포팅 메뉴얼

목차

사용 프로그램 버전

- **Java**: `zulu-17`
- **Spring Boot**: `3.1.3`
- **gradle**: `8.2.1`
- **MySQL**: `8.1`
- **Redis**: `7.0.12`
- **Solidity**: `0.5.17`
- **node**: `v18.16.1`
- **npm**: `9.5.1`
- **React**: `18.2.0`

시스템 구성

Front server

- **React**: `5173:5173`

API server

- **Spring Boot 1**: `8080:8080`
- **Spring Boot 2**: `8081:8081`

Jenkins

- `8090:8090`
- `50000:50000`

Database

- **MySQL**: `3306:3306`
- **Redis**: `6379:6379`

외부 프로그램

- **openvidu**
- **Solidity**

환경 파일 세팅

백엔드

application.yml

```
spring:
  datasource:
    hikari:
      jdbc-url: ${DB_URL}
      username: ${DB_USERNAME}
      password: ${DB_PASSWORD}
      driver-class-name: com.mysql.cj.jdbc.Driver

  jpa:
    hibernate:
      ddl-auto: validate
    show-sql: true
    open-in-view: false
    properties:
      hibernate:
        format_sql: true
        dialect: org.hibernate.dialect.MySQL57Dialect

  sql:
    init:
      mode: always

  h2:
    console:
      enabled: true

  security:
    oauth2:
      client:
        registration:
          kakao:
            client-id: ${KAKAO_CLIENT_ID}
            client-secret: ${KAKAO_CLIENT_SECRET}
            redirect-uri: "{baseUrl}/{action}/oauth2/code/{registrationId}"
            authorization-grant-type: authorization_code
            client-authentication-method: client_secret_post
            client-name: Kakao
          naver:
            client-id: ${NAVER_CLIENT_ID}
            client-secret: ${NAVER_CLIENT_SECRET}
            redirect-uri: "{baseUrl}/{action}/oauth2/code/{registrationId}"
            authorization-grant-type: authorization_code
            scope: name,email,profile_image
            client-name: Naver
          google:
            client-id: ${GOOGLE_CLIENT_ID}
            client-secret: ${GOOGLE_CLIENT_SECRET}
            redirect-uri: "{baseUrl}/{action}/oauth2/code/{registrationId}"
            scope: profile,email
        provider:
          kakao:
            authorization-uri: https://kauth.kakao.com/oauth/authorize
            token-uri: https://kauth.kakao.com/oauth/token
            user-info-uri: https://kapi.kakao.com/v2/user/me
            user-name-attribute: id
          naver:
            authorization-uri: https://nid.naver.com/oauth2.0/authorize
            token-uri: https://nid.naver.com/oauth2.0/token
            user-info-uri: https://openapi.naver.com/v1/nid/me
            user-name-attribute: response

  cache:
    type: redis
    redis:
      host: ${REDIS_HOST}
      port: ${REDIS_PORT}

openvidu:
  url: ${OPENVIDU_URL}
  secret: ${OPENVIDU_SECRET}

client:
  host: ${CLIENT_HOST}
  url: ${CLIENT_URL}
  endpoint: ${CLIENT_ENDPOINT}

jwt:
  access-header: Authorization
  refresh-header: Refresh
  secret: ${JWT_SECRET}
```

docker-compose.yml

```
version: "3.7"

services:
  redis:
    image: redis
    container_name: redis
    hostname: redis
    ports:
      - "6379:6379"
  springboot:
    container_name: blooming-${IDLE_PROFILE}
    image: blooming-image-${IDLE_PROFILE}
    build:
      context: .
      dockerfile: ./spring-dockerfile/Dockerfile
    ports:
            - "${IDLE_PORT}:8080"
    environment:
      Profile: ${IDLE_PROFILE}
      OPENVIDU_URL: https://j9a105.p.ssafy.io:4443
      OPENVIDU_SECRET: ahdkahdkEmldnjEmldnj
      DB_URL: jdbc:mysql://j9a105.p.ssafy.io:3306/blooming_db?serverTimezone=UTC&useUnicode=true&characterEncoding=utf8
      DB_USERNAME: blooming_manager
      DB_PASSWORD: fivengers
      KAKAO_CLIENT_ID: f508bf6ede31a21a675b681c026f47d8
      KAKAO_CLIENT_SECRET: qrULgwSfjHJQyap2giDufmRuzdomuVqQ
      NAVER_CLIENT_ID: en96EgJ33d4S53ymNpuf
      NAVER_CLIENT_SECRET: cOPyRDyiPT
      GOOGLE_CLIENT_ID: 439808219871-4csfkl9jjqdvf5mnsjinboucbs8t2q3k.apps.googleusercontent.com
      GOOGLE_CLIENT_SECRET: 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
      CLIENT_HOST: localhost
      CLIENT_URL: http://localhost:5173
            # CLIENT_URL: https://j9a105.p.ssafy.io
      REDIS_HOST: redis
      REDIS_PORT: 6379
      CLIENT_ENDPOINT: /login-success
      JWT_SECRET: bloomingbloomingbloomingbloomingbloomingbloomingbloomingbloomingbloomingbloomingbloomingbloomingbloomingbloomingbloom
```

