포팅 메뉴얼

0. 목차

1. 프로젝트 기술 스택 및 버전

프론트엔드

React

- Vite 5.2.0
- emotion 11.11.4
- material 5.15.15
- redux 2.2.3
- axios 1.6.8
- react-dom 18.3.1
- react-router-dom 6.23.0
- styled-components 6.1.8
- three 0.164.1
- vite-plugin-pwa 0.19.8

백엔드

Spring 서버

- Springboot 3.2.4

- Spring Data JPA
- Spring Data mongodb
- Spring Data redis
- Spring Security
- Spring Kafka 3.1.3
- JWT
- Java JDK 17
- QueryDSL 5.0.0
- AWS S3
- gson 2.10.1

Fast api 서버

- Python 3.11.2
- pip 22.3.1
- fastapi 0.110.2
- uvicorn 0.29.0
- confluent_kafka 2.3.0
- redis 5.0.4
- tensorflow 2.16.1
- tensorflow_hub 0.16.1
- opencv-python 4.9.0

AI

- Openpose
- Movenet thunder

인프라

CI/CD

- AWS EC2
- AWS S3
- jenkins 2.454
- Docker 26.1.0
- NginX 1.18.0 (Ubuntu)
- MatterMost Webhook
- GitLab Webhook
- MySQL 8.0.36
- Redis 7.2.4

기타

이슈 관리

- Jira

형상 관리

- Git, Gitlab

소통, 협업

- Notion
- Mattermost

개발 환경

- OS: Windows 10
- IDE: Intellij, VSCode
- EC2: Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1051-aws x86_64)
- Reverse Proxy: Nginx
- SSH: WSL , MobaXterm

- SSL: CertBot, Let’s Encrypt

2. 배포 포트

BackEnd

- spring boot application: `8080:8080`
- fast api application: `8000:8000`
- nginx(backend): `443:8080`

database

- MongoDB: `8082:27017`
- MySQL: `3306:3306`
- Redis: `6379:6379`
- kafka: `9092`
- kafka-ui: `10000:8080`
- jenkins: `8090:8080`

FrontEnd

- react: `5173:80`
- nginx(frontend): `443:5173`

3. 서버 설정

EC2

```
# 서버 시간 변경
sudo timedatectl set-timezone Asia/Seoul
# 미러 서버를 카카오
sudo sed -i 's/ap-northeast-2.ec2.archive.ubuntu.com/mirror.k
akao.com/g' /etc/apt/sources.list
# swap 영역 할당
sudo fallocate -l 4G /swapfile # 4GB로 할당
sudo chmod 600 /swapfile # 권한 변경
sudo mkswap /swapfile # swapfile 생성
sudo swapon /swapfile # swapfile 활성화
sudo echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fst
ab # 재부팅 시에도 유지되도록 설정
```

- Docker 관련

  ```
  # 필요한 패키지 설치
  sudo apt-get -y install apt-transport-https ca-certificate
  s curl gnupg-agent software-properties-common
  # - apt-transport-https: 패키지 관리자가 https를 통해 데이터 및
  패키지에 접근할 수 있도록 한다
  # ca-certificates: certificate authority에서 발행되는 디지털 서
  명. SSL 인증서의 PEM 파일이 포함되어 있어 SSL 기반 앱이 SSL 연결이
  되어있는지 확인할 수 있다
  # curl: 특정 웹 사이트에서 데이터를 다운로드 받을 때 사용하는 패키
  # gnupg-agent: OpenPGP 표준 규격의 데이터 통신을 암호화하고 서명할
  수 있는 패키지
  # software-properties-common: PPA를 추가하거나 제거할 때 사용한
  다.
  #     PPA: Personal Package Archive(개인 패키지 저장소)를 의미
  하며, 캐노니컬社의 우분투에서 기본적으로 제공하는 패키지 외의 사적으로
  만든 패키지를 의미한다

  # Docker에 대한 GPC Key 인증 진행
  curl -fsSL https://download.docker.com/linux/ubuntu/gpg |
  sudo apt-key add -
  ```

```
# Docker 레포지토리 등록
sudo add-apt-repository "deb [arch=amd64] https://downloa
d.docker.com/linux/ubuntu $(lsb_release -cs) stable"
# docker 패키지 설치
sudo apt-get -y update
sudo apt-get -y install docker-ce docker-ce-cli container
d.io
sudo usermod -aG docker ubuntu
# docker compose 설치
sudo curl -L "https://github.com/docker/compose/releases/d
ownload/v2.21.0/docker-compose-$(uname -s)-$(uname -m)" -o
/usr/local/bin/docker-compose
```

