🌟 Kubernetes From Scratch – Complete Guide

Welcome to this comprehensive, visually enriched guide to Kubernetes (K8s)!
This guide uses help you understand container orchestration step-by-step.

Let's orchestrate some magic! 🚀

🚀 What is Kubernetes?

Kubernetes is an **open-source container orchestration platform** that automates the deployment, scaling, and management of containerized applications across clusters.
Originally developed by Google, it is now maintained by the **Cloud Native Computing Foundation (CNCF)**.

⭐ Key Benefits

Feature	***Description***
High Availability	Ensures applications run redundantly across nodes to reduce downtime.
Zero-Downtime Deployments	Performs rolling updates without service interruption.
Auto-Scaling	Adjusts resources automatically based on metrics (CPU, memory).
Auto-Healing	Restarts failed containers, reschedules pods, and maintains replicas.

⚙️ What Does Container Orchestration Do?

Kubernetes handles the heavy lifting of managing containers at scale.

Core Functions

- **Provision & Deploy** containers across multiple nodes
- **High Availability** through balanced distribution
- **Auto-Scaling** workloads (via HPA, VPA)
- **Efficient Resource Allocation** (CPU, memory, storage)
- **Load Balancing & Traffic Routing**
- **Service Discovery** between microservices
- **Secure Communication** via Network Policies

🔄 Popular Orchestrators Comparison

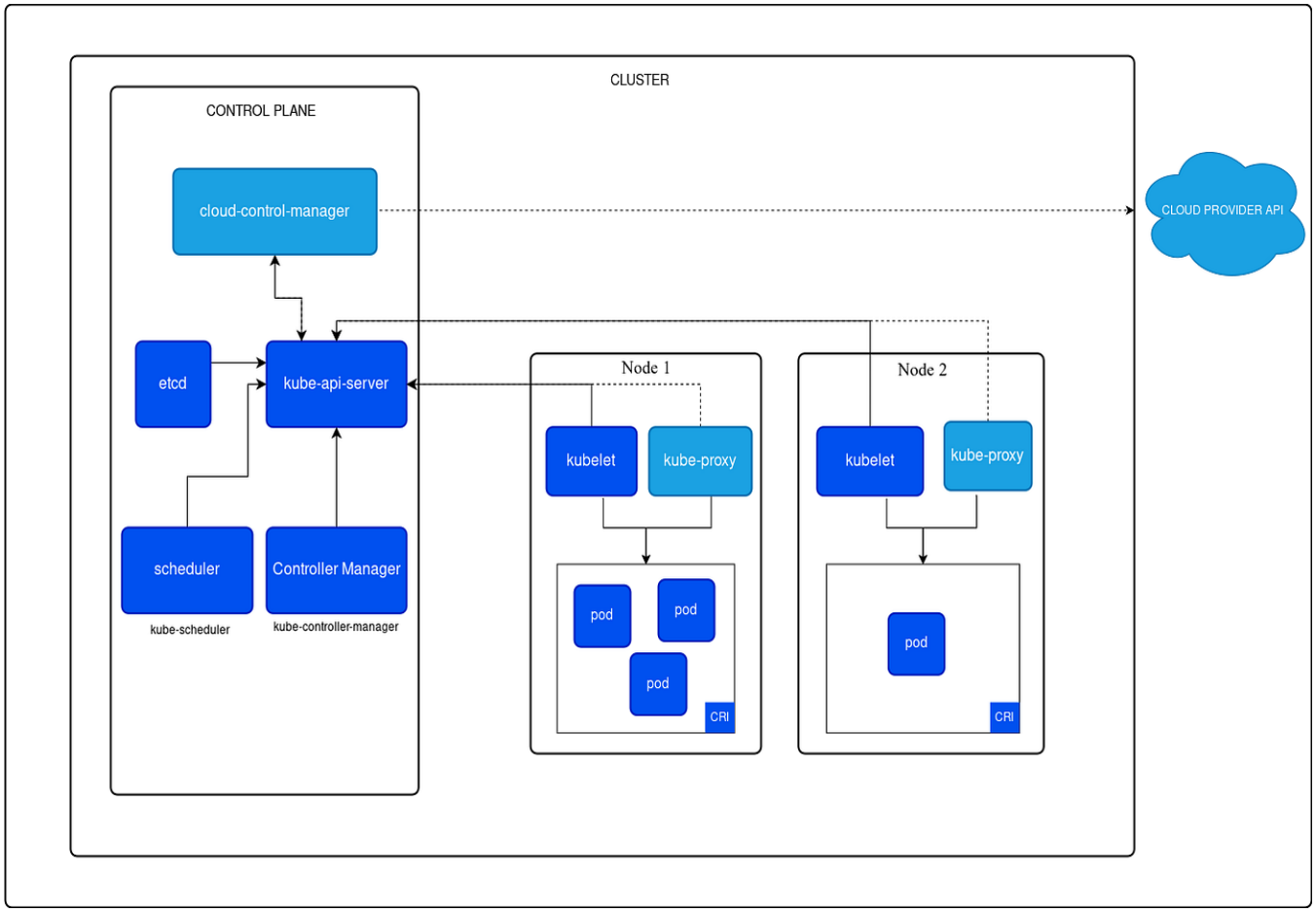
Orchestrator	Strengths	Best For	Drawbacks
Kubernetes	Feature-rich, flexible, huge ecosystem	Enterprise apps	Steep learning curve
Docker Swarm	Simple, Docker-native	Small teams, quick deployments	Less advanced
Apache Mesos	Handles mixed workloads	Big data clusters	Complex, declining adoption

---

🧱 Kubernetes Components & Architecture

Kubernetes operates through a cluster — a group of machines working together.

