Starting an Open Source Project

オープンソースプロジェクトを立ち上げる

Once a company has participated in open source communities long enough to build a reputation, it’s in a position to launch its own open source projects. It’s at this stage of open source participation that companies can realize the greatest benefits from open collaboration. You can open source proprietary projects that could be of use to the community. Another common avenue is to create new open source projects from scratch and benefit from collaboration among external developers at the outset.

企業は、長期に渡ってオープンソースコミュニティに参加し、良い評価を確立すると、やがてその企業独自のオープンソースプロジェクトを立ち上げることが可能な立場になります。オープンな協業から得られる大きなメリットを企業が実感できるのはオープンソースへの参加がこの段階に達した時です。企業は、コミュニティで活用される可能性のあるプロプライエタリなプロジェクトをオープンソース化することができます。もうひとつのやり方は、新規のオープンソースプロジェクトをスクラッチから作り上げ、最初から外部開発者の協業のメリットを得るやり方です。

This guide was created to help enterprises already well versed in open source learn what they need to know to start their own open source projects. We’ll take you through the process, from deciding on what to open source, to budget and legal considerations, and more. The road to creating an open source project may be foreign, but major enterprises including Google, IBM, Facebook, Twitter, and Microsoft have blazed the trail for you. Follow this guide for topical and helpful advice and you will be on your way.

本ガイドの目的は、すでにオープンソースに関して十分に経験を積んだ企業の方々を対象に、企業独自のオープンソースプロジェクトを立ち上げるために何を知る必要があるのかを学ぶ手助けとなることです。本ガイドでは、何をオープンソース化するかを決定することから始めて、予算や法的な考慮事項に至るまで、すべてのプロセスを案内します。オープンソースプロジェクトを作り上げることは、かつて経験したことのないことかもしれませんが、Google、IBM、Facebook、Twitter、Microsoftのような大企業は、後に続く人々のために道を開いてくれています。本ガイドの的確で有益なアドバイスを参考とし、あなたの会社の道を進んで下さい。

Contents

目次

Why create an open source project?

なぜオープンソースプロジェクトを作るのか

When to create an open source project

いつオープンソースプロジェクトを作るか

Where to start

どこから始めるか

Planning

プロジェクトの計画作り

Launch

オープンソースプロジェクトの始動

Final words

結語

Launch checklist

オープンソースプロジェクト始動のチェックリスト

View All Guides »

ガイド一覧

Contribute on GitHub »

GitHub上で貢献

Contributors to this Guide

本ガイドの貢献者

Christine Abernathy

Open Source Developer Advocate at Facebook

オープンソースチーム開発者アドボケート、Facebook

Ibrahim Haddad

VP of R&D and Head of the Open Source Group

Samsung Research America

R&D部門VP兼オープンソースグループ首席、Samsung Research America

Guy Martin

Director, Open at Autodesk

Autodesk

Open部門ディレクター、Autodesk

John Mertic

Director of Program Management at The Linux Foundation

プログラム管理ディレクター、Linux Foundation

Jared Smith

Open Source Community Manager, Capital One

オープンソースコミュニティマネジャー、Capital One

Section 1

第一章

Why create an open source project?

なぜオープンソースプロジェクトを作るのか

There are plenty of reasons for your company to start an open source project. You might aim to innovate faster, achieve quicker time to market, collect new ideas, enable interoperability or de facto standards, recruit talented developers, and gather diverse viewpoints and contributions to produce better code and better products.

企業にはオープンソースプロジェクトを立ち上げるたくさんの理由があります。すなわち、敏速な技術革新を狙ったり、早期市場参入を成し遂げたり、新たなアイデアを収集したり、相互運用性や業界標準を実装したり、才能ある開発者を採用したり、より優れたソフトウェアや製品を開発するために広汎な知見と貢献を集めたりできます。

These benefits can all be realized by using and contributing to open source projects created and managed outside your company. But a comprehensive open source strategy often includes creating and launching your own open source projects as well.

このようなメリットはすべて、企業の外で作られ、運営されるオープンソースプロジェクトを活用したり、貢献したりすることによって実現されます。しかし、広い意味でのオープンソース戦略は、企業が自身でオープンソースプロジェクトを作り、立ち上げることも含む場合が多くあります。

Initiating projects or releasing existing projects as open source to the community strengthens the sense of give and take, which can further build a company’s reputation in open source, and make the company even more attractive to open source developers and more influential in the open source projects where it contributes. Having your codebase as a starting point for discussion can bring other advantages as you look to engage an external ecosystem of partners, vendors, and users.

プロジェクトを始めること、あるいは、既存プロジェクトをオープンソースとしてコミュニティに公開することは、「ギブ・アンド・テーク」への意識を高め、そのことがオープンソースコミュニティにおける企業の良好な評価をもたらし、企業がオープンソース開発者に魅力的に映り、また、企業の貢献するオープンソースプロジェクトに大きな影響力を持つことに繋がります。また、企業の持つコードベースを議論の開始点として公開することで、外部のパートナー、ベンダー、あるいは、ユーザーのエコシステムとの関係が強化され、さらなる利益を生み出すことになります。

By opening up your own code and development practices to outside use and contribution, you are truly embracing open innovation and using open source to its best effect for your business. Code released under an open source license allows anyone to contribute, inspect, modify, and improve it. This collaborative approach to development is now the de facto way of building software and a proven engine of technological innovation.

企業のコードと開発手法を外部の利用と貢献に向けて公開することにより、企業は本当にオープンイノベーションを取り入れ、企業のビジネスに対する最大限の効果とみなされるまでオープンソースを活用したことになります。オープンソースライセンスの下で公開されたコードは、誰でもそのコードに貢献したり、調査したり、変更したり、改善したりすることを許容します。このような開発の協業アプローチは、今やソフトウェア開発の実質的な標準となり、技術革新の推進力となっていると広く認識されています。

This is true whether your primary mission is financial services, providing health care, operating fleets of trucks, selling retail goods in stores or online, providing transportation to commuters and airline passengers, building roads and bridges, or thousands of other specialties. While these enterprises certainly want to keep the applications and technologies that are core to the value they provide to their customers, there exists a myriad of code and software that are dependencies and which are not a high-value differentiator to the organization. Opening these technologies up as a project to outside contributions can create new possibilities and opportunities for growing and strengthening this code.

この状況は、企業の活動分野が、金融サービスであろうと、ヘルスケアの提供であろうと、トラック輸送の運用であろうと、店舗やオンラインでの小売りであろうと、通勤者や航空機搭乗者むけの移動手段の提供であろうと、道路や橋梁の建設であろうと、あるいは、その他何千にも及ぶ専門分野であろうと当てはまります。このような企業でも顧客に提供する価値の中核となるアプリケーションやテクノロジーは社内に保持することを望むでしょう。しかし、一方で、必要性はあるものの、企業にとって差別化要因とするほどの高い価値は認められないような無数のコードやソフトウェアも存在します。このようなテクノロジーを外部の貢献者に向けての公開することによって、当該コードの成長と強化の新たな可能性と機会が作りだされます。

“No matter how many smart people we hire inside the company, there’s always smarter people on the outside. We find it is worth it to us to open source and share our code with the outside world in exchange for getting some great advice from people on the outside who have expertise and are willing to share back with us.”

「どんなにたくさんの優れた技術者を会社が雇用しようと、会社外にはもっと優れた技術者がいつも存在します。専門性があり、しかも喜んで私たちと一緒にコードの改善に取り組んでくれる外部の開発者の素晴らしいアドバイスと引き換えに、ソースを公開し、外部世界とコードを共有することには、私たちに大きな価値があることが分かりました。」

Jared Smith – Open Source Community Manager, Capital One

[Jared Smith](https://twitter.com/jsmith) – オープンソースコミュニティマネジャー、Capital One

Companies look to open source when they want to move their agendas forward in areas where they might not have the needed talent on staff. By moving to open source, they can often speed up their efforts and work with others who are working toward the same software goals, while also cutting costs and improving their end products.

企業は、社内で必要な人材を供給できないような領域で何らかの計画を前に進めようとするときに、オープンソースに期待します。オープンソースに移行することにより、開発スピードを上げ、同じ目標を目指すソフトウェアの開発に向けて活動する人々と協調することができ、同時にコストを削減し、出来上がり製品の価値を高めることができます。

Open source projects provide freedom to collaborate — even among competitors in the same industry — and can accelerate development by placing many eyes on the code in progress. By working together, developers can share openly, gain plenty of feedback, and together build code that is scalable, efficient, and high quality.

オープンソースプロジェクトは、同じ業界の競合企業との間においてさえ、協調の自由をもたらします。また、開発中のコードに多くの人々の着眼を得て開発を加速させることができます。一緒に働くことにより、開発者は、オープンに情報を共有し、多くのフィードバックを受け、そして、拡張性、効率、品質において優れたコードを構築できます。

5 Reasons to Start a New Open Source Project as an Enterprise

　企業が新規にオープンソースプロジェクトを立ち上げる5つの理由

Accelerate an open solution; provide a reference implementation to a standard; share development costs for strategic functions

　　オープンソリューションの加速；標準となるリファレンスをインプリメントして提供。戦略的機能に対する開発コストの共有。

Commoditize a market; reduce prices of non-strategic software components.

　　市場をコモディティ化；非戦略的なソフトウェアコンポーネントの価格を低減。

Drive demand by building an ecosystem for your products.

　　製品のエコシステムを築き需要を喚起。

Partner with others; engage customers; strengthen relationships with common goals.

