포팅 매뉴얼

1. 개요

주식 종목 관련 뉴스를 분석하여 호재/악재 여부를 실시간으로 알려주는 서비스

AI 기반 주식 종목 별 뉴스 분석·알림 및 뉴스레터 제공 서비스

바쁜 현대 사회 속 당신이 놓친 주식 뉴스들을 찾아드립니다. 뉴스톡.

2. 사용 도구

- 이슈 관리 : Jira
- 형상 관리 : GitLab
- 커뮤니케이션 : Notion, MatterMost
- 디자인 : Figma
- CI/CD : GitLab Runner, Docker Compose

3. 개발 도구

- Intellij : 2024.3.1.1
- Android Studio : 2024.2.2
- Visual Studio Code: 1.99.0

4. 개발 환경

4-1. Front(React Native)

항목	버전
Android Studio	latest stable
Node.JS	latest stable

Dependencies

▼ package.json

```
{
  "name": "NewStock",
  "version": "1.0.0",
  "scripts": {
    "android": "expo start --android",
    "ios": "expo start --ios",
    "start": "expo start",
    "prebuild": "expo prebuild",
    "lint": "eslint \"**/*.{js,jsx,ts,tsx}\" && prettier -c \"**/*.{js,jsx,ts,tsx,json}\"
    "format": "eslint \"**/*.{js,jsx,ts,tsx}\" --fix && prettier \"**/*.{js,jsx,ts,tsx
    "web": "expo start --web"
  },
  "dependencies": {
    "@expo/vector-icons": "^14.0.4",
    "@stomp/stompjs": "^7.1.0",
    "@tanstack/react-query": "^5.69.0",
    "axios": "^1.8.4",
    "clsx": "^2.1.1",
    "expo": "^52.0.40",
    "expo-blur": "^14.0.3",
    "expo-constants": "~17.0.8",
    "expo-device": "~7.0.3",
    "expo-linear-gradient": "~14.0.2",
    "expo-linking": "~7.0.5",
    "expo-notifications": "~0.29.14",
    "expo-router": "~4.0.19",
    "expo-secure-store": "~14.0.1",
    "expo-splash-screen": "~0.29.22",
    "expo-status-bar": "~2.0.1",
    "miragejs": "^0.1.48",
    "nativewind": "^4.1.23",
    "react": "18.3.1",
    "react-native": "0.76.7",
    "react-native-calendars": "^1.1310.0",
    "react-native-collapsible": "^1.6.2",
```

```
    "react-native-draggable-flatlist": "^4.0.2",
    "react-native-gesture-handler": "^2.25.0",
    "react-native-gifted-charts": "^1.4.59",
    "react-native-markdown-display": "^7.0.2",
    "react-native-safe-area-context": "4.12.0",
    "react-native-screens": "~4.4.0",
    "react-native-svg": "15.8.0",
    "react-native-toast-message": "^2.2.1",
    "react-native-vector-icons": "^10.2.0",
    "react-native-webview": "^13.12.5",
    "react-native-word-cloud": "^1.0.7",
    "tailwind-merge": "^3.0.2",
    "uri-scheme": "^1.4.0",
    "victory-native": "^37.3.6",
    "ws": "^8.18.1",
    "zustand": "^5.0.3"
  },
  "devDependencies": {
    "@babel/core": "^7.20.0",
    "@types/react": "~18.3.12",
    "@types/react-native": "^0.72.8",
    "@types/ws": "^8.18.0",
    "@typescript-eslint/eslint-plugin": "^7.7.0",
    "@typescript-eslint/parser": "^7.7.0",
    "eslint": "^8.57.0",
    "eslint-config-universe": "^12.0.1",
    "prettier": "^3.2.5",
    "prettier-plugin-tailwindcss": "^0.5.11",
    "react-native-svg-transformer": "^1.5.0",
    "tailwindcss": "^3.4.0",
    "typescript": "~5.3.3"
  },
  "eslintConfig": {
    "extends": "universe/native",
    "root": true
  },
  "main": "index.js",
```

```
  "private": true
}
```

주요 라이브러리	버전
react	18.3.1
react-native	0.76.7
expo	52.0.40
expo-router	4.0.19
@expo/vector-icons	14.0.4
axios	1.8.4
@tanstack/react-query	5.69.0
zustand	5.0.3
nativewind	4.1.23
miragejs	0.1.48
typescript	5.3.3

