LivOn(기업 연계) 프로젝트 포팅 매뉴얼

1. 개발 및 운영 환경 정보

1-1. 공통 환경

- **OS**
  - AWS EC2 Ubuntu (64bit Linux)
  - Docker 기반 컨테이너 운영 (호스트 OS 버전은 `/etc/os-release`로 확인 후 기록 권장)
- **JVM / Java**
  - 빌드용: `gradle:8.10.2-jdk17-alpine` (JDK 17)
  - 실행용: `eclipse-temurin:17-jre-alpine` (JRE 17)
  - → **Java 17** 환경을 전제로 함 (로컬 개발 시도 JDK 17 권장)
- **Web Server**
  - Docker 이미지: `nginx:1.27-alpine`
  - 역할:
    - HTTPS 종료(SSL Termination)
    - 프론트엔드(React) 정적 파일 reverse proxy
    - 백엔드(Spring Boot) API reverse proxy
    - WebSocket (STOMP) 프록시 (`/api/v1/ws/**`)
    - APK 다운로드 경로 제공 (`/download/`)
- **WAS**
  - 별도 Tomcat 설치 없음
  - **Spring Boot 내장 Tomcat (JAR 실행 방식)** 사용
  - Docker 컨테이너에서 `java -jar app.jar --spring.config.location=...` 로 기동
- **IDE (개발용 예시)**
  - 백엔드: IntelliJ IDEA, Eclipse 등 Java 17 지원 IDE

    - 프론트엔드: VSCode, WebStorm 등
    - 모바일: Android Studio (Gradle 기반 빌드 사용)
- **Infra/도구**
    - Docker CE, docker compose plugin
    - Jenkins `jenkins/jenkins:lts-jdk17` 컨테이너
    - GitLab(SSAFY: `lab.ssafy.com`) + Webhook 연동
    - Certbot (Let's Encrypt)로 SSL 인증서 발급 및 `/etc/letsencrypt` 공유

1-2. 주요 설정 값

- **Nginx 설정 파일 경로**
    - Git 저장소:
        - `LivOnInfra/nginx/nginx.dev.conf`
        - `LivOnInfra/nginx/nginx.prod.conf`
    - 컨테이너 내부:
        - `/etc/nginx/conf.d/default.conf` 로 COPY
- **SSL 인증서 경로 (컨테이너 기준)**
    - 호스트: `/etc/letsencrypt/...`
    - Nginx 컨테이너:
        - `v /etc/letsencrypt:/etc/letsencrypt:ro` 마운트
        - 설정 내:
            - `ssl_certificate /etc/letsencrypt/live/k13s406.p.ssafy.io/fullchain.pem;`
            - `ssl_certificate_key /etc/letsencrypt/live/k13s406.p.ssafy.io/privkey.pem;`
- **WAS 메모리 관련**
    - Dockerfile/compose 상에 별도 Xms/Xmx 설정 없음
    - 필요 시 예시:
        - `JAVA_TOOL_OPTIONS="-Xms512m -Xmx2048m"` 등을 `environment:`에 추가하여 조정

2. 빌드 환경 변수 및 구성

2-1. Build Tool

- **백엔드 (LivOnBack)**
  - Gradle 8.x (컨테이너: `gradle:8.10.2-jdk17-alpine`)
  - 빌드: `gradle clean bootJar --no-daemon`
  - 로컬 개발 시: `./gradlew clean bootJar`
- **프론트엔드 Web (LivOnFront/web)**
  - Node.js 20 (컨테이너: `node:20-alpine`)
  - 빌드:
    - `npm install`
    - `npm run build`
  - 런타임: `serve -s build -l 3000`
- **모바일 (LivOnFront/mobile)**
  - Gradle Wrapper (Android 프로젝트)
  - Jenkins에서 Android SDK를 설치 후:
    - Dev: `./gradlew clean assembleDebug`
    - Prod: `./gradlew clean assembleRelease`

2-2. 주요 환경 변수 (Environment Variables)

> 실제 민감 값들은 Git 저장소에 포함하지 않고, Jenkins Secret file/.env 로 관리합니다.

