포팅 메뉴얼

Stacks

```
- Front
    - Vue 3
    - Node.js
    - vite: ^5.0.10
- Back
    - JDK 17
    - SpringBoot 3.2.2
    - SpringBatch 5.1.0
- DB
    - MySQL 8.0.35
- WebRTC
    - React
    - socket.io
```

Build & Distribute

```
    - Nginx
    - Spring Boot
    - Kafka
    - Vue
    - React
```

Nginx Setting

```
server {
        server_name i10b204.p.ssafy.io;

        location / {
                return 301 $scheme://joblog.pro$request_uri;
        }

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/i10b204.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/i10b204.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 {
        listen 80;
        server_name 43.201.249.9;

        location / {
                rewrite ^/coffee(/.*)$ $1 break;
                proxy_pass http://0.0.0.0:3003;
                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;
        }
}

server {
        server_name joblog.pro www.joblog.pro;

        location /api {
                rewrite ^/api(/.*)$ $1 break;
                proxy_pass http://0.0.0.0:8080;
                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;
        }

        location / {
                proxy_pass http://0.0.0.0:3000;
                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;
        }

        location /batch {
                rewrite ^/api(/.*)$ $1 break;
                proxy_pass http://0.0.0.0:8001;
                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/joblog.pro/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/joblog.pro/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 ($host = joblog.pro) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80;
        server_name joblog.pro www.joblog.pro;
```

```
    return 404; # managed by Certbot


}

server {
    if ($host = i10b204.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80;
        server_name 43.201.249.9 i10b204.p.ssafy.io;
    return 404; # managed by Certbot


}
```

Server URL

```
- Front Server : https://joblog.pro
- Back Server : https://joblog.pro/api
- WebRTC Server : http://43.249.201.9
- Batch Server : https://joblog.pro/batch
```

Port

```
- 2181 : zookeeper
- 3000 : frontend
- 3003 : webrtc
- 3306 : mysql
- 8001 : batch
- 8080 : backend
- 8989 : gerrit
- 9092->9093 : kafka
```

Dockerfile

- Back

```
FROM openjdk:17-alpine
COPY ./build/libs/*.jar app.jar
EXPOSE 8080 8085
ENTRYPOINT ["java", "-Dspring.profiles20.active=prod", "-jar","app.jar"]
```

- Front

```
FROM node:lts AS build
WORKDIR /usr/src/app
COPY package*.json ./
COPY .env ./.env
RUN npm install
COPY . .
RUN npm run build
EXPOSE 3000
#RUN npm run start
CMD ["npm", "run", "dev"]
```

- Batch

```
FROM openjdk:17-alpine
WORKDIR /app
```

```
COPY ./build/libs/*.jar app.jar
EXPOSE 8001
ENTRYPOINT ["java", "-Dspring.profiles20.active=prod", "-jar","app.jar"]
```

- WebRTC

```
FROM node:lts AS build
WORKDIR /usr/src/app
RUN npm i express ejs socket.io
RUN npm i uuid
RUN npm i --save-dev nodemon
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3003
CMD ["npm", "run", "devStart"]
```

docker-compose.yml

- Kafka + Zookeeper

```
version: "3"
services:
  zookeeper:
    image: "bitnami/zookeeper:latest"
    container_name: "zookeeper"
    networks:
      - deploy
    ports:
      - "2181:2181"
    environment:
      - ALLOW_ANONYMOUS_LOGIN=yes
  kafka:
    image: "bitnami/kafka:latest"
    container_name: "kafka"
    networks:
      - deploy
    ports:
      - "9092:9093"
    environment:
      - KAFKA_CFG_NODE_ID=0
      - KAFKA_CFG_PROCESS_ROLES=controller,broker
      - KAFKA_CFG_CONTROLLER_QUORUM_VOTERS=0@kafka:9093
      - KAFKA_CFG_LISTENERS=PLAINTEXT://:9092,CONTROLLER://:9093
      - KAFKA_CFG_ADVERTISED_LISTENERS=PLAINTEXT://:9092
      - KAFKA_CFG_LISTENER_SECURITY_PROTOCOL_MAP=CONTROLLER:PLAINTEXT,PLAINTEXT:PLAINTEXT
      - KAFKA_CFG_CONTROLLER_LISTENER_NAMES=CONTROLLER
      - KAFKA_CFG_INTER_BROKER_LISTENER_NAME=PLAINTEXT
    depends_on:
      - "zookeeper"
networks:
  deploy:
    external: true
```

환경설정파일

- Batch
  - application.yml

```
server:
  port: 8001
  servlet:
```

```
    context-path: /
    encoding:
      charset: UTF-8
      enabled: true
      force: true

# default
spring:
  profiles:
    active: local

---
spring:
  config:
    activate:
      on-profile: local
  batch:
    jdbc:
      initialize-schema: ALWAYS
    access-key:
      3arT09ZHgyNKWYkriX0I60XPns9zQoVtXKmgUSbSpkKWym7Lb6
  datasource:
    url: jdbc:mysql://172.18.0.2:3306/batch_test
    driver-class-name: com.mysql.cj.jdbc.Driver
    username: b204
    password: 54321
  jpa:
    show-sql: true
    hibernate:
      ddl-auto: update
---
```

