포팅 메뉴얼

🛠️ 포팅 메뉴얼

▼ 1. 개발 환경

▼ 1.1 BackEnd

▼ JAVA - Spring

- **Java OpenJDK 17**

  > java.toolchain.languageVersion = 17

- **Spring Boot 3.4.3**

  > id 'org.springframework.boot' version '3.4.3'

- **Spring Web / Security / OAuth2 Client**

  > spring-boot-starter-web, spring-boot-starter-security, spring-boot-starter-oauth2-client

- **Spring Data JPA / Redis / Elasticsearch / WebFlux**

  > spring-boot-starter-data-jpa, spring-boot-starter-data-redis, spring-boot-starter-data-elasticsearch, spring-boot-starter-webflux

- **JWT (jjwt 0.11.5)**

  > io.jsonwebtoken:jjwt-api, -impl, -jackson

- **Validation**

  > spring-boot-starter-validation

- **Swagger (SpringDoc OpenAPI 2.2.0)**

  > springdoc-openapi-starter-webmvc-ui

- **Lombok**

  > compileOnly 'lombok' + annotationProcessor 'lombok'

- **Database: PostgreSQL**

  > runtimeOnly 'org.postgresql:postgresql'

- **S3 연동**

  > com.amazonaws:aws-java-sdk-s3:1.12.705

- **Hibernate Types (JSON, ENUM, Array)**

  > hibernate-types-60:2.21.1

- **Kafka 통신 처리**

  > spring-kafka

- **DevTools (개발 편의)**

  > spring-boot-devtools

- **테스트**

  > spring-boot-starter-test, spring-security-test, mockito-core

▼ 1.2 Database

- **PostgreSQL 17.4**

  > 주요 관계형 데이터 저장소

> Spring Data JPA 기반 ORM 매핑

- **Redis 7.4.2**

> 실시간 캐시, 좋아요 상태 저장, 분산 환경 구성에 활용
>
> `jemalloc` 메모리 관리 적용

- **Elasticsearch 8.12.2**

> 검색 기능 및 필터 처리
>
> RESTful API 연동 기반 검색 인덱싱 사용

▼ 1.3 Server / Infra

- **Ubuntu 22.04.5 LTS**

> 서비스 배포용 운영체제 (AWS EC2 기반)

- **Docker 26.1.3 / Docker Compose 2.34.0**

> Spring 서버, Redis, DB 등을 컨테이너화하여 구성 및 실행

- **Jenkins 2.501**

> GitLab Webhook 기반 CI/CD 자동화
>
> `docker-compose` 를 활용한 통합 배포 파이프라인 구성

▼ 1.4 Android

- Kotlin 2.1.10
- Gradle 8.9
- Android Gradle Plugin 8.7.3
- google-service 4.4.2
- compose 1.7.7
- Kakao SDK v2-user 2.12.0
- socket.io-client 2.1.1

▼ 1.5 IDE

- IntelliJ IDEA Community Edition 2024.3.2.2
- Visual Studio Code 1.97.2
- Android Studio 2024.2.2

▼ 2. 빌드 환경 구성

▼ 2.0 EC2 인스턴스에서 Jenkins 컨테이너 실행 명령어

```
sudo docker run -d -p 8080:8080 --name jenkins \
  -v /home/ubuntu/jenkins-data:/var/jenkins_home \
  -v /var/run/docker.sock:/var/run/docker.sock jenkins/jenkins:lts
```

