PAGE: 1

DATE: 31 Aug 2025

Kubernetes Notes

- Introduction
- Design & kubernetes architecture
- Core Concepts
- Scheduling
- Logging & Monitoring
- Application Lifecycle Management
- Cluster Maintenance
- Security
- Storage
- Networking

Introduction

- Kubernetes (K8s) orchestrates containers across a cluster using a desired state model.
- Why it exists: schedule containers, keep them healthy, scale, and self-heal.
- Control plane components: API server, etcd (state store), scheduler, controller manager, cloud-controller.
- Node components: kubelet (agent), container runtime, kube-proxy (service networking).
- Workload types: Deployments (stateless), StatefulSets (stateful), DaemonSets (one per node), Jobs/CronJobs (batch).

Design & kubernetes architecture

- Choose managed (GKE/EKS/AKS) or self-managed (kubeadm, kops, kubespray). Plan HA: multiple control-plane nodes and an odd-sized etcd cluster across zones.
- Core components: API Server (front door), etcd (state), Scheduler (places pods), Controller Manager (reconcilers), Cloud Controller (cloud integration).
- Request flow when creating a Pod: 1) Client sends manifest to API Server; 2) API Server validates and writes the Pod (Pending) to etcd; 3) Scheduler watches and picks a suitable node; 4) Scheduler writes the decision back to API Server; 5) Kubelet on the chosen node watches the API Server, pulls images, creates containers; 6) Probes pass; 7) Kubelet updates Pod status; 8) Service discovery exposes it if matched by a Service.
- Install outline with kubeadm: init control plane, join workers, install CNI, create a default StorageClass, deploy an Ingress Controller, configure RBAC and resource quotas.

- Install outline with kubeadm: init control plane, join workers, install CNI, create a default StorageClass, deploy an Ingress Controller, configure RBAC and resource quotas.

Core Concepts

- Pod: the smallest deployable unit; one or more containers + shared network/storage.
- ReplicaSet: maintains pod count; Deployment manages ReplicaSets for rollout/rollback.
- Service: stable virtual IP/DNS over pods. Types: ClusterIP, NodePort, LoadBalancer, Headless.
- ConfigMap/Secret: inject config and sensitive data; mount as env or volume.
- Labels/Selectors & Annotations: attach metadata; select pods/services; docs and tooling.
- Namespaces: isolate resources; use ResourceQuota + LimitRange to control usage.
- Health: liveness/readiness/startup probes; resource requests/limits for fair scheduling.

Scheduling

- Default scheduler scores nodes; place pods respecting constraints.
- NodeSelector/NodeAffinity: pick nodes by labels; Affinity/Anti-Affinity for co-location or spreading.
- Taints/Tolerations: keep workloads off certain nodes unless tolerated.
- TopologySpreadConstraints: even spread across zones/nodes; avoid hotspots.
- PriorityClass & Preemption: higher-priority pods can evict lower-priority pods.

- PriorityClass & Preemption: higher-priority pods can evict lower-priority pods.
- DaemonSet: ensure one pod per node (e.g., log collectors). Jobs/CronJobs for batch/scheduled tasks.

Application Lifecycle

- Strategies: rolling update (default), recreate, blue/green, canary (progressive delivery).
- Scaling: HPA (CPU/memory/custom metrics), VPA (resource rightsizing), Cluster Autoscaler.
- kubectl rollout: status/history/undo; maxUnavailable and maxSurge tune safety vs speed.
- Config delivery: volumes/env from ConfigMaps/Secrets; watch for changes; use checksum annotations.
- Packaging: Helm charts or Kustomize overlays; GitOps for auditable changes.

Cluster Maintenance

- Upgrade flow: cordon + drain nodes, upgrade control plane, then workers; verify workloads; uncordon.
- etcd: take snapshots; secure with TLS; consider external etcd for HA.
- Backups: manifests, PV snapshots (CSI), and critical secrets.
- Node ops: set taints for maintenance; use PodDisruptionBudgets to protect availability.
- Quotas/limits: prevent noisy neighbors; track usage with Prometheus and reports.

Security

- Identity/AuthN: users/groups via OIDC; ServiceAccounts for workloads.

- Identity/AuthN: users/groups via OIDC; ServiceAccounts for workloads.
- AuthZ: RBAC roles and bindings; least privilege; separate CI/CD and runtime identities.
- NetworkPolicies: default deny; allow minimal pod-to-pod/egress.
- Pod Security: Kubernetes Pod Security Standards (Baseline/Restricted) enforced via admission.
- Secrets: encrypt at rest; mount as tmpfs; avoid logging; consider external secret managers.
- Supply chain: use trusted registries; scan images; sign/verify (cosign); enforce via policy (OPA/Gatekeeper or Kyverno).

Storage

- Volumes: container-attached storage lifecycle with pod; ephemeral or persistent.
- PersistentVolume (PV) & PersistentVolumeClaim (PVC): bind storage to workloads.
- StorageClass: dynamic provisioning via CSI drivers; set reclaim policy and parameters.
- AccessModes: RWO/ROX/RWX; VolumeMode: Filesystem or Block; resizing and snapshots via CSI.
- StatefulSets: volumeClaimTemplates give stable per-pod storage.

Logging & Monitoring

- Metrics pipeline: metrics-server for HPA; Prometheus + Grafana for time series + dashboards.
- cAdvisor exposes container metrics via kubelet; scrape with Prometheus.
- Logging: app writes to stdout/stderr; collect with Fluent Bit/Fluentd/Vector via DaemonSet.
- Tracing/OTel: instrument services; export to Jaeger/Tempo/X-Ray; add exemplars to metrics.
- Alerting: Alertmanager or SaaS (Datadog/New Relic); page on SLO burn and user impact.

Networking

- Flat pod network: every pod gets an IP; pods can reach each other (CNI implements).
- Service discovery: CoreDNS; kube-proxy programs iptables/ipvs for Service VIPs.
- Service types: ClusterIP (internal), NodePort (nodeIP:port), LoadBalancer (external), Headless (no VIP).
- Ingress: L7 routing terminated by an Ingress Controller; use IngressClass and annotations for features.
- Egress: NAT gateways or egress controllers; limit with NetworkPolicies.