RunnerWay 포팅 메뉴얼

1. 개요
    1.1. 프로젝트 개요
    1.2. 프로젝트 사용 도구
    1.3. 개발 환경
    1.4. 기술 스택
    1.5. 추가 기술 스택
2. 빌드
    2.1. 프론트엔드 빌드 방법
    2.2. 백엔드 빌드 방법
    2.3. 추천 알고리즘 빌드 방법
    2.4. 배포하기
3. 프로젝트에서 사용하는 외부 서비스 정보를 정리한 문서
    3.1. Kakao Login API
    3.2. Google Map API
    3.3. S3 Bucket

1. 개요

1.1. 프로젝트 개요

- 프로젝트 명
    - RunnerWay
- 프로젝트 소개
    - 러닝 코스 추천 기반 기록 관리 및 대결 서비스
- 주요 기능
    1. 랭커와 대결
    2. 러닝 코스 추천
    3. 러닝 기록 관리
    4. 랭킹 시스템

1.2. 프로젝트 사용 도구

- 이슈 관리
  - JIRA
- 형상 관리
  - Gitlab
- 커뮤니케이션
  - Notion
  - Mattermost
- 디자인
  - Figma
- DB 설계
  - ERD Cloud
- 동영상 편집
  - 모바비

1.3. 개발 환경

- 프론트엔드
  - Android Studio
- 백엔드
  - IntelliJ IDEA
- 인프라
  - gitbash
- DB
  - MySQL Workbench

1.4. 기술 스택

- 프론트엔드
  - Andriod Studio 2024.1.2
  - Dart 3.5.3
  - Dio 5.7.0
  - Dotenv 5.1.0
  - envied 0.5.4+1
  - Flutter 3.24.3
  - Get 4.6.6
  - SecureStorage 9.2.2
- 백엔드
  - H3 3. 7. 3
  - Java 17
  - Jwt 0.11.5
  - Spring-Boot 3.3.3
  - Spring Data JPA 3.3.3
  - Spring Security 6.3.3
- DB
  - Elasticsearch 7.17.0
  - MySQL 9.0.1
  - Redis 7.4.0
- 인프라
  - AWS EC2
  - AWS S3
  - Docker
  - Jenkins
  - NginX

1.5. 추가 기술 스택

- 추천 알고리즘 관련
  - FastAPI 0.115.0
  - lightFM 1. 17
  - Python 3.9

2. 빌드

2.1. 프론트엔드 빌드 방법

1. 버전
   a. Flutter 3.24.3
2. 라이브러리 설치
   a. pubspec.yaml 파일에 아래 설정 추가

```
environment:
  sdk: ^3.5.2

dependencies:
  flutter:
    sdk: flutter
  get: ^4.6.6
  baseflow_plugin_template: ^2.2.0
  location: ^7.0.0
  geolocator: ^12.0.0
  google_maps_flutter: ^2.9.0
  flutter_polyline_points: ^2.1.0
  flutter_dotenv: ^5.1.0
  envied: ^0.5.4+1
  dio: 5.7.0
  image_picker: ^0.8.7+3
  intl: ^0.18.1
  flutter_secure_storage: ^9.2.2
  connectivity_plus: ^6.0.5
  app_settings: ^5.1.1
```

```
  table_calendar: ^3.0.9
  jwt_decoder: ^2.0.1
  path_provider: ^2.1.2
  loading_animation_widget: ^1.3.0
  flowery_tts: ^1.2.0
  audioplayers: ^6.1.0
  cached_network_image: ^3.2.3
  cupertino_icons: ^1.0.8
  kakao_flutter_sdk_user: ^1.9.5
  flutter_native_splash: ^2.4.1

dev_dependencies:
  flutter_test:
    sdk: flutter
  build_runner: ^2.4.12
  envied_generator: ^0.5.4+1
  flutter_lints: ^4.0.0

flutter:
  assets:
    - assets/images/
    - assets/icons/
    - assets/logo/
    - assets/images/main/
    - assets/images/temp/
    - assets/images/auth/
    - assets/images/medals/
    - .env

  fonts:
    - family: notosans
      fonts:
        - asset: assets/fonts/notosans/NotoSans-Thin.tt
          weight: 100
        - asset: assets/fonts/notosans/NotoSans-ExtraLi
          weight: 200
        - asset: assets/fonts/notosans/NotoSans-Light.t
          weight: 300
```

```
        - asset: assets/fonts/notosans/NotoSans-Regular
          weight: 400
        - asset: assets/fonts/notosans/NotoSans-Medium.
          weight: 500
        - asset: assets/fonts/notosans/NotoSans-SemiBol
          weight: 600
        - asset: assets/fonts/notosans/NotoSans-Bold.tt
          weight: 700
        - asset: assets/fonts/notosans/NotoSans-ExtraBo
          weight: 800
        - asset: assets/fonts/notosans/NotoSans-Black.t
          weight: 900
    - family: playball
      fonts:
        - asset: assets/fonts/playball/Playball-Regular
    - family: MyFlutterApp
      fonts:
        - asset: assets/fonts/MyFlutterApp.ttf
```

