1. CONTAINERIZATION:

* Backend dockerfile-
* # build stage  
  FROM maven:3.8.7-openjdk-18 AS *build*WORKDIR /build  
  COPY /book-network/pom.xml .  
  RUN mvn dependency:go-offline  
  COPY /book-network/src ./src  
  RUN mvn clean package -DskipTests  
    
  # runtime stage  
  FROM amazoncorretto:17  
  #argument for passing values through env  
  ARG *PROFILE*=dev  
  ARG *APP\_VERSION*=1.0.0  
  WORKDIR /app  
  COPY --from=*build* /build/target/book-network-\*.jar /app/  
  EXPOSE 9000  
  ENV *DB\_URL*=jdbc:postgresql://postgres\_sql\_bsn:5432/book\_social\_network  
  ENV *ACTIVE\_PROFILE*=${*PROFILE*}  
  ENV *JAR\_VERSION*=${*APP\_VERSION*}  
  CMD java -jar -Dspring.profiles.active=${*ACTIVE\_PROFILE*} -Dspring.datasource.url=${*DB\_URL*} book-network-${*JAR\_VERSION*}.jar

The ‘bsn-api’ service is the image that is built from this dockerfile.

* Create ngnix.conf in the frontend root foler and put contents below(this is for angular, react may be slightly different)-

events {}

http {

server {

listen 80;

location / {

root /usr/share/nginx/html;

index index.html;

try\_files $uri $uri/ /index.html;

}

# Handle JavaScript | CSS files with the correct MIME type

location ~ \.(js|css)$ {

root /usr/share/nginx/html;

include /etc/nginx/mime.types;

}

# Catch-all location to handle refreshes on any route

location ~ ^/.+$ {

root /usr/share/nginx/html;

index index.html;

try\_files $uri /index.html;

}

# Backend reverse proxy

location /api {

proxy\_pass http://bsn-api:8088;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

}

}

}

* Frontend dockerfile-

FROM node:16 as build-stage

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

RUN npm run build

FROM nginx:alpine

COPY nginx.conf /etc/nginx/nginx.conf

COPY --from=build-stage /app/dist/book-network-ui /usr/share/nginx/html

EXPOSE 80

PUT frontend image in the docker-compose which is ‘bsn-ui’

* Docker-compose-

services:

postgres:

container\_name: postgres-sql-bsn

image: postgres

environment:

POSTGRES\_USER: username

POSTGRES\_PASSWORD: password

PGDATA: /var/lib/postgresql/data

POSTGRES\_DB: book\_social\_network

volumes:

- postgres:/data/postgres

ports:

- 5432:5432

networks:

- spring-demo

restart: unless-stopped

mail-dev:

container\_name: mail-dev-bsn

image: maildev/maildev

ports:

- 1080:1080

- 1025:1025

bsn-api:

container\_name: bsn-api

image: bsn/bsn:1.0.8

ports:

- 8088:8088

networks:

- spring-demo

depends\_on:

- postgres

bsn-ui:

container\_name: bsn-ui

image: bsn/bsn-ui:1.0.0

ports:

- 8080:80

networks:

- spring-demo

depends\_on:

- bsn-api

networks:

spring-demo:

driver: bridge

volumes:

postgres:

driver: local

// ignore maildev service, we used gmail later

* Docker-compose up –d (run containers) // just to test
* Get username and token from dockerhub
* Set some secrets in github like dockerhub username,token, email username, password,hostname, vps username,vps ip
* Create ci/cd pipeline for backend –

name: BSN Backend API Pipeline  
  
on:  
 push:  
 branches:  
 - main  
 paths:  
 - book-network/\*\*  
 - Dockerfile  
 - 'docker-compose.yml'  
 - .github/workflows/\*-backend.yml  
jobs:  
 compile:  
 runs-on: self-hosted  
 name: Compile project  
 steps:  
 - name: Checkout code  
 uses: actions/checkout@v4  
 with:  
 fetch-depth: 0  
 - name: Setup jdk  
 uses: actions/setup-jdk@v4  
 with:  
 java-version: 17  
 distribution: "corretto"  
 - name: Compile code  
 run: |  
 cd book-network  
 ./mvnw clean compile  
   
