1. Deploy a Kubernetes cluster for 3 nodes
2. Create a NGINX deployment of 3 replicas

1. Deploy a Kubernetes cluster for 3 nodes

INSTALL 3 LINUX SERVERS; USING minimum of t2-Medium / 20gb hard disk

On all the nodes install all the prerequisites for kubernetes cluster creation: kubernetes.sh

vi kubernetes.sh

```
sudo yum install -y docker
sudo systemctl enable docker
sudo systemctl start docker
sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.28/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.28/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
sudo systemctl enable --now kubelet
VER=$(curl -s https://api.github.com/repos/Mirantis/cri-dockerd/releases/latest|grep tag_name | cut -d
"" -f 4|sed 's/v//g')
wget https://github.com/Mirantis/cri-dockerd/releases/download/v${VER}/cri-dockerd-$
{VER}.amd64.tgz
tar xvf cri-dockerd-${VER}.amd64.tgz
sudo mv cri-dockerd/cri-dockerd /usr/local/bin/
wget https://raw.githubusercontent.com/Mirantis/cri-dockerd/master/packaging/systemd/cri-
docker.service
wget https://raw.githubusercontent.com/Mirantis/cri-dockerd/master/packaging/systemd/cri-
docker.socket
sudo mv cri-docker.socket cri-docker.service /etc/systemd/system/
sudo sed -i -e 's,/usr/bin/cri-dockerd,/usr/local/bin/cri-dockerd,' /etc/systemd/system/cri-docker.service
sudo systemctl daemon-reload
sudo systemctl enable cri-docker.service
sudo systemctl enable --now cri-docker.socket
sudo kubeadm config images pull --cri-socket unix:///var/run/cri-dockerd.sock
```

