

## installation of docker

1. yum install docker -y
2. systemctl enable docker --now
3. docker info

```
[root@ip-172-31-49-3 ~]# yum install docker -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-
: motd

[root@ip-172-31-49-3 ~]# systemctl enable docker --now
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.
[root@ip-172-31-49-3 ~]# docker info
Client:
 Debug Mode: false

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 0
 Server Version: 19.03.13-ce
 Storage Driver: overlay2
  Backing Filesystem: xfs
  Supports d_type: true
  Native Overlay Diff: true
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
```

## installation of kubeadm

1. yum install kubeadm - fails
2. vi /etc/yum.repos.d/kubernetes.repo - need to setup the repo
3. yum repolist
4. yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
5. systemctl status kubelet
6. systemctl enable kubelet --now

```
[root@ip-172-31-49-3 ~]# vi /etc/yum.repos.d/kubernetes.repo
[root@ip-172-31-49-3 ~]# cat /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-$basearch
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
exclude=kubelet kubeadm kubectl
[root@ip-172-31-49-3 ~]# yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                               | 3.7 kB  00:00:00
kubernetes/x86_64/signature              | 844 B  00:00:00
Retrieving key from https://packages.cloud.google.com/yum/doc/yum-key.gpg
Importing GPG key 0xA7317B0F:

[root@ip-172-31-49-3 ~]# systemctl enable kubelet --now
Created symlink from /etc/systemd/system/multi-user.target.wants/kubelet.service to /usr/lib/systemd/system/kubelet.service.
[root@ip-172-31-49-3 ~]# systemctl status kubelet
● kubelet.service - kubelet: The Kubernetes Node Agent
   Loaded: loaded (/usr/lib/systemd/system/kubelet.service; enabled; vendor preset: disabled)
   Drop-In: /usr/lib/systemd/system/kubelet.service.d
            └─10-kubeadm.conf
       Active: activating (auto-restart) (Result: exit-code) since Wed 2021-02-03 14:20:22 UTC; 7s ago
       Docs: https://kubernetes.io/docs/
   Process: 1889 ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_CONFIG_ARGS $KUBELET_KUBEADM_ARGS $KUBELET_EXTRA_ARGS (code
e=exited, status=255)
   Main PID: 1889 (code=exited, status=255)

Feb 03 14:20:22 ip-172-31-49-3.ec2.internal kubelet[1889]: created by k8s.io/kubernetes/vendor/go.opencensus.io/stats/view.init.0
Feb 03 14:20:22 ip-172-31-49-3.ec2.internal systemd[1]: kubelet.service: main process exited, code=exited, status=255/n/a
Feb 03 14:20:22 ip-172-31-49-3.ec2.internal kubelet[1889]: /usr/bin/kubelet: output/dockerized/go/exe/k8s.io/kubelet: 10x57
```

## joining.... -- fails

```
[root@ip-172-31-58-206 ~]# kubeadm join
discovery: Invalid value: "": bootstrapToken or file must be set
To see the stack trace of this error execute with --v=5 or higher
[root@ip-172-31-58-206 ~]#
```

➤ a token is required to allow worker node to get connected to master

➔ go to master create a new token..

a. kubeadm token list

```
[root@ip-172-31-52-186 ~]# kubeadm token list
TOKEN                                TTL    EXPIRES    USAGES    E
XTRA GROUPS
y3934u.hvqgghz22ouc6rs 22h    2021-02-04T12:16:14Z  authenticati
on,signing The default bootstrap token generated by 'kubeadm init'. s
ystem:bootstrappers:kubeadm:default-node-token
[root@ip-172-31-52-186 ~]#
```

it is the pre-created token, it was also the last time of kubeadm init command.

