차례

Open-Source-in-the-Enterprise

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

```
현재 1차 번역은 완료
PDF를 배포하여 외부 리뷰 준비중
```

Acknowledgments

감사의 글

Creating and growing open source projects and communities takes a village. Throughout this book, we reference projects, processes, books, reports, best practices, and all forms of contributions developed by foundations, companies, communities, and individuals. We trust that these will be incredibly valuable resources on your open source journey and will provide the additional depth needed on each topic.

오픈소스 프로젝트와 커뮤니티를 만들고 성장시키기 위해서는 온 마을이 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 엄청난 가치로운 자원이라고 믿으며 각 주제에 대한 깊이있는 내용을 제공할 것이라고 확신합니다.

On this, the twentieth anniversary of open source, we'd like to acknowledge our deepest thanks to every individual member of an open source community who has contributed to open source in any way for their significant and valuable contributions, in sharing code, tools, lessons, practices, processes, and advocacy for open source for the benefit of all.

오픈소스 20주년을 맞이하여, 의미있는 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.

We'd also like to thank our reviewers who made extensive comments and helpful suggestions Cecilia Donnelly, Karl Fogel, James Vasile, Chris Aniszczyk, Deborah Nicholson, Shane Coughlan, Ricardo Sueiras, Henri Yandell, and Adrian Cockcroft. And finally, thanks to both the O'Reilly Media and AWS teams for supporting this book to bring these resources together.

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 감사드립니다. 마지막으로, O'Reilly 미디어와 AWS 팀이 여러 리소스들을 이렇게 결집된 책을 만들 수 있도록 지원해주셔서 고맙다는 인사를 드립니다.

— Andy and Zaheda

역사 서문

- (작성중 : 이민석)
- (1차 리뷰 :)
- (1차 리뷰 반영 :)
- (2차 리뷰 :)
- (2차 리뷰 반영 :)
- ...

Open Source in the Enterprise

엔터프라이즈에서의 오픈소스

Free and open source software is everywhere, frequently taking over entire fields of computing. GNU/Linux is now the most common operating system, powering data centers and controlling Android devices around the world. Apache Hadoop and its follow-on open source technologies brought the big data revolution to a wide range of organizations, whereas Docker and Kubernetes underpin microservices-based cloud computing, and artificial intelligence (AI) has over-whelmingly become the province of open source technologies such as TensorFlow and Apache MXNet. The major players in computing such as Amazon, Apple, Facebook, Google, Huawei, IBM, Intel, Microsoft, PayPal, Red Hat, and Twitter have launched and maintain open source projects, and they're not doing it out of altruism. Every business and government involved with digital transformation or with building services in the cloud is consuming open source software because it's good for business and for their mission.

자유 소프트웨어와 오픈소스 소프트웨어는 어디에나 있고, 컴퓨팅 분야 전체에 걸쳐서 자주 사용되고 있습니다. GNU/Linux는 지금 가장 널리 쓰이는 운영체제이며, 전세계에 있는 데이터 센터를 움직이고 Android 기기들을 제어하고 있습니다. Apache Hadoop과 마이크로 서비스 기반 클라우드 컴퓨팅을 지원하는 Docker나 Kubernetes와 같은 후속 오픈소스 기술들은 여러 분야의 조직들에 빅데이터 혁명을 불러일으켰습니다. 또한 인공지능(AI) 분야는 TensorFlow와 Apache MXNet과 같은 오픈소스 기술들을 통해 오픈소스 진영의 압도적인 주역이 되었습니다. Amazon, Apple, Facebook, Google, Huawei, IBM, Intel, Microsoft, PayPal, Red Hat, 그리고 Twitter와 같은 컴퓨팅 업계의 주요 기업들은 오픈소스 프로젝트들을 만들고 유지하고 있으며, 그들은 그 기술을 사용하는 다른 사람들을 사랑하는 마음으로 손해를 감수하면서 그 작업을 하고 있는 것이 아닙니다. 전산화 또는 클라우드에서의 서비스 구축과 관련있는 모든 사업 및 정부는 오픈소스 소프트웨어를 사용하고 있으며, 그 이유는 오픈소스 소프트웨어가 그들의 목적과 비즈니스에 적합하기 때문입니다.

It is time for organizations of every size and in every field to include free and open source software in their strategies. This book summarizes decades of lessons from open source communities to present a contemporary view of the trend. We'll help you effectively use the software, contribute to it, and even launch an open source project of your own.

이제는 조직의 분야와 크기에 상관없이 그들의 전략에 자유 소프트웨어와 오픈소스 소프트웨어를 포함해야 할 때입니다. 이 책은 오픈소스 커뮤니티로부터 얻어진 수십 년간의 교훈을 요약하여 지금의 업계 동향에 대한 현대적인 관점을 제시합니다. 우리는 당신이 소프트웨어를 효율적으로 사용하고 그에 기여하며 당신 자신의 오픈소스 프로젝트를 시작할 수 있도록 도울 것입니다.

Not only do companies get better software by utilizing open source, but the dynamics of working in that community-based fashion opens up new channels for creativity and collaboration within these companies. Conversely, institutions that fail to engage with open source will fall behind those that use it effectively.

기업들은 오픈소스를 활용하여 더 나은 소프트웨어를 얻을 수 있으며, 뿐만 아니라 커뮤니티를 중심에 둠으로써 얻어지는 역동성을 통해 창의성과 협업을 위한 새로운 통로를 개척할 수 있습니다. 이와 반대로, 오픈소스를 효율적으로 사용하지 못하는 기관들은 이를 효율적으로 사용하는 기관들에 비해 뒤처질 것입니다.

Finally, it's worth mentioning that trade secrets and confidential business plans can coexist with open source engagement. If even the US National Security Agency and UK Government Communications Headquarters can use open source software, you can, too.

마지막으로, 영업 비밀과 비밀스러운 사업 계획들이 오픈소스 참여와 공존할 수 있다는 점을 언급할 가치가 있습니다. 심지어 미국 국가안보국과 영국 정부통신본부조차도 오픈소스 소프트웨어를 사용할 수 있으니, 당신도 그렇게 할 수 있습니다.

Why Are Companies and Governments Turning to Open Source?

왜 회사와 정부들이 오픈소스로 전환하는가?

There are solid business reasons for using, supporting, and creating open source software. Benefits include the following:

오픈소스 소프트웨어의 사용과 지원 및 생성에 대한 확실한 비즈니스 측면의 이유들이 있습니다. 그로 인해 얻을 수 있는 이익들은 다음과 같습니다:

Multiplying the company's investment

회사에 대한 투자가 배가됨

Open source benefits from the famous principle: "The smartest people in every field are never in your own company." At best, an ecosystem of innovation will grow up around an open project. Evidence that opening a project pays off financially comes from a recent report prepared under World Bank auspices. Careful tracing of contributions to their project —- a form of geospatial software called GeoNode -- showed that the World Bank's subsidiary had invested about one million dollars in the project but had benefited from an estimated two million dollars invested by other organizations.

오픈소스의 이익을 보여주는 유명한 원칙이 있습니다: "어떤 분야에서든 그 중 가장 똑똑한 사람은 당신의 회사에 없습니다." 기껏해야 혁신의 생태계는 개방형 프로젝트를 중심으로 성장할 것입니다. 세계 은행(World Bank)의 후원으로 작성된 최근 보고서에서 프로젝트를 여는 것이 재정적으로 도움이 된다는 증거를 찾을 수 있습니다. 그들의 프로젝트인 GeoNode라는 지형 공간에 관련된 소프트웨어에 대한 기여를 주의 깊게 추적한 결과, 세계 은행의 자회사는 그 프로젝트에 약 백만 달러를 투자했지만 다른 조직들의 투자로 되려 2백만 달러의 이익을 얻었습니다.

Benefiting from the most recent advances

최신 기술로부터의 이익

The AI projects mentioned in the introduction are a good example. Your data scientists will want implementations of the best and most up-to-date algorithms, and these implementations will usually be open source. There is no need to reinvent the wheel in-house. Furthermore, your company can innovate more quickly by using tools and existing code that you can get with a simple download and installation process.

도입부에서 이야기했던 인공지능 프로젝트가 매우 좋은 예시입니다. 당신의 데이터 과학자들은 최고의, 가장 최신인 알고리즘에 대한 구현체를 원할 것이며, 그것들은 대부분 오픈소스일 것입니다. 당신의 회사 안에서 그것들을 다시 개발할 필요는 전혀 없습니다. 나아가, 당신의 회사는 간단한 다운로드와 설치 과정을 통해 얻을 수 있는 도구들과 이미 존재하는 코드들을 통해 더욱 빠르게 혁신할 수 있습니다.

Spreading knowledge of the software

소프트웨어에 대한 지식의 전파

When the code is open and especially when a robust community grows up around it adoption is broader. This initially takes effort on the company's part, but it leads to more people throughout the industry understanding the code and the contribution process.

코드가 공개되어 있을 때, 특히 그 코드를 중심으로 튼튼한 커뮤니티가 성장했을 때 그 프로젝트는 더 널리 사용될 것입니다. 이것은 근본적으로 회사의 노력이 들어가야 하지만, 업계의 더 많은 사람들이 코드와 기여 과정을 이해할 수 있도록 이끌 것입니다.

Increasing the developer base

개발자 인재풀의 증가

Broader adoption, along with wide discussion of the source code, translates into a larger pool of talented developers from which the company can hire to work on the code and related projects.

프로젝트가 소스 코드에 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.

Upgrading internal developer skills

회사에 소속된 개발자들의 능력 향상

The spread of knowledge goes in many directions. Developers already recognize that the best way to learn good coding skills is to work on an open source project because they can study the practices of the top coders in the field. These benefits spread to the company that employs the open source developers.

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

Building reputation

회사에 대한 평판의 향상

Most people want to work for organizations they can boast about. Adopting open source -- both the code and the practices that go with it -- shows that your organization is cool. And if you can release your own code as open source and win adoption for it, you prove that your organization is a leader in your field, adept at the best development practices.

대부분의 사람들은 자랑할 만한 조직에서 일하고 싶어합니다. 오픈소스 프로젝트의 코드와 관행들을 채택하는 것은 당신의 조직이 매우 멋지다는 것을 보여줍니다. 그리고 만약 당신이 당신의 코드를 오픈소스로 배포했고 그것들 다른 사람들이 채택했다면, 당신은 당신의 조직이 그 분야의 리더 역할을 하고 있으며 최고의 개발 관행에 익숙하다는 것을 증명하는 것입니다.

Recruiting and retaining developers

개발자들의 고용과 유지

Good developers want to work on exciting projects that affect large groups of people. They also want their skills and contributions to be widely recognized, and they enjoy the interactions they can have with peers around the world. All these considerations lead them to gravitate toward open source projects, and if your competitors are more successful than you in supporting such projects, developers will bring their talents and reputations to those companies instead of yours.

훌륭한 개발자들은 많은 사람들에게 영향을 끼치는 재미있는 프로젝트에서 일하고 싶어합니다. 또한 그들은 그들의 능력과 기여들이 널리 알려지기를 바라며, 전세계에 있는 동료들과 의견을 교환하는 것을 즐깁니다. 이러한 모든 고민들은 그들이 오픈소스에 끌리도록 유도하며, 만약 당신의 경쟁사들이 어떤 프로젝트를 당신보다 더 성공적으로 지원하였다면 개발자들은 당신의 회사 대신 당신의 경쟁사에 그들의 재능과 명성들을 쏟을 것입니다.

Faster startup of new companies and projects

새로운 회사와 프로젝트의 빠른 시작

In the frenetic pace of today's social and business environments, a startup or new division needs to go from concept to product in months, not years. Working with a community, both on existing software and on your own innovations, saves you time and lets you focus limited employee time on critical competitive parts of your product.

오늘날의 사회 및 비즈니스 환경의 열광적인 속도에서 신생 기업 또는 신규 사업부는 수 년이 아닌 수 개월 만에 컨셉을 제품으로 바꾸어 내야 합니다. 이미 존재하는 소프트웨어와 당신의 혁신에 대한 부분을 커뮤니티와 함께한다면, 당신의 시간을 아껴줄 것이며 제품의 중요한 경쟁력 있는 부분에 한정된 직무 시간을 집중시킬 수 있습니다.

Many governments have launched major open source policies and initiatives. As France and the United States demonstrate, we are now seeing a shift from the use of open source to policies that encourage the development of open source and investment in open source communities. Some have committed to an "open source first" strategy, requiring vendors as well as internal developers to use open source licenses and practices wherever possible. For

example, the government of France has stated that all agencies must do future code work in open source. With such policies, agencies can revise obsolete, expensive, slow procurement practices that have been notorious for causing failed software projects and outrageous cost overruns. For governments, open source becomes a staging ground for the latest, more responsive software practices that have proven more efficient and productive in other sectors.

많은 정부들은 그들의 주요 오픈소스 정책과 이니셔티브들을 발족했습니다. 프랑스와 미국이 보여주었듯, 우리는 정책 분야에서 오픈소스를 사용하는 것에 그치지 않고 오픈소스 개발과 오픈소스 커뮤니티에 대한 투자를 독려하는 것으로의 변화를 보고 있습니다. 일부는 서비스 제공자들과 내부 개발자들에게 어느 곳이든 가능한 곳에서부터 오픈소스 라이선스와 오픈소스 개발 방법론을 사용하는 것을 권장하는 "오픈소스 먼저 (open source first)" 전략을 보여주었습니다. 예를 들어, 프랑스 정부는 모든 기관이 이후의 코드에 관련된 일들을 오픈소스로 해야만 한다고 명시했습니다. 이러한 정책들과 함께, 기관들은 소프트웨어 프로젝트의 실패와 터무니없는 예산 초과의 원인인 악명 높은 쓸모없고 비싸며 느리게 조달되는 관습을 바꿀 수 있습니다. 정부들에게 오픈소스는 다른 분야에서 더 효율적이고 생산성있는 것이 증명된, 최신의 보다 민감한 소프트웨어 개발 방법론들의 시작점이 되었습니다.

Furthermore, governments are realizing that each agency’s needs are similar to other agencies, around the nation and around the world. Open source means, at least, that the investment made by one agency can save money for all the rest and, at best, that the agencies will share requirements and collaborate in the classic open source manner to create software that helps governments better serve their citizens everywhere. Open source collaboration also opens opportunities for smaller companies, citizen developers, and nonprofits to contribute to innovation in government services. Finally, the software creates a common standard that fosters interoperability for many kinds of development.

나아가, 정부들은 국가와 세계를 아울러 각 기관이 가진 필요가 다른 기관들과 서로 비슷하다는 것을 깨닫고 있습니다. 오픈소스는 최소한 한 기관에 의해 이루어진 투자가 전부의 예산을 아낄 수 있으며, 나아가 기관들이 그들의 필요성을 공유하고 전통적인 오픈소스 개발 방식을 통해 협력하며 정부가 그들의 시민들을 위해 봉사하는 데 도움을 주는 소프트웨어를 개발하는 것을 뜻합니다. 또한 오픈소스에서의 협력은 작은 회사들과 시민 개발자들, 그리고 비영리 단체들에게 정부 서비스의 혁신에 기여할 수 있는 기회를 열어줍니다. 결국, 소프트웨어는 다양한 종류의 개발에서 상호 이용성을 키우는 공통적인 표준을 만듭니다.

More Than a License or Even Code

저작권이나 심지어 코드보다 더 중요한 것

In open source, a productive community and its accompanying practices are just as important as the code itself. Officially, of course, open source is defined by a license. Popular ones include the GNU General Public License, the Mozilla Public License, and the Apache License, all of which go through occasional version changes. But in practical terms, you need much more than a license to have a thriving open source project.

