Kubernetes Basics — “Hard Way (Pragmatic)” — Printable Edition

Prepared for print on 19 Feb 2026

Includes: updated commands, on-call addenda, one-page architecture diagram, and triage flowchart.

Full Guide (Updated v8)

```
# Kubernetes Basics — “Hard Way (Pragmatic)” Step■by■Step

> **Goal:** Build a strong Kubernetes foundation by bootstrapping your own cluster (kubeadm) and practicing



## Table of Contents

- [0) Production Command Ladder (Basics → Expert) — for every topic](#0-production-command-ladder-basics-ex
  - [0.1 Prerequisites (Linux/Networking)](#01-prerequisites-linuxnetworking)
  - [0.2 Container runtime (containerd)](#02-container-runtime-containerd)
  - [0.3 Kubernetes binaries (kubeadm/kubelet/kubectl)](#03-kubernetes-binaries-kubeadmkubeletkubectl)
  - [0.4 Bootstrap / Cluster bring-up (kubeadm)](#04-bootstrap-cluster-bring-up-kubeadm)
  - [0.5 Workloads (Pods/Deployments/Services)](#05-workloads-podsdeploymentsservices)
  - [0.6 Troubleshooting / Incidents](#06-troubleshooting-incidents)
  - [0.7 Helm (install/upgrade/rollback)](#07-helm-installupgraderollback)
  - [0.8 Ingress + TLS](#08-ingress-tls)
  - [0.9 NetworkPolicy](#09-networkpolicy)
  - [0.10 Kustomize](#010-kustomize)
  - [0.11 kubeadm upgrades](#011-kubeadm-upgrades)
- [1) Prerequisites](#1-prerequisites)
- [2) Install containerd (both nodes)](#2-install-containerd-both-nodes)
- [3) Install kubeadm, kubelet, kubectl (both nodes)](#3-install-kubeadm-kubelet-kubectl-both-nodes)
- [4) Bootstrap cluster (control-plane)](#4-bootstrap-cluster-control-plane)
- [5) Core objects you must know](#5-core-objects-you-must-know)
  - [Pods](#pods)
  - [Deployments + Rollouts](#deployments-rollouts)
  - [Services + DNS](#services-dns)
- [6) Debugging & Incident Commands (Survival)](#6-debugging-and-incident-commands-survival)
  - [Fast triage](#fast-triage)
  - [Pending](#pending)
  - [CrashLoopBackOff](#crashloopbackoff)
  - [ImagePullBackOff](#imagepullbackoff)
  - [Service not reachable](#service-not-reachable)
  - [Rollout control](#rollout-control)
  - [Node maintenance](#node-maintenance)
- [7) Real■World Survival Commands (Quick Reference)](#7-realworld-survival-commands-quick-reference)
- [8) Control plane internals (must-know)](#8-control-plane-internals-must-know)
- [9) Next steps](#9-next-steps)
  - [9.1 Helm basics (package manager for Kubernetes)](#91-helm-basics-package-manager-for-kubernetes)
  - [9.2 Ingress Controller + TLS (real-world HTTP routing)](#92-ingress-controller-tls-real-world-http-rout
  - [9.3 NetworkPolicy (pod-to-pod security)](#93-networkpolicy-pod-to-pod-security)
  - [9.4 Kustomize (manifest customization without templates)](#94-kustomize-manifest-customization-without-
  - [9.5 kubeadm upgrades (very real-world)](#95-kubeadm-upgrades-very-real-world)
  - [9.6 After this (optional but high impact)](#96-after-this-optional-but-high-impact)
    - [9.6.1 Metrics & Observability (metrics-server / Prometheus / Grafana)](#961-metrics-and-observability
    - [9.6.2 Logging (fluent-bit / fluentd / Loki / EFK)](#962-logging-fluent-bit-fluentd-loki-efk)
    - [9.6.3 Tracing (OpenTelemetry / Jaeger / Tempo)](#963-tracing-opentelemetry-jaeger-tempo)
    - [9.6.4 SLO/SLI + Alerting (error budgets, burn-rate alerts)](#964-slosli-alerting-error-budgets-burn-r
    - [9.6.5 Backup/Restore (etcd + app backups like Velero)](#965-backuprestore-etcd-app-backups-like-veler
    - [9.6.6 Security (PSA / RBAC / supply chain / runtime / secrets)](#966-security-psa-rbac-supply-chain-r
      - [9.6.6.1 Image scanning +

signing (supply-chain) + SBOMs](#9661-image-scanning-signing-supply-chain-sboms)
      - [9.6.6.2 Runtime security (Falco) + audit logs](#9662-runtime-security-falco-audit-logs)
      - [9.6.6.3 Secrets management (External Secrets / Vault-style patterns)](#9663-secrets-management-exte
    - [9.6.7 Policy & Governance (OPA Gatekeeper / Kyverno)](#967-policy-and-governance-opa-gatekeeper-kyver
    - [9.6.8 GitOps (ArgoCD / Flux)](#968-gitops-argocd-flux)
    - [9.6.9 Service Mesh (Istio / Linkerd) — optional](#969-service-mesh-istio-linkerd-optional)
    - [9.6.10 Cost & Capacity (requests/limits hygiene, autoscaling)](#9610-cost-and-capacity-requestslimits
    - [9.6.11 Reliability Practices (rollouts, PDBs, game-days)](#9611-reliability-practices-rollouts-pdbs-g
    - [9.6.12 Multi-cluster & DR (avoid "oops in prod")](#9612-multi-cluster-and-dr-avoid-oops-in-prod)
    - [9.6.13 Platform Patterns (Operators/CRDs, admission webhooks)](#9613-platform-patterns-operatorscrds-
    - [9.6.14 Networking Advanced (Gateway API, egress, CNI deep dive)](#9614-networking-advanced-gateway-ap

Use this as a **company/on■call ready checklist**. For each topic in the Table of Contents, you have:
- **Basics:** quick visibility / confirm status
- **Intermediate:** isolate the issue / validate assumptions
- **Advanced:** inspect controllers/CRDs/components
- **Expert:** deep diagnostics / safe recovery

> **Tip:** keep aliases in your shell:


alias k=kubectl
alias kg='kubectl get'
alias kd='kubectl describe'
alias kl='kubectl logs'
### 0.1 Prerequisites (Linux/Networking)

**Basics**
```

```
sudo swapoff -a
sudo modprobe overlay br_netfilter
sudo sysctl --system


**Intermediate**


lsmod | egrep 'br_netfilter|overlay'
sysctl net.ipv4.ip_forward
sysctl net.bridge.bridge-nf-call-iptables
ip a
ip route


**Advanced**


hostnamectl
timedatectl status
ss -lntp | head
curl -I https://pkgs.k8s.io/ 2>/dev/null | head -n 1 || true


**Expert**


sudo journalctl -b -n 200 --no-pager
sudo dmesg | tail -n 80
sudo iptables -S | head




### 0.2 Container runtime (containerd)

**Basics**


sudo systemctl status containerd --no-pager
containerd --version || true


**Intermediate**


sudo grep -n "SystemdCgroup" /etc/containerd/config.toml || true
sudo crictl info | head || true
sudo crictl ps || true
sudo crictl images | head || true


**Advanced**


sudo ctr plugins ls | head || true
sudo crictl pods | head || true
sudo crictl inspectp <pod_id> | head || true


**Expert**


sudo journalctl -u containerd -n 200 --no-pager
sudo crictl logs <container_id> | tail -n 80




### 0.3 Kubernetes binaries (kubeadm/kubelet/kubectl)

**Basics**


kubeadm version
kubectl version --client
kubelet --version


**Intermediate**


apt-mark showhold | egrep 'kubelet|kubeadm|kubectl' || true
apt-cache policy kubeadm | head || true
**Advanced**
sudo systemctl status kubelet --no-pager
```

```
sudo journalctl -u kubelet -n 200 --no-pager
```

