FOSSLight Scanner

: Free and Open Source Software Light

FINAL EVALUATION

과학기술정보통신부 nipa 정보통신산업진흥원 OPENUP IPA 한국IT비즈니스진흥협회

목차

01 프로젝트 소개

02 컨트리뷰션 진행 과정

03 컨트리뷰션 진행 성과

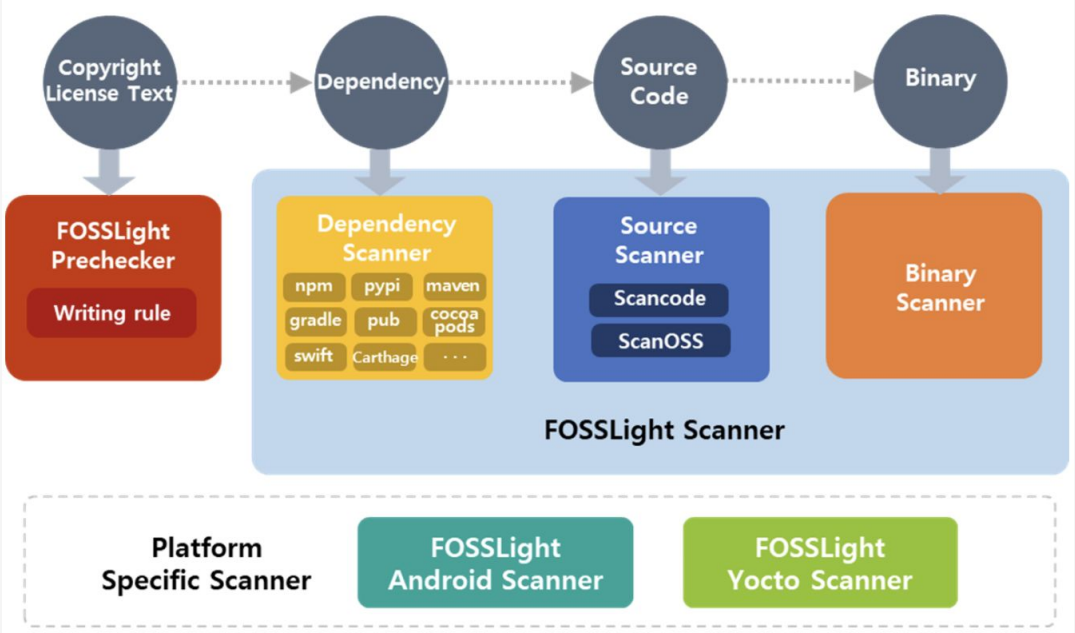
04 활동 후기

04 컨트리뷰션 Q & A

01

프로젝트 소개

1) FOSSLight Scanner 핵심 기능



프로젝트 종속성

+

소스 코드

+

바이너리 파일

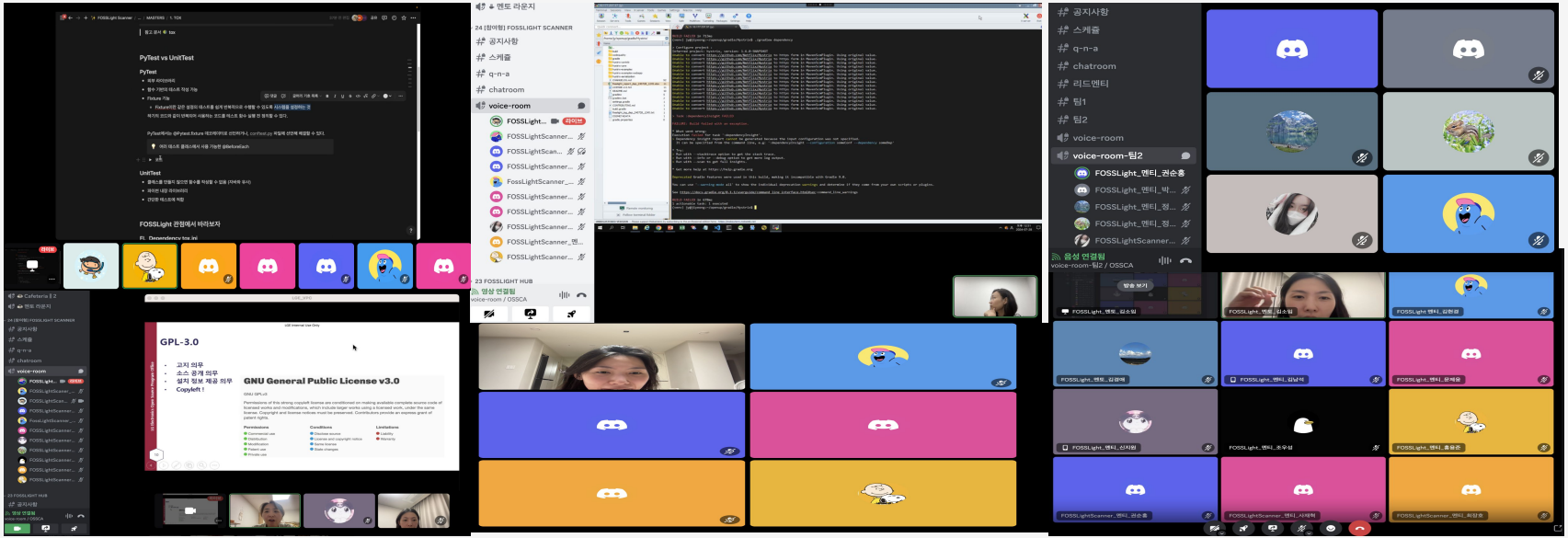
SBOM(소프트웨어 부품 명세서)