오픈소스에서, 생산적 커뮤니티와 그 커뮤니티가 수행하는 프랙티스는 코드만큼이나 중요하다. GNU General Public License, the Mozilla Public License, 그리고 Apache License가 인기 있는 라이센스에 포함되며 종종 버전 변경을 한다. 그러나, 실제적으로 말하자면, 번성하는 오픈소스 프로젝트를 갖기 위해서는 라이센스가 전부가 아니다.

Many people cite the principle “community before code” from the Apache Soft- ware Foundation. At a conference, one open source community leader explained the principle as follows:

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

If you have great code and a dysfunctional community, people will leave and the code will atrophy. If you have dysfunctional code but a great community, people will improve the code.

만약 대단한 코드와 기능을 상실한 오픈소스 커뮤니티가 있다면, 사람들은 떠나고, 코드는 위축될 것이다. 만약 기능을 상실한 코드와 위대한 커뮤니티가 있다면, 사람들은 코드를 고쳐나갈 것이다. 이런 원칙은 회사 문화에도 확대 적용될 수 있다. 여러 다른 팀으로부터 개발자 커뮤니티를 만들어 그들이 더 큰 커뮤니티에서 생산적으로 일하게 하려고 할 때도, 이 원칙은 치명적으로 중요하다.

That observation extends to the culture of your own company, where it becomes crucial to create a community among developers from different teams and let them work productively in the larger project community.

우리는 이 책에서의 오픈소스 여정에 도움될 만한 참고 자료를 제시하면서, 이런 실천을 다음과 같이 요약하려고 한다. 더 자세한 정보는 다음에서 얻을 수 있다.

• An extensive reading list provided by the Linux Foundation. Perhaps the most cited books from this list are Karl Fogel’s Producing Open Source Software (O’Reilly, 2018) and Eric Raymond’s classic The Cathedral and the Bazaar (O’Reilly, 2009).

- Linux Foundation에서 제공하는 더 확대된 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)이다.

• A comprehensive set of guides from the Linux Foundation. These are developed by members of the TODO Group, a collaboration among open source program offices and contributors from companies that have adopted open source principles, practices, and tools.

- Linux Foundation에서 제공하는 포괄 가이드. 이 것은 TODO Group의 회원에 의해 개발되었고,. 오픈소스 프로그램 사무소와 오픈소스 원칙, 실천, 툴을 채용한 여러 회사의 기여자들의 협력이 있었다.

• Resources and answers to questions from the open source community at opensource.com.

- opensource.com에 올라온 많은 질문과 답변, 리소스들.

So powerful are open source practices and community behavior that many companies mimic open source techniques internally, in a process called InnerSource. You can pursue this process, described in another O’Reilly Media report, in parallel with open source participation or on its own.

오픈소스 실천과 커뮤니티 활동은 매우 강력하여, 많은 회사들이 내부적으로 따라하고 있으며 InnerSource라는 내부 프로세스로 불리고 있다. 원한다면 또 다른 O’Reilly Media report에 기술한 것을 참고하여, 오픈소스 활동과 병행하거나, 또는 독립적으로 따라 해 볼 수 있다.

Most organizations—unless they grow organically out of a healthy open source project—greatly underestimate the role of open source culture. This culture is strikingly different from the secretive, hierarchical, management-driven cultures of most companies today. Values of open source projects include listening skills, transparency, collaboration, sharing expertise, mentoring, recognizing merit wherever people demonstrate it, respecting diversity of needs and opinions, and disciplining one's own ego to accept criticism.

오픈소스 활동이 유기적으로 건강한 오픈소스 프로젝트로 자라지 않는한, 대부분 조직은 오픈소스 활동을 과소 평가하고 있다. 이 오픈소스 문화는, 오늘날 대부분 회사의 비밀스럽고, 위계적이고, 관리주도의 문화와 두드러지게 다르다. 시연을 하는 어떤 곳에서나 필요와 의견의 다양성을 존중하며 비판을 받아들이는 자아를 훈련시키면서, 오픈소스 프로젝트의 가치는 경청, 투명성, 협동, 전문성 공유, 멘토링, 장점 인지를 포함하고 있다.

Many companies establish an open source program office (OSPO), where open source is fostered, supported, nurtured, shared, and explained both inside and outside the company. OSPOs are vital for larger organizations that have invested heavily in consuming and contributing to open source software. OSPOs from different companies also collaborate to share best practices that sustain open source development and communities. You can learn more about OSPOs via case studies by the TODO Group.

많은 회사들이 오픈소스 프로그램 오피스(open source program office, OSPO)를 설립하는데, 여기서 오픈소스가 발전되고, 지원되며, 육성되고, 공유되며, 회사 안밖으로 설명된다. 오픈소스 소프트웨어를 사용하고, 기여하는데 있어서 비중있는 투자를 하는 보다 큰 조직에서 OSPO는 필수적이다. OSPO는 다른 회사의 OSPO와도 협력을 하여, 오픈소스 개발과 커뮤니티를 유지하는 모범사례를 공유한다. OSPO에 대해 더 공부하기를 원한다면 TODO Group의 사례연구를 참고하자.

Groundwork for Understanding Open Source

오픈소스를 이해하기 위한 기초

Before discussing open source software from three angles—how to adopt software developed elsewhere, how to contribute to a project, and how to launch a project of your own—let's quickly try to dispel a few myths:

이미 개발된 오픈소스 소프트웨어를 채택하는 방법, 프로젝트에 기여하는 방법 및 자신의 오픈소스 프로젝트를 시작하는 방법등 오픈소스 소프트웨어를 설명하기에 앞서 오픈소스 소프트웨어에 대한 몇몇 오해를 풀어 보고자 한다.

Open source software is low quality or less secure

오픈소스 소프트웨어는 품질이 낮거나 보안성이 떨어진다.

Now that major companies are involved in open source, this myth is not cited so often, but it persists in attitudes that often go unstated. People accustomed to typical procurement processes have trouble believing that something distributed without cost can be high quality. In fact, open source projects have replaced the need to charge for licenses through a number of other funding strategies. But the key issue is that strong open source projects adopt strict quality processes, which your organization can also adopt for your own benefit. As for security, flaws occur in both open source and closed software. Neither is guaranteed to avoid breaches. Experience suggests that transparency and a large development community in open source lead to faster fixes and faster distribution of the fixed software.

최근들어 주요 소프트웨어 기업들이 오픈소스에 참여하면서 이같은 오해가 덜 언급되고는 있지만, 꼭 직접 언급하지 않아더도 아직도 이같은 생각을 하는 경우가 있다. 상용 소프트웨어 구매에 익숙한 사람들에게는 비용을 들이지 않고 분산하여 개발한 소프트웨어가 높은 품질을 보장할 수 있다는 점을 믿기 어려울 것이다. 그러나 실제로 오픈소스 프로젝트는 여러 방법의 자금 지원 전략을 통해 라이선스 비용을 대체했다. 핵심은 엄격한 품질 프로세스를 채택하여 고품질의 소프트웨어를 제공하고 그 결과 고객의 편의를 도모할 수 있다는 것이다. 보안에 대한 결함은 오픈소스와 소스가 공개되지 않은 소프트웨어 모두에서 발생하며, 어느 누구도 이를 피할 수 없다. 오픈소스의 투명성과 큰 규모의 개발 커뮤니티가 보안문제에 대한 보다 신속한 수정과 수정된 소프트웨어 배포로 이어졌었다.

Open source software lacks support

오픈소스 소프트웨어는 고객 지원이 부족하다.

Popular open source projects have many sources of technical support from both organizations and individuals. The open code is a great advantage because you are not locked into a single company for support. Smaller and younger projects might not have yet developed this ecosystem of support, so getting support here might require you to devote more developer time and draw on informal help from the community. Likely enough, you will stumble over a critical bug that requires immediate attention someday, and you will be thankful that you can apply your own developers or a hire a developer to fix the bug instead of waiting for an indifferent vendor's fix.

대중적인 오픈소스 프로젝트는 조직과 개인들로부터 기술 지원을 받고 있다. 지원이 한 회사에 종속되지 않는다는 점이 코드가 오픈된 것의 장점이다. 더 작고 오래되지 않은 프로젝트는 아직 지원 생태계 개발이 충분하지 않을 수 있다. 따라서 개발자들의 더 많은 개발 시간 할애와 커뮤니티로부터 비공식적인 도움이 필요하다. 아마도 언젠가 즉시 수정이 필요한 치명적인 버그를 만났을 때, 무관심한 벤더의 지원을 기다리지 않고 내부 개발자에 요청하여 수정하거나 개발자를 아웃소싱하여 수정할 수 있다.

Open source software projects are unmanaged and chaotic and free for the taking

오픈소스 소프트웨어 프로젝트는 관리되지 않고 혼란스럽고 무료이다.

As you will see through the course of this book, successful open source projects have well-defined processes for decision-making, review of code, and dealing with users like your organization. You must follow certain rules when you use code developed by others. The code almost always has a license, but with different rules from proprietary code. If one of your developers copies code found on the internet into your own products, you will almost certainly be violating a license, and it's a bad practice for legal and other reasons. These points receive more discussion at the Open Source Initiative and Software Freedom Conservancy. Later sections of this book explain current practices for accepting open source code into your organization.

이 책을 통해 알 수 있듯이 성공적인 오픈소스 프로젝트들은 의사 결정, 코드 리뷰 및 사용자에 대한 대응 프로세스를 보통의 조직들과 같이 잘 정의하고 있다. 다른 사람들이 개발한 코드를 사용할 때는 특정 규칙을 따라야 한다. 코드에는 거의 항상 라이선스가 있지만 독점적인 코드와는 다른 규칙이다. 개발자가 인터넷에 있는 코드를 자신의 제품에 복사하면 라이선스를 위반하는 것이 거의 확실하며 법적 또는 다른 이유로 나쁜 관행인 것이다. 오픈소스 이니셔티브 (Open Source Initiative)와 소프트웨어 자유 보호 협회 (Software Freedom Conservancy)에서 이에 대하여 많은 토론이 이루어지고 있다. 다음 섹션들에서 당신의 조직에 오픈소스 코드를 수용하기 위한 몇가지 현재 진행 중인 사례들에 대해 설명하려고 한다.

Using open source software requires you to open your code

오픈소스 소프트웨어를 사용하려면 코드를 공개하여야 한다.

This is something of a reverse of the previous myth. Certainly, you need to be aware of what the license requires before you use open source software. Some licenses have rules for contributing back changes you make, and as you'll see later, you benefit by doing so. (These licenses are sometimes called "viral," but their users dislike the negative connotations of that word; "copy‐ left" is a more neutral term.) Most open source projects, even those with rules for contributing back code, are distributed as libraries that you can link into your own code without opening the functions you write yourself (for instance, the GNU Lesser General Public License).

이는 앞의 오해와 반대된다. 오픈소스 소프트웨어를 사용하기 전에 라이선스에 필요한 사항을 필히 숙지하여야 한다. 일부 라이선스에는 기여한 변경 사항을 반영하기 위한 규칙이 있으며, 나중에 볼 수 있듯이 이렇게하는 것이 좋다. (때로 이러한 라이선스를 "바이럴"하다고 하지만 해당 단어에 대한 부정적 의미 때문에 "카피-레프트"가 보다 중립적인 용어이다.) 대부분의 오픈소스 프로젝트 경우 기여 코드을 포함하여 대부분 개발한 기능을 직접 공개하지 않고 대부분 자신의 코드에 연결할 수 있는 라이브러리 형태로 배포한다.(예 : GNU Lesser General Public License).

You can gain a user base and community by releasing code on a public repository

공개 저장소에 코드를 공개함으로써 사용자 기반과 커뮤니티를 확보할 수 있다.

Opening code out of laziness never works. Open source projects do have an advantage over proprietary ones in gaining adoption, but only if you treat the project as a respected element of your business strategy. Open source projects realize value only as part of an active community. In many cases, the interactions inherent in that process are in and of themselves highly valuable to participants. But open source dynamics reward ongoing investment. You'll see more of how to do this during the course of the book. Inactive projects produce declining benefits over time as the costs of stagnation add up.

공개된 코드가 게으르게 진화되면 결코 작동하지 않는다. 비즈니스 전략 관점에서 오픈소스 프로젝트가 바람직한 요인을 제공할 때 오픈소스 프로젝트는 독점적인 프로젝트보다 채택에 있어서 유리하다. 오픈소스 프로젝트는 활성화된 커뮤니티가 있을 때 그 가치가 실현된다. 그래서 많은 경우에, 그 프로세스에 내재되어 있는 상호 작용은 그 자체로 참여자에게 매우 귀중하다. 이같은 오픈소스 역학은 지속적인 투자를 보상한다. 이 책을 읽는 동안 이 점이 어떻게 이루어 지는가에 대해 더 많이 알 수 있을 것이다. 비활성 프로젝트는 시간이 지남에 따라 정체 비용이 증가하여 혜택이 줄어든다.

An open source project will occupy all your developers' time with support requests

오픈소스 프로젝트에서는 회사의 개발자들이 기술지원을 하는데 대부분의 시간을 쓰게 만든다.

In open source, you trade the time you spend on support for the contributions you get back from the community. Certainly, you must budget time for support, but your company can control how much time to put in and can pull back in order to meet deadlines on internal projects or control excessive support costs. In a successful open source community, all members engage in education, and your company is not solely responsible for providing it.

오픈소스에서는 커뮤니티의 기여에 대한 보상 차원에서 기술 지원에 시간을 투입하는 것이다. 물론 지원을 위한 시간을 미리 정해놓아야하지만 회사가 내부 프로젝트의 데드라인을 맞추거나 과도한 지원 비용을 관리하기 위해 투입 시간을 조절하기도 한다. 성공적인 오픈소스 커뮤니티에서는 모든 회원이 교육을 함께 분담하며 이를 위하여 전적으로 한 회사가 전적으로 책임을 지지 않는다.

Terminology: Free and Open Source

용어: 프리 오픈 소스

The terms free and open source appear interchangeably in this book, because with negligible exceptions, everything that falls under the definition of free software also falls under the definition of open source, and vice versa. The term free soft- ware is used by those who want to emphasize the aspects of liberty, privacy, and sharing, whereas open source is used by those who emphasize its practical and business benefits.

무시할 수있는 예외를 제외하면 자유 소프트웨어의 정의에 해당하는 모든 것은 오픈소스의 정의에 해당하며 그 반대의 경우도 마찬가지이기 때문에 자유 및 오픈소스라는 용어를 이 책에서 서로 바꿔어 사용한다. 자유, 개인 정보 및 공유의 측면을 강조하려는 사람들이 자유 소프트웨어라는 용어를 사용하는 반면, 실용적이고 비즈니스적인 이점을 강조하는 사람들이 오픈소스라는 용어를 사용한다.

Do not use the obsolete term freeware, which used to refer to programs whose developers kept the source code closed while distributing the executable files cost free. This is not free software as currently understood. To be truly free (or open source), the source code must be available and must be under a license that allows its users to modify and redistribute it.

실행 파일을 무료로 배포하지만, 개발자가 소스코드를 공개하지 않는 프로그램을 부르는데 사용하는 예전 용어인 프리웨어라는 말을 더 이상 사용하지 말아야 한다. 그것은 현재 통용되는 프리 소프트웨어는 아니다. 진정으로 프리 (또는 오픈소스)가 되려면 소스코드가 공개되어 이를 사용할 수 있어야 하며 사용자가 소스 코드를 수정하고 재배포 할 수 있는 라이선스가 있어야 한다.

