KUBERNETES ETCD BACKUP AND NAMESPACE SECURITY

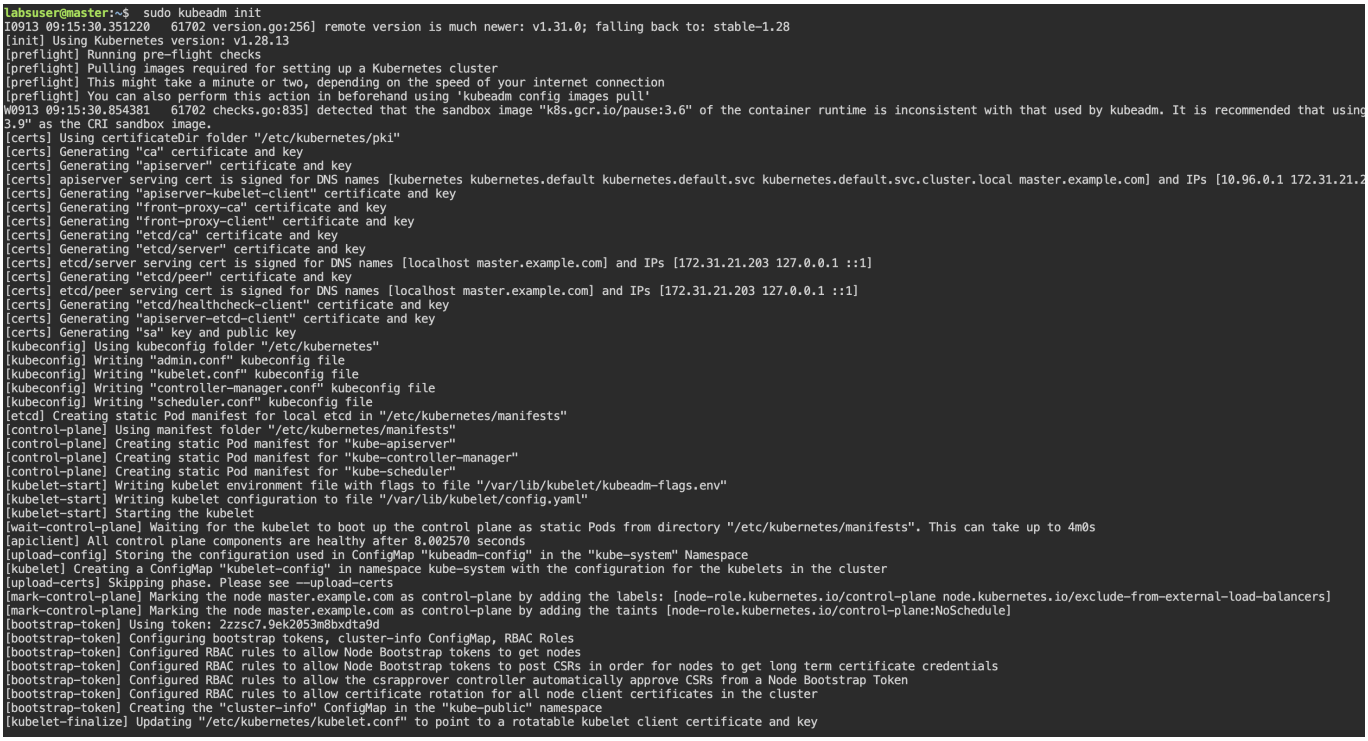
As an infrastructure admin, I was assigned the task of safeguarding a Kubernetes cluster by backing up its etcd data, ensuring network security within a specific namespace, restricting user access to view access only, and upgrading the cluster's master node to the latest version. This project emphasized the importance of data integrity and security within cloud-native environments.