- jenkins Docker container 내 설치

```
# 이미지 가져오기
docker pull jenkins/jenkins:jdk17
# 컨테이너 생성 및 실행
# docker.sock을 이용해 DooD를 가능하게 한다.
docker run -d --restart always --env JENKINS_OPTS=--httpPo
rt=8080 -v /etc/localtime:/etc/localtime:ro -e TZ=Asia/Seo
ul -p 8080:8080 -v /jenkins:/var/jenkins_home -v /var/run/
docker.sock:/var/run/docker.sock -v /usr/local/bin/docker-
compose:/usr/local/bin/docker-compose --name jenkins -u ro
ot jenkins/jenkins:jdk17
# 이후 jenkins 내부에서도 docker, docker compose를 설치해준다.
```

- jenkins plugin

```
# ssh 커맨드 입력에 사용
SSH Agent

# docker 이미지 생성에 사용
Docker
Docker Commons
Docker Pipeline
Docker API

# 웹훅을 통해 브랜치 merge request 이벤트 발생시 Jenkins 자동 빌드
에 사용
Generic Webhook Trigger

# 타사 레포지토리 이용시 사용 (GitLab, Github 등)
GitLab
GitLab API
GitLab Authentication
GitHub Authentication

# Node.js 빌드시 사용
NodeJS
```

- nginx(호스트)

```
sudo apt-get -y install nginx
```

S3

- region: `아시아 태평양(서울) ap-northeast-2`
- bucket: `step-to-dance`
- 퍼블릭 액세스 차단: `비활성`
- 버킷 정책

```
{
    "Version": "2012-10-17",
    "Id": "Policy1715241589222",
    "Statement": [
        {
            "Sid": "Stmt1715241582129",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": [
                "arn:aws:s3:::step-to-dance",
                "arn:aws:s3:::step-to-dance/*"
            ]
        }
    ]
}
```

- CORS

```
[
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "GET",
            "HEAD"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": [
            "x-amz-server-side-encryption",
            "x-amz-request-id",
            "x-amz-id-2",
            "Cross-Origin-Resource-Policy"
        ],
        "MaxAgeSeconds": 3000
    }
]
```

4. 배포 관련 파일

- 모든 설정 파일들은 각 프로젝트(spring, fast api, react) 최상단에 위치합니다.