　　他社との間でパートナーの関係を形成；顧客を取り込む。共通のゴールに向けて関係性を強化。

Offer your customers the ability to self-support: the ability to adapt your code without waiting for you.

　　顧客に自己サポートの能力を提供；企業の提示するコードを企業のサポートなしで利用。

Source: Ibrahim Haddad

出展：[Ibrahim Haddad](http://www.ibrahimatlinux.com/)

Section 2

第二章

When to create an open source project

いつオープンソースプロジェクトを作るか

The decision to release or create a new open source project depends on your circumstances. Your company should first achieve a certain level of open source mastery by using open source software and contributing to existing projects. This is because consuming can teach you how to leverage external projects and developers to build your products. And participation can bring more fluency in the conventions and culture of open source communities. (See our guides on Using Open Source Code and Participating in Open Source Communities) But once you have achieved open source fluency, the best time to start launching your own open source projects is simply “early” and “often.”

コードを公開したり、新規オープンソースプロジェクトを作ったりすることの決定はその企業の置かれた状況に依存します。企業としては、オープンソースソフトウェアの利用や既存プロジェクトへの貢献を通じ、オープンソースに対して一定程度の習熟度に達しているべきです。このような活動によって、企業の製品を作り上げるのに、外部のプロジェクトや開発者をどのように活用できるのかを学ぶことができるからです。また、参加することによって、オープンソースコミュニティの習慣や文化に円滑に対応できるようになります。(本ガイド集の「オープンソースコードの利用」、および、「オープンソースコミュニティへの参加」を参照して下さい。) しかしながら、ひとたびオープンソースに対する円滑さを達成すれば、企業独自のオープンソースプロジェクトを立ち上げるべき時は、単純明快に「早期」、「頻繁」の考え方にに従うのがよいでしょう。

The Release Early, Release Often

早期に公開、頻繁に公開

Development Methodology

開発の方法論

Release Early

早期に公開

Other project participants can provide feedback and contribute to the development.

　　他のプロジェクト参加者は開発にフィードバックを提供し、貢献することが可能

New ideas can be incorporated while code is still flexible.

　　コードにまだ柔軟性が残っている間に新しいアイデアを組み込むことが可能

Problems can be flagged by other participants before getting too far.

　　進行し過ぎない段階で、問題点が他の参加者によって提起されるうる

Release Often

頻繁に公開

Smaller changes are easier to understand, debug, improve, and drive to maturity.

　　小さな修正の方が理解、デバッグ、改善、熟成への到達が容易

Maintain rapid pace of development, QA, and innovation.

　　開発、品質保証、技術革新の速いペースを維持

Maintain high level of visibility towards project members on the progress and current development status.

　　進捗や開発の現況についてプロジェクトメンバーに対して高いレベルの可視性を維持

Source: Ibrahim Haddad

出展：[Ibrahim Haddad](http://www.ibrahimatlinux.com/)

Section 3

第三章

Where to start

どこから始めるか

Perhaps the time to create a new project is when you realize that you have a tough technical problem which you can’t solve on your own. Another motivator is when you’re unable to find and join another project that does what you want it to do. Ultimately, there is no right answer to the question. You start one when you need one and can’t find an existing effort.

新しいプロジェクトを作るのは、おそらく、企業が自社だけでは解決できない困難な技術課題を認識した時でしょう。他の動機としては、企業の期待する活動を実施するプロジェクトを見つけ、参加することができないといった状況もあるでしょう。究極的には、設問に対する正しい答えはありません。企業が新しいプロジェクトを必要とし、既存の活動が見つからない時にプロジェクトを立ち上げるのがいいでしょう。

For enterprises considering a new open source project, you want to start by finding your unique answer to “why?” Begin by asking a lot of questions about what is important to your organization. It’s important to start an open source project for the right reasons.

新たなオープンソースプロジェクトを検討する企業としては、「なぜ？」への独自の答えを見出すことから始めるのがよいでしょう。企業にとって何が重要なのかについてたくさんの問いを発することから始めてください。正しい理由に基づいてオープンソースプロジェクトを立ち上げることが大切です。

“I think it is a crucial thing for a company to be thinking about what they’re hoping to achieve with a new open source project. They must think about the value of it to the community and developers out there and what outcomes they’re hoping to get out of it. And then they must understand all the pieces they must have in place to do this the right way, including legal, governance, infrastructure and a starting community. Those are the things I always stress the most when you’re putting an open source project out there.”

「企業にとって、新しいオープンソースプロジェクトで何を成し遂げたいかを常に考えることが非常に重要なことだと思います。コミュニティや開発者に対するプロジェクトの価値、および、プロジェクトから何が成果物として出てきて欲しいかについて考えなければなりません。その後で、これを正しく行うために備えるべきすべての要素、たとえば、法務、管理・統制、インフラ、起点となるコミュニティなどを理解しなければなりません。それらは、オープンソースプロジェクトを作ろうとするときに私が一番強調する点です。」

John Mertic – Director of Program Management at The Linux Foundation

[John Mertic](https://twitter.com/jmertic) – プログラム管理ディレクター、Linux Foundation

The place to start includes secondary coding projects where an enterprise does not need to be an authority, and where there may be a larger group of technologists around the world who can help you solve a problem. If it is not mission-critical code, then it is likely a good candidate to be open sourced. However, it should also be code that your company is still actively using and maintaining. Commercial dependencies on code enable a continuous feedback loop of bug fixes, patches and new features.

時として、起点となるのが企業として必ずしも権威者とは言えない副次的コーディングプロジェクトながら、世界には当該企業の問題解決を手助けする技術者の大きなグループが存在するようなケースもあります。そのコードがミッションクリティカルな業務に供されるものでなければ、オープンソース化の候補となる可能性があります。そのようなコードが企業内で活発に利用され保守されていることもありえるでしょう。コードがビジネスで利用されているということにより、継続的なバグ修正、パッチ、さらには、新機能のフィードバックが可能となります。

“Many of the projects we share are the projects we are using internally, so we try to share what we actually use in production. That means because of Facebook’s scale, it’s battle-tested at scale; it’s professional grade work we’re contributing to the community. The other key thing is that because we are using these tools, we’re not going to let them languish and be unsupported because our engineers need them to be robust as well.”

「当社が共有に供する多くのプロジェクトは、当社の社内で利用しているものです。私たちは、当社の日常業務で実際に利用しているものを共有することを目指しています。それは、Facebookのシステムの規模から考えて、実戦試験ずみであることを意味します。当社がコミュニティに貢献しているものは、プロ級のソフトウェアです。もう一つのポイントとして、これらのツールは日常業務で使用されているので、当社はこれらを劣化した状況や保守されない状況に置くことはあり得ません。当社の技術者もこれらが良い状態であることを必要としているからです。」

Christine Abernathy – Open Source Team Developer Advocate at Facebook

[Christine Abernathy](https://twitter.com/abernathyca) – オープンソースチーム開発者アドボケート、Facebook

Another issue to consider is whether your project is unique or if others are already working on similar code because they have similar problems. Is the potential open source project something that’s important for your company to offer and manage as a project and will other users seek it as well? If so, then perhaps the idea makes a lot of sense.

もう一つの考慮点は、そのプロジェクトがユニークなものなのか、あるいは、同様の問題に対応するために同様のコードを作ろうと既に活動している人々がいるかどうかです。オープンソースプロジェクトとすべく検討しているものは、プロジェクトとして提案し、管理して行くに値するほど当該企業にとって重要なものでしょうか？しかも、他のユーザーもそれを必要としているでしょうか？もしもそうであれば、おそらくそのアイデアには十分に意味があります。

You will also need to decide whether you want to donate your code to a vendor-neutral, non-profit organization or retain some control by owning and running your own project. Again, the answer depends on what you are trying to achieve.

企業は、ベンダー中立な非営利組織にその企業のコードを寄贈するのか、あるいは、企業独自のプロジェクトを所有・運営して何らかの制御権を維持するかどうかを決定する必要が出て来ます。ここでも、答えはその企業が何を達成しようとしているかに依存します。

“Do an open source project because you recognize there are things within a potential project that are not strictly important to your company, but which you want to maintain some control. Do it when you know that by getting other developers involved with it as an open source project that it helps your ship rise as well. So, think of something to pursue and do it.”

「オープンソースプロジェクトにするが適切なのは、検討中プロジェクトに企業としてそんなに重要じゃないけれども、何らかの制御は維持したいと思うような要素が認められる時です。また、オープンソースプロジェクトに他の開発者を巻き込むことによって、プロジェクトが浮上する可能性が広がると思ったときも、オープンソースプロジェクトを始めて下さい。つまり、何か追求したいものを考えてから、オープンソースプロジェクトを始めてください。」

John Mertic – Director of Program Management at The Linux Foundation

[John Mertic](https://twitter.com/jmertic) – プログラム管理ディレクター、Linux Foundation

Questions to Ask Before Starting an Open Source Project

オープンソースプロジェクトを立ち上げる前に問うべき質問

Can we financially sponsor the project? Do we have an internal executive champion?

プロジェクトを財政的にサポートできますか？社内に経営幹部レベルをサポート、牽引してくれる人がいますか？

Is it possible to join efforts with an existing open source project?

既存のオープンソースプロジェクトと一緒に活動することは可能ですか？

Can we launch and maintain the project using the OSS model?

OSSモデルによってプロジェクトを立ち上げ、維持することはできますか？

What constitutes success?

How do we measure it?

何をもって成功とみなしますか？

どんな方法で評価しますか？

Will the project be able to attract outside enterprise participation (from the start)?

プロジェクトは、外部の企業の参加を引き寄せることが可能ですか(立ち上げの当初から)？

Is there enough external interest to form and grow a developer community?

