**Steps for deploying the application on the Kubeadm**

Create ec2 instance 2

t3 medium

ubuntu os

security - all tcp all traffic...anywhere

in advance setting copy paste below command

*#!/bin/bash*

*#Master*

*# Update and upgrade Ubuntu packages*

*echo "Updating and upgrading Ubuntu packages..."*

*sudo apt-get update -y*

*sudo apt-get upgrade -y*

*# Disable swap*

*echo "Disabling swap..."*

*sudo swapoff -a*

*sudo sed -i '/ swap / s/^\(.\*\)$/#\1/g' /etc/fstab*

*# Add Kernel Parameters*

*echo "Adding kernel parameters..."*

*sudo tee /etc/modules-load.d/containerd.conf <<EOF*

*overlay*

*br\_netfilter*

*EOF*

*sudo modprobe overlay*

*sudo modprobe br\_netfilter*

*sudo tee /etc/sysctl.d/kubernetes.conf <<EOF*

*net.bridge.bridge-nf-call-ip6tables = 1*

*net.bridge.bridge-nf-call-iptables = 1*

*net.ipv4.ip\_forward = 1*

*EOF*

*sudo sysctl --system*

*# Install Containerd Runtime*

*echo "Installing Containerd runtime..."*

*sudo apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates*

*echo "Adding Docker's GPG key and repository..."*

*sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/docker.gpg*

*sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"*

*sudo apt update*

*sudo apt install -y containerd.io*

*echo "Configuring Containerd..."*

*containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1*

*sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml*

*echo "Restarting and enabling Containerd..."*

*sudo systemctl restart containerd*

*sudo systemctl enable containerd*

*# Install Docker*

*echo "Installing Docker..."*

*sudo apt update*

*# sudo apt install -y docker-ce docker-ce-cli*

*sudo apt install -y apt-transport-https ca-certificates curl software-properties-common*

*curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -*

*sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"*

*sudo apt update*

*sudo apt install -y docker-ce docker-ce-cli*

*sudo usermod -aG docker $USER*

*sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose*

*sudo chmod 777 /var/run/docker.sock*

*echo "Starting and enabling Docker..."*

*sudo systemctl start docker*

*sudo systemctl enable docker*

*# Add the Kubernetes signing key and repository*

*echo "Adding Kubernetes signing key and repository..."*

*sudo apt-get update -y*

*sudo apt-get install -y apt-transport-https ca-certificates curl gpg*

*curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg*

*echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list*

*# Update the package list and install kubelet, kubeadm, and kubectl*

*echo "Updating package list and installing kubelet, kubeadm, and kubectl..."*

*sudo apt-get update -y*

*sudo apt-get install -y kubelet kubeadm kubectl*

*sudo apt-mark hold kubelet kubeadm kubectl*

*echo "Kubernetes installation script completed successfully!"*

………………………………………………………………………………………………………………

**On master node terminal**

*sudo hostnamectl hostname Master*

*sudo kubeadm init --pod-network-cidr=192.168.0.0/16*

*mkdir -p $HOME/.kube*

*sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config*

*sudo chown $(id -u):$(id -g) $HOME/.kube/config*

*kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/manifests/calico.yaml*

#Do not install Calico now it is supposed to be installed by the error which I was facing install it after setting everything up and after applying -

*kubectl apply -f .*

*kubeadm token create --print-join-command*

on worker node

*sudo (generated token paste)*

now kubeadm installation has been done.........

……………………………………………………………………………………………………………………..

*kubectl get nodes* - fire this command on master node

*mkdir backend*

*mkdir frontend*

*mkdir k8s*

*cd frontend*

*vim default.conf*

*server {*

*listen 80;*

*server\_name \_;*

*# Serve Angular application*

*root /usr/share/nginx/html;*

*index index.html;*

*location / {*

*try\_files $uri $uri/ /index.html;*

*}*

*# Proxy API requests to Spring Boot service*

*location /api {*

*proxy\_pass http://springboot-service:8080;*

*proxy\_set\_header Host $host;*

*proxy\_set\_header X-Real-IP $remote\_addr;*

*proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;*

*proxy\_set\_header X-Forwarded-Proto $scheme;*

*}*

*# Error handling for Angular routes*

*error\_page 404 /index.html;*

*location = /index.html {*

*allow all;*

*}*

*}*

……………………………………………………………………………………………..

*docker login -u spquantum*

*….password*

**For frontend**

*vim Dockerfile*

*FROM nginx:alpine*

*COPY dist/ /usr/share/nginx/html*

*RUN chmod -R 755 /usr/share/nginx/html*

*COPY default.conf /etc/nginx/conf.d/default.conf*

*EXPOSE 80*

*CMD ["nginx", "-g", "daemon off;"]*

………………………………………………………………………………………………

*docker build -t spquantum/frontend:v11 .*

*docker push -t spquantum/frontend:v11*

*cd dist*

*cd config*

*vim Config.js*

Comment the development line and then edit the ip and add the ip of the master node also add *https://....*

