포팅 메뉴얼

1. 개발환경

1.1 Backend

- Java==21
- Swagger==2.0.2
- Spring Boot==3.3.3
- Gradle==8.8
- lombok==1.18.34
- spring-boot-starter-data-jpa==3.3.3
- spring-boot-starter-web==3.3.3
- spring-boot-starter-oauth2-client==3.3.3
- spring-boot-starter-security==3.3.3

- spring-boot-starter-data-redis==3.3.3
- redisson-spring-boot-starter==3.33.0
- jjwt-api==0.12.6
- jjwt-impl==0.12.6
- jjwt-jackson==0.12.6
- spring-clout-starter-aws==2.2.6

1.2 Frontend

- axios==1.7.7
- react==18.3.1
- react-dom==18.3.1
- react-router-dom==6.26.2
- typescript==5.5.3
- vite==5.4.1
- lucide-react==0.439.0
- react-beautiful-dnd==13.1.1
- @tanstack/react-query==5.56.2
- react-intersection-observer==9.13.1
- tailwind-merge==2.5.2
- tailwind-scrollbar-hide==1.1.7
- tailwind-animate==1.0.7

1.3 Database

- PostgreSQL==16.4
- Redis==7.4.0
- ElasticSearch==8.6.2

1.4 Infra

- docker==27.2.0
- docker-compose==2.29.2

- nginx==1.27.1
- Kibana==8.6.2
- AWS S3
- AWS CloudFront

1.5 Cooperation

- Git
- Gitlab
- Jira
- MattterMost
- Figma
- Notion

2. 환경 변수 설정

2.1 Frontend: .env

- 환경변수 설정 위치

```
S11P21A701
└── frontend
    └── melting
        └── .env
```

- .env

```
VITE_API_BASE_URL=https://j11a701.p.ssafy.io
VITE_REDIRECT_URL=https://j11a701.p.ssafy.io

VITE_API_MEMBERS_PATH=/api/v1/members
VITE_API_ALBUMS_PATH=/api/v1/albums
VITE_API_SONGS_PATH=/api/v1/songs
VITE_API_ORIGINAL_SONGS_PATH=/api/v1
VITE_API_HASHTAGS_PATH=/api/v1/hashtags
```

- 로컬호스트에서 https로 정상 가동을 위해 mkcert SSL 인증 필요

2.2 Backend: application.yml

- 환경변수 설정 위치

```
S11P21A701
└── backend
    └── src
        └── main
                  └── resources
                      ├── application.yml
                      ├── application-local.yml
                      ├── application-prod.yml
                      └── application-secret.yml
```

- application.yml

```
spring:
  application:
    name: melting

logging:
  config: classpath:logging/logback-spring.xml
```

- application-local.yml

```
spring:
  application:
    name: melting

  config:
    import: application-secret.yml

  datasource:
    driver-class-name: org.postgresql.Driver
    url: jdbc:postgresql://localhost:5432/melting
    username: postgres

  jpa:
    hibernate:
      ddl-auto: update
```

```yaml
      properties:
        hibernate:
          format_sql: true
      show-sql: true
      defer-datasource-initialization: true

    mvc:
      problem details:
        enabled: true

    data:
      redis:
        port: 6379
        redisson-prefix: redis://

      elasticsearch:
        repositories:
          enabled: true

    servlet:
      multipart:
        max-file-size: 20MB
        max-request-size: 20MB

    batch:
      job:
        enabled: false
      jdbc:
        initialize-schema: always

    threads:
      virtual:
        enabled: true

  logging:
    level:
      com:
        dayangsung:
          melting: debug
      web: debug
```

