

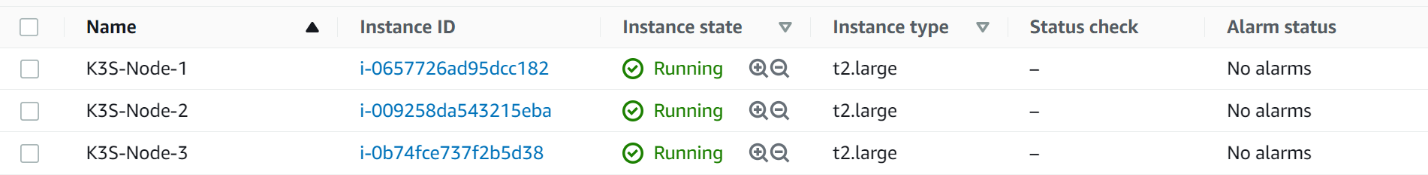
Deploying a Highly Available K3S Cluster:

For a Highly Available Kubernetes cluster, we need to have multiple ETCD databases for fault tolerance. This ETCD cluster can be deployed as a separate cluster or can be installed along with master servers.

Two stages are involved for creating a Highly Available Kubernetes Cluster:

1. Creating a multi node ETCD Cluster.
2. Creating a multi-master Kubernetes cluster.

Deploy 3 Ubuntu Servers with 2CPU and 4GB or 8GB Memory and Storage depend on the project requirements. We also need to create certificates for each node and also copy ca.pem as ectd-ca.crt to all nodes in /etc/etcd/.



Creating Multi-Node ETD Cluster:



Creating Multi-Master K8S Cluster:

Once the ETCD cluster is created , run the following on all the 3 nodes:

#!/bin/bash

curl https://get.docker.com/ | bash

usermod -a -G root ubuntu

usermod -a -G docker ubuntu

sudo systemctl daemon-reload

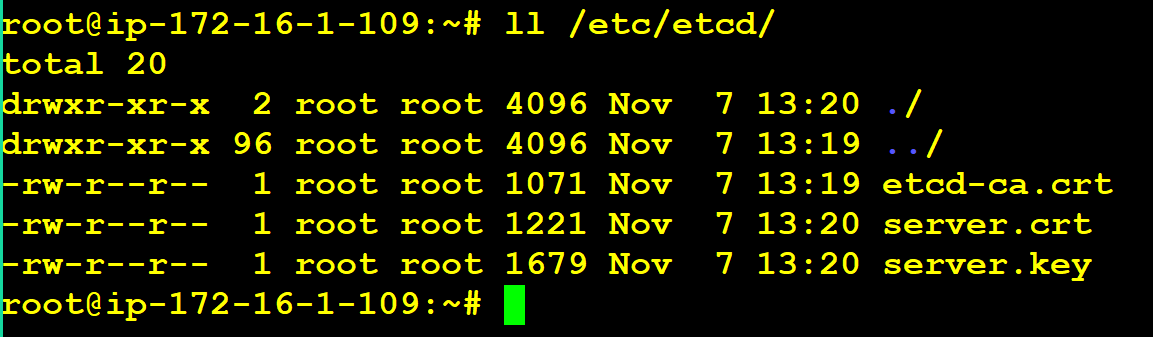
sudo systemctl restart docker

For each node we need to generate certificates as we did for ectd-cluster and copy to /etc/etcd.

export NAME=ip-172-16-1-109

export ADDRESS=172.16.1.109,$NAME

echo '{"CN":"'$NAME'","hosts":[""],"key":{"algo":"rsa","size":2048}}' | cfssl gencert -config=ca-config.json -ca=ca.pem -ca-key=ca-key.pem -hostname="$ADDRESS" - | cfssljson -bare $NAME



Add Following in the .bashrc so that configuration is saved when server reboots.

export K3S\_DATASTORE\_ENDPOINT='https://10.20.1.170:2379,https://10.20.1.201:2379,https://10.20.1.92:2379'

export K3S\_DATASTORE\_CAFILE='/etc/etcd/etcd-ca.crt'

