1. Remove any old cluster information

```
devops@master:~$ sudo kubeadm reset
[sudo] password for devops:
[reset] Reading configuration from the cluster...
[reset] FYI: You can look at this config file with 'kubectl -n kube-system
get cm kubeadm-config -oyaml'
W0505 15:12:11.872744    2967 reset.go:73] [reset] Unable to fetch the
kubeadm-config ConfigMap from cluster: failed to get config map: Get
https://192.168.56.120:6443/api/v1/namespaces/kube-system/configmaps/kubeadm-
config: dial tcp 192.168.56.120:6443: connect: connection refused
[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
W0505 15:12:13.876331    2967 reset.go:234] [reset] No kubeadm config, using
etcd pod spec to get data directory
[reset] Stopping the kubelet service
[reset] unmounting mounted directories in "/var/lib/kubelet"
[reset] Deleting contents of stateful directories: [/var/lib/etcd
/var/lib/kubelet /etc/cni/net.d /var/lib/dockershim /var/run/kubernetes]
[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]

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.
```

---

2. Setup master node for the cluster (Ubuntu 16.04)

```
devops@master:~$ sudo swapoff -a

devops@master:~$ sudo kubeadm init --apiserver-advertise-address
192.168.56.120 --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.14.1
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your
internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm
config images pull'
[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] Activating the kubelet service
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[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 localhost]
and IPs [192.168.56.120 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [master localhost] and
IPs [192.168.56.120 127.0.0.1 ::1]
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [master kubernetes
kubernetes.default kubernetes.default.svc
kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 192.168.56.120]
[certs] Generating "apiserver-kubelet-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"
[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 27.010717 seconds
[upload-config] storing the configuration used in ConfigMap "kubeadm-config"
in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config-1.14" in namespace kube-system
with the configuration for the kubelets in the cluster
[upload-certs] Skipping phase. Please see --experimental-upload-certs
[mark-control-plane] Marking the node master as control-plane by adding the
label "node-role.kubernetes.io/master=''"
[mark-control-plane] Marking the node master as control-plane by adding the
taints [node-role.kubernetes.io/master:NoSchedule]
[bootstrap-token] Using token: xt2cl2.nfwxxyydonngy81i
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC
Roles
[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
[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 192.168.56.120:6443 --token xt2cl2.nfwxxyydonngy81i \
    --discovery-token-ca-cert-hash
sha256:79721de3f8e730b58edc72111ffffbf134ba055fe6fd5827e5b8b6baf5837119
```

---

3. Setup configuration

```
devops@master:~$ mkdir -p $HOME/.kube
devops@master:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
cp: overwrite '/home/devops/.kube/config'? y
devops@master:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

---

4. Installing a pod network add-on

```
devops@master:~$ kubectl apply -f
https://docs.projectcalico.org/v3.3/getting-
started/kubernetes/installation/hosted/rbac-kdd.yaml
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
devops@master:~$ kubectl apply -f
https://docs.projectcalico.org/v3.3/getting-
started/kubernetes/installation/hosted/kubernetes-datastore/calico-
networking/1.7/calico.yaml
configmap/calico-config created
service/calico-typha created
deployment.apps/calico-typha created
poddisruptionbudget.policy/calico-typha created
daemonset.extensions/calico-node created
serviceaccount/calico-node created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.project
calico.org created
```

```
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org
created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectca
lico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org
created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico
.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.project
calico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.proje
ctcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectca
lico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcali
co.org created
```

---

5. list nodes in the cluster

```
devops@master:~$ kubectl get nodes
NAME      STATUS   ROLES    AGE     VERSION
master    Ready    master   2m10s   v1.14.1
```

---

6. Adding worker node (Ubuntu 16.04)

```
devops@worker02:~$ sudo swapoff -a

Note: token shown in following command is for example.  Always use the
"kubeadm join" command with token generated while initializing the master

devops@worker02:~$ sudo kubeadm join 192.168.56.120:6443 --token
xt2cl2.nfwxxyydonngy81i \
>     --discovery-token-ca-cert-hash
sha256:79721de3f8e730b58edc72111ffffbf134ba055fe6fd5827e5b8b6baf5837119
[preflight] Running pre-flight checks
error execution phase preflight: [preflight] Some fatal errors occurred:
        [ERROR FileAvailable--etc-kubernetes-kubelet.conf]:
/etc/kubernetes/kubelet.conf already exists
        [ERROR FileAvailable--etc-kubernetes-bootstrap-kubelet.conf]:
/etc/kubernetes/bootstrap-kubelet.conf already exists
        [ERROR Port-10250]: Port 10250 is in use
        [ERROR FileAvailable--etc-kubernetes-pki-ca.crt]:
/etc/kubernetes/pki/ca.crt already exists
[preflight] If you know what you are doing, you can make a check non-fatal
with `--ignore-preflight-errors=...`
```

---

7. If there is previous cluster information remove them before joining new cluster

