Kubernetes

🔧 What is Kubernetes?

Kubernetes (K8s) is an open-source **container orchestration platform**.

It helps **automate deployment, scaling, and management** of containerized applications (like Docker containers).

🧠 Basic Terminology (with analogies)

Term	Explanation	Analogy
Cluster	A set of machines (nodes) that run your applications.	A company with many computers (employees) working together.
Node	A single machine (VM or physical). Can be Master or Worker.	An employee in a company.
Pod	The smallest unit in Kubernetes. A pod contains one or more containers that run together.	A team (1-2 members) working on a task.
Container	Lightweight, isolated app environment (e.g., Docker).	An app running inside a portable box.
Deployment	Describes the desired state (e.g., 3 pods running your app). Kubernetes maintains this state.	Telling HR: "I need 3 people working on this task at all times."
Service	A stable endpoint (IP/port) to access pods. Handles networking/load balancing.	A receptionist who always connects you to the right team member.
Namespace	A way to divide cluster resources between multiple teams/projects.	Separate departments in a company (HR, IT, Marketing).
ReplicaSet	Ensures a specified number of pod replicas are running at all times.	A manager ensuring there are always 3 workers on a task.
ConfigMap / Secret	Store configuration and sensitive info (passwords, URLs).	A secure locker or file cabinet with instructions.
Volume	Used for persistent storage (outside container life).	A shared drive or hard disk.

🏗️ Kubernetes Architecture Overview

```
              +-------------------------+
              |      Master Node        |
              |-------------------------|
              | - API Server            |
              | - Scheduler             |
              | - Controller Manager    |
              | - etcd (key-value store)|
              +-----------+-------------+
                          |
       +------------------|----------------------+
       |                  |                  |
+---------v---------+  +---------v---------+  +---------v---------+
|   Worker Node     | |   Worker Node     | |   Worker Node     |
|-------------------|  |-------------------|  |-------------------|
| - Kubelet         | | - Kubelet         | | - Kubelet         |
| - Kube Proxy      | | - Kube Proxy      | | - Kube Proxy      |
| - Container Runtime| | - Container Runtime| | - Container Runtime|
| - Pods            || - Pods            || - Pods            |
+-------------------+  +-------------------+  +-------------------+
```

💡 Key Components Explained

Component	Role
API Server	Front door of the cluster. You (or `kubectl`) interact with this.
Scheduler	Decides which pod goes to which node.
Controller Manager	Maintains the desired state (replicas, rollout, etc.).
etcd	Key-value store that stores cluster state.
Kubelet	Agent running on each node that talks to API server and runs containers.
Kube Proxy	Manages networking (routing requests to the correct pod).

Container Runtime	The software that actually runs containers (e.g., Docker, containerd).

⚙️ Real Life Analogy

Imagine you're running a restaurant chain 🍔:

- **Master Node** = Manager who takes orders and assigns tasks.
- **Worker Node** = Cooks who prepare the dishes.
- **Pods** = Stations where food is prepared (burger station, fries station).
- **Services** = Waiters who serve customers, always pointing to the correct station.
- **Deployment** = You instruct that there must always be 2 burger stations open.

🚀 Why Use Kubernetes?

✅ Auto-healing (restarts crashed pods)

✅ Load balancing & service discovery

✅ Easy scaling (1 to 100 pods in seconds)

✅ Rollbacks & rolling updates

✅ Declarative infrastructure (YAML-based)

🧱 1. Basic Cluster Commands

Command	Description
`kubectl version`	Show client and server version
`kubectl cluster-info`	Display cluster info
`kubectl config view`	Show current kubeconfig
`kubectl get nodes`	List all cluster nodes
`kubectl describe node <node-name>`	Detailed info about a node

📦 2. Working with Pods

Command	Example	Description
`kubectl get pods`	List all pods in current namespace	
`kubectl get pods -A`	List all pods across all namespaces	
`kubectl describe pod mypod`	Detailed info about a pod	
`kubectl logs mypod`	Get logs from a pod	
`kubectl logs -f mypod`	Follow pod logs	
`kubectl exec -it mypod -- bash`	SSH into a pod (if shell exists)	
`kubectl delete pod mypod`	Delete a pod manually	

📁 3. Deployments & ReplicaSets

Command	Example	Description
`kubectl create deployment nginx --image=nginx`	Create a deployment	
`kubectl get deployments`	List deployments	
`kubectl describe deployment nginx`	Deployment details	
`kubectl scale deployment nginx --replicas=3`	Scale a deployment	
`kubectl delete deployment nginx`	Delete a deployment	