Expert

```
# upgrade planning (control-plane)
sudo kubeadm upgrade plan 2>/dev/null || true
```

0.4 Bootstrap / Cluster bring-up (kubeadm)

Basics

```
kubectl get nodes -o wide
kubectl get pods -A | head
```

Intermediate

```
kubectl -n kube-system get pods -o wide
kubectl -n kube-system get ds | head
kubectl get events -A --sort-by=.lastTimestamp | tail -n 40
```

Advanced

```
#
 API readiness checks
kubectl get --raw /readyz 2>/dev/null | head || true
kubectl get --raw /livez 2>/dev/null | head || true
```

Expert

```
# token / join recovery (control-plane)
sudo kubeadm token list 2>/dev/null || true
sudo kubeadm token create --print-join-command 2>/dev/null || true
sudo kubeadm certs check-expiration 2>/dev/null || true
```

0.5 Workloads (Pods/Deployments/Services)

Basics

```
kubectl get pods -n <ns> -o wide
kubectl get deploy,rs -n <ns>
kubectl get svc,ep -n <ns>
```

Intermediate

```
kubectl describe pod <pod> -n <ns>
kubectl logs <pod> -n <ns> -c <ctr> --previous
kubectl exec -it <pod> -n <ns> -- sh
```

Advanced

```
kubectl rollout status deploy/<name> -n <ns>
kubectl rollout history deploy/<name> -n <ns>
kubectl set image deploy/<name> <ctr>=<image:tag> -n <ns>

kubectl patch deploy/<name> -n <ns> -p '{"spec":{"replicas":2}}'
```

Expert

```
kubectl debug -it <pod> -n <ns> --image=nicolaka/netshoot --target=<ctr> 2>/dev/null || true
kubectl explain deployment.spec.template.spec --recursive | head
### 0.6 Troubleshooting / Incidents

**Basics**
```

```
kubectl get pods -A | egrep -v 'Running|Completed' | head -n 60
kubectl get events -A --sort-by=.lastTimestamp | tail -n 60
```

Intermediate

```
kubectl describe pod <pod> -n <ns>
kubectl logs <pod> -n <ns> -c <ctr> --previous
kubectl get svc,ep -n <ns>
```

```
kubectl run -it --rm net --image=nicolaka/netshoot --restart=Never -n <ns> -- bash
```

Advanced

```
kubectl get pods -A --field-selector spec.nodeName=<node> -o wide
sudo journalctl -u kubelet -n 200 --no-pager
sudo crictl ps
```

Expert

```
# control-plane components (kubeadm)
ls -l /etc/kubernetes/manifests/ 2>/dev/null || true
sudo ss -lntp | egrep '6443|2379|2380|10250|10257|10259' || true
```

0.7 Helm (install/upgrade/rollback)

Basics

```
helm repo add bitnami https://charts.bitnami.com/bitnami
helm repo update
helm install my-app bitnami/nginx -n dev --create-namespace
```

Intermediate

```
helm list -A
helm status my-app -n dev
helm upgrade my-app bitnami/nginx -n dev --set replicaCount=2
helm rollback my-app 1 -n dev
```

Advanced

```
helm get values my-app -n dev
helm get manifest my-app -n dev | head
helm template my-app bitnami/nginx -n dev | head
helm lint ./chart 2>/dev/null || true
```

Expert (optional)

```
# diff plugin if installed
helm diff upgrade my-app bitnami/nginx -n dev 2>/dev/null || true
```

0.8 Ingress + TLS

Basics

```
kubectl get ingress -A
kubectl describe ingress <name> -n <ns>
```

Intermediate

```
kubectl get svc -n ingress-nginx
kubectl logs -n ingress-nginx -l app.kubernetes.io/name=ingress-nginx --tail=120
```

