공통 포팅매뉴얼

깃랩 웹훅연결
pipeline

클라이언트 구축하기

버전

- react 18.3.1
- typescript 5.5.3
- vite 5.3.4
- sass 1.77.8

환경 변수

.env(직접 생성 필요)

```
VITE_API_URL=https://i11a309.p.ssafy.io
VITE_API_URL_LOCAL=http://localhost:8080
VITE_API_URL_MAIN=http://localhost:3000
VITE_API_URL_LOGIN=https://i11a309.p.ssafy.io/api
VITE_API_URL_LOCAL_SOCKET=http://localhost:8080/ws
VITE_API_URL_SERVER_SOCKET=https://i11a309.p.ssafy.io/wss
VITE_API_URL_RECOMMEND=https://i11a309.p.ssafy.io
VITE_API_URL_RECOMMEND_LOCAL=http://localhost:5000
```

빌드 매뉴얼

nginx.conf

```
    location / {
        proxy_pass http://localhost:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarde
        proxy_set_header X-Forwarded-Proto $scheme;

        # 웹소켓 연결 설정
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_cache_bypass $http_upgrade;
    }
```

Dockerfile

```
# Stage 1: Build the React app with Vite
FROM node:18 AS build

# Set the working directory inside the container
WORKDIR /app

# Copy package.json and package-lock.json (if available)
COPY package*.json ./

# Install dependencies
RUN npm install

# Copy the rest of the application source code
COPY . .

# Build the application
RUN npm run build

# Install `serve` to serve the built files
```

```
RUN npm install -g serve

# Expose the port the app runs on
EXPOSE 3000

# Start the built application using `serve`
CMD ["serve", "-s", "dist"]
```

API 서버, 데이터베이스, RabbitMQ 구축하기

- Java : ZULU-21
- Spring Boot 3
- Spring FrameWork
- Spring Data JPA
- Spring Security
- Gradle
- Redis
- mysql
- mongodb
- rabbitmq

환경변수

application.yml

```
server:
  port: 8080
  forward-headers-strategy: native

springdoc:
```

```
  api-docs:
    path: /api-docs
  packageToScan: AltTab

spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://mysql-db:3306/alttab_db
    username: root
    password: qwer1234

  jpa:
    database-platform: org.hibernate.dialect.MySQL8Dialect
    open-in-view: false
    show-sql: true
    hibernate:
      ddl-auto: update
      naming:
        physical:
          strategy: org.hibernate.boot.model.naming.PhysicalN

  thymeleaf:
    cache: false

  redis:
    host: redis
    port: 6379
    timeout: 5000

  data:
    mongodb:
      uri: mongodb://mongo:27017/alttab_db

  rabbitmq:
    host: rabbitmq
    port: 5672
    username: guest
    password: guest
```

```
  security:
    oauth2:
      client:
        registration:
          github:
            client-id: Ov23lifHK4BZ1RLMnHCe
            client-secret: 721ad19451e62dadd8747fd3b3614f8946
            redirect-uri: "{baseUrl}/api/login/oauth2/code/{r

  mail:
    host: smtp.gmail.com
    port: 587
    username: ssafyalttab
    password: fzehgolzorcuigzf
    properties:
      mail:
        smtp:
          auth: true
          timeout: 5000
          starttls:
            enable: true

axios:
  defaults:
    headers['Access-Control-Allow-Origin']: '*'
    withCredentials: true



app:
  front:
    url: "https://i11a309.p.ssafy.io"

logging:
  level:
    org:
      springframework:
```

```
        security: DEBUG
        oauth2: DEBUG
    com:
      ssafy:
        alttab: DEBUG

jwt:
  secret: your_very_long_and_very_secure_secret_key_here_mini
  access:
    expiration: 3600 # Token expiration time in seconds (1 ho
  refresh:
    expiration: 1209600 # 2 weeks
```

빌드 매뉴얼

nginx.conf

```
    location ^~ /api/ {
        proxy_pass http://localhost:8080;
        proxy_set_header Host $host;
        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;
    }
```

