I will generate CA and create a CA server

cat <<EOF > CAopenssl.cnf

[req]

distinguished\_name = req\_distinguished\_name

x509\_extensions = v3\_ca

prompt = no

[req\_distinguished\_name]

C = BD

ST = Dhaka

L = Dhaka

O = Kubernetes

OU = CA

CN = kubernetes-ca

[v3\_ca]

subjectKeyIdentifier = hash

authorityKeyIdentifier = keyid:always,issuer

basicConstraints = critical, CA:true

keyUsage = critical, keyCertSign, cRLSign

EOF

openssl genpkey -algorithm RSA -out ca.key -pkeyopt rsa\_keygen\_bits:4096

openssl req -x509 -new -nodes -key ca.key -sha256 -days 1000 -out ca.crt -config CAopenssl.cnf -extensions v3\_ca

A screen shot of a computer screen

AI-generated content may be incorrect.

Now I will install etcd server

# vim openssl-etcd.cnf

[req]

req\_extensions = v3\_req

distinguished\_name = req\_distinguished\_name

[req\_distinguished\_name]

[ v3\_req ]

basicConstraints = CA:FALSE

keyUsage = nonRepudiation, digitalSignature, keyEncipherment

subjectAltName = @alt\_names

[alt\_names]

IP.1 = 192.168.60.12 #This my host ip

IP.2 = 127.0.0.1

openssl genpkey -algorithm RSA -out etcd-server.key -pkeyopt rsa\_keygen\_bits:2048

openssl req -new -key etcd-server.key -subj "/CN=etcd-server/O=Kubernetes" -out etcd-server.csr -config openssl-etcd.cnf

openssl x509 -req -in etcd-server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out etcd-server.crt -extensions v3\_req -extfile openssl-etcd.cnf -days 1000

openssl verify -CAfile ca.crt etcd-server.crt

A black screen with a black background

AI-generated content may be incorrect.

Download etcd binary

wget <https://github.com/etcd-io/etcd/releases/download/v3.6.0-rc.3/etcd-v3.6.0-rc.3-linux-amd64.tar.gz>

tar -xvf etcd-v3.6.0-rc.3-linux-amd64.tar.gz

mv etcd-v3.6.0-rc.3-linux-amd64/etcd etcd-v3.6.0-rc.3-linux-amd64/etcdctl /usr/local/bin/

ls /usr/local/bin/

mkdir -p /etc/etcd /var/lib/etcd

chmod 700 /var/lib/etcd

A black background with colorful lines

AI-generated content may be incorrect.

cp etcd-server.key etcd-server.crt ca.crt /etc/etcd/

A black rectangle with white text

AI-generated content may be incorrect.

#vim /etc/systemd/system/etcd.service

[Unit]

Description=etcd

Documentation=https://github.com/coreos

[Service]

ExecStart=/usr/local/bin/etcd \

--name controlplane01 \

--cert-file=/etc/etcd/etcd-server.crt \

--key-file=/etc/etcd/etcd-server.key \

--peer-cert-file=/etc/etcd/etcd-server.crt \

--peer-key-file=/etc/etcd/etcd-server.key \

--trusted-ca-file=/etc/etcd/ca.crt \

--peer-trusted-ca-file=/etc/etcd/ca.crt \

--peer-client-cert-auth \

--client-cert-auth \

--initial-advertise-peer-urls https://192.168.60.12:2380 \

--listen-peer-urls https://192.168.60.12:2380 \

--listen-client-urls https://192.168.60.12:2379,https://127.0.0.1:2379 \

--advertise-client-urls https://192.168.60.12:2379 \

--initial-cluster-token etcd-cluster-0 \

--initial-cluster controlplane01=https://192.168.60.12:2380 \

--initial-cluster-state new \

--data-dir=/var/lib/etcd

Restart=on-failure

RestartSec=5

[Install]

WantedBy=multi-user.target

systemctl daemon-reload

systemctl start etcd

systemctl enable etcd

ETCDCTL\_API=3 etcdctl member list \

--endpoints=https://127.0.0.1:2379 \

--cacert=/etc/etcd/ca.crt \

--cert=/etc/etcd/etcd-server.crt \

--key=/etc/etcd/etcd-server.key

A computer screen shot of a computer screen

AI-generated content may be incorrect.

Now I will install KubeAPI-server along with its dependencies like kube-control-manager and kubescheduler so I will generate all the certificate all together

This certificate generation below is for kubeapi-serve

SERVICE\_CIDR=10.96.0.0/24

API\_SERVICE=$(echo $SERVICE\_CIDR | awk 'BEGIN {FS="."} ; { printf("%s.%s.%s.1", $1, $2, $3) }')

echo $SERVICE\_CIDR

echo $API\_SERVICE

#vim openssl.cnf

req\_extensions = v3\_req

distinguished\_name = req\_distinguished\_name

[req\_distinguished\_name]

[v3\_req]

basicConstraints = critical, CA:FALSE

keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment

extendedKeyUsage = serverAuth

subjectAltName = @alt\_names

[alt\_names]

DNS.1 = kubernetes

DNS.2 = kubernetes.default

DNS.3 = kubernetes.default.svc

DNS.4 = kubernetes.default.svc.cluster

DNS.5 = kubernetes.default.svc.cluster.local

IP.1 = 192.168.60.12

IP.2 = 127.0.0.1

IP.3 = 10.96.0.1 #this one is from $API\_SERVICE ip

A black screen with colorful text

AI-generated content may be incorrect.

openssl genpkey -algorithm RSA -out kube-apiserver.key -pkeyopt rsa\_keygen\_bits:2048

openssl req -new -key kube-apiserver.key -subj "/CN=kube-apiserver/O=Kubernetes" -out kube-apiserver.csr -config openssl.cnf

openssl x509 -req -in kube-apiserver.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out kube-apiserver.crt -extensions v3\_req -extfile openssl.cnf -days 1000

openssl verify -CAfile ca.crt kube-apiserver.crt

A computer screen with colorful lines

AI-generated content may be incorrect.

