포팅 매뉴얼

태그

1) 프로젝트 및 환경 정보

백엔드

- JDK 17 (HotSpot JVM)
- Spring Boot 3.3.5
- MySQL 8.0
- Redis 7.4.1

프론트

- Andriod
  - 32

Infra

- Docker
  - 24.0.7
- Docker Compose
  - 2.30.2

이미지 생성

- OpenAI API
- Skybox AI API
- Python 3.12
- FastAPI

공통 사항

- URL
  - Server
    - https://k11a604.p.ssafy.io
  - Client
    - https://drive.google.com/file/d/1gL74tvTDfhlvfF8H5q-r5zsSZZ4ZwIL1/view?usp=sharing
- DB(MySQL)
  - Driver
    - MySQL 8.0
  - Host
    - k11a604.p.ssafy.io:3306
  - User
    - root
  - password
    - a604
- DB(Redis)
  - Driver
    - Redis 7.4.1
  - Host
    - k11a604.p.ssafy.io:6379

포트 정보

Container Name	Port Number
jenkins	8082
springboot	8080
mysql	3306
redis	6379
fastapi	8000

2) 클론 후 빌드 및 실행 방법

백엔드 빌드 및 실행 방법

💡 **정보**

- 포트 번호 : 8080
- 주의할 점
  - 반드시, Docker 설치 필요

application.yml

프론트엔드 빌드 및 실행 방법

- apk 파일 설치 및 실행
  - https://drive.google.com/file/d/1gL74tvTDfhlvfF8H5q-r5zsSZZ4ZwIL1/view?usp=sharing

fast api 서버 빌드 및 실행 방법

💡 **포트 정보**

- 포트 번호 : 8000

```
source ./venv/Scripts/activate

pip install -r requirements.txt

uvicorn main:app --reload
```

3) 배포

Jenkins를 활용하여 진행 develop 브랜치에 변경 사항이 생긴다면 자동으로 배포를 수행한다.

실행 방법

1. develop 브런치 clone
2. 새로운 브랜치 생성
3. 소스 코드 수정 후 push
4. 생성 브랜치 → develop 브랜치로 MR 요청 후 merge
5. 자동으로 배포 실행

Docker-compose : 스프링백엔드 , MySQL , Redis , FastAPI

Nginx - 스프링백엔드

docker-compose.yml

```
version: '3.8'

services:
  604backend:
    build: ./Classik_Backend
    ports:
      - "8080:8080"
    depends_on:
      604mysql:
        condition: service_healthy
      604redis:
        condition: service_healthy
    command: >
      sh -c "until nc -z k11a604.p.ssafy.io 3306; do echo 'Waiting for MySQL...'; sleep 1; done;
      java -jar app.jar"
    environment:
      - DB_HOST=k11a604.p.ssafy.io
      - DB_PORT=3306
      - DB_USER=root
      - DB_PASSWORD=a604
      - spring.data.redis.host=k11a604.p.ssafy.io
      - spring.data.redis.port=6379
      - spring.data.redis.playback_prefix=classik:playbacks
    networks:
      - 604network

  604mysql:
    image: mysql:8.0
    environment:
      MYSQL_ROOT_PASSWORD: a604
      MYSQL_DATABASE: 604db
    ports:
      - "3306:3306"
    healthcheck:
      test: ["CMD", "mysqladmin", "ping", "-h", "localhost"]
      interval: 10s
      timeout: 5s
      retries: 5
    networks:
      - 604network
```

```
  604redis:
    image: redis
    ports:
      - "6379:6379"
    networks:
      - 604network
    healthcheck:
      test: ["CMD", "redis-cli", "ping"]
      interval: 5s
      retries: 10

  fastapi:
    build:
      context: ./Classik_AI/searchserver
      dockerfile: Dockerfile
    ports:
      - "8000:8000"
    depends_on:
      - backend
    networks:
      - 604network

networks:
  604network:
    driver: bridge

volumes:
  jenkins_home: {}
```

CI/CD 구축

jenkins 설정 파이프라인

```
pipeline {
agent any
environment {
DOCKER_COMPOSE_PATH = "/var/lib/jenkins/workspace/604ound/docker-compose.yml"
BACKEND_JAR_PATH = "build/libs/Classik_Backend-0.0.1-SNAPSHOT.jar"
}
stages {
stage('Clone Repository') {
steps {
// GitLab에서 소스 코드 가져오기
git branch: "develop",
url: "https://lab.ssafy.com/s11-final/S11P31A604.git",
credentialsId: '604ounddd'
}
}
stage('Build Spring Boot') {
steps {
dir('Classik_Backend') {
// Gradle로 빌드
sh './gradlew build'
}
}
}
stage('Test Spring Boot') {
steps {
dir('Classik_Backend') {
// Gradle로 테스트 실행
sh './gradlew test'
}
}
}
stage('Deploy with Docker Compose') {
steps {
sh '''
docker-compose -f $DOCKER_COMPOSE_PATH stop
docker-compose -f $DOCKER_COMPOSE_PATH up -d --build
'''
}
}
}
post {
always {
```

```
echo 'Pipeline execution completed.'
}
success {
echo 'Pipeline succeeded. Application is running successfully.'
}
failure {
echo 'Pipeline failed. Check the logs for errors.'
}
}
}
```