Kubernetes installation (Single node cluster)

Machine: 1 master & 2 worker

Master-node ip : 192.168.1.42
Worker-node1 ip : 192.168.1.43
Worker-node2 ip : 192.168.1.44
—----------
Host Entries: on each machine

192.168.1.42 master-node
192.168.1.43 worker-node1
192.168.1.44 worker-node2

—----------

Install podman in each machine
yum install podman* -y

—----------

Install Cri-o in each machine

curl -L -o /etc/yum.repos.d/devel:kubic:libcontainers:stable.repo
https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/CentOS_9_Stream/devel:kubic:libcontainers:stable.repo

curl -L -o /etc/yum.repos.d/devel:kubic:libcontainers:stable:cri-o:$VERSION.repo
https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable:/cri-o:/1.28/CentOS_9_Stream/devel:kubic:libcontainers:stable:cri-o:1.28.repo

yum install crio
systemctl status crio
systemctl start crio

—----------

Add repo of kubernetes in each machine

```
export KUBE_VERSION=1.28 && \
cat <<EOF | tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
```

```
baseurl=https://pkgs.k8s.io/core:/stable:/v${KUBE_VERSION}/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v${KUBE_VERSION}/rpm/repodata/repomd.xml.key
EOF
```

—----------

Install `kubeadm`, `kubectl`, and `kubelet` (In each machine)

```
yum install -y kubeadm kubectl kubelet
```

```
systemctl enable --now kubelet
```

—----------

Master node

#kubeadm init --apiserver-advertise-address=192.168.1.42/24 --pod-network-cidr=10.244.0.0/16

—----------OUTPUT—----------

[root@master-node ~]# kubeadm init --apiserver-advertise-address=192.168.1.42 --pod-network-cidr=10.244.0.0/16
I0309 17:57:37.459333 10186 version.go:256] remote version is much newer: v1.29.2; falling back to: stable-1.28
[init] Using Kubernetes version: v1.28.7
[preflight] Running pre-flight checks
error execution phase preflight: [preflight] Some fatal errors occurred:
[ERROR FileContent--proc-sys-net-ipv4-ip_forward]: /proc/sys/net/ipv4/ip_forward contents are not set to 1
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
[root@master-node ~]# Vi /etc/sysctl.conf
-bash: Vi: command not found
[root@master-node ~]# vi /etc/sysctl.conf
[root@master-node ~]# sysctl -p
net.ipv4.ip_forward = 1

```
[root@master-node ~]# kubeadm init --apiserver-advertise-address=192.168.1.42
--pod-network-cidr=10.244.0.0/16
I0309 17:58:38.821331   10250 version.go:256] remote version is much newer: v1.29.2;
falling back to: stable-1.28
[init] Using Kubernetes version: v1.28.7
[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'
[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-node] and IPs
[10.96.0.1 192.168.1.42]
[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-node] and
IPs [192.168.1.42 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [localhost master-node] and IPs
[192.168.1.42 127.0.0.1 ::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 7.503427 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-node 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-node as control-plane by adding the taints [node-role.kubernetes.io/control-plane:NoSchedule]
[bootstrap-token] Using token: 0dvfow.sebdmxm6qwi8d6a4
[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
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

```
  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
```

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

Alternatively, if you are the root user, you can run:

export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.1.42:6443 --token 0dvfow.sebdmxm6qwi8d6a4 \
--discovery-token-ca-cert-hash
sha256:4cebf59a7fe8fa00f4b89130b3107e7b0e9b0f7627429d4bd29f56953028e5c5

—----------

Run on Master node
—----------
# mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

Download
curl
https://raw.githubusercontent.com/projectcalico/calico/v3.26.3/manifests/canal.yaml
-O

kubectl apply -f canal.yaml

If cidr not define in master initialization time , then,

Vi canal.yaml

```
# - name: CALICO_IPV4POOL_CIDR
  #   value: "10.244.0.0/16"
```

Set network and save it.

Error:[root@worker-node1 ~]# kubeadm join 192.168.1.42:6443 --token 0dvfow.sebdmxm6qwi8d6a4 --discovery-token-ca-cert-hash sha256:4cebf59a7fe8fa00f4b89130b3107e7b0e9b0f7627429d4bd29f56953028e5c5
[preflight] Running pre-flight checks
[WARNING Service-Kubelet]: kubelet service is not enabled, please run 'systemctl enable kubelet.service'
error execution phase preflight: [preflight] Some fatal errors occurred:
[ERROR FileContent--proc-sys-net-ipv4-ip_forward]: /proc/sys/net/ipv4/ip_forward contents are not set to 1
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher

Solution: # vi /etc/sysctl.conf
Add line: net.ipv4.ip_forward = 1

And save it.
sysctl -p

—----------

Adding working node in custer (run on worker nodes)

kubeadm join 192.168.1.42:6443 --token 0dvfow.sebdmxm6qwi8d6a4 \
--discovery-token-ca-cert-hash sha256:4cebf59a7fe8fa00f4b89130b3107e7b0e9b0f7627429d4bd29f56953028e5c5

—----------

kubectl get nodes -o wide

```
# kubectl get pods -A
#kubectl get pods -o wide -A
```