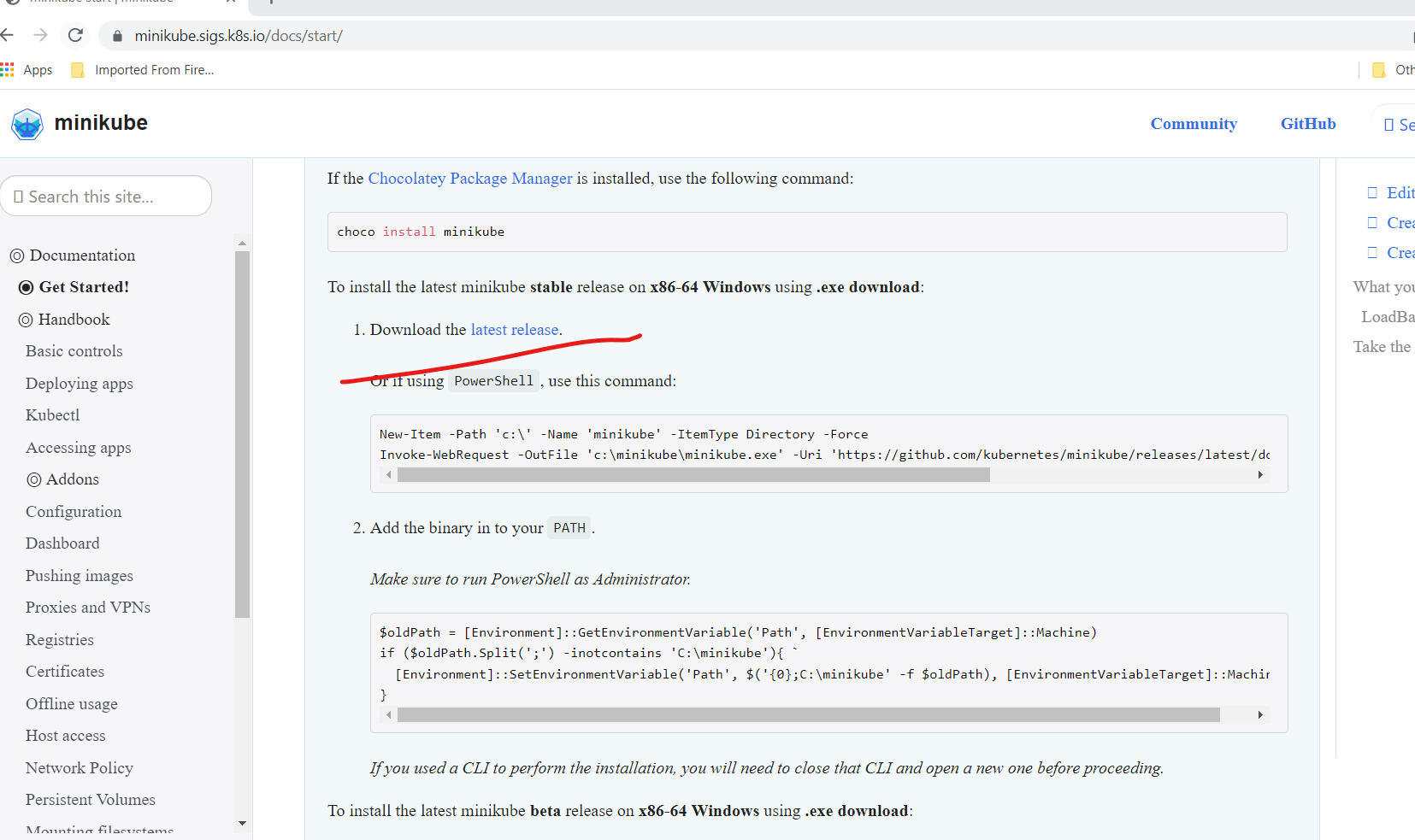
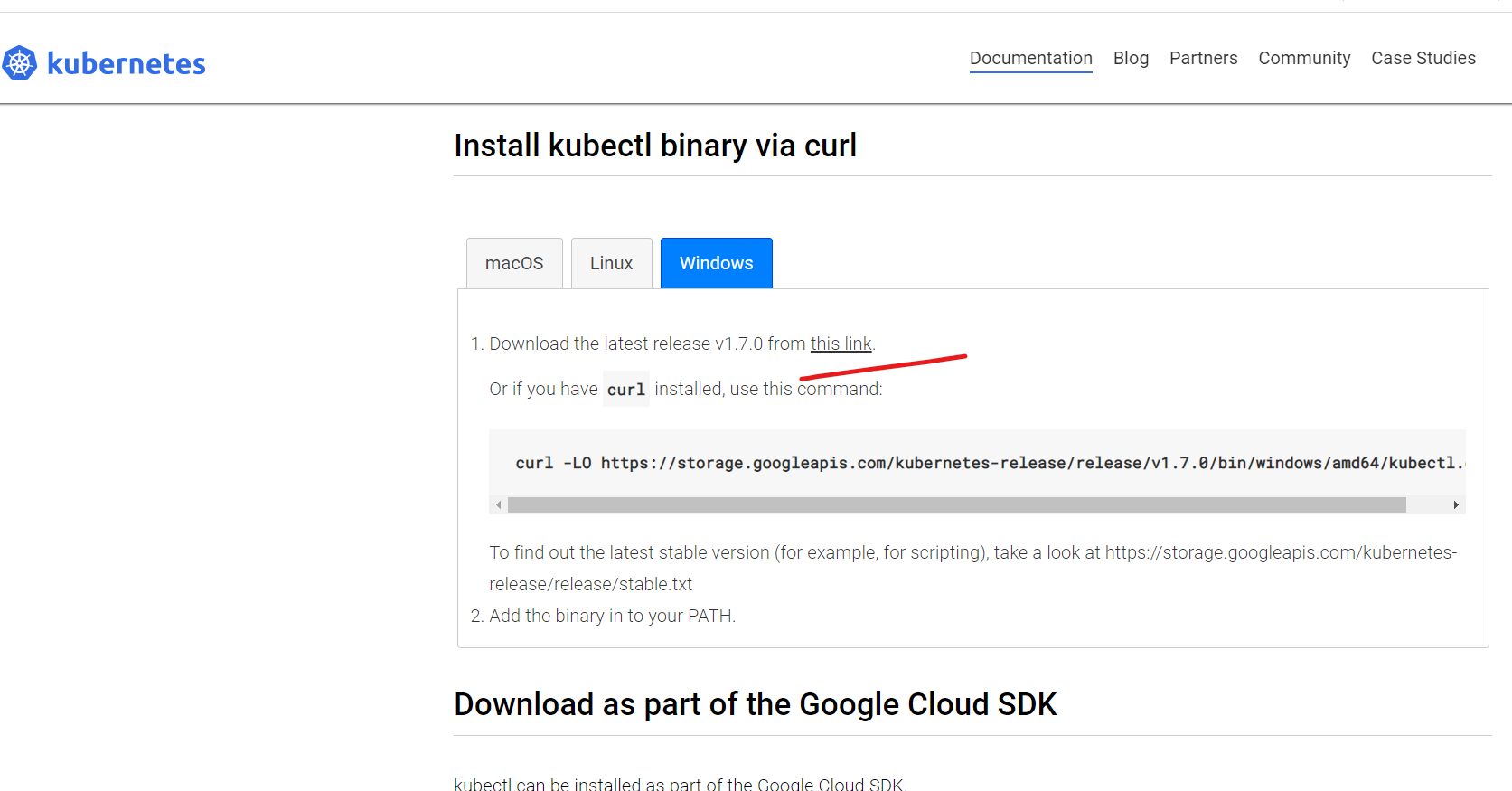
Download the minikube for window OS

<https://minikube.sigs.k8s.io/docs/start/>



**Kubectl download**

[**http://pwittrock.github.io/docs/tasks/tools/install-kubectl/#install-kubectl-binary-via-curl**](http://pwittrock.github.io/docs/tasks/tools/install-kubectl/#install-kubectl-binary-via-curl)

****

Before build the spring boot project you have the environment variable for mysql hostname, db name, username and password.

application.yml

spring:

datasource:

driver-class-name: com.mysql.cj.jdbc.Driver

url: jdbc:mysql://${DB\_HOST:localhost}/${DB\_NAME:healthcare}?useSSL=false

username: ${DB\_USERNAME:root}

password: ${DB\_PASSWORD:root@123}

jpa:

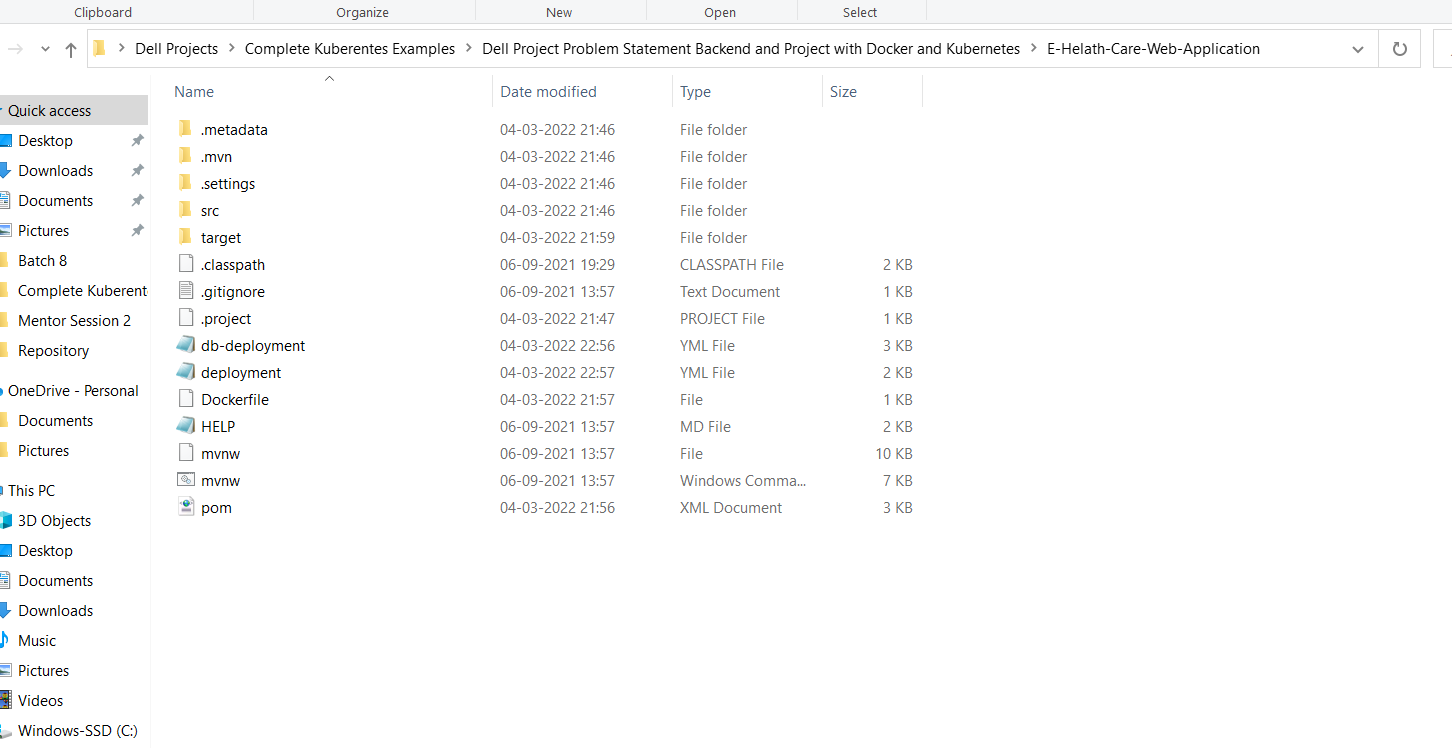
database-platform: org.hibernate.dialect.MySQL5Dialect

generate-ddl: true

show-sql: true

hibernate:

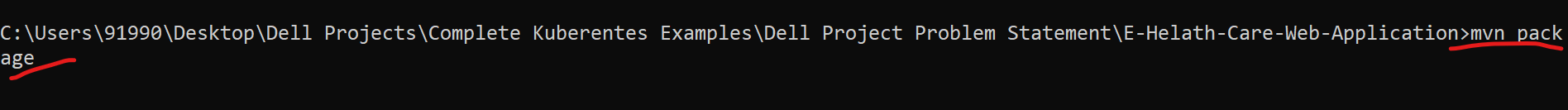
ddl-auto: update



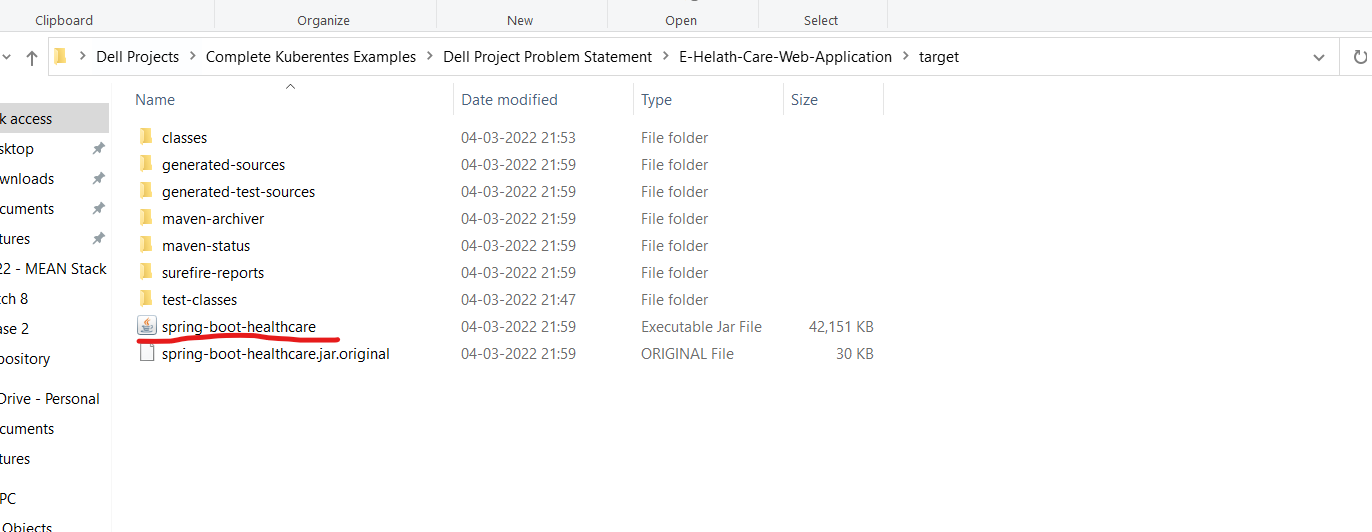
First open the command prompt in backend project and

Build the project using the command as

mvn build



Then that you can see inside a target folder jar file



So the jar file name is

spring-boot-healthcar.jar

Now you have to create the docker image using this jar file.

Dockerfile

FROM openjdk:11

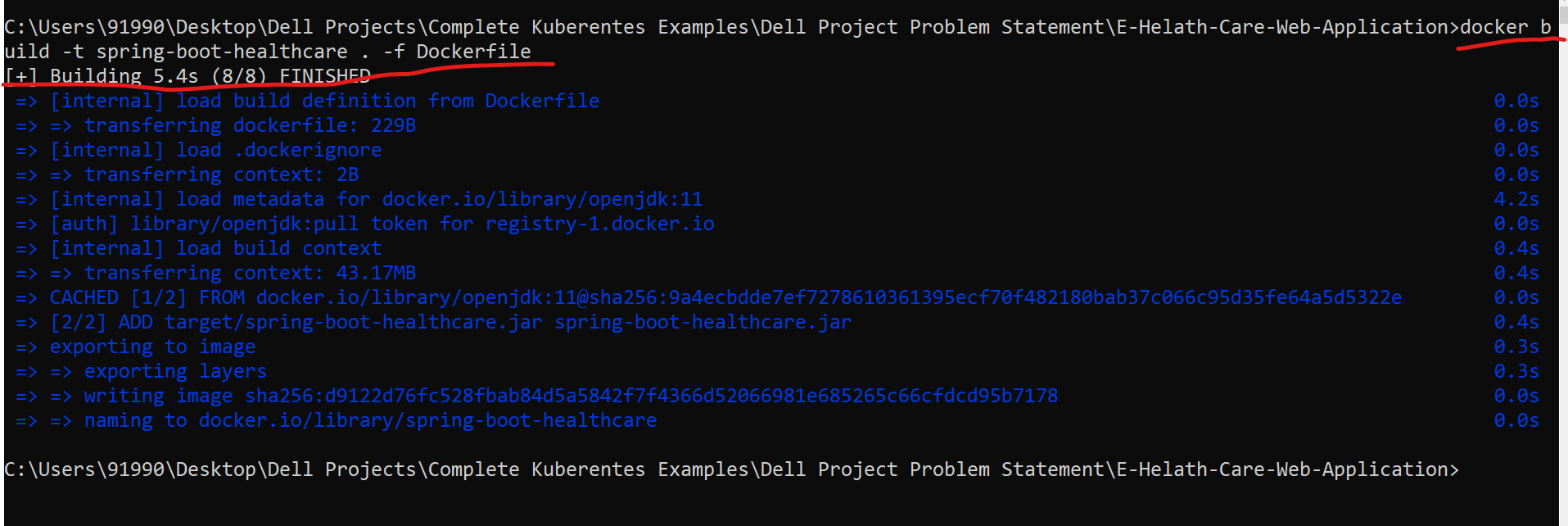
EXPOSE 8080

#COPY target/MySpringRestMySQL.jar .

ADD target/spring-boot-healthcare.jar spring-boot-healthcare.jar

ENTRYPOINT ["java","-jar","/spring-boot-healthcare.jar"]

**docker build –t spring-boot-healthcare . –f Dockerfile**



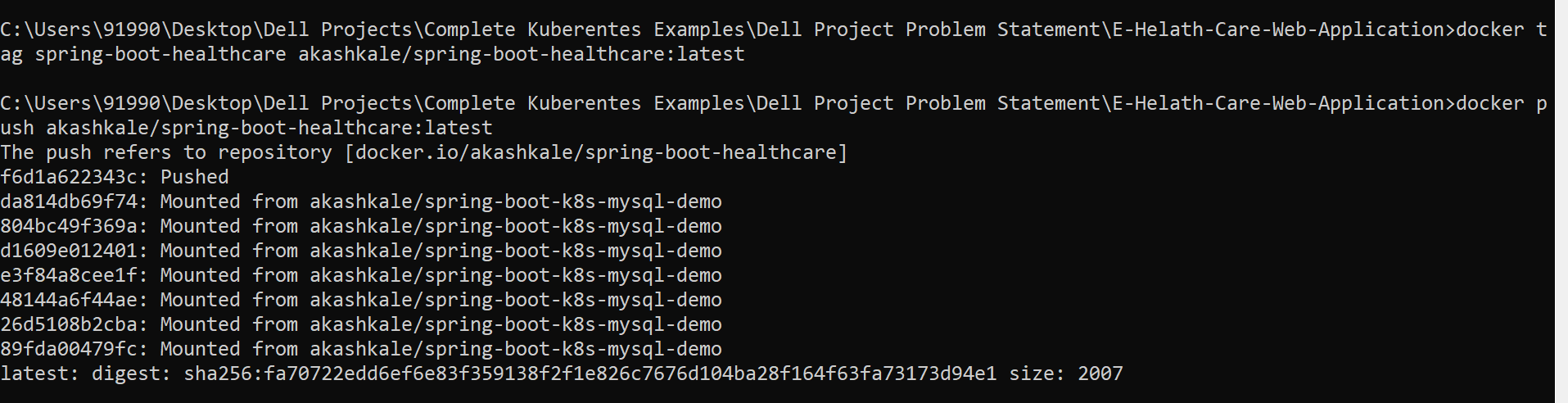
Now you have to push this project in Docker hub account

Before push you have to create the tag for this project.

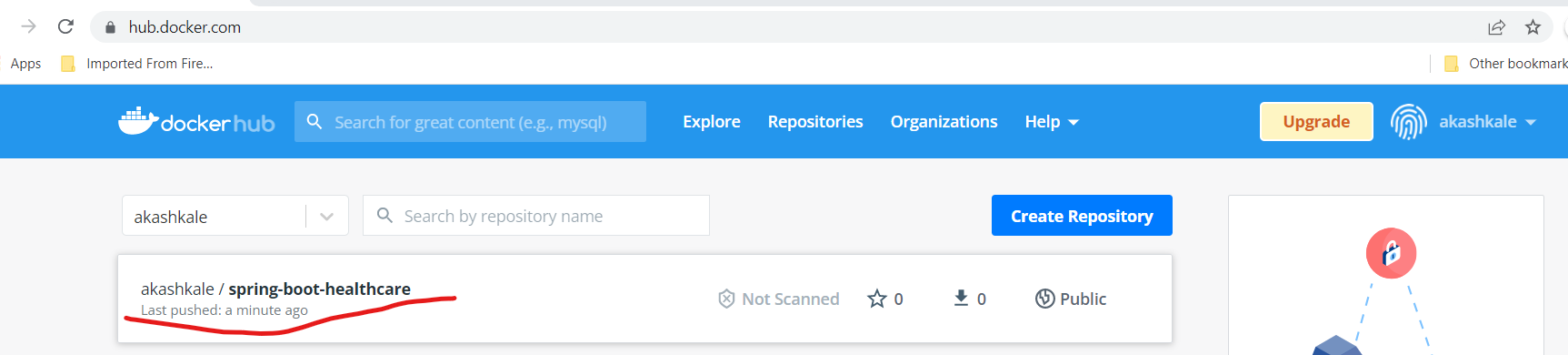
docker tag spring-boot-healthcare akashkale/spring-boot-healthcare:latest