4-2. Backend (Spring Boot)

항목	버전
Java (OpenJDK)	17
Spring Boot	3.3.9
Gradle	7.x 이상
Spring Dependency Management	1.1.7

Dependencies

라이브러리	버전
Spring Boot Starter Web	최신 안정 버전
Spring Boot Starter Webflux	최신 안정 버전
Spring Boot Starter JPA	최신 안정 버전
Spring Boot Starter Security	최신 안정 버전
Spring Boot Starter Mail	최신 안정 버전

Spring Boot Starter Thymeleaf	최신 안정 버전
Spring Boot Kafka	최신 안정 버전
QueryDSL	5.0.0 (jakarta)
MySQL Connector	8.x
Swagger (SpringDoc OpenAPI)	2.2.0
JsonWebToken	0.11.5
Hibernate-Validator	8.0.1
Validator-Api	3.0.2
Jsoup	1.15.4
Google firebase	9.2.0
Lombok	최신 안정 버전
JUnit (테스트 프레임워크)	최신 안정 버전

4-3. AI (Fast API)

- requirements.txt

```
annotated-types==0.7.0
anyio==4.9.0
blinker==1.9.0
certifi==2025.1.31
charset-normalizer==3.4.1
click==8.1.8
colorama==0.4.6
exceptiongroup==1.2.2
fastapi==0.115.12
filelock==3.18.0
Flask==3.1.0
fsspec==2025.3.0
h11==0.14.0
huggingface-hub==0.29.3
idna==3.10
importlib_metadata==8.6.1
itsdangerous==2.2.0
Jinja2==3.1.6
```

```
joblib==1.4.2
kobert-tokenizer @ git+https://github.com/SKTBrain/KoBERT.git@5c46b1c
MarkupSafe==3.0.2
mpmath==1.3.0
networkx==3.2
nltk==3.9.1
numpy==2.0.2
packaging==24.2
pandas==2.2.3
protobuf==3.20.3
pydantic==2.10.6
pydantic_core==2.27.2
python-dateutil==2.9.0.post0
python-dotenv==1.0.1
pytz==2025.2
PyYAML==6.0.2
regex==2024.11.6
requests==2.32.3
safetensors==0.5.3
sentencepiece==0.2.0
six==1.17.0
sniffio==1.3.1
starlette==0.46.1
sympy==1.13.1
tokenizers==0.21.1
torch==2.6.0
tqdm==4.67.1
transformers==4.50.0
typing_extensions==4.12.2
tzdata==2025.2
urllib3==2.3.0
uvicorn==0.34.0
Werkzeug==3.1.3
zipp==3.21.0
krwordrank==1.0.3
konlpy==0.6.0
```

5. 외부 프로그램

- **FireBase**
  - Cloud Messaging
    - Firebase 콘솔 프로젝트 만들어 사용
- **카카오**
  - OAuth
    - 카카오 디벨로퍼스에서 내 앱을 만들어 사용
- **한국투자증권**
  - 실시간 등락률 웹소켓 사용
    - 한국투자증권 키 연결하여 사용
- **Chat GPT**
  - 뉴스레터용 콘텐츠 만들 때 사용
    - GPT 키 연결하여 사용

