컨테이너화 및 배포 가이드

1. 컨테이너 구성

- Frontend: React + Nginx
- Backend: Spring Boot
- AI Module: Python + FastAPI
- Database: MariaDB

2. Docker 설정

2.1.1 Frontend Dockerfile

```
# 1단계: Vite 앱 빌드
FROM node:18-alpine AS build
WORKDIR /app

COPY package*.json ./
RUN npm install

COPY . .
RUN npm run build -- --mode production

# 2단계: Nginx로 서빙
FROM nginx:alpine
COPY docker/nginx.conf /etc/nginx/conf.d/default.conf
COPY --from=build /app/dist /usr/share/nginx/html

EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

2.1.2 Nginx.conf

```
server {
  listen 80;
  server_name localhost;

  root /usr/share/nginx/html;
  index index.html;
```

```
  location / {
    try_files $uri /index.html;
  }

  location /api/ {
    proxy_pass http://health-backend:8080/api/;
    proxy_http_version 1.1;
    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;


    add_header 'Access-Control-Allow-Origin' 'http://13.56.184.142' always;
    add_header 'Access-Control-Allow-Credentials' 'true' always;
    add_header 'Access-Control-Allow-Headers' 'Authorization,Content-Type,Set-
Cookie' always;
    add_header 'Access-Control-Allow-Methods' 'GET,POST,PUT,DELETE,OPTIONS'
always;

    if ($request_method = OPTIONS ) {
        add_header Content-Length 0;
        add_header Content-Type text/plain;
        return 204;
    }
  }
    location /nginx_status {
        stub_status on;
        access_log off;
    }
}
```

2.2 Backend Dockerfile

```
FROM maven:3.9.6-eclipse-temurin-17 AS build
WORKDIR /app
COPY . .
RUN ./mvnw clean package -DskipTests

FROM openjdk:17
```

```
VOLUME /tmp
COPY --from=build /app/target/Thridprojectback.jar app.jar
ENTRYPOINT ["java", "-jar", "/app.jar", "--spring.profiles.active=prod"]
```

2.3 AI Module Dockerfile

```
FROM python:3.12-slim

# system dependencies
RUN apt-get update && apt-get install -y build-essential && rm -rf
/var/lib/apt/lists/*

# create working dir
WORKDIR /app

# copy poetry config and project files
COPY pyproject.toml poetry.lock* ./

# install poetry
RUN pip install --upgrade pip && pip install poetry

# install dependencies (no virtualenvs)
RUN poetry config virtualenvs.create false && poetry install --no-root

# copy the rest of the code
COPY . .

EXPOSE 8003
CMD ["uvicorn", "server.main:app", "--host", "0.0.0.0", "--port", "8003"]
```

3. Docker Compose (로컬 실행용)

```
services:
  backend:
    build: ./ThirdprojectBack
    container_name: health-backend
    ports:
      - "8080:8080"
    depends_on:
      - mariadb
    environment:
```

```
      - SPRING_PROFILES_ACTIVE=prod
      - DB_HOST=mariadb
      - DB_PORT=3306
      - DB_DATABASE=healthdb
      - DB_USERNAME=gym
      - DB_PASSWORD=gym
      - AI_SERVER_URL=http://ai-server:8003/healthai/invoke
      - YOUTUBE_API_KEY=${YOUTUBE_API_KEY}
    networks:
      - health-net

  frontend:
    build: ./frontend
    container_name: health-frontend
    ports:
      - "80:80"
    networks:
      - health-net

  ai-server:
    build: ./ai
    container_name: health-ai
    ports:
      - "8003:8003"
    env_file:
      - ./ai/ai/.env
    networks:
      - health-net

  mariadb:
    image: mariadb:10.9
    container_name: health-db
    restart: always
    environment:
      - MYSQL_ROOT_PASSWORD=root
      - MYSQL_DATABASE=healthdb
      - MYSQL_USER=gym
      - MYSQL_PASSWORD=gym
    ports:
      - "3306:3306"
    volumes:
```

```
    - mariadb-data:/var/lib/mysql
  networks:
    - health-net

prometheus:
  image: prom/prometheus
  container_name: prometheus
  volumes:
    - ./monitoring/prometheus.yml:/etc/prometheus/prometheus.yml
  command:
    - '--config.file=/etc/prometheus/prometheus.yml'
  ports:
    - "9090:9090"
  networks:
    - health-net

grafana:
  image: grafana/grafana
  container_name: grafana
  ports:
    - "3010:3000"
  environment:
    - GF_SECURITY_ADMIN_USER=shieldus
    - GF_SECURITY_ADMIN_PASSWORD=shieldus
  volumes:
    - grafana-storage:/var/lib/grafana
  networks:
    - health-net

node-exporter:
  image: prom/node-exporter
  container_name: node-exporter
  ports:
    - "9113:9100"
  networks:
    - health-net

nginx-exporter:
  image: nginx/nginx-prometheus-exporter:latest
  container_name: nginx-exporter
  command:
```