Then you have to push the project in docker hub account

docker push akashkale/spring-boot-healthcare:latest



Now you can verify this image in in docker hub account

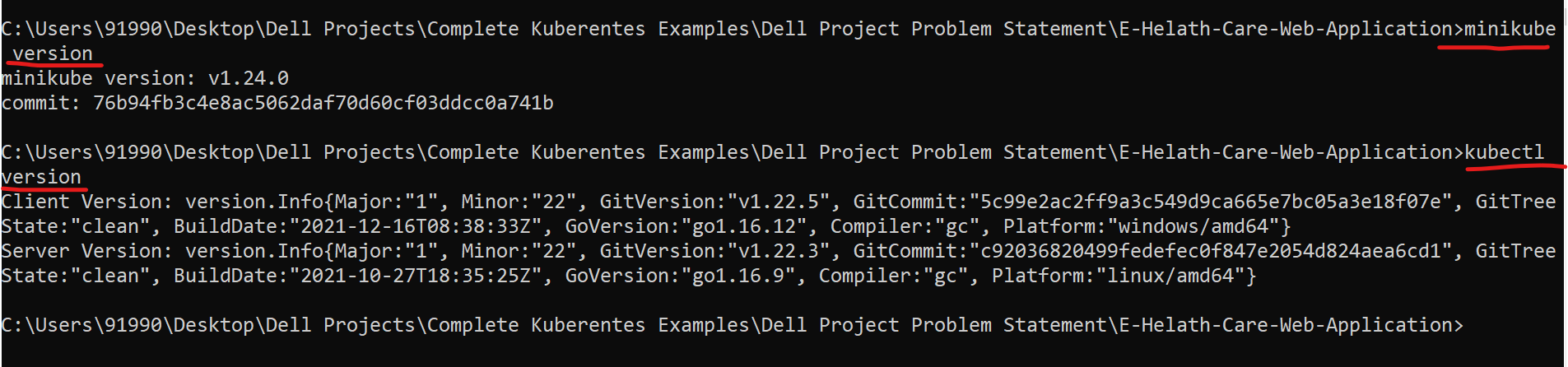


You have to install minikube and kubectl

in respective os

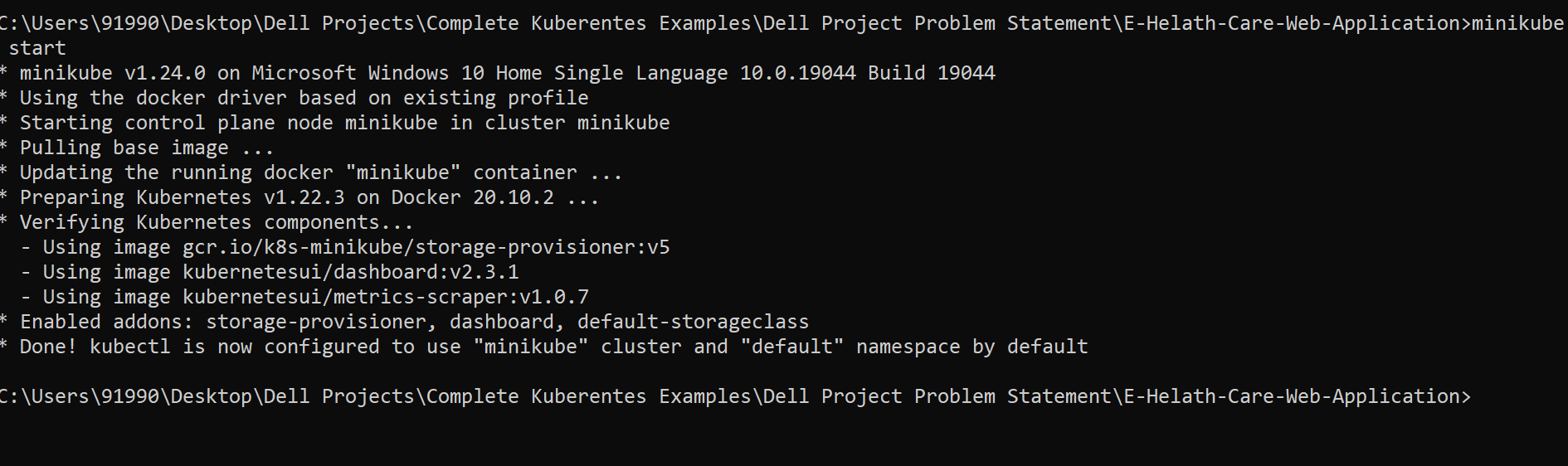
minikube version

kubectl version

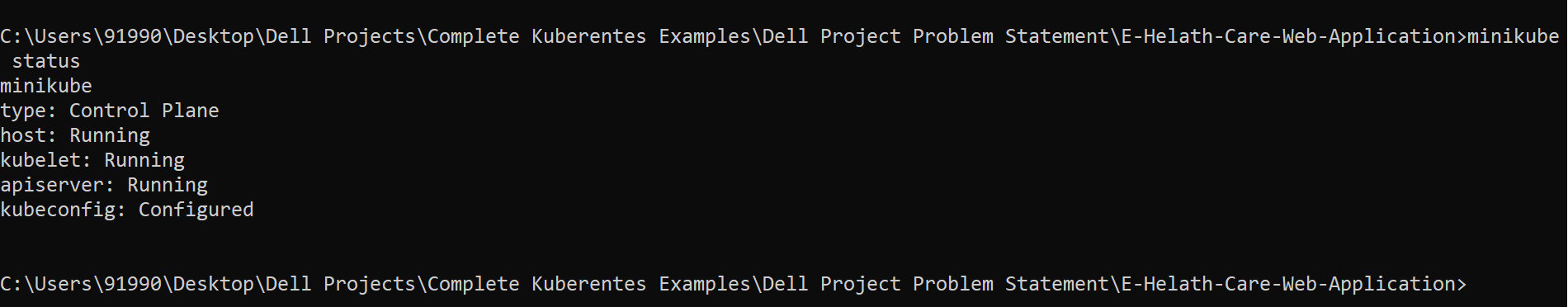


Now you have to start the minikube

minikube start



Now you can check the status of minikube



Now you have to create the deployment object for mysql

db-deployment.yml

# Define a 'Persistent Voulume Claim'(PVC) for Mysql Storage, dynamically provisioned by cluster

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: mysql-pv-claim1 # name of PVC essential for identifying the storage data

labels:

app: mysql

tier: database

spec:

accessModes:

- ReadWriteOnce #This specifies the mode of the claim that we are trying to create.

resources:

requests:

storage: 1Gi #This will tell kubernetes about the amount of space we are trying to claim.

---

# Configure 'Deployment' of mysql server

apiVersion: apps/v1

kind: Deployment

metadata:

name: mysql

labels:

app: mysql

tier: database

spec:

selector: # mysql Pod Should contain same labels

matchLabels:

app: mysql

tier: database

strategy:

type: Recreate

template:

metadata:

labels: # Must match 'Service' and 'Deployment' selectors

app: mysql

tier: database

spec:

containers:

- image: mysql:5.7 # image from docker-hub

args:

- "--ignore-db-dir=lost+found" # Workaround for https://github.com/docker-library/mysql/issues/186

name: mysql

env:

- name : DB\_NAME

value: k8s

- name: DB\_HOST # Setting Root Password of mysql From a 'Secret'

value: mysql

- name: DB\_USERNAME # Setting Root Password of mysql From a 'Secret'

value: root

- name: DB\_PASSWORD # Setting Root Password of mysql From a 'Secret'

value: root@123

ports:

- containerPort: 3306

name: mysql

volumeMounts: # Mounting voulume obtained from Persistent Volume Claim

- name: mysql-persistent-storage

mountPath: /var/lib/mysql #This is the path in the container on which the mounting will take place.

volumes:

- name: mysql-persistent-storage # Obtaining 'vloume' from PVC

persistentVolumeClaim:

claimName: mysql-pv-claim1

---

apiVersion: v1

kind: Service

metadata:

name: mysql # DNS name

labels:

app: mysql

tier: database

spec:

ports:

- port: 3306

targetPort: 3306

selector: # mysql Pod Should contain same labels

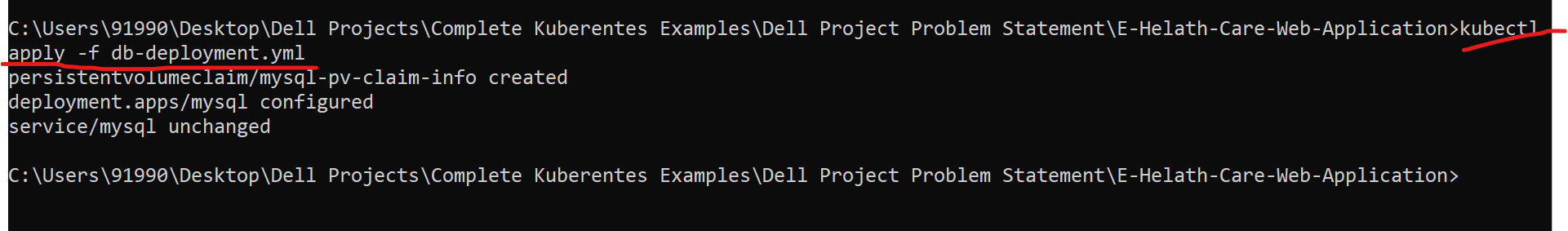
app: mysql

tier: database

clusterIP: None # We Use DNS, Thus ClusterIP is not relevant

---

kubectl apply –f db-deployment.yml

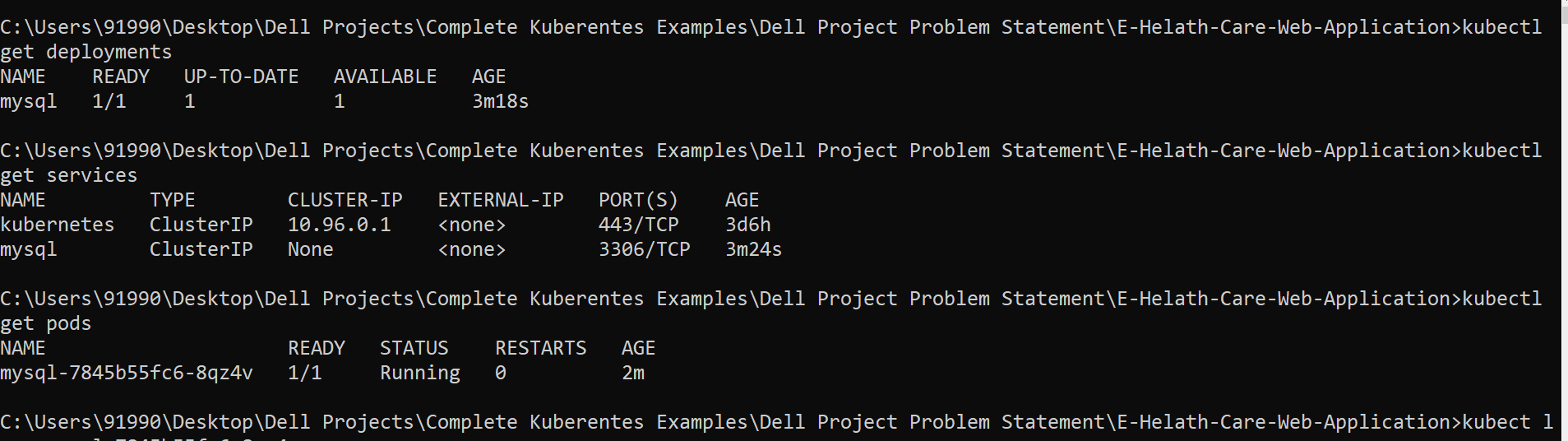


Now you can check the status of deployment, services and pods

kubectl get deployments

kubectl get services

kubectl get pods



Now mysql pods is ready

Now we will login for mysql pods

Please open another terminal and check the mysql

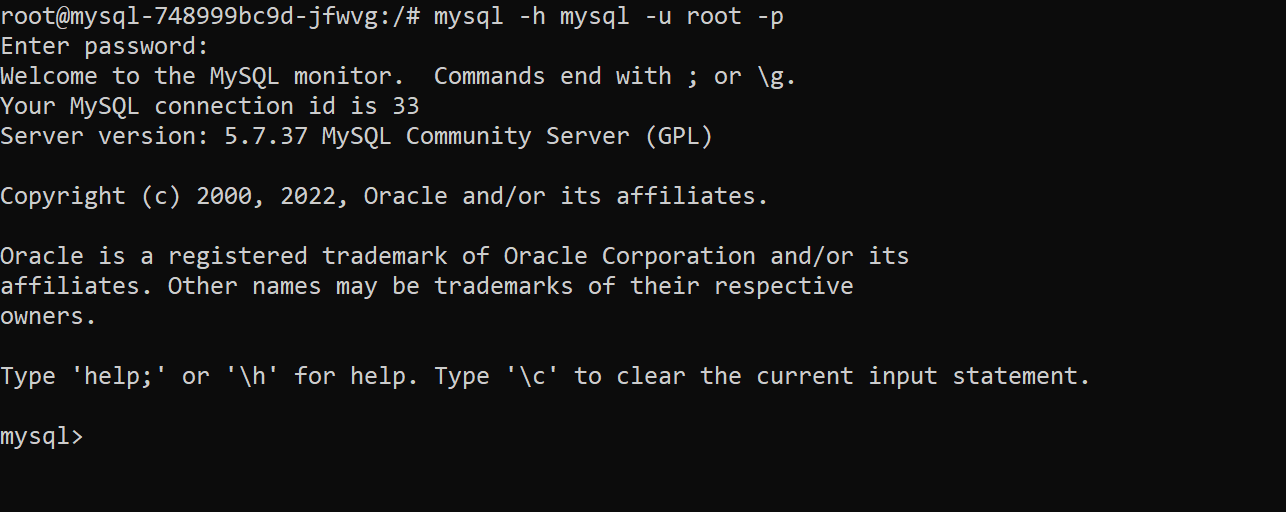
kubectl exec –it podsname /bin/bash

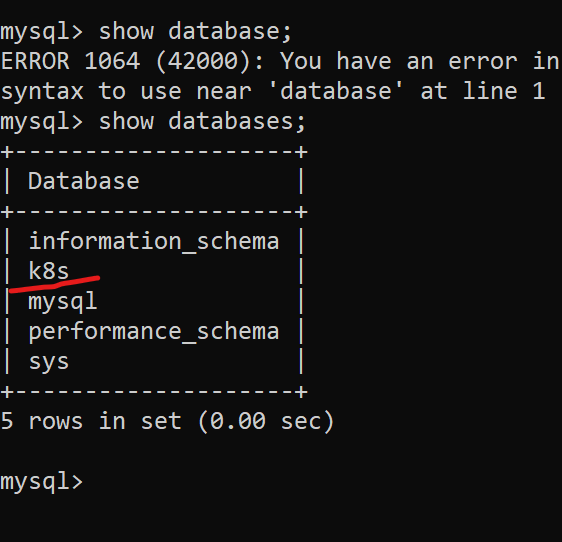
kubectl exec -it mysql-7845b55fc6-8qz4v /bin/bash

