com  1/1  Running   0       15m  172.31.31.207  master.example.com
e>          <none>                         kube-controller-manager-master.example.com  1/1  Running   0       14m  172.31.31.207  master.example.com
e>          <none>                         kube-proxy-xdfmt    1/1  Running   0       14m  172.31.31.207  master.example.com
e>          <none>                         kube-scheduler-master.example.com    1/1  Running   0       14m  172.31.31.207  master.example.com

```

Upgrade kubelet and kubectl

1. Upgrade the kubelet and kubectl:

```
sudo apt-mark unhold kubelet kubectl && sudo apt-get update && sudo apt-get install -y
kubelet='1.28.6-1.1' kubectl='1.28.6-1.1' && sudo apt-mark hold kubelet kubectl
```

```

labsuser@master:~$ sudo apt-mark unhold kubelet kubectl && sudo apt-get update && sudo apt-get
upgrade
kubelet was already not on hold.
kubectl was already not on hold.
Hit:1 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb
Hit:6 https://ppa.launchpadcontent.net/mozillateam/ppa/ubuntu jammy InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
kubectl is already the newest version (1.28.6-1.1).
kubelet is already the newest version (1.28.6-1.1).
0 upgraded, 0 newly installed, 0 to remove and 67 not upgraded.
kubelet set on hold.
kubectl set on hold.

```

2. Restart the kubelet:

```

sudo systemctl daemon-reload
sudo systemctl restart kubelet

```

Now the particular node is disabled for scheduling

```
labsuser@master:~$ kubectl get nodes
```

```
labsuser@master:~$ kubectl get nodes
NAME           STATUS   ROLES      AGE   VERSION
master.example.com   Ready,SchedulingDisabled control-plane 28d   v1.28.6
worker-node-1.example.com Ready    <none>     28d   v1.28.5
worker-node-2.example.com Ready    <none>     28d   v1.28.5
labsuser@master:~$
```

Uncordon the node

Bring the node back online by marking it schedulable:

```
labsuser@master:~$ kubectl uncordon master.example.com
```

```
node/master.example.com uncordoned
```

And Now

```
labsuser@master:~$ kubectl get nodes
NAME           STATUS   ROLES      AGE   VERSION
master.example.com   Ready    control-plane 28d   v1.28.6
worker-node-1.example.com Ready    <none>     28d   v1.28.5
worker-node-2.example.com Ready    <none>     28d   v1.28.5
labsuser@master:~$
```

Upgrading worker nodes

Upgrade kubeadm

```
sudo apt-mark unhold kubeadm && \
sudo apt-get update &&
sudo apt-get install -y kubeadm='1.28.6-1.1' && \
sudo apt-mark hold kubeadm
```

Call "kubeadm upgrade"

For worker nodes this upgrades the local kubelet configuration:

```
sudo kubeadm upgrade node
```

```
labsuser@worker-node-1:~$ sudo kubeadm upgrade node
```

Drain the node

Prepare the node for maintenance by marking it unschedulable and evicting the workloads:

```
# execute this command on a control plane node
```

```
labsuser@master:~$ kubectl drain worker-node-1.example.com --ignore-daemonsets
```

```
kubectl drain worker-node-1.example.com --ignore-daemonsets --force --delete-emptydir-data
```

```
pod/nginx-ingress-884dd4ucd4-zt1vv evicted
pod/mysql-79c547d7fb-qbxrs evicted
pod/postgres-7d97467c58-9w28k evicted
pod/redis-master-84769768cb-5hjbc evicted
pod/secret-pod evicted
pod/task-pv-pod evicted
pod/test-mysql-6cd89db584-p8sld evicted
pod/wordpress-78cbf57fdb-nt5f5 evicted
pod/coredns-5dd5756b68-7n5n2 evicted
pod/envvar-demo evicted
pod/dapi-test-pod02 evicted
pod/liveness-exec evicted
pod/readiness-exec evicted
node/worker-node-1.example.com drained
labsuser@master:~$
```

```
labsuser@master:~$ kubectl get pods -A -o wide | grep worker-node-1 | wc -l
```

3

These are demonset pod

Upgrade kubelet and kubectl

1. Upgrade the kubelet and kubectl:

```
sudo apt-mark unhold kubelet kubectl && sudo apt-get update && sudo apt-get install -y  
kubelet='1.28.6-1.1' kubectl='1.28.6-1.1' && sudo apt-mark hold kubelet kubectl  
  
labsuser@worker-node-1:~$ sudo apt-mark unhold kubelet kubectl && sudo apt-get  
update && sudo apt-get install -y kubelet='1.28.6-1.1' kubectl='1.28.6-1.1' &&  
sudo apt-mark hold kubelet kubectl
```

2. Restart the kubelet:

```
sudo systemctl daemon-reload
```

```
sudo systemctl restart kubelet
```

```
labsuser@master:~$ kubectl get nodes
```

NAME	STATUS	ROLES	AGE
VERSION			
master.example.com	Ready	control-plane	28d
v1.28.6			
worker-node-1.example.com	Ready, SchedulingDisabled	<none>	28d
v1.28.6			
worker-node-2.example.com	Ready	<none>	28d
v1.28.5			

Uncordon the node

Bring the node back online by marking it schedulable:

```
labsuser@master:~$ kubectl uncordon worker-node-1.example.com
```

```
node/worker-node-1.example.com uncordoned
```

```
labsuser@master:~$ kubectl get node
```

NAME	STATUS	ROLES	AGE	VERSION
------	--------	-------	-----	---------

master.example.com	Ready	control-plane	28d	v1.28.6
worker-node-1.example.com	Ready	<none>	28d	v1.28.6
worker-node-2.example.com	Ready	<none>	28d	v1.28.5

```
labsuser@master:~$ kubectl uncordon worker-node-1.example.com
node/worker-node-1.example.com uncordoned
labsuser@master:~$ kubectl get node
NAME           STATUS   ROLES     AGE   VERSION
master.example.com   Ready    control-plane   28d   v1.28.6
worker-node-1.example.com   Ready    <none>      28d   v1.28.6
worker-node-2.example.com   Ready    <none>      28d   v1.28.5
labsuser@master:~$
```

Now our control plane and worker node both are upgraded

Now will do for worker node 2

```
labsuser@worker-node-2:~$ sudo apt-mark unhold kubeADM && sudo apt-get update &&
sudo apt-get install -y kubeADM='1.28.6-1.1' && sudo apt-mark hold kubeADM
```

```
labsuser@master:~$ kubectl drain worker-node-2.example.com --ignore-daemonsets
```

```
absuser@master:~$ kubectl drain worker-node-2.example.com --ignore-daemonsets
--delete-emptydir-data
```

```
labsuser@worker-node-2:~$ sudo kubeADM upgrade node
```

```
labsuser@master:~$ kubectl get node
```

NAME	STATUS	ROLES	AGE	VERSION
master.example.com	Ready	control-plane	28d	v1.28.6
worker-node-1.example.com	Ready	<none>	28d	v1.28.6
worker-node-2.example.com	Ready,SchedulingDisabled	<none>	28d	v1.28.6

Uncordon

```
labsuser@master:~$ kubectl uncordon worker-node-2.example.com
```

```
node/worker-node-2.example.com uncordoned
```

```
labsuser@master:~$ kubectl get nodes
NAME           STATUS   ROLES      AGE   VERSION
master.example.com   Ready   control-plane 28d  v1.28.6
worker-node-1.example.com Ready   <none>     28d  v1.28.6
worker-node-2.example.com Ready   <none>     28d  v1.28.6
```

```
labsuser@master:~$ kubectl uncordon worker-node-2.example.com
node/worker-node-2.example.com uncordoned
labsuser@master:~$ kubectl get nodes
NAME           STATUS   ROLES      AGE   VERSION
master.example.com   Ready   control-plane 28d  v1.28.6
worker-node-1.example.com Ready   <none>     28d  v1.28.6
worker-node-2.example.com Ready   <none>     28d  v1.28.6
```

Task 5. Generating a certificate and private key in the worker node.

1. To generate a certificate and private key on a worker node, you can use tools like OpenSSL. Here are the general steps to generate a self-signed certificate and private key:

Installing OpenSSL

```
sudo apt-get install openssl
```

```
labsuser@master:~$ sudo apt-get install openssl
```

```
labsuser@master:~$ sudo apt-get install openssl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
openssl is already the newest version (3.0.2-0ubuntu1.14).
openssl set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 67 not upgraded.
labsuser@master:~$
```

2. Generate a Private Key:

```
openssl genrsa -out worker.key 2048
```

```
labsuser@master:~$ openssl genrsa -out worker.key 2048
```

This command generates a 2048-bit RSA private key and saves it in a file named **worker.key**.

3. Generate a Certificate Signing Request (CSR):

