오픈소스 소프트웨어(OSS) 소개

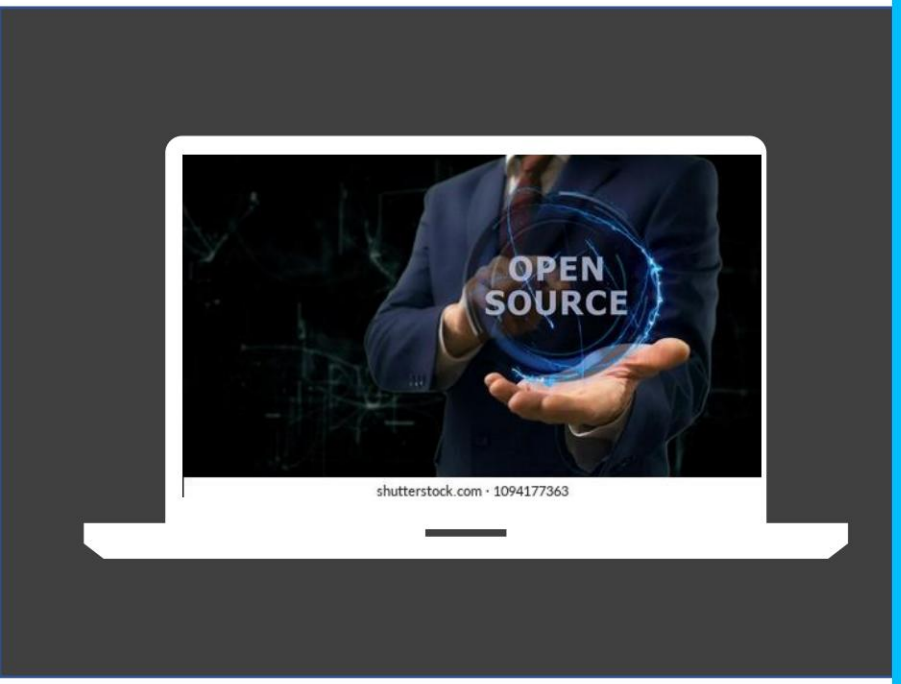
오픈소스 소프트웨어 개발과 관련된 개념, 전략 및 방법론

3주차 – 강의 5

자밀 후세인
jamil@세종.ac.kr
010-6252-8807

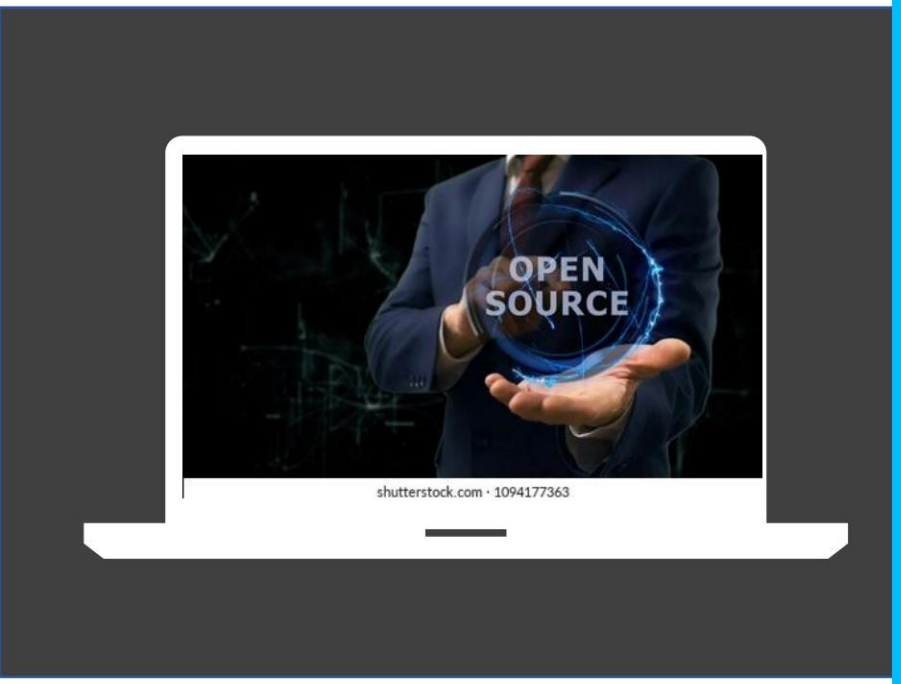
사무실: 421, 혁신 센터
세종대학교

리캡

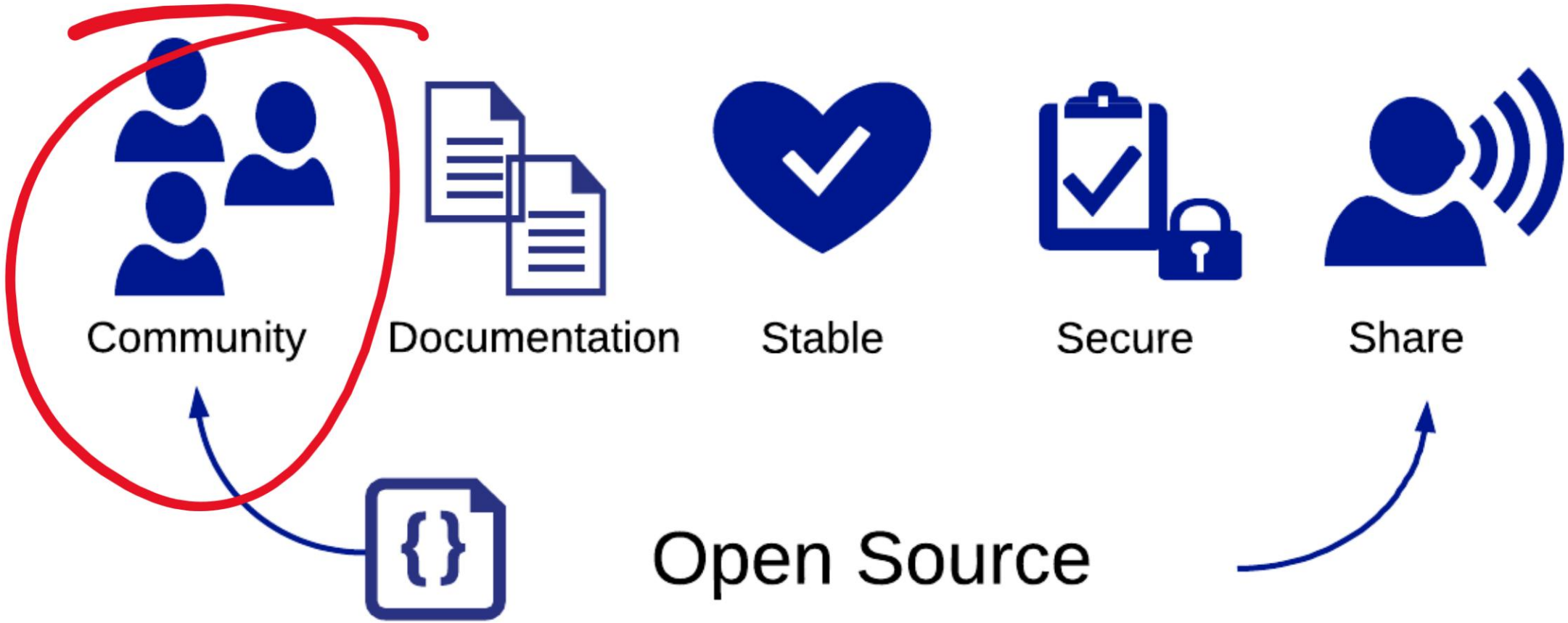


◯ 저작권 및 라이선스 개요 ◯ 오픈 소스의 법적 의미 이해 ◯ 라이선스의 요약 ◯ 소프트웨어 라이선스 범주 ◯ 라이선스의 측면 ◯ GPL 및 라이선스 호환성 ◯ 라이선스 선택

오늘, 의제



◯ 커뮤니티와 그 구조 ◯ 오픈 소스에 기여하는 이유는? ◯ 수업 활동



출처: https://hackernoon.com/what-i-learned-from-building-my-first-successful-open-source-project-72f7429145a0

오픈소스 프로젝트 커뮤니티

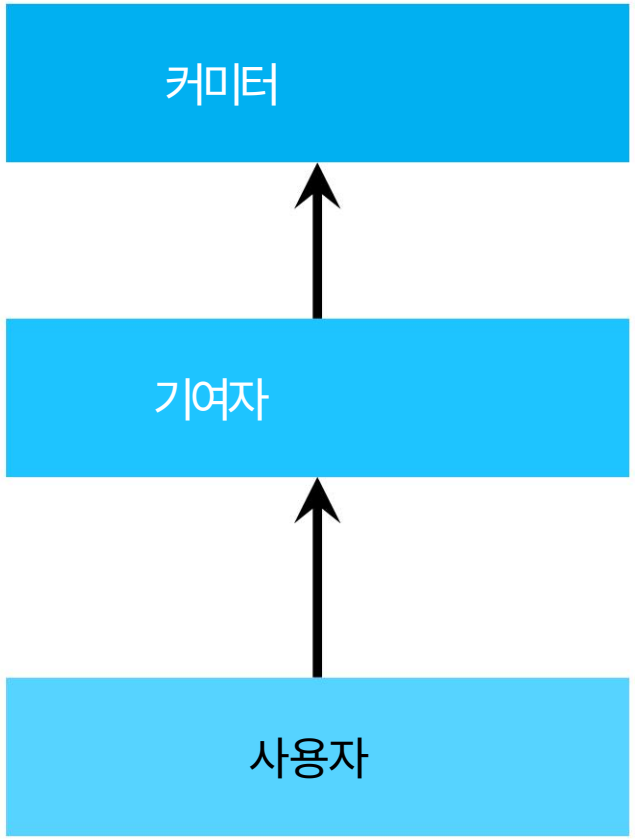