開発者のコミュニティを形成し、拡大してゆくに十分な外部の関心はありますか？

Source: Ibrahim Haddad

出典：[Ibrahim Haddad](https://twitter.com/ibrahimatlinux)

Section 4

第四章

Planning the project

プロジェクトの計画作り

Once you’ve set your plans in motion, there are a myriad of details which must be considered and resolved to bring your open source project to fruition. Let’s take the considerations step by step, starting with how you decide whether to release or donate the project’s source code.

計画を実行に移し始めると、オープンソースプロジェクトから成果を得るために、検討し、解決しなければならないことが無数に出てきます。プロジェクトのソースコードを公開、ないしは、寄贈するかどうかをいかに決定するかという問題から始めて、ひとつひとつ考えてみましょう。

Decide what code to release or donate

どんなコードを公開、ないしは、寄贈するかの決定

First, you must decide if your company wants to create or release the code while maintaining ownership of it and the project, or if you want to donate the code to others to maintain and administer the project. If the code already exists, then there is the related issue of whether you will release all the code in a project or just some of it as an open source project.

最初に、コードやプロジェクトに関する所有権を維持した状態で、コードの作成や公開を行おうと考えるのか、あるいは、他者に寄贈し、プロジェクトの保守と管理を任せたいと考えるのかを決定しなければなりません。もしもコードが既に存在している時には、関連した課題として、オープンソースプロジェクトに供するのは、既存プロジェクトのすべてのコードなのか、その一部のみとするのかを決定しなければなりません。

To make these decisions, consider stepping back to determine the objectives you have in mind for the code.

これらの決定を行うに当たり、コードに託した目的を見定めるべく立ち戻ることを検討してください。

“When our engineers decide they want to open source a project, we look at several things. First, we want to make sure, is the project going to be useful to developers outside the company? And is the project going to be something transformative? Is it something we can showcase? And is there going to be a community built around this that the person maintaining it will be able to support?”

「私たちの技術者がプロジェクトをオープンソース化したいということを決定したら、いくつかのことを考えます。まず、プロジェクトが社外の開発者にとって有用なものになるかどうかを確認します。そして、プロジェクトは斬新なものか？ショーケースにできるようなものか？また、プロジェクトの周りにコミュニティが形成され、それをプロジェクトの面倒を見る人がきちんとサポートできるかどうか？」

Christine Abernathy – Open Source Team Developer Advocate at Facebook

[Christine Abernathy](https://twitter.com/abernathyca) – オープンソースチーム開発者アドボケート、Facebook

For example, you might want to attract fresh insights from other developers on a part of an application that isn’t core to your work. Or you might seek additional real-life algorithms to detect logs in a system monitoring application. Rather than releasing the whole product as open source, you could release only the code related to the algorithms. This allows you to gain the contributions of others and help others who are needing similar help, without compromising your core business.

たとえば、自分の仕事の中核ではないアプリケーションの一部について、他の開発者から新鮮な見識を得たいと考えることがあるでしょう。あるいは、システム監視アプリケーションの中で、ログを検出する現実的なアルゴリズムを追加したいと考えるかもしれません。この際、全ソフトウェアをオープンソースとして公開するのでなく、アルゴリズムに関連したコードだけを公開するというやり方もあります。こうすれば、中核ビジネスを危うくすることなく、他の人々の貢献を受け、なおかつ、同様の手助けを必要としている人々を助けることもできます。

Starting a project and maintaining overall control lets you have oversight and gives you the ability to help shape it into what you need, while still giving freedom and control to the contributing developers to do their work.

プロジェクトを立ち上げ、全体の制御権を維持すれば、プロジェクトを高い場所から見守り、必要な形に持って行くのに必要な影響力を持つことができます。それでも、貢献する開発者には彼らの活動の自由と自己制御を保証します。

Contributing your code is different. It means giving it up and handing off some control to others to maintain and administer. It may be code that is no longer needed by your company, but it still has value to others because it fills important niches for users. Such code, which your company simply may not have time for anymore, can be welcomed and furthered by open source communities and can lead to thriving long-term projects. It may also be critical code for your company, but a project needs a neutral home in order to attract more participants and grow a broader ecosystem.

企業が持っているコードを貢献するのは少し様子が異なります。それは、コードを放棄し、保守や管理に関する制御権を他者に引き渡すことを意味しています。企業にとってはもはや必要でないコードかもしれませんが、ユーザーの特定分野のニーズを満たしているので、他の人々には価値があるのです。企業としては単に時間をかけられないというようなコードでも、オープンソースコミュニティから歓迎され、前進の力を与えられ、長期的なプロジェクトとして成長する可能性があります。コードは企業にとって重要なものである可能性もありまありますが、より多くの参加者を引き寄せ、より大きなエコシステムへと成長するために、プロジェクトは中立的な受け皿を必要としています。

Do not just contribute code that is no longer useful or of interest to you and expect a community will keep it up to date – that’s not what we mean. On the contrary, never use open source communities or projects to dump old code to see if it gains traction. If it’s not really an important project, you’re going to lose credibility in the world of open source and when you later try to open source other code, developers won’t be interested. They’ll remember that you wasted their time in the past, which is not something you want to have happen.

企業にとってもはや有用性がなくなった、あるいは、関心がなくなったというようなコードを貢献し、コミュニティが最新の環境に適合させることを期待してはいけません。私たちはそのようなことを勧めているのではありません。オープンソースコミュニティやプロジェクトを古いコードの捨て場として使い、弾みがつくかどうか試すようなことは決してしないでください。それがまったく重要なプロジェクトでないということになると、オープンソースの世界で信用を失い、その後、他のコードをオープンソース化しようとしても、開発者が関心を示さなくなります。過去にその企業が時間の浪費を強いたことは記憶に残るので、企業としてそのようなことを行ってはいけません。

“If you create three open source projects this year and they’re all really good and they’ve attracted a good community and there’s lots of value in them, that’s much more important than creating 10 open source projects a year. Honestly, the open source community values quality over quantity and they will self-select which projects to join. If you create 10 that are horrible, you’re not going to create any traction. You want to open source good stuff.”

「もしもあなたの会社が今年、3件のオープンソースプロジェクトを立ち上げ、それらのどれもが素晴らしいもので、良質なコミュニティを惹きつけたとすると、1年に10件のオープンソースプロジェクトを立ち上げるのよりも大きな値打ちがあります。オープンソースコミュニティは、量よりも質の方を高く評価し、どのプロジェクトに参加するのか自ら選択します。10件のプロジェクトが低質なものであれば、どれも牽引力を得ることはできないでしょう。よいものだけをオープンソース化するように考えるべきです。」

Guy Martin – Director of Open at Autodesk

[Guy Martin](https://twitter.com/guyma) – Open部門ディレクター、Autodesk

Build a business case

ビジネスケースを作る

It’s also a good time to start an open source project after you have developed a sound business case that’s bolstered with outcomes that can be achieved, like any other product you’re bringing to market. Then there is the needed executive buy-in as well, because management needs to understand why it is being undertaken, what the goals and budgets are, what the roadmap is going to be, what intellectual property will be opened up, and what code will and will not be involved.

オープンソースプロジェクトの立ち上げは、市場に製品を送りだすときと同じように、達成の見込みのある成果によって裏付けられた健全なビジネスケースを作った後で行うのがいいでしょう。そうすれば、経営幹部の了承も得られるでしょう。なぜなら、経営幹部としては、なぜプロジェクトが実施されるのか、ゴールは何か、予算はどの程度か、ロードマップはどうなるのか、どんな知財が開示されるのか、どのコードが関係しており、また、関係してないのかを理解する必要があるからです。

Allocate resources

資源の割り当て

You’ll also need to decide whether you can make the necessary resource commitments, including developer time to be contributed to the project. Developer time will likely be similar initially to the amount of time they spend on internal code efforts. You’ll also need to consider what time, materials or help your developers will need to provide to help others in the new community get up to speed on the codebase. There will also be resources needed for the legal team that will be involved in creating an open source project that could involve competitors. And marketing investments will ensure that the project gains support and contributors after launch.

また、プロジェクトに貢献する開発者の労働時間のような、資源提供のコミットメントを行うことができるかどうかも決定する必要があります。開発者の労働時間は、社内のコード開発の時に費やしていた時間と当初は同等になるでしょう。また、新しいコミュニティで他の人々がコードベースを素早く習得する手助けをするために、当該企業の開発者が、どれくらいの時間、どんな資材、どんな援助を提供する必要があるかを検討しなければなりません。また、競合他社を巻き込む可能性のあるオープンソースプロジェクトを作る際には、法務部門のチームの資源も必要とされます。さらに、マーケッティング投資を行うことによって、プロジェクトの立ち上げ後、プロジェクトへのサポートと貢献者獲得が確かなものとなります。

You’ll also have to set budgets for the infrastructure used to begin and maintain the project. This includes a project hosting and source control website like GitHub, where the code will reside and be maintained, as well as bug resolution, and other needed tools.

また、プロジェクトの開設と維持に要するインフラのための予算も準備しなければなりません。例えば、プロジェクトのサポートとGitHubのようなソース管理のためのWebサイトが必要とされ、そこに、コードが格納され、そこでバグ解決などの保守作業なされ、必要なツールも用意されます。

Test the code quality

コードの品質

The readiness and maturity of the code you are considering for use in your open source project can also be an indicator that you are ready to start your planning. You want to ensure that the code is in good shape, and as we mentioned earlier, not junk code that will lead to a credibility disaster with the open source community.

