K3s Lightweight Kubernetes

START HERE

`curl -sfL https://get.k3s.io | sh -`
Install a server

`sudo k3s kubectl get nodes`
Check the node

`sudo cat /var/lib/rancher/k3s/server/node-token`
Get the join token

`K3S_URL=https://<server>:6443 K3S_TOKEN=<token>`
Join env for an agent

`systemctl status k3s`
Service state

`journalctl -u k3s -f`
Follow the logs

`k3s etcd-snapshot save`
Back up the cluster

`k3s-uninstall.sh`
Remove K3s

CONCEPTS

K3s is one small binary: the install script drops it in /usr/local/bin, adds kubectl, crictl and ctr links, and sets up a systemd service.

Server control plane + datastore
↓ node-token
Agent K3S_URL + K3S_TOKEN
↓ registers
Cluster kubectl, manifests, snapshots
Install, join, operate

INSTALL

`curl -sfL https://get.k3s.io | sh -`
Install a single-node server

`curl -sfL https://get.k3s.io | K3S_URL=https://<server>:6443 K3S_TOKEN=<token> sh -`
Install as agent and join
Setting K3S_URL makes the installer choose agent mode.

`curl -sfL https://get.k3s.io | INSTALL_K3S_VERSION=<version> sh -`
Pin a version

`INSTALL_K3S_CHANNEL=latest`
Pick channel: stable, latest, testing

`INSTALL_K3S_EXEC="server --cluster-init"`
Set the command and flags

`INSTALL_K3S_SKIP_START=true`
Install without starting

`K3S_NODE_NAME=<name>`
Unique node name
Each machine needs a unique hostname.

`K3S_KUBECONFIG_MODE=644`
Make the kubeconfig readable

Install variables are saved into the service. Re-running the installer without them loses the old values.

SERVER AND AGENT

`k3s server`
Run a server node

`k3s agent`
Run a worker node

`--cluster-init`
Start an embedded-etcd cluster

`--server https://<server>:6443`
Join an existing cluster

SERVER AND AGENT (cont.)

`--token <token>`
Shared join secret

`--tls-san <host-or-ip>`
Add a name to the API cert

`--disable traefik`
Skip a packaged component
Also coredns, servicelb, local-storage, metrics-server.

`--node-taint <key>=<value>:<effect>`
Taint the node at start

`--flannel-backend none`
Set or turn off Flannel
Must match on all servers.

`--secrets-encryption`
Encrypt Secrets at rest

`--write-kubeconfig-mode 644`
Kubeconfig file mode

CONFIG FILE

`/etc/rancher/k3s/config.yaml`
Main config file

`/etc/rancher/k3s/config.yaml.d/*.yaml`
Drop-ins, read alphabetically

config.yaml

```
# flag names without the dashes
write-kubeconfig-mode: "0644"
tls-san:
  - k3s.example.com
disable:
  - traefik
node-taint+:
  - "dedicated=infra:NoSchedule"
```

CLI flags override the file. A key ending in + appends to the earlier value instead of replacing it.

`/etc/rancher/k3s/registries.yaml`
Mirrors and registry auth

`/var/lib/rancher/k3s/server/manifests/`
YAML here is auto-applied

`/var/lib/rancher/k3s/agent/images/`
Air-gap image tarballs

CLUSTER AND NODES

`/etc/rancher/k3s/k3s.yaml`
Server kubeconfig

`export KUBECONFIG=/etc/rancher/k3s/k3s.yaml`
Use it with plain kubectl

`k3s kubectl get nodes`
Embedded kubectl

CLUSTER AND NODES (cont.)

`k3s kubectl get pods -A`
All pods

`/var/lib/rancher/k3s/server/node-token`
Token to join agents

`/var/lib/rancher/k3s/server/token`
Server token (join servers)

`k3s crictl ps`
Containers via the CRI

`k3s crictl images`
Images on the node

`k3s check-config`
Check kernel and system setup

`k3s --version`
Show the version

Copy k3s.yaml to your laptop and change the server address from 127.0.0.1 to the server's name.

SERVICE

`systemctl status k3s`
Server service state

`systemctl status k3s-agent`
Agent service state

`sudo systemctl restart k3s`
Restart after config changes

`journalctl -u k3s -f`
Follow server logs

`journalctl -u k3s-agent -f`
Follow agent logs

`/usr/local/bin/k3s-killall.sh`
Stop all K3s processes

`/usr/local/bin/k3s-uninstall.sh`
Remove a server

`/usr/local/bin/k3s-agent-uninstall.sh`
Remove an agent

TOKENS AND CERTIFICATES

`k3s token create`
Create a bootstrap token

`k3s token list`
List bootstrap tokens

`k3s token rotate --new-token <token>`
Replace the server token
Restart all servers and agents afterwards. Old snapshots need the old token.

`k3s certificate check`
Show cert expiry

TOKENS AND CERTIFICATES (cont.)

`k3s certificate rotate`
Rotate certs (stop K3s first)

`k3s certificate rotate --service <name>,<name>`
Rotate only some certs

ETCD AND BACKUPS

`k3s etcd-snapshot save`
Snapshot now

`k3s etcd-snapshot save --name <name>`
Named snapshot

`k3s etcd-snapshot ls`
List snapshots

`k3s etcd-snapshot delete <name>`
Delete a snapshot

`k3s etcd-snapshot prune --snapshot-retention <n>`
Keep only n on-demand snapshots

`--etcd-snapshot-schedule-cron "0 */6 * * *"`
Change the schedule (default 12 h)

`k3s server --cluster-reset --cluster-reset-restore-path=<snapshot>`
Restore a snapshot
Stop K3s first. On a new host also pass the old --token.

`/var/lib/rancher/k3s/server/db/snapshots/`
Default snapshot directory

RECIPES

Add a worker

```
# on the server
sudo cat /var/lib/rancher/k3s/server/node-token
# on the worker
curl -sfL https://get.k3s.io | K3S_URL=https://<server>:6443 K3S_TOKEN=<token> sh -
# back on the server
sudo k3s kubectl get nodes
```

HA with embedded etcd

```
# first server
curl -sfL https://get.k3s.io | K3S_TOKEN=<secret> sh -s - server --cluster-init
# more servers
curl -sfL https://get.k3s.io | K3S_TOKEN=<secret> sh -s - server --server https://<first>:6443
```

Restore from a snapshot

```
sudo systemctl stop k3s
sudo k3s server --cluster-reset \
  --cluster-reset-restore-path=<snapshot-path>
sudo systemctl start k3s
```