02

컨트리뷰션 진행 과정

2-1) 주간 온라인 미팅

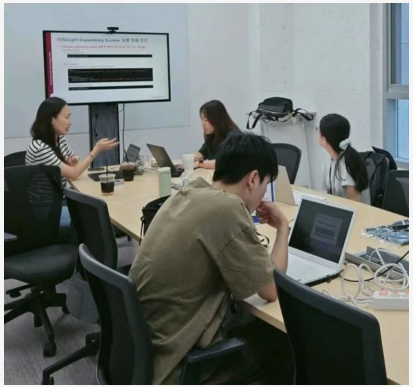
90% 넘는 인원이 적극적으로 매주 참여



2-2) 주간 오프라인 미팅

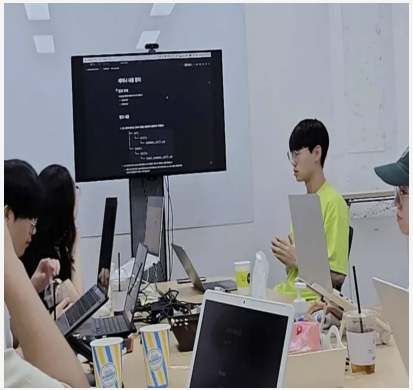
주간 오프라인 미팅 활성도 100%

2-3) 격주 세미나 진행



FOSSLight 실행 방법
및 dependency 가이드
24.07.28

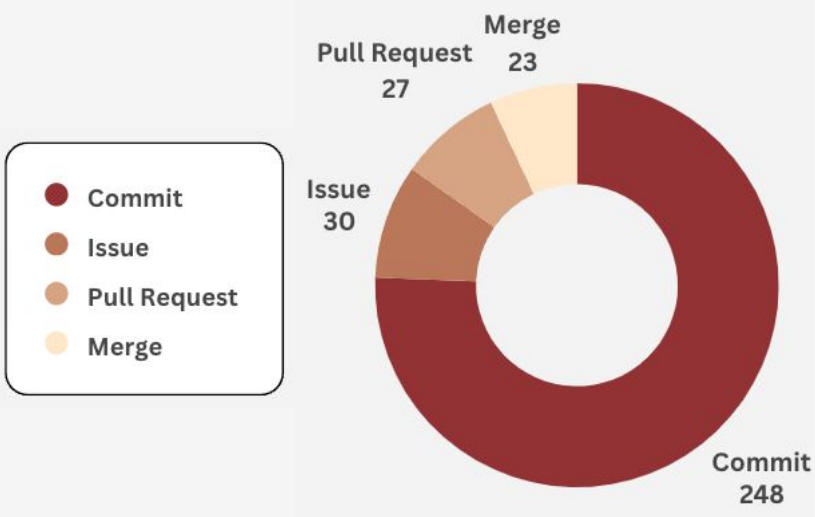
적극적으로 컨트리뷰션에
임할 수 있는 방법은
무엇인가
24.08.25

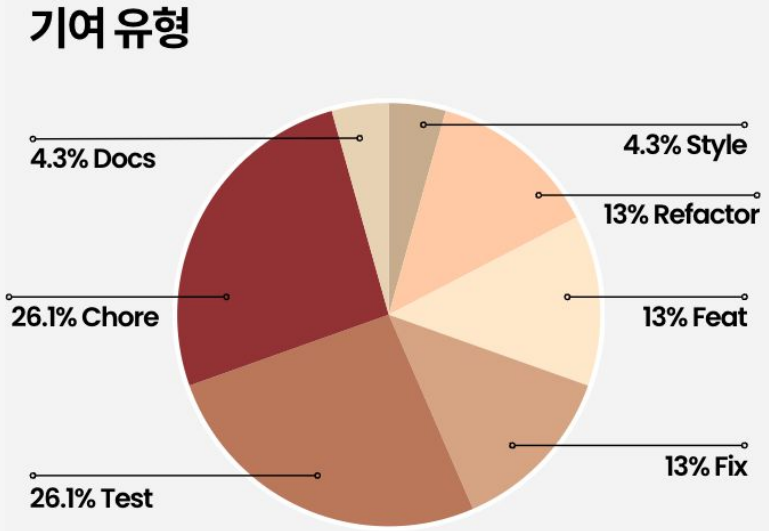


테스트코드를 바꾸게된
이유. 그에 대한 효과
24.09.08

오픈소스 보안 취약점을
관리하는데 있어
FOSSLight의 필요성
24.09.28

3) 다양한 방면의 기여 활동 - 정량적 기여 성과





달성률: 27(PR) / 30(ISSUE) = 90%

기여 유형 수: 7개

03

컨트리뷰션 진행 성과

1) 컨트리뷰션 모토

FOSSLight Scanner 컨트리뷰션 모토

누구나 쉽게 컨트리뷰션 할 수 있는 환경을 조성하자!

Test Code 팀

체계적인 테스트 코드 구축으로
다수의 코드 기여도 안정적으로 관리하고,
리뷰 시간을 단축

DevOps 팀

Package 배포 과정을 자동화하고,
Docker 개발 환경을 개선하여 처음 접근하는 기여자도
쉽게 개발 환경을 세팅할 수 있게 함

FL scanner

FL Prechecker

FL Dependency

FL source

FL Binary

FL Util

Test Code 팁

리뷰 프로세스를 더욱 효과적으로

체계적인 테스트 코드 구축으로
다수의 코드 기여도 안정적으로 관리하고,
리뷰 시간을 단축

2-1) 테스트 코드 강화

[리팩토링] 테스트 코드 중 Pytest 기능 추가

기존 테스트 코드 흐름도

Github Action → Tox → Run Test

기존 테스트 코드의 문제점

- 파일 내부 값 검증 부족
- 다양한 상황에 대한 테스트 X
- 테스트 관리의 어려움

Tox란?

Python 프로젝트의
테스트 자동화 도구

여러 Python 버전에서
테스트를 실행하게
해주는 가상환경 관리자

2-1) 테스트 코드 강화

[리팩토링] 테스트 코드 중 Pytest 기능 추가

Pytest란?

