포팅 메뉴얼

상태	제니

사용 도구

기본 도구

도구	버전	설명
Ubuntu	22.04.5 LTS	서버 운영체제
Docker	27.5.1	컨테이너 관리
Docker Compose	2.32.4	다중 컨테이너 관리
GitLab	17.6.2	버전 관리
Open JDK	17.0.14	Java 실행 환경
Node.js	22.13.1	JavaScript 실행 환경
Jira	-	이슈관리
Notion	-	회의관리
Figma	-	디자인

라이브러리/프레임 워크

이름	버전	용도
React.js	18.3.1	프론트엔드 프레임워크
Spring Boot	3.4.2	백엔드 프레임워크
KafKa	4.1.0	메시징 분산 도구
Nginx	1.18.0	웹 서버, 프록시 서버
Jenkins	2.492.1	CI/CD 자동화 도구

데이터베이스

이름	버전	용도
MySQL	22.04.1	관계형 데이터베이스
Redis	7.4.1	인메모리 데이터베이스
ElasticSeatch	8.12.1	NoSQL 데이터베이스

환경 변수

Frontend

```
VITE_BASE_URL=${backend-Url}
예) https://k12d204.p.ssafy.io/api
```

Backend

application.yml

```
spring:
  application:
    name: book-shy  # 🪪 서비스 이름 설정

  profiles:
  active: dev     # ✅ 현재 활성화할 profile 지정 (dev, prod 등)
  include: key     # 🔐 보안 정보 포함한 외부 yml 파일 분리 (application-key.yml)

jackson:
  time-zone: Asia/Seoul

data:
  redis:
    host: k12d204.p.ssafy.io
    port: 6379
    password: ${REDIS.PASSWORD}  # 🔐 외부 설정에서 불러오기

kafka:
  bootstrap-servers: ${KAFKA.BOOTSTRAP-SERVERS}  # 🔗 Kafka 클러스터 주소

  producer:
    key-serializer: org.apache.kafka.common.serialization.StringSerializer
    value-serializer: org.springframework.kafka.support.serializer.JsonSerializer
    properties:
      partitioner.class: org.apache.kafka.clients.producer.RoundRobinPartitioner  # 🔄 라운드 로빈 파티셔너
      spring.json.add.type.headers: false  # ✅ 헤더 정보 생략
      spring.json.trusted.packages: "*"  # ✅ DTO 역직렬화 허용 패키지

  consumer:
    group-id: ${KAFKA.CONSUMER.BOOK-GROUP-ID}  # 👥 Kafka 컨슈머 그룹 ID
    key-deserializer: org.apache.kafka.common.serialization.StringDeserializer
```

```yaml
    value-deserializer: org.springframework.kafka.support.serializer.JsonDeserializer
    auto-offset-reset: earliest  # 🕒 최초 시작 시 earliest부터 읽음
    properties:
      spring.json.trusted.packages: "*"  # ✅ DTO 역직렬화 허용 패키지

  listener:
    ack-mode: manual_immediate  # 🎯 수동 커밋, 메시지 중복 처리 방지

server:
  port: 8080  # 🌐 기본 포트 (profile별로 override 가능)

logging:
  level:
    root: INFO  # 📝 로깅 레벨 (DEBUG/INFO/WARN/ERROR)

management:
  endpoints:
    web:
      exposure:
        include: health,info  # ✅ 헬스체크 및 서비스 정보 노출
  endpoint:
    health:
      show-details: always  # 헬스체크 상세 정보 응답 포함

file:
  upload-dir: /home/ubuntu/bookshy/images/coverImage

elasticsearch:
  url: ${ELK_URL}

# 개발자 식별자 설정 추가
app:
  developer:
    id: ${DEV_ID}
```

application-dev.yml

```yaml
spring:
  datasource:
    url: jdbc:postgresql://${DB.HOST}:${DB.PORT}/${DB.NAME}  # ✅ PostgreSQL 접속 URL (개발용)
    username: ${DB.USERNAME}                                 # ✅ DB 유저명
    password: ${DB.PASSWORD}                                 # ✅ DB 비밀번호
    driver-class-name: org.postgresql.Driver

  jpa:
    hibernate:
      ddl-auto: update   # ✅ 개발 시에는 테이블 자동 생성/업데이트 허용
    database-platform: org.hibernate.dialect.PostgreSQLDialect
    show-sql: true      # ✅ SQL 로그 출력
    properties:
      hibernate:
        format_sql: true # ✅ SQL을 보기 좋게 포맷팅
        jdbc:
          time_zone: Asia/Seoul

  kafka:
    bootstrap-servers: ${KAFKA.BOOTSTRAP-SERVERS}
    consumer:
      # ✅ 그룹별 ID를 설정하여 메시지 처리를 역할별로 분리
      book-group-id: ${KAFKA.CONSUMER.BOOK-GROUP-ID}
      match-group-id: ${KAFKA.CONSUMER.MATCH-GROUP-ID}
      trade-group-id: ${KAFKA.CONSUMER.TRADE-GROUP-ID}
      chat-group-id: ${KAFKA.CONSUMER.CHAT-GROUP-ID}
      recommend-group-id: ${KAFKA.CONSUMER.RECOMMEND-GROUP-ID}

server:
  port: 8080  # ✅ 개발용 포트
```