6. 환경변수 형태

- **Backend**
  - **application.properties**

```
spring.config.import=optional:file:.env[.properties]
spring.application.name=${APP_NAME}

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

# JPA
spring.jpa.hibernate.ddl-auto=${JPA_DDL_AUTO}
```

```
spring.jpa.show-sql=${JPA_SHOW_SQL}
spring.jpa.properties.hibernate.format_sql=${JPA_FORMAT_SQL}
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8D

server.servlet.context-path=${SERVER_CONTEXT_PATH}
spring.web.resources.add-mappings=true

# JWT
jwt.secret=${JWT_SECRET}

# Kakao
kakao.client-id=${KAKAO_CLIENT_ID}
kakao.redirect-uri=${KAKAO_REDIRECT_URI}
kakao.redirect-url=${KAKAO_REDIRECT_URL}

# Redis
spring.redis.host=${REDIS_HOST}
spring.redis.port=${REDIS_PORT}

# Lettuce
spring.data.redis.connect-timeout=2s
spring.data.redis.lettuce.pool.max-wait=2s

# fast api
news.ai.url=${NEWS_AI_URL}

# OpenAi
openai.api.key=${OPENAI_API_KEY}

# KIS
kis.app-key=${KIS_APP_KEY}
kis.secret-key=${KIS_SECRET_KEY}

# Gmail SMTP
spring.mail.host=${MAIL_HOST}
spring.mail.port=${MAIL_PORT}
spring.mail.username=${MAIL_USERNAME}
spring.mail.password=${MAIL_PASSWORD}
```

```
# SMTP
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true

spring.mail.default-encoding=UTF-8

logging.level.org.hibernate.SQL=DEBUG
logging.level.org.hibernate.type.descriptor.sql.BasicBinder=trace
logging.level.org.springframework.security=DEBUG

fcm.project-id=${FCM_PROJECT_ID}
fcm.service-account-file=firebase-service-account.json
```

- **.env**

```
APP_NAME=newstock

# MySQL 설정
MYSQL_HOST={MySQL 서버 주소}
MYSQL_PORT={PORT}
MYSQL_DATABASE=newstock
MYSQL_USERNAME={MySQL UserName}
MYSQL_PASSWORD={MySQL Password}

# JPA 설정
JPA_DDL_AUTO=update
JPA_SHOW_SQL=true
JPA_FORMAT_SQL=true

# 서버 설정
SERVER_CONTEXT_PATH=/api

# 카카오
KAKAO_CLIENT_ID={KAKAO CLIENT ID}
KAKAO_REDIRECT_URI={KAKAO REDIRECT URI}
KAKAO_REDIRECT_URL={KAKAO REDIRECT URL}
```

```
# 카프카
KAFKA_BOOTSTRAP_SERVERS={KAFKA 서버 주소}

# Fast API
NEWS_AI_URL={AI 서버 주소}

# JWT
JWT_SECRET={JWT SECRET}

# 한국투자증권
KIS_APP_KEY={한국투자증권 APP KEY}
KIS_SECRET_KEY={한국투자증권 SECRET KEY}

# Redis
REDIS_HOST={REDIS 서버 주소}
REDIS_PORT={REDIS PORT}
REDIS_TIMEOUT=60000

# FCM
FCM_PROJECT_ID={Firebase 프로젝트 ID}
FCM_SERVICE_ACCOUNT_FILE={FCM_SERVICE_ACCOUNT_FILE}

# Gmail
MAIL_HOST=smtp.gmail.com
MAIL_PORT=587
MAIL_USERNAME={MAIL_USER_NAME}
MAIL_PASSWORD={MAIL_PASSWORD}

# Open AI
OPENAI_API_KEY={OPEN_API_KEY}
```

- **FrontEnd**
  - .env

```
EXPO_PUBLIC_API_URL={백엔드 배포 주소}
EXPO_PUBLIC_API_VERSION={백엔드 API 버전} // api/v1
EXPO_PUBLIC_KAKAO_CLIENT_ID={카카오 OAuth client id}
```

7. CI/CD 구축

1. 프로젝트 구조 개요

프로젝트는 다음과 같은 주요 구성 요소로 이루어져 있습니다:

- `backend/`: Spring Boot 백엔드 애플리케이션
- `nginx/`: Nginx 웹 서버 설정
- `.gitlab-ci.yml`: GitLab CI/CD 파이프라인 설정
- `docker-compose.yml`: Docker 컨테이너 구성

2. 개발자 코드 푸시 프로세스

2.1 코드 변경 및 푸시

개발자가 로컬에서 코드를 수정하고 GitLab 저장소에 푸시합니다:

```
git add .
git commit -m "feat: API 기능 개선"
git push origin feat/infra  # 현재 CI/CD 파이프라인은 feat/infra 브랜치에 설정됨
```