- application-prod.yml

```
spring:
  application:
    name: melting
  datasource:
    driver-class-name: org.postgresql.Driver
    url: ${DATASOURCE_URL}
    username: ${DATASOURCE_USERNAME}
    password: ${DATASOURCE_PASSWORD}
  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        format_sql: true
    show-sql: true
    defer-datasource-initialization: true
  security:
    oauth2:
      client:
        registration:
          kakao:
            client-name: kakao
            client-authentication-method: client_secret_post
            client-id: ${OAUTH2_KAKAO_CLIENT_ID}
            client-secret: ${OAUTH2_KAKAO_CLIENT_SECRET}
            redirect-uri: https://j11a701.p.ssafy.io/oauth2/log
in/code/kakao
            authorization-grant-type: authorization_code
            scope:
              - account_email
          google:
            client-name: google
            client-id: ${OAUTH2_GOOGLE_CLIENT_ID}
            client-secret: ${OAUTH2_GOOGLE_CLIENT_SECRET}
            redirect-uri: https://j11a701.p.ssafy.io/oauth2/log
in/code/google
            authorization-grant-type: authorization_code
            scope:
              - email
        provider:
          kakao:
            authorization-uri: https://kauth.kakao.com/oauth/au
```

```yaml
thorize
            token-uri: https://kauth.kakao.com/oauth/token
            user-info-uri: https://kapi.kakao.com/v2/user/me
            user-name-attribute: kakao_account
  mvc:
    problem details:
      enabled: true
  data:
    redis:
      host: redis
      port: 6379
      redisson-prefix: redis://
      password: ${REDIS_PASSWORD}
    elasticsearch:
      repositories:
        enabled: true
      username: ${ELASTICSEARCH_USERNAME}
      password: ${ELASTICSEARCH_PASSWORD}
      uris: ${ELASTICSEARCH_HOST}
  servlet:
    multipart:
      max-file-size: 20MB
      max-request-size: 20MB
  jwt:
    secret: ${JWT_SECRET}
  batch:
    job:
      enabled: false
    jdbc:
      initialize-schema: always
  threads:
    virtual:
      enabled: true

fastapi:
  baseurl: ${FASTAPI_BASEURL}

logging:
  level:
    com:
      dayangsung:
        melting: debug
    web: debug
```

```
cloud:
  aws:
    s3:
      bucket: ${CLOUD_AWS_S3_BUCKET}
    region:
      static: ap-northeast-2
      auto: false
    credentials:
      access key: ${CLOUD_AWS_CREDENTIALS_ACCESS_KEY}
      secret key: ${CLOUD_AWS_CREDENTIALS_SECRET_KEY}
    stack:
      auto: false

openai:
  api-key: ${OPENAI_API_KEY}

server:
  tomcat:
    connection-timeout: 30000

mattermost-logger:
  base-url: ${ERROR_WEBHOOK_URL}
```

- application-scret.yml

```
spring:
  security:
    oauth2:
      client:
        registration:
          kakao:
            client-name: kakao
            client-authentication-method: client_secret_post
            client-id: b87fb8ff7abf90f7dae1c4b2b54b2fd1
            client-secret: M0lyhUNbiDUOTA0dvxt0qE928f4Fm78P
            redirect-uri: http://localhost:8080/login/oauth2/co
de/kakao
            authorization-grant-type: authorization_code
            scope:
              - account_email
          google:
            client-name: google
            client-id: 343589914003-ls5qicmojabecs5slgpc2300eud
d68el.apps.googleusercontent.com
            client-secret: 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
            redirect-uri: http://localhost:8080/login/oauth2/co
de/google
            authorization-grant-type: authorization_code
            scope:
              - email
        provider:
          kakao:
            authorization-uri: https://kauth.kakao.com/oauth/au
thorize
            token-uri: https://kauth.kakao.com/oauth/token
            user-info-uri: https://kapi.kakao.com/v2/user/me
            user-name-attribute: kakao_account
  jwt:
    secret: d24084c2a22bc6f5c2d44700000044355fe9085070538c36c6a6
f6ea8831ab6f7\

  datasource:
    password: 1234

  profiles:
    active: local

  data:
    redis:
        host: localhost
        password: melting1234!

    elasticsearch:
      repositories:
        enabled: true
      uris: j11a701.p.ssafy.io:9200
      username: elastic
      password: melting1234!

cloud:
  aws:
    s3:
      bucket: melting-data-bucket
    region:
      static: ap-northeast-2
      auto: false
    credentials:
      access key: AKIAQZFG43ZCVMPQ4DGH
      secret key: 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
    stack:
      auto: false

fastapi:
  baseurl: http://70.12.246.171:8000

openai:
  api-key: sk-proj-bMl8I3J26kfeY77JR6O9AoLSLsl83UPUxEYIIF-93KKJ
65oLY9GWEuAc7INIfaOyU0z7BsmOR6T3BlbkFJdIfUQ3vgE3jGyVaLubuczCpqg
_fovYhbmiC8fcK57neJjEMOxu5Io7_0JR9yO6om-uuA_ZzgcA # Melting APP
KEY
  # api-key: sk-viY__ljuhNxevkfrMbGhQX4glQJjBGmmRbyUL9o6aiT3Blb
kFJnR-UJPkNCbANEDBWb3EQQSy390S57CN1KgJ_c2L4UA # 규영 계정 KEY
  # api-key: sk-cd9kIexTA_0rcYgpMPNTsfq-1uKbLYqHWB2oaSPuK2T3Blb
kFJve_kIgJA7N7-BflDye5RAQ0FE0wq1UZ0tsE3K8sagA # 연습용, 크레딧 더
충전되어 있습니다.

mattermost-logger:
  base-url: https://meeting.ssafy.com/hooks/hs1ta4i5njdf7dyk7wh
qnyeaow
```

