포팅 메뉴얼

```
프로젝트에서 사용할 개발 도구, 환경 설정, Convention등의 방법과 지침을
1. 개발 환경(IDE)
- 통합개발환경, 버전 관리(Git)의 설정과 Convention 에 대해 적습니다.
2. 개발 언어 및 프레임워크
- 개발 언어와 버전을 명시하고 필요한 경우 언어 벌전 관리 도구 혹은 사용
Dependency를 적습니다.
3. 데이터베이스
- 프로젝트에서 사용할 데이터 베이스의 종류와 버전, 포트번호, 접속방법,
4. 테스팅 도구
- 단위 테스트, 통합테스트, 기능 테스트 등의 테스트 도구와 프레임워크 혹
5. 빌드 및 배포 두구
- 빌드 도구(ex: Maven,Gradle, WebPack,Vite 등등)을 명시하고 빌드
적습니다.
6. 개발 가이드 라인
- 코드 스타일 가이드: BackEnd , FrontEnd 각각 사용 언어나 환경에 맞
정하거나 ESLint 혹은 IDE 설정을 동일하게 공유합니다.
```

1. 빌드

버전 정보

[Server]

- EC2: Ubuntu 20.04.6 LTS
- Nginx: v1.18.0
- Docker: v25.0.0

[Project]

Back

- Spring: v2.7.7
- Gradle: v8.5
- JVM: 11
- Openvidu: v2.29.0

Front

- Node: v20.11.0
- NPM: v10.4.0
- React: v18.2.0
- Vite: v5.0.8
- Zustand : v4.5.0
- vite-plugin-pwa : v0.17.5
- axios : v1.6.7
- Tailwind CSS: v3.4.1

AI

- python: 3.9.13

▼ requirements.txt

```
annotated-types==0.6.0
anyio==4.2.0
asgiref==3.7.2
aiomysql==0.2.0
attrs==23.2.0
boto3==1.34.34
botocore==1.34.34
certifi==2024.2.2
charset-normalizer==3.3.2
click==8.1.7
colorama==0.4.6
coloredlogs==15.0.1
Django==4.2.9
```

```
exceptiongroup==1.2.0
fastapi==0.109.2
filelock==3.13.1
flatbuffers==23.5.26
fsspec==2024.2.0
h11==0.14.0
httpcore==1.0.2
httpx==0.26.0
huggingface-hub==0.20.3
humanfriendly==10.0
idna==3.6
image==1.5.33
imageio==2.33.1
Jinja2==3.1.3
jmespath==1.0.1
jsonschema==4.21.1
jsonschema-specifications==2023.12.1
lazy_loader==0.3
llvmlite==0.41.1
MarkupSafe==2.1.5
mpmath==1.3.0
networkx==3.1
numba==0.58.1
numpy==1.24.4
onnxruntime==1.16.3
opencv-python-headless==4.9.0.80
packaging==23.2
pillow==10.2.0
platformdirs==4.2.0
pooch==1.8.0
protobuf==4.25.2
pydantic==2.6.1
pydantic_core==2.16.2
PyMatting==1.1.12
pyreadline3==3.4.1
python-dateutil==2.8.2
python-dotenv==1.0.1
python-multipart==0.0.7
```

```
PyYAML==6.0.1
referencing==0.33.0
regex==2023.12.25
rembg==2.0.54
requests==2.31.0
rpds-py==0.17.1
s3transfer==0.10.0
safetensors==0.4.2
scikit-image==0.21.0
scipy==1.10.1
six==1.16.0
sniffio==1.3.0
sqlparse==0.4.4
starlette==0.36.3
sympy==1.12
tifffile==2023.7.10
tokenizers==0.15.1
torch==2.2.0
tqdm==4.66.1
typing_extensions==4.9.0
tzdata==2023.4
urllib3==1.26.18
uvicorn==0.27.0.post1
```

[DB]

- MariaDB: v10.3.23
- MongoDB: v5.0.24

[IDE]

- IntelliJ:
- Visual Studio Code: v1.85.1

1. 환경변수 형태

▼ 예시

.env.local :

