차례

Introduction	1.1
감사의 글	1.2
역자 서문	1.3
개괄: 기업에서의 오픈소스	1.4
왜 기업과 정부는 오픈소스로 전환하는가?	1.5
저작권이나 심지어 코드보다 더 중요한 것	1.6
오픈소스를 이해하기 위한 기초	1.7
오픈소스 코드의 도입과 사용	1.8
프로젝트 커뮤니티에의 참여	1.9
오픈소스 프로젝트에 기여하기	1.10
오픈소스 프로젝트의 시작	1.11
오픈소스와 클라우드	1.12
결어	1.13
저자 소개	1.14
역자 소개	1.15
외부 링크	1.16
용어 사전	1.17

Open-Source-in-the-Enterprise 한글 리뷰본

=====

리뷰 안내

현재 pdf 파일은 한글 리뷰본입니다. 리뷰가 모두 끝나면 정리되어, 온라인 및 책자로 배포될 예정입니다. 리뷰는 이 pdf 파일에 메모(스티커노트)를 추가하거나 고치기는 뭐한데 영 이상한 부분은 형광펜(텍스트강조)으로 줄을 그어주시면 잘 보고 반영하겠습니다.

여러 사람이 번역을 하느라 말투가 조금 다릅니다. 그리고 일부는 존댓말로 일부는 반말로 번역이 되었습니다. 우선 잘못 전달된 의미와 중대한 단어 오류, 문맥상 더 의역이 필요한 경우, 그리고 아름다운 문장으로 만들기 위한 리뷰를 해서 아래 주소로 보내주시면 감사하겠습니다.

- slack : opensource-enterprise.slack.com
- slack 초대 link: https://join.slack.com/t/opensource-enterprise/shared_invite/enQtNTMzMzgyNTk5NjAxLTE3NTI1NTg0OTM4MzdhOTk2MTFjZGQ5ZjkwOTdiNjIwYWVhZmMzMmNiY2I2YWNhMzlhNDk0MzJmYmIyMmU3OTE
- Email : ykhl1itj@gmail.com
- Facebook Message : https://www.facebook.com/minsuk.lee0
- 또는 이 PDF 파일을 발견한 곳에 댓글로 알려주시면 됩니다.

=====

Open Source in the Enterprise의 한글 버전입니다.

이 책의 원본은 여기에서 PDF 파일을 받으실 수 있습니다.

이 책의 번역과 리뷰는 여러 분들의 기여에 의해 이루어졌습니다.

- (임시) 작업 분담표 : http://bit.ly/OSE-E2K
- 번역을 위한 단어장

개발 환경

이 프로젝트는 markdown 문서 형태로 바로 읽거나 편집해도 상관없지만, 오픈소스 커뮤니티에서 널리 쓰이는 전자출판 방식인 gitbook을 사용해서 책의 형태로 배포할 수도 있습니다. (예시: #)

개발 환경 설정

gitbook을 설치하기 위해서는 아래와 같은 조건이 필요합니다.

- Windows, Mac OS X 혹은 Linux 운영 체제.
- NodeJS (v4.0.0 이상 권장)

설치법은 아래와 같습니다:

```
npm install gitbook-cli -g
```

local에서 띄우기

프로젝트가 저장된 디렉토리로 이동한 뒤 아래 명령을 입력합니다:

```
gitbook serve
```

이제 웹 브라우저 상에서 `localhost:4000` 으로 접속하면 결과물을 바로 확인할 수 있습니다. commit하기 전 작업물을 확인하는 데 유용합니다.

ebook 생성하기

ebook 생성 기능을 사용하기 위해서는 calibre toolset과 npm의 ebook-convert 패키지가 추가로 필요합니다. 설치하기 위해서는 아래 명령어를 입력합니다:

```
npm install ebook-convert
```

프로젝트를 전자책 (enterprise-oss-book-kr.pdf) 형식으로 변환하기 위해서는 아래와 같이 하면 됩니다:

```
gitbook pdf ./ ./enterprise-oss-book-kr.pdf    # pdf 파일 생성
gitbook epub ./ ./enterprise-oss-book-kr.epub  # epub 파일 생성
gitbook mobi ./ ./enterprise-oss-book-kr.mobi  # mobi 파일 생성
```

기타

gitbook의 자세한 사용법에 대해서는 아래를 참조하시기 바랍니다:

- Setup and Installation of GitBook
- Generating eBooks and PDFs

목차 및 작업 현황

```
1차 번역 완료
1차 리뷰 완료
PDF를 배포하여 외부 리뷰 준비중
```


- Acknowledgments
- 역자 서문
- Table of Contents
- Open Source in the Enterprise
- Why Are Companies and Governments Turning to Open Source?
- More Than a License or Even Code
- Groundwork for Understanding Open Source
  - Terminology: Free and Open Source
- Adopting and Using Open Source Code
  - Create and Document an Internal Open Source Policy
  - Formalize Your Strategy Through an OSPO
  - Build Ties Throughout the Company
  - Assess Potential Projects
  - Comply with the License
  - Manage Community Code as Seriously as the Code You Create
  - Change Your Reward and Management Structure

    - Ego and Open Source
- Participating in a Project's Community
    - Quality and Security: A Comparison of Open and Closed Source
- Contributing to Open Source Projects
    - Establish the "Why" Throughout the Company
    - Hire from the Community
    - Develop Mentoring and Support
    - Set Rules for Participation
    - Foster Open Communication
- Launching an Open Source Project
    - Choose a License
    - Open the Code Right Out of the Gate
    - Use Best Practices for Stable Code
    - Set Up Public Discussion Forums
    - Make Life Easy for Newbies
    - Keep Up Communication
    - Adopt Metrics and Measurement
    - Release the Project to an Independent Governance Organization
- Open Source and the Cloud
- Conclusion
- About Authors
- 역자 소개 (각자 자기 소개 추가 필요)
- 이 책에 있는 모든 외부 링크 (확인 필요)

감사의 글

오픈소스 프로젝트와 커뮤니티를 만들고 성장시키기 위해서는 많은 이의 참여가 필요합니다. 이 책은 여러 프로젝트, 프로세스, 책, 보고서, 모범 사례 그리고 재단, 기업, 커뮤니티와 개인들이 했던 모든 형태의 기여 결과를 참조한 결과물입니다. 우리가 참조한 내용들이야말로 이 책의 독자들이 오픈소스로 나아가는데 도움이 되는 엄청나게 가치로운 자원이라고 믿으며 각 주제에 대한 깊이있는 내용을 제공할 것이라고 확신합니다.

오픈소스 20주년을 맞이하여, 의미 있는 코드, 도구, 강좌, 경험, 프로세스의 공유 그리고 오픈소스가 모든 면에서 이롭다는 지지를 표명해주는 등 어떤 형태로든 오픈소스에 기여를 해주신 오픈소스 커뮤니티의 모든 구성원 개개인에게 깊은 감사의 뜻을 표합니다.

광범위한 의견과 조언을 해주신 리뷰어들인 Cecilia Donnelly, Karl Fogel, James Vasile, Chris Aniszczyk, Deborah Nicholson, Shane Coughlan, Ricardo Sueiras, Henri Yandell 및 Adrian Cockcroft 씨에게 감사를 드립니다. 마지막으로, O'Reilly 미디어와 AWS 팀이 여러 리소스들을 이렇게 결집된 책을 만들 수 있도록 지원해주셔서 고맙다는 인사를 드립니다.

— Andy and Zaheda

역사 서문

기업이 잘 되는 이유 가운데 하나는 놀라운 서비스나 제품을 적절한 시점에 시장에 출시하는 것 입니다. 그 이면에 늘 오픈소스가 있습니다. 시장이 기대하는 기술적 혁신의 수준과 속도는 너무 높고 빨라서 기업 내부 개발자들의 노력만으로는 얻어내기 힘든 세상이 되었습니다. 다른 누군가에 의해 만들어진 오픈소스 결과물과 그들로 구성된 커뮤니티와의 협력을 통해서만 가능합니다.

우리 기업들은 오픈소스를 매우 많이 활용하고 있으면서도 아직 오픈소스를 잘 알지는 못하고 있습니다. 그래서 우리가 오픈소스를 잘 쓰면 얻을 수 있는 많은 것들을 놓치고 있습니다.

이 책은 오픈소스가 왜 우리 기업의 성장에 도움이 되는지 그리고 우리가 오픈소스를 통해 얻을 수 있는 것이 무엇인지를 알려줍니다. 이 두껍지 않은 책은 오픈소스를 온전히 활용하여 최고의 이득을 얻어내는 단계적인 방법과 구체적인 액션 아이템들을 잘 설명하고 있습니다. 아마도 소프트웨어 개발을 약간만이라도 이해하고 있는 분이라면 이 책에서 안내하는 방법들을 회사에 어떤 식으로 적용할 수 있을지, 어떤 어려움이 있을지 잘 이해할 수 있을 것으로 보입니다. 이미 오픈소스로 프로젝트를 운영하고 있거나 오픈소스 커뮤니티에 능동적으로 참여하고 있는 기업들은 물론이고 아직 소극적으로 활용하는 단계에 있는 모든 기업들에게 이 책은 훌륭한 지침을 제공합니다.

다시 말하지만, 우리 기업들은 오픈소스를 매우 많이 활용하고 있으면서도 아직 오픈소스를 잘 알지 못하고 있습니다. 그래서 이 책을 번역하는데 참여한 우리들을 포함한 많은 개발자들이 오픈소스 커뮤니티에 대한 우리 기업들의 기여가 아직 부족하다는 느낌을 받습니다. 이 책을 계기로 우리 기업들이 오픈소스를 더 잘 기술적으로 활용하고, 문화적으로도 변화하여 커뮤니티와 함께 성장할 수 있기를 기대합니다.

이 책은, 기꺼이 시간을 내주신 많은 분들이 조금씩 나누어 번역하고, 리뷰해 완성되었습니다. 이 책을 번역하는 과정 자체도 오픈소스 커뮤니티의 협업 개발 방식으로 진행되었습니다. 최종적으로 책으로 편찬하는 과정에는 한빛출판사의 도움도 있었습니다. 함께한 작업 과정도 재미있고 그 결과물이 자랑스럽습니다.

모쪼록 이 책을 읽은 분들과 기업들이 오픈소스 가치를 공유하고 특히 우리나라의 오픈소스 커뮤니티에 더 많은 기여를 해주시면 감사하겠습니다.

역자 일동.

엔터프라이즈에서의 오픈소스

자유 오픈소스 소프트웨어는 어디에나 있고, 전 컴퓨팅 분야에 걸쳐 사용되고 있습니다. GNU/Linux는 현재 가장 널리 쓰이고 있는 운영체제로서 전세계의 데이터센터를 움직이고 Android 기기들을 제어하고 있습니다. Apache Hadoop과 마이크로서비스 기반 클라우드 컴퓨팅을 지원하는 Docker나 Kubernetes와 같은 후속 오픈소스 기술들은 여러 분야에서 빅데이터 혁명을 불러일으켰습니다. 인공지능(AI) 분야는 TensorFlow와 Apache MXNet과 같은 기술들로 인하여 압도적으로 오픈소스가 주도하는 영역이 되었습니다. Amazon, Apple, Facebook, Google, Huawei, IBM, Intel, Microsoft, PayPal, Red Hat, Twitter 등 컴퓨팅 업계의 주요 기업들은 오픈소스 프로젝트들을 시작하고 유지하고 있습니다. 그들은 그 기술을 사용하는 다른 사람들을 사랑하는 마음으로 손해를 감수하면서 그 작업을 하고 있는 것이 아닙니다. 디지털 전환 또는 클라우드 상에서 서비스를 구축하는 모든 기업과 정부 역시 오픈소스 소프트웨어를 사용하고 있으며, 그 이유는 오픈소스 소프트웨어가 그들의 목적과 비즈니스에 이득이 되기 때문입니다.

이제는 조직의 크기나 분야와 상관없이 그들의 전략에 자유 오픈소스 소프트웨어를 포함해야 할 때입니다. 이 책은 오픈소스 커뮤니티로부터 얻어진 수십 년 간의 교훈을 요약하여, 업계 동향에 대한 지금 시점에서의 관점을 제시합니다. 이 책은 여러분들을 오픈소스 소프트웨어를 효과적으로 사용하고, 그에 기여하며 직접 오픈소스 프로젝트를 시작할 수 있도록 도울 것입니다.

기업들은 오픈소스를 활용하여 더 나은 소프트웨어를 얻을 수 있을 뿐만 아니라 커뮤니티가 중심에 있는 개발이 주는 역동성을 통해 창의성과 참여 회사들 사이의 협업을 위한 새로운 길을 개척할 수 있습니다. 이와 반대로, 오픈소스를 활용하지 못하는 기관들은 이를 효율적으로 사용하는 기관들에 비해 뒤처질 것입니다.

마지막으로, 영업 비밀과 비밀스러운 사업 계획들이 오픈소스 참여와 공존할 수 있다는 점을 언급할 가치가 있습니다. 미국 국가안보국과 영국 정부통신본부에서도 오픈소스 소프트웨어를 사용하는 것을 보면 누구에게나 오픈소스가 문제 없음을 알 수 있습니다.

왜 기업과 정부는 오픈소스로 전환하는가?

오픈소스 소프트웨어를 생산하고 사용하고 지원하는 데는 확실한 비즈니스 관점의 이유들이 있습니다. 그 이득은 다음과 같습니다:

배가되는 회사의 투자

오픈소스의 이익을 보여주는 유명한 원칙이 있습니다: "어떤 분야에서든 그 분야에서 가장 똑똑한 사람은 당신 회사에 없습니다." 그래서 오픈 프로젝트를 중심으로 혁신의 생태계는 성장합니다. 세계은행(World Bank)의 후원으로 작성된 최근 보고서에서도 프로젝트를 오픈하는 것이 재정적으로 도움이 된다는 증거를 찾을 수 있습니다. 지리공간과 관련된 그들의 프로젝트인 GeoNode에 대한 기여를 주의 깊게 추적한 결과, 세계은행의 자회사는 그 프로젝트에 약 백만 달러를 투자했지만 2백만 달러로 평가되는 다른 조직들의 투자가 들어옴으로써 이득을 얻었습니다.