3. 빌드 및 배포 문서

3.1 소프트웨어 설치

3.1.1 Docker 설치

```
# 1. 도커 apt 리포지토리 설정
# 도커 공식 GPG key 추가
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 /e
tc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

```
# Apt 소스에 도커 리포지토리 추가
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyri
ngs/docker.asc] https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

# 2. 도커 최신버전 설치
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-
buildx-plugin docker-compose-plugin
```

3.1.2 docker-compose 설치

```
# 1. docker-compose 설치
$ sudo curl -SL https://github.com/docker/compose/releases/downloa
d/v2.29.2/docker-compose-linux-x86_64 -o /usr/local/bin/docker-com
pose

# 2. 실행 권한 주기
$ sudo chmod +x /usr/local/bin/docker-compose
```

3.1.3 Java 설치

```
# 1. java 설치
$ sudo apt-get install openjdk-21-jdk
$ java -version
$ which java
$ readlink -f /usr/bin/java

# 2. 환경변수 추가
$ sudo vi /etc/profile

# 아래 내용 추가
export JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bin
export CLASSPATH=$JAVA_HOME/jre/lib:$JAVA_HOME/lib/tools.jar
```

```
# 3. 환경변수 적용
$ source /etc/profile
```

3.1.4 Node 설치

```
$ sudo apt-get update
$ curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash
-
```

3.2 Ngnix

3.2.1 nginx cerbot docker-compose.yml

```
version: "3"

services:
  proxy:
    image: "nginx:latest"
    container_name: "nginx"
    ports:
      - "80:80"
      - "443:443"
    restart: always
    volumes:
      - ./conf:/etc/nginx/conf.d
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
      - /home/ubuntu/melting/melting-fe/dist:/usr/share/nginx/html
    networks:
      - my_network

    command: '/bin/sh -c ''while :; do sleep 6h & wait $${!}; ngin
x -s reload; done & nginx -g "daemon off;"'''

  certbot:
    image: "certbot/certbot"
    restart: unless-stopped
    volumes:
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
    entrypoint: "/bin/sh -c 'trap exit TERM; while :; do certbot r
enew; sleep 12h & wait $${!}; done;'"
networks:
  my_network:
    driver: bridge
```

3.2.2 default.conf

```
server {
     client_max_body_size 0;
     listen 80;
     server_name j11a701.p.ssafy.io;

     location /.well-known/acme-challenge/ {
             root /var/www/certbot;
     }

     location / {
        proxy_set_header Host $host;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_fo
r;
        proxy_set_header X-Forwarded-Proto $scheme;
        return 301 https://$host$request_uri;
     }

}

server {
        client_max_body_size 0;
        listen 443 ssl;
        server_name j11a701.p.ssafy.io;

        ssl_certificate /etc/letsencrypt/live/j11a701.p.ssafy.io/f
ullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/j11a701.p.ssafy.
io/privkey.pem;
        include /etc/letsencrypt/options-ssl-nginx.conf;
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

        location / {
                root   /usr/share/nginx/html;
                index  index.html index.htm;
                try_files $uri /index.html;