Now login with hostname, username and password.

mysql -h mysql -u root –p

hostname is mysql, username is root and password is root@123





Now mysql pods is running

Now you have to start the spring-boot application deployment in Kubernetes

kubectctl apply –g deployment.yml

**deployment.yml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: springboot-k8s-mysql

spec:

selector:

matchLabels:

app: springboot-k8s-mysql-test

replicas: 3

template:

metadata:

labels:

app: springboot-k8s-mysql-test

spec:

containers:

- name: springboot-k8s-mysql-test

image: akashkale/spring-boot-healthcare

ports:

- containerPort: 8080

env: # Setting Enviornmental Variables

- name: DB\_HOST # Setting Database host address from configMap

value: mysql

- name: DB\_NAME # Setting Database name from configMap

value: k8s

- name: DB\_USERNAME # Setting Database username from Secret

value: root

- name: DB\_PASSWORD # Setting Database password from Secret

value: root@123

---

apiVersion: v1 # Kubernetes API version

kind: Service # Kubernetes resource kind we are creating

metadata: # Metadata of the resource kind we are creating

name: springboot-k8s-svc

spec:

selector:

app: springboot-k8s-mysql-test

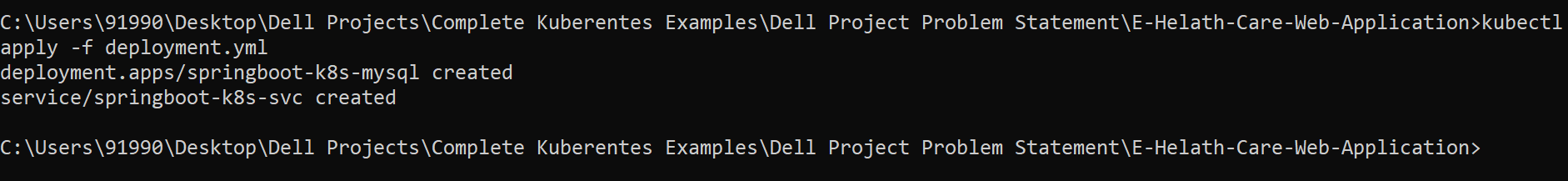
ports:

- protocol: "TCP"

port: 8080 # The port that the service is running on in the cluster

targetPort: 8080 # The port exposed by the service

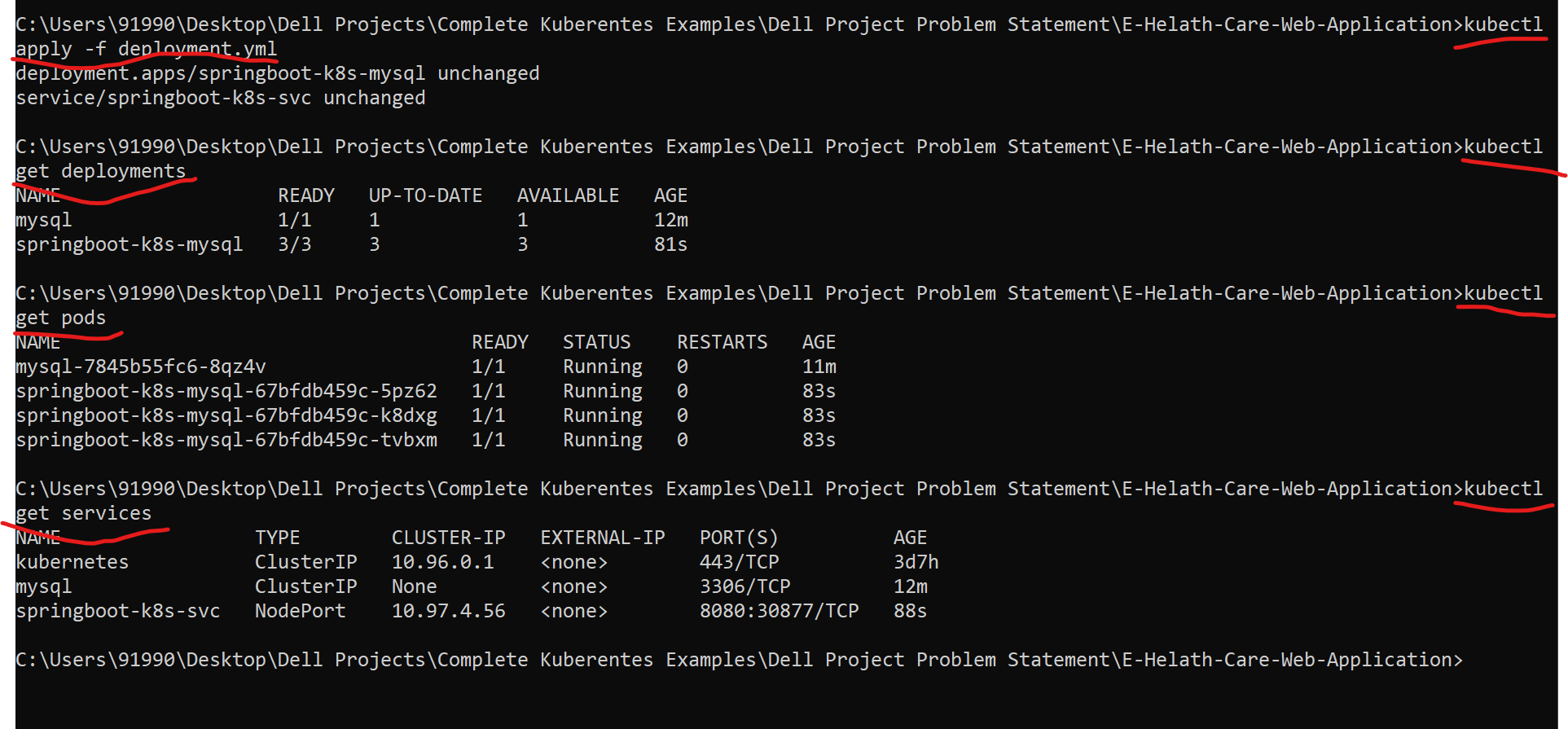
type: NodePort # type of the service.



