포팅메뉴얼

📌 개발 환경

FrontEnd

- Node.js `20.15.0`
- TypeScript `5.5.3`
- vite `5.4.1`
- React `18.3.1`
  - Recoil `0.7.7`
  - react-query `5.59.0`
- axios `1.7.7`
- chart.js `4.4.4`
- styled-components `6.1.13`

BackEnd

- Java
  - Java OpenJDK `17.0.12`
  - Spring Boot `3.3.2`
    - Spring Data JPA `3.3.3`
    - Spring Data redis `3.3.3`
    - Spring Data mongodb `4.3.3`
    - Spring Data elasticsearch `5.3.3`
    - Spring Security `6.3.3`
    - OAuth2.0 `6.3.3`
    - Lombok `1.18.20`
  - JWT `0.12.3`
  - elasticsearch `8.15.1`
  - logstash `8.15.1`
  - kibana `8.15.1`
  - Gradle `8.10`
  - AWS S3 Bucket Cloud `2.2.6`
- Python
  - Python `3.12.3`
  - Django `5.1.1`
  - apscheduler `2.2.6`
  - requests `2.32.3`
  - beautifulsoup4 `4.12.3`
  - google-api-core `2.20.0`
  - fastapi `0.115.0`
  - SQLAlchemy `2.0.35`
  - Scikit-learn `1.5.2`
  - numpy `2.1.1`

UI/UX

- Figma

IDE

- IntelliJ `2024-01`
- Visual Studuio Code `1.94.1`
- Pycharm `2024-02-03`

Server 배포 환경

- AWS EC2 `ubuntu 20.04.6 LTS`
- Docker `27.2.0`
- Docker Compose `2.29.2`
- Nginx `1.18.0`
- SSL
- Docker Hub

CI/CD

- jenkins `2.475`

DB

- MySQL `8.0.38`
- redis `7.4.0`
- mongoDB `7.0.14`
- elasticsearch `8.15.1`
- AWS S3

Collaboration

형상관리

- GitLab

커뮤니케이션

- Mattermost
- Notion

이슈관리

- Jira

📌 환경 변수 설정

[FrontEnd]

⌨️ .env

```
VITE_SPEECH_API_KEY=${AZURE_SPEECH_API_KEY}
VITE_SPEECH_REGION=${AZURE_SPEECH_REGION}
VITE_BACKEND_URL="https://j11d105.p.ssafy.io/api"
```

[BackEnd]

⌨️ application.yml

```
spring:
  autoconfigure:
    exclude: org.springframework.boot.autoconfigure.data.elasticsearch.ReactiveElasticsearchRepositoriesAutoConfiguration
  application:
    name: backend
  servlet:
    multipart:
      max-request-size: 50MB
      max-file-size: 50MB
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://${HOSTNAME}:${PORT}/${DBNAME}?useSSL=false&serverTimezone=Asia/Seoul&characterEncoding=UTF-8
    username: {MYSQL_USERNAME}
    password: {MYSQL_PASSWORD}
    hikari:
      maximum-pool-size: 50
      minimum-idle: 10
      idle-timeout: 30000
      max-lifetime: 600000
      connection-timeout: 20000
      register-mbeans: true
  jpa:
    properties:
      hibernate.format_sql: true
      dialect: org.hibernate.dialect.MySQL8InnoDBDialect

  data:
    redis:
      host: ${HOST}
      port: ${PORT}
      password: ${REDIS_PASSWORD}
      username: ${REDIS_USERNAME}
    mongodb:
      host: ${HOST}
      port: ${PORT}
```

```
      username: ${MONGO_USERNAME}
      password: ${MONGO_PASSWORD}
      authentication-database: admin
      database: ${DBNAME}
      # uri: mongodb: //${HOST}:${PORT}/${DBNAME}

  security:
    oauth2:
      client:
        registration:
          kakao:
            client-id: ${KAKAO_CLIENT_ID}
            client-secret: ${KAKAO_CLIENT_SECRET}
            redirect-uri: ${KAKAO_REDIRECT_URL}
            authorization-grant-type: authorization_code
            client-authentication-method: client_secret_post
            client-name: Kakao
            scope:
              - profile_nickname
              - account_email
          naver:
            client-id: ${NAVER_CLIENT_ID}
            client-secret: ${NAVER_CLIENT_SECRET}
            redirect-uri: ${NAVER_REDIRECT_URL}
            authorization-grant-type: authorization_code
            client-name: Naver
        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

logging:
  level:
    org.springframework.security: DEBUG
    org.springframework.web: DEBUG
    org.springframework.security.oauth2: DEBUG
    org.springframework.web.servlet: DEBUG
    org.springframework.web.client.RestTemplate: DEBUG

frontend:
  url: ${DOMAIN}

oauth2:
  baseUrl: ${DOMAIN}

cors:
  allowed-origin: http://192.168.100.167:8080, http://192.168.100.167:5173, http://192.168.100.166:3000, http://192.168.100.
  allowed-methods: '*'

jwt:
  secret: ${JWT_SECRET:ZZIRrOGPAAtiMOXANWzDUrc90lurrc5dlg7O1hSJnYg=}
  accessToken-expiration: ${JWT_ACCESS_TOKEN_EXPIRATION:604800000} # 7일
  refreshToken-expiration: ${JWT_REFRESH_TOKEN_EXPIRATION:1209600000} # 1,209,600,000 ms = 14일
  oauth-sign-up-expiration: ${JWT_REFRESH_TOKEN_EXPIRATION:600000} # 10분

cloud:
  aws:
    s3:
      bucket: nonakim
    credentials:
      access-key: AKIAQEIP3C7IIFCCTNEG
      secret-key: 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
    region:
      static: ap-northeast-2
      auto: false
    stack:
      auto: false
fast-api:
  base:
    url: http://${IP}:8003
```

