CKA · CKAD Shell set-up, aliases and the kubectl commands you reach for most in CKA and CKAD tasks, grouped by topic.

Kubernetes cheat sheet

3 sections

PREPARED FOR
You

GENERATED
27 November 2026

VERSION
v0.2.449

IN SHORT

Kubernetes shell set-up and commands for CKA and CKAD practice. Generated 2026-11-27.

1 Exam-day set-up

Type these in the first minute. Lines marked with a star in the tables below are the same set.

```bash
```

2 Aliases and settings

3 Commands by topic

Pods

Run a pod:

```bash
kubectl run web --image=nginx --port=80
```

Write the YAML instead of creating it:

```bash
kubectl run web --image=nginx --dry-run=client -o yaml > web.yaml
```

A pod with a command and labels:

```bash
kubectl run box --image=busybox --labels=app=box,tier=tools -- sleep 3600
```

A shell inside a pod:

```bash
kubectl exec -it web -- sh
```

Replace a pod from an edited file:

```bash
kubectl replace --force -f web.yaml
```

Deployments

Create a deployment:

```bash
kubectl create deployment api --image=nginx --replicas=3
```

Scale it:

```bash
kubectl scale deployment api --replicas=5
```

Change the image:

```bash
kubectl set image deployment/api nginx=nginx:1.27
```

Watch the rollout, then roll back:

```bash
kubectl rollout status deployment/api
kubectl rollout undo deployment/api
```

Rollout history:

```bash
kubectl rollout history deployment/api
```

Services

Expose a deployment inside the cluster:

```bash
kubectl expose deployment api --port=80 --target-port=8080
```

Expose on a node port:

```bash
kubectl expose deployment api --type=NodePort --port=80
```

A service for a pod:

```bash
kubectl expose pod web --port=80 --name=web-svc
```

See where a service sends traffic:

```bash
kubectl get endpointslices -l kubernetes.io/service-name=api
```

ConfigMaps and Secrets

ConfigMap from values:

```bash
kubectl create configmap app-config --from-literal=MODE=prod
  --from-literal=LEVEL=info
```

ConfigMap from a file:

```bash
kubectl create configmap nginx-conf --from-file=nginx.conf
```

Secret from values:

```bash
kubectl create secret generic db --from-literal=user=app
  --from-literal=password=s3cret
```

Read one secret value:

```bash
kubectl get secret db -o jsonpath='{.data.password}' | base64 -d
```

Jobs and CronJobs

A one-off job:

```bash
kubectl create job hello --image=busybox -- echo hello
```

A cron job every five minutes:

```bash
kubectl create cronjob tick --image=busybox --schedule='*/5 * * * *' -- date
```

Run a cron job now:

```bash
kubectl create job tick-now --from=cronjob/tick
```

RBAC

A service account:

```bash
kubectl create serviceaccount app
```

A role that reads pods:

```bash
kubectl create role pod-reader --verb=get,list,watch --resource=pods
```

Bind it to the service account:

```bash
kubectl create rolebinding app-reads --role=pod-reader --serviceaccount=default:app
```

Check what it may do:

```bash
kubectl auth can-i list pods --as=system:serviceaccount:default:app
```

Networking

Test a service from a throwaway pod:

```bash
kubectl run tmp --rm -it --image=busybox --restart=Never -- wget -qO- http://api
```

DNS lookup inside the cluster:

```bash
kubectl run tmp --rm -it --image=busybox --restart=Never -- nslookup
  api.default.svc.cluster.local
```

Network policies in a namespace:

```bash
kubectl get networkpolicy -n team-a
```

Forward a port to your node:

```bash
kubectl port-forward svc/api 8080:80
```

Debugging

Why a pod is not running:

```bash
kubectl describe pod web
```

Logs of the previous crash:

```bash
kubectl logs web --previous
```

Logs of one container:

```bash
kubectl logs web -c sidecar
```

Recent events, newest last:

```bash
kubectl get events --sort-by=.lastTimestamp
```

Node health and the kubelet:

```bash
kubectl describe node node-1
sudo journalctl -u kubelet --since '10 min ago'
```

Output formats

Wide, with node and IP:

```bash
kubectl get pods -o wide
```

One field with JSONPath:

```bash
kubectl get pod web -o jsonpath='{.status.podIP}'
```

Your own columns:

```bash
kubectl get pods -o custom-columns=NAME:.metadata.name,NODE:.spec.nodeName
```

Sort by a field:

```bash
kubectl get pods --sort-by=.metadata.creationTimestamp
```

Labels as columns:

```bash
kubectl get pods --show-labels
```

Helm

Add a repository and refresh it:

```bash
helm repo add bitnami https://charts.bitnami.com/bitnami
helm repo update
```

Install a chart with a value:

```bash
helm install web bitnami/nginx --set replicaCount=2 -n web --create-namespace
```

List releases everywhere:

```bash
helm list -A
```

Upgrade, then roll back:

```bash
helm upgrade web bitnami/nginx --set replicaCount=3 -n web
helm rollback web 1 -n web
```

From https://gazeek.com/resources/kubernetes-cheat-sheet. For your own practice: the commands need a Kubernetes cluster to run against.