최신 기술로부터의 얻는 이득

앞서 이야기했던 인공지능 프로젝트가 매우 좋은 예입니다. 당신 회사의 데이터 과학자들은 최고이자 가장 최신인 알고리즘의 구현 결과를 원하기 마련이며, 그것은 대부분 오픈소스로 존재합니다. 회사 내부에서 그것들을 다시 개발할 필요는 전혀 없습니다. 나아가, 회사는 그저 다운로드하여 설치만 하면 쓸 수 있는 도구들과 누군가 이미 만들어 공개한 코드들을 이용하여, 더 빠른 혁신을 이뤄낼 수 있습니다.

소프트웨어에 관한 지식의 전파

코드가 공개되고 특히 그 코드를 중심으로 튼튼한 커뮤니티가 성장하고 있을 때 그 프로젝트는 더 널리 사용됩니다. 처음에는 회사의 노력이 들어가야 하지만, 업계의 더 많은 사람들이 코드를 이해하고 기여할 수 있도록 만듭니다.

개발자 인재풀의 증가

프로젝트가 소스코드에 대한 여러 토론들과 함께 더 많이 사용된다면, 회사에 고용되어 코드와 그에 관련된 프로젝트에서 일할 수 있는 재능있는 개발자들이 포함된 큰 인재풀을 얻는 것과 다름이 없습니다.

회사에 소속된 개발자들의 능력 향상

지식의 전파는 많은 방향으로 이루어집니다. 개발자들은 코딩 역량을 높이는 가장 좋은 방법이 오픈소스 프로젝트에 참여하는 것임을 이미 알고 있습니다. 프로젝트 참여를 통해 그 분야에서 가장 잘하는 개발자들의 코드들을 보고 배울 수 있기 때문입니다. 이러한 이득은 오픈소스 개발자들을 고용하는 회사에 퍼지게 됩니다.

좋은 평판 쌓기

대부분의 사람들은 자랑하고 싶은 조직에서 일하고 싶어합니다. 오픈소스 프로젝트의 코드와 관행들을 채택하고 있다는 것은 그 조직이 매우 멋지다는 것을 의미합니다. 그리고 만약 당신이 코드를 오픈소스로 배포할 수 있고 그 코드를 다른 사람들이 채택한다면, 그 조직은 그 분야에서 선도적인 위치에 있다는 것과 개발 관행도 최고 수준에 있다는 증명하는 것입니다.

개발자들의 고용과 유지

좋은 개발자들은 많은 사람들에게 영향을 끼치는 재미있는 프로젝트에 참여하고 싶어합니다. 또 자신들의 능력과 기여가 널리 알려지기를 바라며, 전세계에 있는 동료들과 의견을 교환하는 것을 즐깁니다. 이러한 모든 고려사항들이 그들을 오픈소스로 끌어들이고 있으며, 만약 경쟁사가 어떤 프로젝트를 당신 회사보다 더 성공적으로 지원하고 있다면 개발자들은 당신 회사 대신 그 경쟁사에 자신의 재능과 명성을 사용할 것입니다.

더 빠른 스타트업과 프로젝트

오늘날 미친듯한 속도의 사회 및 비즈니스 환경 변화 속에서, 새로운 회사나 사업을 시작한다는 것은 이제 수 년이 아닌 수 개월안에 컨셉을 제품으로 만들어낸다는 것을 의미합니다. 커뮤니티와 함께 하면, 이미 존재하는 소프트웨어와 당신 스스로의 혁신을 기반으로, 개발 시간을 단축하고 제품의 핵심 경쟁력 부분에 개발자들이 쓰는 시간을 집중할 수 있습니다.

많은 국가들은 주요 오픈소스 정책과 이니셔티브들을 발족했습니다. 프랑스와 미국이 보여주듯이, 우리는 이제 오픈소스를 사용하는 것에 그치지 않고 오픈소스 개발과 오픈소스 커뮤니티에 대한 투자를 장려하는 정책적 변화를 관찰하고 있습니다. 일부 국가는 벤더들과 내부 개발자들에게 어느 곳이든 가능한 곳에서부터 오픈소스 라이선스와 오픈소스 개발 방법론을 사용할 것을 권장하는 "오픈소스 퍼스트 (open source first)" 전략을 시행하고 있습니다. 예를 들어, 프랑스 정부는 모든 기관이 만들어 낸 코드를 오픈소스화 해야 한다고

명시했습니다. 이런 정책들이 있기 때문에, 기관들은 소프트웨어 프로젝트를 실패하게 만들거나 엄청난 예산 초과를 부르는 악명높고, 구시대적이며, 느린 조달 관행 자체를 바꿀 수 있습니다. 다른 영역에서 이미 더 효율적이고 생산성이 높다고 증명된 오픈소스 방식은 최신이면서 변화에 빠르게 반응하기 때문에, 정부에서도 소프트웨어 개발 방식의 시작점이 되고 있습니다.

이제 여러 나라들은 각 기관이 가진 요구 사항들이 자기 나라 또는 전 세계의 다른 기관들과 서로 비슷하다는 것을 알고 있습니다. 적어도 오픈소스는 한 기관에 의해 이루어진 투자가 나머지 모두의 예산을 아끼는데 기여하며, 나아가 기관들은 각자의 요구 사항들을 공유하고 전통적인 오픈소스 개발 방식에 따른 협력을 함으로써 정부가 시민을 위해 더 좋은 서비스를 제공할 수 있는 소프트웨어를 개발할 수 있게 됩니다. 그리고 오픈소스를 통한 협업은 중소기업들, 개발 역량을 가진 시민들, 그리고 비영리 단체들에게 정부 서비스의 혁신에 기여할 수 있는 기회를 열어줍니다. 최종적으로 그렇게 만들어진 소프트웨어는 다양한 개발에서 상호 운용성을 높일 수 있는 공통적인 표준을 만들어 냅니다.

저작권이나 심지어 코드보다 더 중요한 것

오픈소스에서, 생산적인 커뮤니티와 그 커뮤니티의 개발 방식은 코드만큼이나 중요합니다. 공식적으로 오픈소스는 라이선스에 의해 정의됩니다. GNU General Public License, the Mozilla Public License, 그리고 Apache License 등이 자주 사용되는 라이선스에 속하며, 각 라이선스는 가끔 개정되기도 합니다. 그러나 현실적으로, 성공적인 오픈소스 프로젝트로 만들기 위해서는 라이선스 그 이상이 필요합니다.

많은 사람들이 아파치 재단의 "코드 이전에 커뮤니티(community before code)" 라는 원칙을 인용합니다. 한 컨퍼런스에서, 어떤 오픈소스 커뮤니티 리더는 이 원칙을 다음과 같이 설명했습니다:

만약 대단한 코드는 있지만 잘 동작하지 않는 커뮤니티가 있다면, 사람들은 떠나고, 코드는 위축될 것입니다. 거꾸로 잘 동작하지 않는 코드를 가진 훌륭한 커뮤니티가 있다면, 사람들은 코드를 고쳐나갈 것입니다.

이런 원칙은 회사 문화에도 확대 적용될 수 있습니다. 여러 다른 팀에서 온 개발자들로 커뮤니티를 만들고 그들이 더 큰 프로젝트 커뮤니티에서 생산적으로 일하게 하려고 할 때도, 이 원칙은 너무나 중요합니다.

우리는 이 책에서, 오픈소스 여정에 도움될 만한 참고 자료를 제시하면서, 이런 오픈소스 방법론들을 요약하려고 합니다. 더 자세한 정보는 다음에서 얻을 수 있습니다.

- Linux Foundation의 추가적인 읽을 거리 목록. 아마도 이 목록에서 가장 자주 언급되는 책은 Karl Fogel의 Producing Open Source Software(O'Reilly, 2018)와 Eric Raymond의 고전인 The Cathedral and the Bazaar(O'Reilly, 2009)입니다.
- Linux Foundation의 포괄적인 가이드. 이 가이드는 TODO Group에서 만들어졌으며, 오픈소스 원칙, 개발 방법론, 도구들을 채택한 여러 회사들의 오픈소스 프로그램 부서와 기여자들의 협업 결과물 입니다.
- opensource.com에 올라온 많은 질문과 답변, 리소스들.

오픈소스 개발 방식과 커뮤니티 활동은 매우 강력해서, 많은 회사들이 InnerSource라고 흔히 부르는 내부협동개발 프로세스에서도 따라하고 있습니다. 원한다면 또 다른 O'Reilly Media report에 기술한 것을 참고하여, 오픈소스 활동과 병행하면서 또는 독립적으로 따라해볼 수도 있습니다.

잘 진행되는 오픈소스 프로젝트를 바탕으로 유기적인 성장을 경험하지 못했던 조직들은 오픈소스 문화의 중요성을 과소 평가합니다. 오픈소스 문화는, 오늘날 많은 회사들이 가진 비밀스럽고, 위계적이고, 관리주도적인 문화와는 상당히 다릅니다. 오픈소스 프로젝트의 가치는 경청, 투명성, 협업, 전문성의 공유, 멘토링, 장점에 대한 인정, 기능 요구와 견해에 있어서의 다양성 존중, 비판을 받아드리는 자세 등을 포함합니다.

많은 회사들이 오픈소스 프로그램 부서(open source program office, OSPO)를 설립하고 있으며, 여기서 오픈소스가 길러지고, 지원되며, 육성되고, 공유되며, 회사 안과 밖에서 설명됩니다. 오픈소스 소프트웨어를 사용하고 기여하는데 있어서 비중있는 투자를 해왔던 큰 조직이라면 OSPO는 필수적입니다. OSPO는 다른 회사의 OSPO와도 협력을 하여, 오픈소스 개발과 커뮤니티를 지속가능하게 하는 모범 사례들을 공유합니다. OSPO에 대해 더 자세히 알고 싶다면 TODO Group의 사례연구를 참고하시면 됩니다.

오픈소스를 이해하기 위한 기초

오픈소스 소프트웨어를 설명하기에 앞서, 다른 곳에서 개발된 소프트웨어를 채택하는 방법, 어떤 프로젝트에 기여하는 방법, 자신의 오픈소스 프로젝트를 시작하는 방법 등 세 관점에서 오픈소스 소프트웨어에 대한 몇몇 오해를 풀어보고자 합니다.

오픈소스 소프트웨어는 품질이 낮거나 보안성이 떨어진다.

주요 소프트웨어 기업들이 오픈소스에 참여하면서, 이제 이 같은 오해는 자주 언급되지 않지만, 직접 이야기하지는 않더라도 아직 그렇게 생각하는 경우가 있습니다. 상용 소프트웨어 구매에 익숙한 사람들에게는 무료로 얻을 수 있는 소프트웨어가 높은 품질을 가질 수 있다는 점을 믿기 어려울 것입니다. 그러나 실제로 오픈소스 프로젝트는 다양한 자금 지원 전략으로 라이선스 비용을 대체해 왔습니다. 핵심은 오픈소스 프로젝트들이 당신의 회사에서도 도입하면 이득이 될 수 있는 수준의 엄격한 품질 프로세스를 채택하여 고품질의 소프트웨어 개발 프로세스를 가지고 있다는 것입니다. 보안에 대한 결함은 오픈소스와 그렇지 않은 소프트웨어 모두에서 발생합니다. 누구도 이를 피할 수 있다고 장담할 수 없습니다. 우리가 경험한 바로는 투명하고 큰 규모의 개발 커뮤니티를 가진 오픈소스가 보다 신속한 수정과 수정된 버전의 배포를 할 수 있다는 것입니다.

오픈소스 소프트웨어는 기술 지원이 부족하다.

많이 사용되는 오픈소스 프로젝트들은 조직과 개인 개발자들 양쪽으로부터 기술 지원을 받고 있습니다. 기술 지원이 한 회사에 종속되지 않는다는 점이 코드가 오픈되어 있다는 것의 큰 장점이기도 합니다. 작거나 시작한지 얼마 안 된 프로젝트의 경우에는 아직 지원 생태계가 충분히 활성화되지 않을 수 있어서 개발자들이 더 많은 시간을 쓰거나 커뮤니티로부터의 비공식적인 도움이 필요할 수도 있습니다. 당신이 즉시 수정이 필요한 치명적인 버그를 발견한 상황에서, 오픈소스가 아닌 소프트웨어라면 무관심한 벤더가 버그를 고쳐줄 때까지 기다리려야 하는 반면, 오픈소스의 경우 내부의 개발자가 고치거나 해당 코드를 잘 아는 외부 개발자를 고용하여 수정할 수 있다는 점이 감사할 수도 있습니다.

오픈소스 소프트웨어 프로젝트는 관리되지 않고 혼란스럽고 무료이다.

이 책을 읽어가면서 점점 명확해지겠지만, 성공적인 오픈소스 프로젝트들은 의사 결정, 코드 리뷰, 사용자 대응을 위한 프로세스를 보통의 조직들처럼 잘 정의하고 있습니다. 다른 사람들이 개발한 코드를 사용할 때는 정해진 규칙을 따라야 합니다. 코드에는 거의 항상 라이선스가 있지만 상용 코드와는 다릅니다. 개발자가 인터넷에 있는 코드를 자신의 제품에 복사해 사용하는 것은 거의 확실히 라이선스를 위반하는 것이며 법적으로 또는 다른 이유로 좋은 방법이 아닙니다. 오픈소스 이니셔티브 (Open Source Initiative)와 소프트웨어 자유보호협회 (Software Freedom Conservancy)에서 이에 관한 많은 토론이 이루지고 있습니다. 다음 섹션들에서 어떤 조직이 오픈소스 코드를 수용하기 위하여 최근에 사용되는 방법들에 대해 설명하려고 합니다.

오픈소스 소프트웨어를 사용하려면 내 코드도 공개하여야 한다.

이 부분은 앞의 오해와 반대됩니다. 분명히, 오픈소스 소프트웨어를 사용하기 전에 라이선스에 필요한 사항을 숙지하여야 합니다. 일부 라이선스에는 변경 사항들을 원 소스에 기여해야 한다는 규칙이 있으며, 나중에 이야기하겠지만 그 규칙이 없더라도 그렇게 하는 것이 이득으로 돌아옵니다. (때로 이러한 라이선스를 "바이러스 성"이라 일컫기도 하지만 해당 단어에 대한 부정적 의미 때문에 "카피-레프트, copy-left"가 보다 중립적인 용어입니다.) 변경한 부분을 기여해야한다는 규칙이 있는 경우를 포함하여 대부분 오픈소스 프로젝트는 라이브러리 형태로 배포되는데, 직접 개발한 코드를 그 라이브러리를 링크하는 경우에는 자신의 코드를 공개하지 않을 수도 있습니다 (예: GNU Lesser General Public License).