オープンソースプロジェクトで使用されることを想定しているコードの完成度・成熟度は、計画の実行開始に向けた準備の指標ともなりえます。当然ながら、コードは良い状態にあることが望まれます。以前にも述べたように、ガラクタのようなコードは、オープンソースコミュニティにおける信用崩壊に繋がります。

At the same time, the trap you must be careful of is that perfect code isn’t required. If you think that the code has to be perfect, you’ll likely never get it started. Go with the best you’ve got and know others will be helping to make it better. You do also want to be sure that the code you start with doesn’t include code comments with trade secrets, references to your proprietary interfaces or profanities and other issues as you ensure it is mature enough to get out there into the community.

同時に、注意しなければならないのは完璧なコードが要求されているわけでもないという点です。もしもコードが完璧でなければならないと考えていると、プロジェクトは立ち上がりません。出来上がりで最善のものを提供し、他の人々が改良に向けて支援してくれるものと考えるのが良いでしょう。また、コミュニティに提供できるレベルの完成度を確かめる時は、最初に提供するコードには、コードコメントに企業秘密や、企業のプロプライエタリインターフェースへの参照、あるいは、不快な言葉その他の問題を含んでいないことを確かめるべきです。

Make sure it’s useful

有用性を確認する

Your project is also ready to move forward when you can see and demonstrate that it will be useful to others as well who are seeking such answers to their IT problems, which can be gleaned through traditional market analysis. You want to be sure that it’s something that others will seek out and be willing to contribute to so it can be a successful project. Do some research and ask around. Attend open source events and talk to developers and presenters about their problems and projects.

そのプロジェクトが前に進むべく準備万端となるのは、IT課題に対して同様の答えを追求する他の人々にとってもそのプロジェクトが有用であることが明らかとなり、それを証明できた時です。そのような状況は、伝統的な市場分析によっても情報収集できます。そのプロジェクトは他の人々が探し求めるものであり、喜んで貢献したいと思うものであること、それゆえにプロジェクトが成功する可能性が高いことを確かめるのが良いでしょう。何らかの調査を行い、また、周りの人々に聞いて回ってください。また、オープンソースイベントに参加し、開発者や講演者と会話し、それらの人々の課題やプロジェクトについて聴いてください。

If you find that others have already started a similar project to solve a similar problem, then you can consider whether you want to join that effort rather than duplicate it. If a similar project already exists, it’s more powerful to team up, even if your competitor is driving it because collaboration is a critical part of open source communities.

もしも他の人々が既に同じ問題を解決するために同様のプロジェクトを立ち上げていることが分かった時には、重複するよりも、その活動に参加することを検討してみるのが良いでしょう。同様のプロジェクトが存在しているなら、競合企業が推進していたとしても、チームになる方がより強力になります。なぜなら、オープンソースコミュニティでは、協調することが非常に重要な要因だからです。

Working with competitors on open source projects is important to consider. If your company starts an open source project and draws in competitors to get involved, you can build collaboration and goodwill, all for the greater good of the code, while leading instead of following.

オープンソースプロジェクト上で競合企業と一緒に活動することは、検討すべき重要なことがらです。企業が独自のオープンソースプロジェクトを立ち上げ、競合企業を巻き込むことができれば、協調、友好を築くことができ、より優れたコードの完成に繋がります。しかも、追随するのではなく、先導できるのです。

Consult your team

意見を求める

Through every consideration we’ve listed, the technical team can work with the executive team to make these decisions and help guide the process to make it successful. Your developers and IT staff can illustrate where and when it becomes useful to collaborate.

上記のすべての考慮点について、技術チームは、これらの決定や、プロジェクトの成功に必要なプロセスの方向付けを手助けするのに、経営幹部のチームの意見を求めることができます。企業内の開発者とITスタッフは、どこで、いつ協調すれば有効かを示すことができます。

“We’ll start an open source project if we can’t find something we are looking for, or if the thing that we found that has worked in the past doesn’t work for us as we’re moving forward. Sometimes that’s for performance reasons. Sometimes that’s simply cost reasons or vendor lock-in reasons. Sometimes it’s simply we’re moving a bunch of our infrastructure over to more modern technologies and some of the legacy vendors we’ve used in the past just aren’t prepared or aren’t willing to run their software in a cloud or containerized environment.”

「私たちがオープンソースプロジェクトを立ち上げるのは、私たちが探し求めていたものが見つからなかったときや、過去にうまく動いていたものが、私たちの環境の進化のために、動かなくなったということが分かったときです。それは性能的な理由であることもあります。また、単にコストが理由であったり、ベンダーロックインが理由であったりします。私たちが、単純に、一団のインフラをより最新のテクノロジーに移行させたのに、過去に利用していたレガシーベンダーのいくつかがクラウド環境やコンテナ環境で彼らのソフトウェアが間に合わない、あるいは、動作させる予定がないということもあります。」

Jared Smith – Open Source Community Manager, Capital One

[Jared Smith](https://twitter.com/jsmith) – オープンソースコミュニティマネジャー、Capital One

Section 5

第五章

Launching your open source project

オープンソースプロジェクトの始動

Once you’ve properly planned your project, it’s time to take some more formal steps to set up your project, starting with legal preparations. This includes code scanning and scrubbing to ensure the code is safe to use, choosing the right open source license for your project, and setting project governance for smooth operations. Related topics to tackle include: setting up the proper infrastructure, preparing the code to get started, and then finally communicating the launch to the community and providing ongoing documentation.

プロジェクトの計画作りができたら、次は、法務的な準備を始め、プロジェクトの設立に向けより公式のステップに踏み出すべき時となります。これには、コードを安全に利用できることを保証するためのコードのスキャンと改善の実施、プロジェクトに対する適切なライセンスの選択、円滑な運用のための管理・統制のルール作りなどが含まれます。関連した話題として、インフラの開設、プロジェクトの開始に向けたコードの設置、そして、最後にコミュニティに対するプロジェクト始動の通知やドキュメントの提供もあります。

Legal review

法務的レビュー

One of the worst things that can happen to a project is for the community to have distrust in the legal cleanliness of a codebase. It’s important to ensure the code you release has clear licensing and provenance. A full legal review is often helpful in making sure what gets contributed will be acceptable by others in the community. A key aspect of this review is verifying that your company has the right to release all of the code. Your legal review should include trademark due diligence and registration. Note that if you are contributing the project into a foundation, make sure you’re aligned on the trademark strategy before open sourcing your codebase.

プロジェクトに起こりえる最悪の事態は、コードベースの法的潔白性に関してコミュニティに不信感を持たれることです。企業が公開するコードが、ライセンス、および、出所に関して明確であることを保証することが重要です。貢献として提供されるものがコミュニティ中の他の人々に受け入れ可能であることを確かなものとするのに、すべてにわたる法務的レビューが役立ちます。このレビューの重要な側面は、企業として当該コード全体を公開する権限を有していることを検証することです。法務的レビューには、商標の審査、登録も含まれるべきです。ただし、プロジェクトをファウンデーションの中で推進しようとしているなら、コードベースのオープンソース化の前にファウンデーションの商標戦略に合致しているかを確認する必要があります。

You’ll also need to choose a license for your project. It’s important to document any licensing or intellectual property requirements. An IP Policy can be a helpful document to create to make all licensing and contribution requirements clear. Also, make sure your code has license headers or SPDX license identifiers embedded in each file. Another best practice is to require a Developer Certificate of Origin (DCO) ‘signed-off-by’ on each commit to help improve the provenance of the code. GitHub, for example, has built a tool for requiring this in any repository, available at <https://probot.github.io/apps/dco>.

プロジェクトのライセンスを選択する必要もあります。ライセンスや知的財産に関わる要件があればそれらを文書化しておくことが重要です。IPポリシーを文書化しておくと、ライセンスや貢献の仕方に関わるすべての要件を明確にするのに役立ちます。また、コードについては、各ファイルにライセンスヘッダー、あるいは、SPDXライセンス識別子を記述するようにしてください。各コミットにDCO(Developer Certificate of Origin) 、すなわち、「signed-off-by行」を要求し、コードの由来を証明しやすくすることも取り入れることが推奨される手法です。たとえば、GitHubでは、どんなリポジトリーにおいてもこれを要求するようなツールを用意しており、 <https://probot.github.io/apps/dco>で利用できます。

It’s important to be familiar with the common open source licenses and their tradeoffs. Some have explicit patent grants, some have defensive termination rights, some protect the rights of users, some offer cure provisions and some are just simpler or more commonly accepted in your industry. You also need to consider the licenses that are in use for your dependencies and other codebases your software may be combined with or integrate with.

一般的に採用されるいくつかのオープンソースライセンスと、それぞれのトレードオフについて精通しておくことが重要です。それらの中には、明示的なパテント許諾の条項を持つもの、防御的な停止権限の条項を持つもの、ユーザーの権利を保護するもの、救済条項を提示するもの、簡易なものや業界で広く受け入れられているものがあります。また、当該コードが依存するソフトウェアや、それが組み込まれた、あるいは、統合されたコードベースで採用されたライセンスについても考慮の必要があります。

In addition to the software source code, consider the license requirements for other aspects of your project. If you anticipate needing corporate commitments for patent grants, or the ability to relicense the project later, you may want to look at some of the more common Contributor License Agreements (commonly referred to as CLAs). Not all CLAs are alike, so consider this option carefully. Also be aware that CLAs can be an impediment to participation as developers often have to go through arduous approval processes to get them signed.

