DevOps Project

You need to create an application which should consist of microservices (3 at least) and they communicate with each other and some (at least 1) or all of them should connect to any DB(MySQL, Postgres, Mongo, etc). If you are uncomfortable with coding, I have bootstrapped a sample app
https://github.com/kahootali/doctor-appointment-system
You can clone it and then create a private Github repo of your own, and copy and add files there.

Make sure the repo should be private and you add my user `kahootali` as a collaborator to your repo, if it's a public repo you will have 50% mark deduction.

The above repository has a base skeleton for a simple doctors & appointments application, it has

- Frontend
- Appointments: hardcoded appointments
- Doctors: hardcoded doctors

You need to remove hardcoding and use db for it, and have some sort of initial bootstrapping to add some initial values.

You need to submit

- Google Doc link containing all the items below and screenshots of implementation of below steps and actual running of them
- Repo link: **make sure my github user (kahootali) is added as collaborator, else I won't be able to access it and won't mark your project**

Github Repo: https://github.com/ameerahaider/Doctors-Appointment-App

Phase 1 (30):

- Need to dockerize the microservices, add dockerfiles for each microservice in their folder
- Build & push image to Dockerhub, share url of dockerhub repos

Appointments

```dockerfile
# Use the official Python 3.9 image as the base image
FROM python:3.9

# Set the working directory in the container
WORKDIR /app

# Copy the requirements.txt file to the container
COPY requirements.txt .

# Install the Python dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Copy the rest of the application code to the container
COPY . .

# Expose the port that the application will listen on
EXPOSE 7070

# Start application
CMD [ "python", "app.py" ]
```

```
ameera@ameera-VirtualBox:~/Desktop/Doctors-Appointment-App/appointments$ docker build -t amee
rahaider/appointments .
[+] Building 20.6s (10/10) FINISHED                                 docker:default
 => [internal] load .dockerignore                                           0.1s
 => => transferring context: 2B                                             0.0s
 => [internal] load build definition from Dockerfile                        0.1s
 => => transferring dockerfile: 515B                                        0.0s
 => [internal] load metadata for docker.io/library/python:3.9               2.6s
 => CACHED [1/5] FROM docker.io/library/python:3.9@sha256:b2e47a7eca3178e   0.0s
 => [internal] load build context                                           0.1s
 => => transferring context: 1.53kB                                         0.0s
 => [2/5] WORKDIR /app                                                      0.3s
 => [3/5] COPY requirements.txt .                                           0.2s
 => [4/5] RUN pip install --no-cache-dir -r requirements.txt               15.6s
 => [5/5] COPY . .                                                          0.5s
 => exporting to image                                                      0.7s
 => => exporting layers                                                     0.7s
 => => writing image sha256:86859d0f223603ac704de49a5ce82b6a0c00438d9d43e   0.0s
 => => naming to docker.io/ameerahaider/appointments                        0.0s
```

```
ameera@ameera-VirtualBox:~/Desktop/Doctors-Appointment-App/appointments$ docker push ameeraha
ider/appointments
Using default tag: latest
The push refers to repository [docker.io/ameerahaider/appointments]
eb30514c20c1: Pushed
5f57b1c36809: Pushed
1dfa55b55297: Pushed
08ed3be2a3d6: Pushed
d5ad3ac69862: Mounted from library/python
4929be171cab: Mounted from library/python
f021e1878a27: Mounted from library/python
a04a14a911a5: Mounted from library/python
80bd043d4663: Mounted from library/python
30f5cd833236: Mounted from library/python
7c32e0608151: Mounted from library/python
7cea17427f83: Mounted from library/python
latest: digest: sha256:69af796b2be839c31f81ec17e9882acc59f1ca114f765563c09b1a1b40574409 size:
 2838
```