Kubeapiserver communicate with kubelet so I need to create kubeapiserver-kubelet-client certificate

#vim openssl-kubelet.cnf

[req]

req\_extensions = v3\_req

distinguished\_name = req\_distinguished\_name

[req\_distinguished\_name]

[v3\_req]

basicConstraints = critical, CA:FALSE

keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment

extendedKeyUsage = clientAuth

openssl genpkey -algorithm RSA -out apiserver-kubelet-client.key -pkeyopt rsa\_keygen\_bits:2048

openssl req -new -key apiserver-kubelet-client.key -subj "/CN=kube-apiserver-kubelet-client/O=system:masters" -out apiserver-kubelet-client.csr -config openssl-kubelet.cnf

openssl x509 -req -in apiserver-kubelet-client.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out apiserver-kubelet-client.crt -extensions v3\_req -extfile openssl-kubelet.cnf -days 1000

openssl verify -CAfile ca.crt apiserver-kubelet-client.crt

A computer screen with a black background

AI-generated content may be incorrect.

Create certificate for controller-manager

openssl genpkey -algorithm RSA -out kube-controller-manager.key -pkeyopt rsa\_keygen\_bits:2048

openssl req -new -key kube-controller-manager.key -subj "/CN=system:kube-controller-manager/O=system:kube-controller-manager" -out kube-controller-manager.csr

openssl x509 -req -in kube-controller-manager.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out kube-controller-manager.crt -days 1000

openssl verify -CAfile ca.crt kube-controller-manager.crt

A screen shot of a computer

AI-generated content may be incorrect.

Now I will create for kube-scheduler

openssl genpkey -algorithm RSA -out kube-scheduler.key -pkeyopt rsa\_keygen\_bits:2048

openssl req -new -key kube-scheduler.key -subj "/CN=system:kube-scheduler/O=system:kube-scheduler" -out kube-scheduler.csr

openssl x509 -req -in kube-scheduler.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out kube-scheduler.crt -days 1000

openssl verify -CAfile ca.crt kube-scheduler.crt

A screen shot of a computer

AI-generated content may be incorrect.

Now I will create a service account certificate

openssl genpkey -algorithm RSA -out service-account.key -pkeyopt rsa\_keygen\_bits:2048

openssl req -new -key service-account.key -subj "/CN=service-accounts/O=Kubernetes" -out service-account.csr

openssl x509 -req -in service-account.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out service-account.crt -days 1000

openssl verify -CAfile ca.crt service-account.crt

A screen shot of a computer

AI-generated content may be incorrect.

Now we will download the latest binaries for kubeapi-server, kubectl,kube-controller,kubescheduler

wget https://dl.k8s.io/v1.32.3/bin/linux/amd64/kubectl

wget https://dl.k8s.io/v1.32.3/bin/linux/amd64/kube-apiserver

wget https://dl.k8s.io/v1.32.3/bin/linux/amd64/kube-controller-manager

wget <https://dl.k8s.io/v1.32.3/bin/linux/amd64/kube-scheduler>

chmod +x kubectl kube-apiserver kube-controller-manager kube-scheduler

mv kube-apiserver kube-controller-manager kube-scheduler kubectl /usr/local/bin/

ll -lah /usr/local/bin/

A screenshot of a computer

AI-generated content may be incorrect.

For kubeapi-server store data with encryption we will create an encryption key

#Generate an encryption key. This is simply 32 bytes of random data, which we base64 encode:

ENCRYPTION\_KEY=$(head -c 32 /dev/urandom | base64)

echo $ENCRYPTION\_KEY



#vim encryption-config.yaml

kind: EncryptionConfig

apiVersion: v1

resources:

- resources:

- secrets

providers:

- aescbc:

keys:

- name: key1

secret: 8vhLpvUaaH5B+zvHBpBxbFpyWUs59zY5uxqMqxTnKrc=

- identity: {}

Now we are ready to deploy kubeapi-server as a service

Transfer all the certificate to the right place

mkdir -p /var/lib/Kubernetes

mkdir -p /var/lib/kubernetes/pki/

cp encryption-config.yaml /var/lib/kubernetes/

cp ca.key ca.crt etcd-server.crt etcd-server.key apiserver-kubelet-client.crt apiserver-kubelet-client.key service-account.crt service-account.key kube-apiserver.crt kube-apiserver.key /var/lib/kubernetes/pki/



#vim /etc/systemd/system/kube-apiserver.service

[Unit]

Description=Kubernetes API Server

Documentation=https://github.com/kubernetes/kubernetes

[Service]

ExecStart=/usr/local/bin/kube-apiserver \

--advertise-address=192.168.60.12 \

--allow-privileged=true \

--apiserver-count=2 \

--audit-log-maxage=30 \

--audit-log-maxbackup=3 \

--audit-log-maxsize=100 \

--audit-log-path=/var/log/audit.log \

--authorization-mode=Node,RBAC \

--bind-address=0.0.0.0 \

--client-ca-file=/var/lib/kubernetes/pki/ca.crt \

--enable-admission-plugins=NodeRestriction,ServiceAccount \

--enable-bootstrap-token-auth=true \

--etcd-cafile=/var/lib/kubernetes/pki/ca.crt \

--etcd-certfile=/var/lib/kubernetes/pki/etcd-server.crt \

--etcd-keyfile=/var/lib/kubernetes/pki/etcd-server.key \

--etcd-servers=https://192.168.60.12:2379\

--event-ttl=1h \

--encryption-provider-config=/var/lib/kubernetes/encryption-config.yaml \

--kubelet-certificate-authority=/var/lib/kubernetes/pki/ca.crt \

--kubelet-client-certificate=/var/lib/kubernetes/pki/apiserver-kubelet-client.crt \

