Kubernetes (k8s)

P by Praveen Kumar

What is Kubernetes?

- Kubernetes (K8s) is an open-source **container orchestration platform**.
- Automates deployment, scaling, and management of containerized applications.
- Originally developed by **Google**, now maintained by **CNCF**.
- De facto industry standard for modern cloud-native applications.

Why Kubernetes?

- Manual container management is complex.
- Ensures applications are:
  - **Highly available**
  - **Auto-scaled**
  - **Self-healing**
  - **Efficiently scheduled**
- Runs containers consistently across **dev, test, prod**.

Problems Without Kubernetes

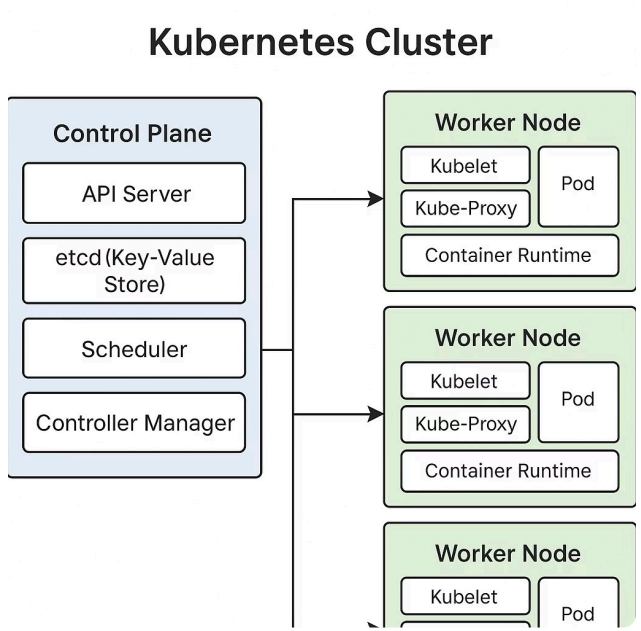
- Manual container deployment
- No automatic scaling
- No auto restart on failure
- Hard to update containers
- Difficult load balancing between containers
- Poor resource utilization

Kubernetes Features

- Automated Deployment & Rollbacks
- Self-healing (restart / replace failed containers)
- Horizontal Auto Scaling
- Service Discovery & Load Balancing
- Storage Orchestration
- Infrastructure abstraction (runs anywhere)
- Secret & Config management

Kubernetes Architecture Overview

- **Master/Control Plane** – manages the cluster.
- **Worker Nodes** – run container workloads.
- Cluster → Group of Nodes
- Each node runs multiple Pods


Kubernetes Cluster
Control Plane
API Server
etcd (Key-Value Store)
Scheduler
Controller Manager
Worker Node
Kubelet
Kube-Proxy
Pod
Container Runtime
Worker Node
Kubelet
Kube-Proxy
Pod
Container Runtime
Worker Node
Kubelet
Pod

Control Plane Components

1. **API Server**
2. **etcd (Key-Value Store)**
3. **Scheduler**
4. **Controller Manager**
5. **Cloud Controller Manager**

API Server

- The **central communication point** for the cluster.
- All kubectl commands go through API server.
- Validates and processes API requests.

etcd

- Distributed, consistent key–value store.
- Stores cluster state (pods, nodes, services, configs).
- Highly available and fault-tolerant.

Scheduler

- Assigns Pods to nodes.
- Considers:
  - Resource requirements
  - Node capacity
  - Constraints/affinities
  - Taints & tolerations

Controller Manager

- Ensures desired cluster state.
- Types of controllers:
  - Node Controller
  - Replication Controller
  - Endpoint Controller
  - Service Account & Token Controller

Worker Node Components

1. **Kubelet**
2. **Kube-Proxy**
3. **Container Runtime (Docker / containerd)**

Kubelet

- Runs on every node.
- Ensures containers in a Pod are running.
- Talks to container runtime.

