포팅 메뉴얼

1. 개발환경

1.1 Frontend

- Node.js 20.11.0
- Vue 3
- Flutter 3.19.5
- Dart 3.3.3

1.2 Backend

- Java 21
- SpringBoot 3.2.5
- Spring Data JPA
- Lombok

1.3 Database

- MySQL 8.0.34

1.4 UI/UX

- Figma

1.5 Server

- 배포서버
  - AWS EC2 1대
  - Ubuntu 22.04.1
  - Docker 26.1.2
  - Docker Compose 1.27.4

1.6 형상 / 이슈관리

- Gitlab
- Jira

2. 환경변수

2.1 Backend - Spring Boot

```
MYSQL_USERNAMEMYSQL_PASSWORD
JWT_SECRET_KEY
KAKAO_API_KEY
```

3. 배포 서버 세팅

3.1 Docker설치

```
# 1. Set up Docker's apt repository
 3-2. Docker Compose
설치
# Add Docker's official GPG key:
 sudo apt-get update
 sudo apt-get install ca-certificates curl
 sudo install -m 0755 -d /etc/apt/keyrings
 sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
 sudo chmod a+r /etc/apt/keyrings/docker.asc
 # Add the repository to Apt sources:
 echo \
 "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.dock
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
 sudo apt-get update
 # 2. Install the Docker packages.
 sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

3.2 Docker Compose설치

```
# Install the Compose plugin
 # 1. Update the package index, and install the latest version of Docker Compose
 sudo apt-get update
 sudo apt-get install docker-compose-plugin
```

3.3 컨테이너 구축

- docker-compose.yml

```
version: "3.0"
 services:
 nginx:
 ports:- "80:80"- "443:443"
 networks:- appnet
 image: nginx
 mysql:
 ports:- "3306:3306"
 volumes:- mysql_data:/var/lib/mysql
 networks:- appnet
 environment:
 MYSQL_ROOT_PASSWORD:
 MYSQL_DATABASE:
 MYSQL_USER:
 MYSQL_PASSWORD:
 command: --character-set-server=utf8mb4 --collation-server=utf8mb4_unicode_ci
 image: mysql:8.0.34
 jenkins:
 build: .
 ports:- "8080:8080"
 networks:- jenkinsnet
 volumes:- /var/run/docker.sock:/var/run/docker.sock
portainer:
ports:- "9443:9443"
volumes:- /var/run/docker.sock:/var/run/docker.sock- portainer_data:/data
image: portainer/portainer-ce:latest
volumes:
mysql_data: {}
portainer_data: {}
networks:
jenkinsnet: {}
appnet: {}
```

3.4 Nginx proxy 설정 + SSL 인증서 발급

```
# 1. 프록시 서버 컨테이너 접속
docker exec -it docker-compose-nginx-1 bash

#2. 도메인 구매, 등록

#3. certbot install
 sudo snap install certbot --classic

#4. SSL 발급 및 적용
cd /etc/nginx/conf.d
sudo certbot --nginx -d { 도메인 }

#5. 설정 파일 수정( 프록시 설정 )
vi /etc/nginx/conf.d/default.conf

 server {
    root /var/www/html;
    server_name 도메인명
    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/choonong.store/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/choonong.store/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
 ### admin
    location /admin {
        proxy_pass http://nginx-vue;
    }
 ### springboot
    location /api {
       proxy_pass http://docker-compose-spring-1:8081;
    }
### flutter
    location /api {
       proxy_pass http://nginx-flutter;
    }

 server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name loadlogix.store www.loadlogix.store;

    location /api/simulation {
       proxy_pass http://docker-compose-spring-1:8081;
    }

    if ($host = 도메인 명) {
       return 301 https://$host$request_uri;
    } # managed by Certbot
    if ($host = 도메인 명) {
       return 301 https://$host$request_uri;
    } # managed by Certbo
 }
```

4. CI/CD

- DooD(Docker out of Docker) 방식 채택
- GitLab WebHooks설정으로 자동 빌드, 배포 수행

4.1 Backend pipline

```
pipeline {
    agent any
    stages {
        stage('Clone Repository') {
            steps {
                script {
                    git branch: 'release', credentialsId: 'gitlabToken', url: 'https://lab.ssafy.com/s10-final/S10P31B308.git'
                }
            }
        }

        stage('Build') {
            steps {
                dir('backend') {
                    sh 'chmod +x gradlew'
                    sh './gradlew clean build -x test'
                }
            }
        }

        stage('Copy') {
            steps {
                dir('backend/build/libs') {
                    sh 'cp load-0.0.1-SNAPSHOT.jar ~/workspace/deploy/springboot/app.jar'
                }
            }
        }

        stage('Deploy') {
            steps {
                dir('../deploy/springboot') {
                    sh 'docker-compose down'
                    sh 'docker-compose build'
                    sh 'docker-compose up -d --build'
                }
            }
        }
    }
}
```

- docker-compose.yml

```
version: "3.0"
services:
  springboot:
    build: .
    networks:
      - docker-compose_appnet
    ports:
      - "8081:8081"
    environment:
      MYSQL_URL: jdbc:mysql://43.201.31.37:3306/load_db?serverTimezone=UTC&useUniCode=yes&characterEncoding=UTF-8
      MYSQL_USERNAME:
      MYSQL_PASSWORD: ""
      KAKAO_API_KEY:
      JWT_SECRET_KEY:
networks:
  docker-compose_appnet:
    external: true
```

- Dockerfile

```
FROM openjdk:21

WORKDIR /

COPY app.jar .