📄 .env (Django)

```
GEN_AI_SECRET_KEY=${GEMINI_AI_SECRET_KEY}
CRAWLING_USER_AGENT='Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0
OPEN_AI_API_KEY=${GEMINI_AI_API_KEY}

REDIS_HOST=${IP_ADDR}
REDIS_USERNAME=${REDIS_USERNAME}
REDIS_PASSWORD=${REDIS_PASSWORD}

MONGO_HOST=${IP_ADDR}
MONGO_PORT=${PORT}
MONGO_USERNAME=${MONGO_USERNAME}
MONGO_PASSWORD=${MONGO_PASSWORD}

S3_ACCESS_KEY=${AWS_S3_ACCESS_KEY}
S3_SECRET_KEY=${AWS_S3_SECRET_KEY}
S3_REGION=ap-northeast-2

MYSQL_USER=${MYSQL_USER}
MYSQL_PWD=${MYSQL_PASSWORD}
MYSQL_HOST=${IP_ADDR}
MYSQL_DB_NAME=${MYSQL_DBNAME}
```

📄 .env (fastAPI)

```
MYSQL_USERNAME=${MYSQL_USER}
MYSQL_PASSWORD=${MYSQL_PASSWORD}
MYSQL_HOST=${DOMAIN}
MYSQL_PORT=${PORT}
MYSQL_DBNAME=${MYSQL_DBNAME}

MONGO_USERNAME=${MONGO_USERNAME}
MONGO_PASSWORD=${MONGO_PASSWORD}
MONGO_HOST=${IP_ADDR}
MONGO_PORT=${PORT}
MONGO_DBNAME=${MONGO_DBNAME}
MONGO_TABLE_NAME=${TABLE_NAME}
```

📌 배포 환경 설정

0. 초기 세팅

1. EC2 접속

```
# sudo ssh -i [pem키 위치] [접속 계정]@[접속할 도메인]
$ sudo ssh -i J11D105T.pem ubuntu@j11d105.p.ssafy.io
```

2. Docker & Docker Engine 설치
3. Docker Compose 설치

1. Docker 컨테이너 생성

: 백엔드 Spring서버, 크롤링 Django서버, 추천 fastAPI, 프론트엔드 react
mysql, mongodb, redis, elasticsearch, logstash, kibana, jenkins, sonarqube

- `docker ps` 결과

```
ubuntu@ip-172-26-8-1:~$ docker ps
CONTAINER ID   IMAGE                                  NAMES
f3e533bc62bb   seryoii/newlearn-react:latest          newlearn-5173
8e9f7a808d45   nahyun1616/newlearn-recommend-image:latest   recommend-container
037acbff0c09   nahyun1616/newlearn-spring:latest       newlearn-8080
f239b612519e   yechanissm2/crawling-image:latest      crawling-image-container
82a09031f03e   sonarqube:lts-community                sonarqube-container
dc7e13d0ef6c   docker.elastic.co/kibana/kibana:8.15.1  kibana
48bcf92c273c   logstash:8.15.1                        logstash
eae38a6ccad6   docker.elastic.co/elasticsearch/elasticsearch:8.15.1   elasticsearch
64631ee10966   mongo:latest                           mongodb-container
a65f29f4487f   jenkins/jenkins                        jenkins-container
87ceec0b82dc   redis                                  redis-container
5297a33b67d8   mysql:8.0.38                           mysql-container

CREATED          STATUS          PORTS
29 minutes ago   Up 29 minutes   0.0.0.0:5173->5173/tcp, :::5173->5173/tcp
About an hour ago   Up About an hour   0.0.0.0:8003->8003/tcp, :::8003->8003/tcp
About an hour ago   Up About an hour   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp
2 hours ago      Up 2 hours
8 hours ago      Up 8 hours      0.0.0.0:9006->9000/tcp, [::]:9006->9000/tc
2 days ago       Up 2 days       0.0.0.0:5601->5601/tcp, :::5601->5601/tcp
2 days ago       Up 25 hours     0.0.0.0:5044->5044/tcp, :::5044->5044/tcp,
2 days ago       Up 2 days       0.0.0.0:9200->9200/tcp, :::9200->9200/tcp,
3 weeks ago      Up 2 days       0.0.0.0:27017->27017/tcp, :::27017->27017/
4 weeks ago      Up 2 days       50000/tcp, 0.0.0.0:9090->8080/tcp, [::]:90
4 weeks ago      Up 2 days       0.0.0.0:6379->6379/tcp, :::6379->6379/tcp
4 weeks ago      Up 2 days       0.0.0.0:3306->3306/tcp, :::3306->3306/tcp,
```

`/home/ubuntu/ Dockerfiles` 경로에 docker-compose파일 모아둠

```
$tree ./Dockerfiles
./Dockerfiles
├── 📁jenkins
│   └── docker-compose.yml
├── 📁mongodb
│   └── docker-compose.yml
├── 📁mysql
│   └── docker-compose.yml
├── 📁recomm
│   └── docker-compose.yml
└── 📁redis
    └── docker-compose.yml
```

`/home/ubuntu/ elk` 경로에 ElasticSearch 관련 파일 모아둠

```
$tree ./elk
./elk
├── docker-compose.elk.yml
├── 📁elasticsearch
│   └── 📁data
│       └── ...
└── 📁logstash
    ├── Dockerfile
    └── 📁pipeline
        └── logstash.conf
```