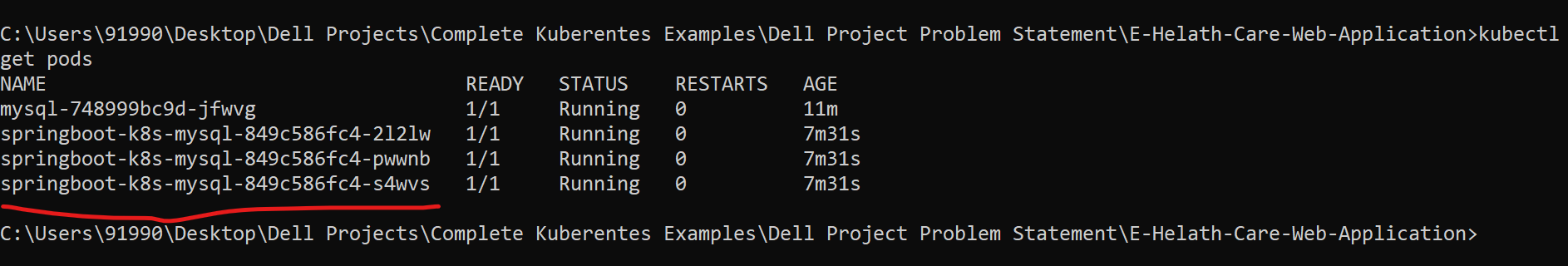
kubectl get deployments

kubectl get pods

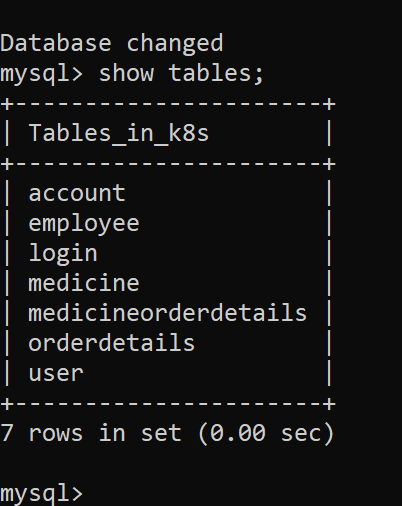
kubectl get services



Now you can check all three pods are running base upon the replica

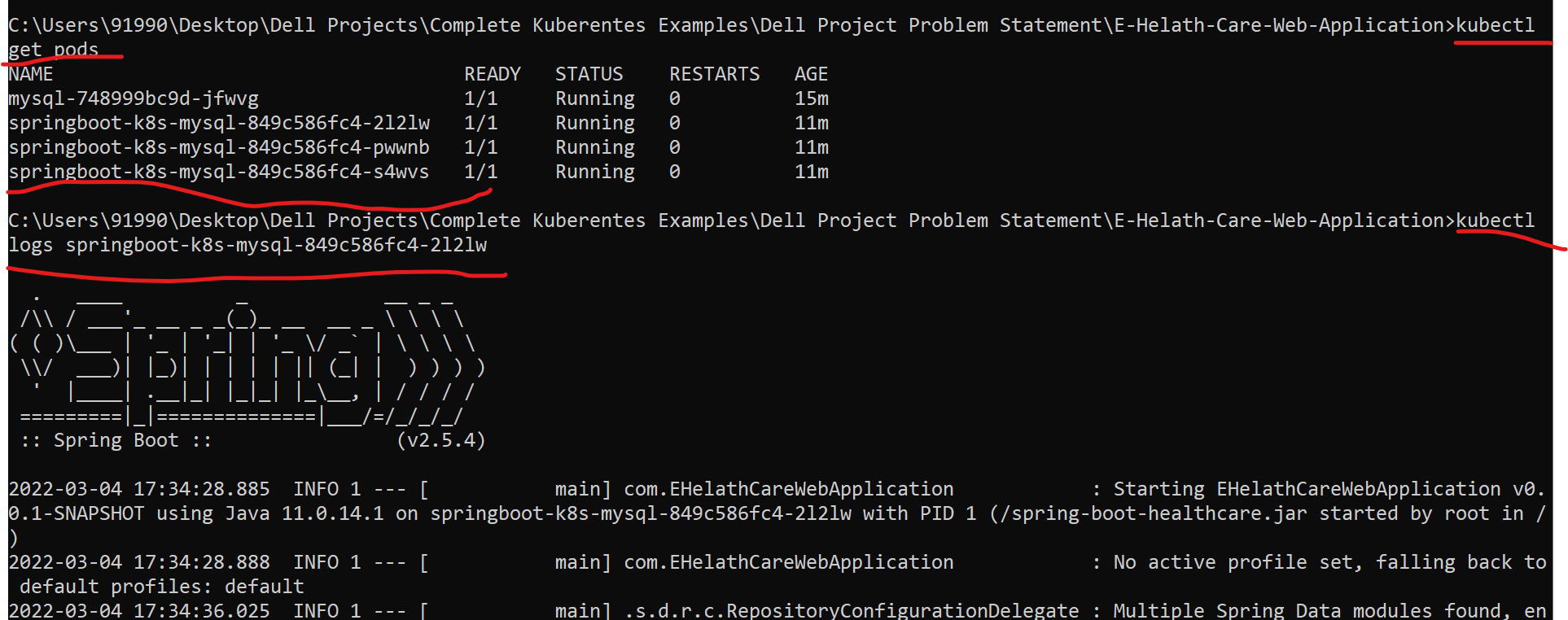


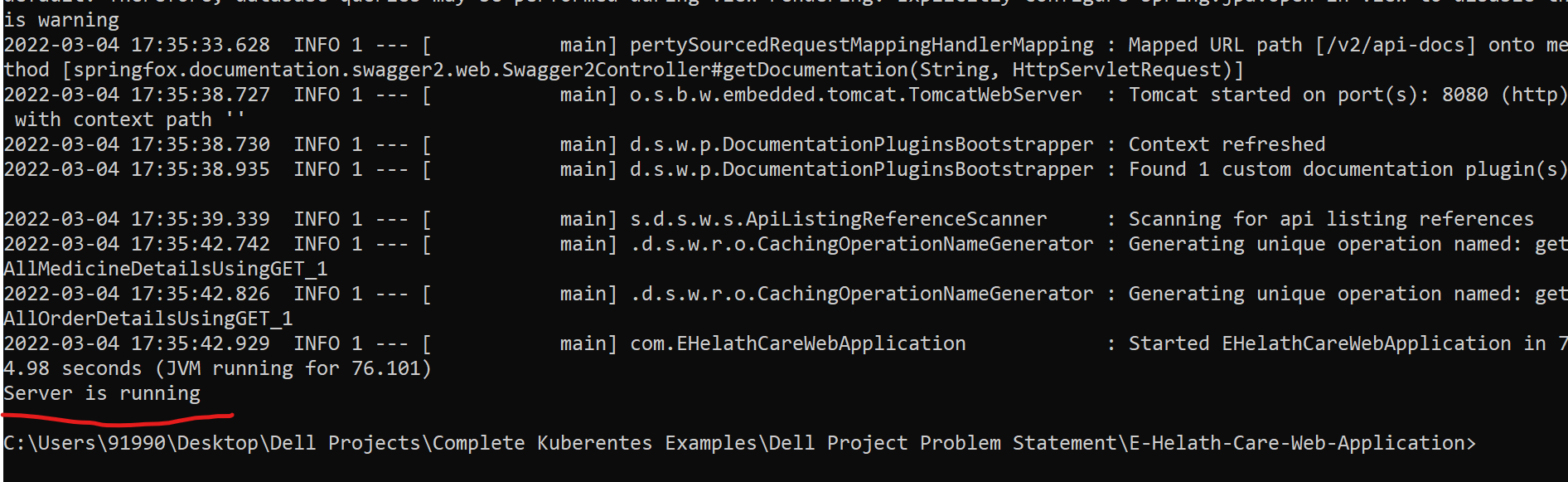
Now you can check all tables created in mysql pods



