포팅매뉴얼

프로젝트 구성도

개발 환경

- Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1063-aws x86_64)
- IntelliJ IDEA (2024.1.4)
- Visual Studio Code (1.86)

FrontEnd

- Next.JS 15
- TypeScript 5

- Sass 1.8

BackEnd

- Java 17 (openjdk-17)
- Spring Boot 3.3.5
- Hibernate 6.5.3

DB

- PostgreSQL 17.0
- Redis 7.4.1

INFRA

- Ubuntu 20.04.6
- Docker 27.3.1
- Docker Compose 2.29.7
- Jenkins 2.13.0
- NginX 1.27.2

소프트웨어 버전

소프트웨어	버전
Java	17 (openjdk-17)
Spring Boot	3.3.5
PostgreSQL	17.0
Redis	7.4.1
Ubuntu	20.04.6
Docker	27.3.1
Docker Compose	2.29.7

소프트웨어	버전
Jenkins	2.13.0
nginx	1.27.2
Next.JS	15
TypeScript	5
Sass	1.8

⚙️ 환경 설정

🔒 환경변수

```
/*.env*/
NEXT_PUBLIC_API_URL=https://j11a304.p.ssafy.io/api/api



/*application.yml*/
OPENAI_API_MODEL=gpt-4o
OPENAI_API_KEY=sk-proj-hwjMr9RSSBuF-jExtATg0-S1Xdaq18Or0XqGIT
KOREAN_API_URL=https://stdict.korean.go.kr/api/search.do
KOREAN_API_KEY=69E398DCF8DC5D67162322B845E64F47
SPRING_DATASOURCE_URL=jdbc:postgresql://dannae.kr:5432/dannae
SPRING_DATASOURCE_USERNAME=postgres
SPRING_DATASOURCE_PASSWORD=!dannaepw
SPRING_DATA_REDIS_HOST=dannae.kr
SPRING_DATA_REDIS_PASSWORD=!redispw
SPRING_DATA_REDIS_PORT=6379
SPRING_JPA_HIBERNATE_DDL_AUTO=update
SPRING_JPA_SHOW_SQL=true
```

🐳 Docker 설치

```
# Docker 설치를 위한 패키지 업데이트
sudo apt update
# 필수 패키지 설치
sudo apt install -y apt-transport-https ca-certificates curl
# Docker GPG 키 추가
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sud
# Docker 저장소 추가
sudo add-apt-repository "deb [arch=amd64] https://download.do
# 패키지 목록 업데이트
sudo apt update
# Docker 설치
sudo apt install -y docker-ce
# Docker 설치 확인
docker --version
# Docker 서비스 시작 및 부팅 시 자동 실행 설정
sudo systemctl start docker
sudo systemctl enable docker
```

🔐 SSL 인증서 발급

```
sudo apt update
sudo apt install certbot python3-certbot-nginx
sudo certbot --nginx -d ${domain}
```

🚪 EC2 포트 번호

서비스	포트번호
Back-End	8080
Front-End	3000
PostgreSQL	5432
Redis	6379
Jenkins	8005
Nginx	80(http), 443(https)