▼ 2.1 Jenkins 파이프라인 설정 (Jenkinsfile)

```
pipeline {
    agent any

    options {
        skipDefaultCheckout(true)
    }


    environment {
        SPRING_IMAGE = "my-spring-app"
        REACT_IMAGE = "my-react-app"
        ANDROID_IMAGE = "my-android-app"
        NETWORK = "givu_nginx-network"
        KAFKA_NETWORK = "kafka-network"
        COMPOSE_FILE = "/var/jenkins_home/workspace/givu/docker-compose.yml"
        SPRINGBOOT_PORT = credentials('Springboot-Port')
    }

    stages {



        stage('Checkout') {
            steps {
                checkout scm
            }
        }

                stage('Start Infra Services') {
            steps {
                 sh "docker network create givu_nginx-network || true"
                 sh "docker-compose -f ${COMPOSE_FILE} up -d postgres
redis kafka kafka-ui elasticsearch kibana"

            }
        }


        stage('Build Spring Boot') {
            steps {
                dir('BE/givu') {
                    withCredentials([file(credentialsId: 'SPRING_APPLICATIO
N', variable: 'SPRING_YML')]) {
                        sh '''
                            echo "[INFO] 복사 중..."
                            cp $SPRING_YML src/main/resources/application.y
ml
                            echo "[DEBUG] application.yml 내용:"
                            cat src/main/resources/application.yml
                        '''
                    }

                sh 'chmod +x gradlew'
                sh './gradlew build -x test -Dspring.profiles.active=test --no
-daemon'

                // Docker build
                sh "docker build -t ${SPRING_IMAGE} -f Dockerfile ."
        }
    }
}
```

```
        stage('Build React') {
            steps {
                script {
                    withCredentials([file(credentialsId: 'REACT_ENV_FILE', va
riable:'REACT_ENV_PATH')]) {
                sh 'cp $REACT_ENV_PATH FE/GIVU/.env'
                sh 'echo "[DEBUG] .env 내용:"'
                sh 'cat FE/GIVU/.env'
            }

            sh "docker build -t ${REACT_IMAGE} -f FE/GIVU/Dockerfile F
E/GIVU"
        }
    }
}



        stage('Deploy App (Blue-Green)') {
            steps {
                script {
                    def nginxTemplatePath = "/home/ubuntu/nginx/nginx.te
mplate.conf"
                    def nginxConfPath = "/home/ubuntu/nginx/nginx.conf"

                    def backendNew = 'backend-v1'
                    def frontendNew = 'frontend-v1'

                    // 살아있는 게 v1인지 v2인지 확인해서 교체할 쪽으로 선택
                    if (sh(script: "docker ps --format '{{.Names}}' | grep bac
kend-v1 || true", returnStdout: true).trim() == 'backend-v1') {
                        backendNew = 'backend-v2'
                    }
                    if (sh(script: "docker ps --format '{{.Names}}' | grep fron
tend-v1 || true", returnStdout: true).trim() == 'frontend-v1') {
                        frontendNew = 'frontend-v2'
                    }
                    def backendPort = (backendNew == 'backend-v1') ? '111
5' : '1116'
                    def frontendPort = (frontendNew == 'frontend-v1') ? '30
00' : '3001'
                    // 이걸 기준으로 구버전 컨테이너 명확히 계산
                    def backendOld = (backendNew == 'backend-v1') ? 'bac
kend-v2' : 'backend-v1'
                    def frontendOld = (frontendNew == 'frontend-v1') ? 'fron
tend-v2' : 'frontend-v1'

                    // 새 컨테이너 실행
                    sh """
                        docker rm -f ${backendNew} || true
                        docker run -d --name ${backendNew} \
                            --network ${NETWORK} \
                            -e PORT=${SPRINGBOOT_PORT} \
                            -v /etc/localtime:/etc/localtime:ro \
                            -v /etc/timezone:/etc/timezone:ro \
                            -p ${backendPort}:8080 \
                            ${SPRING_IMAGE}

                            docker run -d --name ${frontendNew} \
                                --network ${NETWORK} \
                                -p ${frontendPort}:80 \
                                -v /home/ubuntu/nginx/front-nginx/react-default.
conf:/etc/nginx/conf.d/default.conf:ro \
                                ${REACT_IMAGE}
                    """

                    sleep time: 15, unit: 'SECONDS'

                    // nginx.conf 생성
                    def sedCommand = """
                        sed -e 's|\\\${BACKEND}|${backendNew}|g' \\
                            -e 's|\\\${FRONTEND}|${frontendNew}|g' \\
                            ${nginxTemplatePath} > ${nginxConfPath}
                    """
                    sh script: sedCommand
```

```
                // DNS가 등록될 때까지 대기 (최대 10번 시도)
                sh """
                for i in {1..10}; do
                docker run --rm --network ${NETWORK} busybox ping -
c 1 ${backendNew} && break
                echo "[⏳] ${backendNew} not ready, retrying..."
                sleep 2
                done

                for i in {1..10}; do
                docker run --rm --network ${NETWORK} busybox ping -
c 1 ${frontendNew} && break
                echo "[⏳] ${frontendNew} not ready, retrying..."
                sleep 2
                done
                """


                def nginxExists = sh(script: "docker ps -a --format '{{.Na
mes}}' | grep nginx || true", returnStdout: true).trim()

                def restartScript = """
                if [ "${nginxExists}" = "nginx" ]; then
                   docker restart nginx
                else
                   docker run -d --name nginx \
                      --network ${NETWORK} \
                      -p 80:80 -p 443:443 \
                      -v ${nginxConfPath}:/etc/nginx/nginx.conf:ro \
                      -v /home/ubuntu/nginx/empty:/etc/nginx/conf.d:ro \
                      -v /etc/letsencrypt:/etc/letsencrypt:ro \
                      nginx:latest
                fi
                """

                sh script: restartScript

                // 이전 컨테이너 제거
```

```
                    // nginx 재시작 후 이전 것 제거
                    sh """
                        docker stop ${backendOld} || true
                        docker rm ${backendOld} || true
                        docker stop ${frontendOld} || true
                        docker rm ${frontendOld} || true
                    """


                }
            }
        }
    }
}
```