- Backend
  - application.yml

```
spring:
  config:
    import: application-jwt.yml

  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://172.18.0.2:3306/joblogdb?serverTimezone=Asia/Seoul
    username: b204
    password: 54321
security:
  user:
    name: user
    password: 1234
  oauth2:
    client:
      registration:
        google:
          client-id: 670125628464-tuj6s6g18vht93kd3a2le6n3aivpsfta.apps.googleusercontent.com
          client-secret: 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
          redirect-uri: https://joblog.pro/api/login/oauth2/code/google
          scope:
            - profile
            - email
        naver:
          client-id: 9UJ3Z3P0oY0mQPIpbP34
          client-secret: Mhf43fXRWd
          scope:
            - name
            - email
```

```
            client-name: Naver
            authorization-grant-type: authorization_code
            redirect-uri: https://joblog.pro/api/login/oauth2/code/naver
          kakao:
            client-id: 67f6bd42ea307bd92f4d4d858ca644b4
            client-secret: XRrYJlDyDqeQcSsI8HRU2fTO2R059u0u
            redirect-uri: https://joblog.pro/api/login/oauth2/code/kakao
            authorization-grant-type: authorization_code
            scope:
              - profile_nickname
            client-name: Kakao
            client-authentication-method: client_secret_post

        provider:
          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
          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
      jpa:
        properties:
          hibernate:
            format_sql: true
        hibernate:
          ddl-auto: update #create update none
    logging:
      level:
        org.hibernate.SQL: debug
    kafka:
      bootstrap-servers: 172.18.0.5:9092
      consumer:
        auto-offset-reset: earliest
    openai:
      secret-key: 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
      url:
        legacy-prompt: https://api.openai.com/v1/completions
```

- application-jwt.yml

```
spring:
  jwt:
    secret: "am9ibG9nam9ibG9nam9ibG9nam9ibG9nam9ibG9nam9ibG9nam9ibG9nam9ibG9nam9ibG9n"
```

- Front
  - .env

```
VITE_API_BASE_URL=https://joblog.pro/api/api

VITE_GOOGLE_API_KEY="AIzaSyBDjyrmorSZShfb9Rf6Mc62EjG3PuatfxI"
VITE_GOOGLE_CLIENT_ID="1034174863741-s8j7thfdb1imkqgg1lqbq8f9dnkj2frg.apps.googleusercontent.com"
```

Deployment Command

- mysql

```
- 아래 명령어로 실행
    docker run -d \ --network=deploy \ --name ssafy-db \ -p 3306:3306 \ -e MYSQL_ROOT_PASSWORD=ssafy \
mysql:latest
```

- Back

```
- JDK17 설치
    sudo apt install openjdk-17-jdk
- GitClone 받아서 Jar 파일 생성
    sudo sh ./gradlew clean bootJar --stacktrace
- joblog-backend의 docker image 생성(뒤에는 Dockerfile위치)
    sudo docker build -t joblog-backend ~/joblog/S10P12B204/backend/joblog/
- docker container 실행
    docker run -d -p 8080:8080 -p 8085:8085 --network deploy --name joblog-backend joblog-backend
```

- Batch

```
- JDK17 설치
    sudo apt install openjdk-17-jdk
- GitClone 받아서 Jar 파일 생성
    sudo sh ./gradlew clean bootJar --stacktrace
- joblog-batch의 docker image 생성(뒤에는 Dockerfile위치)
    sudo docker build -t joblog-batch ~/joblog/S10P12B204/batch/joblog/
- docker container 실행
    docker run -d -p 8001:801 --network deploy --name joblog-batch joblog-batch
```

- Frontend

```
- npm 설치
    sudo apt install npm
- node.js 설치
    curl -fsSL https://deb.nodesource.com/setup_19.x | sudo -E bash - &&sudo apt-get install -y nodejs
- joblog-frontend의 docker image 생성(뒤에는 Dockerfile위치)
    sudo docker build -t joblog-frontend ~/joblog/S10P12B204/FrontEnd
- docker container 실행
    sudo docker run -d -p 3000:3000 --network deploy --name joblog-frontend joblog-frontend
```

- WebRTC

```
- npm 설치
    sudo apt install npm
- node.js 설치
    curl -fsSL https://deb.nodesource.com/setup_19.x | sudo -E bash - &&sudo apt-get install -y nodejs
- joblog-webrtc의 docker image 생성(뒤에는 Dockerfile위치)
    sudo docker build -t joblog-webrtc~/joblog/webrtc/coffeechat
- docker container 실행
    sudo docker run -d -p 3000:3000 --network deploy --name joblog-webrtc joblog-webrtc
```

- Kafka + Zookeeper

```
- kafka + zookeeper의 docker-compose.yml 폴더로 이동
    cd ~/kafka
- docker compose 파일을 detach 모드로 실행
    sudo docker-compose up -d
```

외부 API 정보

```
- 소셜 로그인
    - 카카오 로그인 API
        https://developers.kakao.com/docs/latest/ko/kakaologin/rest-api
    - 네이버 로그인 API
        https://developers.naver.com/products/login/api/api.md
    - 구글 로그인 API
        https://console.cloud.google.com/apis/dashboard?pli=1
- OpenAI API
    - ChatGPT API
        - https://platform.openai.com/
```

```
- 맞춤법 검사 API
    - 부산대 맞춤법 검사기 API
        - https://github.com/9beach/hanspell
```