Question 1

```
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ ls
README.md  dockerfile  node_modules  package-lock.json  package.json  public  src
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ nano deployment.yaml
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ ls
README.md  deployment.yaml  dockerfile  node_modules  package-lock.json  package.json  public  src
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ nano service.yml
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ ls
README.md  deployment.yaml  dockerfile  node_modules  package-lock.json  package.json  public  service.yml  src
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
error: error validating "deployment.yaml": error validating data: failed to download openapi: Get "https://127.0.0.1:32769/openapi/v2?timeout=32s": dial tcp 127.0.0.1:32769: connect: connection refused; if you choose to ignore th
ese errors, turn validation off with --validate=false
error: the path "service.yaml" does not exist
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ kubectl apply -f deployment.yaml
error: error validating "deployment.yaml": error validating data: failed to download openapi: Get "https://127.0.0.1:32769/openapi/v2?timeout=32s": dial tcp 127.0.0.1:32769: connect: connection refused; if you choose to ignore th
ese errors, turn validation off with --validate=false
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ minikube status
minikube
type: Control Plane
host: Stopped
kubelet: Stopped
apiserver: Stopped
kubeconfig: Stopped

kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ minikube start
😄  minikube v1.32.0 on Ubuntu 22.04 (amd64)
✨  Using the docker driver based on existing profile
👍  Starting control plane node minikube in cluster minikube
🚜  Pulling base image ...
🔄  Restarting existing docker container for "minikube" ...
🐳  Preparing Kubernetes v1.28.3 on Docker 24.0.7 ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🔎  Verifying Kubernetes components...
🌟  Enabled addons: storage-provisioner, default-storageclass
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ kubectl apply -f deployment.yaml
deployment.apps/my-react-app created
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ kubectl apply -f service.yaml
error: the path "service.yaml" does not exist
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ ls
README.md  deployment.yaml  dockerfile  node_modules  package-lock.json  package.json  public  service.yml  src
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ kubectl apply -f service.yml
service/my-react-app-service created
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$
```


```
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ kubectl apply -f deployment.yaml
deployment.apps/my-react-app created
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ kubectl apply -f service.yaml
error: the path "service.yaml" does not exist
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ ls
README.md  deployment.yaml  dockerfile  node_modules  package-lock.json  package.json  public  service.yml  src
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ kubectl apply -f service.yml
service/my-react-app-service created
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 6

TERMINAL

GNU nano 6.2 stateful.yml

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mongo
spec:
  serviceName: "mongo"
  replicas: 3
  selector:
    matchLabels:
      app: mongo
  template:
    metadata:
      labels:
        app: mongo
    spec:
      containers:
      - name: mongo
        image: mongo:latest
        ports:
        - containerPort: 27017
        volumeMounts:
        - name: mongo-data
          mountPath: /data/db
  volumeClaimTemplates:
  - metadata:
      name: mongo-data
    spec:
      accessModes: [ "ReadWriteOnce" ]
      resources:
        requests:
          storage: 1Gi
```

Stateful yaml file

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS 6
TERMINAL
  GNU nano 6.2
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mongo-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/path/to/host/dir"
```

Persistent volume yaml

```
TERMINAL
  GNU nano 6.2
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mongo-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Persistent volume claim yaml

```
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ kubectl apply -f stateful.yml
statefulset.apps/mongo created
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ kubectl apply -f mongo-pv.yml
persistentvolume/mongo-pv created
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ kubectl apply -f persistentvolumeclaim.yml
persistentvolumeclaim/mongo-pvc created
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$
```

Question 2

1. Create a docker image with nodejs installed

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

TERMINAL
  GNU nano 6.2
# Use an official Node.js image as the base image
FROM node:14-alpine

# Set the working directory inside the container
WORKDIR /app

# Copy package.json and package-lock.json files to the working directory
COPY package*.json ./

# Install dependencies
RUN npm install

# Copy the rest of the application files to the working directory
COPY . .

# Build the React app
RUN npm run build

# Expose port 3000 to the outside world
EXPOSE 3000

# Command to run the React app when the container starts
CMD ["npm", "start"]
```

2.

```
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ docker images
REPOSITORY                                          TAG       IMAGE ID       CREATED             SIZE
my-react-app                                        latest    a63f49be5484   About an hour ago   820MB
ansible-image
```

```
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ docker tag a63f49be5484 docker-4034:v1.0
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$
```

3.