b. pubget 실행

```
flutter pub get
```

3. 빌드

a. build 생성을 통한 env.g.dart 파일 생성

```
flutter pub run build_runner build
```

b. keytool을 통한 앱 서명

```
keytool -genkey -v -keystore runnerway.jks -keyalg RSA
```

c. apk 추출(안드로이드)

```
flutter build apk --release --target-platform=android-a
```

d. 파일명 수정 후 배포

2.2. 백엔드 빌드 방법

1. 버전
    a. JAVA Open-JDK 17
    b. SpringBoot 3.3.3
    c. Gradle 8.10
2. 빌드

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.3.3'
    id 'io.spring.dependency-management' version '1.1.6'
}

group = 'chuchu'
version = '0.0.1-SNAPSHOT'

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(17)
    }
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-s
    implementation 'org.springframework.boot:spring-boot-s
```

```
    implementation 'org.springframework.boot:spring-boot-s
    implementation 'org.springframework.boot:spring-boot-s

    //Swagger
    implementation 'org.springdoc:springdoc-openapi-starte

    //JWT
    implementation 'io.jsonwebtoken:jjwt-api:0.11.5'
    implementation 'io.jsonwebtoken:jjwt-impl:0.11.5'
    implementation 'io.jsonwebtoken:jjwt-jackson:0.11.5'

    //ModelMapper
    implementation group: 'org.modelmapper', name: 'modelm

    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring-boot-
    runtimeOnly 'com.mysql:mysql-connector-j'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-bo
    testImplementation 'org.springframework.security:sprin
    testRuntimeOnly 'org.junit.platform:junit-platform-lau

    // elasticsearch
    implementation 'org.springframework.boot:spring-boot-s
    implementation 'org.elasticsearch.client:elasticsearch

    // redis
    implementation 'org.springframework.boot:spring-boot-s

    // 날짜/시간 파싱
    implementation 'com.fasterxml.jackson.datatype:jackson

    // mapstruct
    implementation 'org.mapstruct:mapstruct:1.5.3.Final'
    annotationProcessor 'org.mapstruct:mapstruct-processor

    // webclient
    implementation 'org.springframework.boot:spring-boot-s
```

```
    // h3
    implementation group: 'com.uber', name: 'h3', version:
    implementation group: 'org.json', name: 'json', versio
}

tasks.named('test') {
    useJUnitPlatform()
}
```

```
./gradlew build
```

3. Properties

```
server:
  servlet:
    context-path: /api
spring:
  profiles:
    include: key

  datasource:
    url: jdbc:mysql://${serverhost}:${dbport}/runnerway
    username: ${mysqlusername}
    password: ${mysqlpassword}
    driver-class-name: com.mysql.cj.jdbc.Driver
    hikari:
      maximum-pool-size: 10

  jpa:
    hibernate:
      ddl-auto: none
    properties:
      hibernate:
        diarect: org.hibernate.dialect.MysqlDialect
        jdbc:
          batch_size: 1000
        show_sql: false
```