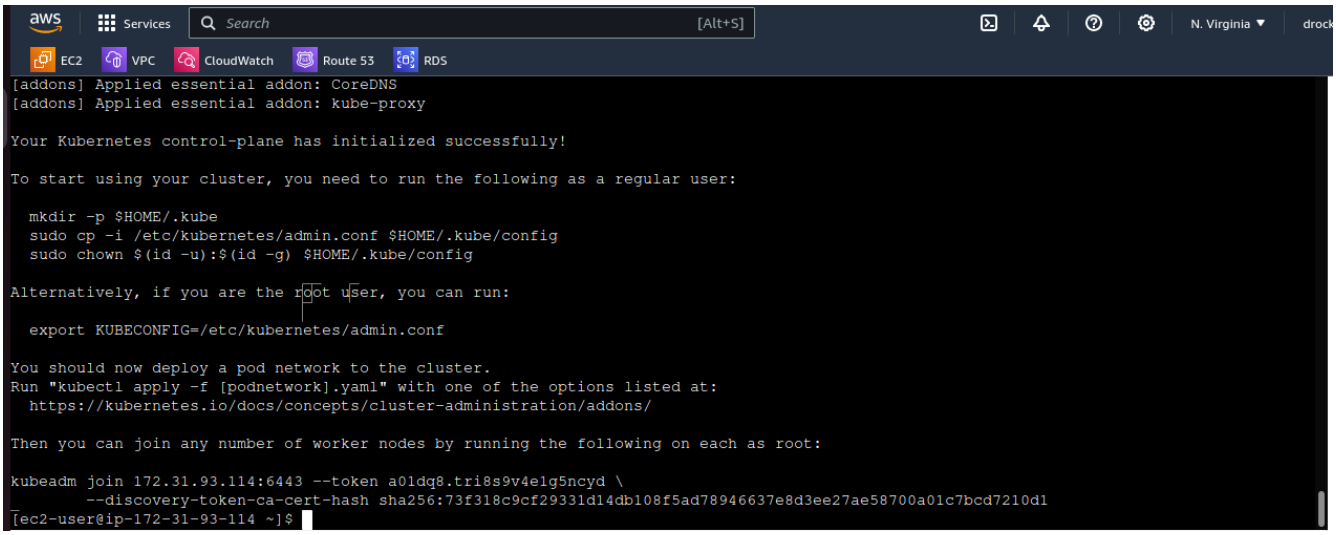
chmod +x kubernetes.sh
./kubernetes.sh

```
aws  Services  Search [Alt+S]  N. Virginia ▾  drock
EC2  VPC  CloudWatch  Route 53  RDS
      _/m/'
[ec2-user@ip-172-31-93-114 ~]$ vi kubernetes.sh
[ec2-user@ip-172-31-93-114 ~]$ sudo chmod +x kubernetes.sh
[ec2-user@ip-172-31-93-114 ~]$ ./kubernetes.sh
Last metadata expiration check: 0:05:40 ago on Fri Mar  8 10:44:32 2024.
Dependencies resolved.
================================================================================================================================
 Package                          Architecture          Version                              Repository                  Size
================================================================================================================================
Installing:
 docker                           x86_64                25.0.3-1.amzn2023.0.1                amazonlinux                  44 M
Installing dependencies:
 containerd                       x86_64                1.7.11-1.amzn2023.0.1                amazonlinux                  35 M
 iptables-libs                    x86_64                1.8.8-3.amzn2023.0.2                 amazonlinux                 401 k
 iptables-nft                     x86_64                1.8.8-3.amzn2023.0.2                 amazonlinux                 183 k
 libcgroup                        x86_64                3.0-1.amzn2023.0.1                   amazonlinux                  75 k
 libnetfilter_conntrack           x86_64                1.0.8-2.amzn2023.0.2                 amazonlinux                  58 k
 libnfnetlink                     x86_64                1.0.1-19.amzn2023.0.2                amazonlinux                  30 k
 libnftnl                         x86_64                1.2.2-2.amzn2023.0.2                 amazonlinux                  84 k
 pigz                             x86_64                2.5-1.amzn2023.0.3                   amazonlinux                  83 k
 runc                             x86_64                1.1.11-1.amzn2023.0.1                amazonlinux                 3.0 M

Transaction Summary
================================================================================================================================
```

```
aws  Services  Search [Alt+S]  N. Virginia ▾  drock
EC2  VPC  CloudWatch  Route 53  RDS
2024-03-08 10:50:46 (74.4 MB/s) - 'cri-docker.service' saved [1319/1319]

--2024-03-08 10:50:46--  https://raw.githubusercontent.com/Mirantis/cri-dockerd/master/packaging/systemd/cri-docker.socket
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 204 [text/plain]
Saving to: 'cri-docker.socket'

cri-docker.socket                   100%[===========================================================>]     204  --.-KB/s    in 0s

2024-03-08 10:50:46 (12.8 MB/s) - 'cri-docker.socket' saved [204/204]

Created symlink /etc/systemd/system/multi-user.target.wants/cri-docker.service → /etc/systemd/system/cri-docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/cri-docker.socket → /etc/systemd/system/cri-docker.socket.
I0308 10:50:48.352369   27335 version.go:256] remote version is much newer: v1.29.2; falling back to: stable-1.28
[config/images] Pulled registry.k8s.io/kube-apiserver:v1.28.7
[config/images] Pulled registry.k8s.io/kube-controller-manager:v1.28.7
[config/images] Pulled registry.k8s.io/kube-scheduler:v1.28.7
[config/images] Pulled registry.k8s.io/kube-proxy:v1.28.7
[config/images] Pulled registry.k8s.io/pause:3.9
[config/images] Pulled registry.k8s.io/etcd:3.5.10-0
[config/images] Pulled registry.k8s.io/coredns/coredns:v1.10.1
[ec2-user@ip-172-31-93-114 ~]$
```

Run the below command on the master server.
sudo kubeadm init --pod-network-cidr=192.168.0.0/16 --apiserver-advertise-address=172.31.39.26(Change it to the private IP of ur Master Server) --cri-socket unix:///var/run/cri-dockerd.sock

I.e
sudo kubeadm init --pod-network-cidr=192.168.0.0/16 --apiserver-advertise-address=172.31.93.114 --cri-socket unix:///var/run/cri-dockerd.sock



join the worker node in the cluster with this flag
--cri-socket unix:///var/run/cri-dockerd.sock
Example:

sudo kubeadm join 172.31.93.114:6443 --token a01dq8.tri8s9v4e1g5ncyd \
    --discovery-token-ca-cert-hash sha256:73f318c9cf29331d14db108f5ad78946637e8d3ee27ae58700a01c7bcd7210d1 --cri-socket unix:///var/run/cri-dockerd.sock

```
[ec2-user@ip-172-31-91-218 ~]$ sudo kubeadm join 172.31.93.114:6443 --token a01dq8.tri8s9v4e1g5ncyd \
        --discovery-token-ca-cert-hash sha256:73f318c9cf29331d14db108f5ad78946637e8d3ee27ae58700a01c7bcd7210d1 --cri-socket unix:///var/r
un/cri-dockerd.sock
[preflight] Running pre-flight checks
        [WARNING FileExisting-tc]: tc not found in system path
[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.

[ec2-user@ip-172-31-91-218 ~]$
```

i-0e85835b18a623c9c (kub-slave1)