```
      - '-nginx.scrape-uri=http://frontend/nginx_status'
    ports:
      - "9114:9113"
    depends_on:
      - frontend
    networks:
      - health-net

volumes:
  mariadb-data:
  grafana-storage:

networks:
  health-net:
    driver: bridge
```

4. 환경 변수

4.1 개발 환경

```
spring.datasource.url=jdbc:mariadb://127.0.0.1:3306/healthdb
spring.datasource.username=gym
spring.datasource.password=gym
spring.datasource.driverClassName=org.mariadb.jdbc.Driver

spring.jpa.hibernate.ddl-auto=create
spring.jpa.show-sql=true
spring.jpa.database-platform=org.hibernate.dialect.MariaDBDialect

logging.level.org.springframework.security=DEBUG
logging.level.org.hibernate.SQL=DEBUG
logging.level.org.hibernate.orm.jdbc.bind=TRACE

ai.server.url=http://localhost:8003/healthai/invoke
```

4.2 운영 환경

```
# DB 설정
spring.datasource.url=jdbc:mariadb://${DB_HOST}:${DB_PORT}/${DB_DATABASE}
spring.datasource.username=${DB_USERNAME}
spring.datasource.password=${DB_PASSWORD}
```

```
spring.datasource.driverClassName=org.mariadb.jdbc.Driver

# JPA 설정
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.show-sql=true
spring.jpa.database-platform=org.hibernate.dialect.MariaDBDialect

# Actuator + Prometheus
management.endpoints.web.exposure.include=*
management.endpoint.prometheus.enabled=true
management.metrics.export.prometheus.enabled=true
management.metrics.enable.jvm=true
management.metrics.tags.application=health-backend
```

5. 클라우드 배포

5.1 AWS EC2

```
cd ~/health-ai
docker compose pull
docker compose up -d
docker image prune -a
# git hub action의 script입니다.
```

6. CI/CD

```
name: Build and Push All Service Images

on:
  push:
    branches: [main]

jobs:
  build-and-push:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout source code
        uses: actions/checkout@v3
```

```
- name: Set up Docker Buildx
  uses: docker/setup-buildx-action@v3

- name: Login to DockerHub
  uses: docker/login-action@v3
  with:
    username: ${{ secrets.DOCKER_USERNAME }}
    password: ${{ secrets.DOCKER_PASSWORD }}

- name: Create .env.production
  run: |
    echo "VITE_API_URL=/api" > frontend/.env.production
# backend
- name: Build and push backend image
  uses: docker/build-push-action@v5
  with:
    context: ./ThirdprojectBack
    push: true
    tags: ${{ secrets.DOCKER_USERNAME }}/health-ai:backend-v1

# frontend
- name: Build and push frontend image
  uses: docker/build-push-action@v5
  with:
    context: ./frontend
    push: true
    tags: ${{ secrets.DOCKER_USERNAME }}/health-ai:frontend-v1

# ai server
- name: Build and push AI server image
  uses: docker/build-push-action@v5
  with:
    context: ./ai
    push: true
    tags: ${{ secrets.DOCKER_USERNAME }}/health-ai:ai-v1
# Deploy to EC2 via SSH
- name: Deploy on EC2
  uses: appleboy/ssh-action@v1.0.0
  with:
    host: ${{ secrets.EC2_HOST }}
    username: ${{ secrets.EC2_USER }}
```

```
        key: ${{ secrets.EC2_SSH_KEY }}
        script: |
          cd ~/health-ai
          docker compose pull
          docker compose up -d
          docker image prune -a
        timeout: 20m
```

7. 네트워크

- Frontend: 포트 80
- Backend: 포트 8080
- AI Module: 포트 8003
- Database: 포트 3306
- Prometheus: 포트 9090
- Grafana: 포트 3010
- Node Exporter: 포트 9113
- Nginx Exporter: 포트 9114

8. 완료 체크리스트

- [o] Docker 이미지 빌드
- [o] 컨테이너 실행
- [o] 네트워크 연결 확인
- [o] 헬스체크 통과
- [o] 클라우드 배포