application-key.yml

```yaml
DB:
  HOST: k12d204.p.ssafy.io
  PORT: 5432
  NAME: d204
  USERNAME: bookshy
  PASSWORD: ssafyd204@!

REDIS:
  PASSWORD: ssafyd204@!
```

```yaml
KAFKA:
  BOOTSTRAP-SERVERS: k12d204.p.ssafy.io:19092,k12d204.p.ssafy.io:19093,k12d204.p.ssafy.io:19094
  CONSUMER:
    BOOK-GROUP-ID: book-consumer-group
    MATCH-GROUP-ID: match-consumer-group
    TRADE-GROUP-ID: trade-consumer-group
    CHAT-GROUP-ID: chat-consumer-group
    RECOMMEND-GROUP-ID: recommend-consumer-group

jwt:
  secret-key: "여기에_강력한_비밀키를_입력할까말까_i_LOVe_Altong_dackk"
  expiration-time: 1296000000   # 액세스 토큰 만료 시간 (1시간)
  refresh-expiration: 1296000000 # 리프레시 토큰 만료 시간
  token-prefix: "Bearer "
  header-string: "Authorization"

issuer: "bookshy-application"

oauth:
  kakao:
    user-info-uri: https://kapi.kakao.com/v2/user/me
    client-id: c05b2b71032f579f88db700fa66c2cc1
    redirect-uri: http://localhost:5173/oauth

naver:
  ocr:
    url: https://chzqmycph5.apigw.ntruss.com/custom/v1/41690/97b3ca6d63855dfa51e7e316d6afc29025e8cb13e0051df9f200d177d5cb46f0/general
    secretKey: UUJ6eEpJQ0NWc3ZkQ1JobVhUZ0xRb2dRbEdheHRXdmU=

aladin:
  ttb:
    key: ttbkwonjm12031421001
  api:
    base-url: https://www.aladin.co.kr/ttb/api

elasticsearch:
  url: k12d204.p.ssafy.io:9200

app:
  developer:
    id: subi
```

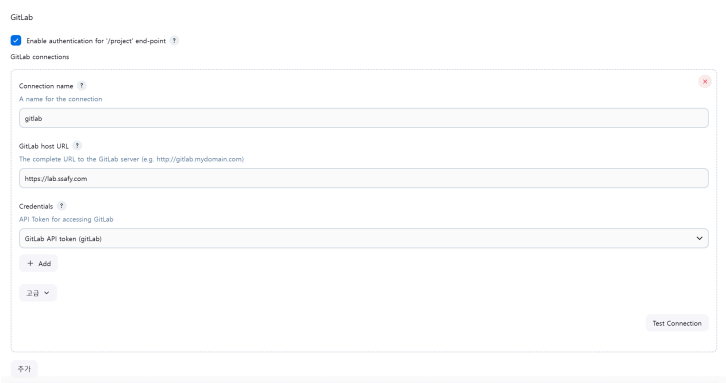
application-prod.yml

```yaml
spring:
  datasource:
    url: ${DB_URL}
    username: ${DB_USERNAME}
    password: ${DB_PASSWORD}
    driver-class-name: org.postgresql.Driver

  jpa:
    hibernate:
      ddl-auto: validate  # ✅ 운영에서는 테이블 구조 변경을 금지하고 유효성만 검사
    database-platform: org.hibernate.dialect.PostgreSQLDialect
    show-sql: false       # ✅ 운영에서는 SQL 로그 미출력
    properties:
      hibernate:
        jdbc:
          time_zone: Asia/Seoul

  kafka:
    bootstrap-servers: ${KAFKA_BOOTSTRAP_SERVERS}
    consumer:
      # ✅ 운영에서도 동일하게 각 역할별 그룹 ID 지정
      book-group-id: book-consumer-group
      match-group-id: match-consumer-group
      trade-group-id: trade-consumer-group
      chat-group-id: chat-consumer-group
      recommend-group-id: recommend-consumer-group

aladin:
  api:
    base-url: ${ALADIN_BASE_URL}
  ttb:
    key: ${ALADIN_TTB_KEY}

naver:
  ocr:
    secretKey: ${NAVER_OCR_SECRET_KEY}
    url: ${NAVER_OCR_URL}
```