ソフトウェアのソースコードに加えて、プロジェクトの他の側面に関わるライセンス要件も検討しなければなりません。参加企業がパテント許諾の取り決めを必要とすることや、後でプロジェクトをリライセンスできるようにすることを見越すなら、一般的なContributor License Agreements (CLAs)をいくつか参照してみるのがよいでしょう。しかし、すべてのCLAsが必ずしも似通っているわけでもないので、注意深く検討する必要があります。また、CLAsは、開発者が社内の承認を得るのに骨の折れるプロセスを要することもあり、参加の障害となる可能性もあります。

Your project may also produce deliverables that are not software. If your project is producing documentation, discuss whether you should use a specific license for the documentation. For example, many open source projects will use an open source license for the software and a Creative Commons license for the documentation. Additionally, some projects seek to create specifications that may be implemented in various ways by others. Those projects should consider the option to use a specification license. One example of this would be the Open Container Initiative (OCI) which uses the OWFa 1.0 – Patent Only specification license for the specifications and the Apache License, Version 2.0 on the open source software implementation they’re building.

当該プロジェクトがソフトウェアではない成果物を作りだすこともあります。プロジェクトがドキュメントを作るのなら、それらのドキュメントに特別なライセンスを適用するのかどうかを議論する必要があります。たとえば、多くのオープンソースプロジェクトは、ソフトウェアに対してはオープンソースライセンスを適用し、ドキュメントに対してはクリエィティブコモンズライセンスを適用します。加えて、プロジェクトの中には、他の人々がさまざまな方法で実装することを許すようなスペシフィケーションを作ろうとするものもあります。そのようなプロジェクトは、スペシフィケーションライセンスを適用するという選択肢を検討すべきです。この実例は[Open Container Initiative (OCI)](https://www.opencontainers.org/)です。同プロジェクトは、スペシフィケーションに対しては、OWFa 1.0 – Patent Onlyスペシフィケーションライセンスを適用し、作成したオープンソースソフトウェアの実装に対しては、Apache License Version 2.0を適用しています。

Another common consideration in licensing is choosing between copyleft and permissive licenses. Copyleft is commonly used to describe licenses that require reciprocal sharing and in general try to guarantee the rights of users to receive the source code for the software they’re provided. Permissive licenses tend to favor making it easy for others to participate and share contributions without downstream obligations. This particularly favors software segments that require the ability for software producers to distribute proprietary software based on the open source codebase without revealing their changes.

ライセンス検討におけるもうひとつの共通的な考慮点は、コピーレフトライセンスとパーミッシブライセンスの間でどんな選択をするかという点です。コピーレフトという用語は、互恵的な共有を要求し、提供されたソフトウェアに対応したソースコードをユーザーが受領する権利を保証しようとするライセンスを言及するのによく使われます。パーミッシブライセンスは、下流における義務なしに他の人々が貢献に参加することや、成果物を共有することを容易にするのに都合の良い傾向のあるライセンスです。このライセンスは、修正部分を開示することなしに、ソフトウェア開発者がオープンソースコードベースを元としたプロプラエタリーソフトウェアを頒布することを必要とするソフトウェアセグメントに特に好都合です。

Each licensing approach has its advantages and disadvantages, but be aware of the potential for fragmenting your project, which is a particular problem for software that needs to be interoperable or provide portability across various vendor solutions. Often this issue is addressed by creating conformance programs that permit usage of the project trademark if the commercial solutions pass a community created test or set of requirements. Thinking about this up front will help inform your legal review and plans for the project. (For more extensive reading on open source legal issues and considerations, see our recommended Reading List.)

それぞれのライセンスの取り組みには、有利な点と不利な点があります。しかし、プロジェクトがバラバラになってしまう可能性には要注意です。すなわち、ソフトウェアで特に課題となるのは、異なるベンダーのソリューション間で、相互運用性と、可搬性の必要がある点です。この問題は、しばしば、適合性プログラムを作ることによって解決しており、それらの適合性プログラムは、商用ソリューションがコミュニティの制定したテストや一連の要件をパスすることを条件として、プロジェクトの商標使用を許可しています。前もってこのことを考慮しておくと、法務的レビューに情報を与え、プロジェクトの計画作りに役立ちます。(オープンソースの法務的な課題、検討事項に関するさらなる情報として、[Reading List](https://www.linuxfoundation.org/open-source-guides-reading-list/)をお勧めします。)

In summary, the steps in the legal review process include:

要約すると、法務的レビューのステップでは、以下を実施します。

Consider the impact of open sourcing on your company’s intellectual property

Ensure full compliance with open source licenses

Select an open source license for the source code to be released, document all licensing requirements very clearly in your project

Decide if you need a contributor agreement

Consider the possible non-software outputs from the community and the appropriate licenses, such as documentation and specifications

Decide on any trademark related considerations

Decide whether there are additional factors to build into your plans for an ecosystem, such as conformance testing

　　オープンソース化することによる企業の知的財産への影響を検討

　　オープンソースライセンスへの完全な準拠性を確認

　　公開するソースコードに対するオープンソースライセンスを選択、当該プロジェクトにおけるライセンス要件を明確に文書化

　　Contributor License Agreements (CLAs)を必要とするかを決定

　　コミュニティの非ソフトウェア生産物を考慮し、たとえば、ドキュメントやスペシフィケーションに対して適切なライセンスを選択

　　商標に関連した考慮事項を決定

　　プロジェクトに関連したエコシステムの構築を計画に組み込むために、たとえば、適合性テストのような要素をどうするか決定

Technical review

技術的レビュー

The technical review verifies that the source code can function without dependencies on other internal code or development practices, and that it does not include third-party code the company cannot include in the open source release.

技術的レビューにおいては、ソースコードが他の社内コードや社内の開発技法に依存することなく機能すること、そして、オープンソースとして公開することのできない第三者コードが含まれていないことを確認します。

You’ll want to ensure that no portion of the code you plan to release violates intellectual property of another company, such as patents. There are plenty of patent trolls out there who will go after code that violates patents held by others. This is a huge issue with major negative implications and you must get it right from the start. To do this, companies often scan the code with specialized scanning tools to ensure it is clean. Add license and copyright notices, as well as documentation describing what it is and how it can be used.

公開を計画しているコードが特許など他社知財を侵害していないことを確認してください。他企業の特許を侵害する可能性のあるコードを付け回っているパテントトロールが数多く活動しています。これはネガティブな意味合いをもつ大きな問題ですが、プロジェクトの立ち上げ時から正しく対処しなければなりません。これを行うのに、多くの企業が、コードに問題がないことを確認する特別なスキャンツールを使っています。また、ライセンス告知や著作権告知、さらには、ソースコードがどんなものなのか、どう利用されるのかを記述したドキュメントをプロジェクト情報として用意してください。

The technical review should include verification of all license and copyright notices, and private code comments should be scrubbed. Steps include:

技術的レビューでは、すべてのライセンス告知と著作権告知を検証し、個人的なコードコメントを除去すべきです。実行すべきステップとしては、以下があります。

Remove critical dependencies on non-public components

Provide documentation and use case examples

Remove internal comments, references to other internal code, etc.

Ensure coding style is consistent

Update copyright notices in source code files

Add license notice in source code files

Add license text as a file in the root directory

　　非公開コンポーネントへの依存性を排除

　　ドキュメントと使用例を提供

　　個人的なコメント、社内向けのコメント、他の社内コードへの言及を除去

　　コーディングスタイルの一貫性を確認

　　ソースコードファイルの著作権告知を適切に更新

　　ソースコードファイルにライセンス告知を付加

　　ライセンス文書をルートディレクトリー下のファイルとして追加

Governance

管理・統制

To ready a project for launch, you also must define the technical requirements for project governance. Governance is the process by which the project makes decisions regarding strategy, releases, direction, and development priorities. Decision making should be public and open, to help ensure that all participants are aware of changes to the project and to maintain transparency. Also, consider if your governance should include paths for escalating issues.

プロジェクト始動の前に、プロジェクトの管理・統制に関する技術的要件を定義しなければなりません。管理・統制とは、プロジェクトの戦略、改版・公開、方向性、開発優先項目を決定するプロセスを指しています。すべての参加者がプロジェクトに起こる変化を知り、透明性を維持するために、決定の仕方は公開かつオープンでなければなりません。また、管理・統制に、問題をエスカレートするためのパスを持つべきかについても検討すべきです。

It’s important to decide early in the process what criteria must be met to participate in the project governance body. Decisions should be formalized on how features and bugs will be tracked, how code will be submitted, and who will manage the release process.

早期の段階で、プロジェクトの管理・統制組織(たとえば、理事会)に参加するためにどんな基準を満たさなければならないかを決定することが重要です。どのように機能追加やバグがトラックされるのか、どのようにコードが提出されるのか、誰が改版・公開のプロセスを管理するのかなどは、公式な決定を行うべきです。

You want to make sure the people entrusted with the project have the tools they need to operate and maintain the project. This is where your open source program office and manager comes in.

プロジェクトの責任を任せられた人々には、プロジェクトを運営・管理するのに必要なツールが確実に使えるようにしなければなりません。[オープンソースプログラムオフィスとプログラムマネージャー](https://www.linuxfoundation.org/?page_id=34557)は、このような場面を想定して設けられます。

“You need to make sure the people that need to get these things done are well empowered to be successful. You also need to be conscious of not intermixing the business half of the project with the technical half of the project – they need to have distinct leadership. That way you don’t get things stuck in the tracks. You’re not getting people making out-of-context decisions. Let the business unit help make the technical unit more successful.”

