Kubernetes FROM SCRATCH

📘 Kubernetes From Scratch

Beginner → Practical → Production Mindset Guide

✅ Kubernetes Concepts Explained Simply
✅ Architecture + Components
✅ YAML Basics + kubectl Commands
✅ Cluster Setup (kubeadm)
✅ Networking + Services + Ingress
✅ Storage (PV/PVC)
✅ Security (RBAC + Secrets)
✅ Monitoring & Logging

📚 Table of Contents

- Services
- Namespaces
- Desired State
- Secrets
- CoreDNS
- ReplicaSet vs Deployment
- DaemonSet
- Labels & Selectors
- Annotations
- Node Affinity / Anti-Affinity
- Taints & Tolerations
- ConfigMaps
- Volumes

4️⃣ kubectl Commands + Essentials

- kubectl syntax
- Common commands list
- Vim setup for YAML
- PODS, Deployments, Services, Volumes
- Secrets, ConfigMaps, Ingress
- Scheduler, RBAC, Troubleshooting

5️⃣ Kubernetes YAML (Basics + Examples)

- YAML vs JSON
- Required YAML fields
- apiVersion and Kind reference

6️⃣ Kubernetes Installation (kubeadm on CentOS)

- Prepare nodes
- Install Docker + Kubernetes tools
- Initialize cluster
- Join worker nodes
- Deploy first Pod + Service

7️⃣ Kubernetes Namespaces

- What is namespace?
- Why namespaces matter
- Resource quotas in namespaces

8️⃣ Kubernetes Deployments

- Deployment vs ReplicationController
- Strategies (Recreate vs RollingUpdate)
- Upgrade, Rollback, History
- Scaling + NodeSelector

9️⃣ Kubernetes Networking

- Container-to-container

- Pod-to-pod
- Pod-to-service
- External-to-service
- ClusterIP vs NodePort vs LoadBalancer vs Ingress
- Ingress rules + Ingress Controller

🔟 Kubernetes Storage

- PersistentVolume (PV)
- PersistentVolumeClaim (PVC)
- NFS example
- Volume types overview

1️⃣1️⃣ Kubernetes Security

- RBAC (Roles, Bindings)
- Users vs ServiceAccounts
- Namespace-limited user example
- Secrets usage (env + mount)

1️⃣2️⃣ Kubernetes Logging & Monitoring

- Prometheus
- Sematext Docker Agent
- Kubernetes log metadata

📌 Resources

Official + learning links

1️⃣ Introduction to Kubernetes

✅ What is Kubernetes?

Kubernetes (K8s) is an **open-source container orchestration platform** originally developed by Google. It is used to manage **containerized applications** across a cluster of machines.

Kubernetes provides:

- ✅ **High availability**
- ✅ **Zero downtime deployments**
- ✅ **Auto scaling**
- ✅ **Self-healing**
- ✅ **Automatic rollbacks**

📌 Kubernetes was inspired by Google's internal systems **Borg & Omega** (used since ~2003). Google open-sourced Kubernetes in **2014**.

✅ What does orchestration do?

Orchestration is the automation of running containers in production.

It handles:

- Configuring and scheduling containers
- Provisioning and deployments
- High availability (HA)

- Application configurations
- Scaling workloads automatically
- Hardware resource allocation (CPU/RAM)
- Load balancing + service discovery
- Health monitoring
- Securing container interactions

✅ Famous Container Orchestrators

- Docker Swarm
- Apache Mesos (Mesosphere)
- Nomad
- Cloud Foundry
- Cattle
- Managed Cloud Kubernetes (Azure, AWS, GCP, IBM, Alibaba)

2️⃣ Kubernetes Architecture & Components

🧩 Kubernetes Architecture Overview

A Kubernetes cluster consists of:

✅ Master Node (Control Plane)

The master node manages the cluster and controls scheduling, monitoring, scaling, and maintaining the desired state.

✅ Worker Nodes

Worker nodes run applications in the form of **Pods** (containers).

📌 You can run **multiple master nodes** for High Availability.

🧠 Master Node Components

1) API Server (`kube-apiserver`)

The **main entry point** for all cluster operations.

It:

- receives commands from kubectl, UI, or automation
- validates requests
- coordinates everything in Kubernetes

---

2) Controller Manager (`kube-controller-manager`)

Runs control loops to maintain the desired state, such as:

- creating Pods
- scaling Pods
- healing failures

---

3) Replication Controller

Ensures the specified number of Pod replicas are running at all times.

