SETUP KUBERNETES – RED HAT [[ALMALINUX]]…

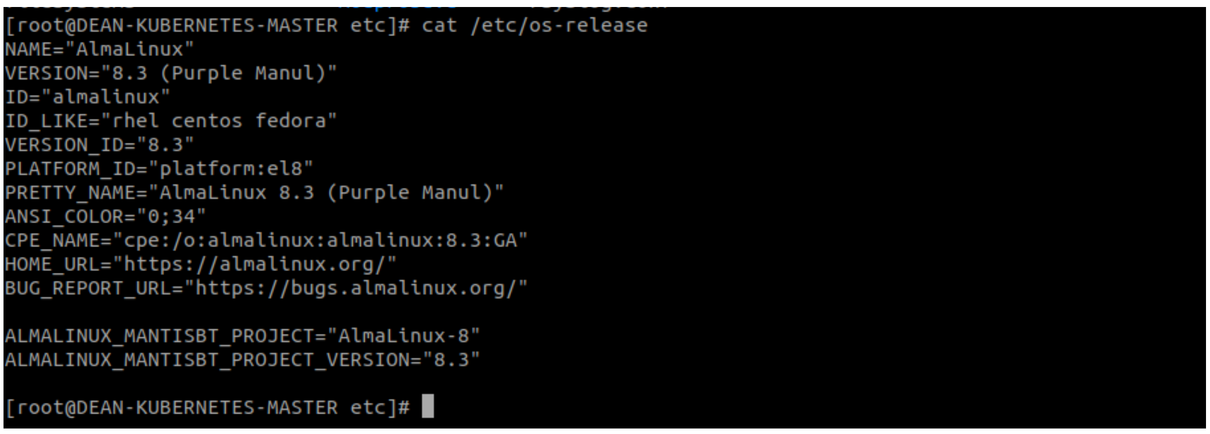
Requirements

1. Download Alma Linux 9 -> http://alma.mirror.ac.za/9.0/isos/x86_64/
2. Download WHMCS Ver 8.4.1 February 21st 2022
3. 1-GRID Server Setup - 2 x Raid Master and Multiple - Worker Raid Nodes.. For redundancy in a Kubernetes setup in this example I am using the following:
   - example used: **master or main** -> 41.185.61.18
   - example used: **worker one** -> 41.185.61.28
   - example used: **worker two** -> 41.185.61.142
4. User with Sudo privileges on every server

How to install Kubernetes on Alma Linux 9

Step 1. After Install check version of OS.

```
$ cat /etc/os-release
```



```
[root@DEAN-KUBERNETES-MASTER etc]# cat /etc/os-release
NAME="AlmaLinux"
VERSION="8.3 (Purple Manul)"
ID="almalinux"
ID_LIKE="rhel centos fedora"
VERSION_ID="8.3"
PLATFORM_ID="platform:el8"
PRETTY_NAME="AlmaLinux 8.3 (Purple Manul)"
ANSI_COLOR="0;34"
CPE_NAME="cpe:/o:almalinux:almalinux:8.3:GA"
HOME_URL="https://almalinux.org/"
BUG_REPORT_URL="https://bugs.almalinux.org/"

ALMALINUX_MANTISBT_PROJECT="AlmaLinux-8"
ALMALINUX_MANTISBT_PROJECT_VERSION="8.3"

[root@DEAN-KUBERNETES-MASTER etc]#
```

Step 2. Install Docker on all Alma Linux 9 systems

Update the package database

```
$ sudo yum check-update
```

Install the dependencies

```
$ sudo yum install -y yum-utils device-mapper-persistent-data lvm2
```

At the time of this writing, the closest thing we have to a Docker repository for AlmaLinux is the one made for CentOS. We can add the Docker repository to our system with the following command.

```
$ sudo dnf config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

Before we begin installing Docker, we need to remove the **podman** and **buildah** packages from our system, as they conflict with Docker and will inhibit it from being installed.

```
$ sudo dnf remove podman buildah
```

Finally, we can install the three Docker packages we'll need by executing the following command.

```
$ sudo dnf install docker-ce docker-ce-cli containerd.io
```

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
```

