HOSTAFRICA SOUTH AFRICA

New Technologies •

How to Install Kubernetes Cluster on CentOS 7

Published: 01/09/2020 · Updated: 02/09/2022

How to install Kubernetes and deploy a cluster with Docker on CentOS 7

Kubernetes (**k8s**) is an open-source, cloud-native, container orchestration and management platform. It's the go-to way to automate the deployment, scaling, and maintenance of containerised applications across different nodes. From service discovery to auto-restarts, and from resource allocation tracking to compute utilisation and scaling; a well-configured **k8s** cluster can manage a lot on its own.

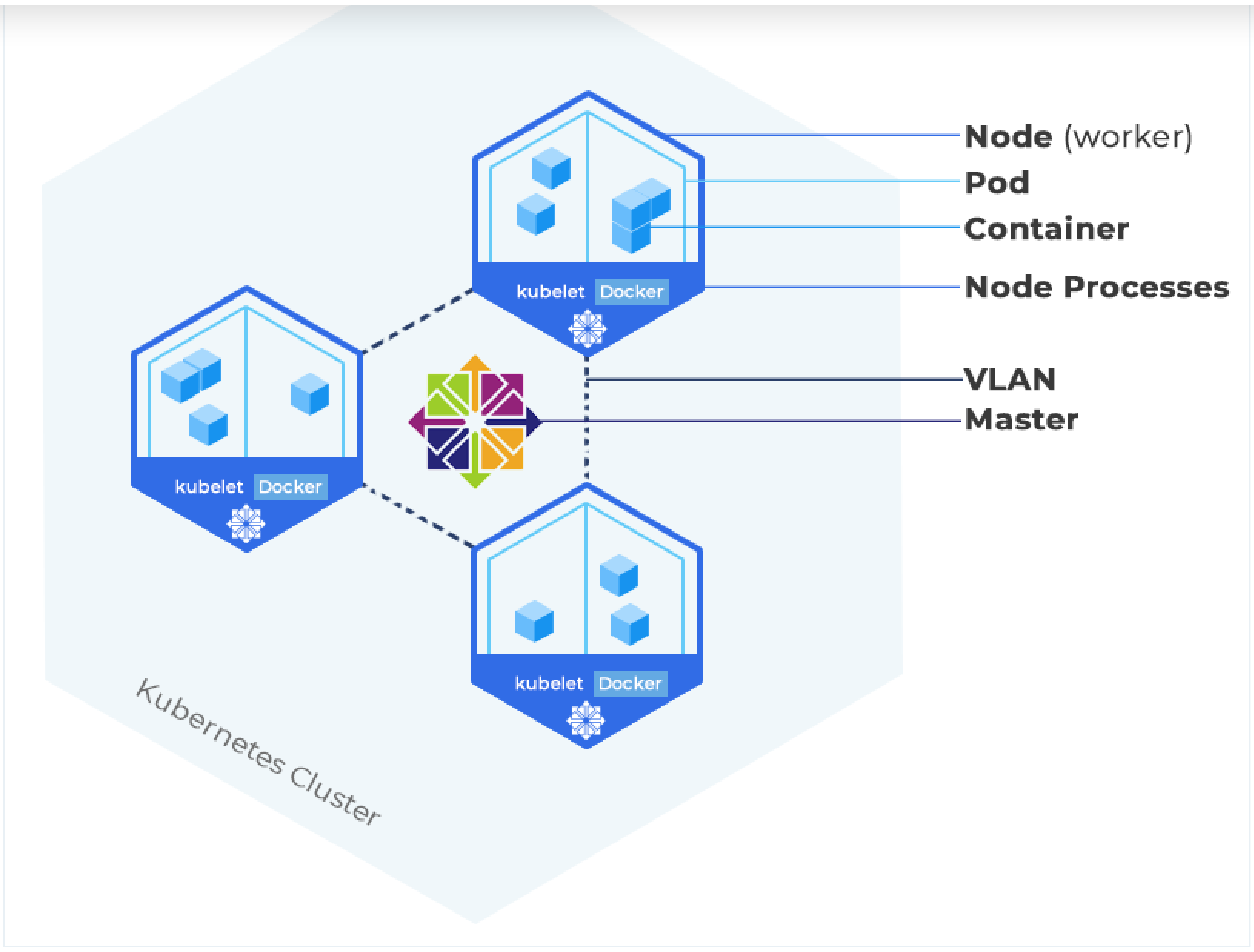
Contents

Got Ubuntu on your VMs? Learn How to install Kubernetes and deploy a cluster with Docker on Ubuntu 18.04

What is a Kubernetes Cluster?

A Kubernetes cluster consists of a **Master** and at least one to several **worker** node(s). The Master is the virtual machine (VM) that administers all activities on your cluster. A node is a VM that serves as a **worker** machine in your **k8s** cluster to host running applications. We strongly recommend you only use VMs aka Cloud Servers to run Kubernetes, not **system containers** aka VPS, as these can cause issues with k8s.

0.0 ★★★★★
No rating available



A node is comprised of the **Kubelet**, a container runtime, and the kube-proxy. The **k8s** installation's three core modules: Kubelet, **kubeadm**, and **kubectl** are agents that control the node and communicate with the Kubernetes Master. Once they have been installed and other configurations done, you will be able to create your first k8s cluster. You can manage this cluster from the command line on your kubemaster node.

Every Kubernetes instance runs on top of a **container runtime,** which is software responsible for managing container operations. Containers in this case are not virtualised servers but rather a solution that packages code and dependencies to run a single application (service) in an isolated (containerised) environment, essentially disassociating applications from the host machine. The most popular and recommended one is Docker, and it's the one we will use for the purpose of this guide. However, if you want to install a different underlying container runtime, you can harness the power of the Container Runtime Interface and use basically any runtime you want.

Kubernetes groups containers into pods, its most basic operational unit, which are basically just groups of containers running on the same node. Pods are connected over a network and share storage resources.

In order to connect your nodes or VMs and make them private, make sure to choose a hosting company who provides a Virtual Local Area Network (VLAN) with their VMs. We offer a VLAN add-on to our Cloud Servers for R200 per month.

Prerequisites
• Multiple CentOS 7 VMs (Cloud Servers) to house the **Master** and **worker** nodes.
• Docker or any other container runtime.
• User with `sudo` or `root` privileges on every server.