- **공통**
  - `JAVA_HOME` (로컬 개발용, JDK 17 설치 경로)
  - `SPRING_PROFILES_ACTIVE`
    - dev: `dev`
    - prod: `prod`
    - docker-compose에서 각 컨테이너에 설정
  - `GCP_KEY_FILE`
    - 예: `/app/keys/xxxx.json`

    - docker-compose의 `environment` 에서 설정 후, 해당 경로로 key 파일 COPY
- **Android 빌드(Jenkins 내부)**
    - `ANDROID_SDK_ROOT=/var/jenkins_home/android-sdk`
    - `ANDROID_HOME=/var/jenkins_home/android-sdk`
    - `PATH` 에 `cmdline-tools/latest/bin`, `platform-tools` 추가 (Jenkinsfile에 설정)
- **프론트엔드 Web (.env 내용 – Jenkins Secret file)**
    - dev (`frontend-env-dev`):
        - `REACT_APP_API_BASE_URL` (예: `https://k13s406.p.ssafy.io:8443/api/v1`)
        - `REACT_APP_SOCKET_URL` (예: `https://k13s406.p.ssafy.io/api/v1/ws/chat`)
        - `REACT_APP_LIVEKIT_URL` (예: `wss://ov.s406.site:443/`)
    - prod (`frontend-env-prod`):
        - 동일 키 이름, 도메인/포트만 운영용으로 변경
- **백엔드 (application.yml – Jenkins Secret file)**
    - `yml-dev`, `yml-prod` Credential 파일로 관리
    - DB, Redis, MongoDB, S3, LiveKit, GCP, MinIO 등 외부 연동 설정 포함

      → 포팅 시 **해당 값은 새 환경에 맞게 재발급/수정 필수**

2-3. 빌드 명령어 예시

- **로컬(또는 CI 내) 백엔드 빌드**

```
cd LivOnBack
./gradlew clean bootJar
```

- **로컬 프론트엔드 빌드**

```
cd LivOnFront/web
npm install
npm run build
```

- **모바일 APK 빌드 (로컬 예시)**

```
cd LivOnFront/mobile
./gradlew clean assembleDebug   # 또는 assembleRelease
```

3. 배포 시 특이사항

3-1. GitLab 소스 클론

1. 서버에서 Git 사용 가능 확인 (`git --version`)
2. 원하는 경로에서 프로젝트 클론:

```
git clone https://lab.ssafy.com/s13-final/S13P31S406.git
cd S13P31S406
```

3. 브랜치 전략
   - `dev` 브랜치: 개발/테스트 환경
   - `master` 브랜치: 운영 환경
   - Jenkinsfile은 루트 (`/Jenkinsfile`)에 위치

3-2. 서버 선행 셋업 (공통)

1. **Docker & docker compose 설치**
   - 공식 Docker 저장소 등록 후 설치 (문서에 명령어 상세) LivOn(기업_연계)
   - `docker --version` , `docker compose version` 확인
   - `ubuntu` 사용자에 docker 그룹 추가:

     ```
     sudo usermod -aG docker $USER
     logout / 재접속
     ```

2. **Docker 네트워크 생성**

```
docker network create livon-dev-net
docker network create livon-prod-net
```

3. **MySQL 컨테이너 (공용 DB)**

```
docker run -d \
  --name livon-mysql \
  -e MYSQL_ROOT_PASSWORD=<비밀번호> \
  -e MYSQL_DATABASE=livon_db \
  -p 3306:3306 \
  -v livon_mysql_data:/var/lib/mysql \
  --restart always \
  mysql:8.0

docker network connect livon-dev-net livon-mysql
docker network connect livon-prod-net livon-mysql
```

4. **Redis 컨테이너**

```
docker run -d \
  --name livon-redis \
  -p 6379:6379 \
  --restart always \
  redis:alpine \
  redis-server --requirepass <비밀번호>

docker network connect livon-dev-net livon-redis
docker network connect livon-prod-net livon-redis
```

