# 11기 공통PJT 배포 메뉴얼 A702

| 상태 | 시작 전 |
| --- | --- |

# 온방 포팅 매뉴얼

## I. 기술 스택 & 개발 환경

### 사용 도구, 개발 도구

🔨 **사용 도구, 개발 도구**

- 사용 도구
  - 이슈 관리: JIRA
  - 형상 관리: GitLab
  - 커뮤니케이션: Mattermost, Notion, Google Docs
  - 코드 리뷰: GitLab Merge Requests
  - 디자인: Figma, Canva
  - UCC: Monavi
  - 배포: EC2, Docker
- 개발 도구
  - Visual Studio Code: 1.91.1
  - Intellij: 2024.1.4

### 개발 환경

🔧 **개발 환경**

- **Frontend**
  - Node.js: 20
  - npm: 10.8.2
  - React: 18.3.1
  - eslint: 8.57.0
  - zustand: 4.5.4
- **Backend**
  - JDK: OpenJDK 17 (openjdk-17-jdk)
  - SpringBoot: 3.3.1
  - Gradle: 8.3
- **WebRTC**
  - Openvidu: 2.30
- **Database**
  - MariaDB: 10.3.23
- **Caching**
  - redis: 7.4.0
- **Search & Analytics**
  - Elasticsearch: 7.6.2
  - Kibana: 7.6.2
- **Infra**
  - Server
    - AWS EC2 Ubuntu 20.04
    - Nginx: 1.27.0
  - Storage
    - Amazon S3
  - Content Delivery Network (CDN)
    - Amazon CloudFront
  - Containerization
    - Docker Engine - Community: 27.1.1
      - buildx: 0.16.1
      - compose: 2.29.1

## 외부 API

💡 **외부 API**

- Kakao Map API

## .gitignore

- Frontend

```
# Created by https://www.toptal.com/developers/gitignore/api/node,visualst
# Edit at https://www.toptal.com/developers/gitignore?templates=node,visua

### Node ###
# Logs
logs
*.log
npm-debug.log*
yarn-debug.log*
yarn-error.log*
lerna-debug.log*
.pnpm-debug.log*

# Diagnostic reports (https://nodejs.org/api/report.html)
report.[0-9]*.[0-9]*.[0-9]*.[0-9]*.json

# Runtime data
pids
*.pid
*.seed
*.pid.lock

# Directory for instrumented libs generated by jscoverage/JSCover
lib-cov

# Coverage directory used by tools like istanbul
coverage
*.lcov

# nyc test coverage
.nyc_output

# Grunt intermediate storage (https://gruntjs.com/creating-plugins#storing
.grunt

# Bower dependency directory (https://bower.io/)
bower_components
```

```
# node-waf configuration
.lock-wscript

# Compiled binary addons (https://nodejs.org/api/addons.html)
build/Release

# Dependency directories
node_modules/
jspm_packages/

# Snowpack dependency directory (https://snowpack.dev/)
web_modules/

# TypeScript cache
*.tsbuildinfo

# Optional npm cache directory
.npm

# Optional eslint cache
.eslintcache

# Optional stylelint cache
.stylelintcache

# Microbundle cache
.rpt2_cache/
.rts2_cache_cjs/
.rts2_cache_es/
.rts2_cache_umd/

# Optional REPL history
.node_repl_history

# Output of 'npm pack'
*.tgz

# Yarn Integrity file
.yarn-integrity

# dotenv environment variable files
.env
.env.development.local
.env.test.local
.env.production.local
.env.local

# parcel-bundler cache (https://parceljs.org/)
```

```
.cache
.parcel-cache

# Next.js build output
.next
out

# Nuxt.js build / generate output
.nuxt
dist

# Gatsby files
.cache/
# Comment in the public line in if your project uses Gatsby and not Next.j
# https://nextjs.org/blog/next-9-1#public-directory-support
# public

# vuepress build output
.vuepress/dist

# vuepress v2.x temp and cache directory
.temp

# Docusaurus cache and generated files
.docusaurus

# Serverless directories
.serverless/

# FuseBox cache
.fusebox/

# DynamoDB Local files
.dynamodb/

# TernJS port file
.tern-port

# Stores VSCode versions used for testing VSCode extensions
.vscode-test

# yarn v2
.yarn/cache
.yarn/unplugged
.yarn/build-state.yml
.yarn/install-state.gz
.pnp.*
```

```
### Node Patch ###
# Serverless Webpack directories
.webpack/

# Optional stylelint cache

# SvelteKit build / generate output
.svelte-kit

### react ###
.DS_*
**/*.backup.*
**/*.back.*

node_modules

*.sublime*

psd
thumb
sketch

### VisualStudioCode ###
.vscode/*
!.vscode/settings.json
!.vscode/tasks.json
!.vscode/launch.json
!.vscode/extensions.json
!.vscode/*.code-snippets

# Local History for Visual Studio Code
.history/

# Built Visual Studio Code Extensions
*.vsix

### VisualStudioCode Patch ###
# Ignore all local history of files
.history
.ionide

# End of https://www.toptal.com/developers/gitignore/api/node,visualstudio
```

