kubeadm Bootstrap Kubernetes

START HERE

`kubeadm init`
Create a control plane

`kubeadm init --pod-network-cidr=<cidr>`
Init, sized for a CNI

`kubeadm join <host>:6443 --token <token>`
Add a node

`kubeadm token create --print-join-command`
Get a fresh join command

`kubeadm certs check-expiration`
When do certs expire?

`kubeadm upgrade plan`
What can I upgrade to?

`kubeadm upgrade apply v<version>`
Upgrade first control plane

`kubeadm reset`
Undo init or join

Install runtime, kubelet, kubeadm → `kubeadm init` → **Control plane** static pods, admin.conf → `apply a CNI` → **Network** nodes become Ready → `kubeadm join` → **Workers** kubelet joins

CONCEPTS

kubeadm only bootstraps: it needs a container runtime and kubelet already installed, and you install a CNI plugin yourself afterwards.

Run kubeadm as root. Swap must be off and the node needs a unique hostname and reachable port 6443.

Upgrade one minor version at a time; never skip a minor.

BOOTSTRAP A CLUSTER

`kubeadm init`
Initialise a control-plane node

`kubeadm init --pod-network-cidr=<cidr>`
Reserve the pod IP range for a CNI

`kubeadm init --apiserver-advertise-address=<ip>`
Choose the API server IP

`kubeadm init --control-plane-endpoint=<dns>:6443`
Stable endpoint for HA control plane

`kubeadm init --upload-certs`
Share certs for more control planes

`kubeadm init --config <file>`
Init from a config file

`kubeadm init --dry-run`
Show what init would do

`kubeadm init --kubernetes-version <version>`
Pin the control-plane version

`kubeadm init --cri-socket <socket>`
Pick the container runtime socket

`kubeadm init --service-cidr <cidr>`
Change the Service IP range

`kubeadm init --apiserver-cert-extra-sans <name>`
Extra names on the API cert

`kubeadm init --ignore-preflight-errors=<check>`
Downgrade a preflight error

`kubeadm init --skip-phases=<phase>`
Skip a phase, e.g. addon/kube-proxy

`kubeadm init phase <phase>`
Run a single init phase

BOOTSTRAP A CLUSTER (cont.)

`mkdir -p $HOME/.kube && sudo cp /etc/kubernetes/admin.conf $HOME/.kube/config`
Let kubectl talk to the cluster

JOIN NODES

`kubeadm join <host>:6443 --token <token> --discovery-token-ca-cert-hash sha256:<hash>`
Join a worker node

`kubeadm join <host>:6443 ... --control-plane --certificate-key <key>`
Join as a control-plane node

`kubeadm join --config <file>`
Join from a config file

`kubeadm join ... --node-name <name>`
Override the node name

`kubeadm join ... --cri-socket <socket>`
Pick the container runtime socket

`kubeadm join phase <phase>`
Run a single join phase

`kubeadm init phase upload-certs --upload-certs`
Re-upload control-plane certs

TOKENS

`kubeadm token create --print-join-command`
New token plus full join command

`kubeadm token list`
List bootstrap tokens

`kubeadm token create`
Create a token (24h by default)

`kubeadm token create --ttl 0`
Token that never expires

`kubeadm token delete <token>`
Revoke a token

`kubeadm token generate`
Print a random token locally

`kubeadm certs certificate-key`
Generate a certificate key

`openssl x509 -pubkey -in /etc/kubernetes/pki/ca.crt | openssl rsa -pubin -outform der 2>/dev/null | openssl dgst -sha256 -hex | sed 's/^.* //'`
Compute the CA cert hash

CERTIFICATES

`kubeadm certs check-expiration`
Show expiry of all certs

`kubeadm certs renew all`
Renew every certificate

`kubeadm certs renew <cert>`
Renew one cert, e.g. apiserver

`kubeadm kubeconfig user --client-name <user>`
Make a kubeconfig for a user

After renewing, restart the control-plane static pods and copy the new admin.conf to your kubeconfig.

UPGRADE

`kubeadm upgrade plan`
Show available versions and checks

`kubeadm upgrade apply v<version>`
Upgrade the first control plane

`kubeadm upgrade node`
Upgrade other nodes' local config

`kubeadm upgrade diff v<version>`
Preview static pod manifest changes

`kubeadm upgrade apply v<version> --dry-run`
Rehearse an upgrade

`kubectl drain <node> --ignore-daemonsets`
Evict pods before upgrading kubelet

`kubectl uncordon <node>`
Make the node schedulable again

`kubeadm version -o short`
Print the kubeadm version

CONFIG AND IMAGES

`kubeadm config print init-defaults`
Print the default init config

`kubeadm config validate --config <file>`
Check a config file

`kubeadm config migrate --old-config <file> --new-config <file>`
Convert to the newest API version

`kubeadm config images list`
List images kubeadm needs

`kubeadm config images pull`
Pre-pull the control-plane images

CONFIG AND IMAGES (cont.)

`kubectl -n kube-system get cm kubeadm-config -o yaml`
Read the stored cluster config

RESET

`kubeadm reset`
Revert init or join on this node

`kubeadm reset --force`
Reset without the prompt

`sudo rm -rf /etc/cni/net.d`
Remove leftover CNI config

`rm -rf $HOME/.kube`
Remove the stale kubeconfig

reset leaves CNI config, iptables rules and $HOME/.kube behind; clean them by hand before re-initialising.

RECIPES

Single-node cluster

```
sudo kubeadm init --pod-network-cidr=<cidr>
mkdir -p $HOME/.kube
sudo cp /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
# install a CNI plugin, then let pods run on the control plane
kubectl taint nodes --all node-role.kubernetes.io/control-plane-
```

Add a worker later

```
# on the control plane
kubeadm token create --print-join-command
# on the new node, as root
kubeadm join <host>:6443 --token <token> \
  --discovery-token-ca-cert-hash sha256:<hash>
```

Upgrade a control plane node

```
# upgrade the kubeadm package first, then
sudo kubeadm upgrade plan
sudo kubeadm upgrade apply v<version>
kubectl drain <node> --ignore-daemonsets
# upgrade kubelet and kubectl packages
sudo systemctl daemon-reload &&
sudo systemctl restart kubelet
kubectl uncordon <node>
```

Check and renew certs

```
sudo kubeadm certs check-expiration
sudo kubeadm certs renew all
# restart kube-apiserver, controller-manager, scheduler, etcd static pods
sudo cp /etc/kubernetes/admin.conf $HOME/.kube/config
```