Python 테스트 프레임워크의 표준

간단한 유닛테스트부터 복잡한 기능 테스트까지 가능한 도구

변경된 테스트 코드 흐름도

Github Action → Tox → Pytest → Run Test

Pytest가 추가된 패키지 목록


FL Dependency


FL source


FL Util


FL Binary


FL Prechecker


FL scanner

2-1) 테스트 코드 강화

[리팩토링] 테스트 코드 중 Pytest 기능 추가

Pytest 기능을 추가하면서 얻게된 이점

- 상세 에러 메시지
- subprocess 활용
- 테스트 결과 검증 강화

상세 에러 메세지의 이점

- 어떤 커맨드로 인해 실패를 했는지 알 수 있음
- 테스트 결과값과 예상값을 비교 가능
- 결과 유무 확인 가능

정확한 문제 진단을 통해 신속한 해결책을 제시 ⇨ 사회적 비용 및 시간 절감, 서비스 안정성 향상

2-1) 테스트 코드 강화

[리팩토링] 테스트 코드 중 Pytest 기능 추가

Pytest 기능을 추가하면서 얻게된 이점

- 상세 에러 메시지
- subprocess 활용
- 테스트 결과 검증 강화

subprocess 활용의 이점

- 명령어 실행 제어 가능
- 쉘 주입 공격에 대한 취약점 감소
- OS에 관계없이 일관된 방식으로 실행가능

안전하고 통일된 시스템 운영 ⇨ 사이버 보안 강화, 디지털 격차 해소

2-1) 테스트 코드 강화

[리팩토링] 테스트 코드 중 Pytest 기능 추가

