**Kubernetes Study Notes - Day 30**

**1. Introduction to Kubernetes**

**What is Kubernetes?**

* **Definition**: Kubernetes (K8s) is an **open-source container orchestration platform** designed to automate the deployment, scaling, and management of containerized applications.
* **Key Features**:
  + **Auto-scaling** (Handles traffic spikes)
  + **Auto-healing** (Self-recovery of failed containers)
  + **Multi-host deployment** (Cluster-based architecture)
  + **Enterprise-grade support** (Load balancing, security, networking)

**Image: Kubernetes Overview**

**Why Kubernetes?**

* Docker is great for **single-host container management**, but lacks:
  + **Scalability**
  + **High availability**
  + **Enterprise features** (Firewall, API Gateway, Load Balancing)
* Kubernetes solves these problems by providing a **cluster-based approach**.

**2. Problems with Docker & How Kubernetes Solves Them**

| **Problem with Docker** | **Kubernetes Solution** |
| --- | --- |
| **Single-host limitation** (All containers run on one machine) | **Multi-node cluster** (Pods can run across multiple nodes) |
| **No Auto-healing** (If a container crashes, manual restart is needed) | **ReplicaSets & Deployments** (Automatically restarts failed containers) |
| **No Auto-scaling** (Manual scaling required) | **Horizontal Pod Autoscaler (HPA)** (Scales based on traffic) |
| **Lacks Enterprise features** (No built-in load balancer, firewall, etc.) | **Ingress Controllers, Network Policies, Custom Resources** |

**3. Key Kubernetes Concepts**

**1. Kubernetes Cluster**

* A **group of nodes** (servers) that run containerized applications.
* **Components**:
  + **Master Node (Control Plane)** – Manages the cluster.
  + **Worker Nodes** – Run the actual workloads (Pods).

**Image: Kubernetes Cluster Architecture**

**2. Pods**

* **Definition**: The smallest deployable unit in Kubernetes.
* **Contains**:
  + One or more containers (usually **one main container + sidecars**).
  + Shared storage & network.

**Example YAML:**

apiVersion: v1

kind: Pod

metadata:

name: nginx-pod

spec:

containers:

- name: nginx

image: nginx:latest

**3. ReplicaSets & Deployments**

* **ReplicaSet**: Ensures a specified number of Pod replicas are running.
* **Deployment**: Manages ReplicaSets (enables rolling updates & rollbacks).

**Example YAML:**

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:latest

**4. Services & Load Balancing**

* **Service**: Exposes Pods to the network (ClusterIP, NodePort, LoadBalancer).
* **Ingress**: Manages external access (HTTP/HTTPS routing).

**Example YAML:**

apiVersion: v1

kind: Service

metadata:

name: nginx-service

spec:

selector:

app: nginx

ports:

- protocol: TCP

port: 80

targetPort: 80

type: LoadBalancer

**5. Horizontal Pod Autoscaler (HPA)**

* Automatically scales Pods based on CPU/memory usage.

**Example YAML:**

apiVersion: autoscaling/v1

kind: HorizontalPodAutoscaler

metadata:

name: nginx-hpa

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: nginx-deployment

minReplicas: 1

maxReplicas: 10

targetCPUUtilizationPercentage: 80

**4. Kubernetes Architecture**

**Control Plane (Master Node) Components**

| **Component** | **Role** |
| --- | --- |
| **API Server** | Entry point for all commands |
| **Scheduler** | Assigns Pods to Nodes |
| **Controller Manager** | Ensures desired state (e.g., ReplicaSets) |
| **etcd** | Key-value store for cluster data |

**Worker Node Components**

| **Component** | **Role** |
| --- | --- |
| **Kubelet** | Ensures Pods are running |
| **Kube-proxy** | Manages networking rules |
| **Container Runtime** (Docker, containerd) | Runs containers |

**Image: Kubernetes Architecture**

**5. Summary & Key Takeaways**

✅ **Kubernetes vs. Docker**: Docker is for **single-host containers**, while Kubernetes is for **orchestrating multi-host clusters**.  
✅ **Auto-healing**: Kubernetes **restarts failed Pods automatically**.  
✅ **Auto-scaling**: Uses **HPA** to scale based on demand.  
✅ **Enterprise-ready**: Supports **load balancing, security policies, and networking**.

**Next Steps**

🔹 Learn **Kubernetes CLI (kubectl)**  
🔹 Practice **Deploying a Sample App**  
🔹 Explore **Helm for Package Management**

📌 **Quote from the Video**:

*"Kubernetes is easy. If you understand the* ***why****, the* ***how*** *becomes simple."*

🚀 **Stay tuned for Day 31: Deep Dive into Kubernetes Architecture!** 🚀

**Feedback & Discussion**

💬 **Questions?** Drop them in the comments!  
👍 **Like & Share** if you found this helpful!

