

OLIVE Action 도입기

OLIVE Platform에 GitHub Actions 기반 자동 의존성 분석 적용

김영환

KAKAO



CONTENTS

- 01 OLIVE Platform 소개
- 02 OLIVE Action 도입 배경
- 03 OLIVE Action 동작 구조
- 04 OLIVE Action 사용 방법
- 05 개발 과정 회고
- 06 앞으로의 계획





01

OLIVE Platform 소개



01 OLIVE Platform 소개

- 도입 배경

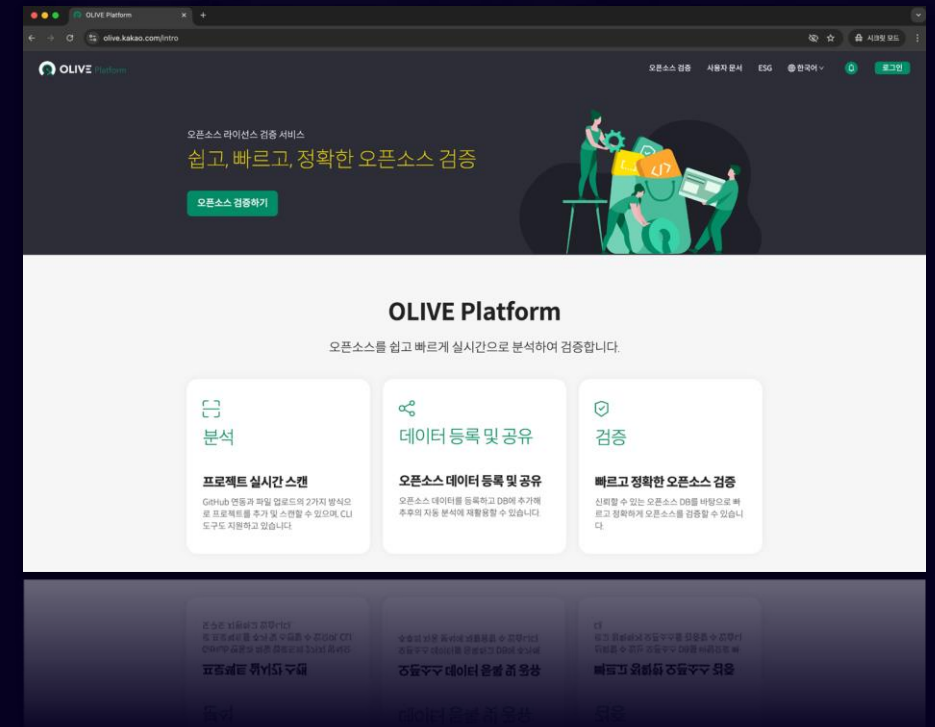
다양한 개발 환경과 플랫폼을 쉽고 빠르게 분석할 수 있는
개발자 주도형 검증 시스템의 필요

- 카카오에서 개발하고 공개한 오픈소스 라이선스 및 컴플라이언스 관리 자동화 플랫폼
"쉽고, 빠르고, 정확한 오픈소스 관리"를 목표로 함

- 기대 효과

개발자의 오픈소스 라이선스 의무사항 준수
소프트웨어 보안 및 라이선스 리스크의 사전 대응
수동검증 시간 대폭 단축: 10일 이상 → Just One Day

<https://olive.kakao.com/>



02

OLIVE Platform 주요 기능

- **프로젝트 분석:**
GitHub 연동, 파일 업로드, CLI를 통해 의존성 및 코드 스니펫 분석
- **매핑 & 사용자 정의:**
자동 매핑 및 직접 매핑, 매핑 데이터를 DB에 저장하여 자동 매핑
- **라이선스 검증 및 의무사항 안내:**
라이선스 자동 확인, 고지/배포 조건 등 의무사항 및 해결 가이드 제공
- **리포트 생성 및 고지문 제공:**
스캔 후 체크리스트, 리포트, 고지문 자동 생성 및 다운로드
- **CLI 연동:**
로컬 환경에서 코드 전송 없이 안전하게 사용 가능