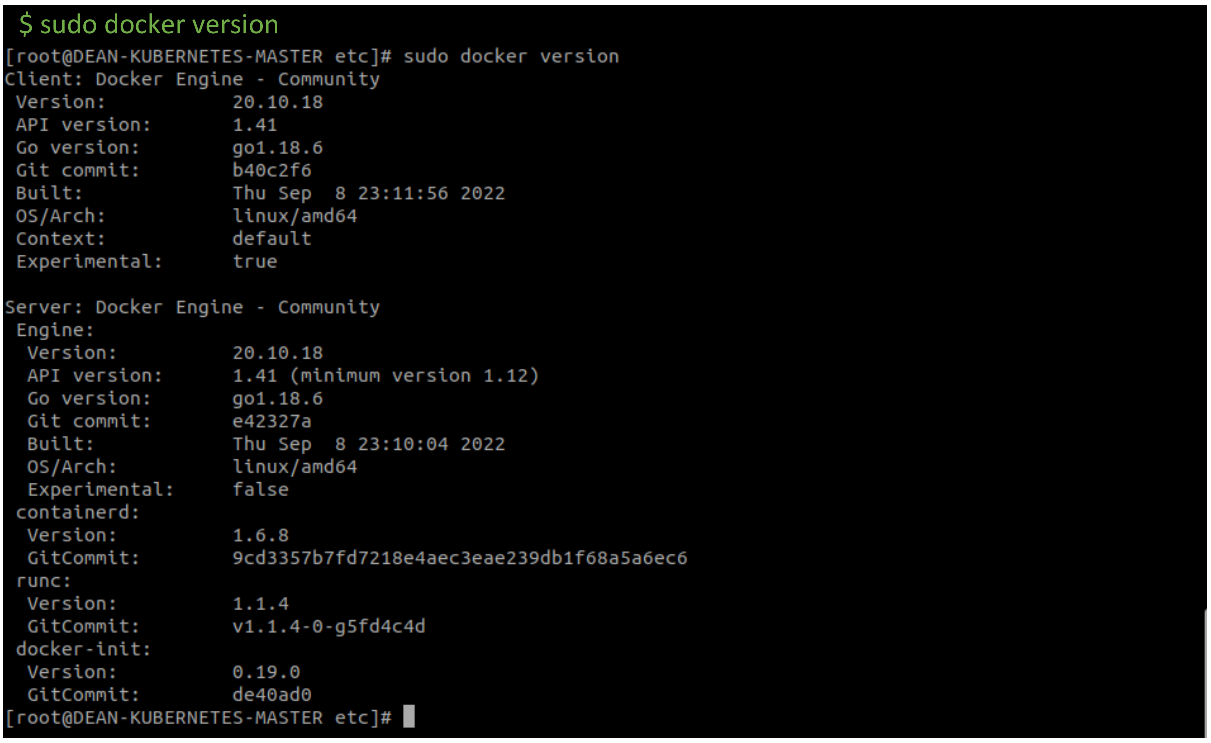
Step 3: Manage Docker Service

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
```

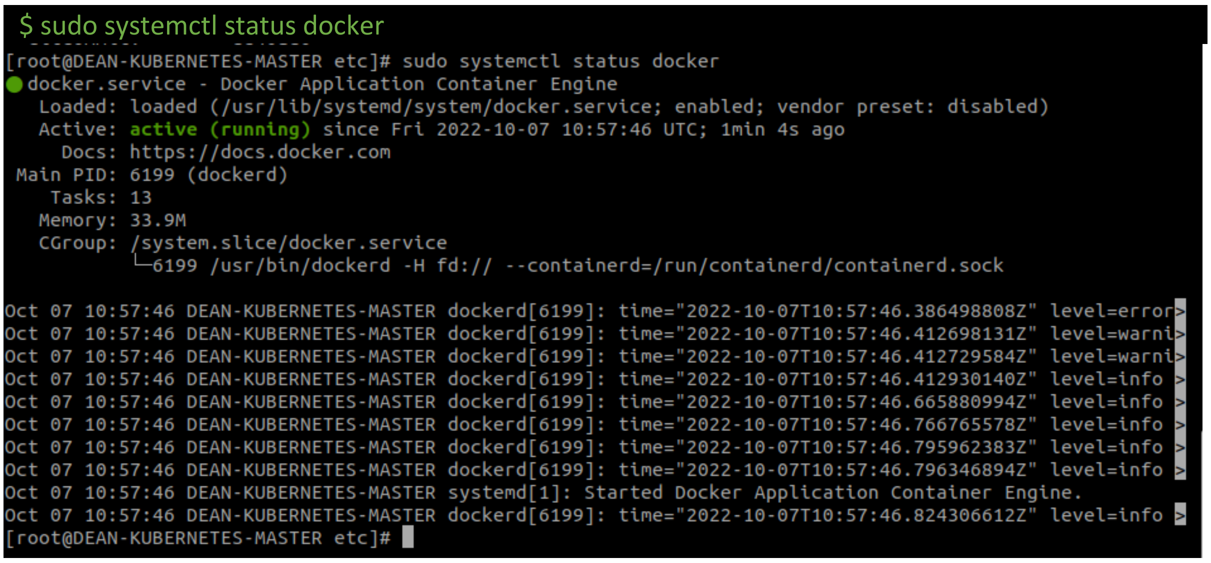
2nd Floor
360 Business Park
8 Wessex Street
Paarden Eiland
Cape Town
7405
South Africa

You can verify that Docker is installed and gather some information about the current version by entering this command and checking if it is running:



$ sudo docker version

```
[root@DEAN-KUBERNETES-MASTER etc]# sudo docker version
Client: Docker Engine - Community
 Version:           20.10.18
 API version:       1.41
 Go version:        go1.18.6
 Git commit:        b40c2f6
 Built:             Thu Sep  8 23:11:56 2022
 OS/Arch:           linux/amd64
 Context:           default
 Experimental:      true

Server: Docker Engine - Community
 Engine:
  Version:          20.10.18
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.18.6
  Git commit:       e42327a
  Built:            Thu Sep  8 23:10:04 2022
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.6.8
  GitCommit:        9cd3357b7fd7218e4aec3eae239db1f68a5a6ec6
 runc:
  Version:          1.1.4
  GitCommit:        v1.1.4-0-g5fd4c4d
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
[root@DEAN-KUBERNETES-MASTER etc]#
```



$ sudo systemctl status docker

```
[root@DEAN-KUBERNETES-MASTER etc]# sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-10-07 10:57:46 UTC; 1min 4s ago
     Docs: https://docs.docker.com
 Main PID: 6199 (dockerd)
    Tasks: 13
   Memory: 33.9M
   CGroup: /system.slice/docker.service
           └─6199 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Oct 07 10:57:46 DEAN-KUBERNETES-MASTER dockerd[6199]: time="2022-10-07T10:57:46.386498808Z" level=error>
Oct 07 10:57:46 DEAN-KUBERNETES-MASTER dockerd[6199]: time="2022-10-07T10:57:46.412698131Z" level=warni>
Oct 07 10:57:46 DEAN-KUBERNETES-MASTER dockerd[6199]: time="2022-10-07T10:57:46.412729584Z" level=warni>
Oct 07 10:57:46 DEAN-KUBERNETES-MASTER dockerd[6199]: time="2022-10-07T10:57:46.412930140Z" level=info >
Oct 07 10:57:46 DEAN-KUBERNETES-MASTER dockerd[6199]: time="2022-10-07T10:57:46.665880994Z" level=info >
Oct 07 10:57:46 DEAN-KUBERNETES-MASTER dockerd[6199]: time="2022-10-07T10:57:46.766765578Z" level=info >
Oct 07 10:57:46 DEAN-KUBERNETES-MASTER dockerd[6199]: time="2022-10-07T10:57:46.795962383Z" level=info >
Oct 07 10:57:46 DEAN-KUBERNETES-MASTER dockerd[6199]: time="2022-10-07T10:57:46.796346894Z" level=info >
Oct 07 10:57:46 DEAN-KUBERNETES-MASTER systemd[1]: Started Docker Application Container Engine.
Oct 07 10:57:46 DEAN-KUBERNETES-MASTER dockerd[6199]: time="2022-10-07T10:57:46.824306612Z" level=info >
[root@DEAN-KUBERNETES-MASTER etc]#
```

Step 4. Set up the Kubernetes Repository

Since the Kubernetes packages aren't present in the official Alma Linux 9 or CentOS 7/8 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
```