**Author**: Abhishek  
**Channel**: Complete DevOps Course  
**Day**: 30  
**Topic**: Kubernetes Introduction

# \*\*Kubernetes Study Notes - Day 30\*\*

## \*\*1. Introduction to Kubernetes\*\*

### \*\*What is Kubernetes?\*\*

- \*\*Definition\*\*: Kubernetes (K8s) is an \*\*open-source container orchestration platform\*\* designed to automate the deployment, scaling, and management of containerized applications.

- \*\*Key Features\*\*:

- \*\*Auto-scaling\*\* (Handles traffic spikes)

- \*\*Auto-healing\*\* (Self-recovery of failed containers)

- \*\*Multi-host deployment\*\* (Cluster-based architecture)

- \*\*Enterprise-grade support\*\* (Load balancing, security, networking)

### \*\*Why Kubernetes?\*\*

- Docker is great for \*\*single-host container management\*\*, but lacks:

- \*\*Scalability\*\*

- \*\*High availability\*\*

- \*\*Enterprise features\*\* (Firewall, API Gateway, Load Balancing)

- Kubernetes solves these problems by providing a \*\*cluster-based approach\*\*.

---

## \*\*2. Problems with Docker & How Kubernetes Solves Them\*\*

| \*\*Problem with Docker\*\* | \*\*Kubernetes Solution\*\* |

|-------------------------|-------------------------|

| \*\*Single-host limitation\*\* (All containers run on one machine) | \*\*Multi-node cluster\*\* (Pods can run across multiple nodes) |

| \*\*No Auto-healing\*\* (If a container crashes, manual restart is needed) | \*\*ReplicaSets & Deployments\*\* (Automatically restarts failed containers) |

| \*\*No Auto-scaling\*\* (Manual scaling required) | \*\*Horizontal Pod Autoscaler (HPA)\*\* (Scales based on traffic) |

| \*\*Lacks Enterprise features\*\* (No built-in load balancer, firewall, etc.) | \*\*Ingress Controllers, Network Policies, Custom Resources\*\* |

---

## \*\*3. Key Kubernetes Concepts\*\*

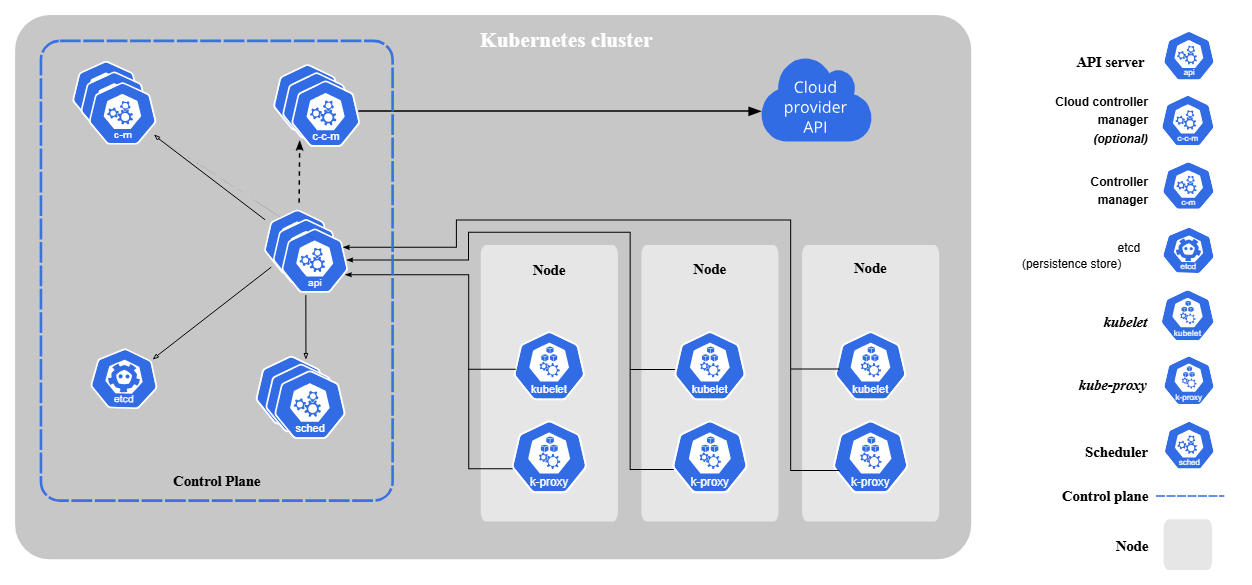
### \*\*1. Kubernetes Cluster\*\*

- A \*\*group of nodes\*\* (servers) that run containerized applications.