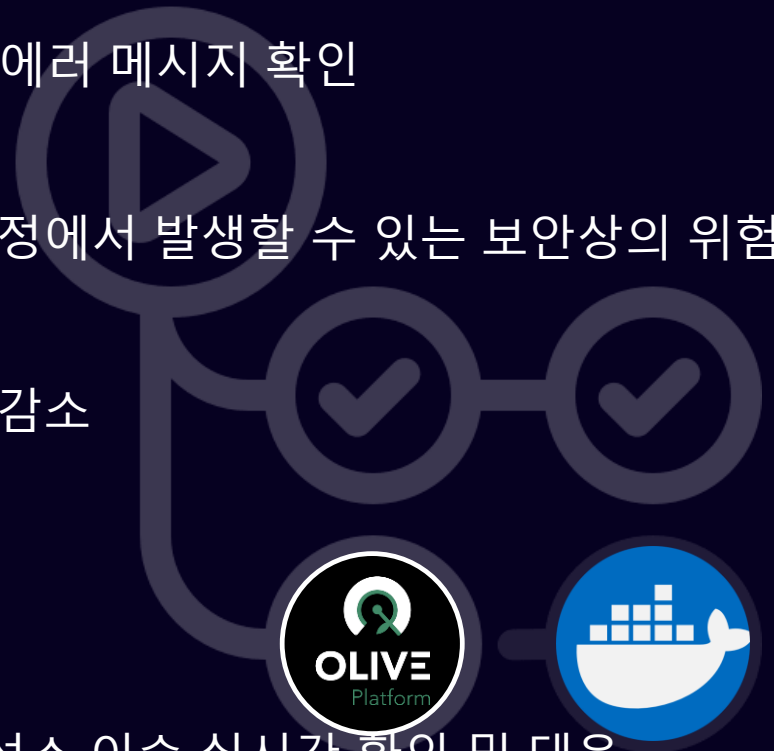
02

OLIVE Action 도입 배경

03

OLIVE Action 목표

- **분석 로그 확인:**
OLIVE CLI 스캔이 실패한 경우, OLIVE Action의 로그를 통해 에러 메시지 확인
- **잠재적 보안 위험:**
스캐너가 외부 코드를 내부 서버에 다운로드하여 실행하는 과정에서 발생할 수 있는 보안상의 위험 방지
- **개발 프로세스 자동화:**
CI/CD에 통합하여 수동 분석의 번거로움을 없애고 휴먼 에러 감소
- **라이선스 이슈 대응:**
의존성 분석을 통한 라이선스 이슈 사전 대응
- **투명한 코드 리뷰 문화 조성:**
Pull Request에 의존성 분석 결과를 자동으로 제공하여, 라이선스 이슈 실시간 확인 및 대응
- **개발자 경험(DX) 향상:**
복잡한 설정 없이 Workflow에 몇 줄만 추가하면 바로 사용 가능





03

OLIVE Action 동작 구조



04 OLIVE Action 동작 흐름

PR 생성/업데이트

GitHub Actions
실행

Docker 컨테이너
구동 및 OLIVE CLI
실행

결과 처리 및
리포팅

개발자가 코드를
Push
Trigger 조건 실행

Workflow에 정의된
OLIVE Action 자동
실행

격리된 분석 환경에서
소스 코드 분석

분석 결과를 가공하여
아티팩트 생성 및
PR 코멘트 작성

05

OLIVE Action 주요 기능

- **의존성/라이선스 분석**
프로젝트의 의존성 및 라이선스 정보 분석
- **PR 자동 코멘트**
분석 결과의 핵심 요약을 PR에 자동으로 게시
- **분석 아티팩트 생성**
상세 분석 결과(CSV, JSON)를 아티팩트로 저장하여 다운로드 가능
- **유연한 사용자 정의 설정**
분석 환경, 보관 기간, 분석 모드 등 다양한 옵션 제공





04

OLIVE Action 사용 방법



06 사전 준비

사전 준비 (Prerequisites)

- OLIVE Platform에서 API 토큰 발급
- 발급받은 토큰을 GitHub 저장소 Secrets로 추가
Settings > Secrets and variables > Actions
예시 이름: OLIVE_TOKEN

