Docker & Kubernetes – Beginner to Interview Ready Guide

What is Docker?

Docker is a containerization platform that packages an application along with its dependencies so it can run consistently across different environments.

Why Docker is Needed

It solves the 'works on my machine' problem by ensuring the same environment everywhere.

Core Docker Components

Dockerfile (recipe), Image (blueprint), Container (running app), Docker Engine, Docker Registry.

Docker Flow

Code → Dockerfile → Image → Container → Running Application.

Limitations of Docker

Docker cannot manage containers at scale. No auto-healing, auto-scaling, or load balancing.

What is Kubernetes?

Kubernetes is a container orchestration system that manages Docker containers at scale.

Why Kubernetes is Needed

It ensures applications are always running, scalable, and highly available.

Kubernetes Cluster

A cluster is the complete Kubernetes environment consisting of control plane and worker nodes.

Node

A node is a machine (VM or physical) that runs pods.

Pod

A pod is the smallest deployable unit in Kubernetes. It wraps one or more containers.

Deployment

Deployment manages pods and provides scaling, rolling updates, and rollback.

Service

Service provides a stable network endpoint and load balancing for pods.

Ingress

Ingress manages external HTTP/HTTPS access to services.

Auto-Healing

Kubernetes automatically replaces failed pods to maintain desired state.

Auto-Scaling

Kubernetes increases or decreases pods based on load.

Load Balancing

Traffic is evenly distributed across healthy pods.

Zero-Downtime Deployment

New versions are deployed gradually without interrupting users.

Docker vs Kubernetes

Docker runs containers. Kubernetes manages and scales containers.

Interview One-Liner

Docker packages applications; Kubernetes runs them reliably in production.