Advanced

```
kubectl get secret -n <ns> | egrep -i 'tls|cert' || true
kubectl describe secret <tls-secret> -n <ns>

kubectl run -it --rm net --image=nicolaka/netshoot --restart=Never -n <ns> -- bash
# curl -H 'Host: demo.local' http://<ingress-controller-ip>


**Expert**


kubectl get events -n <ns> --sort-by=.lastTimestamp | tail -n 40
kubectl get endpointslices -n <ns> 2>/dev/null || true




### 0.9 NetworkPolicy

**Basics**


kubectl get netpol -A
kubectl describe netpol <name> -n <ns>


**Intermediate**


kubectl run -it --rm net --image=nicolaka/netshoot --restart=Never -n <ns> -- bash


**Advanced**


kubectl -n kube-system get pods -o wide | egrep -i 'calico|cilium|weave|flannel'
kubectl -n kube-system logs -l k8s-app=calico-node --tail=100 2>/dev/null || true


**Expert**


kubectl -n kube-system logs ds/kube-proxy --tail=80 2>/dev/null || true

### 0.10 Kustomize

**Basics**


kubectl apply -k ./overlays/dev


**Intermediate**


kubectl kustomize ./overlays/dev | head
kubectl diff -k ./overlays/dev 2>/dev/null || true


**Advanced**


kubectl kustomize ./overlays/dev | less


**Expert (optional)**


kustomize build ./overlays/dev | head 2>/dev/null || true




### 0.11 kubeadm upgrades

**Basics**


kubeadm version
sudo kubeadm upgrade plan 2>/dev/null || true


**Intermediate**


kubectl cordon <node>
kubectl drain <node> --ignore-daemonsets --delete-emptydir-data
```

Advanced

```
sudo kubeadm certs check-expiration 2>/dev/null || true
```

Expert

```
kubectl uncordon <node>
kubectl get nodes -o wide
kubectl -n kube-system get pods | head
```

> ■ For **9.6 Production Engineer track**, use the dedicated section below (already includes tiered commands

1) Prerequisites

- 2 Linux VMs (Ubuntu 22.04+ recommended)
  - Control-plane: 2 CPU, 4–8 GB RAM
  - Worker: 2 CPU, 4 GB RAM
- Disable swap, enable kernel modules, sysctl for bridged traffic.

```
sudo swapoff -a
sudo sed -i.bak '/ swap / s/^/#/' /etc/fstab

sudo modprobe overlay
sudo modprobe br_netfilter

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
EOF
sudo sysctl --system
```

2) Install containerd (both nodes)

```
sudo apt-get update
sudo apt-get install -y containerd
sudo mkdir -p /etc/containerd
containerd config default | sudo tee /etc/containerd/config.toml >/dev/null
sudo sed -i 's/SystemdCgroup = false/SystemdCgroup = true/' /etc/containerd/config.toml
sudo systemctl restart containerd
sudo systemctl enable containerd
```

3) Install kubeadm, kubelet, kubectl (both nodes)

```
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/d

sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

4) Bootstrap cluster (control-plane)

```
sudo kubeadm init \
  --pod-network-cidr=192.168.0.0/16 \
  --apiserver-advertise-address=<CONTROL_PLANE_IP>

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Install Calico:

```
kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.27.0/manifests/calico.yaml
kubectl get pods -A -w


Join worker (run join command from kubeadm output):


sudo kubeadm join <CONTROL_PLANE_IP>:6443 --token <...> --discovery-token-ca-cert-hash sha256:<...>



## 5) Core objects you must know

### Pods

kubectl run nginx-pod --image=nginx:1.25 --restart=Never
kubectl describe pod nginx-pod
kubectl exec -it nginx-pod -- bash
kubectl delete pod nginx-pod


### Deployments + Rollouts

kubectl create deployment web --image=nginx:1.25
kubectl scale deploy/web --replicas=3
kubectl set image deploy/web nginx=nginx:1.26
kubectl rollout status deploy/web
kubectl rollout undo deploy/web


### Services + DNS

kubectl expose deploy web --port=80 --target-port=80 --name=web-svc
kubectl get svc,ep
kubectl run -it --rm dns --image=busybox:1.36 --restart=Never -- sh
nslookup web-svc
wget -qO-

http://web-svc
exit



## 6) Debugging & Incident Commands (Survival)

### Fast triage

kubectl get nodes -o wide
kubectl get pods -A | egrep -v 'Running|Completed' | head
kubectl get events -A --sort-by=.lastTimestamp | tail -n 50
kubectl get pods -A --sort-by='.status.containerStatuses[0].restartCount' | tail -n 20


### Pending

kubectl describe pod <pod> -n <ns>
kubectl describe node <node>
kubectl get pvc -A


### CrashLoopBackOff

kubectl logs <pod> -n <ns> -c <ctr> --previous
kubectl describe pod <pod> -n <ns>
kubectl exec -it <pod> -n <ns> -- sh


### ImagePullBackOff

kubectl describe pod <pod> -n <ns>


### Service not reachable

kubectl get svc,ep -n <ns>
kubectl describe svc <svc> -n <ns>


### Rollout control

kubectl rollout pause deploy/<name>
kubectl rollout resume deploy/<name>
kubectl rollout status deploy/<name>
kubectl rollout undo deploy/<name>
```

```
kubectl rollout restart deploy/<name>


### Node maintenance

kubectl cordon <node>
kubectl drain <node> --ignore-daemonsets --delete-emptydir-data
kubectl uncordon <node>



## 7) Real■World Survival Commands (Quick Reference)


# contexts / namespace
kubectl config get-contexts
kubectl config use-context <context>
kubectl config set-context --current --namespace=<ns>

# logs
kubectl logs -f <pod> -n <ns> -c <ctr>
kubectl logs <pod> -n <ns> -c <ctr> --previous

# emergency scale
kubectl scale deploy/<name> -n <ns> --replicas=0
kubectl scale deploy/<name> -n <ns> --replicas=<n>

# debug pod
kubectl run -it --rm netshoot --image=nicolaka/netshoot --restart=Never -- bash

# stuck resources (DANGER)
kubectl delete pod <pod> -n <ns> --grace-period=0 --force
kubectl patch <type> <name> -n <ns> -p '{"metadata":{"finalizers":[]}}' --type=merge

# node runtime triage
sudo journalctl -u kubelet -n 200 --no-pager
sudo systemctl status containerd --no-pager
sudo crictl ps
sudo crictl logs <container_id>

# cert expiry (kubeadm)
sudo kubeadm certs check-expiration



## 8) Control plane internals (must-know)


sudo ls -l /etc/kubernetes/manifests/
sudo ls -l /etc/kubernetes/pki/



## 9) Next steps

Below is a **practical roadmap** to move from “basics” → “real production + interviews”.
Each item includes **what to learn**, **what to practice**, and **must-know commands**.



### 9.1 Helm basics (package manager for Kubernetes)

**Learn:** charts, values, releases, upgrades/rollbacks.

**Practice (in any namespace):**


# add repo + search
helm repo add bitnami https://charts.bitnami.com/bitnami
helm repo update
helm search repo nginx | head

# install a chart
helm install my-nginx bitnami/nginx -n dev --create-namespace

# inspect what got installed
helm list -A
helm status my-nginx -n dev
kubectl get all -n dev

# upgrade with new values
helm upgrade my-nginx bitnami/nginx -n dev --set replicaCount=2
# rollback
```

```
helm rollback my-nginx 1 -n dev

# uninstall
helm uninstall my-nginx -n dev
```

