자율 포팅 메뉴얼

목차

사용 프로그램 버전

- **Java**: `zulu-17`
- **Spring Boot**: `3.1.5`
- **gradle**: `8.3`
- **MariaDB**: `10.5`
- **Redis**: `6.2.6`
- **mongoDB**: `7.0.2`
- **Jenkins**: `2.414.3`
- **npm**: `9.5.1`
- **yarn**: `1.22.19`
- **React**: `18.2.0`

시스템 구성

Frontend

- **CloudFront**: `https://app.zupzup.shop`

Development Server

domain: `https://zupzup.shop`

- **Spring Boot:** `8080:8080`
- **Jenkins:** `https://jenkins.zupzup.shop`
  - `9090:9090`
  - `50000:50000`
- **MariaDB**: `3306:3306`
- **Redis**: `6379:6379`
- **mongoDB**: `27017:27017`

Production Server

domain: `https://api.zupzup.shop`

- **Spring Boot:** `8080:8080`
- **Jenkins:** `https://jenkins.api.zupzup.shop`
  - `9090:9090`
  - `50000:50000`
- **MariaDB**: `3306:3306`
- **Redis**: `6379:6379`
- **mongoDB**: `27017:27017`

환경 파일 세팅

백엔드

application-prod.yml

```
spring:
  cache:
    jcache:
      config: classpath:ehcache.xml
  datasource:
    driver-class-name: org.mariadb.jdbc.Driver
    url: ${MARIA_URL}
    username: ${MARIA_USER}
    password: ${MARIA_PASSWORD}
  jpa:
    hibernate:
```

```
      ddl-auto: validate
  data:
    redis:
      host: ${REDIS_HOST}
      port: ${REDIS_PORT}
      password: ${REDIS_PASSWORD}
    mongodb:
      host: ${MONGO_HOST}
      port: ${MONGO_PORT}
      database: ${MONGO_DATABASE}
      username: ${MONGO_USERNAME}
      password: ${MONGO_PASSWORD}
      authentication-database: admin
  security:
    oauth2:
      client:
        registration:
          google:
            client-id: ${GOOGLE_CLIENT_ID}
            client-secret: ${GOOGLE_CLIENT_SECRET}
            redirect-uri: ${GOOGLE_REDIRECT_URL}
            scope: email, profile, openid
          kakao:
            client-id: ${KAKAO_CLIENT_ID}
            client-secret: ${KAKAO_CLIENT_SECRET}
            redirect-uri: ${KAKAO_REDIRECT_URL}
            scope: profile_nickname, openid
            client-name: Kakao
            authorization-grant-type: authorization_code
            client-authentication-method: client_secret_post
        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/v1/oidc/userinfo
            jwk-set-uri: https://kauth.kakao.com/.well-known/jwks.json
oidc:
  google:
    iss: https://accounts.google.com
    iss2: accounts.google.com
  kakao:
    iss: https://kapi.kakao.com
    iss2: kapi.kakao.com

client:
  url: ${CLIENT_URL}
  redirect:
    login-success: ${CLIENT_REDIRECT_LOGIN_SUCCESS}

jwt:
  secret-key: ${JWT_SECRET_KEY}
  auth-token-expired_second: 60
  access-expired-second: 7200
  refresh_expired_second: 1209600

management:
  endpoints:
    enabled-by-default: false
    jmx:
      exposure:
        exclude: "*"
    web:
      exposure:
        include: health, info, metrics
  endpoint:
    health:
      enabled: true
    info:
      enabled: true
    metrics:
      enabled: true
  server:
    port: ${HEALTH_PORT}
```