- 특정 애플리케이션 이나 도구 에 관심이 있는 기술자 커뮤니티가 공동으로 개발한 후, 이를 활용 할 수 있는 더 많은 개인 커뮤니티 에 무료로 배포하는 소프트웨어입니다 .

- 오픈 소스 프로젝트 커뮤니티는
  - 오픈소스 프로젝트에 참여하는 사람들과 회사 들의 그룹
- 개발자 커뮤니티 는
  - 소프트웨어를 개발 하는 프로젝트 커뮤니티의 하위 집합
- 사용자 커뮤니티는 소프트웨어 를 사용하는 프로젝트 커뮤니티의 하위 집합 입니다.

오픈소스 프로젝트 커뮤니티-FOSS의 Onion 모델

- FOSS 커뮤니티의 구조와 역학을 이해하기 위해 "Onion 모델"은 유용한 프레임워크를 제공합니다.
  - 리더: 프로젝트 창립자 또는 장기 유지 관리자. 비전을 설정하고 중요한 결정을 내린다.
  - 핵심 멤버: 주요 섹션을 관리하는 신뢰할 수 있는 기여자 코드베이스.
  - 활동적인 개발자: 코딩, 버그 수정, 기능 개발에 정기적으로 기여하는 사람입니다.
  - 주변 장치 개발자: 특정 작업에 집중하여 가끔 기여하는 사람입니다.
  - 버그 수정: 문제를 식별하고 수정하여 안정성을 강화합니다.
  - 독자: 기여하지 않고도 문서와 코드에 참여합니다.
  - 수동 사용자: 적극적으로 참여하지 않고 소프트웨어를 사용합니다.

