포팅 매뉴얼 - Fitmily 프로젝트

목차

1. 사용 도구

- 형상 관리: GitLab, Git
- 커뮤니케이션: MatterMost
- 이슈 관리: Jira
- CI/CD: Docker, Jenkins, Docker-compose
- 데이터베이스: MySQL, MongoDB, Redis

2. 개발 도구

- Backend: IntelliJ IDEA 2023.3.8
- Frontend: Android Studio 2024.1.1
- Server Management: Visual Studio Code 1.97.2

3. 개발 환경

OS

- Ubuntu: 22.04.5 LTS (EC2 서버)
- Windows 11 / macOS Ventura 13.0+ (개발 환경)

Backend

- Java OpenJDK: 17.0.15
- Spring Boot: 3.4.5
- Spring Security: 6.3.1
- Spring Data JPA
- Spring Data MongoDB
- Spring WebSocket: 6.2.2
- Gradle: 8.12
- JWT: 0.11.5
- Lombok: 1.18.36
- MyBatis

Android

- Kotlin: 2.0.21
- Android Gradle Plugin: 8.9.2
- Jetpack Compose: 2024.09.00
- Dagger Hilt: 2.56.2
- Krossbow STOMP (WebSocket): 9.3.0
- Moshi: 1.15.2
- Firebase Messaging: 24.1.1
- AndroidX Paging: 3.3.5

Database

- MySQL: 8.0
- MongoDB: 6.0
- Redis: 7.0

- AWS S3

Infra

- AWS EC2
- GitLab Webhook
- Docker: 26.1.3
- Docker-compose: 2.24.1
- Jenkins: 2.479.3
- Nginx: 1.27.5
- Certbot

포트 정보

- Backend: 8080
- MySQL: 3306
- MongoDB: 27017
- Redis: 6379
- Jenkins: 8080
- RabbitMQ: 5672
- Nginx: 80/443

4. 환경 변수

application-prod.yml (.gitignore로 보안 관리)

```
spring:
  application:
    name: fitmily

  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url:
```

```
jdbc:mysql://k12d208.p.ssafy.io:3306/mydb?useSSL=false&useUnicode=true&serverTimezone=Asia/Seoul&a
llowPublicRetrieval=true
    username:
    password:
    hikari:
      connection-test-query: SELECT 1
      maximum-pool-size: 10
      connection-timeout: 30000

  data:
    mongodb:
      host: mongodb
      port: 27017
      database: fitmily

    redis:
      host: redis
      port: 6379

  rabbitmq:
    host: rabbitmq
    port: 5672
    username: guest
    password: guest

  jwt:
    secret:

  cloud:
    aws:
      region:
        static:
        auto: false
      credentials:
        accessKey:
        secretKey:
      s3:
        bucket:

mybatis:
  mapper-locations: classpath*:mappers/*.xml
  type-aliases-package: com.d208.fitmily.domain
  configuration:
    map-underscore-to-camel-case: true
    use-column-label: true
    auto-mapping-behavior: PARTIAL
    return-instance-for-empty-row: true
    cache-enabled: false
    call-setters-on-nulls: true

springdoc:
  api-docs:
    path: /v3/api-docs
    enabled: true
  swagger-ui:
    path: /swagger-ui.html
    config-url: /v3/api-docs/swagger-config
    enabled: true
  packages-to-scan: com.d208.fitmily

server:
  port: 8080
  address: 0.0.0.0
  forward-headers-strategy: FRAMEWORK
```

.env (GitLab CI/CD 환경변수)

```
# GitLab 설정
GITLAB_TOKEN=

# EC2 배포 정보
EC2_HOST=k12d208.p.ssafy.io
EC2_USER=ubuntu

# Jenkins 설정
JENKINS_URL=http://k12d208.p.ssafy.io:8080
JENKINS_USER=admin
JENKINS_TOKEN=
# MySQL
MYSQL_ROOT_PASSWORD=
MYSQL_DATABASE=mydb
MYSQL_USER=d208
MYSQL_PASSWORD=

# MongoDB
MONGO_INITDB_ROOT_USERNAME=fitmily
MONGO_INITDB_ROOT_PASSWORD=
MONGO_INITDB_DATABASE=fitmily

# Spring
SPRING_PROFILES_ACTIVE=prod

# S3
AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=
S3_BUCKET_NAME=
```