「これらの仕事を成し遂げることを課された人々には、それを推進できる十分な権限を与えられる必要があります。また、プロジェクトのビジネスの領分と技術の領分を入り混じらせないように気をつける必要があります。それらには明確なリーダーシップが必要です。そのようにして流れがとどこおらないようにします。また、筋違いな決定をしないようにします。技術側のより大きな成功のためにビジネス側に協力させます。」

John Mertic – Director of Program Management at The Linux Foundation

[John Mertic](https://twitter.com/jmertic) – プログラム管理ディレクター、Linux Foundation

Technical processes

技術的プロセス

Prior to a launch, it is often helpful to create a standard release process to schedule regular releases of the code as changes and improvements are made by the project maintainers. A schedule should be set up with well-defined and visible details for the development community and the business side of the project.

プロジェクトの始動に先立ち、改版・公開の標準的なプロセスを作っておき、プロジェクトのメンテナーによる変更・強化の実装とともに行われるコードの改版・公開の定常スケジュールを決めておくと役に立ちます。スケジュールは、プロジェクトの開発コミュニティとビジネス側の人々のために、細部にわたり明確で可視的な情報とともに提示されるべきです。

How often those releases should be made depends on your community’s expectations. If the project is enterprise-focused and you’re trying to build something that is very hardened, your release schedule might be twice a year. If the project is smaller in scope and more agile and you are trying to get pieces of it out there, perhaps you might be releasing the code monthly or weekly. The key to the schedule is that the community must dictate the timeline and understand its ability to support the project from a velocity standpoint while giving users what they need and expect.

どれくらいの頻度で改版・公開を行うかは、コミュニティがどれくらい待ち望んでいるかに依存します。プロジェクトが企業向けのもので、完成度の高いものを開発しようとしているのなら、改版・公開のスケジュールは、年間2回程度でしょう。また、プロジェクトが、より限定されたスコープと機敏さを特性として持ち、プロジェクトの一部を取り出して利用することを想定しているようなら、コードは毎月とか毎週のような改版・公開も考えられるでしょう。スケジュールの考え方における鍵は、ユーザーに対しては必要なもの、待ち望むものを提供する一方で、速度の観点で、プロジェクトをサポートしていく自身の力量を十分に理解し、コミュニティ自ら予定表を作らなければならないという点です。

If the community provides feedback that the releases are coming too soon or too slowly, then you need to look at the process and make some adjustments. The big thing here is consistency, predictability, and visibility.

もし、改版・公開が速過ぎる、あるいは、遅過ぎるという声が上がり、コミュニティがそれにフィードバックを与えるなら、プロセスを見直し、調整を加える必要があります。ここにおいて重要なことは、一貫性、予測可能性、可視性です。

Leadership

指導体制

Setting leadership roles is also important before getting underway. That can mean different things for different projects. If you are starting a multi-company project with several enterprise participants, the project may need more formal governance, such as a governing board or other management groups. Other arrangements could simply require a technical committee that oversees all facets of the open source project from a technical perspective. The composition of the committee would include mostly technical leadership, as well as a liaison back to the executive team to provide updates about progress and project needs. The legal team can then be brought in as the technical members and executives see fit.

プロジェクトの活動開始前に指導体制を作っておくことが重要です。それはプロジェクトが異なれば、異なったものになりえます。複数の大企業が参加する企業プロジェクトを立ち上げるのなら、監理役員会議とか管理グループのような、より形式にこだわった管理・統制が必要になるでしょう。他の構成では、オープンソースプロジェクトのすべての細部を技術的観点から指導する技術委員会だけで足ることもありえます。委員会は主に技術的なリーダー、また、企業の経営幹部に対して進捗とプロジェクトのニーズに関する情報更新を行う連絡員で構成されるでしょう。法務チームの要員は、技術メンバーや経営幹部が適切だと判断した時に参加します。

Your top architectural minds will also certainly be there, as well as others who know how the code base will work. Altogether, the members of the committee will have a vision for where the project is going as well as support from within the developer community. Those are the people you want there at the table to plan the process.

企業において最もアーキテクチャーにたけた人物が、コードベースがどう動作するかを良く知る人々とともに、指導体制に参加するのがよいでしょう。委員会メンバーは、プロジェクトの進む方向に対するビジョンを持つとともに、開発者コミュニティからのサポートも受けます。これらはプロセスの計画作りの際に参加してもらいたい人々です。

“You have a fiduciary responsibility to your organization that is contributing this code, to make sure this is in line with your board of directors and your shareholders and all of that who are entrusted for this IP. You have to be sure they’re in alignment with this. But then you have to think of potential liabilities, risks, and pieces like that that open yourself up to problems. Don’t take this lightly.”

「指導体制の人々は、そのコードを貢献した企業・組織に対する受託者責任があり、プロジェクトが企業の役員・株主、さらには、この知的財産を託してくれたすべての人々の意に即したものとなるようにしなければなりません。指導体制の人々はこの方向性で一致していなければなりません。同時にまた、降りかかってくる責任、危険、また、それに類する事柄が問題にとなる状況も想定していなければなりません。これを過小評価してはいけません。」

John Mertic – Director of Program Management at The Linux Foundation

[John Mertic](https://twitter.com/jmertic) – プログラム管理ディレクター、Linux Foundation

Infrastructure

インフラ

Before the launch can ever happen, you need to prepare a repository for the project. This is essentially the infrastructure for a code repository where the project will be available to contributors all the time. Many projects use the well-known GitHub or GitLab repositories or host their own repositories with tools such as Gerrit. There are many other options available as well, but remember it’s often useful to consider making it simple for developers to participate and engage in your project. Choose your platform, open an account, and prepare a place for the code, set up workflows and get ready for the magic to happen.

プロジェクト始動の前に、プロジェクトのリポジトリーを用意する必要があります。これは、コードリポジトリーのインフラですが、貢献者はそこにアクセスすればいつでもプロジェクトを利用できます。多くのプロジェクトは良く知られた[GitHub](https://github.com/)や[GitLab](https://about.gitlab.com/)のリポジトリーを利用するか、あるいは、Gerritのようなツールを使って独自のリポジトリーでプロジェクトをホストします。その他にも多くの選択肢はありますが、開発者がプロジェクトに参加し、関わることを容易となるように考えることがよいでしょう。プラットフォームを選択し、アカウントを開設してください。そして、コード設置場所を用意し、ワークフローを定めれば、魔法の始まりに準備万端です。

Bug, issue, and feature tracking also should be included as part of a project’s infrastructure plans. You want to provide an easy place for contributors to make reports of problems that need to be fixed as well as requests for new features that might be useful added. Then there are automated build and test system processes which may need to be included as part of your project’s GitHub repository to keep your systems and project flowing smoothly while scanning and checking the code to ensure its quality.

プロジェクトのインフラ計画の一部として、バグ、問題点、機能開発をトラックする仕組みを含めるべきです。貢献者が、解決を必要とする問題の発生を報告し、また、追加すると有益だとみなされる新規機能を要望するための場を提供することが重要です。それから、プロジェクトのGitHubリポジトリーには、自動的なビルドシステムと自動テストシステムのプロセスを含めることが必要となる可能性があります。それにより、コードの質を確保するためのコードスキャンとコードチェックを実施しつつ、常にシステムとして円滑に作業を進めることができるようになります。

Website

Webサイト

The next step is creating a company-neutral web presence or wiki page for the project. The web page provides a place for the community to find information about the project including documentation, links to download the code, and more. The web page can also provide details about the project’s leadership, scope, users and contributors, budget, governance, and other details.

次のステップは、企業からは中立なプロジェクトのWebサイト、あるいは、Wikiページの開設です。Webページは、コミュニティに対してドキュメント、コードのダウンロード、さらにはもっといろいろな情報を見つける場を提供します。また、Webページは、プロジェクトの指導体制、スコープ、ユーザーと貢献者、予算、管理・統制、その他について詳細な情報を提供することもできます。

Communication

コミュニケーション

Offering communication channels for your community to ask questions for help is critical. You’ll want to find tools that can be integrated into the entire development workflow (for example, receiving notifications for support requests, code check-ins, error logs, and other tasks). You’ll also want to provide critical discussion forums and a mechanism for community members to get quick answers from others who are also working on a project. All are important means of communication that help move projects forward in real-time.

コミュニティとして支援要請のためのコミュニケーションチャネルを提供することは非常に重要です。開発ワークフロー全体 (たとえば、サポート要求、コードのチェックイン、エラーログ、その他の作業の通知を受信) に組み込むことが可能なツールが必要です。また、重要な問題の議論のためのフォーラムや、コミュニティのメンバーが同じプロジェクトに取り組む他の人々から迅速な回答を受け取る仕組みも必要です。これらすべてがコミュニケーションの重要な手法であり、プロジェクトをリアルタイムで前進させるのに役立ちます。

One project tool to consider is Slack, an online team project management and communications platform where users can access and share messages and files, organize workflows, perform searches for information, and more. However, Slack is a proprietary tool and can cost money to maintain. There are other open source options out there such as IRC, Gitter.im and others. For example, the Hyperledger project uses a communication system called Rocket.Chat which is fully open source and provides similar features to Slack. If you need modern forums, Discourse is a great option that is fully open source and also provides optional hosting.