CMD ["java", "-jar", "/app.jar"]
```

- 스프링 서버 환경 변수

```
environment:
      MYSQL_URL:
      MYSQL_USERNAME:
      MYSQL_PASSWORD:
      KAKAO_API_KEY:
      JWT_SECRET_KEY:
```

4.2 Backend PipeLine (python)

```
pipeline {
    agent any
    stages {
        stage('Clone Repository') {
            steps {
                script {
                    git branch: 'release', credentialsId: 'gitlabToken', url: 'https://lab.ssafy.com/s10-final/S10P31B308.git'
                }
            }
        }

        stage('Clean'){
            steps{
                script{
                    try{
                        sh "docker stop loadAlgo"
                        sleep 1
                        sh "docker rm loadAlgo"
                    }catch(e){
                        sh 'exit 0'
                    }
                }
            }
        }

        stage('Copy') {
            steps {
                sh 'cp -r algorithm/load/* ~/workspace/deploy/python'
            }
        }

        stage('Build') {
            steps {
                dir('../deploy/python') {
                    sh 'pwd'
                    sh 'docker build -t loadalgo .'
                }
            }
        }

        stage('Deploy') {
            steps {
                dir('../deploy/python/loadAlgo') {
                    sh 'docker run -d --name="loadAlgo" -p 8083:8083 --network=docker-compose_appnet loadalgo:latest'
                }
            }
        }
    }

}
```

- Dockerfile

```
FROM python:3.9
WORKDIR /python
COPY ./requirements.txt /python/requirements.txt
RUN pip install --no-cache-dir --upgrade -r /python/requirements.txt
COPY ./py3dbp/main.py /python/py3dbp/main.py
COPY ./py3dbp/__init__.py /python/py3dbp/__init__.py
COPY ./py3dbp/auxiliary_methods.py /python/py3dbp/auxiliary_methods.py
COPY ./py3dbp/constants.py /python/py3dbp/constants.py
COPY ./loadAlgo.py /python/loadAlgo.py
ENV PYTHONPATH "$PYTHONPATH:/python"
CMD ["python3", "loadAlgo.py"]
```

4.3 Frontend PipeLine (vue)

```
pipeline {
    agent any
     tools {
        nodejs 'nodejs'
    }
    environment {
        VUE_APP_API_URL = ''
    }

    stages {
        stage('Clone Repository') {
            steps {
                script {
                    git branch: 'release', credentialsId: 'gitlabToken', url: 'https://lab.ssafy.com/s10-final/S10P31B308.git'
                }
            }
        }

        stage('Clean'){
            steps{
                script{
                    try{
                    sh "docker stop nginx-vue"
                    sleep 1
                    sh "docker rm nginx-vue"
                }catch(e){
                    sh 'exit 0'
                }
            }
        }
    }


    stage('Build') {
        steps {
            dir('./frontend_admin') {
                sh 'npm -v'
                sh 'npm install'
                sh 'npm run build'
                sh 'rm -r ~/workspace/deploy/vue/dist'
                sh 'cp -r ./dist ~/workspace/deploy/vue/dist'
            }
        }
    }

    stage('Copy') {
        steps {
            dir('./frontend_admin') {
                sh 'rm -r ~/workspace/deploy/vue/dist'
                sh 'cp -r ./dist ~/workspace/deploy/vue/dist'
            }
        }
    }

        stage('Deploy') {
            steps {
                dir('../deploy/vue') {
                    sh 'docker-compose up -d'
                    sh 'docker cp ./dist nginx-vue:/usr/share/nginx/html/admin'
                }
            }
        }
    }

}
```

- docker-compose.yml

```
version: "3.0"

services:
  nginx:
    container_name: nginx-vue
    ports:
      - "8090:8090"
    networks:
      - docker-compose_appnet
    image: nginx

networks:
  docker-compose_appnet:
    external: true
```

4.4 Frontend PipeLine (flutter)

```
pipeline {
    agent any
    environment {
        FLUTTER_HOME = '/usr/local/flutter'
        PATH = "${FLUTTER_HOME}/bin:${env.PATH}"
    }
    stages {
        stage('Clone Repository') {
            steps {
                script {
                    git branch: 'release', credentialsId: 'gitlabToken', url: 'https://lab.ssafy.com/s10-final/S10P31B308.git'
                }
            }
        }

        stage('Clean'){
            steps{
                script{
                    try{
                        sh "docker stop nginx-flutter"
                        sleep 1
                        sh "docker rm nginx-flutter"
                    }catch(e){
                        sh 'exit 0'
                    }
                }
            }
        }


        stage('Build') {
            steps {
                dir('./frontend/Load_Frontend') {
                    sh 'flutter build web'
                    sh 'rm -r ~/workspace/deploy/flutter/web'
                    sh 'cp -r ./build/web ~/workspace/deploy/flutter/web'
                }
            }
        }

        stage('copy') {
            steps {
                dir('./frontend/Load_Frontend') {
                    sh 'rm -r ~/workspace/deploy/flutter/web'
                    sh 'cp -r ./build/web ~/workspace/deploy/flutter/web'
                }
            }
        }


        stage('Deploy') {
            steps {
                dir('../deploy/flutter/web') {
                    sh 'docker-compose up -d'
                    sh 'docker cp . nginx-flutter:/usr/share/nginx/html/'
                }
            }
        }
    }

}
```

- docker-compose.yml

```
version: "3.0"

services:
  nginx:
    container_name: nginx-flutter
    ports:
      - "8091:8091"
    networks:
      - docker-compose_appnet
    image: nginx

networks:
  docker-compose_appnet:
    external: true
```

- Flutter 환경 변수

```
GOOGLE_MAPS_API_KEY=
BASE_URL=
```