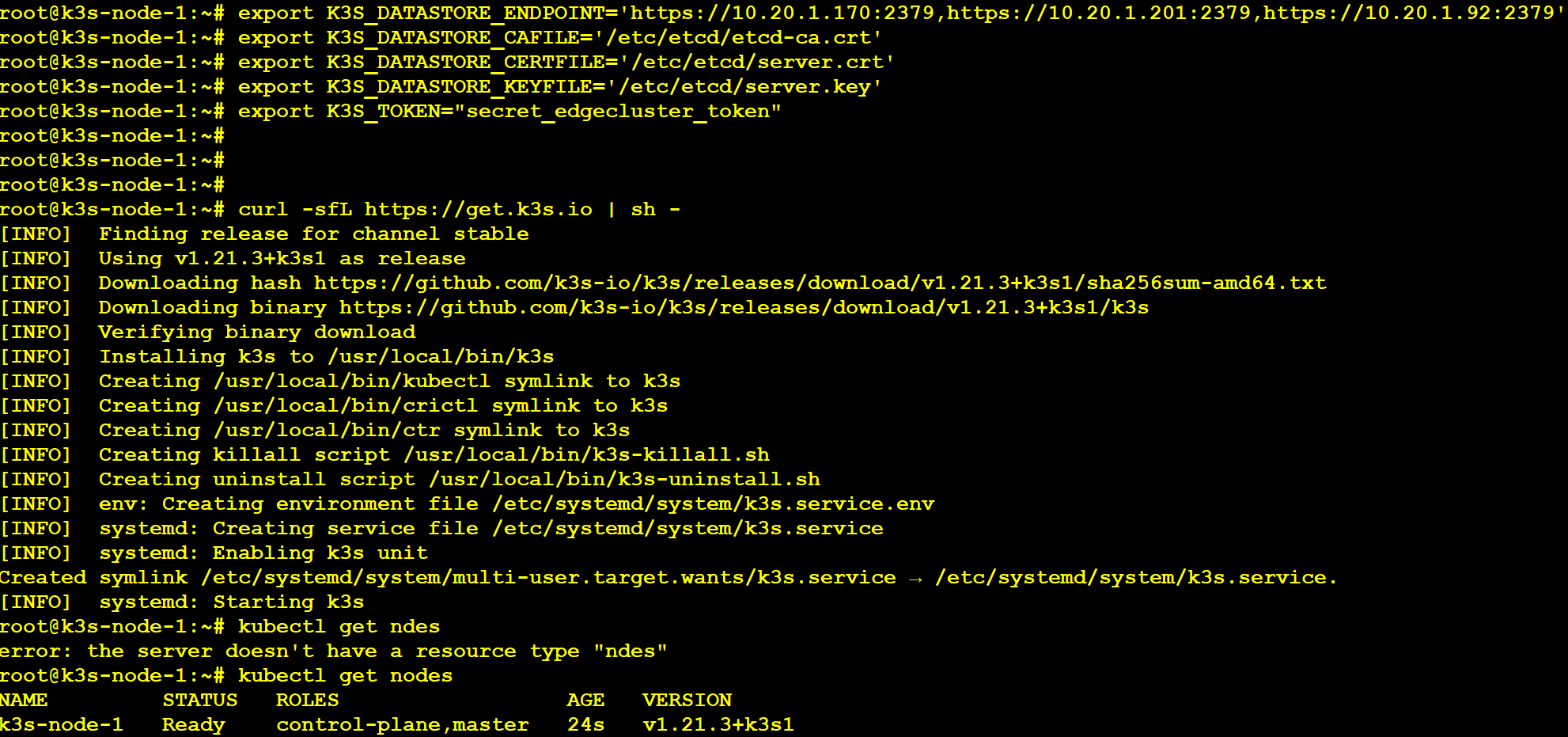
export K3S\_DATASTORE\_CERTFILE='/etc/etcd/server.crt'