Step 1: Setting up Cluster

`sudo kubeadm init`

```
    sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
    sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

`kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml`



```
labsuser@master:~$  sudo kubeadm init
I0913 09:15:30.351220   61702 version.go:256] remote version is much newer: v1.31.0; falling back to: stable-1.28
[init] Using Kubernetes version: v1.28.13
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
W0913 09:15:30.854381   61702 checks.go:835] detected that the sandbox image "k8s.gcr.io/pause:3.6" of the container runtime is inconsistent with that used by kubeadm. It is recommended that using
3.9" as the CRI sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local master.example.com] and IPs [10.96.0.1 172.31.21.2
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [localhost master.example.com] and IPs [172.31.21.203 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [localhost master.example.com] and IPs [172.31.21.203 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Starting the kubelet
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests". This can take up to 4m0s
[apiclient] All control plane components are healthy after 8.002570 seconds
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config" in namespace kube-system with the configuration for the kubelets in the cluster
[upload-certs] Skipping phase. Please see --upload-certs
[mark-control-plane] Marking the node master.example.com as control-plane by adding the labels: [node-role.kubernetes.io/control-plane node.kubernetes.io/exclude-from-external-load-balancers]
[mark-control-plane] Marking the node master.example.com as control-plane by adding the taints [node-role.kubernetes.io/control-plane:NoSchedule]
[bootstrap-token] Using token: 2zzsc7.9ek2053m8bxdta9d
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[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
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key
```

Worker Nodes initialization - Worker1 & 2

```
-discovery-token-ca-cert-hash
sha256:fb2d578ac95eff98926aa1f7bee6c983c1e0c161224047ab02423d0ac041ea
```

```
labsuser@worker-node-1:~$ sudo kubeadm join 172.31.21.203:6443 --token 2zzsc7.9ek2053m8bxdta9d \
        --discovery-token-ca-cert-hash sha256:fb2d578ac95eff98926aa1f7bee6c983fc1e0c1612240e47ab02423d0ac041ea
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

labsuser@worker-node-1:~$
```

```
labsuser@worker-node-2:~$ sudo kubeadm join 172.31.21.203:6443 --token 2zzsc7.9ek2053m8bxdta9d \
        --discovery-token-ca-cert-hash sha256:fb2d578ac95eff98926aa1f7bee6c983fc1e0c1612240e47ab02423d0ac041ea
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

labsuser@worker-node-2:~$
```

Step 2: Backup the ETCD Cluster Data

```
etcdctl snapshot save /tmp/myback --endpoints=https://172.31.21.203:2379 --
cacert=/etc/kubernetes/pki/etcd/ca.crt --
cert=/etc/kubernetes/pki/etcd/server.crt --
key=/etc/kubernetes/pki/etcd/server.key

etcdctl snapshot status /tmp/myback --endpoints=https://172.31.21.203:2379 --
cacert=/etc/kubernetes/pki/etcd/ca.crt --
cert=/etc/kubernetes/pki/etcd/server.crt --
key=/etc/kubernetes/pki/etcd/server.key
```

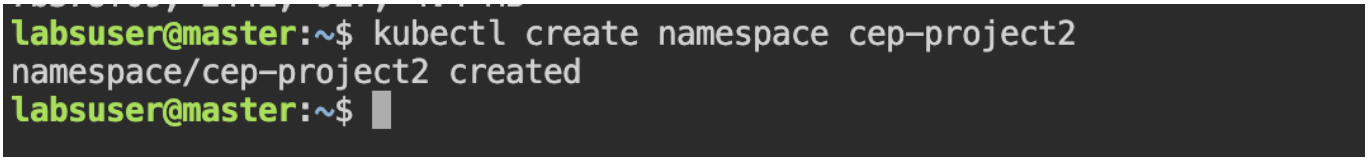
Step 3: Creating the Namespace and Configuring Network Policies: The next task was to isolate a specific environment for the project within Kubernetes. I created a namespace named cep-project2. Afterward, I applied a strict network policy to ensure that only the Pods within this namespace could communicate with each other, blocking all external access

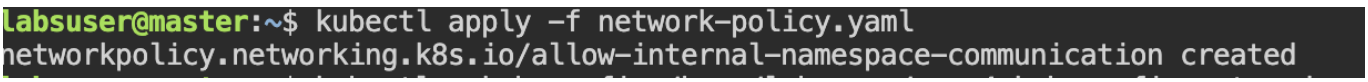
```
kubectl create namespace cep-project2

vi network-policy.yaml

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

```
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-same-namespace
  namespace: cep-project2
spec:
  podSelector: {}
  ingress:
  - from:
    - podSelector: {}
```





This allowed only intra-namespace communication and denied access from Pods outside the cep-project2 namespace.