데이터베이스 docker-compose

```
  mysql-db:
    image: mysql:8.0
    container_name: mysql-db-container
    ports:
      - "3306:3306"
    environment:
      MYSQL_ROOT_PASSWORD: qwer1234
```

```
      MYSQL_DATABASE: alttab_db
    volumes:
      - mysql-data:/var/lib/mysql

  redis:
    image: redis:latest
    container_name: redis-container
    ports:
      - "6379:6379"
    volumes:
      - redis-data:/data

  mongo:
    image: mongo:latest
    container_name: mongo-container
    ports:
      - "27017:27017"
    volumes:
      - mongo-data:/data/db

  rabbitmq:
    image: rabbitmq:management
    container_name: rabbitmq-container
    ports:
      - "5672:5672"
      - "15672:15672" # RabbitMQ management UI

volumes:
  mysql-data:
    external: true
  redis-data:
    external: true
  mongo-data:
    external: true
```

Dockerfile

```
# Use an official Gradle image to build the application
FROM gradle:7.6.0-jdk17 AS build

# Set the working directory inside the container
WORKDIR /app

# Copy Gradle project files
COPY build.gradle settings.gradle ./

# Copy the rest of the application source code
COPY src ./src

# Build the application (create the JAR file)
RUN gradle clean build -x test --no-daemon

# Use an official OpenJDK runtime as a parent image
FROM openjdk:21-jdk-slim

# Set the working directory inside the container
WORKDIR /app

# Copy the JAR file from the previous stage
COPY --from=build /app/build/libs/*.jar app.jar

# Expose the port the application runs on
EXPOSE 8080

# Run the JAR file
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Executor 서버

버전

- Java : ZULU-21

- Spring Boot 3
- Spring FrameWork
- Gradle

환경변수

application.yml

```
server:
  port: 8081

  rabbitmq:
    host: rabbitmq
    port: 5672
    username: guest
    password: guest
```

빌드 매뉴얼

Dockerfile

```
# Use an official Gradle image to build the application
FROM gradle:7.6.0-jdk17 AS build

# Set the working directory inside the container
WORKDIR /app

# Copy Gradle project files
COPY build.gradle settings.gradle ./

# Copy the rest of the application source code
COPY src ./src
```

```
# Build the application (create the JAR file)
RUN gradle clean build -x test --no-daemon

# Use an official OpenJDK runtime as a parent image
FROM openjdk:21-jdk-slim

# Set the working directory inside the container
WORKDIR /app

# Copy the JAR file from the previous stage
COPY --from=build /app/build/libs/*.jar app.jar

# Expose the port the application runs on
EXPOSE 8081

# Run the JAR file
ENTRYPOINT ["java", "-jar", "app.jar"]
```

flask 서버(문제 추천 서버)

버전

- pathon 3.9
- flask
- scikit-learn

환경변수

.env(직접 생성 필요)

```
DB_HOST=mysql-db-container
DB_USER=root
DB_PASSWORD=qwer1234
DB_NAME=alttab_db

MONGO_URI=mongodb://i11a309.p.ssafy.io:27017/
MONGO_DB_NAME=baekjoon
MONGO_COLLECTION_NAME=problems_html
```

빌드 매뉴얼

nginx.conf

```
    location /flask {
        proxy_pass http://localhost:5000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarde
        proxy_set_header X-Forwarded-Proto $scheme;
    }
```

Dockerfile

```
# Use an official Python runtime as a parent image
FROM python:3.9

# Set the working directory in the container
WORKDIR /app

# Copy the requirements file into the container
COPY requirements.txt requirements.txt
```

```
# Install any dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Copy the current directory contents into the container
COPY . .

# Make port 5000 available to the world outside this containe
EXPOSE 5000

# Run app.py when the container launches
CMD ["gunicorn", "--bind", "0.0.0.0:5000", "app:app"]
```

requirements.txt

```
Flask
pandas
scikit-learn
pymysql
cryptography
gunicorn
mysql-connector-python
python-dotenv
flask-cors
pymongo
```

EC2 서버 세팅하기

Nignx + SSL을 이용한 HTTPS 세팅

```
server {
    listen 80 default_server;
```

```
    listen [::]:80 default_server;

    server_name _;

    # 모든 HTTP 요청을 HTTPS로 리디렉션
    return 301 https://i11a309.p.ssafy.io$request_uri;
}

server {
    listen 443 ssl;
    server_name i11a309.p.ssafy.io;

    root /var/www/html;
    index index.html index.htm index.nginx-debian.html;