Spring

docker-compose.yml

```
version: "3"

services:
  mysql:
    container_name: mysql
    image: mysql:8.0
    ports:
      - ${MYSQL_BINDING_PORT}:3306
    volumes:
      - ${MYSQL_DATA_PATH}:/var/lib/mysql
    environment:
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_USERNAME: ${MYSQL_USERNAME}
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    restart: unless-stopped

  redis:
    container_name: redis
    image: redis:6.2.6-alpine
    command: redis-server --requirepass ${REDIS_PASSWORD} --p
ort 6379
    ports:
      - ${REDIS_BINDING_PORT}:6379
    restart: unless-stopped

  mongodb:
    container_name: mongodb
    image: mongo
    ports:
      - ${DANCE_MONGODB_BINDING_PORT}:27017
    volumes:
      - ${DANCE_MONGODB_DATA_PATH}:/data/db
    environment:
      MONGO_INITDB_DATABASE: dance
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: ${DANCE_MONGODB_PASSWORD}
      DEV_TRIP_MONGODB_BINDING_PORT: ${DANCE_MONGODB_BINDING_
PORT}
    restart: no

  spring:
    container_name: spring
    image: dance_spring
    ports:
      - ${SPRING_BINDING_PORT}:8080
    build:
      context: .
      dockerfile: Dockerfile
    depends_on:
      - mysql
      - redis
    volumes:
      - /data/spring/vod/:/data/vod/
    environment:
      MYSQL_BINDING_PORT: ${MYSQL_BINDING_PORT}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_USERNAME: ${MYSQL_USERNAME}
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      REDIS_BINDING_PORT: ${REDIS_BINDING_PORT}
      KAFKA1_BINDING_PORT: ${KAFKA1_BINDING_PORT}
      KAFKA2_BINDING_PORT: ${KAFKA2_BINDING_PORT}
      KAFKA3_BINDING_PORT: ${KAFKA3_BINDING_PORT}
      JWT_SECRET_KEY: ${JWT_SECRET_KEY}
      KAKAO_KEY: ${KAKAO_KEY}
      KAKAO_REDIRECT_URL: ${KAKAO_REDIRECT_URL}
      FFMPEG_LOCATION: ${FFMPEG_LOCATION}
      FFPROBE_LOCATION: ${FFPROBE_LOCATION}
      REDIS_PASSWORD: ${REDIS_PASSWORD}
      DANCE_MONGODB_PASSWORD: ${DANCE_MONGODB_PASSWORD}
      DANCE_MONGODB_BINDING_PORT: ${DANCE_MONGODB_BINDING_POR
T}
      S3_ACCESS_KEY: ${S3_ACCESS_KEY}
      S3_SECRET_KEY: ${S3_SECRET_KEY}
      S3_BUCKET_NAME: ${S3_BUCKET_NAME}
    restart: always

  kafka1:
    image: confluentinc/cp-kafka:7.6.0
    hostname: kafka1
    container_name: kafka1
    ports:
      - ${KAFKA1_BINDING_PORT}:9092
    environment:
      KAFKA1_BINDING_PORT: ${KAFKA1_BINDING_PORT}
      KAFKA_BROKER_ID: 1
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_PROCESS_ROLES: broker,controller
      KAFKA_CONTROLLER_LISTENER_NAMES: CONTROLLER
      KAFKA_LISTENERS: PLAINTEXT://0.0.0.0:9092,CONTROLLER://
0.0.0.0:9093
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: CONTROLLER:PLAINT
EXT,PLAINTEXT:PLAINTEXT
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka1:9092
      KAFKA_CONTROLLER_QUORUM_VOTERS: 1@kafka1:9093,2@kafka2:
9093,3@kafka3:9093
      KAFKA_AUTO_LEADER_REBALANCE_ENABLE: "true"
      KAFKA_DELETE_TOPIC_ENABLE: "true"
      KAFKA_GROUP_INITIAL_REBALANCE_DELAY_MS: 100
      CLUSTER_ID: kafka-docker-cluster-1
      ALLOW_PLAINTEXT_LISTENER: 'yes'
      KAFKA_AUTO_CREATE_TOPICS_ENABLE: 'true'
      KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_MIN_ISR: 1
      KAFKA_ALLOW_EVERYONE_IF_NO_ACL_FOUND: "true"
      KAFKA_MESSAGE_MAX_BYTES: 20971520
  kafka2:
    image: confluentinc/cp-kafka:7.6.0
    hostname: kafka2
    container_name: kafka2
    ports:
      - ${KAFKA2_BINDING_PORT}:9093
    environment:
      KAFKA2_BINDING_PORT: ${KAFKA2_BINDING_PORT}
      KAFKA_BROKER_ID: 2
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_PROCESS_ROLES: broker,controller
      KAFKA_CONTROLLER_LISTENER_NAMES: CONTROLLER
      KAFKA_LISTENERS: PLAINTEXT://0.0.0.0:9092,CONTROLLER://
0.0.0.0:9093
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: CONTROLLER:PLAINT
EXT,PLAINTEXT:PLAINTEXT
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka2:9092
      KAFKA_CONTROLLER_QUORUM_VOTERS: 1@kafka1:9093,2@kafka2:
9093,3@kafka3:9093
      KAFKA_AUTO_LEADER_REBALANCE_ENABLE: "true"
      KAFKA_DELETE_TOPIC_ENABLE: "true"
      KAFKA_GROUP_INITIAL_REBALANCE_DELAY_MS: 100
      CLUSTER_ID: kafka-docker-cluster-1
      ALLOW_PLAINTEXT_LISTENER: 'yes'
      KAFKA_AUTO_CREATE_TOPICS_ENABLE: 'true'
      KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_MIN_ISR: 1
      KAFKA_ALLOW_EVERYONE_IF_NO_ACL_FOUND: "true"
      KAFKA_MESSAGE_MAX_BYTES: 20971520
  kafka3:
    image: confluentinc/cp-kafka:7.6.0
    hostname: kafka3
    container_name: kafka3
    ports:
      - ${KAFKA3_BINDING_PORT}:9094
    environment:
      KAFKA3_BINDING_PORT: ${KAFKA3_BINDING_PORT}
      KAFKA_BROKER_ID: 3
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_PROCESS_ROLES: broker,controller
      KAFKA_CONTROLLER_LISTENER_NAMES: CONTROLLER
      KAFKA_LISTENERS: PLAINTEXT://0.0.0.0:9092,CONTROLLER://
0.0.0.0:9093
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: CONTROLLER:PLAINT
EXT,PLAINTEXT:PLAINTEXT
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka3:9092
      KAFKA_CONTROLLER_QUORUM_VOTERS: 1@kafka1:9093,2@kafka2:
9093,3@kafka3:9093
      KAFKA_AUTO_LEADER_REBALANCE_ENABLE: "true"
      KAFKA_DELETE_TOPIC_ENABLE: "true"
      KAFKA_GROUP_INITIAL_REBALANCE_DELAY_MS: 100
      CLUSTER_ID: kafka-docker-cluster-1
      ALLOW_PLAINTEXT_LISTENER: 'yes'
      KAFKA_AUTO_CREATE_TOPICS_ENABLE: 'true'
      KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_MIN_ISR: 1
      KAFKA_ALLOW_EVERYONE_IF_NO_ACL_FOUND: "true"
      KAFKA_MESSAGE_MAX_BYTES: 20971520
  kafka-ui:
    image: provectuslabs/kafka-ui:latest
    container_name: kafka-ui
    ports:
      - ${KAFKA_UI_BINDING_PORT}:8080
    depends_on:
      - kafka1
      - kafka2
      - kafka3
    environment:
      KAFKA_CLUSTERS_0_NAME: kafka-docker-cluster-1
      KAFKA_CLUSTERS_0_BOOTSTRAPSERVERS: kafka1:9092,kafka2:9
092,kafka3:9092
```