- \*\*Components\*\*:

- \*\*Master Node (Control Plane)\*\* – Manages the cluster.

- \*\*Worker Nodes\*\* – Run the actual workloads (Pods).



![Kubernetes Cluster](https://d33wubrfki0l68.cloudfront.net/2475489eaf20163ec0f54ddc1d92aa8d4c87c96b/e7c81/images/docs/components-of-kubernetes.svg)

### \*\*2. Pods\*\*

- \*\*Definition\*\*: The smallest deployable unit in Kubernetes.

- \*\*Contains\*\*:

- One or more containers (usually \*\*one main container + sidecars\*\*).

- Shared storage & network.

\*\*Example\*\*:

```yaml

apiVersion: v1

kind: Pod

metadata:

name: nginx-pod

spec:

containers:

- name: nginx

image: nginx:latest

```

### \*\*3. ReplicaSets & Deployments\*\*

- \*\*ReplicaSet\*\*: Ensures a specified number of Pod replicas are running.

- \*\*Deployment\*\*: Manages ReplicaSets (enables rolling updates & rollbacks).

\*\*Example\*\*:

```yaml

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:latest

```

### \*\*4. Services & Load Balancing\*\*

- \*\*Service\*\*: Exposes Pods to the network (ClusterIP, NodePort, LoadBalancer).

- \*\*Ingress\*\*: Manages external access (HTTP/HTTPS routing).

\*\*Example\*\*:

```yaml

apiVersion: v1

kind: Service

metadata:

name: nginx-service

spec:

selector:

app: nginx

ports:

- protocol: TCP

port: 80

targetPort: 80

type: LoadBalancer

```

### \*\*5. Horizontal Pod Autoscaler (HPA)\*\*

- Automatically scales Pods based on CPU/memory usage.

\*\*Example\*\*:

```yaml

apiVersion: autoscaling/v1

kind: HorizontalPodAutoscaler

metadata:

name: nginx-hpa

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: nginx-deployment

minReplicas: 1

maxReplicas: 10

targetCPUUtilizationPercentage: 80

```

---

## \*\*4. Kubernetes Architecture\*\*

### \*\*Control Plane (Master Node) Components\*\*

| \*\*Component\*\* | \*\*Role\*\* |

|--------------|---------|

| \*\*API Server\*\* | Entry point for all commands |

| \*\*Scheduler\*\* | Assigns Pods to Nodes |

| \*\*Controller Manager\*\* | Ensures desired state (e.g., ReplicaSets) |

| \*\*etcd\*\* | Key-value store for cluster data |

### \*\*Worker Node Components\*\*

| \*\*Component\*\* | \*\*Role\*\* |

|--------------|---------|

| \*\*Kubelet\*\* | Ensures Pods are running |

| \*\*Kube-proxy\*\* | Manages networking rules |

| \*\*Container Runtime\*\* (Docker, containerd) | Runs containers |

![Kubernetes Architecture](https://www.weave.works/docs/cloud/latest/images/kubernetes-architecture.png)

---

## \*\*5. Summary & Key Takeaways\*\*

✅ \*\*Kubernetes vs. Docker\*\*: Docker is for \*\*single-host containers\*\*, while Kubernetes is for \*\*orchestrating multi-host clusters\*\*.

✅ \*\*Auto-healing\*\*: Kubernetes \*\*restarts failed Pods automatically\*\*.

✅ \*\*Auto-scaling\*\*: Uses \*\*HPA\*\* to scale based on demand.

✅ \*\*Enterprise-ready\*\*: Supports \*\*load balancing, security policies, and networking\*\*.

---

## \*\*Next Steps\*\*

🔹 Learn \*\*Kubernetes CLI (`kubectl`)\*\*

🔹 Practice \*\*Deploying a Sample App\*\*

🔹 Explore \*\*Helm for Package Management\*\*

📌 \*\*Quote from the Video\*\*:

> \*"Kubernetes is easy. If you understand the \*\*why\*\*, the \*\*how\*\* becomes simple."\*

🚀 \*\*Stay tuned for Day 31: Deep Dive into Kubernetes Architecture!\*\* 🚀

---

### \*\*Feedback & Discussion\*\*

💬 \*\*Questions?\*\* Drop them in the comments!

👍 \*\*Like & Share\*\* if you found this helpful!

---

\*\*Author\*\*: Abhishek

\*\*Channel\*\*: Complete DevOps Course

\*\*Day\*\*: 30

\*\*Topic\*\*: Kubernetes Introduction