- https://hub.docker.com/repository/docker/ameerahaider/appointments

Doctors

```dockerfile
# Use the official Python 3.9 image as the base image
FROM python:3.9

# Set the working directory in the container
WORKDIR /app

# Copy the requirements.txt file to the container
COPY requirements.txt .

# Install the Python dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Copy the rest of the application code to the container
COPY . .

# Expose the port that the application will listen on
EXPOSE 9090

# Start app
CMD [ "python", "app.py" ]
```

-
```
ameera@ameera-VirtualBox:~/Desktop/Doctors-Appointment-App/doctors$ docker build
 -t ameerahaider/doctors .
[+] Building 3.4s (11/11) FINISHED                              docker:default
 => [internal] load build definition from Dockerfile                      0.2s
 => => transferring dockerfile: 515B                                      0.0s
 => [internal] load .dockerignore                                         0.3s
 => => transferring context: 2B                                           0.0s
 => [internal] load metadata for docker.io/library/python:3.9             2.1s
 => [auth] library/python:pull token for registry-1.docker.io             0.0s
 => [1/5] FROM docker.io/library/python:3.9@sha256:b2e47a7eca3178e4ce6c09  0.0s
 => [internal] load build context                                         0.2s
 => => transferring context: 1.31kB                                       0.0s
 => CACHED [2/5] WORKDIR /app                                             0.0s
 => CACHED [3/5] COPY requirements.txt .                                  0.0s
 => CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt       0.0s
 => [5/5] COPY . .                                                        0.2s
 => exporting to image                                                    0.2s
 => => exporting layers                                                   0.2s
 => => writing image sha256:f00a5f1c9bdd6ab4c5a941207d20c9829d10aa6ffd1f4  0.0s
 => => naming to docker.io/ameerahaider/doctors                           0.0s
```
-
```
ameera@ameera-VirtualBox:~/Desktop/Doctors-Appointment-App/doctors$ docker push
ameerahaider/doctors
Using default tag: latest
The push refers to repository [docker.io/ameerahaider/doctors]
5eebb1cb22f3: Pushed
5f57b1c36809: Mounted from ameerahaider/appointments
1dfa55b55297: Mounted from ameerahaider/appointments
08ed3be2a3d6: Mounted from ameerahaider/appointments
d5ad3ac69862: Mounted from ameerahaider/appointments
4929be171cab: Mounted from ameerahaider/appointments
f021e1878a27: Mounted from ameerahaider/appointments
a04a14a911a5: Mounted from ameerahaider/appointments
80bd043d4663: Mounted from ameerahaider/appointments
30f5cd833236: Mounted from ameerahaider/appointments
7c32e0608151: Mounted from ameerahaider/appointments
7cea17427f83: Mounted from ameerahaider/appointments
latest: digest: sha256:1b43de7600bb4218f4f2e878816eca141f5aca4483acd38c85fd1793f
fe45f6a size: 2838
```
-
- https://hub.docker.com/repository/docker/ameerahaider/doctors

Frontend

```
# Use the official Node.js 14 image as the base image
FROM node:14

# Set the working directory in the container
WORKDIR /app

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

# Install the dependencies
RUN npm install

# Copy the rest of the application code to the container
COPY . .

# Expose the port that the application will listen on
EXPOSE 3000

# Start application
CMD [ "node", "app.js" ]
```

-

```
ameera@ameera-VirtualBox:~/Desktop/Doctors-Appointment-App/frontend$ docker buil
d -t ameerahaider/frontend .
[+] Building 2.8s (11/11) FINISHED                                docker:default
 => [internal] load build definition from Dockerfile                       0.1s
 => => transferring dockerfile: 615B                                       0.0s
 => [internal] load .dockerignore                                          0.2s
 => => transferring context: 2B                                            0.0s
 => [internal] load metadata for docker.io/library/node:14                 1.8s
 => [auth] library/node:pull token for registry-1.docker.io                0.0s
 => [1/5] FROM docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c59   0.0s
 => [internal] load build context                                          0.1s
 => => transferring context: 2.12kB                                        0.0s
 => CACHED [2/5] WORKDIR /app                                              0.0s
 => CACHED [3/5] COPY package*.json ./                                     0.0s
 => CACHED [4/5] RUN npm install                                           0.0s
 => [5/5] COPY . .                                                         0.3s
 => exporting to image                                                     0.2s
 => => exporting layers                                                    0.1s
 => => writing image sha256:35526f2928d91d683104023b15fdf28eed91dcdad94a0  0.0s
 => => naming to docker.io/ameerahaider/frontend                           0.0s
```

```
ameera@ameera-VirtualBox:~/Desktop/Doctors-Appointment-App/frontend$ docker push
 ameerahaider/frontend
Using default tag: latest
The push refers to repository [docker.io/ameerahaider/frontend]
942b7628c12f: Pushed
c6fc30914e43: Pushed
7e10bf820044: Pushed
5410bacca143: Pushed
0d5f5a015e5d: Mounted from library/node
3c777d951de2: Mounted from library/node
f8a91dd5fc84: Mounted from library/node
cb81227abde5: Mounted from library/node
e01a454893a9: Mounted from library/node
c45660adde37: Mounted from library/node
fe0fb3ab4a0f: Mounted from library/node
f1186e5061f2: Mounted from library/node
b2dba7477754: Mounted from library/node
latest: digest: sha256:6080afacf87ab31f56f4ec748980e28798cb9440db0e9a8a757e99c48
5d7cd65 size: 3047
```