kubectl get pods

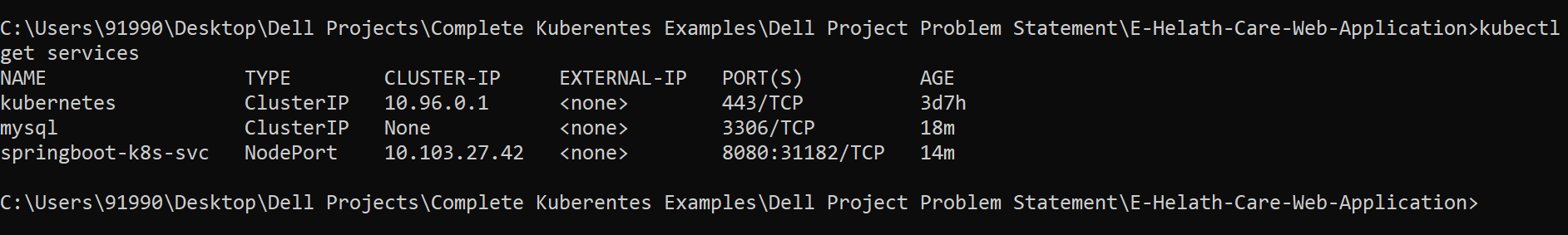
**kubectl logs podsname**

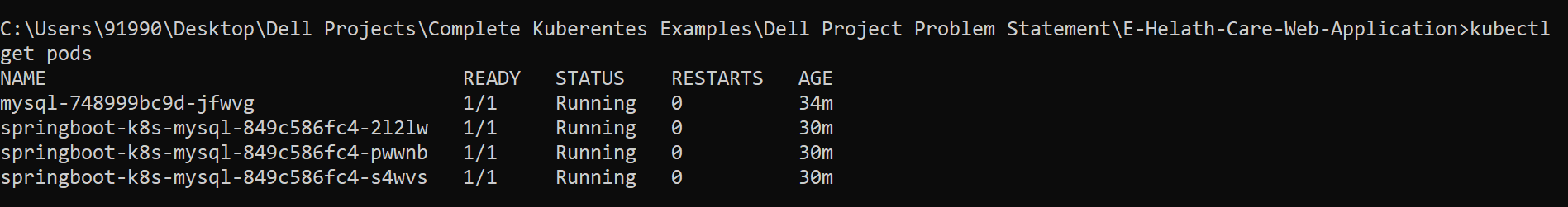


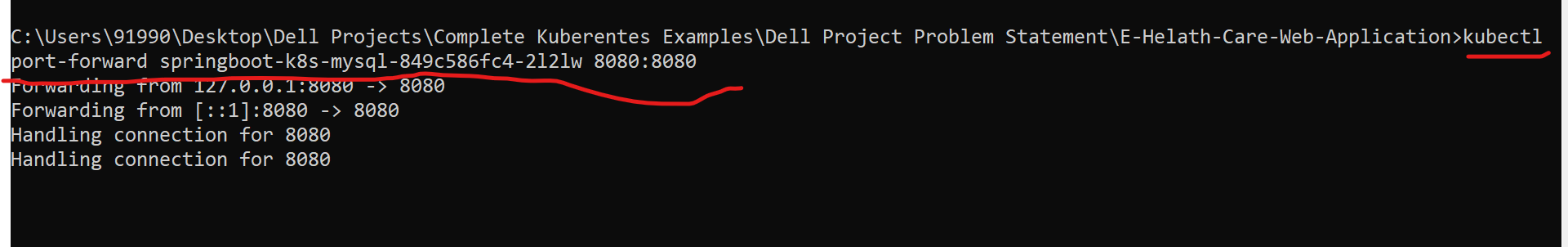


Now you can see two services are running it mysql and spring-boot service

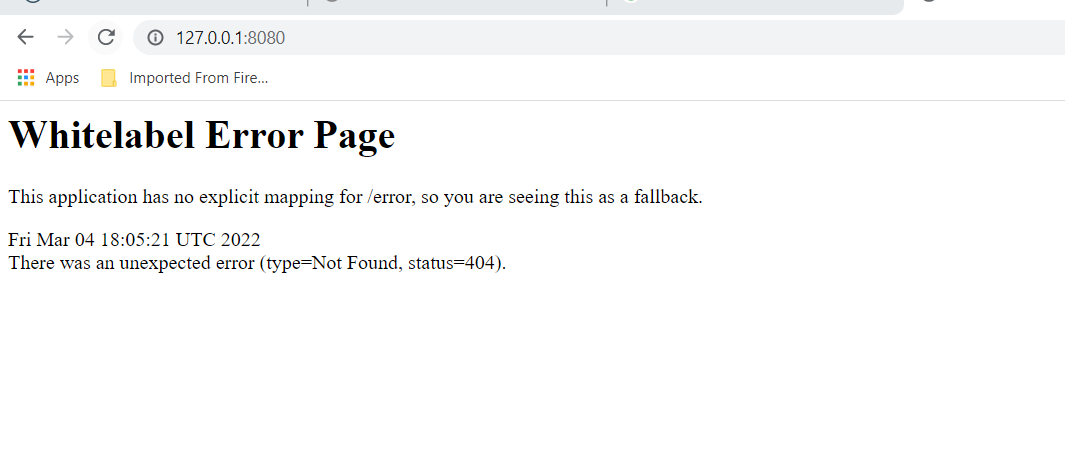
kubectl get services



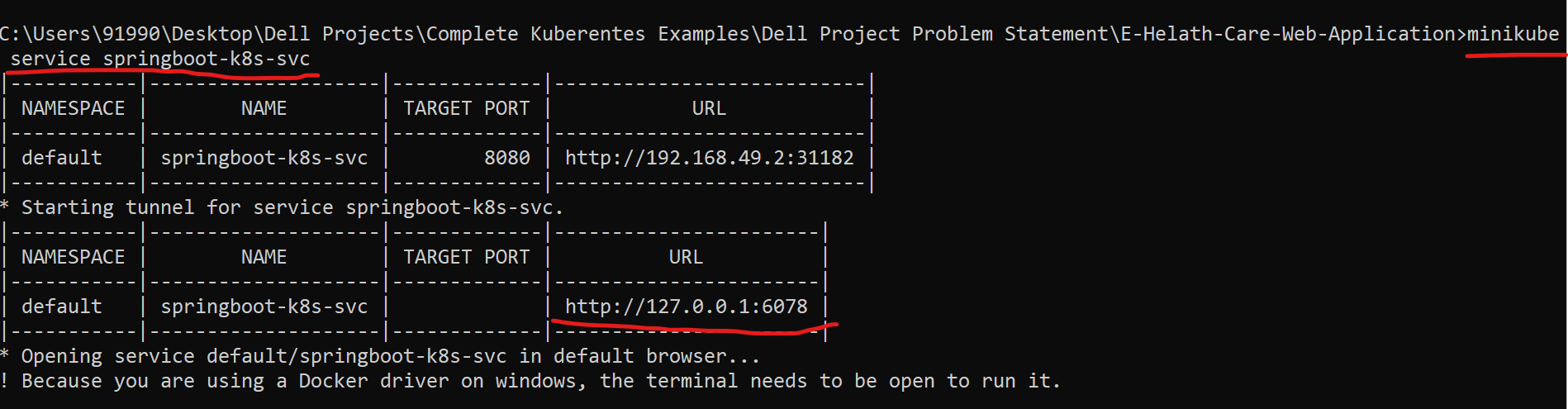


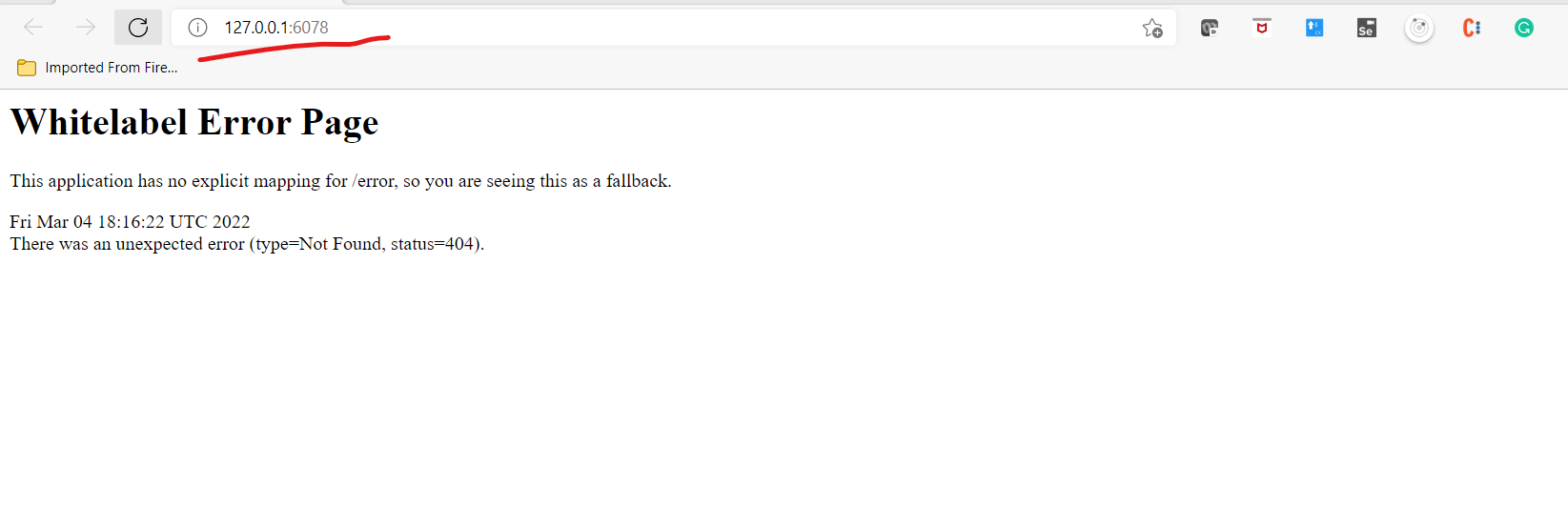


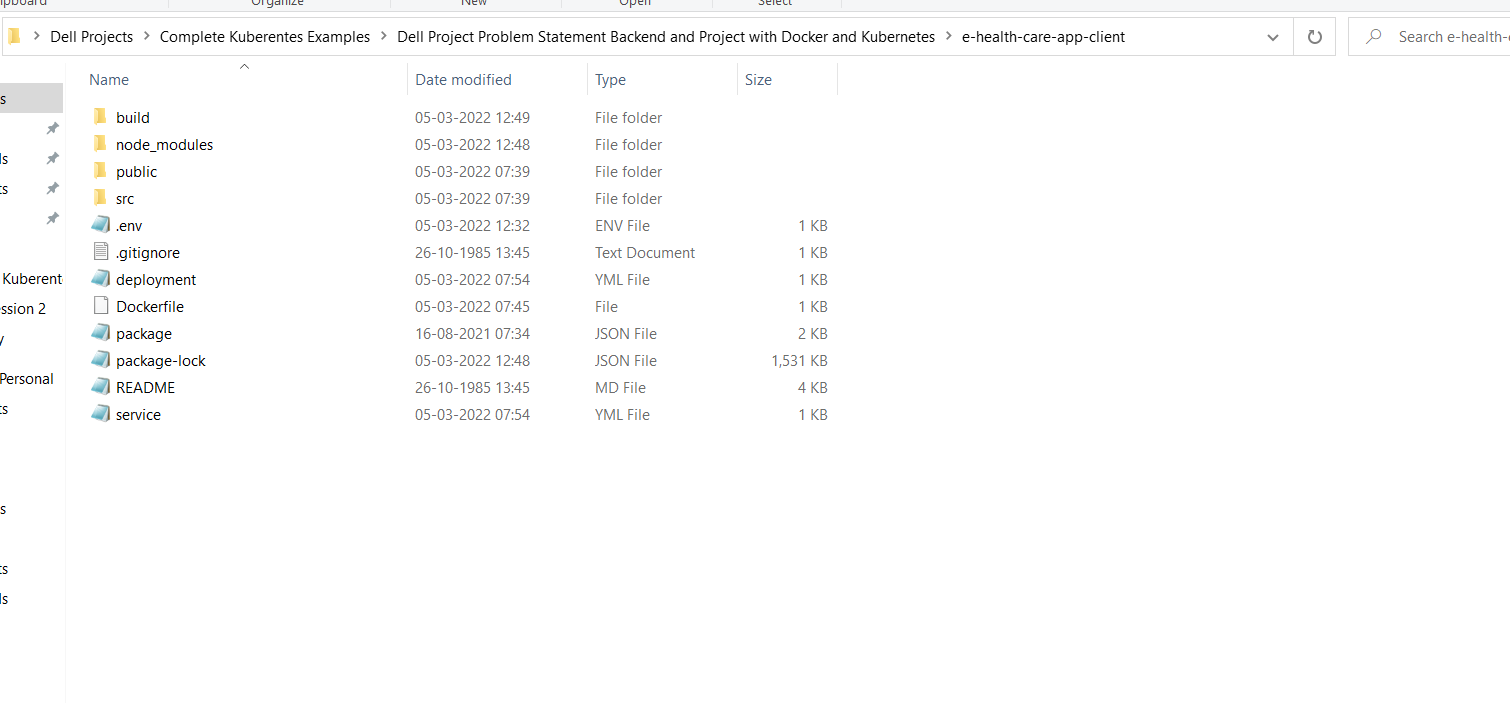
Now you can see the application running



Another way to run the application







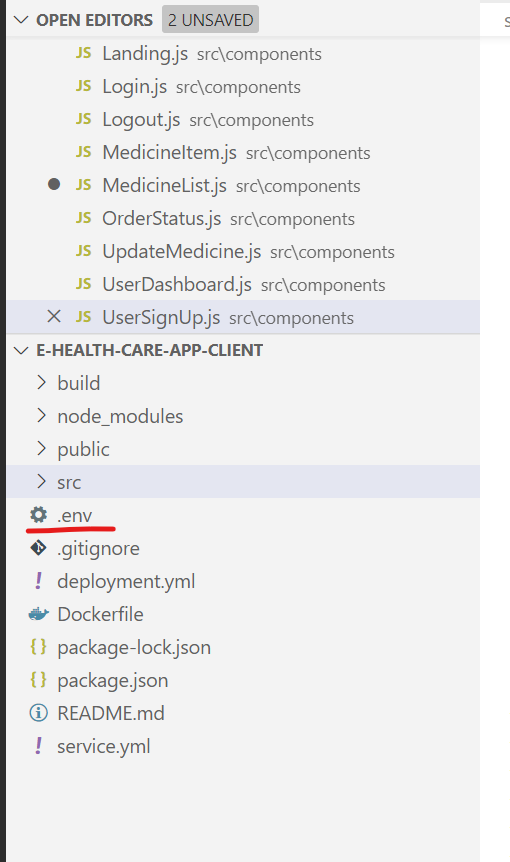
Now open the Frontend project ie React JS application

First open the command prompt in React JS project run the command to download the dependencies.

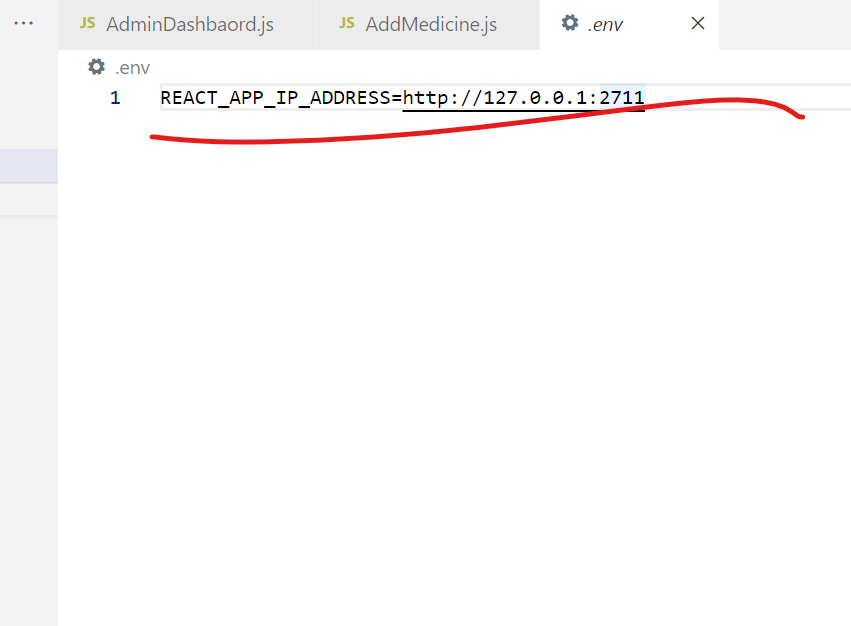
Inside the project folder create .env folder to set the global IP address of Spring boot application running on different ip with port number

As of now application running on port number as

127.0.0.1:6078

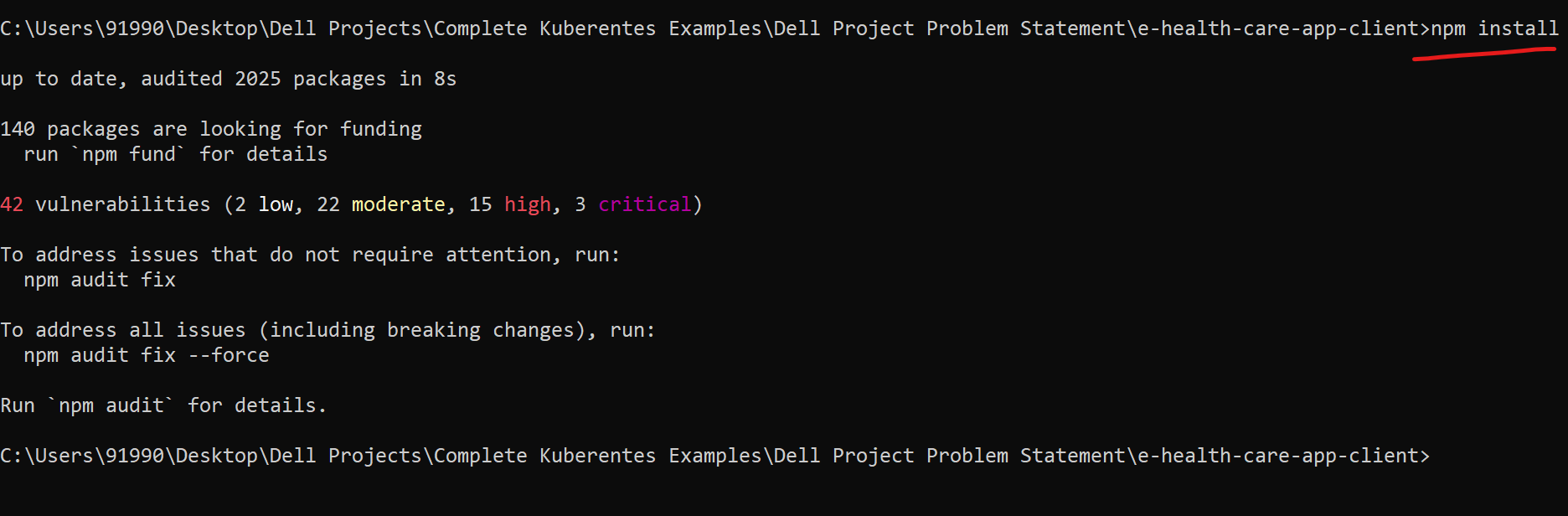


In .evn file contains environment details

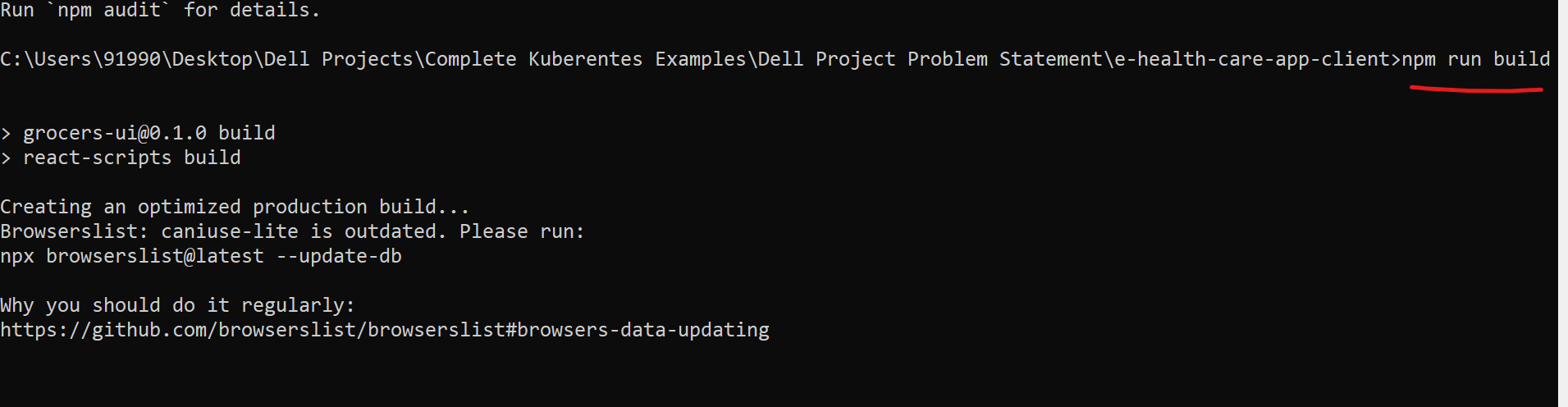


So one run the spring boot service you will get the IP and port number please replace in this file.