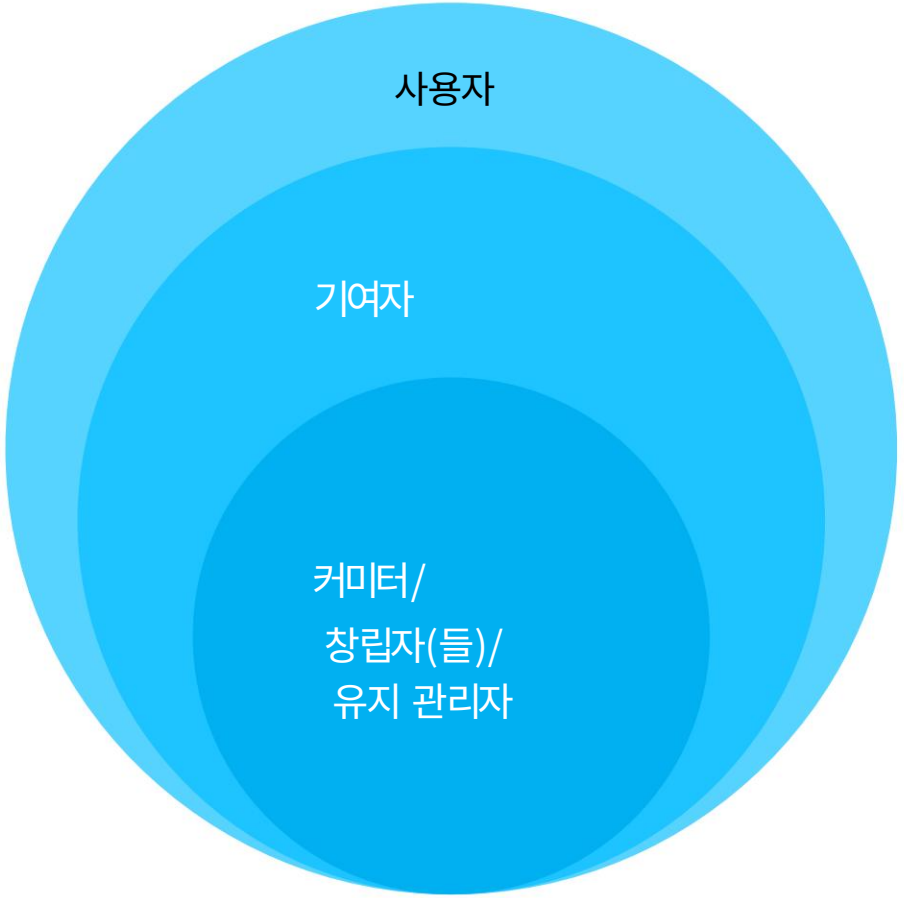
오픈소스 프로젝트 커뮤니티-FOSS의 Onion 모델



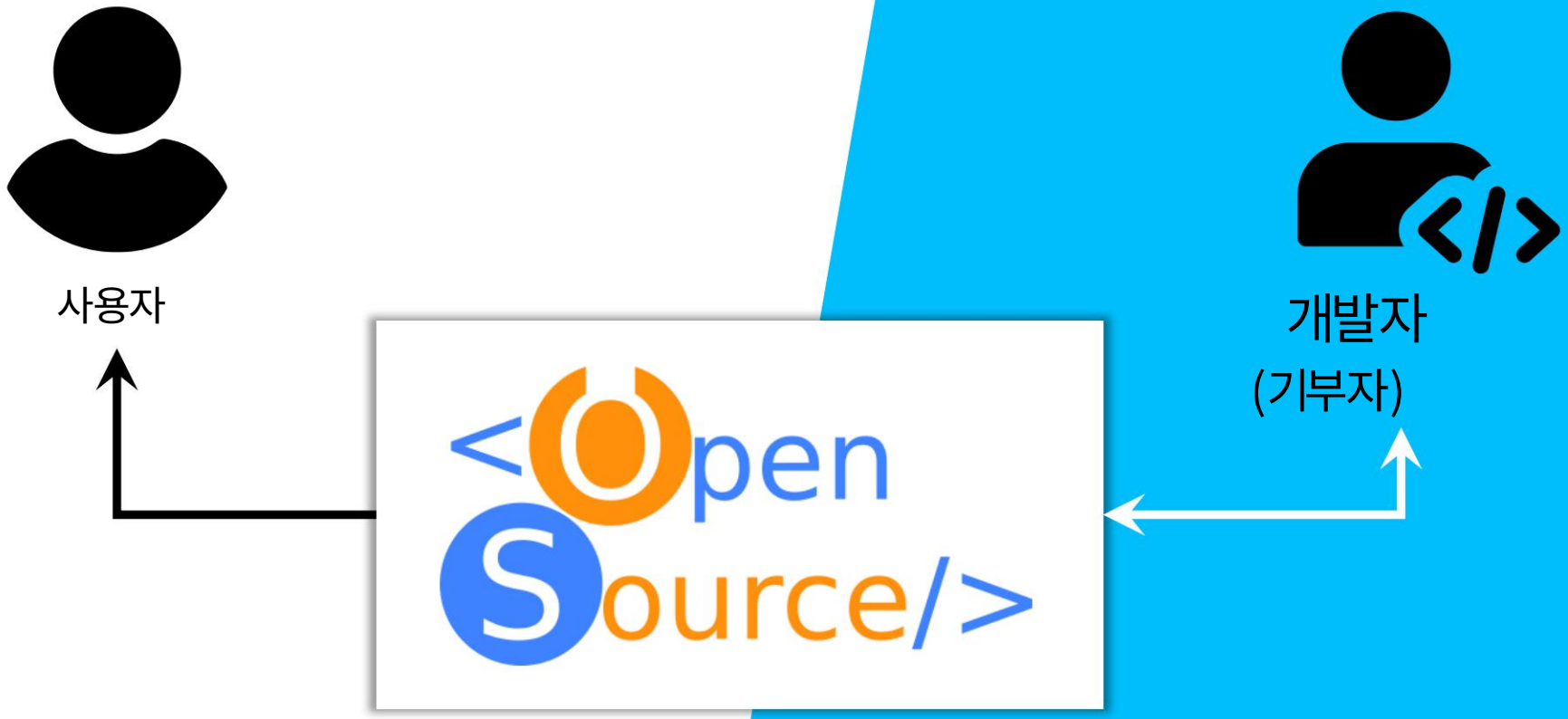
• 프로젝트 리더 • 핵심 멤버
• 커밋(쓰기) 권한이 있음 •
대량 작업 수행, 패치 검토

