# k3d k3s in Docker

**START HERE**

**`k3d cluster create <name>`**
Make a cluster

**`k3d image import <image> -c <name>`**
Load a local image

**`k3d cluster list`**
List clusters

**`k3d cluster create -p "8080:80@loadbalancer"`**
Reach ingress from host

**`k3d cluster stop <name>`**
Pause, keep state

**`k3d kubeconfig merge <name> --kubeconfig-switch-context`**
Point kubectl at it

**`k3d cluster delete <name>`**
Remove a cluster

**`k3d cluster create --config <file>`**
Cluster from YAML

## CONCEPTS

**Docker host** your machine
**Cluster network** k3d-<name>
**Nodes** serverlb, server-N, agent-N

One Docker network per cluster

Every node is a Docker container named k3d-<cluster>-<role>-<n>. Host ports reach the cluster through the k3d-<cluster>-serverlb container, so map ports with @loadbalancer.

Node filters pick targets: server:0, agent:*, agent:0,1, loadbalancer. Repeatable flags like -p, -v, -e and --k3s-arg accept them after an @.

## CLUSTERS

**`k3d cluster create <name>`**
Create a cluster

**`k3d cluster list`**
List clusters and node counts

**`k3d cluster delete <name>`**
Delete a cluster

**`k3d cluster delete --all`**
Delete every cluster

**`k3d cluster stop <name>`**
Stop the cluster containers

**`k3d cluster start <name>`**
Start a stopped cluster

**`k3d cluster create <name> -s <n> -a <n>`**
Set server and agent count
*Several servers switch k3s to embedded etcd HA.*

**`k3d cluster create <name> -i <image>`**
Pick the k3s version via image
*Example image: rancher/k3s:<tag>.*

**`k3d cluster list -o json`**
Machine-readable list

**`k3d cluster create --no-lb`**
Skip the load balancer

**`k3d cluster create --timeout <dur>`**
Roll back if too slow

## PORTS AND NETWORKING

**`k3d cluster create -p "8080:80@loadbalancer"`**
Map host port to ingress

## PORTS AND NETWORKING (cont.)

**`k3d cluster create -p "8082:30080@agent:0"`**
Map a NodePort via an agent

**`k3d cluster create --api-port 6550`**
Fix the API server host port

**`k3d node edit k3d-<name>-serverlb --port-add 8080:80`**
Add a port to a live cluster
*Experimental; k3d cluster edit --port-add does the same.*

**`k3d cluster create --network <net>`**
Join an existing Docker network

## KUBECONFIG

**`k3d kubeconfig merge <name> --kubeconfig-switch-context`**
Merge and switch context

**`k3d kubeconfig get <name>`**
Print a cluster's kubeconfig

**`k3d kubeconfig merge <name> --kubeconfig-merge-default`**
Merge into ~/.kube/config

**`k3d kubeconfig merge <name> -o <file>`**
Write to a separate file

**`k3d cluster create --kubeconfig-update-default=false`**
Leave default kubeconfig alone

**`kubectl config use-context k3d-<name>`**
Switch context by hand

create already updates your default kubeconfig and switches to context k3d-<name>. Use merge for clusters made elsewhere.

## NODES

**`k3d node list`**
List all nodes

**`k3d node create <name> -c <cluster>`**
Add an agent to a cluster

**`k3d node create <name> --role server -c <cluster>`**
Add a server node

**`k3d node stop <node>`**
Stop one node

**`k3d node start <node>`**
Start one node

**`k3d node delete <node>`**
Remove a node

## IMAGES AND REGISTRIES

**`k3d image import <image> -c <name>`**
Copy a local image into nodes
*Default cluster name is k3s-default.*

**`k3d image import <archive.tar> -c <name>`**
Import from a tarball

**`k3d image import <image> -m direct`**
Skip the tools node

**`k3d registry create <name> --port <port>`**
Create a local registry
*The container is named k3d-<name>.*

**`k3d registry list`**
List registries

**`k3d registry delete <name>`**
Delete a registry

**`k3d cluster create --registry-create <name>`**
Make a registry with the cluster

**`k3d cluster create --registry-use k3d-<name>:<port>`**
Use an existing registry

**`k3d registry create <name> --proxy-remote-url <url>`**
Pull-through cache

## MOUNTS, ENV AND K3S ARGS

**`k3d cluster create -v <host>:<node>@<filter>`**
Mount a host path

**`k3d cluster create -e "KEY=val@server:0"`**
Set a node env var

**`k3d cluster create --k3s-arg "<arg>@server:*"`**
Pass a flag to k3s

**`k3d cluster create --k3s-node-label "k=v@agent:*"`**
Label k3s nodes

**`k3d cluster create --runtime-ulimit "nofile=1024:1024"`**
Set a ulimit

## CONFIG FILE

**`k3d cluster create --config <file>`**
Create from a config file

**`k3d cluster create <name> -c <file>`**
Flags override the file

## CONFIG FILE (cont.)

**`k3d cluster delete -c <file>`**
Delete the file's cluster

**`k3d config init`**
Print a default config
*Check k3d config init --help for output flags.*

### Common config keys

| Key | Sets |
|---|---|
| `servers / agents` | Node counts |
| `image` | k3s image |
| `ports[].port + nodeFilters` | Port maps |
| `volumes[].volume` | Mounts |
| `registries.create / use` | Registry setup |
| `options.k3s.extraArgs` | k3s flags |
| `options.k3d.wait` | Wait for ready |

Only apiVersion and kind are required in the file. $VAR and ${VAR} are expanded before parsing. The format is still alpha (v1alpha5).

## MAINTENANCE

**`k3d version`**
Show k3d and k3s versions

**`k3d completion <shell>`**
Generate shell completion

**`k3d <cmd> --verbose`**
Debug logging

**`docker ps -f name=k3d-`**
See the node containers

## RECIPES

### Dev cluster with ingress and registry

```
k3d registry create dev --port 5000
k3d cluster create dev -a 2 \
  -p "8080:80@loadbalancer" \
  --registry-use k3d-dev:5000
# push to localhost:5000, pull as k3d-dev:5000
```

### Local image without a registry

```
docker build -t app:dev .
k3d image import app:dev -c dev
kubectl create deployment app --image=app:dev
kubectl set image deploy/app app=app:dev
```

### Throw away and rebuild

```
k3d cluster delete dev
k3d cluster create --config dev.yaml
kubectl get nodes
```