        }
        location /.well-known/ {
             root /home/;
        }
        location /oauth/ {
                proxy_pass http://j11a701.p.ssafy.io:8080;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forw
arded_for;
                proxy_set_header X-Forwarded-Proto $scheme;

        }
        location /oauth2/ {
                proxy_pass http://j11a701.p.ssafy.io:8080;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forw
arded_for;
                proxy_set_header X-Forwarded-Proto $scheme;

        }
        location /api/ {
                proxy_pass http://j11a701.p.ssafy.io:8080;
                proxy_set_header Host $host;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forw
arded_for;
                proxy_set_header X-Forwarded-Proto $scheme;
        }

        location /fastapi/ {
                proxy_pass http://j11a701.p.ssafy.io:8000;  # Fast
API가 실행 중인 Docker 컨테이너 포트
```

```
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forw
arded_for;
                proxy_set_header X-Forwarded-Proto $scheme;
        }
}
```

3.2.3 init-letsencrypt.sh 실행

```
#!/bin/bash

if ! [ -x "$(command -v docker-compose)" ]; then
  echo 'Error: docker-compose is not installed.' >&2
  exit 1
fi

domains=(j11a701.p.ssafy.io)
rsa_key_size=4096
data_path="./data/certbot"
email="thswlals0219@gmail.com" # Adding a valid address is strongl
y recommended
staging=0 # Set to 1 if you're testing your setup to avoid hitting
request limits

if [ -d "$data_path" ]; then
  read -p "Existing data found for $domains. Continue and replace
existing certificate? (y/N) " decision
  if [ "$decision" != "Y" ] && [ "$decision" != "y" ]; then
    exit
  fi
fi


if [ ! -e "$data_path/conf/options-ssl-nginx.conf" ] || [ ! -e "$d
ata_path/conf/ssl-dhparams.pem" ]; then
  echo "### Downloading recommended TLS parameters ..."
  mkdir -p "$data_path/conf"
  curl -s https://raw.githubusercontent.com/certbot/certbot/maste
r/certbot-nginx/certbot_nginx/_internal/tls_configs/options-ssl-ng
inx.conf > "$data_path/conf/options-ssl-nginx.conf"
```

```
  curl -s https://raw.githubusercontent.com/certbot/certbot/maste
r/certbot/certbot/ssl-dhparams.pem > "$data_path/conf/ssl-dhparam
s.pem"
  echo
fi

echo "### Creating dummy certificate for $domains ..."
path="/etc/letsencrypt/live/$domains"
mkdir -p "$data_path/conf/live/$domains"
docker-compose run --rm --entrypoint "\
  openssl req -x509 -nodes -newkey rsa:$rsa_key_size -days 1\
    -keyout '$path/privkey.pem' \
    -out '$path/fullchain.pem' \
    -subj '/CN=localhost'" certbot
echo


echo "### Starting nginx ..."
docker-compose up --force-recreate -d nginx
echo

echo "### Deleting dummy certificate for $domains ..."
docker-compose run --rm --entrypoint "\
  rm -Rf /etc/letsencrypt/live/$domains && \
  rm -Rf /etc/letsencrypt/archive/$domains && \
  rm -Rf /etc/letsencrypt/renewal/$domains.conf" certbot
echo


echo "### Requesting Let's Encrypt certificate for $domains ..."
#Join $domains to -d args
domain_args=""
for domain in "${domains[@]}"; do
  domain_args="$domain_args -d $domain"
done

# Select appropriate email arg
case "$email" in
  "")./ email_arg="--register-unsafely-without-email" ;;
  *) email_arg="--email $email" ;;
esac

# Enable staging mode if needed
```

```
if [ $staging != "0" ]; then staging_arg="--staging"; fi

docker-compose run --rm --entrypoint "\
  certbot certonly --webroot -w /var/www/certbot \
    $staging_arg \
    $email_arg \
    $domain_args \
    --rsa-key-size $rsa_key_size \
    --agree-tos \
    --force-renewal" certbot
echo

echo "### Reloading nginx ..."
docker-compose exec nginx nginx -s reload
```

```
$ chmod +x init-letsencrypt.sh
$ sudo ./init-letsencrypt.sh
```

3.3 Spring

3.3.1 Dockerfile