```
        format_sql: false
        use_sql_comments: false
        order_updates: true
        default_batch_fetch_size: 16

    open-in-view: false

  data:
    redis:
      host: ${serverhost}
      port: ${redisport}
      password: ${redispassword}
  cache:
    type: redis

springdoc:
  swagger-ui:
    path: ${swagger-url}

logging:
  level:
    org.hibernate.SQL: info
    org.springframework.data.elasticsearch: DEBUG

jwt:
  expiration_time: 3600000 #1시간
  secret: ${JWT}

kakao:
  grant-type: authorization_code
  client-id: ${kakao-client-id}
  redirect-uri: ${kakao-redirect-uri}

elasticsearch:
  username: ${elastic-user-name}
  password: ${elastic-password}
```

2.3. 추천 알고리즘 빌드 방법

1. 버전
    a. Python 3.9
2. 빌드

```
uvicorn app.main:app --reload
```

3. requirements.txt

```
affine==2.4.0
aiohappyeyeballs==2.4.0
aiohttp==3.10.5
aiosignal==1.3.1
alabaster==0.7.16
annotated-types==0.7.0
anyio==4.6.0
argon2-cffi==23.1.0
argon2-cffi-bindings==21.2.0
arrow==1.3.0
astroid==3.3.4
asttokens==2.4.1
async-lru==2.0.4
async-timeout==4.0.3
asyncssh==2.17.0
atomicwrites==1.4.1
attrs==24.2.0
autopep8==2.0.4
babel==2.16.0
backports.tarfile==1.2.0
beautifulsoup4==4.12.3
binaryornot==0.4.4
black==24.8.0
bleach==6.1.0
blinker==1.8.2
Brotli==1.1.0
```

```
category-encoders==2.6.3
certifi==2024.8.30
cffi==1.17.1
chardet==5.2.0
charset-normalizer==3.3.2
click==8.1.7
click-plugins==1.1.1
cligj==0.7.2
cloudpickle==3.0.0
colorama==0.4.6
comm==0.2.2
ConfigArgParse==1.7
contourpy==1.3.0
cookiecutter==2.6.0
cornac==1.18.0
cryptography==43.0.1
cycler==0.12.1
debugpy==1.8.5
decorator==5.1.1
defusedxml==0.7.1
Deprecated==1.2.14
diff-match-patch==20230430
dill==0.3.8
docstring-to-markdown==0.15
docutils==0.21.2
dump==0.0.5
exceptiongroup==1.2.2
executing==2.1.0
fastapi==0.115.0
fastjsonschema==2.20.0
filelock==3.16.1
flake8==7.1.1
Flask==3.0.3
Flask-Cors==5.0.0
Flask-Login==0.6.3
fonttools==4.54.0
fqdn==1.5.1
frozenlist==1.4.1
```

```
fsspec==2024.9.0
future==1.0.0
gevent==24.2.1
geventhttpclient==2.3.1
greenlet==3.1.1
h11==0.14.0
h3==3.7.7
httpcore==1.0.5
httpx==0.27.2
huggingface-hub==0.25.1
hyperopt==0.2.7
hypothesis==6.112.1
idna==3.10
imagesize==1.4.1
importlib_metadata==8.5.0
importlib_resources==6.4.5
inflection==0.5.1
intervaltree==3.1.0
ipykernel==6.29.5
ipython==8.18.1
isoduration==20.11.0
isort==5.13.2
itsdangerous==2.2.0
jaraco.classes==3.4.0
jaraco.context==6.0.1
jaraco.functools==4.0.2
jedi==0.19.1
jellyfish==1.1.0
Jinja2==3.1.4
joblib==1.4.2
json5==0.9.25
jsonpointer==3.0.0
jsonschema==4.23.0
jsonschema-specifications==2023.12.1
jupyter-events==0.10.0
jupyter-lsp==2.2.5
jupyter_client==8.6.3
jupyter_core==5.7.2
```