공개 저장소에 코드를 공개함으로써 사용자 기반과 커뮤니티를 확보할 수 있다.

아주 천천히 진화하는 오픈소스는 결코 잘 작동하지 않습니다. 오픈소스 프로젝트가 비즈니스 전략 상 의미있는 요소로 존재하는 경우에만 독점적인 프로젝트 대신에 채택될 것입니다. 오픈소스 프로젝트는 활성화된 커뮤니티의 일부로 존재할 때만 그 가치가 실현됩니다. 많은 경우에, 프로세스에 프로젝트 참여자들 사이의 상호 작용이 내재되어 있고, 그 상호 작용은 그 자체로 참여자들에게 매우 소중합니다. 이 같은 오픈소스의 역동성이 계속된 투자를 보상합니다. 책을 읽는 동안 이 부분이 어떻게 이루어지는 가에 대해 더 잘 알게 될 것입니다. 활성화되지 않은 프로젝트는 시간이 지남에 따라 정체 비용이 증가하여 혜택이 줄어듭니다.

오픈소스로 프로젝트를 하면 회사의 개발자들이 기술지원을 하는데 대부분의 시간을 쓰게 만든다.

오픈소스에서는 자신이 기술지원에 사용한 시간을 다른 사람이 커뮤니티에 기여한 결과로 보상받습니다. 물론 지원을 위해 쓸 수 있는 시간을 미리 정해야하지만, 회사는 내부 프로젝트의 데드라인을 맞추거나 과도한 지원 비용을 관리하기 위해 기술지원에 투입하는 시간을 조절할 수 있습니다. 성공적인 오픈소스 커뮤니티에서는 모든 멤버들이 교육이나 지원을 분담하며 한 회사가 전적으로 책임을 지지는 않습니다.

용어: 프리 오픈소스

무시할 수 있는 수준의 몇 예외 사항을 제외하면 자유 소프트웨어의 정의에 해당하는 모든 내용은 오픈소스의 정의에도 해당되고 그 반대의 경우도 마찬가지이기 때문에 자유 및 오픈소스라는 용어를 이 책에서는 서로 바꿔가며 사용합니다. 자유, 프라이버시 및 공유의 측면을 강조하려는 사람들이 자유 소프트웨어라는 용어를 사용하는 반면, 실용적이고 비지니스 관점의 이득을 강조하는 사람들이 오픈소스라는 용어를 사용합니다.

개발자가 소스코드를 공개하지 않으면서 무료로 배포하는 프로그램을 의미하는 프리웨어라는 용어는 더 이상 사용하지 말아야 합니다. 그런 소프트웨어는 지금 통용되는 프리 소프트웨어 분류에 속하지 않습니다. 진정으로 프리(또는 오픈소스)가 되려면 사용자가 소스코드를 수정하고 재배포 할 수 있는 라이선스 형태로 소스코드가 공개되어야 합니다.

오픈소스의 도입과 사용

우리는 이 책의 독자들이 오픈소스가 무엇을 제공할 수 있는지를 궁금해하고, 어쩌면 각자 가진 비지니스 요구를 해결할 수 있는 코드를 찾고 있다고 믿습니다. 이 절에서는 다른 사람의 오픈소스 코드를 성공적으로 사용하기 위해 도입해야 하는 주요 절차들을 요약합니다. 이 책의 앞부분에 인용된 자료들에는 이 절에서 언급할 내용들이 훨씬 더 자세하게 설명되어 있습니다.

내부 오픈소스 정책의 수립과 문서화

개발팀은 어떤 오픈소스 코드가 어디에 사용 중인지 정확히 알고 있어야합니다. 이에 대한 관리는 OSPO나 OSPO가 아직 만들어지지 않은 경우 내부 직원들오 구성된 가상적인 팀이 수행하도록 합니다. 이 관리에는 두 가지 주요 목적이 있습니다. 코드를 올바르게 사용하고 있음을 입증하기 위한 감사 기록을 남기고, 사용된 외부 오픈소스가 가진 라이선스 의무들을 준수하는지 확인하고자 하는 것입니다. 이 정보를 수집하는 것은 여러 가지 이유로 매우 중요해서 대부분의 조직에서는 개발 사이클 안의 자동화된 도구를 이용하여 이 정보를 수집합니다.

전략 문서를 작성하는 것은 관리자와 직원을 교육할 때 유용합니다. 크게 생각하고 당신이 성취하고자 하는 최종 상태를 목표로 삼으십시오. 동시에 비지니스 성과 측면에서 광범위한 상위 목표를 수립하십시오. 다음은 오픈소스가 조직을 위해 할 수 있는 것을 잘 설명하기 위하여 언급되는 몇가지 사항입니다:

- 인재를 끌어들이고 유지합니다.
- 사업의 민첩성을 높이고, 혁신을 유도하며, 비지니스 가치 창출을 가속화합니다.
- 직원들이 쓸데없는 일에 낭비하는 시간을 제거하고 비지니스 로직 작성에 집중함으로써 비용을 절감하고 효율성을 향상시킵니다.
- 제품 또는 사고 리더쉽을 통해 수익을 창출하거나 시장 점유율을 높힙니다.

전략을 마일스톤 단위로 세분화하십시오. 이를 통해 필요한 각 프로세스의 책임자를 정하여 더 신속하게 프로세스들을 구동할 수 있습니다. 전략 측면에서 다음 사항을 고려하십시오.

- 회사 전체에서 오픈소스를 언제 어떻게 사용할 것인지를 명확하게 하는 오픈소스 거버넌스 및 정책
- 개발자가 외부 오픈소스 프로젝트에 기여할 수 있는 방법, 즉 역할 및 소요시간을 명시한 정책
- 기술 리더쉽 및 엔터프라이즈 아키텍처 그룹들의 적용 가능한 소프트웨어 프로젝트에 처음부터 오픈 정책을 사용할 것을 독려하기
- 내부 개발 프로젝트에 대하여 회사 전체에 걸친 오픈소스 개발 방법론을 적용하는 내부협동개발(InnerSource) 모델을 시작

오픈소스 개발방식의 채택은 조직 사이의 경계를 넘게 만들고 새로운 조직 구조를 유도하기 때문에, 회사 내부에서 개발자들이 그런 구조를 넘나들며 협력하는 것을 허용하는 새로운 정책을 만들기 위한 노력이 필요할 수 있습니다.

당신의 주장을 옹호하면서 경계를 허무는 것을 도와줄 고위층 후원자를 찾는 것이 중요합니다. 이 부분은 우리가 이 절에서 언급한 것 가운데 가장 어려운 작업일 수 있지만, 매우 중요합니다. 당신이 준비하는 이 과정을 지지해줄 후원자가 필요합니다. 전략 보고서를 설득력 있게 작성하면, 그들이 당신의 고차원적이면도 장기적인 비전을 받아들일 것입니다. CTO 또는 CIO의 지원을 목표로 삼되, 그렇게 되기까지 최선을 다해 준비해야 합니다.

법무팀의 직원들은 이전에 다루었던 것들과는 근본적으로 다를 수 있는 라이선스를 이해할 수 있도록 교육을 받아야 합니다. 또한, 마케팅 및 홍보팀은 오픈소스가 개발 방법, 품질 그리고 고객 반응에 미치는 영향을 살펴봐야 합니다. 오픈소스 사례를 학습하고, 커뮤니티와의 소통에 협력하며, 이 새로운 개발 방법을 고객과의 상호 작용으로 전환하는 일을 해야합니다. 어쩌면 당신이 거기에서 고용된 경우도 있겠지만 당신이 참여하고 있는 오픈소스 커뮤니티의 구성원들이 그 커뮤니티에서 개최하는 이벤트에 참석하고 지원하는 것의 중요성을 마케팅 팀에 설명할 시간을 마련하는 것도 좋습니다. 또한, 영업팀은 고객에게 솔루션을 제시할 때, 코드의 사용 및 확장에 대한 질문에 잘 대답할 수 있도록 라이선스를 충분히 이해해야 합니다.

명확한 프로세스가 없으면, 누군가 올바른 방법을 따르지 않고 비공식적으로 오픈소스를 우리 코드에 포함시킬 위험이 있습니다. 이것은 어쩌면 그 오픈소스가 가진 라이선스를 위반할 수도 있을 뿐만 아니라, 오픈소스를 올바르게 활용하여 얻을 수 있는 이득을 놓치게 할 수도 있습니다. 예를 들어, 중요한 버그 수정은 오픈소스 프로젝트에서 주기적으로 릴리즈되는데, 그 수정을 적용하기 위해서는 해당 코드가 어디에 사용되는지 알아야 합니다. 또한, 명확한 프로세스가 수립되면 당신 조직의 직원들이 커뮤니티의 소중한 구성원이 되는 것이 허용되며, 거기에서 조직의 요구사항을 대변할 수 있습니다. 마지막으로, 명확하고 잘 소통되는 정책은 회사 전체에 걸쳐 당신의 오픈소스 계획에 대한 참여와 인식을 훨씬 더 높여줍니다.

OSPO를 통한 전략 수립

개발자들은 비공식적으로 오픈소스 커뮤니티의 일원으로 참여할 수 있지만, 회사가 그로부터 확실한 이득을 얻기를 원한다면 법률적 심사, 프로젝트 심사, 코드를 작성할 개발자를 채용하는 것, 프로젝트와 이벤트를 지원하는 것, 커뮤니테이션과 커뮤니티 릴레이션를 관리하는 것 등, 모든 오픈소스 지원을 위한 창구를 일원화해야 합니다. 따라서 오픈소스를 시작하는 대부분의 회사들은 프로젝트를 홍보하고 관련된 업무를 처리하기 위해서 OSPO(Open Source Program Office)를 만듭니다. 여러 OSPO의 책임자들과 리눅스 재단의 TODO 그룹이 협력해서 만든 오픈소스 템플릿과 정책 예제들이 이제 막 시작하는 회사들에게 유용하게 사용될 수 있을 것입니다. 당신의 OSPO는 다음 절들에서 서술되는 활동들을 지원할 수 있습니다.

회사 전체를 아우르기

정책을 수립한 뒤에는, 회사의 모든 개발자들이 이 정책을 숙지하도록 합니다. 여기에는 법무팀, 구매조달팀과 같은 회사의 여러 다른 영역들 또한 포함되어야 합니다.

먼저 회사 안에 오픈소스 실무자로 이루어진 지원 커뮤니티를 만드십시오. 경영진의 지원이나 가이드가 있든 없든, 개발자를 포함하여 오픈소스 커뮤니티에서 일해본 경험이 있는 사람들이 기본적으로는 지원 커뮤니티 활동을 할 수 있을 것입니다. 이들 지지자들은 정기적인 점심시간 세션, 웹 세미나, 그리고 팀 회의에서의 발표와 같은 활동들을 통해서 다른 직원들에게 오픈소스를 전파하고 교육할 수 있습니다.

회사에 있는 모든 개발자를 대상으로 진행하는 교육 과정은 오픈소스를 사용하고 기여하는데 있어서 모두가 같은 이해와 기대를 가지고 있도록 만들어 줍니다.

이와 같은 지원 활동은 오픈소스에 대한 이해를 돕고 직원들이 그 문화에 익숙해지도록 합니다. 더 중요한 것은, 이 활동이 잠재적인 역량을 가진 사람을 발굴하여 더 일찍 그들과 같이 일을 시작할 수 있게 해준다는 것입니다.

잠재력 있는 프로젝트의 평가

오픈소스를 찾을 수 있는 곳은 많이 있습니다. GitHub과 GitLab은 잘 알려진 오픈소스 호스팅 사이트입니다. (두 사이트 모두 Linus Torvalds가 개발하고 많이 사용하는 버전 관리 소프트웨어인 Git을 사용합니다). 필요한 키워드 (예를들어, "employee management")를 검색하면 보토 수 천개의 프로젝트가 검색됩니다. 따라서 관심이 가는 프로젝트가 우리 요구 사항에 잘 맞는지를 잘 살펴봐야합니다. 리눅스 재단의 오픈소스 활용 가이드는 프로세스에 중점을 두고 있으며 Dan Woods와 Gautam Guliani의 책 엔터프라이즈 오픈 소스(O'Reilly, 2015)는 프로젝트가 얼마나 쓸만한가를 판단하는 가이드 라인을 제시합니다. 다음은 평가시 확인할 사항에 대한 목록입니다.

코드 품질

코드를 직접 검사하거나, 프로젝트가 GitHub에 있다면 Stars 등급의 확인, 버그리포트의 수, 그 프로젝트에 대한 사람들의 평판을 온라인에서 확인함으로써 평가할 수 있습니다. 모든 코드에는 오류가 있기 마련이기 때문에, 사람들이 버그가 있다는 것을 보고한다는 사실은 그 프로젝트가 유망하다는 뜻입니다. 중요한 버그가 수정되고 있기만 한다면 말이죠.

개발 활성도

프로젝트가 최근에 새 릴리즈를 발표했거나 아니면 적어도 버그를 수정하고 있는가? 코드에 대한 관심을 나타내는 지표로서 풀-리퀘스트가 많이 있는가?

프로젝트 성숙도

프로젝트가 얼마나 오래된 것인가? 커뮤니티가 활성화되어 있는가? 코드를 관리하는 사람들이 많이 있는가? 프로젝트에 대한 펀딩이 이루어지고 있나? 책, 동영상이나 다른 교육 자료가 있는가?

지원 수준

소프트웨어의 설치, 유지 보수에 도움을 받을 수 있는가? 프로젝트에 관한 좋은 문서가 있는가?

커뮤니티 활성도

메일링 리스트가 활성화되어 있나? 버그리포트에 대하여 개발자들이 신속하고 긍정적으로 대응하는가? 제기된 이슈에 사람들이 예의바르고 생산적으로 대응하는가? 커뮤니티가 성장하는 좋은 징조이지만 반드시 그래야 할 필요는 없습니다. 작은 사용자 기반을 가진 프로젝트가 우리에게 딱 맞는 것일 수도 있기 때문입니다. 한편, 커뮤니티가 활성화되지 않았거나 줄어들고 있다면 그것은 일종의 경고 신

호입니다.