PublicIPs: 44.201.153.44 PrivateIPs: 172.31.91.218

TO GET RID OF SUDO

cd /etc/kubernetes

ls -al

cd ~

mkdir -p $HOME/.kube

cd .kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config .....COPY THE admin.conf to current user path

sudo chown $(id -u):$(id -g) $HOME/.kube/config

kubectl get nodes ,,,,,,,,,NOT READY ,

```
[ec2-user@ip-172-31-93-114 ~]$ cd /etc/kubernetes
[ec2-user@ip-172-31-93-114 kubernetes]$ ls -la
total 60
drwxr-xr-x.  4 root root   125 Mar  8 11:01 .
drwxr-xr-x. 83 root root 16384 Mar  8 10:50 ..
-rw-------.  1 root root  5649 Mar  8 11:01 admin.conf
-rw-------.  1 root root  5677 Mar  8 11:01 controller-manager.conf
-rw-------.  1 root root  2061 Mar  8 11:01 kubelet.conf
drwxr-xr-x.  2 root root   113 Mar  8 11:01 manifests
drwxr-xr-x.  3 root root 16384 Mar  8 11:01 pki
-rw-------.  1 root root  5625 Mar  8 11:01 scheduler.conf
[ec2-user@ip-172-31-93-114 kubernetes]$ mkdir -p $HOME/.kube
[ec2-user@ip-172-31-93-114 kubernetes]$ cd
[ec2-user@ip-172-31-93-114 ~]$ mkdir -p $HOME/.kube
[ec2-user@ip-172-31-93-114 ~]$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[ec2-user@ip-172-31-93-114 ~]$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
[ec2-user@ip-172-31-93-114 ~]$ kubectl get nodes
NAME                           STATUS     ROLES           AGE     VERSION
ip-172-31-81-175.ec2.internal  NotReady   <none>          5m59s   v1.28.7
ip-172-31-91-218.ec2.internal  NotReady   <none>          6m32s   v1.28.7
ip-172-31-93-114.ec2.internal  NotReady   control-plane   12m     v1.28.7
[ec2-user@ip-172-31-93-114 ~]$
```

CALICO NETWORK CONFIGURE

pass the 3 commands:

kubectl create -f https://raw.githubusercontent.com/projectcalico/calico/v3.27.0/manifests/tigera-operator.yaml

curl https://raw.githubusercontent.com/projectcalico/calico/v3.27.0/manifests/custom-resources.yaml -O

kubectl create -f custom-resources.yaml

```
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/apiservers.operator.tigera.io created
customresourcedefinition.apiextensions.k8s.io/imagesets.operator.tigera.io created
customresourcedefinition.apiextensions.k8s.io/installations.operator.tigera.io created
customresourcedefinition.apiextensions.k8s.io/tigerastatuses.operator.tigera.io created
serviceaccount/tigera-operator created
clusterrole.rbac.authorization.k8s.io/tigera-operator created
clusterrolebinding.rbac.authorization.k8s.io/tigera-operator created
deployment.apps/tigera-operator created
[ec2-user@ip-172-31-93-114 ~]$ curl https://raw.githubusercontent.com/projectcalico/calico/v3.27.0/manifests/custom-resources.yaml -O
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   810  100   810    0     0   8091      0 --:--:-- --:--:-- --:--:--  8181
[ec2-user@ip-172-31-93-114 ~]$ kubectl create -f custom-resources.yaml
installation.operator.tigera.io/default created
apiserver.operator.tigera.io/default created
[ec2-user@ip-172-31-93-114 ~]$
```

kubectl get nodes ,,,,,,,,,NOW READY

```
[ec2-user@ip-172-31-93-114 ~]$ kubectl get nodes
NAME                           STATUS   ROLES           AGE   VERSION
ip-172-31-81-175.ec2.internal  Ready    <none>          16m   v1.28.7
ip-172-31-91-218.ec2.internal  Ready    <none>          17m   v1.28.7
ip-172-31-93-114.ec2.internal  Ready    control-plane   23m   v1.28.7
[ec2-user@ip-172-31-93-114 ~]$
```

2. Create a NGINX deployment of 3 replicas

vi nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

kubectl apply -f nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
~
~
~
                                                                    20,0-1        All
```

kubectl get deployment

```
[ec2-user@ip-172-31-93-114 ~]$ vi nginx-deployment.yaml
[ec2-user@ip-172-31-93-114 ~]$ kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
[ec2-user@ip-172-31-93-114 ~]$ kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3/3     3            3           53s
[ec2-user@ip-172-31-93-114 ~]$
```