```
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ docker tag a63f49be5484 docker-4034:v1.0
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ docker run my-react-app  npm --version
6.14.18
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$
```

4. Update docker file to accept volume

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

TERMINAL

  GNU nano 6.2
# Use an official Node.js image as the base image
FROM node:14-alpine

# Set the working directory inside the container
WORKDIR /app

# Copy package.json and package-lock.json files to the working directory
COPY package*.json ./

# Install dependencies
RUN npm install

# Copy the rest of the application files to the working directory
COPY . .

# Build the React app
RUN npm run build

# Expose port 3000 to the outside world
EXPOSE 3000

# Specify a placeholder directory as a volume mount point
VOLUME /app/public

# Command to run the React app when the container starts
CMD ["npm", "start"]
```

5. List the docker volume that was created for the container

```
TERMINAL
  GNU nano 6.2
# Use an official Node.js image as the base image
FROM node:14-alpine

# Set the working directory inside the container
WORKDIR /app

# Copy package.json and package-lock.json files to the working directory
COPY package*.json ./

# Install dependencies
RUN npm install

# Copy the rest of the application files to the working directory
COPY . .

# Build the React app
RUN npm run build

# Expose port 3000 to the outside world
EXPOSE 3000

# Specify a placeholder directory as a volume mount point
VOLUME /app/public

# Command to run the React app when the container starts
CMD ["npm", "start"]
```

```
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ docker volume ls
DRIVER    VOLUME NAME
local     0a2189a5b8d9515e8b81f8f26974f0f7a8026586ecbfa60b3aa27813e352d386
local     1aa99c7998e53ce368425f3515caa25b52dd5e220aaf794246b23209d655d528
local     3fbd9964ba056fe03a9d2d8d6b9c488d91a244ae508a1201206df127efd61e9a
local     5e4edd2842da63f9a89736ed52428f32bc05f037c6db77743fea295da23309d4
local     5e261a7e21bd4ecc939001644b0a5e01fbbdc1692dc122240c61d74c2ef8e0e9
local     05edb197e1dfe387023ffd5e3b7110f40bbda2a827655418b30ba7f3bf2e7ef6
local     6c0be79dd6496ea7ba7d3d6173312754f9c03aa7fcbb5912d32cb49c1cde1dc9
local     6cf2a3c65b766e856913dac86c71d45145c83799fa9d15e949fa424eabf12621
local     7b91fee886534522066b3ba347933ac879e0eafe58eb4667b5af0da4968e8a2e
local     7ee5262dc02f73bcf0ae087d43708eb99165f5f310b66cd08df3e2b52a1d3059
local     08debbd80f5e033c3e0321e7e4b3836cb59fbf28e1facc7e46e2dcc485ff99a5
local     9dcd1458e6fe173800c5826ef20907f9b2f865f100202186ef3bd3aac85303c4
local     54c73c319d2db9cb872b8b8b37986bd815dd5993bff648996b670c868062d28f
local     68b13afe74365e2bf32ca53f8ef528d76fe91a312608566d54aaa5dcbe39ceb8
local     97abcbc969f19a11e0924af12752dcec39c44d4aec9c7a43358f75cb91b09b40
local     304a6a3834b6514d9151506e321244672a73311eb61ff1b15aaa7d8019a6effa
local     616f405e937bdd0882c27357aebf3c5ab852f875cf2181ca699afee033fe0402
local     739c574380c4263d0981b601addc061738b53471cebd9715937e533488e8195d
local     0766aa059ad634ac260df7c2004d1f4510e2c40c12df49b83597f5eea895b3ac
local     2556b7a1e220d51d21f15c469a38f704fcdbdd7b1d4433f16fb79ecde389f20f
local     05138bf94d54f7f30c7ad36e8b988900a2fa9b264d1489ab63e89f1a1c98b31e
local     8404d9e99e573a42d673ebea14904faac02a56bce045e85d6d90a3a75cea4988
local     19086e7a7bf9aa6857b73a008438779a749f102f264697cdec9d37eaf2a4b72f
local     62409bf9b5c4d70fb2475b380e26787f83339a8905665c4b264c9ea08c1fb534
local     0529665e532cd15f29c3a33502a1458a6da8058f684b7f7458eb609eef3864b8
local     853898efd5160c003c670d30214f1cd0f91ff80b426e6976292d5ffd53616eff
local     a5abb55b23a1440bf0ebd889fdf910c235aee9ff368b7046f66f72492ba73b39
local     a16a7362fa8cce9eea52c664b1cb4beb59f7951256394170bc3e3e0c7db4dd05
local     a502ddde6beca5ab24a38f65356c247786d44879a6904d964847e3fb3e73d1bf
local     b7a9568bcbc9b165bc6e3078e66a656a635a931e37b8bfafcf353c4de1176f7d
local     b263f574aef61cf0acaeff8b6b8ac1e708b10b90df465e5ca550eb2cfbdd3ce0
local     c0f2b99f652a68a3b143c10f926dce2823aef0ef5e13049a6589cb3875353197
local     c456bb47d05d23d2c8481a9f876605d184d1e0c21647c57afa2506f97fa57f16
local     ce1585dd64300f41acd3cbf56b00f919a119280d0c5e2655820662d351e7c6eb
local     cf5814c2ed7d3fd763056d995f2dbc9e762f5dbfaa8587e2f87c00071f2c0d67
local     compose_orderer.example.com
local     compose_peer0.org1.example.com
local     compose_peer0.org2.example.com
local     ea0f9079f14841c3d22b13392b61bf3800955d55d6b7fbdad372477aa559828b
local     f3c43432c402d0cc972909acbb88a08a88c0ab64d1fa0213304e928200a4016f
local     f59b43a580ff8c648f771012364957b29f5dd461d31ad513e070fca1b0ab4ce4
local     f770fc3f8fadb5c992fd6a8c0ec9b6543b1abb434b15981a9df651b2ab720096
local     f9620abefbbbf3b3b2a70e9040c1395a924ef6d1612c32dcc844b5b2bd82dc72
local     fabric-blockchain-explorer_pgdata
local     minikube
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$
```