MySQL

볼륨 생성 $ `docker volume create mysql-volume`

```
services:
  mysql:
    image: mysql:8.0.38
    container_name: mysql-container
    restart : always
    ports:
      - "3306:3306"
    volumes:
      - /mysql-volume:/var/lib/mysql
    environment:
      MYSQL_DATABASE: ${MYSQL_DBNAME}
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      TZ: "Asia/Seoul"
```

Redis

볼륨 생성 $ `docker volume create redis-volume`

```
services:
  redis:
    image: redis
    container_name: redis-container
    ports:
      - "6379:6379"
    command: redis-server --requirepass ${REDIS_PASSWORD}
    volumes:
      - /redis-volume:/data
    restart: on-failure
```

mongodb

볼륨 생성 $ `docker volume create mongodb-volume`

```
services:
  mongodb:
    image: mongo:latest
    container_name: mongodb-container
    restart: always
    environment:
      - TZ=Asia/Seoul
      - MONGO_INITDB_ROOT_USERNAME= ${MONGO_ROOT_USERNAME}
      - MONGO_INITDB_ROOT_PASSWORD= ${MONGO_ROOT_PASSWORD}
    command : [--auth]
    ports:
      - "27017:27017"
    volumes:
      - /mongodb-volume:/data/db
      - ./mongod.conf:/etc/mongod.conf
    entrypoint: ["mongod", "-f", "/etc/mongod.conf"]
    mem_limit: 900m
    ulimits:
      nproc: 64000
      nofile:
        soft: 64000
        hard: 64000
```

Jenkins

$ `cd /home/ubuntu && mkdir jenkins-backup`
$ `sudo chown 1000 /home/ubuntu/jenkins-backup`

```
services:
  jenkins:
    image: jenkins/jenkins
    container_name: jenkins-container
    ports:
      - "9090:8080"
    volumes:
      - /home/ubuntu/jenkins-backup:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
      - /usr/bin/docker:/usr/bin/docker
    environment:
      TZ: "Asia/Seoul"
```

- 젠킨스 컨테이너 비밀번호 확인
  `docker exec jenkins-container cat /var/jenkins_home/secrets/initialAdminPassword`

SonarQube

```
sudo docker run -d --restart=always \
-e TZ=Asia/Seoul \
--name sonarqube-container -p 9006:9000 \
sonarqube:lts-community
```

- 실행
  $ `docker compose up -d`
- 컨테이너 접속
  $ `sudo docker exec -it [컨테이너 이름] bash`

ElasticSearch / logstash / kibana

docker-compose.yml

```
services:
  elasticsearch:
    container_name: elasticsearch
    image: docker.elastic.co/elasticsearch/elasticsearch:8.15.1
```

```
    restart: always
    ports:
      - "9200:9200"
      - "9300:9300"
    environment:
      - ES_JAVA_OPTS=-Xms2g -Xmx2g
      - discovery.type=single-node
      - xpack.security.enabled=false
      - TZ=Asia/Seoul
    ulimits:
      memlock:
        soft: -1
        hard: -1
    volumes:
      - ./elasticsearch/data:/usr/share/elasticsearch/data
    networks:
      - elk_network

  logstash:
    container_name: logstash
    image: logstash:8.15.1
    build:
      context: ./logstash
    volumes:
      - ./logstash/pipeline/logstash.conf:/usr/share/logstash/pipeline/logstash.conf
      - ./logstash/.logstash_jdbc_last_run:/usr/share/logstash/.logstash_jdbc_last_run
    ports:
      - "5044:5044"
    environment:
      - ELASTICSEARCH_HOSTS=http://j11d105.p.ssafy.io:9200
      - xpack.monitoring.collection.enabled=true
      - TZ=Asia/Seoul
    depends_on:
      - elasticsearch
    networks:
      - elk_network

  kibana:
    container_name: kibana
    image: docker.elastic.co/kibana/kibana:8.15.1
    restart: always
    ports:
      - "5601:5601"
    environment:
      - ELASTICSEARCH_HOSTS=http://j11d105.p.ssafy.io:9200
      - xpack.security.enabled=false
      - TZ=Asia/Seoul
    depends_on:
      - elasticsearch
    networks:
      - elk_network

networks:
  elk_network:
    driver: bridge
```

2. Nginx 설치 + SSL 인증키 발급

1. Nginx 설치

   $ `sudo apt update && sudo apt upgrade`

   $ `sudo apt install nginx`

   $ `sudo service nginx start`

2. Encrypt, Certbot 설치

   $ `sudo apt-get install letsencrypt`

   $ `sudo apt-get install certbot python3-certbot-nginx`

3. SSL 인증서 발급

```
# Certbot 동작 (nginx 중지하고 해야함)
$ sudo systemctl stop nginx

# Nginx 상태확인 & 80번 포트 확인
$ sudo service nginx status
$ netstat -na | grep '80.*LISTEN'
```

```
# SSL 인증서 발급 (인증서 적용 및 .pem 키 발급)
$ sudo certbot --nginx
$ sudo letsencrypt certonly --standalone -d j11d105.p.ssafy.io

# 설치한 인증서 확인 및 위치 확인
$ sudo certbot certificates

# nginx 설정 적용
# nginx 재시작
$ sudo service nginx restart
$ sudo systemctl reload nginx
```

Nginx conf 설정

- service-url.inc 파일 생성

  $ `sudo vim /etc/nginx/conf.d/service-url.inc`

```
set $service_url http://127.0.0.1:8080;
```