--kubelet-client-key=/var/lib/kubernetes/pki/apiserver-kubelet-client.key \

--runtime-config=api/all=true \

--service-account-key-file=/var/lib/kubernetes/pki/service-account.crt \

--service-account-signing-key-file=/var/lib/kubernetes/pki/service-account.key \

--service-account-issuer=https://192.168.60.12:6443 \

--service-cluster-ip-range=10.96.0.0/16 \

--service-node-port-range=30000-32767 \

--tls-cert-file=/var/lib/kubernetes/pki/kube-apiserver.crt \

--tls-private-key-file=/var/lib/kubernetes/pki/kube-apiserver.key \

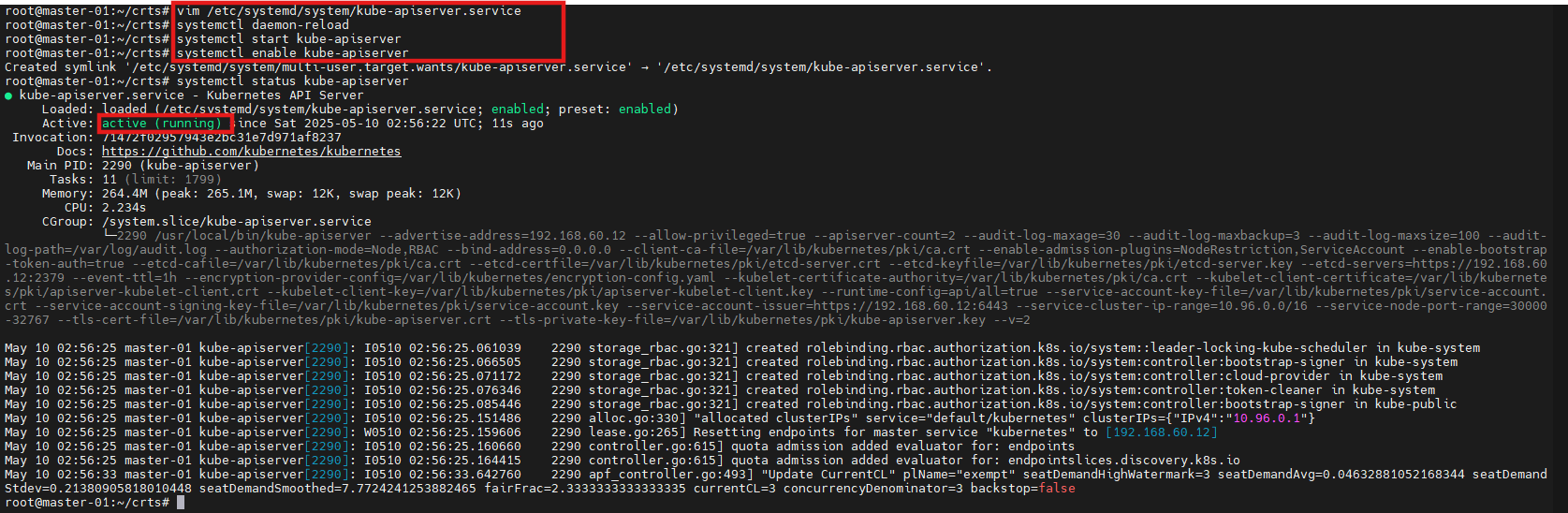
--v=2

Restart=on-failure

RestartSec=5

[Install]

WantedBy=multi-user.target



Now we will create kube-controller-manager.kubeconfig to communicate with kube-apiserver

kubectl config set-cluster kubernetes-the-hard-way \

--certificate-authority=/var/lib/kubernetes/pki/ca.crt \

--server=https://127.0.0.1:6443 \

--kubeconfig=kube-controller-manager.kubeconfig

kubectl config set-credentials system:kube-controller-manager \

--client-certificate=/var/lib/kubernetes/pki/kube-controller-manager.crt \

--client-key=/var/lib/kubernetes/pki/kube-controller-manager.key \

--kubeconfig=kube-controller-manager.kubeconfig

kubectl config set-context default \

--cluster=kubernetes-the-hard-way \

--user=system:kube-controller-manager \

--kubeconfig=kube-controller-manager.kubeconfig

kubectl config use-context default --kubeconfig=kube-controller-manager.kubeconfig

cp kube-controller-manager.kubeconfig /var/lib/kubernetes/

A screen shot of a computer

AI-generated content may be incorrect.

#vim /etc/systemd/system/kube-controller-manager.service

[Unit]

Description=Kubernetes Controller Manager

Documentation=https://github.com/kubernetes/kubernetes

[Service]

ExecStart=/usr/local/bin/kube-controller-manager \

--allocate-node-cidrs=true \

--authentication-kubeconfig=/var/lib/kubernetes/kube-controller-manager.kubeconfig \

--authorization-kubeconfig=/var/lib/kubernetes/kube-controller-manager.kubeconfig \

--bind-address=127.0.0.1 \

--client-ca-file=/var/lib/kubernetes/pki/ca.crt \

--cluster-cidr=10.244.0.0/16 \

--cluster-name=kubernetes \

--cluster-signing-cert-file=/var/lib/kubernetes/pki/ca.crt \

--cluster-signing-key-file=/var/lib/kubernetes/pki/ca.key \

--controllers=\*,bootstrapsigner,tokencleaner \

--kubeconfig=/var/lib/kubernetes/kube-controller-manager.kubeconfig \

--leader-elect=true \

--node-cidr-mask-size=24 \

--requestheader-client-ca-file=/var/lib/kubernetes/pki/ca.crt \

--root-ca-file=/var/lib/kubernetes/pki/ca.crt \

--service-account-private-key-file=/var/lib/kubernetes/pki/service-account.key \

--service-cluster-ip-range=10.96.0.0/16 \

--use-service-account-credentials=true \

--v=2

Restart=on-failure

RestartSec=5

[Install]

WantedBy=multi-user.target

cp ca.key kube-controller-manager.key kube-controller-manager.crt /var/lib/kubernetes/pki/

systemctl daemon-reload

systemctl start kube-controller-manager

systemctl enable kube-controller-manager

systemctl status kube-controller-manager

A computer screen shot of a computer screen

AI-generated content may be incorrect.

Now I will create kube-scheduler.kubeconfig

kubectl config set-cluster kubernetes-the-hard-way \

--certificate-authority=/var/lib/kubernetes/pki/ca.crt \

--server=https://127.0.0.1:6443 \

--kubeconfig=kube-scheduler.kubeconfig

kubectl config set-credentials system:kube-scheduler \

--client-certificate=/var/lib/kubernetes/pki/kube-scheduler.crt \

--client-key=/var/lib/kubernetes/pki/kube-scheduler.key \

--kubeconfig=kube-scheduler.kubeconfig

kubectl config set-context default \

--cluster=kubernetes-the-hard-way \

--user=system:kube-scheduler \

--kubeconfig=kube-scheduler.kubeconfig

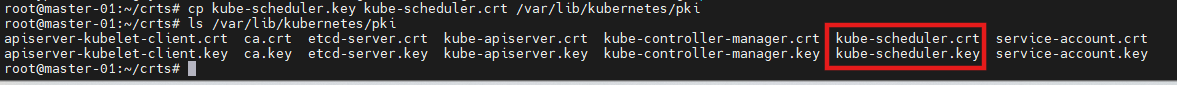
kubectl config use-context default --kubeconfig=kube-scheduler.kubeconfig

cp kube-scheduler.kubeconfig /var/lib/kubernetes/

A screen shot of a computer program

AI-generated content may be incorrect.

cp kube-scheduler.key kube-scheduler.crt /var/lib/kubernetes/pki



#vim /etc/systemd/system/kube-scheduler.service

[Unit]

Description=Kubernetes Scheduler

Documentation=https://github.com/kubernetes/kubernetes

[Service]

ExecStart=/usr/local/bin/kube-scheduler \

--kubeconfig=/var/lib/kubernetes/kube-scheduler.kubeconfig \

--leader-elect=true \

--v=2

Restart=on-failure

RestartSec=5

[Install]

WantedBy=multi-user.target

systemctl daemon-reload

systemctl start kube-scheduler

systemctl enable kube-scheduler

A computer screen with many colorful lines

AI-generated content may be incorrect.

