Kubernetes Project — NodeJS + MongoDB

=======================================

===========================================================

🪜 Steps to Setup and Deploy

-----------------------------------------------------------

1) Launch EC2

- Use Ubuntu 22.04(recommended)

- instance type t3.medium

- ppk key

- all-tcp security group

- 30gb, gp2

-----------------------------------------------------------

2) Install Docker

------------------

sudo apt-get update -y &&

sudo apt-get install -y curl apt-transport-https ca-certificates conntrack &&

curl -fsSL https://get.docker.com | sh &&

sudo usermod -aG docker $USER &&

newgrp docker

-----------------------------------------------------------

3) Install kubectl

-------------------

curl -LO "https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl" &&

chmod +x kubectl &&

sudo mv kubectl /usr/local/bin/

-----------------------------------------------------------

4) Install Kind

----------------

curl -Lo ./kind https://kind.sigs.k8s.io/dl/v0.23.0/kind-linux-amd64 &&

chmod +x ./kind &&

sudo mv ./kind /usr/local/bin/kind

-----------------------------------------------------------

5) Create Kind cluster with port mapping

------------------------------------------

cat <<EOF > kind-config.yaml

kind: Cluster

apiVersion: kind.x-k8s.io/v1alpha4

nodes:

- role: control-plane

extraPortMappings:

- containerPort: 30080

hostPort: 30080

protocol: TCP

EOF

kind create cluster --config kind-config.yaml

kubectl get nodes

-----------------------------------------------------------

6) Deploy Mongo + Employee App

----------------------------------

vi k8s-employee-mongo.yaml

Paste this YAML:

-------------------

apiVersion: apps/v1

kind: Deployment

metadata:

name: mongo

spec:

replicas: 1

selector:

matchLabels:

app: mongo

template:

metadata:

labels:

app: mongo

spec:

containers:

- name: mongo

image: mongo:6

ports:

- containerPort: 27017

---

apiVersion: v1

kind: Service

metadata:

name: mongo

spec:

selector:

app: mongo

ports:

- port: 27017

targetPort: 27017

protocol: TCP

type: ClusterIP

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: employee

spec:

replicas: 1

selector:

matchLabels:

app: employee

template:

metadata:

labels:

app: employee

spec:

containers:

- name: employee

image: devopsedu/employee

ports:

- containerPort: 8888

env:

- name: DB\_HOST

value: "mongo"

- name: MONGO\_URL

value: "mongodb://mongo:27017"

---

apiVersion: v1

kind: Service

metadata:

name: employee-service

spec:

selector:

app: employee

ports:

- port: 8888

targetPort: 8888

nodePort: 30080

protocol: TCP

type: NodePort

-----------------------------------------------------------

Save and apply:

-----------------

kubectl apply -f k8s-employee-mongo.yaml

kubectl get pods,svc

-----------------------------------------------------------

7) Open the App in Browser(Final Step)

---------------------------------------

http://<EC2\_PUBLIC\_IP>:30080/

-----------------------------------------------------------

8) Test the App

- Click Add Employee → enter details

- Go to Get Employee → confirm record is visible

===========================================================

Deletion Step (Proper Manner)

-----------------------------------------------------------

1) Delete Kubernetes resources

---------------------------------

kubectl delete -f k8s-employee-mongo.yaml

2) Verify

-----------

kubectl get pods,svc

3) Delete Kind cluster

-------------------------

kind delete cluster

4) Remove config files (optional)

----------------------------------

rm -f kind-config.yaml k8s-employee-mongo.yaml

5) Remove binaries (optional)

---------------------------------

sudo rm -f /usr/local/bin/kubectl /usr/local/bin/kind

6) Remove Docker containers/images (optional, full reset)

------------------------------------------------------------

docker ps -a &&

docker images &&

docker rm -f $(docker ps -aq) &&

docker rmi -f $(docker images -q)