- Backend

```
HELP.md
.gradle
build/
```

```
!gradle/wrapper/gradle-wrapper.jar
!**/src/main/**/build/
!**/src/test/**/build/

### STS ###
.apt_generated
.classpath
.factorypath
.project
.settings
.springBeans
.sts4-cache
bin/
!**/src/main/**/bin/
!**/src/test/**/bin/

### IntelliJ IDEA ###
.idea
*.iws
*.iml
*.ipr
out/
!**/src/main/**/out/
!**/src/test/**/out/

### NetBeans ###
/nbproject/private/
/nbbuild/
/dist/
/nbdist/
/.nb-gradle/

### VS Code ###
.vscode/

### macOS ###
# General
.DS_Store
.AppleDouble
.LSOverride

### macOS Patch ###
# iCloud generated files
*.icloud

### Windows ###
# Windows thumbnail cache files
Thumbs.db
```

```
Thumbs.db:encryptable
ehthumbs.db
ehthumbs_vista.db

# Dump file
*.stackdump

# Folder config file
[Dd]esktop.ini

# Recycle Bin used on file shares
$RECYCLE.BIN/

# Windows Installer files
*.cab
*.msi
*.msix
*.msm
*.msp

# Windows shortcuts
*.lnk

/logs
```

## 환경 변수

- Frontend (.env)

```
VITE_KAKAO_SDK_KEY={카카오맵 API 키}
VITE_API_URL={사이트 URL}:8887
```

- Backend (docker 환경변수)

```
ENV aws_accessKey={AWS 액세스키}
ENV aws_cloudfront={AWS Cloudfront DNS}
ENV aws_region={AWS region}
ENV aws_s3bucket={AWS S3 버킷 이름}
ENV aws_secretKey={AWS 시크릿키}
ENV aws_stack_auto=false
ENV DB_URL={MariaDB URL}
ENV DB_USER={MariaDB 유저명}
ENV DB_PASSWORD={MariaDB 패스워드}
ENV FFMPEG_PATH={ffmpeg 설치 경로(ubuntu의 경우 /usr/bin/ffmpeg}
ENV FFPROBE_PATH={ffmpeg 설치 경로(ubuntu의 경우 /usr/bin/ffprobe}
ENV JWT_SECRET_KEY={JWT 시크릿키}
ENV OPENVIDU_SECRET={Openvidu 비밀번호}
```

```
ENV OPENVIDU_URL=https://localhost:5443

ENV PORT=8887
ENV REDIS_HOST=redis
ENV REDIS_PASSWORD={redis 비밀번호}
ENV REDIS_PORT=6379
ENV S3_ROOMIMAGE_PATH={S3 내 매물 이미지 경로}
ENV VIDEO_RECORDINGS_PATH={서버 내 Openvidu 녹화 경로}
ENV S3_HLS_PATH={S3 내 HLS 저장 경로}
ENV S3_USERIMG_PATH={S3 내 유저 이미지 경로}

ENV ELASTIC_SEARCH_HOST=elasticsearch
ENV ELASTIC_SEARCH_PORT=9200
```