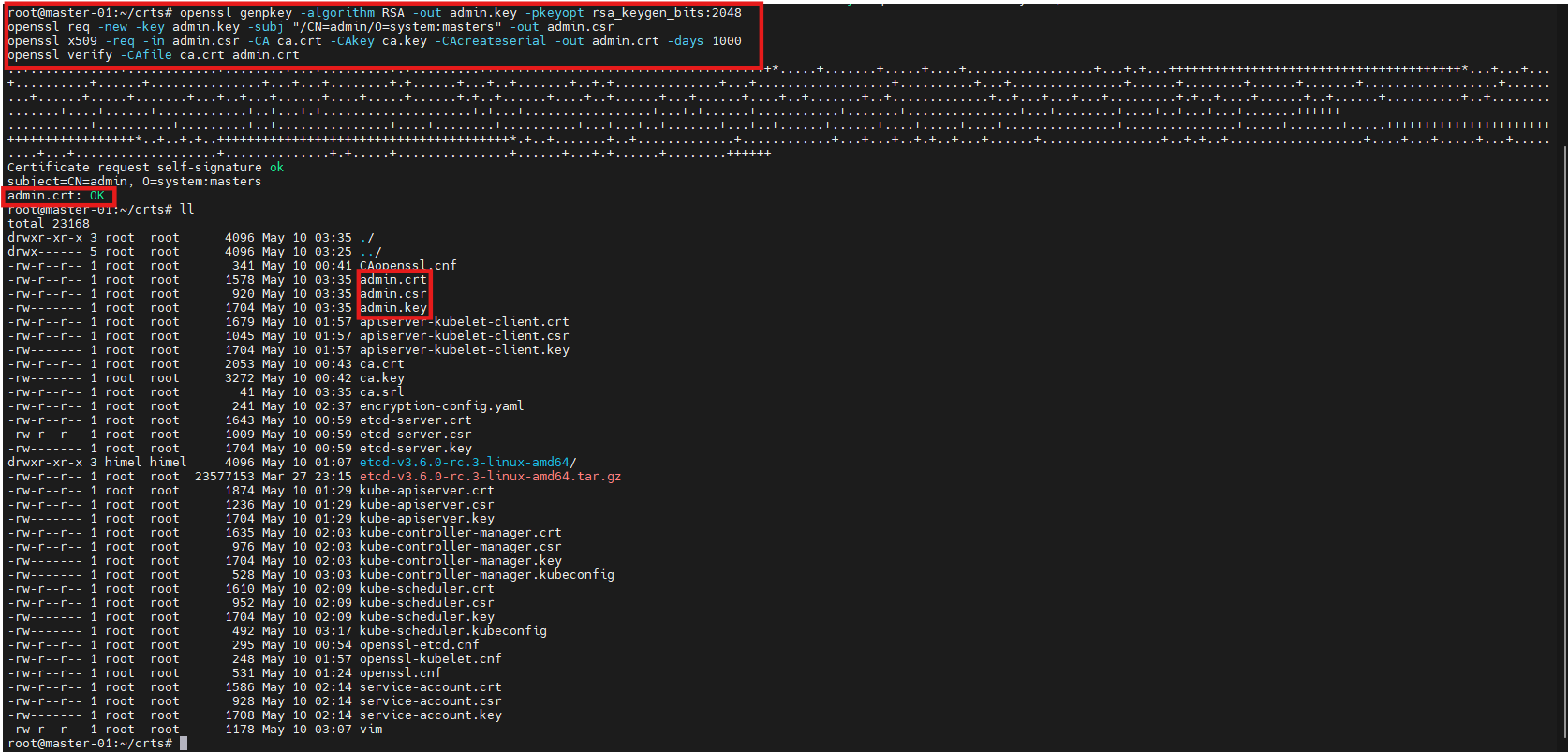
Now I will create master user to communicate with kubeapi user

openssl genpkey -algorithm RSA -out admin.key -pkeyopt rsa\_keygen\_bits:2048

openssl req -new -key admin.key -subj "/CN=admin/O=system:masters" -out admin.csr

openssl x509 -req -in admin.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out admin.crt -days 1000

openssl verify -CAfile ca.crt admin.crt



Generate a kubeconfig file for the admin user:

kubectl config set-cluster kubernetes-the-hard-way \

--certificate-authority=ca.crt \

--embed-certs=true \

--server=https://192.168.60.12:6443 \

--kubeconfig=admin.kubeconfig

kubectl config set-credentials admin \

--client-certificate=admin.crt \

--client-key=admin.key \

--embed-certs=true \

--kubeconfig=admin.kubeconfig

kubectl config set-context default \

--cluster=kubernetes-the-hard-way \

--user=admin \

--kubeconfig=admin.kubeconfig

kubectl config use-context default --kubeconfig=admin.kubeconfig

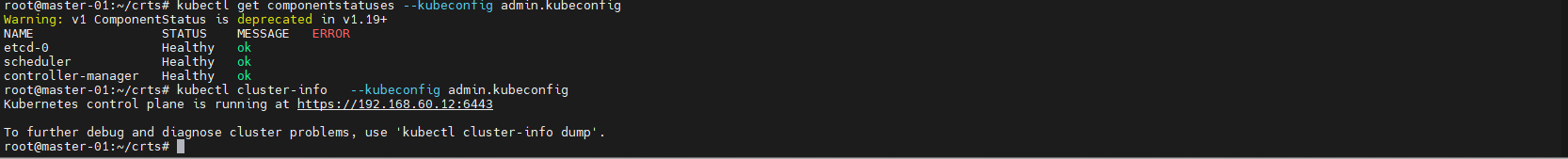
A black screen with white text

AI-generated content may be incorrect.