Interview: “Helm is to K8s what apt/yum is to Linux packages (templates + release lifecycle).”

9.2 Ingress Controller + TLS (real-world HTTP routing)

Learn: Service vs Ingress, L4 vs L7, TLS termination, annotations, ingressClass.

Practice: install **NGINX Ingress Controller** and expose an app.

```
# quick: use Helm to install ingress-nginx
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm repo update
helm install ingress-nginx ingress-nginx/ingress-nginx -n ingress-nginx --create-namespace

kubectl get pods -n ingress-nginx
kubectl get svc -n ingress-
```

nginx

Create a TLS secret (self-signed for lab):

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
  -keyout tls.key -out tls.crt -subj "/CN=demo.local"

kubectl create secret tls demo-tls --cert=tls.crt --key=tls.key -n default
```

Debug commands (critical):

```
kubectl get ingress -A
kubectl describe ingress <name> -n <ns>
kubectl logs -n ingress-nginx -l app.kubernetes.io/name=ingress-nginx --tail=100
```

9.3 NetworkPolicy (pod-to-pod security)

Learn: default allow, deny-all baseline, allow by labels, egress control.

Practice: create a **deny-all** then allow only required traffic.

```
# Deny all ingress in a namespace
cat <<'EOF' | kubectl apply -n dev -f -
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-all-ingress
spec:
  podSelector: {}
  policyTypes:
  - Ingress
EOF
```

Debug commands:

```
kubectl get netpol -n dev
kubectl describe netpol deny-all-ingress -n dev

# connectivity test pod
kubectl run -it --rm net --image=nicolaka/netshoot --restart=Never -n dev -- bash
```

Interview: show you understand that NetworkPolicy requires a supporting CNI (like Calico/Cilium).

9.4 Kustomize (manifest customization without templates)

Learn: bases/overlays, patches, generators.

Practice: create a base deployment and overlay for dev/prod.

```
# build and apply from a kustomization
kubectl kustomize ./overlays/dev
kubectl apply -k ./overlays/dev

# view final rendered YAML
kubectl kustomize ./overlays/dev | less

# common: patch image
kubectl kustomize ./overlays/dev | grep image -n
```

Interview: contrast Helm (templating + releases) vs Kustomize (patch-based composition).

9.5 kubeadm upgrades (very real-world)

Learn: version skew rules, upgrade order, draining nodes.

Practice (lab only):

```
# On control-plane: check current
kubeadm version
kubectl get nodes

# plan upgrade
sudo kubeadm upgrade plan

# Upgrade control-plane (example minor/patch)
# 1) upgrade kubeadm package to target version
# 2) run: sudo kubeadm upgrade apply v1.xx.y
# 3) upgrade kubelet + kubectl packages and restart kubelet

# Node lifecycle during maintenance
kubectl cordon <node>
kubectl drain <node> --ignore-daemonsets --delete-emptydir-data
kubectl uncordon <node>
```

Must-know checks:

```
# certificate expiry issues can break clusters
sudo kubeadm certs check-expiration

# control-plane static pods
sudo ls -l /etc/kubernetes/manifests/
```

9.6 After this (optional but high impact)

This is the “**production engineer**” track. If you learn these, you’ll be confident in real clusters and to

How to use this section: for each topic below, you get **Basics → Intermediate → Advanced → Expert** c
- **Basics:** quick visibility + confirm symptoms
- **Intermediate:** isolate root cause + validate assumptions
- **Advanced:** inspect controllers/CRDs, drill into platform components
- **Expert:** multi-cluster safety, deep diagnostics, recovery patterns

> ■ Most commands are pure `kubectl`. Some are **optional** (Helm / security CLIs) and are marked.

9.6.1 Metrics & Observability (metrics-server / Prometheus / Grafana)

Basics

```
kubectl top nodes || true
kubectl top pods -A || true
kubectl describe node <node> | egrep -i 'Allocatable|Allocated|Pressure|Taints'
```

Intermediate

```
# is metrics API available?
kubectl get apiservices | grep -i metrics || true
kubectl -n kube-system get pods | grep -i metrics || true
kubectl -n kube-system logs -l k8s
-app=metrics-server --tail=200 2>/dev/null || true
# HPA signals
```

```
kubectl get hpa -A
kubectl describe hpa <hpa> -n <ns>


**Advanced**


# Prometheus/Grafana components (names vary by stack)
kubectl get pods -A | egrep -i 'prometheus|alertmanager|grafana|kube-state-metrics|node-exporter'

# Prometheus Operator CRs (if installed)
kubectl get servicemonitors,podmonitors,prometheusrules -A 2>/dev/null || true

# port-forward Grafana (example)
kubectl -n monitoring port-forward svc/grafana 3000:80 2>/dev/null || true


**Expert**


# query API endpoints (advanced debugging)
kubectl get --raw /healthz || true
kubectl get --raw /metrics | head -n 20 || true

# see top resource consumers quickly (if metrics exist)
kubectl top pods -A --sort-by=cpu | tail -n 20 || true
kubectl top pods -A --sort-by=memory | tail -n 20 || true