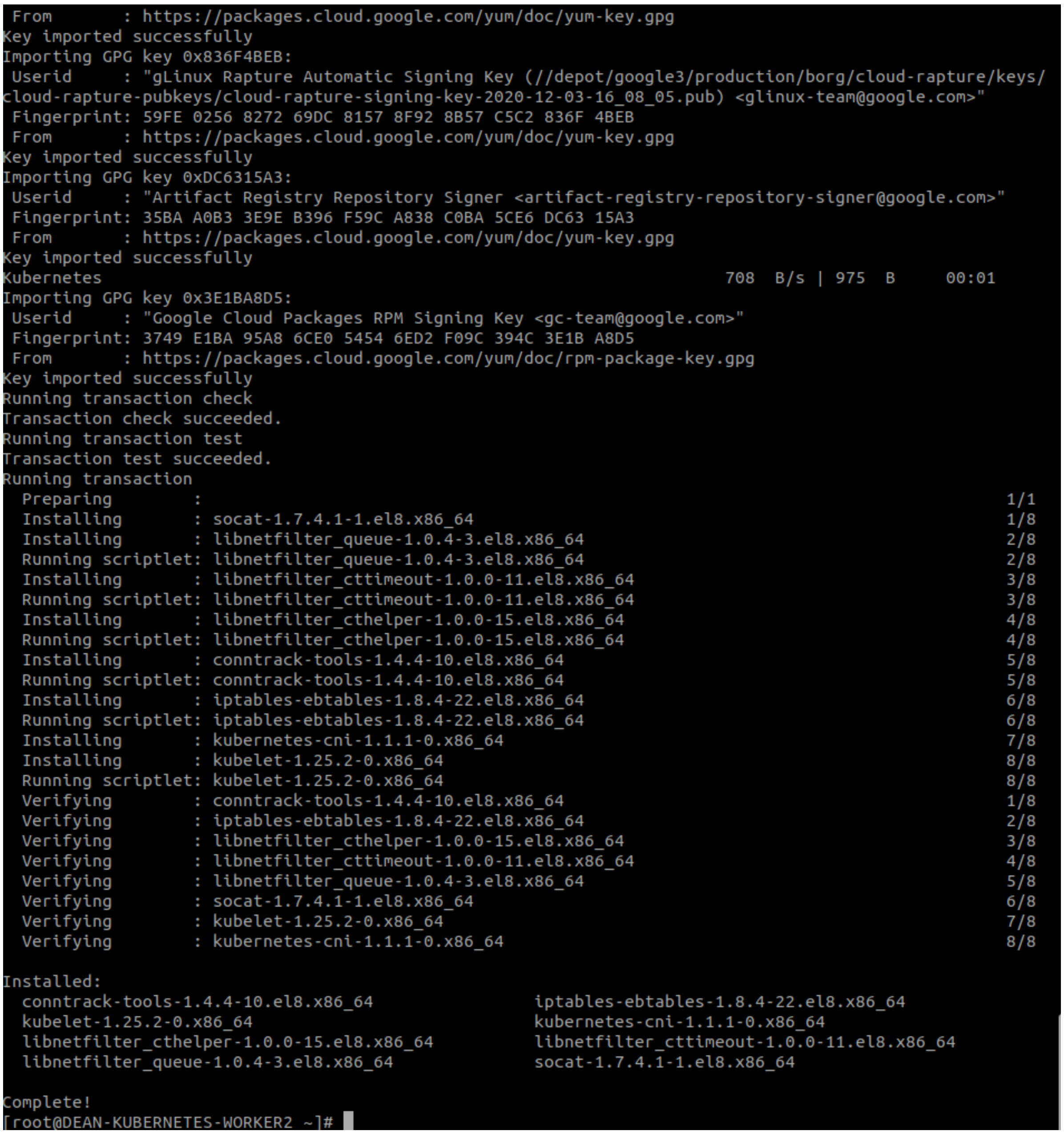
Once pasted, press escape to exit insert mode. Then enter :x to save the file and exit.

Step 5. Install Kubelet Alma Linux [[Now the fun part]]

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:



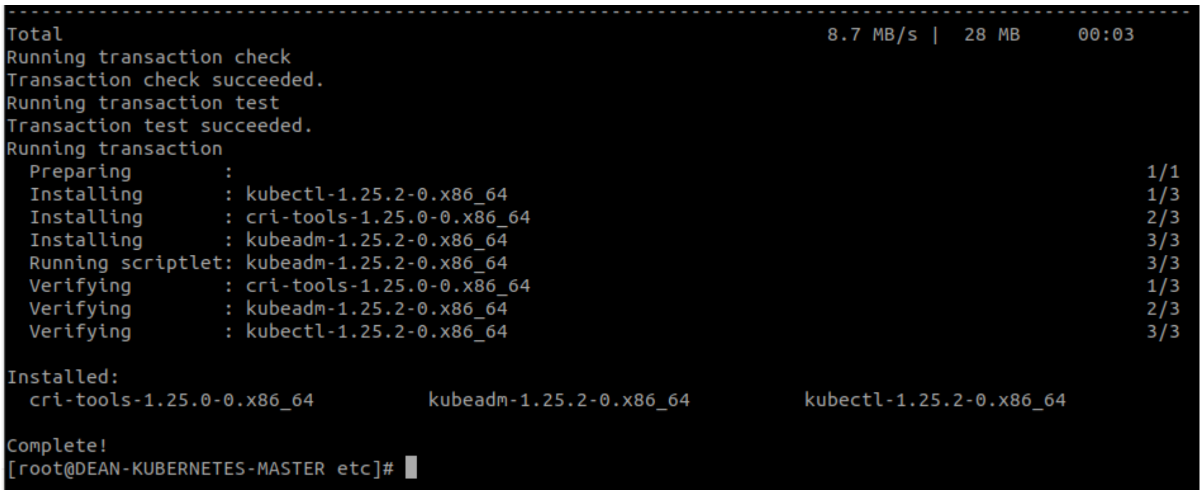
```
 From       : https://packages.cloud.google.com/yum/doc/yum-key.gpg
Key imported successfully
Importing GPG key 0x836F4BEB:
 Userid     : "gLinux Rapture Automatic Signing Key (//depot/google3/production/borg/cloud-rapture/keys/
cloud-rapture-pubkeys/cloud-rapture-signing-key-2020-12-03-16_08_05.pub) <glinux-team@google.com>"
 Fingerprint: 59FE 0256 8272 69DC 8157 8F92 8B57 C5C2 836F 4BEB
 From       : https://packages.cloud.google.com/yum/doc/yum-key.gpg
Key imported successfully
Importing GPG key 0xDC6315A3:
 Userid     : "Artifact Registry Repository Signer <artifact-registry-repository-signer@google.com>"
 Fingerprint: 35BA A0B3 3E9E B396 F59C A838 C0BA 5CE6 DC63 15A3
 From       : https://packages.cloud.google.com/yum/doc/yum-key.gpg
Key imported successfully
Kubernetes                                                            708  B/s | 975  B     00:01
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
Key imported successfully
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                          1/1
  Installing       : socat-1.7.4.1-1.el8.x86_64                                               1/8
  Installing       : libnetfilter_queue-1.0.4-3.el8.x86_64                                    2/8
  Running scriptlet: libnetfilter_queue-1.0.4-3.el8.x86_64                                    2/8
  Installing       : libnetfilter_cttimeout-1.0.0-11.el8.x86_64                               3/8
  Running scriptlet: libnetfilter_cttimeout-1.0.0-11.el8.x86_64                               3/8
  Installing       : libnetfilter_cthelper-1.0.0-15.el8.x86_64                                4/8
  Running scriptlet: libnetfilter_cthelper-1.0.0-15.el8.x86_64                                4/8
  Installing       : conntrack-tools-1.4.4-10.el8.x86_64                                      5/8
  Running scriptlet: conntrack-tools-1.4.4-10.el8.x86_64                                      5/8
  Installing       : iptables-ebtables-1.8.4-22.el8.x86_64                                    6/8
  Running scriptlet: iptables-ebtables-1.8.4-22.el8.x86_64                                    6/8
  Installing       : kubernetes-cni-1.1.1-0.x86_64                                            7/8
  Installing       : kubelet-1.25.2-0.x86_64                                                  8/8
  Running scriptlet: kubelet-1.25.2-0.x86_64                                                  8/8
  Verifying        : conntrack-tools-1.4.4-10.el8.x86_64                                      1/8
  Verifying        : iptables-ebtables-1.8.4-22.el8.x86_64                                    2/8
  Verifying        : libnetfilter_cthelper-1.0.0-15.el8.x86_64                                3/8
  Verifying        : libnetfilter_cttimeout-1.0.0-11.el8.x86_64                               4/8
  Verifying        : libnetfilter_queue-1.0.4-3.el8.x86_64                                    5/8
  Verifying        : socat-1.7.4.1-1.el8.x86_64                                               6/8
  Verifying        : kubelet-1.25.2-0.x86_64                                                  7/8
  Verifying        : kubernetes-cni-1.1.1-0.x86_64                                            8/8

Installed:
  conntrack-tools-1.4.4-10.el8.x86_64               iptables-ebtables-1.8.4-22.el8.x86_64
  kubelet-1.25.2-0.x86_64                           kubernetes-cni-1.1.1-0.x86_64
  libnetfilter_cthelper-1.0.0-15.el8.x86_64         libnetfilter_cttimeout-1.0.0-11.el8.x86_64
  libnetfilter_queue-1.0.4-3.el8.x86_64             socat-1.7.4.1-1.el8.x86_64

Complete!
[root@DEAN-KUBERNETES-WORKER2 ~]#
```