// FIREBASE RTDB 프론트 키

VUE_APP_FIREBASE_APIKEY=

VUE_APP_FIREBASE_AUTHDOMAIN=

VUE_APP_FIREBASE_DATABASEURL=

VUE_APP_FIREBASE_PROJECTID=

VUE_APP_FIREBASE_STORAGEBUCKET=

VUE_APP_FIREBASE_MESSAGING_SENDERID=

VUE_APP_FIREBASE_APPID=

VUE_APP_FIREBASE_MEASUREMENTID=

// YOUTUBE DATA API KEY

VUE_APP_YOUTUBE_API_KEY=

// 카카오 공유 서비스 키

VUE_APP_KAKAO_API_KEY=

.env

DB_USER=DB호스트 유저 정보

DB_PASS=DB 비밀번호

DB_URL= DB 주소

.application.yml

REDIS

redis:

database: 0

host: 호스트 주소

password: 비밀 번호

port: 포트 번호

마리아 DB(배포)

datasource:

url: 마리아 DB 주소

driver-class-name: org.mariadb.jdbc.Driver

username: 유저 이름

password: 유저 비밀번호

Security OAuth

security:

oauth2.client:

registration:

google:

clientId: 클라이언트 아이디

clientSecret: 비밀번호

scope:

- email

- profile

JWT 시크릿 키

jwt:

secret: 시크릿 키

카카오 ADMIN 키

key:

kakao:

admin: 카카오 페이 API 키

.serviceAccountKey

Google Cloud Platform 유료 회원이 지급 받을 수 있는 프로젝트 전권한 루트 key

▼ env

▼ back

~~~/resources/applicaion-env.properties

```
# database
SPRING_DATASOURCE_URL={{MARIADB_URL}}
SPRING_DATASOURCE_USERNAME={{MARIADB_USERNAME}}
SPRING_DATASOURCE_PASSWORD={{PASSWORD_}}
MONGODB_URI={{MONGODB_URI}}
MONGODB_USERNAME={{MONGODB_USERNAME}}
```
~~~

```
MONGODB_PASSWORD={{MONGODB_PASSWORD}}
AUTO_TYPE=create

# S3
## access key
cloud.aws.credentials.accessKey={{aws 엑세스키}}
## secret key
cloud.aws.credentials.secretKey={{aws 시크릿키}}
cloud.aws.s3.bucketName=gwwmbucket
cloud.aws.region.static=ap-northeast-2
#cloud formation ??? ???? ?? ??.
cloud.aws.stack.auto=false

# OAuth
KAKAO_CLIENT_ID={{카카오 클라이언트ID}}
GOOGLE_CLIENT_ID={{구글 클라이언트ID}}
GOOGLE_CLIENT_SECRET={{구글 클라이언트 시크릿키}}

OPENVIDU_SECRET=MY_SECRET
```

application.properties

```
spring.profiles.active=${ACTIVE}
spring.profiles.include=db,env
spring.mvc.pathmatch.matching-strategy=ant_path_match


spring.jpa.hibernate.ddl-auto=${AUTO_TYPE}
spring.jpa.generate-ddl=true
spring.jpa.show-sql=true

spring.jpa.database-platform=org.hibernate.dialect.My
spring.data.mongodb.uri=${MONGODB_URI}
spring.data.mongodb.username=${MONGODB_USERNAME}
spring.data.mongodb.password=${MONGODB_PASSWORD}

spring.jpa.hibernate.use-new-id-generator-mappings=tr
spring.servlet.multipart.max-file-size=5MB
```

```
spring.jackson.serialization.fail-on-empty-beans=fals

# database
spring.datasource.url=${SPRING_DATASOURCE_URL}
spring.datasource.driver-class-name=org.mariadb.jdbc.
spring.datasource.username=${SPRING_DATASOURCE_USERNA
spring.datasource.password=${SPRING_DATASOURCE_PASSWO

spring.session.store-type=jdbc
spring.session.jdbc.initialize-schema=always

# registration
spring.security.oauth2.client.registration.kakao.clie

spring.security.oauth2.client.registration.kakao.clie
spring.security.oauth2.client.registration.kakao.auth
spring.security.oauth2.client.registration.kakao.scop
spring.security.oauth2.client.registration.kakao.clie

spring.security.oauth2.client.registration.google.cli
spring.security.oauth2.client.registration.google.cli
spring.security.oauth2.client.registration.google.scc

# provider
spring.security.oauth2.client.provider.kakao.authoriz
spring.security.oauth2.client.provider.kakao.token-ur
spring.security.oauth2.client.provider.kakao.user-inf
spring.security.oauth2.client.provider.kakao.user-nam

# Spring log colors
spring.output.ansi.enabled=always

# basic url path
server.servlet.context-path=/gawm
server.servlet.session.cookie.path=/
# openvidu
server.ssl.enabled=false
```

```
OPENVIDU_URL=https://i10e203.p.ssafy.io:4443/
#OPENVIDU_URL=http://localhost:4443/

OPENVIDU_SECRET=${OPENVIDU_SECRET}

server.servlet.session.cookie.same-site=none
server.servlet.session.cookie.secure=true
server.servlet.session.cookie.name=userSessionId
```

