**What Is Kubernetes?**

Kubernetes (often abbreviated as **K8s**) is an **open-source container orchestration platform** designed to automate the deployment, scaling, and management of containerized applications. It was originally developed by Google and is now maintained by the **Cloud Native Computing Foundation (CNCF)**.

**Key Features of Kubernetes**

1. **Automated Deployment & Scaling** – Deploy applications automatically and scale them up or down based on demand.
2. **Load Balancing** – Distributes traffic across containers to ensure high availability.
3. **Self-Healing** – Automatically restarts failed containers and replaces unresponsive nodes.
4. **Service Discovery & Networking** – Provides built-in networking and service discovery.
5. **Storage Orchestration** – Supports persistent storage solutions like AWS EBS, Google Persistent Disk, NFS, etc.
6. **Secret & Configuration Management** – Manages sensitive data (secrets) and configurations separately from the application.
7. **Rolling Updates & Rollbacks** – Deploy updates without downtime and roll back if needed.

**Why Use Kubernetes?**

* Manages **multiple containers** efficiently.
* Provides a **declarative way** to define infrastructure.
* Works with **various cloud providers** (AWS, GCP, Azure) and on-premises.
* Ensures **high availability** and **fault tolerance**.

**Types of Kubernetes clusters**

**Single-Node Cluster**:

* Contains only one node running both the control plane and worker node components.
* Used for development, testing, or learning purposes (e.g., Minikube, Kind).

**Multi-Node Cluster**:

* Contains multiple nodes: one or more control plane nodes and multiple worker nodes.
* Suitable for production workloads and high availability.

**2. Based on Deployment Environment**

* **On-Premises Kubernetes Cluster**
  + Deployed within a data center or on private infrastructure.
  + Uses tools like **kubeadm** or OpenShift for setup.
  + Requires manual network and storage configuration.
* **Cloud-Based Kubernetes Cluster**
  + Managed Kubernetes services in public cloud environments:
    - **Amazon EKS (Elastic Kubernetes Service)**
    - **Google GKE (Google Kubernetes Engine)**
    - **Azure AKS (Azure Kubernetes Service)**
    - **DigitalOcean Kubernetes, Linode Kubernetes, etc.**
  + Cloud provider manages control plane and networking.
* **Hybrid Kubernetes Cluster**
  + Combines on-premises and cloud resources.
  + Uses tools like **Anthos (Google), Azure Arc, or OpenShift Hybrid Cloud**.
  + Helps in smooth migration and multi-cloud deployments.
* **Edge Kubernetes Cluster**
  + Runs Kubernetes clusters on edge devices or remote locations.
  + Used in IoT, 5G, and real-time applications.
  + Example tools: **K3s (lightweight Kubernetes), MicroK8s**.

**3. Based on High Availability (HA)**

* **Non-HA (Single Master) Cluster**
  + Single control plane node.
  + Risky for production as failure of the master node causes downtime.
  + Common in development environments.
* **HA (High Availability) Cluster**
  + Multiple control plane nodes (at least 3) for redundancy.
  + Used in mission-critical production systems.
  + Ensures failover and minimizes downtime.

**4. Based on Cluster Management**

* **Standalone Kubernetes Cluster**
  + A single independent cluster managed manually.
  + Requires setting up networking, storage, and monitoring separately.
* **Federated Kubernetes Cluster**
  + A system for managing multiple clusters across different regions/clouds.
  + Allows for **multi-cluster deployments and workload balancing**.
  + Implemented using **Kubernetes Federation (kubefed)**.

**What is control plane?**

In Kubernetes, the **Control Plane** is the brain of the cluster that manages and orchestrates all cluster activities. It consists of several components that work together to maintain the cluster’s desired state, schedule workloads, and manage networking and storage.