Step 4: Configuring Kubernetes Client for User4 (Worker Node 3): The next challenge was to configure access for user4, ensuring that this user had read-only permissions on Worker Node 3. I generated the necessary certificates for user4 and used kubectl commands to assign only view access to the cep-project2 namespace.

```
openssl genrsa -out user4.key 2048

openssl req -new -key user4.key -out user4.csr -subj "/CN=user4
```

```
labsuser@master:~$ openssl genrsa -out user4.key 2048
labsuser@master:~$ openssl req -new -key user4.key -out user4.csr -subj "/CN=user4"
```

Then i Created a Certificate Signing Request

```
vi user4-csr.yaml
```

```
kind: CertificateSigningRequest
metadata:
  name: user4
spec:
  request:
  signerName: kubernetes.io/kube-apiserver-client
  expirationSeconds: 86400  # one day
  usages:
  - client auth
```

```
kubectl apply -f user4-csr.yaml
```

```
labsuser@master:~$ kubectl apply -f user4-csr.yaml
certificatesigningrequest.certificates.k8s.io/user4 created
```

```
cat user4.csr | base64 | tr -d "\n"
```

This command gets the base64 encoded value of the CSR file content, copy and paste it in the request of the user4-csr.yaml file.

Approve the CertificateSigningRequest

```
kubectl certificate approve user4
```

```
labsuser@master:~$ kubectl certificate approve user4
certificatesigningrequest.certificates.k8s.io/user4 approved
```

Get Certificate and Export the issued certificate from the CertificateSigningRequest.

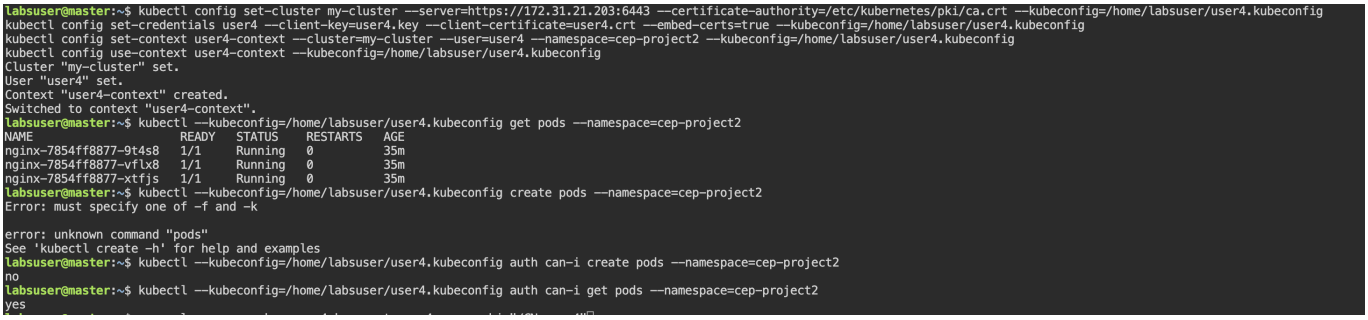
```
kubectl get csr/user4 -o yaml

kubectl get csr user4 -o jsonpath='{.status.certificate}'| base64 -d > user4.crt
```

Create the user4.kubeconfig File Use the signed certificate (user4.crt), key (user4.key), and Kubernetes CA (ca.crt) to generate user4.kubeconfig

```
kubectl config set-credentials user4 --client-key=user4.key --client-
certificate=user4.crt --embed-certs=true --
kubeconfig=/home/labsuser/user4.kubeconfig
kubectl config set-context user4-context --cluster=my-cluster --user=user4
--namespace=cep-project2 --kubeconfig=/home/labsuser/user4.kubeconfig
kubectl config use-context user4-context --
kubeconfig=/home/labsuser/user4.kubeconfig
```

I encountered an issue here where i did not specify the full path to my user4.kubeconfig file where it kept asking for username.