 unit-tests:  
 runs-on: self-hosted  
 name: Unit tests  
 steps:  
 - name: Checkout code  
 uses: actions/checkout@v4  
 with:  
 fetch-depth: 0  
 - name: Setup jdk  
 uses: actions/setup-jdk@v4  
 with:  
 java-version: 17  
 distribution: "corretto"  
 - name: Running unit tests  
 run: |  
 cd book-network  
 ./mvnw clean test  
 build:  
 runs-on: self-hosted  
 name: Build backend  
 needs: [ compile, unit-tests ]  
 steps:  
 - name: Checkout code  
 uses: actions/checkout@v4  
 with:  
 fetch-depth: 0  
 - name: Setup jdk  
 uses: actions/setup-jdk@v4  
 with:  
 java-version: 17  
 distribution: "corretto"  
 - name: Build project  
 run: |  
 cd book-network  
 ./mvnw clean package  
 build-image:  
 name: Build docker image  
 runs-on: self-hosted  
 needs: [ build ]  
 steps:  
 - name: Checkout code  
 uses: actions/checkout@v4  
 with:  
 fetch-depth: 0  
 - name: Setup jdk  
 uses: actions/setup-jdk@v4  
 with:  
 java-version: 17  
 distribution: "corretto"  
 - name: Build project  
 run: |  
 cd book-network  
 ./mvnw clean package  
 - name: Extract project version  
 id: extract\_version *# to use later by id  
 # getting project version* run: |  
 cd book-network  
 echo "VERSION=$(./mvnw -q -Dexec.executable='echo' -Dexec.args='${project.version}' --non-recursive exec:exec)" >> $GITHUB\_OUTPUT  
 - name: Login to docker hub  
 uses: docker/login-action@v3  
 with:  
 username: ${{ secrets.DOCKERHUB\_USERNAME }}  
 password: ${{ secrets.DOCKERHUB\_TOKEN }}  
 - name: Build & Push to Dockerhub  
 uses: docker/build-push-action@v5  
 with:  
 context: book-network  
 file: Dockerfile  
 push: true  
 platform: linux/amd64  
 tags: ${{ secrets.DOCKERHUB\_USERNAME }}/bsn-api:${{ steps.extract\_version.outputs.VERSION }},${{ secrets.DOCKERHUB\_USERNAME }}/bsn-api:latest  
 build-args: |  
 PROFILE=dev  
 APP\_VERSION=${{ steps.extract\_version.outputs.VERSION }}  
 deploy:  
 name: Deploy backend  
 runs-on: self-hosted  
 needs: [build-image]  
 steps:  
 - name: Create deployment folder  
 run: ssh ${{ secrets.VPS\_USERNAME}}@${{ secrets.VPS\_IP}} "mkdir -p ci-cd" *# connect to instance and making folder* - name: Copy docker-compose file  
 run: scp docker-compose.yml ${{ secrets.VPS\_USERNAME}}@${{ secrets.VPS\_IP}}:ci-cd/docker-compose.yml  
 - name: Set ENV variables and deploy  
 run: |  
 ssh ${{ secrets.VPS\_USERNAME}}@${{ secrets.VPS\_IP}} <<EOF  
 export EMAIL\_HOSTNAME=${{ secrets.EMAIL\_HOSTNAME}}  
 export EMAIL\_USERNAME=${{ secrets.EMAIL\_USERNAME}}  
 export EMAIL\_PASSWORD=${{ secrets.EMAIL\_PASSWORD}}  
 cd ci-cd  
 docker-compose -f docker-compose.yml pull -q  
 docker-compose -f docker-compose.yml up -d  
 EOF

* Create ci/cd pipleline for frontend –

(same as backend, just change dockerfile location)

* Push code to main branch
* finsh