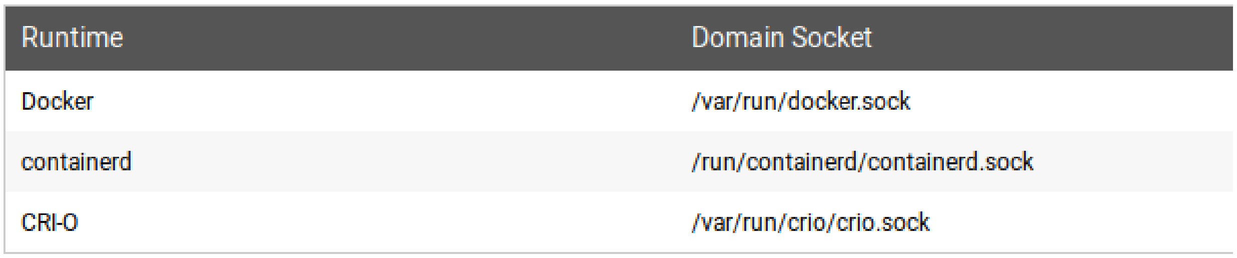
Install container runtime (Docker)

```
sudo apt-get update
sudo apt-get install apt-transport-https ca-certificates curl
software-properties-common
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key
add -
sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) \
stable"
sudo apt-get update
sudo apt-get install docker-ce=18.06.2~ce~3-0~ubuntu -y
cat | sudo tee /etc/docker/daemon.json <<EOF
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
EOF
sudo mkdir -p /etc/systemd/system/docker.service.d
sudo systemctl daemon-reload
sudo systemctl restart docker
```

Installation using kubeadm

- kubeadm will try to automatically detect the container runtime on Linux nodes by scanning through a list of well known domain sockets.



Runtime	Domain Socket
Docker	/var/run/docker.sock
containerd	/run/containerd/containerd.sock
CRI-O	/var/run/crio/crio.sock

- If both Docker and containerd are detected together, Docker takes precedence. This is needed, because Docker 18.09 ships with containerd and both are detectable. If any other two or more runtimes are detected, kubeadm will exit with an appropriate error message.
- On non-Linux nodes the container runtime used by default is Docker.

Install components (kubeadm, kubelet, kubectl)

- kubeadm: the command to bootstrap the cluster.
- kubelet: the component that runs on all of the machines in your cluster and does things like starting pods and containers.
- kubectl: the command line util to talk to your cluster.

```
apt-get update && apt-get install -y apt-transport-https curl
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add
-
cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF
apt-get update
apt-get install -y kubelet kubeadm kubectl
apt-mark hold kubelet kubeadm kubectl
```

Cgroup driver

- When using Docker, kubeadm will automatically detect the cgroup driver for the kubelet and set it in the **/var/lib/kubelet/kubeadm-flags.env** file during runtime.
- This file will be used by **kubeadm init and kubeadm join** to source extra user defined arguments for the kubelet.

Initialize control plane node(kubeadm init)

Using flannel network...

- **Kubeadm init --pod-network-cidr=10.244.0.0/16**

1. kubeadm will try to detect the container runtime on Linux by using a list of well known domain socket paths
   To specify manually
   --cri-socket
2. kubeadm uses the network interface associated with the default gateway to advertise the master's IP
   --apiserver-advertise-address=<ip-address>

3. (Optional) Run **kubeadm config images pull** prior to kubeadm init to verify connectivity to gcr.io registries.
4. Details steps are listed in link below
   https://kubernetes.io/docs/reference/setup-tools/kubeadm/kubeadm-init/