How to install Kubernetes on CentOS 7

Step 1. Install Docker on all CentOS 7 VMs

Update the package database

```
sudo yum check-update
```

Install the dependencies

0.0 ★★★★★
No rating available

```
all -y yum-utils device-mapper-persistent-data lvm2
```

Install the latest Docker version on CentOS 7

```
sudo yum install docker-ce
```

A successful installation output will be concluded with a `Complete!`

You may be prompted to accept the GPG key, this is to verify that the fingerprint matches. The format will look as follows. If correct, accept it.

```
060A 61C5 1B55 8A7F 742B 77AA C52F EB6B 621E 9F35
```

Step 4: Manage Docker Service

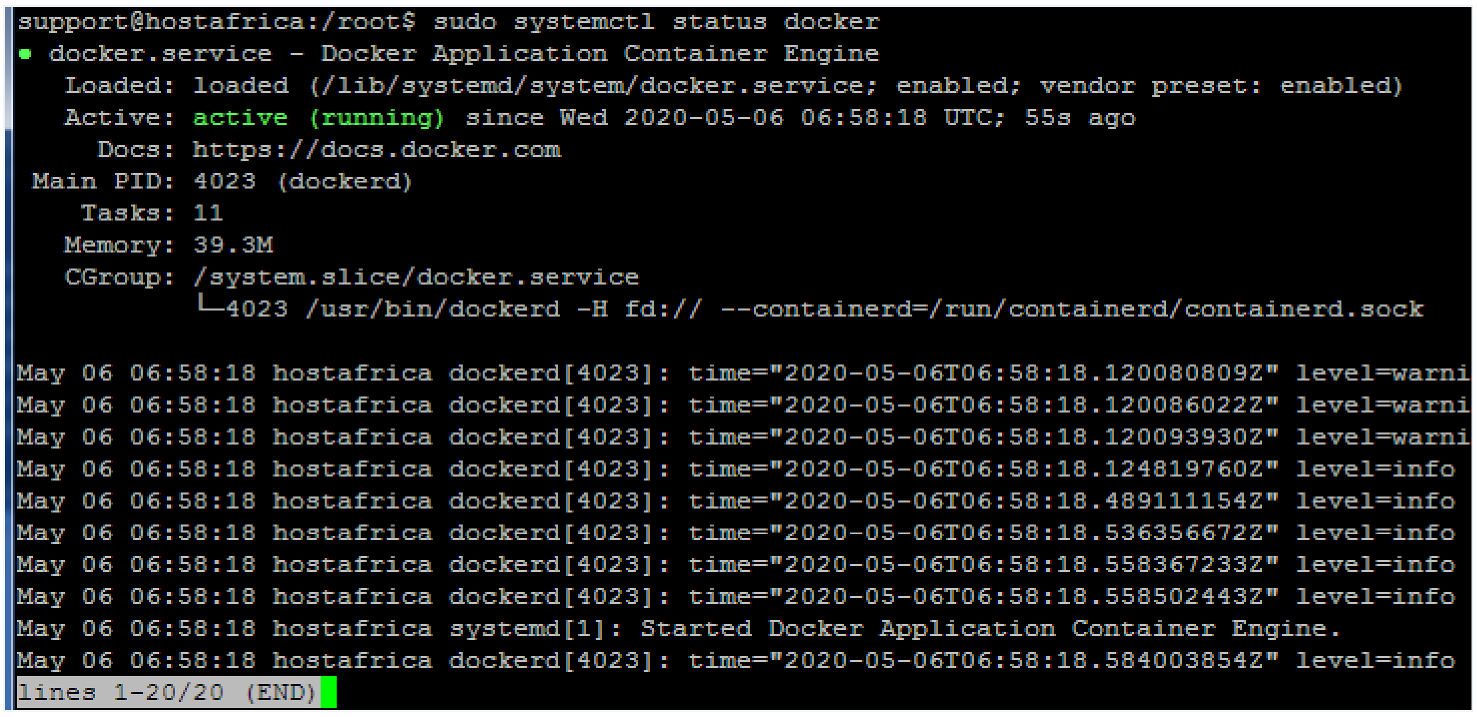
Now Docker is installed, but the service is not yet running. Start and enable Docker using the commands

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```

To confirm that Docker is active and running use

```
sudo systemctl status docker
```



```
support@hostafrica:/root$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2020-05-06 06:58:18 UTC; 55s ago
     Docs: https://docs.docker.com
 Main PID: 4023 (dockerd)
    Tasks: 11
   Memory: 39.3M
   CGroup: /system.slice/docker.service
           └─4023 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

May 06 06:58:18 hostafrica dockerd[4023]: time="2020-05-06T06:58:18.120808092" level=warni
May 06 06:58:18 hostafrica dockerd[4023]: time="2020-05-06T06:58:18.120086022Z" level=warni
May 06 06:58:18 hostafrica dockerd[4023]: time="2020-05-06T06:58:18.120093930Z" level=warni
May 06 06:58:18 hostafrica dockerd[4023]: time="2020-05-06T06:58:18.124819760Z" level=info
May 06 06:58:18 hostafrica dockerd[4023]: time="2020-05-06T06:58:18.489111154Z" level=info
May 06 06:58:18 hostafrica dockerd[4023]: time="2020-05-06T06:58:18.536356672Z" level=info
May 06 06:58:18 hostafrica dockerd[4023]: time="2020-05-06T06:58:18.558367233Z" level=info
May 06 06:58:18 hostafrica dockerd[4023]: time="2020-05-06T06:58:18.558502443Z" level=info
May 06 06:58:18 hostafrica systemd[1]: Started Docker Application Container Engine.
May 06 06:58:18 hostafrica dockerd[4023]: time="2020-05-06T06:58:18.584003854Z" level=info
lines 1-20/20 (END)
```

Step 2. Set up the Kubernetes Repository

Since the Kubernetes packages aren't present in the official CentOS 7 repositories, we will need to add a new repository file. Use the following command to create the file and open it for editing:

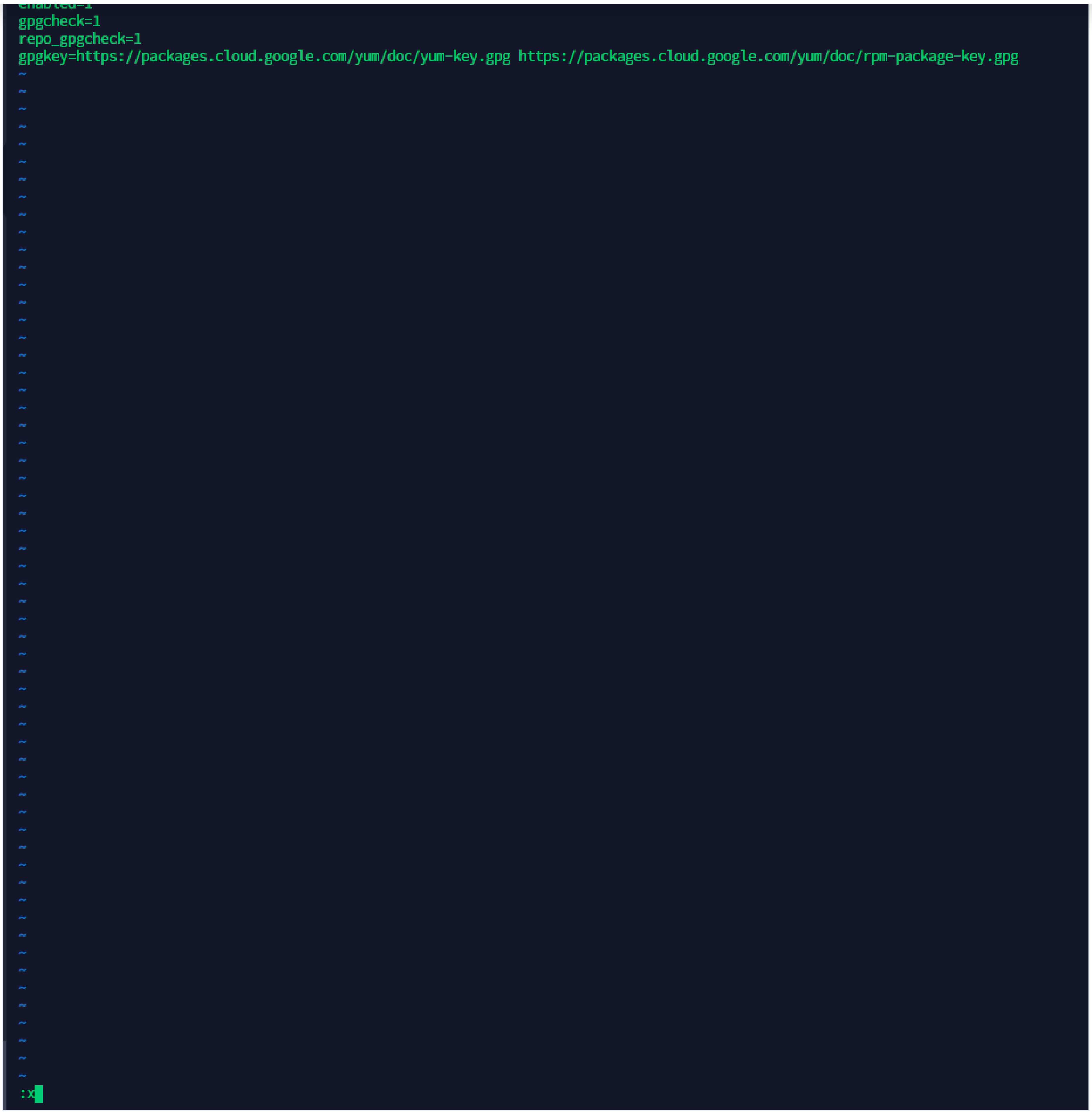
```
sudo vi /etc/yum.repos.d/kubernetes.repo
```

Once the file is open, press **I** key to enter **insert mode**, and paste the following contents:

```
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
```

Once pasted, press **escape** to exit **insert mode**. Then enter `:x` to save the file and exit.

0.0 ★★★★★
No rating available



```
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
~
:x
```

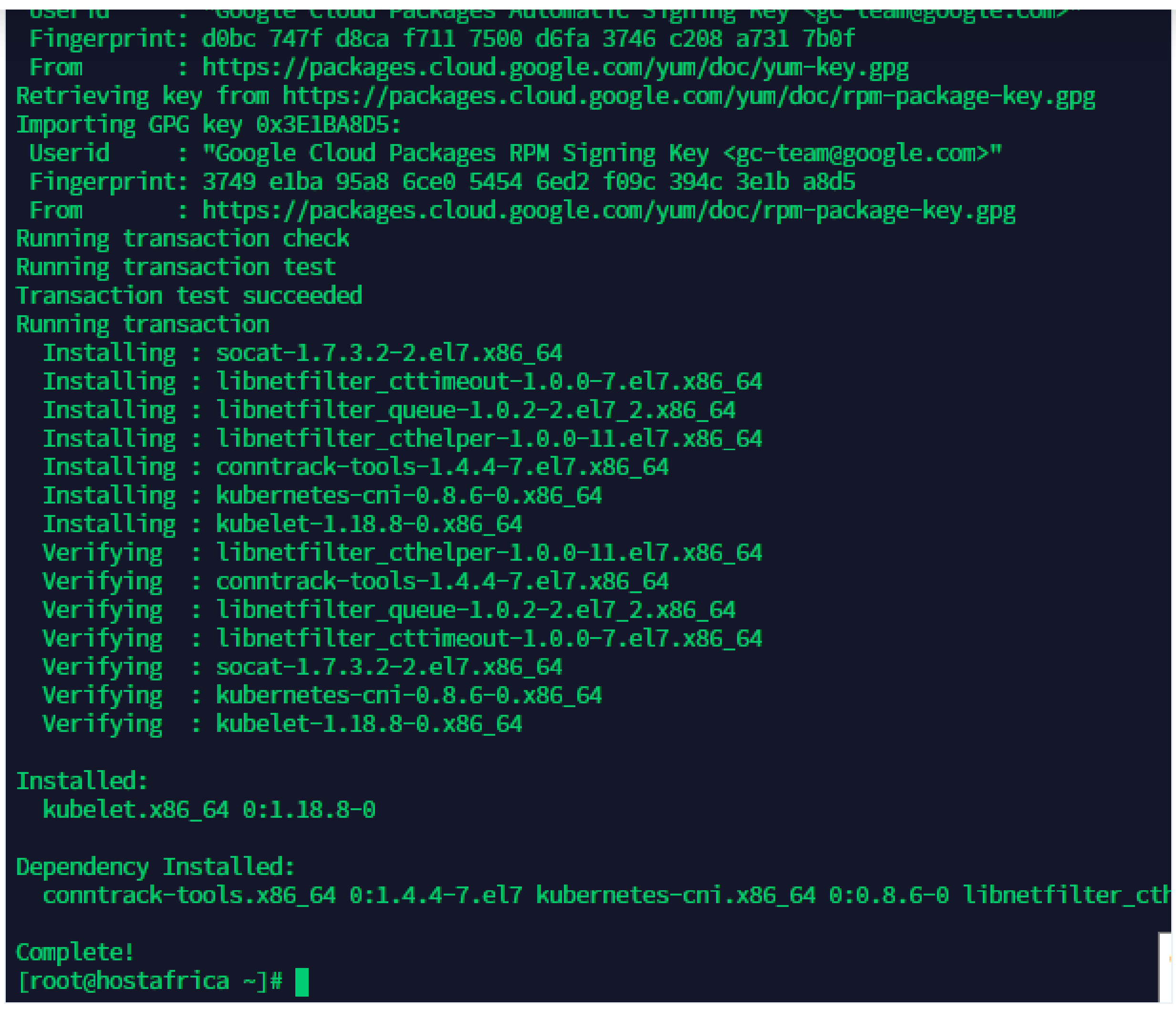
Step 3. Install Kubelet on CentOS 7

The first core module that we need to install on every node is Kubelet. Use the following command to do so:

```
sudo yum install -y kubelet
```