```
jupyter_server==2.14.2
jupyter_server_terminals==0.5.3
jupyterlab==4.2.5
jupyterlab_pygments==0.3.0
jupyterlab_server==2.27.3
keyring==25.4.1
kiwisolver==1.4.7
lightfm==1.17
lightgbm==4.5.0
llvmlite==0.43.0
locust==2.31.6
markdown-it-py==3.0.0
MarkupSafe==2.1.5
matplotlib==3.9.2
matplotlib-inline==0.1.7
mccabe==0.7.0
mdurl==0.1.2
memory-profiler==0.61.0
mistune==3.0.2
more-itertools==10.5.0
mpmath==1.3.0
msgpack==1.1.0
multidict==6.1.0
multimethod==1.10
mypy-extensions==1.0.0
nbclient==0.10.0
nbconvert==7.16.4
nbformat==5.10.4
nest-asyncio==1.6.0
networkx==3.2.1
nltk==3.9.1
notebook==7.2.2
notebook_shim==0.2.4
numba==0.60.0
numpy==2.0.2
numpydoc==1.8.0
overrides==7.7.0
packaging==24.1
```

```
pandas==2.2.3
pandera==0.20.4
pandocfilters==1.5.1
parso==0.8.4
pathspec==0.12.1
patsy==0.5.6
pexpect==4.9.0
pickleshare==0.7.5
pillow==10.4.0
platformdirs==4.3.6
pluggy==1.5.0
powerlaw==1.5
prettytable==3.11.0
prometheus_client==0.21.0
prompt_toolkit==3.0.47
psutil==6.0.0
ptyprocess==0.7.0
pure_eval==0.2.3
py4j==0.10.9.7
pycodestyle==2.12.1
pycparser==2.22
pydantic==2.9.2
pydantic_core==2.23.4
pydocstyle==6.3.0
pyflakes==3.2.0
PyGithub==2.4.0
Pygments==2.18.0
PyJWT==2.9.0
pylint==3.3.0
pylint-venv==3.0.3
pyls-spyder==0.4.0
PyMySQL==1.1.1
PyNaCl==1.5.0
pyparsing==3.1.4
pyproj==3.6.1
PyQt5==5.15.11
PyQt5-Qt5==5.15.2
PyQt5_sip==12.15.0
```

```
PyQtWebEngine==5.15.7
PyQtWebEngine-Qt5==5.15.2
python-dateutil==2.9.0.post0
python-dotenv==1.0.1
python-json-logger==2.0.7
python-lsp-black==2.0.0
python-lsp-jsonrpc==1.1.2
python-lsp-server==1.12.0
python-slugify==8.0.4
pytoolconfig==1.3.1
pytz==2024.2
pyuca==1.2
PyYAML==6.0.2
pyzmq==26.2.0
QDarkStyle==3.2.3
qstylizer==0.2.3
QtAwesome==1.3.1
qtconsole==5.6.0
QtPy==2.4.1
rasterio==1.4.0
recommenders==1.2.0
referencing==0.35.1
regex==2024.9.11
requests==2.32.3
retrying==1.3.4
rfc3339-validator==0.1.4
rfc3986-validator==0.1.1
rich==13.8.1
rope==1.13.0
rpds-py==0.20.0
Rtree==1.3.0
safetensors==0.4.5
scikit-learn==1.5.2
scikit-surprise==1.1.4
scipy==1.13.1
seaborn==0.13.2
Send2Trash==1.8.3
six==1.16.0
```

