

KUBERNETES FROM ZERO · FOUNDATIONS 3 OF 12 · LESSON 3 · LESSON KIT · CHEAT SHEET

Kubernetes from Zero, part 3: your local lab with kind, kubectl and kubeconfig

The four lab tools, how kubeconfig picks a cluster, and the commands from Lesson 3.

Installed and checksummed kubectl v1.37.2 and kind v0.33.0; ran 1- and 3-node clusters, 10 Nov 2026. Cluster: first and lab.

THE ONE RULE

Use kind for a disposable, multi-node lab, and always know your current context before you type a command.

Four ways to run a learning cluster

every one of them gives you a real API server

TOOL	HOW THE LESSON DESCRIBES IT	THIS SERIES
kind	Each node is a container on Docker, Podman or nerdctl, bootstrapped with kubeadm. Several nodes come from a small config file. The version is the node image.	Every lab in this series.
minikube	A cluster in a VM, a container, or on the host, depending on the driver. It asks for 2 CPUs, 2 GB of memory and 20 GB of disk, and it has an add-on catalogue.	A good fit for a VM lab or its add-ons.
k3d	Runs k3s in Docker. k3s is a lightweight, fully compliant Kubernetes in a single binary under 100 MB. k3d is a community project, not an official Rancher product.	Part 53 covers k3s.
Docker Desktop	A built-in cluster: kubeadm (one node, version set by Docker Desktop) or kind (several nodes, version of your choice). Check the licence. Personal, education, non-commercial open source and small businesses are free.	Professional use in larger organizations, and government entities, need a paid subscription.

kubeconfig

clusters, users, contexts, and one pointer

FIELD	WHAT IT HOLDS	HOW KUBECTL USES IT
clusters	A name, the API server address (`server`), and the certificate authority used to trust it.	kind fills this in for you.
users	A name and the credentials. For kind, a client certificate and key.	In clouds this is usually a token plugin.
contexts	A name that joins one cluster, one user, and an optional default namespace.	`kubectl config get-contexts`
current-context	The context kubectl uses when you do not pass `--context`.	The star in `get-contexts`.

Lookup order: the `--kubeconfig` flag if you pass one, otherwise the files in `KUBECONFIG` (colon-separated on Linux and macOS, semicolon on Windows), otherwise `$HOME/.kube/config`. When files are merged, the first file to set a value wins. A kubeconfig is a credential: `chmod 600`, and read an unknown file the way you would read a shell script.

What the lab showed

ports, IDs and timings differ on your machine

1	**Checksums**	`kubectl: OK` and `kind: OK` mean the binary matches the published checksum. The lesson's client is kubectl v1.37.2 and kind v0.33.0. If you see `FAILED`, download both files again.
2	**kind-first**	`kind create cluster --name first` sets the context to `kind-first`. `NotReady` right after creation is normal. A few seconds later the node is `Ready`. kind prefixes the cluster name with `kind-`.
3	**kind-lab**	`kind-lab.yaml` asks for one control-plane and two workers. The lesson's nodes are `lab-control-plane`, `lab-worker` and `lab-worker2`, Kubernetes v1.37.0. Only the control-plane containers publish port 6443.

SCAN FOR THE LESSON

Lesson: **Kubernetes from Zero, part 3: your local lab with kind, kubectl and kubeconfig**

https://aitamer.news/posts/k8s-from-zero-03-your-local-lab/?utm_source=lessonkit&utm_medium=pdf&utm_campaign=k8s-03&utm_content=cheatsheet

NEXT IN THE SERIES **Next: Lesson 4, your first Pod.** Write a Pod manifest by hand and inspect it with get, describe, logs, exec and port-forward. Lesson page

Free to share with credit: © aitamer.news

4 **Contexts** `use-context` changes the kubeconfig. `--context` targets one command. `--minify` shows only the current context and hides certificate data unless you add `--raw`. `set-context --current --namespace` sets the default namespace for commands that omit `-n`.

With `KUBECONFIG` listing several files, the first file's `current-context` wins. `kind delete cluster` also removes that cluster's context. Deleting a cluster that does not exist is not an error. After that cleanup, step 10 creates `k8s-from-zero` for part 4.

