E201_그로밋_자율프로젝트_포팅메뉴얼

목차

1. 사전 준비사항

백엔드 서비스를 위한 준비

- Docker 및 Docker Compose 설치 (버전 20.10.x 이상)
- 배포할 서버(리눅스 권장): 최소 4GB RAM, 2vCPU
- PostgreSQL 데이터베이스 (또는 Docker Compose로 자동 구성)
- 도메인 및 SSL 인증서 (HTTPS 지원을 위한 Nginx 구성)

API 키 및 서비스 계정

- Kakao 개발자 계정 및 앱 등록 (OAuth 인증용)
- Firebase 프로젝트 설정 및 서비스 계정
- 필요한 경우 OpenAI API 키, Claude API 키, Pinecone API 키
- Brave Search API 키 (검색 기능용)
- Google Search API 키 및 검색 엔진 ID (검색 기능용)

안드로이드 앱 빌드를 위한 준비

- JDK 17
- Android SDK
- Ruby 및 Bundler (Fastlane용)
- Firebase App Distribution 계정
- 앱 서명용 keystore 파일

2. 프로젝트 구조

프로젝트는 다음과 같은 주요 구성요소로 이루어져 있습니다:

- **front**: 프론트엔드 애플리케이션
  - **frontend**: 안드로이드 앱 (React Native)
  - **apk-fe**: 웹 프론트엔드 (Next.js)
- **back**: 백엔드 서비스
  - **gateway**: API 게이트웨이 (Spring Cloud Gateway)
  - **auth**: 인증 서비스 (Spring Boot)
  - **backend**: 메인 비즈니스 로직 서비스 (Spring Boot)
  - **rag**: Retrieval Augmented Generation 서비스 (FastAPI)
  - **search**: 검색 서비스 (FastAPI)
  - **mcp**: MCP API 서비스 (FastAPI)
  - **brave-search**: Brave 검색 프록시 서비스 (Node.js)
  - **google-web-search**: Google 검색 프록시 서비스 (Node.js)

3. 환경 변수 설정

1. 프로젝트 루트 디렉토리에 있는 `env.template` 파일을 복사하여 `.env.prod` 파일을 생성합니다.

```
cp env.template .env.prod
```

1. `.env.prod` 파일을 편집하여 필요한 환경 변수를 설정합니다.

```
# database (auth)
POSTGRES_AUTH_USER=your_auth_db_user
```

```
POSTGRES_AUTH_PASSWORD=your_auth_db_password
POSTGRES_AUTH_DB_NAME=auth_db

# database (scheduler)
POSTGRES_SCHED_USER=your_sched_db_user
POSTGRES_SCHED_PASSWORD=your_sched_db_password
POSTGRES_SCHED_DB_NAME=sched_db

# database (rag)
POSTGRES_RAG_USER=your_rag_db_user
POSTGRES_RAG_PASSWORD=your_rag_db_password
POSTGRES_RAG_DB_NAME=rag_db

# AI keys
OPENAI_API_KEY=your_openai_api_key
PINECONE_API_KEY=your_pinecone_api_key
PINECONE_INDEX_NAME=your_pinecone_index_name
CLAUDE_API_KEY=your_claude_api_key

# Firebase
FIREBASE_CREDENTIALS_JSON_BASE64=your_base64_encoded_firebase_
credentials
FIREBASE_APP_ID=your_firebase_app_id

# Kakao OAuth
KAKAO_CLIENT_ID=your_kakao_client_id
KAKAO_NATIVE_APP_KEY=your_kakao_native_app_key

# JWT
JWT_SECRET_KEY=your_jwt_secret_key
JWT_AT_VALIDITY=900000
JWT_RT_VALIDITY=604800000

# Search API keys
BRAVE_API_KEY=your_brave_api_key
GOOGLE_SEARCH_API_KEY=your_google_search_api_key
GOOGLE_SEARCH_ENGINE_ID=your_google_search_engine_id
```

```
# Web Frontend
FRONTEND_API_URL=https://your-api-domain.com
SERVER_BASE_URL=https://your-api-domain.com

# 앱 환경 구분
ENV=prod
SPRING_PROFILES_ACTIVE=prod
```

1. Firebase 서비스 계정 JSON 파일을 Base64로 인코딩하여 `FIREBASE_CREDENTIALS_JSON_BASE64` 환경변수에 설정합니다.

```
cat path/to/firebase-service-account.json | base64 -w 0
```

1. GCP 서비스 계정 키 파일을 대상 서버의 안전한 위치에 복사합니다.

4. 백엔드 서비스 배포

1. 서버에 프로젝트를 클론합니다.

```
git clone https://your-repository-url.git
cd project-directory
```