• 개발자 • 작은 기능
제공, 버그 수정 • 패치 제출(커밋 권한 없음)

• 프로젝트를 알고 사용하는 사람들 그렇다면, 의견과 피드백을 통
• 해 도움을 주세요.



오픈소스 프로젝트 커뮤니티-FOSS의 Onion 모델



오픈소스 프로젝트는 관심 있는 모든 사람에게 전환이 환영받을 수 있는 방식으로 운영되어야 합니다.

오픈소스 프로젝트 커뮤니티-FOSS의 Onion 모델

기여자 유형

핵심 기여자

- 프로젝트에 참여한 가장 선임 및 경험이 풍부한 사람들
- 지침을 제공하고 새로운 커뮤니티 멤버 에 대한 멘토링
- 커밋 비트를 유지합니다. 프로젝트에 대한 변경 사항을 승인할 수 있습니다.

비핵심 기여자

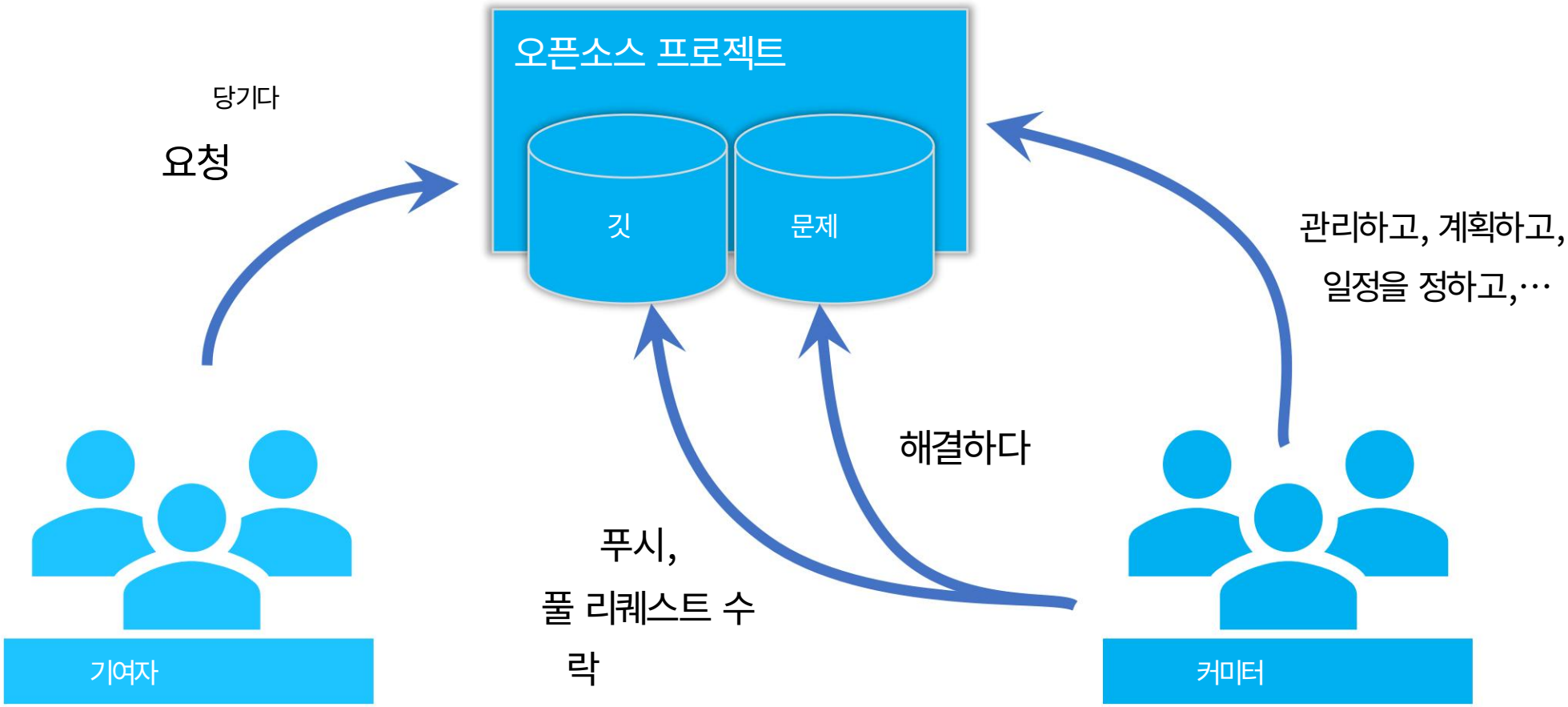
- 정기적인 기부를 하세요
- 이슈 논의에 적극 참여
- 커뮤니티 의 새로운 멤버 에게 피드백을 제공합니다 .

