Kubernetes Tutorial for Beginners

What Is Kubernetes?

Kubernetes (often abbreviated as K8s) is an open-source platform used to automate the deployment, scaling, and management of applications running in containers.

Think of Kubernetes as a manager for containerized applications. If you have dozens or hundreds of containers running across multiple servers, Kubernetes helps keep everything organized and running smoothly.

Why Kubernetes?

Before Kubernetes, administrators had to manually manage servers and applications. This became difficult as applications grew larger and more complex.

Kubernetes helps by:

- Automatically deploying applications

- Scaling applications up or down based on demand
- Restarting failed containers
- Distributing traffic across multiple instances
- Managing updates with minimal downtime

What Are Containers?

Containers package an application along with everything it needs to run, including libraries and dependencies.

A popular container platform is Docker.

Benefits of containers:

- Consistent behavior across environments
- Fast startup times
- Efficient use of resources
- Easy deployment

Kubernetes Architecture

Cluster

A Kubernetes cluster is a group of machines that work together.

The cluster consists of:

- Control Plane (Master Node)
- Worker Nodes

Control Plane

The control plane manages the entire cluster.

Main components include:

API Server

The front door of Kubernetes.

Receives commands and communicates with other components.

Scheduler

Decides which worker node should run a container.

Controller Manager

Monitors the cluster and ensures the desired state is maintained.

etcd

A distributed database that stores cluster information.

Worker Nodes

Worker nodes run applications.

Each worker node contains:

Kubelet

An agent that communicates with the control plane.

Container Runtime

Software that runs containers.

Examples:

- containerd
- CRI-O

Kube Proxy

Handles networking and traffic routing.

Important Kubernetes Objects

Pod

A Pod is the smallest deployable unit in Kubernetes.

A Pod can contain:

- One container
- Multiple closely related containers

Example:

A web application running inside a container.

Deployment

A Deployment manages Pods.

Benefits:

- Automatic updates
- Rollbacks
- Scaling

Example:

Run 3 copies of a web application.

Service

A Service provides stable network access to Pods.

Without Services, Pods may receive new IP addresses whenever they restart.

Namespace

Namespaces organize resources into separate environments.

Examples:

- Development
- Testing
- Production

Installing Kubernetes

Common options include:

Minikube

Best for learning Kubernetes on a local machine.

Kind

Runs Kubernetes clusters using Docker containers.

Managed Kubernetes Services

Cloud providers offer managed Kubernetes solutions:

- Amazon EKS
- Google Kubernetes Engine (GKE)
- Azure Kubernetes Service (AKS)

Creating Your First Deployment

Example deployment file:

```
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
```

Save the file as:

nginx-deployment.yaml

Apply it:

```
kubectl apply -f nginx-deployment.yaml
```

Useful kubectl Commands

View nodes:

kubectl get nodes

View pods:

kubectl get pods

View deployments:

kubectl get deployments

Describe a pod:

kubectl describe pod <pod-name>

View logs:

kubectl logs <pod-name>

Delete a deployment:

kubectl delete deployment nginx-deployment

Scaling an Application

Increase replicas:

kubectl scale deployment nginx-deployment --replicas=5

Kubernetes will automatically create additional Pods.

Rolling Updates

Update an application image:

kubectl set image deployment/nginx-deployment nginx=nginx:1.27

Kubernetes updates Pods gradually to avoid downtime.

Self-Healing

If a Pod crashes:

1. Kubernetes detects the failure.
2. Kubernetes automatically creates a replacement Pod.
3. The application continues running.

This is one of Kubernetes' most valuable features.

Real-World Example

Imagine a shopping website:

- Frontend service
- Product catalog service
- Payment service
- Database

Kubernetes can:

- Run each service in containers
- Scale busy services automatically
- Replace failed containers
- Balance incoming traffic
- Deploy updates safely

Best Practices

1. Use Deployments instead of creating Pods manually.
2. Store configuration separately using ConfigMaps.
3. Store secrets securely using Secrets.
4. Monitor applications with tools such as Prometheus.
5. Use namespaces to separate environments.
6. Limit resource usage with CPU and memory requests.

Summary

Kubernetes is a container orchestration platform that automates the deployment, scaling, and management of containerized applications.

Key concepts to remember:

- Cluster = Group of machines
- Node = A machine in the cluster
- Pod = Smallest deployable unit
- Deployment = Manages Pods
- Service = Network access to Pods
- kubectl = Command-line management tool

Learning Kubernetes takes time, but understanding Pods, Deployments, Services, and kubectl provides a strong foundation for working with modern cloud-native applications.

This tutorial is suitable for a 10–15 minute introductory video and can be expanded with live demonstrations using Docker, Minikube, and `kubectl` commands.