-
- https://hub.docker.com/repository/docker/ameerahaider/frontend
- Add a single docker-compose file at root path which will start all microservices, networks and db on a single deployment

```yaml
version: '3.9'

services:

  doctors:
    build:
      context: doctors/
      dockerfile: Dockerfile
    init: true
    deploy:
      replicas: 2
    networks:
      - mynet

  appointments:
    build:
      context: appointments/
      dockerfile: Dockerfile
    init: true
    deploy:
      replicas: 2
    networks:
      - mynet

  frontend:
    build:
      context: frontend/
```

```
    dockerfile: Dockerfile
  init: true
  environment:
    DOCTORS_SERVICE_URL: doctors:9090
    APPOINTMENTS_SERVICE_URL: appointments:7070
  deploy:
    replicas: 1
  networks:
    - mynet
  ports:
    - "3000:3000"
  depends_on:
    - doctors
    - appointments

db:
  image: mysql:latest
  ports:
    - 3306:3306
  environment:
    MYSQL_ROOT_PASSWORD: example
    MYSQL_DATABASE: mydatabase
    MYSQL_USER: myuser
    MYSQL_PASSWORD: mypassword
  volumes:
    - ./data:/var/lib/mysql
```

-

```
networks:
  mynet:
    driver: bridge
```

-

Phase 2 (30):

Add CI/CD to the repo, can use Jenkins or Github Actions, which will

- Pipeline should run on **Pull Request** & **on main branch**
- **For versioning, you can use the commit hash or semantic versioning or any other way (but dont use latest tag)**
- Build & push docker images with respective version of the microservices to dockerhub

- Use path based filtering so that if there is a change in the **appointments** folder, only the pipeline will build appointments microservice.
- **Which means there should be separate pipelines for all microservices and if a change is in one microservice, only that pipeline should be triggered**
- Update docker-compose file to update the image tag to the one that is built in this pipeline and push back to repo

Appointments

```yaml
name: CI/CD Pipeline for Appointments

on:
  pull_request:
    branches:
      - '*'
    paths:
      - 'appointments/**'
  push:
    branches:
      - main
    paths:
      - 'appointments/**'

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Build and push Docker image
      run: |
        DOCKER_VERSION=$(git rev-parse --short HEAD)
        APPOINTMENT_VERSION=${DOCKER_VERSION}
        echo "APPOINTMENT_VERSION=${APPOINTMENT_VERSION}" >> $GITHUB_ENV
        docker build -t ${{ secrets.DOCKER_USERNAME }}/appointments:${DOCKER_VERSION} ./appointments
        docker login -u ${{ secrets.DOCKER_USERNAME }} -p ${{ secrets.DOCKER_PASSWORD }}
        docker push ${{ secrets.DOCKER_USERNAME }}/appointments:${DOCKER_VERSION}

    - name: Update .env file with APPOINTMENT_VERSION
      run: echo "APPOINTMENT_VERSION=${APPOINTMENT_VERSION}" >> .env
```