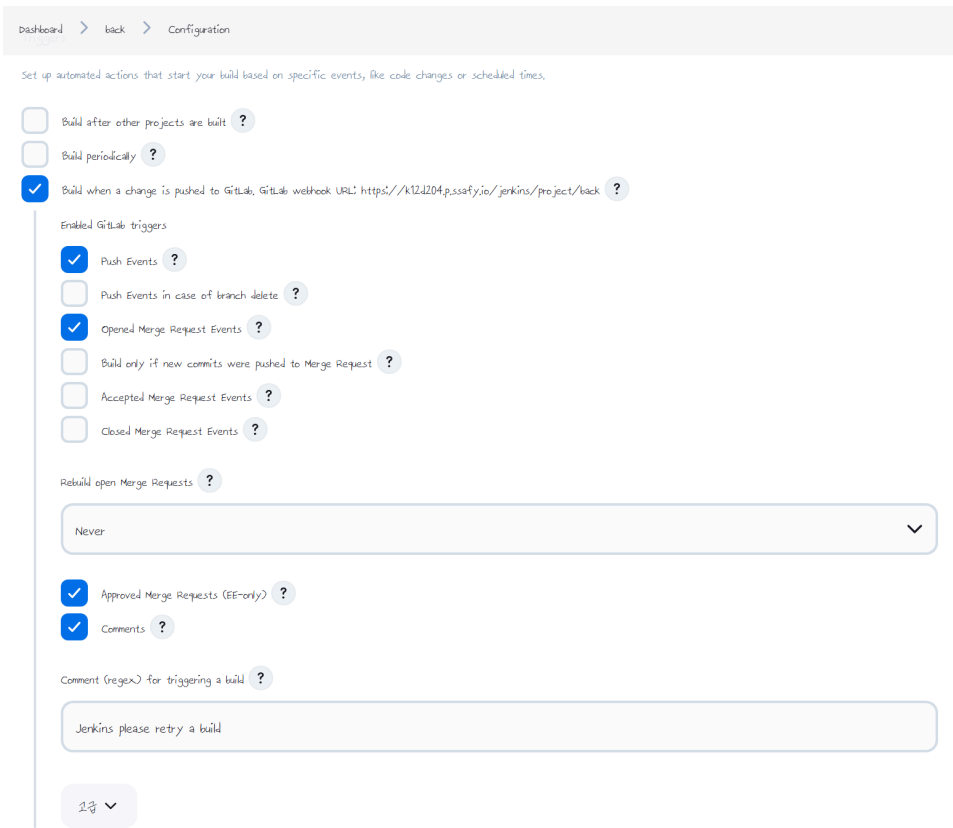
```
oauth:
  kakao:
    user-info-uri: ${KAKAO_USER_INFO_URI}
    client-id: ${KAKAO_CLIENT_ID}
    redirect-uri: ${KAKAO_REDIRECT_URI}

server:
  port: 8080  # ✅ 운영용 포트

elasticsearch:
  url: ${ELK_URL}

app:
  developer:
    id: ""  # 비워둠
```

🦥 Jenkins CI/CD 구축

1. **Jenkins PlugIn**
    a. gitLab, gitLab api, gitLab athuntication
    b. docker, docker pipeline, docker api
    c. ssh, ssh agent
    d. node
2. **gitLab tool 추가**



