# **Kubernetes Complete Notes**

## **1. Introduction to Kubernetes**

### **What is Kubernetes?**

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

### **Why Use Kubernetes?**

* Automates deployment and scaling
* Manages containerized workloads efficiently
* Provides high availability and fault tolerance
* Supports multi-cloud and hybrid cloud environments

### **Kubernetes Architecture**

* **Master Node (Control Plane)**: Manages the cluster
  + API Server
  + Controller Manager
  + Scheduler
  + etcd (Key-Value Store)
* **Worker Nodes**: Runs application containers
  + Kubelet
  + Container Runtime (Docker, CRI-O, etc.)
  + Kube Proxy

### **Installing Kubernetes (Minikube)**

#### **Linux Installation**

curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64

sudo install minikube-linux-amd64 /usr/local/bin/minikube

minikube start

#### **Windows/macOS Installation**

Download Minikube from [Minikube's official website](https://minikube.sigs.k8s.io/docs/start/).

## **2. Kubernetes Basic Commands**

### **Check Cluster Status**

kubectl cluster-info

### **List Nodes**

kubectl get nodes

### **Get Cluster Events**

kubectl get events

## **3. Kubernetes Objects**

### **Pods**

A pod is the smallest deployable unit in Kubernetes.

#### **Creating a Pod**

apiVersion: v1

kind: Pod

metadata:

name: nginx-pod

spec:

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

kubectl apply -f nginx-pod.yaml

### **List Pods**

kubectl get pods

### **Delete a Pod**

kubectl delete pod nginx-pod

## **4. Deployments**

A Deployment ensures that the desired number of pod replicas are running.

#### **Creating a Deployment**

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

ports:

- containerPort: 80

kubectl apply -f nginx-deployment.yaml

### **List Deployments**

kubectl get deployments

### **Scaling a Deployment**

kubectl scale deployment nginx-deployment --replicas=5

### **Delete a Deployment**

kubectl delete deployment nginx-deployment

## **5. Services in Kubernetes**

Kubernetes **Services** expose applications inside or outside the cluster.

### **Types of Services**

* **ClusterIP**: Accessible only within the cluster (default)
* **NodePort**: Exposes service on a static port
* **LoadBalancer**: Uses cloud provider's load balancer

#### **Creating a Service**

apiVersion: v1

kind: Service

metadata:

name: nginx-service

spec:

selector:

app: nginx

ports:

- protocol: TCP

port: 80

targetPort: 80

type: NodePort

kubectl apply -f nginx-service.yaml

### **List Services**

kubectl get services

### **Delete a Service**

kubectl delete service nginx-service

## **6. Kubernetes Networking**

### **List Cluster Networks**

kubectl get svc,po -o wide

### **Accessing a Pod from Another Pod**

kubectl exec -it <pod-name> -- /bin/sh

## **7. ConfigMaps and Secrets**

ConfigMaps store non-sensitive configuration data, while Secrets store sensitive data.

### **Creating a ConfigMap**

apiVersion: v1

kind: ConfigMap

metadata:

name: app-config

labels:

app: my-app

data:

DATABASE\_URL: mysql://db-service:3306

kubectl apply -f app-config.yaml

### **Creating a Secret**

echo -n "mypassword" | base64

apiVersion: v1

kind: Secret

metadata:

name: db-secret

type: Opaque

data:

password: bXlwYXNzd29yZA==

kubectl apply -f db-secret.yaml

## **8. Kubernetes Volumes and Persistent Storage**

Kubernetes provides **Persistent Volumes (PV)** and **Persistent Volume Claims (PVC)**.

#### **Creating a Persistent Volume**

apiVersion: v1

kind: PersistentVolume

metadata:

name: pv-volume

spec:

capacity:

storage: 1Gi

accessModes:

- ReadWriteOnce

hostPath:

path: /data/pv-storage

kubectl apply -f pv.yaml

#### **Creating a Persistent Volume Claim**

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: pvc-volume

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 500Mi

kubectl apply -f pvc.yaml

## **9. Kubernetes Ingress Controller**

Ingress is used to expose HTTP(S) services externally.

#### **Creating an Ingress Resource**

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: my-ingress

spec:

rules:

- host: example.com

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: nginx-service

port:

number: 80

kubectl apply -f ingress.yaml

## **10. Kubernetes Logging and Monitoring**

### **Viewing Logs**

kubectl logs <pod-name>

### **Monitoring Resource Usage**

kubectl top pod

## **11. Kubernetes Role-Based Access Control (RBAC)**

RBAC restricts access to Kubernetes resources.

#### **Creating a Role**

apiVersion: rbac.authorization.k8s.io/v1

kind: Role

metadata:

name: pod-reader

rules:

- apiGroups: [""]

resources: ["pods"]

verbs: ["get", "watch", "list"]

kubectl apply -f role.yaml

## **Conclusion**

Kubernetes is essential for modern containerized applications, providing **scalability, resilience, and automation**. 🚀