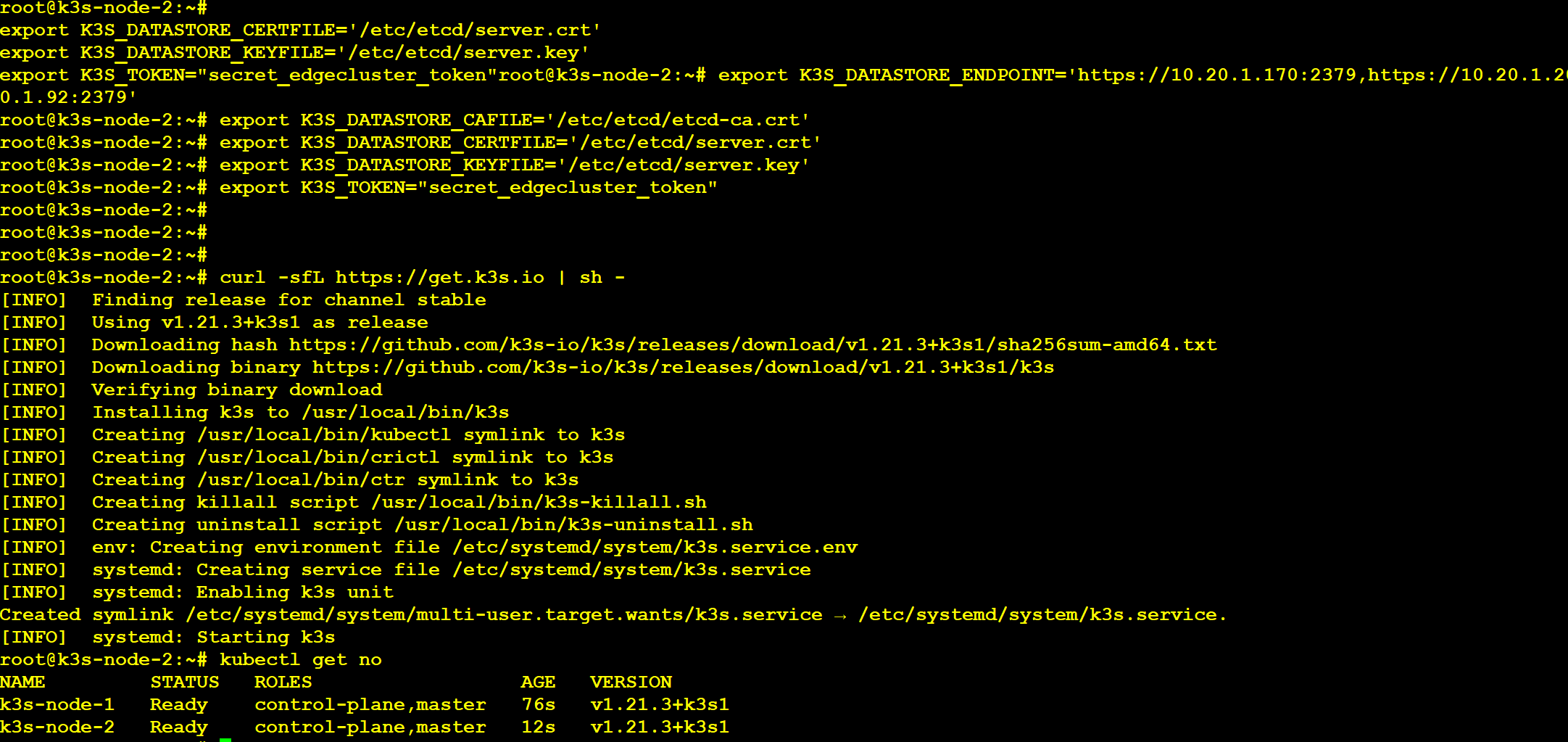
export K3S\_DATASTORE\_KEYFILE='/etc/etcd/server.key'

export K3S\_TOKEN="secret\_edgecluster\_token"