⚙️ 4. Services

Command	Example	Description
`kubectl expose deployment nginx --port=80 --type=NodePort`	Expose deployment as a service	
`kubectl get svc`	List all services	
`kubectl describe svc nginx`	Service details	
`kubectl delete svc nginx`	Delete a service	

📁 5. Apply/Manage YAML Files

Command	Example	Description
`kubectl apply -f myapp.yaml`	Apply/create resource from YAML	
`kubectl create -f myapp.yaml`	Same as above (older style)	
`kubectl delete -f myapp.yaml`	Delete resources defined in YAML	

`kubectl diff -f myapp.yaml`	See diff before applying	
`kubectl get -f myapp.yaml`	Get resource defined in YAML	

🛡️ 6. ConfigMaps & Secrets

Command	Example	Description
`kubectl create configmap myconfig --from-literal=key1=value1`	Create ConfigMap from CLI	
`kubectl create secret generic mysecret --from-literal=password=1234`	Create secret	
`kubectl get configmap`	List configmaps	
`kubectl get secret`	List secrets	
`kubectl describe configmap myconfig`	ConfigMap details	
`kubectl describe secret mysecret`	Secret details (base64 encoded)	

🚀 7. Namespaces

Command	Example	Description
`kubectl get namespaces`	List all namespaces	
`kubectl create namespace dev`	Create a new namespace	
`kubectl delete namespace dev`	Delete a namespace	
`kubectl get pods -n dev`	Get pods in specific namespace	
`kubectl config set-context --current --namespace=dev`	Set default namespace	

🔄 8. Rollouts & Updates

Command	Example	Description
`kubectl rollout status deployment nginx`	Check rollout status	
`kubectl rollout restart deployment nginx`	Restart a deployment	
`kubectl rollout undo deployment nginx`	Rollback deployment	

📊 9. Monitoring & Debugging

Command	Example	Description
`kubectl top nodes`	CPU/memory usage of nodes	
`kubectl top pods`	CPU/memory usage of pods	
`kubectl describe <resource> <name>`	Debug any resource	
`kubectl events` (with plugin)	View recent cluster events	

🧹 10. Clean Up & Utilities

Command	Example	Description
`kubectl delete all --all`	Delete all resources in namespace	
`kubectl get all`	Get all resources (pods, services, deployments)	
`kubectl explain pod`	Show detailed structure and docs for a resource	
`kubectl port-forward pod/mypod 8080:80`	Access pod locally on port 8080	

Bonus: YAML Generator Shortcut

```
bash
CopyEdit
kubectl create deployment myapp --image=myimage --dry-run=client -o y
aml > myapp.yaml
```

📝 Creates a YAML file you can edit and apply!

Want this list as a **cheat sheet PDF**, or want to practice these commands in a Minikube or Docker Desktop setup? I can guide you step-by-step.

Deploying Spring Boot:

1. **Dockerize your Spring Boot app**

2. **Create Deployment YAML**
3. **Create Service YAML**
4. **Apply to Kubernetes**

🔧 Step 1: Dockerize Your Spring Boot App

Make sure you have a `Dockerfile` like this:

```
# Use a base image with Java
FROM openjdk:17-jdk-slim

# Copy your JAR file into the container
COPY target/myapp.jar app.jar

# Run the app
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Then build and push the Docker image:

```
docker build -t your-dockerhub-username/springboot-app:latest .
docker push your-dockerhub-username/springboot-app:latest
```

📦 Step 2: Deployment YAML (`springboot-deployment.yaml`)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: springboot-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: springboot-app
  template:
    metadata:
      labels:
        app: springboot-app
    spec:
      containers:
      - name: springboot-app
        image: your-dockerhub-username/springboot-app:latest
        ports:
        - containerPort: 8080
```

🌐 Step 3: Service YAML (`springboot-service.yaml`)

```
apiVersion: v1
kind: Service
metadata:
  name: springboot-service
spec:
  selector:
    app: springboot-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
  type: LoadBalancer   # or NodePort if you're on Minikube
```

🚀 Step 4: Apply to Kubernetes

```
kubectl apply -f springboot-deployment.yaml
kubectl apply -f springboot-service.yaml
```

```

```

Check everything is running:

```
kubectl get pods
kubectl get svc
```

💡 Accessing the App

- **If you're using Minikube:**

  ```
  minikube service springboot-service
  ```

- **If on Cloud (EKS, GKE, etc):**
  - Use the **External IP** shown in `kubectl get svc`
