포팅 메뉴얼

1. 기술 스택 및 개발 환경

사용 도구

- **이슈 관리**: JIRA
- **형상 관리**: GitLab
- **커뮤니케이션**: Mattermost, Notion
- **디자인**: Figma
- **영상 편집**: Movavi
- **DB 설계**: ERD Cloud
- **CI/CD**: Docker(26.1.3), Git Runner, AWS EC2

개발 도구

- **IDE**: VS Code, IntelliJ IDEA 2024.3.1.1 Ultimate
- **AI 코딩 보조**: Cursor

개발 환경

Frontend

- **언어/프레임워크**: React (v19.x), TypeScript (v5.7.x)
- **번들러**: Vite (v6.3.1)
- **PWA 지원**: vite-plugin-pwa (v1.0.0)
- **스타일링**: Tailwind CSS (v4.1.4)

Backend

- **JDK**: 17.0.13 LTS
- **프레임워크**: Spring Boot (v3.4.4), FastAPI
- **빌드 툴**: Gradle (v8.13)

Infra & Server

- **클라우드**: AWS EC2 (t2.xlarge), S3, CloudFront
- **웹 서버**: Nginx (v1.18.0)
- **보안 스캔**: ClamAV (v0.103.12)

Database & Storage

- **RDBMS**: MySQL (v8.0.42)
- **NoSQL**: MongoDB (v6.0.22)
- **인메모리**: Redis (v7.0.15)
- **벡터 DB**: Qdrant (v1.14.0)

Messaging & Streaming

- **프로토콜**: STOMP, WebSocket (v3.4.1)

Authentication

- **토큰**: JWT (v0.11.5)
- **OAuth**

문서화

- **API 문서**: Swagger / Spring RestDocs
- **코드 문서**: Javadoc

1.2 환경변수

Front

- `.env`

```
VITE_API_URL=https://checkmate.ai.kr
VITE_KAKAO_MAP_KEY=
VITE_REST_API=
VITE_REDIRECT_URL=https://checkmate.ai.kr/login
```

Back

- `.env`

```
MYSQL_HOST=k12c103.p.ssafy.io
MYSQL_PORT=3306
MYSQL_DATABASE=checkmate
MYSQL_USERNAME=ssafy6B
MYSQL_PASSWORD=

REDIS_HOST=k12c103.p.ssafy.io
REDIS_PORT=6379
REDIS_PASSWORD=

MONGODB_HOST=k12c103.p.ssafy.io
MONGODB_PORT=27017
MONGODB_DATABASE=checkmate
MONGODB_USERNAME=ssafy
MONGODB_PASSWORD=
MONGODB_AUTH_DB=admin

JWT_SECRET=ssafy-gwangju-C103-cpzm-mate-pjt-6B-check-apdlxm-wkdbf
JWT_ACCESS_EXPIRE_TIME=3600000
JWT_REFRESH_EXPIRE_TIME=2592000000

ENCRYPTION_PASSWORD=cpzmapdlxmc103password!
ENCRYPTION_SALT=a1b2c3d4e5f60708
AES_ENCRYPTION_KEY=CUNEAri8up1LEuDtR92MVrB3L/qDGlflld1SU1KLzws=

AWS_S3_BUCKET=checkmate-b6
AWS_S3_ACCESS=
AWS_S3_SECRET=
AWS_S3_REGION=ap-northeast-2
AWS_CLOUDFRONT_DOMAIN=d24rvdqlfsgh19.cloudfront.net
AWS_CLOUDFRONT_KEY_PAIR_ID=
AWS_CLOUDFRONT_PRIVATE_KEY_PATH=/etc/cloudfront/private_key.pem

CLAMAV_HOST=k12c103.p.ssafy.io
CLAMAV_PORT=3310
```

```
NAVER_API_CLIENT_ID=
NAVER_API_CLIENT_SECRET=
NAVER_API_URL=https://openapi.naver.com/v1/search/news.json

OPENAI_API_KEY=
OPENAI_API_URL=https://api.openai.com

HS_API_KEY=
HS_CLIENT_ID=

WEBHOOK_API_KEY=
```