- nginx 설정파일

  $ `sudo vim /etc/nginx/sites-available/default`

  + https (ssl 키 적용) , `service-url.inc` 를 통한 무중단 배포 진행

```
server {

    index index.html index.htm index.nginx-debian.html;

    server_name j11d105.p.ssafy.io;
    include /etc/nginx/conf.d/service-url.inc;

    listen 443 ssl; # managed by Certbot
    listen [::]:443 ssl ipv6only=on;

  # frontend
    location / {
        proxy_pass http://j11d105.p.ssafy.io:5173;
        proxy_set_header Host $host:$http_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_cookie_path / "/";

    }

    # backend - spring
    location /api {
        proxy_pass $service_url;

        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-Host $server_name;
        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 Cookie $http_cookie;
        proxy_pass_header Set-Cookie;
        proxy_cookie_path / "/; HttpOnly; Secure; SameSite=None";

        proxy_redirect off;
    }

    location ~ ^/(oauth2|login/oauth2) {
        proxy_pass $service_url;
        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;
    }

    ssl_certificate /etc/letsencrypt/live/j11d105.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/j11d105.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 ($host = j11d105.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

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

        server_name j11d105.p.ssafy.io;
    return 404; # managed by Certbot


}
```

- S3에 파일 업로드 시 용량 제한 늘리기

  $ `sudo vim /etc/nginx/nginx.conf`

  해결하기 위해 http block 에 아래의 옵션 추가

```
client_max_body_size 50M;
```

- nginx 재시작 : 파일 수정 사항 적용

  $ `sudo systemctl restart nginx`

- Nginx 로그 확인

  $ `cd /var/log/nginx`

  ㄴ access.log, error.log 존재

3. Jenkins 설정

젠킨스 파이프라인 스크립트

: 특정 브랜치(backend-develop, frontend-develop)를 추적하여 자동 배포가 진행하도록 한다.

post{} 는 mattermost 알림을 위한 설정

▼ 백엔드

```
pipeline {
    agent any
    stages {
        stage('Git Clone'){
            steps {
                git branch: 'backend-develop',
                credentialsId: 'gitlab',
                url: 'https://lab.ssafy.com/s11-bigdata-recom-sub1/S11P21D105.git'
            }
            post {
                failure {
                  echo 'Repository clone failure !'
                }
                success {
                  echo 'Repository clone success !'
                }
            }
        }
        stage('application.yml download') {
            steps {
                withCredentials([file(credentialsId: 'application.yml', variable: 'applicationymlFile')]) {
                    script {
                        sh 'rm /var/jenkins_home/workspace/newlearn-spring/backend/src/main/resources/application.yml'
                        sh 'cp $applicationymlFile /var/jenkins_home/workspace/newlearn-spring/backend/src/main/resources
                    }
                }
            }
        }
        stage('BE-Build'){
            steps {
                dir('/var/jenkins_home/workspace/newlearn-spring/backend/'){
                    sh 'pwd'
                    sh 'ls -al'
                    sh 'chmod +x ./gradlew'
                    sh 'chmod +x ./gradlew.bat'
                    sh 'java --version'
                    sh './gradlew clean build '
                }
            }
```

```
        }

        stage('Docker Hub Login'){
            steps{
                withCredentials([usernamePassword(credentialsId: 'DOCKERHUB_USER', passwordVariable: 'DOCKER_PASSWORD', u
                    sh 'echo "$DOCKER_PASSWORD" | docker login -u $DOCKER_USERNAME --password-stdin'
                }
            }
        }
        stage('Docker Build and Push') {
            steps {
                withCredentials([usernamePassword(credentialsId: 'DOCKER_REPO', passwordVariable: 'DOCKER_PROJECT', usern
                    sh 'cd ./backend && docker build -f Dockerfile -t $DOCKER_USER/$DOCKER_PROJECT .'
                    sh 'cd ./backend && docker push $DOCKER_USER/$DOCKER_PROJECT'
                    echo 'docker push Success!!'
                }
                echo 'docker push Success!!'
            }
        }

        stage('BE Deploy to EC2') {
            steps {
                //백엔드 이미지 땡겨오고 배포
                sshagent(credentials: ['ssh-key']) {
                  withCredentials([string(credentialsId: 'EC2_SERVER_IP', variable: 'IP')]) {
                     sh 'ssh -o StrictHostKeyChecking=no ubuntu@$IP "sudo sh deploy.sh"'
                  }
                }
            }
        }
    }

    post {
        always {
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
                def Commit_Message = sh(script: "git log -1 --pretty=%B", returnStdout: true).trim()
                def Build_Status = currentBuild.result ?: 'SUCCESS'
                def Status_Color = Build_Status == 'SUCCESS' ? 'good' : (Build_Status == 'UNSTABLE' ? 'warning' : 'danger
                def Status_Text = Build_Status == 'SUCCESS' ? '빌드 성공' : (Build_Status == 'UNSTABLE' ? '빌드 불안정' : '빌

                def branchName = sh(script: "git rev-parse --abbrev-ref HEAD", returnStdout: true).trim() //$env.GIT_BRAN


                //def allCommits = sh(script: "git log --pretty=format:'%h - %s (%an)' $env.GIT_PREVIOUS_SUCCESSFUL_COMMI
                def previousCommit = env.GIT_PREVIOUS_SUCCESSFUL_COMMIT ?: 'HEAD~1' // 이전 커밋이 없으면 HEAD~1으로 설정
                def allCommits = sh(script: "git log --pretty=format:'%h - %s (%an)' $previousCommit..HEAD", returnStdout
                def formattedCommits = allCommits.split('\\n').collect { line ->
                    def escapedLine = line.replaceAll("([\\[\\]\\(\\)])", '\\\\$1')
                    "• ${escapedLine}"
                }.join('\\n')

                def message = """
                    #### 💻BE $Status_Text
                    **빌드 번호** $env.JOB_NAME #$env.BUILD_NUMBER
                    **브랜치:** $branchName
                    **작성자:** $Author_ID ($Author_Name)
                    **빌드 URL:** [Details]($env.BUILD_URL)
                    **포함된 커밋:**
                    $formattedCommits
                """.stripIndent()
                mattermostSend(
                    color: Status_Color,
                    message: message,
                    endpoint: 'https://meeting.ssafy.com/hooks/45o43om36fnkpnfym1jmfz4z8o',
                    channel: 'd105notification'
                )
            }
        }
    }
}
```