Once you enter the command, you should see a lot of logs being printed. A successful installation will be indicated by the `Complete!` keyword at the end. See below:

0.0 ★★★★★
No rating available



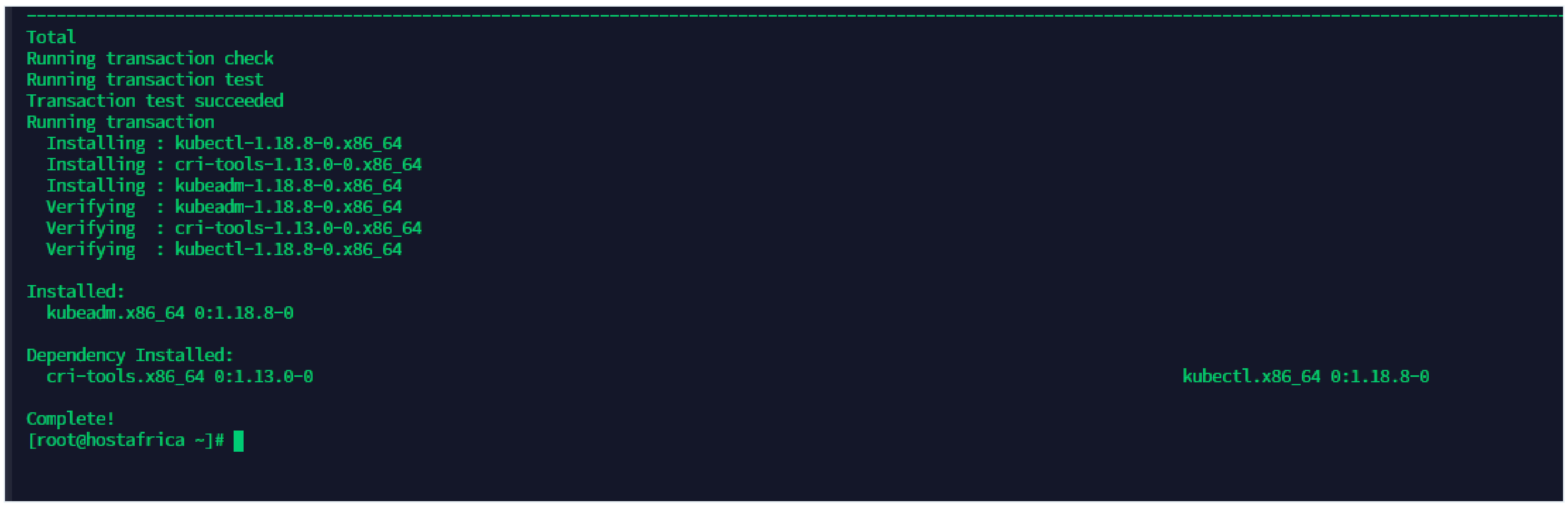
```
 Userid     : "Google Cloud Packages Automatic Signing Key <gc-team@google.com>"
 Fingerprint: d0bc 747f d8ca f711 7500 d6fa 3746 c208 a731 7b0f
 From       : https://packages.cloud.google.com/yum/doc/yum-key.gpg
Retrieving key from https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
Importing GPG key 0x3E1BA8D5:
 Userid     : "Google Cloud Packages RPM Signing Key <gc-team@google.com>"
 Fingerprint: 3749 e1ba 95a8 6ce0 5454 6ed2 f09c 394c 3e1b a8d5
 From       : https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : socat-1.7.3.2-2.el7.x86_64
  Installing : libnetfilter_cttimeout-1.0.0-7.el7.x86_64
  Installing : libnetfilter_queue-1.0.2-2.el7_2.x86_64
  Installing : libnetfilter_cthelper-1.0.0-11.el7.x86_64
  Installing : conntrack-tools-1.4.4-7.el7.x86_64
  Installing : kubernetes-cni-0.8.6-0.x86_64
  Installing : kubelet-1.18.8-0.x86_64
  Verifying  : libnetfilter_cthelper-1.0.0-11.el7.x86_64
  Verifying  : conntrack-tools-1.4.4-7.el7.x86_64
  Verifying  : libnetfilter_queue-1.0.2-2.el7_2.x86_64
  Verifying  : libnetfilter_cttimeout-1.0.0-7.el7.x86_64
  Verifying  : socat-1.7.3.2-2.el7.x86_64
  Verifying  : kubernetes-cni-0.8.6-0.x86_64
  Verifying  : kubelet-1.18.8-0.x86_64

Installed:
  kubelet.x86_64 0:1.18.8-0

Dependency Installed:
  conntrack-tools.x86_64 0:1.4.4-7.el7 kubernetes-cni.x86_64 0:0.8.6-0 libnetfilter_cth

Complete!
[root@hostafrica ~]#
```

Step 4. Install **kubeadm** and **kubectl** on CentOS 7

kubeadm, the next core module, will also have to be installed on every machine. Use the following command:

```
sudo yum install -y kubeadm
```

Successful installation should result in the following output:
(Note that **kubeadm** automatically installs **kubectl** as a dependency)



```
--------------------------------------------------------------------------------------------------------------------------------
Total
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : kubectl-1.18.8-0.x86_64
  Installing : cri-tools-1.13.0-0.x86_64
  Installing : kubeadm-1.18.8-0.x86_64
  Verifying  : kubeadm-1.18.8-0.x86_64
  Verifying  : cri-tools-1.13.0-0.x86_64
  Verifying  : kubectl-1.18.8-0.x86_64

Installed:
  kubeadm.x86_64 0:1.18.8-0

Dependency Installed:
  cri-tools.x86_64 0:1.13.0-0                                                                  kubectl.x86_64 0:1.18.8-0

Complete!
[root@hostafrica ~]#
```

Step 5. Set hostnames

On your **Master** node, update your hostname using the following command:

```
sudo hostnamectl set-hostname master-node
```

```
sudo exec bash
```

And

```
sudo hostnamectl set-hostname W-node1
```

```
sudo exec bash
```

Now open the */etc/hosts* file and edit the hostnames for your **worker** nodes:

0.0 ★★★★★
No rating available

```
10.168.10.208 node1 W-node1
10.168.10.209 node2 W-node2
EOF
```

Step 6. Disable SElinux

To allow containers to be able to access the file system, we need to enable the "**permissive**" mode of SElinux. Use the following commands:
(Note: For these commands to take effect, you will have to reboot)

```
sudo setenforce 0
```

```
sudo sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux
```