새로운 기여자

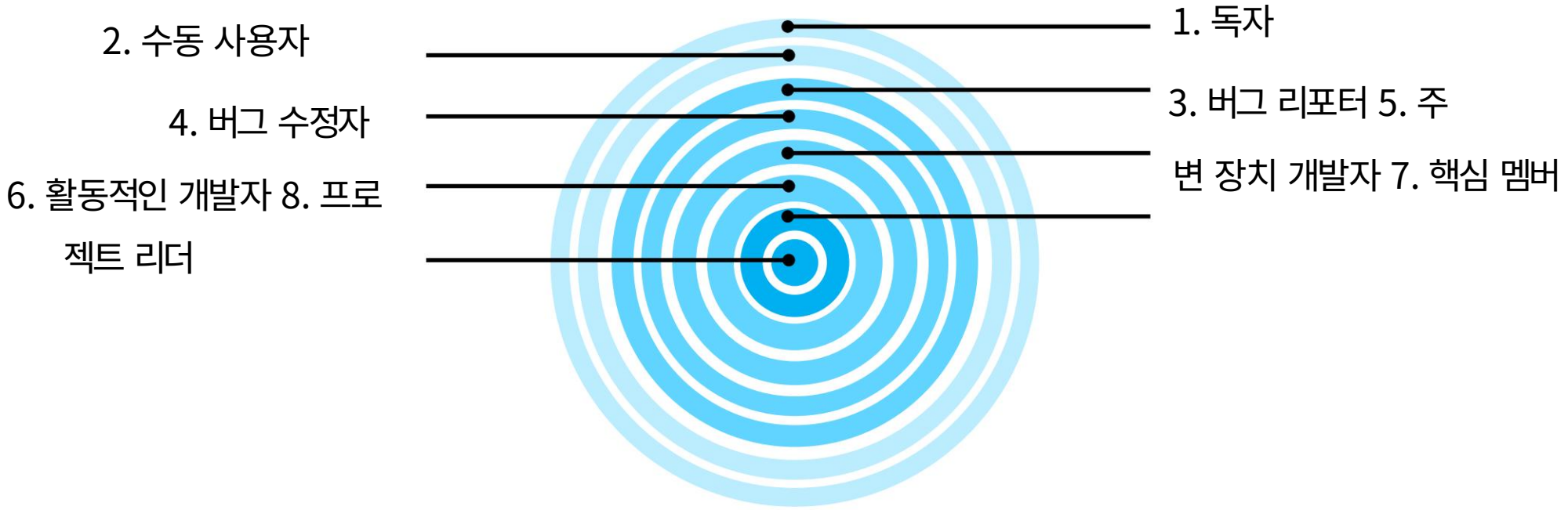
- 프로젝트에 기여하고자 하는 사람들
- 프로젝트 구조와 상호 작용하는 방법을 계속 학습하고 있습니다.

지역 사회

오픈소스 프로젝트 커뮤니티-FOSS의 Onion 모델

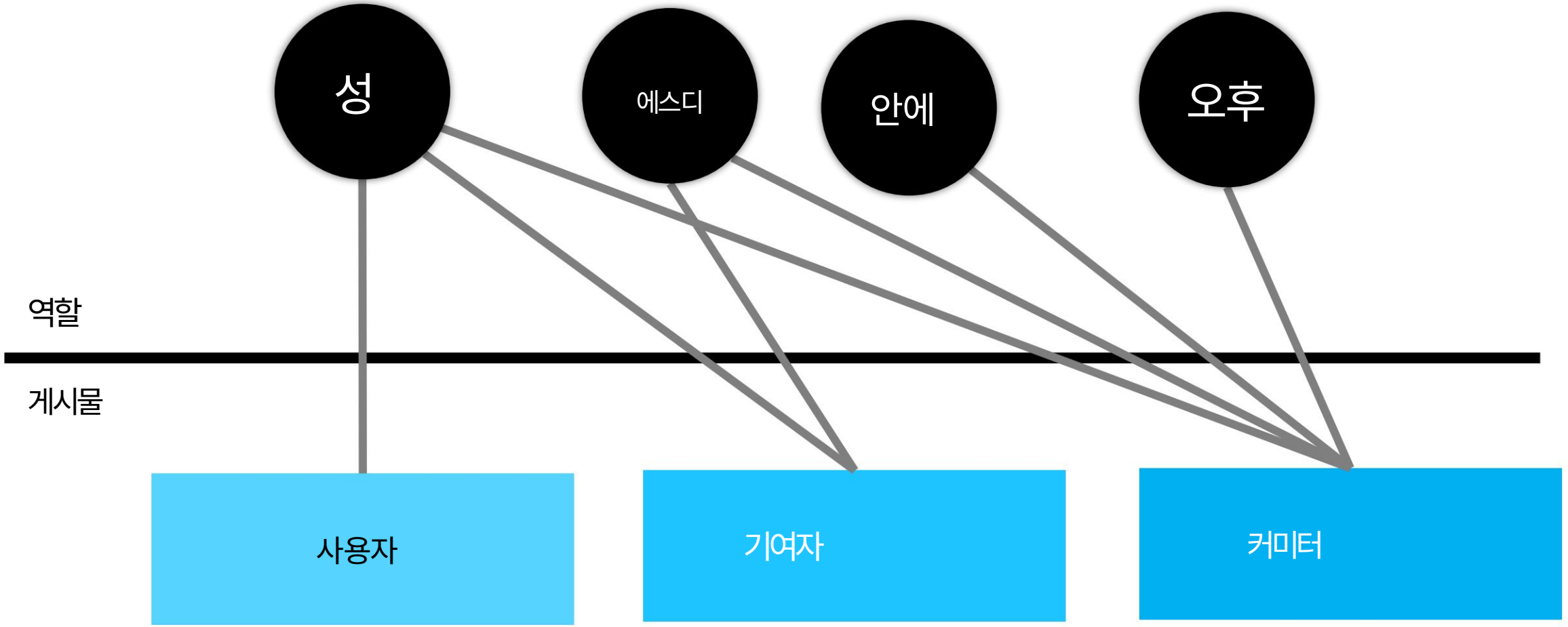


오픈소스 프로젝트 커뮤니티-FOSS의 Onion 모델



사용자 | 기여자 | 커미터

오픈소스 프로젝트의 역할과 직위



오픈소스에 기여하는 이유는 무엇입니까?