2.2 Dockerfile 위치

구성 요소	Dockerfile 경로
Backend	`/BE/givu`
Front	`/FE/GIVU`
Android	`/Android/GIVU`

2.3 docker-compose.yml

> 주요 서비스: Kafka, Redis, Spring Backend, elasticsearch, kibana, postgres, nginx

```
version: '3.8'

services:
  # ---------------------
  # Kafka (KIP-500 mode)
  # ---------------------
  kafka:
    image: bitnami/kafka:3.9.0
    container_name: kafka
    environment:
      - KAFKA_CFG_NODE_ID=1
      - KAFKA_CFG_PROCESS_ROLES=broker,controller
    - KAFKA_CFG_CONTROLLER_QUORUM_VOTERS=1@kafka:9093
    - KAFKA_CFG_LISTENER_SECURITY_PROTOCOL_MAP=PLAINTEXT
    - KAFKA_CFG_LISTENERS=PLAINTEXT://:9092,CONTROLLER://:909
    - KAFKA_CFG_ADVERTISED_LISTENERS=PLAINTEXT://kafka:9092
    - KAFKA_CFG_CONTROLLER_LISTENER_NAMES=CONTROLLER
    - KAFKA_CFG_LISTENER_NAME_CONTROLLER_SSL_CLIENT_AUTH=
    - KAFKA_CFG_OFFSETS_TOPIC_REPLICATION_FACTOR=1
    - KAFKA_CFG_TRANSACTION_STATE_LOG_MIN_ISR=1
    - KAFKA_CFG_TRANSACTION_STATE_LOG_REPLICATION_FACTOR=
    - ALLOW_PLAINTEXT_LISTENER=yes
  ports:
    - "9092:9092"
    - "9093:9093"
  volumes:
    - kafka-data:/bitnami/kafka
  networks:
    - givu_nginx-network

kafka-ui:
  image: provectuslabs/kafka-ui:latest
  container_name: kafka-ui
  depends_on:
    - kafka
  ports:
    - "8081:8080"
  environment:
    - KAFKA_CLUSTERS_0_NAME=local
    - KAFKA_CLUSTERS_0_BOOTSTRAPSERVERS=kafka:9092
  networks:
    - givu_nginx-network

# ---------------------
# Elasticsearch + Kibana
# ---------------------
elasticsearch:
  image: docker.elastic.co/elasticsearch/elasticsearch:8.12.2
  container_name: elasticsearch
  environment:
    - discovery.type=single-node
    - xpack.security.enabled=false
  ports:
    - "9200:9200"
  volumes:
    - esdata:/usr/share/elasticsearch/data
  networks:
    - givu_nginx-network

kibana:
  image: docker.elastic.co/kibana/kibana:8.12.2
  container_name: kibana
  ports:
    - "5601:5601"
  environment:
    - ELASTICSEARCH_HOSTS=http://elasticsearch:9200
  depends_on:
    - elasticsearch
  networks:
    - givu_nginx-network

# ---------------------
# DB & Cache
# ---------------------
postgres:
  image: postgres:17
  container_name: postgres
  ports:
    - "5432:5432"
  environment:
    - POSTGRES_DB=givudb
    - POSTGRES_USER=d107
    - POSTGRES_PASSWORD=d107password
  volumes:
    - pgdata:/var/lib/postgresql/data
  networks:
    - givu_nginx-network
```

```
redis:
  image: redis
  container_name: redis
  ports:
    - "6379:6379"
  command: redis-server --requirepass d107password
  volumes:
    - redis-data:/data
  networks:
    - givu_nginx-network

# ---------------------
# App Containers (for local dev)
# ---------------------
backend-v1:
  image: my-spring-app
  container_name: backend-v1
  ports:
    - "1115:8080"
  environment:
    - PORT=8080
  networks:
    - givu_nginx-network

backend-v2:
  image: my-spring-app
  container_name: backend-v2
  ports:
    - "1116:8080"
  environment:
    - PORT=8080
  networks:
    - givu_nginx-network

frontend-v1:
  image: my-react-app
  container_name: frontend-v1
  ports:
      - "3000:80"
    networks:
      - givu_nginx-network

  frontend-v2:
    image: my-react-app
    container_name: frontend-v2
    ports:
      - "3001:80"
    networks:
      - givu_nginx-network

  nginx-local:
    image: nginx
    container_name: nginx-local
    ports:
      - "80:80"
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro
    networks:
      - givu_nginx-network

# ---------------------
# 네트워크 및 볼륨
# ---------------------
networks:
  givu_nginx-network:
    name: givu_nginx-network
    external: true

volumes:
  kafka-data:
  pgdata:
  redis-data:
  esdata:
```

▼ 2.4 사용 포트 번호

서비스	포트
Spring	1115
kibana	5601
elasticsearch	9200
kafka	9092
postgres	5432
Redis	6379
jenkins	8080

2.5 Android - API Key

`local.properties`

```
OPENAI_API_KEY='your_api_key'
```

▼ 2.6 nginx리버스 프록시 설정

- nginx.config

```
worker_processes auto;

events {
    worker_connections 1024;
}

http {
    # ✅ Backend Blue-Green
    upstream backend {
        server backend-v1:8080;
    }

    # ✅ Frontend Blue-Green
    upstream frontend {
        server frontend-v2:80;
    }

    # HTTP 서버 설정 - HTTP를 HTTPS로 리다이렉트
```

```
   server{
         listen 80;                               # HTTP 포트
         server_name j12d107.p.ssafy.io;             # 도메인 이름
         return 301 https://$server_name$request_uri;  # HTTPS로 영구
리다이렉트

   }

   server {
         listen 443 ssl;                           # HTTPS 포트
         server_name j12d107.p.ssafy.io;             # 도메인 이름



         client_max_body_size 20M;
      # SSL 인증서 설정
      ssl_certificate /etc/letsencrypt/live/j12d107.p.ssafy.io/fullchain.p
em;
      ssl_certificate_key /etc/letsencrypt/live/j12d107.p.ssafy.io/privke
y.pem;

      # ✅ Swagger UI
      location /swagger-ui/ {
         proxy_pass http://backend-v1:8080/swagger-ui/;
         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;
      }

      location /v3/api-docs {
         proxy_pass http://backend-v1:8080/v3/api-docs;
         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;
      }
```

```
        # ✅ Backend API
        location /api/ {
            proxy_pass http://backend-v1:8080;
            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;
        }

        # ✅ Frontend
         location / {
            proxy_pass http://frontend-v2:80;
            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;
        }
    }
}
```