3. GitLab CI/CD 파이프라인 실행

`.gitlab-ci.yml` 파일에 정의된 파이프라인이 실행됩니다 ← 이 작업을 깃랩 러너가 하는거임

깃랩이 이 파일을 읽고 저 특정 태그가 걸린 러너를 실행시킴

```
stages:
  - deploy

variables:
  # Docker 관련 변수 제거 (Runner의 Docker를 사용)

deploy:
  stage: deploy
  tags:
    - newstock  # 특정 Runner 태그 지정
  image: alpine:latest
```

```
  before_script:
    - apk add --no-cache openssh-client
    - eval $(ssh-agent -s)
    - echo "$SSH_PRIVATE_KEY" | tr -d '\\r' | ssh-add -
    - mkdir -p ~/.ssh
    - chmod 700 ~/.ssh
    - echo "$SSH_KNOWN_HOSTS" >> ~/.ssh/known_hosts
    - chmod 644 ~/.ssh/known_hosts
  script:
    - |
      ssh ubuntu@$SERVER_IP "
        cd /home/ubuntu/S12P21A304 &&
        git pull &&

        # 환경 변수 파일 생성
        echo \\"APP_NAME=$APP_NAME\\" > backend/.env &&
        echo \\"MYSQL_HOST=$MYSQL_HOST\\" >> backend/.env &&
        echo \\"MYSQL_PORT=$MYSQL_PORT\\" >> backend/.env &&
        echo \\"MYSQL_DATABASE=$MYSQL_DATABASE\\" >> backend/.env
&&
        echo \\"MYSQL_USERNAME=$MYSQL_USERNAME\\" >> backend/.env
&&
        echo \\"MYSQL_PASSWORD=$MYSQL_PASSWORD\\" >> backend/.en
v &&
        echo \\"JPA_DDL_AUTO=$JPA_DDL_AUTO\\" >> backend/.env &&
        echo \\"JPA_SHOW_SQL=$JPA_SHOW_SQL\\" >> backend/.env &&
        echo \\"JPA_FORMAT_SQL=$JPA_FORMAT_SQL\\" >> backend/.env &&
        echo \\"SERVER_CONTEXT_PATH=$SERVER_CONTEXT_PATH\\" >> ba
ckend/.env &&

        # 루트 .env 파일도 생성 (docker-compose에서 사용)
        echo \\"MYSQL_DATABASE=$MYSQL_DATABASE\\" > .env &&
        echo \\"MYSQL_USERNAME=$MYSQL_USERNAME\\" >> .env &&
        echo \\"MYSQL_PASSWORD=$MYSQL_PASSWORD\\" >> .env &&
        echo \\"MYSQL_ROOT_PASSWORD=$MYSQL_ROOT_PASSWORD\\" >
> .env &&

        # 서버에서 직접 빌드 및 배포
```

```
      docker-compose build &&
      docker-compose down &&
      docker-compose up -d
    "
  only:
    - feat/infra
```

3.1 파이프라인 단계별 진행

1. **SSH 클라이언트 설정**: Alpine 컨테이너에 SSH 클라이언트 설치 및 설정
2. **SSH 키 설정**: GitLab에 저장된 개인 키 변수(`$SSH_PRIVATE_KEY`)를 사용하여 SSH 인증 설정
3. **서버 연결 및 명령 실행**: 환경 변수(`$SERVER_IP`)를 사용하여 서버에 SSH로 연결하고 다음 작업 수행:
   - 프로젝트 디렉토리로 이동
   - 최신 코드 가져오기
   - 환경 변수 파일 생성 (두 가지: 백엔드용, docker-compose용)
   - Docker 이미지 빌드 및 컨테이너 재시작

4. 서버에서의 배포 과정

4.1 최신 코드 가져오기

```
cd /home/ubuntu/S12P21A304
git pull
```

4.2 환경 변수 파일 생성

`backend/.env` 파일에 Spring Boot 애플리케이션에 필요한 환경 변수가 저장됩니다:

```
APP_NAME=MobileApp
MYSQL_HOST=mysql
MYSQL_PORT=3306
MYSQL_DATABASE=appdb
MYSQL_USERNAME=appuser
MYSQL_PASSWORD=secretpassword
```

```
JPA_DDL_AUTO=update
JPA_SHOW_SQL=false
JPA_FORMAT_SQL=false
SERVER_CONTEXT_PATH=/api
```