#### 9.6.2 Logging (fluent-bit / fluentd / Loki / EFK)

**Basics**


kubectl logs <pod> -n <ns> -c <ctr> --tail=200
kubectl logs <pod> -n <ns> -c <ctr> --previous
kubectl describe pod <pod> -n <ns> | sed -n '/Events:/,$p'


**Intermediate**


# find log agents
kubectl get pods -A | egrep -i 'fluent-bit|fluentd|promtail|vector|logstash'

# common: daemonset-based log agent
kubectl get ds -A | egrep -i 'fluent-bit|fluentd|promtail|vector'

# check agent logs (label varies)
kubectl logs -A -l app=fluent-bit --tail=200 2>/dev/null || true


**Advanced**


# Loki stack visibility
kubectl get pods -A | egrep -i 'loki|promtail|grafana'

# validate configmaps/secrets used by agents
kubectl get cm,secret -A | egrep -i 'fluent|promtail|loki|logstash' || true


**Expert**


# node-level log triage (when cluster logging is broken)
# (run on node)
sudo journalctl -u kubelet -n 200 --no-pager
sudo journalctl -u containerd -n 200 --no-pager

# correlate pod -> node quickly
kubectl get pod <pod> -n <ns> -o wide




#### 9.6.3 Tracing (OpenTelemetry / Jaeger / Tempo)

**Basics**


kubectl get pods -A | egrep -i 'otel|opentelemetry|jaeger|tempo' || true
kubectl get svc -A | egrep -i 'jaeger|tempo|otel' || true
**Intermediate**
```

```
# check collector/agent logs
kubectl logs -A -l app=opentelemetry-collector --tail=200 2>/dev/null || true
kubectl logs -A | egrep -i 'otel|collector|jaeger|tempo' || true


**Advanced**


# inspect configs (names vary)
kubectl get cm -A | egrep -i 'otel|opentelemetry|jaeger|tempo' || true
kubectl describe cm <cm> -n <ns>

# port-forward UI (adjust namespace/service)
kubectl -n observability port-forward svc/jaeger-query 16686:16686 2>/dev/null || true


**Expert**


# verify app instrumentation is exporting
kubectl logs <pod> -n <ns> -c <ctr> --tail=200
kubectl get events -A --sort-by=.lastTimestamp | tail -n 40




#### 9.6.4 SLO/SLI + Alerting (error budgets, burn-rate alerts)

**Basics**


kubectl get events -A --sort-by=.lastTimestamp | tail -n 50
kubectl get pods -A | egrep -v 'Running|Completed' | head -n 50


**Intermediate**


# alerting stack presence
kubectl get pods -A | egrep -i 'alertmanager|prometheus'

# HPA is often tied to SLO symptoms
kubectl get hpa -A
kubectl describe hpa <hpa> -n <ns>


**Advanced**


# Prometheus Operator rules
kubectl get prometheusrules -A 2>/dev/null || true
kubectl describe prometheusrule <rule> -n <ns> 2>/dev/null || true

# port-forward Alertmanager UI
kubectl -n monitoring port-forward svc/alertmanager 9093:9093 2>/dev/null || true


**Expert**


# make sure you are on the correct cluster before changing alerts
kubectl config current-context
kubectl cluster-info

# sanity check for kube-system health
kubectl -n kube-system get pods | head




#### 9.6.5 Backup/Restore (etcd + app backups like Velero)

**Basics**


kubectl get pvc -A
kubectl get pv
kubectl get sts -A


**Intermediate (etcd snapshot: kubeadm clusters)**


ETCD_POD=$(kubectl -n kube-system get pod -l component=etcd -o jsonpath='{.items[0].metadata.name}')

kubectl -n kube-system exec $ETCD_POD -- sh -c '\
```

```
ETCDCTL_API=3 etcdctl \\
  --endpoints=https://127.0.0.1:2379 \\
  --cacert=/etc/kubernetes/pki/etcd/ca.crt \\
  --cert=/etc/kubernetes/pki/etcd/server.crt \\
  --key=/etc/kubernetes/pki/etcd/server.key \\
  snapshot save /var/lib/etcd/snapshot.db\
'

kubectl -n kube-system cp ${ETCD_POD}:/var/lib/etcd/snapshot.db ./snapshot.db
ls -lh ./snapshot.db


**Advanced (Velero if installed)**


kubectl get crd | grep -i velero || true
kubectl get backups,restores -n velero 2>/dev/null || true
kubectl get schedules -n velero 2>/dev/null || true

kubectl -n velero get pods 2>/dev/null || true
kubectl -n velero logs deploy/velero --tail=200 2>/dev/null || true


**Expert**


# restore drill validation
kubectl get all -n <ns>
kubectl get pvc -n <ns>



#### 9.6.6 Security (PSA / RBAC / supply chain / runtime / secrets)

> Goal: **least privilege + safe workloads + trusted images + auditable cluster + safe secrets**.

**Basics (RBAC)**


kubectl auth can-i get pods -n <ns>
kubectl auth can-i create deployments -n <ns>
kubectl auth can-i '*' '*' --all-namespaces
kubectl auth whoami 2>/dev/null || true


**Intermediate (Pod Security Admission)**


kubectl label ns dev pod-security.kubernetes.io/enforce=restricted --overwrite
kubectl label ns dev pod-security.kubernetes.io/audit=restricted --overwrite
kubectl label ns dev pod-security.kubernetes.io/warn=restricted --overwrite
kubectl describe ns dev | egrep -i 'pod-security' -n


**Advanced (cluster quick audit)**


# privileged/hostNetwork quick audit
kubectl get pods -A -o jsonpath='{range .items[*]}{.metadata.namespace}/{.metadata.name}{"\t"}{.spec.hostNet

# images inventory
kubectl get pods -A -o jsonpath='{range .items[*]}{.metadata.namespace}/{.metadata.name}{"\t"}{range .spec.c


**Expert (security add-on presence)**


kubectl get pods -A | egrep -i 'falco|trivy|kyverno|gatekeeper|vault|external-secrets' || true