Adopting and Using Open Source Code

오픈소스 코드의 도입과 사용

We trust you are curious what open source code can offer, and perhaps eager to find code that can solve a business need. This section summarizes the key processes you need to adopt to successfully use other people's open source code. The resources cited earlier in the book go into much more detail.

오픈소스 코드가 무엇을 제공할 수 있는지 궁금해하고, 비지니스 요구를 해결할 수 있는 코드를 찾고 싶을 것입니다. 이 섹션에는 다른 사람의 오픈소스 코드를 성공적으로 사용하기 위해 채택해야 하는 주요 프로세스가 요약되어 있습니다. 이 책의 앞부분에 인용된 자료는 훨씬 더 자세하게 설명되어 있습니다.

Create and Document an Internal Open Source Policy

내부 오픈소스 정책 생성 및 문서화

Your development team should know exactly what open source code it is using, and where. This tracking is done by your OSPO or by a virtual team of employees if you have not yet set up an OSPO. Tracking has two main purposes: establishing an audit trail to demonstrate that you are using the code properly, and ensuring that you comply with the license obligations on your third-party open source dependencies. Collecting this information is critical for many reasons; most organizations do so through automated tools in their development cycle.

개발팀은 어떤 오픈소스 코드가 어디에 사용 중인지 정확히 알고 있어야합니다. 이 추적은 OSPO에서 수행하거나 또는 OSPO가 아직 만들어지지 않은 경우 직원으로 구성된 가상적인팀에 의해 수행합니다. 추적에는 두가지 주요 목적이 있습니다. 코드를 올바르게 사용하고 있음을 입증하기 위한 감사 기록을 남기고, 제3자 오픈소스 종속성에 대한 라이선스 의무를 준수하는지 확인하는 것입니다. 이 정보를 수집하는 것은 여러 가지 이유로 매우 중요합니다. 대부분의 조직에서는 개발 사이클 안에 자동화된 도구를 포함시켜 이 정보를 수집합니다.

Writing a strategy paper is valuable to educate managers and employees. Think big and aim for the end state that you are trying to achieve. At the same time, frame the broad, high-level goals in the context of business outcomes. Here are some points that have been have used successfully to explain what open source can do for an organization:

전략 보고서를 작성하는 것은 관리자와 직원을 교육하는데 중요합니다. 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 설명하는데 성공적으로 사용된 몇가지 사항입니다:

- Attract and retain talent
- Increase agility, drive innovation, and accelerate the creation of business value
- Reduce costs and improve efficiency by focusing your staff on writing business logic and by eliminating reinvent-the-wheel heavy lifting
- Generate revenue or gain market share, either through your product or through thought leadership

- 인재를 끌어들이고 유지합니다.

- 민첩성을 높이고, 혁신을 주도하며, 비지니스 가치 창출을 가속화합니다.
- 비지니스 로직 작성에 집중하고, 쓸데 없는 일에 낭비하는 시간을 제거함으로 비용을 절감하고 효율성을 향상 시킵니다.
- 제품 또는 리더쉽 사고를 통해 수익을 창출하거나 시장 점유율을 확보합니다.

Break down your strategy into milestones. This allows you to assign ownership and speed up the delivery of the multiple processes that are needed. In terms of strategy, think about these:

전략을 이정표로 세분화하십시오. 이를 통해 소유권을 할당하고 필요한 여러 프로세스를 신속하게 제공할 수 있습니다. 전략 측면에서 다음 사항을 고려하십시오.

- Open source governance and policies that clarify to the broader company how and when it can use open source

- Policies specifying how developers can contribute to external open source projects: roles and time spent
- Encouraging an open policy in applicable software projects from the start among technology leadership and enterprise architecture groups
- Starting an InnerSource model in tandem, in which you adopt open source practices for internal development across your entire company

- 오픈소스를 사용할 수 있는 방법과 시기에 대해 회사 전반에 알리는 오픈소스 거버넌스 및 정책

- 개발자가 외부 오픈소스 프로젝트에 기여할 수 있는 방법을 명시한 정책 : 역할 및 소요 시간
- 기술 리더쉽 및 엔터프라이즈 아키텍처 그룹 중 처음부터 적용 가능한 소프트웨어 프로젝트의 오픈 정책 장려
- 회사 전체에 걸쳐 내부 개발을 위해 오픈소스 사례를 적용하는 InnerSource 모델을 동시에 시작

Because the adoption of open source crosses many organizational boundaries and can lead to new organizational structures, you might need to explore the creation of new policies that allow developers to collaborate internally across those structures.

오픈소스의 채택은 많은 조직의 경계를 넘어 새로운 조직 구조로 이어질 수 있기 때문에, 당신은 개발자들이 이러한 구조 전체에 걸쳐 내부적으로 협업할 수 있게하는 새로운 정책의 생성을 조사해야 할 수 있습니다.

It is important to find a senior sponsor who will help open doors and champion your cause. This can be the most difficult task we describe in this section, but it is critical. You will need the sponsor on your journey. Make your strategy paper compelling, and they will buy into your high-level, long-range vision. Aim for CTO or CIO support, but be prepared to have to work your way up to that.

당신의 주장을 허용하고 옹호하는데 도움을 줄 수 있는 고위의 후원자를 찾는 것이 중요합니다. 이것은 우리가 이 섹션에서 설명하는 가장 어려운 작업일 수 있지만, 매우 중요합니다. 당신은 여행에서 후원자가 필요할 것입니다. 전략 보고서를 설득력 있게 작성하면, 그들은 당신의 높은 수준의 장기적인 비전을 받아들일 것입니다. CTO 또는 CIO의 지원을 목표로 삼되, 이를 위해 최선을 다해 준비해야 합니다.

Legal staff need to be trained to understand licenses that might be radically different from anything they have dealt with previously. The marketing and PR teams also need to discuss the effects of open source on your practices, quality, and responsiveness to customers. This requires them to study open source practices, collaborate on communication with the communities, and translate these new practices into interactions with your customers. Members of the open source community that you're joining (and probably hiring from) can explain to your marketing team the importance of attending and supporting events held by the community. In addition, the sales team needs to understand licensing well enough to answer questions about using and extending the code, when presenting your solution to customers.

법률 직원은 이전에 그들이 다루었던 것과 근본적으로 다를 수 있는 라이선스를 이해할 수 있도록 교육을 받아야 합니다. 또한, 마케팅 및 홍보팀은 오픈소스가 고객에 대한 관행, 품질 및 반응에 미치는 영향에 대해 논의해야 합니다. 이를 위해서는 오픈소스 사례를 연구하고, 커뮤니티와의 커뮤니케이션에 협력하며, 이러한 새로운 사례를 고객과의 상호 작용으로 전환해야 합니다. 당신이 참여하고 있는 (그리고 아마 고용하고 있는) 오픈소스 커뮤니티의 구성원은 커뮤니티에서 개최하는 이벤트에 참석하고 지원하는 것의 중요성을 당신의 마케팅 팀에 설명할 수 있습니다. 또한, 영업팀은 고객에게 솔루션을 제시할 때, 코드 사용 및 확장에 대한 질문에 대답할 수 있도록 라이선스를 충분히 이해해야 합니다.

If you don't have a clear process, you risk having someone incorporate open source code informally without following good practices. Not only could this violate the code's license, but it deprives you of the benefits of properly using open source code. For instance, critical bug fixes are released regularly by open source projects, and you need to know where you're using this code in order to install the fixes. Instituting clear processes also allows your employees to become valued members of the community and represent your organization's needs there. Finally, a clear and well-communicated policy leads to dramatically more participation and more awareness of your open source initiative throughout the company.

명확한 프로세스가 없다면, 누군가 오픈소스 코드를 좋은 사례를 따르지 않고 비공식적으로 통합할 위험이 있습니다. 이것은 코드의 라이선스에 위배될 뿐만 아니라, 오픈소스 코드를 올바르게 사용하면 얻을 수 있는 이익을 박탈할 수 있습니다. 예를 들어, 중요한 버그 수정은 오픈소스 프로젝트에서 주기적으로 릴리스되는데, 이를 설치하기 위해서는 이 코드가 어디에 사용되는지 알아야 합니다. 또한, 명

확한 프로세스를 수립하는 것은 당신의 직원들이 커뮤니티의 소중한 구성원이 되고, 거기에서 당신 조직의 필요를 대변할 수 있습니다. 마지막으로, 명확하고 잘 전달된 정책은 회사 전체에 걸쳐 당신의 오픈소스 계획에 대한 참여와 인식을 훨씬 더 높여줍니다.

Formalize Your Strategy Through an OSPO

OSPO를 통한 전략을 수립하세요

Developers can participate informally as members of open source communities, but companies who want to fully benefit need to centralize the logistics for supporting open source: legal vetting, project vetting, recruiting developers to work on the code, sponsoring projects and events, managing communications and community relations, and so on. Most companies entering open source have therefore created an OSPO to promote it and handle the associated tasks. Several OSPO leads have collaborated via the Linux Foundation's TODO group to assemble sample open source templates and policies that can be useful to companies getting started. Your OSPO can sponsor the activities we describe in subsequent sections.

개발자들은 오픈소스 커뮤니티의 일원으로서 비공식적으로 참여할 수 있습니다. 하지만 확실한 이득을 얻기를 원한다면 회사는, 법률적 심사, 프로젝트 심사, 코드를 작성할 개발자를 채용하는 것, 프로젝트와 이벤트를 지원하는 것, 커뮤니테이션과 커뮤니티 관계를 관리하는 것 등의 오픈소스 지원을 위한 창구를 일원화해야 합니다. 따라서 오픈소스를 시작하는 대부분의 회사들은 프로젝트를 홍보하고 관련된 업무를 처리하기 위해서 OSPO(역자 주:Open Source Project 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 아래 섹션들에서 서술되는 활동들을 지원할 수 있습니다.

Build Ties Throughout the Company

회사 전체의 연결 관계를 만드세요

After creating a policy, make sure that all of your developers know it. Many other areas of the company, such as the legal and procurement teams, need to be on board as well.

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

First, create a supportive community of open source practitioners within the company. Developers and others who have worked in open source communities can do this at a grassroots level, with or without support and guidance from management. The advocates spend time evangelizing and educating other employees about open source through activities such as regular lunchtime ses- sions, webinars, and presentations at team meetings.

먼저 회사 안에 오픈소스 실무자로 이루어진 지원 커뮤니티를 만드세요. 오픈소스 커뮤니티에서 일한 적이 있는 개발자나 다른 사람들은 경영진의 지원이나 가이드가 있든 없든 이것을 기본적인 정도로 할 수 있을 것입니다. 이들 지지자들은 정기적인 점심시간 세션, 웹 세미나, 그리고 팀 회의에서의 발표와 같은 활동들을 통해서 다른 직원들에게 오픈소스를 전파하고 교육을 합니다.

A training course given to all developers in the company will make sure everyone has the same understanding and expectations about using and contributing to open source.

회사에 있는 모든 개발자를 대상으로 진행되는 교육 과정은 오픈소스를 사용하고 기여하는데 있어서 모두가 같은 이해와 기대를 가지고 있다는 것을 확실하게 할 수 있습니다.

These outreach activities all help demystify open source and accustom the employees to its culture. Most important, those activities uncover other potential champions, so you can begin working with them early on.

이와 같은 지원 활동은 오픈소스에 대한 이해를 돕고 직원들이 그 문화에 익숙해지도록 합니다. 그런 활동들이 중요한 이유는 잠재적인 역량을 가진 사람을 발굴하여 그들과 더 일찍 일을 시작할 수 있게 만든다는 것입니다.

Assess Potential Projects

잠재력있는 프로젝트의 평가

There are plenty of places to find open source code. GitHub and GitLab are two well-known code hosting sites (both use the popular Git version control software developed originally by Linus Torvalds). A search for keywords describing your need (for instance, “employee management”) might well turn up thousands of projects. So be careful to determine that an interesting project really meets your needs. One Linux Foundation guide focuses on the process, and the book Open Source for the Enterprise by Dan Woods and Gautam Guliani (O’Reilly, 2015) also offers guidelines for judging project readiness. Following is a list of typical things to check for:

오픈소스를 찾을 수 있는 곳은 많이 있습니다. GitHub과 GitLab은 잘 알려진 코드 호스팅 사이트입니다. (둘 다 원래 Linus Torvalds가 개발하여 많이 사용되는 버전 관리 소프트웨어인 Git을 사용합니다). 필요한 키워드 (예 : "employee management")를 검색하면 수 천개의 프로젝트가 검색됩니다. 따라서 관심이 가는 프로젝트가 우리 요구 사항에 잘 맞는지를 잘 살펴봐야합니다. 리눅스 재단 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 확인할 사항에 대한 목록입니다.

Code quality

코드 품질

You can assess this by examining the code, checking the ratings (stars) if the project is on GitHub, looking at the number of reported bugs, and seeing what people say about the project online. All code has errors, and you want to see that people report them, so the presence of bug reports is a good thing —so long as the important bugs get fixed.

코드를 직접 검사하고, 프로젝트가 GitHub에 있다면 Stars 등급을 확인하고, 버그 레포팅의 수를 보고, 그 프로젝트에 대한 사람들의 평을 온라인에서 확인함으로써 평가할 수 있습니다. 모든 코드에는 오류가 있기 마련이기 때문에, 중요한 버그가 수정되고 있기만 한다면, 사람들이 버그가 있다는 것을 보고한다는 사실은 그 프로젝트가 유망하다는 뜻입니다.

Active development

개발 활성도

Has the project put out any new releases recently, or at least bug fixes? Are there many pull requests, which indicate interest in the code?

프로젝트가 최근에 새 릴리스를 발표했거나 아니면 적어도 버그를 수정하고 있는가? 코드에 대한 관심을 나타내는 지표인, 풀리퀘스트가 많이 있는가?

Project maturity

프로젝트 성숙도

How long has the project been in existence? Is there an active community? Are there a number of people maintaining the code? Does it have funding? Are books, videos, and other forms of education available?

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

Level of support

지원 수준

Can you get help to install and maintain the software? Is there good docu- mentation for the project?

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

Health of the community

커뮤니티 활성도

Are the mailing lists active? Do developers respond quickly and positively when bugs are reported? Are people polite and productive when responding to issues? A growing community is a good sign but not a necessity, because a project that serves a small user base might turn out to be just right for you. On the other hand, an inactive or declining community is a warning sign.

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

Public decision making

의사 결정의 개방성

Are all choices debated on the public list? If you get a sense that leaders make important decisions privately and just report results to a mailing list or repository, it's a red flag indicating that you might not be able to influence the direction of the project. You might not think such influence is important now, but as you become dependent on the software, you might want to have a say in the future.

모든 선택 사항이 공개되고 논의되고 있는가? 리더가 개인적으로 중요한 결정을 내리고 메일링리스트나 코드 저장소에 결과를 보고한다는 느낌을 받는다면, 외부에서 프로젝트의 방향에 영향을 줄 수 없다는 것을 나타내는 경고 신호입니다. 지금은 그러한 영향력이 중요하지 않다고 생각할 수 있지만, 그 소프트웨어에 대한 의존성이 커지면 나중에 뭔가 말해야할 때 문제가 될 수 있습니다.

Governance/commit access

거버넌스/커밋 권한

Is there a documented way for new contributors to gain commit access? If you invested your time and expertise in contributing to the project, you will want it to support an increase in your responsibilities.

새로운 기여자가 커밋 권한을 얻게 되는 과정이 문서화되어 있는가? 프로젝트에 내 시간과 전문성 투자하고 있다면, 프로젝트에서 그 기여에 상응하는 책임을 맡는 것이 맞을 수도 있습니다.

Security reporting

보안 문제 보고 체계

Professional projects will provide a way for people who discover security flaws to contact the leadership privately so that the flaws can be fixed before advertising their existence.