• 오픈 소스에 기여하는 것은 상상할 수 있는 모든 기술을 배우고, 가르치고, 경험을 쌓는 보람 있는 방법
이 될 수 있습니다 .

• 사람들이 오픈 소스에 기여하는 이유는? 수많은 이유가 있습니다! • 의존하는 소프트웨어 개선 • 기존
기술 개선 • 비슷한 것에 관심이 있는 사람들
을 만나기 • 멘토를 찾아 다른 사
람들에게 가르치기

• 명성(및 경력)을 키우는 데 도움이 되는 공개 아티팩트 구축 • 사람 기술 학습 • 아무리
작은 변화라도 변화를 만
들 수 있는 능력은 힘을 줍니다.

기여한다는 것은 무엇을 의미합니까?

새로운 오픈소스 기여자라면 그 과정이 무섭기도 합니다. 어떻게 하면 적합한 프로젝트를 찾을 수 있을까요? 코딩하는 법을 모른다면요? 뭔가 잘못되면요?

걱정하지 마세요! 오픈소스 프로젝트에 참여하는 방법은 다양하며 , 몇 가지 팁을 통해 경험을 최대한 활용하는 데 도움이 될 것입니다.

기여한다는 것은 무엇을 의미합니까?

- 이벤트 계획을 좋아하시나요?
  - @fzamperin이 한 것처럼 프로젝트에 대한 워크숍이나 미팅을 조직합니다.
    노드스쿨
  - 프로젝트 컨퍼런스를 조직합니다(컨퍼런스가 있는 경우)
  - 커뮤니티 구성원이 적합한 컨퍼런스를 찾고 제안서를 제출하도록 돕습니다.
    말하기 위해
- 디자인을 좋아하시나요?
  - 프로젝트의 유용성을 개선하기 위해 레이아웃을 재구성합니다.
  - 프로젝트 탐색을 재구성하고 개선하기 위해 사용자 연구를 수행합니다.
    Drupal이 제안하는 것과 같은 메뉴
  - 프로젝트가 일관된 시각적 특징을 갖도록 돕기 위해 스타일 가이드를 작성합니다.
    설계
  - hapi.js의 기여자들이 한 것처럼 티셔츠나 새로운 로고를 위한 아트를 만들어보세요.

기여한다는 것은 무엇을 의미합니까?

- 글쓰기를 좋아하시나요?
  - 프로젝트 설명서 작성 및 개선 • 프로젝트 사용 방법을 보여주는
    예제 폴더 정리 • 프로젝트에 대한 뉴스레터 시작 또는 메일링 목록에서 하이라이트
    정리 • PyPA 기여자처럼 프로젝트에 대한 튜토리얼 작성 • 프로젝트 문서에 대한 번역을 작성합니다.

- 정리하는 것을 좋아하시나요? •
  중복된 문제에 대한 링크와 새로운 문제 레이블을 제안하여 문제를 정리하세요.
  조직된
  - @nzakas가 한 것처럼 열려 있는 문제를 살펴보고 오래된 문제를 닫을 것을 제안합니다.
    ESLint
  - 최근에 열린 문제에 대해 논의를 진행하기 위해 설명 질문을 합니다.
    앞으로

기여한다는 것은 무엇을 의미합니까?

• 코딩을 좋아하시나요? • @dianjin

이 Leaflet에서 한 것처럼 해결해야 할 열린 이슈를 찾으세요. • 새로운 기능 작성
에 도움을 줄 수 있는지 물어보세요. • 프로젝트 설정 자
동화 • 도구 및 테스트 개선 • 사람들
을 돕는 것을 좋아하시나요?

• 예를 들어 Stack Overflow (이 Postgres 예제와 같은) 에서 프로젝트에 대한 질문에 답하십시오. 또는 Reddit • 공개 문제에 대한 사
람들의 질문에 답변 • 토론 게시판이나 대화 채널의 조정을 돕습
니다.

기여한다는 것은 무엇을 의미합니까?

- 다른 사람의 코딩을 돕는 걸 좋아하시나요?
  - 다른 사람의 제출물에 대한 코드 검토 • 프로젝트 사용 방법에 대한 튜토리얼 작성 • @ereichert가 @bronzdoc에 대해 한 것처럼 다른 기여자를 멘토링하겠다고 제안

기여할 프로젝트 찾기

- 이제 오픈소스 프로젝트가 어떻게 작동하는지 파악했으니, 기여할 프로젝트를 찾을 시간입니다!

- 다음 리소스 중 하나를 사용하여 새 프로젝트를 발견하고 기여할 수도 있습니다. • GitHub Explore • 오픈소스 금요일
  • 첫 타이머에 한함 • 코드 트리아지 • 24개의

풀 리퀘스트 • 획득 가능 • 기

여자-닌자 • 첫 번째 기여

- 소스 정렬

수업 활동

오픈소스 프로젝트의 역할 과 책임

오픈소스 프로젝트의 역할과 책임

– 카드 분류 활동

- 사례 연구: 도전 속에서 오픈소스 프로젝트 관리
  - 여러분은 웹 개발에 사용되는 인기 있는 JavaScript 라이브러리를 관리하는 오픈 소스 프로젝트에 참여하고 있습니다. 이 프로젝트는 빠르게 성장하고 있지만 커뮤니티가 확장되면서 새로운 과제가 생겨났습니다. 여러분은 복잡한 프로젝트 시나리오를 탐색하면서 역할과 책임을 분류하기 위해 그룹 으로 일하게 될 것입니다 .