Step 6. Install kubeadm and kubectl on Alma Linux 9

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                                                                     8.7 MB/s |  28 MB     00:03
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                                1/1
  Installing       : kubectl-1.25.2-0.x86_64                                                        1/3
  Installing       : cri-tools-1.25.0-0.x86_64                                                      2/3
  Installing       : kubeadm-1.25.2-0.x86_64                                                        3/3
  Running scriptlet: kubeadm-1.25.2-0.x86_64                                                        3/3
  Verifying        : cri-tools-1.25.0-0.x86_64                                                      1/3
  Verifying        : kubeadm-1.25.2-0.x86_64                                                        2/3
  Verifying        : kubectl-1.25.2-0.x86_64                                                        3/3

Installed:
  cri-tools-1.25.0-0.x86_64          kubeadm-1.25.2-0.x86_64          kubectl-1.25.2-0.x86_64

Complete!
[root@DEAN-KUBERNETES-MASTER etc]#
```

Step 7. Set hostnames

On your Master node < 41.185.61.18 >, update your hostname using the following command:

```
$ sudo hostnamectl set-hostname master-node
$ sudo exec bash
```

And on server -> worker1 -> 41.185.61.28

```
$ sudo hostnamectl set-hostname W-node1
$ sudo exec bash
```

And on server -> worker2 -> 41.185.61.142

```
$ sudo hostnamectl set-hostname W-node2
$ sudo exec bash
```

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

```
$ sudo yum install nano
$ sudo nano /etc/hosts
```

Add these to the file at the bottom

```
41.185.61.18 master-node
41.185.61.28 node1 W-node1
41.185.61.142 node2 W-node2
```

Step 8. 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 9. Add a firewall CSF to Alma Linux 9

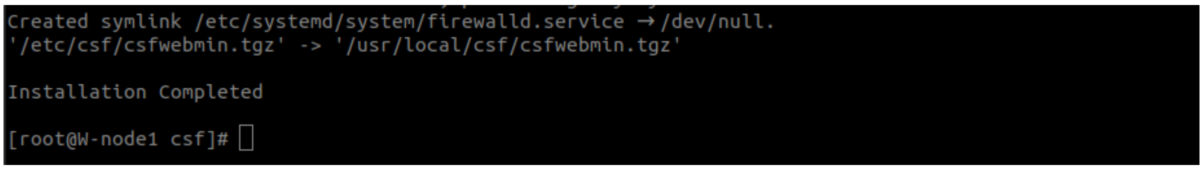
Begin first by installing the prerequisites using the following command:

```
$ dnf install perl-libwww-perl.noarch perl-LWP-Protocol-https.noarch perl-GDGraph wget
tar perl-Math-BigInt
```

Next, execute the following commands to download, extract and install CSF Firewall:

```
$ cd /usr/src
$ wget https://download.configserver.com/csf.tgz
$ tar -xzf csf.tgz
$ cd csf
$ sh install.sh
```

Wait for the Installation to **Complete.**

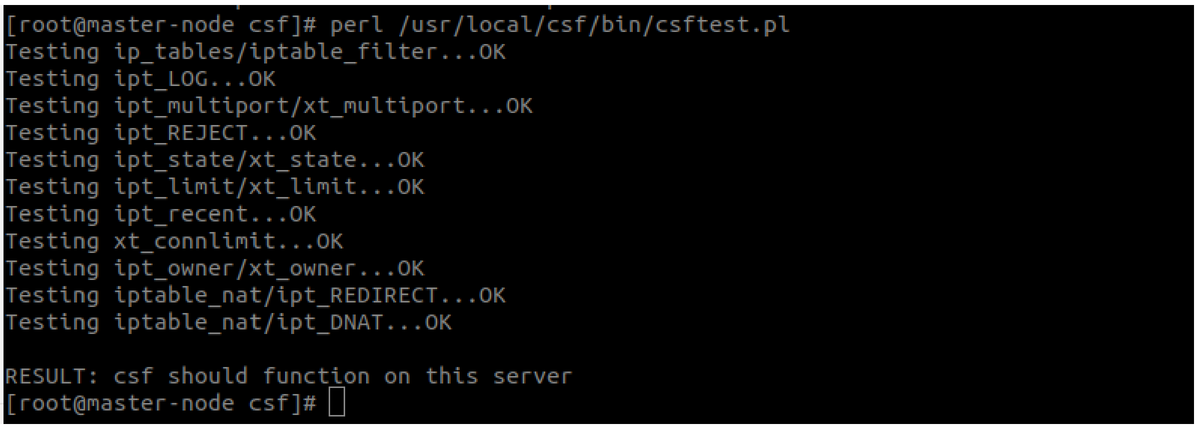


```
Created symlink /etc/systemd/system/firewalld.service → /dev/null.
'/etc/csf/csfwebmin.tgz' -> '/usr/local/csf/csfwebmin.tgz'

Installation Completed

[root@W-node1 csf]# 
```

Finally, you can now run this command to check if your cloud server has the required iptable modules:

```
$ sudo perl /usr/local/csf/bin/csftest.pl
```

Now, you should expect to see the following output, confirming everything is well:



```
[root@master-node csf]# perl /usr/local/csf/bin/csftest.pl
Testing ip_tables/iptable_filter...OK
Testing ipt_LOG...OK
Testing ipt_multiport/xt_multiport...OK
Testing ipt_REJECT...OK
Testing ipt_state/xt_state...OK
Testing ipt_limit/xt_limit...OK
Testing ipt_recent...OK
Testing xt_connlimit...OK
Testing ipt_owner/xt_owner...OK
Testing iptable_nat/ipt_REDIRECT...OK
Testing iptable_nat/ipt_DNAT...OK