Dockerfile

```
FROM openjdk:17-jdk
ARG JAR_FILE=spring-dockerfile/build/libs/Blooming-0.0.1-SNAPSHOT.jar
COPY ${JAR_FILE} Blooming-0.0.1-SNAPSHOT.jar

COPY ./deploy.sh ./deploy.sh

RUN chmod +x ./deploy.sh
RUN chmod +x Blooming-0.0.1-SNAPSHOT.jar
ENTRYPOINT ["java","-jar", "Blooming-0.0.1-SNAPSHOT.jar", "--spring.profiles.active=${Profile}"]
#CMD ["./deploy.sh"]

EXPOSE 8081
```

프론트엔드

build.sh

```
#!/bin/bash
cd client/ || exit 1
# Docker Compose를 사용하여 컨테이너 중지 및 삭제
docker-compose down

# Docker 이미지 삭제 (필요에 따라)
docker rmi client_front:latest

# Docker Compose를 사용하여 컨테이너 실행
docker-compose up -d
```

docker-compose.yml

```
version: "3.1"
services:
  front:
    container_name: front
    build:
      context: ~/front/html/client
      dockerfile: Dockerfile
    ports:
      - 5173:5173
        #environment:
        #- VITE_BUCKET_REGION='ap-northeast-2'
        #- VITE_BUCKET_NAME='blooming-image'
        #- VITE_IDENTITY_POOLID='ap-northeast-2:01b0b700-68e1-41d4-bc17-1318e1b139de'      - VITE_APP_CLIENT="http://j9a105.p.ssafy.io'
        #- VITE_APP_SERVER="http://j9a105.p.ssafy.io:8080"
```

Dockerfile

```
FROM node:18 as builder

COPY . .
RUN yarn install

EXPOSE 5173
USER root
RUN yarn build

RUN yarn global add serve

CMD ["serve", "-s", "dist", "-l", "5173"]
```

무중단 배포

deploy.sh

```
#!/bin/bash
echo "> 현재 구동중인 profile 확인"
CURRENT_PROFILE=$(curl -s http://127.0.0.1/profile)
echo "> $CURRENT_PROFILE"

if [ $CURRENT_PROFILE == was1 ]
then
  IDLE_PROFILE=was2
  IDLE_PORT=8082
elif [ $CURRENT_PROFILE == was2 ]
then
  IDLE_PROFILE=was1
  IDLE_PORT=8081
else
  echo "> 일치하는 Profile이 없습니다. Profile: $CURRENT_PROFILE"
  echo "> was1을 할당합니다. IDLE_PROFILE: was1"
  IDLE_PROFILE=was1
  IDLE_PORT=8081
fi

echo "> $IDLE_PROFILE 배포"
REPOSITORY=/home/ubuntu/
```

```
cd $REPOSITORY
docker stop blooming-${IDLE_PROFILE}
docker rm blooming-${IDLE_PROFILE}
docker rmi blooming-image-${IDLE_PROFILE}

source ~/.bashrc

export IDLE_PROFILE
export IDLE_PORT

docker-compose --env-file ./.env.${IDLE_PROFILE} up -d

echo "> $IDLE_PROFILE 10초 후 Health check 시작"
echo "> curl -s http://localhost:$IDLE_PORT/actuator/health "
sleep 10

for retry_count in {1..10}
do
  response=$(curl -s http://localhost:$IDLE_PORT/actuator/health)
  up_count=$(echo $response | grep 'UP' | wc -l)

  if [ $up_count -ge 1 ]
  then
    echo "> Health check 성공"
    break
  else
    echo "> Health check의 응답을 알 수 없거나 혹은 status가 UP이 아닙니다."
    echo "> Health check: ${response}"
  fi

  if [ $retry_count -eq 10 ]
  then
    echo "> Health check 실패. "
    echo "> Nginx에 연결하지 않고 배포를 종료합니다."
    exit 1
  fi

  echo "> Health check 연결 실패. 재시도..."
  sleep 10
done

echo "> 스위칭을 시도합니다..."
sleep 10
```

switch.sh

```
#!/bin/bash
echo "> 현재 구동중인 Port 확인"
CURRENT_PROFILE=$(curl -s http://localhost/profile)

if [ $CURRENT_PROFILE == set1 ]
then
  IDLE_PORT=8082
elif [ $CURRENT_PROFILE == set2 ]
then
  IDLE_PORT=8081
else
  echo "> 일치하는 Profile이 없습니다. Profile:$CURRENT_PROFILE"
  echo "> 8081을 할당합니다."
  IDLE_PORT=8081
fi

PROXY_PORT=$(curl -s http://localhost/profile)
echo "> 현재 구동중인 Port: $PROXY_PORT"

echo "> 전환할 Port : $IDLE_PORT"
echo "> Port 전환"
echo "set \$service_url http://127.0.0.1:${IDLE_PORT};" | sudo tee /etc/nginx/conf.d/service-url.inc

echo "> Nginx Reload"
sudo service nginx reload
```