The "init" command executes the following phases:

```
preflight                    Run pre-flight checks
kubelet-start                Write kubelet settings and (re)start the kubelet
certs                        Certificate generation
  /etcd-ca                     Generate the self-signed CA to provision
identities for etcd
  /apiserver-etcd-client       Generate the certificate the apiserver uses to
access etcd
  /etcd-healthcheck-client     Generate the certificate for liveness probes
to healtcheck etcd
  /etcd-server                 Generate the certificate for serving etcd
  /etcd-peer                   Generate the certificate for etcd nodes to
communicate with each other
  /ca                          Generate the self-signed Kubernetes CA to
provision identities for other Kubernetes components
  /apiserver                   Generate the certificate for serving the
Kubernetes API
  /apiserver-kubelet-client    Generate the certificate for the API server to
connect to kubelet
  /front-proxy-ca              Generate the self-signed CA to provision
identities for front proxy
  /front-proxy-client          Generate the certificate for the front proxy
client
  /sa                          Generate a private key for signing service
account tokens along with its public key
kubeconfig                   Generate all kubeconfig files necessary to
establish the control plane and the admin kubeconfig file
  /admin                       Generate a kubeconfig file for the admin to
use and for kubeadm itself
  /kubelet                     Generate a kubeconfig file for the kubelet to
use *only* for cluster bootstrapping purposes
  /controller-manager          Generate a kubeconfig file for the controller
manager to use
  /scheduler                   Generate a kubeconfig file for the scheduler
to use
control-plane                Generate all static Pod manifest files necessary
to establish the control plane
  /apiserver                   Generates the kube-apiserver static Pod
manifest
  /controller-manager          Generates the kube-controller-manager static
Pod manifest
  /scheduler                   Generates the kube-scheduler static Pod
manifest
```

```
etcd                           Generate static Pod manifest file for local etcd
  /local                         Generate the static Pod manifest file for a
local, single-node local etcd instance
upload-config                  Upload the kubeadm and kubelet configuration to
a ConfigMap
  /kubeadm                       Upload the kubeadm ClusterConfiguration to a
ConfigMap
  /kubelet                       Upload the kubelet component config to a
ConfigMap
upload-certs                   Upload certificates to kubeadm-certs
mark-control-plane             Mark a node as a control-plane
bootstrap-token                Generates bootstrap tokens used to join a node
to a cluster
addon                          Install required addons for passing Conformance
tests
  /coredns                       Install the CoreDNS addon to a Kubernetes
cluster
  /kube-proxy                    Install the kube-proxy addon to a Kubernetes
cluster
```

Uninstall

Removing Node:

- Drain
- Cordon
- Kubeadm reset
- Cleanup IP tables

```
imsrv01@master-0:/etc$ kubectl drain worker-0
--ignore-daemonsetsnode/worker-0 cordonedWARNING: ignoring
DaemonSet-managed Pods: kube-system/kube-flannel-ds-amd64-r5jn4,
kube-system/kube-proxy-j7h84
evicting pod "nginx"
evicting pod "busybox-64b47fb7f6-6tqfb"
pod/nginx evicted
pod/busybox-64b47fb7f6-6tqfb evicted
node/worker-0 evicted
```

```
imsrv01@master-0:/etc$ kubectl cordon worker-0
node/worker-0 already cordoned
```

```
imsrv01@worker-0:~$ sudo kubeadm reset
[reset] WARNING: Changes made to this host by 'kubeadm init' or 'kubeadm
join' will be reverted.
[reset] Are you sure you want to proceed? [y/N]: y
[preflight] Running pre-flight checks
W0811 17:08:10.559852   12655 removeetcdmember.go:79] [reset] No kubeadm
config, using etcd pod spec to get data directory
[reset] No etcd config found. Assuming external etcd
[reset] Please, manually reset etcd to prevent further issues
[reset] Stopping the kubelet service
[reset] Unmounting mounted directories in "/var/lib/kubelet"
[reset] Deleting contents of config directories: [/etc/kubernetes/manifests
/etc/kubernetes/pki]
[reset] Deleting files: [/etc/kubernetes/admin.conf
/etc/kubernetes/kubelet.conf /etc/kubernetes/bootstrap-kubelet.conf
/etc/kubernetes/controller-manager.conf /etc/kubernetes/scheduler
.conf]
[reset] Deleting contents of stateful directories: [/var/lib/kubelet
/etc/cni/net.d /var/lib/dockershim /var/run/kubernetes]
The reset process does not reset or clean up iptables rules or IPVS tables.
If you wish to reset iptables, you must do so manually.
For example:
iptables -F && iptables -t nat -F && iptables -t mangle -F && iptables -X
If your cluster was setup to utilize IPVS, run ipvsadm --clear (or similar)
to reset your system's IPVS tables.
The reset process does not clean your kubeconfig files and you must remove
them manually.
Please, check the contents of the $HOME/.kube/config file.
```

```
root@worker-0:~# iptables -F && iptables -t nat -F && iptables -t mangle -F
&& iptables -X
root@worker-0:~# ipvsadm -C
```

Upgrade

- Upgrade the primary control plane node.
- Upgrade additional control plane nodes.
- Upgrade worker nodes.