kubectl get componentstatuses --kubeconfig admin.kubeconfig

kubectl cluster-info --kubeconfig admin.kubeconfig

cp admin.kubeconfig ../.kube/config



For tls bootscraping worker node

Step 1 Create the Boostrap Token to be used by Nodes (Kubelets) to invoke Certificate API

EXPIRATION=$(date -u --date "+7 days" +"%Y-%m-%dT%H:%M:%SZ")

echo $EXPIRATION

#vim bootstrap-token-07401b.yaml

apiVersion: v1

kind: Secret

metadata:

  # Name MUST be of form "bootstrap-token-<token id>"

  name: bootstrap-token-07401b

  namespace: kube-system

# Type MUST be 'bootstrap.kubernetes.io/token'

type: bootstrap.kubernetes.io/token

stringData:

  # Human readable description. Optional.

  description: "The default bootstrap token generated by 'kubeadm init'."

  # Token ID and secret. Required.

  token-id: 07401b

  token-secret: f395accd246ae52d

  # Expiration. Optional.

  expiration: 2025-05-17T03:59:27Z

  # Allowed usages.

  usage-bootstrap-authentication: "true"

  usage-bootstrap-signing: "true"

  # Extra groups to authenticate the token as. Must start with "system:bootstrappers:"

  auth-extra-groups: system:bootstrappers:worker

kubectl apply -f bootstrap-token-07401b.yaml

kubectl get secret -n kube-system



Step 2 Authorize nodes (kubelets) to create CSR

#vim csrs-for-bootstrapping.yaml

# enable bootstrapping nodes to create CSR

kind: ClusterRoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

  name: create-csrs-for-bootstrapping

subjects:

- kind: Group

  name: system:bootstrappers

  apiGroup: rbac.authorization.k8s.io

roleRef:

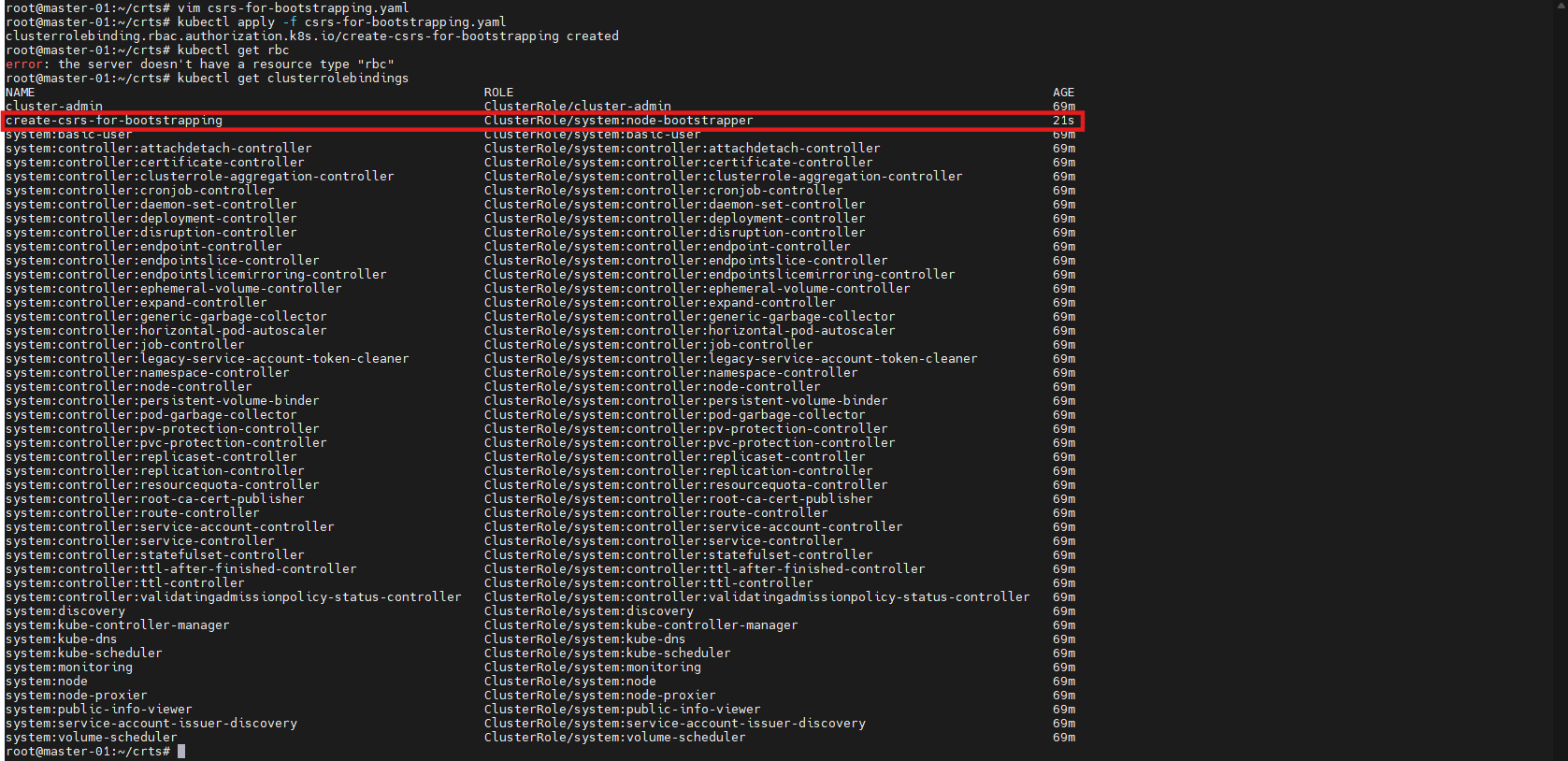
  kind: ClusterRole

  name: system:node-bootstrapper

  apiGroup: rbac.authorization.k8s.io

kubectl apply -f csrs-for-bootstrapping.yaml

kubectl get clusterrolebindings



Step 3 Authorize nodes (kubelets) to approve CSRs

#vim auto-approve-csrs-for-group.yaml

# Approve all CSRs for the group "system:bootstrappers"

kind: ClusterRoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

  name: auto-approve-csrs-for-group

subjects:

- kind: Group

  name: system:bootstrappers

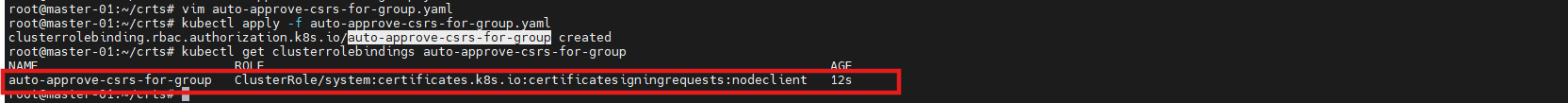
  apiGroup: rbac.authorization.k8s.io

roleRef:

  kind: ClusterRole

  name: system:certificates.k8s.io:certificatesigningrequests:nodeclient

  apiGroup: rbac.authorization.k8s.io



Step 4 Authorize nodes (kubelets) to Auto Renew Certificates on expiration

#vim auto-approve-renewals-for-nodes.yaml

# Approve renewal CSRs for the group "system:nodes"

kind: ClusterRoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: auto-approve-renewals-for-nodes

subjects:

- kind: Group

name: system:nodes

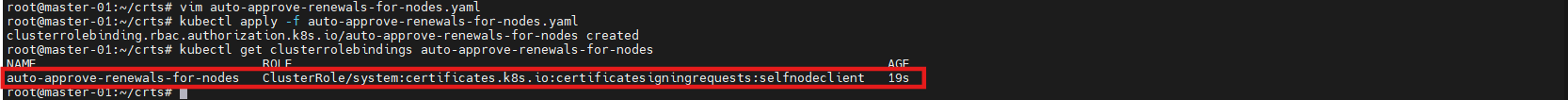
apiGroup: rbac.authorization.k8s.io

roleRef:

kind: ClusterRole

name: system:certificates.k8s.io:certificatesigningrequests:selfnodeclient

apiGroup: rbac.authorization.k8s.io



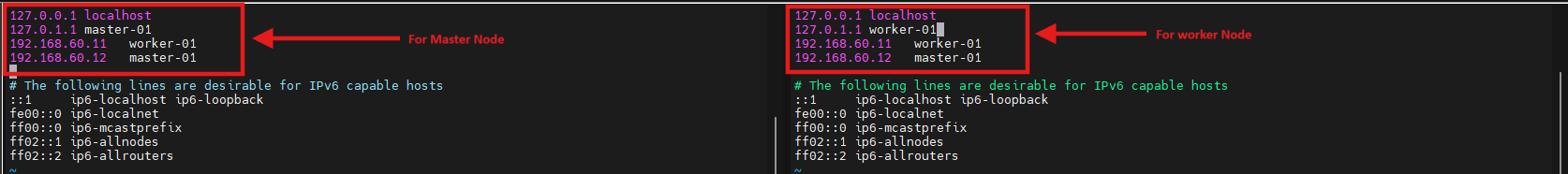
Step 5 Configure the Binaries on the Worker node

Lets ssh on the worker node

Now I will provition a worker node from scratch

we are going to create a tls bootstrapping a worker node

vim /etc/hosts



A black rectangle with white text

AI-generated content may be incorrect.

Download CNI , Containerd, crictl,runc

wget https://github.com/containernetworking/plugins/releases/download/v1.6.2/cni-plugins-linux-amd64-v1.6.2.tgz

wget <https://github.com/containerd/containerd/releases/download/v2.1.0-beta.0/containerd-2.1.0-beta.0-linux-amd64.tar.gz>

wget https://github.com/kubernetes-sigs/cri-tools/releases/download/v1.32.0/crictl-v1.32.0-linux-amd64.tar.gz

wget <https://github.com/opencontainers/runc/releases/download/v1.3.0-rc.1/runc.amd64>

A red square with black background

AI-generated content may be incorrect.

mkdir -p \

/etc/cni/net.d \

/opt/cni/bin \

/var/lib/kubelet \

/var/lib/kube-proxy \

/var/lib/Kubernetes/pki \

/var/run/Kubernetes

cd /opt/cni/bin/

tar -xvf /root/cni-plugins-linux-amd64-v1.6.2.tgz

A black background with white and blue dots

AI-generated content may be incorrect.

apt install -y runc

cd ~

tar -xvf containerd-2.1.0-beta.0-linux-amd64.tar.gz

cd bin/

mv containerd\* /bin

A screenshot of a computer

AI-generated content may be incorrect.

mv bin/ctr /usr/local/bin/

chmod +x runc.amd64

mv runc.amd64 /usr/local/bin/

tar -xvf crictl-v1.32.0-linux-amd64.tar.gz

mv crictl /usr/local/bin/

ls /usr/local/bin/

A black screen with red and white dots

AI-generated content may be incorrect.

To ensure network traffic crossing the CNI bridge network is processed by iptables, load and configure the br-netfilter kernel module:

modprobe br-netfilter

echo "br-netfilter" >> /etc/modules-load.d/modules.conf

echo "net.bridge.bridge-nf-call-iptables = 1" \

>> /etc/sysctl.d/kubernetes.conf

echo "net.bridge.bridge-nf-call-ip6tables = 1" \

>> /etc/sysctl.d/kubernetes.conf

sysctl -p /etc/sysctl.d/kubernetes.conf

mkdir -p /etc/containerd/

cd /etc/containerd/

#vim /etc/containerd/config.toml

version = 2

[plugins."io.containerd.grpc.v1.cri"]

[plugins."io.containerd.grpc.v1.cri".containerd]

snapshotter = "overlayfs"

default\_runtime\_name = "runc"

[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc]

runtime\_type = "io.containerd.runc.v2"

[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]

SystemdCgroup = true

[plugins."io.containerd.grpc.v1.cri".cni]

bin\_dir = "/opt/cni/bin"

conf\_dir = "/etc/cni/net.d"

#vim /etc/systemd/system/containerd.service

[Unit]

Description=containerd container runtime

Documentation=https://containerd.io

After=network.target local-fs.target

[Service]

ExecStartPre=-/sbin/modprobe overlay

ExecStart=/bin/containerd

Restart=always

RestartSec=5

Delegate=yes

KillMode=process

OOMScoreAdjust=-999

#LimitNOFILE=1048576

LimitNPROC=infinity

LimitCORE=infinity

[Install]

WantedBy=multi-user.target

systemctl daemon-reload

systemctl start containerd

systemctl enable containerd

systemctl status containerd

A screen shot of a computer

AI-generated content may be incorrect.