```
FROM openjdk:21-jdk

# JAR 파일을 컨테이너에 복사
COPY melting-0.0.1-SNAPSHOT.jar app.jar

# 환경 변수 설정 (Spring Profile 설정)
ENV SPRING_PROFILES_ACTIVE=prod

# 추가로 필요한 환경 변수 설정 가능
ENV DATASOURCE_URL=jdbc:postgresql://172.26.12.66:5432/melting
ENV DATASOURCE_USERNAME=dayangsung
ENV DATASOURCE_PASSWORD=melting1234!
ENV OAUTH2_KAKAO_CLIENT_ID=b87fb8ff7abf90f7dae1c4b2b54b2fd1
ENV OAUTH2_KAKAO_CLIENT_SECRET=M0lyhUNbiDUOTA0dvxt0qE928f4Fm78P
ENV OAUTH2_GOOGLE_CLIENT_ID=343589914003-ls5qicmojabecs5slgpc2300eud
ENV OAUTH2_GOOGLE_CLIENT_SECRET=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
ENV JWT_SECRET=d24084c2a22bc6f5c2d44700000044355fe9085070538c36c6a6f6
ENV CLOUD_AWS_S3_BUCKET=melting-data-bucket
ENV CLOUD_AWS_CREDENTIALS_ACCESS_KEY=AKIAQZFG43ZCVMPQ4DGH
ENV CLOUD_AWS_CREDENTIALS_SECRET_KEY=LQXfJHEwGjwSmdTwcOCkSRBqBQi4mFa
ENV REDIS_PASSWORD=melting1234!
ENV FASTAPI_BASEURL=http://70.12.246.171:8000
ENV OPENAI_API_KEY=sk-proj-bMl8I3J26kfeY77JR6O9AoLSLsl83UPUxEYIIF-93
ENV ELASTICSEARCH_USERNAME=elastic
ENV ELASTICSEARCH_PASSWORD=melting1234!
ENV ELASTICSEARCH_HOST=172.26.12.66
ENV ERROR_WEBHOOK_URL=https://meeting.ssafy.com/hooks/f688z3u6sib4zm


ENTRYPOINT ["java", "-jar", "app.jar"]
```

3.3.2 spring.sh 스크립트

```
#!/bin/bash

IMAGE_NAME="melting-spring"
IMAGE_TAG="latest"
CONTAINER_NAME="melting-spring-container"
LOG_DIR=~/melting_log
WEBHOOK_URL="https://meeting.ssafy.com/hooks/145xrm8t33f1by3az1p3mfp
now_port=8080

TIMEOUT=20
SECONDS=0

# Docker 이미지 빌드
docker build -t ${IMAGE_NAME}:${IMAGE_TAG} ~/melting/spring-docker

if [ $? -eq 0 ]; then
    curl -i -X POST -H 'Content-Type: application/json' -d '{"text":
else
    curl -i -X POST -H 'Content-Type: application/json' -d '{"text":
    exit 1
fi

# 기존에 실행 중인 컨테이너가 있으면 중지 및 제거
existing_container=$(docker ps -aq -f name=${CONTAINER_NAME})

if [ -n "$existing_container" ]; then
```

```
    echo "Stopping and removing existing container..."
    docker stop ${CONTAINER_NAME} || true  # 이미 중지된 경우 무시
    docker rm -f ${CONTAINER_NAME}  # 강제 삭제
fi

# 도커 컨테이너 실행
docker run -d --name ${CONTAINER_NAME} --network melting-network -p 

# 사용하지 않는 이미지를 제거
dangling_images=$(docker images -f "dangling=true" -q)
if [ -n "$dangling_images" ]; then
    docker rmi $dangling_images
fi

sleep 5

# webhook
while true; do
    response=$(curl -s -o /dev/null -w "%{http_code}" http://172.26.

    if [ "$response" -eq 200 ]; then
        curl -i -X POST -H 'Content-Type: application/json' -d '{"te
        break
    fi