설정 후 실행 명령어:

```
docker run -d \
  --name nginx \
  -p 80:80 -p 443:443 \
  -v /path/to/your/nginx.conf:/etc/nginx/nginx.conf \
  -v /etc/letsencrypt:/etc/letsencrypt \
  nginx
```

▼ 3. 주요 설정 파일 위치

3.1 Spring - application.yml

설정 일부:

```
spring:
  application:
    name: givu

  servlet:
    multipart:
      max-file-size: 5MB
      max-request-size: 30MB

  data:
    redis:
      host: redis
      port: 6379
      password: d107password

    # ✅ Elasticsearch 추가
  elasticsearch:
    uris: http://elasticsearch:9200
    connection-timeout: 3s
    socket-timeout: 3s
```

3.2 React - .env

```
VITE_BASE_URL=https://j12d107.p.ssafy.io/api
VITE_KAKAO_API_KEY=1780d7796c7c3dfe3212a35480508b06
VITE_KAKAO_REST_API_KEY=08cd78d237da6ad8828d168ac223e313
VITE_KAKAO_REDIRECT_URI=https://j12d107.p.ssafy.io/auth/kakao/callb
ack
VITE_API_BASE_URL=https://j12d107.p.ssafy.io/api
```

3.3 Android Studio

- `google-services.json` → `/Android/app` 경로에 위치

▼ 4. 외부 서비스 연동

카카오 소셜 로그인

- https://developers.kakao.com/docs/latest/ko/kakaologin/common

싸피 금융망 API

- project.ssafy → 개발자 센터 → SSAFY 오픈 API → SSAFY 금융망

▼ 5. Elasticsearch 매핑 설정

▼ Elastic 매핑 설정

```
products
{
  "settings": {
    "analysis": {
      "tokenizer": {
        "whitespace_tokenizer": {
          "type": "whitespace"
        }
      },
      "filter": {
        "edge_ngram_filter": {
          "type": "edge_ngram",
          "min_gram": 1,
          "max_gram": 20
        }
      },
      "analyzer": {
        "custom_ngram_analyzer": {
          "tokenizer": "whitespace_tokenizer",
          "filter": ["lowercase", "edge_ngram_filter"]
        }
      }
    }
  },
  "mappings": {
    "properties": {
      "id": {
        "type": "integer"
      },
      "productName": {
        "type": "text",
        "analyzer": "custom_ngram_analyzer",
        "search_analyzer": "standard"
      },
      "price": {
        "type": "integer"
      },
      "image": {
        "type": "text"
      },
      "favorite": {
        "type": "integer"
      },
      "star": {
        "type": "double"
      },
      "description": {
        "type": "text",
        "analyzer": "custom_ngram_analyzer",
        "search_analyzer": "standard"
      },
      "createdAt": {
        "type": "date",
        "format": "date_time"
      },
      "category": {
        "type": "keyword"
      }
    }
  }
}
---------
funding-index
{
  "settings": {
    "analysis": {
      "tokenizer": {
        "whitespace_tokenizer": {
```

```
        "type": "whitespace"
      }
    },
    "filter": {
      "edge_ngram_filter": {
        "type": "edge_ngram",
        "min_gram": 1,
        "max_gram": 20
      }
    },
    "analyzer": {
      "custom_ngram_analyzer": {
        "tokenizer": "whitespace_tokenizer",
        "filter": ["lowercase", "edge_ngram_filter"]
      }
    }
  }
},
"mappings": {
  "properties": {
    "title": {
      "type": "text",
      "analyzer": "custom_ngram_analyzer",
      "search_analyzer": "standard"
    },
    "description": {
      "type": "text",
      "analyzer": "custom_ngram_analyzer",
      "search_analyzer": "standard"
    }
  }
}
}
```