Step 5: Grant user4 View Access Create a role and rolebindingwith view access in the cep-project2 namespace

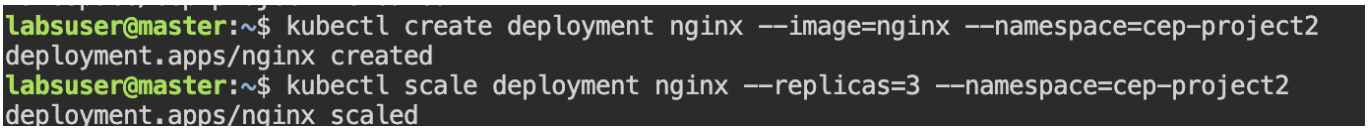
```
kubectl create role view-role --verb=get,list,watch --resource=pods,services --
namespace=cep-project2
```

```
kubectl create rolebinding user4-view-binding --role=view-role --user=user4 --
namespace=cep-project2
```

Then i created a deployment in the namespace

```
kubectl create deployment nginx --image=nginx --namespace=cep-project2
```

```
kubectl scale deployment nginx --replicas=3 --namespace=cep-project2
```



Step 6: I Configured a Kubernetes client on worker node 2 in such a way that user4 has only view access to cep-project2.

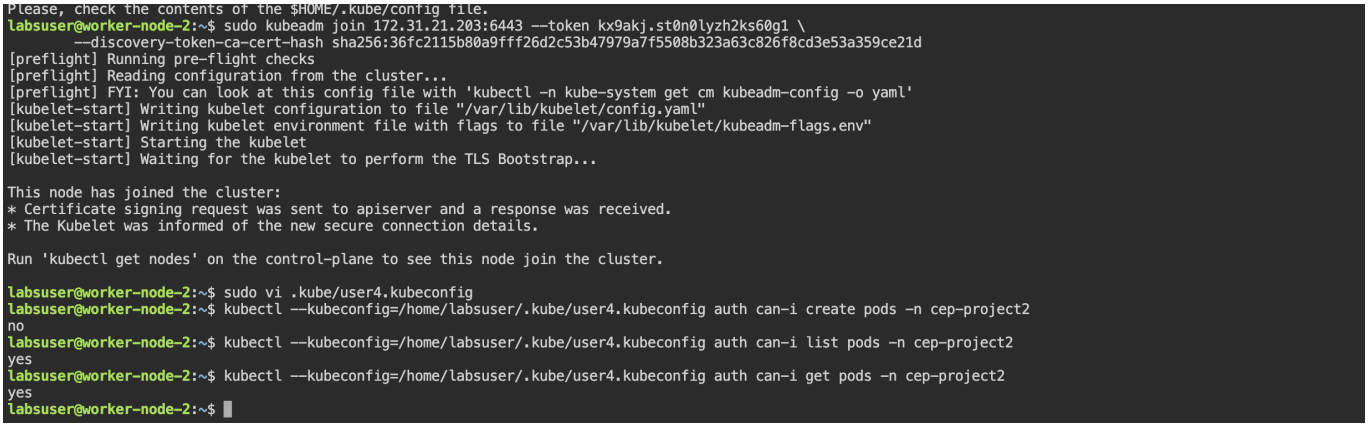
I copied my use4.kubeconfig file on my master node to my workernode2 and validated if user4 truly has view access only.

```
kubectl --kubeconfig=/home/labsuser/.kube/user4.kubeconfig auth can-i get pods --namespace=cep-project2
```

```
kubectl --kubeconfig=/home/labsuser/.kube/user4.kubeconfig auth can-i create pods --namespace=cep-project2
```

```
kubectl --kubeconfig=/home/labsuser/.kube/user4.kubeconfig get pods --namespace=cep-project2
```

```
kubectl --kubeconfig=/home/labsuser/.kube/user4.kubeconfig create pods --namespace=cep-project2
```



Step7: Upgrading Kubernetes Cluster

1. Update the Package List

```
sudo apt update
```

2. Check Available Versions of kubeadm

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

3. Unhold kubeadm

```
sudo apt-mark unhold kubeadm
```

If kubeadm was held from being updated previously, this command releases it.

4. Install the Chosen Version of kubeadm

```
sudo apt-get install -y kubeadm=1.28.12-1.1
```

5. Hold kubeadm Again

```
sudo apt-mark hold kubeadm
```

6. Check Installed kubeadm Version

```
kubeadm version
```

7. Plan the Upgrade

```
sudo kubeadm upgrade plan
```

8. Apply the Upgrade

```
sudo kubeadm upgrade apply v1.28.12-1.1
```

9. Drain the Control Plane Node

```
kubectl drain master.example.com --ignore-daemonsets
```

This safely evicts workloads from the control plane node. The --ignore-daemonsets flag allows the drain to complete even if there are DaemonSets running.

