빌드 / 배포 관련

🟢 FE (React)

1. 환경 세팅

- Node.js: 20.x LTS
- npm: 10.x 이상
- Yarn (선택): 3.x
- React: 21.x (최신 안정 버전)
- 브라우저: Chrome, Edge 최신 버전 권장
- IDE: VSCode 1.90.x, WebStorm 2025.1

---

2. 의존성 설치

```
# npm
npm install

# 또는 Yarn
yarn install
```

---

3. 개발 서버 실행

```
npm start
# 또는
yarn dev
```

- 기본 포트: `http://localhost:3000`

---

4. 빌드

```
npm run build
# 또는
```

```
yarn build
```

- 빌드 결과: `/build` 폴더 생성

🟡 BE (Spring Boot + Gradle + MySQL)

1. 환경 세팅

- JDK: 21 (LTS)
- Gradle: 9.x
- Spring Boot: 3.3.x
- IDE: IntelliJ 2025.1
- JVM: HotSpot 21
  - 권장 옵션: `Xms512m -Xmx2g -XX:+UseG1GC`
- 웹서버/WAS: 내장 Tomcat 11 (Spring Boot 기본), 외부 Tomcat/Jetty 가능
  - Tomcat 외부 사용 시 포트: 8080, 커넥터 설정: `server.tomcat.max-threads=200`

---

2. 의존성 빌드

```
./gradlew clean build
```

---

3. 애플리케이션 실행

```
# 개발 모드
./gradlew bootRun

# 또는 jar 실행
java -jar build/libs/your-app.jar
```

- 기본 포트: `http://localhost:8080`

🟣 AI (FastAPI)

1. 환경 세팅

- Python: 3.12
- FastAPI: 1.1.x
- Uvicorn: 0.24.x
- Poetry: 1.9.x (권장)
- PyTorch / TensorFlow 최신 버전 (프로젝트 의존)
- IDE: VSCode 1.90.x, PyCharm 2025.1

---

2. 가상환경 및 의존성 설치

```
# Poetry 사용
poetry install

# 또는 pip
python -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

---

3. 개발 서버 실행

```
uvicorn app.main:app --reload --host 0.0.0.0 --port 8000
```

- 기본 포트: `http://localhost:8000`

🟣 배포 관련

ci/cd 자동 빌드 & 배포 파이프라인 (dockerfile 및 gitlab-ci.yml 참고)

nginx 설정 파일 예시:

```
server {
    listen 443 ssl;
    server_name i13a602.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/i13a602.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/i13a602.p.ssafy.io/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
```

```
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;


    client_max_body_size 100M;

    # 파일 서빙- PROD
    location /files/ {
        alias /srv/moya/files/;
        autoindex off;
        add_header Access-Control-Allow-Origin *;
    }

    # 파일 서빙 - DEV
    location /files-dev/ {
        alias /srv/moya/files-dev/;
        autoindex off;
        add_header Access-Control-Allow-Origin *;
    }

    # 프론트- PROD (only)
    location / {
        root /var/www/html;
        index index.html;
        try_files $uri /index.html;
    }

    # 백엔드 - PROD
    location /api/ {
        rewrite ^/api(/.*)$ $1 break;
        proxy_pass http://localhost:8080/;
    }

    # 백엔드 - DEV
    location /api-dev/ {
        rewrite ^/api-dev(/.*)$ $1 break;
        proxy_pass http://localhost:8081/;
    }
```

```
# websocket
location /ws {
    proxy_pass http://localhost:8080/ws;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "Upgrade";
    proxy_set_header Host $host;
}

location /ws-dev {
    proxy_pass http://localhost:8081/ws;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "Upgrade";
    proxy_set_header Host $host;
}

# AI server
location /ai-dev/ {
    rewrite ^/ai-dev(/.*)$ $1 break;
    proxy_pass http://localhost:8001/;
    proxy_connect_timeout 180s;
    proxy_send_timeout 180s;
    proxy_read_timeout 180s;

location /ai/ {
    rewrite ^/ai(/.*)$ $1 break;
    proxy_pass http://localhost:8000/;
    proxy_connect_timeout 180s;
    proxy_send_timeout 180s;
    proxy_read_timeout 180s;
}

    # dozzle
location /logs/ {
     proxy_pass         http://localhost:9999;
     proxy_http_version 1.1;
```

```
        proxy_set_header   Upgrade $http_upgrade;
        proxy_set_header   Connection 'upgrade';
        proxy_set_header   Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}

server {
    listen 80;
    server_name i13a602.p.ssafy.io;

    if ($host = i13a602.p.ssafy.io) {
        return 301 https://$host$request_uri;
    }

    return 404;
}
```