See its same, I have copied this form master previous ss.

```
kubeadm join 172.31.52.186:6443 --token y3934u.hvqgghz22ouc6rs \
--discovery-token-ca-cert-hash sha256:c140d470d1bd13e48adf450bae48aae0d88a94f42472a5aa06842302d95f295e
```

b. master's IP : ifconfig

```
[root@ip-172-31-52-186 ~]# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 9001
    inet 172.31.52.186 netmask 255.255.240.0 broadcast 172.31.63.255
    inet6 fe80::4c5:9dff:fe79:af95 prefixlen 64 scopeid 0x20<link>
    ether 06:c5:9d:79:af:95 txqueuelen 1000 (Ethernet)
    RX packets 301944 bytes 422451131 (402.8 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 43071 bytes 4042135 (3.8 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

c. kubeadm token create --print-join-command

```
To see the stack trace of this error execute with --v=5 or higher
[root@ip-172-31-52-186 ~]# kubeadm token create --print-join-command
kubeadm join 172.31.52.186:6443 --token 6m5ot5.kihdgb1std1zjup --dis
covery-token-ca-cert-hash sha256:c140d470d1bd13e48adf450bae48aae0d88a94f4
2472a5aa06842302d95f295e
[root@ip-172-31-52-186 ~]#
```

```
kubeadm join 172.31.52.186:6443 --token 6m5ot5.kihdgb1std1zjup --discovery-token-ca-cert-hash
sha256:c140d470d1bd13e48adf450bae48aae0d88a94f42472a5aa06842302d95f295e
```

go to Worker(172-21-49-3)

➔ same setup we have to kubeadm all trouble shoot all the errors.

```
[root@ip-172-31-49-3 ~]# kubeadm join 172.31.52.186:6443 --token 6m5ot5.kihdgb1std1zjup --discovery-token-ca-cert-hash sha256:c140d470d1bd13e48adf450bae48aae0d88a94f42472a5aa06842302d95f295e
[preflight] Running pre-flight checks
[WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommended driver is "systemd". Please follow the guide at https://kubernetes.io/docs/setup/cri/
[WARNING FileExisting-tc]: tc not found in system path
error execution phase preflight: [preflight] Some fatal errors occurred:
[ERROR FileContent--proc-sys-net-bridge-bridge-nf-call-iptables]: /proc/sys/net/bridge/bridge-nf-call-iptables 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@ip-172-31-49-3 ~]#
```

1st error: [WARNING IsDockerSystemdCheck]: we have to use system instead of cgroups.

```
vi /etc/docker/daemon.json
```

```
{
    "exec-opts": ["native.cgroupdriver=systemd"]
}
```

```
cat /etc/docker/daemon.json
```

```
systemctl restart docker
```

```
docker info | grep Driver
```

```
[root@ip-172-31-49-3 ~]# vi /etc/docker/daemon.json
[root@ip-172-31-49-3 ~]# cat /etc/docker/daemon.json
{
    "exec-opts": ["native.cgroupdriver=systemd"]
}
[root@ip-172-31-49-3 ~]# systemctl restart docker
[root@ip-172-31-49-3 ~]# docker info | grep Driver
WARNING: bridge-nf-call-iptables is disabled
WARNING: bridge-nf-call-ip6tables is disabled
Storage Driver: overlay2
Logging Driver: json-file
Cgroup Driver: systemd
...

[root@ip-172-31-49-3 ~]# kubeadm join 172.31.52.186:6443 --token 6m5ot5.kihdgb1stdlzpup --discovery-token-ca-cert-hash sha256:c140
d470d1bd13e48adf450bae48aae0d88a94f42472a5aa06842302d95f295e
[preflight] Running pre-flight checks
[WARNING FileExisting-tc]: tc not found in system path
error execution phase preflight: [preflight] Some fatal errors occurred:
[ERROR FileContent--proc-sys-net-bridge-bridge-nf-call-iptables]: /proc/sys/net/bridge/bridge-nf-call-iptables 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
```

2nd error: [WARNING FileExisting-tc]: tc not found in system path

```
yum install -y iproute-tc
```

```
[root@ip-172-31-49-3 ~]# yum install -y iproute-tc
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core | 3.7 kB 00:00:00
...