Then build the project.



npm run build



Now you have to create the Docker file

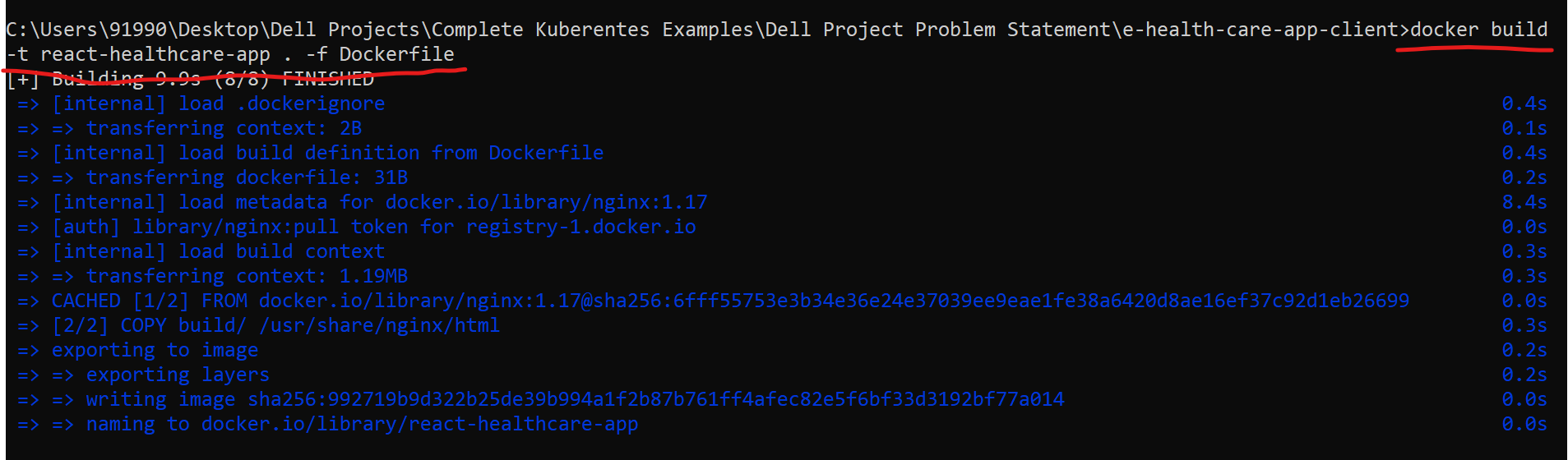
**Dockerfile**

FROM nginx:1.17

COPY build/ /usr/share/nginx/html

Now we have to create the image for this project

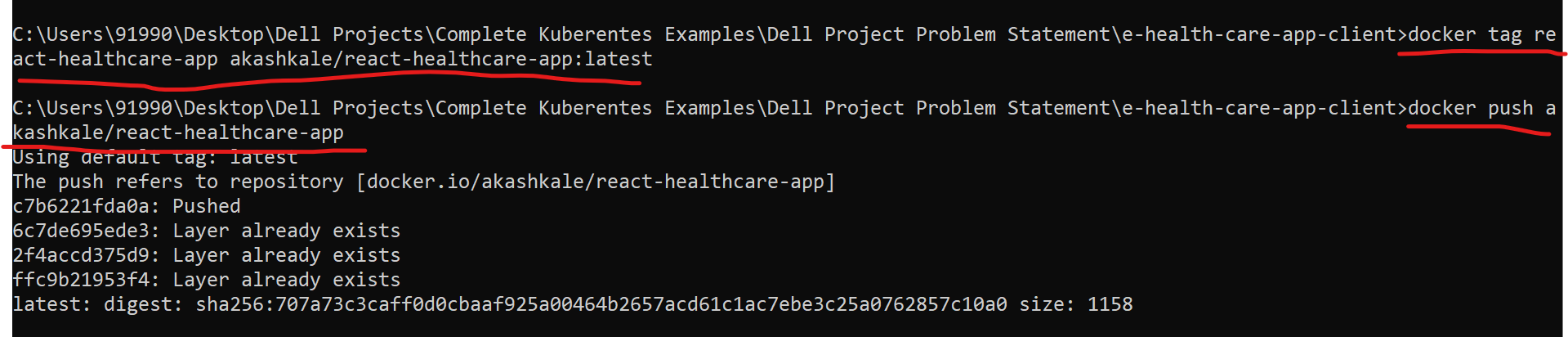
docker build –t react-healthcare-app . –f Dockerfile



Now we have create the tag for this image

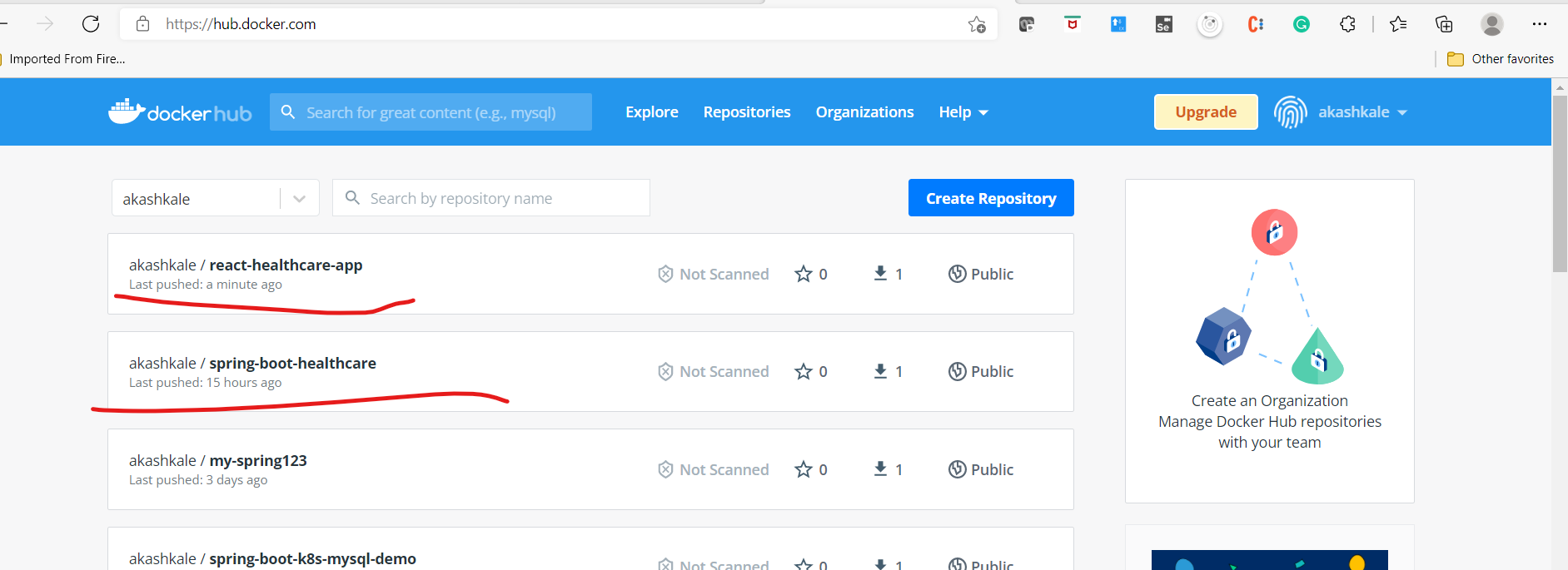
docker tag react-healthcare-app akashkale/react-healthcare-app:latest

Now you have to push this image in docker hub account



Now you can see the react js project : front end image

And spring boot project : backend image available in Docker hub account



Now you have to deploy the project and service

**deployment.yml**

kind: Deployment

apiVersion: apps/v1

metadata:

  name: my-react-app

spec:

  replicas: 2

  selector:

    matchLabels:

      app: my-react-app

  template:

    metadata:

      labels:

        app: my-react-app

    spec:

      containers:

        - name: my-react-app

          image: akashkale/react-healthcare-app

          imagePullPolicy: Always

          ports:

            - containerPort: 80

      restartPolicy: Always

**service.yml**

kind: Service

apiVersion: v1

metadata:

  name: my-react-app

spec:

  type: NodePort

  ports:

    - port: 80

      targetPort: 80

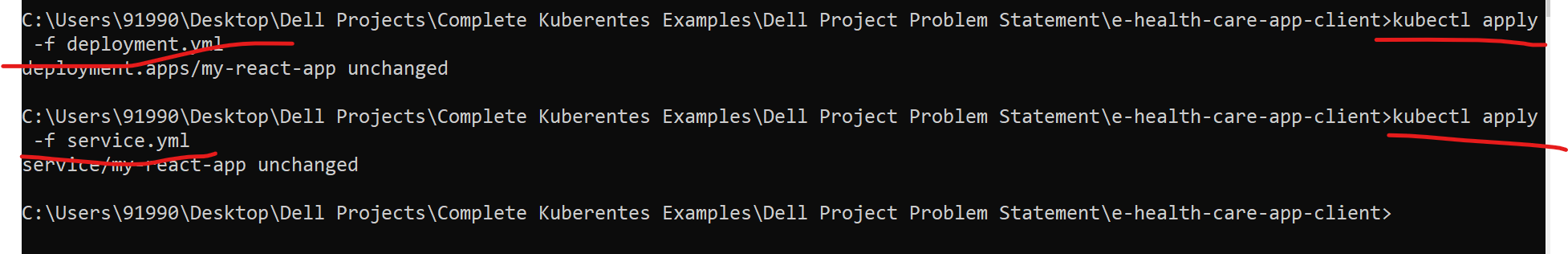
      protocol: TCP

      nodePort: 31000

  selector:

    app: my-react-app

now create the run the deployment.yml and service.yml file using kubectl command.