---

4) Node Controller

Manages node health and availability in the cluster.

---

5) Scheduler (`kube-scheduler`)

Decides which worker node should run a new Pod based on:

- CPU/RAM availability
- node selectors
- taints and tolerations
- affinity / anti-affinity rules

---

6) etcd Cluster

etcd is the cluster's key-value database that stores:

- cluster state
- node info
- workloads
- configurations

✅ etcd is critical for Kubernetes.

🔌 Add-ons in Kubernetes

✅ DNS (CoreDNS)

Provides name resolution inside the cluster.

✅ Web UI (Dashboard)

A visual dashboard to manage and troubleshoot the cluster.

✅ Container Runtime

Kubernetes supports runtimes such as:

- Docker
- containerd
- CRI-O

⚙ Worker Node Components

1) kubelet

The kubelet runs on each node and:

- connects the node to the master
- ensures Pods are running
- reports node health

---

2) kube-proxy

Handles networking and routing by:

- forwarding traffic to the correct Pod
- maintaining iptables rules
- enabling Services to work

🔹 kubectl (Kubernetes CLI)

`kubectl` is the command-line tool used to send commands to Kubernetes API Server.

Every kubectl command becomes an API call to the master.

3️⃣ Kubernetes Core Concepts

🧱 Pod

A Pod is the **smallest deployable unit in Kubernetes**.

A Pod can run:
✅ One container (most common)
✅ Multiple tightly coupled containers (advanced use case)

📌 Key Pod notes:

- Every Pod gets **one IP address**
- Containers inside the same Pod share:
    - network namespace
    - IP address
    - resources
- Communication between different Pods happens via **Pod IPs**

🚀 Deployment

A Deployment provides **declarative updates** for your app.

It can:

- deploy Pods via ReplicaSets
- update and rollback versions
- scale replicas
- pause/resume deployments

🌐 Service

A Service provides a stable endpoint to access Pods.

Pods are volatile (can be recreated), but Services provide:
✅ stable IP
✅ stable DNS name
✅ load balancing across replicas

🗂️ Namespace

A Namespace is a logical partition inside a cluster.

✅ Used for:

- team isolation
- project separation
- resource quota allocation

📌 Resources inside the same namespace must be unique.

🎯 Desired State

Kubernetes works using **desired state** declared in YAML files.

Example:

- Desired: 3 replicas
- Current: 2 replicas
  ✅ Kubernetes automatically creates 1 more Pod.

🔐 Secret

Secrets store sensitive values like:

- passwords
- tokens
- API keys
- SSH keys

Secrets can be used as:
✅ environment variables
✅ mounted files

⚠ Note: Kubernetes stores secrets as **Base64 encoded**, not encrypted by default.

🧠 CoreDNS

CoreDNS provides DNS resolution inside Kubernetes clusters.

🔁 ReplicaSet & Deployment

ReplicaSet ensures a number of replicas are running.

Deployment manages ReplicaSets and supports:

- rolling updates

- rollback
- scaling

✅ ReplicaSet is part of Deployment.

📌 DaemonSet

DaemonSet ensures that a specific Pod runs on:
✅ all nodes
✅ or selected nodes

Use cases:

- log collection (e.g., fluentd, logstash)
- node monitoring (collectd)
- storage daemons (glusterd)

🏷️ Labels & Selectors

✅ Labels

Key/value pairs attached to objects:

```
stage=production
owner=ahmed
```

Example: Add labels

```
kubectl get pods --show-labels
kubectl label pod labelex owner=ahmed
kubectl get pods --show-labels
```

✅ Selectors

Select objects using labels:

```
kubectl get pods -l owner=ahmed
kubectl get pods -l stage=production
kubectl get pods -l 'stage in (production, development)'
```

📝 Annotations

Annotations store non-identifying metadata used by external tools.
✅ No selectors apply here (unlike labels).

📌 Node Affinity & Anti-Affinity

Affinity is a more powerful version of `nodeSelector`. It supports:

- flexible matching rules
- soft preferences
- pod-to-pod placement rules

🚫 Taints & Tolerations

Taints repel Pods from nodes.