Key commands

commands verbatim from the lesson

Contexts

```
kubectl config get-contexts
```
`*` marks the current one

```
kubectl config current-context
```
run it before every change

```
kubectl config use-context kind-lab
```
changes the kubeconfig file

```
kubectl get nodes --context kind-lab
```
one command, other cluster

```
kubectl config set-context --current --namespace=kube-system
```
applies to commands without `-n`

```
kubectl config view --minify
```
`--raw` would print the keys

```
kind get kubeconfig --name lab > lab.kubeconfig
```
then `chmod 600`

```
KUBECONFIG=a.kubeconfig:b.kubeconfig kubectl config view \
  --flatten
```
first file wins on conflicts

When a multi-node create fails

```
sudo sysctl fs.inotify.max_user_instances=512
```
Ubuntu's default is 128

```
sudo sysctl fs.inotify.max_user_watches=524288
```
Ubuntu's default is 8192

Check and install

```
curl -LO "https://dl.k8s.io/release/v1.37.2/bin/linux/amd64/kubectl"
```
pins kubectl v1.37.2, the client the lab printed

```
curl -LO "https://dl.k8s.io/release/v1.37.2/bin/linux/amd64/kubectl.sha256"
```
the checksum file for that binary

```
echo "$(cat kubectl.sha256)  kubectl" | sha256sum --check
```
must print `OK`

```
curl -L -s https://dl.k8s.io/release/stable.txt
```
prints the latest stable version. On 10 Nov 2026 it printed v1.37.2

```
[ "$(uname -s)-$(uname -m)" = Linux-x86_64 ] && KIND_FILE=kind-linux-amd64
```
Linux x86-64

```
[ "$(uname -s)-$(uname -m)" = Linux-aarch64 ] && KIND_FILE=kind-linux-arm64
```
Linux arm64

```
[ "$(uname -s)-$(uname -m)" = Darwin-arm64 ] && KIND_FILE=kind-darwin-arm64
```
Apple silicon

```
curl -Lo ./kind "https://kind.sigs.k8s.io/dl/v0.33.0/${KIND_FILE}"
```
the binary named above

```
curl -Lo kind.sha256sum "https://github.com/kubernetes-sigs/kind/releases/download/v0.33.0/${KIND_FILE}.sha256sum"
```
the checksum file that matches that binary

```
echo "$(cut -d' ' -f1 kind.sha256sum)  kind" | sha256sum --check
```
must print `kind: OK` (Linux; on a Mac see the lesson)

```
kubectl version --client
```
client only. `kubectl version` also asks the server

```
kind version
```
the lesson prints kind v0.33.0



SCAN FOR THE LESSON

Lesson: **Kubernetes from Zero, part 3: your local lab with kind, kubectl and kubeconfig**

https://aitamer.news/posts/k8s-from-zero-03-your-local-lab/?utm_source=lessonkit&utm_medium=pdf&utm_campaign=k8s-03&utm_content=cheatsheet

NEXT IN THE SERIES **Next: Lesson 4, your first Pod**. Write a Pod manifest by hand and inspect it with get, describe, logs, exec and port-forward. Lesson page

Free to share with credit: © aitamer.news

Clusters

`kind create cluster --name first`
context is `kind-first`

`kind create cluster --config kind-lab.yaml`
roles: control-plane, worker

```
kind create cluster --name pinned --image \
  kindest/node:v1.37.0@sha256:a1ed56cfb0e7b93589bdf97c8cd566405a265939e3620fc4f5de89adff580ae5
```
digests are in the kind release notes

`kind get clusters`
names of your clusters

`kind get nodes --name lab`
nodes are Docker containers

`kind delete cluster --name lab`
also removes its context

`kind create cluster --name k8s-from-zero`
after the cleanup. Part 4 needs this cluster

`docker ps --format 'table {{.Names}}\t{{.Image}}\t{{.Ports}}'`
only control planes publish 6443

`docker exec lab-worker crictl ps`
containers as the runtime sees them

SCAN FOR THE LESSON

Lesson: **Kubernetes from Zero, part 3: your local lab with kind, kubectl and kubeconfig**

https://aitamer.news/posts/k8s-from-zero-03-your-local-lab/?utm_source=lessonkit&utm_medium=pdf&utm_campaign=k8s-03&utm_content=cheatsheet

NEXT IN THE SERIES **Next: Lesson 4, your first Pod**. Write a Pod manifest by hand and inspect it with get, describe, logs, exec and port-forward. Lesson page

Free to share with credit: © aitamer.news