```
devops@worker02:~$ 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
W0505 15:26:32.922021    3740 reset.go:234] [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 stateful directories: [/var/lib/kubelet
/etc/cni/net.d /var/lib/dockershim /var/run/kubernetes]
[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]

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.
```

---

8. Join the new cluster using the token provided while initializing the master node

```
devops@worker02:~$ sudo kubeadm join 192.168.56.120:6443 --token
xt2cl2.nfwxxyydonngy81i \
>     --discovery-token-ca-cert-hash
sha256:79721de3f8e730b58edc72111ffffbf134ba055fe6fd5827e5b8b6baf5837119
[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-1.14" 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] Activating the kubelet service
[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.
```

9. List nodes on Master node:

```
devops@master:~$ kubectl get nodes
NAME       STATUS     ROLES    AGE    VERSION
master     Ready      master   5m1s   v1.14.1
worker02   NotReady   <none>   11s    v1.14.1

devops@master:~$ kubectl get nodes
NAME       STATUS   ROLES    AGE     VERSION
master     Ready    master   5m28s   v1.14.1
worker02   Ready    <none>   38s     v1.14.1
```

10. Join the new cluster using the token provided while initializing the master node (CentOS 7)

A fix for iptables in CentOS:

```
[root@worker01 ~]# echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
[root@worker01 ~]# cat /proc/sys/net/bridge/bridge-nf-call-iptables
1
```

Disable memory swap – another requirement

```
[devops@worker01 ~]$ sudo swapoff -a
[sudo] password for devops:
```

Make this node worker01 to join the cluster.

```
[devops@worker01 ~]$ sudo kubeadm join 192.168.56.120:6443 --token
ubrhkd.vvs4vg0zzmjzypyd     --discovery-token-ca-cert-hash
sha256:cd8f1f150c035f61184d46ea811468e7a2def48abf3f5dc4ec92198310be5c4a
[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-1.14" 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] Activating the kubelet service
[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.
```

```
      devops@master:~$ kubectl get nodes
      NAME        STATUS    ROLES     AGE     VERSION
      master      Ready     master    10m     v1.14.1
      worker01    Ready     <none>    113s    v1.14.1
      worker02    Ready     <none>    8m11s   v1.14.1
```

11. list token on Master node:

```
devops@master:~$ sudo kubeadm token list
TOKEN                      TTL       EXPIRES                     USAGES
DESCRIPTION                                                 EXTRA GROUPS
xt2cl2.nfwxxyydonngy81i   23h       2019-05-06T15:22:08+05:30
authentication,signing   The default bootstrap token generated by 'kubeadm
init'.   system:bootstrappers:kubeadm:default-node-token
```

---

12. configuration information

```
devops@master:~$ sudo kubeadm config print init-defaults
apiVersion: kubeadm.k8s.io/v1beta1
bootstrapTokens:
- groups:
  - system:bootstrappers:kubeadm:default-node-token
  token: abcdef.0123456789abcdef
  ttl: 24h0m0s
  usages:
  - signing
  - authentication
kind: InitConfiguration
localAPIEndpoint:
  advertiseAddress: 1.2.3.4
  bindPort: 6443
nodeRegistration:
  criSocket: /var/run/dockershim.sock
  name: master
  taints:
  - effect: NoSchedule
    key: node-role.kubernetes.io/master
---
apiServer:
  timeoutForControlPlane: 4m0s
apiVersion: kubeadm.k8s.io/v1beta1
certificatesDir: /etc/kubernetes/pki
clusterName: kubernetes
controlPlaneEndpoint: ""
```

```
controllerManager: {}
dns:
  type: CoreDNS
etcd:
  local:
    dataDir: /var/lib/etcd
imageRepository: k8s.gcr.io
kind: ClusterConfiguration
kubernetesVersion: v1.14.0
networking:
  dnsDomain: cluster.local
  podSubnet: ""
  serviceSubnet: 10.96.0.0/12
scheduler: {}
```

```
devops@master:~$ sudo kubeadm config print join-defaults
apiVersion: kubeadm.k8s.io/v1beta1
caCertPath: /etc/kubernetes/pki/ca.crt
discovery:
  bootstrapToken:
    apiServerEndpoint: kube-apiserver:6443
    token: abcdef.0123456789abcdef
    unsafeSkipCAVerification: true
  timeout: 5m0s
  tlsBootstrapToken: abcdef.0123456789abcdef
kind: JoinConfiguration
nodeRegistration:
  criSocket: /var/run/dockershim.sock
  name: master
```

---

13. Labelling the role in node list

```
devops@master:~$ kubectl get nodes
NAME        STATUS    ROLES     AGE    VERSION
master      Ready     master    45m    v1.14.1
worker02    Ready     <none>    40m    v1.14.1

devops@master:~$ kubectl label node worker02 node-
role.kubernetes.io/worker=worker
node/worker02 labeled

devops@master:~$ kubectl get nodes
NAME        STATUS    ROLES     AGE    VERSION
master      Ready     master    46m    v1.14.1
worker02    Ready     worker    41m    v1.14.1
```