루트 디렉토리의 `.env` 파일에 Docker Compose에서 사용할 환경 변수가 저장됩니다:

```
MYSQL_DATABASE=appdb
MYSQL_USERNAME=appuser
MYSQL_PASSWORD=secretpassword
MYSQL_ROOT_PASSWORD=rootpassword
```

4.3 Docker 이미지 빌드

```
docker-compose build
```

이 명령은 `docker-compose.yml` 파일에 정의된 서비스 중 빌드 설정이 있는 서비스의 이미지를 빌드합니다:

```
version: '3'

services:
  nginx:
    build:
      context: ./nginx
      dockerfile: Dockerfile
    ports:
      - "80:80"
      - "443:443"
    depends_on:
      - backend
    restart: always
    volumes:
      - ./nginx/default.conf:/etc/nginx/conf.d/default.conf
      - /opt/certs:/certs:ro

  backend:
    build:
```

```
    context: ./backend
    dockerfile: Dockerfile
  ports:
    - "8080:8080"
  environment:
    - SPRING_PROFILES_ACTIVE=prod
  env_file:
    - ./.env
  restart: always
  depends_on:
    - mysql

 mysql:
  image: mysql:8.0
  ports:
    - "3306:3306"
  volumes:
    - db-data:/var/lib/mysql
  environment:
    - MYSQL_DATABASE=${MYSQL_DATABASE}
    - MYSQL_USER=${MYSQL_USERNAME}
    - MYSQL_PASSWORD=${MYSQL_PASSWORD}
    - MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
  restart: always

volumes:
 db-data:
```

4.3.1 백엔드 이미지 빌드

`backend/Dockerfile`을 사용하여 Spring Boot 애플리케이션이 빌드됩니다:

```
FROM eclipse-temurin:17-jre

WORKDIR /app

COPY build/libs/*.jar app.jar
```

```
ENTRYPOINT ["java", "-jar", "app.jar"]
```

빌드 과정:

1. Eclipse Temurin JRE 17 이미지를 기반으로 컨테이너 생성
2. 작업 디렉토리 설정
3. 빌드된 JAR 파일을 컨테이너에 복사
4. 실행 명령 설정

4.3.2 Nginx 이미지 빌드

`nginx/Dockerfile`을 사용하여 Nginx 웹 서버가 빌드됩니다:

```
FROM nginx:latest

# 기본 설정 파일 제거
RUN rm /etc/nginx/conf.d/default.conf

# 사용자 정의 설정 파일 복사
COPY default.conf /etc/nginx/conf.d/

# 포트 노출
EXPOSE 80 443

# Nginx 실행
CMD ["nginx", "-g", "daemon off;"]
```

빌드 과정:

1. 공식 Nginx 이미지를 기반으로 컨테이너 생성
2. 기본 설정 파일 제거
3. 사용자 정의 설정 파일 복사
4. 포트 노출 (80, 443)
5. Nginx 실행 명령 설정

4.4 컨테이너 중지 및 재시작

```
docker-compose down
docker-compose up -d
```

`docker-compose down` 명령은 실행 중인 모든 컨테이너를 중지하고 제거합니다.

`docker-compose up -d` 명령은 새로 빌드된 이미지를 사용하여 컨테이너를 시작합니다.

5. 각 서비스의 배포 및 초기화 과정

5.1 MySQL 서비스

MySQL 8.0 이미지를 사용하여 데이터베이스 서비스가 시작됩니다:

- 환경 변수: `.env` 파일에서 가져온 데이터베이스 설정
- 볼륨: `db-data` 볼륨을 사용하여 데이터 영속성 보장
- 포트: 3306 포트 노출

MySQL 컨테이너가 시작되면 다음 작업이 수행됩니다:

1. 지정된 데이터베이스 생성
2. 지정된 사용자 생성 및 권한 부여
3. 기존 데이터 복원 (볼륨에 데이터가 있는 경우)

5.2 백엔드 서비스

빌드된 Spring Boot 애플리케이션이 시작됩니다:

- 환경 변수: `.env` 파일에서 가져온 설정 및 `SPRING_PROFILES_ACTIVE=prod`
- 포트: 8080 포트 노출
- 의존성: MySQL 서비스에 의존

백엔드 컨테이너가 시작되면 다음 작업이 수행됩니다:

1. Spring Boot 애플리케이션 시작
2. 환경 변수를 통한 MySQL 데이터베이스 연결 설정
3. JPA/Hibernate를 통한 데이터베이스 스키마 초기화/업데이트
4. REST API 엔드포인트 노출

5.3 Nginx 서비스

빌드된 Nginx 웹 서버가 시작됩니다:

- 볼륨: 설정 파일 및 SSL 인증서 마운트
- 포트: 80(HTTP), 443(HTTPS) 포트 노출
- 의존성: 백엔드 서비스에 의존

Nginx 설정(`nginx/default.conf`)은 다음과 같이 구성됩니다:

```
server{
    listen 80;
    server_name j12a304.p.ssafy.io;

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

server {
    listen 443 ssl;
    server_name j12a304.p.ssafy.io;

    # SSL 인증서 설정
    ssl_certificate /certs/fullchain.pem;
    ssl_certificate_key /certs/privkey.pem;

    # SSL 설정 최적화
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 10m;

    # API 요청을 백엔드로 프록시
    location /api {
        proxy_pass <http://backend: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;

        # CORS 설정
        add_header 'Access-Control-Allow-Origin' '*';
```

```
        add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS, P
UT, DELETE';
        add_header 'Access-Control-Allow-Headers' 'DNT,User-Agent,X-Requ
ested-With,If-Modified-Since,Cache-Control,Content-Type,Range,Authoriz
ation';

        # OPTIONS 요청 처리
        if ($request_method = 'OPTIONS') {
            add_header 'Access-Control-Allow-Origin' '*';
            add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTION
S, PUT, DELETE';
            add_header 'Access-Control-Allow-Headers' 'DNT,User-Agent,X-Re
quested-With,If-Modified-Since,Cache-Control,Content-Type,Range,Author
ization';
            add_header 'Access-Control-Max-Age' 1728000;
            add_header 'Content-Type' 'text/plain; charset=utf-8';
            add_header 'Content-Length' 0;
            return 204;
        }
    }

    # 기본 상태 체크용 엔드포인트
    location / {
        root /usr/share/nginx/html;
        index index.html;
    }
}
```

Nginx 컨테이너가 시작되면 다음 작업이 수행됩니다:

1. 설정 파일 로드
2. SSL 인증서 로드
3. HTTP 요청을 HTTPS로 리다이렉트
4. `/api` 경로의 요청을 백엔드 컨테이너로 프록시
5. 루트 경로(`/`)에서 HTML 상태 페이지 제공

6. 배포 과정 요약 및 특징

6.1 CI/CD 파이프라인 자동화

- GitLab 저장소에 코드 푸시하면 자동으로 파이프라인 시작
- SSH를 통한 원격 서버 접속 및 명령 실행
- 환경 변수 파일 자동 생성
- Docker 빌드 및 재시작 자동화

6.2 환경 변수 관리

- GitLab CI/CD 변수에 민감한 정보 저장
- 백엔드와 Docker Compose에 각각 필요한 환경 변수 별도 관리
- 환경 변수 파일을 통한 설정 분리

6.3 컨테이너 구성

- MySQL: 데이터베이스 서비스
- 백엔드: Spring Boot 애플리케이션
- Nginx: 웹 서버 및 프록시
- 각 서비스는 독립적인 컨테이너로 실행되며 필요한 의존성 설정

6.4 SSL 인증서 관리

- `/opt/certs` 디렉토리에 인증서 파일 저장
- Nginx 컨테이너에 인증서 디렉토리 마운트
- HTTPS 통신 제공 및 HTTP를 HTTPS로 리다이렉트

6.5 볼륨 관리

- MySQL 데이터를 위한 영구 볼륨 사용
- Nginx 설정 및 인증서 파일을 위한 볼륨 마운트

7. 서비스 관리 및 모니터링

7.1 컨테이너 상태 확인

```
docker ps
```

7.2 로그 확인

```
docker logs nginx
docker logs backend
docker logs mysql
```

7.3 서비스 재시작

```
docker-compose restart nginx
docker-compose restart backend
docker-compose restart mysql
```

7.4 전체 서비스 재배포

```
cd /home/ubuntu/S12P21A304
git pull
docker-compose build
docker-compose down
docker-compose up -d
```