##### 9.6.6.1 Image scanning + signing (supply-chain) + SBOMs

**Basics**


# running images
kubectl get pods -n <ns> -o jsonpath='{range .items[*]}{.metadata.name}{"\t"}{range .spec.containers[*]}{.im

# pulled image digest (imageID)
kubectl get pod <pod> -n <ns> -o jsonpath='{.status.containerStatuses[*].imageID}' ; echo


**Intermediate (optional scanners)**
trivy image <image:tag>
grype <image:tag>
```

Advanced (SBOM generation — optional)

```
syft <image:tag> -o spdx-json > sbom.spdx.json
ls -lh sbom.spdx.json
```

Expert (sign + verify + enforce — optional)

```
# use digest, not tags
cosign sign <image@sha256:digest>
cosign verify <image@sha256:digest>

# policy engines used to enforce controls
kubectl get clusterpolicies,policies -A 2>/dev/null || true
kubectl get constrainttemplates,constraints 2>/dev/null || true
```

9.6.6.2 Runtime security (Falco) + audit logs

Basics

```
kubectl get ds -A | egrep -i 'falco' || true
kubectl get pods -A | egrep -i 'falco' || true
kubectl logs -A -
l app=falco --tail=200 2>/dev/null || true
```

Intermediate

```
kubectl get cm -A | egrep -i 'falco' || true
kubectl describe cm <falco-rules-cm> -n <ns>

kubectl logs <falco-pod> -n <ns> --tail=200 | egrep -i 'warning|notice|critical|rule' || true
```

Advanced (apiserver audit)

```
kubectl -n kube-system get pods | grep kube-apiserver
kubectl -n kube-system logs -l component=kube-apiserver --tail=200

# control-plane node (kubeadm)
# sudo grep -n "audit" /etc/kubernetes/manifests/kube-apiserver.yaml
```

Expert (API down)

```
# run on control-plane node
sudo journalctl -u kubelet -n 200 --no-pager
sudo crictl ps | egrep -i 'kube-apiserver|etcd'
# sudo crictl logs <apiserver_container_id> | tail -n 200
```

9.6.6.3 Secrets management (External Secrets / Vault-style patterns)

Basics

```
kubectl get secrets -n <ns>
kubectl describe secret <name> -n <ns>

# find secret usage patterns
kubectl get pods -n <ns> -o yaml | egrep -n 'secretName:|secretKeyRef:' | head
```

Intermediate (detect operators)

```
kubectl get crd | egrep -i 'externalsecret|secretstore|clustersecretstore' || true
kubectl get externalsecrets -A 2>/dev/null || true
kubectl get secretstores,clustersecretstores -A 2>/dev/null || true

kubectl get crd | egrep -i 'secretproviderclass' || true
kubectl get secretproviderclasses -A 2>/dev/null || true
```

Advanced (debug sync/mount failures)

```
kubectl get pods -A | egrep -i 'external-secrets|secrets-store-csi|vault' || true
```

```
kubectl logs -n external-secrets deploy/external-secrets --tail=200 2>/dev/null || true
kubectl logs -n kube-system ds/secrets-store-csi-driver --tail=200 2>/dev/null || true

kubectl describe externalsecret <name> -n <ns> 2>/dev/null || true
kubectl get events -n <ns> --sort-by=.lastTimestamp | tail -n 30


**Expert (rotation + least privilege)**


kubectl get pod <pod> -n <ns> -o jsonpath='{.spec.serviceAccountName}' ; echo
kubectl describe sa <sa> -n <ns>

kubectl rollout restart deploy/<name> -n <ns>
kubectl rollout status deploy/<name> -n <ns>

kubectl auth can-i get secrets -n <ns>



#### 9.6.7 Policy & Governance (OPA Gatekeeper / Kyverno)

**Basics**


kubectl get crd | egrep -i 'kyverno|gatekeeper|constrainttemplate|constraints' || true


**Intermediate**


kubectl get clusterpolicies,policies -A 2>/dev/null || true
kubectl get constrainttemplates 2>/dev/null || true
kubectl get constraints 2>/dev/null || true


**Advanced**


kubectl apply -f <resource.yaml> --dry-run=server
kubectl get events -A --sort-by=.lastTimestamp | tail -n 50


**Expert**


kubectl get validatingwebhookconfigurations,mutatingwebhookconfigurations
kubectl get pods -A | egrep -i 'kyverno|gatekeeper' || true



#### 9.6.8 GitOps (ArgoCD / Flux)

**Basics**


kubectl get ns | egrep -i 'argocd|flux' || true
kubectl get pods -n argocd 2>/dev/null || true
kubectl get pods -n flux-system 2>/dev/null || true


**Intermediate**


kubectl get applications -n argocd 2>/dev/null || true
kubectl describe application <app> -n argocd 2>/dev/null || true

kubectl get kustomizations,helmreleases,gitrepositories -A 2>/dev/null || true


**Advanced**


kubectl -n argocd logs deploy/argocd-application-controller --tail=200 2>/dev/null || true
kubectl -n flux-system logs deploy/source-controller --tail=200 2>/dev/null || true


**Expert**


kubectl config current-context
kubectl --context <prod-context> get applications -n argocd 2>/dev/null || true
#### 9.6.9 Service Mesh (Istio / Linkerd) — optional
**Basics**
```

```
kubectl get ns | e
grep -i 'istio|linkerd' || true
kubectl get pods -A | egrep -i 'istio|linkerd' || true
```

Intermediate

```
kubectl get virtualservices,destinationrules,gateways -A 2>/dev/null || true
kubectl get peerauthentication,authorizationpolicy -A 2>/dev/null || true
```

Advanced

```
kubectl get ns --show-labels | grep -i istio || true
kubectl describe pod <pod> -n <ns> | egrep -i 'istio-proxy|sidecar' || true
```

Expert (optional CLI)

```
istioctl proxy-status 2>/dev/null || true
```

9.6.10 Cost & Capacity (requests/limits hygiene, autoscaling)

Basics

```
kubectl top nodes || true
kubectl describe node <node> | egrep -i 'Allocatable|Allocated|Pressure'
```

Intermediate