Dockerfile

```
FROM ubuntu:latest

# 필요한 패키지 설치
RUN apt-get update && apt-get install -y \
    curl \
    ffmpeg \
    openjdk-17-jdk && \
    rm -rf /var/lib/apt/lists/*

# Docker buildx 설치
RUN mkdir -p /usr/libexec/docker/cli-plugins && \
    curl -L https://github.com/docker/buildx/releases/downloa
d/v0.6.3/buildx-v0.6.3.linux-amd64 -o /usr/libexec/docker/cli
-plugins/docker-buildx && \
    chmod a+x /usr/libexec/docker/cli-plugins/docker-buildx

# 애플리케이션 JAR 파일 추가
ADD ./build/libs/steptodance-0.0.1-SNAPSHOT.jar /app.jar

# 애플리케이션 실행 명령 지정
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

build.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.2.4'
    id 'io.spring.dependency-management' version '1.1.4'
}

group = 'com.dance101'
version = '0.0.1-SNAPSHOT'

java {
    sourceCompatibility = '17'
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

dependencies {
    /* Spring */
    implementation 'org.springframework.boot:spring-boot-star
ter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-star
ter-security'
    implementation 'org.springframework.boot:spring-boot-star
ter-web'
    implementation 'org.springframework.boot:spring-boot-star
ter-validation'
    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'
    implementation 'org.springframework.boot:spring-boot-star
ter-webflux'
    /* ffmpeg */
    implementation 'net.bramp.ffmpeg:ffmpeg:0.8.0'
    /* javaCV */
    implementation 'in.hocg.boot:javacv-spring-boot-starter:
1.0.40'
    /* QueryDSL */
    implementation 'com.querydsl:querydsl-jpa:5.0.0:jakarta'
    annotationProcessor "com.querydsl:querydsl-apt:5.0.0:jaka
rta"
    annotationProcessor "jakarta.annotation:jakarta.annotatio
n-api"
    annotationProcessor "jakarta.persistence:jakarta.persiste
nce-api"
    /* DataBase */
    implementation 'org.springframework.boot:spring-boot-star
ter-data-redis'
    implementation 'org.springframework.boot:spring-boot-star
ter-data-mongodb'
    runtimeOnly 'com.mysql:mysql-connector-j'
    /* Kafka */
    implementation 'org.springframework.kafka:spring-kafka'
    /* JWT */
    implementation group: 'io.jsonwebtoken', name: 'jjwt-ap
i', version: '0.11.2'
    runtimeOnly group: 'io.jsonwebtoken', name: 'jjwt-impl',
version: '0.11.2'
    runtimeOnly group: 'io.jsonwebtoken', name: 'jjwt-jackso
n', version: '0.11.2'
    /* AWS */
    implementation "com.amazonaws:aws-java-sdk-s3:1.12.281"
    /* ETC */
    implementation 'com.google.code.gson:gson:2.10.1'
    implementation 'com.fasterxml.jackson.core:jackson-databi
nd:2.16.1'
```

```
    implementation 'com.fasterxml.jackson.core:jackson-core:
2.16.1'
    /* Test */
    testImplementation 'org.springframework.boot:spring-boot-
starter-test'
    testImplementation 'org.springframework.security:spring-s
ecurity-test'
}

tasks.named('test') {
    useJUnitPlatform()
}

/* QueryDSL Start */
clean {
    delete file('src/main/generated')
}
/* QueryDSL End */
```

