THE LINUX FOUNDATION | Research

オープンソース プロジェクトを GitHub でホスト & 管理するための推奨プラクティス

2023 年 3 月

Ibrahim Haddad, Ph.D.
Vice President, Strategic Programs (AI & Data)
The Linux Foundation

序文 Jeff McAffer
Senior Director of Product
GitHub

オープンソース プロジェクトを GitHub でホスト & 管理するための推奨プラクティス

GitHubは、**開発者が協力してコードを共有できる**プラットフォームであり、オープンソース開発とプロジェクト管理をサポートする幅広いツールを提供します。

ドキュメントは、GitHub上のオープンソース プロジェクトに**不可欠なコンポーネント**であり、プロジェクトの目的、コード、使用方法、コントリビューション、指示、ガイドラインを説明します。

イシュー トラッカー、フィードバック プラットフォーム、コミュニティ フォーラムなどのコミュニケーション チャンネルを通じて、**ユーザー サポートを管理します。**

Open Source Initiativeによって承認されたライセンスなど、プロジェクトに必要な使用、変更、共有のレベルをサポートする**ライセンスの種類を選択します。**

2要素認証、アクセス制御、コード レビュー、スキャン ツールなどのセキュリティ機能を実装することにより、**プロジェクトのコードを保護します。**

2つの一般的なライセンスの概念は、Developer Certificate of Origin(DCO)とContributor License Agreement(CLA)です。これらは、**コントリビューションの条件と権利の概要を示すものです。**

英語は世界中で広く話され、理解されているため、Githubのコンテンツの作成やコミュニケーションをする際に使用するのに**最適な言語**です。

オープンソースの中核的な原則である**ピア レビュー、早期かつ頻繁なリリース、継続的なテストと結合**は、協力的で透明性のあるプロジェクトの確立に役立ちます。

GitHubでホストされるオープンソース プロジェクトでは、**正確なライセンス情報を提供することが重要です。**

GitHubのバージョン管理であるGitを使用すると、開発者はコードの変更を時間の経過とともに追跡できます。

DCOは、開発者が**プロジェクトへの貢献が自分のものであり、**コードを登録するために必要な権利を持っていることを**証明する方法**です。

CLAは、開発者とプロジェクトの所有者または保守担当者との間の法的合意であり、コントリビューションの条件を概説し、プロジェクトがコードを使用および配布するために必要な権利を持つことを保証します。

目次

序文

オープンソースの原則は比較的単純ですが、「それを正しくする」ことは困難な場合があります。オープンソースは、テクノロジー、コミュニティ、ビジネス、個人のアイデンティティの交差点に位置するため、ある程度のツールと厳密なーベストプラクティスーが成功への道をスムーズにするのは当然です。

私が Microsoft Open Source Programs Office(OSPO) のディレクターを務めていた頃、同社は GitHub に数百人の開発者とリポジトリを持つ企業から、数万人の開発者とリポジトリを持つ企業へと進化しました。私たちは、アド ホックな運用から、形式的ではあるが依然として手動のプロセスへと移行し、GitHub での配備をほぼ無人化し、数十人から数千人のプロジェクト コミュニティへと移行しました。私たちは、その道をスムーズにするために、大量のツールとベストプラクティスを開発しました。

私が 5 年近く前に GitHub に移ったのは、これらのツールと学んだことのいくつかを GitHub 製品に、そして究極的には世界中のオープンソース開発者にもたらすことを目的としていました。現在、GitHub には、ディスカッションからアクション / チェック、パッケージ、リリース、セキュリティに至るまで、あなたとあなたのプロジェクトを支援する多くの機能と可能性があります。私はそれらをすべてリストアップすることさえできません。

オープンソースの成功は、コードだけでなく、人やコミュニティにも関係しています。この記事で Ibrahim 氏は、オープンソースにおける彼の長い歴史を応用して、GitHub でのオープンソースの取り組みを最大限に活用するための出発点を作成します。ここで概説するベストプラクティスを使用することで、プロジェクトを中心とした、安全で堅牢で活気に満ちたコミュニティを構築できます。GitHub はインフラストラクチャの多くを提供し、ユーザーはイノベーションとコミュニティを提供します。