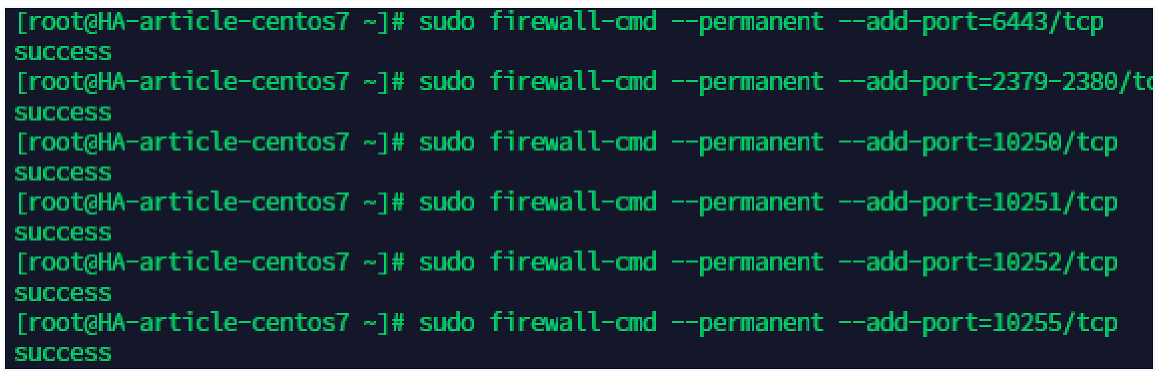
```
reboot
```

Step 7. Add firewall rules

To allow seamless communication between pods, containers, and VMs, we need to add rules to our firewall on the **Master** node. Use the following commands:

```
sudo firewall-cmd --permanent --add-port=6443/tcp
sudo firewall-cmd --permanent --add-port=2379-2380/tcp
sudo firewall-cmd --permanent --add-port=10250/tcp
sudo firewall-cmd --permanent --add-port=10251/tcp
sudo firewall-cmd --permanent --add-port=10252/tcp
sudo firewall-cmd --permanent --add-port=10255/tcp
sudo firewall-cmd -reload
```

All your firewall rule commands should output `success` like below:



```
[root@HA-article-centos7 ~]# sudo firewall-cmd --permanent --add-port=6443/tcp
success
[root@HA-article-centos7 ~]# sudo firewall-cmd --permanent --add-port=2379-2380/tc
success
[root@HA-article-centos7 ~]# sudo firewall-cmd --permanent --add-port=10250/tcp
success
[root@HA-article-centos7 ~]# sudo firewall-cmd --permanent --add-port=10251/tcp
success
[root@HA-article-centos7 ~]# sudo firewall-cmd --permanent --add-port=10252/tcp
success
[root@HA-article-centos7 ~]# sudo firewall-cmd --permanent --add-port=10255/tcp
success
```

You will also need to run the following commands on each **worker** node:

```
sudo firewall-cmd --permanent --add-port=10251/tcp
sudo firewall-cmd --permanent --add-port=10255/tcp
sudo firewall-cmd -reload
```

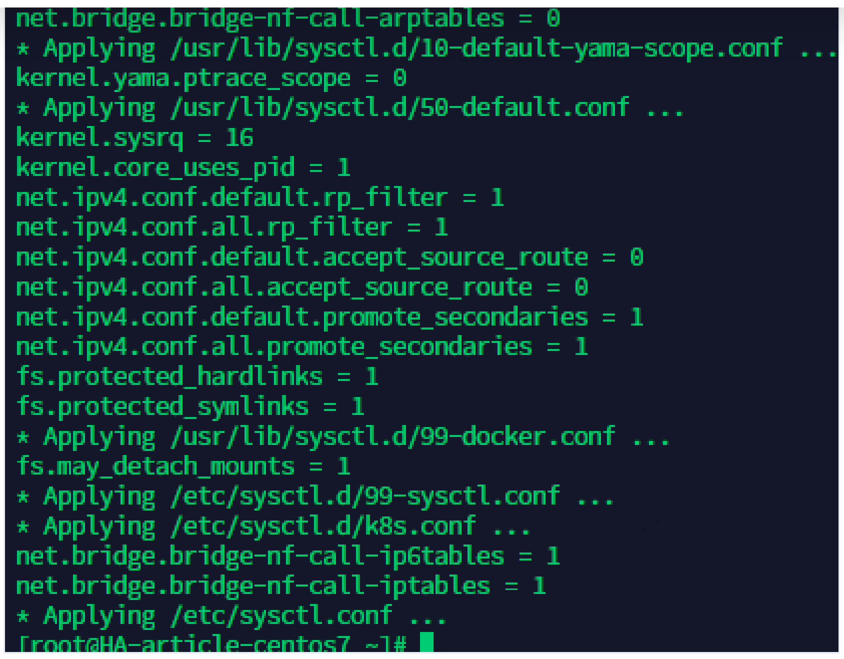
Step 8. Update iptables config

We need to update the `net.bridge.bridge-nf-call-iptables` parameter in our *sysctl* file to ensure proper processing of packets across all machines. Use the following commands:

```
cat <<EOF > /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sudo sysctl --system
```

You should get the following output:

0.0 ★★★★★
No rating available



```
net.bridge.bridge-nf-call-arptables = 0
* Applying /usr/lib/sysctl.d/10-default-yama-scope.conf ...
kernel.yama.ptrace_scope = 0
* Applying /usr/lib/sysctl.d/50-default.conf ...
kernel.sysrq = 16
kernel.core_uses_pid = 1
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.default.promote_secondaries = 1
net.ipv4.conf.all.promote_secondaries = 1
fs.protected_hardlinks = 1
fs.protected_symlinks = 1
* Applying /usr/lib/sysctl.d/99-docker.conf ...
fs.may_detach_mounts = 1
* Applying /etc/sysctl.d/99-sysctl.conf ...
* Applying /etc/sysctl.d/k8s.conf ...
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
* Applying /etc/sysctl.conf ...
[root@HA-article-centos7 ~]#
```

Step 9. Disable **swap**

For Kubelet to work, we also need to disable **swap** on all of our VMs:

```
sudo sed -i '/swap/d' /etc/fstab
```

```
sudo swapoff -a
```

This concludes our installation and configuration of Kubernetes on CentOS 7. We will now share the steps for deploying a k8s cluster.