⚒️ 빌드방법

1. 도커 컴포즈를 이용, 젠킨스, 데이터베이스 컨테이너 실행

```
docker network create dockernetwork
docker compose up -d jenkins postgres redis
docker exec -it postgres bash
sed -i 's/trust/scram-sha-256/g' /var/lib/postgresql/data/pg_
psql -U postgres -c "SELECT pg_reload_conf();"
```

```
# docker-compose.yml
services:
  jenkins:
    image: jenkins/jenkins:lts
    container_name: jenkins
    restart: unless-stopped
    privileged: true
    user: root
    ports:
      - "8005:8080"
      - "50000:50000"
    networks:
      - dockernetwork
    volumes:
      - jenkins_home:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
      - /usr/bin/docker:/usr/bin/docker
      - /usr/libexec/docker/cli-plugins/docker-compose:/usr/l
      - /home/ubuntu:/home/ubuntu
    environment:
      - TZ=Asia/Seoul
  backend:
    image: backend:latest
    container_name: backend_service
    ports:
      - "8080:8080"
    environment:
      - OPENAI_API_KEY=${OPENAI_CREDS_PSW}
      - OPENAI_API_MODEL=${OPENAI_CREDS_USR}
      - JWT_SECRET_KEY=${JWT_SECRET_KEY}
      - TZ=Asia/Seoul
      - JAVA_OPTS=-Duser.timezone=Asia/Seoul
      - SPRING_DATASOURCE_URL=${DB_URL}
      - SPRING_DATASOURCE_USERNAME=${DB_USER}
      - SPRING_DATASOURCE_PASSWORD=${DB_PASSWORD}
      - SPRING_DATA_REDIS_HOST=redis
      - SPRING_DATA_REDIS_PORT=6379
      - SPRING_DATA_REDIS_PASSWORD=!redispw
      - SPRING_JPA_HIBERNATE_DDL_AUTO=update
      - SPRING_JPA_SHOW_SQL=true
      - SPRING_REDIS_HOST=redis
      - KOREAN_API_URL=${KOREAN_CREDS_USR}
      - KOREAN_API_KEY=${KOREAN_CREDS_PSW}
    depends_on:
      - postgres
      - redis
    networks:
      - dockernetwork
    restart: unless-stopped

  frontend:
    image: frontend:latest
    container_name: frontend_service
    ports:
      - "3000:3000"
    environment:
      - API_URL=http://backend_service:8080
      - TZ=Asia/Seoul
    networks:
      - dockernetwork
    restart: unless-stopped

  postgres:
    image: postgres:17
    container_name: postgres
    networks:
      - dockernetwork
    environment:
      - POSTGRES_DB=dannae
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=!dannaepw
      - timezone=Asia/Seoul
      - POSTGRES_HOST_AUTH_METHOD=scram-sha-256
      - POSTGRES_INITDB_ARGS=--auth-host=scram-sha-256 --auth
    command:
      - "postgres"
      - "-c"
      - "password_encryption=scram-sha-256"
    volumes:
      - postgres_data:/var/lib/postgresql/data
      #- ./pg_hba.conf:/var/lib/postgresql/data/pg_hba.conf:r
      - /home/ubuntu/postgres_backup.sql:/docker-entrypoint-i
    restart: unless-stopped

  redis:
    image: redis:alpine
    container_name: redis
    networks:
      - dockernetwork
    environment:
      - TZ=Asia/Seoul
    expose:
      - "6379"
    ports:
      - "6379:6379"
    volumes:
      - redis_data:/data
    command: redis-server --appendonly yes --requirepass "!re
    restart: unless-stopped

  nginx:
    image: nginx:alpine
    container_name: nginx
    ports:
      - "80:80"
      - "443:443"
    environment:
      - TZ=Asia/Seoul
    volumes:
      - /etc/nginx/conf.d:/etc/nginx/conf.d
      - /etc/letsencrypt:/etc/letsencrypt
    networks:
      - dockernetwork
    restart: unless-stopped
    user: root
    depends_on:
      - frontend
      - backend

networks:
  dockernetwork:
    external: true

volumes:
  postgres_data:
  redis_data:
  jenkins_home:
    driver: local
```

2. 젠킨스 파이프라인 이용, 백, 프론트 컨테이너 실행

```
pipeline {
    agent any
    environment {
        FE_DOCKER_IMAGE = 'frontend'
        BE_DOCKER_IMAGE = 'backend'
        DOCKER_TAG = "${BUILD_NUMBER}"
        MATTERMOST_WEBHOOK = credentials('mattermost-webhook'
        // Backend specific credentials
        OPENAI_CREDS = credentials('openai-credentials')
        KOREAN_CREDS = credentials('korean-api')
        JWT_SECRET_KEY = credentials('jwt-secret')
        DB_URL = credentials('DB_URL')
        DB_USER = credentials('DB_USER')
        DB_PASSWORD = credentials('DB_PASSWORD')
    }

    stages {
        stage('Checkout') {
            steps {
                git branch: 'master',
                    credentialsId: 'gitlab-token',
                    url: 'https://lab.ssafy.com/s11-final/S11
            }
        }

        stage('Build and Deploy') {
            parallel {
                stage('Frontend') {
                    stages {
                        stage('Build Frontend Docker Image')
                            steps {
                                dir('frontend') {
                                    script {
                                        try {
                                            sh "docker build
                                            sh "docker tag ${
                                        } catch (Exception e)
                                            sh "docker rmi ${
                                            sh "docker rmi ${
                                            throw e
                                        }
                                    }
                                }
                            }
                        }

                        stage('Deploy Frontend') {
                            steps {
                                script {
                                    try {
                                        sh """
                                            cd /home/ubuntu
                                            docker compose st
                                            docker compose rm
                                            docker compose up
                                        """
                                    } catch (Exception e) {
                                        echo "Frontend deploy

                                        sh """
                                            docker rmi ${FE_D
                                            docker rmi ${FE_D
                                        """

                                        def rollbackTag = sh(
                                            script: """
                                                for ((i=${BUI
                                                    if docker
                                                        echo
                                                        exit
                                                    fi
                                                done
                                                echo 'not_fou
                                            """,
                                            returnStdout: tru
                                        ).trim()

                                        if (rollbackTag != 'n
                                            echo "Rolling bac
                                            sh """
                                                cd /home/ubun
                                                docker compos
                                                docker compos
                                                docker tag ${
                                                docker compos
                                            """
                                        } else {
                                            echo "Warning: No
                                        }
                                        throw e
                                    }
                                }
                            }
                        }
                    }
                }

                stage('Backend') {
                    stages {
                        stage('Build Backend') {
                            steps {
                                dir('backend/dannae') {
                                    sh 'chmod +x ./gradlew'
                                    sh './gradlew clean build
                                }
                            }
                        }

                        stage('Build Backend Docker Image') {
                            steps {
                                dir('backend/dannae') {
                                    script {
                                        try {
                                            sh "docker build
                                            sh "docker tag ${
                                        } catch (Exception e)
                                            sh "docker rmi ${
                                            sh "docker rmi ${
                                            throw e
                                        }
                                    }
                                }
                            }
```

```
                    }

                    stage('Deploy Backend') {
                        steps {
                            script {
                                try {
                                    sh """
                                        cd /home/ubuntu
                                        docker compose st
                                        docker compose rm
                                        docker compose up
                                    """
                                } catch (Exception e) {
                                    echo "Backend deploym

                                    sh """
                                        docker rmi ${BE_D
                                        docker rmi ${BE_D
                                    """

                                    def rollbackTag = sh(
                                        script: """
                                            for ((i=${BUI
                                                if docker
                                                    echo
                                                    exit
                                                fi
                                            done
                                            echo 'not_fou
                                        """,
                                        returnStdout: tru
                                    ).trim()

                                    if (rollbackTag != 'n
                                        echo "Rolling bac
                                        sh """
                                            cd /home/ubun
                                            docker compos
```

```
                                                docker compos
                                                docker tag ${
                                                docker compos
                                            """
                                        } else {
                                            echo "Warning: No
                                        }
                                        throw e
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }

    post {
        always {
            script {
                // 작업 공간 정리
                cleanWs()
                sh 'docker image prune -f --filter "until=24h
            }
        }
    }
}
```