프로페셔널한 프로젝트라면 보안상의 허점을 발견한 사람이 프로젝트 리더에게 개인적으로 연락하여, 문제가 있음이 외부에 알려지기 전에 수정될 수 있어야 합니다.

Comply with the License

라이선스의 준수

The OSPO or team responsible in your organization for tracking the use of open source code should also make sure you obey the license. This requires attention from both developers, who understand the technical implications of the license, and legal staff. Participants from the open source community have formed the OpenChain Project to make open source license compliance simpler and more consistent and thus encourage more companies to use open source software. Recognizing that it can be difficult to find the license or attribution of code, another open source project called ClearlyDefined helps users to find and maintain this compliance information. The book Open Source Compliance in the Enterprise by Ibrahim Haddad (The Linux Foundation, 2016) covers compliance in depth.

OSPO 또는 조직 내에서 오픈소스 사용을 추적하는 팀은 라이선스를 준수하는지 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. 라이선

스 또는 코드의 속성 찾기 자체도 어려울 수 있기 때문에, ClearlyDefined라는 또 다른 오픈소스 프로젝트로 사용자들이 라이선스 정합성을 위한 정보를 찾고 유지 관리할 수 있도록 도움주고 있습니다. Ibrahim Haddad의 책 기업에서의 오픈 소스 준수(리눅스 재단, 2016)는 라이선스 정합성 문제를 깊이있게 다루고 있습니다.

Build a strong relationship with your legal team and partner with them. They will almost certainly be dealing with incoming work around open source: for instance, they need to approve doing the work under an open source license and check the contributor agreements under which your developers submit code to the projects. This is an opportunity to collaborate on the unfamiliar aspects of open source, build a strong relationship, and earn trust. Use these relationships to help position open source policies in a more favorable light for development and developers.

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

Also form ties with your procurement colleagues. You can add value here by recommending specific additional items to put into procurement contracts around open source and help the team review contracts with third parties.

또한 조달 업무 담당자들과의 관계도 필요합니다. 오픈소스와 관련된 구매 계약에 필요한 추가적인 세부 항목들을 추천하고 그 팀이 써드파티와의 계약을 검토하는 과정에 도움을 줄 수 있기 때문에 이 관계는 가치롭습니다.

Manage Community Code as Seriously as the Code You Create

커뮤니티의 코드를 직접 만든 코드만큼 중요하게 관리하기

Even though an outside organization developed the open source code, you need to manage its use within your organization. It should be subject to testing and to checking for back doors and other flaws. You need to set goals and deadlines for incorporating the code into your own, just as when your employees write code for your company, and devote resources to making sure everything goes as planned.

외부 조직에서 오픈소스 코드를 개발했다 하더라도, 우리 조직 내에서 사용하려면 그 코드를 관리해야 합니다. 외부 코드도 테스트되어야 하고, 백도어가 있는지 확인해야하고, 기타 결함에 대한 검사를 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.

Of course, some tasks are different when you get code from outside. You need to act as part of the community that maintains the code. For instance, you need to integrate your developers with the community's version control system; this might require an account where the community hosts the code, such as GitHub or GitLab. If you alter the code (discussed in more detail in "Contributing to Open Source Projects" on page 17), you should set up a branch of the version control system for your version or use your own internal version control.

물론 외부에서 코드를 가져오는 경우 일부 작업은 다르게 진행됩니다. 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 (**17 페이지의** "오픈 소스 프로젝트에 기여"에서 자세히 설명 됨), 버전 관리 시스템에서 수정을 위한 브랜치 만들거나, 작업을 위한 내부 버전 관리 시스템을 사용해야 합니다.

Change Your Reward and Management Structure

보상과 관리 체계의 변경

Developers working with an open source community will do things differently from closed projects and will need additional skills in communication and negotiation. Your reward structure must reflect the extra tasks you require of these developers. You must set aside developer time for such tasks as learning the tools used by the project, participating in chats and on mailing lists, mentoring less experienced colleagues, checking and modifying other peoples' contributed code, and attending conferences or leadership meetings for the open source project.

오픈소스 커뮤니티와 함께 일하는 개발자는 비공개 프로젝트에서와는 다른 방식으로 일을 하며 의사 소통과 협의를 하는 과정에서 추가적인 역량이 필요합니다. 개발자에게 필요한 추가 작업을 반영한 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.

In particular, as developers and other employees spend time on efforts less directly related to the deliverables of their specific teams, you will need to adjust performance evaluation criteria for both them and their managers. This is especially true for companies that pay performance bonuses. It’s crucial to align the goals and incentives within the company with success factors on the open source project. In turn, when employees attend and speak at conferences, blog about their work on the project, and participate in governance, their contributions reflect well on your company and allow you to reap later benefits such as hiring new developers.

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.

Besides approving these changes, which can be extensive, to the compensation plan for many employees, managers must also recognize that software deadlines are dependent on activities by the community, not just internal employees. Deadlines for software projects are notoriously difficult to set accurately, even when they are run totally by internal teams, so the new dependencies on outsiders might simply help management recognize reality. If the community decides that a feature your company needs is low priority, go ahead and use your own resources to finish it in a timely manner—and contribute the improvements back to the community. Other companies will be doing similar things, and all will benefit.

많은 직원들에 대한 광범위할 수도 있는 보상 계획을 수용하는 것과는 별개로, 관리자들은 소프트웨어 데드라인이 내부 직원뿐만 아니라, 커뮤니티의 활동에 의해서도 좌우될 수 있음을 인식해야 합니다. 소프트웨어 프로젝트의 데드라인은 온전히 내부 팀에 의해 실행되는 경우에도 정확하게 설정하기가 어렵습니다. 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.

Ego and Open Source

오픈소스와 자부심

It is natural to take pride in creative output. Software is rarely any longer a single person’s ingenious inspiration in the way Donald Knuth developed TEX and Metafont on his own. But even when software requires team effort, working tightly together as a team provides its own sense of a unique shared identity. Furthermore, the team can boost a developer’s personal needs. It’s easy in a closeknit group for each member to know the other’s strengths. A manager can see exactly how someone contributed and base pay increases and promotions on those contributions.

창의적인 결과물에 자부심을 가지는 것은 자연스러운 일입니다. 소프트웨어는 더 이상 도널드 커누스(Donald Knuth)가 혼자서 독창적인 영감으로 TEX와 Metafont를 개발했던 것과 같은 방법으로는 개발하지 않습니다. 팀 내에서 긴밀한 협업을 통하여 소프트웨어 개발을 진행하면, 그 팀만의 정체성이 생깁니다. 게다가 팀으로 하는 작업을 통하여 개발자의 스스로의 역량도 키울 수 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.

This psychological and practical reward structure is profoundly changed by open source. We must consider how each developer finds personal fulfillment and anacknowledgment of their contributions in an open source context.

이러한 정신적, 실질적 보상이 오픈 소스로 인하여 많이 바뀌고 있습니다. 오픈 소스 관점에서, 개발자스스로의 기여에 대한 개인적인 성취와 보상을 어떤 식으로 원하는지를 알아야 합니다.

When starting or joining an open source project, a team must give up some control and must learn to work within much more amorphous groups of people who come and go, all motivated by different goals. But ultimately, the individual has an even greater chance to shine on an open source project, because every contribution is available for the world to see. It might turn up in distantly related projects that were never imagined. And demonstrating solid contributions that other people choose to adopt is a wonderful boost to a job application. Open source also rewards people who have skills in communication, project and people management, design, web development, documentation, translation, and much more in addition to coding chops. All contributions matter in a project.

오픈 소스 프로젝트를 시작하거나 참여할 때, 프로젝트를 인위적으로 관리하려 해서는 안 되고, 서로 다른 목적을 가지고 참여했다 사라지는 불특정의 사람들과 함께 일하는 법을 배워야 합니다. 프로젝트에 기여한 내용을 누구나 확인할 수 있기 때문에, 오픈소스로 인해서 개인이 빛이 나게 됩니다. 그리고, 상상하지도 못 했던 엉뚱한 프로젝트에서 그 결과가 나타나기도 합니다. 또한 다른 여러 곳에서 이 결과를 사용하게 되면, 취업의 기회는 놀랄만큼 확대 됩니다. 코딩 뿐만이 아니라, 소통, 프로젝트와 인력의 관리, 디자인, 웹 개발, 문서화, 번역 등 많은 분야에서 오픈소스에 기여할 수 있습니다.

This high visibility of open source projects can also scare managers who worry that their best contributors will be lured by higher salaries or more prestigious jobs offered by a competitor. The solution to that problem is to be that competitor. Make sure your employees have the time to contribute to the open source software you use, both as coders and in a leadership role. Give them the freedom to run with good ideas and provide them the resources to implement useful tools that developers or teams come up with. If you have star performers, encourage them to join boards and speak at conferences. They will help turn your companyinto a magnet for other top developers

오픈 소스 프로젝트를 통해 노출되는 놀라운 성과로 인하여 관리자는 프로젝트의 최고 기여자를 높은 연봉이나 좋은 자리로 경쟁사에게 빼앗기지 않을까하는 우려를 가질 수 있습니다. 이 문제를 해결하는 방법은 바로, 경쟁자가 되는 것입니다. 직원들이 지금 사용하고 있는 오픈 소스 프로젝트에 기여할 수 있도록 하십시오. 지금 사용하고 있는 오픈 소스 프로젝트에 코더나 리더로 기여할 수 있는 기회를 주십시오. 좋은 아이디어를 실행할 수 있는 자유를 주고, 개발자나 팀에서 만들어 낸 유용한 도구를 활용할 수 있는 자원을 제공하십시오. 만일 유명한 개발자를 데리고 있다면, 이사회에 참여시키고, 컨퍼런스를 통해 발표하도록 하십시오. 이렇게 하면, 다른 최고의 개발자를 유혹할 수 있습니다.

Participating in a Project's Community

프로젝트 커뮤니티에의 참여

The health and success of your company will now depend, to a greater or lesser degree, on the health and success of the open source project. Not only should you devote employee time to participation as community members (a topic covered in more detail in the section "Change Your Reward and Management Structure" on page 12), but your company can be a valuable resource. Your experiences with the software, including pull requests, bug fixes, and change requests, can help the community improve the code for everybody's benefit.

이제 회사의 성공 여부는 오픈소스 프로젝트의 성공 여부에 달려있다고 볼 수 있습니다. 단순히 직원이 커뮤니티의 구성원으로써 활동하도록 하는 것 뿐만이 아니라 (이 주제는 12 페이지의 "Change Your Reward and Management Structure"에서 더 자세히 다뤄지고 있습니다), 회사 자체를 귀중한 자원으로 보는 것 입니다. 풀리퀘스트나 버그수정 등, 당신이 소프트웨어와 관련하여 한 경험들은 커뮤니티가 해당 오픈소스 프로젝트를 사용하는 모두의 이익을 위해서 코드를 개선시키는데 도움을 줄 수 있습니다.

Participation means doing the same things other contributors do. Developers working with the code can do the following:

여기서 "참여"라는 것은, 다른 기여자들이 하는 것을 똑같이 하는 것을 말합니다. 코드를 수정하는 개발자들은 아래와 같은 것들을 할 수 있습니다.

- Post questions, ideas, and bug reports
- Contribute fixes and new features, signing the contributor agreements that give the project rights to the contributed code
- Attend code-a-thons and conferences, always being transparent about whom they work for
- Become committers (advanced developers who are trusted to change the core code repository) and participate in the project's governance, as they grow into their community and leadership roles

- 프로젝트에 대한 질문글 작성하기, 새로운 아이디어나 버그에 대해서 이슈 작성하기

- 수정사항과 신규기능을 개발하여 기여하고, 기여한 코드에 대한 권한을 프로젝트에 넘기는 기여자 계약에 서명하기
- 코더톤이나 컨퍼런스에 참가하고, 누구를 위해서 일하는지를 투명하게 밝히기
- 커미터(신뢰받는 숙련된 개발자들로써 저장소의 핵심 코드를 수정 할 수 있는 자)가 되어 커뮤니티와 리더십 역량을 키우며 프로젝트 관리에 참여하기

Their participation will help them gain recognition from the community of users and developers for both their own work and your company's support.

이러한 참여 활동들은, 개발자는 자신의 작업 결과물과 회사로부터 받은 지원을 커뮤니티의 프로젝트 사용자들과 개발자들로부터 인정을 받을 수 있도록 도와줍니다.

Many successful projects in free and open source software follow community principles commonly known as the Apache Way. Although members of these projects might not use exactly the same terms as defined by leaders of the Apache Software Foundation, the principles are recognizable in the projects' day-to-day conduct.

성공한 수많은 자유 소프트웨어 및 오픈소스 소프트웨어 중 대부분은 흔히 Apache Way라고 불리는 원칙들을 따르고 있습니다. 비록 이런 소프트웨어 프로젝트들의 구성원들이 아파치 소프트웨어 재단의 리더들이 정의한 것과 완벽하게 동일한 단어를 사용하지 않을 수는 있지만, 프로젝트의 일상적인 행위에서 이 원칙들이 사용된다는 것을 볼 수 있습니다.

Developers will come to your company with these values or will learn them as they work on open source projects. What's really difficult for the company is that similar values—tempered by the need for confidentiality in certain areas, which every organization experiences—must percolate from the developers through the entire organization. Advocates for open source work need to schedule discussions with management and get their buy-in on the culture

needed to reap the benefits of open source. The value of transparency and community participation must be accepted at all levels of the company by managers, giving developers the freedom to determine where best to invest their efforts.

개발자들은 이러한 가치관을 가지고 당신의 회사에 오거나 오픈소스 프로젝트를 하면서 그러한 원칙들을 배울 것입니다. 회사에 있어서 정말로 어려운 것은 비슷한 가치들이 특정 분야에서 기밀의 필요성에 의해 완화된다는 점인데, 그것은 모든 조직 경험들이 개발자로부터 전체 조직을 통해 퍼져야 합니다. 오픈소스 사업을 옹호하는 사람들은 경영진과의 토론을 계획하고 오픈소스로부터 이익을 얻는 데 필요한 문화를 입수할 필요가 있습니다. 투명성과 커뮤니티 참여의 가치는 관리자들에 의해 회사의 모든 수준에서 받아들여져야 하며, 개발자들에게 그들의 노력을 어디에 투자해야 할지를 결정할 수 있는 자유를 주어야 합니다.

Here are some of the activities developers need to do when joining an open source community:

아래의 목록은 개발자가 오픈소스 커뮤니티에 참여하기 위해서 할 수 있는 활동들입니다.

- Check the code of conduct and contributor agreements, which are important to adhere to
- Become adept at the tools used by the project
- Review the project's documentation (which varies in comprehensiveness and completeness), starting with the README file
- Become comfortable with the process for making contributions provided by the project
- Ask questions and become familiar with other contributors through partici- pation in mailing lists or groups
- Start taking items to work on from the project's TODO list
- Submit bug fixes

- 활동하며 반드시 준수해야 할 행동강령과 기여자 계약을 확인하기

- 프로젝트에서 사용되는 도구들에 대해서 익숙해지기
- README 파일을 시작으로 프로젝트의 (프로젝트에 따라 완성도와 범위가 다양한)문서들 검토하기
- 프로젝트에서 정해져있는 기여를 하는 절차를 숙지하가
- 질문을 하고, 메일링 리스트나 그룹을 통해서 다른 기여자들과 익숙해지기
- 프로젝트의 할 일 목록에서 일거리를 찾아 가져오기
- 버그 수정 제출하기

After your employees gain an understanding of the open source code, your organization might find reasons to make your own changes. Often, a project will not have the exact features that you want. You might also decide to make different performance trade-offs. Open source projects encourage users to send all of their changes back "upstream" to the project repository. Depending on the license and how you plan to use the code, you might be required to do so. Well-designed projects are modularized so that people who want your changes can adopt them and others can ignore them.