- `application.yml`

```
spring:
  datasource:
    url: jdbc:mysql://${MYSQL_HOST}:${MYSQL_PORT}/${MYSQL_DATABASE
    username: ${MYSQL_USERNAME}
    password: ${MYSQL_PASSWORD}
    driver-class-name: com.mysql.cj.jdbc.Driver

  data:
    redis:
      host: ${REDIS_HOST}
      port: ${REDIS_PORT}
      password: ${REDIS_PASSWORD}

    mongodb:
      host: ${MONGODB_HOST}
      port: ${MONGODB_PORT}
      database: ${MONGODB_DATABASE}
      username: ${MONGODB_USERNAME}
      password: ${MONGODB_PASSWORD}
      authentication-database: ${MONGODB_AUTH_DB}

  ai:
    openai:
```

```yaml
      api-key: ${OPENAI_API_KEY}
      chat:
        options:
          model: gpt-4.1-mini
          temperature: 0.5
      base-url: ${OPENAI_API_URL}

jwt:
  secret: ${JWT_SECRET}
  access-token-validity: ${JWT_ACCESS_EXPIRE_TIME}
  refresh-token-validity: ${JWT_REFRESH_EXPIRE_TIME}

encryption:
  password: ${ENCRYPTION_PASSWORD}
  salt: ${ENCRYPTION_SALT}

aes:
  encryption:
    base64-key: ${AES_ENCRYPTION_KEY}

aws:
  s3:
    bucket: ${AWS_S3_BUCKET}
    access: ${AWS_S3_ACCESS}
    secret: ${AWS_S3_SECRET}
    region: ${AWS_S3_REGION}
  cloudfront:
    domain: ${AWS_CLOUDFRONT_DOMAIN}
    key-pair-id: ${AWS_CLOUDFRONT_KEY_PAIR_ID}
    private-key-path: ${AWS_CLOUDFRONT_PRIVATE_KEY_PATH}

clamav:
  host: ${CLAMAV_HOST}
  port: ${CLAMAV_PORT}

naver:
  api:
    client-id: ${NAVER_API_CLIENT_ID}
    client-secret: ${NAVER_API_CLIENT_SECRET}
    url: ${NAVER_API_URL}

hs:
  api:
    key: ${HS_API_KEY}
  client:
    id: ${HS_CLIENT_ID}

webhook:
  api-key: ${WEBHOOK_API_KEY}
```

FastAPI

- `.env`

```
OPENAI_API_KEY=
QDRANT_URL=http://k12c103.p.ssafy.io:6333
QDRANT_API_KEY=
EMBEDDING_REPO_ID=
MONGO_HOST=k12c103.p.ssafy.io
MONGO_PORT=27017
MONGO_DB=checkmate
MONGO_USERNAME=ssafy
MONGO_PASSWORD=
MONGO_AUTH=admin

AI_ANALYSIS_REPORT_COLLECTION=ai_analysis_report
IMPROVEMENT_REPORT_COLLECTION=improvement_report
MISSING_CLAUSE_REPORT_COLLECTION=missing_clause_report
RISK_CLAUSE_REPORT_COLLECTION=risk_clause_report

MYSQL_HOST=k12c103.p.ssafy.io
MYSQL_PORT=3306
MYSQL_DATABASE=checkmate
MYSQL_USERNAME=ssafy6B
MYSQL_PASSWORD=
REDIS_HOST=k12c103.p.ssafy.io
REDIS_PORT=6379
REDIS_PASSWORD=

ENCRYPTION_BASE64_KEY=S7lLIdyzGROzgM9HVBtYMmUL61lVALeiYOn/7T
AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_REGION=ap-northeast-2
S3_BUCKET_NAME=

UPSTAGE_OCR_API_URL=https://api.upstage.ai/v1/document-digitization
UPSTAGE_OCR_API_KEY=

SPRINGBOOT_WEBHOOK_URL=https://k12c103.p.ssafy.io/api/webhook
WEBHOOK_API_KEY=

HUGGINGFACE_API_KEY=
```

2. CI/CD 구축

GitLab Runner 세팅

- GitLab 프로젝트에 Runner 등록
- 프로젝트 루트에 .gitlab-ci.yml 연결
- Runner 태그 필터링
- Trigger - GitLab hook 설정
- GitLab Variables에서 변수 관리