Run the k3s binary on all 3 nodes:

curl -sfL https://get.k3s.io | sh –







apt install -y jq

kubectl run nginx --image sreeharshav/rollingupdate:v5

etcdctl --endpoints https://10.20.1.170:2379 --cert /etc/etcd/server.crt --cacert /etc/etcd/etcd-ca.crt --key /etc/etcd/server.key get /registry/pods/default/nginx --prefix=true -w json

etcdctl --endpoints https://10.20.1.170:2379 --cert /etc/etcd/server.crt --cacert /etc/etcd/etcd-ca.crt --key /etc/etcd/server.key get /registry/pods/default/nginx --prefix=true -w json | jq -r .kvs[].value

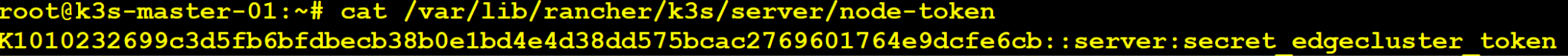
etcdctl --endpoints https://10.20.1.170:2379 --cert /etc/etcd/server.crt --cacert /etc/etcd/etcd-ca.crt --key /etc/etcd/server.key get /registry/pods/default/nginx --prefix=true -w json | jq -r .kvs[].value | base64 -d

export K3S\_TOKEN="secret\_edgecluster\_token"

export K3S\_URL=https://K3S-MASTER-NLB-3d91395c69dc7f68.elb.us-east-1.amazonaws.com:6443

curl -sfL https://get.k3s.io | sh –

/usr/local/bin/k3s-uninstall.sh





Internal aws network load balancer is enough for master server load balancing.

#!/bin/bash

curl https://get.docker.com/ | bash

usermod -a -G root ubuntu

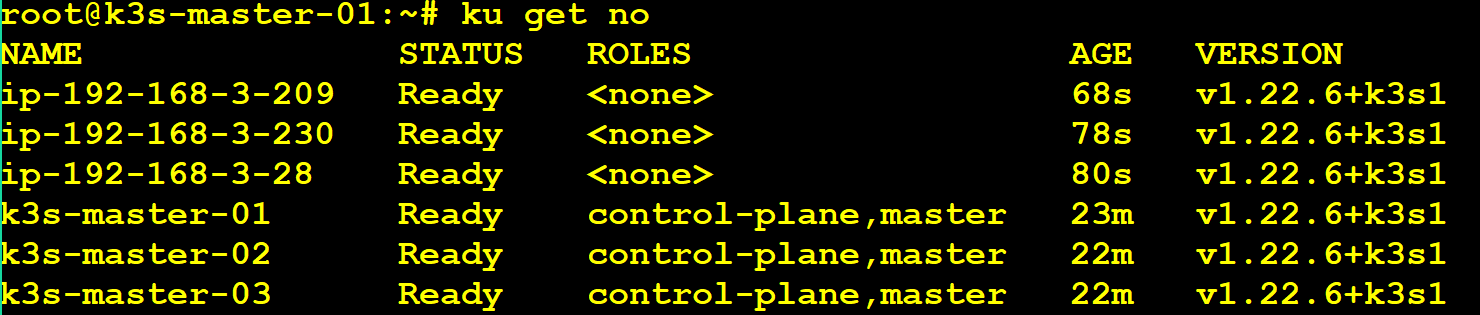
usermod -a -G docker ubuntu

sudo systemctl daemon-reload

sudo systemctl restart docker

TOKEN='K1010232699c3d5fb6bfdbecb38b0e1bd4e4d38dd575bcac2769601764e9dcfe6cb::server:secret\_edgecluster\_token'