Deploying a Kubernetes Cluster on CentOS 7

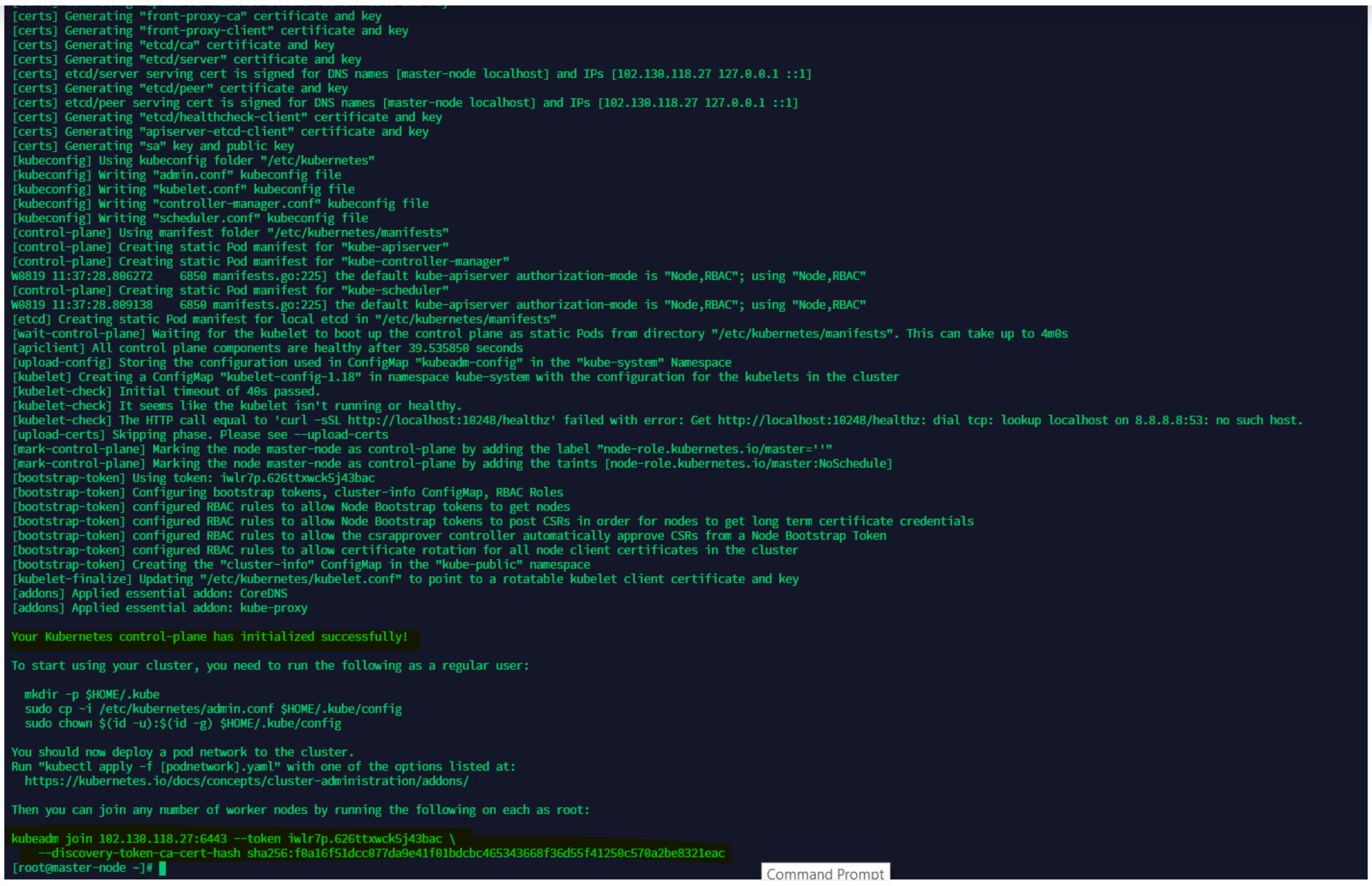
Step 1. **kubeadm** initialization

To launch a new Kubernetes cluster instance, you need to initialize **kubeadm**. Use the following command:

```
sudo kubeadm init
```

This command may take several minutes to execute. Upon success, you should get logs similar to those in this screenshot:

0.0 ★★★★★
No rating available



```
[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 [master-node localhost] and IPs [102.130.118.27 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [master-node localhost] and IPs [102.130.118.27 127.0.0.1 ::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
[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"
W0819 11:37:28.806272    6850 manifests.go:225] the default kube-apiserver authorization-mode is "Node,RBAC"; using "Node,RBAC"
[control-plane] Creating static Pod manifest for "kube-scheduler"
W0819 11:37:28.809138    6850 manifests.go:225] the default kube-apiserver authorization-mode is "Node,RBAC"; using "Node,RBAC"
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[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 39.535850 seconds
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config-1.18" in namespace kube-system with the configuration for the kubelets in the cluster
[kubelet-check] Initial timeout of 40s passed.
[kubelet-check] It seems like the kubelet isn't running or healthy.
[kubelet-check] The HTTP call equal to 'curl -sSL http://localhost:10248/healthz' failed with error: Get http://localhost:10248/healthz: dial tcp: lookup localhost on 8.8.8.8:53: no such host.
[upload-certs] Skipping phase. Please see --upload-certs
[mark-control-plane] Marking the node master-node as control-plane by adding the label "node-role.kubernetes.io/master=''"
[mark-control-plane] Marking the node master-node as control-plane by adding the taints [node-role.kubernetes.io/master:NoSchedule]
[bootstrap-token] Using token: iwlr7p.626ttxwck5j43bac
[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

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 102.130.118.27:6443 --token iwlr7p.626ttxwck5j43bac \
    --discovery-token-ca-cert-hash sha256:f0a16f51dcc077da9e41f01bdcbc465343668f36d55f41250c570a2be8321eac
[root@master-node ~]#
```

You will also get an auto-generated command at the end of the output. Copy the text following the line `Then you can join any number of worker nodes by running the following on each as root:` as highlighted in the above screenshot and save it somewhere safe. We will use this to add worker nodes to our cluster.

