# Kubernetes Basics Training Tutorial

## Module 1: Introduction to Kubernetes

### Objective

Understand what Kubernetes is, why it is needed, and how it fits into the container ecosystem.

### 1.1 What is Kubernetes?

Kubernetes (K8s) is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications.

* Originally developed by Google, now maintained by the Cloud Native Computing Foundation (CNCF).

### 1.2 Why Use Kubernetes?

**Challenges with Containers:**

* Managing many containers manually is hard.
* Ensuring high availability, scaling, rolling updates, and resource management are complex without automation.

**Kubernetes Provides:**

* Container orchestration
* Self-healing (auto-restart, reschedule)
* Load balancing
* Declarative configuration
* Scalability

### 1.3 Kubernetes Architecture Overview

**Master Node Components:**

* **kube-apiserver**: Frontend of the control plane, receives REST commands.
* **etcd**: Key-value store for all cluster data.
* **kube-scheduler**: Assigns Pods to Nodes based on resource availability.
* **kube-controller-manager**: Runs controllers to manage cluster state.

**Worker Node Components:**

* **kubelet**: Agent that ensures containers are running in a Pod.
* **kube-proxy**: Handles networking and forwarding traffic.
* **Container Runtime**: Software to run containers (Docker, containerd).

### Example:

Imagine you want to deploy a web app with 3 containers. Kubernetes:

* Schedules containers across available nodes.
* Monitors health and restarts failed containers.
* Exposes your app through a service.

## Module 2: Core Concepts

### Objective

Understand the basic building blocks of a Kubernetes cluster.

### 2.1 Cluster

A Kubernetes cluster consists of at least one master node and multiple worker nodes.

### 2.2 Node

A node is a machine (virtual or physical) where Kubernetes runs your application.

### 2.3 Pod

A Pod is the smallest deployable unit in Kubernetes.

* Can contain one or more containers.
* Containers in a Pod share storage/network and are scheduled together.

**Example YAML:**

apiVersion: v1  
kind: Pod  
metadata:  
 name: mypod  
spec:  
 containers:  
 - name: myapp  
 image: nginx  
 ports:  
 - containerPort: 80

### 2.4 ReplicaSet

Ensures a specified number of pod replicas are running at any time.

**Example YAML:**

apiVersion: apps/v1  
kind: ReplicaSet  
metadata:  
 name: myapp-rs  
spec:  
 replicas: 3  
 selector:  
 matchLabels:  
 app: myapp  
 template:  
 metadata:  
 labels:  
 app: myapp  
 spec:  
 containers:  
 - name: myapp  
 image: nginx

### 2.5 Deployment

Manages ReplicaSets and provides declarative updates for Pods.

**Example YAML:**

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

### 2.6 Namespace

A namespace is a Kubernetes object used for organizing and separating resources within a cluster.

* Helps isolate environments like dev, test, and prod.
* Useful for multi-team and multi-tenant setups.

**Example Commands:**

kubectl create namespace dev  
kubectl get namespaces  
kubectl run nginx --image=nginx -n dev

**Example YAML:**

apiVersion: v1  
kind: Namespace  
metadata:  
 name: dev

### 2.7 Service

Provides network access to a set of Pods.

* **ClusterIP**: Internal access.
* **NodePort**: Exposes service on each node’s IP.
* **LoadBalancer**: Provisioned by cloud providers for external access.

**Example YAML (NodePort):**

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

### Hands-On Tasks:

1. Use kubectl to create a Pod and inspect it.
2. Create a Deployment with 3 replicas.
3. Expose the Deployment using a Service.
4. Create and switch to a new Namespace.
5. Deploy resources inside the new Namespace.

End of Module 2