application-prod.properties

```
auth-redirect-url={{프론트서버 URI, ex) https://i10e203
spring.security.oauth2.client.registration.kakao.redi
spring.security.oauth2.client.registration.google.red
spring.datasource.hikari.maximum-pool-size=10
```

▼ front

./GAWM_FRONT_SERVER/.env

```
# 옷 이미지 base URL
VITE_CLOTHES_BASE_URL = {{옷 이미지 저장하고 있는 서버 URL

# GAWM API 서버 주소
VITE_GAWM_API_URL = {{API 서버 URL}}
VITE_GAWM_AI_API_URL = {{AI 서버 URL}}
```

▼ ai

./GAWM_AI_SERVER/.env

```
OMNICOMMERS_MANAGEMENT_KEY={{옴니커머스 MANAGEMENT_API키
OMNICOMMERS_TAGGING_KEY={{옴니커머스 TAGGING_API키}}
OMNICOMMERS_URL=https://api.kr.omnicommerce.ai/2022-0
AWS_ACCESS_KEY_ID={{AWS_ACCESS_KEY_ID}}
AWS_SECRET_ACCESS_KEY={{ AWS_SECRET_ACCESS_KEY}}
AWS_BUCKET_NAME={{AWS_BUCKET_NAME}}
AWS_REGION_NAME={{AWS_REGION_NAME}}
```

```
MARIADB_HOST={{HOST}}
MARIADB_PORT={{PORT}}
MARIADB_USER={{유저명}}@{{호스트명}}
MARIADB_PASSWORD={{PASSWORD}}
MARIADB_DB={{DB명}}
```

2. 빌드하기

1) Front

```
npm i
npm run build
```

2) Back-spring

Gradle 실행

```
stage('BE-Build') {
            steps {
                dir("./gawm_web_server/gawm") {
                    sh "sudo chmod a+rx gradlew"
                    sh "sudo ./gradlew clean build -x test"
                }
            }
        }
```

Bootjar 실행

```
stage('Deploy') {
            steps {
                dir("./scripts") {
                    sh "sudo chmod a+rx run_api_server.sh"
                    sh "./run_api_server.sh"
                }
```

```
            }
        }
```

```
#!/bin/bash

sudo chmod a+rx ../gawm_web_server/gawm/build/libs/gawm-0.0
java -jar -Dspring.profiles.active=prod ../gawm_web_server/
```

3) Back-fastAPI

- 가상환경 설치

```
sudo apt-get update
sudo apt-get install python3.10
sudo apt-get install python3-pip
sudo apt-get install virtualenv
sudo virtualenv aienv
source aienv/bin/activate
pip install -r requirements.txt
(권한문제 뜰경우) sudo chown -R (유저명) /aienv 진행 후 다시 insta
python main.py
```

- 젠킨스 pipeline

```
stage('AI-Deploy') {
            steps {
                dir("./scripts") {
                    sh "sudo chmod a+rx run_ai_server.sh"
                    sh "nohup ./run_ai_server.sh &"
                }
            }
        }
```

3. 배포하기

NGINX 설정

```
server {
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";

        location /openvidu {
                proxy_pass http://localhost:4443;
        }

        location /gawm/back{
                proxy_pass http://localhost:8080;
        }

        location /gawm/ai{
                proxy_pass http://localhost:8000;
        }

        location /gawm/login{
                proxy_pass http://localhost:8080;
        }

        location /{
               alias /var/lib/jenkins/workspace/gawm/fro
nt/dist/;
               try_files $uri $uri/ /index.html;
        }

        root /var/lib/jenkins/workspace/gawm/front/dist;
        index index.html index.htm;

        listen 443 ssl; # managed by Certbot
        ssl_certificate /etc/letsencrypt/live/i10e203.p.
ssafy.io/fullchain.pem; # managed by Certbot
        ssl_certificate_key /etc/letsencrypt/live/i10e20
3.p.ssafy.io/privkey.pem; # managed by Certbot
}

server {
```