[root@ip-172-31-49-3 ~]# kubeadm join 172.31.52.186:6443 --token 6m5ot5.kihdgb1stdlzpup --discovery-token-ca-cert-hash sha256:c140
d470d1bd13e48adf450bae48aae0d88a94f42472a5aa06842302d95f295e
[preflight] Running pre-flight checks
error execution phase preflight: [preflight] Some fatal errors occurred:
[ERROR FileContent--proc-sys-net-bridge-bridge-nf-call-iptables]: /proc/sys/net/bridge/bridge-nf-call-iptables 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
```

3rd error: FileContent—proc-sys-net-bridge-bridge-nf-call-iptables

```
vim /etc/sysctl.d/k8s.conf
```

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

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

```
cat /etc/sysctl.d/k8s.conf
```

```
sysctl --system
```

```
sysctl -a | grep bridge-bridge-nf-call
```

```
[root@ip-172-31-49-3 ~]# vim /etc/sysctl.d/k8s.conf
[root@ip-172-31-49-3 ~]# cat /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
[root@ip-172-31-49-3 ~]# sysctl -system
sysctl: cannot stat /proc/sys/-system: No such file or directory
[root@ip-172-31-49-3 ~]# sysctl --system
* Applying /etc/sysctl.d/00-defaults.conf ...
kernel.printk = 8 4 1 7
kernel.panic = 30
net.ipv4.neigh.default.gc_thresh1 = 0
```

```
[root@ip-172-31-49-3 ~]# sysctl -a | grep bridge-bridge-nf-call
sysctl: reading key "net.ipv6.conf.all.stable_secret"
sysctl: reading key "net.ipv6.conf.default.stable_secret"
sysctl: reading key "net.ipv6.conf.docker0.stable_secret"
sysctl: reading key "net.ipv6.conf.eth0.stable_secret"
sysctl: reading key "net.ipv6.conf.lo.stable_secret"
```

DONE..

```
[root@ip-172-31-49-3 ~]# kubeadm join 172.31.52.186:6443 --token 6m5ot5.kihdgb1stsd1zjup --discovery-token-ca-cert-hash sha256:c140d470d1bd13e48adf450bae48aae0d88a94f42472a5aa06842302d95f295e
[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 -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@ip-172-31-49-3 ~]#
```

---

Go to master

➤ kubectl get nodes

```
[root@ip-172-31-52-186 ~]# kubectl get node
NAME                                STATUS    ROLES    AGE    VERSION
ip-172-31-49-3.ec2.internal        NotReady <none>   8m11s  v1.20.2
ip-172-31-52-186.ec2.internal      NotReady control-plane,master 150m   v1.20.2
[root@ip-172-31-52-186 ~]#
```

➤ kubectl create deploy myd --image=httpd

```
[root@ip-172-31-52-186 ~]# kubectl create deploy myd --image=httpd
deployment.apps/myd created
[root@ip-172-31-52-186 ~]# kubectl get deploy
NAME    READY    UP-TO-DATE    AVAILABLE    AGE
myd     0/1      1             0            40s
[root@ip-172-31-52-186 ~]#
[root@ip-172-31-52-186 ~]# kubectl get pods
NAME                                READY    STATUS    RESTARTS    AGE
myd-7cf9bb6c54-ppssg                0/1     Pending   0            53s
[root@ip-172-31-52-186 ~]#
```