1. Upgrade kubeadm

a. Get latest version to apply
b. Upgrade

```
apt update
apt-cache policy kubeadm
# find the latest 1.14 version in the list
# it should look like 1.14.x-00, where x is the latest patch

# replace x in 1.14.x-00 with the latest patch version
apt-mark unhold kubeadm kubelet && \
apt-get update && apt-get install -y kubeadm=1.14.x-00 && \
apt-mark hold kubeadm

kubeadm version
```

2. Upgrade plan

```
root@master-0:~# kubeadm upgrade plan
[preflight] Running pre-flight checks.
[upgrade] Making sure the cluster is healthy:
[upgrade/config] Making sure the configuration is correct:
[upgrade/config] Reading configuration from the cluster...
[upgrade/config] FYI: You can look at this config file with 'kubectl -n
kube-system get cm kubeadm-config -oyaml'
[upgrade] Fetching available versions to upgrade to
```

```
[upgrade/versions] Cluster version: v1.13.9
[upgrade/versions] kubeadm version: v1.14.1
I0811 17:42:17.712882   16389 version.go:240] remote version is much newer:
v1.15.2; falling back to: stable-1.14
[upgrade/versions] Latest stable version: v1.14.5
[upgrade/versions] Latest version in the v1.13 series: v1.13.9

Components that must be upgraded manually after you have upgraded the
control plane with 'kubeadm upgrade apply':
COMPONENT   CURRENT     AVAILABLE
Kubelet     1 x v1.13.5   v1.14.5

Upgrade to the latest stable version:

COMPONENT          CURRENT   AVAILABLE
API Server         v1.13.9   v1.14.5
Controller Manager   v1.13.9   v1.14.5
Scheduler          v1.13.9   v1.14.5
Kube Proxy         v1.13.9   v1.14.5
CoreDNS            1.2.6       1.3.1
Etcd               3.2.24      3.3.10

You can now apply the upgrade by executing the following command:

      kubeadm upgrade apply v1.14.5

Note: Before you can perform this upgrade, you have to update kubeadm to
v1.14.5.
```

3. Upgrade apply

```
sudo kubeadm upgrade apply v1.14.x
```

4. Upgrade kubectl and kubelet

```
# replace x in 1.14.x-00 with the latest patch version
```

```
apt-mark unhold kubelet kubectl && \
apt-get update && apt-get install -y kubelet=1.14.x-00 kubectl=1.14.x-00 &&
\
apt-mark hold kubelet kubectl
```

```
sudo systemctl restart kubelet
```

Upgrade worker Nodes

https://kubernetes.io/docs/tasks/administer-cluster/kubeadm/kubeadm-upgrade-1-14/#upgrade-worker-nodes