RESULT: csf should function on this server
[root@master-node csf]#
```

Run the following commands to disable firewalld

```
$ sudo systemctl stop firewalld
$ sudo systemctl disable firewalld
```

Note that if you have already installed any other iptables firewall script, such as the APF, do remove or disable it too.

Start using CSF Firewall

Let us being by running the following command to turn off testing mode:

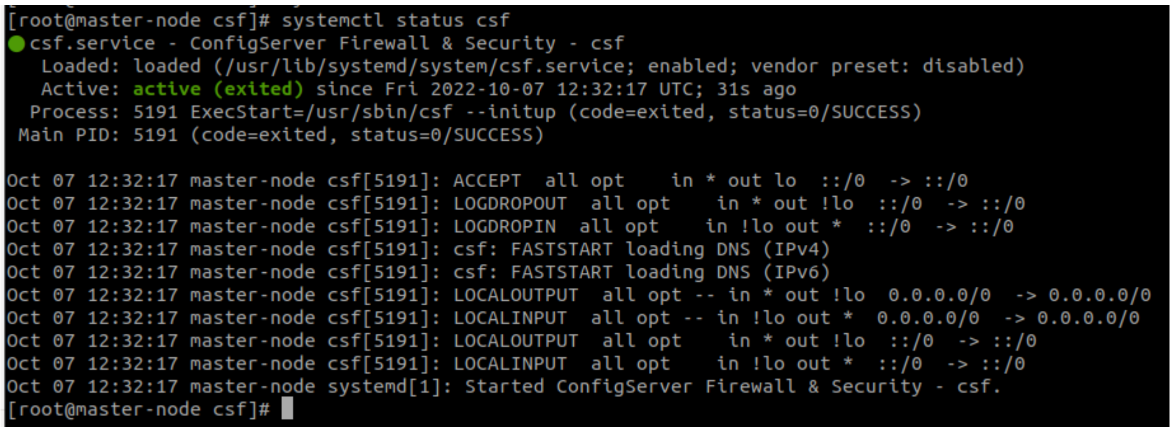
```
$ sudo sed 's/TESTING = "1"/TESTING = "0"/g' /etc/csf/csf.conf
```

You can now start up CSF firewall

```
$ sudo systemctl start csf
$ sudo systemctl enable csf
```

Now lets check if all is in good standing you can run the following command:

```
$ sudo systemctl status csf
```



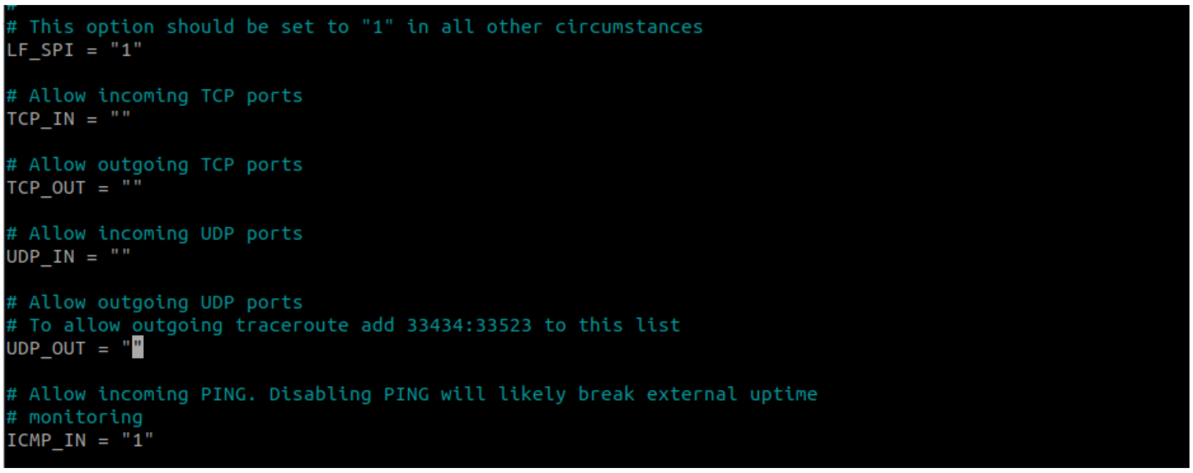
```
[root@master-node csf]# systemctl status csf
● csf.service - ConfigServer Firewall & Security - csf
   Loaded: loaded (/usr/lib/systemd/system/csf.service; enabled; vendor preset: disabled)
   Active: active (exited) since Fri 2022-10-07 12:32:17 UTC; 31s ago
  Process: 5191 ExecStart=/usr/sbin/csf --initup (code=exited, status=0/SUCCESS)
 Main PID: 5191 (code=exited, status=0/SUCCESS)

Oct 07 12:32:17 master-node csf[5191]: ACCEPT  all opt    in * out lo  ::/0  -> ::/0
Oct 07 12:32:17 master-node csf[5191]: LOGDROPOUT  all opt    in * out !lo  ::/0  -> ::/0
Oct 07 12:32:17 master-node csf[5191]: LOGDROPIN  all opt    in !lo out *  ::/0  -> ::/0
Oct 07 12:32:17 master-node csf[5191]: csf: FASTSTART loading DNS (IPv4)
Oct 07 12:32:17 master-node csf[5191]: csf: FASTSTART loading DNS (IPv6)
Oct 07 12:32:17 master-node csf[5191]: LOCALOUTPUT  all opt -- in * out !lo  0.0.0.0/0  -> 0.0.0.0/0
Oct 07 12:32:17 master-node csf[5191]: LOCALINPUT  all opt -- in !lo out *  0.0.0.0/0  -> 0.0.0.0/0
Oct 07 12:32:17 master-node csf[5191]: LOCALOUTPUT  all opt    in * out !lo  ::/0  -> ::/0
Oct 07 12:32:17 master-node csf[5191]: LOCALINPUT  all opt    in !lo out *  ::/0  -> ::/0
Oct 07 12:32:17 master-node systemd[1]: Started ConfigServer Firewall & Security - csf.
[root@master-node csf]#
```

Step 10. 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:

Remove all ports coming in by running this command then save and exit the file:

```
$ sudo nano /etc/csf/csf.conf
```

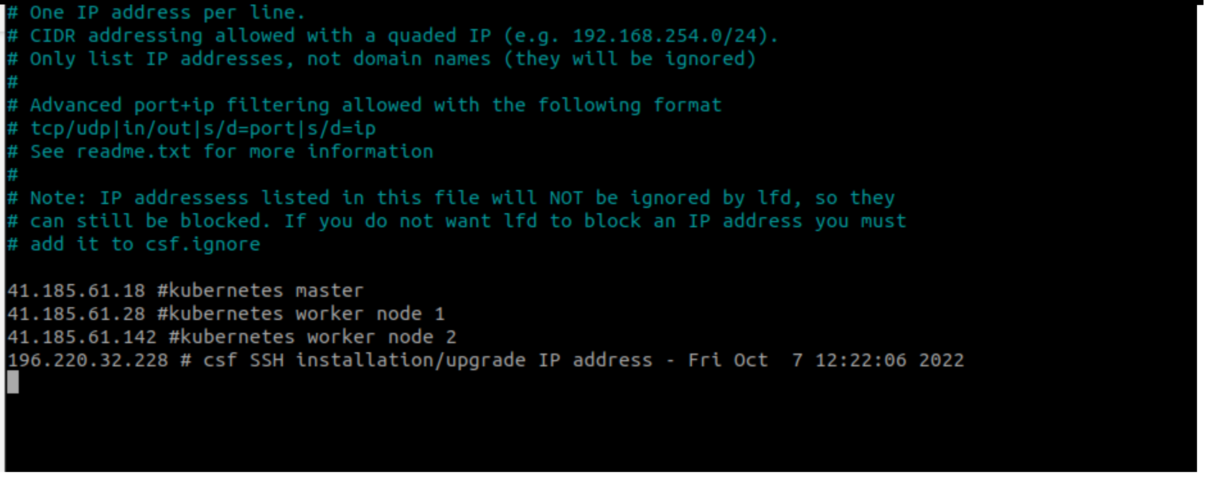


```
# This option should be set to "1" in all other circumstances
LF_SPI = "1"

# Allow incoming TCP ports
TCP_IN = ""

# Allow outgoing TCP ports
TCP_OUT = ""

# Allow incoming UDP ports
UDP_IN = ""

# Allow outgoing UDP ports
# To allow outgoing traceroute add 33434:33523 to this list
UDP_OUT = ""

# Allow incoming PING. Disabling PING will likely break external uptime
# monitoring
ICMP_IN = "1"
```

2nd Floor
360 Business Park
8 Wessex Street
Paarden Eiland
Cape Town
7405
South Africa

Add the master and worker ipaddresses to the csf.allow file:

```
$ sudo nano /etc/csf/csf.allow
```



```
# One IP address per line.
# CIDR addressing allowed with a quaded IP (e.g. 192.168.254.0/24).
# Only list IP addresses, not domain names (they will be ignored)
#
# Advanced port+ip filtering allowed with the following format
# tcp/udp|in/out|s/d=port|s/d=ip
# See readme.txt for more information
#
# Note: IP addressess listed in this file will NOT be ignored by lfd, so they
# can still be blocked. If you do not want lfd to block an IP address you must
# add it to csf.ignore

41.185.61.18 #kubernetes master
41.185.61.28 #kubernetes worker node 1
41.185.61.142 #kubernetes worker node 2
196.220.32.228 # csf SSH installation/upgrade IP address - Fri Oct  7 12:22:06 2022
```

Now Restart the firewall

```
$ sudo csf –r
```

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

Step 11. 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:

```
$ sudo nano /etc/sysctl.d/k8s.conf
```

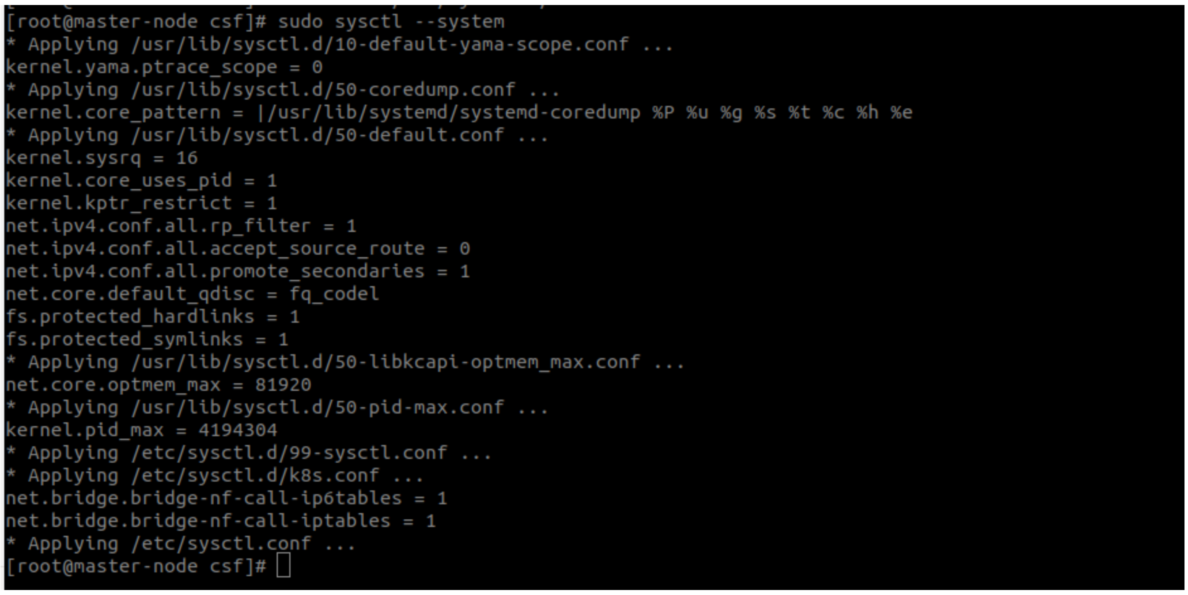
Add the following two lines to the file at the bottom

```
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
```

Now Run the following command to update the iptable rules for kubernetes

```
$ sudo sysctl –system
```

You should get the following output:



```
[root@master-node csf]# sudo sysctl --system
* Applying /usr/lib/sysctl.d/10-default-yama-scope.conf ...
kernel.yama.ptrace_scope = 0
* Applying /usr/lib/sysctl.d/50-coredump.conf ...
kernel.core_pattern = |/usr/lib/systemd/systemd-coredump %P %u %g %s %t %c %h %e
* Applying /usr/lib/sysctl.d/50-default.conf ...
kernel.sysrq = 16
kernel.core_uses_pid = 1
kernel.kptr_restrict = 1
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.all.promote_secondaries = 1
net.core.default_qdisc = fq_codel
fs.protected_hardlinks = 1
fs.protected_symlinks = 1
* Applying /usr/lib/sysctl.d/50-libkcapi-optmem_max.conf ...
net.core.optmem_max = 81920
* Applying /usr/lib/sysctl.d/50-pid-max.conf ...
kernel.pid_max = 4194304
* 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@master-node csf]# 
```

Step 12. Disable swap

For Kubelet to work, we also need to disable swap on all Master and Slaves:

```
$ sudo sed -i '/swap/d' /etc/fstab
$ sudo swapoff –a
```

KUBERNETES DEPLOYMENT CLUSTER – RED HAT [[ALMALINUX]]…

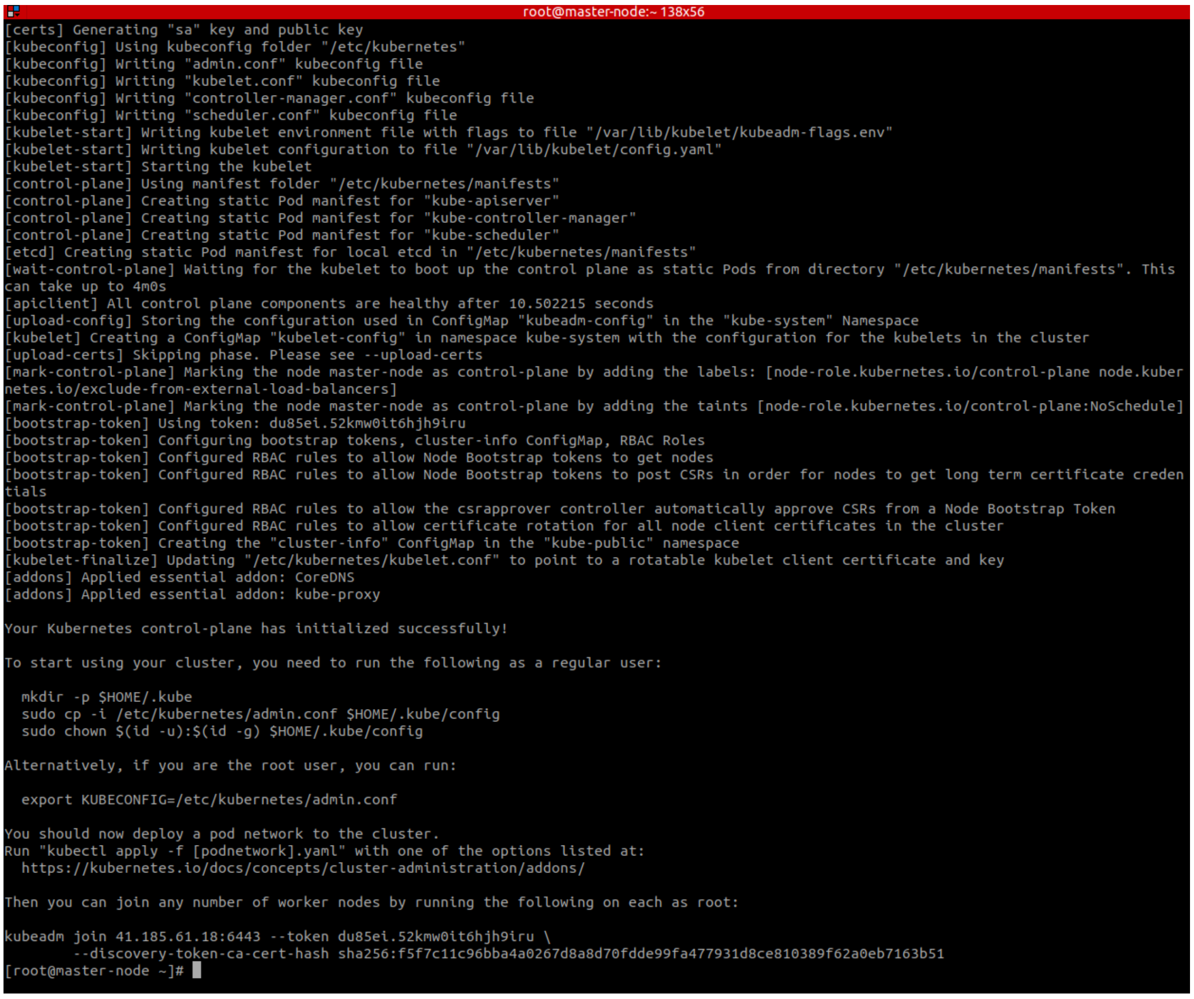
----

1. Kubeadm initialization

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

```
$ dnf install -y iproute-tc
$ swapoff -a
$ sudo systemctl enable kubelet.service
$ sudo systemctl start kubelet.service
$ systemctl restart containerd
$ sudo kubeadm init
```

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



```
[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
[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
[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"
[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 10.502215 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.kuber
netes.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: du85ei.52kmw0it6hjh9iru
[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 creden
tials
[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 41.185.61.18:6443 --token du85ei.52kmw0it6hjh9iru \
        --discovery-token-ca-cert-hash sha256:f5f7c11c96bba4a0267d8a8d70fdde99fa477931d8ce810389f62a0eb7163b51
[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.

```
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 41.185.61.18:6443 --token du85ei.52kmw0it6hjh9iru --discovery-token-ca-
cert-hash
sha256:f5f7c11c96bba4a0267d8a8d70fdde99fa477931d8ce810389f62a0eb7163b51
```

! Important: 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
```

Run these commands on worker-nodes before trying to join node to master:

```
$ dnf install -y iproute-tc
$ swapoff -a
$ sudo systemctl enable kubelet.service
$ sudo systemctl start kubelet.service
$ systemctl restart containerd
```

2. Create required directories and start managing Kubernetes clusters

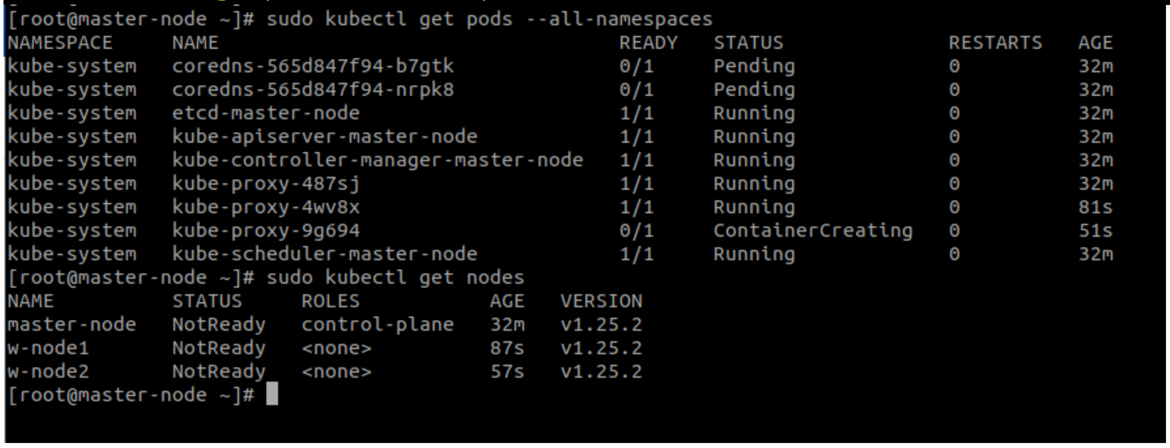
In order to start managing your cluster, you need to create a directory and assume ownership. Run the following commands as a regular user on all systems:

```
$ 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
```

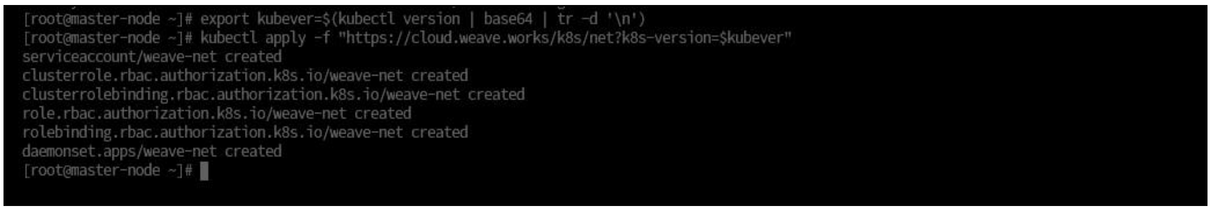