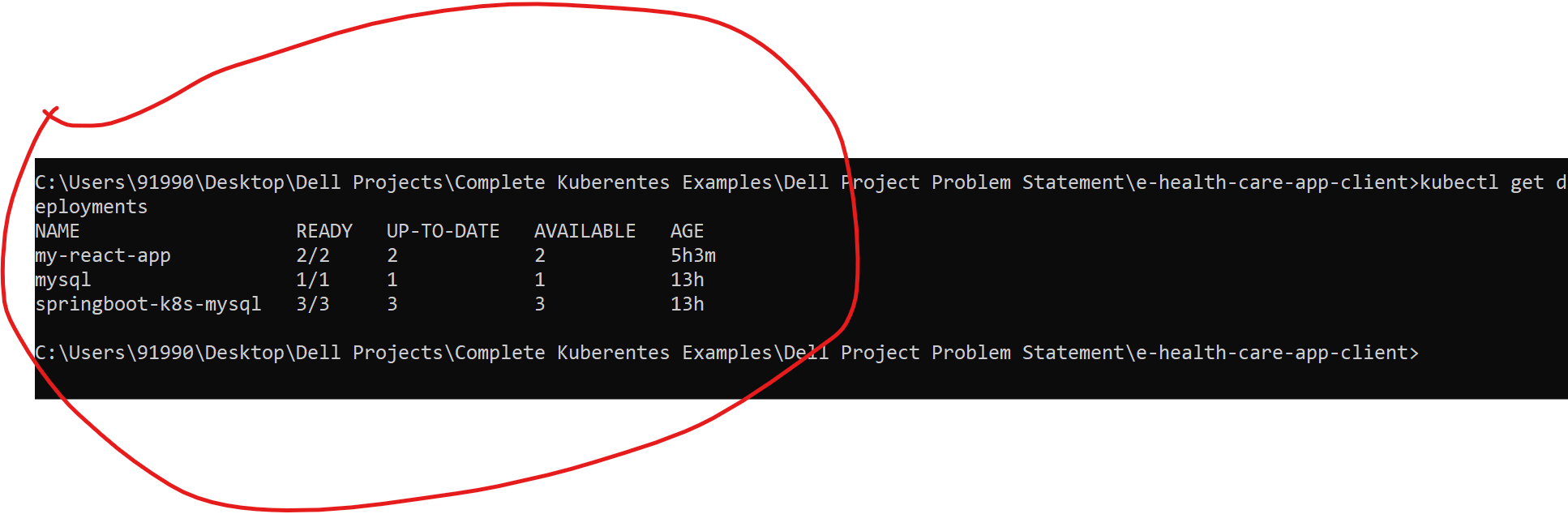


**kubectl apply –f deployment.yml**

**kubectl apply –f service.yml**

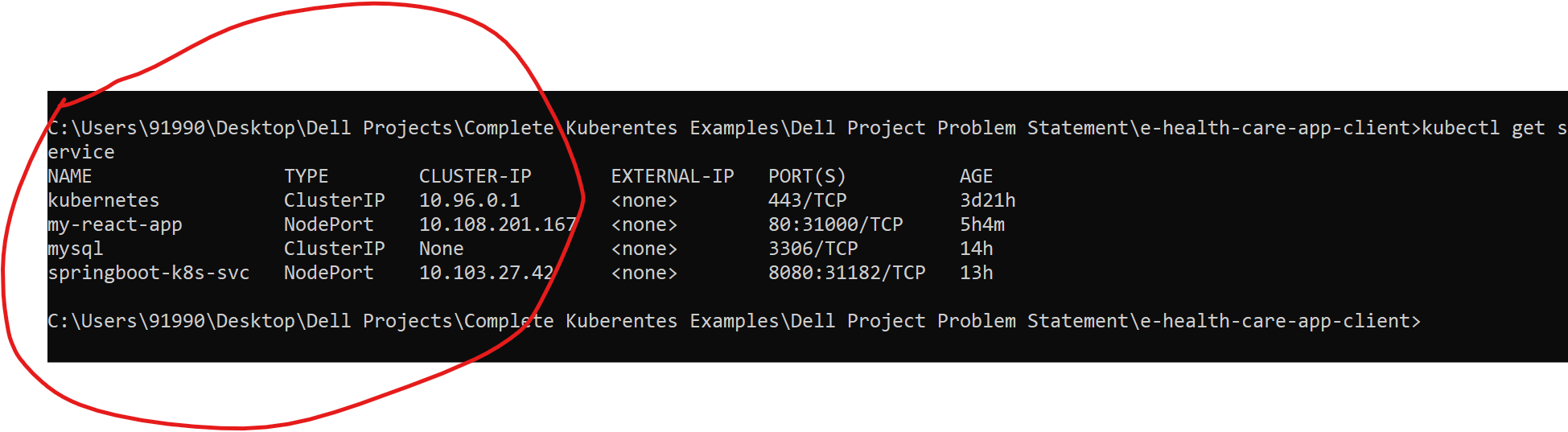
**now you can see the all deployments**

**kubectl get deployments**



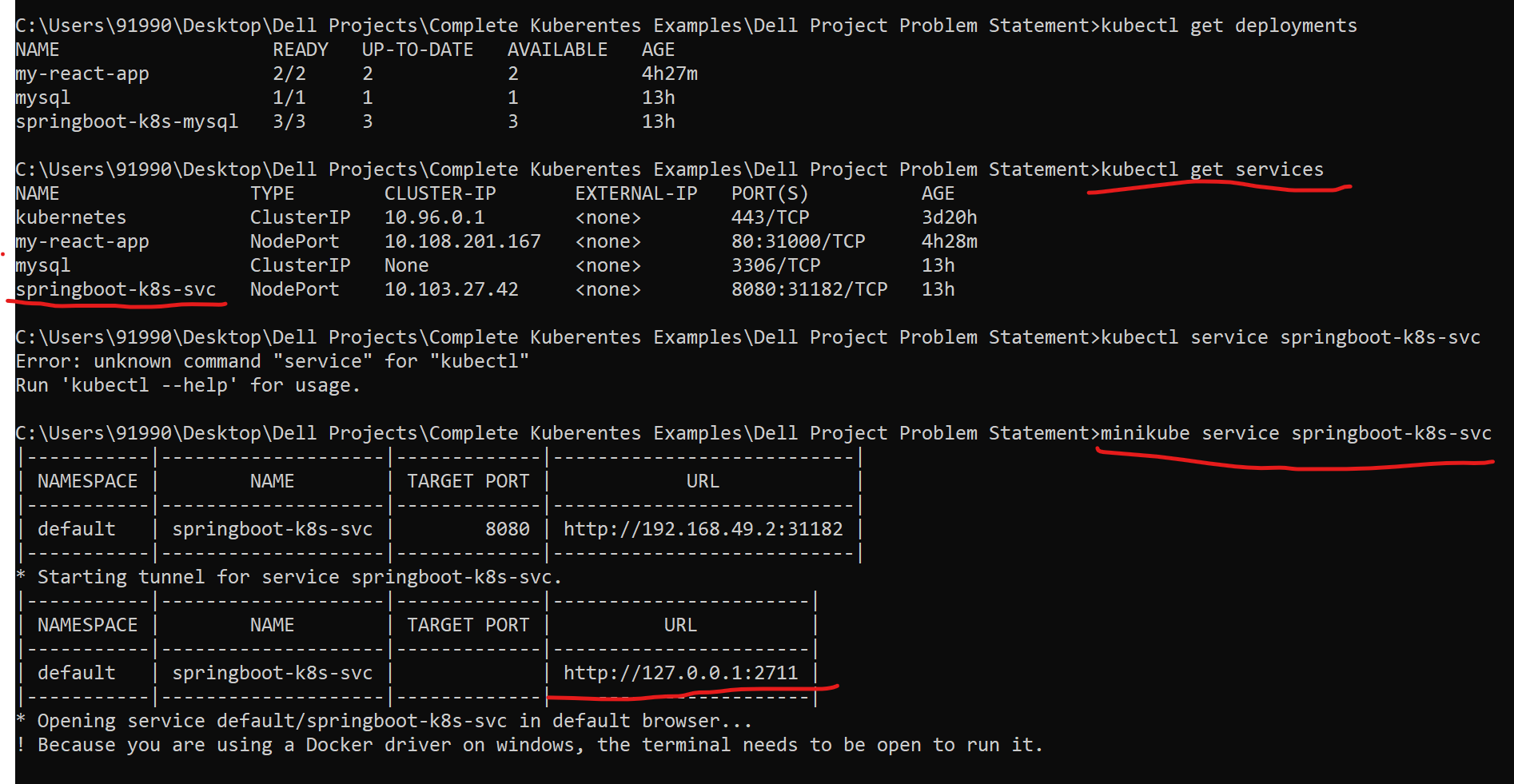
So here react deployment, mysql deployment and spring boot deployment is ready

kubectl get services



Now run both services ie

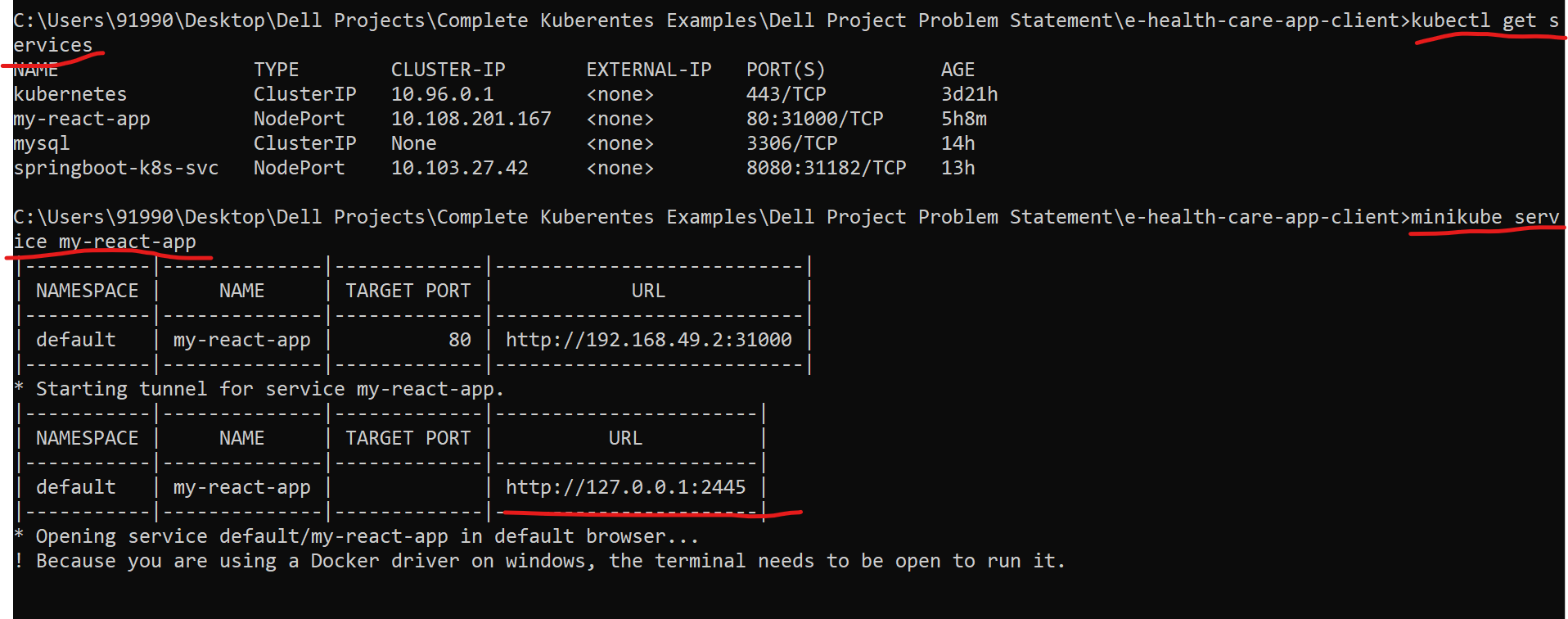
First backend service using the command as

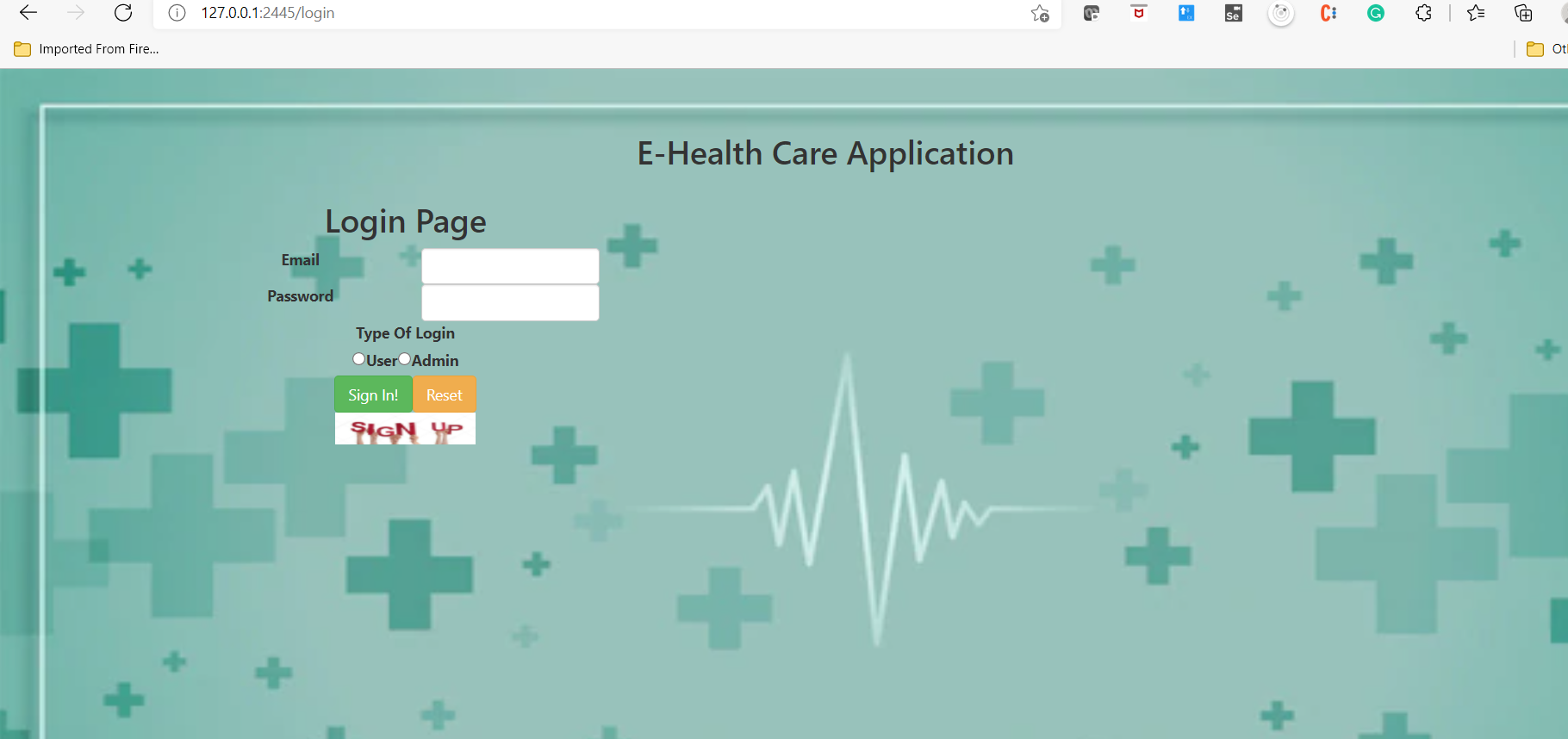


**kubectl get services**

**minikube service serviceName**

Now run the frontend service using the command as





Now front end application running