Note: If you forgot to copy the command, or have misplaced it, don't worry. You can retrieve it again by entering the following command:

```
sudo kubeadm token create --print-join-command
```

Step 2. Create required directories and start managing Kubernetes cluster

In order to start managing your cluster, you need to create a directory and assume ownership. Run the following commands as a regular user:

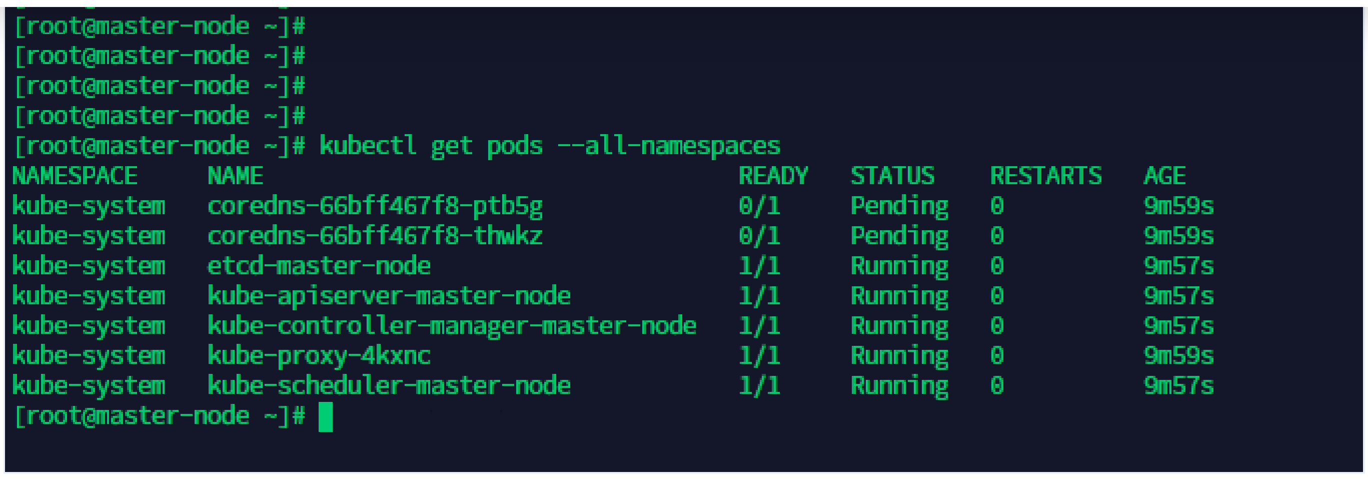
```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Step 3. Set up Pod network for the Cluster

Pods within a cluster are connected via the pod network. At this point, it's not working. This can be verified by entering the following two commands:

```
sudo kubectl get nodes
```

```
sudo kubectl get pods --all-namespaces
```



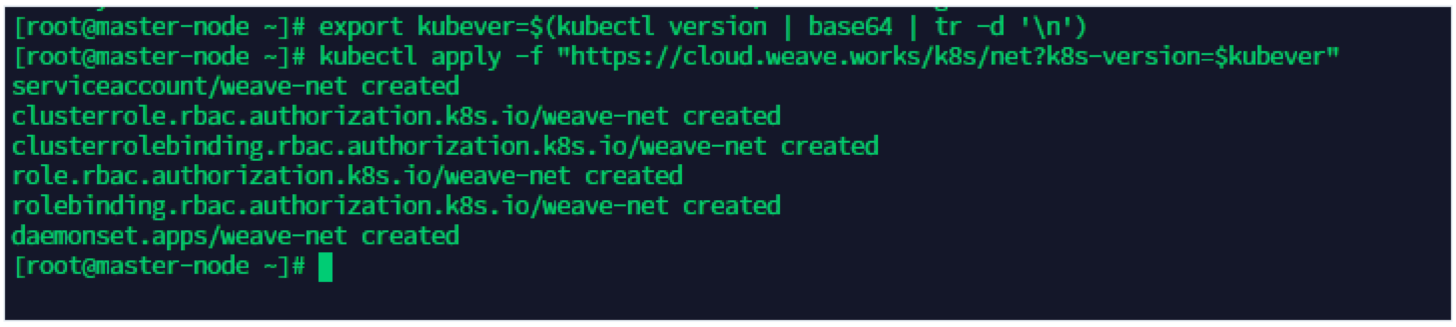
```
[root@master-node ~]#
[root@master-node ~]#
[root@master-node ~]#
[root@master-node ~]#
[root@master-node ~]# kubectl get pods --all-namespaces
NAMESPACE     NAME                                  READY   STATUS    RESTARTS   AGE
kube-system   coredns-66bff467f8-ptb5g              0/1     Pending   0          9m59s
kube-system   coredns-66bff467f8-thwkz              0/1     Pending   0          9m59s
kube-system   etcd-master-node                      1/1     Running   0          9m57s
kube-system   kube-apiserver-master-node            1/1     Running   0          9m57s
kube-system   kube-controller-manager-master-node   1/1     Running   0          9m57s
kube-system   kube-proxy-4kxnc                      1/1     Running   0          9m59s
kube-system   kube-scheduler-master-node            1/1     Running   0          9m57s
[root@master-node ~]#
```

As you can see, the status of master–node is `NotReady`. The CoreDNS service is also not running. To fix this, run the following commands:

```
sudo export kubever=$(kubectl version | base64 | tr -d '\n')
```

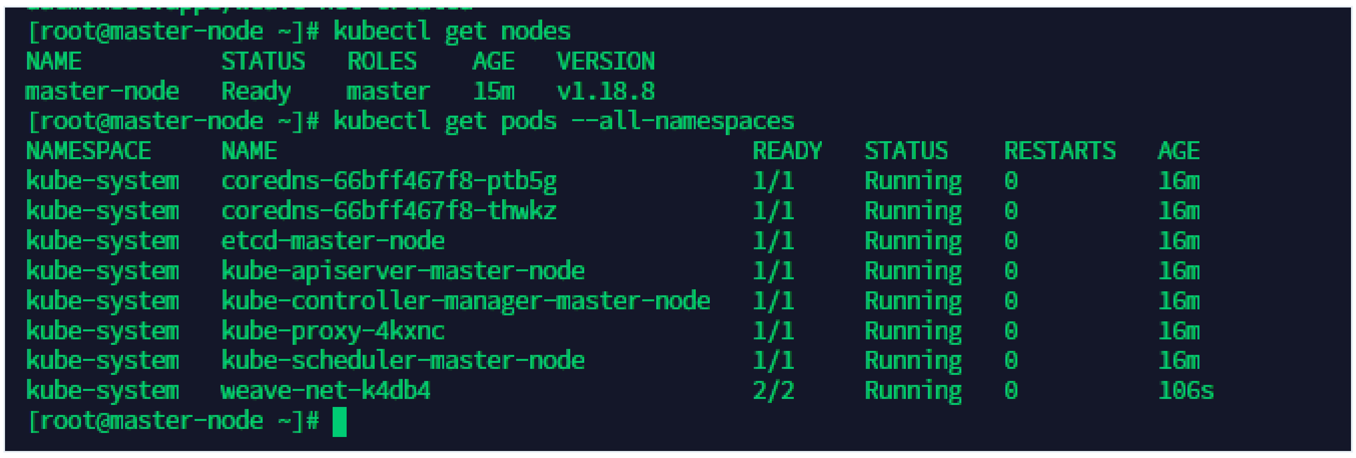
```
sudo kubectl apply -f https://cloud.weave.works/k8s/net?k8s-version=$kubever
```

You should get the following output:



```
[root@master-node ~]# export kubever=$(kubectl version | base64 | tr -d '\n')
[root@master-node ~]# kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$kubever"
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.apps/weave-net created
[root@master-node ~]#
```

And now if you verify the statuses of your node and CoreDNS service, you should get `Ready` and `Running` like seen below:



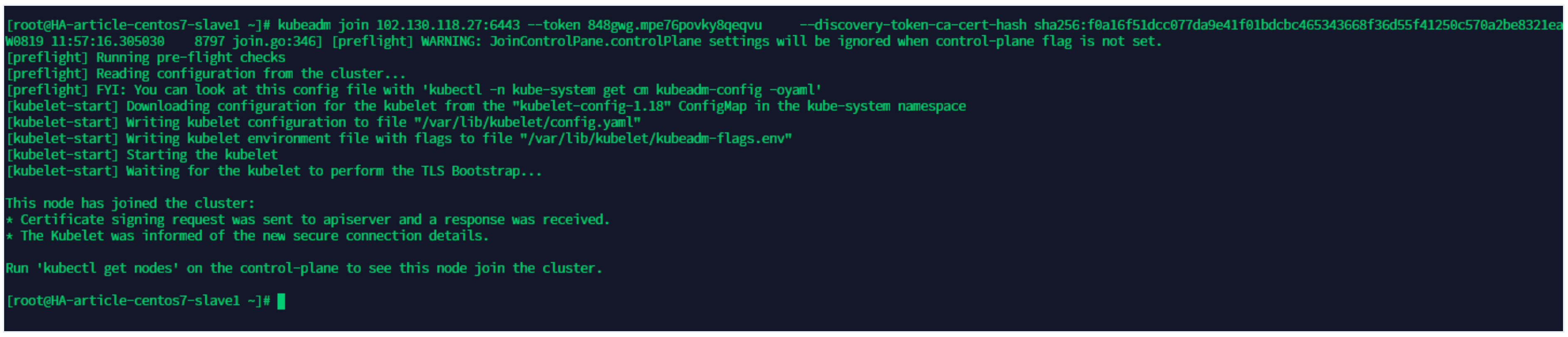
```
[root@master-node ~]# kubectl get nodes
NAME          STATUS   ROLES    AGE   VERSION
master-node   Ready    master   15m   v1.18.8
[root@master-node ~]# kubectl get pods --all-namespaces
NAMESPACE     NAME                                  READY   STATUS    RESTARTS   AGE
kube-system   coredns-66bff467f8-ptb5g              1/1     Running   0          16m
kube-system   coredns-66bff467f8-thwkz              1/1     Running   0          16m
kube-system   etcd-master-node                      1/1     Running   0          16m
kube-system   kube-apiserver-master-node            1/1     Running   0          16m
kube-system   kube-controller-manager-master-node   1/1     Running   0          16m
kube-system   kube-proxy-4kxnc                      1/1     Running   0          16m
kube-system   kube-scheduler-master-node            1/1     Running   0          16m
kube-system   weave-net-k4db4                       2/2     Running   0          106s
[root@master-node ~]#
```

Step 4. Add nodes to your cluster

As a final step, you need to add **worker** nodes to your cluster. We will use the **kubeadm** join auto-generated token in [Step 1.] here. Run your own version of the following command on all of the worker node VMs:

```
sudo kubeadm join 102.130.118.27:6443 --token 848gwg.mpe76povky8qeqvu --discovery-token-ca-cert-hash sha256:f0a16f51dcc077d
```

On successful addition, you should get the following output:



```
[root@HA-article-centos7-slave1 ~]# kubeadm join 102.130.118.27:6443 --token 848gwg.mpe76povky8qeqvu     --discovery-token-ca-cert-hash sha256:f0a16f51dcc077da9e41f01bdcbc465343668f36d55f41250c570a2be8321ea
W0819 11:57:16.305030    8797 join.go:346] [preflight] WARNING: JoinControlPane.controlPlane settings will be ignored when control-plane flag is not set.
[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 -oyaml'
[kubelet-start] Downloading configuration for the kubelet from the "kubelet-config-1.18" ConfigMap in the kube-system namespace
[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.

[root@HA-article-centos7-slave1 ~]#
```

Running the following command on the `master-node` should show your newly added node.

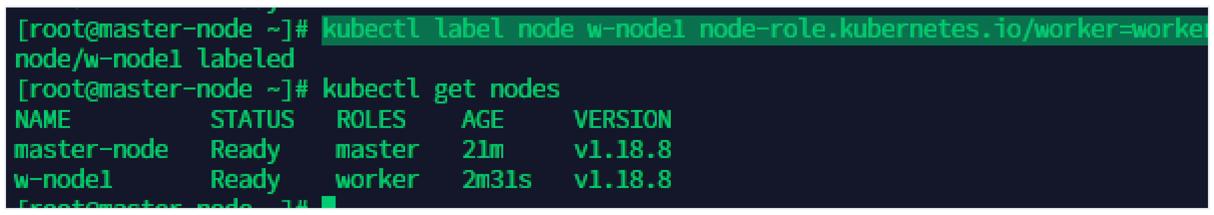
```
get nodes
```

0.0 ★★★★★
No rating available

```
w-node1        Ready    <none>   84s   v1.18.8
```

To set the role for your **worker** node, use the following command:

```
sudo kubectl label node w-node1 node-role.kubernetes.io/worker=worker
```



```
[root@master-node ~]# kubectl label node w-node1 node-role.kubernetes.io/worker=worker
node/w-node1 labeled
[root@master-node ~]# kubectl get nodes
NAME          STATUS   ROLES    AGE     VERSION
master-node   Ready    master   21m     v1.18.8
w-node1       Ready    worker   2m31s   v1.18.8
```

Now you're all set up.

Happy Hosting!

Got Ubuntu on your VMs? Learn [How to install Kubernetes and deploy a cluster with Docker on Ubuntu 18.04]

> [Back to top of page]

Related posts

Tags

/ TUTORIALS

SHARE THIS ARTICLE:

0.0 ★★★★★
No rating available

HOSTAFRICA SOUTH AFRICA

| Automation | Backup | Business | Cloud | cPanel | Cybersecurity | DNS | Domain Name | Domains | E-mail | Hosting | Kubernetes | Linux | Marketing | Networks | New Technologies | News | Our Portfolio | PHP | Reseller | Security | Servers | Site Builder | Software | SSL certificates | VPS | Web Design | Web Development | Web Tutorials | Websites | Windows | WordPress

HOSTAFRICA

Contact Details

12 Helena Ave,
Helena Heights
Somerset West, 7130
South Africa

Phone: +27 21 554 3096
Network Abuse
Submit a Ticket
Sales Enquiry

We Accept: EFT, Debit Cards, Credit Cards and Mobile Payments

PayPal SnapScan PayFast

0.0 ★★★★★
No rating available