Doctors

```yaml
name: CI/CD Pipeline for Doctors

on:
  pull_request:
    branches:
      - '*'
    paths:
      - 'doctors/**'
  push:
    branches:
      - main
    paths:
      - 'doctors/**'

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Build and push Docker image
      run: |
        DOCKER_VERSION=$(git rev-parse --short HEAD)
        DOCTORS_VERSION=${DOCKER_VERSION}
        echo "DOCTORS_VERSION=${DOCTORS_VERSION}" >> $GITHUB_ENV
        docker build -t ${{ secrets.DOCKER_USERNAME }}/doctors:${DOCKER_VERSION} ./doctors
        docker login -u ${{ secrets.DOCKER_USERNAME }} -p ${{ secrets.DOCKER_PASSWORD }}
        docker push ${{ secrets.DOCKER_USERNAME }}/doctors:${DOCKER_VERSION}

    - name: Update .env file with DOCTORS_VERSION
      run: echo "DOCTORS_VERSION=${DOCTORS_VERSION}" >> .env
```

Frontend

```yaml
name: CI/CD Pipeline for Frontend

on:
  pull_request:
    branches:
      - '*'
    paths:
      - 'frontend/**'
  push:
    branches:
      - main
    paths:
      - 'frontend/**'

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Build and push Docker image
      run: |
        DOCKER_VERSION=$(git rev-parse --short HEAD)
        FRONTEND_VERSION=${DOCKER_VERSION}
        echo "FRONTEND_VERSION=${FRONTEND_VERSION}" >> $GITHUB_ENV
        docker build -t ${{ secrets.DOCKER_USERNAME }}/frontend:${DOCKER_VERSION} ./frontend
        docker login -u ${{ secrets.DOCKER_USERNAME }} -p ${{ secrets.DOCKER_PASSWORD }}
        docker push ${{ secrets.DOCKER_USERNAME }}/frontend:${DOCKER_VERSION}

    - name: Update .env file with FRONTEND_VERSION
      run: echo "FRONTEND_VERSION=${FRONTEND_VERSION}" >> .env
```

Updated Docker-Compose

```yaml
version: '3.9'

services:

  doctors:
    image: ${DOCKER_USERNAME}/doctors:${DOCTORS_VERSION}
    build:
      context: doctors/
      dockerfile: Dockerfile
    init: true
    deploy:
      replicas: 2
    networks:
      - mynet
    env_file:
      - .env

  appointments:
    image: ${DOCKER_USERNAME}/appointments:${APPOINTMENT_VERSION}
    build:
      context: appointments/
      dockerfile: Dockerfile
    init: true
    deploy:
      replicas: 2
    networks:
      - mynet
    env_file:
      - .env

  frontend:
    image: ${DOCKER_USERNAME}/frontend:${FRONTEND_VERSION}
    build:
      context: frontend/
      dockerfile: Dockerfile
    init: true
```

```
    environment:
      DOCTORS_SERVICE_URL: doctors:9090
      APPOINTMENTS_SERVICE_URL: appointments:7070
    deploy:
      replicas: 1
    networks:
      - mynet
    ports:
      - "3000:3000"
    depends_on:
      - doctors
      - appointments
    env_file:
      - .env

  db:
    image: mysql:latest
    ports:
      - 3306:3306
    environment:
      MYSQL_ROOT_PASSWORD: example
      MYSQL_DATABASE: mydatabase
      MYSQL_USER: myuser
      MYSQL_PASSWORD: mypassword
    volumes:
      - ./data:/var/lib/mysql

networks:
  mynet:
    driver: bridge
```

Phase 3 (30):

- Add Kubernetes Deployment & Service for all the microservices in respective folder in a single file e.g.

appointments/k8s/app.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: appointments-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: appointments
  template:
    metadata:
      labels:
        app: appointments
    spec:
      containers:
        - name: appointments
          image: ameerahaider/appointments:latest
          ports:
            - containerPort: 7070
---
apiVersion: v1
kind: Service
metadata:
  name: appointments-service
spec:
  selector:
    app: appointments
  ports:
    - protocol: TCP
      port: 7070
      targetPort: 7070
```

-

doctors/k8s/app.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: doctors-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: doctors
  template:
    metadata:
      labels:
        app: doctors
    spec:
      containers:
        - name: doctors
          image: ameerahaider/doctors:latest
          ports:
            - containerPort: 9090
---
apiVersion: v1
kind: Service
metadata:
  name: doctors-service
spec:
  selector:
    app: doctors
  ports:
    - protocol: TCP
      port: 9090
      targetPort: 9090S
```

