

KUBERNETES FROM ZERO · FOUNDATIONS 5 OF 12 · LESSON 5 · LESSON KIT · CHEAT SHEET

Kubernetes from Zero, part 5: kubectl essentials

The three ways to manage objects, the lab commands, and the selector forms from Lesson 5.

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

Keep manifests in files and use diff then apply. Use dry-run to write YAML, explain to read fields, -o to shape output, and labels with -l to pick objects.

Three ways to manage an object

the object management page, as the lesson states it

TECHNIQUE	WHAT THE LESSON SAYS	THE COMMANDS
Imperative commands	Act on live objects with flags. Fast, and good for learning and one-off tasks. They leave no file, so there is nothing to review, nothing in Git, and no record except the cluster itself.	`kubectl run`, `kubectl create deployment`, `kubectl label`
Imperative object configuration	A file, and a verb you choose. You can review the file, and you must pick the right verb. `create` fails if the object exists. `replace` overwrites the whole object with your file, dropping anything that was set elsewhere.	`kubectl create -f`, `replace -f`, `delete -f`
Declarative object configuration	You describe what you want and kubectl works out whether to create or patch each object. Pair it with `kubectl diff` to see the change first. This is how the rest of the series works.	`kubectl apply -f` on files or directories

Manage each object with only one technique. Mixing them on the same object gives results the docs call undefined. The lab shows a mild version: an edit made outside the file lasts only until the next apply.

Selectors

equality-based and set-based, from the labels page as the lesson states it

FORM	WHAT THE LESSON SAYS	WHAT THE LAB MATCHED
Equality	`=`, `==` or `!=`. A comma between requirements means AND. There is no OR operator.	`-l tier=frontend` matched shop-web-dev and shop-web-prod
AND	Both requirements must match.	`-l tier=frontend,environment=prod` matched only shop-web-prod
Inequality	The docs spell this out: inequality matches objects without the key too.	`-l tier!=frontend` also matched scratch
Set-based	`in`, `notin`, and plain existence (`key` or `!key`). `in (a,b)` is how you match either value. Use single quotes so the shell leaves the parentheses and `!` alone.	`environment in (dev,prod)` matched the three shop Pods. `!environment` matched scratch

Life of Pod web

generate, diff, apply, then watch a live edit lose

1	**kubectl run**	`--dry-run=client -o yaml` builds the object locally and sends nothing. The lesson's YAML has labels `app.kubernetes.io/name: web` and `tier: frontend`, image `nginx:1.29-alpine`, and `containerPort: 80`. Redirect it with `> web.yaml`. You can delete `resources: {}` and `status: {}`.
2	**kubectl apply**	The first run prints `pod/web created`. The second prints `pod/web unchanged`. apply is idempotent.

SCAN FOR THE LESSON

Lesson: **Kubernetes from Zero, part 5: kubectl essentials**

https://aitamer.news/posts/k8s-from-zero-05-kubectl-essentials/?utm_source=lessonkit&utm_medium=pdf&utm_campaign=k8s-05&utm_content=cheatsheet

NEXT IN THE SERIES **Next: Lesson 6, namespaces, labels and annotations**. Kubernetes from Zero, part 6, namespaces, labels and annotations. Lesson page

Free to share with credit: © aitamer.news

3	**kubectl diff**	After you add `environment: dev` under labels, the diff shows that added line. Exit code `0` means no differences, `1` means differences, and more than `1` means an error. The lesson checked `0` before the edit and `1` after it. apply then prints `pod/web configured`.
4	**kubectl edit**	Opens the live object in `KUBE_EDITOR` or `EDITOR`. The lesson changed `environment` from `dev` to `test`. The next `kubectl apply -f web.yaml` put `dev` back, because that field belongs to the file.
5	**kubectl label**	`owner=me` is a label the file never mentions. The next apply prints `pod/web unchanged` and `--show-labels` still lists `owner=me`.

apply remembers the last applied object in the annotation `kubectl.kubernetes.io/last-applied-configuration`. A JSON merge patch can change one label. Patching the container port is refused: most of a running Pod's spec is fixed. The Pods page lists the fields a Pod update may change.