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

        root /var/lib/jenkins/workspace/gawm/front/dist;
        index index.html index.htm;

        location / {
                try_files $uri $uri/ /index.html;
        }

}

server {

    listen 80;
    server_name i10e203.p.ssafy.io;
    return 301 https://$host$request_uri;
}
```

젠킨스 pipeline

```
pipeline {

    agent any

    stages {
        stage('Git Clone') {
            steps {
                git branch: 'develop', credentialsId: 'ssaf

            }
        }
        stage('Kill Process') {
            steps {
                dir("./scripts") {
                    sh "sudo chmod a+rx kill_all_process.sh
```

```
                    sh "nohup ./kill_all_process.sh &"
                }
            }
        }
        stage('Remove-package-lock.json') {
            steps {
                dir("./front") {
                    sh "sudo rm -rf package-lock.json node_
                }
            }
        }
        stage('BE-Build') {
            steps {
                dir("./gawm_web_server/gawm") {
                    sh "sudo chmod u+w build/libs"
                    sh "sudo chmod a+rx gradlew"
                    sh "sudo ./gradlew clean build -x test"
                }
            }
        }
        stage('Deploy') {
            steps {
                dir("./scripts") {
                    sh "sudo chmod a+rx run_api_server.sh"
                    sh "nohup ./run_api_server.sh &"
                }
            }
        }
        stage('FE-Build') {
            steps {
                dir("./front") {
                    sh "sudo npm install"
                    sh "sudo npm run build"
                }
            }
        }
        stage('FE-Deploy') {
            steps {
```

```
                dir("./scripts") {
                    sh "sudo chmod a+rx run_react.sh"
                    sh "sudo ./run_react.sh"
                }
            }
        }
        stage('Remove-manifest.json') {
            steps {
                dir("./front") {
                    sh "sudo rm -rf ./public/manifest.json"
                }
            }
        }
        stage('Openvidu-dev-Deploy') {
            steps {
                dir("./scripts") {
                    sh "chmod a+rx run_openvidu_dev.sh"
                    sh "nohup ./run_openvidu_dev.sh &"
                }
            }
        }
        stage('AI-Deploy') {
            steps {
                dir("./scripts") {
                    sh "sudo chmod a+rx run_ai_server.sh"
                    sh "nohup ./run_ai_server.sh &"
                }
            }
        }
    }
}
```

/scripts/kill_all_process.sh

```
#!/bin/bash
# 4443 도커 프로세스 kill
# sudo docker container stop $(sudo docker container ls -q
```

```
# 8080번 포트에 해당하는 PID 찾기
PID_8080=$(sudo lsof -t -i:8080)

# 8080번 포트에 해당하는 프로세스가 있는지 확인
if [ -z "$PID_8080" ]; then
    echo "포트 8080을 사용하는 프로세스가 실행 중이지 않습니다."
else
    # 해당 프로세스 종료
    sudo kill -9 $PID_8080
    echo "포트 8080을 사용하는 프로세스를 종료했습니다."
fi

# 8000번 포트에 해당하는 PID 찾기
PID_8000=$(sudo lsof -t -i:8000)

# 8000번 포트에 해당하는 프로세스가 있는지 확인
if [ -z "$PID_8000" ]; then
    echo "포트 8000을 사용하는 프로세스가 실행 중이지 않습니다."
else
    # 해당 프로세스 종료
    sudo kill -9 $PID_8000
    echo "포트 8000을 사용하는 프로세스를 종료했습니다."
fi
```

/scripts/run_ai_server.sh

```
#!/bin/bash

cd ../gawm_ai_server
sudo aienv/bin/python main.py
```

/scripts/run_api_server.sh

```
#!/bin/bash

sudo chmod a+rx ../gawm_web_server/gawm/build/libs/gawm-0.0
sudo java -jar -Dspring.profiles.active=prod ../gawm_web_se
```

/scripts/run_openvidu_dev.sh

```
#!/bin/bash
sudo docker run -p 4443:4443 --rm -e OPENVIDU_SECRET=MY_SEC

cd /opt/openvidu
sudo ./openvidu start
```

/scripts/run_react.sh

```
#!/bin/bash

sudo cp -r /var/lib/jenkins/workspace/gawm/front/dist /var/
# sudo npm run dev
sudo systemctl restart nginx
```

4. 특이사항