3. nginx 컨테이너 실행

```
#/etc/nginx/conf.d/default.conf
server {
    listen 80;
    server_name dannae.kr www.dannae.kr;
    return 301 https://$host$request_uri;
}
server {
    #access_log /var/log/nginx/access.log main;
    #error_log /var/log/nginx/error.log;
    listen 443 ssl;
    server_name dannae.kr www.dannae.kr;
    ssl_certificate /etc/letsencrypt/live/dannae.kr/fullchain
    ssl_certificate_key /etc/letsencrypt/live/dannae.kr/privk
    location /api/next/ {
        proxy_pass http://frontend_service:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
    # Frontend
    location / {
        proxy_pass http://frontend_service:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
    # Backend API
    location /api {
        proxy_pass http://backend_service:8080;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }

    # WebSocket
    location /ws {
      proxy_pass http://backend_service:8080/ws;
      proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection "upgrade";
      proxy_set_header Host $host;
      proxy_cache_bypass $http_upgrade;

      # WebSocket
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_
      proxy_set_header X-Forwarded-Proto $scheme;
      proxy_buffering off;
  }
}


#/etc/nginx/con.f/k11a308.conf
server {
    listen 80;
    server_name k11a308.p.ssafy.io;
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl;
    server_name k11a308.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/p.ssafy.io/fullchai
    ssl_certificate_key /etc/letsencrypt/live/p.ssafy.io/priv

    location /api/next/ {
        proxy_pass http://frontend_service:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }

    location / {
        proxy_pass http://frontend_service:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }

    location /api {
        proxy_pass http://backend_service:8080;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }

    location /ws {
        proxy_pass http://backend_service:8080/ws;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_buffering off;
    }
}
```

도커 파일

```
#/frontend/Dockerfile
# 1. Node.js 20.15.1 버전을 사용한 베이스 이미지 선택
FROM node:20.15.1-alpine

# 2. 작업 디렉토리 생성
```

```
WORKDIR /app

# 3. package.json과 package-lock.json을 복사하고 의존성 설치
COPY package*.json ./
RUN npm install

# 4. Next.js 소스 파일 복사
COPY . .

# 5. Next.js 빌드
RUN npm run build

# 6. 포트 노출
EXPOSE 3000

# 7. Next.js 앱 실행
CMD ["npm", "start"]

#/backend/dannae/Dockerfile
FROM openjdk:17-jdk AS builder
WORKDIR /app

# RUN apt-get update && apt-get install -y findutils
RUN microdnf install findutils

# Gradle 파일들만 먼저 복사
COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .

# Gradle 실행 권한 부여
RUN sed -i 's/\r$//' gradlew && \
    chmod +x ./gradlew

# 소스 복사
COPY src src
```

```
# 빌드
RUN ./gradlew bootJar

# 실행 환경
FROM openjdk:17-slim
WORKDIR /app

# builder에서 생성된 jar 파일만 복사
COPY --from=builder /app/build/libs/*.jar app.jar

EXPOSE 8080
ENTRYPOINT ["java", "-jar", "app.jar"]
```