10. Upgrade kubelet and kubectl

```
sudo apt-get install -y kubelet=1.28.12-1.1 kubectl=1.28.12-1.1
sudo apt-mark hold kubelet kubectl
```

This upgrades kubelet and kubectl on the control plane to the same version as kubeadm.

11. Restart kubelet

```
sudo systemctl restart kubelet
```

Reloads systemd to apply any configuration changes and restarts the kubelet service.

12. Uncordon the Control Plane Node:

```
kubectl uncordon master.example.com
```

This marks the control plane node as schedulable again, allowing workloads to be placed on it.

13. Check Node Status

```
kubectl get nodes
```

```
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 [128 kB]
Get:3 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Hit:4 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 [2023 kB]
Hit:7 https://ppa.launchpadcontent.net/mozillateam/ppa/ubuntu jammy InRelease
Get:8 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1124 kB]
Get:9 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [28.8 kB]
Get:10 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f Metadata [672 B]
Fetched 3560 kB in 2s (1835 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
135 packages can be upgraded. Run 'apt list --upgradable' to see them.
   kubeadm | 1.28.14-2.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
   kubeadm | 1.28.13-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
   kubeadm | 1.28.12-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
   kubeadm | 1.28.11-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
   kubeadm | 1.28.10-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
   kubeadm | 1.28.9-2.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
   kubeadm | 1.28.8-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
   kubeadm | 1.28.7-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
   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:~$
```

```
labsuser@master:~$ sudo apt-get install -y kubelet='1.28.12-1.1' kubectl='1.28.12-1.1'
sudo apt-mark hold kubelet kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages will be upgraded:
  kubectl kubelet
2 upgraded, 0 newly installed, 0 to remove and 132 not upgraded.
Need to get 30.1 MB of archives.
After this operation, 1050 kB of additional disk space will be used.
Get:1 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb  kubectl 1.28.12-1.1 [10.4 MB]
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb  kubelet 1.28.12-1.1 [19.6 MB]
Fetched 30.1 MB in 1s (36.5 MB/s)
(Reading database ... 218769 files and directories currently installed.)
Preparing to unpack .../kubectl_1.28.12-1.1_amd64.deb ...
Unpacking kubectl (1.28.12-1.1) over (1.28.9-2.1) ...
Preparing to unpack .../kubelet_1.28.12-1.1_amd64.deb ...
Unpacking kubelet (1.28.12-1.1) over (1.28.9-2.1) ...
dpkg: warning: unable to delete old directory '/etc/sysconfig': Directory not empty
Setting up kubectl (1.28.12-1.1) ...
Setting up kubelet (1.28.12-1.1) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Running kernel seems to be up-to-date.

Restarting services...
 systemctl restart kubelet.service


No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
kubelet set on hold.
kubectl set on hold.
labsuser@master:~$ sudo systemctl daemon-reload
sudo systemctl restart kubelet
labsuser@master:~$ kubectl uncordon master.example.com
node/master.example.com already uncordoned
labsuser@master:~$ kubectl get nodes
NAME                      STATUS   ROLES           AGE   VERSION
master.example.com        Ready    control-plane   34h   v1.28.12
worker-node-1.example.com   Ready    <none>          34h   v1.28.9
worker-node-2.example.com   Ready    <none>          34h   v1.28.9
labsuser@master:~$ kubectl get nodes -o wide
NAME                      STATUS   ROLES           AGE   VERSION    INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
master.example.com        Ready    control-plane   34h   v1.28.12   172.31.21.203   <none>        Ubuntu 22.04.3 LTS   6.5.0-1023-aws   containerd://1.6.8
worker-node-1.example.com   Ready    <none>          34h   v1.28.9    172.31.25.251   <none>        Ubuntu 22.04.3 LTS   6.5.0-1023-aws   containerd://1.6.8
worker-node-2.example.com   Ready    <none>          34h   v1.28.9    172.31.22.254   <none>        Ubuntu 22.04.3 LTS   6.5.0-1023-aws   containerd://1.6.8
```