```
sniffio==1.3.1
snowballstemmer==2.2.0
sortedcontainers==2.4.0
soupsieve==2.6
Sphinx==7.4.7
sphinxcontrib-applehelp==2.0.0
sphinxcontrib-devhelp==2.0.0
sphinxcontrib-htmlhelp==2.1.0
sphinxcontrib-jsmath==1.0.1
sphinxcontrib-qthelp==2.0.0
sphinxcontrib-serializinghtml==2.0.0
spyder==6.0.1
spyder-kernels==3.0.0
SQLAlchemy==2.0.35
stack-data==0.6.3
starlette==0.38.6
statsmodels==0.14.3
superqt==0.6.7
tabulate==0.9.0
terminado==0.18.1
text-unidecode==1.3
textdistance==4.6.3
threadpoolctl==3.5.0
three-merge==0.1.1
tinycss2==1.3.0
tokenizers==0.19.1
tomli==2.0.1
tomlkit==0.13.2
tornado==6.4.1
tqdm==4.66.5
traitlets==5.14.3
transformers==4.44.2
typeguard==4.3.0
types-python-dateutil==2.9.0.20240906
typing==3.7.4.3
typing-inspect==0.9.0
typing_extensions==4.12.2
tzdata==2024.2
```

```
ujson==5.10.0
uri-template==1.3.0
urllib3==1.26.20
uvicorn==0.30.6
watchdog==5.0.2
wcwidth==0.2.13
webcolors==24.8.0
webencodings==0.5.1
websocket-client==1.8.0
Werkzeug==3.0.4
whatthepatch==1.0.6
wrapt==1.16.0
yapf==0.40.2
yarl==1.12.1
zipp==3.20.2
zope.event==5.0
zope.interface==7.0.3
boto3==1.35.29
```

2.4. 배포하기

1. 배포 환경
    a. AWS EC3 (Ubuntu 20.04.6)
    b. AWS S3
    c. Docker 27.2.0
    d. Jenkins 2.462.2
2. 배포 방법
    a. 설치
        i. Docker 설치

```
# apt update
sudo apt update

# 패키지 설치
```

```
$sudo apt-get install apt-transport-https ca-certi
ficates curl gnupg-agent software-properties-commo
n

# gpg 설치 및 도커 저장소 key 저장
sudo curl -fsSL https://download.docker.com/linux/
ubuntu/gpg | sudo apt-key add -

# 도커 다운로드 및 레포지토리에 추가
sudo add-apt-repository "deb [arch=amd64] https://
download.docker.com/linux/ubuntu $(lsb_release -c
s) stable"

# 도커 설치
apt-cache policy docker-ce
sudo apt install docker-ce
```

ii. jenkins 설치

```
# port 설정: 9090포트로 외부에서 접속
# volum mount 통해 로컬과 연결
# sock 통해 인증
docker run -d --name jenkins \
    -p 9090:8080 \
    -v /var/jenkins_home:/var/jenkins_home \
    -v /var/run/docker.sock:/var/run/docker.sock \
    -v /usr/bin/docker:/usr/bin/docker \
    -v /usr/local/bin/docker-compose:/usr/local/bin/
    -e TZ=Asia/Seoul \
    --privileged --user root jenkins/jenkins:lts-jdk
```

b. CI/CD 환경 구축

i. NginX를 위해 미리 네트워크 생성

```
docker network create runnerway
```

ii. Gitlab과 연결

1. Gitlab에서 access token 발급
2. Jenkins credentials에 access token 등록
    a. Username with password로 생성하여, 본인 Gitlab ID 토큰 입력
3. Jenkinsdp pipe line item 생성
4. Gitlab build trigger 설정
    a. push
5. Gitlab Web Hook 걸기
    a. project hook에서 jenkins project의 url 넣기
    b. token 넣기

iii. NginX 설정

1. /etc/nginx/templates/runnerway.conf.template

```
server {
        listen 80;
        server_name j11b304.p.ssafy.io;

        return 308 https://j11b304.p.ssafy.io$r
}

server {
        listen 443 ssl;
        server_name j11b304.p.ssafy.io;

        ssl_certificate /etc/letsencrypt/live/j
        ssl_certificate_key /etc/letsencrypt/li

location /api{
        charset utf-8;
        proxy_pass http://spring:8080;
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr
        proxy_set_header X-Forwarded-For $proxy
        proxy_set_header X-Forwarded-Proto $sch
        }
```