# II. 빌드 및 배포

## S3 및 Cloudfront 설정

- S3 버킷 구성
  - 일반 구성
    - AWS 리전 확인: `아시아 태평양(서울) ap-northeast-2`
    - 버킷 이름 설정
  - 이 버킷의 퍼빌릭 액세스 차단 설정: `모든 퍼블릭 액세스 차단`
- CloudFront
  - 원본
    - Origin domain: `만들어 둔 S3로 설정`
    - 원본 액세스: `원본 액세스 제어 설정(권장)`
  - 동작 설정
    - 자동으로 객체 압축: `Yes`
    - 뷰어
      - 뷰어 프로토콜 정책: `HTTP and HTTPS`
      - 허용된 HTTP 방법: `GET, HEAD`
      - 뷰어 엑세스 젱한: `No`
    - 캐시 키 및 원본 요청
      - `Cache policy and origin request policy`
      - 응답 헤더 정책: `CORS-With-Preflight`

## Openvidu 배포

- Openvidu 설치

```
# root 권한 얻기
sudo su

# openvidu 설치 권장 경로로 이동 및 설치
cd /opt
curl <https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_

# 경로 이동 및 환경설정
$ cd openvidu
$ nano .env
```

- Openvidu 환경설정 (/opt/openvidu/.env)
  - 아래 내용의 항목들을 찾아 수정

```
# OpenVidu configuration 수정
# ----------------------
# 도메인 또는 퍼블릭IP 주소
DOMAIN_OR_PUBLIC_IP={도메인 주소}

# 오픈비두 서버와 통신을 위한 시크릿
OPENVIDU_SECRET={openvidu 비밀번호}

# Certificate type
CERTIFICATE_TYPE=letsencrypt

# 인증서 타입이 letsencrypt일 경우 이메일 설정
LETSENCRYPT_EMAIL={인증서 받을 이메일}

# HTTP port
HTTP_PORT=5442

# HTTPS port(해당 포트를 통해 오픈비두 서버와 연결)
HTTPS_PORT=5443

# Openvidu 녹화 설정
OPENVIDU_RECORDING=true

# Openvidu 녹화 Debug 모드 설정
OPENVIDU_RECORDING_DEBUG=false

# Openvidu 녹화 저장 경로 지정
OPENVIDU_RECORDING_PATH=/opt/openvidu/recordings

# Openvidu 녹화 layout 경로 설정
OPENVIDU_RECORDING_CUSTOM_LAYOUT=/opt/openvidu/custom-layout
```

```
# openvidu 녹화 영상 공개 여부
OPENVIDU_RECORDING_PUBLIC_ACCESS=false

# 클라이언트 측에서 녹화 이벤트(recordingStarted, recordingStopped)를 수신할 사용지
OPENVIDU_RECORDING_NOTIFICATION=publisher_moderator

# 녹화가 자동으로 중지되고 (이와 관련된 세션이 종료됨) 하는 타임아웃 시간(초) 설정
OPENVIDU_RECORDING_AUTOSTOP_TIMEOUT=120

# 클라이언트에서 OpenVidu 서버로 전송되는 최대 비디오 대역폭, kbps 단위
OPENVIDU_STREAMS_VIDEO_MAX_RECV_BANDWIDTH=1000

# 클라이언트에서 OpenVidu 서버로 전송되는 최소 비디오 대역폭, kbps 단위.
OPENVIDU_STREAMS_VIDEO_MIN_RECV_BANDWIDTH=300

# OpenVidu 서버에서 클라이언트로 전송되는 최대 비디오 대역폭, kbps 단위.
OPENVIDU_STREAMS_VIDEO_MAX_SEND_BANDWIDTH=1000

# OpenVidu 서버에서 클라이언트로 전송되는 최소 비디오 대역폭, kbps 단위.
OPENVIDU_STREAMS_VIDEO_MIN_SEND_BANDWIDTH=300
```

- Openvidu 실행

```
# 오픈비두 실행
$ ./openvidu start
```

## 로컬 빌드

- Frontend

```
# 의존성 설치
npm install

# 프로젝트 빌드
npm run build
```

- Backend

```
# 프로젝트 빌드
./gradlew build # 또는 Gradle > Tasks > build > bootJar)

# 빌드된 jar 파일 실행
java -jar build/libs/{프로젝트명}.jar
```

## 배포 빌드

- Frontend Dockerfile

```
# 베이스 이미지로 Node.js 20-alpine을 사용
FROM node:20-alpine as builder

# 앱 디렉토리를 생성하고 작업 디렉토리로 설정
WORKDIR /app

# package.json과 package-lock.json을 복사 (가능한 경우)
COPY ./package*.json ./

# 의존성 설치
RUN npm install

# 앱 소스 복사
COPY . .

# 빌드
RUN npm run build

# nginx 설정
FROM nginx
EXPOSE 5442
# EXPOSE 5173 -> 지도 확인 됨
COPY ./nginx/default.conf /etc/nginx/conf.d/default.conf
COPY --from=builder /app/dist /usr/share/nginx/html
```