-

frontend/k8s/app.yaml

```yaml
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: ameerahaider/frontend:latest
          ports:
            - containerPort: 3000
          env:
            - name: DOCTORS_SERVICE_URL
              value: "10.109.51.236:9090"
            - name: APPOINTMENTS_SERVICE_URL
              value: "10.111.250.3:7070"
---
apiVersion: v1
kind: Service
metadata:
  name: frontend-service
spec:
  selector:
    app: frontend
  ports:
    - protocol: TCP
      port: 3000
      targetPort: 3000
```

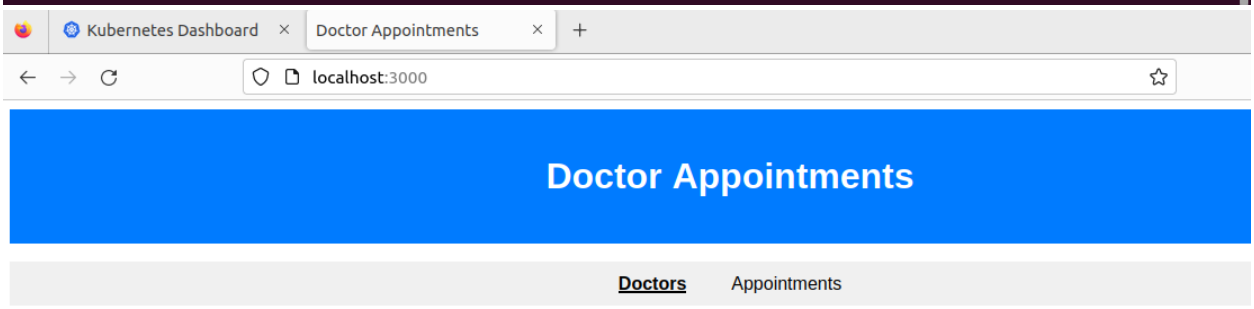
-

- Make sure you are following best practices of using environment variables for inter-communication, and how to pass those env vars to your pod in best way.

Running Services/Pods/Deployments

-
```
ameera@ameera-VirtualBox:~/Desktop/Doctors-Appointment-App/appointments$ kubectl get services
NAME                   TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
appointments-service   ClusterIP   10.111.250.3    <none>        7070/TCP   128m
doctors-service        ClusterIP   10.109.51.236   <none>        9090/TCP   18m
frontend-service       ClusterIP   10.98.166.148   <none>        3000/TCP   10m
kubernetes             ClusterIP   10.96.0.1       <none>        443/TCP    147m
ameera@ameera-VirtualBox:~/Desktop/Doctors-Appointment-App/appointments$ kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
appointments-deployment-6f449c8548-c8knn   1/1     Running   0          22m
doctors-deployment-65457d88cd-22w2z        1/1     Running   0          18m
frontend-deployment-5bdf5dfdcc-gqdc9       1/1     Running   0          10m
ameera@ameera-VirtualBox:~/Desktop/Doctors-Appointment-App/appointments$ kubectl get deployments
NAME                     READY   UP-TO-DATE   AVAILABLE   AGE
appointments-deployment   1/1     1            1           22m
doctors-deployment        1/1     1            1           18m
frontend-deployment       1/1     1            1           10m
```

-
```
^Cameera@ameera-VirtualBox:~/Desktop/Doctors-Appointment-App/appointments$ kubectl port-forward se
rvice/frontend-service 3000:3000
Forwarding from 127.0.0.1:3000 -> 3000
Forwarding from [::1]:3000 -> 3000
Handling connection for 3000
Handling connection for 3000
```

- 

-

Phase 4 (20):

- All of the microservice's Deployment should have K8s Resources for request and limits on memory & CPU (need to read it yourself), add K8s Probes(readiness & liveness) to your container