Key commands

commands verbatim from the lesson

See the request

```
kubectl get pods -n k8s-from-zero -v=6 2>&1 | grep -E \
  "Config loaded|https://"
```
each API request; your kubeconfig path will differ

```
kubectl get --raw /api/v1/namespaces/k8s-from-zero | head -c \
  160; echo
```
core objects live under `/api/v1`

```
kubectl api-resources | grep -E \
  "^NAME|^pods |^namespaces |^deployments "
```
short names and API versions

Generate and apply

```
kubectl run quick --image=nginx:1.29-alpine -n k8s-from-zero
```
imperative, no file

```
kubectl run web --image=nginx:1.29-alpine --port=80 \
  --labels=app.kubernetes.io/name=web,tier=frontend -n \
  k8s-from-zero --dry-run=client -o yaml
```
redirect with `> web.yaml`

```
kubectl apply -f web.yaml
```
`created`, then `unchanged`, then `configured`

```
kubectl diff -f web.yaml
```
exit 0 none, 1 changes, above 1 error

```
kubectl apply -f web.yaml --dry-run=server
```
admission checks, stores nothing

```
kubectl explain pod.spec.containers.imagePullPolicy
```
field docs from the API schema

Change a live object

```
KUBE_EDITOR="sed -i s/environment:\ dev/environment:\ test/" \
  kubectl edit pod web -n k8s-from-zero
```
the lab's editor; a person uses a normal editor

```
kubectl get pod web -n k8s-from-zero -L environment
```
one column per label key

```
kubectl label pod web -n k8s-from-zero owner=me
```
a key the file never mentions

```
kubectl get pod web -n k8s-from-zero --show-labels
```
all labels in the last column

```
kubectl patch pod web -n k8s-from-zero --type=merge -p \
  '{"metadata":{"labels":{"environment":"test"}}}'
```
a JSON merge patch

```
kubectl get pod web -n k8s-from-zero -o \
  jsonpath="{.metadata.annotations}" | head -c 120; echo
```
`last-applied-configuration`

Select by label

```
kubectl apply -f shop.yaml
```
four Pods; `---` separates objects

```
kubectl get pods -n k8s-from-zero -L tier,environment
```
two extra columns

```
kubectl get pods -n k8s-from-zero -l \
  tier=frontend,environment=prod
```
comma means AND

```
kubectl get pods -n k8s-from-zero -l \
  'environment in (dev,prod)'
```
set-based; quote it

```
kubectl get pods -n k8s-from-zero -l '!environment'
```
objects without the key

```
kubectl label pod scratch -n k8s-from-zero environment=test \
  --overwrite
```
required when the key already has a value

```
kubectl label pod scratch -n k8s-from-zero environment-
```
a trailing `-` removes the key

```
kubectl label pods -n k8s-from-zero -l \
  app.kubernetes.io/name=shop team=payments
```
label every match

```
kubectl delete pods -n k8s-from-zero -l environment=dev
```
run the same `-l` with get first

SCAN FOR THE LESSON

Lesson: **Kubernetes from Zero, part 5: kubectl essentials**

https://aitamer.news/posts/k8s-from-zero-05-kubectl-essentials/?utm_source=lessonkit&utm_medium=pdf&utm_campaign=k8s-05&utm_content=cheatsheet

NEXT IN THE SERIES **Next: Lesson 6, namespaces, labels and annotations**. Kubernetes from Zero, part 6, namespaces, labels and annotations. Lesson page

Free to share with credit: © aitamer.news

Shape the output

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

`kubectl get pods -n k8s-from-zero -o name`
`pod/quick`, `pod/web`

`kubectl get pods -n k8s-from-zero -o jsonpath='{.items[*].metadata.name}{"\n"}'`
one field

`kubectl get pods -n k8s-from-zero -o jsonpath='{range .items[*]}{.metadata.name}{"\t"}{.status.podIP}{"\n"}{end}'`
`range ... end` loops

```
kubectl get pods -n k8s-from-zero -o \
  custom-columns=NAME:.metadata.name,IMAGE:.spec.containers[0].image,NODE:.spec.nodeName,PHASE:.status.phase
```
your own table

`kubectl delete pod quick web -n k8s-from-zero`
before the selector section

SCAN FOR THE LESSON

Lesson: **Kubernetes from Zero, part 5: kubectl essentials**

https://aitamer.news/posts/k8s-from-zero-05-kubectl-essentials/?utm_source=lessonkit&utm_medium=pdf&utm_campaign=k8s-05&utm_content=cheatsheet

NEXT IN THE SERIES **Next: Lesson 6, namespaces, labels and annotations**. Kubernetes from Zero, part 6, namespaces, labels and annotations. Lesson page

Free to share with credit: © aitamer.news