**Create Git Repository**

**Clone the repository**

**Run frontend & backend on the local system**

For frontend : run command : **npm install**  
 **npm run**

For backend: **run node index.js**; after applying this command backend is not running due to database connection error.  
 So we need to follow below steps:  
 step: 1 : Goto config.env file and change the “ATLAS URL” and connect a database with it.  
 step: 2: In config.env expose the port 5000.  
 step: 3: go in index.js file and in line number 3; change port number, and save it.  
 step: 4: run command again node index.js; this tym backend is running successfully on port 5000.

**Now our application is running fine on local.**

**Build docker image for frontend & backend**

Run command to create docker image : **docker build –t <your\_image\_name> .**

**Image is successfully created for frontend & backend.**

**Push docker images from docker desktop to docker hub**

Run command: **docker tag <your\_image\_name> <dockerhub\_username>/<repository\_name>:<tag>  
 docker login  
 docker push <dockerhub\_username>/<repository\_name>:<tag>**

**Image is successfully pushed from docker desktop to docker hub for frontend & backend.**

**Deployment on Kubernetes through Minikube**

Create a folder “Kubernetes/k8” in frontend & backend.

Create **deployment.yaml** and **service.yaml** files in the both folders.

Adjust all the necessary details in the deployment.yaml file:

* In metadata, mention the name:
* In spec, add replicas (if you need)
* In templetes, mention the app:
* In specs, mention the name
* In spec, mention the image: (put your docker imager name from docker hub)
* In port, mention the container port (your image running port)

**In frontend deployment file need to do an extra change we have to increase the limit of CPU.**

Adjust all the necessary details in the service.yaml file:

* In metadata, mention the name:
* In specs, mention the name
* In ports, add a “targetPort” heading and mention the port.

**Now modification of the files is completed.**

Open the powershell and check status of minkube:

* Run command: **minikube status**
* if minikube is not running then, we have to run the minikube use command: **minikube start**
* After opening the minikube run command: **minikube dashboard**, with this command minikube browser dashboard is opend.
* Check kubectl status.
* Now we have to run **kubectl apply -f <file name>** ; it is used to apply Kubernetes resource configuration from a file. (e.g. kubectl apply –f deployment.yaml)
* We have to run above command for deployment.yaml & service.yaml file for frontend and backend
* After applying the above command we have to check our files configure or not properly so we need to run command: **kubectl get all**
* To check the pods run command: **kubectl get pod**
* To check the svc run command: **kubectl get svc**
* If we got any error while configuration run command: **kubectl logs**

**Creation of HELM chart to streamline the deployment process**

Create a **folder on root** with any name <learner-chart>

Run command: **helm create <file name>** (create file for frontend and backend)

After creation the file for backend we have to make some changes in it.  
 goto values.yml file and do as below:

* Step: 1 : Add replicas count
* Step: 2 : In images --- repository add the docker image name.
* Step: 3 : In images --- tag mention the version of the image.
* Step: 4 : In service --- add a new heading target port and define the port number “5000”.
* Step: 5 : In resources --- remove the curli braces, and uncomment the from “limits to memory:”, And comment from line number 75 to 82.

Now In templetes—deploymnent.yaml

* Step:1 : In line 41 “Container port” instead of .Values.service.Port to .Values.service.targetPort.

Now In templetes—service.yaml

* Step:1 : In line 11 “targetPort” instead of .Values.service.Port to .Values.service.targetPort.

After creation the file for frontend we have to make some changes in it.  
 goto values.yml file and do as below:

* Step: 1 : Add replicas count
* Step: 2 : In images --- repository add the docker image name.
* Step: 3 : In images --- tag mention the version of the image.
* Step: 4 : In service --- add a new heading target port and define the port number “3000”.
* Step: 5 : In resources --- remove the curli braces, and uncomment the from “limits to memory and make CPU 1000mi and memory 512mi”, And comment from line number 75 to 82.

Now In templetes—deploymnent.yaml

* Step:1 : In line 41 “Container port” instead of .Values.service.Port to .Values.service.targetPort.

Now In templetes—service.yaml

* Step:1 : In line 11 “targetPort” instead of .Values.service.Port to .Values.service.targetPort.

After save the all the above changes.

Now make the packages; run command: **helm package <chart name>** (frontend & backend), we have to create packages for both.

After creation of the packages need to install; run command : **helm install <chart name>** (e.g. helm install backend-chart ./backend-chart).

**Jenkins Pipeline:**

* Set up a Jenkins pipeline on your localhost.
* Configure the pipeline to build, test, and deploy the frontend and backend services.

**Deployment on AWS EKS:**

* Use EKSCTL to create an AWS EKS cluster.
* Deploy the Helm charts on the AWS EKS cluster using the Jenkins pipeline.