5. **APK 공유 폴더 (모바일 빌드 결과 공유용)**

```
sudo mkdir -p /var/livon/downloads
# Jenkins 컨테이너 UID 확인 후
sudo chown -R 1000:1000 /var/livon/downloads
```

6. **SSL 인증서 (Let's Encrypt, Certbot)**
   - 80 포트 비우기 (기존 Nginx/컨테이너 down)
   - Certbot 설치 후:

     ```
     sudo certbot certonly --standalone -d k13s406.p.ssafy.io
     ```

   - `/etc/letsencrypt/live/<도메인>/fullchain.pem`, `privkey.pem` 생성 확인

3-3. Jenkins 설치 및 설정

1. **Jenkins 컨테이너 실행**

```
docker run -d \
  -p 8080:8080 \
  -p 50000:50000 \
  -v jenkins_home:/var/jenkins_home \
  -v /var/run/docker.sock:/var/run/docker.sock \
  -v /var/livon/downloads:/downloads \
  --group-add 999 \
  --name jenkins \
  --restart=on-failure \
  jenkins/jenkins:lts-jdk17
```

2. **Jenkins 컨테이너 내부에 Docker CLI 설치**
   - `docker exec -it -u root jenkins /bin/bash`
   - Docker 공식 repo 등록 후 `docker-ce-cli`, `docker-compose-plugin` 설치
3. **Jenkins Credentials**
   - GitLab HTTPS 계정: `gitlab-https-credential`
   - GitLab API Token: `gitlab-api-token`
   - Secret file:
     - `frontend-env-dev`, `frontend-env-prod`
     - `yml-dev`, `yml-prod`
     - `mobile-local-properties`
   - Mattermost Webhook: `mattermost-webhook`
4. **파이프라인 Job 생성**
   - Type: *Pipeline*
   - Definition: *Pipeline script from SCM*
   - SCM: Git
     - Repo URL: `https://lab.ssafy.com/s13-final/S13P31S406.git`
     - Credentials: `gitlab-https-credential`

    - Branch Specifier: `/dev` (dev용), `/master` (운영용) 등
  - Script Path: `Jenkinsfile`

5. **GitLab Webhook 연동**
   - Jenkins Job에서 “Build when a change is pushed to GitLab” 설정
   - 생성된 Webhook URL & Secret token 을 GitLab Webhooks에 등록
   - 일반적으로 **Merge request events** 중심으로 트리거

3-4. docker compose 배포 (Jenkins 또는 수동)

- **개발 환경(dev)**

```
cd LivOnInfra
docker compose -p livon-dev -f docker-compose.dev.yml up -d --build
```

- **운영 환경(prod)**

```
cd LivOnInfra
docker compose -p livon-prod -f docker-compose.prod.yml up -d --bui
ld
```

- 컨테이너 구성:
  - `livon-be-dev` / `livon-be-prod` (백엔드)
  - `livon-fe-dev` / `livon-fe-prod` (프론트엔드)
  - `nginx-dev` / `nginx-prod` (Nginx HTTPS 프록시)

3-5. 주의 사항

- **기존 프로세스 중지 필요 여부**
  - Nginx 80/443 포트를 쓰는 다른 서비스가 있을 경우 **중지 후 배포**
  - Certbot 실행 전에는 반드시 80 포트 사용 중인 서비스 down
- **권한 관련**
  - Docker 명령 사용 계정은 `docker` 그룹에 포함 필수
  - `/var/livon/downloads`는 Jenkins UID/GID 로 소유자 설정
- **로그 확인 경로**

- 컨테이너 로그:

```
docker logs livon-be-dev
docker logs livon-fe-dev
docker logs nginx-dev
```

- Jenkins 빌드 로그: Jenkins 웹 UI → Job → Build Console Output
- Nginx 로그: 컨테이너 내 `/var/log/nginx/access.log`, `error.log` (이미지 기본값)