회사 내의 개발자들이 오픈소스 프로젝트의 코드를 이해해 나아가며 자연스럽게 직접 수정을 해야할 필요성을 느낄 것 입니다. 이유는, 프로젝트에는 내가 원하는 기능이 없을 수 있기 때문입니다. 한 예로, 기존의 프로젝트와는 다른 성능적 등가교환(trade-off) 방식을 선택 할 수도 있습니다. 오픈소스 프로젝트들은 사용자들이 직접 수정한 내용을 프로젝트의 업스트림(Upstream)에 넣어주는 것을 권장하고 있습니다. 프로젝트의 라이선스나 코드의 사용 용도에 따라서는 수정한 내용을 업스트림에 넣는 것이 강제되기도 합니다. 설계가 잘 되어있는 프로젝트들은 필요한 사람들이 이렇게 추가된 기능을 쉽게 가져다 쓰거나 제거할 수 있도록 모듈화가 되어있습니다.

It can take a long time—even years—to get your changes into the main project, or core. The project leaders might hold off for many reasons: they might decide that no one else needs your enhancements, or be afraid that they're buggy, or worry that they'll have a negative effect on performance. You also can run aground over cultural barriers: your developers simply might not understand the community well enough to gain its attention and trust. But it's worth addressing any concerns the project leaders have and persist in trying to work together with them, not on your own.

당신 회사의 개발자들의 수정사항을 주 프로젝트나 코어모듈에 넣는것은 꽤 긴 시간(때에 따라서는 수년)이 걸릴 수도 있습니다. 프로젝트의 리더들은 당신의 조직에서 개발하여 PR을 요청한 수정사항이 다른 사용자들에게 필요가 없다고 판단이 되거나, 버그에 대한 우려가 되거나, 혹은 성능적 저하를 가져오지 않을까에 대한 걱정 때문에 당신의 수정사항을 머지하지 않고 계류시킬 수 있습니다. 또는 문화

적인 견해 차이에 부딪힐 수도 있습니다. 당신 회사의 개발자들이 아직은 커뮤니티의 관심과 신뢰를 얻지 못한 경우일 수도 있습니다. 하지만 이러한 경우들에도 혼자서가 아닌, 커뮤니티의 개발자들과 함께 소통하고, 프로젝트 리더들이 가지고 있는 문제점들에 대해서 해명을 하는것이 좋습니다.

You might need to set up a separate branch for development or clone the code and take it internal to your organization. This is called a fork. There are usually ways to reintegrate your fork with the main branch later, if the core committers agree to that.

당신의 개발 조직을 위해서 브랜치(branch)를 별도로 만들거나 원래의 코드를 클론(clone)하여 내부적으로 새로운 리파지토리(repo)를 만들어야 할 수도 있습니다. 이러한 과정을 포크(fork)라고 합니다. 포크를 한 브랜치를 추후 메인 브랜치로 병합하는 과정이 있기는 하지만, 이는 프로젝트의 기여자들이 동의해야지만 가능합니다.

When you contribute or push your fixes back, you get the enormous benefit of a thorough and uncompromising code review. The code that ultimately goes in will end up much better than the code you submitted: fewer bugs, better perfor‐ mance, and an architecture that is more generalized for future uses.

코드를 기여하거나 문제를 수정한 경우에 처음부터 끝까지 그리고 타협이란 없는 코드리뷰를 통해 큰 이득을 얻습니다. 결과적으로 프로젝트에 들어가는 코드는 최초에 당신이 넣었던 코드보다 더 적은 버그들, 더 나은 성능수치, 그리고 미래를 고려되어 설계된 아키택쳐(architecture)가 반영된 코드가 들어가게 됩니다.

Furthermore, if your code becomes part of the core, you can simply install any updates that come from the project, secure in the knowledge that they have your enhancements and will work with them. If you fork the code, you must either give up getting new versions (and suffer from any security flaws and bugs the original code contained) or tediously reapply your enhancements to the new ver‐ sion, testing carefully and checking for changes that invalidate your own work.

추가적으로, 만약 당신의 코드가 프로젝트의 핵심적인 부분이 된다면, 그 프로젝트에 발생하는 업데이터를 지속적으로 설치하고 그 업데이트에 당신의 코드가 있다는 것을 확인하면서 같이 일을 하게 됩니다. 만약 코드를 포크하게 된다면, 프로젝트의 새로운 버전들을 포기하여 수많은 버그들과 보안 취약점으로부터 고통받거나 혹은 새로 나오는 버전에 직접 개발했던 수정사항을 매번 다시 넣어주며 수정사항에 대한 테스트와 검증을 직접 하는 것 둘 중 한가지를 선택해야 합니다.

The cost of maintaining forks is quite high over time, and most companies that lack the training to contribute back to the original project also lack the internal processes that would allow them to benefit from the original project’s continued development. They don’t download and incorporate upstream bug fixes, security patches, and feature improvements because the effort of coordinating with the original project increases as the original and custom versions diverge.

포크된 수정사항들을 유지보수하는데는 시간이 지날수록 큰 비용이 듭니다. 그리고 원 프로젝트에 기여하는 훈련이 부족한 회사들의 대부분은 원 프로젝트에 들어가있는 개발사항을 다시 가져오는 내부 절차도 미흡하게 되어있습니다. 원 프로젝트의 업스트림(upstream)에서 수정된 버그와 보안 패치, 그리고 기능 개선을 다운받고 적용하지 않는데, 이 이유는 포크를 한 시점으로부터 시간이 지날수록 업스트림과의 버전차이가 커지고, 그에 따라서 업스트림의 수정사항들을 포크된 곳으로 가져오는 작업 비용이 증가하기 때문입니다.

All that said, you might find that the changes you make to code have no prospect of being accepted into the core. Or, you might have reasons for withholding your changes; for instance, if you need to bury secrets in the code for regulatory or competitive reasons. Some companies choose to maintain a separate branch, publicly or privately, and do the grunt work of reapplying enhancements to new versions. Sometimes, companies contribute part of their changes back but with‐ hold others. Any withholding, however, can reduce the value and appeal of the project.

앞의 이야기에도 불구하고 프로젝트 코드에 우리가 수정한 내용은 받아들여지지 않을 수도 있습니다. 또는 수정사항을 PR 하지 않는 이유가 있을 수도 있습니다. 한 예시로, 규정이나 경쟁 구도 때문에 코드를 숨겨야 하는 이유가 있을수도 있습니다. 몇몇 회사들은 별도의 브랜치를, 공개적 혹은 내부적으로, 유지하기로 결정하고 업스트림의 개선사항들을 포크된 브랜치로 가져오는 작업에 비용을 들이기도 합니다. 또 가끔은 회사에서 직접 개발한 수정사항들중 일부분만을 원 프로젝트에 기여를 하기도 합니다. 하지만 이런 행위는 프로젝트의 가치와 appeal을 저하시킬 수도 있습니다.

Quality and Security: A Comparison of Open and Closed Source

품질과 보안 : 오픈소스와 오픈소스가 아닌 소스의 비교

Just as you evaluate a vendor for quality, good business practices, customer support, and general reputation before buying its goods, you need to evaluate open source software. The section “Assess Potential Projects” on page 9 offers some pointers as a start. With open source software, the risks of making the wrong choice are that you don’t get the quality you expected or that interest in the project dries up and contributors move elsewhere. If the latter problem happens, you probably should move too, although you have the option of taking responsibility for the code and recruiting others to keep it going.

당신은 어떤 공급업체의 상품을 사기 전에 품질을 평가하고, 좋은 사업적인 관례가 있는지, 소비자에게 지원을 잘 하는지 등의 일반적인 평판을 하는 것처럼 먼저 오픈 소스의 품질을 평가할 필요가 있습니다. 9페이지에 있는 "잠재적인 프로젝트의 평가" 부분이 처음에 몇가지 포인트들을 제공해 줄 것입니다. 잘못된 오픈 소스 소프트웨어를 선택을 할 경우 당신이 예상했던 품질을 얻지 못할수 있고, 또는 프로젝트가 엎어지고 기여자들은 다들 다른곳으로 흩어질 수도 있습니다. 만약에 후자의 문제가 발생한다면, 당신은 비록 프로젝트를 유지하기 위해 코드에 관한 책임을 지고 사람들을 고용할 수 있을지라도, 당신 또한 달라져야 합니다.

In return, open source has several advantages over a closed-software vendor. Open source offers you other options if your vendor no longer suits your needs; that is, goes out of business, abandons the field you’re in and takes the product in a different direction, ratchets up the cost of the product precipitously, refuses to fix a bug or add a feature that is important to you, or even inserts malware that tracks your activity. All of these things have happened in proprietary software. Open source software grants you control over your future direction. You can switch vendors and decide how to fix problems in ways that best suit your needs.

오픈 소스는 오픈소스가 아닌 것을 제공하는 사업자에 비해 몇가지 이점을 가지고 있습니다. 오픈 소스는 당신의 공급업체가 더 이상 당신의 요구에 맞추지 못할 경우 다른 옵션을 제공합니다. 즉, 사업을 접게 만들고, 당신이 생각하는 제품의 방향이 아닌 다른 방향으로 제품을 몰게 하며, 제품의 비용을 급격히 상승시키고, 버그를 고치거나 또는 당신에게 중요한 기능을 추가하는것을 거부하거나, 심지어 당신의 활동을 추적하게 하는 악성 소프트웨어를 삽입하는 것입니다. 이 모든 위의 일들은 독점적인 소프트웨어에서 일어났습니다. 오픈 소스 소프트웨어는 당신이 당신의 미래 방향을 정할 수 있도록 합니다. 당신은 공급업체를 바꾸고 당신의 필요에 가장 적합한 방법으로 문제를 해결하는 방법을 결정할 수 있습니다.

Much ink has been strewn about on claims that closed source is higher quality or lower quality or more secure or less secure than open source code. The question has not been resolved either way. To encourage robust development practices, the Core Infrastructure Initiative at the Linux Foundation has created a “best practices” guide covering a wide range of common issues, from source control to testing and code analysis.

오픈소스가 아닌 소스가 오픈소스에 비해 더 품질이 높거나 낮은지, 더 안전한지 덜 안전한지에 대해 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 가이드는 소스코드 관리에서부터 테스트, 코드 분석에 이르는 광범위한 범위에서 공통적으로 발생하는 이슈들에 대해 다룹니다.

Security experts almost universally agree that an open source process is better for secure code and standards, because experts everywhere can evaluate it. On the other hand, the notorious Heartbleed flaw, which lay dormant in the widely deployed open source security software OpenSSL for many years, shows that open source is no panacea. Fortunately, major open source projects made significant, externally verifiable, process changes to prevent something like Heartbleed from happening again. For instance, companies working with the Linux Foundation now collaborate to support the developers working on OpenSSL as full-time contributors.

보안 전문가들은 거의 모든 곳에서 전문가들이 소스를 평가할 수 있기 때문에, 오픈 소스 프로세스가 보안 코드와 표준에 더 좋다는 데 동의하였습니다. 다른 한편으로는. 널리 배포된 오픈 소스 보안 소프트웨어 OpenSSL에서 수년간 휴면했던 악명 높은 하트블리드 결함은 오픈 소스가 만병통치약은 아니라는 것을 보여줍니다. 다행히도, 주요 오픈 소스 프로젝트들은 하트블리드 결함과 같은 일이 다시 일어나는 걸 막기 위해 중요하고, 외부적으로 검증 가능한 프로세스 변화를 만들었습니다. 예를 들어, 리눅스 재단과 함께 일하는 회사들은 이제 OpenSSL에 기여하는 개발자들을 지원하기 위해 협력합니다.

At any rate, security in these highly networked times has become much more complicated than hardening your system. You can assiduously install all patches and avoid opening suspicious attachments but still download malware from a compromised website you visit or fall victim to a phishing scheme. Security must always be viewed holistically.

It is cultural and policy-driven as much as technical.

어찌되었든, 이러한 고도로 네트워크화된 시대의 보안은 시스템을 강화하는 것보다 훨씬 더 복잡해졌습니다. 당신은 모든 패치를 부지런하게 설치하고 의심스러운 첨부 파일을 열지 않더라도, 당신이 피싱 계획의 희생자가 되는 손상된 웹 사이트를 방문하여 멀웨어를 다운로드 할 수 있습니다. 보안은 항상 전체적인 관점에서 보아야 합니다. 그것은 기술만큼이나 문화적이고 정책 중심적입니다.

Contributing to Open Source Projects

오픈소스 프로젝트에 기여하기

If you use open source code, you will get much more from the project by contributing back, both through code and through other practices such as sponsorship and donations. It makes sense to begin by creating an open source project around noncritical software such as a tool in your support software, a platform add-on, or a plug-in.

오픈소스 코드를 사용하고 있다면 소스코드로든 스폰서나 기부 등의 방법으로 그 오프소스 코드 프로젝트에 기여함으로써 더 많은 것을 얻을 수 있습니다. 여러분이 지원하는 소프트웨어에 포함된 도구나 플랫폼 애드온, 플러그인과 같은 중요하지 않는 오픕 소스 코드 프로젝트를 만들어보면서 시작해 볼 수 있습니다.

This section covers making contributions to existing projects, and the next section looks at the extra steps involved in launching one of your own.

존재하는 프로젝트에 기여하는 방법을 설명할 것입니다. 그리고, 이어서 여러분 자신만의 오픈소스 프로젝트를 만들어보는 과정도 살펴볼 것입니다.

Establish the “Why” Throughout the Company

회사 전반에 기여해야하는 "이유" 확립하기

You understand by now that adopting open source is not a matter of copy and paste but represents a serious commitment from your organization. Managers must approve the use of resources for the open source project, including the kinds of participation in the community discussed earlier. You need a story that justifies the investment of checking the code and community and then participating as community members. Thus, you need to explain clearly how the use of the code contributes to the business goals of the organization, and you need to check regularly to make sure that the local goals of the developers stay aligned with these larger goals. A whitepaper from Mozilla and Open Tech Strategies offers ten different overall strategies for creating and running open source projects. The variety of governance structures and motivational frameworks discussed in that paper and the impacts they have on outcomes show how important such choices are.

이제 오픈소스를 적용하는 것은 소스를 그저 복사해 붙여넣기 하는 것이 아니라, 여러분의 속한 조직의 진지한 의지를 의미한다는 것을 이해했을 것입니다. 매니저들은 앞서 이야기했던 커뮤니티에 참여하기 등의 활동을 포함하여 오픈소스에 투입하는 리소스를 승인해야 합니다. 여러분은 코드와 커뮤니티를 확인하고 커뮤니티 구성원으로서 하는 활동에 여러분의 자원을 투자하는 것을 정당화할 만 한 근거가 필요합니다. 그렇기 때문에 오픈 소스를 이용하는 것이 조직의 비즈니스 목표에 어떻게 도움이 되는지 명확하게 설명할 수 있어야 합니다. 또한, 개발자들의 개별적인 목표가 이러한 조직의 기업적 목표에 부합할 수 있도록 주기적으로 확인해야 합니다. Mozilla and Open Tech Strategies에서 발간한 백서에서는 오픈소스 프로젝트를 만들고 운영하는 전반에 대한 10가지 전략이 적혀 있습니다. 백서에서 설명하는 다양한 거버넌스 구조와 동기 부여를 위한 프레임워크들 그리고 그 것들이 미친 영향은 이를 선택하는 것이 얼마나 중요한 지 보여줍니다.