```
  location /{
          charset utf-8;
          proxy_pass http://fastapi:8000;
          proxy_set_header Host $http_host;
          proxy_set_header X-Real-IP $remote_addr
          proxy_set_header X-Forwarded-For $proxy
          proxy_set_header X-Forwarded-Proto $sch
          }
```

1. 443 port로 들어오면 url 바탕으로 proxy
    a. /api가 붙어 있는 경우, Spring Container의 8080 port로 이동
    b. 그 외, fastapi container의 8000 port로 이동
    c. docker 내부 dns 서버를 활용하여 spring, fastapi 컨테이너 찾기
    d. 인증서를 사용하여 https 설정

iv. 백엔드 CI/CD 구축

1. Pipe line 구축

```
pipeline {
    agent any

    environment {
        SPRING_IMAGE = 'spring' // Spring 백엔드 이
        COMPOSE_FILE = 'docker-compose.yml' // 도커
    }
    tools {
        gradle 'gradle' // Gradle 도구 사용
    }

    stages {
        stage('Checkout') {
            steps {
                // Git 리포지토리에서 'develop/BE' 브
                git branch: 'develop/BE',
                url: 'https://lab.ssafy.com/s11-b
```

```
                credentialsId: 'jsj046@naver.com'
            }
        }

        stage('application-key download') {
            steps {
                dir('BackEnd'){
                    withCredentials([file(credent
                        script {
                            sh 'cp $applicationKe
                            sh 'ls src/main/resou
                            sh 'cat src/main/reso
                        }
                    }
                }
            }
        }

        stage('Build Spring Boot') {
            steps {
                script {
                    dir('BackEnd') { // BackEnd ㄷ
                        sh 'chmod +x gradlew' //
                        sh './gradlew build -x te
                    }
                }
            }
        }

        stage('Stop and Remove Existing Container
            steps {
                script {
                    sh 'docker-compose -f ${COMPO
                }
            }
        }

        stage('Remove Existing Docker images') {
```

```
            steps {
                script {
                    sh "docker rmi -f \$(docker i
                }
            }
        }

        stage('Build Docker Image Spring') {
            steps {
                script {
                    dir('BackEnd'){
                        sh 'docker build -t ${SPR
                    }
                }
            }
        }

        stage('docker-compose up') {
            steps {
                script {
                    sh 'docker-compose -f ${COMPO
                }
            }
        }

        stage('Nginx Restart') {
            steps {
                script {
                    // Nginx 컨테이너 재시작
                    sh 'docker restart nginx'
                }
            }
        }
    }
}
```

1. push, merge 액션 발생
2. branch 이동

3. seceret config 파일 download
    a. Jenkins Credentials에 등록된 applicaion-key.yml
4. build
5. 생성된 build 파일의 jar 파일 image화
    a. docker file을 미리 설정
    b. docker file 코드

```
FROM openjdk:17-alpine
RUN apk add --no-cache tzdata
ARG JAR_FILE=/build/libs/runnerway-0.0.1-SN
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

6. 해당 image를 Build ⇒ 이미지 생성
7. docker-compose를 내리고, 생성된 image를 다시 실행

```
version: '3.8'

services:
  spring:
    container_name: spring
    image: spring
    environment:
      - TZ=Asia/Seoul
    privileged: true
    networks:
      - runnerway

networks:
  runnerway:
    external: true
```

    1. image 실행하여 container 생성
    2. network 포함하여 연결, 해당 container에 접근 가능하도록 함
8. NginX 재실행