Front

- `.gitlab-ci.yml`

```
image: node:22.12.0

stages:
  - build
  - deploy

variables:
  MATTERMOST_WEBHOOK: $MATTERMOST_WEBHOOK

before_script:
  - echo "GitLab CI/CD 시작!"
  - node -v
  - npm -v

build_checkmate_fe:
  stage: build
  only:
    - fe
  script:
    - cd checkmate-fe
    - echo "$FRONTEND_ENV" > .env
    - cat .env
    - npm ci
    - npm run build
  artifacts:
    paths:
      - checkmate-fe/dist
  after_script:
    - |
      if [ "$CI_JOB_STATUS" == "success" ]; then
        curl -X POST -H 'Content-Type: application/json' \
        -d "{\"text\": \"✅ [프론트 빌드 성공] $CI_PROJECT_NAME - $CI_COMMIT_
      else
        curl -X POST -H 'Content-Type: application/json' \
        -d "{\"text\": \"❌ [프론트 빌드 실패] $CI_PROJECT_NAME - $CI_COMMIT_
      fi

deploy_checkmate_fe:
  stage: deploy
  only:
    - fe
  script:
    - echo "🚀 Docker 배포 시작"
    - cd checkmate-fe
    - docker stop checkmate-fe-container || true
    - docker rm checkmate-fe-container || true
    - docker rmi checkmate-fe || true
    - docker build -t checkmate-fe .  # 이때 Dockerfile은 dist만 복사
    - docker run -d --name checkmate-fe-container -p 3000:80 checkmate-fe
  after_script:
    - |
      if [ "$CI_JOB_STATUS" == "success" ]; then
        curl -X POST -H 'Content-Type: application/json' \
        -d "{\"text\": \"✅ [프론트 배포 성공] $CI_PROJECT_NAME - $CI_COMMIT_
      else
        curl -X POST -H 'Content-Type: application/json' \
        -d "{\"text\": \"❌ [프론트 배포 실패] $CI_PROJECT_NAME - $CI_COMMIT_
      fi
```

Back

- .gitlab-ci.yml

```
image: docker:latest

stages:
  - build
  - deploy

variables:
  DOCKER_IMAGE: checkmate-be

before_script:
  - echo "🚀 백엔드 GitLab CI 시작"
  - docker --version
```

```
# ------------------ build ------------------
build_checkmate_be:
  stage: build
  only:
    - be
  script:
    - cd checkmate-be
    # ✅ application-secret.yml 생성
    - echo "$APPLICATION_SECRET_YML" > src/main/resources/application-se
    - echo "✅ application-secret.yml 파일 생성 완료"
    # ✅ .env 파일 생성 (프로젝트 루트로 돌아와서 만들기)
    - cd ..
    - cp "$BACKEND_ENV" .env
    - echo "✅ .env 파일 생성 완료"

    # ✅ Docker 이미지 정리 및 빌드
    - docker stop $DOCKER_IMAGE || true
    - docker rm   $DOCKER_IMAGE || true
    - docker rmi  $DOCKER_IMAGE || true
    - docker build -t $DOCKER_IMAGE checkmate-be
  artifacts:
    paths:
      - .env
      - checkmate-be/build/libs/

  after_script:
    - |
      if [ "$CI_JOB_STATUS" == "success" ]; then
        curl -X POST -H 'Content-Type: application/json' \
             -d "{\"text\": \"✅ [백엔드 빌드 성공] $CI_PROJECT_NAME - $CI_COMMI
      else
        curl -X POST -H 'Content-Type: application/json' \
             -d "{\"text\": \"❌ [백엔드 빌드 실패] $CI_PROJECT_NAME - $CI_COMMI
      fi

# ------------------ deploy ------------------
deploy_checkmate_be:
  stage: deploy
  needs: [build_checkmate_be]
  only:
    - be
  script:
    - docker stop $DOCKER_IMAGE || true
    - docker rm   $DOCKER_IMAGE || true
    - mkdir -p cloudfront
    - echo "$CLOUDFRONT_PRIVATE_KEY" > cloudfront/private_key.pem
    - ls -l cloudfront/private_key.pem

    - >
      docker run -d --name $DOCKER_IMAGE \
        -p 8080:8080 \
        --env-file .env \
        -v $PWD/cloudfront/private_key.pem:/etc/cloudfront/private_key.pem \
        $DOCKER_IMAGE
  after_script:
    - |
      if [ "$CI_JOB_STATUS" == "success" ]; then
        curl -X POST -H 'Content-Type: application/json' \
             -d "{\"text\": \"✅ [백엔드 배포 성공] $CI_PROJECT_NAME - $CI_COMMI
      else
        curl -X POST -H 'Content-Type: application/json' \
             -d "{\"text\": \"❌ [백엔드 배포 실패] $CI_PROJECT_NAME - $CI_COMMI
      fi
```

Docker 파일

1. **MySQL**

```
docker run -d --name mysql \
  -e MYSQL_DATABASE=checkmate \
  -e MYSQL_USER=ssafy6B \
  -e MYSQL_PASSWORD='checkmateC103!^^7' \
  -e MYSQL_ROOT_PASSWORD='checkmateC103!^^7' \
  -e TZ=Asia/Seoul \
  -p 3306:3306 \
  -v mysql_data:/var/lib/mysql \
  --restart always \
  mysql:8.0
```