    # /api location 설정 (정규표현식 사용)
    location ^~ /api/ {
        proxy_pass http://localhost:8080;
        proxy_set_header Host $host;
        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;
    }


location /wss {
        proxy_pass http://localhost:8080;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_http_version 1.1;
}



    # /recommend location 설정
    location /flask {
        proxy_pass http://localhost:5000;
        proxy_set_header Host $host;
```

```
        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;
    }


    # / location 설정
    location / {
        proxy_pass http://localhost:3000;
        proxy_set_header Host $host;
        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;

        # 웹소켓 연결 설정
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_cache_bypass $http_upgrade;
    }

    ssl_certificate /etc/letsencrypt/live/i11a309.p.ssafy.io/
    ssl_certificate_key /etc/letsencrypt/live/i11a309.p.ssafy
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}
```

배포하기

젠킨스 세팅하기

젠킨스 접속을 위한 변수

```
URL: http://43.203.248.134:8090
```

```
계정명 alttab_admin
암호 qwer1234
```

환경 변수 저장

Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	gitlab_token	ssh2957@naver.com/******
		System	(global)	alttab	ubuntu
		System	(global)	docker_hub_token	도커
		System	(global)	executor-config-yml	application.yml
		System	(global)	backend-config-yml	application.yml (ㄴㄴ)
		System	(global)	frontend-env	.env (ㄴㄴ)
		System	(global)	flask-env	.env (ㄴㄴ)

깃랩 웹훅연결

-아래 블로그 참고

[Jenkins] GitLab Webhooks를 이용한 젠킨스 연동 및 빌드유발

Summary Gitlab의 Webhooks은 프로젝트 내에서 어떠한 일이 발생할 때, 이벤트를 바인딩 하는데 사용 할 수 있습니다. 이 기능을 이용하면, 다양한 Event를 발생 시킬 수 있는데요. Gitlab에 있는 Webhooks 를 이용해

https://holjjack.tistory.com/288

pipeline

```
pipeline {
    agent any

    environment {
        DOCKER_HUB_CREDENTIALS = 'docker_hub_credentials'
        DOCKER_HUB_REPO = 'alttab-app-repo'
        DOCKER_HUB_USERNAME = 'ssh2957'
        HOST = 'ubuntu@43.203.248.134'
        IMAGE_TAG = 'latest'
```

```
        NETWORK = 'ubuntu_default' // 네트워크 이름 추가
    }

    stages {
        stage('Checkout') {
            steps {
                script {
                    git credentialsId: 'gitlab_token'
                    , branch: 'develop', url:
                    'https://lab.ssafy.com/s11-webmobile2-sub
                }
            }
        }

        stage('Copy Config Files') {
            steps {
                script {
                    withCredentials([
                        file(credentialsId: 'backend-config-y
                         variable: 'BACKEND_CONFIG_YML'),
                        file(credentialsId: 'executor-config-
                         variable: 'EXECUTOR_CONFIG_YML'),
                        file(credentialsId: 'frontend-env', va
                         'FRONTEND_ENV'),
                        file(credentialsId: 'flask-env', vari
                        'FLASK_ENV'),
                    ]) {
                        sh """
                            # Copy backend config files
                            mkdir -p backend/src/main/resourc
                            cp \$BACKEND_CONFIG_YML backend/s
                            /resources/application.yml

                            # Copy executor config files
                            mkdir -p executor/src/main/resour
                            cp \$EXECUTOR_CONFIG_YML executor
                            /resources/application.yml
```

```
                        # Copy frontend .env file
                        cp \$FRONTEND_ENV frontend/.env