```
kubectl get pods -A -o jsonpath='{range .items[*]}{.metadata.namespace}/{.metadata.name}{"\t"}{range .spec.c
kubectl get resourcequota -A
kubectl get limitrange -A
```

Advanced

```
kubectl get hpa -A
kubectl get vpa -A 2>/dev/null || true
kubectl get deploy -A | egrep -i 'cluster-autoscaler' || true
```

Expert

```
kubectl get events -A --sort-by=.lastTimestamp | egrep -i 'FailedScheduling|Preempt' | tail -n 30
```

9.6.11 Reliability Practices (rollouts, PDBs, game-days)

Basics

```
kubectl rollout status deploy/<name> -n <ns>
kubectl rollout history deploy/<name> -n <ns>
```

Intermediate

```
kubectl get pdb -A
kubectl describe pdb <name> -n <ns>
```

Advanced

```
kubectl describe pod <pod> -n <ns> | egrep -i 'Readiness|Liveness|Probe|Killing'
kubectl get events -n <ns> --sort-by=.lastTimestamp | tail -n 50
```

Expert

```
kubectl rollout pause deploy/<name> -n <ns>
```

```
kubectl rollout undo deploy/<name> -n <ns>
```

9.6.12 Multi-cluster & DR (avoid "oops in prod")

Basics

```
kubectl config get-contexts
kubectl config current-context
kubectl cluster-info
```

Intermediate

```
kubectl --context <context> get nodes
kubectl --context <context> get ns
```

Advanced

```
kubectl --context <context> get ns kube-system -o jsonpath='{.metadata.uid}' ; echo
```

Expert

```
kubectl get nodes
kubectl get pods -A | egrep -v 'Running|Completed' | head
```

9.6.13 Platform Patterns (Operators/CRDs, admission webhooks)

Basics

```
kubectl get crd | head
kubectl api-resources | head
```

Intermediate

```
kubectl get <crd-kind> -A
kubectl describe <crd-kind> <name> -n <ns>
```

Advanced

```
kubectl get pods -A | egrep -i 'operator|controller'
kubectl logs -n <ns> deploy/<controller-deploy> --tail=200
```

Expert

```
kubectl get validatingwebhookconfigurations,mutatingwebhookconfigurations
kubectl get events -A --sort-by=.lastTimestamp | tail -n 50
```

9.6.14 Networking Advanced (Gateway API, egress, CNI deep dive)

Basics

```
kubectl get svc,ep -A
kubectl get endpointslices -A 2>/dev/null || true
```

Intermediate

```
kubectl run -it --rm dns --image=busybox:1.36 --restart=Never -- sh
nslookup kubernetes.default
exit

kubectl get netpol -A
```

```
kubectl describe netpol <name> -n <ns>
```

Advanced

```
kubectl -n kube-system get pods -o wide | egrep -i 'calico|cilium|weave|flannel|kube-proxy'
kubectl -n kube-system logs ds/kube-proxy --tail=80 2>/dev/null || true
kubectl -n kube-system logs -l k8s-app=calico-node --tail=100 2>/dev/null || true
```

Expert

```
kubectl get crd | egrep -i 'gatewayclasses\.gateway\.networking\.k8s\.io|gateways\.gateway\.networking\.k8s\
|| true
kubectl get gatewayclass,gateway,httproute -A 2>/dev/null || true

# node-level network triage (run on node)
sudo ss -lntp | egrep '6443|10250' || true
sudo iptables -S | head
```

---

10) Addenda — Real■World Commands You'll Be Glad You Had

> This section complements your original guide with frequently■used, on■call friendly commands and small YA

10.1 Storage (PVC/PV/CSI/StatefulSets)

```
# List storage classes & PVCs
kubectl get sc
kubectl get pvc -A

# Quick PVC (5Gi) example (namespace: dev)
cat <<'EOF' | kubectl apply -n dev -f -
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: data-pvc
spec:
  accessModes: ["ReadWriteOnce"]
  resources:
    requests:
      storage: 5Gi
EOF

# Minimal StatefulSet skeleton using the PVC
cat <<'EOF' | kubectl apply -n dev -f -
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web-ss
spec:
  serviceName: web-ss
  replicas: 2
  selector:
    matchLabels: {app: web}
  template:
    metadata: {labels: {app: web}}
    spec:
      containers:
      - name: nginx
        image: nginx:1.25
        volumeMounts:
        - name: data
          mountPath: /var/www/html
  volumeClaimTemplates:
  - metadata: {name: data}
    spec:
      accessModes: ["ReadWriteOnce"]
      resources: {requests: {storage: 5Gi}}
EOF

# PVC troubleshoot
kubectl describe pvc data-pvc -n dev
kubectl get events -n dev --sort-by=.lastTimestamp | tail -n 50
```

10.2 Scheduling (taints/tolerations, affinity)

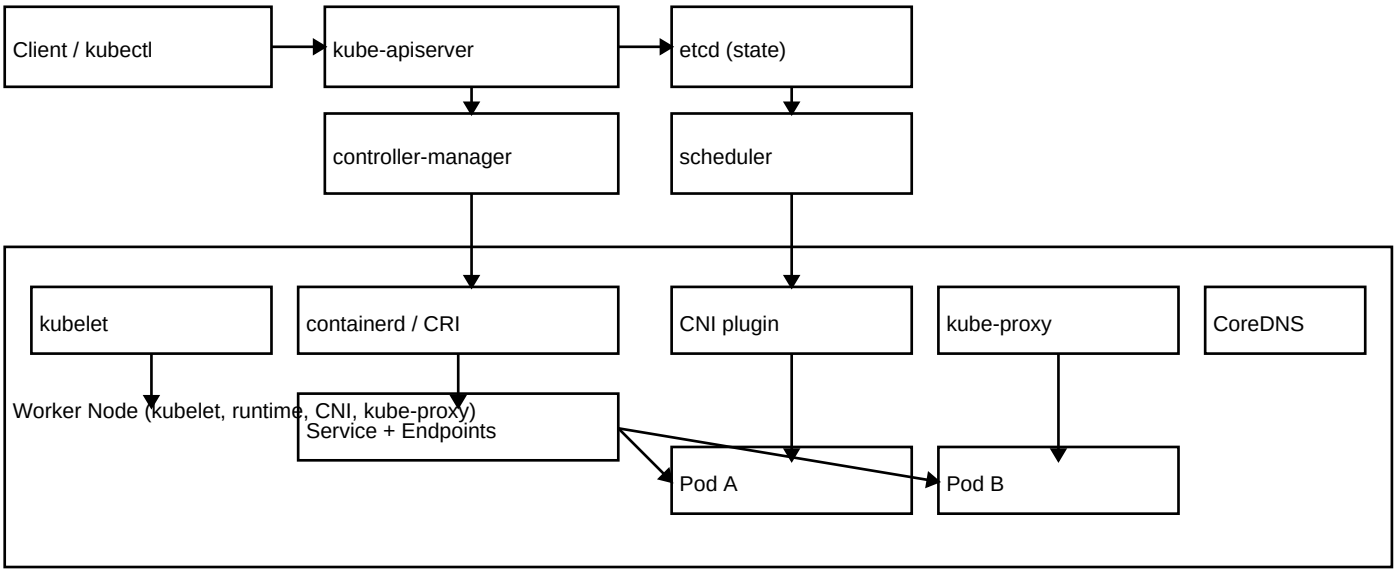
```
# Show taints
kubectl describe node <node> | egrep -i 'Taints|Roles'