오픈소스 프로젝트의 역할과 책임

- 지침: 1. 역할:

1. 사용자 2.
기여자
3. 커미터 4. 프로젝트
리더 2. 책임: 1. 버그 보고
2. 기능 개발 3. 코드 검토 4. 릴리
스 관리 5. 문서 작성
6. 보안 감사 수행 7. 프로젝트
로드맵 계획 3. 시나리
오: 각 그룹은 다음과 같은 실
제 과제를 제시하는 시나리오 카드를
받습니다. 1. 주요 보안 취약성 2. 충돌하
는 기능 요청 3. 새로운 기여자 코드 품
질 문제 4. 버그 백로
그 위기 5. 커뮤니티 갈등

샘플 시나리오: 핵심 팀을 압도하는 새로운 기여자 유입

- 귀하의 오픈소스 프로젝트가 인기를 얻어 많은 사람들의 관심을 끌었습니다.
  코드를 제출하는 새로운 기여자. 그러나 그들의 기여의 상당수는 제대로 테스트되지 않아 핵심 팀이 문제를 검토하고 수정해야 할 일이 더 많아 졌습니다 . 핵심 팀은 풀 리퀘스트의 양 에 압도당해 따라잡기 위해 고군분투하고 있어 기여자들 사이에 지연과 좌절이 발생하고 있습니다 .

- 과제:
  - 품질이 낮은 대량의 기여 관리
  - 프로젝트 표준을 유지하면서 새로운 참여자의 참여를 유지합니다.
  - 검토 프로세스를 가속화하는 동시에 코드 품질을 보장합니다.

샘플 시나리오: 핵심 팀을 압도하는 새로운 기여자 유입

- 관련 역할:
  - 새로운 기여자: 작은 기능이나 수정 사항을 제공하고 멘토링을 받습니다.
  - 경험이 풍부한 기여자: 멘토 역할을 하여 새로운 기여자를 검토하고 안내합니다.
    기여자.
  - 커미터: 단계적 검토 시스템을 구현하고 우선순위가 높은 사항에 집중합니다.
    풀 리퀘스트.
  - 프로젝트 리더: 품질과 효율성을 유지하기 위해 기여 지침을 개발하고 시행합니다.

샘플 시나리오: 핵심 팀을 압도하는 새로운 기여자 유입

- 행동 계획:
  1. 온보딩 및 멘토링:
     - 경험이 풍부한 기여자를 신규 기여자의 멘토로 지정하십시오. 이를 통해 프로젝트 교육에 도움이 됩니다.
       표준을 준수하고 코드 제출의 품질을 개선합니다.
     - 새로운 기여자가 다음을 이해하도록 돕기 위해 온보딩 문서 및 비디오 튜토리얼을 만듭니다.
       코드를 제출하기 전에 워크플로와 코딩 표준을 확인합니다.
  2. 간소화된 검토 프로세스:
     - 간단한 기여가 핵심 커미터에게 에스컬레이션되기 전에 새로운 기여자 또는 주니어 개발자가 먼저 검토하는 계층적 코드 검토 프로세스를 구
       현합니다 . 이렇게 하면 상급 검토자의 부담이 줄어듭니다.
     - CI(Continuous Integration) 도구를 사용하여 스타일 오류와 같은 기본적인 문제를 감지하여 검사를 자동화합니다.
       인간 검토자가 시간을 들여 검토하기 전에 테스트가 누락되었거나 빌드가 실패한 경우가 있습니다.
  3. 기여 지침 개선:
     - 프로젝트 리더는 명확한 지침을 포함하여 자세한 기여 지침을 작성해야 합니다.
       풀 리퀘스트, 코드 스타일 규칙, 테스트 요구 사항 제출. 이러한 가이드라인은 제출물이 최소 품질 기준을 충족하는지 확인하기 위해
       새로운 기여자와 공유할 수 있습니다.
  4. 참여 및 피드백:
     - 풀 리퀘스트에 대한 새로운 기여자에게 정기적인 피드백을 제공하고 문제점을 지적합니다.
       개선 및 모범 사례 장려. 이를 통해 기여자로서 성장하고 지원을 받는 느낌을 받는 데 도움이 됩니다.
     - 기여도가 낮더라도 새로운 기여자의 기여를 인정하여 동기를 부여합니다 .
       작습니다. 여기에는 커뮤니티 회의나 블로그 게시물에서의 외침이 포함될 수 있습니다.

시나리오 1 - 팀 1: 주요 보안 취약점

- 귀하의 프로젝트에서 심각한 보안 결함이 발견되었습니다. 사용자들은 이 결함이 잠재적으로 민감한 데이터를 노출시킬 수 있다고 보고하고 있습니다.
  이 문제는 즉각적인 주의가 필요하지만, 일부 주요 기여자는 앞으로 며칠 동안 이용할 수 없습니다. 이 보안 수정을 우선 순위로 지정하고 가능한 한 빨리 패치를 릴리스 해야 합니다.
- 과제:
  - 시간적 압박: 문제를 해결하기 위해 48시간이 주어집니다.
  - 수석 보안 관리자는 코드 검토에 참여할 수 없습니다.