1. Nginx 설정 파일을 준비합니다. `/home/ubuntu/nginx/nginx.conf` 파일을 생성하고 SSL 인증서를 설정합니다.

```
# /home/ubuntu/nginx/nginx.conf 예시
user  nginx;
worker_processes  auto;

error_log  /var/log/nginx/error.log notice;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}

http {
    include       /etc/nginx/mime.types;
```

```
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$reques
t" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    keepalive_timeout  65;

    # 파일 업로드 크기 제한 설정
    client_max_body_size 20M;

    server {
        listen 80;
        server_name your-domain.com;

        # HTTP to HTTPS 리다이렉트
        location / {
            return 301 https://$host$request_uri;
        }
    }

    server {
        listen 443 ssl;
        server_name your-domain.com;

        ssl_certificate /etc/letsencrypt/live/your-domain.com/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/your-domain.com/privkey.pe
m;

        # SSL 설정
        ssl_protocols TLSv1.2 TLSv1.3;
        ssl_prefer_server_ciphers on;
        ssl_ciphers "EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES25
6+EDH";
```

```
        ssl_session_cache shared:SSL:10m;

        # 프론트엔드 서비스
        location / {
            proxy_pass http://frontend: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;
        }

        # API 게이트웨이
        location /api/ {
            proxy_pass http://gateway:8000;
            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;
        }

        # MCP API 엔드포인트
        location /mcp/ {
            proxy_pass http://mcp:8050;
            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;
        }
    }
}
```

1. Docker Compose를 사용하여 백엔드 서비스를 실행합니다.

```
docker-compose -f docker-compose-prod.yml up -d
```

1. 서비스 상태를 확인합니다.

```
docker-compose -f docker-compose-prod.yml ps
```

5. 프론트엔드(웹) 배포

웹 프론트엔드는 Docker Compose로 자동 배포됩니다. 백엔드 서비스 배포 과정에서 함께 배포됩니다.

6. 안드로이드 앱 빌드 및 배포

안드로이드 앱을 빌드하기 위해 제공된 `copy_and_deploy.sh` 스크립트를 사용합니다. 이 과정은 로컬 개발 환경에서 수행하는 것이 좋습니다.

1. 필요한 환경 변수를 설정합니다.

```
export ANDROID_SDK_ROOT=/path/to/android/sdk
export KAKAO_NATIVE_APP_KEY=your_kakao_native_app_key
export SERVER_BASE_URL=https://your-api-domain.com
export FIREBASE_APP_ID=your_firebase_app_id
```

1. 앱 배포에 필요한 파일들을 준비합니다. 이 파일들은 `$HOME/frontend/` 디렉토리에 위치해야 합니다.

```
mkdir -p $HOME/frontend
cp /path/to/firebase_service_account.json $HOME/frontend/
cp /path/to/google-services.json $HOME/frontend/
cp /path/to/release.keystore $HOME/frontend/
```

1. 배포 스크립트를 실행합니다. 여기서 `1.0.0`은 배포 버전입니다.

```
./copy_and_deploy.sh 1.0.0
```

1. 스크립트는 다음 작업을 수행합니다:
    - 필요한 파일을 올바른 위치에 복사
    - `local.properties` 파일 생성 및 환경 변수 설정
    - Bundler 및 Fastlane 설정
    - Firebase App Distribution을 통한 앱 배포

7. 문제 해결

데이터베이스 연결 문제

- 방화벽 설정 확인
- 데이터베이스 자격 증명 확인
- Docker 네트워크 설정 확인

API 키 관련 문제

- 환경 변수가 올바르게 설정되었는지 확인
- API 서비스 상태 및 할당량 확인

빌드 오류

- JDK 및 Android SDK 버전 확인
- 필요한 모든 파일이 올바른 위치에 있는지 확인
- 빌드 로그 확인

```
# 빌드 로그 확인 방법
cd front/frontend/android
./gradlew assembleRelease --info
```

인증 문제

- JWT 비밀키 설정 확인
- Kakao OAuth 설정 확인
- Firebase 구성 확인

서비스 로그 확인

서비스 로그를 확인하여 문제를 진단할 수 있습니다.

```
# 전체 서비스 로그 확인
docker-compose -f docker-compose-prod.yml logs -f

# 특정 서비스의 로그만 확인
docker-compose -f docker-compose-prod.yml logs -f [service-name]
```

여기서 `[service-name]`은 확인하려는 서비스 이름입니다. 예: `gateway`, `auth`, `backend` 등.

