Kubernetes (K8s)

What is Kubernetes?

Kubernetes (K8s) is a powerful **container orchestration tool** that helps **manage and scale containerized applications automatically.**

Think of Kubernetes like a **traffic controller** for your applications, ensuring everything runs smoothly, even when demand increases or servers fail.

Real-Life Example:

Imagine you run a **food delivery service** like Swiggy or Zomato. Your application needs to handle thousands of orders at once.

- **Without Kubernetes**: You manually add/remove servers to manage traffic, which is slow and inefficient.
- **With Kubernetes**: It **automatically scales** your service based on demand, restarting failed services and balancing traffic.

Main Purpose of Kubernetes

Kubernetes helps in:
✅ **Scaling applications automatically** (handling more customers in peak hours)
✅ **Ensuring high availability** (restarting services if they fail)
✅ **Load balancing traffic** (distributing orders evenly among kitchens)
✅ **Efficient resource utilization** (using the right number of servers to avoid wastage)
✅ **Automating deployment & updates** (updating your app without downtime)

Main Components of Kubernetes

Kubernetes has **two main parts**:

1. **Control Plane** (Management Layer)
2. **Worker Nodes** (Execution Layer)

1️⃣ Control Plane (Brain of Kubernetes)

This manages and controls everything in Kubernetes.
📌 **Key Components:**

- **API Server**: The entry point for all requests (like a restaurant's ordering system).
- **Scheduler**: Assigns work (like deciding which kitchen prepares which dish).
- **Controller Manager**: Ensures desired state (like making sure every order is delivered).
- **etcd (Key-Value Store)**: Stores configuration data (like a database of active orders).

2️⃣ Worker Nodes (Chefs & Kitchens)

These are the actual machines running the applications.
📌 **Key Components:**

- **Kubelet**: Ensures containers are running properly (like a kitchen manager).
- **Container Runtime** (Docker, containerd): Runs the containers (like a chef cooking meals).
- **Kube-Proxy**: Manages network traffic between containers (like a waiter delivering food).

Kubernetes Key Objects (Resources)

1. **Pods**: The smallest unit in Kubernetes, like a food order containing one or more dishes.
2. **Deployments**: Manages multiple pods, ensuring they are always available (like a restaurant ensuring chefs are available).
3. **Services**: Provides a stable network endpoint for pods (like a restaurant's front desk that takes orders).
4. **Ingress**: Manages external access (like a food delivery app connecting customers to restaurants).
5. **ConfigMaps & Secrets**: Store configuration settings securely (like secret recipes and discount codes).

Kubernetes Key Components

Kubernetes has multiple components that work together to **orchestrate and manage containers** efficiently. Here's a breakdown of the most important ones, explained using **real-life examples**.

1️⃣ kubectl (User Interface for Kubernetes)

- `kubectl` is the **command-line tool** used to interact with the Kubernetes cluster.
- It sends requests to the **API Server**, which then manages the cluster.

Real-Life Example:

Think of **kubectl** as a **food delivery app (like Swiggy/Zomato)** where you place an order.

- When you use `kubectl apply -f deployment.yaml`, it's like ordering food from the app.
- The order request is sent to the **API Server**, which then assigns it to a restaurant (worker node).

Common `kubectl` Commands:

```
kubectl get nodes        # View all nodes in the cluster

kubectl get pods         # List all running pods

kubectl describe pod <pod-name>  # Get details of a specific pod

kubectl delete pod <pod-name>  # Delete a pod

kubectl scale deployment myapp --replicas=5  # Scale application to 5 instances
```

2️⃣ Kubelet (Worker Node Agent)

🔹 `kubelet` is an agent that runs on **each worker node** and ensures that containers are running properly.
🔹 It communicates with the **API Server** and executes commands for managing containers.

Real-Life Example:

Think of **kubelet** as a **kitchen manager** in a restaurant.

- The **API Server (Owner)** tells the kitchen manager what dishes (containers) need to be prepared.
- The **kubelet (kitchen manager)** ensures that chefs (containers) are cooking the right food (running properly).
- If a chef (container) stops working, kubelet informs the owner (API Server).

How Kubelet Works:

1. Checks the **desired state** of a container (like verifying the order list).
2. Ensures the container is **running** properly.
3. Reports the health status of the node to the API Server.

3️⃣ API Server (Main Communication Hub)

🔹 The **API Server** is the heart of Kubernetes. It is responsible for **processing requests** from kubectl and other services.
🔹 It acts as the **entry point** for all operations in Kubernetes.

Real-Life Example:

Imagine the **API Server** as a **restaurant's front desk**:

- You (user) place an order using the **kubectl app**.
- The **front desk (API Server)** receives your order.
- It sends the request to the right **kitchen (worker node)** for processing.
- The API Server ensures everything is recorded in a database (**etcd**).

How API Server Works:

1. Receives requests from `kubectl`, dashboards, or external applications.
2. Validates and processes requests.
3. Stores configuration details in **etcd**.
4. Sends instructions to controllers, schedulers, and worker nodes.