**Key Components of the Control Plane:**

1. API Server (kube-apiserver)
2. Controller Manager (kube-controller-manager)
3. Scheduler (kube-scheduler)
4. Etcd (Distributed Key-Value Store)
5. **Cloud Controller Manager (cloud-controller-manager)** (only in cloud environments)

The **Control Plane** runs on the **Master Node(s)** and ensures that the cluster remains functional and self-healing.

**Explanation:**

## **1. API Server (**kube-apiserver**)**

### ****Role:****

* The **API Server** is the entry point for all administrative tasks in the Kubernetes cluster.
* It exposes a **REST API** that allows users, CLI tools (like kubectl), and other components to interact with the cluster.
* It is **the only Control Plane component that directly communicates with the worker nodes**.
* The kubectl command contacts the kube-apiserver, which fetches the requested data from etcd and returns the response.
* The API Server also performs authentication, authorization, and request validation.

**2. Scheduler (kube-scheduler).**

### ****Role:****

* Assigns newly created **Pods** to worker nodes.
* Makes decisions based on **resource availability**, **node affinity**, **taints & tolerations**, and **custom scheduling policies**.

### ****How It Works:****

1. A new Pod is created but remains in a **Pending** state.
2. The Scheduler picks a worker node based on:
   * CPU & Memory availability
   * Node Selector, Affinity, Taints & Tolerations
   * Pod Priority and Resource Requests/Limits
3. After selection, the **Scheduler binds the Pod** to the chosen node.

Example: If a node has **insufficient CPU**, the scheduler will pick another node

## **3. Controller Manager (**kube-controller-manager**)**

### ****Role:****

* Runs multiple **controllers** that ensure the cluster’s desired state matches the actual state.
* Monitors objects like **Pods, Nodes, Replicas**, and ensures their count and health.

### ****Important Controllers:****

1. **Node Controller**
   * Monitors worker nodes and marks them **unhealthy** if they fail.
   * If a node is down, it moves its Pods to another node.
2. **Replication Controller**
   * Ensures the correct number of replicas for a Deployment/ReplicaSet.
   * If a Pod crashes, it spins up a new one.
3. **Endpoints Controller**
   * Updates the list of available **Pods** for a **Service**.
   * Ensures traffic goes only to healthy pods.
4. **Service Account & Token Controller**
   * Manages authentication for Pods to communicate with the API Server.

## **4. Etcd (Key-Value Store)**

### ****Role:****

* Stores **all cluster configuration and state**.
* Acts as a distributed, consistent, and highly available **database**.
* Every update in the cluster is recorded in etcd.

### ****How It Works:****

1. When you create a **Pod**, the API Server writes this change to etcd.
2. Other components (Scheduler, Controllers) **watch etcd** for changes and act accordingly.
3. If the API Server restarts, it reloads the cluster state from etcd.

Since etcd is **critical for Kubernetes**, it is recommended to:

* **Back it up regularly**
* Run it in **HA mode** for reliability

## **5. Cloud Controller Manager (**cloud-controller-manager**)** (Used in Cloud Deployments)

*If Kubernetes is running* ***on-premises****, this component* ***is not needed****.*

### ****Role:****

* Allows Kubernetes to integrate with cloud providers like AWS, Azure, and GCP.
* It separates cloud-specific logic from the rest of the cluster.

### ****Key Responsibilities:****

1. **Node Management:** Automatically detects cloud VM failures and replaces them.
2. **Load Balancers:** Creates external Load Balancers (e.g., AWS ELB) for Services.
3. **Storage Provisioning:** Dynamically creates **cloud storage volumes** (EBS, GCE Persistent Disk).
4. **Network Routes:** Manages **network routing** between cloud resources.

### ****How the Control Plane Works Together****

1. You deploy an **application** using kubectl apply -f deployment.yaml.
2. The **API Server** stores the new deployment request in etcd.
3. The **Scheduler** assigns Pods to the best worker nodes.
4. The **Controller Manager** ensures the correct number of Pods are running.
5. The **Cloud Controller Manager** (if using a cloud) provisions resources like Load Balancers or Volumes.