Pytest 기능을 추가하면서 얻게된 이점

- 상세 에러 메시지
- subprocess 활용
- 테스트 결과 검증 강화

테스트 결과 검증 강화의 이점

- 버그와 결함을 조기에 발견할 수 있음
- 테스트 프로세스와 데이터에 대한 품질검사를 통해 신뢰성이 향상

체계적인 테스트 코드 구축

- 다수의 코드 기여 안정적 관리
- 획기적인 리뷰 시간 단축

FOSSLight GUI

Docker

VS code extension

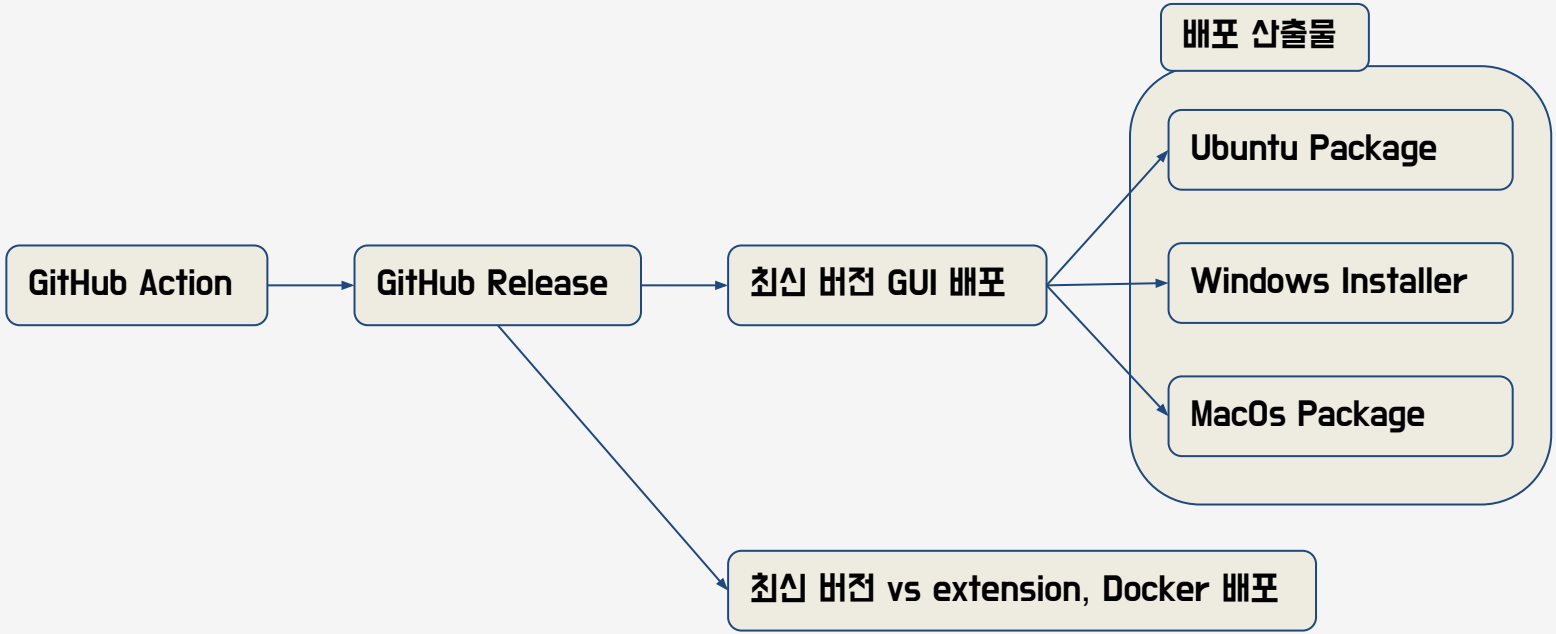
DevOps 팀

개발 진입 장벽을 낮춰라

Package 배포 과정을 자동화하고.
Docker 개발 환경을 개선하여
처음 접근하는 기여자도 쉽게 개발
환경을 세팅할 수 있게 함

3-1) 패키지 배포과정 자동화 (1)

[신규 기능] GitHub release시 FOSSLight 최신 버전 패키지 배포 자동화



3-1) 패키지 배포기능 자동화 (2)

[신규 기능] FOSSLight vs extension 실행 명령 간소화

기존 vs extension 실행 방법

ctrl + shift + p → analyze root directory / analyze currnet file 선택

변경된 vs extension 실행 방법

ctrl + shift + r → analyze root directory

ctrl + shift + c → analyze current file

3-2) 쉽고 간편해진 포스라이트 (1)