3. **gitLab Webhook 연결**



4. **Item**

5. **Credentials**

Dashboard > Jenkins 관리 > Credentials

Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	gitlab	GitLab API token (gitlab)
		System	(global)	lab-token	lab-token
		System	(global)	SONAR_KEY	SONAR_KEY
		System	(global)	sonarqube-token	sonarqube-token
		System	(global)	docker-hub-credentials	subihwang/****** (docker-hub-credentials)
		System	(global)	remote-server-credentials	ubuntu (remote-server-credentials)
		System	(global)	DB_URL	DB_URL
		System	(global)	DB_USERNAME	DB_USERNAME
		System	(global)	DB_PASSWORD	DB_PASSWORD
		System	(global)	VITE_BASE_URL	VITE_BASE_URL
		System	(global)	REDIS_PASSWORD	REDIS_PASSWORD
		System	(global)	KAFKA_BOOTSTRAP_SERVERS	KAFKA_BOOTSTRAP_SERVERS
		System	(global)	JWT_KEY	JWT_KEY
		System	(global)	ALADIN_TTB_KEY	ALADIN_TTB_KEY
		System	(global)	ALADIN_BASE_URL	ALADIN_BASE_URL
		System	(global)	NAVER_OCR_SECRET_KEY	NAVER_OCR_SECRET_KEY
		System	(global)	NAVER_OCR_URL	NAVER_OCR_URL
		System	(global)	KAKAO_USER_INFO_URI	KAKAO_USER_INFO_URI
		System	(global)	KAKAO_CLIENT_ID	KAKAO_CLIENT_ID
		System	(global)	KAKAO_REDIRECT_URI	KAKAO_REDIRECT_URI
		System	(global)	VITE_FIREBASE_API_KEY	firebase api key
		System	(global)	VITE_FIREBASE_AUTH_DOMAIN	firebase auth domain
		System	(global)	VITE_FIREBASE_PROJECT_ID	firebase project id
		System	(global)	VITE_FIREBASE_STORAGE_BUCKET	firebase storage bucket
		System	(global)	VITE_FIREBASE_MESSAGING_SENDER_ID	firebase messaging sender id
		System	(global)	VITE_FIREBASE_APP_ID	firebase app id
		System	(global)	VITE_FIREBASE_MEASUREMENT_ID	firebase mesurment id
		System	(global)	VITE_FIREBASE_VAPID_KEY	firebase vapid key
		System	(global)	ELK_URL	ELK_URL
		System	(global)	firebase-service-account	firebase-service-account.json
		System	(global)	VITE_KAKAO_REST_API_KEY	VITE_KAKAO_REST_API_KEY

🦥 Nginx 설정

```
server {
    listen 80;
    server_name k12d204.p.ssafy.io;

    # HTTP를 HTTPS로 리다이렉트
    location / {
        try_files $uri $uri/ /index.html;
        return 301 https://$host$request_uri;
    }

    # .well-known 디렉토리 접근 허용 (인증서 갱신용)
    location /.well-known {
        root /usr/share/nginx/html;
        allow all;
    }

    location /images/profile/ {
        alias /usr/share/nginx/html/images/profile;
    }
```

```
    location /images/coverImage/ {
        alias /usr/share/nginx/html/images/coverImage/;
        add_header 'Access-Control-Allow-Origin' '*';
        add_header 'Access-Control-Allow-Methods' 'GET, OPTIONS';
        add_header 'Access-Control-Allow-Headers' 'Origin, Content-Type, Accept';
    }

}

server {
    listen 443 ssl; # ssl 매개변수가 이미 여기에 있음
    server_name k12d204.p.ssafy.io;

    # SSL 설정
    ssl_certificate /etc/ssl/certificate.crt;
    ssl_certificate_key /etc/ssl/private.key;

    # 책 커버 이미지 경로 설정
    location /images/coverImage/ {
        alias /usr/share/nginx/html/images/coverImage/;
          add_header 'Access-Control-Allow-Origin' '*';
        add_header 'Access-Control-Allow-Methods' 'GET, OPTIONS';
        add_header 'Access-Control-Allow-Headers' 'Origin, Content-Type, Accept';
    }

    # 유저 프로필 이미지 경로 설정
    location /images/profile/ {
      alias /usr/share/nginx/html/images/profile/;
      add_header 'Access-Control-Allow-Origin' '*';
      add_header 'Access-Control-Allow-Methods' 'GET, OPTIONS';
      add_header 'Access-Control-Allow-Headers' 'Origin, Content-Type, Accept';
    }

    # Jenkins 경로 설정
    location /jenkins/ {
        proxy_pass http://jenkins:8080/jenkins/;
        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;
        proxy_redirect http:// https://;
    }

    # SonarQube 경로 설정
    location /sonarqube/ {

        proxy_pass http://sonarqube:9000/sonarqube/;
        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;
    }

    # 백엔드 API 경로 설정
    location /api/{
        proxy_pass http://backend:8080/api/;
        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;
    }


    # 프론트엔드 설정
     location / {
        proxy_pass http://frontend:80/;
        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;
        proxy_redirect http:// https://;

      }


    # 채팅 웹소켓 설정
    location /ws-chat {
      proxy_pass        http://backend:8080/ws-chat;
      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;
    proxy_read_timeout 600s;  # ⏳ WebSocket 연결 유지 시간 연장
  }


}
```