2. **Redis**

```
docker run -d --name redis \
  -p 6379:6379 \
  -v redis_data:/data \
  --restart always \
  redis:7.0 \
  sh -c 'exec redis-server --requirepass '\''checkmateC103!^^6'\'''
```

3. **MongoDB**

```
docker run -d --name mongodb \
  -p 27017:27017 \
  -v mongo_data:/data/db \
  -e TZ=Asia/Seoul \
  -e MONGO_INITDB_ROOT_USERNAME=ssafy \
  -e MONGO_INITDB_ROOT_PASSWORD='checkmateC103^^5' \
  --restart always \
  mongo:6.0
```

4. **Qdrant**

```
docker run -d --name qdrant\
  -p 6333:6333\
  -v qdrant_data:/qdrant/storage \
  -e QDRANT__SERVICE__API_KEY='checkmateC103^^1' \
  qdrant/qdrant
```

5. **SprinBoot Dockerfile**

```
# 1단계: 빌드 단계
FROM gradle:8.8-jdk17 AS build
```

```
# 작업 디렉토리 설정
WORKDIR /home/app

# Gradle 파일 복사
COPY build.gradle /home/app/
COPY settings.gradle /home/app/

# 종속성 미리 다운로드
RUN gradle build -x test --parallel --continue || true

# 소스 코드 복사
COPY src /home/app/src

# 프로젝트 빌드
RUN gradle clean build -x test

# 2단계: 실행 단계
FROM eclipse-temurin:17-jre

# 빌드 결과물 복사
COPY --from=build /home/app/build/libs/*.jar /app.jar

# 애플리케이션 실행
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

6. **Front Dockerfile**

```
# 정적 파일만 Nginx로 복사
FROM nginx:latest

COPY nginx.conf /etc/nginx/nginx.conf
COPY dist /usr/share/nginx/html

EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

7. **FastAPI Dockerfile**

```
FROM python:3.12-slim

WORKDIR /app

# 시스템 패키지 설치
RUN apt-get update && apt-get install -y \
    git \
    wget \
    && rm -rf /var/lib/apt/lists/*

# Python 패키지 설치
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# 앱 코드 복사
COPY . /app

# FastAPI 실행
EXPOSE 7860
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "7860"]
```

ClamAV

1. ubuntu에 `ClamAV` 다운

```
sudo apt install -y clamav clamav-daemon
```

2. 바이러스 데이터베이스 업데이트

```
sudo systemctl stop clamav-freshclam
sudo freshclam
sudo systemctl start clamav-freshclam
```

3. ClamAV 데몬 시작 및 자동 실행 설정

```
sudo systemctl enable clamav-daemon
sudo systemctl start clamav-daemon
```

3. 빌드 및 배포

1. Back

빌드

0. Trigger: `be` 브랜치에 push될 때만 실행
1. `cd checkmate-be` 후 `src/main/resources/application-secret.yml`로 덤프
2. 프로젝트 루트로 돌아와 `.env` 생성 (`BACKEND_ENV` 변수)
3. 기존 이미지/컨테이너 정리 (`docker stop/rm/rmi`)
4. `docker build -t checkmate-be checkmate-be` 로 이미지 빌드

배포

0. **Needs:** `build_checkmate_be` 성공 시에만 실행
1. Trigger: `be` 브랜치
2. 기존 컨테이너/이미지 정리 (`docker stop/rm`)
3. Docker 실행

```
docker run -d --name checkmate-be \
    -p 8080:8080 \
    --env-file .env \
    -v
```

4.
알림: Mattermost Webhook으로 성공/실패 메시지 전

2. Front

빌드

0. Trigger: `fe` 브랜치에 push될 때만 실행
1. `cd checkmate-fe` 후 `.env` 파일 생성
2. 의존성 설치 & 빌드
    a. `checkmate-fe/dist/` (빌드된 정적 파일)

배포

0. **Needs:** `build_checkmate_fe` 성공 시에만 실행
1. Trigger: `fe` 브랜치
2. 기존 컨테이너/이미지 정리 (`docker stop/rm`)
3. Docker 실행

```
docker run -d \
  --name checkmate-fe-container \
  -p 3000:80 \
  checkmate-fe
```

4.
알림: Mattermost Webhook으로 성공/실패 메시지 전

3. Nginx

Nginx 설정 파일

1. `checkmate-fe` 폴더 안에 `nginx.conf`

```
events { }

http {
  include       mime.types;
  default_type  application/octet-stream;
  sendfile        on;
  keepalive_timeout  65;
  server {
    listen 80;
    server_name _;

    root /usr/share/nginx/html;
    index index.html;

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

2. `ubuntu` 에 `/etc/nginx/sites-available` 폴더 안에 `checkmate-fe` nginx 설정파일