웹 서버 (NGINX)

```
server {
        root /var/www/html/client/dist;

        index index.html index.htm;
        server_name j9a105.p.ssafy.io; # managed by Certbot

        include /etc/nginx/conf.d/service-url.inc;

        location / {
                proxy_pass http://localhost:5173;
                proxy_redirect off;
                charset utf-8;

                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
        }

        location ^~ /html {
                root /home/ubuntu/spring-dockerfile/src/docs/asciidoc;
                index index.html;
        }

        location ~ ^/ws/blooming(/.*)?$ {
                proxy_pass http://j9a105.p.ssafy.io:8081/ws/blooming$1;
                proxy_redirect off;
                charset utf-8;

                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;

        }

        location ~ /(api|oauth2|login)/ {
                proxy_pass http://j9a105.p.ssafy.io:8081;
                proxy_redirect off;
                charset utf-8;

                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;
                proxy_set_header X-NginX-Proxy true;

                client_max_body_size 500M;

                proxy_buffer_size          128k;
                proxy_buffers              4 256k;
                proxy_busy_buffers_size    256k;

                proxy_connect_timeout 300s;
                proxy_read_timeout 600s;
                proxy_send_timeout 600s;
        }

        location /actuator {#actuator social login ^~ location ^~ /(api|actuator)
                proxy_pass http://j9a105.p.ssafy.io:8081/actuator;
                proxy_redirect off;
                charset utf-8;

                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;
                proxy_set_header X-NginX-Proxy true;

                client_max_body_size 500M;

                proxy_buffer_size          128k;
                proxy_buffers              4 256k;
                proxy_busy_buffers_size    256k;

                proxy_connect_timeout 300s;
                proxy_read_timeout 600s;
                proxy_send_timeout 600s;
        }
        location /node {
                proxy_pass http://j9a105.p.ssafy.io:8084;
                proxy_redirect off;
                charset utf-8;

                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
        }


    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/j9a105.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/j9a105.p.ssafy.io/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

}
server {
    if ($host = j9a105.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

        listen 80 ;
        listen [::]:80 ;
    server_name j9a105.p.ssafy.io;
    return 404; # managed by Certbot
}
```

빌드

백엔드

windows

1. `sudo chmod +x ./gradlew.bat`
2. `./gradlew.bat clean build`

Linux, macOS

1. `sudo chmod +x ./gradlew`
2. `./gradlew clean build`

프론트엔드

- npm
    - `npm run build`
- yarn
    - `yarn build`

배포 서버 구성

OS

- Ubuntu 20.04

SSL 인증서(Certbot)

1. 우분투 시스템 패키지 업데이트
    - `sudo apt update`
2. let's encrypt 설치
    - `sudo apt-get install letsencrypt`
3. 인증서 발급
    - `sudo certbot certonly -d "*.ssafy.io" --manual --preferred-challenges dns`

Docker

1. 우분투 시스템 패키지 업데이트
    - `sudo apt-get update`
2. 필요한 패키지 설치
    - `sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common`
3. Docker의 공식 GPG키 추가
    - `curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -`
4. Docker의 공식 apt 저장소 추가
    - `sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"`
5. 시스템 패키지 업데이트
    - `sudo apt-get update`
6. Docker 설치
    - `sudo apt-get install docker-ce docker-ce-cli containerd.io`
7. 실행상태 확인
    - `sudo systemctl status docker`

NGINX

1. 우분투 패키지 업데이트
    - `sudo apt-get update`
2. NGINX 설치
    - `sudo apt-get install nginx`
3. NGINX 버전 확인
    - `sudo nginx -v`
4. NGINX 시작
    - `sudo systemctl start nginx`

Openvidu

1. Openvidu 이미지 다운로드
    - `docker pull openvidu/openvidu-dev`
2. opt 디렉터리 이동
    - `cd /opt`
3. .env 파일에 환경 변수 작성
    - `vim .env`
4. Openvidu 실행
    - `./openvidu start`

MySQL

1. MySQL 이미지 다운로드
    - `docker pull mysql:latest`
2. MySQL 실행
    - `docker run --name some-mysql -e MYSQL_ROOT_PASSWORD=password -d mysql:latest`

Redis

1. Redis 이미지 다운로드

- `docker pull redis:latest`

2. Redis 실행

- `docker run -p 6379:6379 --name some-redis -d redis:latest`