docker-compose.yml

```
version: "3"
services:
  spring:
    container_name: spring
    build:
      context: ./backend/zupzup
      args:
        JWT_KEY: ${JWT_SECRET_KEY}
    environment:
      MARIA_URL: ${MARIA_URL}
      MARIA_USER: ${MARIA_USER}
      MARIA_PASSWORD: ${MARIA_PASSWORD}
      REDIS_HOST: ${REDIS_HOST}
      REDIS_PORT: ${REDIS_PORT}
      REDIS_PASSWORD: ${REDIS_PASSWORD}
      MONGO_HOST: ${MONGO_HOST}
      MONGO_PORT: ${MONGO_PORT}
      MONGO_DATABASE: ${MONGO_DATABASE}
      MONGO_USERNAME: ${MONGO_USERNAME}
      MONGO_PASSWORD: ${MONGO_PASSWORD}
      CLIENT_URL: ${CLIENT_URL}
      CLIENT_REDIRECT_LOGIN_SUCCESS: ${CLIENT_REDIRECT_LOGIN_SUCCESS}
      GOOGLE_CLIENT_ID: ${GOOGLE_CLIENT_ID}
      GOOGLE_CLIENT_SECRET: ${GOOGLE_CLIENT_SECRET}
      GOOGLE_REDIRECT_URL: ${GOOGLE_REDIRECT_URL}
      KAKAO_CLIENT_ID: ${KAKAO_CLIENT_ID}
      KAKAO_CLIENT_SECRET: ${KAKAO_CLIENT_SECRET}
      KAKAO_REDIRECT_URL: ${KAKAO_REDIRECT_URL}
      JWT_SECRET_KEY: ${JWT_SECRET_KEY}
      HEALTH_PORT: ${HEALTH_PORT}
      DATADOG_API_KEY: ${DATADOG_API_KEY}
      DATADOG_APPLICATION_KEY: ${DATADOG_APPLICATION_KEY}
      SECURITY_PERMITTED_URLS: ${SECURITY_PERMITTED_URLS}
    ports:
      - "8080:8080"
    networks:
      - deploy
    restart: always

networks:
  deploy:
    external: true
```

.env

```
MARIA_URL=jdbc:mariadb://mariadb:3306/zupzup
MARIA_USER=twoeasy
MARIA_PASSWORD=mzUWudjUsbTpxqBR53VM5FZS

REDIS_HOST=redis
REDIS_PORT=6379
REDIS_PASSWORD=9rMeP1Tk1RHuZRJLSs1ujjYfe

MONGO_HOST=api.zupzup.shop
MONGO_PORT=27017
MONGO_DATABASE=admin
MONGO_USERNAME=root
MONGO_PASSWORD=L25iryBLMjTu0jnfa8R1viA2r

CLIENT_URL=https://app.zupzup.shop
# CLIENT_URL=https://localhost:5173
CLIENT_REDIRECT_LOGIN_SUCCESS=/login-success

GOOGLE_CLIENT_ID=688197126446-0ncc44hv52kf76nph3j2brrehmjic5h3.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET=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
GOOGLE_REDIRECT_URL=https://api.zupzup.shop/login/oauth2/code/google

KAKAO_CLIENT_ID=128c4e7bb850bb6e4e34900450cf1445
KAKAO_CLIENT_SECRET=LV3jDh8Ng7ADiVstoqcrJC5HE3ou7rWg
KAKAO_REDIRECT_URL=https://api.zupzup.shop/login/oauth2/code/kakao

JWT_SECRET_KEY=wingardiumleviosaexpectopetroniumpetrificustotaluswingardiumleviosaexpectopetroniumpetrificustotaluswingardiumleviosaexpectopetroniumpetrificustotaluswingardiumleviosaexpectopetroniumpetrificust

HEALTH_PORT=8090

DATADOG_API_KEY=bf4728a40b3f6b90cb5b08cfd2a8ecf5
DATADOG_APPLICATION_KEY=38b6a6fb92d42b8a524b0359514a2226add6824

SECURITY_PERMITTED_URLS=login/**, error, api/v1/auth, actuator/**, api/v1/members/register, api/v1/docs/api
```