4️⃣ etcd (Cluster Database - Stores Configurations)

🔹 `etcd` is a **key-value database** that **stores the cluster state and configuration.**
🔹 It is **highly available** and **distributed**, ensuring Kubernetes knows what's happening at all times.

Real-Life Example:

Think of **etcd** as a **restaurant's order history database**:

- Every time you order, the restaurant records the details in a system (**etcd**).
- If the kitchen manager (kubelet) forgets an order, they can **check the database**.
- Even if one system crashes, the order history is safe.

What etcd Stores?

✔️ Cluster state (list of running nodes, pods, services)
✔️ Configuration details
✔️ Security policies

5️⃣ Scheduler (Assigns Work to Nodes)

🔹 The **Scheduler** decides where to run new applications (pods) inside the cluster.
🔹 It finds the **best node** based on resource availability (CPU, memory, network).

Real-Life Example:

Imagine the **Scheduler** as a **food delivery dispatcher**:

- When an order is placed, the dispatcher (Scheduler) finds the **nearest available restaurant (worker node)**.

- It assigns the order to a restaurant **that has enough resources** (enough chefs to cook).
- If one restaurant is too busy, it assigns the order to another.

How Scheduler Works?

1. Checks the **resource availability** on nodes.
2. Selects a **suitable node** to run the pod.
3. Sends the assignment details to the API Server.

6️⃣ Controller Manager (Maintains Desired State)

🔹 The **Controller Manager** ensures that the cluster always matches the **desired state** defined by the user.
🔹 It has multiple controllers, such as:

- **Node Controller** (manages node failures)
- **Replication Controller** (ensures correct number of pods)
- **Service Controller** (handles networking services)

Real-Life Example:

Think of the **Controller Manager** as a **restaurant supervisor**:

- If a chef (pod) stops cooking, the supervisor **hires a new one** (starts a new pod).
- If there are **too many chefs** and not enough orders, the supervisor **removes extra chefs** (scales down pods).
- If an ingredient is missing (node failure), the supervisor **finds alternatives**.

How Controller Manager Works?

✔️ Continuously **monitors the cluster state**.
✔️ Fixes issues when something is not as expected.
✔️ Automates tasks like **restarting failed pods, scaling applications, and managing nodes**.

Summary of Components with Examples

Component	Purpose	Real-Life Example
kubectl	User interface to control Kubernetes	A food delivery app (Swiggy/Zomato)
Kubelet	Ensures containers are running on worker nodes	A kitchen manager
API Server	Entry point for all Kubernetes requests	Restaurant front desk
etcd	Stores cluster state and configurations	Order history database
Scheduler	Assigns work to worker nodes	Food delivery dispatcher
Controller Manager	Ensures the cluster matches the desired state	Restaurant supervisor

---

Basic Kubernetes Commands

Here are some important **kubectl** commands:

1️⃣ Cluster Information & Nodes

```
kubectl cluster-info    # Get details about the cluster
kubectl get nodes       # List all worker nodes in the cluster
```

2️⃣ Working with Pods

```
kubectl get pods          # List all running pods
kubectl describe pod <pod-name>  # Get detailed info about a specific pod
kubectl logs <pod-name>   # View logs of a pod
kubectl delete pod <pod-name>  # Delete a pod
```

3️⃣ Deployments

```
kubectl create deployment myapp --image=nginx  # Deploy an application
kubectl get deployments       # List all deployments
kubectl scale deployment myapp --replicas=5  # Scale deployment to 5 replicas
kubectl delete deployment myapp  # Delete a deployment
```

4️⃣ Services & Networking

```
kubectl expose deployment myapp --type=NodePort --port=80  # Expose a deployment as a service
kubectl get services     # List all services
kubectl delete service myapp  # Delete a service
```

5️⃣ Configuration & Secrets

```
kubectl create configmap myconfig --from-literal=ENV=production  # Create a ConfigMap
kubectl create secret generic mysecret --from-literal=DB_PASS=admin123  # Create a secret
kubectl get configmaps  # List all ConfigMaps
kubectl get secrets  # List all secrets
```

Real-Life Example: Running an E-commerce Website

Let's say you are running **Amazon** or **Flipkart** and need to handle traffic spikes during the **Big Billion Days Sale**.

- 🔹 **Pods:** Handle user requests (like ordering a phone).
- 🔹 **Deployments:** Ensure enough pods are running at all times.
- 🔹 **Services:** Ensure customers can always access the website.
- 🔹 **Autoscaling:** Increases/decreases servers based on demand.
- 🔹 **Rolling Updates:** Deploy new features **without downtime**.

Final Thoughts

💡 Kubernetes automates application management by distributing workloads efficiently and ensuring high availability.
🚀 It is widely used by companies like **Google, Netflix, Amazon, and Uber** for managing large-scale applications.

Would you like a **hands-on demo** on setting up a Kubernetes cluster? 😊