```
[root@ip-172-31-52-186 ~]# kubectl describe pods myd
Name:          myd-7cf9bb6c54-ppssg
Namespace:     default
```

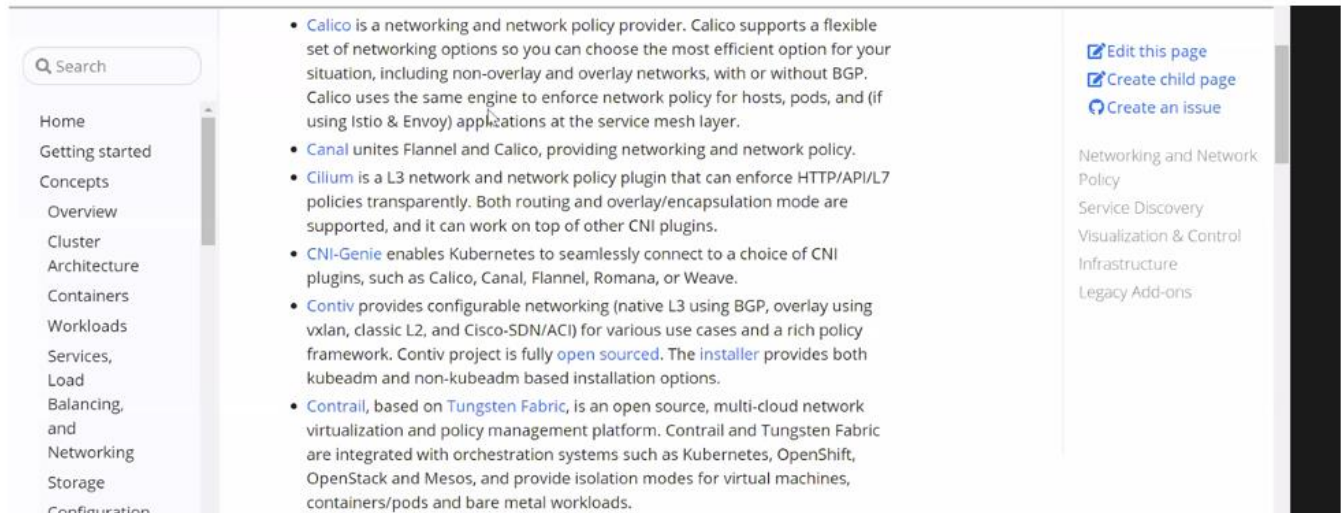
```
node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason             Age          From          Message
  ----     -
Warning   FailedScheduling   84s (x3 over 2m37s)  default-scheduler  0/2 nodes are available: 1 node(s) had taint {node-role.kubernetes.io/master: }, that the pod didn't tolerate, 1 node(s) had taint {node.kubernetes.io/not-ready: }, that the pod didn't tolerate.
[root@ip-172-31-52-186 ~]#
```

failing scheduling.

master scheduler looking for pod and they cant find it.

➔ how to make it ready?

lots of CNI need to develop



The screenshot shows the Kubernetes documentation page for CNI plugins. On the left is a navigation menu with links like Home, Getting started, Concepts, Overview, Cluster, Architecture, Containers, Workloads, Services, Load Balancing, and Networking. The main content area lists several CNI plugins: Calico, Canal, Cilium, CNI-Genie, Contiv, and Contrail. Each plugin description includes its purpose and supported features. On the right, there are links to 'Edit this page', 'Create child page', and 'Create an issue', along with a table of contents for the 'Networking and Network Policy' section.

- **Calico** is a networking and network policy provider. Calico supports a flexible set of networking options so you can choose the most efficient option for your situation, including non-overlay and overlay networks, with or without BGP. Calico uses the same engine to enforce network policy for hosts, pods, and (if using Istio & Envoy) applications at the service mesh layer.
- **Canal** unites Flannel and Calico, providing networking and network policy.
- **Cilium** is a L3 network and network policy plugin that can enforce HTTP/API/L7 policies transparently. Both routing and overlay/encapsulation mode are supported, and it can work on top of other CNI plugins.
- **CNI-Genie** enables Kubernetes to seamlessly connect to a choice of CNI plugins, such as Calico, Canal, Flannel, Romana, or Weave.
- **Contiv** provides configurable networking (native L3 using BGP, overlay using vxlan, classic L2, and Cisco-SDN/ACI) for various use cases and a rich policy framework. Contiv project is fully [open sourced](#). The [installer](#) provides both kubeadm and non-kubeadm based installation options.
- **Contrail**, based on [Tungsten Fabric](#), is an open source, multi-cloud network virtualization and policy management platform. Contrail and Tungsten Fabric are integrated with orchestration systems such as Kubernetes, OpenShift, OpenStack and Mesos, and provide isolation modes for virtual machines, containers/pods and bare metal workloads.

here flannel is req:

`kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml`