Android local.properties (gitignore)

```
sdk.dir=C₩:₩₩Users₩₩USERNAME₩₩AppData₩₩Local₩₩Android₩₩Sdk
BACKEND_BASE_URL=https://k12d208.p.ssafy.io
WEBSOCKET_URL=wss://k12d208.p.ssafy.io/api/ws-connect
```

5. CI/CD 구축

기본 설정

AWS EC2 접속

```
# ssh -i [pem키] [접속 계정]@[접속 도메인]
ssh -i K12D208T.pem ubuntu@k12d208.p.ssafy.io
```

업데이트

```
sudo apt update
sudo apt upgrade -y
sudo apt install -y build-essential
```

ufw 포트 설정

```
sudo ufw status
sudo ufw allow 8080  # Jenkins
sudo ufw allow 80    # HTTP
sudo ufw allow 443   # HTTPS
sudo ufw allow 8000  # Spring Boot
sudo ufw allow 3306  # MySQL
sudo ufw allow 27017 # MongoDB
sudo ufw allow 6379  # Redis
sudo ufw allow 5672  # RabbitMQ
sudo ufw show added
```

Docker & Docker-compose 설치

```
# 필요한 패키지 설치
sudo apt-get install -y ₩
  apt-transport-https ₩
  ca-certificates ₩
  curl ₩
  gnupg-agent ₩
  software-properties-common

# Docker 공식 GPG 키 추가
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

# Docker 레포지토리 추가
sudo add-apt-repository ₩
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu ₩
   $(lsb_release -cs) ₩
   stable"
```

```
# Docker 엔진 설치
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io

# Docker 서비스 시작
sudo systemctl start docker
sudo systemctl enable docker

# 현재 사용자를 docker 그룹에 추가 (sudo 없이 docker 명령어 실행 가능)
sudo usermod -aG docker $USER

# Docker Compose 설치
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose

# 설치 확인
docker --version
docker-compose --version
```

Docker-compose.yml

```
version: '3.8'

services:
  springboot-app:
    build:
      context: ./fitmily_backend/user-service
      dockerfile: Dockerfile
    container_name: springboot-app
    ports:
      - "8080:8080"
    environment:
      - SPRING_PROFILES_ACTIVE=prod
    networks:
      - app-network
    depends_on:
      - mysql
      - mongodb
      - redis
      - rabbitmq
    restart: always

  mysql:
    image: mysql:8.0
    container_name: mysql
    ports:
      - "3306:3306"
    environment:
      - MYSQL_ROOT_PASSWORD=
      - MYSQL_DATABASE=mydb
      - MYSQL_USER=d208
      - MYSQL_PASSWORD=
    volumes:
      - mysql-data:/var/lib/mysql
    networks:
      - app-network
    restart: always

  mongodb:
    image: mongo:6.0
    container_name: mongodb
    ports:
      - "27017:27017"
    environment:
      - MONGO_INITDB_ROOT_USERNAME=fitmily
      - MONGO_INITDB_ROOT_PASSWORD=
      - MONGO_INITDB_DATABASE=fitmily
    volumes:
      - mongo-data:/data/db
    networks:
      - app-network
    restart: always

  redis:
    image: redis:7.0
    container_name: redis
    ports:
      - "6379:6379"
    volumes:
      - redis-data:/data
    networks:
      - app-network
    restart: always

  rabbitmq:
    image: rabbitmq:3-management
    container_name: rabbitmq
    ports:
      - "5672:5672"
      - "15672:15672"
    networks:
      - app-network
    restart: always

  nginx:
    image: nginx:1.27.5
    container_name: nginx
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx/conf.d:/etc/nginx/conf.d
      - /etc/letsencrypt:/etc/letsencrypt:ro
    networks:
      - app-network
    depends_on:
      - springboot-app
    restart: always

  prometheus:
    image: prom/prometheus:latest
    container_name: prometheus
    ports:
      - "9090:9090"
    volumes:
      - ./prometheus:/etc/prometheus
      - prometheus-data:/prometheus
    networks:
      - app-network
    restart: always

  grafana:
    image: grafana/grafana:latest
    container_name: grafana
    ports:
      - "3000:3000"
    volumes:
      - grafana-data:/var/lib/grafana
    networks:
      - app-network
    restart: always

  jenkins:
    image: jenkins/jenkins:lts
    container_name: jenkins
    ports:
      - "8080:8080"
      - "50000:50000"
    volumes:
      - jenkins_home:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
    networks:
      - app-network
    restart: always

networks:
  app-network:
    driver: bridge

volumes:
  mysql-data:
  redis-data:
  mongo-data:
  prometheus-data:
  grafana-data:
  jenkins_home:
```