v. 추천 알고리즘 CI/CD 구축

1. Pipe line 구축

```
pipeline {
    agent any

    environment {
        PYTHON_IMAGE = 'fastapi' // fastapi 이미지
        COMPOSE_FILE = 'docker-compose.yml' // 도커

    }


    stages {
        stage('Checkout') {
            steps {
                // Git 리포지토리에서 'develop/PY' 브
                git branch: 'develop/PY',
                url: 'https://lab.ssafy.com/s11-b
                credentialsId: 'jsj046@naver.com'

            }
        }

        stage('.env download') {
            steps {
                dir('lightfm-api'){
                    withCredentials([file(credent
                        script {
                            // .env 파일을 lightfm-
                            sh 'cp $envFile .env'
                            // 확인을 위해 .env 파일
                            sh 'ls -la'
                            sh 'cat .env'
                        }
                    }
                }
            }
        }
```

```
        stage('Stop and Remove Existing Container
            steps {
                dir('lightfm-api'){
                    script {
                        sh 'docker-compose -f ${C
                    }
                }
            }
        }

        stage('Remove Existing Docker images') {
            steps {
                script {
                    sh "docker rmi -f \$(docker i
                }
            }
        }

        stage('Build Docker Image fastapi') {
            steps {
                script {
                    dir('lightfm-api'){
                        sh 'docker build --no-cac
                    }
                }
            }
        }

        stage('docker-compose up') {
            steps {
                script {
                    dir('lightfm-api'){
                        sh 'docker-compose -f ${C
                    }
                }
            }
```

```
        }

        stage('Nginx Restart') {
            steps {
                script {
                    // Nginx 컨테이너 재시작
                    sh 'docker restart nginx'
                }
            }
        }
    }
}
```

1. push, merge 액션 발생
2. branch 이동
3. .env 파일 download
    a. Jenkins Credentials에 등록된 .env
4. build
5. docker file 통해 image 생성
    a. docker file 코드

```
# Python 3.9 이미지 사용
FROM python:3.9

# 작업 디렉토리 설정
WORKDIR /app

# requirements.txt 파일 복사
COPY requirements.txt .

# 의존성 설치
RUN pip install --no-cache-dir --upgrade -r

# 필요한 시스템 패키지 설치
RUN apt-get update && apt-get install -y li
```

```
# 애플리케이션 소스 코드 복사
COPY . .

# FastAPI 서버 실행
CMD ["uvicorn", "app.main:app", "--host", "
```

1. requirements.txt 통해 의존성 설치

6. docker-compse 통해 container 생성

```
version: '3.8'

services:
  fastapi:
    container_name: fastapi
    build:
      context: .
      dockerfile: Dockerfile
    env_file:
      - .env
    environment:
      - TZ=Asia/Seoul
    ports:
      - "8000:8000"
    networks:
      - runnerway

networks:
  runnerway:
    external: true
```

1. fastapi container 생성
2. network 연결

7. NginX 재실행

c. 포트 개방

i. 443 ⇒ https 접속

ii. 3306 ⇒ MySQL

iii. 6379 ⇒ Redis

iv. 8000 ⇒ 추천 알고리즘 Fast API 통신

v. 8080 ⇒ 백엔드 통신

vi. 9090 ⇒ Jenkins 접속

vii. 9200 ⇒ Elasticsearch

3. 프로젝트에서 사용하는 외부 서비스 정보를 정리한 문서

3.1. Kakao Login API

- 사용 목적: 소셜 로그인
- 환경 변수 설정

```
// Back
kakao:
  grant-type: authorization_code
  client-id: ${kakao-client-id}
  redirect-uri: ${kakao-redirect-uri}

// Front
KAKAO_NATIVE_APP_KEY=****
```

- 필요 정보
  - Kakao Developers 계정 생성
  - 애플리케이션 등록 후 앱 키 발급

3.2. Google Map API

- 사용 목적 : 지도
- 환경 변수 설정

```
GOOGLE_MAPS_API_KEY=****
```

- 필요 정보

- Google 개발자 계정 생성
- Google Maps API KEY 발급

3.3. S3 Bucket

- 사용 목적 : 사진, 파일 저장
- 환경 변수 설정

```
S3_NAME=****
S3_REGION=****
S3_ACCESS_KEY_ID=****
S3_SECRET_ACCESS_KEY=****
```

- 필요 정보
  - AWS 계정 생성
  - S3 Bucket 생성
  - 접근 키 및 비밀 키 발급