🦥 Docker 설정

전체 도커 컨테이너

```
ubuntu@ip-172-26-7-33:~$ docker ps
CONTAINER ID   IMAGE                                                    COMMAND
         CREATED          STATUS          PORTS

                                                     NAMES
722507b9337b   subihwang/frontend:latest                                "/docker-entrypo
int.…"   55 seconds ago   Up 54 seconds   0.0.0.0:3000->80/tcp, [::]:3000->80/tcp

                                                     frontend
10c9cef4397b   subihwang/backend:latest                                 "java -Xmx512m -
Xms2…"   6 hours ago      Up 6 hours      0.0.0.0:8080->8080/tcp, [::]:8080->8080/tcp

                                                     backend
931d48530dc4   docker.elastic.co/kibana/kibana:8.12.1                   "/bin/tini -- /u
sr/l…"   9 days ago       Up 9 days       0.0.0.0:5601->5601/tcp, [::]:5601->5601/tcp

                                                     kibana
ad30c1d7d30f   docker.elastic.co/logstash/logstash:8.12.1               "/usr/local/bin/
dock…"   9 days ago       Up 6 hours      0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp,
 0.0.0.0:5044->5044/tcp, [::]:5044->5044/tcp, 0.0.0.0:9600->9600/tcp, 0.0.0.0:5000->50
00/udp, [::]:9600->9600/tcp, [::]:5000->5000/udp   logstash
a6e307ca2bdf   docker.elastic.co/elasticsearch/elasticsearch:8.12.1     "/bin/tini -- /u
sr/l…"   9 days ago       Up 6 days       0.0.0.0:9200->9200/tcp, [::]:9200->9200/tcp,
 9300/tcp
                                                     elasticsearch
b7814a5a6e45   nginx:latest                                             "/docker-entrypo
int.…"   13 days ago      Up 4 days       0.0.0.0:80->80/tcp, [::]:80->80/tcp, 0.0.0.0
:443->443/tcp, [::]:443->443/tcp
                                                     nginx
7d115515cc25   sonarqube:8.6-community                                  "bin/run.sh bin/
sona…"   2 weeks ago      Up 10 days      0.0.0.0:9000->9000/tcp, [::]:9000->9000/tcp

                                                     sonarqube
39b5c3e4c230   jenkins/jenkins:lts                                      "/usr/bin/tini -
- /u…"   2 weeks ago      Up 10 days      50000/tcp, 0.0.0.0:8081->8080/tcp, [::]:8081
->8080/tcp
                                                     jenkins
c75db096d28a   obsidiandynamics/kafdrop:latest                          "/kafdrop.sh"
         2 weeks ago      Up 10 days      0.0.0.0:9500->9000/tcp, [::]:9500->9000/tcp

                                                     kafdrop
61a461f62104   confluentinc/cp-kafka:latest                             "/etc/confluent/
dock…"   2 weeks ago      Up 10 days      9092/tcp, 0.0.0.0:19093->19093/tcp, [::]:190
93->19093/tcp
                                                     kafka-2
20426d758d6d   confluentinc/cp-kafka:latest                             "/etc/confluent/
dock…"   2 weeks ago      Up 10 days      9092/tcp, 0.0.0.0:19092->19092/tcp, [::]:190
92->19092/tcp
                                                     kafka-1
761fd8cb0bec   confluentinc/cp-kafka:latest                             "/etc/confluent/
dock…"   2 weeks ago      Up 10 days      9092/tcp, 0.0.0.0:19094->19094/tcp, [::]:190
94->19094/tcp
                                                     kafka-3
359df14018e6   confluentinc/cp-zookeeper:latest                         "/etc/confluent/
dock…"   2 weeks ago      Up 10 days      2888/tcp, 0.0.0.0:2181->2181/tcp, [::]:2181-
>2181/tcp, 3888/tcp
                                                     zookeeper
8fee955a6a6e   redis                                                    "docker-entrypoi
nt.s…"   2 weeks ago      Up 10 days      0.0.0.0:6379->6379/tcp, [::]:6379->6379/tcp

                                                     redis-container
f14121499e23   postgres:15                                              "docker-entrypoi
nt.s…"   2 weeks ago      Up 10 days      0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp

                                                     postgres
```

🦥 SSL 인증서 설정

1. 도메인 준비
2. ZeroSSL 인증서 발급
3. Nginx 적용