  - Nginx 설정 파일

```
server{
    listen 80;

    location / {
        root /usr/share/nginx/html;

        index index.html index.html;

        try_files $uri $uri/ /index.html;
    }
}
```

- Backend Dockerfile

```
FROM ubuntu:22.04

# 환경 변수 설정
ENV DEBIAN_FRONTEND=noninteractive

# 필수 패키지 설치
```

```
RUN apt-get update && apt-get install -y openjdk-17-jdk ffmpeg wget unzip

# Gradle 8.x 설치
RUN wget https://services.gradle.org/distributions/gradle-8.3-bin.zip -P /
    unzip /tmp/gradle-8.3-bin.zip -d /opt && \
    ln -s /opt/gradle-8.3/bin/gradle /usr/bin/gradle

# Gradle 버전 확인 (선택 사항)
RUN gradle -v

WORKDIR /app

COPY . .

RUN gradle build -x test

WORKDIR /app/build/libs

ENV aws_accessKey={AWS 액세스키}
ENV aws_cloudfront={AWS Cloudfront DNS}
ENV aws_region={AWS region}
ENV aws_s3bucket={AWS S3 버킷 이름}
ENV aws_secretKey={AWS 시크릿키}
ENV aws_stack_auto=false
ENV DB_URL={MariaDB URL}
ENV DB_USER={MariaDB 유저명}
ENV DB_PASSWORD={MariaDB 패스워드}
ENV FFMPEG_PATH={ffmpeg 설치 경로(ubuntu의 경우 /usr/bin/ffmpeg}
ENV FFPROBE_PATH={ffmpeg 설치 경로(ubuntu의 경우 /usr/bin/ffprobe}
ENV JWT_SECRET_KEY={JWT 시크릿키}
ENV OPENVIDU_SECRET={Openvidu 비밀번호}

ENV OPENVIDU_URL=https://localhost:5443

ENV PORT=8887
ENV REDIS_HOST=redis
ENV REDIS_PASSWORD={redis 비밀번호}
ENV REDIS_PORT=6379
ENV S3_ROOMIMAGE_PATH={S3 내 매물 이미지 경로}
ENV VIDEO_RECORDINGS_PATH={서버 내 Openvidu 녹화 경로}
ENV S3_HLS_PATH={S3 내 HLS 저장 경로}
ENV S3_USERIMG_PATH={S3 내 유저 이미지 경로}

ENV ELASTIC_SEARCH_HOST=elasticsearch
ENV ELASTIC_SEARCH_PORT=9200
```

```
EXPOSE 8887
ENTRYPOINT ["java", "-jar", "ownBang-0.0.1-SNAPSHOT.jar"]
```

- docker-compose.yml

```
version: '3.8'
services:
  back:
    build: ./BACKEND/ownBang
    environment :
      REDIS_HOST : redis
      REDIS_PORT : 6379
      REDIS_PASSWORD : {redis 비밀번호}
    ports:
      - "8887:8887"
    depends_on:
      - redis
      - elasticsearch
      - kibana

  redis:
    image: redis:latest
    command: redis-server --requirepass {redis 비밀번호}
    environment:
      REDIS_PASSWORD: {redis 비밀번호}
    ports:
      - "8886:6379"
    healthcheck:
      test: ["CMD", "redis-cli", "ping"]
      interval: 10s
      timeout: 5s
      retries: 1

  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.6.2
    container_name: elasticsearch
    environment:
      - discovery.type=single-node
    ports:
      - "9200:9200"  # Elasticsearch의 기본 포트
      - "9300:9300"

  kibana:
    image: docker.elastic.co/kibana/kibana:7.6.2
    container_name: kibana
    environment:
      - ELASTICSEARCH_HOSTS=http://elasticsearch:9200
    ports:
```

```
      - "5601:5601"  # Kibana의 기본 포트
    depends_on:
      - elasticsearch

  front:
    build: ./FRONT
    ports:
      - "5442:80"
```