                        # Copy frontend .env file
                        cp \$FLASK_ENV bigdata/.env

                    """
                }
            }
        }
    }

    stage('Build Docker Images') {
        steps {
            script {
                dir('frontend') {
                    sh 'docker build -t ${DOCKER_HUB_USER
                    }/${DOCKER_HUB_REPO}-react-app:${IMAG
                }
                dir('backend') {
                    sh 'docker build -t ${DOCKER_HUB_USER
                    /${DOCKER_HUB_REPO}-springboot-app:${
                }
                dir('executor') {
                    sh 'docker build -t ${DOCKER_HUB_USER
                    ${DOCKER_HUB_REPO}-executor-app:${IMA
                }
                dir('bigdata') { // Add this block
                    sh 'docker build -t ${DOCKER_HUB_USER
                    /${DOCKER_HUB_REPO}-flask-app:${IMAGE_
                }
            }
        }
    }

    stage('Push Docker Images') {
        steps {
            script {
```

```
                    withCredentials([string(credentialsId: 'd
                    variable: 'DOCKER_HUB_TOKEN')]) {
                        sh 'echo $DOCKER_HUB_TOKEN | docker l
                        _HUB_USERNAME} --password-stdin'
                        sh 'docker push ${DOCKER_HUB_USERNAME
                        REPO}-react-app:${IMAGE_TAG}'
                        sh 'docker push ${DOCKER_HUB_USERNAME
                        REPO}-springboot-app:${IMAGE_TAG}'
                        sh 'docker push ${DOCKER_HUB_USERNAME
                        REPO}-executor-app:${IMAGE_TAG}'
                        sh 'docker push ${DOCKER_HUB_USERNAME
                        REPO}-flask-app:${IMAGE_TAG}' // Add
                    }
                }
            }
        }

        stage('Deploy to Server') {
            steps {
                script {
                    sshagent(credentials: ['alttab']) {
                        sh """
                            ssh -o StrictHostKeyChecking=no $
                                echo \$DOCKER_HUB_TOKEN | doc
                                DOCKER_HUB_USERNAME} --passwo
                                docker stop react-app-contain
                                pp-container executor-app-con
                                -container || true &&
                                docker rm -f react-app-contai
                                pp-container executor-app-con
                                -container || true &&
                                docker pull ${DOCKER_HUB_USER
                                UB_REPO}-react-app:${IMAGE_TA
                                docker pull ${DOCKER_HUB_USER
                                B_REPO}-springboot-app:${IMAG
                                docker pull ${DOCKER_HUB_USER
                                UB_REPO}-executor-app:${IMAGE_
                                docker pull ${DOCKER_HUB_USER
```

```
                                B_REPO}-flask-app:${IMAGE_TAG
                                docker run -d --name react-ap
                                etwork ${NETWORK} -p 3000:300
                                ERNAME}/${DOCKER_HUB_REPO}-re
                                docker run -d --name springbo
                                 --network ${NETWORK} -p 8080
                                 BBITMQ_HOST=rabbitmq ${DOCKE
                                 CKER_HUB_REPO}-springboot-ap
                                docker run -d --name executor
                                --network ${NETWORK} -p 8081:
                                BITMQ_HOST=rabbitmq ${DOCKER_
                                ER_HUB_REPO}-executor-app:${I
                                docker run -d --name flask-ap
                                work ${NETWORK} -p 5000:5000
                                ME}/${DOCKER_HUB_REPO}-flask-
                            '
                        """
                    }
                }
            }
        }

        post {
            success {
                script {
                    dir("${env.WORKSPACE}") {
                        def Author_ID = sh(script: "git show -s -
                        rnStdout: true).trim()
                        def Author_Name = sh(script: "git show -s
                        urnStdout: true).trim()
                        mattermostSend(
                            color: 'good',
                            message: "빌드 성공: ${env.JOB_NAME} #$
                            y ${Author_ID}(${Author_Name})\n(<${e
                            endpoint: 'https://meeting.ssafy.com/
                            arbncefbufh',
                            channel: '#TEST'
```

```
                    )
                }
            }
            cleanWs()
        }
        failure {
            script {
                dir("${env.WORKSPACE}") {
                    def Author_ID = sh(script: "git show -s -
                    turnStdout: true).trim()
                    def Author_Name = sh(script: "git show -s
                    returnStdout: true).trim()
                    mattermostSend(
                        color: 'danger',
                        message: "빌드 실패: ${env.JOB_NAME} #$
                        MBER} by ${Author_ID}(${Author_Name})
                        Details>)",
                        endpoint: 'https://meeting.ssafy.com/
                        bydiywarbncefbufh',
                        channel: '#TEST'
                    )
                }
            }
            cleanWs()
        }
    }
}
```