의사 결정의 개방성

프로젝트의 모든 선택 사항들이 공개되어 논의되고 있는가? 리더들이 개인적으로 중요한 결정을 내리고 메일링리스트나 코드 저장소에 결과를 보고한다는 느낌을 받는다면, 그것은 당신이 프로젝트의 방향에 대한 영향력을 발휘하기 어렵다는 것을 나타내는 경고 신호입니다. 처음에는 그 영향력이 중요하지 않다고 생각할 수 있지만, 소프트웨어에 대한 의존성이 커진 나중에 뭔가 말해야 한다면 문제가 될 수 있습니다.

거버넌스와 커밋 권한

새로운 기여자가 커밋 권한을 얻게 되는 과정이 문서화되어 있는가? 프로젝트에 당신의 시간과 전문성을 투자하고 있다면, 당신이 원할 때 그 기여에 상응하는 책임을 프로젝트에서 맡는 것이 맞을 수도 있습니다.

보안 문제 보고 체계

프로페셔널한 프로젝트라면 보안상의 허점을 발견한 사람이 프로젝트 리더에게 개인적으로 연락하도록 하여, 보안 문제가 있다는 사실이 외부에 알려지기 전에 수정될 수 있어야 합니다.

라이선스의 준수

OSPO 또는 조직 내에서 오픈소스 사용을 추적하는 팀은 라이선스가 준수되고 있는지 확인해야합니다. 라이선스의 기술적 의미를 이해하는 개발자들과 법무팀 모두 주의가 필요합니다. 오픈소스 커뮤니티 참여자들은 오픈소스 라이선스 정합성(compliance)을 보다 간단하고 일관성있게 만들고 더 많은 기업들이 오픈소스 소프트웨어를 사용하도록 권장하기 위해 OpenChain Project를 구성했습니다. 라이선스 또는 코드의 원저자 찾기 자체도 어려울 수 있기 때문에, ClearlyDefined라는 또 다른 오픈소스 프로젝트로 사용자들이 라이선스 정합성을 위한 정보를 찾고 유지 관리할 수 있도록 도움을 주고 있습니다. Ibrahim Haddad의 책 기업에서의 오픈 소스 준수(리눅스 재단, 2016)는 라이선스 정합성 문제를 깊이있게 다루고 있습니다.

법무팀과 관계를 잘 설정하고 협력하십시오. 그들은 오픈소스와 관련하여 발행하는 일들을 거의 확실하게 처리해 줄 것입니다. 예를 들어, 오픈소스 라이선스의 적용을 받는 작업을 승인해주며 회사의 개발자들이 코드를 외부 프로젝트에 제출할 때 필요한 기여자 동의서를 검토해줍니다. 이 과정은 오픈소스의 친숙하지 않은 측면에 대해 협력하고, 끈끈한 관계를 구축하고, 신뢰를 얻을 수 있는 기회입니다. 이 관계를 이용하여 개발 자체와 개발자들에게 보다 유리한 오픈소스 정책을 자리잡게 할 수 있습니다.

또한 구매조달팀과의 관계도 필요합니다. 오픈소스와 관련된 구매 계약에 필요한 추가적인 세부 항목들을 추천하거나 그 팀이 써드파티와의 계약을 검토하는 과정에서 도움을 줄 수 있습니다.

커뮤니티의 코드를 직접 만든 코드만큼 중요하게 관리하기

외부 조직에서 오픈소스 코드를 개발했다 하더라도, 우리 조직 내에서 사용하려면 그 코드를 관리해야 합니다. 외부 코드도 테스트되어야 하고, 백도어가 있는지 확인해야하고, 기타 결함에 대한 테스트를 해야합니다. 외부의 코드를 우리 코드와 통합하는 작업도 내부에서 회사 코드를 작성하는 것과 마찬가지로 목표와 기한을 설정하고 모든 것이 계획대로 진행될 수 있도록 리소스를 투입해야 합니다.

물론 외부에서 코드를 가져오는 경우 일부 작업은 다르게 진행됩니다. 우리 개발자들이 코드를 관리하는 커뮤니티의 일원이 되어야 합니다. 예를 들어, 우리 개발자들이 커뮤니티의 버전 관리 시스템 안에서 일을 해야 합니다. 커뮤니티에서 코드를 호스팅하는 GitHub 또는 GitLab의 계정이 필요합니다. 코드를 수정하려면 (뒤쪽의 "오픈 소스 프로젝트에 기여하기" 장에서 자세히 설명 됨), 버전 관리 시스템에서 수정을 위한 브랜치 만들거나, 작업을 위한 내부 버전 관리 시스템을 사용해야 합니다.

보상과 관리 체계의 변경

오픈소스 커뮤니티와 함께 일하는 개발자는 비공개 프로젝트에서와는 다른 방식으로 일하며 의사소통과 협의를 하는 과정에서 추가적인 역량이 필요합니다. 개발자에게 필요한 추가 작업이 반영된 보상 체계를 만들어야 합니다. 프로젝트에서 사용하는 도구를 배우고, 채팅 및 메일링 리스트에 참여하고, 경험이 부족한 동료를 멘토링하고, 다른 사람이 기여한 코드를 확인 및 수정하고, 오픈소스 프로젝트에 대한 컨퍼런스나 리더 미팅에 참석하기 위한 시간도 할당해 주어야 합니다.

특히 개발자들과 다른 직원들이 특정 팀의 직접적인 성과물과 관련이 적은 일에 시간을 투자해야 한다면, 해당 직원과 관리자에 대한 성과 평가 기준도 조정해야 합니다. 이는 성과보너스를 지급하는 회사에 특히 해당됩니다. 오픈소스 프로젝트의 성공 요인과 회사 내부의 목표와 인센티브의 결을 맞추는 것이 중요합니다. 직원들이 컨퍼런스에서 발표하고 프로젝트 작업에 대한 블로그를 작성하고, 프로젝트의 거버넌스에 참여한다면, 그들의 기여가 회사의 명성에도 좋게 작용하며, 향후 개발자 채용과 같은 면에서 이득이 됩니다.

많은 직원들에 대한 광범위할 수도 있는 보상 계획을 수용하는 것과는 별개로, 관리자들은 소프트웨어 데드라인이 내부 직원뿐만 아니라, 커뮤니티의 활동에 의해서도 좌우될 수 있음을 알아야 합니다. 소프트웨어 프로젝트의 데드라인은 온전히 내부 팀에 의해 실행되는 경우에도 정확하게 설정하는 것이 거의 불가능에 가깝기 때문에 외부인에 의한 이 새로운 의존성은 관리자들이 현실을 인식하는데 도움이 될 수 있습니다. 커뮤니티가 우리가 필요로 하는 기능에 대한 우선순위를 낮게 설정했다면 자체 리소스를 투입하여 적시에 완료하고 개선 내용을 커뮤니티에 기여하십시오. 다른 회사도 비슷한 방식으로 일을 할 것이며 모두에게 이득이 됩니다.

오픈소스와 자부심

창의적인 결과물에 자부심을 가지는 것은 자연스러운 일입니다. 소프트웨어는 더 이상 도널드 커누스(Donald Knuth)가 혼자서 독창적인 영감으로 TEX와 Metafont를 개발했던 것과 같은 방법으로는 개발되지 않습니다. 그러나 팀 단위로 개발을 하는 경우에도 긴밀한 협업을 통하여 개발이 진행된다면, 그 팀 고유의 정체성이 생깁니다. 나아가 팀으로 하는 작업을 통하여 개발자 개인의 요구 사항을 담아낼 수도 있습니다. 서로 긴밀하게 작업을 진행하는 팀에서는 서로의 역량을 쉽게 파악할 수 있습니다. 관리자는 누가 어떻게 기여를 했는지를 정확히 파악할 수 있기 때문에, 그 기여에 따라 연봉 인상 및 승진을 결정할 수 있습니다.

이러한 정신적, 실질적 보상 구조는 오픈소스로 인하여 많이 바뀌고 있습니다. 오픈소스 관점에서, 개발자 스스로 기여에 대한 개인적인 성취와 보상을 어떤 식으로 원하는지를 고려해야 합니다.

오픈소스 프로젝트를 시작하거나 참여할 때, 프로젝트를 인위적으로 관리하려 해서는 안되며, 서로 다른 목적을 가지고 참여했다 사라지는 불특정의 사람들과 함께 일하는 법을 배워야 합니다. 궁극적으로는, 기여한 내용을 세상 누구나 확인할 수 있기 때문에, 오픈소스 프로젝트에서는 개개인이 훨씬 더 빛이 날 기회를 가지게 됩니다. 상상하지도 못했던 엉뚱한 프로젝트에서 그런 일이 발생하기도 합니다. 또한 다른 여러 사람이 채택하는 확실한 기여를 하면, 취업의 기회가 놀랄만큼 확대됩니다. 코딩뿐만 아니라, 소통, 프로젝트와 인력의 관리, 디자인, 웹 개발, 문서화, 번역 등 많은 분야에서 오픈소스에 기여할 수 있습니다. 모든 기여가 프로젝트에는 중요합니다.

모든 것이 잘 노출되는 오픈소스 프로젝트의 특징 때문에, 관리자는 프로젝트의 최고 기여자를 높은 연봉이나 더 좋은 자리를 제안하는 경쟁사에게 빼앗기지 않을까하는 우려를 할 수 있습니다. 이 문제를 해결하는 방법은 바로, 그 경쟁자가 되는 것입니다. 직원들이 지금 사용하고 있는 오픈소스 프로젝트에 개발자로 또 리더로 활동할 수 있는 기회를 주십시오. 그들에게 좋은 아이디어를 실현할 자유를 주고, 개발자나 개발팀에 유용한 도구를 만들어 낼 수 있는 자원을 제공하십시오. 만일 당신이 유명한 개발자들을 데리고 있다면, 그를 프로젝트 이사회에 참여시키고, 컨퍼런스를 통해 발표하도록 하십시오. 그들이 다른 최고의 개발자들을 자석처럼 당신 회사에 끌어올 수 있습니다.

프로젝트 커뮤니티에의 참여

이제 회사의 성공 여부는 많든 적든 오픈소스 프로젝트의 성공 여부에 달려있다고 볼 수 있습니다. 업무 시간에 커뮤니티의 구성원으로써 활동해야 하는 것 뿐만 아니라 (이 주제는 앞장의 “보상과 관리 체계의 변경” 절에서 더 자세히 다뤄지고 있습니다), 회사 자체가 프로젝트에는 소중한 자원이 될 수 있습니다. 풀-리퀘스트나 버그 수정, 수정 요구 등 해당 소프트웨어와 관련된 당신의 경험은 커뮤니티가 모두의 이익을 위해서 코드를 개선시키는데 도움이 될 수 있습니다.

여기서 참여라는 것은, 다른 기여자들이 하는 것을 똑같이 하는 것을 말합니다. 코드를 개발하는 개발자들은 아래와 같은 것들을 할 수 있습니다.

- 프로젝트에 대한 질문글 작성하기, 새로운 아이디어 제시, 버그리포트 하기
- 버그 수정이나 신규 기능을 개발하여 기여하고, 기여한 코드에 대한 권한을 프로젝트에 넘기는 기여자 계약에 서명하기
- 코더톤이나 컨퍼런스에 참가하고, 누구를 위해서 일하는지를 투명하게 밝히기
- 커미터(숙련된 개발자로 저장소의 핵심 코드를 수정할 수 있는 수준이라고 신뢰받는 개발자)가 되어 커뮤니티와 리더십 역할을 늘려가며 프로젝트 관리에 참여하기

이러한 참여 활동들은, 개발자들의 작업 결과물과 회사가 지원한 내용을 커뮤니티의 사용자들과 다른 개발자들로부터 인정을 받을 수 있도록 도와줍니다.

많은 성공적인 자유 오픈소스 소프트웨어 대부분은 흔히 Apache Way라고 불리는 원칙을 따르고 있습니다. 비록 소프트웨어 프로젝트들의 구성원들이 아파치 재단의 리더들이 정의한 것과 완벽하게 동일한 단어를 사용하지 않을 수는 있지만, 프로젝트의 일상적인 진행에서 이 원칙들이 사용된다는 것을 볼 수 있습니다.

개발자들은 이러한 가치들을 가지고 당신의 회사에 오거나 오픈소스 프로젝트를 하면서 그러한 원칙들을 배울 것입니다. 회사에게 정말로 어려운 점은, 모든 조직에서 경험하듯이 특정 분야에서의 기밀유지 필요성에 의해 완화된 유사한 가치들이 개발자로부터 전체 조직을 통해 퍼져나간 다는 점입니다. 오픈소스 사업을 옹호하는 사람들은 경영진과의 논의을 계획하고 오픈소스로부터 이익을 얻기 위하여 필요한 문화적 요소들을 도입할 필요가 있습니다. 투명성과 커뮤니티 참여의 가치를 관리자들이 회사의 모든 수준에서 받아들여야 하며, 개발자들에게 그들의 노력을 어디에 투자해야 하는지를 결정할 수 있는 자유를 주어야 합니다.

아래의 목록은 개발자가 오픈소스 커뮤니티에 참여하기 위해서 해야하는 활동들입니다.

- 활동하며 반드시 준수해야 할 행동 강령과 기여자 계약 확인하기
- 프로젝트에서 사용되는 도구들에 대해서 익숙해지기
- README 파일을 시작으로 프로젝트의 (프로젝트에 따라 완성도와 범위가 다양한) 문서들을 검토하기
- 프로젝트에서 정한 기여 절차 숙지하기
- 질문을 하고, 메일링 리스트나 그룹을 통해서 다른 기여자들과 친해지기
- 프로젝트의 할 일 목록에서 일거리를 찾아 가져오기
- 버그 수정 제출하기

회사 내의 개발자들이 오픈소스 프로젝트의 코드를 이해하고 나면, 당신의 조직에서는 그 코드를 직접 수정해야할 필요성을 느끼게 될 가능성이 높습니다. 대개, 프로젝트에는 내가 원하는 바로 그 기능이 없을 것이기 때문입니다. 또 이전과는 다른 성능 트레이드오프를 만들어야 할 수도 있습니다. 오픈소스 프로젝트들은 사용자들이 직접 수정한 내용을 프로젝트의 리파지터리, 즉 업스트림(Upstream)에 반영할 것을 권장하고 있습니다. 프로젝트의 라이선스와 코드를 어떻게 사용하는지에 따라 수정한 내용을 업스트림에 반영하는 것이 강제되기도 합니다. 설계가 잘 되어있는 프로젝트들은 이렇게 추가된 기능을 필요한 사람들은 채택하고, 다른 사람은 무시할 수 있도록 모듈화가 되어있습니다.