Docker 컨테이너 상태 확인

```
# 실행 중인 모든 컨테이너 확인
docker ps

# 특정 컨테이너 상세 정보 확인
docker inspect [container-id]

# 컨테이너 재시작
docker-compose -f docker-compose-prod.yml restart [service-name]
```

8. 부록

부록 A: 환경별 설정

개발 환경 설정

개발 환경에서는 `docker-compose-local.yml` 파일을 사용하여 서비스를 실행합니다.

```
# 개발 환경을 위한 환경 변수 파일 생성
cp env.template .env.dev

# 개발 환경 실행
docker-compose -f docker-compose-local.yml up -d
```

테스트 환경 설정

테스트 환경에서는 모든 서비스가 실행되지만, 실제 데이터는 사용하지 않습니다.

```
# 테스트 환경을 위한 환경 변수 파일 생성
cp env.template .env.test

# 환경 변수 설정
echo "ENV=test" >> .env.test
echo "SPRING_PROFILES_ACTIVE=test" >> .env.test
```

```
# 테스트 환경 실행
docker-compose -f docker-compose-prod.yml --env-file .env.test up -d
```

부록 B: 배포 자동화

배포 자동화를 위해 CI/CD 파이프라인을 구축할 수 있습니다. 다음은 Jenkins를 사용한 예시입니다.

Jenkinsfile 예시

```
pipeline {
    agent any

    environment {
        DOCKER_COMPOSE_FILE = 'docker-compose-prod.yml'
        ENV_FILE = '.env.prod'
    }

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

        stage('Build and Deploy Backend') {
            steps {
                sh 'docker-compose -f ${DOCKER_COMPOSE_FILE} --env-file
${ENV_FILE} build'
                sh 'docker-compose -f ${DOCKER_COMPOSE_FILE} --env-file
${ENV_FILE} up -d'
            }
        }

        stage('Build and Deploy Android App') {
            steps {
                sh './copy_and_deploy.sh ${APP_VERSION}'
            }
        }
```

```
    }

    post {
        always {
            sh 'docker-compose -f ${DOCKER_COMPOSE_FILE} logs > deploym
ent_logs.txt'
            archiveArtifacts artifacts: 'deployment_logs.txt', fingerprint: true
        }
    }
}
```

부록 C: 필요한 API 키 획득 방법

Kakao API 키 획득

1. Kakao Developers 사이트에 가입 및 로그인합니다.
2. 애플리케이션 추가를 선택하고 앱 이름과 회사명을 입력합니다.
3. 생성된 애플리케이션에서 "플랫폼" 메뉴로 이동하여 안드로이드 플랫폼을 추가합니다.
4. "앱 키" 메뉴에서 네이티브 앱 키를 확인할 수 있습니다.

Firebase 설정

1. Firebase Console에 로그인합니다.
2. "프로젝트 추가"를 클릭하고 프로젝트 이름을 입력합니다.
3. 안드로이드 앱을 추가하고 패키지 이름을 입력합니다.
4. "google-services.json" 파일을 다운로드하여 저장합니다.
5. "프로젝트 설정" > "서비스 계정" 탭에서 새 비공개 키 생성을 클릭하여 서비스 계정 키를 다운로드합니다.

OpenAI API 키 획득

1. OpenAI 사이트에 가입 및 로그인합니다.
2. API 키 섹션에서 새 API 키를 생성합니다.

Claude API 키 획득

1. Anthropic 사이트에 가입 및 로그인합니다.

2. API 키 섹션에서 새 API 키를 생성합니다.

Pinecone API 키 획득

1. Pinecone 사이트에 가입 및 로그인합니다.
2. 새 인덱스를 생성하고 API 키를 확인합니다.

부록 D: 안드로이드 앱 서명 설정

키스토어 생성

안드로이드 앱 배포를 위해서는 서명 키스토어가 필요합니다. 다음 명령어로 키스토어를 생성할 수 있습니다:

```
keytool -genkey -v -keystore release.keystore -alias app_alias -keyalg RSA -keysize 2048 -validity 10000
```

키스토어 정보 설정

생성한 키스토어 정보를 안드로이드 프로젝트에 설정합니다:

1. 키스토어 파일을 `$HOME/frontend/` 디렉토리에 복사합니다.
2. 다음 환경 변수를 설정하여 키스토어 정보를 제공합니다 (선택적):

```
export KEYSTORE_PASSWORD=your_keystore_password
export KEYSTORE_ALIAS=app_alias
```

1. `copy_and_deploy.sh` 스크립트에서 키스토어 설정 부분의 주석을 해제하여 사용합니다.

참고: 이 포팅 메뉴얼은 OpenBoard Korean 애플리케이션의 설치 및 배포를 위한 문서입니다.
문의사항이 있으면 프로젝트 관리자에게 연락하세요.

문서 버전: 1.0
최종 업데이트: 2025년 5월 21일