🏛 Cluster Overview



Master Node = Control plane = Brain

Worker Nodes = Muscle

Multi-Master Setup = High Availability

🟦 Master Node Components

Component	Role	Key Features
API Server	Central endpoint for all REST/K8s requests	Handles kubectl, validates config, exposes `/metrics`
Controller Manager	Maintains desired state	Includes node/replica controllers
Scheduler	Assigns pods to nodes	Considers resources, affinities, taints
etcd	Distributed key-value store	Stores entire cluster state

Essential Add-ons

- **CoreDNS** – Service discovery
- **Kubernetes Dashboard** – Web UI
- **Container Runtime** – Docker, containerd, CRI-O

🟩 Worker Node Components

Component	***Role***	***Key Features***
Kubelet	Node agent managing pod lifecycle	Runs containers, performs health checks
Kube-Proxy	Network routing for services	Implements load balancing via iptables/IPVS

Pro Tip: Kubelet communicates only with the API Server, never directly with other nodes.

🛠 kubectl — The Swiss Army Knife CLI

General syntax:

```
kubectl [command] [type] [name] [flags]
```

Examples of Commands

- **Commands:** `get`, `apply`, `create`, `delete`, `describe`
- **Types:** `pod`, `deployment`, `service`, `namespace`
- **Flags:** `-n`, `-o yaml`, `--all-namespaces`

📘 kubectl Cheat Sheet

Action	Command Example
List resources	`kubectl get pods -A`
Describe	`kubectl describe pod my-pod`
Create/Apply	`kubectl apply -f deployment.yaml`
Scale	`kubectl scale deploy/my-app --replicas=5`
Logs & Exec	`kubectl logs my-podkubectl exec -it my-pod -- /bin/sh`
Delete	`Kubectl delete pod my-pod`
Port-forward	`kubectl port-forward svc/my-svc 8080:80`
Events	`kubectl get events --sort-by=.metadata.creationTimestamp`

📦 Core Kubernetes Concepts

🟦 Pod — The Smallest Unit

Pods wrap around one or more containers that share:

- Network namespace (one IP)
- Storage volumes
- Lifecycle management rules

Pod Types

- **Single-container Pod**
- **Multi-container Pod** (Sidecar pattern)

🟧 Deployment — Declarative Scaling

- Manages ReplicaSets
- Handles rolling updates/rollbacks
- Ensures stateless workloads scale

Deployment YAML

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: nginx
        image: nginx:1.14
```

🌐 Service — Stable Networking

Type	*Use Case*	*Exposure*
ClusterIP	Internal-only	Virtual IP inside cluster
NodePort	External access via nodes	`<NodeIP>:<NodePort>`
LoadBalancer	Cloud LB	Public external IP

🧭 Namespace — Virtual Clusters

Used for isolation, organization, and resource limits.

Default namespaces:

- `default`
- `kube-system`
- `kube-public`

🔐 Secrets, ConfigMaps & Storage

Concept	*Purpose*	*Security Notes*
Secrets	Store sensitive info	Base64 encoded, not encrypted
ConfigMaps	Store config data	Mounted as env or volumes
Volumes	Persist data	Supports emptyDir, hostPath, PVC

🔄 ReplicaSet & DaemonSet

- **ReplicaSet** – Maintains N identical Pods
- **DaemonSet** – Runs one Pod per node (logs, monitoring)

🏷 Labels, Selectors & Annotations

- **Labels:** metadata to organize resources
- **Selectors:** query/filter resources
- **Annotations:** metadata for external tools

🎛 Affinity, Anti-Affinity & Taints

Control where Pods are scheduled:

- **Node Affinity** – Match node labels
- **Pod Affinity/Anti-Affinity** – Group or separate pods
- **Taints/Tolerations** – Restrict scheduling on specific nodes

🔐 RBAC — Role-Based Access Control

Resource	Scope	Purpose
Role	Namespace	Defines permissions
ClusterRole	Cluster	Global permissions
RoleBinding	Namespace	Binds Role
ClusterRoleBinding	Cluster	Binds ClusterRole

Best Practice: Always follow **least privilege**.

🌐 Networking & Ingress

Ingress — Smart HTTP Routing

Feature	Benefit
Path-based routing	`/api` → backend1, `/web` → frontend
TLS termination	HTTPS security
Host-based routing	`api.example.com`, `app.example.com`

⚡ Scaling & Autoscaling

Horizontal Pod Autoscaler (HPA)

Automatically scales deployments.

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: my-app-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: my-app
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```

Vertical Pod Autoscaler (VPA)

Adjusts CPU/memory requests automatically (may restart pods).

📊 Monitoring, Logging & Troubleshooting

Built-in Tools

- `kubectl get events -w`
- `kubectl logs my-pod -f`
- `kubectl top pods`

Observability Stack

Tool	***Purpose***
Prometheus	Metrics
Grafana	Dashboards
EFK Stack	Centralized logs
Jaeger	Tracing

Troubleshooting Flow:

- `kubectl describe pod <name>`
- `kubectl exec -it <pod> -- curl <service>`

🛡️ Security Best Practices

- Enforce Pod Security Standards
- Default-deny Network Policies
- Image scanning (Trivy, Clair)
- External Secret Stores
- Admission Controllers
- API server audit logs