개발자들이 수정한 내용이 메인 프로젝트나 코어 모듈에 반영될 때까지는 꽤 긴 시간(때에 따라서는 수년)이 걸릴 수도 있습니다. 프로젝트의 리더들은 수정사항이 다른 사용자들에게 필요가 없다고 판단이 되거나, 버그에 대한 우려가 있거나, 혹은 성능에 나쁜 영향 가져오지 않을까에 대한 걱정 등으로 그 수정사항을 보류할 수 있습니다. 또 문화적인 견해 차이에 부딪힐 수도 있습니다. 당신 회사의 개발자들이 아직은 커뮤니티의 관심과 신뢰를 얻는데 충분할 정도로 커뮤니티를 잘 이해하지 못했을 수도 있습니다. 하지만 이러한 경우에도 프로젝트 리더들이 가지고 있는 우려를 불식할 가치가 있으며, 혼자가 아니라 커뮤니티의 개발자들과 일을 하는 노력을 계속해야 합니다.

당신의 개발 조직을 위해서 브랜치를 별도로 만들거나 원래의 코드를 복제하여 내부적으로 새로운 리파지토리를 만들어야 할 수도 있습니다. 이러한 과정을 포크(fork)라고 합니다. 포크를 한 브랜치를 추후 메인 브랜치에 병합하는 것이 가능하기는 하지만, 프로젝트의 주요 커미터들의 동의를 얻어야만 합니다.

코드를 기여하거나 버그를 수정한 경우, 코드 전체에 대하여 전혀 타협이 없는 코드 리뷰를 통해 큰 이득을 얻습니다. 최종적으로 프로젝트에 포함되는 코드는 최초에 당신이 제출했던 코드보다 더 적은 버그, 더 나은 성능, 그리고 미래에 개선될 것을 고려한 더 일반화된 소프트웨어 구조의 코드가 됩니다.

추가적으로, 만약 당신의 코드가 프로젝트의 핵심적인 부분이 된다면, 그 프로젝트에 발생하는 업데이트를 지속적으로 설치하고 그 업데이트에 당신의 코드가 있다는 것을 확인하면서 같이 일을 하게 됩니다. 만약 수정내용을 기여를 하지 않고 코드를 포크하여 따로 사용하게 된다면, 프로젝트의 새로운 버전들을 포기하여 원래의 코드가 가지고 있던 많은 버그들과 보안 취약점으로부터 고통받게 되거나 혹은 매번 버전이 바뀔 때마다 직접 개발했던 수정 사항을 다시 적용하고 시험하는 작업을 지루하게 해야하고 또 직접 개발했던 기능이 새 버전에 포함되어 내가 수정한 내용이 의미가 없어졌는지도 매번 검사해야합니다.

기여없이 코드를 포크하여 유지하는 경우, 시간이 지날수록 유지 보수 비용이 커집니다. 원 프로젝트에 기여하는 훈련이 부족한 회사들의 대부분은 원래 프로젝트의 지속적인 개발 내용을 가져와 적용하는 내부 절차도 미흡하게 되어있습니다. 원 프로젝트의 업스트림(upstream)에서 수정된 버그와 보안 패치, 그리고 기능 개선을 다운받고 적용하지 않는데, 그 이유는 포크를 한 시점으로부터 시간이 지날수록 업스트림과의 버전 차이가 커지고, 그에 따라서 업스트림에서 수정된 내용들을 포크된 코드에 적용하는 작업 비용이 증가하기 때문입니다.

앞의 이야기에도 불구하고, 우리가 수정하고 기여한 내용이 프로젝트에서 받아들여지지 않을 가능성도 있습니다. 또는 수정 사항을 기여하지 않는 이유가 있을 수도 있습니다. 예를 들면, 규정이나 경쟁 구도 때문에 코드에 포함된 비밀들을 공개하지 말아야할 할 수도 있습니다. 몇몇 회사들은 별도의 브랜치를 공개적으로 혹은 내부적으로 유지하기로 결정하고 업스트림의 개선 사항들을 포크된 브랜치로 가져오는 작업에 비용을 들이기도 합니다. 또 가끔은 회사에서 직접 개발한 수정사항들 가운데 일부분만을 원 프로젝트에 기여를 하기도 합니다. 하지만 수정 코드를 기여하지 않는 것은 프로젝트의 존재 가치를 저하시킬 수 있습니다.

품질과 보안 : 오픈소스와 오픈소스가 아닌 소스의 비교

우리가 어떤 벤더로부터 상품을 사기 전에 품질을 평가하고, 좋은 사업 관행을 가지고 있는지, 고객 지원은 잘 하는지, 그리고 일반적인 평판도 확인하는 것처럼 먼저 오픈소스를 평가할 필요가 있습니다. 이 책의 "잠재력 있는 프로젝트의 평가" 절이 처음 확인할 몇가지 포인트들을 제공합니다. 잘못된 오픈소스 소프트웨어를 선택을 하는 경우, 예상했던 품질을 얻지 못하거나 프로젝트에 대한 관심이 줄어들어 기여자들이 다른 곳으로 옮겨갈 수 있다는 위험 요소가 있습니다. 만약에 후자의 문제가 발생한다면, 비록 프로젝트를 유지하기 위해 코드에 관한 책임을 지고 사람들을 고용하는 선택을 할 수도 있지만, 아마도 당신도 같이 다른 곳으로 옮겨가는 것이 맞을 수도 있습니다.

대신 오픈소스는 그렇지 않은 소프트웨어를 제공하는 벤더에 비해 몇가지 이점을 가지고 있습니다. 사업을 접거나, 당신이 생각하는 제품의 방향이 아닌 다른 쪽으로 제품을 바꾸거나, 제품의 비용을 급격히 상승시키거나, 버그 수정이나 당신에게는 중요한 기능의 추가를 거부하거나, 심지어 당신의 활동을 추적하는 악성 소프트웨어를 삽입하는 등, 벤더가 더 이상 당신의 요구에 맞추지 못할 경우 오픈소스는 다른 옵션을 제공합니다. 이 모든 일들은 독점적인 소프트웨어에서 발생해 왔습니다. 오픈소스 소프트웨어는 당신이 나아가야 할 방향을 스스로 정할 수 있도록 합니다. 당신은 벤더를 바꾸고 당신의 필요에 가장 적합한 방법으로 문제를 해결하는 방법을 결정할 수 있습니다.

오픈소스가 아닌 소프트웨어가 오픈소스에 비해 더 품질이 높거나 낮은지, 더 안전한지 덜 안전한지에 관한 많은 주장이 있었습니다. 그 질문에 대한 해답은 어느 쪽인지 결론이 나지 않았습니다. 탄탄한 개발 관행을 확산시키기 위해 리눅스 재단의 Core Infrastructure Initiative는 모범 사례 가이드를 작성했습니다. 이 가이드에는 소스코드 관리에서부터 테스트, 코드 분석까지 공통적으로 발생하는 광범위한 이슈들에 관한 내용이 적혀있습니다.

세상 모든 전문가들이 소스를 검증할 수 있다는 점 때문에 오픈소스 프로세스가 더 안전하고 표준적인 코드를 만들 수 있다는 점에 보안 전문가들은 동의하고 있습니다. 그와 반대로, 널리 배포된 오픈소스 보안 소프트웨어인 OpenSSL에서 수 년 간 발견되지 않고 있다가 발견된 악명 높은 하트블리드 결함은 오픈소스가 만병통치약도 아님을 보여줍니다. 다행히도, 주요 오픈소스 프로젝트들은 하트블리드 결함과 같은 일이 다시 발생하는 걸 막기 위해 외부에서 검증가능한 프로세스 변화를 의미있게 만들었습니다. 예를 들어, 리눅스 재단과 함께 일하는 회사들이 협력하여 OpenSSL에 기여하는 개발자들이 전업으로 그 일을 할 수 있도록 지원합니다.

어찌되었든, 이러한 고도로 네트워크화된 시대의 보안은 시스템을 강화하는 것보다 훨씬 더 복잡해졌습니다. 당신이 모든 패치를 부지런하게 설치하고 의심스러운 첨부 파일을 열지 않더라도, 누군가 조작을 한 웹 사이트를 방문하여, 멀웨어를 다운로드 하거나 피싱의 희생자가 될 수 있습니다. 보안은 항상 전체적인 관점에서 보아야 합니다. 그것은 기술적이기도 하고 그 만큼이나 문화적이고 정책 중심적이기도 합니다.

오픈소스 프로젝트에 기여하기

오픈소스를 사용하고 있다면, 코드 기여를 통해서 또는 스폰서나 현금 기부 등의 방법으로 그 오픈소스 프로젝트에 기여함으로써 더 많은 것을 얻을 수 있습니다. 여러분이 지원하는 소프트웨어에서 활용할 수 있는 도구나 플랫폼의 추가 기능, 플러그인과 같이, 핵심적이지는 않은 오픈소스 프로젝트를 만드는 것부터 시작할 수 있습니다.

이 절은 이미 진행되고 있는 프로젝트에 기여하는 방법을 설명하고, 이어서 다음 장에서는 직접 오픈소스 프로젝트를 시작하기 위해 밟아야하는 단계들도 살펴보겠습니다.

회사 전반에 기여해야하는 "이유" 확립하기

이제 오픈소스를 적용하는 것은 소스를 그저 복사해 붙여넣기 하는 것이 아니라, 당신의 조직이 오픈소스를 진지하게 생각하고 있다는 것을 의미한다는 것을 이해했을 것입니다. 관리자들은 앞서 이야기했던 커뮤니티에 참여 활동을 포함하여 오픈소스에 투입하는 리소스를 승인해야 합니다. 그를위해 코드의 검증과 커뮤니티 자체, 그리고 커뮤니티 구성원으로서의 활동에 자원을 투자하는 것을 정당화하기 위한 근거를 만들어야합니다. 다시 말하면, 오픈소스를 이용하는 것이 조직의 사업 목표들에 어떻게 도움이 되는지 명확하게 설명할 수 있어야 합니다. 또한, 개발자들의 개별적인 목표가 조직의 상위 목표들에 부합한다는 점을 확신할 수 있도록 정기적으로 확인해야 합니다. Mozilla and Open Tech Strategies에서 발간한 백서에서는 오픈소스 프로젝트를 만들고 운영하는 전반에 대한 열 가지 전략이 적혀 있습니다. 백서에서는 다양한 거버넌스 구조와 동기 부여를 위한 프레임워크들이 설명되어 있으며, 그것들이 성과에 미친 영향을 볼 때 오픈소스를 선택하는 것이 얼마나 중요한지 보여줍니다.

커뮤니티로부터 고용하기

기존의 오픈소스 프로젝트를 단번에 활용하는 가장 좋은 방법은 해당 프로젝트에 활발하게 기여하고 있는 역량 높은 컨트리뷰터를 고용하는 것입니다. 몇몇 기업들은 프로젝트 리더나 주요 커미터들을 해당 프로젝트를 위해 풀타임으로 일하는 조건으로 고용하기도 합니다. 또 어떤 기업들은 직원들에게 일정 시간을 오픈소스에 기여하고 그 외의 시간은 해당 오픈소스를 기업에 적용하는 데에 쓰거나 다른 내부 프로젝트에 쓰도록 하는 식으로 균형을 맞추기도 합니다.

사실 내부 프로젝트와 오픈소스 프로젝트 참여 사이에서 균형을 유지하는 것은 쉽지 않습니다. 관리자들은 언제나 내부 프로젝트에 더 힘을 쏟기를 바라고 있고, 개발자의 일정에 더 많은 내부 프로젝트 작업을 끼워넣기를 원하기 때문입니다. 회사가 커뮤니티에서 개발자를 고용하기 위해서는, 비즈니스 계획과 프로젝트 실제 참여를 통해 해당 프로젝트에 대한 진지한 의지가 있다는 것을 보여주어야 합니다. 개발자들이 자기 시간을 오픈소스 프로젝트에 투자할 수 있도록 고용 계약을 다시 확인하고 수정해야 할 수도 있습니다. 개발자가 고용이 되면, 관리자와 개발자 모두 시간을 어떻게 분배하기로 했는지에 대한 약속에 꾸준히 신경써야 합니다.

여러분이 오픈소스 프로젝트에 얼만큼의 시간을 들였던지 적재적소에 활용한 건강한 오픈소스는 "왜 기업과 정부는 오픈소스로 전환하는가?" 장에서 언급했던 대로 여러분에게 반드시 득이 될 것입니다.

개발자 멘토링과 지원

회사의 직원이 회사의 새로운 코드를 가지고 작업하거나 오픈소스 프로젝트에 참여하려면 어느 정도 훈련이 필요합니다. 만약 오픈소스 환경에서 일하는 것에 익숙한 직원이 없다면 - 한번 조사해보면 이미 오픈소스 경험이 있는 직원들이 이미 꽤 있다는 점에 놀랄지도 모릅니다 - 앞 절에서 이야기한 것과 같이 오픈소스 경험이 있는 개발자를 고용하는 것도 좋습니다. 이 개발자들이 다른 사람들을 멘토링하도록 하면 됩니다. 회사 내부이든 외부의 더 큰 커뮤니티이든 개발자들을 오픈소스 프로젝트에 참여시키기 위해서는 이 부분이 중요합니다. 앞서 언급했듯, 오픈소스 프로젝트에 참여하는 것은 어떤 회사의 개발 역량을 향상시키는 최고의 방법입니다.

참여를 위한 규칙 설정하기

외부 프로젝트 하나에 참여하는 것만으로도 오픈소스는 그것을 채택하는 모든 조직을 변화시킵니다. 조직의 다양한 직급의 개발자들과 다른 직원들의 대화나 결정 사항들이 모두 공개되어 외부 사람들이 볼 수 있는 상태가 됩니다. 오픈소스 프로젝트 온라인 포럼에서 개발자가 했던 말이 전세계에 공개되고, 온라인 상에 영원히 남겨진다고 가정해 봅시다. 이 상황은 오픈소스 프로젝트에 참여할 의지를 떨어뜨릴 이유처럼 들릴 수 있지만, 실제로는 그렇지 않습니다. 이건 단지 정직하고 존중이 담긴 대화를 하도록 하고, 차별과 학대 금지하는

인권 존중 규정이 담긴 행동 강령을 만들어 지키도록 할 필요가 있다는 것을 의미할 뿐 입니다. 많은 오픈소스 프로젝트가 가져다 쓸만한 좋은 행동 강령을 제공하고 있습니다. 주의 깊게 시험하고 검증한 여러 강령을 TODO Group이 이미 제공하고 있습니다. 이 행동 강령의 개정본이 아마존웹서비스와 같은 회사의 오픈소스 프로젝트들에도 적용되고 있습니다.