```
labsuser@master:~$ openssl req -new -key worker.key -out worker.csr
```

```
labsuser@master:~$ openssl req -new -key worker.key -out worker.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

```
labsuser@master:~$ openssl x509 -req -days 365 -in worker.csr -signkey
```

```
worker.key -out worker.crt
```

```
Certificate request self-signature ok
```

```
subject=C = AU, ST = Some-State, O = Internet Widgits Pty Ltd
```

```
labsuser@master:~$ openssl x509 -req -days 365 -in worker.csr -signkey worker.key -out worker.crt
Certificate request self-signature ok
subject=C = AU, ST = Some-State, O = Internet Widgits Pty Ltd
```

creating a valid Kubernetes configuration file named myconfig,

```
kubectl config view --minify --flatten > myconfig
```

Verify Configuration File:

- Once the command is executed, verify that the myconfig file has been created in your current directory. You can use the ls command to list the files in the directory.

```
ls
```

```
labsuser@master:~$ kubectl config view --minify --flatten > myconfig
labsuser@master:~$ ls
DCV-Storage          cj           daemonset.yaml.1   exe1       kubesample.yaml    pod01      sec-pod
Desktop              cj.yaml      daemonset.yaml.2   exe10      kubesample.yaml.1  pod02      secret
Documents             cluster-admin db           exe1k      liveness.yaml   postgres.yaml  secrete-pod.yaml
Downloads             cluster-admin1 de           exe2       m           private_key.key  secretes
Music                cluster-role-binding.yaml deploy.yaml  exe8       myconfig        pv.yaml    snap
Pictures              cluster-role.yaml deployment.yaml exer      mysql-pod      pvc.yaml  snapshot.db.part
Public               cm           env           exet1     mysql.yaml    quota      storage
ServiceAccount.yaml cm-pod.yaml e10.yaml      g          n           quotaaz     user.yaml
Templates             cni-plugins-linux-amd64-v1.1.1.tgz emptyDir    gogs      networkpolicy.yaml readiness.yaml  wordpress.yaml
Videos                configmap   env           h          nginx-deployment role.yaml  worker.crt
ar                   configmap.yaml env.yaml     helm101   nginx-service.yaml rolebinding.yaml worker.cs
b                   containerd-1.6.8-linux-amd64.tar.gz ex          ingress-deployment.yaml nginx.yaml runc.amd64  worker.key
blue                 cronjob.yaml ex10      j           job       pod           sa         workload
cep-project2-namespace.yaml d           ex4       job      pod           scheduler
cep-project2-network-policy.yaml daemonset.yaml example-voting-app job.yaml pod.yaml
labsuser@master:~$
```

```
kubectl get pods --kubeconfig=myconfig
```

NAME	READY	STATUS	AGE
frontend-58cf9cd597-625h9	1/1	Running	2 (29m ago)
frontend-58cf9cd597-l2t54	1/1	Running	2 (29m ago)
frontend-58cf9cd597-xxpcx	1/1	Running	2 (29m ago)

```
labsuser@master:~$ kubectl get deployment --kubeconfig=myconfig
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
cart          1/1     1            1           15d
db            1/1     1            1           17d
example-deployment 1/1     1            1           3d7h
exe9-deployment 1/1     1            1           3d7h
frontend       3/3     3            3           22d
game-demo-deployment 0/5     5            0           16d
gogs          1/1     1            1           17d
kucc8          2/2     2            2           21d
kucc8-blue    10/10   10           10          21d
kucc8-green   10/10   10           10          21d
mysql          1/1     1            1           19d
nginx-deployment 1/1     1            1           22d
postgres       1/1     1            1           17d
redis          1/1     1            1           17d
redis-master   1/1     1            1           22d
redis-slave    2/2     2            2           22d
result         1/1     1            1           17d
test-mysql     1/1     1            1           9d
vote           1/1     1            1           17d
wordpress      1/1     1            1           19d
worker         1/1     1            1           17d
your-app       0/3     3            0           12d
```

```
labsuser@master:~$ kubectl get nodes --kubeconfig=myconfig
NAME           STATUS   ROLES      AGE   VERSION
master.example.com   Ready    control-plane   29d   v1.28.6
worker-node-1.example.com   Ready    <none>     29d   v1.28.6
worker-node-2.example.com   Ready    <none>     29d   v1.28.6
```