외부 서비스 정보

외부 서비스 가입 및 활용 정보 (카카오 Oauth, 카카오 맵)

https://developers.kakao.com/ 접속 후 로그인

내 애플리케이션 → 애플리케이션 추가하기 → 추가한 애플리케이션 클릭

카카오 앱 키

‘앱 키’ 탭 클릭

- REST API 키를 Backend application.yml 파일에 저장 (로그인을 위한 키)
- Javascript 키를 Frontend .env 파일에 저장 (지도 사용을 위한 키)

카카오 로그인 설정

카카오 로그인 탭 클릭 후 활성화 → Redirect URI에 로그인 후 redirect할 uri 추가 (ex. http://i12a506.p.ssafy.io/api/users/kauth) → 카카오 로그인 탭의 동의항목 클릭 → 닉네임, 카카오 계정 필수 동의 설정

‘플랫폼’ 탭 클릭 → Web 사이트 도메인 추가 (ex. http://i12a506.p.ssafy.io) (지도 및 oauth)

‘비즈니스’ 탭 클릭 → 비즈 앱 등록

Oauth 사용을 위한 정보 :
https://developers.kakao.com/docs/latest/ko/kakaologin/rest-api

지도 사용을 위한 정보 :

https://apis.map.kakao.com/web/guide/

Jenkins CI/CD

jenkins 콘솔 - 새로운 item - pipeline - Build when a change is pushed to Gitlab 선택 및 url 복사 - 고급 - secret token 생성 후 복사

gitlab 레포지토리 접속 - webhook 설정 - add new webhook - 복사한 url, secret token 붙여넣기 - push event - wildcard pattern 선택 후 clone 할 브랜치 입력 - add webhook

pipeline 작성

```
# Backend, Batch CI/CD

pipeline {
    agent any
    environment {
        DOCKER_IMAGE_BE = 'muinus-be'
        DOCKER_TAG = 'latest'
    }
    stages {
        stage('Clone') {
            steps {
                git branch: 'master', credentialsId: 'aseongjun99', url: 'https://lab.ssa
            }
        }
         stage('create yml') {
             steps {
                 withCredentials([
                    file(credentialsId: 'application', variable: 'application'),
                    file(credentialsId: 'batch', variable: 'batch')
                    ]) {
                   script {
                      dir('Backend/src/main/resources') {
                         sh 'rm application.yml || true'
                         sh 'cp $application application.yml'
                      }
                      dir('Batch/src/main') {
                         sh 'mkdir -p resources'
                         sh 'rm -f resources/application.yml || true'
                         sh 'cp $batch resources/application.yml'
```

```
                }
            }
        }
    }
}
stage('Build') {
    steps {
        script {
            dir('Backend') {
                // Gradle로 빌드
                sh 'chmod +x gradlew'
                sh './gradlew clean build -x test'
            }
            dir('Batch') {
                sh 'chmod +x gradlew'
                sh './gradlew clean build -x test'
            }
        }
    }
}
stage('Docker Build') {
    steps {
        script {
            dir('Backend') {
                // Docker 이미지 빌드
                sh "docker build -t ${DOCKER_IMAGE_BE}:${DOCKER_TAG} ."
            }
            dir('Batch') {
                sh "docker build -t ${DOCKER_IMAGE_BATCH}:${DOCKER_TAG
            }
        }
    }
}
stage('Docker Compose Up') {
    steps {
        script {
            dir('Backend') {
                sh 'docker-compose down'
```

```
                sh 'docker-compose up --build -d'
            }
            dir('Batch') {
                sh 'docker-compose down || true'
                sh 'docker-compose up --build -d'
            }
        }
    }
  }
  post {
    success {
      echo 'CI/CD 파이프라인이 성공적으로 완료되었습니다.'
    }
    failure {
      echo 'CI/CD 파이프라인에서 오류가 발생했습니다.'
    }
  }
}
```

```
# Frontend CI/CD

pipeline {
  agent any
  environment {
    DOCKER_IMAGE = 'muinus-fe'
    DOCKER_TAG = 'latest'
    CI = 'false'
  }
  stages {
    stage('Clone') {
      steps {
        git branch: 'Frontend', credentialsId: 'aseongjun99', url: 'https://lab.s
      }
    }
    stage('create env') {
      steps {
        withCredentials([file(credentialsId: 'env', variable: 'env')]) {
```

```
                script {
                    dir('Frontend') {
                        sh 'rm env || true'
                        sh 'chmod +w .env.production'
                        sh 'cp $env .env.production'
                        sh 'cat .env.production'
                    }
                }
            }
        }
    }
    stage('Docker Build') {
        steps {
            script {
                dir('Frontend') {
                    // Docker 이미지 빌드
                    sh "docker build -t ${DOCKER_IMAGE}:${DOCKER_TAG} ."
                }
            }
        }
    }
    stage('Docker Compose Up') {
        steps {
            script {
                dir('Frontend') {
                    sh 'docker-compose down'
                    sh 'docker-compose up --build -d'
                }
            }
        }
    }
}
post {
    success {
        echo 'CI/CD 파이프라인이 성공적으로 완료되었습니다.'
    }
    failure {
        echo 'CI/CD 파이프라인에서 오류가 발생했습니다.'
```

```
      }
    }
}
```

각각 master, Frontend 브랜치에 push/merge시 웹훅을 통해 Jenkins가 클론 후 빌드 및 배포