최근 컴퓨터 산업(과 사회의 다른 부분에서) 여성과 소수자들의 기여를 좌절시키거나 비협조적이고 고통을 주는 환경을 조성하는 관행에 대한 주의가 높아지고 있습니다. 여러분들도 공개 조사로 들어났는지의 여부를 떠나서 이러한 악습들이 조직에서 근절되길 원할 것입니다. 기업과 커뮤니티는 다양성과 포용성을 위한 긍정적이면서도 차별을 철폐하는 조치를 취해야만 합니다.

열린 소통문화의 장려

존중을 넘어, 여러분의 개발자들이 오픈소스 커뮤니티에 생산적으로 참여할 필요가 있으며, 그를 위해서는 문화적인 변화도 요구됩니다. 개발자들은 두 명에서 네 명 정도의 비공식 미팅을 통해서 설계와 관련된 결정을 내리는 데에 익숙했을 것입니다. 일부 애자일과 스크럼 방법론은 이런 활동을 장려하기도 하고, 조직 내에서 폭 넓은 의견을 수렴하는 기술도 제공합니다. 따라서, 애자일 스크럼 방법론은 오픈소스에 적용할 수 있습니다. 실제로 오픈소스는 다양한 종류의 개발 방법론에 열려 있습니다.

개발자들은 이제 다른 사람들도 볼 수 있고, 모든 기록이 지워지지 않고 보존되는 포럼에서 중요한 설계 결정에 대한 논의를 수행하는 법을 배워야 합니다. 커뮤니티 코드에 기여하고자 하는 변경 사항에 대해서는 커뮤니티 메일링 리스트 혹은 모든 사람이 참여할 수 있는 슬랙, 지터, 스택 오버플로우, 구글 그룹스와 같은 커뮤니케이션 채널을 통해 토론해야 합니다. 그렇게 하지 않고 변경 사항을 제출하면, 커뮤니티는 화들짝 놀라게 되고 아마도 변경 사항은 받아들여지지 않을 것입니다.

시의적절하게 정보를 제공하는 것 뿐만 아니라, 개발자들은 소속 팀 내의 좁은 시야를 넘어, 사내 다른 부서나 심지어는 지구 반대편에 있는 외부인에 이르기까지 다양한 범위의 사람들로부터 의견을 얻는 것의 가치를 인식해야만 합니다. 존중이 담긴 대화의 기술이 여러분의 기업 문화에 꼭 도입되어야 하는 것은 이 때문입니다. 이제 개발자들에게 다음과 같은 다른 새로운 기술도 요구됩니다: 타인의 말을 잘 들어주어야 하고, 다양한 개발자들로부터 피드백을 받아야 하며, 부적절한 기여를 거절하는 한편으로는 기여자가 학습을 한 뒤 다시 시도하도록 격려하고, 다양한 언어 능력을 가진 서로 다른 배경의 사람들을 다루어야 합니다.

개발자들이 복도에서의 대화나 전화 통화를 통해 설계 결정을 내리는 데에 익숙해져 있다면, 좀 더 공개된 방식으로 결정을 내리고 메일링 리스트나 버그 트래킹 시스템이 제공하는 덜 직관적인 소통 방식을 배우는 데에 다소 시간이 필요할 수도 있습니다. 오픈소스 프로젝트에 참여하는 것은 여러분의 조직 내에서 훌륭한 커뮤니케이션과 솔직한 반대, 그리고 존중이 담긴 열린 대화의 우선 순위를 높이는 훌륭한 방법입니다.

더 많은 사람이 참여하게 되면 결정에 더 오랜 시간이 걸릴 수 있습니다. 제품을 빨리 내놓기 위해 오픈소스 토론 절차를 생략하고 빠른 결정을 내리게 되면, 수행한 작업의 일부를 롤백하여 되돌려 놓아야 할지도 모릅니다. 그렇게해야 코드는 유지 보수가 될 수 있고, 미래의 요구 사항들을 반영할 수 있게 됩니다.

성실하게 결정 사항을 기록할 때에 얻을 수 있는 중요한 부수적인 이점은 그 프로젝트가 특정 개인에게 덜 의존하게 되고, 프로젝트의 핵심 인물들이 떠나더라도 무리없이 재구성될 수 있게 된다는 것입니다. 어느 한 사람이 유일하거나 대체불가능한 정보를 가지고 있으면 안됩니다.

오픈소스 프로젝트의 시작

오픈소스 개발 모델의 장점을 여러분이 속한 조직이 알게 되면서, 직접 오픈소스 프로젝트를 시작하기로 결정했을지도 모릅니다. 사려 깊은 계획이, 빨리 잊혀지는 실패와 번창하면서 지속되는 코드베이스의 차이를 만듭니다. Linux Founation 가이드에서는 그 과정에 대해서 자세히 설명하며, 포괄적인 오픈소스 프로젝트 시작에 관한 체크리스트를 제공합니다. 앞 절들에서 설명한 모든 원칙들이 적용되며, 좀 더 고려할 만한 사항들이 이 절에서 설명될 것입니다.

라이선스의 선택

법무팀에서는 앞으로 회사 내부에서만 사용하기 위하여 비공개로 개선할 것인지를 포함하여, 코드를 어떻게 활용할지를 파악하고 있어야 합니다. 라이선스의 선택은 다른 회사나 개인들이 그 소프트웨어의 채택 여부에 대한 결정뿐만 아니라 회사 내에서의 사용에도 영향을 줍니다. 그렇기에, 라이선스는 회사의 비지니스 전략을 고려해서 결정해야 합니다.

그럼에도 불구하고, 팬시한 라이선스를 채택할 필요는 없습니다. 이 책의 앞 부분에서 몇 가지의 유명한 라이선스를 소개하였습니다. OSI(Open Source Initiative)에 보면 많은 오픈소스 라이선스가 나열되어 있기는 하지만, 그 중에서 잘 알려져있고 최신에 만들어진 라이선스를 선택할 것을 강력하게 권합니다. GitHub은 잘 만들어진 라이선스 선별 가이드를 제공하고 있습니다. 마음에 맞는 라이선스를 찾을 수 없다면, 오픈소스로 프로젝트를 진행하는 동기를 다시 한번 진지하게 생각해 보십시오. 만일 잘 알려진 라이선스를 사용할 수 없다면, 오픈소스의 원칙과는 다른 방향으로 프로젝트를 진행하려고 하기 때문일 가능성이 있으며, 그로 인하여 우리가 프로젝트에 끌어들이고 싶었던 사람들을 떨어져나가게 할 수 도 있습니다.

코드를 문 밖에 내놓기

코드의 첫번째 라인을 작성하기 전부터 공개된 코드 저장소를 만들고 회사 외부에 참여를 요청하는 것이 가장 좋습니다. 그렇지 않으면 공개되서는 안되는 내용들과 낯 뜨거운 코멘트들을 제거하기 위한 철저한 코드 리뷰 전까지는 코드를 공개할 수 없을 것입니다. "오픈소스 퍼스트" 방식으로 처음부터 코드를 공개하면 개발자들이 최상의 코딩 방법을 따르게 될 것입니다.

프로젝트 이름은 커뮤니티와 공감할 수 있도록 선정하십시오. 이름은 아주 개인적일 수도 있고(예 : Linux는 운영체제 작성자의 이름을 따서 명명됨), 오피스 소프트웨어인 [LibreOffice] (https://www.libreoffice.org/)처럼 소프트웨어를 설명할 수 있는 것도 좋습니다. 마스코트의 이름을 따서 짓기도 하고 기억하기 쉽도록 의미없는 단어를 쓰기도 (예 : Hadoop) 합니다.
법무팀이 상표에 대해 수행하는 것과 동일한 수준의 조사와 확인을 거쳐, 같은 이름이 존재하지 않으며, 어떤 언어에서는 나쁜 의미를 가지지 않는 이름을 선택합니다. 프로젝트 및 브랜드 홍보에 대한 정보는 아래 '프로젝트 소통' 절을 참조하십시오.

오픈소스 개발 방식을 따르지 않고 내부 팀 작업에 익숙한 사람들이 만든 첫 번째 공개 코드는 지저분할 수 있습니다. 완성도가 낮은 코드를 공개하는 것은 개발자와 관리자 모두 수용하기 어려울 수 있습니다. 그러나 이미 진행된 내용과 커뮤니티로부터의 도움이 필요한 사항을 문서화하고, 프로젝트를 완성하겠다는 점과 새로운 사용자에 대한 멘토링을 약속하면 커뮤니티로부터 보상을 받게 됩니다. 당신이 좋은 아이디어를 가지고 있다면 많은 사람들이 손을 내밀어 같이 문제를 해결해 줄 것이고, 아이디어가 좋지 않다면, 외부로부터 좋지 않은 반응들이 들릴 것이며, 더 이상 개발자 시간을 낭비하지 않고 프로젝트를 포기할 수도 있습니다.

안정된 코드를 만들기 위한 모범적인 개발 방법의 사용

공개 리파지토리에 코드와 문서를 저장하거나, 당신이 사용하는 버전 관리 시스템을 누구나 접근하게 설정해도 됩니다. 개발자가 매일 또는 필요한 만큼 자주 변경 사항을 적용할 수 있게 하십시오. 개발자들은 이 방법을 "release early, release often"이라고 부릅니다. 오픈 소스 커뮤니티에서 모범 사례로 간주되는 지속적 통합(Continuous Integration) 및 회귀 테스트(Regression Test)로 개발자가 잘 되던 것을 망가뜨리지 않도록 해야합니다. 대부분의 프로젝트는 아직도 안정성을 보장하기 위해 (특히 주요 기업 사용자들에게) 공식 릴리스를 제공하기도 하지만 오픈소스에서는 끊임없는 피드백과 개선을 허용합니다.

공개된 토론을 위한 포럼 개설하기

프로젝트에 관한 결정 과정을 숨긴다면 사람들이 프로세스를 존중하거나 프로젝트에 기여하는 것을 기대하기 어려울 것입니다. "열린 소통 문화의 장려" 절에서 설명한 것처럼, 회사 안의 개발자들은 이제 기대하건대 회사 외부의 프로젝트로 커가는 더 큰 커뮤니티의 한 부분이 될 것입니다. 단순한 메일링 리스트 하나면 충분할 수도 있습니다. 토론에 사용하는 모든 도구들은 공개되어야 하며, 그 도구들 자체도 자유 오픈소스 소프트웨어로 만들어져야 하는 등, 참여를 막는 장벽을 두지 말아야합니다.

처음 시작하는 사람들이 쉽게 할 수 있도록 만들기

프로젝트를 시작할때 당신의 코드와 조직의 설립에 익숙하지 않은 사람들을 끌어모으는 것이 매우 중요합니다. 프로젝트가 잘 안착된 이후라고 해도 그것은 중요합니다. 이미 몇 년동안 참가한 사람들만 이해하는 편협한 문화 안에서만 떠다니면 안됩니다. 새로운 사람들을 프로젝트에 끌어들이기 위하여 지속적으로 다음과 같은 활동에 자원을 투입해야합니다:

문서화

빠르게 시작하기 (Getting Started) 안내서부터 당신의 프로젝트가 무엇이 특별하고 가치있는지를 설명하기 위한 기술적 구조 설명까지를 포함합니다. 코드를 기여하기에는 스킬이 부족한 많은 사람들은 문서를 작성하는 기여를 좋아할 가능성이 높으므로, 그런 사람들을 찾아, 프로젝트에 데려와 참여시켜야 합니다.

컨퍼런스와 코더톤*(code-a-thons)*

이 활동은 당신의 의지를 커뮤니티에 보여주며 열정을 불러일으킵니다. 많은 사람들이 이런 행사에 참가하고 난 뒤에 장기적인 기여자가 됩니다. 그래서 이런 행사는 새로운 기여자들을 확보하고 기존 기여자들이 더 많은 것을 할 수 있도록 격려할 수 있는 하나의 좋은 방법입니다. 또 이런 행사을 통해 주요 결정 사항들에 대하여 더 넓은 커뮤니티의 의견을 들을 수 있습니다. 참여하는 것 뿐만 아니라 이벤트를 준비하는 과정에도 커뮤니티의 참여를 유도하는 것이 좋습니다. 회사의 공간에서 피자와 샌드위치 정도를 제공하는 지역 밋업(meetup)은 비용이 적게들면서도 강력한 커뮤니티를 만들 수 있게 합니다.

행동 강령

프로젝트를 시작할 때 부터 존중이라는 것을 주요 가치로 설정하십시오. 너무나 당연하게 들릴지라도, 글로 쓰여진 행동 강령은 매우 중요합니다. 사람들이 무례하거나 독설을 할 때 프로젝트 리더들이 빠르게 개입하여 정리하여야 합니다. 심지어 단 한번의 이슈가 잠재적으로 많은 사용자들을 떠나게 할 수 있습니다. 만약 다양한 젠더나 다른 그룹들을 기꺼이 수용하지 않는다면 재능이 있는 큰 그룹에서 프로젝트 참여자를 찾을 수 있는 기회를 어쩌면 영원히 잃게 될 것입니다. 그리고 또한 나쁜 평판은 당신의 조직에도 영향을 끼칩니다.

해야할 일 목록

사람들이 어느 정도 코드를 사용해 왔다면, 그들은 어떤 일이 도움이 될지를 물어보기 시작합니다. 프로젝트에 대하여 당신과 커뮤니티의 다른 회원들이 우선순위에 입각하여 만든 명확한 할 일 목록을 제공하여야 합니다.