Download kubeproxy and kubelet

wget https://dl.k8s.io/v1.32.3/bin/linux/amd64/kube-proxy

wget <https://dl.k8s.io/v1.32.3/bin/linux/amd64/kubelet>

chmod +x kubelet kube-proxy

mv kubelet kube-proxy /usr/local/bin



For kube proxy we need to create certificate so we are going back to masternode and create certificate and copy to worker node /var/lib/Kubernetes/pki

openssl genpkey -algorithm RSA -out kube-proxy.key -pkeyopt rsa\_keygen\_bits:2048

openssl req -new -key kube-proxy.key -subj "/CN=system:kube-proxy/O=system:node-proxier" -out kube-proxy.csr

openssl x509 -req -in kube-proxy.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out kube-proxy.crt -days 1000

openssl verify -CAfile ca.crt kube-proxy.crt

A blurry image of a line of colorful lights

AI-generated content may be incorrect.

Now transfer kube-proxy.crt, kube-proxy.key, ca.crt to worker-01 to path /var/lib/Kubernetes/pki



From master node we have to create kube-proxy.kubeconfig and kubeconfig

{

kubectl config set-cluster kubernetes-the-hard-way \

--certificate-authority=/var/lib/kubernetes/pki/ca.crt \

--server=https://192.168.60.12:6443 \

--kubeconfig=kube-proxy.kubeconfig

kubectl config set-credentials system:kube-proxy \

--client-certificate=/var/lib/kubernetes/pki/kube-proxy.crt \

--client-key=/var/lib/kubernetes/pki/kube-proxy.key \

--kubeconfig=kube-proxy.kubeconfig

kubectl config set-context default \

--cluster=kubernetes-the-hard-way \

--user=system:kube-proxy \

--kubeconfig=kube-proxy.kubeconfig

kubectl config use-context default --kubeconfig=kube-proxy.kubeconfig

}



Now copy kube-proxy.kubeconfig from masternode to worker node /var/lib/kube-proxy

#vim /var/lib/kube-proxy/kube-proxy.kubeconfig

apiVersion: v1

clusters:

- cluster:

certificate-authority: /var/lib/kubernetes/pki/ca.crt

server: https://192.168.60.12:6443

name: kubernetes-the-hard-way

contexts:

- context:

cluster: kubernetes-the-hard-way

user: system:kube-proxy

name: default

current-context: default

kind: Config

preferences: {}

users:

- name: system:kube-proxy

user:

client-certificate: /var/lib/kubernetes/pki/kube-proxy.crt

client-key: /var/lib/kubernetes/pki/kube-proxy.key

#vim /var/lib/kube-proxy/kube-proxy-config.yaml

kind: KubeProxyConfiguration

apiVersion: kubeproxy.config.k8s.io/v1alpha1

clientConnection:

kubeconfig: /var/lib/kube-proxy/kube-proxy.kubeconfig

mode: iptables

clusterCIDR: 10.244.0.0/16

#vim /etc/systemd/system/kube-proxy.service

[Unit]

Description=Kubernetes Kube Proxy

Documentation=https://github.com/kubernetes/kubernetes

[Service]

ExecStart=/usr/local/bin/kube-proxy \

--config=/var/lib/kube-proxy/kube-proxy-config.yaml

Restart=on-failure

RestartSec=5

[Install]

WantedBy=multi-user.target

A black screen with many colorful lights

AI-generated content may be incorrect.

Install kubelet

#vim /var/lib/kubelet/kubelet-config.yaml

kind: KubeletConfiguration

apiVersion: kubelet.config.k8s.io/v1beta1

authentication:

anonymous:

enabled: false

webhook:

enabled: true

x509:

clientCAFile: /var/lib/kubernetes/pki/ca.crt

authorization:

mode: Webhook

containerRuntimeEndpoint: unix:///var/run/containerd/containerd.sock

cgroupDriver: systemd

clusterDomain: "cluster.local"

clusterDNS:

- 10.96.0.10

registerNode: true

resolvConf: /run/systemd/resolve/resolv.conf

rotateCertificates: true

runtimeRequestTimeout: "15m"

serverTLSBootstrap: true

#vim /var/lib/kubelet/bootstrap-kubeconfig

apiVersion: v1

clusters:

- cluster:

    certificate-authority: /var/lib/kubernetes/pki/ca.crt

    server: https://192.168.60.12:6443

  name: bootstrap

contexts:

- context:

    cluster: bootstrap

    user: kubelet-bootstrap

  name: bootstrap

current-context: bootstrap

kind: Config

preferences: {}

users:

- name: kubelet-bootstrap

  user:

    token: 07401b.f395accd246ae52d



# vim /etc/systemd/system/kubelet.service

[Unit]

Description=Kubernetes Kubelet

Documentation=https://github.com/kubernetes/kubernetes

After=containerd.service

Requires=containerd.service

[Service]

ExecStart=/usr/local/bin/kubelet \

--bootstrap-kubeconfig="/var/lib/kubelet/bootstrap-kubeconfig" \

--config=/var/lib/kubelet/kubelet-config.yaml \

--kubeconfig=/var/lib/kubelet/kubeconfig \

--cert-dir=/var/lib/kubelet/pki/ \

--node-ip=192.168.60.11 \

--v=2

Restart=on-failure

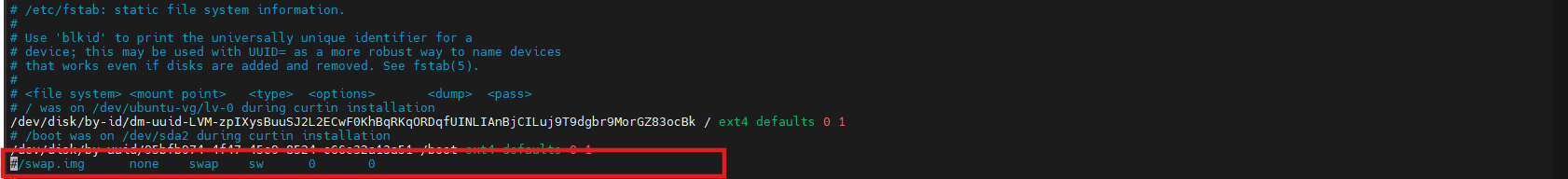
RestartSec=5

[Install]

WantedBy=multi-user.target

swapoff -a

vim /etc/fstab



systemctl daemon-reload

systemctl start kubelet

systemctl enable kubelet

systemctl status kubelet

A computer screen shot of a city

AI-generated content may be incorrect.