Hire from the Community

커뮤니티로부터 고용하기

A great way to jump-start your use of an existing open source project is to hire qualified contributors who are currently working on the project. Some companies recruit the leaders or key committers, paying them to work full time on the open source project. Other companies strike some kind of balance, allowing a developer to work part time on the open source code and the rest of the time integrating the code into the company or doing other company-specific projects.

기존의 오픈 소스 프로젝트를 이용하는 가장 좋은 출발점은 해당 프로젝트에 활발하게 기여하고 있는 컨트리뷰터를 고용하는 것입니다. 몇몇 기업들은 프로젝트의 리더나 중요 커미터들을 해당 프로젝트에 풀 타임으로 작업하도록 고용하기도 합니다. 또 어떤 기업들은 직원들에게 일정 시간을 오픈소스에 기여하고 그 외의 시간은 해당 오픈소스를 기업에 녹여내는 데에 쓰거나 기업 내부 프로젝트를 진행

하는 식으로 균형을 맞추기도 합니다.

The balance between internal and open source work can be difficult to maintain because managers are always eager to get more help for internal projects and are liable to slip more and more of this work onto the developer's schedule. To recruit developers from the community, your company must show your sincere commitment to the project, through your business plan and your participation in the project. You might also need to review your employee contract and amend it to support developers who want to contribute to their open source projects in their own time. When developers are hired, both management and the developers themselves must stay alert and stick to the original deal about how they divide their time.

사실 내부 프로젝트와 오픈소스 프로젝트를 균형있게 진행하는 것은 쉽지 않을 수 있습니다. 매니저들은 언제나 내부 프로젝트에 더 힘을 쏟기를 바라고 있고, 개발자의 일정에 더 많은 내부 프로젝트 작업을 끼워넣을 수 있기 때문입니다. 회사가 커뮤니티에서 개발자를 고용하기 위해서는 비즈니스 계획과 프로젝트 실제 참여를 통해 해당 프로젝트에 대한 진지한 의지를 보여주어야 합니다. 개발자들이 그들의 시간을 오픈 소스 프로젝트에 투자할 수 있도록 고용 계약을 다시 확인 하고 수정해야 할 수도 있습니다. 개발자가 고용이 되면 관리 부서와 개발자 자신 모두 시간을 어떻게 분배하기로 했는지 꾸준히 신경써야 합니다.

Regardless of how much time you invest in the open source project, a healthy project that you adopt for the right reasons will pay you back handsomely for the reasons stated in "Why Are Companies and Governments Turning to Open Source?" on page 2 earlier in the book.

여러분이 오픈소스 프로젝트에 얼만큼의 시간을 들였던지 적재적소에 활용한 건강한 오픈소스는 4장의 왜 기업과 정부는 오픈소스로 전환하는가?에서 언급했던 대로 여러분에게 반드시 득이 될 것입니다.

Develop Mentoring and Support

멘토링과 지원 마련하기

Your employees will need training, both to work with the new code you are handing them and to join the open source project. If you don't already have employees comfortable with working in an open source environment—and it's worth polling your developers to find out, because you might be surprised to learn that some of them have experience with open source—it's worth hiring developers who do, as explained in the previous section. These developers should also mentor others, because this is a key part of bringing developer resources into open source projects, whether within the company or in the wider community. As mentioned earlier, working on open source projects is an excellent way to improve a company's developer skills.

여러분 회사의 직원이 회사의 새로운 코드 작업과 오프소스 참여 두가지 모두를 하려면 어느 정도 훈련이 필요할 것입니다. 만약 오픈소스 환경에서 일하는 것이 익숙한 직원이 없다면 - 한번 조사해보면 이미 오픈소스 경험이 있는 직원들도 꽤 된다는 점에 놀랄 것 입니다 - 앞 섹션에서 이야기한 것과 같이 이미 익숙한 직원을 고용하는 것도 괜찮습니다. 이 개발자들은 다른 이들을 멘토링 해야할 의무도 있습니다. 그럼으로써 회사 내부에서든 더 넓은 커뮤니티에서든 개발자들은 오픈 소스 프로젝트에 참여시킬 수 있기 때문입니다. 앞서 언급했듯, 오픈 소스프로젝트에 참여하는 것은 회사의 개발 능력을 향상시키는 훌륭한 방법입니다.

Set Rules for Participation

참여를 위한 규칙 설정하기

Open source changes every organization that adopts it, even just to participate in one outside project. Developers at many levels of the organization, and sometimes other employees, are now exposing their conversations and decisions to public scrutiny. Assume that what a developer says in an online forum for an open source project will be seen by the entire world and will last online forever. This might sound like a reason to discourage participation, but it's not. It just means that you need to establish rules for online participation, such as a code of conduct that enforces honest but respectful dialog and that lays out the rights of people who regularly experience discrimination or abuse. Many open

source projects provide good codes of conduct that you can adopt. Several codes that have been carefully tested and vetted already are offered by the TODO Group. Revised versions of this code of conduct are implemented by companies such as Amazon Web Services across their open source projects.

오픈소스는 단지 외부 프로젝트 하나에 참여하더라도 오픈소스를 채택하는 모든 조직을 변화하게 합니다. 해당 조직에 속한 다양한 수준의 개발자들, 때로는 다른 직원들도 이제는 대화와 결정 사항을 공개 조사에 노출하고 있습니다. 오픈소스 프로젝트 온라인 포럼에서 한 개발자가 한 말이 전세계에 보여지고, 온라인 상에 영원히 남겨진다고 가정해 봅시다. 이게 참여할 의지를 떨어뜨릴 이유 같이 들릴 수 있지만, 실제로는 그렇지 않습니다. 이건 단지 정직하지만 존중이 담긴 대화를 강제하고, 자주 차별과 학대를 경험하는 사람들의 권리를 규정하는 등의 행동 강령(Code of conduct)과 같은 온라인 참여 시의 규칙을 확립할 필요가 있다는 것을 의미할 뿐 입니다. 많은 오픈소스 프로젝트가 채용할 만한 좋은 행동 강령을 제공하고 있습니다. TODO Group은 주의 깊게 시험하고 검증한 여러 강령을 이미 제공하고 있습니다. 이런 행동 강령의 개정본이 아마존웹서비스와 같은 회사의 오픈소스 프로젝트들에 걸쳐 시행되고 있습니다.

There has recently been a heightened awareness of practices in the computer industry (and other parts of society) that discourage the contributions of women and minorities, or that create a harassing and unsupportive environment. These are habits you'll want to root out of your organization, regardless of whether they're exposed to public scrutiny. The company and community must take positive and affirmative steps at diversity and inclusion.

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

Foster Open Communication

열린 커뮤니케이션을 촉진하기

Beyond showing respect, your developers need to engage with the open source community productively, which might require a culture change. Developers might be used to calling an informal meeting of two to four people and making design decisions informally. Some kinds of Agile and Scrum methodologies encourage this behavior, but they also provide techniques for accepting input from wide swaths of the organization. Therefore, Agile and Scrum can be adapted to open source. In fact, open source is open to a wide range of methodologies.

존중을 보이는 것을 넘어, 여러분의 개발자들은 오픈소스 커뮤니티에 생산적으로 참여하기 위해 문화의 변화가 필요할 수 있습니다. 개발자들은 두 명에서 네 명 정도의 비공식 미팅을 소집해서 설계 결정을 내리는 데에 익숙했을 것입니다. 일부 애자일 스크럼 방법론(Agile and Scrum methodologies)은 이런 방식을 지지하기도 하지만, 조직 내의 폭 넓은 의견을 수렴하는 기술 또한 제공합니다. 따라서, 애자일 스크럼 방법론은 오픈소스에 적용할 수 있습니다. 실제로 오픈소스는 다양한 종류의 방법론에 열려 있습니다.

Developers must now learn to conduct all discussions of significant design issues in forums where the decisions are archived for others to view. For changes that you intend to contribute back to the community, discussions must be on the community mailing list or communication channels everyone can join, such as Slack, Gitter, Stack Overflow, or Google Groups. Otherwise, you will take the community by surprise when you submit changes and will probably fail to get changes accepted.

개발자들은 이제 다른 사람들도 볼 수 있도록 기록이 보관되는 포럼에서 중요한 설계 결정에 대한 모든 논의를 수행하는 법을 배워야만 합니다. 커뮤니티에 역으로 기여하고자 하는 변경 사항에 대해 커뮤니티 메일링 리스트 혹은 모든 사람이 참여할 수 있는 슬랙, 지터, 스택 오버플로, 구글 그룹스와 같은 커뮤니케이션 채널을 통해 토론해야 합니다. 그렇지 않으면, 변경 사항을 제출할 때 커뮤니티를 화들짝 놀라게 하고 변경 사항은 아마도 받아들여지지 못할 것입니다.

In addition to providing information in a timely manner, developers must also look beyond the narrow horizons of their team and recognize the value of input from a diverse range of people, either from other parts of your company or from outsiders who might live halfway around the world. This is where techniques of respectful dialog must enter your corporate culture. Developers need other new skills, as well: they must be good listeners, take feedback from a variety of developers, reject unsuitable contributions while encouraging the contributor to learn and try again, and deal with people from different backgrounds with varying language skills.

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 개발자들은 다른 새로운 기술도 필요로 합니다: 타인의 말을 잘 들어주어야 하고, 다양한 개발자들로부터 피드백을 받아야 하며, 부적절한 기여를 거절하는 한편으로는 기여자가 공부하여 다시 시도하도록 격려하고, 다양한 언어 능력을 가진 서로 다른 배경의 사람들을 다루어야 합니다.

When developers are accustomed to making design decisions in hallway conversations or over the telephone, they might need some time to learn to make decisions in more open ways and to use the less-intuitive communication media provided by mailing lists and bug trackers. Joining an open source project is an excellent way to raise the priority of good communications, honest disagreement, and open, respectful dialog throughout your organization.

개발자들이 복도에서의 대화나 전화 통화를 통해 설계 결정을 내리는 데에 익숙해져 있다면, 좀 더 공개된 방식으로 결정을 내리고 메일링 리스트나 버그 추적기가 제공하는 덜 직관적인 커뮤니케이션 매체를 사용하는 방법을 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.

Decisions can take longer when the crowd become involved. If you short-circuit the open source discussion process and make a quick decision in order to get a product out the door, you might need to roll back some of the work done later so that the code can be maintained and can adapt to future needs.

군중이 참여하게 되면 결정에 더 오랜 시간이 걸릴 수 있습니다. 제품을 빨리 내놓기 위해 오픈소스 토론 절차를 짧게 줄이고 빠른 결정을 내려 버리게 되면, 코드가 유지되고 미래의 요구의 받아들일 수 있게 하기 위해 수행한 작업의 일부를 롤백해야 할 수도 있습니다.

An important side benefit of diligently recording decisions is that the project becomes less dependent on particular individuals and can reorganize itself gracefully if key individuals leave. No single person has unique or irreplaceable information.

성실하게 결정 사항을 기록할 때에 한 가지 중요한 부차적인 이점은 프로젝트가 특정 인물에 덜 의존하게 되고, 핵심 인물이 떠나더라도 무리없이 프로젝트를 재구성할 수 있게 해준다는 것입니다. 어느 한 사람도 유일하거나 대체불가능한 정보를 가질 수 없습니다.

Launching an Open Source Project

오픈소스 프로젝트의 시작

As your organization comes to recognize the benefits of the open source development model, you might decide to start a project of your own. Careful planning can make the difference between a quickly forgotten fiasco and a thriving, sustainable code base. A Linux Foundation guide(http://bit.ly/2sWo0cO) offers more details about the process and a comprehensive project launch checklist. All the principles of the previous sections apply, and some further considerations are summarized in this section.

오픈 소스 개발 모델의 장점을 여러분이 속한 조직이 알게 되면서, 새로운 자신만의 오픈 소스 프로젝트를 시작하고자 결정했을지도 모릅니다. 사려깊게 계획을 해야 소스코드가 빠른 흥망성쇠를 통해 잊혀질지 계속 유지하게 될지 만들 수 있습니다. 리눅스 재단 가이드(http://bit.ly/2sWo0cO)에서는 그 과정에 대해서 자세히 설명하며, 종합적인 오픈 소스 프로젝트을 시작할 수 있는 체크리스트를 제공합니다. 앞에서 설명한 모든 이론들이 당연히 적용되며, 좀 더 고려할 만한 사항들을 설명할 것입니다.

Choose a License

라이선스의 선택

Your legal staff must understand how you plan to use the code, including whether you will make closed enhancements for internal use. The license you choose will control your own use as well as the decisions other companies and individuals make to adopt your software. So, make sure it is aligned to the business's strategy.

법무팀에서는 앞으로 내부 사용을 위하여 비공개로 개선할 것인지를 포함하여, 코드를 어떻게 활용할지를 파악하고 있어야 합니다. 라이선스의 선택은 다른 회사나 개인들이 그 소프트웨어의 채택 여부를 결정하는 것뿐만 아니라 회사 자신의 사용에도 영향을 줍니다. 그렇기에, 라이선스는 회사의 비지니스 전략을 고려해야 결정해야 합니다.

There is no reason to get fancy, though. Earlier in this book, we mentioned a few of the most popular licenses. Although there is a long list of open source licenses at the Open Source Initiative, we strongly urge that you to go with one of the popular and recent licenses. GitHub offers a guide with a selection of well-crafted licenses. If you don't think one of these meets your needs, go back and candidly reevaluate your motivation for going open source. If you can't use a popular license, you are probably trying to do something out of sync with open source principles, and could end up driving away the people whom you want to attract.

복잡한 라이선스를 채택할 필요는 없습니다. 이 책의 앞부분에서 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 원칙과 어긋나는 방향으로 프로젝트를 진행하려는 것일 수 있으며, 그로 인하여 우리가 프로젝트에 끌어들이고 싶었던 사람들을 떨어져 나가게 할 수 도 있습니다.

Open the Code Right Out of the Gate

코드를 문 밖에 내놓기

It's best to create a public repository and invite participation from outside your company before you create a single line of code. Otherwise, you won't be able to open your code until you review it thoroughly to strip out proprietary secrets and embarrassing comments. Opening the code from the start—part of an "open source first" approach—will encourage your own developers to follow best coding practices.

코드의 첫번째 라인을 작성하기 전에 먼저 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 우선" 방식으로 처음부터 코드를 공개하면 개발자들이 최상의 코딩 방법을 따르게 될 것입니다.

Select a name for your project that will resonate with the community. Names can be very personal (for example, Linux was named after the creator of the operating system) or can describe the software, as in the office software called LibreOffice. The project can be named after a mascot or even be a nonsense word chosen to be easy to remember (such as Hadoop). Run through the same checks and due diligence that your legal team does for trademarks to ensure that no duplicate exists, that the name is not offensive in some language, and so on. See the section “Keep Up Communication” on page 22 for information on promoting your project and brand.

프로젝트 이름은 커뮤니티와 공감할 수 있도록 선정하십시오. 이름은 아주 개인적일 수도 있고(예 : Linux는 운영체제 작성자의 이름을 따서 명명됨) [LibreOffice] (https://www.libreoffice.org/)라는 오피스 소프트웨어처럼 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 커뮤니케이션 유지하기 섹션을 참조하십시오.

The first code you open could be messy, if it’s created by people used to internal team work instead of open source development. Opening immature code can be difficult for both your developers and your managers to accept. But simply docu- ment the progress you’ve made and what you need from the community, and you will be rewarded by them—given, that is, a commitment to winning over and mentoring new users. If you have a good idea, many hands will reach in to fix your problems. If you don’t have a good idea, you’ll hear that unpleasant news from outsiders and will be able to abandon the project before wasting more developer time.

