Kubernetes Basics with MERN Example

1. What is Kubernetes?

- **Container Orchestration Platform** → manages Docker containers.
- Handles deployment, scaling, networking, and self-healing.
- Works across clusters of machines.

Why Needed? - Docker runs containers, but scaling & reliability = manual. - Kubernetes automates scaling, restarts crashed containers, and provides stable networking.

2. Kubernetes Architecture (Simple)

- **Control Plane (Master):** API Server, Scheduler, Controller Manager, etcd.
- **Worker Nodes:** Run actual Pods (containers).
- **kubectl:** CLI to interact with cluster.

Analogy: - Control Plane = Air traffic control tower ✈️- Worker Nodes = Runways - Pods = Planes (carry containers)

3. Core Kubernetes Objects

- **Pod:** Smallest deployable unit (wraps one/more containers).
- **Deployment:** Manages Pods (replicas, rolling updates).
- **Service:** Provides stable networking & load balancing.

Flow:

```
Pod → Deployment → Service
```

4. Install Tools (Local Practice)

```
# Install kubectl (CLI)
choco install kubernetes-cli

# Install Minikube (local cluster)
choco install minikube
```

```
# Start cluster
minikube start --driver=docker

# Verify
kubectl get nodes
```

---

5. Prepare MERN Images

```
# Build images locally
docker build -t mern-backend:dev ./server
docker build -t mern-frontend:dev ./client

# Load into Minikube
minikube image load mern-backend:dev
minikube image load mern-frontend:dev
```

---

6. Deploy Backend (Node API)

- **Pod (imperative):**

  ```
  kubectl run backend-pod
    --image=mern-backend:dev
    --port=5000
    --env="MONGODB_URI=<atlas-uri>"
  ```

- **Check:**

  ```
  kubectl get pods
  kubectl logs backend-pod
  ```

- **Expose Pod (temporary):**

  ```
  kubectl expose pod backend-pod --type=NodePort --port=5000
  minikube service backend-pod
  ```

---

7. Backend Deployment

```
kubectl create deployment backend --image=mern-backend:dev

# Add env
kubectl set env deployment/backend MONGODB_URI="<atlas-uri>"

# Scale to 2 pods
kubectl scale deployment backend --replicas=2
```

- **Port Forward for Testing:**

  ```
  kubectl port-forward deployment/backend 5000:5000
  ```

---

8. Backend Service

```
kubectl expose deployment backend
  --name=backend-svc
  --port=5000
  --target-port=5000
  --type=ClusterIP
```

- **Use DNS inside cluster:**

```
http://backend-svc:5000/api
```

---

9. Frontend Deployment

```
kubectl create deployment frontend --image=mern-frontend:dev

kubectl set env deployment/frontend VITE_API_URL=http://backend-svc:5000/api
```

- **Expose as NodePort:**

  ```
  kubectl expose deployment frontend
    --name=frontend-svc
    --port=80
  ```

```
  --target-port=80
  --type=NodePort
```

- **Open in browser:**

```
minikube service frontend-svc
```

---

10. Summary Flow (MERN in K8s)

1. **Backend Deployment** → Pods running Node API.
2. **Backend Service (ClusterIP)** → Stable DNS inside cluster.
3. **Frontend Deployment** → React app Pods.
4. **Frontend Service (NodePort)** → Accessible in browser.

Result: MERN app runs fully in Kubernetes with scaling, self-healing, and stable networking.