    # 시간 초과 확인
    if [ $SECONDS -ge $TIMEOUT ]; then
        curl -i -X POST -H 'Content-Type: application/json' -d '{"te
        exit 1
    fi

    sleep 1  # 1초 대기 후 다시 확인
    echo "다시 확인중"
done
```

```
# 실행
chmod +x ./spring.sh
./spring.sh
```

3.4 React

3.4.1 react.sh

```
#!/bin/bash

WEBHOOK_URL="https://meeting.ssafy.com/hooks/145xrm8t33f1by3az1p3m
fpm3w"
BUILD_FILE=~/melting/melting-fe/dist/index.html
TIMEOUT=60

if [ -f "$BUILD_FILE" ] && [ $(($(date +%s) - $(stat -c %Y "$BUILD
_FILE"))) -lt $TIMEOUT ]; then
    curl -i -X POST -H 'Content-Type: application/json' \
    -d '{"text": "**[FE]** React 빌드 완료"}' $WEBHOOK_URL
else
    curl -i -X POST -H 'Content-Type: application/json' \
    -d '{"text": "**[FE]** React 빌드 실패: 빌드 결과를 찾을 수 없거나 최
신 빌드가 아닙니다."}' $WEBHOOK_URL
    exit 1
fi
```

3.5 Fast API

> rvc_ai_api 폴더 바로 아래

3.5.1 Dockerfile

```
# syntax=docker/dockerfile:1

FROM python:3.10-bullseye

EXPOSE 7865

WORKDIR /app

COPY . .

RUN apt-get update && \
    apt-get install -y ffmpeg && \
    rm -rf /var/lib/apt/lists/*

RUN pip3 install -r requirements.txt

VOLUME [ "/app/weights", "/app/input", "/app/logs", "/app/TEMP"]

CMD ["python3", "run_uvicorn.py"]
```

3.5.2 docker-compose.yml

```
version: "3.8"
services:
  rvc:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: rvc-api
    volumes:
      - ./weights:/app/weights
      - ./input:/app/input
      - ./TEMP:/app/TEMP
      - ./logs:/app/logs
      # - ./dataset:/app/dataset # you can use this folder in orde
r to provide your dataset for model training
    ports:
      - 8000:8000
    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              count: 1
              capabilities: [gpu]
```

4. Database

4.1 Postgresql

4.1.1 postgres 도커 이미지 받기

```
docker pull postgres:latest
```

4.1.2 postgres 실행

```
$ docker run -d -p 5432:5432 --name postgres \
-e POSTGRES_PASSWORD=melting1234! \
-e TZ=Asia/Seoul \
-v /home/ubuntu/melting/pgdata:/var/lib/postgresql/data \
-d postgres
```

```
$ docker run -d -p 5432:5432 --name postgres -e POSTGRES_PASSWORD=
melting1234! -e TZ=Asia/Seoul -v /home/ubuntu/melting/pgdata:/var/
lib/postgresql/data -d postgres
```

4.1.3 PostgreSQL 데이터베이스 및 유저 생성

```
docker exec -it [CONTAINER ID] bash #  postgres  컨테이너 접속
psql -U postgres # postgresql 접속
create database melting
CREATE USER dayangsung WITH PASSWORD 'melting1234!' SUPERUSER;
GRANT ALL PRIVILEGES ON DATABASE melting TO dayangsung;
```

4.2 Redis

4.2.1 Redis 설치 및 실행

```
$ docker pull redis:latest
$ docker run --name redis --restart=always --network melting-netwo
rk \
-p 6379:6379 -v /home/ubuntu/melting/redis/redis.conf:/usr/local/e
tc/redis/redis.conf \
-d redis redis-server /usr/local/etc/redis/redis.conf
```

4.3 Elasticsearch

4.3.1 Elasticsearch & Kibana docker-compose.yml

```
# Dockfile
FROM docker.elastic.co/elasticsearch/elasticsearch:8.6.2
RUN~ elasticsearch-plugin install analysis-nori
RUN bin/elasticsearch-plugin install https://github.com/netcrazy/e
lasticsearch-jaso-analyzer/releases/download/v8.6.2/jaso-analyzer-
plugin-8.6.2-plugin.zip
```