Dockerfile

```
FROM azul/zulu-openjdk:17-latest AS builder
WORKDIR /app

COPY gradlew ./
COPY gradle ./gradle
COPY build.gradle ./
COPY settings.gradle ./
COPY src ./src

RUN chmod +x ./gradlew
RUN ./gradlew spotlessApply
RUN ./gradlew clean build -x test -x asciidoctor

FROM azul/zulu-openjdk:17-latest
WORKDIR /app
COPY --from=builder /app/build/libs/*.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java","-jar","-Dspring.profiles.active=prod" ,"app.jar"]
```

docker-compose.mariadb.yml

```
version: "3"
services:
  mariadb:
    image: mariadb:10.5
    container_name: mariadb
    environment:
      TZ: Asia/Seoul
      MYSQL_ROOT_PASSWORD: mzUWudjUsbTpxqBR53VM5FZS
      MYSQL_USER: twoeasy
      MYSQL_PASSWORD: mzUWudjUsbTpxqBR53VM5FZS
      MYSQL_DATABASE: zupzup
      MYSQL_CHARACTER_SET_SERVER: utf8mb4
      MYSQL_COLLATION_SERVER: utf8mb4_unicode_ci
    ports:
      - 3306:3306
    volumes:
      - ./conf.d:/etc/mysql/conf.d
      - mariadb:/var/lib/mysql
    networks:
      - deploy
    restart: always

networks:
  deploy:
    external: true
volumes:
  mariadb:
```

docker-compose.mongodb.yml

```
version: '3.8'
services:
  mongodb:
    image: mongo:latest
    container_name: mongodb
    restart: always
    ports:
      - 27017:27017
    volumes:
      - ./mongodb:/data/db
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: L25iryBLMjTu0jnfa8R1viA2r
      MONGO_INITDB_DATABASE: admin
    networks:
      - deploy

networks:
  deploy:
    external: true
```

docker-compose.redis.yml

```
version: "3"
services:
  redis:
    container_name: redis
    image: redis:6.2.6-alpine
    command: redis-server --requirepass ${REDIS_PASSWORD}
    ports:
      - 6379:6379
    networks:
      - deploy
    restart: always

networks:
  deploy:
    external: true
```

redis/.env

```
REDIS_PASSWORD=9rMeP1Tk1RHuZRJLSs1ujjYfe
```

웹 서버 (NGINX, conf.d/default.conf)

```
upstream backend {
    server 0.0.0.0:8080;
}

server {
    server_name api.zupzup.shop www.api.zupzup.shop;

    location / {
        proxy_pass http://backend;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        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;
    }

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/api.zupzup.shop/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/api.zupzup.shop/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 ($host = api.zupzup.shop) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    server_name api.zupzup.shop www.api.zupzup.shop;
    return 404; # managed by Certbot
}
```