このようなプロジェクトツールとして検討すべきものの一つは[Slack](https://slack.com/)です。Slackは、オンラインのチームプロジェクト管理とコミュニケーションの機能を提供するプラットフォームです。ユーザーは、メッセージおよびファイルへのアクセスと共用、ワークフローの編成、情報の検索、さらに多くのことを行うことができます。しかし、Slackはプロプラエタリーなツールであり、使い続けると課金が生じます。オープンソースの選択肢としては、IRC、[Gitter.im](https://gitter.im/)、その他のものも存在します。たとえば、Hyperledgerプロジェクトでは、[Rocket Chat](https://rocket.chat/)と呼ばれるコミュニケーションシステムを採用しており、完全にオープンソースでありながら、Slack同様の機能を提供しています。最新のフォーラム機能が必要なら、[Discourse](https://www.discourse.org/)は、完全にオープンソースの素晴らしい選択肢であり、ホスティングサービスをオプションとして提供しています。

When choosing a communication system, be mindful about any form of lock-in, financial costs, and how easy it is to migrate to a new system in the future. As your community grows, you need to be able to adapt to any new communication channels out there, it wasn’t so long ago that newsgroups were the preferred communication style for a lot of open source projects.

コミュニケーションシステムを選択する際は、何らかの形のロックイン、課金、将来的な新システムへの移行の容易さに気を付けるべきでしょう。コミュニティの成長とともに、新たなコミュニケーションチャネルを採用できるようにしておく必要があります。newsgroupがたくさんのオープンソースプロジェクトで使われるコミュニケーションスタイルであったのはそんなに昔の話ではありません

“Some 190 of our various open source projects are arranged in one central spot in an Autodesk section on GitHub. We used to have at least 14 different sections that focused on Autodesk open source projects. By using some code from Twitter, we brought the 14 sections into a single view where they can be seen by visitors. It’s important from a company perspective to make sure people see what you’ve launched and that there’s a central place they can go to find it and ask questions.”

「当社のおよそ190件の異なったオープンソースプロジェクトは、ただひとつの中心的開発スポットであるGitHub上のAutodeskセクション中に配置しました。かつて、私たちはAutodeskのオープンソースプロジェクトとして少なくとも14セクションを運営していました。Twitterのコードをいくつか使うことによって、14セクションはひとつのビューにまとめられ、外部訪問者からも見てもらえるようにしました。企業の立場として、何を始めたのか、そして、人々が訪問してプロジェクトの内容を見たり、質問したりできる中心的スポットがあるということを外部の人々が確実に分かるようにすることが重要です。」

Guy Martin – Director of Open at Autodesk

[Guy Martin](https://twitter.com/guyma) – Open部門ディレクター、Autodesk

Launch and maintain

始動と保守

Eventually, after all the planning, preparations and multi-faceted reviews and steps along the path, you’ll be ready to launch and begin to maintain your open source project. You’ll get there through public planning, open communications and the completion of your top to bottom infrastructure, along with the steps you created for governance, technical processes, and everything in between.

いろいろな計画作り、準備、そして、多面的レビューと数々のステップの後、いよいよ、独自のオープンソースプロジェクトを始動させ、保守を開始する段となります。そこに至るまでには、広範囲にわたる計画作り、オープンになされたコミュニケーション、上層・下層インフラの整備があったでしょう。さらには、管理・統制、技術プロセス、および、両者の間の諸々の事柄のために作ったいくつものステップもあったはずです。

Once these critical parts are all in place, it is time to open it to the world and gain input from contributors. When interested contributors inspect the project and see it is thoughtful, concise, and valuable, they’ll be excited to participate out of the gate because it’s something they can use.

これらの必須の要素が整うと、当該プロジェクトを世界に公開し、貢献者からの入力を得る時となります。関心を持つ貢献者たちがプロジェクトを調べ、それが思慮に富み、簡明で、価値あるものであるとみなされると、彼らは、自ら利用できる何かがあるということで、堰を切ったようにわくわくして参加します。

Essential tasks prior to launch:

始動に先立つ重要な仕事

Pre-brief launch partners

Ensure that all project infrastructure is running, secure, and scalable

Ensure developers join and monitor communication channels (IRC, mailing lists, etc.)

Release source code

Follow the open source development model

　　パートナーに始動を事前に説明

　　プロジェクトのインフラが動作すること、安全であること、拡張性があることを確認

　　開発者がコミュニケーションチャネル(IRC、メーリングリスト、その他)に参加し、受信することを確認

　　ソースコードを公開

　　オープンソース開発モデルを踏襲

Don’t forget marketing!

マーケッティング活動を忘れないで

Of course, the launch doesn’t mean the end of your work. To keep the project moving along, there are a series of additional business and marketing steps that need attention as well. They include promoting the project, laying out a successful operational strategy, providing a realistic budget and project branding, as well as establishing lively social media accounts and useful domain names to bolster its long-term success.

もちろん、プロジェクトの始動が活動の終わりを意味している訳ではありません。プロジェクトの前進を続けさせるためには、注意が必要な一連のビジネス上、および、マーケッティング上の追加のステップがあります。それらの中には、プロジェクトの普及活動、上手くいく運営戦略の設定、現実的な予算の提供、および、プロジェクトのブランディングが含まれ、さらに、長期的成功を助けるような活動的なソーシアルメディアアカウントおよび使いやすいドメイン名を確保することも含まれます。

A marketing review establishes guidelines for branding. This is particularly important, as it helps to ensure a consistent message in the market. Steps in the marketing review include:

マーケッティングレビューによってブランディングに向けたガイドラインを確立させます。これは、市場における一貫したメッセージを確かなものとするのに役立つため特に重要です。マーケッティングレビューのステップには次があります。

Design a project logo, color scheme, website, collateral, etc.

Formalize branding guidelines

Register social media accounts for the project (Twitter, Facebook, LinkedIn, etc.)

Register domain names for the project

　　プロジェクトロゴ、色使い、Webサイト、マーケッティング・コラテラル(販促資料)、その他をデザイン

　　ブランディングに向けたガイドラインを定める

　　プロジェクトのソーシアルメディアアカウントを登録(Twitter、Facebook、LinkedIn等)

プロジェクトのドメイン名を登録

Now that you have the project, it’s your job to promote it and let people know it’s out there so they can use it and work on it. As a marketer, it’s a fun challenge because your litmus test for success here is how many people you can drive to the project to join in to contribute code, participate in forums, offer bug fixes and report issues.

プロジェクトの実働が始まると、プロジェクトを普及させ、そこにプロジェクトが存在しているので、人々が利用し、活動に参加できるということを知らしめる責任があります。人々がプロジェクトに引きつけられ、コードの貢献、フォーラムへの参加、バグ修正の提出、問題点の報告などの活動にいかに多くの人が加わってくれるかが成功を示すリトマス試験となるため、マーケッティング活動に携わる者にとって、これは楽しみな挑戦となります。

“Since the community is essential to this, you want to make sure that you are taking care of the community. This can manifest itself in small things like acting quickly on requests and making sure your projects are helping. So, when someone comes to your project they can look at it and know how it’s doing.”

「この活動にコミュニティは不可欠なので、しっかりとコミュニティの面倒を見ていくことが必要です。そのことは、小さなことですが、要望に対して敏速に行動することや、プロジェクトが確実に役に立つことで証明できます。誰かがプロジェクトを訪問したとき、彼らはプロジェクトの様子を見て、どうやっているかを知ることができるのです。」

Christine Abernathy – Open Source Team Developer Advocate at Facebook

[Christine Abernathy](https://twitter.com/abernathyca) –オープンソースチーム開発者アドボケート、Facebook

Build the community

コミュニティの構築

After the project has launched, it is essential to monitor the vitality of the external community.

プロジェクトの始動後、外部のコミュニティの活度をよく見ていることが重要です。

Community building does not happen automatically. In the early stages of the project, it may be necessary to host developer events or sponsor meetups at major conferences to build momentum.

コミュニティの構築は自動的に起きるものではありません。プロジェクトの初期の段階では、はずみをつけるために開発者イベントを主催したり、大きなコンファレンス中にミーティングを行うための資金援助をしたりすることが必要となるでしょう。

It’s also extremely important to manage expectations and fulfill obligations for project governance and transparency.

また、プロジェクトに対する期待を正しく管理し、義務を果たすことはた、プロジェクトの管理・統制と透明性のために非常に重要なことです。

Ongoing activities include:

引き続き実施すべき活動には以下があります。

Designate a community manager or community advocate

Ensure any changes to direction or governance are clearly communicated

Follow best practices of other similar communities

Encourage and provide opportunities for face-to-face community building

Identify appropriate events and have the community submit talks

Consider doing local meetups

　　コミュニティマネージャー、あるいは、コミュニティアドボケートを指名

　　方向性や管理・統制のいかなる変更も確実に伝達

　　他の類似コミュニティのベストプラクティスを踏襲

　　直接対面のコミュニティ構築を奨励し、その機会を提供

　　適切なイベントを決めて、コミュニティが講演する機会を得る

　　地域ごとのミーティング開催を検討

By building up a diverse group of contributors, you can later decide to move your project to the next level by having discussions with other enterprises and organizations that see the work as valuable to determine if they are interested in investing their time, money and other resources to expand your initial efforts. By gaining input and resources from others, the project can be expanded and grown to do more for additional contributors.

貢献者の多様なグループを構築することができれば、当初のコードを拡張することに対して彼らの時間、資金、その他の資源を投資することに関心があるほどの価値があるかを見極めている、他企業・組織と議論を行い、プロジェクトを次の段階に進めるかどうかを決定することができるようになります。その議論では、当初のコードを拡張することに対して彼らの時間、資金、その他の資源を投資することに関心があるかどうかが見極められます。他企業・組織からの入力と資源を得ることによって、プロジェクトは拡大し、また、成長し、貢献者の増加によってさらに多くのことができるようになります。

Such growth means that additional businesses may want to contribute more money to bring their own developers in to join the efforts and help move the work forward by putting their weight behind the efforts you’ve begun. That may involve $10,000 or $250,0000 or more, depending on the importance of the project and what it means to other companies. Once your project begins, other companies can come in to contribute funding toward the work if it will aid their missions.