Now lets go back to the master node since kubelet and kube proxy is running

Now check node status

Kubectl get nodes

  
its not ready state

We have to approve the certificate from the master node to include in the cluster

kubectl get csr

A red line in a black background

AI-generated content may be incorrect.

kubectl certificate approve csr-pmrvj

Installing calico and coreDNS

mkdir network

cd network

wget <https://raw.githubusercontent.com/projectcalico/calico/v3.27.0/manifests/calico.yaml>

vim calico.yaml #add thi line given below

4971 initialDelaySeconds: 10 #Increase from default 5s

kubectl apply -f calico.yaml

A screen shot of a computer

AI-generated content may be incorrect.

wget <https://raw.githubusercontent.com/mmumshad/kubernetes-the-hard-way/master/deployments/coredns.yaml>

kubectl apply -f coredns.yaml



Test corDNS working or not

kubectl run busybox -n default --image=busybox:1.28 --restart Never --command -- sleep 180

kubectl get pods

kubectl exec -ti -n default busybox -- nslookup Kubernetes

A screen shot of a computer

AI-generated content may be incorrect.

Smoke test

kubectl create deployment nginx --image=nginx:alpine

kubectl expose deploy nginx --type=NodePort --port 80

kubectl get services

curl 192.168.60.11:31560

A black screen with white lines

AI-generated content may be incorrect.

Deploy metrics-server server in kubernatis

Step1:create certificates on /var/lib/kubernetes/pki where I have ca.crt and ca.key

openssl genpkey -algorithm RSA -out front-proxy-client.key -pkeyopt rsa\_keygen\_bits:2048

openssl req -new -key front-proxy-client.key -subj "/CN=front-proxy-client" -out front-proxy-client.csr

openssl x509 -req -in front-proxy-client.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out front-proxy-client.crt -days 365 -extensions v3\_req -extfile <(cat <<EOF

[ v3\_req ]

basicConstraints = CA:FALSE

keyUsage = digitalSignature, keyEncipherment

extendedKeyUsage = clientAuth

EOF

)

Step2: change kubeapi server configuration from directory

[Unit]

Description=Kubernetes API Server

Documentation=https://github.com/kubernetes/kubernetes

[Service]

ExecStart=/usr/local/bin/kube-apiserver \

--advertise-address=192.168.60.12 \

#-------------------For Matrics Server --------------------------

--enable-aggregator-routing=true \

--requestheader-client-ca-file=/var/lib/kubernetes/pki/ca.crt \

--proxy-client-cert-file=/var/lib/kubernetes/pki/front-proxy-client.crt \

--proxy-client-key-file=/var/lib/kubernetes/pki/front-proxy-client.key \

--requestheader-allowed-names=front-proxy-client \

--requestheader-extra-headers-prefix=X-Remote-Extra- \

--requestheader-group-headers=X-Remote-Group \

--requestheader-username-headers=X-Remote-User \

#-------------------- END ---------------------------------------

--allow-privileged=true \

--apiserver-count=2 \

--audit-log-maxage=30 \

--audit-log-maxbackup=3 \

--audit-log-maxsize=100 \

--audit-log-path=/var/log/audit.log \

--authorization-mode=Node,RBAC \

--bind-address=0.0.0.0 \

--client-ca-file=/var/lib/kubernetes/pki/ca.crt \

--enable-admission-plugins=NodeRestriction,ServiceAccount \

--enable-bootstrap-token-auth=true \

--etcd-cafile=/var/lib/kubernetes/pki/ca.crt \

--etcd-certfile=/var/lib/kubernetes/pki/etcd-server.crt \

--etcd-keyfile=/var/lib/kubernetes/pki/etcd-server.key \

--etcd-servers=https://192.168.60.12:2379\

--event-ttl=1h \

--encryption-provider-config=/var/lib/kubernetes/encryption-config.yaml \

--kubelet-certificate-authority=/var/lib/kubernetes/pki/ca.crt \

--kubelet-client-certificate=/var/lib/kubernetes/pki/apiserver-kubelet-client.crt \

--kubelet-client-key=/var/lib/kubernetes/pki/apiserver-kubelet-client.key \

--runtime-config=api/all=true \

--service-account-key-file=/var/lib/kubernetes/pki/service-account.crt \

--service-account-signing-key-file=/var/lib/kubernetes/pki/service-account.key \

--service-account-issuer=https://192.168.60.12:6443 \

--service-cluster-ip-range=10.96.0.0/16 \

--service-node-port-range=30000-32767 \

--tls-cert-file=/var/lib/kubernetes/pki/kube-apiserver.crt \

--tls-private-key-file=/var/lib/kubernetes/pki/kube-apiserver.key \

--v=2

Restart=on-failure

RestartSec=5

[Install]

WantedBy=multi-user.target

systemctl daemon-reexec

systemctl daemon-reload

systemctl restart kube-apiserver

kubectl get apiservice v1beta1.metrics.k8s.io -o yaml

|  |
| --- |
| status:  conditions:  - type: Available  status: "True" |

Then you will deploy matrix server using yaml file

kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml

Then change some configuration

|  |
| --- |
| kubectl patch deployment metrics-server -n kube-system --type='json' -p='[  {  "op": "add",  "path": "/spec/template/spec/hostNetwork",  "value": true  },  {  "op": "replace",  "path": "/spec/template/spec/containers/0/args",  "value": [  "--cert-dir=/tmp",  "--secure-port=4443",  "--kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname",  "--kubelet-use-node-status-port",  "--metric-resolution=15s",  "--kubelet-insecure-tls"  ]  },  {  "op": "replace",  "path": "/spec/template/spec/containers/0/ports/0/containerPort",  "value": 4443  }  ]' |

|  |
| --- |
| kubectl -n kube-system get pods -l k8s-app=metrics-server  kubectl get apiservices -l k8s-app=metrics-server  root@master-01:/var/lib/kubernetes/pki# kubectl get apiservices -l k8s-app=metrics-server  NAME SERVICE AVAILABLE AGE  v1beta1.metrics.k8s.io kube-system/metrics-server True 41m  root@master-01:/var/lib/kubernetes/pki# kubectl top node  NAME CPU(cores) CPU(%) MEMORY(bytes) MEMORY(%)  worker-01 59m 0% 962Mi 29%  root@master-01:/var/lib/kubernetes/pki# |