[신규 기능] One Click FOSSLight

외부 issue 발생

docker를 이용할 때 -p output 경로가 정상적으로 작동하지 않고.
이용하기가 너무 까다롭습니다 :(

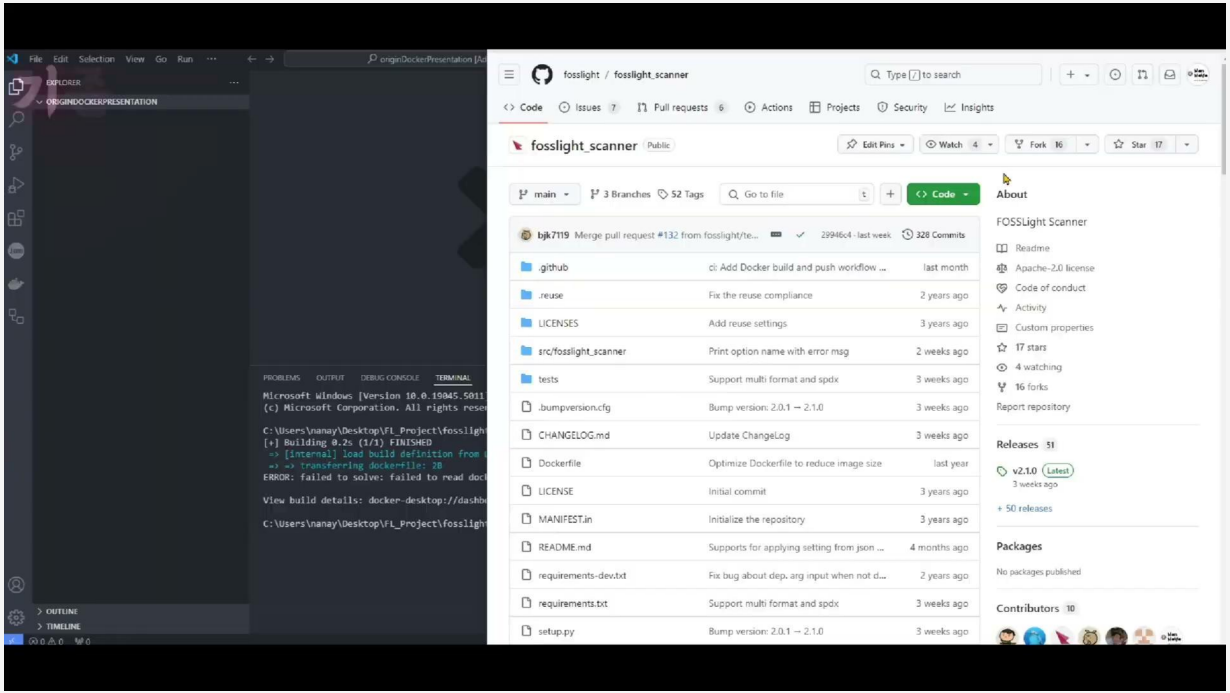
기존 docker 입력 방법

git clone FOSSLight scanner → docker build . → docker run -it -v /Users/fosslight_scanner/test_output:/app/output fosslight -p tests/test_files -o output ...

커맨드가 너무 길고. 한글자만 잘못 입력해도 실행이 안되네..
어떻게 하면 이 문제를 해결할 수 있을까?

3-2) 쉽고 간편해진 포스라이트 (1)

[신규 기능] One Click FOSSLight



3-2) 쉽고 간편해진 포스라이트 (2)

[신규 기능] FOSSLight 추가 옵션 제어를 쉽고 빠르게!

예시 요구사항

포스라이트 리포트 파일 포맷은 yaml로.
OSS 정보 수정은 안하고 싶어!

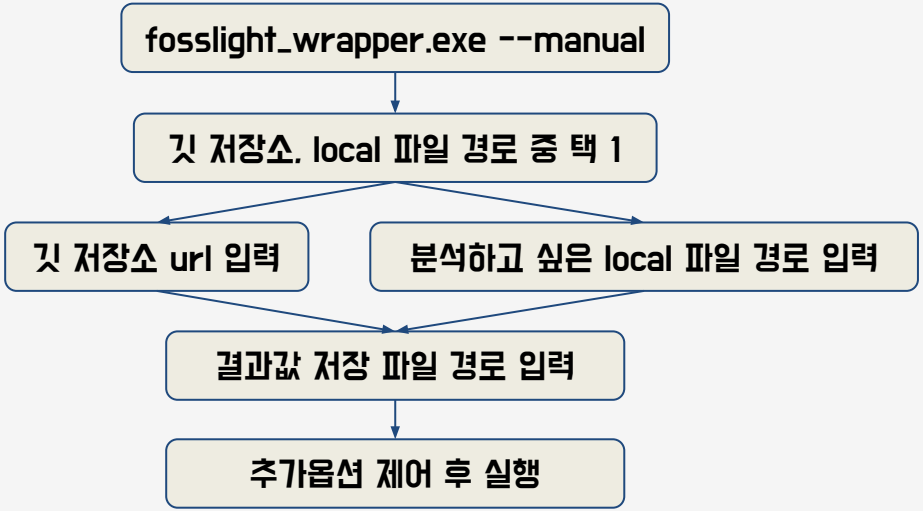
기존 docker 입력 방법

```
docker run -it -v
/Users/fosslight_scanner/test_output:/a
pp/output fosslight -p tests/test_files -o
output -f yaml –no_correction …
```

추가옵션을 입력하였더니 커맨드가 더 길어졌어..
어떻게 하면 이 문제를 해결할 수 있을까?