기본 사용법 (Quick Start)

- .github/workflows/ 경로에
예시 내용으로 yml 파일 생성

<https://github.com/kakao/olive-action>

```
name: OLIVE Action

on:
  pull_request:
    types: [opened, synchronize, reopened]
    branches:
      - develop
      - main

permissions:
  contents: read
  issues: write
  pull-requests: write

jobs:
  olive-scan:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Run OLIVE Action
        uses: kakao/olive-action@v1
        with:
          olive-token: ${ secrets.OLIVE_TOKEN }
          github-token: ${ secrets.GITHUB_TOKEN }
```



다양한 설정 값

입력값(Inputs) 으로 상세 설정하기

- “with” 키워드를 사용하여 다양한 동작 제어 가능

이름	설명	필수	기본값
olive-token	OLIVE Platform API 토큰	Y	
github-token	PR 코멘트 작성을 위한 GitHub 토큰	Y	
olive-project-name	OLIVE Platform에 등록될 프로젝트 이름	N	저장소 이름
source-path	분석할 소스코드의 루트 경로	N	.
user-config-path	사용자 정의 config 파일 경로	N	
analyze-only	분석만 수행 (OLIVE Platform 전송 X)	N	false



08 설정 값 활용 -1

사용 예시 (Advanced Usage)

- 프로젝트 이름, 분석할 소스 경로, 아티팩트 보관 기간, PR에 코멘트 작성 여부 등

```
- name: Run OLIVE Action with custom settings
  uses: kakao/olive-action@v1
  with:
    olive-token: ${ secrets.OLIVE_TOKEN }
    github-token: ${ secrets.GITHUB_TOKEN }
    olive-project-name: "my-custom-project"
    source-path: "./src" # 분석할 소스코드 경로
    artifact-retention-days: "7" #아티팩트 보관
    기간 (일)
    comment-on-pr: "true" #PR에 코멘트 작성 여부
```

- 분석만 수행 (OLIVE Platform 에 결과 미전송)
- analyze-only를 "true"로 설정

```
- name: Run OLIVE Action (analysis only)
  uses: kakao/olive-action@v1
  with:
    olive-token: ${ secrets.OLIVE_TOKEN }
    github-token: ${ secrets.GITHUB_TOKEN }
    analyze-only: "true"
```



09 설정 값 활용 -2

사용자 정의 config 파일 사용하기

- user-config-path에 프로젝트에 사용자 정의 파일(user-config.yml)을 생성하여 OLIVE CLI의 세부 동작을 제어

```
.github/workflows/olive.yml
- name: Run OLIVE Action with custom config
  uses: kakao/olive-action@v1
  with:
    olive-token: ${ secrets.OLIVE_TOKEN }
    github-token: ${ secrets.GITHUB_TOKEN }
    user-config-path: "./user-config.yml"
```



user-config.yml 파일 예시

```
isOpenSource: false # 소스 코드 공개 여부 (기본값: false)
excludePaths: # 분석에서 제외할 경로 (기본값: 빈 목록)
- ".git"
- "build"
analysisType: "BUILDER" # Gradle 의존성 분석 방식
("PARSER" 또는 "BUILDER", 기본값: "PARSER")
onlyDirect: false # 직접 의존성만 분석할지 여부 (기본값:
false)
gradleBuildVariant: ""
# 분석할 Build Variant 지정 (예: "debug", "release",
"freeDebug" 등, 기본값: "")
# 값이 비어 있으면 모든 Build Variant에 대한 의존성이
포함됩니다.
excludeGradle: # Gradle 빌드 수행 시 제외할 모듈 이름 목록
(기본값: 빈 목록)
- ":test"
```

10

실행 결과 -1

● Pull Request 코멘트

comment-on-pr: true (기본값)일 경우 자동
생성/업데이트

PR Comment

● 표시 정보

- OLIVE CLI 버전
- 프로젝트 이름
- 실행 로그 링크
- 라이선스 및 컴포넌트 매핑 요약
- 매핑되지 않은 의존성 목록
- 아티팩트 목록