それではリポジトリでお会いしましょう。

Jeff McAffer
GitHub

概要

オープンソース ソフトウェア (OSS) は、私たちの世界を変革し、デジタル経済のバックボーンとなり、デジタル世界の基盤となっています。今日、OSS はデジタル経済に力を与え、私たちの生活を向上させる科学的および技術的革新を可能にします。インターネットや私たちが日常的に使用するモバイル アプリケーションから、私たちが未来を構築するために使用するオペレーティング システムやプログラミング言語にいたるまで、OSS は重要な役割を果たしてきました。OSS はテクノロジー業界の生命線です。

オープンソース開発における GitHub の利用は、コラボレーションやコード共有のためのプラットフォームを提供することで、近年ますます普及してきています。

適切なガイドラインがあれば、GitHub でのオープンソース プロジェクトの管理はそれほど難しくなくなります。この資料では、オープンソース開発に GitHub を使用するためのベストプラクティスの概要を提供します。これらのプラクティスに従うことで、オープンソース開発者は、GitHub 上のプロジェクトの組織性、分かりやすさ、コラボレーションを向上させ、他の開発者が採用したり貢献したりしやすくなるようにできます。

はじめに

OSS の可用性は、組織が製品を開発および提供する方法を変化させています。透過的な開発コミュニティと公開ソースコードへのアクセスにより、組織はソフトウェアの調達、実装、テスト、導入、および保守について異なる考え方をすることができます (図 1)。

OSS は、関係するすべての人に多くのメリットをもたらすエコシステムを構築しました。さまざまな業界の組織が、OSPO の下でオープンソースの運用を構築し、成長させています。これは、オープンソース プロジェクトをより効率的かつ効果的に利用し、貢献することを支援し、その戦略的影響から利益を得るためです (図 2)。

OSS は、数十億ドル規模の OSS のメリットを組織が享受できるようにすることで、開発の共有を可能にし、研究開発コストを削減します。これにより、組織はより優れた製品やサービスを開発することができます。さらに、OSS は、ビジネス ニーズを上流のオープンソース プロジェクトと連携させることで、製品開発を加速し、市場投入までの時間を短縮するのに役立ちます。組織がオープンソース プロジェクトに参加するのは、それが楽しいからではなく、それがビジネスや製品戦略の一部であるからです。