K8sHardWay

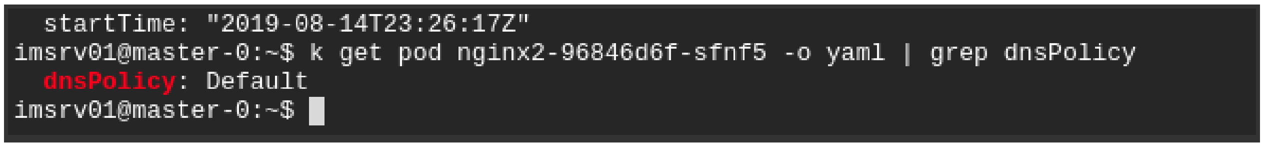
Worker

Kubelet

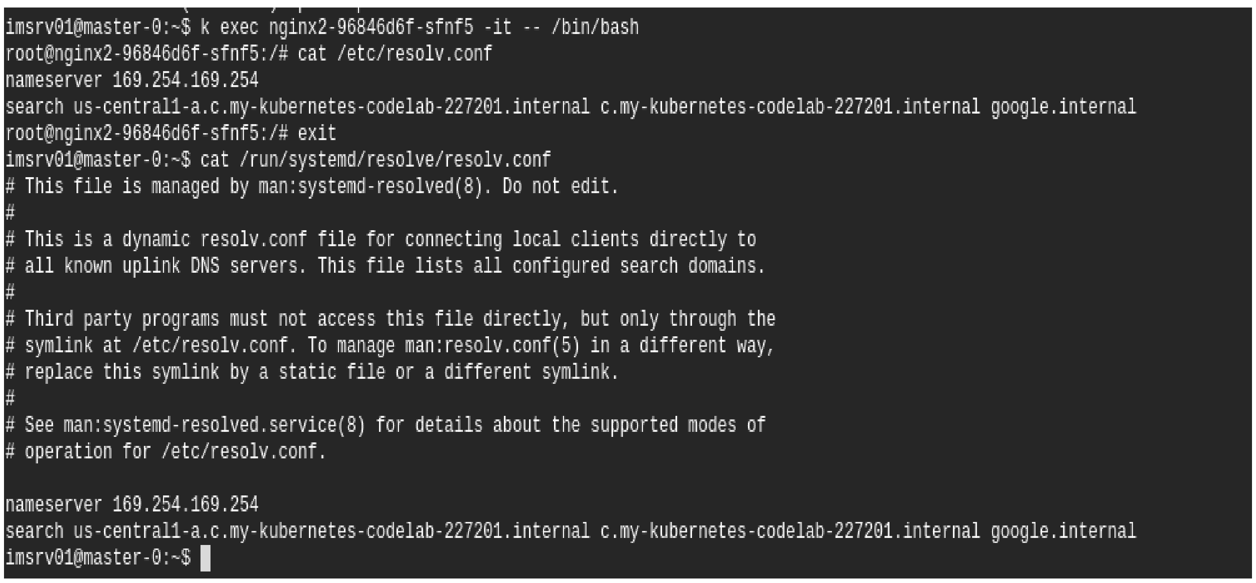
```
[Service]
ExecStart=/usr/local/bin/kubelet \\
  --config=/var/lib/kubelet/kubelet-config.yaml \\
  --container-runtime=remote \\
// use container runtime other than docker..
  --container-runtime-endpoint=unix:///var/run/containerd/containerd.sock \\
// use this container runtime
  --image-pull-progress-deadline=2m \\
  --kubeconfig=/var/lib/kubelet/kubeconfig \\
  --network-plugin=cni \\
  --register-node=true \\
  --v=2
Restart=on-failure
RestartSec=5
```

```
cat <<EOF | sudo tee /var/lib/kubelet/kubelet-config.yaml
kind: KubeletConfiguration
apiVersion: kubelet.config.k8s.io/v1beta1
authentication:
  anonymous:
    enabled: false
  webhook:
    enabled: true
  x509:
    clientCAFile: "/var/lib/kubernetes/ca.pem"
authorization:
  mode: Webhook
clusterDomain: "cluster.local"
// kubelet inserts this local domain value in pod/container - /etc/resolv.conf
clusterDNS:
  - "10.32.0.10"
// kubelet inserts this DNS server value in pod/container - /etc/resolv.conf
podCIDR: "${POD_CIDR}"
resolvConf: "/run/systemd/resolve/resolv.conf"
// POD dnsPolicy set to default - POD will inherit DNS resolution from node using above
//conf file from the node..(Default - /etc/resolv.conf)
//See screenshot below
runtimeRequestTimeout: "15m"
tlsCertFile: "/var/lib/kubelet/${HOSTNAME}.pem"
tlsPrivateKeyFile: "/var/lib/kubelet/${HOSTNAME}-key.pem"
EOF
```

Pod inheriting DNS configuration from node
POD - dnsPolicy: Default
Kubelet start - `resolvConf: "/run/systemd/resolve/resolv.conf"`