Upgrade Worker Nodes 1 to v1.28.12-1.1

1. Update the Worker Node: On each worker node, start by updating the package list

```
sudo apt-get update
```

2. Unhold kubeadm on the Worker Node

```
sudo apt-mark unhold kubeadm
```

3. Install the Updated Version of kubeadm

```
sudo apt-get install -y kubeadm=1.28.12-1.1
```

4. Hold kubeadm Again

```
sudo apt-mark hold kubeadm
```

5. Verify kubeadm Version

```
kubeadm version
```

Confirm that kubeadm on the worker node is upgraded to the correct version

6. Upgrade the Worker Node

```
sudo kubeadm upgrade node
```

This upgrades the kubelet configuration for the worker node.

7. Drain the Worker Node from the Master: On the control plane node, run

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

This safely evicts workloads from the worker node.

8. Upgrade kubelet and kubectl on the Worker Node

```
sudo apt-get install -y kubelet=1.28.12-1.1 kubectl=1.28.12-1.1
sudo apt-mark hold kubelet kubectl
```

9. Restart kubelet

```
sudo systemctl restart kubelet
```

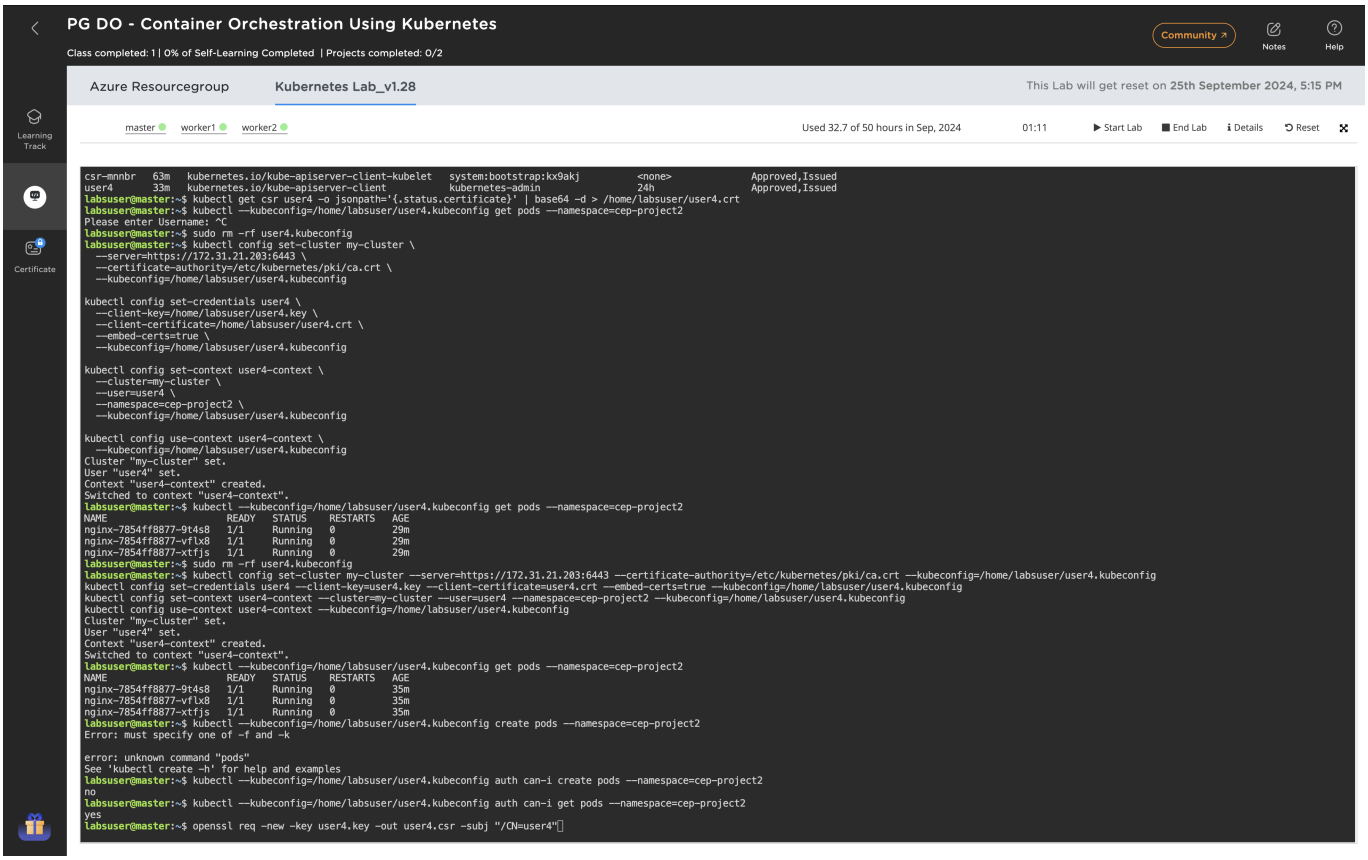
10. Uncordon the Worker Node from the Master

```
kubectl uncordon worker-node-1.example.com
```

