Steps that needs to be followed to complete the Task:

1. Install Docker

- sudo apt update
- sudo apt install docker.io
- sudo systemctl start docker
- sudo systemctl enable docker

2. Install Kubectl

- curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
- curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"

- sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
- chmod +x kubectl
- mkdir -p ~/.local/bin
- mv ./kubectl ~/.local/bin/kubectl
- kubectl version --client

3. Install Minikube

Once both Docker and Kubectl are installed, you may use the following set of commands to install Minikube.

- $ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
- $ sudo install minikube-linux-amd64 /usr/local/bin/minikube

4. Minikube start with Docker driver

- Minikube start –driver=docker

5. Install Nginx Ingress Controller

- minikube addons enable ingress
- kubectl get pods -n ingress-nginx

6. Install Helm v3

- $ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
- $ chmod 700 get_helm.sh
- $ ./get_helm.sh

6. Find out demo node js app for Hello World and create a Dockerfile for it. Dockerfile is on Github repo.

- Build Docker image and Push it into Dockerhub.

- **docker build -t node-hello-world .** : Builds a Docker image named node-hello-world from the current directory (.) using the Dockerfile located there.

- **docker tag node-hello-world trushal123/k8s:node-hello-world2**: Tags the local Docker image node-hello-world with a repository name and tag (trushal123/k8s:node-hello-world2).

- **docker push trushal123/k8s:node-hello-world2**: Pushes the tagged Docker image to a Docker registry (like Docker Hub) under the specified repository name and tag (trushal123/k8s:node-hello-world2). This makes the image accessible to others.

7. Create Dockerhub secret on Minikube so that it can be able to pull docker image

- kubectl create secret docker-registry dockerhub-secret \
  --docker-server=https://index.docker.io/v1/ \
  --docker-username=<your-dockerhub-username> \
  --docker-password=<your-dockerhub-password> \
  --docker-email=<your-dockerhub-email>

8. Create helm chart for nodejs-app

- **helm create nodejs-app**: Generates a new Helm chart named nodejs-app.
- **helm template nodejs-app ./nodejs-app**: Renders Kubernetes manifest files without deploying.
- **helm install nodejs-app ./nodejs-app**: Installs the Helm chart, deploying the application to Kubernetes.
- **helm upgrade nodejs-app ./nodejs-app**: Upgrades the deployed Helm release with changes from the chart.
- **helm ls -a**: Lists all Helm releases, including historical ones.
- **helm history nodejs-app**: Shows the revision history of a Helm release.
- **helm rollback nodejs-app 3**: Rolls back the Helm release to a specific revision.

- Make necessary changes into values.yaml file and then follow these commands to Install, Upgrade, Uninstall, Rollback release.

9. Run below mentioned commands to check pods, services, deployment and Ingress.

- Kubectl get pods
- Kubectl get services
- Kubectl get ingress
- Kubectl get deployments

Try curl http:///(ingress url) to access the application locally.

Note: As a Service in Kubernetes I have used ClusterIP and the application is accessible via Ingress Controller. I have deployed 2 nodejs “hello world” applications.

Both applications are exposed on 3000 ports and the host system port assigned is 80. Able to get output from both the applications using Ingress URL. Image pull policy i kept on Always, it can be changed as per the project requirement.

To Adapt this deployment on Production, there are a couple of things that needs to be taken care of like exposing all APIs on Cluster IP, and attaching SSL/TLS certificates with the Ingress controller to enhance security.

Here I have deployed 2 apps using host based routing. For production we can implement path based routing as well for microservices APIs.