Kube-Proxy

- Handles networking.
- Provides internal/external load balancing.
- Manages network routing rules.

Container Runtime

- Software responsible for running containers.
- Supported runtimes:
  - containerd
  - CRI-O
  - Docker (deprecated as runtime after 1.20)

Kubernetes Objects

- Pod
- ReplicaSet
- Deployment
- Service
- ConfigMap
- Secret
- Namespace
- StatefulSet
- DaemonSet
- Job & CronJob

Pod

- Smallest deployable unit in K8s.
- Contains one or more containers.
- Has its own IP address.
- Ephemeral (temporary).

ReplicaSet

- Ensures the desired number of pod replicas are running.
- Maintains high availability.

Deployment

- Most commonly used object.
- Manages ReplicaSets.
- Supports:
  - Rolling updates
  - Rollbacks
  - Zero-downtime deploys

Service

- Provides a stable IP and DNS name.
- Types:
  - ClusterIP (internal)
  - NodePort (external access)
  - LoadBalancer (cloud LB)
  - ExternalName (DNS mapping)

ConfigMap

- Stores non-sensitive configuration data.
- Injected as environment variables or files.

Secret

- Stores sensitive information:
  - Passwords
  - Tokens
  - Certificates

Encoded using Base64.

Namespace

- Logical cluster partition.
- Useful for:
  - Multi-environment (dev, test, prod)
  - RBAC & resource limits

StatefulSet

- Used for stateful apps:
  - Databases
  - Message queues
- Provides stable network identities and storage.

DaemonSet

- Ensures a pod runs on **all nodes**.
- Used for:
  - Log collectors
  - Monitoring agents

Job

- Runs a task **once** and completes.

CronJob

- Runs tasks on a schedule (like Linux cron).

Kubernetes Networking

- Every Pod gets a unique IP.
- Pods communicate without NAT.
- Service provides stable networking.
- CNI Plugins:
  - Flannel
  - Calico
  - Weave
  - Cilium

Kubernetes Storage

- Supports static & dynamic provisioning.
- Persistent Volume (PV)
- Persistent Volume Claim (PVC)
- Storage classes

Kubernetes Workflow

1. Developer writes Deployment YAML.
2. `kubectl apply -f deployment.yaml`.
3. API server validates request.
4. Scheduler assigns Pod to node.
5. Kubelet runs container.
6. Service exposes the Pod.
7. Autoscaler adjusts replicas if needed.

Scaling in Kubernetes

Horizontal Pod Autoscaler (HPA)

- Scales pods based on:
  - CPU usage
  - Memory usage
  - Custom metrics

Load Balancing in Kubernetes

- Internal LB → ClusterIP
- External LB → NodePort, LoadBalancer
- Ingress Controller for advanced routing

Ingress

- HTTP/HTTPS routing to internal services.
- Supports SSL termination.
- Used for real-world web applications.

Kubernetes Deployment Strategies

- Rolling Update
- Recreate Deployment
- Blue-Green Deployment
- Canary Deployment

YAML Example (Deployment + Service)

Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

Service

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  type: NodePort
  selector:
    app: nginx
  ports:
  - port: 80
    targetPort: 80
    nodePort: 30007
```

kubectl Commands

- `kubectl get pods`
- `kubectl get svc`
- `kubectl get deployment`
- `kubectl describe pod <name>`
- `kubectl logs <pod>`
- `kubectl scale deployment nginx-depl --replicas=5`
- `kubectl apply -f file.yaml`
- `kubectl delete -f file.yaml`

Kubernetes Advantages

- Cloud vendor-agnostic
- Auto scaling
- High availability
- Declarative configuration
- Large ecosystem
- Zero downtime deployments

Kubernetes Use Cases

- Scalable web applications
- Microservices
- Big data workloads
- CI/CD automation
- Edge computing
- Real-time analytics

Kubernetes in Cloud

- Amazon EKS
- Google GKE
- Azure AKS
- DigitalOcean Kubernetes
- RedHat OpenShift