얼마나 많은 시간을 커뮤니티에 쏟을 수 있을지를 개발 단계 별로 가늠해야합니다. 바라건대 다른사람들이 할 일을 고르고 질문에 답변하기 시작할 것입니다. 그러나 궁극적으로는, 사용자들이 원하는 기능을 추가하고, 목소리를 내도록 해야합니다. 사용자들의 요구를 진지하게 받아들이지 않는다면 그들을 잃게될 것이기 때문입니다. 커뮤니티 매니져를 두는 것은 좋은 투자입니다. 직원 가운데 한 사람이 회사 직원들과 외부 커뮤니티 사람들에게 어떻게 프로젝트의 진행을 부드럽고 생산적으로 도울 수 있을지에 대해서 교육할 수 있을 것입니다. 프로젝트가 크게 성장하고 널리 채택되면, 독립적인 관리 조직를 가진 단체로 프로젝트 통제권을 넘겨줄 때가 올지도 모릅니다. 이에 대한 설명은 바로 다음장 "독립된 관리 조직에 프로젝트 릴리즈하기" 절에서 다뤄집니다.

커뮤니티를 잘 운영하기 위한 방법에 대한 심도있는 설명이 필요하다면 Jono Bacon의 책 *The Art of Community* (O'Reilly, 2012)가 매우 유용할 것입니다.

지속적인 소통

보통 주요 이벤트에 대한 보도에 집중하는 홍보 활동과 달라서 커뮤니티와의 소통은 그 이상을 필요로 합니다. 회사라면 커뮤니티와 전 세계가 프로젝트의 각 단계별로 그 이전, 진행 중 및 그 이후의 진화에 대해 알기를 원할 것입니다. 직원들이 블로그에 글을 게시하거나 소셜미디어를 활용하여 새로운 소식을 전하도록 독려해야 합니다. 커뮤니티 내의 누군가가 적어도 2 주일에 한 번 이상은 (큰 프로젝트의 경우 더 자주) 정기적인 트윗을 반드시 해야한다고 정할 수도 있습니다. 커뮤니티 멤버들이 노트북 컴퓨터나 양말 및 티셔츠 등에 붙일 수 있는 스티커를 만드는 것과 같은 작은 브랜드 투자는 프로젝트에 대한 자부심을 드러내고, 참여를 하면 좋겠다고 생각하는 사람들이 볼 수 있도록 프로젝트 이름을 알리는데 도움이 됩니다. 이런 활동은 새로운 사용자를 유치하고 기존 커뮤니티 멤버들에게 이 프로젝트가 활발한 활동을 하는 프로젝트 임을 상기시킵니다.

측정 기준을 정하고 그에 따라 측정하기

데이터가 주도하는 사회입니다. 모든 조직은 의미있는 측정 데이터를 수집하는 방법을 배워야하며, 데이터에 기반한 의사 결정 방법을 직원들에게 가르쳐야합니다. 일부 측정 데이터는 다른 것들보다 모으기 쉽기 때문에, 측정 기준을 정할 때는 실제로 사용자들에게 유용한 것이 무엇인지를 고려하여야 합니다. 따라서 먼저 다양한 측정 데이터를 모은 뒤에, 시간을 두고 어떤 측정 기준이 정말 유용한 지를 찾아내어 줄여나가야 합니다. 일반적인 측정 기준들은 다음과 같습니다:

- 기여자 수와 기여 수, 어떤 사람인지, 소속이 어디인지
- 다운로드 수, 메일링 리스트 참여자 수, 다른 간접 데이터 등을 이용하여 통계적으로 산출 가능한 사용자 수
- 기여자 메일링 리스트 참여자 수의 증가 혹은 감소
- 제보된 이슈 및 버그의 수와 해결된 수
- GitHub의 포크 수 및 스타 개수
- 프로젝트의 웹 페이지, 블로그 방문자 수, 프로젝트와 관련된 트윗 수

CHAOSS 커뮤니티에선 대부분의 프로젝트에 적용 가능한 측정 기준 들을 정의합니다.

보통 측정 결과가 더 높게 나오기를 원할 겁니다. 심지어 버그리포트가 늘어나는 것도 좋은 일 입니다. 이는 코드가 그만큼 유용하다는 뜻이며, 오히려 버그 해결 속도가 더 중요한 측정 기준이 될 수 있습니다. 만약 PR(Pull Request)이 들어오는 횟수가 점점 줄어든다면, 프로젝트를 홍보하거나 프로젝트를 더 매력적으로 만들어 줄 새 기능을 추가하기 위한 작업을 시작해야할 때임을 의미합니다.

대부분의 기여가 소수의 조직에서만 오고 있다면 다양성을 확보하기 위한 노력이 필요합니다. 코드가 한 두 명의 주요 사용자에게 맞춰서 최적화 될 경우, 다른 잠재적 사용자들을 잃을 수 있습니다. 그리고 그 주요 기여자가 갑자기 그만두게 되면 중요한 지원이 끊어지는 사태를 맞게 됩니다.

측정 기준은 상황에 따라 달라질 수 있습니다. 예를 들어, 꼭 필요한 몇몇 사람들을 위한 좁은 응용 범위의 프로젝트에서는 PR 수나 커뮤니티 참여가 안정적으로 유지된다 해도 문제가 없습니다.

측정 결과는 지속적인 개선 과정의 일부가 될 수 있습니다. 관리자가 이 결과를 대시보드로 볼 수 있게 하여 회의나 향후 작업의 우선 순위를 결정하는데 참고할 수 있게 하십시오. 거의 모든 소프트웨어와 마찬가지로, 측정 기준을 표시하는 대시 보드 및 시각화에 적합한 좋은 오픈소스 툴들이 존재합니다. Bitergia는 오픈소스 대시 보드를 제공하고, Amazon은 자신들의 오픈소스 관리부서에서 프로젝트를 관리하는 데 사용하는 OSS attribution builder와 OSS contribution tracker를 공개하였습니다.

독립된 관리 조직에 프로젝트 릴리즈하기

여러분이 이 책의 조언들과 제시된 참고 자료의 내용을 잘 따라왔다고 가정해 봅시다. 여러분의 프로젝트는 성공한 것처럼 보일 것이며, 조직 외부사람들에게도 알려지고 채택될 겁니다. 프로젝트가 커지고 충분히 안정되면, 프로젝트를 회사로부터 독립시키는 것이 중요합니다. 이는 다른 조직들이 재정적으로, 혹은 다른 방법으로 그 프로젝트를 더 잘 지원할 수 있게 만듭니다. 또 이를 통해서 해당 프로젝트가 지속성 있고, 프로젝트의 미래에 대한 의사결정이 특정 회사의 경영적 판단에 의존하지 않을 것임을 세상에 알리는 길이기도 합니다. 그럼으로써 더 많은 사람들이 프로젝트를 사용하고, 기여할 것입니다. 하지만 그 프로젝트를 위해 독립적인 재단을 설립하는 것은 일이 매우 많아서, 정말 재단 설립이 중요한지에 대한 명확한 이유, 즉 주요 기여자들의 요청이나 외부의 조직에서 자금을 모아야 할 필요가 생긴 후에 진행하는 것이 좋습니다.

재단을 세운다는건 복잡한 일입니다. 물론 Linux, Mozilla, OpenStack 등 몇몇 주요 프로젝트는 독립 재단을 세웠지만, 대부분의 오픈소스 프로젝트는-Spark 데이터 처리 툴 등의 유명한 툴도 포함해서-기존 재단의 후원 하에 있습니다. Apache 소프트웨어 재단, Eclipse 재단, Linux 재단 등은 자신들의 원래 미션을 뛰어넘는 매우 다양한 소프트웨어를 후원합니다. 그밖에도 헬스케어 분야의 HL7이나 자동차 분야의 Automotive Grade Linux 등과 같이 특정 산업 영역만을 지원하는 조직들도 있습니다.

오픈소스와 클라우드

지난 십여년동안 인프라, 데이터, 서비스등 다양한 수준에서 클라우드 컴퓨팅이 적용되어 왔으며 이런 움직임은 오픈소스와 궤를 같이 하고 있습니다. 실제로 프리 오픈소스 소프트웨어는 클라우드와 서비스 공급자들에게는 주요한 동력이었습니다. 이 책의 앞부분에는 많은 서비스 공급자들이 나열되어 있으며, 이들은 오픈소스 소프트웨어를 만들어내는 주요 공급자이기도 합니다. 클라우드 사업자들은 Linux와 같은 오픈소스 소프트웨어, KVM과 Xen 같은 가상화 소프트웨어에 크게 의존하고 있습니다. 고객들이 클라우드에서 소프트웨어를 선택할 때 오픈소스 소프트웨어를 사용하는 것도 같은 이유입니다.

클라우드 서비스 공급자와 클라우드 사용자에게 오픈소스 소프트웨어가 제공하는 주된 잇점은 무료 배포입니다. 시작할 때는 10개의 가상머신에서 인스턴스를 사용하고, 피크 때 수요를 감당하기 위하여 인스턴스를 30개로 빠르게 늘릴 수 있습니다. 그리고 다시 10개로 줄여서 비용을 절약할 수 있으며, 이때 개발자당 비용을 따지거나, 비용 구조의 조정 노력이 필요없습니다.

오픈소스 소프트웨어는 경험많은 개발자에게 널리 알려지고 깊이 이해되는 링구아 프랑카(lingua franca, 링구아 프랑카는 서로 다른 모어를 사용하는 화자들이 의사소통을 하기 위해 공통어로 사용하는 제3의 언어를 말하며 국가나 단체에서 공식적으로 정한 언어를 뜻하는 공용어와는 다른 개념이다 - 위키백과)가 되었습니다. 이 점이 각 프로젝트를 사용하는 것의 엄청난 파급효과를 이해하고 있는 클라우드 공급자와 그 고객들에게는 더 매력으로 다가옵니다. 또 테스트가 잘 된 이미 존재하는 소프트웨어를 기반으로 클라우드 사업을 하는 경우 보다 빨리 성장하고, 더 많은 향상된 기능을 적용할 수 있습니다. 예를 들어, 대부분 클라우드 공급자들은 딥러닝같은 최신 기술을 추가하여 경쟁력을 유지하려고 합니다. 공급자들은 오픈소스 도구의 운영을 더 단순화하는 것이 고객들에게 엄청난 가치를 가져다 준다는 것을 알고 있습니다. 성공적인 새로운 프로젝트는 자연스럽게 새로운 서비스의 가장 훌륭한 후보가 됩니다. 이 분야에서 품질 높은 오픈소스 옵션이 많다는 점은 빠른 서비스 업그레이드를 가능하게 하고, 고객들의 여러 개발활동 지원을 더 잘하기 위한 플랫폼 개선도 빠르다는 것을 의미합니다.

고객 또한 공급자가 오픈소스 도구를 사용할 때, 보다 안전하다고 생각합니다. 오픈소스는 락인(lock-in, 특정 벤더에 메임, 역자 주)의 위험을 줄이면서 하이브리드 솔루션을 채택할 수 있게 해주는데, 이는 어플리케이션을 여러 클라우드 서비스 가운데 하나 위에서 실행하거나, 온-프레미스와 클라우드 상에서 동시에 돌리는 선택을 할 수도 있게 해줍니다.

클라우드 공급자들은 자신의 서비스를 소유하고 관리하므로, 도구와 지원 소프트웨어를 오픈소스로 릴리즈하고, 그 소프트웨어들을 개선하기 위해 모인 커뮤니티로부터 이득을 얻습니다.

결어

지난 십 년 동안 오픈소스 소프트웨어의 생산과 활용은 엄청나게 성숙해졌습니다. 어떤 "자비로운 독재자" 주변에서 이루어지던 형식이 없는 협력에서 하나의 규율로 발전해왔습니다. 커뮤니티 관리자, 오픈소스 관리부서, 행동 강령과 잘 정리된 개발 방법과 같은 다른 측면들도 널리 보급되었습니다. 웹사이트는 더욱 정교해지고 좋은 의사소통 방식과 프로세스는 TODO 그룹과 같은 그룹에 의해 GitHub와 같은 협업 개발 사이트에 녹아들었니다. 그리고 프로젝트들은 정말 오랫동안 중요성이 무시되어 왔던, 소스코드의 사촌격인, 문서의 중요성을 인식하고 있습니다.

이 책에서는 오픈소스 활동이 2018년에 가장 기술 주도적인 기업 및 정부 기관들에 의해 미래지향적으로 진행되고 있다고 기술했습니다. 또 여러 정보와 참고 자료들을 종합하여 오픈소스를 추진하려는 시도가 성공으로 이어지게 하고, 우려되는 부분에 관한 질문들에 답을 했습니다. 그렇습니다. 오픈소스 코드를 채택하여 사용하고 릴리스하려면 배워야 할 것들이 많습니다. 그럼에도 불구하고 많은 기업들이 꽤 성공적으로 진행하고 있습니다. 그 기업들은 회사 내의 문화를 바꾸고 이러한 노력에 참여하는 개발자와 모든 직원들을 지원함으로써 오픈소스를 채택의 전략적 가치를 극대화하고 있습니다.

수 많은 회사가 오픈소스 소프트웨어를 사용하고 있으며, 많은 회사들이 오픈소스 소프트웨어에 기여하고 있습니다. 이 책은 파일럿 프로젝트로 개발자 및 기타 주요 이해 관계자들과 협력하면서 이 과정을 시작하는 방법을 설명했습니다. 개발자들을 대상으로 이미 오픈소스 코드를 사용하고 있는지 확인을 위한 조사해보십시오. 조직의 구성원들이 직접 경험을 통해 배울 수 있게 하고 오픈소스 관련한 좀 더 공식적인 정책을 가지는 것이 중요합니다. 커뮤니티가 이런 활동을 시작하는 과정에서 기꺼이 멘토 역할을 해줄 것입니다. 이 책이 참조하고 있는 문서들 가운데 일부를 읽으시고, 오픈소스 프로젝트의 품질을 확인하기 위한 몇 가지 기본적이고 모범적인 사례를 따르십시오. 진행 상황을 문서화하고 그 경험을 바탕으로 다음 단계를 결정하십시오.

기업 업무 관행에 대한 조정과 개정을 통해 딥러닝이나 웹 개발과 같은 영역의 오픈소스 라이브러리를 우선 사용할 수 있을 것입니다. 그 다음 중요한 단계는 오픈소스 소프트웨어를 제품에 통합하는 것입니다. 그리고 그 다음의 큰 단계는 우리 소스코드를 오픈하고 새로운 프로젝트를 오픈소스로 시작하는 것입니다. 이 책은 더 구체적인 정보를 얻을 수 있는 자료와 함께 이러한 노력을 수행하는 조직이 당면할 여러 일들을 요약하여 설명했습니다.