After the upgrade allows it to start receiving new pods again for scheduling.

11. Check Node Status

```
kubectl get nodes
```



Upgrade Worker Nodes 2 to v1.28.12-1.1

1. Update the Worker Node: On each worker node, start by updating the package list

```
sudo apt-get update
```

2. Unhold kubeadm on the Worker Node

```
sudo apt-mark unhold kubeadm
```

3. Install the Updated Version of kubeadm

```
sudo apt-get install -y kubeadm=1.28.12-1.1
```

4. Hold kubeadm Again

```
sudo apt-mark hold kubeadm
```

5. Verify kubeadm Version

```
kubeadm version
```

Confirm that kubeadm on the worker node is upgraded to the correct version

6. Upgrade the Worker Node

```
sudo kubeadm upgrade node
```

This upgrades the kubelet configuration for the worker node.

7. Drain the Worker Node from the Master: On the control plane node, run

```
kubectl drain worker-node-2.example.com --ignore-daemonsets --delete-emptydir-data
```

This safely evicts workloads from the worker node.

8. Upgrade kubelet and kubectl on the Worker Node

```
sudo apt-get install -y kubelet=1.28.12-1.1 kubectl=1.28.12-1.1
sudo apt-mark hold kubelet kubectl
```

9. Restart kubelet

```
sudo systemctl restart kubelet
```

10. Uncordon the Worker Node from the Master

```
kubectl uncordon worker-node-2.example.com
```

After the upgrade allows it to start receiving new pods again for scheduling.

11. Check Node Status

```
kubectl get nodes
```

```
labsuser@worker-node-2:~$ sudo apt-mark unhold kubelet kubectl
sudo apt-get update
sudo apt-get install -y kubelet='1.28.12-1.1' kubectl='1.28.12-1.1'
sudo apt-mark hold kubelet kubectl
Canceled hold on kubelet.
Canceled hold on kubectl.
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 [128 kB]
Get:3 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 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 [2023 kB]
Get:7 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1124 kB]
Hit:8 https://ppa.launchpadcontent.net/mozillateam/ppa/ubuntu jammy InRelease
Get:9 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [28.8 kB]
Get:10 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f Metadata [672 B]
Fetched 3560 kB in 1s (2423 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages will be upgraded:
  kubectl kubelet
2 upgraded, 0 newly installed, 0 to remove and 133 not upgraded.
Need to get 30.1 MB of archives.
After this operation, 1050 kB of additional disk space will be used.
Get:1 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb  kubectl 1.28.12-1.1 [10.4 MB]
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb  kubelet 1.28.12-1.1 [19.6 MB]
Fetched 30.1 MB in 1s (40.6 MB/s)
(Reading database ... 218715 files and directories currently installed.)
Preparing to unpack .../kubectl_1.28.12-1.1_amd64.deb ...
Unpacking kubectl (1.28.12-1.1) over (1.28.9-2.1) ...
Preparing to unpack .../kubelet_1.28.12-1.1_amd64.deb ...
Unpacking kubelet (1.28.12-1.1) over (1.28.9-2.1) ...
dpkg: warning: unable to delete old directory '/etc/sysconfig': Directory not empty
Setting up kubectl (1.28.12-1.1) ...
Setting up kubelet (1.28.12-1.1) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Running kernel seems to be up-to-date.

Restarting services...
 systemctl restart kubelet.service

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
kubelet set on hold.
kubectl set on hold.
labsuser@worker-node-2:~$ sudo systemctl daemon-reload
sudo systemctl restart kubelet
labsuser@worker-node-2:~$
```

