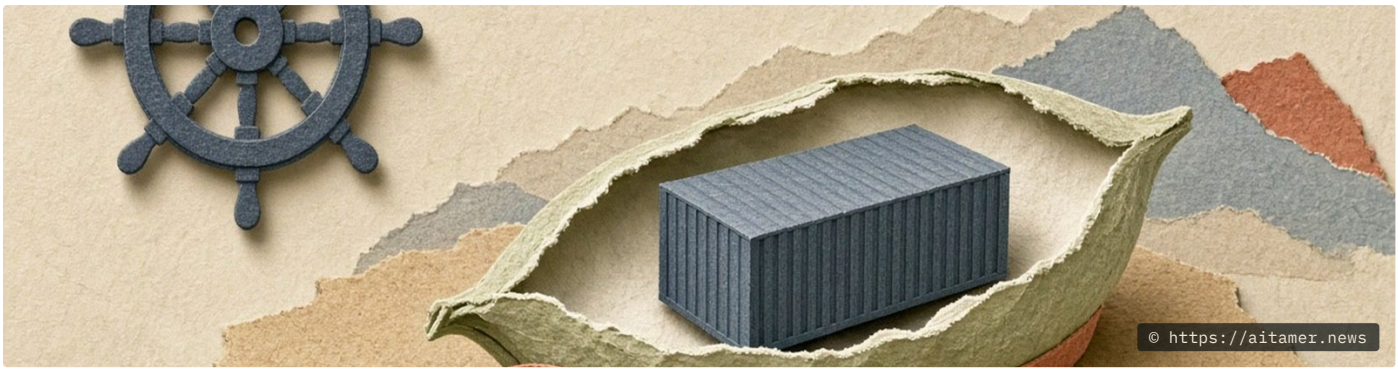


KUBERNETES FROM ZERO · FOUNDATIONS 4 OF 12 · LESSON 4 · LESSON KIT · CHEAT SHEET

Kubernetes from Zero, part 4: Your first Pod

The Pod, its phase, and the commands from Lesson 4.

Every command ran on a kind v0.33.0 cluster (node v1.37.0, kubectl v1.37.2) on 10 Nov 2026. Cluster: k8s-ep04.

THE ONE RULE

A Pod is the smallest thing Kubernetes runs: one IP around one or more containers. Learn get, describe, logs and exec on it, and treat it as disposable.

The manifest four fields you write, and one the cluster writes

FIELD	WHAT THE LESSON SAYS	IN POD.YAML
apiVersion	Which version of the Kubernetes API this object uses. Pods are in the core group, version `v1`.	`v1`
kind	Which type of object.	`Pod`
metadata	Who the object is. At least a `name`. The lesson also sets a namespace and a label.	`hello` in `k8s-from-zero`
spec	The desired state. For a Pod, mainly the list of containers.	You write this.
status	What is true right now: the phase, the Pod IP, the conditions. You never put `status` in your file.	The cluster writes it.

One network identity: the IP belongs to the Pod, and containers in that Pod talk on `localhost`. One hostname: containers see the Pod name. One place: all containers of a Pod run on the same node. Pods are disposable. Kubernetes never moves a Pod. It replaces it with a new one that has a new UID.

Phase, container state, STATUS three different stories, from the Pod lifecycle docs as the lesson states them

WORD	WHAT IT MEANS	WHAT THE LAB SHOWED
Pending	Accepted, but at least one container is not running yet. Waiting for a node and pulling images both count.	`typo` stayed here.
Running	Bound to a node, all containers created, at least one running or restarting.	`hello`, and `crash` the whole time.
Succeeded	All containers exited with success and will not restart.	`once`. STATUS column: `Completed`.
Failed	All containers have terminated and at least one failed.	A phase, not the STATUS word `Error`.
Unknown	The control plane cannot get the state, usually because the node is unreachable.	The fifth phase.
Container state	`Waiting`, `Running` or `Terminated`. A waiting container has a reason, such as `ContainerCreating` or `ImagePullBackOff`. A terminated one has an exit code.	`describe` shows State and Last State.
restartPolicy	`Always` (the default), `OnFailure`, or `Never`. The kubelet waits 10 seconds, then 20, then 40, doubling up to 5 minutes, and resets that timer after 10 minutes without trouble.	`once` sets `Never`. `crash` keeps the default.