OLIVE Action PR 테스트 #1
Knext wants to merge 1 commit into `main` from `feature/olive-ac...`

github-actions bot commented 9 minutes ago · edited

OLIVE Action

- OLIVE CLI 버전: 2.9.0
- 프로젝트 이름: sample-app-for-android
- 상세 로그: [OLIVE Action 실행 결과](#)
- OLIVE Platform 분석결과: [OLIVE Platform scan 결과 자세히보기](#)

라이선스 정보

주의: 이슈가 있는 라이선스가 발견되었습니다. 의무사항 확인해서 준수해주세요.

Licenses: [2]

ID	Name	isIssued	Url
1	Apache-2.0	X	http://www.apache.org/licenses/LICENSE-2.0
2	MPL-2.0	O	http://www.mozilla.org/MPL/2.0/ Disclose source code (files or modul

컴포넌트 매핑 정보

Mapping Components: [5]

ID	Type	Name	Url
1	REVIEWED	Android - platform - frameworks - support	https://android.googlesource.com/platform/fr
2	REVIEWED	ConstraintLayout	https://github.com/androidx/constraintlayout
3	REVIEWED	Cozodb for Android	https://github.com/cozodb/cozodb-android
4	REVIEWED	Material Components for Android	https://github.com/material-components/material-components-android
5	REVIEWED	Square Picasso	https://github.com/square/picasso

확인 필요한 의존성 정보

Unmapping Dependencies: [3]

ID	Count	PackageManager	GroupID	ArtifactID	Version	Scope	Package Name
1	1	LIBRARY		kakao-olive-camera	1.3.5		...akao-olive-camera-1.3.5.jar
2	1	LIBRARY		kakao-olive-side-menu	1.22.2		...olive-side-menu-1.22.2.jar
3	1	LIBRARY		kakao-olive-super	1.5.6		...kakao-olive-super-1.5.6.jar



실행 결과 -2

- **GitHub 아티팩트 (Artifacts)**
Action 실행 결과 페이지에서 다운로드 가능, Artifacts
 - dependency.csv, dependency.json (의존성 분석 결과)
 - mapping.csv, unmapping.csv (의존성 매핑 결과)

← OLIVE Action

✓ OLIVE Action PR 테스트 #1

Summary

Jobs

olive-scan

Run details

Usage

Workflow file

Re-run triggered 4 minutes ago

Knex #1 feature/olive-action

Status

Success

Total duration

2m 22s

Artifacts

3

olive-action.yml

on: pull_request

olive-scan 2m 19s

Artifacts

Produced during runtime

Name	Size
apply-analysis	5.16 KB
dependency-analysis	1.83 KB
local-config.yaml	541 Bytes

12

실행 결과 -3

• OLIVE Platform 연동

OLIVE Platform

오른소스 검증 사용자 문서 ESG 한국어 감영환 (Sean)

+ 프로젝트 추가하기

내 프로젝트

컴포넌트

라이선스

오른소스 가이드

내 프로젝트 / sample-app-for-android

sample-app-for-android

OLIVE Action

2025.09.29 14:31:28 [RESOLVING](#)

요약

분석 상세

컴포넌트

라이선스

취약성

리포트

스캔 정보 +

이슈를 해결하고 검증을 완료하세요.

분석 상세에서 매핑되지 않은 의존성 3개를 컴포넌트에 매핑하세요.

라이선스에서 이슈 있는 라이선스 1개를 확인하세요.

의존성

확인이 필요한 의존성: 3 개

매핑된 의존성: 5 개

간접 의존성 분석 결과가 없습니다.

코드스니펫

라이브러리를 이용해 코드스니펫 분석을 진행할 수 있습니다.

OLIVE CLI로 생성한 프로젝트는 소스코드가 존재하지 않아 코드스니펫 분석이 불가능합니다.

라이선스

확인이 필요한 라이선스: 1개

컴포넌트 매핑이 완료되면 정확한 라이선스 계수를 확인할 수 있습니다.

취약성

발견된 의존성별 취약성 정보가 없습니다.