▼ 프론트엔드

```
pipeline {
    agent any
```

```
    environment {
        PATH = "/usr/local/bin:/usr/bin:$PATH"
    }
    stages {
        stage('Git Clone') {
            steps {
                git branch: 'frontend-develop',
                credentialsId: 'gitlab',
                url: 'https://lab.ssafy.com/s11-bigdata-recom-sub1/S11P21D105.git'
            }
        }
        stage('.env download') {
            steps {
                withCredentials([file(credentialsId: 'REACT_ENV', variable: 'ENV_FILE')]) {
                    script {
                        sh 'cp $ENV_FILE /var/jenkins_home/workspace/newlearn-react/frontend/'
                        sh 'grep VITE_FIREBASE_PROJECT_ID /var/jenkins_home/workspace/newlearn-react/frontend/.env || ech
                    }
                }
            }
        }
        stage('FE-Build') {
            steps {
                dir('/var/jenkins_home/workspace/newlearn-react/frontend/') {
                    sh 'npm install'
                    sh 'npm run build'
                }
            }
        }
        stage('Docker Hub Login'){
            steps{
                withCredentials([usernamePassword(credentialsId: 'DOCKER_FE_USER', passwordVariable: 'DOCKER_PASSWORD', u
                    sh 'echo "$DOCKER_PASSWORD" | docker login -u $DOCKER_USERNAME --password-stdin'
                }
            }
        }
        stage('Docker Build and Push') {
            steps {
                withCredentials([usernamePassword(credentialsId: 'DOCKER_FE_REPO', passwordVariable: 'DOCKER_PROJECT', us
                    sh 'set -o allexport; . $ENV_FILE; set +o allexport'
                    sh 'cd ./frontend && docker build -f Dockerfile -t $DOCKER_USER/$DOCKER_PROJECT .'
                    sh 'cd ./frontend && docker push $DOCKER_USER/$DOCKER_PROJECT'
                }
                echo 'docker push Success!!'
            }
        }
        stage('FE Deploy to EC2') {
            steps {
                sshagent(credentials: ['ssh-key']) {
                  withCredentials([string(credentialsId: 'EC2_SERVER_IP', variable: 'IP')]) {
                      sh 'ssh -o StrictHostKeyChecking=no ubuntu@$IP "sudo sh deploy-frontend.sh"'
                  }
                }
            }
        }
    }

    post {
        always {
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
                def Commit_Message = sh(script: "git log -1 --pretty=%B", returnStdout: true).trim()
                def Build_Status = currentBuild.result ?: 'SUCCESS'
                def Status_Color = Build_Status == 'SUCCESS' ? 'good' : (Build_Status == 'UNSTABLE' ? 'warning' : 'danger
                def Status_Text = Build_Status == 'SUCCESS' ? '빌드 성공' : (Build_Status == 'UNSTABLE' ? '빌드 불안정' : '빌

                def branchName = sh(script: "git rev-parse --abbrev-ref HEAD", returnStdout: true).trim() //$env.GIT_BRAN


                //def allCommits = sh(script: "git log --pretty=format:'%h - %s (%an)' $env.GIT_PREVIOUS_SUCCESSFUL_COMMI
                def previousCommit = env.GIT_PREVIOUS_SUCCESSFUL_COMMIT ?: 'HEAD~1' // 이전 커밋이 없으면 HEAD~1으로 설정
                def allCommits = sh(script: "git log --pretty=format:'%h - %s (%an)' $previousCommit..HEAD", returnStdout
                def formattedCommits = allCommits.split('\\n').collect { line ->
                    def escapedLine = line.replaceAll("([\\[\\]\\(\\)])", '\\\\$1')
                    "• ${escapedLine}"
                }.join('\\n')
```

```
                def message = """
                    #### 🎨FE $Status_Text
                    **빌드 번호** $env.JOB_NAME #$env.BUILD_NUMBER
                    **브랜치:** $branchName
                    **작성자:** $Author_ID ($Author_Name)
                    **빌드 URL:** [Details]($env.BUILD_URL)
                    **포함된 커밋:**
                    $formattedCommits
                """.stripIndent()
                mattermostSend(
                    color: Status_Color,
                    message: message,
                    endpoint: 'https://meeting.ssafy.com/hooks/45o43om36fnkpnfym1jmfz4z8o',
                    channel: 'd105notification'
                )
            }
        }
    }
}
```