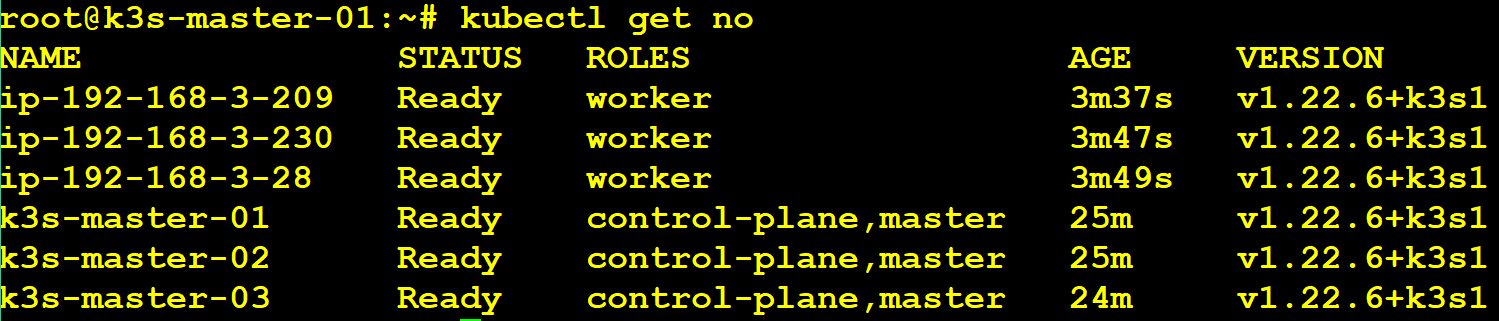
curl -sfL https://get.k3s.io | K3S\_URL=https://k3s-masters-lb-2dc2480295074dfa.elb.us-east-1.amazonaws.com:6443 K3S\_TOKEN=$TOKEN sh -



ku label node ip-192-168-3-28 node-role.kubernetes.io/worker=true

ku label node ip-192-168-3-209 node-role.kubernetes.io/worker=true

ku label node ip-192-168-3-230 node-role.kubernetes.io/worker=true



apiVersion: v1

kind: Pod

metadata:

labels:

run: nginx01

name: nginx01

spec:

containers:

- image: sreeharshav/rollingupdate:v5

name: nginx01

nodeSelector:

node-role.kubernetes.io/worker: "true"

---

apiVersion: v1

kind: Pod

metadata:

labels:

run: nginx02

name: nginx02

spec:

containers:

- image: sreeharshav/rollingupdate:v5

name: nginx02

nodeSelector:

node-role.kubernetes.io/worker: "true"

---

apiVersion: v1

kind: Pod

metadata:

labels:

run: nginx03

name: nginx03

spec:

containers:

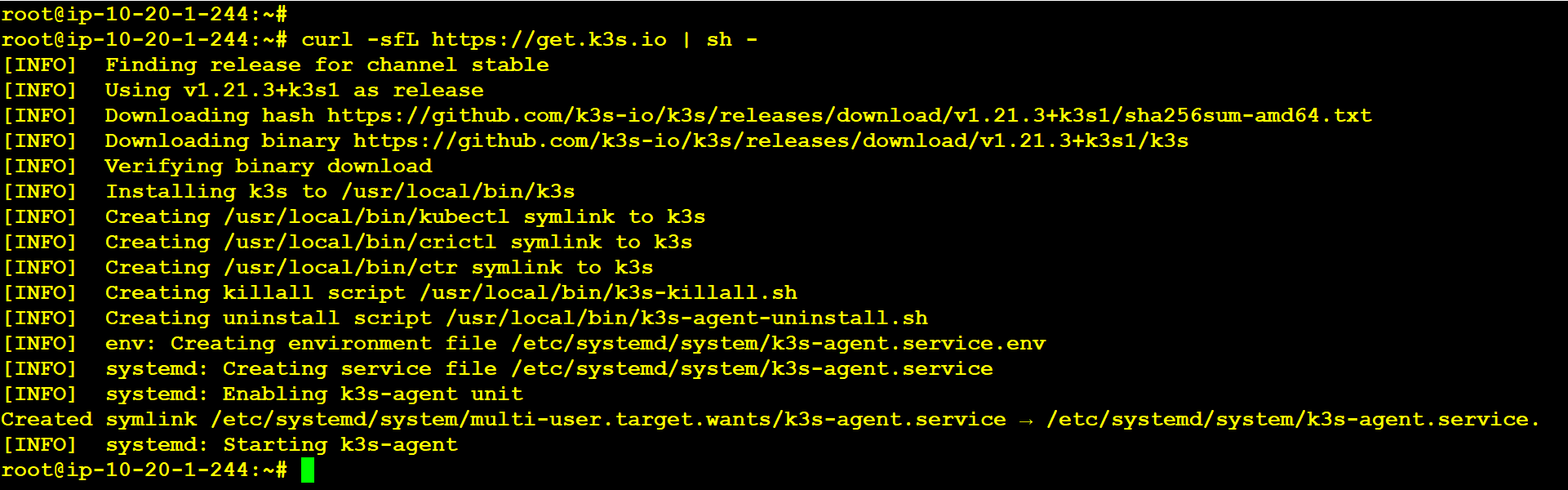
- image: sreeharshav/rollingupdate:v5

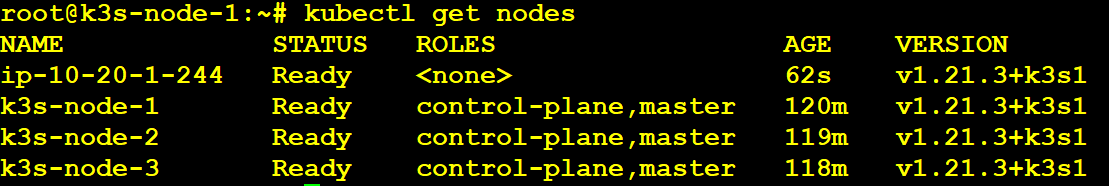
name: nginx03

nodeSelector:

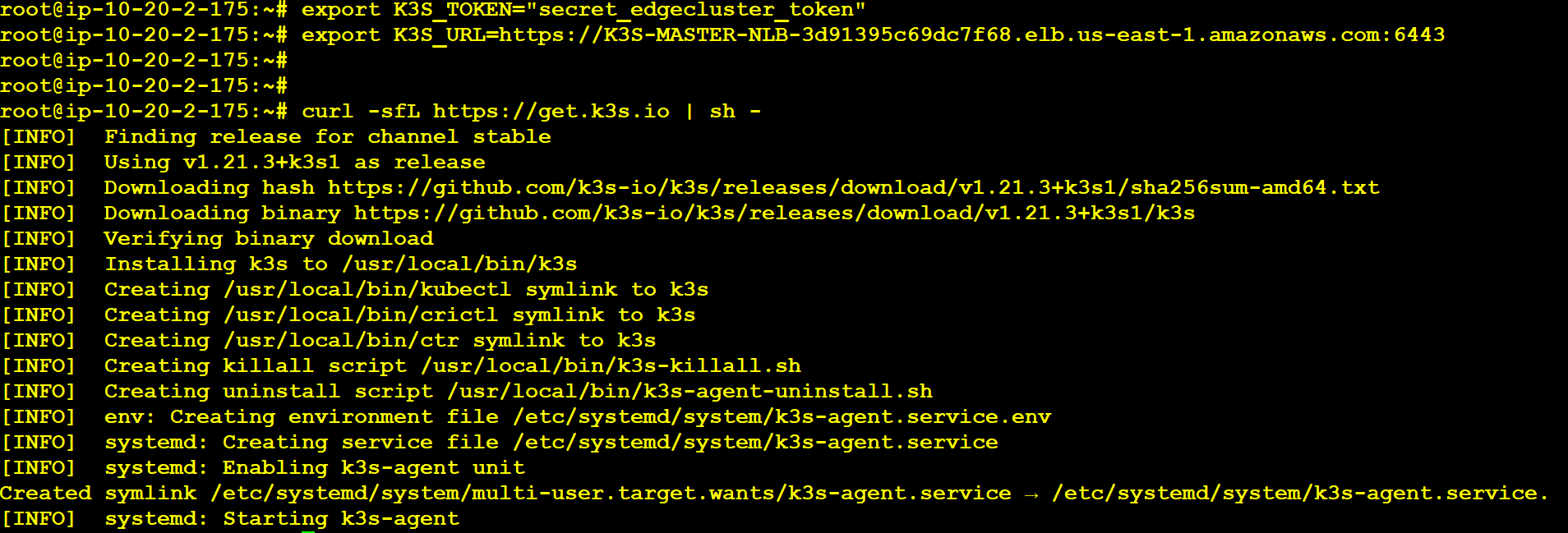
node-role.kubernetes.io/worker: "true"