.env

```
MYSQL_BINDING_PORT=3306
MYSQL_DATA_PATH=/data/mysql/steptodance/data
MYSQL_DATABASE=steptodance
MYSQL_USERNAME=root
MYSQL_ROOT_PASSWORD=a101
REDIS_BINDING_PORT=6379
JWT_SECRET_KEY=SecretKeyForOurTeamA101AndAllForThoseWhoUseOur
ServiceHopefullyThisSecretKeysGonnaWorkAndThisTeamIsGonnaRipT
heAcademyOFF
KAKAO_KEY={카카오 api 키를 적어주세요}
# 서버로그인
KAKAO_REDIRECT_URL=https://www.steptodance.site/auth/login
# 로컬로그인
# KAKAO_REDIRECT_URL=http://localhost:5173/auth/login
```

```
KAFKA1_BINDING_PORT=9092
KAFKA2_BINDING_PORT=9093
KAFKA3_BINDING_PORT=9094
KAFKA_UI_BINDING_PORT=10000
SPRING_BINDING_PORT=8080
FFMPEG_LOCATION=/usr/bin/ffmpeg
FFPROBE_LOCATION=/usr/bin/ffprobe
REDIS_PASSWORD=A101PASS
DANCE_MONGODB_BINDING_PORT=8082
DANCE_MONGODB_PASSWORD=a101
DANCE_MONGODB_DATA_PATH=/data/mongodb/steptodance/data
S3_ACCESS_KEY={aws s3 액세스 키}
S3_SECRET_KEY={aws s3 시크릿 키}
S3_BUCKET_NAME=step-to-dance
```

fast api

docker-compose.yml

```
version: "3"

services:
  web:
    build:
      context: .
      dockerfile: Dockerfile
    image: fast-api
    container_name: fast-api
    ports:
      - "8000:8000"
    environment:
      REDIS_HOST: ${REDIS_HOST}
      REDIS_PORT: ${REDIS_PORT}
      REDIS_PASSWORD: ${REDIS_PASSWORD}
    restart: always
    networks:
      - steptodance_default
networks:
  steptodance_default:
    external: true
```

Dockerfile

```
FROM python:3.11

# Install necessary system packages
RUN apt-get update \
    && apt-get install -y --no-install-recommends \
        libgl1-mesa-glx \
        mesa-utils \
    && rm -rf /var/lib/apt/lists/*

# Copy and install Python dependencies
COPY ./requirements.txt /fastApiProject/requirements.txt
RUN pip install --no-cache-dir --upgrade -r /fastApiProject/r
equirements.txt

# Copy the source code
COPY ./src /fastApiProject/src
COPY ./resources /fastApiProject/resources

# Set working directory
WORKDIR /fastApiProject/src

# Command to run the FastAPI application
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8
000"]
```

requirements.txt

```
fastapi
uvicorn
confluent_kafka
python-dotenv
redis
tensorflow
tensorflow_hub
opencv-python
```

.env

```
KAFKA1_BINDING_PORT=9092
KAFKA2_BINDING_PORT=9093
KAFKA3_BINDING_PORT=9094
KAFKA_UI_BINDING_PORT=8091

REDIS_PORT=6379
REDIS_HOST=k10a101.p.ssafy.io
```

React

docker-compose.yml

```
version: "3"

networks:
  steptodance_default:
    external: true

services:
  web:
    build:
      context: .
      dockerfile: Dockerfile
    image: react
    container_name: react
    ports:
      - "5173:80"
    restart: always
    networks:
      - steptodance_default
    environment:
      TZ: Asia/Seoul
```

Dockerfile

```
# 사용할 이미지 선택
FROM node:20-alpine as build

# 작업 디렉토리 설정
WORKDIR /app

# 컨테이너 내부로 package.json 파일들을 복사
COPY package*.json ./

# 명령어 실행 (의존성 설치)
RUN yarn install --network-timeout 1000000

COPY . .

#yarn build
RUN yarn build

# prod environment
FROM nginx:stable-alpine
```

```
# 이전 빌드 단계에서 빌드한 결과물을 /usr/share/ngnix/html으로 복사
COPY --from=build /app/dist /app/dist

# 기본 nginx 설정 파일을 삭제
RUN rm /etc/nginx/conf.d/default.conf

# custom 설정파일을 컨테이너 내부로 복사
COPY nginx.conf /etc/nginx/conf.d

# 연결할 포트번호
EXPOSE 80

# 앱 실행
CMD ["nginx","-g", "daemon off;"]
```

nginx.conf

```
server {
    listen 80;
    location / {
        root    /app/dist;
        index   index.html;
        try_files $uri $uri/ /index.html;
    }
}
```

.env

```
VITE_APP_KAKAO_KEY={카카오 api 키를 적어주세요}

VITE_APP_KAKAO_REDIRECT_URL=http://localhost:5173/auth/login
```