▼ **sonarQube**

```
pipeline {
   agent any

   environment {
        SONAR_PROJECT_KEY = credentials('sonarQubeProjectKey')
        SONAR_HOST_URL = 'http://j11d105.p.ssafy.io:9006'
        SONAR_LOGIN = credentials('sonarQubeLogin')
    }

   stages{
       stage('SCM') {
           steps {
               git branch: "backend-develop",
               credentialsId: 'gitlab',
               // (Gitlab URL)
               url: 'https://lab.ssafy.com/s11-bigdata-recom-sub1/S11P21D105.git'
           }
       }
      stage('Build') {
            steps {
                        dir('/var/jenkins_home/workspace/newlearn-spring/backend/'){
                        // 프로젝트 권한 변경
                        sh 'chmod +x ./gradlew'
                        // 프로젝트 빌드
                        sh './gradlew clean build'
               }
           }
        }
       stage('SonarQube analysis') {
           steps {
                withSonarQubeEnv('sonarQubeToken') {
                    sh '''
                    cd ./backend && chmod +x ./gradlew && ./gradlew sonarqube \
                        -Dsonar.projectKey=$SONAR_PROJECT_KEY \
                        -Dsonar.host.url=$SONAR_HOST_URL \
                        -Dsonar.login=$SONAR_LOGIN
                    '''
               }
            }
       }
   }
}
```

🔑Credential 관리

빌드에 필요한 env 파일들을 저장해두고 배포 시 파일을 옮겨 서버에 올린다.

Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	gitlab_token	GitLab API token
		System	(global)	gitlab	selene0106@naver.com/******
		System	(global)	ssh-key	ssh-key
		System	(global)	DOCKERHUB_USER	hyunstu16@gmail.com/******
		System	(global)	DOCKER_REPO	nahyun1616/******
		System	(global)	EC2_SERVER_IP	EC2_SERVER_IP
		System	(global)	DOCKER_FE_USER	hhhky9900@gmail.com/******
		System	(global)	DOCKER_FE_REPO	seryoii/******
		System	(global)	REACT_ENV	.env
		System	(global)	sonarQubeToken	sonarQubeToken
		System	(global)	sonarQubeLogin	sonarQubeLogin
		System	(global)	sonarQubeProjectKey	sonarQubeProjectKey
		System	(global)	application.yml	application.yml (application.yml)

- **GitLab** : gitlab의 프로젝트를 clone 해오기위한 credential
  - `gitlab_token` : gitlab API 토큰
  - `gitlab` : gitlab ID/PW
- `ssh-key` : jenkins에서 우리의 aws ec2의 ssh에 접속하기위한 credential
- **EC2 Server IP** : Pipeline에서 EC2 Server IP 감추기 위해
  - `EC2_SERVER_IP` : 서버주소
- **Docker Hub** : dockerhub에 있는 이미지를 끌어오기 위함
  - `DOCKER_USER`, `DOCKER_FE_USER` : 도커허브 아이디 / 비밀번호
  - `DOCKER_REPO`, `DOCKER_FE_REPO` : 도커허브 nameSpace / 도커허브 RepositoryName
- **백엔드, 프론트엔드 설정파일들**

  프로젝트 최종 배포시 중요한 정보들이 들어있는 Spring, React 설정 파일들을 gitlab에 올리지않기 때문에 Jenkins에 미리 저장 해두고 파이프라인 속 build 전 단계에 가져오기 위함
  - `REACT_ENV` : 프론트엔드 env파일
  - `application.yml` : 백엔드 SpringBoot yml파일
- **SonarQube 관련**
  - `sonarQubeToken`, `sonarQubeLogin`, `sonarQubeProjectKey`

Gitlab 웹훅 설정

- 백엔드 ; backend-develop 브랜치
- 프론트 ; frontend-develop 브랜치
- sonarqube ; backend-develop 브랜치 → spring 코드 정적 분석

젠킨스 플러그인 추가 설치

- GitLab
- SSH Agent
- Pipeline Graph View
- Mattermost Notification
- SonarQube Scanner
- react

4. 배포 위한 파일 생성

[Backend - Spring]

1. **SpringBoot Dockerfile 생성**

   Dockerfile

```
# open jdk 17 버전의 환경 구성
FROM openjdk:17-alpine

# tzdata 패키지 설치 및 타임존 설정
RUN ln -snf /usr/share/zoneinfo/Asia/Seoul /etc/localtime && echo Asia/Seoul > /etc/timezone

# build가 되는 시점에 JAR_FILE 경로에 jar파일 생성
ARG JAR_FILE=/build/libs/backend-0.0.1-SNAPSHOT.jar

COPY ${JAR_FILE} /newlearnspring.jar

# 운영 및 개발에서 사용되는 환경 설정을 분리
ENTRYPOINT ["java","-jar","-Dspring.profiles.active=prod", "-Duser.timezone=Asia/Seoul", "/newlearnspring.jar"]
```

2. **DockerHub에 올린 이미지를 가져와 docker compose로 서버 띄우기**

$ `vi /home/ubuntu/docker-compose.newlearn8080.yml`

$ `vi /home/ubuntu/docker-compose.newlearn8081.yml`

docker-compose.newlearn8080.yml

```
services:
  api:
    image: nahyun1616/newlearn-spring:latest
    container_name: newlearn-8080
    environment:
      - TZ=Asia/Seoul
      - LANG=ko_KR.UTF-8
      - HTTP_PORT=8080
    ports:
      - '8080:8080'
```

docker-compose.newlearn8081.yml

```
services:
  api:
    image: nahyun1616/newlearn-spring:latest
    container_name: newlearn-8081
    environment:
      - TZ=Asia/Seoul
      - LANG=ko_KR.UTF-8
      - HTTP_PORT=8081
    ports:
      - '8081:8080'
```