빌드

백엔드

windows

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

Linux, macOS

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

Jenkins pipeline

```
pipeline {
  agent any
  stages {
    stage('Clone Repository') {
      steps {
        echo "Clone repository"
        git branch: "develop", url: "https://lab.ssafy.com/s09-final/S09P31A202", credentialsId: "lio8625"
      }
    }
    stage("Set environment") {
      steps {
        echo "Copy require file to pipeline folder"
        sh 'cp /var/jenkins_home/util/zupzup/docker-compose.yml .'
        sh 'cp /var/jenkins_home/util/zupzup/.env .'
        sh 'cp /var/jenkins_home/util/zupzup/Dockerfile ./backend/zupzup'
      }
    }
    stage('Docker down') {
      steps {
          sh 'pwd'
        echo "Docker compose down"
        sh "sudo docker-compose -f docker-compose.yml down --rmi all"
      }
    }
    stage('Docker build') {
      steps {
        echo "docker compose build"
```

```
        sh "sudo docker-compose -f docker-compose.yml build"
      }
      post {
        success {
        echo "Success to build"
      }
        failure {
          echo "Docker build failed. clear unused file"
          sh "sudo docker system prune -f"
          error 'pipeline aborted'
        }
      }
    }
    stage('Docker up') {
      steps {
        echo "docker compose up"
        sh "sudo docker-compose -f docker-compose.yml up -d"
      }
    }
    stage('HealthCheck') {
      steps {
        script {
          for (int i = 0; i < 10; i++) {
            try {
              sh 'curl spring:8090/actuator/health > /dev/null'
              currentBuild.result = 'SUCCESS'
              break
            } catch (Exception e) {
              if (i == 9) {
                  currentBuild.result = 'FAILURE'
              } else {
                echo "The server is not alive yet. Retry health check in 3 seconds..."
                sleep(4)
              }
            }
          }
        }
      }
      post {
        failure {
          echo "Release Fail. clear unused file"
          sh "sudo docker system prune -f"
          error 'pipeline aborted'
        }
        success {
          echo "Release Success"
        }
      }
    }
    stage('Docker clear') {
      steps{
        sh "docker system prune -f"
      }
    }
  }
}
```

프론트엔드

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

배포 서버 구성

OS

- Ubuntu 20.04

Docker Compose

- `docker-compose up`

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 "*.zupzup.shop" --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`

MariaDB

1. MariaDB 이미지 다운로드
    - `docker pull mariadb:10.5`
2. MariaDB 실행
    - `docker run --detach --name some-mariadb --env MARIADB_ROOT_PASSWORD=my-secret-pw  mariadb:10.5`

Redis

1. Redis 이미지 다운로드
    - `docker pull redis:6.2.6-alpine`
2. Redis 실행
    - `docker run -p 6379:6379 --name some-redis -d redis:6.2.6-alpine`

MongoDB

1. MongoDB 이미지 다운로드
    - `docker pull mongo:latest`
2. MongoDB 실행
    - `docker run -itd --name mongodb mongo:latest`

시연 시나리오

- 카카오 or 구글로 로그인 및 회원 가입
    - 사용자 신체 정보(몸무게, 키) 입력
    - 미입력 시 대한민국 평균치로 임시 저장
- 메인 화면
    - 현재 플로깅 중인 유저 수 확인
    - 최근 플로깅 일자 및 이동 거리 확인
    - 지도 내 현재 위치 표시
- 플로깅 중
    - 실시간 플로깅 정보 화면
        - 타이머에 소요 시간 표시
        - 여태까지 플로깅하며 이동한 거리 표시
        - 여태까지 소모한 칼로리 표시
        - 여태까지 획득한 코인 표시
        - 지도 확인 버튼
        - 카메라 버튼(쓰레기 인식)
        - 종료하기 버튼
    - 지도 화면
        - 현재까지 이동한 경로 표시
        - 현재 위치로 지도 시점 이동 버튼
        - 현재 사용자 근처 쓰레기통 조회
        - 카메라 버튼(쓰레기 인식)
        - 종료하기 버튼
    - 카메라 화면
        - 쓰레기 촬영
        - 쓰레기 종류 구분 및 코인 제공
- 플로깅 종료 시
    - 지도 내 이동한 경로 표시
    - 총 플로깅 소요 시간 표시
    - 총 회득한 코인 표시
    - 총 이동 거리 표시
    - 마이페이지로 이동 or 메인 화면으로 이동
- 마이 페이지

    - 현재 캐릭터(펭깅) 애니메이션 출력
    - 플로깅한 일수 표시
    - 현재 레벨 및 경험치 표시
    - 사용자 건의함 제공
    - 튜토리얼 제공
    - 누적 레포트 제공
    - 테마 변경
    - 로그 아웃
- 상점 페이지
    - 아이템 정보 제공
    - 아이템 구매
- 캘린더 화면
    - 과거 플로깅 정보 조회
    - 과거 플로깅 기록 내 주운 쓰레기 정보 상세 조회