Dockerfile 설정

Backend (Spring Boot)

```
FROM eclipse-temurin:17-jdk

WORKDIR /app

COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src ./src

RUN chmod +x ./gradlew
RUN ./gradlew clean build -x test

EXPOSE 8080
ENTRYPOINT ["java", "-jar", "/app/build/libs/*.jar"]
```

Nginx 설정 (nginx/conf.d/default.conf)

```
map $http_upgrade $connection_upgrade {
    default upgrade;
    '' close;
}

server {
    listen 80;
    server_name k12d208.p.ssafy.io;

    # HTTP를 HTTPS로 리디렉션
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl;
    server_name k12d208.p.ssafy.io;
```

```
# SSL 설정
ssl_certificate /etc/letsencrypt/archive/fullchain1.pem;
ssl_certificate_key /etc/letsencrypt/archive/privkey1.pem;
ssl_protocols TLSv1.2 TLSv1.3;
ssl_prefer_server_ciphers off;

# 웹소켓 라우팅
location ^~ /api/ws-connect {
    proxy_pass http://springboot-app:8080;

    # WebSocket 필수 설정
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "Upgrade";

    # 추가 WebSocket 최적화 설정
    proxy_read_timeout 3600s;
    proxy_send_timeout 3600s;
    proxy_buffering off;

    # 표준 프록시 헤더
    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;
}

# 스프링부트 앱으로 요청 전달
location /api/ {
    proxy_pass http://springboot-app: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;
}

# Swagger UI 접근 허용
location /swagger-ui/ {
    proxy_pass http://springboot-app: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;
        proxy_set_header X-Forwarded-Prefix "";
    }

    # API 문서 접근
    location /v3/api-docs {
        proxy_pass http://springboot-app: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-Prefix "";
    }

    # Swagger Config 접근
    location /v3/api-docs/swagger-config {
        proxy_pass http://springboot-app:8080/v3/api-docs/swagger-config;
        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-Prefix "";
    }

    # Jenkins 접근 설정
    location /jenkins/ {
        proxy_pass http://jenkins: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;
        proxy_set_header X-Forwarded-Port 443;
        proxy_set_header X-Forwarded-Host $host;
    }
}
```

SSL 인증서 발급 (Certbot)

```
sudo apt-get update
sudo apt-get install python3-certbot-nginx
sudo certbot certonly --nginx -d k12d208.p.ssafy.io
```

Jenkins 설정

```
# Jenkins 볼륨 생성
docker volume create jenkins_home

# Jenkins 컨테이너 실행
docker run -d ₩
  --name jenkins-docker ₩
  --restart=unless-stopped ₩
  -p 8080:8080 ₩
  -p 50000:50000 ₩
  -v jenkins_home:/var/jenkins_home ₩
  -v /var/run/docker.sock:/var/run/docker.sock ₩
  jenkins/jenkins:lts

# 초기 비밀번호 확인
docker exec jenkins-docker cat /var/jenkins_home/secrets/initialAdminPassword
```