4. 주요 계정 및 설정 파일 목록

4-1. DB 접속 정보

- **DBMS**
  - MySQL 8.0 (Docker 이미지: `mysql:8.0`)
- **Schema**
  - Database: `livon_db`
- **접속 정보 (애플리케이션 관점)**
  - Host: `livon-mysql`
  - Port: `3306`
  - DB: `livon_db`
  - User: `root`
  - Password: (application.yml 또는 Secret에서 관리, 문서에는 값 직접 기록 지양)
- **접속 정보 파일 위치**
  - 로컬 개발 기본:
    - `LivOnBack/src/main/resources/application.yml`
  - 서버/운영:
    - Jenkins Secret file (`yml-dev`, `yml-prod`) → 배포 시 `LivOnBack/application.yml` 로 복사 후 컨테이너 `/app/config/application.yml` 로 COPY

4-2. 외부 시스템 / 프로젝트 관련 계정

> 실제 키/비밀번호는 절대 Git에 커밋하지 말고, Jenkins Secret, 환경 변수, 별도 Vault 등으로 관리해야 합니다. 업로드된 application.yml에는 실제 값이 포함되어 있으므로, 운영 환경에서는 반드시 키를 재발급하고 교체하는 것을 권장합니다. LivOn(기업_연계)

- **Redis**
  - Host: `livon-redis`
  - Port: `6379`
  - Password: (application.yml에 설정)
- **MongoDB**
  - 외부 MongoDB Atlas 클러스터 URI 사용
  - spring.data.mongodb.uri 로 설정
- **AWS S3**
  - Bucket 예: `livon-s406-bucket` 등
  - AccessKey/SecretKey는 application.yml에 있었으나, 포팅 시 **새 IAM 사용자/키 발급 후 반영**
- **GCP (Vertex AI, GCS 등)**
  - Service Account 키 JSON 파일
  - 환경 변수 `GCP_KEY_FILE`로 경로 전달
  - GCS Bucket 예: `livon-video-uploads` 등
- **LiveKit / OpenVidu v3**
  - LiveKit API Key / Secret
  - URL 예: `https://ov.s406.site/`
  - Egress 설정 (녹화 파일 S3/MinIO 저장 등) 별도 구성
- **MinIO (OpenVidu용)**
  - Endpoint 예: `http://127.0.0.1:9100`
  - access-key / secret-key (application.yml에서 관리)
- **메일 시스템**
  - Gmail 계정 및 앱 비밀번호 사용
  - 포팅 시 별도 메일 계정 및 앱 비밀번호를 새로 생성하여 적용

- **기타**
  - GMS API (SSAFY 제공 gpt-5 프록시)
  - Mattermost Webhook (빌드 알림)

4-3. 주요 프로퍼티 / 설정 파일

- **백엔드**
  - `LivOnBack/src/main/resources/application.yml`
    - Spring Boot 기본 설정 (로컬용)
  - Jenkins Secret file (서버용)
    - `yml-dev` : 개발 환경 DB/외부 서비스 설정
    - `yml-prod` : 운영 환경 설정
  - 컨테이너 내 사용 경로:
    - `/app/config/application.yml`
- **프론트엔드 Web**
  - `.env` (로컬 개발용)
  - Jenkins Secret file:
    - `frontend-env-dev` : 개발 URL/Socket/LiveKit
    - `frontend-env-prod` : 운영 URL/Socket/LiveKit
- **모바일**
  - `LivOnFront/mobile/local.properties`
    - `sdk.dir` 등 Android SDK 경로 설정
    - Jenkins Secret file `mobile-local-properties`에서 복사
- **Infra 관련**
  - Nginx:
    - `LivOnInfra/nginx/nginx.dev.conf`
    - `LivOnInfra/nginx/nginx.prod.conf`
  - docker compose:
    - `LivOnInfra/docker-compose.dev.yml`

        - `LivOnInfra/docker-compose.prod.yml`
    - Jenkins 파이프라인:
        - `Jenkinsfile` (루트 디렉터리)
            - BE/FE 컨테이너 재빌드 & 재기동
            - 모바일 APK 빌드 및 `/download` 디렉터리에 배포
            - Mattermost 알림 전송 등 자동화 포함