3-2) 쉽고 간편해진 포스라이트 (2)

[신규 기능] FOSSLight 추가 옵션 제어를 쉽고 빠르게!




```
Manage additional options:
1. Add new option
2. Remove option
3. View current options
4. Finish and proceed

Enter your choice (1-4): 1

Available additional options:
1. -f <format>: FOSSLight Report file format (excel, yaml)
2. -c <number>: Number of processes to analyze source
3. -r: Keep raw data
4. -t: Hide the progress bar
5. -s <path>: Path to apply setting from file
6. --no_correction: Don't correct OSS information
7. --correct_fpath <path>: Path to the sbom-info.yaml file
8. -u <db_url>: DB Connection (for 'all' or 'bin' mode)
9. -d <dependency_argument>: Additional arguments for dependency analysis

Enter the number of the option you want to add:
```

window. mac os. ubuntu 환경에서 **모두** 이용 가능

04

활동 후기 및 향후 계획

1) 활동 후기

오픈소스 컨트리뷰션 아카데미 활동을 마치며

멘티 문제웅 활동 후기

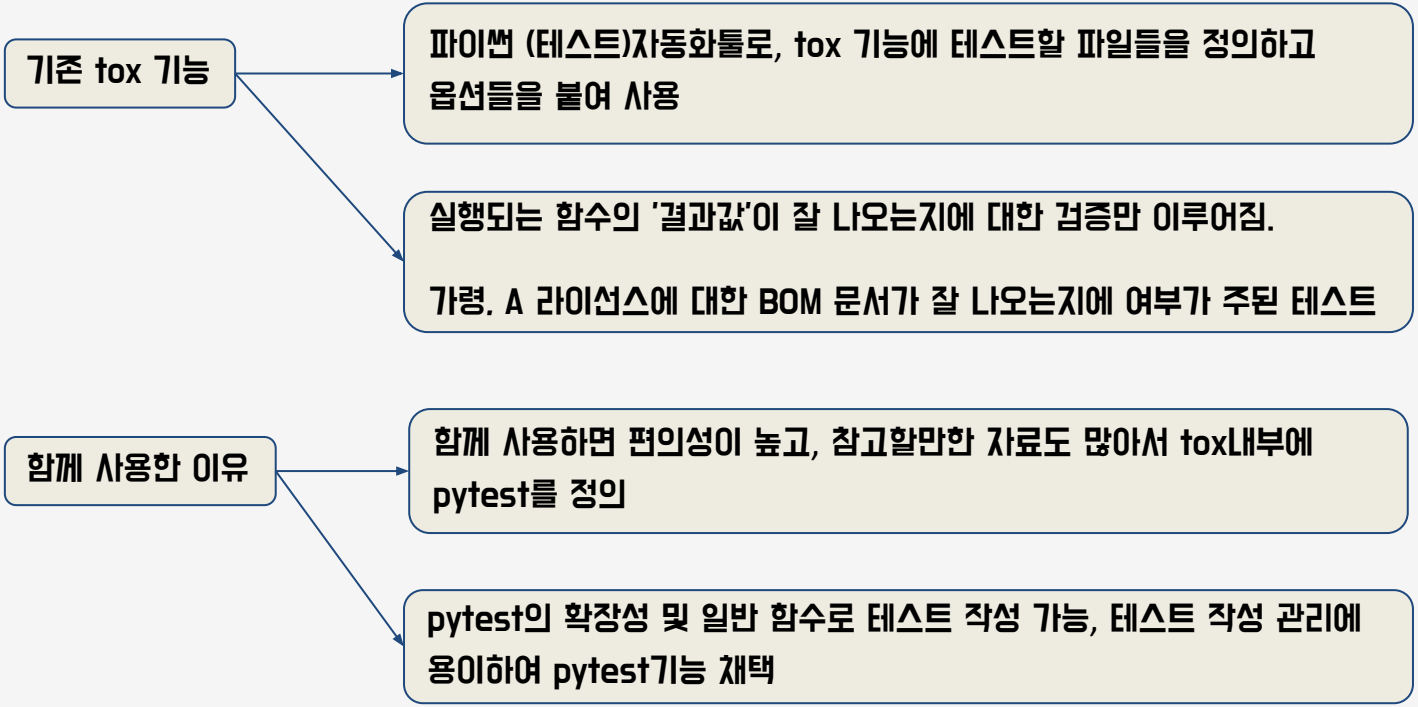
메인 기술 스택이 다르더라도, 테스트 코드 작성과 같은 공통적인 부분에 대한 지식으로도 충분히 오픈소스에 기여할 수 있다는 경험을 쌓을 수 있었습니다. 이를 통해 앞으로 오픈소스를 접할 때 내가 기여할 부분은 없는지 유심히 살펴보는, 개발자로서 한층 더 **성숙한 관점**을 가질 수 있게된 계기가 되었습니다.