Updated appointments/k8s/app.yaml

Open

app.yaml
~/Desktop/Doctors-Appointment-App/appointments/k8s

```yaml
        - name: appointments
          image: ameerahaider/appointments:latest
          ports:
            - containerPort: 7070
          resources:
            limits:
              memory: "256Mi"
              cpu: "200m"
            requests:
              memory: "128Mi"
              cpu: "100m"
          livenessProbe:
            httpGet:
              path: /hello
              port: 7070
            initialDelaySeconds: 3
            periodSeconds: 3
          readinessProbe:
            httpGet:
              path: /hello
              port: 7070
            initialDelaySeconds: 5
            periodSeconds: 5
---
apiVersion: v1
kind: Service
metadata:
  name: appointments-service
spec:
  selector:
```

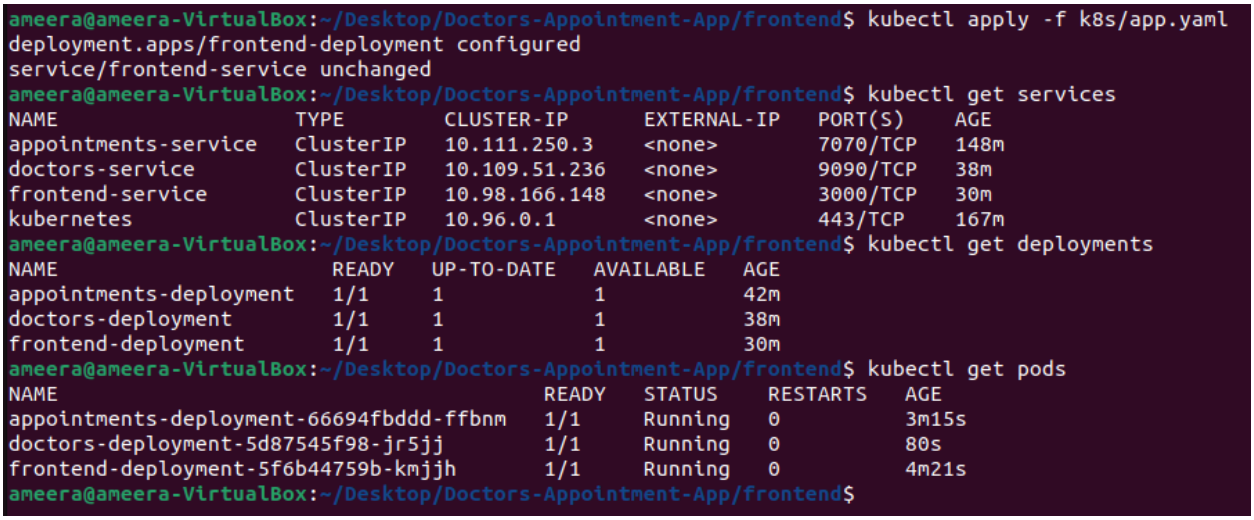
Updated doctors/k8s/app.yaml

```
        app: doctors
    spec:
      containers:
        - name: doctors
          image: ameerahaider/doctors:latest
          ports:
            - containerPort: 9090
          resources:
            limits:
              memory: "256Mi"
              cpu: "200m"
            requests:
              memory: "128Mi"
              cpu: "100m"
          livenessProbe:
            httpGet:
              path: /hello
              port: 9090
            initialDelaySeconds: 3
            periodSeconds: 3
          readinessProbe:
            httpGet:
              path: /hello
              port: 9090
            initialDelaySeconds: 5
            periodSeconds: 5
---
- apiVersion: v1
```

Updated frontend/k8s/app.yaml

```yaml
    ports:
      - containerPort: 3000
    env:
      - name: DOCTORS_SERVICE_URL
        value: "10.109.51.236:9090"
      - name: APPOINTMENTS_SERVICE_URL
        value: "10.111.250.3:7070"
    resources:
      limits:
        memory: "256Mi"
        cpu: "200m"
      requests:
        memory: "128Mi"
        cpu: "100m"
    livenessProbe:
      httpGet:
        path: /hello
        port: 3000
      initialDelaySeconds: 3
      periodSeconds: 3
    readinessProbe:
      httpGet:
        path: /hello
        port: 3000
      initialDelaySeconds: 5
      periodSeconds: 5
---
```