3. **BLUE/GREEN 무중단 배포 script 작성**

$ `vi /home/ubuntu/deploy.sh`

deploy.sh

: EC2환경에서 배포하기 위한 스크립트

```
DOCKER_APP_NAME=newlearn
# 0
# 이미지 갱신
sudo docker compose -p ${DOCKER_APP_NAME}-8080 -f /home/ubuntu/docker-compose.${DOCKER_APP_NAME}8080.yml pull
sudo docker compose -p ${DOCKER_APP_NAME}-8081 -f /home/ubuntu/docker-compose.${DOCKER_APP_NAME}8081.yml pull

# 1 현재 떠 있는 컨테이너 체크
EXIST_8080=$(sudo docker compose -p ${DOCKER_APP_NAME}-8080 -f /home/ubuntu/docker-compose.${DOCKER_APP_NAME}8080.yml ps
EXIST_8081=$(sudo docker compose -p ${DOCKER_APP_NAME}-8081 -f /home/ubuntu/docker-compose.${DOCKER_APP_NAME}8081.yml ps

# 2 컨테이너 스위칭
if [ -n "$EXIST_8081" ]; then
    echo "8080 컨테이너 실행"
    sudo docker compose -p ${DOCKER_APP_NAME}-8080 -f /home/ubuntu/docker-compose.${DOCKER_APP_NAME}8080.yml up -d --forc
    BEFORE_COLOR="8081"
    AFTER_COLOR="8080"
    BEFORE_PORT=8081
    AFTER_PORT=8080
else
    echo "8081 컨테이너 실행"
    sudo docker compose -p ${DOCKER_APP_NAME}-8081 -f /home/ubuntu/docker-compose.${DOCKER_APP_NAME}8081.yml up -d --forc
    BEFORE_COLOR="8080"
    AFTER_COLOR="8081"
    BEFORE_PORT=8080
    AFTER_PORT=8081
fi

# 3 서버 상태 체크
SERVER_OK=false
for cnt in `seq 1 10`; do
    echo "서버 응답 확인 : (${cnt}/10)"
    UP=$(curl -s http://127.0.0.1:${AFTER_PORT}/api/server-check)
    if [ "${UP}" = "OK" ]; then
        SERVER_OK=true
        break
    fi
    sleep 10
done


if [ "$SERVER_OK" = true ]; then
    echo "${AFTER_COLOR} server up(port:${AFTER_PORT})"

    # 4 nginx 설정 변경사항 reload
    sudo sed -i "s/${BEFORE_PORT}/${AFTER_PORT}/" /etc/nginx/conf.d/service-url.inc
    sudo nginx -s reload
    echo "Nginx reload"

    # 5 새로운 컨테이너가 제대로 떴는지 재확인
    EXIST_AFTER=$(docker compose -p ${DOCKER_APP_NAME}-${AFTER_PORT} -f docker-compose.${DOCKER_APP_NAME}${AFTER_COLOR}.y
    if [ -n "$EXIST_AFTER" ]; then
        # 6 이전 컨테이너 종료
        echo "$BEFORE_COLOR server down(port:${BEFORE_PORT})"
        docker compose -p ${DOCKER_APP_NAME}-${BEFORE_PORT} -f docker-compose.${DOCKER_APP_NAME}${BEFORE_COLOR}.yml down
```

```
        # 7 사용되지 않는 이미지 삭제
        sudo docker image prune -f
    else
        echo "새 컨테이너 실행 실패. 이전 상태로 롤백합니다."
        docker compose -p ${DOCKER_APP_NAME}-${AFTER_PORT} -f docker-compose.${DOCKER_APP_NAME}${AFTER_COLOR}.yml down
    fi
else
    echo "서버에 문제가 있어요. 배포를 중단하고 이전 상태를 유지합니다."
    # 새로 시작한 컨테이너 종료
    docker compose -p ${DOCKER_APP_NAME}-${AFTER_PORT} -f docker-compose.${DOCKER_APP_NAME}${AFTER_COLOR}.yml down
fi
```

새로 배포한 버전에 이상이 없으면 새로운 컨테이너로 교체, 이상이 있으면 기존 컨테이너 유지함

[Backend - Django]

크롤링 서버

1. 🌫️ **Dockerfile**

```
# Python 3.12.6 slim 이미지 사용
FROM python:3.12.6-slim

# 작업 디렉토리 설정
WORKDIR /app

# 필요한 패키지 설치를 위한 APT 패키지 업데이트 및 필수 패키지 설치
RUN apt-get update && apt-get install -y \
    gcc \
    python3-dev \
    libpq-dev \
    dos2unix \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*

# 필요한 패키지 설치
COPY requirements.txt .
RUN pip install --upgrade pip
RUN pip install --no-cache-dir -r requirements.txt

# 소스 코드 및 .env 파일 복사
COPY . .

# Python 경로 설정 (Crawling 모듈을 찾을 수 있도록)
ENV PYTHONPATH=/app:$PYTHONPATH

# Django 환경 변수 설정
ENV DJANGO_SETTINGS_MODULE=Crawling.settings
ENV ENV_FILE=/app/.env

# entrypoint.sh 복사 및 실행 권한 부여
COPY entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh

# .env 파일의 줄바꿈 형식을 UNIX 스타일로 변환
RUN dos2unix /app/.env

# entrypoint 설정
ENTRYPOINT ["/entrypoint.sh"]

# 스케줄러 스크립트 실행
CMD ["python", "crawled_data/scheduler.py"]
```

2. 컨테이너 생성 및 실행

```
docker run -d --name crawling_container yechanissm2/crawling-image
```

[Backend - fastAPI]