오픈소스 개발 방식 대신 내부 팀 작업에 익숙한 사람들이 만든 첫 번째 코드는 지저분할 수 있습니다. 완성도가 낮은 코드를 공개하는 것은 개발자와 관리자 모두 수용하기 어려울 수 있습니다. 그러나 이미 진행된 내용과 커뮤니티로부터의 도움이 필요한 내용을 문서화하면, 새로운 사용자를 수용하고 그들에 대한 멘토링을 약속하면 커뮤니티로부터 보상을 받게 됩니다. 공개한 내용이 좋은 생각이라면 많은 사람들이 손을 내밀어 같이 문제를 해결할 것입니다. 혹시 그 내용이 좋은 생각이 아니라면, 외부로부터 좋지 않은 반응들이 들릴 것이며, 더 이상 개발자 시간을 낭비하지 않고 프로젝트를 포기할 수 있습니다.

Use Best Practices for Stable Code

안정된 코드를 만들기 위한 모범적인 개발 방법의 사용

You can use one of the popular public repositories to store code and documentation, or can set up a version control system of your own with public access. Have developers check in their changes daily or as often as needed. Developers refer to this practice as “release early, release often.” Continuous integration and regres- sion tests (both considered best practices in open source communities) should ensure that the developer doesn’t break anything. Most projects still offer formal releases in order to guarantee stability (especially to major corporate users), but open source permits feedback and improvements on a continuous basis.

공개 코드 저장소에 코드와 문서를 저장하거나 내 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 테스트로 개발자가 잘 되던 것을 망가뜨리지 않도록 해야합니다. 대부분의 프로젝트는 아직도 안정성을 보장하기 위해 (특히 주요 기업 사용자들에게)공식 릴리스를 제공하지만 오픈소스에서는 끊임없는 피드백과 개선을 허용합니다.

Set Up Public Discussion Forums

공개 논의 포럼을 설정하기

You can’t expect people to respect your process and make contributions if you hide decisions from them. As explained earlier in “Foster Open Communication” on page 19, developers within your company are now part of the wider community that hopefully will flock to the project outside your company. A simple mailing list might be all that you need. Any tools you use for discussions should be open to the public and should be based themselves on free and open source software so that you don’t put up barriers to participation.

결정을 숨기게 된다면 사람들이 프로세스를 존중하는 것과 기여를 만드는 것에 대해서는 기대하기 어려울 것입니다. "열린 커뮤니케이션을 촉진하기"에서 설명한 것처럼, 회사 안의 개발자들은 이제 기대하건대 회사 외부의 프로젝트로 몰려가는 더 큰 커뮤니티의 한 부분이 될 것입니다. 필요한 것은 단순한 메일링 리스트 하나일지도 모릅니다. 논의에서 사용하는 모든 도구들은 공개되어야하며, 이것 자체가 자유 소프트웨어 그리고 오픈소스 소프트웨어의 기반이 되어야 하기 때문에 참여하기 위한 장벽을 두지 말아야합니다.

Make Life Easy for Newbies

처음 시작하는 사람들이 쉽게 할 수 있도록 만들기

Attracting people who are unfamiliar with your code and organizational setup is critical at the very start of your project, and it remains important even after you are well established. Do not drift into an insular culture understood only by people who have participated for several years. Devote resources constantly to activities that draw in new people, such as the following:

프로젝트를 시작할때 당신의 코드와 조직의 설립에 익숙하지 않은 사람들을 끌어모으는 것은 매우 중요합니다. 그리고 이것은 잘 설립이 된 이후에도 중요하게 남습니다. 이미 몇 년동안 참가한 사람들만 이해하는 편협한 문화안에서만 떠다니지 마십시오. 다음과 같은 새로운 사람들을 활동에 참가시키기 위해서는 지속적으로 리소스들을 쏟아야합니다:

Documentation

문서화

This ranges from quick Getting Started guides to architectural descriptions that explain what is unique and valuable about your project. Many people who lack the skills to contribute code would be happy to write documentation; you need to find, recruit, and engage them.

빠르게 시작하기 안내서(quick getting started guides) 부터 이 프로젝트가 무엇이 특별하고 가치있는지를 설명하는 구조적인 기술까지를 포함합니다. 코드를 기여하기에는 스킬이 부족한 많은 사람들은 문서를 작성하는 것에 매우 기뻐할 것입니다. 이런 사람들을 찾고, 채용하고, 참여시켜야 합니다.

Conferences and code-a-thons

컨퍼런스와 코더톤*(code-a-thons)*

These demonstrate your commitment to the community and foster enthusi- asm. Many people become long-term contributors after attending such events, which are a good way to recruit new contributors and inspire existing contributors to do even more. They also ensure that the wider community is heard when key decisions are made. Get community involvement in organizing the events as well as attending them. Local meetups can be cheap to organize —offer a space in your offices and buy a few pizzas and sandwiches— and can build strong community support.

이것들은 당신의 헌신(몰입,약속,책임,의지)을 커뮤니티에 증명하고 열정을 가속화시키는 것입니다. 많은 사람들은 이런 종류의 행사를 참가하고 난 뒤에 장기적인 기여자가 됩니다. 그렇기 때문에 이런 행사는 새로운 기여자들을 채용하고 기존의 기여자들이 더 많은 것을 할 수 있도록 격려할 수 있는 하나의 좋은 방법입니다. 이것은 또한 주요 결정이 더 넓은 커뮤니티가 들을 수 있도록 합니다. 참석하는 것뿐만 아니라 이벤트를 준비하는 데 있어서 커뮤니티의 참여를 유도하세요. 지역 밋업(meetup)은 준비하는데 비용이 적게 — 사무실의 공간을 제공하고 피자와 샌드위치 같은 것을 제공하세요. — 들 수 있습니다. 그리고 강력한 커뮤니티 지원을 형성할 수 있습니다.

Code of conduct

행동 규범

Establish respect as a key value of your project from the start. A written code of conduct is critical, even if what it says seems obvious to you. Make sure that project leaders intervene quickly when people are rude or abusive. Even a single tense exchange can drive away a substantial number of users. If you are not welcoming to diverse genders and other groups, you will lose (perhaps forever) the chance to recruit from a big group of talent. And the bad reputation will stain your organization, as well.

프로젝트를 시작할 때 부터 존중이라는 것을 주요 가치로 설립하세요. 보기에 너무나도 당연하게 보일지라도, 글로 쓰여진 행동 규범은 중요합니다. 사람들이 무례하거나 독설을 할 때 프로젝트 리더(지도자)들이 빠르게 중재할 수 있도록 하세요. 심지어 하나의 긴장이 오고 가는 것이 잠재적인 많은 사용자들을 떠나게 할 수 있습니다. 만약 다양한 성별과 다른 그룹들을 환영하지 않는다면 재능의 큰 그룹으로부터 채용할 수 있는 기회를 (어쩌면 영원히) 잃게 될 것입니다. 그리고 또한 나쁜 평판은 조직을 얼룩지게 만들 것입니다.

To-do lists

해야할 일 목록

When people have been using your code for a while, they begin to ask how they can help. Provide a prominent list of tasks that you—and other members of the community—have identified as priorities.

사람들이 일정 기간동안 코드를 사용할 때, 어떻게 그들이 도울 수 있을지 물어보기 시작합니다. 해야할 일에 대해서 직접— 그리고 커뮤니티의 다른 회원들이— 지정한 우선순위에 따르는 명확한 목록을 제공하세요.

You need to assess at different stages in development how much time you can dedicate to the community; hopefully others will start to pick up the task and answer questions. Ultimately, though, you'll lose users unless you take their needs seriously, support them in creating the features they want, and give them a say. Hiring a community manager is a good investment: such a member of your staff can educate both company employees and outside community members about how to help the project progress smoothly and productively. As the project grows big and widely adopted, it might become time to hand control over to an independent governance organization altogether, which is the topic of an upcoming section ("Release the Project to an Independent Governance Organization" on page 24).

얼마나 많은 시간을 커뮤니티에 쏟을 수 있을지를 개발 단계에 따라 가늠해야합니다; 바라건대 다른사람들이 할 일을 고르고 질문에 답변하기 시작할 것입니다. 그러나 궁극적으로 사용자들의 요구를 진지하게 받아들이지 않는다면 그들을 잃게될 것이기 때문에, 그들이 원하는 기능을 추가하고, 목소리를 내도록 해야합니다. 커뮤니티 관리자를 고용하는 것은 좋은 투자입니다: 가령 직원중의 한 사람은 회사 직원들과 외부 커뮤니티 사람들 모두에게 어떻게 프로젝트의 진행을 부드럽고 생산적으로 도울 수 있을지에 대해서 교육할 수 있을 것입니다. 프로젝트가 크게 성장하고 널리 채택되면, 독립적인 운영 체계를 가진 단체로 통제권을 넘겨줄 때가 올지도 모릅니다. 이에 대한 설명은 바로 다음장 (독립된 관리 조직에 프로젝트 릴리즈하기)에서 다뤄집니다.

For an in-depth discussion of how to work well with a community, the book The Art of Community by Jono Bacon (O'Reilly, 2012) is very useful.

어떻게 커뮤니티 안에서 잘 할 수 있는지에 대한 심도있는 논의에 대해서는, Jono Bacon의 책 *The Art of Community* (O'Reilly, 2012) 가 굉장히 유용할 것입니다.

Keep Up Communication

지속적인 소통

Talking with a community goes beyond public relations, which typically focus on press releases about major events. You want the community and the world at large to know about evolution in the project before, during, and after each step. Encourage your employees to blog and use social media in appropriate ways to get the news out. Consider a commitment to recruit an informative post at least once every two weeks from someone in the community (and even more often for large projects) along with regular tweets. A small investment in branding, such as stickers that community members can put on their laptops, or socks and t-shirts, shows pride in the project and gets the name where it is seen by the people you want. Such practices attract new users and remind existing community members that you have a vibrant project.

보통 주요 이벤트에 대한 보도에 집중하는 PR(public relation) 활동과 달리 커뮤니티와의 소통은 그 이상을 필요로 합니다. 회사라면 커뮤니티와 전 세계가 프로젝트 진행 단계의 이전, 과정 및 이후의 진화에 대해 알기를 원할 것입니다. 직원들이 블로그에 글을 게시하거나 소셜미디어 사용하여 그 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 노트북이나 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.

Adopt Metrics and Measurement

측정 기준을 선택하고 그에 따라 측정하기

We are a data-driven society. All organizations must learn how to collect useful metrics and educate employees on how to use the data when making decisions. Some metrics are easier to collect than others, so you need to determine what's really useful to you. Begin by collecting lots of metrics; then, over time, as you find out which ones are really useful, you can scale back. Typical metrics include the following:

모든 조직은 유용한 측정 기준(metrics)들을 수집하는 방법을 배우고 의사 결정시 데이터를 다루는 법을 직원에게 가르쳐야합니다. 일부 측정 기준은 다른 것들 보다 모으기 쉽기 때문에, 측정 기준을 고를 땐 실제로 사용자들에게 유용한 것이 무엇인지 결정해야 합니다. 따라서 먼저 많은 측정 기준들을 모은 뒤에 그 중 유용한 것들만 골라내며 수를 줄여야 합니다. 일반적인 측정 기준들은 다음과 같습니다:

- Numbers of contributors and contributions, and what people and places they come from
- Number of users, which you might be able to calculate roughly from statistics on downloads, mailing list participation, and other proxies
- Growth or shrinkage of the contributor mailing list
- Numbers of reported issues, bugs, and fixes
- Number of forks and stars on GitHub
- Page views of web pages, blogs, and tweets associated with your project

- 컨트리뷰터 인원과 기여 수

- 다운로드 수, 메일 목록 등의 접근 통계로 산출 가능한 사용자 수
- 컨트리뷰터의 증가 혹은 감소
- 제보된 이슈 및 버그와 해결 수
- GitHub 에서 포크 된 횟수 및 스타 개수
- 프로젝트의 웹사이트, 블로그, 트위터 등의 방문자 수

The CHAOSS Community] is defining metrics that are useful to collect across most projects.

CHAOSS 커뮤니티 에선 대부분의 프로젝트에 적용 가능한 측정 기준 들을 정의합니다.

Generally, you want to see the measures increase. Even an increase in reported bugs can be a good thing because it shows that the code is useful. The speed with which reported bugs are fixed can be a more important metric. If pull requests stagnate or go down, you need to think of ways to promote the project—or perhaps it's time to launch an effort to add new features that make the project more appealing.

보통 측정 결과가 더 높게 나오기를 원할 겁니다. 하지만 버그 제보가 늘어난다 하더라도 낙담할 필요는 없습니다. 이는 코드가 그만큼 유용하다는 뜻이며, 오히려 버그 해결 속도가 더 중요한 측정 기준이 될 수 있습니다. 만약 PR 이 들어오는 횟수가 점점 줄어든다면, 프로젝트를 홍보하거나 프로젝트를 더 매력적으로 만들어 줄 새 기능을 추가할 때입니다.

If most of your contributions are coming from a couple of organizations, you might need to encourage more diversity. There is a risk that your code will be optimized for one or two major users, losing value for other potential users. And, if a major contributor suddenly pulls out, you can be left without crucial support.

대부분의 기여가 소수의 조직에서만 올 경우 다양성을 장려해야 합니다. 코드가 한 두명의 주요 사용자에게 맞춰서 최적화 될 경우, 다른 잠재적 유저들을 잃을 수 있는데다가 그 주요 기여자가 갑자기 그만두게 되면 중요한 지원이 끊어지게 됩니다.

Different metrics are useful in different circumstances. For some projects, it's all right for pull requests and community participation to stabilize. Perhaps your project has a narrow application but is very useful for the people who need it.

측정 기준은 상황에 따라 달라질 수 있습니다. 예를 들어, 특정 사람들을 위한 특수 목적을 갖는 몇몇 프로젝트는 커뮤니티 참여와 PR이 안정되는 편이 더 좋습니다.

Your measurements can become part of a continuous improvement process. Make them available to managers through dashboards and encourage managers to pull them up at meetings and during the process of prioritizing future work. As with nearly all software, good open source tools exist for dashboards and visualizations displaying metrics. Bitergia offers open source dashboards, and Amazon has released an OSS attribution builder and OSS contribution tracker that their OSPO uses to manage its open source projects.

측정 결과는 지속되는 발전 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, 아마존은 자신들의 오픈소스 프로젝트 부서에서 프로젝트를 관리하는 데 사용하는 OSS attribution builder와 OSS contribution tracker를 출시했습니다.

Release the Project to an Independent Governance Organization

독립된 관리 조직에 프로젝트 릴리즈하기

Suppose that you have followed the advice of this book and the resources to which we've pointed you. Your project looks like a success and is being adopted by people outside your organization. When the project is big and stable enough, it's probably valuable to make it independent from your company. This will further encourage other organizations to support it, financially and otherwise. It will announce to the world that the project is sustainable and does not depend on your own management decisions for its future, which in turn will draw more people to use it and contribute to it. But because making an independent foundation is a lot of work, you should wait for clear evidence that it's important; for instance, requests from major contributors or the need to raise funds outside your own organization.

여러분이 이 책의 조언과 자료들을 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 지원하는 것을 도와줍니다. 또 이를 통해서 해당 프로젝트가 지속성 있고, 프로젝트의 미래에 대한 의사결정이 회사의 경영적 판단에 의존하지 않을 것을 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 이유가 생긴 후에 진행합니다. 예를 들면, 주요 기여자들의 요청이 있거나 외부의 조직에서 자금을 모아야 할때 말이죠.