SCAN FOR THE LESSON

Lesson: **Kubernetes from Zero, part 4: Your first Pod**

https://aitamer.news/posts/k8s-from-zero-04-your-first-pod/?utm_source=lessonkit&utm_medium=pdf&utm_campaign=k8s-04&utm_content=cheatsheet

NEXT IN THE SERIES **Next: Lesson 5, kubectl essentials.** Kubernetes from Zero, part 5, kubectl essentials. Lesson page

Free to share with credit: © aitamer.news

Life of Pod hello

the events on the lesson's describe, top to bottom

1	scheduler	`Scheduled`. Successfully assigned `k8s-from-zero/hello` to the node. The lesson's node is `k8s-ep04-control-plane`.
2	kubelet	`Pulled`. On the lab the image was already present. On your machine you see `Pulling`, then `Successfully pulled image`.
3	kubelet	`Created`, then `Started`. `READY 1/1` means one of one containers is ready. `kubectl get` without `-n` looks in `default` and finds nothing.
4	you	`logs`, `exec` and `port-forward` all go through the API server. An edit inside the container is gone after you delete the Pod and apply the file again.

`containerPort` is documentation. It does not open or block anything by itself. With a tag other than `latest`, the default pull policy is `IfNotPresent`: the node pulls the image once and reuses it. Deleting a Pod waits up to the default grace period, which the lesson says is 30 seconds.

Key commands

commands verbatim from the lesson

Run the Pod

```
kubectl create namespace k8s-from-zero
```
once per cluster

```
kubectl apply -f pod.yaml
```
prints `created`, `configured` or `unchanged`

```
kubectl get pods -n k8s-from-zero
```
add `-o wide` for IP and node

```
kubectl get pods -n k8s-from-zero -w
```
Ctrl+C to stop

```
kubectl describe pod hello -n k8s-from-zero
```
read Events first

```
kubectl explain pod.spec.restartPolicy
```
works offline from the API schema

Look inside

```
kubectl logs hello -n k8s-from-zero
```
`-f` follows, `--previous` reads the last crashed run

```
kubectl exec hello -n k8s-from-zero -- nginx -v
```
`-it ... -- sh` for a shell

```
kubectl port-forward pod/hello 8080:80 -n k8s-from-zero
```
debugging only

```
curl -s http://localhost:8080 | grep title
```
the lesson prints the nginx title

```
kubectl get pod hello -n k8s-from-zero -o \
  jsonpath="{.status.phase}{\"\n\"}"
```
part 5 covers JSONPath

```
kubectl get pod hello -n k8s-from-zero -o yaml
```
includes `status`

Three small Pods

```
kubectl apply -f once.yaml -f crash.yaml -f typo.yaml
```
`once` finishes, `crash` restarts, `typo` cannot pull

```
kubectl logs once -n k8s-from-zero
```
counting, then 1 2 3

```
kubectl logs crash -n k8s-from-zero --previous
```
the run that crashed

```
kubectl get pods crash typo -n k8s-from-zero -o \
  custom-columns='NAME:.metadata.name,PHASE:.status.phase,WAITING-REASON:.status.containerStatuses[0].state.waiting.reason'
```
phase stays `Running` while STATUS says `Error`

```
kubectl delete -f once.yaml -f crash.yaml -f typo.yaml
```
waits up to the 30 s grace period

```
kubectl delete -f pod.yaml
```
keep the namespace

Free to share with credit: © aitamer.news