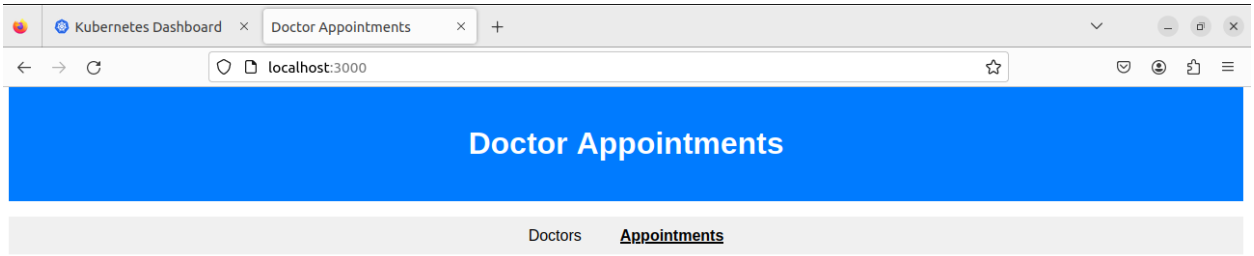
Updated Running Services/Pods/Deployments



```
ameera@ameera-VirtualBox:~/Desktop/Doctors-Appointment-App/frontend$ kubectl apply -f k8s/app.yaml
deployment.apps/frontend-deployment configured
service/frontend-service unchanged
ameera@ameera-VirtualBox:~/Desktop/Doctors-Appointment-App/frontend$ kubectl get services
NAME                   TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
appointments-service   ClusterIP   10.111.250.3    <none>        7070/TCP   148m
doctors-service        ClusterIP   10.109.51.236   <none>        9090/TCP   38m
frontend-service       ClusterIP   10.98.166.148   <none>        3000/TCP   30m
kubernetes             ClusterIP   10.96.0.1       <none>        443/TCP    167m
ameera@ameera-VirtualBox:~/Desktop/Doctors-Appointment-App/frontend$ kubectl get deployments
NAME                     READY   UP-TO-DATE   AVAILABLE   AGE
appointments-deployment   1/1     1            1           42m
doctors-deployment        1/1     1            1           38m
frontend-deployment       1/1     1            1           30m
ameera@ameera-VirtualBox:~/Desktop/Doctors-Appointment-App/frontend$ kubectl get pods
NAME                                      READY   STATUS    RESTARTS   AGE
appointments-deployment-66694fbddd-ffbnm   1/1     Running   0          3m15s
doctors-deployment-5d87545f98-jr5jj        1/1     Running   0          80s
frontend-deployment-5f6b44759b-kmjjh       1/1     Running   0          4m21s
ameera@ameera-VirtualBox:~/Desktop/Doctors-Appointment-App/frontend$
```

-

```
ameera@ameera-VirtualBox:~/Desktop/Doctors-Appointment-App/frontend$ kubectl port-forward service/frontend-ser
vice 3000:3000
Forwarding from 127.0.0.1:3000 -> 3000
Forwarding from [::1]:3000 -> 3000
Handling connection for 3000
Handling connection for 3000
```



Doctor Appointments

Doctors Appointments

Appointments List

ID	Doctor	Date	Rating
1	1	21 Nov 2023	Good
2	1	22 Nov 2023	Bad
3	2	22 Nov 2023	Good
4	1	22 Nov 2023	Bad
5	2	22 Nov 2023	Good

-

Bonus 1

- **Appointments microservice should have environment variables NAME and PASSWORD that should be set from a Secret named appointment**
- **Doctors microservice should mount a file details.txt on path /user/details.txt that should be set from Configmap named doctors where you can have an intro about yourself.**
- **The Configmap & Secret manifests should be added to respective file as mentioned in Phase 3**

Bonus 2

- **Pod should run as non-root user and follow hardened container best practices**
- **Add initContainer(read yourself) based on ubuntu image, which will print your name and sleep for 5 seconds and exit and then main container will start**