**For Backend**

*vim Dockerfile*

*FROM eclipse-temurin:23-jdk-alpine*

*# Set the*

*WORKDIR /app*

*# Copy the jar file into the container*

*COPY QuantomSoft-0.0.1-SNAPSHOT\_4.jar QuantomSoft-0.0.1-SNAPSHOT\_4.jar*

*# Make the port available to the outside world*

*EXPOSE 8080*

*# Run the jar file*

*CMD ["java", "-jar", "QuantomSoft-0.0.1-SNAPSHOT\_4.jar"]*

………………………………………………………………………………………………………

*docker build -t spquantum/backend:v11 .*

*docker push spquantum/backend:v11*

*cd k8s*

*vim frontend-deployment.yml*

*apiVersion: v1*

*kind: Service*

*metadata:*

*name: frontend-service*

*spec:*

*selector:*

*app: frontend*

*ports:*

*- protocol: TCP*

*port: 80*

*targetPort: 80*

*type: NodePort*

*---*

*apiVersion: apps/v1*

*kind: Deployment*

*metadata:*

*name: frontend-deployment*

*spec:*

*replicas: 1*

*selector:*

*matchLabels:*

*app: frontend*

*template:*

*metadata:*

*labels:*

*app: frontend*

*spec:*

*containers:*

*- name: frontend*

*image: spquantum/frontend:v11*

*imagePullPolicy: Always*

*ports:*

*- containerPort: 80*

…………………………………………………………………………………………………………

*vim backend-deployment.yml*

*apiVersion: v1  
kind: Service  
metadata:  
  name: springboot-service  
spec:  
  selector:  
    app: backend  
  ports:  
    - protocol: TCP  
      port: 8080  
      targetPort: 8080  
---  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: backend-deployment  
spec:  
  replicas: 1  
  selector:  
    matchLabels:  
      app: backend  
  template:  
    metadata:  
      labels:  
        app: backend  
    spec:  
      containers:  
        - name: backend  
          image: spquantum/backend:v11*

*imagePullPolicy: Always  
          ports:  
            - containerPort: 8080  
          env:  
            - name: MYSQL\_HOST  
              value: "mysql-service"  
            - name: MYSQL\_PORT  
              value: "3306"  
            - name: MYSQL\_USER  
              value: "root"  
            - name: MYSQL\_PASSWORD  
              value: "root"*

*……………………………………………………………………………………………………….*

*vim mysql-deployment.yml*

*apiVersion: v1*

*kind: Service*

*metadata:*

*name: mysql-service*

*spec:*

*selector:*

*app: mysql*

*ports:*

*- port: 3306*

*targetPort: 3306*

*type: ClusterIP # Changed from headless to default ClusterIP*

*---*

*apiVersion: apps/v1*

*kind: Deployment*

*metadata:*

*name: mysql*

*spec:*

*replicas: 1*

*selector:*

*matchLabels:*

*app: mysql*

*template:*

*metadata:*

*labels:*

*app: mysql*

*spec:*

*containers:*

*- name: mysql*

*image: mysql:8*

*ports:*

*- containerPort: 3306*

*env:*

*- name: MYSQL\_ROOT\_PASSWORD*

*value: "root"*

*- name: MYSQL\_DATABASE*

*value: "quantumsoft"*

………………………………………………………………………………………………………………

*kubectl apply -f .*

*kubectl get pods*

The pods will be in the “pending state” as there is no calico configured in master node/control panel

Now run the calico command

*kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/manifests/calico.yaml*

then again

*kubectl delete all –all*

*kubectl apply -f .*

*kubectl get pods*

*kubectl get svc*

*kubectl rollout restart deployment* – pods restart karnyasathi

*kubectl get deployment*

on browser paste master node ip : with frontend node port

*kubectl exec -it (pod id) -- /bin/bash* – specific pods cha bash open honyasathi

*mysql -u root -p - mysql* database madhe janyasathi

*password – root*

*show databases;*

*use (database name);*

*show tables;*

*select \* from (table name);*

*describe admin;*

notes:

pod crash zalyas tyacha name n id copy karne

*kubectl logs (pod name&id) –*

*kubectl get svc – kontya service*

*Your kubeadm init command failed because the previous cluster components are* ***still running or weren't cleaned up properly****. The reset wasn't complete.*

*Here’s how to* ***fully clean up your master node*** *before running kubeadm init again:*

***✅ Step-by-Step: Fix "Port in Use" and Manifest File Errors***

***1. Stop kubelet service***

*sudo systemctl stop kubelet*

***2. Manually remove leftover Kubernetes components***

*Delete manifest files:*

*sudo rm -rf /etc/kubernetes/manifests/\**

*Kill any processes still using the required ports:*

*sudo lsof -i :6443*

*sudo lsof -i :10250*

*sudo lsof -i :10257*

*sudo lsof -i :10259*

*sudo lsof -i :2379*

*sudo lsof -i :2380*

*Kill them if running (example):*

*sudo kill -9 <PID>*

***3. Remove etcd data and Kubernetes config***

*sudo rm -rf /var/lib/etcd*

*sudo rm -rf /etc/kubernetes*

*sudo rm -rf ~/.kube*

***4. (Optional but Recommended) Reset Docker or Containerd containers***

*If you're using Docker:*

*sudo docker ps -a # View containers*

*sudo docker stop $(sudo docker ps -aq)*

*sudo docker rm $(sudo docker ps -aq)*

*sudo docker system prune -af --volumes*

*If using containerd:*

*sudo crictl pods*

*sudo crictl stopp $(sudo crictl pods -q)*

*sudo crictl rmp $(sudo crictl pods -q)*

***5. Run kubeadm reset again to be safe***

*sudo kubeadm reset -f*

***6. Now re-initialize the master node***

*sudo kubeadm init --pod-network-cidr=192.168.0.0/16*

*After this, don't forget:*

*mkdir -p $HOME/.kube*

*sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config*

*sudo chown $(id -u):$(id -g) $HOME/.kube/config*

*Then reapply Calico or your chosen CNI plugin.*

*Let me know if you're using Docker or containerd — I can tailor the cleanup more precisely.*