멘티 양지원 활동 후기

오픈소스 컨트리뷰션 활동을 통해 다양한 기술을 새롭게 접하며 기술적 시야를 넓힐 수 있었습니다. 특히, 각기 다른 전문성을 가진 분들과의 코드 리뷰와 세션을 통해 깊이 있는 개발적 인사이트를 얻을 수 있었고, 함께 더 나은 로직을 모색하며 **상호 협력의 가치**를 느낄 수 있었습니다.

Thank you for listening :)

tox 와 pytest를 혼용한 이유



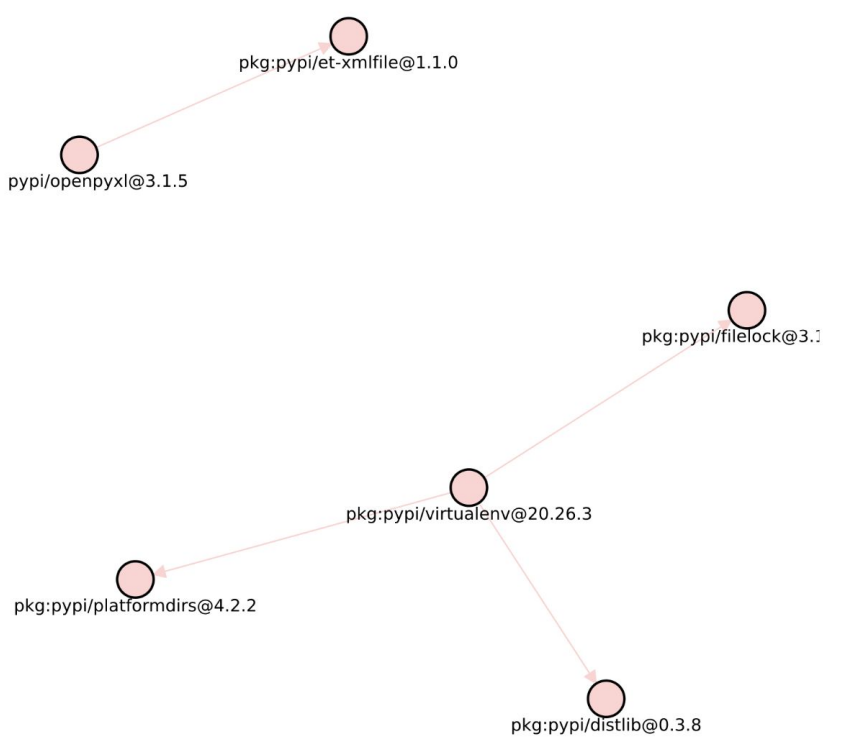
05

컨트리뷰션 Q&A

06

Appendix

Dependency Graph Network



각 패키지 별 의존 중인 패키지가 무엇인지 한눈에 직관적으로 알 수 있는 그래프

어떠한 패키지에서 Log4j 백도어 사태 같은 해킹 이슈가 있을 때, 해당 패키지를 쉽게 찾음으로써 빠르게 제거 가능