Setting up a foundation is a complex task. A few major projects set up independent foundations, such as Linux, Mozilla, and OpenStack, but the vast majority of open source projects—even such popular tools as the Spark data processing tool —work under the auspices of an existing foundation. The Apache Software Foundation, Eclipse Foundation, and Linux Foundation sponsor wide varieties of software that extend beyond their original missions. Other organizations serve particular industries, such as HL7 for health care and Automotive Grade Linux for software in cars.

재단을 세운다는건 복잡한 일입니다. 물론 Linux, Mozilla, OpenStack 등의 몇몇 주요 프로젝트는 독립 재단을 세웠지만, 대부분의 오픈소스 프로젝트는-Spark 데이터 처리 툴 등의 유명한 툴도 포함해서-기존 재단의 후원 하에 있습니다. 아파치 소프트웨어 재단, 이클립스 재단, 리눅스 재단 등은 자신들의 원래 미션을 뛰어넘는 다양한 소프트웨어를 후원합니다. 그밖에도 헬스케어 분야의 HL7이나 자동차 분야의 Automotive Grade Linux 등, 특정 산업을 취급하는 조직들도 있습니다.

Open Source and the Cloud

오픈소스와 클라우드

The move to open source during the past decade or so has been paralleled by the adoption of cloud computing at many levels: infrastructure, data, and services. In fact, free and open source software is a major driver of the cloud, and service providers—many of them listed at the beginning of this book—are major crea‐ tors of open source software. Cloud providers rely heavily on open source soft‐ ware such as Linux, and virtualization software such as KVM and Xen. Customers running their software in the cloud choose open source software for the same reason.

지난 십여년동안 오픈소스로 이동은 인프라스트럭처, 데이터, 서비스등 다양한 수준에서 클라우드 컴퓨팅을 채택과 병행되었다. 실제로 무료 소프트웨어, 오픈소스 소프트웨어는 클라우드와 서비스 프로바이더의 주된 동력이었다. 서비스 프로바이더는 이 책의 앞 부분에 나열한 바 있으며, 이들은 오픈소스 소프트웨어의 주된 제작자이다. 클라우드 프로바이더는 Linux와 같은 오픈소스 소프트웨어, KVM과 Xen 같은 가상화 소프트웨어에 크게 의존 하고 있다. 고객들이 클라우드에서 소프트웨어를 선택할 때 오픈소스 소프트웨어를 사용하는 것도 같은 이유다.

The key advantage open source software offers both cloud providers and cloud users is its cost-free deployment. You can start up 10 instances of a virtual machine, expand quickly to 30 to meet peak needs, and then shrink back to 10 without trying to keep track of cost per seats, or adjust payments.

클라우드 프로바이더와 클라우드 사용자에게 오픈소스 소프트웨어가 제공하는 주된 잇점은 배포를 무료로 하는 것이다. 시작할 때는 10개의 가상 머신에서 인스턴스를 사용하고, 피크 때는 감당할 수 있도록 인스턴스를 30개로 빠르게 늘릴 수 있다. 그리고 다시 10개로 줄여서 비용을 절약할 수 있으며, 이때 시트 라이센스나 비용을 조정하지 않아도 된다.

Open source software has become a *lingua franca*, widely known and deeply understood by experienced developers. This increases its appeal to cloud provid‐ ers and customers, because they understand the impacts of using each project. Also, basing a cloud business on well-tested, preexisting software allows you to spin up faster and add more enhancements. For instance, most cloud providers remain competitive by adding all the latest hot technologies such as deep learn‐ ing. The providers realize that making it simpler to operate open source tools has tremendous value to their customers. The most successful new projects are natu‐ rally great candidates for new services. The wealth of high-quality open source options in these areas allows rapid upgrades to services and quickly enhances their platforms for customers' development efforts.

오픈소스 소프트웨어는 경험많은 개발자에게 널리 알려지고 깊이 이해되므로써, 링구아 프랑카(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 되었다. 이 점이 클라우드 프로바이더와 소비자에게 더 매력적으로 다가온다. 왜냐하면 이들은 각 프로젝트를 사용하는 것의 엄청난 파급효과를 이해하고 있기때문이다. 또한, 이미 존재하며, 잘 테스트된 소프트웨어를 기반으로 클라우드 사업을 하는 것은 보다 빨리 스핀 업하고, 더 많은 향상된 기능을 사용할 수 있다. 예를 들어, 대부분 클라우드 프로바이더는 딥 러닝 같은 최신 기술을 추가하여 창의적으로 자리매김한다. 프로바이더는 오픈소스 도구를 사용하여 운영을 단순화 하는 것이 고객들에게 엄청난 가치를 가져다 준다는 것을 알고 있다. 가장 성공적인 새 프로젝트는 자연스럽게 가장 훌륭한 서비스 후보가 된다. 이 영역에서 고품질의 오픈소스 옵션은 빠른 서비스 업그레이드를 가능하게 하고, 고객의 개발 노력을 위한 플랫폼을 빠르게 강화한다.

Customers also feel safer when cloud providers use open source tools. It reduces the risk of lock-in and allows customers to adopt hybrid solutions that run their applications on multiple cloud providers or using on-premises software as well as cloud providers.

고객 또한 프로바이더가 오픈소스 도구를 사용할 때, 보다 안전하다고 생각한다. 오픈소스는 락인(기존 시스템 대체비용이 어마어마하여 기술 전환을 하지 못하는 상태, 네이버 사전)의 위험을 줄인다. 또한, 오픈소스는 하이브리드 솔루션을 채택할 수 있게 해주는데, 이는 여러 클라우드 프로바이더 위에서 어플리케이션을 실행하는 클라우드 프로바이더를 사용할 뿐 아니라, 온 프레미스 소프트웨어를 사용할 수도 있게 해준다.

Because cloud providers own and manage their services, they are empowered to release tools and support software as open source and thus benefit from com‐ munities that form to improve the software.

클라우드 프로바이더는 자신의 서비스를 소유하고 관리하므로, 도구와 지원 소프트웨어를 오픈소스로 릴리즈 하도록 권한을 부여받는다. 이리하여, 소프트웨어를 향항시키는 커뮤니티 이점을 얻는다.

Conclusion

결어

The production and use of open source software has matured tremendously over the past decade. From informal collaboration, often around a “benevolent dicta‐ tor,” it has evolved into a discipline. Community managers, open source program offices, codes of conduct, and other facets of organized development practices are widespread. Websites have become more sophisticated, good communication practices and processes are codified on collaboration sites such as GitHub by groups like the TODO Group, and projects recognize the importance of that long-neglected cousin to source code: documentation.

지난 10 년 동안 오픈소스 소프트웨어의 생산과 활용은 엄청나게 성숙해졌습니다. 가끔은 어떤 "자비로운 독재자" 주변에서 이루어지던 형식이 없는 협력에서 하나의 규율로 발전했습니다. 커뮤니티 관리자, OSPO(오픈소스 프로그램 사무소), 행동 규범과 조직적인 개발 관행과 같은 다른 측면들도 널리 보급었습니다. 웹사이트는 더욱 정교해지고 좋은 의사소통 방식과 프로세스는 TODO 그룹과 같은 그룹에 의해 GitHub와 같은 공동 작업 사이트에 녹아들었으며, 프로젝트들은 정말 오랫동안 중요성이 인식되지 않았던, 소스 코드의 사촌격인, 문서의 중요성을 인식하고 있습니다.

This book described open source as it is conducted by the most advanced, tech‐ nology led companies and government agencies in 2018, with a look toward the future. We have consolidated resources and references to materials that will make your open source ventures successful and answered questions where you might have had concerns. Yes, there’s a lot to learn in the adoption, use, and release of open source code. Yet many companies are doing so successfully. They maximize the strategic value of adopting open source through culture change and by investing in support for developers and all employees engaged in these efforts.

이 책에서는 2018년에 가장 기술 주도적인 기업 및 정부 기관들에 의해 오픈소스활동이 미래지향적으로 진행되고 있다고 기술했습니다. 또 여러 정보와 참고 자료들을 종합하여 오픈소스를 추진하려는 시도를 성공할 수 있도록 하고, 우려되는 부분에 관한 질문에 답변했습니다. 그렇습니다. 오픈소스 코드를 채택하여 사용하고 릴리스 하려면 배워야 할 것이 많습니다. 그럼에도 불구하고 많은 기업들이 꽤 성공적으로 진행하고 있습니다. 그 기업들은 회사 내의 컬처를 바꾸고 이러한 노력에 참여하는 개발자와 모든 직원들을 지원함으로써 오픈소스를 채택의 전략적 가치를 극대화하고 있습니다.

Thousands of companies use open source software, and many contribute to it. This book showed you how to start your own journey with a pilot project, working with developers and other key stakeholders. Poll your developers to find out whether they are already using open source code. It’s important for the members of your organization to learn from their experiences and perhaps to involve them in a more formal policy regarding open source. The community will be a willing mentor as you embark on this journey.

수 많은 회사가 오픈소스 소프트웨어를 사용하고 있으며, 많은 회사들이 오픈소스 소프트웨어에 기여하고 있습니다. 이 책은 개발자 및 기타 주요 이해 관계자들과 협력하여 파일럿 프로젝트로 이 과정을 시작하는 방법을 설명했습니다. 개발자들을 대상으로 이미 오픈소스 코드를 사용하고 있는지 확인을 위한 조사를 해보십시오. 조직 구성원들이 그들의 경험을 통해 배우고 오픈소스 관련한 좀 더 공식적인 정책을 가지는 것이 중요합니다. 커뮤니티는 이런 활동을 시작하는 과정에서 기꺼이 멘토 역할을 해줄 것입니다.

Read some of the documents to which this book points and follow some basic good practices for checking the quality of the open source project. Document what you do and use your experience to determine your next steps.

이 책이 참고하고 있는 문서들 가운데 일부를 읽고 오픈소스 프로젝트의 품질을 확인하기 위한 몇 가지 기본적이고 모범적인 사례를 따르십시오. 진행 상황을 문서화하고 그 경험을 바탕으로 다음 단계를 결정하십시오.

With adjustments and revisions to your corporate practices, you can use open source libraries for such things as deep learning or web development. The big next step is incorporating open source software into your products. An even big‐ ger step is to open your own code and start a new open source project. This book offered an overview of the tasks facing organizations that undertake these efforts, along with pointers to more detailed sources of information.

기업 업무 관행에 대한 조정과 개정을 통해 딥러닝이나 웹 개발과 같은 영역의 오픈소스 라이브러리를 사용할 수 있습니다. 그 다음 단계는 오픈소스 소프트웨어를 제품에 통합하는 것입니다. 그리고 그 다음의 큰 단계는 우리 코드를 오픈하고 새로운 프로젝트를 오픈소스로 시작하는 것입니다. 이 책은 더 구체적인 정보를 얻을 수 있는 자료와 함께 이러한 노력을 수행하는 조직이 당면할 여러 일들에 관하여 대략 이야기했습니다.

Large corporations' embrace of open source demonstrates that it is a fixture of software development and becoming the new normal. We can see that software is changing markets and driving the value of all sorts of organizations, ranging from governments to financial services, and including traditional fields such as agriculture and construction. Each organization chooses a balance between building its own software, purchasing closed-source products or services, and consuming or creating open source.

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 포함한 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.

Startups also recognize the power of community. For them, the open source pro‐ cesses are a multiplier for their limited, precious resource of developer time.

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

Similar principles apply in other areas of creative production (although each type differs in the details): open source hardware such as Arduino, artwork released under some Creative Commons licenses, data provided under an open license, open standards (such as the Open Standards principles defined by the UK government, information provided through an open license by governments, and so on.

이러한 원리는 세부적인 내용이 조금 다른 형태이기는 하지만 Arduino와 같은 오픈소스 하드웨어, Creative Commons 라이선스로 공개된 예술 작품들, 오픈 라이선스로 제공되는 데이터, 영국 정부가 정의한 공개 표준 원칙과 같은 공개 표준, 정부가 공개 라이선스를 통해 제공하는 정보 등에도 적용됩니다.

Thus, open source software provides value across many fields. It's time to incorporate the best of open source tools and methods into your company strategy and culture, which will increase your competitive advantage.

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

Furthermore, adopting open source practices—including InnerSource for soft‐ ware you don't want to open to the world—can make your organization more productive, your innovation faster-moving, your employees happier, and your decision-making more efficient. These are the extra gifts of open source you will come to appreciate during your journey.

또한 외부에는 오픈하고 싶지 않은 소프트웨어의 내부 개발 과정을 포함하여, 오픈소스 관행을 채택하면 조직의 생산성을 높이고 혁신을 보다 빠르게 수행하며 직원들을 더 행복하게 만들고 의사 결정을 보다 효율적으로 수행할 수 있습니다. 이런 장점들은 오픈소스로 나아가는 여러분들을 위해 준비한 사은품 같은 것입니다.

About the Authors

저자 소개

Andy Oram is an editor at O’Reilly Media, a highly respected book publisher and technology information provider. His work for O’Reilly includes the influential 2001 title *Peer-to-Peer*, the ground-breaking 2005 book *Running Linux*, and the 2007 bestseller *Beautiful Code*.

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

Zaheda Bhorat is the head of open source strategy at AWS. A computer scientist, Zaheda is a long-time active contributor to open source and open standards communities. Previously, she shaped the first-ever open source program office at Google; launched successful programs, including Google Summer of Code; and represented Google on many industry standards executive boards across multiple technologies. She also served as a senior technology advisor for the Office of the CTO at the UK Government Digital Service, where she co-led the open standards policy, which is in use by the UK government on open document formats.

Zaheda Bhorat는 AWS의 오픈소스 전략 책임자이다. Zaheda는 컴퓨터과학자로서 오랫동안 오픈소스 및 오픈스탠다드 커뮤니티에 적극적으로 기여했다. 구글에서 그녀는 최초로 오픈소스 프로그램 부서를 설립했으며, Google Summer of Code(역자 주, 구글의 오픈소스 여름 인턴 프로그램)를 포함한 성공적인 프로그램들을 만들였다. 또 다양한 기술에 대한 업계 표준(industry standard) 집행위원회에서 구글을 대표했다. 그녀는 또한 영국의 정부 디지털 서비스의 CTO실을 위한 선임 기술고문으로서 영국 정부가 ODF(Open Document Format)를 사용하도록 하는 오픈스탠다드 정책을 주도했다.

Zaheda was responsible for OpenOffice.org, and later NetBeans.org, at Sun Microsystems, where she built a thriving global volunteer community and deliv‐ ered the first user version, OpenOffice 1.0. Zaheda is passionate about technology, education, open source, and the positive impact of collaboration for social good. She serves on the UK Government’s Open Standards Board, which determines the standards government should adopt. She also serves on the board of directors of the Mifos Initiative, an open source effort that is positioning financial institutions to become digitally connected providers of financial services to the poor. Zaheda speaks internationally on topics related to open source and social good. You can find her on Twitter *@zahedab*.

Zaheda는 선마이크로시스템즈(Sun Microsystems)의 OpenOffice.org, 이후에는 NetBeans.org를 맡아 열정적인 글로벌 자원봉사 커뮤니티를 구축하고, 최초의 사용자 버전인 OpenOffice 1.0를 출시하였다. Zaheda는 기술, 교육, 오픈소스 및 공익을 위한 협업의 긍정적 영향에 대하여 열정적이다. 그녀는 영국 정부가 채택해야하는 표준을 결정하는 오픈스탠다드 위원회의 일원이다. 또한. Mifos Initiative의 이사회 이사로 재직하고 있다. 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 있다. 그녀의 트위터 아이디는 *@zahedab*이다.

역자 소개

(가나다순) '이 민석' 예시를 참조하여 각자 소개를 쓰세요

- **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** :
- **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.

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

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

- 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

- 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.