```
services:
  elasticsearch:
    build:
        context: .
        dockerfile: Dockerfile
    container_name: elasticsearch
    environment:
      - discovery.type=single-node
      - xpack.security.enabled=false
      - xpack.security.transport.ssl.enabled=false
      - "ES_JAVA_OPTS=-Xms1g -Xmx1g"
    ulimits:
      memlock:
        soft: -1
        hard: -1
    volumes:
      - /home/ubuntu/melting/elk/elasticsearch:/usr/share/elastics
earch/data
      - /home/ubuntu/melting/elk/config/elasticsearch.yml:/usr/sha
re/elasticsearch/config/elasticsearch.yml
    ports:
      - "9200:9200"
    networks:
      - melting-network

  kibana:
    image: docker.elastic.co/kibana/kibana:8.6.2
    container_name: kibana
    ports:
      - "5601:5601"
    environment:
      - ELASTICSEARCH_HOSTS=http://elasticsearch:9200
      - ELASTICSEARCH_USERNAME=kibana_system
      - ELASTICSEARCH_PASSWORD=melting1234!
    depends_on:
      - elasticsearch
    networks:
      - melting-network

networks:
  melting-network:
    external: true
```

```
# elasticsearch.yml
network.host: 0.0.0.0
```

4.3.2 비밀번호 설정

```
$ docker exec -it elasticsearch sh
$ bin/elasticsearch-setup-passwords interactive
# 비밀번호 입력
```

5. Gitlab CI/CD

5.1 gitlab-runner

5.1.1 gitlab-runner 설치

```
# Download the binary for your system
$ sudo curl -L --output /usr/local/bin/gitlab-runner https://gitla
b-runner-downloads.s3.amazonaws.com/latest/binaries/gitlab-runner-
linux-amd64

# Give it permission to execute
$ sudo chmod +x /usr/local/bin/gitlab-runner

$ gitlab-runner --version
$ gitlab-runner register  --url https://lab.ssafy.com  --token glr
t-6yrBxyLiinzxwEQycne2

$ sudo gitlab-runner run
```

5.1.2 gitlab-runner 등록

```

gitlab-runner register  --url https://lab.ssafy.com  --token glrt-
KjuPQndZH_BRscF4LgvA

sudo gitlab-runner run # 포그라운드 실행
sudo gitlab-runner start # 백그라운드 실행

# 백그라운드 실행
sudo systemctl start gitlab-runner
sudo systemctl enable gitlab-runner
sudo systemctl status gitlab-runner
sudo systemctl stop gitlab-runner
```

5.1.3 .gitlab-runner/config.toml

```
concurrent = 1
check_interval = 0
shutdown_timeout = 0

[session_server]
  session_timeout = 1800
  listen_address = "0.0.0.0:8093"   # 서버가 리슨할 IP 주소와 포트
  advertise_address = "43.203.201.104:8093"   # 외부에서 접근할 수 있
는 IP 주소와 포트

[[runners]]
  name = "ip-172-26-12-66"
  url = "https://lab.ssafy.com"
  id = 785
  token = "glrt-MPK8rw9FW1NbR-xuSRC4"
  token_obtained_at = 2024-09-19T03:41:34Z
  token_expires_at = 0001-01-01T00:00:00Z
  executor = "shell"
  [runners.custom_build_dir]
  [runners.cache]
    MaxUploadedArchiveSize = 0
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]
```

5.2 gitlab-cli

5.2.1 .gitlab-cli.yml

```
stages:
  - build

build_BE:
  stage: build
  image: gradle:8.8-jdk21
  before_script:
    - echo "[INFO] YML Settings"
    - chmod +x backend/gradlew
  script:
    - cd backend
    - ./gradlew clean build -x test -Dspring.profiles.active=prod
    - cp build/libs/melting-0.0.1-SNAPSHOT.jar ~/melting/spring-dock
    - sh ~/melting/spring-docker/spring.sh
  rules:
    - if: '$CI_COMMIT_BRANCH == "dev-be"'
  tags:
    - prod

build_FE:
  stage: build
  image: node
  before_script:
    - echo "[INFO] YML Settings"
    - cd frontend/melting
    - printenv | grep 'VITE_' > .env
  script:
    - yarn install
    - yarn build
    - sudo cp -R dist ~/melting/melting-fe
    - sh ~/melting/melting-fe/react.sh
  rules:
    - if: '$CI_COMMIT_BRANCH == "dev-fe"'
  tags:
    - prod
```