Example:

```
kubectl taint nodes node1 key=value:NoSchedule
```

Remove taint:

```
kubectl taint nodes node1 key=value:NoSchedule-
```

⚙ ConfigMap

ConfigMaps store configuration like:

- env values
- configs
- ports
- args

✅ keeps configuration separated from Pod YAML.

💾 Volume

Volumes are Pod-level storage shared across containers in the Pod.

✅ Data survives container restarts
❌ Removed when the Pod is removed (unless PV/PVC used)

4️⃣ kubectl Commands (Essentials)

✅ kubectl Syntax

```
kubectl [command] [TYPE] [NAME] [flags]
```

Examples:

```
kubectl get pod pod1
kubectl get pods pod1
kubectl get po pod1
kubectl get pods
kubectl describe pod pod1
kubectl delete pod pod1
```

✅ Most Common kubectl Commands

PODs

```
kubectl get pods
kubectl get pods -A
kubectl describe pod <pod>
kubectl get pod <pod> -o wide
kubectl logs <pod>
kubectl exec -it <pod> -- sh
```

Deployments

```
kubectl get deploy
kubectl create -f deployment.yaml
kubectl apply -f deployment.yaml
kubectl rollout status deploy/<name>
kubectl rollout history deploy/<name>
kubectl rollout undo deploy/<name> --to-revision=2
```

Services

```
kubectl get svc
kubectl describe svc <svc>
kubectl expose deploy nginx --port=80 --type=NodePort
```

5️⃣ Kubernetes YAML Basics

✅ Why YAML?

YAML is a superset of JSON, but:

- easier to read

- supports comments
- uses fewer characters

✅ Required Fields in Kubernetes YAML

Every Kubernetes YAML must include:

- **apiVersion**
- **kind**
- **metadata**
- **spec**

Example Pod YAML:

```
apiVersion: v1
kind: Pod
metadata:
  name: website
  labels:
    name: web
spec:
  containers:
    - name: web-server
      image: nginx
      ports:
        - containerPort: 80
```

6️⃣ Kubernetes Installation (kubeadm on CentOS)

✅ Step 1: Prepare All Nodes

Configure `/etc/hosts`

```
vim /etc/hosts
```

Example:

```
192.168.179.133 k8s-master
192.168.179.131 node01
192.168.179.132 node02
```

Firewall Configuration (Master)

```
firewall-cmd --permanent --add-port=6443/tcp
firewall-cmd --permanent --add-port=2379-2380/tcp
firewall-cmd --permanent --add-port=10250/tcp
firewall-cmd --permanent --add-port=10251/tcp
firewall-cmd --permanent --add-port=10252/tcp
firewall-cmd --permanent --add-port=10255/tcp
firewall-cmd --reload
```

Worker nodes:

```
firewall-cmd --permanent --add-port=10251/tcp
firewall-cmd --permanent --add-port=10252/tcp
firewall-cmd --reload
```

Check firewall rules:

```
firewall-cmd --list-all
```

Enable Kernel Module + IPTables

```
cat <<EOF > /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
```

```
EOF

sysctl --system
modprobe br_netfilter
echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
```

Disable SELinux

```
setenforce 0
sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

Disable Swap

```
swapoff -a
```

Edit `/etc/fstab` and comment swap line.

✅ Install Docker

```
yum install -y yum-utils device-mapper-persistent-data
yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo
yum install -y docker-ce
systemctl enable docker
systemctl start docker
```

✅ Install Kubernetes Packages

Create repo:

```
cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
```

```
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86
_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
       https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOF
```

Install:

```
yum install -y kubelet kubeadm kubectl
systemctl enable kubelet
systemctl start kubelet
```

✅ Step 2: Initialize Master Node

```
kubeadm init --apiserver-advertise-address=10.0.15.10
--pod-network-cidr=10.244.0.0/16
```

Setup kubeconfig:

```
mkdir -p $HOME/.kube
cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
chown $(id -u):$(id -g) $HOME/.kube/config
```

Install flannel:

```
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/
kube-flannel.yml
```

Verify:

```
kubectl get nodes
kubectl get pods -A
```

✅ Step 3: Join Worker Nodes

Run this on workers (from init output):

```
kubeadm join <MASTER_IP>:6443 --token <TOKEN>
--discovery-token-ca-cert-hash sha256:<HASH>
```

Verify in master:

```
kubectl get nodes
kubectl get pods -A
```