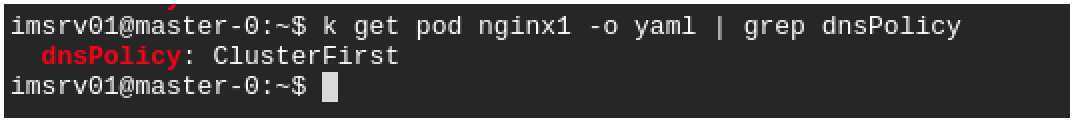
```
  startTime: "2019-08-14T23:26:17Z"
imsrv01@master-0:~$ k get pod nginx2-96846d6f-sfnf5 -o yaml | grep dnsPolicy
  dnsPolicy: Default
imsrv01@master-0:~$
```



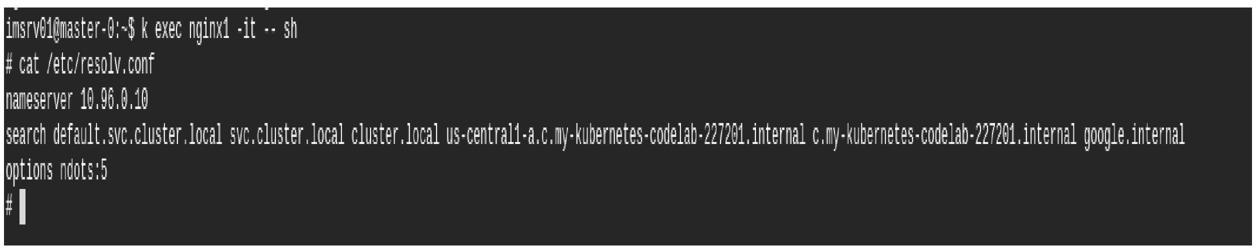
```
imsrv01@master-0:~$ k exec nginx2-96846d6f-sfnf5 -it -- /bin/bash
root@nginx2-96846d6f-sfnf5:/# cat /etc/resolv.conf
nameserver 169.254.169.254
search us-central1-a.c.my-kubernetes-codelab-227201.internal c.my-kubernetes-codelab-227201.internal google.internal
root@nginx2-96846d6f-sfnf5:/# exit
imsrv01@master-0:~$ cat /run/systemd/resolve/resolv.conf
# This file is managed by man:systemd-resolved(8). Do not edit.
#
# This is a dynamic resolv.conf file for connecting local clients directly to
# all known uplink DNS servers. This file lists all configured search domains.
#
# Third party programs must not access this file directly, but only through the
# symlink at /etc/resolv.conf. To manage man:resolv.conf(5) in a different way,
# replace this symlink by a static file or a different symlink.
#
# See man:systemd-resolved.service(8) for details about the supported modes of
# operation for /etc/resolv.conf.

nameserver 169.254.169.254
search us-central1-a.c.my-kubernetes-codelab-227201.internal c.my-kubernetes-codelab-227201.internal google.internal
imsrv01@master-0:~$
```

Pod with dnsPolicy set to clusterFirst

- **Resolv.conf file created by kubelet (not inherited) using the values passed to it at startup**



```
imsrv01@master-0:~$ k get pod nginx1 -o yaml | grep dnsPolicy
  dnsPolicy: ClusterFirst
imsrv01@master-0:~$
```

