포팅 매뉴얼

목차

1. Outline

1.1. 프로젝트 사용 도구

이슈 관리 : JIRA

형상 관리 : GitLab

커뮤니케이션 : Mattermost

디자인 : Figma

UCC : Adobe Premiere, Mango Board

CI/CD : Jenkins

1.2. 개발 환경

Visual Studio Code : 1.81.1

IntelliJ IDEA Ultimate : 17.0.7+10-b8291.16 amd64

JDK : Corretto-17.0.7.7.1 LTS

Node.js : 18.17.0

React : 18.2.0

Python : 3.7.16

Mysql : 8.0.34

JPA : 3.1.2

AWS EC2 : Ubuntu 20.04.6 LTS

Docker : 24.0.5

Docker-Compose : 1.24.0

Jenkins : 2.416

NGINX : 1.18.0 (Ubuntu)

Swagger : 2.2.9

AR.js : 3.4.5

Aframe : 1.3.0

1.3. .gitignore

/front-end/.gitignore

```
# dependencies
/node_modules
/.pnp
.pnp.js

# testing
/coverage

# production
/build

# misc
.DS_Store
.env.local
.env.development.local
.env.test.local
.env.production.local

npm-debug.log*
yarn-debug.log*
yarn-error.log*

# .dockerignore
.git
node_modules
.gitignore
```

/back-end/lions/.gitignore

```
HELP.md
.gradle
build/
!gradle/wrapper/gradle-wrapper.jar
```

```
!**/src/main/**/build/
!**/src/test/**/build/

### STS ###
.apt_generated
.classpath
.factorypath
.project
.settings
.springBeans
.sts4-cache
bin/
!**/src/main/**/bin/
!**/src/test/**/bin/

### IntelliJ IDEA ###
.idea
*.iws
*.iml
*.ipr
out/
!**/src/main/**/out/
!**/src/test/**/out/

### NetBeans ###
/nbproject/private/
/nbbuild/
/dist/
/nbdist/
/.nb-gradle/

### VS Code ###
.vscode/
```

---

2. Install

2.1. Docker

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

2.2. Jenkins

```
sudo docker pull jenkins/jenkins:lts
```

```
sudo docker run -d -p 9090:9090 -v /var/jenkins:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock --name jenkins -u root
```

2.3. MySql

```
docker pull mysql:latest
docker volume create mysql-volume
$ docker run -d --name mysql-container -p 3306:3306 -v mysql-volume:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=1234 mysql:latest
```

---

3. Detail

3.1. Docker-Compose

~/docker-compose.yaml

```
version: '3'
services:
  api:
    container_name: api-container
    image: kimta2hwan/api
    expose:
      - 8080
    volumes:
      - ./secret:/secret

  web:
    container_name: web-container
    image: kimta2hwan/web
    expose:
      - 3000
    volumes:
      - ./images:/images

  nginx:
    container_name: nginx-container
    image: nginx:latest
    restart: always
    volumes:
      - ./conf/:/etc/nginx/conf.d
      - ./etc/letsencrypt:/etc/;letsencrypt
      - ./images:/home/root/images
    ports:
      - 80:80
      - 443:443
    depends_on:
      - api
      - web
```

3.2. Dockerfile

front-end

```
FROM node:alpine as builder
WORKDIR /usr/src/app
COPY package.json .
RUN npm install
COPY ./ ./
RUN npm run build

FROM nginx
EXPOSE 3000
COPY ./default.conf /etc/nginx/conf.d/default.conf
COPY --from=builder usr/src/app/build  /usr/share/nginx/html
```

back-end

```
FROM amazoncorretto:17
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

3.3. NGINX

~/conf/nginx.conf

```
server {
        listen 80;
        client_max_body_size 20M;

        server_name laon.info;
        return 308 https://laon.info$request_uri;
}

server {
        listen 443 ssl;
        client_max_body_size 20M;

        server_name laon.info;

        # Certificate
        ssl_certificate /etc/letsencrypt/live/laon.info/fullchain.pem;

        # Private Key
        ssl_certificate_key /etc/letsencrypt/live/laon.info/privkey.pem;

        location /api {
                proxy_pass http://api:8080;
                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;
        }

        location / {
                proxy_pass http://web:3000;
                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;
        }

        location /images {
                alias /home/root/images;
        }

}
```

3.4. Jenkins

Command

```
# Backend Build and Push
cd back-end/lions

chmod +x ./gradlew
./gradlew clean build -x test
docker ps -f name=api-container -q | xargs --no-run-if-empty docker container stop
```

```
docker container ls -a -f name=api-container -q | xargs -r docker container rm

# Build image with environment variables and Dockerfile
docker build \
-t kimta2hwan/api .

#docker run -it -d --rm -p 8080:8080 --name=backend backend -h bserver
docker rmi -f $(docker images -f "dangling=true" -q) || true

# Frontend Build and Push
cd ../../front-end

# Build image with Dockerfile
docker build -t kimta2hwan/web .

cd /var/jenkins_home
docker-compose up -d
docker image prune -f
```

---

4. Build

4.1. Comman

```
docker-compose up -d
```