# Add / remove taints (maintenance or dedicated nodes)
kubectl taint nodes <node> dedicated=web:NoSchedule
```

```
kubectl taint nodes <node> dedicated-   # remove

# Node/Pod affinity snippets
cat <<'EOF' | kubectl apply -n dev -f -
apiVersion: apps/v1
kind: Deployment
metadata: {name: web-aff}
spec:
  replicas: 2
  selector: {matchLabels: {app: web}}
  template:
    metadata: {labels: {app: web}}
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: node-pool
                operator: In
                values: ["blue"]
      tolerations:
      - key: dedicated
        operator: Equal
        value: web
        effect: NoSchedule
      containers:
      - name: nginx
        image: nginx:1.25
EOF

### 10.3 Connectivity & Port■forwarding (CoreDNS, svc reachability)
# CoreDNS health
kubectl -n kube-system get pods -l k8s-app=kube-dns -o wide ||   kubectl -n kube-system get pods -l k8s-app=
kubectl -n kube-system logs -l k8s-app=coredns --tail=200

# DNS test from a temporary toolbox pod
kubectl run -it --rm dns --image=busybox:1.36 --restart=Never -- sh
nslookup kubernetes.default
wget -qO- http://web-svc.default.svc.cluster.local
exit

# Port-forward when Ingress/LB isn’t ready yet
kubectl -n dev port-forward svc/web-svc 8080:80
kubectl -n dev port-forward deploy/web 8080:80

### 10.4 Jobs & CronJobs
# One-off Job
cat <<'EOF' | kubectl apply -n dev -f -
apiVersion: batch/v1
kind: Job
metadata: {name: echo-job}
spec:
  template:
    spec:
      restartPolicy: OnFailure
      containers:
      - name: main
        image: busybox:1.36
        command: ["/bin/sh","-c","echo hello; sleep 2"]
EOF

# CronJob (every 5 minutes)
cat <<'EOF' | kubectl apply -n dev -f -
apiVersion: batch/v1
kind: CronJob
metadata: {name: hello-cj}
spec:
  schedule: "*/5 * * *
*"
  jobTemplate:
    spec:
      template:
        spec:
          restartPolicy: OnFailure
          containers:
          - name: main
            image: busybox:1.36
            args: ["/bin/sh","-c","date; echo hello"]
EOF

### 10.5 Kubeconfig & contexts (merge/switch)
# View / switch
kubectl config get-contexts
kubectl config current-context
```

```
kubectl config use-context <ctx>

# Merge another kubeconfig file
KUBECONFIG=$HOME/.kube/config:/path/other.kubeconfig kubectl config view --raw > /tmp/merged
mv /tmp/merged $HOME/.kube/config

### 10.6 kubectl productivity (selectors, jsonpath, sorting)
# Label selectors and wide output
kubectl get pods -n dev -l app=web -o wide

# Sort by restarts
kubectl get pods -A --sort-by='.status.containerStatuses[0].restartCount' | tail -n 20

# Quick images inventory (per ns)
kubectl get pods -n dev -o jsonpath='{range .items[*]}{.metadata.name}■{range .spec.containers[*]}{.image}{'
{end}'

### 10.7 HPA quickstart
kubectl autoscale deployment web -n dev --min=2 --max=5 --cpu-percent=70
kubectl get hpa -A
kubectl describe hpa web -n dev

### 10.8 Copy files in/out of pods
# To pod
kubectl -n dev cp ./local.txt <pod>:/tmp/local.txt
# From pod
kubectl -n dev cp <pod>:/etc/os-release ./os-release

### 10.9 Node & runtime deep dives (ephemeral debug)
# Ephemeral debug container into a failing Pod (1.25+)
kubectl debug -n dev pod/<pod> -it --image=nicolaka/netshoot --target=<container>

# Debug a node
kubectl debug node/<node> -it --image=ubuntu:22.04 -- chroot /host

### 10.10 Clean-up / recovery (use with care)
# Reset a broken kubeadm node (DANGER: removes cluster state from node)
sudo kubeadm reset -f && sudo systemctl restart containerd

# Remove stuck finalizers on a resource (last resort)
kubectl patch <type> <name> -n <ns> -p '{"metadata":{"finalizers":[]}}' --type=merge

---

## 11) One■Page: Kubernetes Request Flow & Components (Visual)
> See the PDF for a high■level architecture diagram showing: kubectl → API Server → etcd, Scheduler/Contro

---

## 12) Triage Flowchart — “Something is Down”
> The PDF includes a one■page flowchart that guides you from **API health** → **workload state** → **netwo
```

One■Page Architecture — Request Flow

Kubernetes — High-Level Architecture & Request Flow



Triage Flowchart — “Something is Down”