컴포넌트

5

확인된 컴포넌트: 5개

미확인 컴포넌트: 0개

라이선스

2

오류사항 이슈가 있는 라이선스: 1개

리포트

체크리스트 확인하기

고지문 미리보기

스캔 이력

2025.09.29 14:31:28

스캔 요청

감영환 (Sean)

2025.09.29 14:31:28

스캔 시작

감영환 (Sean)

해결 중

감영환 (Sean)

OLIVE Action
바로가기



05

개발 과정 회고

13

Lessons Learned -1

1: 내/외부 환경의 이중 관리

- **문제:** Action을 사내용과 외부 공개용(Marketplace)으로 동시에 관리해야 했음
 - 버전 호환성:** 외부 마켓플레이스는 최신 버전을 요구, 사내에서는 특정 버전이 필요
 - 코드 의존성:** 사내 서비스에 의존하는 코드를 외부 배포 시 모두 제거해야 하는 수고
 - 프록시 이슈:** 사내에서는 별도의 프록시 설정이 되어야 테스트 가능
- **해결:** 초기에는 수동으로 동기화, 현재는 워크플로우를 이용한 자동화 작업 중

2: 예상치 못한 과금 이슈

- **사건:** 개발 중 테스트 Action 실행이 갑자기 차단됨
- **원인:** 유료 러너(macOS Large) 사용으로 인한 과금 발생. 하지만 결제 수단이 등록되지 않아 GitHub에서 즉시 실행을 차단
- **교훈:** GitHub Actions의 과금 정책, 특히 러너 종류별 과금 여부를 명확히 이해해야 함.
(Public 저장소 + 표준 러너 = 무료)
- **참고:** GitHub Actions billing <https://docs.github.com/en/billing/concepts/product-billing/github-actions>



Lessons Learned -2

3: 액션 공개와 친절한 문서화의 교훈

- **과정:** GitHub Marketplace 등록 및 공개
- **교훈:** action.yml 메타데이터, 상세한 README의 중요성 체감
초기 사용자 가이드가 미흡하여 여러 차례 보완 작업을 진행
사용자 친화적인 문서화가 좋은 도구의 완성도를 결정

4: 바이브 코딩 적극 활용

- **과정:** AI를 이용한 OLIVE Action 스크립트 및 Workflow 작성
- **교훈:** Cursor, Claude 등 AI를 이용해서 개발 진행
대부분의 코드를 직접 작성하지 않고 바이브 코딩 형태로 진행
GitHub Action 문법을 잘 모르는 상황에서도 매우 빠르고 정확하게 개발 가능
바이브 코딩이 실제 개발 환경에서도 잘 적용되는 것을 확인

15

드디어! OLIVE Action 마켓 등록!

- **준비**

Public 저장소의 루트 경로에 action.yml과 상세한 README.md 작성
액션 메타데이터 작성 (<https://github.com/kakao/olive-action/blob/main/action.yml>)

- name, description, author **필수 입력**
- branding (아이콘, 색상)을 지정하여 액션을 돋보이게 할 수 있음

- **등록**

새 릴리스(Release) 생성 시 "Publish this Action to GitHub Marketplace" 옵션 체크
적절한 카테고리 선택 (**계정에 2단계 인증(2FA) 활성화 필수**)
<https://github.com/marketplace/actions/olive-action>

- **관리**

등록 완료 후 릴리스 페이지를 통해 수정 및 삭제 관리





06

앞으로의 계획



앞으로의 계획

- **Globalization:**
OLIVE Action 영문 문서화 및 키워드 최적화를 통해 글로벌 검색성과 접근성 개선
- **적극적인 홍보:**
OLIVE Action을 통해 OLIVE Platform에 더 쉽게 접근할 수 있는 부분 적극적 홍보
- **OLIVE Platform 고도화:**
OLIVE Platform에서 OLIVE Action 상태 관리 및 수행할 수 있도록 사용성 개선
- **도커 환경 최적화:**
패키지 매니저별 경량화된 도커 이미지를 제공하여 분석 속도와 효율성 증대

감사합니다.