Question 3

1. Create a docker image using docker file for a simple web application. It can be as simple as a single html file.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

TERMINAL

  GNU nano 6.2
# Use an official Node.js image as the base image
FROM node:14-alpine

# Set the working directory inside the container
WORKDIR /app

# Copy package.json and package-lock.json files to the working directory
COPY package*.json ./

# Install dependencies
RUN npm install

# Copy the rest of the application files to the working directory
COPY . .

# Build the React app
RUN npm run build

# Expose port 3000 to the outside world
EXPOSE 3000
```

2. Deploy the application using deployment workload.

```
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ kubectl apply -f deployment.yaml
deployment.apps/my-react-app created
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ kubectl apply -f service.yaml
error: the path "service.yaml" does not exist
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ ls
README.md  deployment.yaml  dockerfile  node_modules  package-lock.json  package.json  public  service.yml  src
kj@Wdevice50:~/BCDV4034/BCDV4034/calculatorapp$ kubectl apply -f service.yml
service/my-react-app-service created
```

3. Enable loadbalancer as a service within the deployment workload.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
TERMINAL
  GNU nano 6.2
apiVersion: v1
kind: Service
metadata:
  name: my-react-app-service
spec:
  selector:
    app: my-react-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 3000
  type: LoadBalancer
```

4.

5.

6.