図1
オープンソースは技術市場の加速者

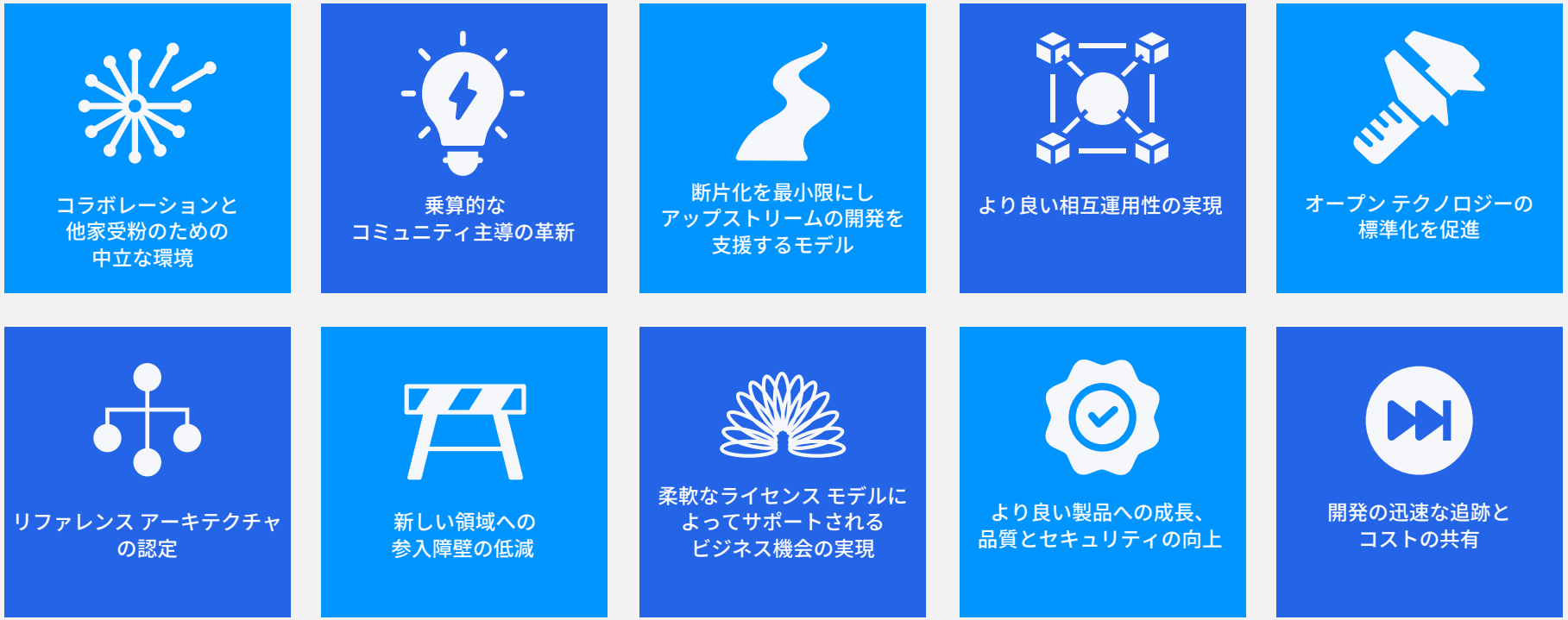

図2
OSS の戦略的影響力

- オープン ソリューションの開発を促進
- オープン スタンダードの実装を提供

- 市場のコモディティ化
- 非戦略的なソフトウェア資産の価値の引き下げ
- 開発コストの共有

- 製品やサービスのエコシステム構築による需要喚起

- 他者との連携
- 顧客との関わり
- 共通の目標を持つ関係の強化

ほとんどのオープンソース開発はどこで行われていますか？　そう、**GitHub** です。GitHub は、オープンソース開発にとって重要なプラットフォームです。これは、Git バージョン管理システムを使用して、開発者が共同でコードを共有できるようにする Web ベースのホスティング サービスです。オープンソース開発者がコードベースの変更を保存、管理、追跡し、他のコントリビュータと共同で開発作業を行うための一元化されたプラットフォームを提供します。

GitHub の重要な利点の 1 つは、開発者がコードを他の人と簡単に共有できるため、他の人が簡単に貢献、レビュー、変更のマージができることです。開発者が物理的な場所に関係なく大規模なプロジェクトで協力できるため、オープンソース開発にとって強力なツールです。GitHub はまた、ユーザーがオープンソース プロジェクトを作成し、問題を作成および管理し、プル リクエスト (PR) やコメントを介して他の開発者とコミュニケーションできるようにすることで、コミュニティ構築とコラボレーションのためのプラットフォームを提供します。

さらに、GitHub は、オープンソース開発者のニーズに合わせて用意された幅広いツールと機能を提供します。プロジェクト管理ツール、コードレビュー機能、パッケージング、リリース、デプロイメント機能、継続的インテグレーション サービスなどの他の開発者ツールとの組み込み統合などの機能が含まれており、オープンソース開発のためのオール インワン プラットフォームとなっています。

この資料では、GitHub のプレゼンスを向上させ、より多くのユーザと開発者をプロジェクトに惹きつけるための推奨プラクティスについて説明します。

ドキュメンテーション

GitHub でホストされているオープンソース プロジェクトでは、明確で詳細なドキュメントが不可欠です。これにより、プロジェクトが他の開発者にとって理解しやすく、使用しやすくなり、コントリビュータとユーザの数を増やすことができます。優れたドキュメントには、プロジェクトの

目的、使用手順、依存関係や要件の概要が含まれています。また、コード、その動作、既知の問題や制限の詳細な説明も含まれる必要があります。明確なドキュメントを持つことで、他の開発者がプロジェクトを理解し、貢献することが容易になります。また、開発者がコードの動作をよりよく理解し、自信を持って変更できるようになるため、プロジェクトを長期的に維持することも容易になります。さらに、優れたドキュメントは、ユーザーがソフトウェアの使用方法を知り、遭遇する可能性のある問題のトラブルシューティングを容易にします。