```
[root@ip-172-31-52-186 ~]# kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
podsecuritypolicy.policy/psp.flannel.unprivileged created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
[root@ip-172-31-52-186 ~]#
```

➤ again checking now.

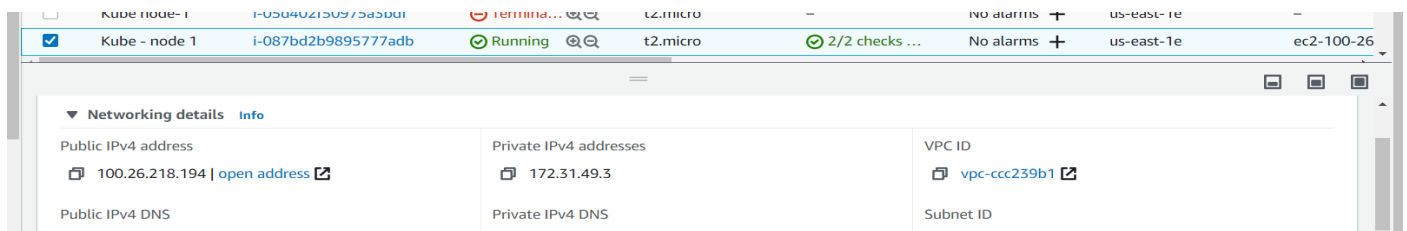
```
[root@ip-172-31-52-186 ~]# kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
myd-7cf9bb6c54-ppsg 1/1     Running   0           7m26s
[root@ip-172-31-52-186 ~]#
```

➤ -o wide will tell in which ip the pod is running.

`kubectl get pods -o wide`

```
[root@ip-172-31-52-186 ~]# kubectl get pods -o wide
NAME                READY   STATUS    RESTARTS   AGE   IP            NODE                NOMINATED NODE   READINESS GATES
myd-7cf9bb6c54-ppsg 1/1     Running   0           8m5s  10.240.1.4    ip-172-31-49-3.ec2.internal  <none>            <none>
[root@ip-172-31-52-186 ~]#
```

none => worker node, also verify the ip.



The screenshot shows the AWS Management Console interface for a Kubernetes node. The top bar indicates the node is 'Running' and has '2/2 checks' passed. Below this, the 'Networking details' section is expanded, showing the node's public and private IP addresses, DNS information, VPC ID, and Subnet ID. The node is identified as 'Kube - node 1' with the ID 'i-087bd2b9895777adb'.

➤ the ip range is also as we mentioned in kubeadm init

➤ lets expose the port now

`kubectl expose deploy myd --port=80 --type=NodePort`

```
[root@ip-172-31-52-186 ~]# kubectl expose deploy myd --port=80 --type=NodePort
service/myd exposed
[root@ip-172-31-52-186 ~]#
```

```
[root@ip-172-31-52-186 ~]# kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
kubernetes   ClusterIP    10.96.0.1        <none>            443/TCP          3h8m
myd          NodePort     10.110.49.182    <none>            80:31022/TCP     58s
[root@ip-172-31-52-186 ~]#
```

port exposed is : 31022

worker node public IP is mentioned above as : 100.26.218.194

← → ↻ ⚠ Not secure | 100.26.218.194:31022 ☆ 🏠 ⚙️ 👤 ⋮

**It works!**

working fine.....