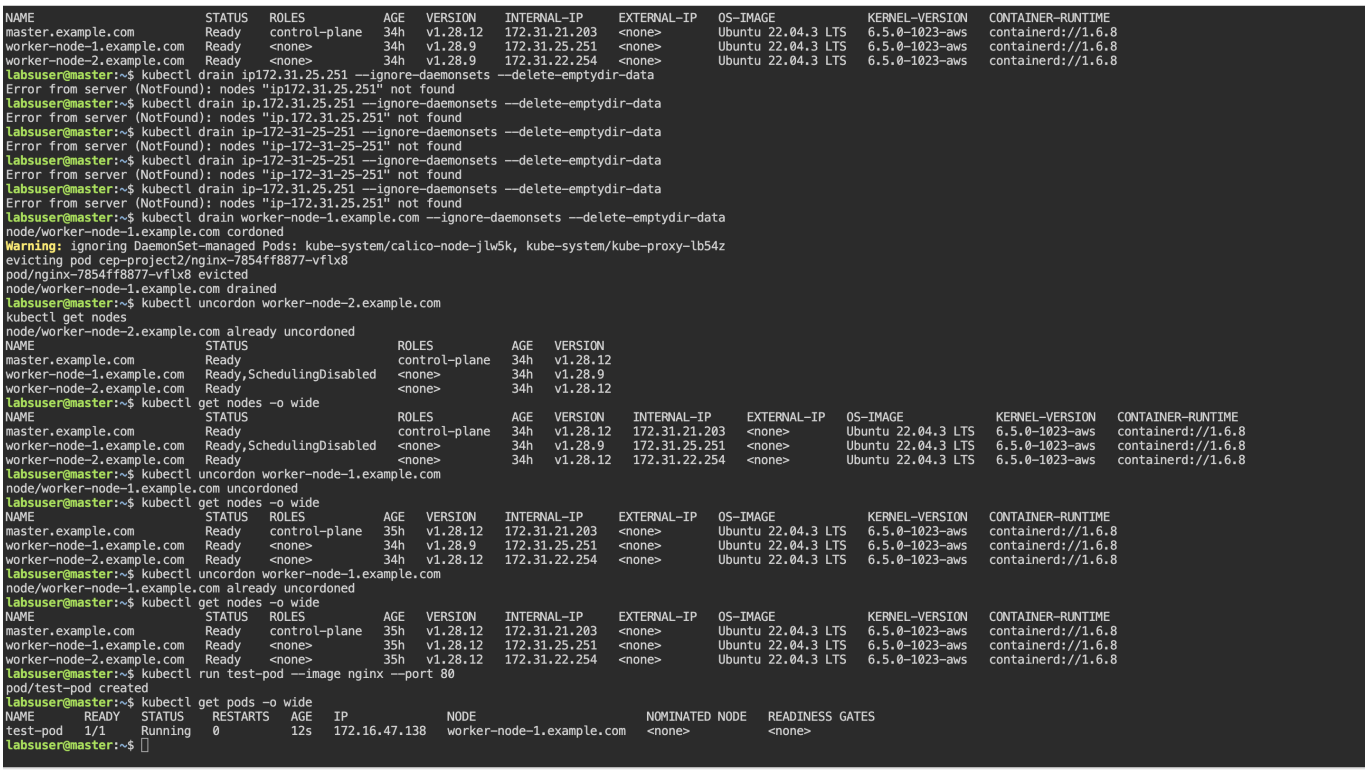
Validate the Cluster Upgrade

1. Run a Test Pod: On the control plane, run the following to ensure everything is functioning correctly

```
kubectl run test-pod --image nginx --port 80
```

2. Check Pod

```
kubectl get pods -o wide
```



In Conclusion, this project demonstrated essential Kubernetes admin tasks like securing etcd data, applying network policies, managing user access, and upgrading the cluster. It emphasized the importance of data integrity, access control, and seamless upgrades in maintaining a secure and efficient Kubernetes environment.