```
[root@master-node ~]# sudo kubectl get pods --all-namespaces
NAMESPACE     NAME                                  READY   STATUS              RESTARTS   AGE
kube-system   coredns-565d847f94-b7gtk              0/1     Pending             0          32m
kube-system   coredns-565d847f94-nrpk8              0/1     Pending             0          32m
kube-system   etcd-master-node                      1/1     Running             0          32m
kube-system   kube-apiserver-master-node            1/1     Running             0          32m
kube-system   kube-controller-manager-master-node   1/1     Running             0          32m
kube-system   kube-proxy-487sj                      1/1     Running             0          32m
kube-system   kube-proxy-4wv8x                      1/1     Running             0          81s
kube-system   kube-proxy-9g694                      0/1     ContainerCreating   0          51s
kube-system   kube-scheduler-master-node            1/1     Running             0          32m
[root@master-node ~]# sudo kubectl get nodes
NAME          STATUS     ROLES           AGE   VERSION
master-node   NotReady   control-plane   32m   v1.25.2
w-node1       NotReady   <none>          87s   v1.25.2
w-node2       NotReady   <none>          57s   v1.25.2
[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 –ouput=json | base64 | tr -d '\n')
$ sudo kubectl apply -f https://cloud.weave.works/k8s/net?k8s-version=$kubever
```

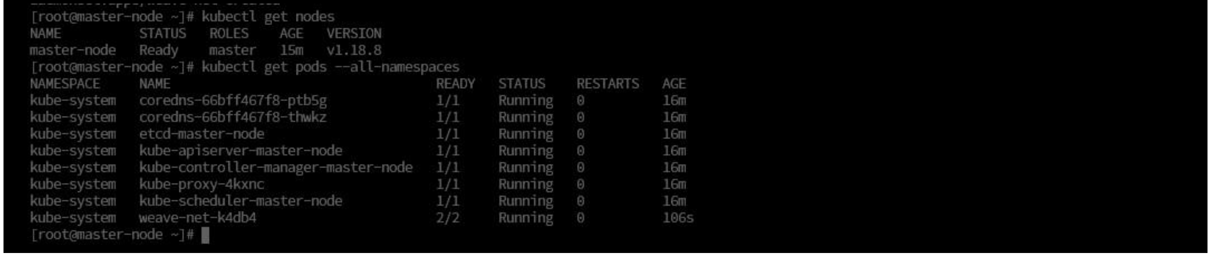
You should get the following output:



```
[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:

```
$ kubectl get pod --all-namespaces -o wide
```



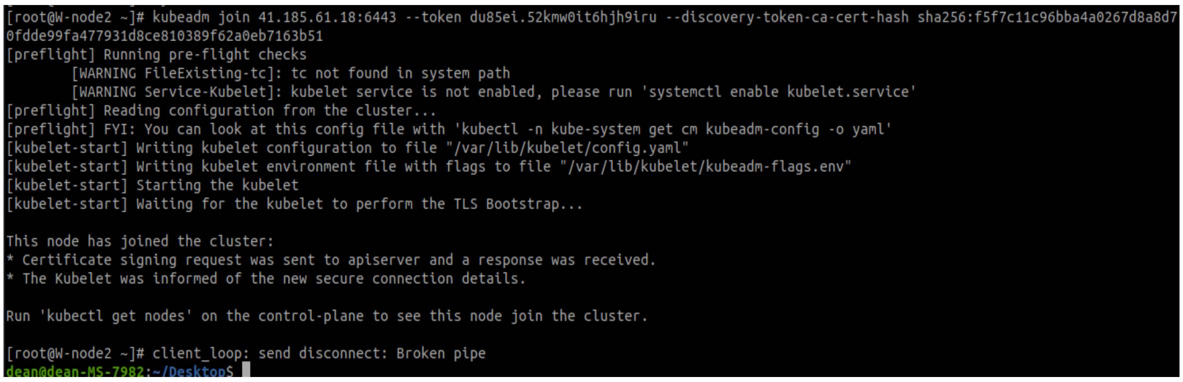
```
[root@master-node ~]# kubectl get nodes
NAME          STATUS   ROLES    AGE   VERSION
master-node   Ready    master   15m   v1.18.8
[root@master-node ~]# kubectl get pods --all-namespaces
NAMESPACE     NAME                                  READY   STATUS    RESTARTS   AGE
kube-system   coredns-66bff467f8-ptb5g              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 on all of the worker node:

```
$ sudo kubeadm join 41.185.61.18:6443 --token du85ei.52kmw0it6hjh9iru --discovery-token-ca-cert-hash sha256:f5f7c11c96bba4a0267d8a8d70fdde99fa477931d8ce810389f62a0eb7163b51
```

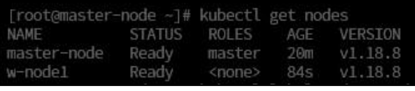
On successful addition, you should get the following output:



```
[root@W-node2 ~]# kubeadm join 41.185.61.18:6443 --token du85ei.52kmw0it6hjh9iru --discovery-token-ca-cert-hash sha256:f5f7c11c96bba4a0267d8a8d7
0fdde99fa477931d8ce810389f62a0eb7163b51
[preflight] Running pre-flight checks
        [WARNING FileExisting-tc]: tc not found in system path
        [WARNING Service-Kubelet]: kubelet service is not enabled, please run 'systemctl enable kubelet.service'
[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.

[root@W-node2 ~]# client_loop: send disconnect: Broken pipe
dean@dean-MS-7982:~/Desktop$
```

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

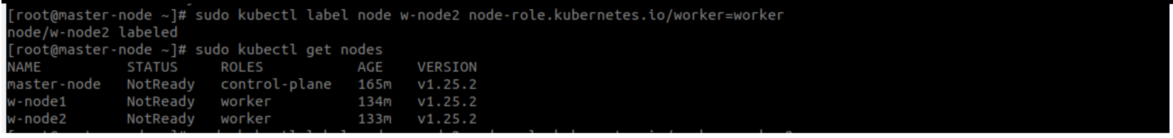
```
$ sudo kubectl get nodes
```



```
[root@master-node ~]# kubectl get nodes
NAME          STATUS   ROLES    AGE   VERSION
master-node   Ready    master   20m   v1.18.8
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 ~]# sudo kubectl label node w-node2 node-role.kubernetes.io/worker=worker
node/w-node2 labeled
[root@master-node ~]# sudo kubectl get nodes
NAME          STATUS     ROLES           AGE    VERSION
master-node   NotReady   control-plane   165m   v1.25.2
w-node1       NotReady   worker          134m   v1.25.2
w-node2       NotReady   worker          133m   v1.25.2
```

END OF FILE ….