큰 기업들이 오픈소스를 포용하고 있다는 것은 그것이 소프트웨어 개발 방법으로 정착되고 있고 새로운 표준이 되고 있음을 보여줍니다. 우리는 소프트웨어가 정부, 금융 서비스 등 뿐만아니라 농업 및 건설과 같은 전통적인 분야를 포함하는 모든 조직에 더 큰 가치를 부여하고 시장을 변화시키고 있음을 알 수 있습니다. 각 조직은 자체적인 소프트웨어 개발, 오픈소스가 아닌 소프트웨어 제품 또는 서비스 구매, 아니면 오픈소스의 사용 또는 개발 사이의 균형을 선택합니다.

스타트업들도 커뮤니티의 힘을 인식하고 있습니다. 그들에게 오픈소스 프로세스는 자신들이 투입할 수 있는 개발자의 시간이란 제한된 자산을 몇 배로 만들어주는 도구입니다.

이러한 원리는 세부적인 내용이 조금 다르기는 하지만 Arduino와 같은 오픈소스 하드웨어, Creative Commons 라이선스로 공개된 예술 작품들, 오픈 라이선스로 제공되는 데이터, 영국 정부가 정의한 오픈 스탠다드 원칙의 의한 오픈 스탠다드, 정부가 오픈 라이선스를 통해 제공하는 정보 등에도 적용됩니다.

그렇게 오픈소스 소프트웨어는 많은 분야에서 가치를 제공합니다. 이제는 회사의 전략과 문화에 오픈소스 도구와 방법론을 최대한 활용하여 경쟁 우위를 더 높일 때입니다.

또한 외부에는 공개하고 싶지 않은 소프트웨어의 내부협업개발 과정을 포함하여, 오픈소스 개발방법론을 채택하면 조직의 생산성을 높이고 혁신을 보다 빠르게 이루게하며 직원들을 더 행복하게 만들고 의사 결정을 보다 효율적으로 수행할 수 있습니다. 이런 장점들은 오픈소스로 나아가는 여러분들을 위해 준비한 사은품같은 것들입니다.

저자 소개

Andy Oram은 존경받는 출판사이자 기술정보 제공 업체인 O'Reilly Media의 편집자이다. 그는 O'Reilly에서, 영향력 높았던 *Peer-to-Peer*(2001), 획기적인 *Running Linux*(2005)과 베스트셀러 *Beautiful Code*(2007)를 편찬했다.

Zaheda Bhorat는 AWS의 오픈소스 전략 책임자이다. 컴퓨터 과학자로서 Zaheda는 오랫동안 오픈소스 및 오픈 스탠다드 커뮤니티에 적극적으로 기여했다. 구글에서 그녀는 최초로 오픈소스 관리부서를 설립했으며, Google Summer of Code(역자 주, 구글의 오픈소스 여름 인턴 프로그램)을 포함한 성공적인 프로그램들을 만들었다. 또 다양한 기술에 대한 업계 표준(industry standard) 집행위원회에서 구글을 대표했다. 그녀는 또한 영국의 정부 디지털 서비스(Government Digital Service) CTO 실의 선임 기술고문으로서 영국 정부가 ODF(Open Document Format)를 사용하도록 하는 오픈스탠다드 정책을 주도했다.

Zaheda는 선마이크로시스템즈(Sun Microsystems)의 OpenOffice.org, 이후에는 NetBeans.org의 책임자로 열정적인 글로벌 자원봉사 커뮤니티를 구축하였고, 최초의 사용자 버전인 OpenOffice 1.0를 출시하였다. Zaheda는 기술, 교육, 오픈소스, 그리고 공익을 위한 협업의 긍정적 영향에 대하여 열정적이다. 그리고 그녀는 영국 정부가 채택해야하는 표준을 결정하는 오픈스탠다드 위원회의 일원으로 일하고 있으며 Mifos Initiative 이사회의 이사이기도 하다. Mifos Initiative는 금융 기관들이 디지털 방식으로 빈곤층에게 금융 서비스를 제공할 수 있도록 하는 오픈소스 플랫폼을 제공하고 있다. Zaheda는 오픈소스 및 공익과 관련된 주제에 대해 국제적인 강연을 하고 있으며 그녀의 트위터 아이디는 *@zahedab*이다.

역자 소개

(가나다순)

- **김 대희** : 비상교육에서 웹 백엔드 개발자로 일하고 있으며, 현재는 Popit 에서 블로그를 간간히 올리는 중. 스타1,2등의 게임을 좋아하며 평범하고 즐겁게 오래가기를 원하는 개발자입니다.
- **김 영하** :
- **김 정묵** : 오픈소스 스타트업 구성원으로 일하고 있습니다. 가족에게 밥 해주기를 좋아하며 쪼렙 역덕입니다.
- **박 소은** : 행복하게 살기 위해 노력하는 개발자. 웹 서버 관련 기술과 머신러닝에 관심이 많고 게임을 좋아하는 덕후.
- **오 연호** :
- **윤 건영** :
- **윤 종민** : 기계공학을 전공했지만, 여러 소프트웨어 회사를 거쳐 지금은 인사이너리(Insignary Inc.)에서 소프트웨어 엔지니어로 일하고 있다. 리눅스 커널, 밤 하늘, 사진, 그리고 움직이는 모든 것에 대해 관심이 많고, 자유소프트웨어 운동을 지지하고 있음.
- **이 민석** : 국민대학교 소프트웨어학부 오픈소스 소프트웨어 연구실 교수. 어린 시절 오랫동안 하드웨어와 땜질에 심취했었으나 90년대 중반에 소프트웨어로 전향. 많은 회사들과 하드웨어와 소프트웨어를 개발했고, 선후배들과 사업도 했었으나 돈버는 데는 실패하고 인력만 양성하는데만 성공함. 어쩌다 소프트웨어 교육에 발을 담그게 되었고, 지금은 오픈소스 소프트웨어를 널리 퍼뜨리는 일, 개발자가 행복해지는 세상을 만들기 위한 일들을 하고 있음. 페이스북 활동을 하며, 가끔 개인 블로그에 글을 쓰기도 함.
- **이 서연** :
- **이 윤준** : KAIST 전산학부 교수, 1973년 대학 1학년때 KIST 전산교육센터에서 FORTRAN과 COBOL를 배우면서 소프트웨어 분야에 발을 들여 놓아 46년째 이 길을 걷고 있음. 84년 KAIST 전산학과 교수로 부임하여 은퇴 직전임. 하나 DB, BADA DBMS 등 여러 DBMS 개발에 참여했으나 자신의 제품은 없음. 아직 소프트웨어 개발에 흥미를 갖고 있음. 특히 최근에는 소프트웨어 공학에 관심을 갖고 공부하고 있음. 페이스북에 계정이 있으며, 읽은 책들을 book log에 정리하고 있음.
- **장 학성** : https://github.com/hakssung
- **정 원혁** : "전공따위" - 전공에 얽매이지 않고 삽니다. 사회학사, 이러닝석사. 이랜드 개발자로 시작, 마이크로소프트 엔지니어, 퇴사 후 프리랜서 8년. 실직자 구제를 목적으로 (주)필라넷 공동창업. 데이터베이스 컨설팅을 하는 (주)씨퀄로 분사후 11년차. 이제 또 새로 스타트업, "데이터 분석을 돕는 자" 디플러스 대표 3년차.
- **허 경영** :
- **홍 승환** : 국민대학교 소프트웨어학부의 학부생이자, 한 블록체인 스타트업의 소프트웨어 엔지니어이다. 많은 것을 보고 들으며 경험을 쌓고 싶어 다양한 스타트업들에서 일해보며, 오픈소스의 중요성을 몸으로 체감하고 있다. 좋아하는 사람들과 같은 관심사를 공유하는 것을 좋아하며, 지금은 블록체인에 깊은 관심을 갖고 공부하고 있다.

리뷰어

- **이 홍현** : 한성대학교 IT융합공학부

이 책에 나온 URL들

(책에 나열된 순서)

Acknowledgments

Open Source in the Enterprise

- GNU/Linux : https://www.linuxfoundation.org/
- Hadoop : http://hadoop.apache.org/
- Docker : https://www.docker.com/
- Kubernetes : https://kubernetes.io/
- TensorFlow : https://www.tensorflow.org/
- MXNet : https://mxnet.apache.org/
- 미국 국가안보국 : https://thehackernews.com/2017/06/nsa-github-projects.html
- 영국 정부통신본부 : https://github.com/gchq

Why Are Companies and Governments Turning to Open Source?

- 유명한 원칙 : https://quoteinvestigator.com/2018/01/28/smartest/
- 세계 은행(World Bank)의 후원으로 작성된 최근 보고서 : https://opendri.org/wp-content/uploads/2017/03/OpenDRI-and-GeoNode-a-Case-Study-on-Institutional-Investments-in-Open-Source.pdf
- GeoNode : http://geonode.org/
- 많은 정부들 : https://www.csis.org/analysis/government-open-source-policies
- 프랑스 : http://circulaire.legifrance.gouv.fr/pdf/2012/09/cir_35837.pdf
- 미국 : https://code.gov/

More Than a License or Even Code

- GNU General Public License : https://www.gnu.org/licenses/licenses.en.html
- Mozilla Public License : https://www.mozilla.org/en-US/MPL/
- Apache License : https://www.apache.org/licenses/
- 코드 이전에 커뮤니티(community before code) : https://community.apache.org/newbiefaq.html
- reading list : https://www.linuxfoundation.org/resources/open-source-guides/open-source-guides-reading-list/
- Producing Open Source Software : https://producingoss.com
- Cathedral and the Bazaar : http://www.catb.org/esr/writings/cathedral-bazaar/
- guides : https://www.linuxfoundation.org/resources/open-source-guides/
- TODO Group : https://todogroup.org/blog/todo-becomes-lf-collaborative-project/
- opensource.com : https://opensource.com/resources
- InnerSource : https://paypal.github.io/InnerSourceCommons/
- another O'Reilly Media report : https://www.oreilly.com/programming/free/getting-started-with-innersource.csp
- open source program office (OSPO) : https://github.com/todogroup/guides/blob/master/creating-an-open-source-program.md
- case studies by the TODO Group : https://github.com/todogroup/guides)

Groundwork for Understanding Open Source

Adopting and Using Open Source Code

- open source templates and policies : https://github.com/todogroup/policies
- GitHub : https://github.com/

- GitLab : https://about.gitlab.com/
- Linux Foundation Guide : https://www.linuxfoundation.org/using-open-source-code/
- Open Source for the Enterprise : http://shop.oreilly.com/product/9780596101190.do

Participating in a Project's Community

- Heartbleed : http://heartbleed.com/

Contributing to Open Source Projects

- whitepaper : https://drive.google.com/file/d/1woaZ0wjQMbLQhyfB8ZOYveh8cW-jlDPG/view
- TODO Group : https://todogroup.org/opencodeofconduct/
- Amazon Web Services : https://aws.github.io/code-of-conduct.html
- Slack : https://slack.com/
- Gitter : https://gitter.im/
- Stack Overflow : https://stackoverflow.com/
- Google Groups : https://groups.google.com

Launching an Open Source Project

- Linux Foundation Guide : http://bit.ly/2sWo0cO
- Open Source Initiative : https://opensource.org/licenses
- 라이선스 선별 가이드 : https://choosealicense.com/
- LibreOffice : https://www.libreoffice.org/
- The Art of Community : http://shop.oreilly.com/product/0636920021995.do
- CHAOSS 커뮤니티 : https://chaoss.community/
- 측정 기준(Metrics) : http://bit.ly/2sWYq7u
- Bitergia : https://bitergia.com/opensource/
- OSS attribution builder : http://bit.ly/2LOzGVL
- OSS contribution tracker : (http://bit.ly/2JDTA9k)
- Linux : http://bit.ly/2LQAtWa
- Mozilla : https://mzl.la/2HOXeaO
- OpenStack : http://bit.ly/2LPYjBx
- Spark data processing tool : https://spark.apache.org/
- The Apache Software Foundation : http://bit.ly/2JCHspm
- Eclipse Foundation : http://bit.ly/2MsxzZd
- Linux Foundation : http://bit.ly/2JAiZRr
- HL7 : http://www.hl7.org/
- Automotive Grade Linux : https://www.automotivelinux.org/

Open Source and the Cloud

KVM : https://www.linux-kvm.org/page/Main_Page Xen : https://www.xenproject.org

Conclusion

- Arduino : https://www.arduino.cc/
- Creative Commons Licenses : https://creativecommons.org/
- Open License : https://opendatacommons.org/
- Open Standards Principles Defined by the UK Government : http://bit.ly/2JI5H1m
- Open License by Government : http://bit.ly/2JBTxLm

About Authors

- 영국의 Government Digital Service : https://www.gov.uk/
- OpenOffice.org : http://www.openoffice.org/
- Mifos Initiative : http://mifos.org/ (역자추가)
- NetBeans.org : http://www.netbeans.org/
- Twitter @zahedab

역자 소개

- 국민대학교 오픈소스 소프트웨어 연구실 : https://KMU-OSS-Laboratory.github.io
- 이민석 페이스북 : https://www.facebook.com/minsuk.lee0
- 이민석 블로그 : http://hl1itj.tistory.com

애자일 및 스크럼 방법론

원문은 Agile and Scrum methodologies.

애자일 방법론이란 관련자들 사이의 활발한 의견 교환과 정기적인 제품 출시를 반복함으로써 점진적으로 소프트웨어를 완성해 나가는 일련의 소프트웨어 개발 방법론을 가리킨다. 원래는 소프트웨어 개발에 있어 속도와 유연성이 중요해진 인터넷 시대를 맞아 전통적인 소프트웨어 개발 방법론의 대안으로 제안되었으나 이후 그 효율성이 입증되면서 소프트웨어 엔지니어링 뿐 아니라 다양한 전문 분야로 확산되어 가고 있다. 스크럼이란 애자일 방법론의 하나로, 주어진 작업을 스프린트(sprint)라 불리는 1주~1개월 가량의 일정한 기간의 반복으로 나누어서 진행한다. 각 스프린트는 하나의 작동하는 제품 출시를 목표로 하며, 매일 15분 가량의 일일 스크럼(daily scrums)이라 불리는 회의를 진행함으로써 작업 내역과 추후 계획을 점검한다.

덧붙이자면 이 표현은 엄밀히 말하면 잘못된 것이다 - 왜냐하면 애자일속에 스크럼이 포함되기 때문. 예컨대, 춤과 힙합 같은 표현.