시나리오 2 - 팀 2: 대기업의 기능 요청

- 귀하의 프로젝트에 의존하는 대규모 기업이 새로운 기능을 요청했습니다. 그러나 이 기능을 구현하려면 기존 아키텍처를 크게 변경해야 합니다. 일부 기여자는 이러한 변경의 잠재적 위험에 대해 우려하고, 다른 기여자는 다른 중요한 기능의 개발이 늦어질 것이라고 생각합니다.

- 과제:
  - 이해 관계자의 요구 사항과 지속적인 프로젝트 개발 간의 균형을 유지합니다.
  - 변경으로 인해 기존 기능이 손상되지 않는지 확인합니다.

시나리오 3- 팀 3: 코드베이스 재구성 제안

- 핵심 기여자 중 일부는 장기 유지관리성을 개선하기 위해 전체 코드베이스를 재구성할 것을 제안했습니다. 그러나 이러한 재구성은 기능 개발을 지연시키고 현재 사용자의 이전 버전과의 호환성을 손상시킬 위험을 초래합니다.

- 과제:
  - 장기적인 유지 관리 가능성과 즉각적인 사용자 요구 사항 간의 균형 유지.
  - 구조 조정 중에 이전 버전과의 호환성 문제를 방지합니다.

시나리오 4- 팀 4: 버그 백로그 위기

- 사용자가 보고한 버그 수가 상당히 증가했고, 프로젝트는 따라잡기 위해 고군분투하고 있습니다. 일부 사용자는 프로젝트의 신뢰성에 대한 신뢰를 잃기 시작했습니다. 팀은 다른 중요한 작업을 계속 진행하는 동시에 버그 백로그를 효율적으로 처리할 방법을 알아내야 합니다 .

- 과제:
  - 증가하는 버그 백로그 관리.
  - 버그 수정과 기능 개발 사이에 리소스 할당.

시나리오 5 - 팀 5: 커뮤니티 내 갈등

- 핵심 기여자 두 명이 프로젝트 방향에 대해 격렬한 의견 불일치를 겪고 있습니다. 이 갈등으로 인해 의사 결정이 늦어지고 커뮤니티 내에서 긴장이 고조되고 있습니다. 상황을 중재하고 프로젝트 목표와 일치하는 해결책을 찾아야 합니다.

- 과제:
  - 지역 사회 내 갈등 중재.
  - 의견 불일치를 해소하면서도 프로젝트의 추진력을 유지합니다.

시나리오 6- 팀 6: 제한된 리소스로 새로운 릴리스 마감일

- 프로젝트의 주요 릴리스를 계획했지만, 몇몇 기여자가 개인적인 이유로 작업을 할 수 없는 경우.

게다가 이 프로젝트는 테스트와 품질 보증에 필요한 리소스가 제한적입니다.

출시 기한이 빠르게 다가오고 있으며, 사용자와 이해관계자들은 이를 맞춰야 한다는 압력을 받고 있습니다.

- 과제:
  - 기여자의 참여 가능성이 제한적입니다.
  - 테스트 리소스가 부족합니다.
  - 발표 기한을 맞춰야 한다는 압력.

시나리오 7- 팀 7: 외부 종속성 업데이트로 인한 문제 발생

- 귀하의 프로젝트는 여러 외부 라이브러리에 의존하고 있으며, 최근 해당 라이브러리 중 하나에 대한 업데이트로 인해 호환성 문제가 발생하고 있습니다. 일부 기여자는 이전 버전으로 되돌리기를 주장하는 반면, 다른 기여자는 프로젝트가 새로운 변경 사항에 적응해야 한다고 생각합니다 .

- 과제:
  - 외부 변화에 대한 적응과 안정성 간의 균형.
  - 외부 의존성으로 인해 발생하는 잠재적 중단 관리

시나리오 8- 팀 8: 빠른 새로운 기능에 대한 사용자 요청

- 귀하의 프로젝트는 인기를 얻었고, 사용자들은 엄청난 속도로 새로운 기능을 요청하고 있습니다. 기여자들은 수요를 따라잡기 위해 고군분투하고 있으며, 일부 기능은 서두르면서 버그와 성능 문제가 발생합니다. 빠른 기능 개발과 품질 유지를 균형 있게 조절할 방법을 찾아야 합니다 .

- 과제:
  - 신속한 기능 개발을 위한 사용자 기대치 관리.
  - 새로운 기능을 빠르게 도입하는 동시에 코드 품질을 유지합니다.

시나리오 9- 팀 9: 기여자 번아웃

- 몇몇 핵심 기여자는 증가하는 업무량으로 인해 번아웃을 느꼈다고 표현했습니다 . 그들은 프로젝트에 필수적이지만, 물러서기 시작했습니다. 프로젝트가 계속 진행되도록 하면서 번아웃을 줄일 방법을 찾아야 합니다.

- 과제:
  - 기여자 소진을 해결하고 주요 기여자를 유지합니다.
  - 기여자 감소에도 불구하고 프로젝트 추진력 지속 보장 유효성.

독서 자료

- 책
  - 오픈소스 소프트웨어 제작 성공적인 무료 소프트웨어 운영 방법
    프로젝트 → 9장
  - 오픈소스 및 자유 소프트웨어 라이선싱 이해 → 1장: 오픈소스 라이선싱, 계약 및 저작권법

- https://www.youtube.com/watch?v=02m1T9c7CFk
- https://www.coursera.org/lecture/open-source-software-development-methods/open-소스-거버넌스-모델-TEhwM
- https://hackernoon.com/what-i-learned-from-building-my-first-successful-open-source-프로젝트-72f7429145a0
- https://opensource.guide/how-to-contribute/
- https://fossa.com/blog/apply-license-open-source-software-project/
- https://opensource.guide/legal/

감사해요

영업시간 : 월요일-금요일 (1000 - 1800)

수업과 관련된 회의나 어떤 종류의 토론이든 이메일을 보내주세요.

jamil@세종.ac.kr