そのようなコミュニティの成長は、拡張されるビジネスは、企業が開発者を送り込んで開発活動に参加し、そして、当初のコードの拡張に彼らの開発力を注ぐことによって前に進めるたには、より多くの資金がサポートとされなければならないということです。それは、プロジェクトの重要性、および、そのプロジェクトが他企業に何を意味するかによって、$10,000、あるいは、$250,000、さらには、それ以上の投資を意味します。プロジェクトが始まれば、他企業はプロジェクトがその企業の役に立てば、プロジェクト活動を前に進めるために、財政的な貢献を行うでしょう。

This happens regularly today, as enterprises and organizations realize that the technology problems they are trying to solve are larger than any of them individually. That’s when they can begin to see strategic value in joining together with other companies in vendor-neutral joint projects that are working for the greater good in solving technical problems faced by businesses.

今日、このようなことは頻繁に起きています。今や、企業や組織が解決しようと試みる技術的な課題の大きさは、個別に解決できる課題の上限をはるかに超えているからです。企業や組織が他企業とともにベンダー中立なジョイントプロジェクトに参加することの戦略的価値にようやく気づき始めたのです。そのようなジョイントプロジェクトは、各企業が直面する技術課題を解決する解を得るという社会的な大義のために活動しています。

Several examples of this approach are the open source projects for Hyperledger, which is a collaborative effort sponsored by The Linux Foundation to advance cross-industry blockchain technologies, and for the Cloud Native Computing Foundation, which is open source software used to create modern private and public clouds. Enterprises contribute not only developers to work on these large-scale projects but also provide plenty of dollars to help fund, promote and drive the technologies forward.

このアプローチをとる例を2つ挙げます。1つ目は、[Hyperledger](https://www.hyperledger.org/)のためのオープンソースプロジェクトです。同プロジェクトは、Linux Foundationが後援している共同開発活動であり、業界を跨るブロックチェーン技術の強化・発展を目指しています。2つ目は、[Cloud Native Computing Foundation (CNCF)](https://www.cncf.io/) で、こちらは最新のプライベートクラウド、および、パブリッククラウドを構築するために利用されるオープンソースソフトウェアです。多くの企業は、これらの大規模プロジェクトに開発者として貢献するだけではなく、多くの資金を提供し、これら技術の普及・強化を手助けしています。

Section 6

第六章

Final words

結語

Taking the leap to start an open source project can be a bit mystifying and even scary, at least the first time. But when your enterprise sees and quantifies the exponential value it can receive in the process, that first project may be only the start of your corporate journey to using open source software more strategically. Learn how others have taken this path to help make your next open source endeavor a successful one.

オープンソースプロジェクトを立ち上げ、飛躍させることは、少なくとも始めは、ちょっと神秘的で、恐ろしいところさえあります。しかし、企業がそのプロセスによって受け取る指数的価値を理解し、定量化すると、その最初のプロジェクトはオープンソースソフトウェアをより戦略的に活用する企業活動の最初の一歩となるでしょう。それに続くオープンソース活動への挑戦が成功するために、他の企業がこの道のりをどのようにたどったかを学んでください。

Section 7

第七章

Open source project launch checklist

オープンソースプロジェクト始動のチェックリスト

Considerations

検討事項

Evaluate possibility of joining an existing open source project

Evaluate the company’s ability to launch and maintain the project using the open source model

Evaluate the likelihood that other companies may join the project from the start

Evaluate success factors and set appropriate metrics for the open source project

既存のオープンソースプロジェクトに参加することの可否を評価

企業にオープンソース開発モデルに従ってプロジェクトを立ち上げ、維持していく力量があるかを評価

立ち上げ当初から他企業がプロジェクトに参加することの確度を評価

オープンソースプロジェクトの成功要因を評価、また、成功の評価基準を設定

Business strategy & plan

ビジネス戦略、および、計画作り

Determine and set goals for your project

Gather reasons for doing it from stakeholder

Select code to be considered for the project

Decide whether the project will include all code for an application or just parts of it

Create a business case for the selected proposal

Determine if there is executive buy-in for the move

Plan resource commitments for developers and funding

Set budgets for costs, including development time, infrastructure and related expenses

Gather executives and tech staff for project discussions and decision-making

Debate and finalize project scope and code selection

プロジェクトのゴールを決定し、設定

プロジェクト実施の理由を利害関係者から収集

プロジェクトに提供するコードを選定

プロジェクトにアプリのコード全体を提供するのか、一部なのかを決定

選定されたプロポーザルに対してビジネスケースを作成

プロジェクトに対して経営幹部の支持があるかを見極める

開発者、および、資金の面で資源提供のコミットメントを計画

費用を予算化、開発者の人月、インフラ、その他の関連費用

プロジェクトの議論、および、決定のために経営幹部や必要な技術スタッフを招集

プロジェクトのスコープとコード選択について議論し、最終決定

Legal Review

法務的レビュー

Consider the impact of open sourcing on your company’s intellectual property

Ensure full compliance with open source licenses

Select an open source license for the source code to be released, document all licensing requirements very clearly in your project

Decide if you need a contributor agreement

Consider the possible non-software outputs from the community and the appropriate licenses, such as documentation and specifications

Decide on any trademark related considerations

Decide whether there are additional factors to build into your plans for an ecosystem, such as conformance testing

オープンソース化が企業の知財に与える影響を検討

オープンソースライセンスへの完全な準拠性を確認

公開するソースコードのオープンソースライセンスを選定、当該プロジェクトにおけるすべてのライセンス要件を明確に文書化

CLAs(Contributor License Agreement)の必要性があるか決定

コミュニティから、ドキュメントやスペシフィケーションのような非ソフトウェア生産物が出て来るかを考慮し、それらに対するライセンスを検討

商標に関わる考慮事項を決定

プロジェクトに関連したエコシステム形成のために適合性テストのような要素を加えるかを決定

Technical Review

技術的レビュー

Remove critical dependencies on non-public components

Provide documentation and use case examples

Remove internal comments, references to other internal code, etc.

Ensure coding style is consistent

Update copyright notices in source code files

Add license notice in source code file

Add license text as a file in the root directory

公開しないコンポーネントに対する依存性を除去

ドキュメントと使用例を提供

社内開発者を想定したコメントや他の社内コードへの参照を排除

コーディングスタイルの一貫性を確認

ソースコードファイル中の著作権告知を更新

ソースコードファイルにライセンス告知を付加

ルートディレクトリー下にひとつのファイルとしてライセンス文全体を追加

Governance and Processes

管理・統制とプロセス

Define project governance steps and structure

Set up a code repository, bug reporting, and code testing infrastructure

Create supporting Slack channels, forums, and Wikis

Create open lines of communication with contributors for project success

プロジェクトの管理・統制に対するステップと構造を定義

コードリポジトリー、バグ報告、および、テストのためのインフラを開設

プロジェクト支援のためのSlackチャネル、フォーラム、Wikiを用意

貢献者とのオープンなコミュニケーション手法を用意

Branding and Marketing

ブランディングとマーケッティング

Set marketing strategy to promote an active contributor community

Design project logo, color scheme, website, collateral, etc.

Formalize branding guidelines

Register social media accounts for the project (Twitter, Facebook, LinkedIn, etc.)

Register domain names for the project

貢献者コミュニティを活性化するためのマーケッティング戦略を設定

プロジェクトロゴ、色使い、Webサイト、マーケッティング・コラテラル、その他をデザイン

ブランディングのガイドラインを定める

プロジェクト用のソーシアルメディアアカウントを登録(Twitter、Facebook、LinkedInなど)

プロジェクト用のドメイン名を登録

Launch and Maintain

始動と保守

Open project and begin development work and contributions process

Designate a community manager or community advocate

Ensure any changes to direction or governance are clearly communicated

Follow best practices of other similar communities

Encourage and provide opportunities for face-to-face community building

プロジェクトの開設、開発活動と貢献プロセスの開始

コミュニティマネージャー、あるいは、コミュニティアドボケートを指名

方向性や管理・統制に関わる変更は明確に伝達されるよう確認

他の類似コミュニティのベストプラクティスを踏襲

コミュニティ構築のために直接対面を奨励、ミーティングの機会を提供

These resources were created in partnership with the TODO (Talk Openly, Develop Openly) Group – the professional open source networking group at The Linux Foundation. A special thanks goes out to the open source program managers who contributed their time and knowledge to making these comprehensive guides. Participating companies include Autodesk, Comcast, Dropbox, Facebook, Google, Intel, Microsoft, Netflix, Oath (Yahoo + AOL), Red Hat, Salesforce, Samsung and VMware. To learn more, visit: todogroup.org

本資料集は、Linux Foundationにおいて形成されたオープンソース・ネットワーキングの専門家グループであるTODO (Talk Openly, Develop Openly) Groupとの協力によって作成されました。本ガイド集のために時間と識見を貢献してくれたオープンソースのプログラムマネージャーの皆様に特に謝意を表します。本ガイドの作成に参加してくれた企業は、Autodesk、Comcast、Dropbox、Facebook、Google、Intel、Microsoft、Netflix、Oath (Yahoo + AOL)、Red Hat、Salesforce、Samsung、VMwareの各社です。詳細は、[todogroup.org](http://todogroup.org/)をご参照ください。

This work is licensed under a Creative Commons Attribution ShareAlike 4.0 International License.

*本資料は、Creative Commons Attribution ShareAlike 4.0 International Licenseのもとに利用が許諾されます。*