Jenkins Pipeline 설정 (Jenkinsfile)

```
pipeline {
    agent any

    environment {
        GIT_AUTHOR_ID = ''
        GIT_AUTHOR_EMAIL = ''
    }

    options {
        skipDefaultCheckout(true)
    }

    stages {
        stage('Clean Workspace') {
            steps {
                cleanWs()
            }
        }

        stage('Checkout') {
            steps {
                checkout([
                    $class: 'GitSCM',
                    branches: [[name: 'backend']],
                    extensions: [
                        [$class: 'CleanBeforeCheckout'],
                        [$class: 'CloneOption', depth: 0, noTags: false, reference: '', shallow: false]
                    ],
                    userRemoteConfigs: [[credentialsId: 'gitlab-credentials', url: 'https://lab.ssafy.com/s12-
final/S12P31D208.git']]
                ])

                script {
                    def authorName = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                    def authorEmail = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()

                    env.GIT_AUTHOR_NAME = authorName
                    env.GIT_AUTHOR_EMAIL = authorEmail

                    echo "Git Author: ${env.GIT_AUTHOR_NAME}, Email: ${env.GIT_AUTHOR_EMAIL}"
                }
            }
        }

        stage('Build SpringBoot') {
            steps {
                dir('fitmily_backend/user-service') {
                    sh 'chmod +x ./gradlew'
                    sh './gradlew clean build -x test'
                }
            }
        }
        stage('Docker Build & Deploy') {
            steps {
                dir('fitmily_backend') {
                    sh 'docker-compose build --no-cache'
                    sh 'docker-compose up -d'
                }
            }
        }

        stage('Update Nginx Config') {
            steps {
                dir('fitmily_backend') {
                    sh 'echo "Applying latest Nginx configuration..."'
                    sh 'ls -la nginx/conf.d/'
                    sh 'docker-compose restart nginx'
                }
            }
        }

        stage('Health Check') {
            steps {
                sh 'sleep 15'
                sh 'curl -f http://localhost:8080/actuator/health || true'
            }
        }
    }

    post {
        success {
            script {
                mattermostSend (
                    color: 'good',
                    message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by
${env.GIT_AUTHOR_NAME}(${env.GIT_AUTHOR_EMAIL})₩n(<${env.BUILD_URL}|Details>)",
                    endpoint: 'https://meeting.ssafy.com/hooks/93zyedse8b8m7dmkjfwk6rhtfo',
                    channel: 'D208-Build-Bot'
                )
            }
        }
        failure {
            script {
                mattermostSend (
                    color: 'danger',
                    message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by
${env.GIT_AUTHOR_NAME}(${env.GIT_AUTHOR_EMAIL})₩n(<${env.BUILD_URL}|Details>)",
                    endpoint: 'https://meeting.ssafy.com/hooks/93zyedse8b8m7dmkjfwk6rhtfo',
                    channel: 'D208-Build-Bot'
                )
            }
        }
        always {
            cleanWs()
        }
    }
}
```

6. 외부 서비스 사용

- AWS S3: 파일 저장소
- Firebase Cloud Messaging: 안드로이드 푸시 알림
- AWS EC2: 서버 호스팅
- GitLab CI/CD: 자동 빌드 및 배포
- Swagger: API 문서화 및 테스트

7. Android 앱 배포

앱 빌드 및 서명

1. Android Studio에서 app 모듈 선택
2. Build > Generate Signed Bundle/APK 선택
3. APK 선택 후 다음
4. 키 스토어 정보 입력 (신규 생성 또는 기존 키 사용)
5. 릴리즈 빌드 설정 후 Finish

Google Play 스토어 배포

1. Google Play Console에 로그인
2. 앱 선택 또는 신규 앱 생성
3. 앱 릴리즈 > 프로덕션 선택
4. 새 릴리즈 만들기
5. APK/AAB 파일 업로드
6. 릴리즈 노트 작성
7. 롤아웃 설정 후 검토 > 출시

FCM 설정

1. Firebase 콘솔에서 프로젝트 생성
2. 앱 등록 (패키지명 입력)
3. google-services.json 다운로드
4. 앱 프로젝트의 app 디렉토리에 파일 배치
5. 앱 모듈의 build.gradle에 Firebase SDK 추가

```
// 앱 모듈의 build.gradle (or build.gradle.kts)
plugins {
    id("com.google.gms.google-services")
}

dependencies {
    implementation(libs.firebase.messaging.ktx)
}
```

⁂