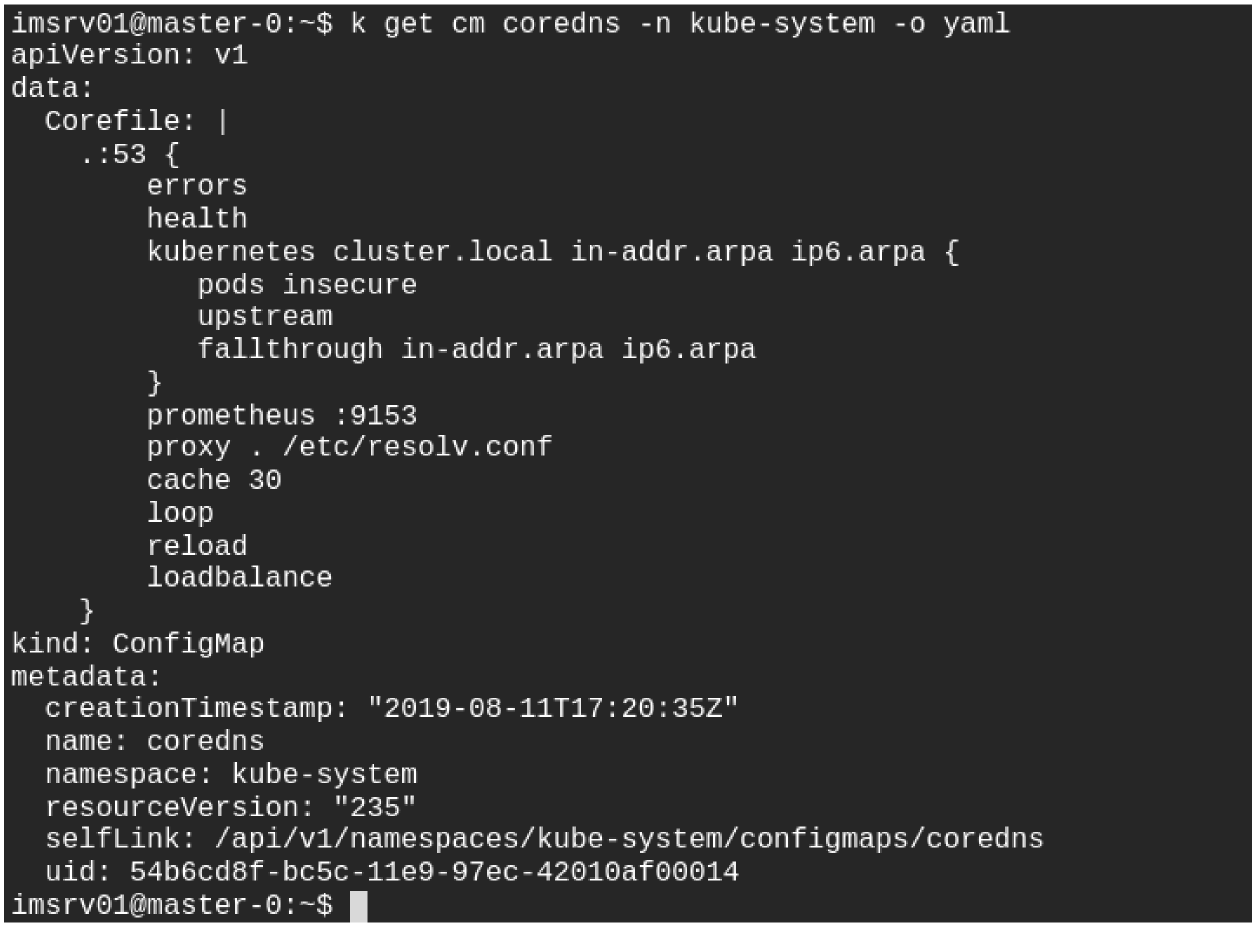


```
imsrv01@master-0:~$ k exec nginx1 -it -- sh
# cat /etc/resolv.conf
nameserver 10.96.0.10
search default.svc.cluster.local svc.cluster.local cluster.local us-central1-a.c.my-kubernetes-codelab-227201.internal c.my-kubernetes-codelab-227201.internal google.internal
options ndots:5
#
```

coreDNS config map

Has associated CoreFile:
https://coredns.io/2017/07/23/corefile-explained/

Corefile defines plugins
https://coredns.io/plugins/



```
imsrv01@master-0:~$ k get cm coredns -n kube-system -o yaml
apiVersion: v1
data:
  Corefile: |
    .:53 {
        errors
        health
        kubernetes cluster.local in-addr.arpa ip6.arpa {
           pods insecure
           upstream
           fallthrough in-addr.arpa ip6.arpa
        }
        prometheus :9153
        proxy . /etc/resolv.conf
        cache 30
        loop
        reload
        loadbalance
    }
kind: ConfigMap
metadata:
  creationTimestamp: "2019-08-11T17:20:35Z"
  name: coredns
  namespace: kube-system
  resourceVersion: "235"
  selfLink: /api/v1/namespaces/kube-system/configmaps/coredns
  uid: 54b6cd8f-bc5c-11e9-97ec-42010af00014
imsrv01@master-0:~$
```