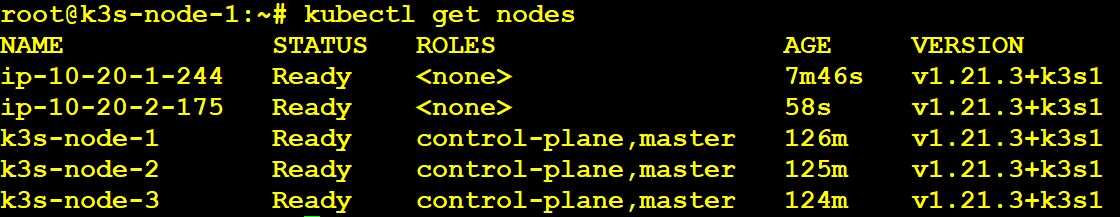
**Adding Worker Node 1**



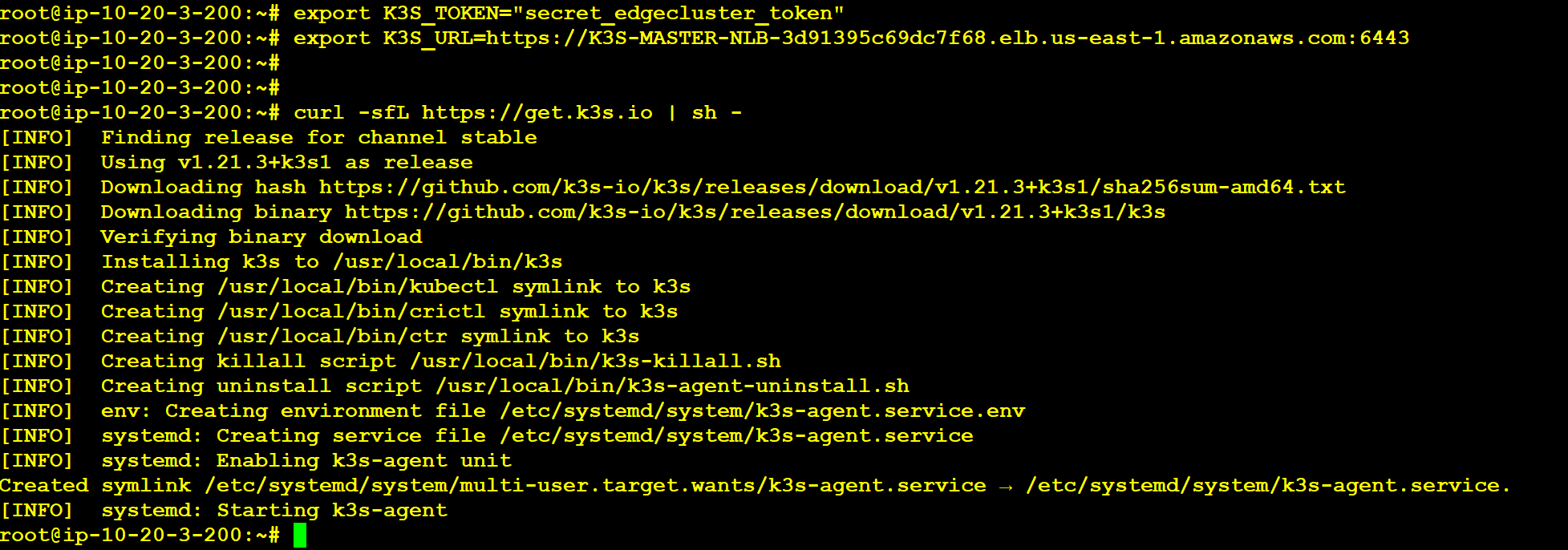


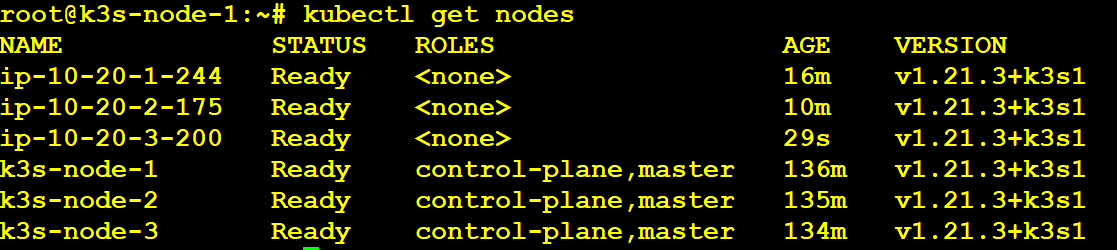
**Adding Worker Node 2**

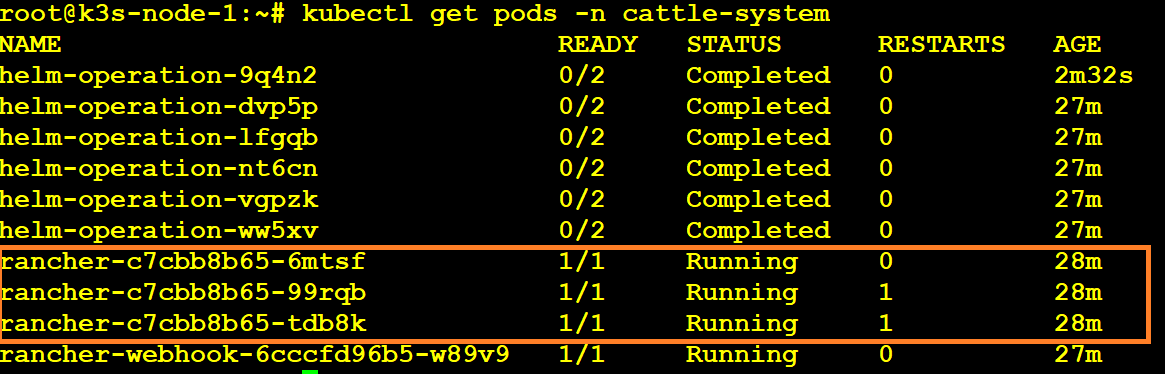




**Adding Worker Node 3**







**Installing Rancher on a Kubernetes Cluster Using Helm:**

<https://rancher.com/docs/rancher/v2.x/en/installation/other-installation-methods/behind-proxy/install-rancher/>

| helm3 error using k3s: |
| --- |
|  |

Error: Kubernetes cluster unreachable: Get "http://localhost:8080/version?timeout=32s": dial tcp 127.0.0.1:8080: connect: connection refused

| Above Issue Resolved By Creating KUBECONFIG as below: |
| --- |
|  |

export KUBECONFIG=/etc/rancher/k3s/k3s.yaml

helm repo add jetstack https://charts.jetstack.io

kubectl create namespace cert-manager

kubectl apply --validate=false -f https://github.com/jetstack/cert-manager/releases/download/v0.15.2/cert-manager.crds.yaml

helm upgrade --install cert-manager jetstack/cert-manager --namespace cert-manager --version v0.15.2 --set http\_proxy=http://${proxy\_host} --set https\_proxy=http://${proxy\_host} --set no\_proxy=127.0.0.0/8\\,10.0.0.0/8\\,cattle-system.svc\\,172.16.0.0/12\\,192.168.0.0/16\\,.svc\\,.cluster.local

helm repo add rancher-latest https://releases.rancher.com/server-charts/latest

kubectl create namespace cattle-system

https://thenewstack.io/tutorial-set-up-a-secure-and-highly-available-etcd-cluster/

https://thenewstack.io/tutorial-install-a-highly-available-k3s-cluster-at-the-edge/

helm upgrade --install rancher rancher-latest/rancher \

--namespace cattle-system \

--set hostname=rancher.example.com \

--set proxy=http://${proxy\_host} \

--set no\_proxy=127.0.0.0/8\\,10.0.0.0/8\\,cattle-system.svc\\,172.16.0.0/12\\,192.168.0.0/16\\,.svc\\,.cluster.local