```
# HTTP → HTTPS 강제 리디렉션
server {
    listen 80;
    server_name k12c103.p.ssafy.io www.checkmate.ai.kr checkmate.ai.kr;

    return 301 https://$host$request_uri;
}

# HTTPS 설정
server {
    listen 443 ssl;
    server_name k12c103.p.ssafy.io www.checkmate.ai.kr checkmate.ai.kr;

    server_tokens off;

    ssl_certificate /etc/letsencrypt/live/k12c103.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/k12c103.p.ssafy.io/privkey.pem;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;

    # 백엔드 API 요청 프록시
    location /api/ {
        proxy_pass http://127.0.0.1:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    # WebSocket → 소켓 서버
    location /app/ {
        proxy_pass http://127.0.0.1:8080;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Real-Ip $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

        # 타임아웃 버퍼
        proxy_read_timeout 60s;
    }

    # 프론트엔드 React 정적 파일
    location / {
        proxy_pass http://127.0.0.1:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

SSL 인증서 적용

1. `Cerbot` **설치**

```
sudo apt install -y certbot python3-certbot-nginx
```

2. **SSL 인증서 설정** (`Let`s Encrypt`)

```
# SSL 인증서 설정 (Let`s Encrypt)
/etc/letsencrypt/live/k12c103.p.ssafy.io/fullchain.pem
/etc/letsencrypt/live/k12c103.p.ssafy.io/privkey.pem
```

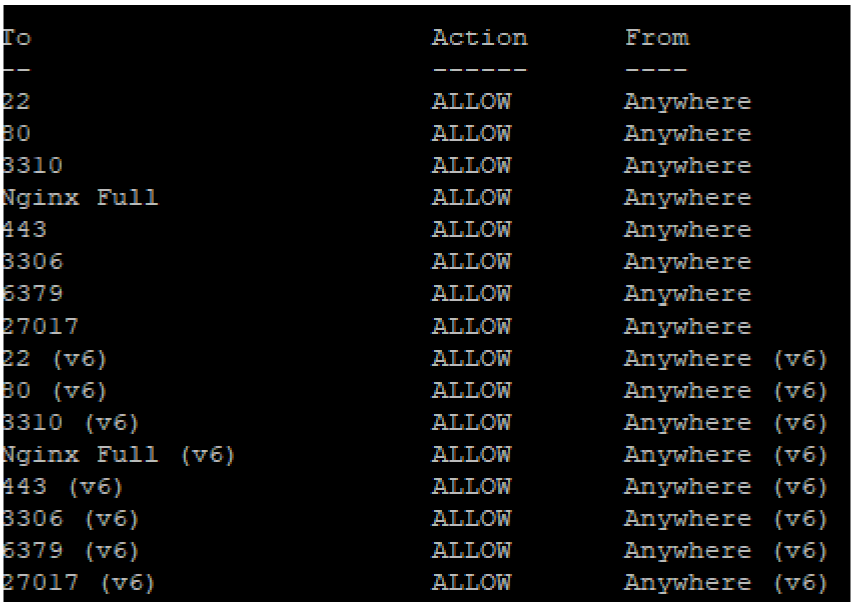
SSL 인증서 갱신 방법

```
sudo certbot renew --dry-run # SSL 갱신 테스트
sudo certbot renew # SSL 갱신 실행
```

3. **TSL/SSL 인증서 발급**

```
sudo certbot --nginx -d k12c103.p.ssafy.io -d www.checkmate.ai.kr
```

4. **포트번호**



```
To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
80                         ALLOW       Anywhere
3310                       ALLOW       Anywhere
Nginx Full                 ALLOW       Anywhere
443                        ALLOW       Anywhere
3306                       ALLOW       Anywhere
6379                       ALLOW       Anywhere
27017                      ALLOW       Anywhere
22 (v6)                    ALLOW       Anywhere (v6)
80 (v6)                    ALLOW       Anywhere (v6)
3310 (v6)                  ALLOW       Anywhere (v6)
Nginx Full (v6)            ALLOW       Anywhere (v6)
443 (v6)                   ALLOW       Anywhere (v6)
3306 (v6)                  ALLOW       Anywhere (v6)
6379 (v6)                  ALLOW       Anywhere (v6)
27017 (v6)                 ALLOW       Anywhere (v6)
```

4. 외부 서비스 및 활용 정보

- Dropbox Sign API

- Kakao Login API
- Huggingface spaces
- 국토교통부 지오코더 API (법원 데이터 넣을때 사용)
- OpenAI API
- Naver News API
- Daum 우편번호 API
- Kakao Local REST API