추천

1. 🌫️ **Dockerfile**

```
# 기본 이미지로 Python 3.12 사용
FROM python:3.12.3-slim

# 로케일 설정을 위한 패키지 설치
RUN apt-get update && apt-get install -y locales
```

```
# 한국어 로케일 생성
RUN localedef -i ko_KR -c -f UTF-8 -A /usr/share/locale/locale.alias ko_KR.UTF-8

# 환경 변수 설정
ENV LANG ko_KR.UTF-8
ENV LC_ALL ko_KR.UTF-8

# 작업 디렉토리를 설정
WORKDIR .

# 애플리케이션 코드를 복사
COPY . .

# 시스템 패키지 업데이트 및 의존성 설치
RUN apt-get update && \
    pip install --no-cache-dir --upgrade pip && \
    pip install --no-cache-dir -r requirements.txt

EXPOSE 8003

# FastAPI 애플리케이션 실행
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8003"]
```

2. **docker-compose.yml 파일 생성**

```
services:
  newlearn-recommend:
    image: nahyun1616/newlearn-recommend-image:latest
    container_name: recommend-container
    restart: unless-stopped
    ports:
      - "8003:8003"
    env_file:
      - ./.env
    command: uvicorn main:app --host 0.0.0.0 --port 8003
```

[logstash]

Elastic Search 로그 수집 도구

1. **Dockerfile**

$ `vi /home/ubuntu/elk/logstash`

```
FROM docker.elastic.co/logstash/logstash:8.15.1

# MySQL JDBC 드라이버 버전 설정
ENV MYSQL_CONNECTOR_J_VERSION 8.0.30
ENV MYSQL_CONNECTOR_J_URL https://downloads.mysql.com/archives/get/p/3/file/mysql-connector-java-$MYSQL_CONNECTOR_J_VERSI

# 드라이버 다운로드 및 복사
RUN curl -L -O $MYSQL_CONNECTOR_J_URL && \
    tar -xvf mysql-connector-java-$MYSQL_CONNECTOR_J_VERSION.tar.gz && \
    cp mysql-connector-java-$MYSQL_CONNECTOR_J_VERSION/mysql-connector-java-$MYSQL_CONNECTOR_J_VERSION.jar /usr/share/log
    rm -rf mysql-connector-java-$MYSQL_CONNECTOR_J_VERSION.tar.gz mysql-connector-java-$MYSQL_CONNECTOR_J_VERSION

CMD ["logstash", "-f", "/usr/share/logstash/pipeline/logstash.conf"]
```

2. **logstash.conf**

$ `vi /home/ubuntu/elk/logstash/pipeline`

```
input {
  jdbc {
    jdbc_driver_library => "/usr/share/logstash/mysql-connector-java-8.0.30.jar"
    jdbc_driver_class => "com.mysql.cj.jdbc.Driver"
    jdbc_connection_string => "jdbc:mysql://${IP_ADDR}:${PORT}/${MYSQL_DBNAME}?useSSL=false&serverTimezone=Asia/Seoul&cha
    jdbc_user => ${MYSQL_USER}
    jdbc_password => ${MYSQL_PASSWORD}
    jdbc_paging_enabled => true
    use_column_value => true
    tracking_column => "news_id"
    tracking_column_type => "numeric"
    schedule => "*/5 * * * *"
    statement => "SELECT news_id, title, title_eng FROM news WHERE news_id > :sql_last_value ORDER BY news_id ASC"
    last_run_metadata_path => "/usr/share/logstash/.logstash_jdbc_last_run/.last_run"
  }
}
```

```
filter {
  mutate {
    copy => {"news_id" => "[@metadata][_id]"}
    remove_field => ["@version"]
  }
}
output {
  elasticsearch {
    hosts => ["http://j11d105.p.ssafy.io:9200"]
    index => "news"
    document_id => "%{[@metadata][_id]}"
  }
  elasticsearch {
    hosts => ["http://j11d105.p.ssafy.io:9200"]
    index => "news_aggregation"
    document_id => "%{[@metadata][_id]}"
  }
}
```

[Frontend]

1. **Dockerfile 생성**

   ⌨️ **Dockerfile (프론트엔드 프로젝트 내부)**

```
# Node.js 20 버전 이미지 기반 새로운 이미지 생성
FROM node:20

# 컨테이너 내 작업할 디렉토리 설정
WORKDIR /app

# package.json, package-lock.json 컨테이너에 복사
COPY package*.json ./

RUN rm -rf node_modules

# 의존성 설치
RUN npm ci

# 나머지 파일 컨테이너에 복사
COPY . .

# 빌드 실행
RUN npm run build

CMD ["npm", "run", "start"]
```

2. **docker-compose.yml 파일 생성**

   $ `vi /home/ubuntu/docker-compose.newlearn5173.yml`

   ⌨️ **docker-compose.newlearn5173.yml**

```
services:
   api:
     image: seryoii/newlearn-react:latest
     container_name: newlearn-5173
     environment:
         - TZ=Asia/Seoul
         - LANG=ko_KR.UTF-8
         - HTTP_PORT=5173
     ports:
         - '5173:5173'
     command: npm run dev
```

3. **배포 script 작성**

   $ `vi /home/ubuntu/deploy-frontend.sh`

   ⌨️ **deploy.sh**

```
sudo docker compose -p newlearn-5173 -f /home/ubuntu/docker-compose.newlearn5173.yml pull
sudo docker compose -p newlearn-5173 -f /home/ubuntu/docker-compose.newlearn5173.yml up -d --force-recreate
sudo docker image prune -f
```

5. 참고: EC2내 파일구조

```
$ pwd
/home/ubuntu
$ tree
.
├── 📁 Dockerfiles
├── deploy-frontend.sh
├── deploy.sh
├── docker-compose.newlearn5173.yml
├── docker-compose.newlearn8080.yml
├── docker-compose.newlearn8081.yml
├── 📁 elk
└── 📁 jenkins-backup
```