オープンソース プロジェクトは、そのコミュニティのユーザーと開発者を支援するために、さまざまなタイプのドキュメントを提供できます。歴史的に、ドキュメントは改善が必要な分野でした。しかし、状況は改善され、多くのプロジェクトがプロジェクトのすべての分野をカバーする優れたドキュメントを持っています。

以下のサブセクションでは、文書化が不可欠な 3 つの主要分野を取り上げます。

1. プロジェクト
    a. 目的
    b. 管理
    c. コミュニティ組織
    d. リリース周期
    e. ロード マップと優先度
    f. ユース ケース
    g. FAQ
2. ユーザーのためのドキュメンテーション
    a. ユーザー ガイドとチュートリアル
    b. API ガイド
    c. アーキテクチャの概要
    d. インストール ガイド
    e. 機能追加の要望とセキュリティ脆弱性の報告プロセス
    f. 経験共有コーナー
3. 開発者のためのドキュメンテーション
    a. アーキテクチャの詳細と該当する場合コード サブシステム サービスへのマッピング
    b. 開発プロセス
    c. 参加について
    d. 参加のガイドライン
    e. 機能追加の要望プロセス
    f. パッチ登録プロセス
    g. 該当する場合、Signed-off-by プロセス
    h. 開発ガイドとチュートリアル
    i. API ガイド

推奨事項:

1. TODO Group[1] によって作成された REPOLINTER ツールを使用して、GitHub リポジトリ内の共通の問題を特定します。
2. 新しいコミュニティ メンバーをプロジェクトに歓迎し、プロジェクトの価値と開始方法を説明する `README.md` ファイルを追加します。
3. プロジェクトへのコントリビューション方法を説明する `CONTRIBUTING.md` ファイルを追加します。このファイルには、必要なコントリビューションの種類とプロセスの仕組みが説明されています。

4. CODEOWNERS ファイルを追加して、リポジトリ内のコードを担当する個人またはチームを定義します。
5. CODE_OF_CONDUCT.md ファイルを追加します。このファイルは、参加者の行動の基本ルールを設定し、友好的で歓迎される環境を促進するのに役立ちます。 CODE _ OF _ CONDUCT.md ファイルは、このプロジェクトが貢献することを歓迎するプロジェクトであることを示し、プロジェクトのコミュニティと関わるための基準を定義しています。
6. プロジェクトのセキュリティ問題を報告する方法をユーザーに提示する SECURITY.md ファイルを追加します。
7. リリース方法、周期、基準などに関する文書を提供します。
8. プロジェクト ガバナンスを文書化し、プロジェクトのリポジトリで利用できるようにします。

GitHub は、組織レベルで、その組織内で作成されたすべてのレポに、このようなコミュニティヘルスファイルを自動的に適用する機能を提供します。詳しくは、GitHub Docs をご覧ください。

サポート チャンネル

GitHub でオープンソース プロジェクトを維持するためには、優れたユーザー サポートが不可欠です。以下では、オープンソース プロジェクトのコミュニティに対してより良いサポートを提供するために、オープンソース プロジェクトに推奨されるプラクティスを提供します。

1. SUPPORT.md ファイルを作成します。このファイルには、プロジェクトに関するヘルプを取得する方法が記述されています。
2. 詳細でよく整理されたドキュメントを作成します:これには、プロジェクトのインストールと使用に関する手順、トラブルシューティングのヒント、および日常的な使用例を含めることができます。
3. イシュー トラッカーを使用します : これにより、ユーザーはバグを報告し、機能を要求し、ヘルプを求めることができます。迅速に対応し、正確で詳細な対応を提供することは、ユーザーとの信頼と信用を構築するのに役立ちます。
4. フィードバックや提案に対してオープンであること:ユーザーにフィードバックを提供するよう促し、調査やインタビューを通じて積極的にフィードバックを求めます。これは、ユーザーが何を求めているのかや、プロジェクトを改善する方法を理解するのに役立ちます。
5. 積極的かつ透過的なコミュニケーション：ブログ、ニュースレター、またはソーシャル メディアを使用して、プロジェクトの進捗状況、ロード マップ、および最新情報を共有することで、ユーザーは、何が起こっているのか、および将来の変更に対する計画を知ることができます。
6. GitHub Discussions を使用して、ユーザーと開発者をサポートし、関与させます。プロジェクトでは、GitHub Discussions を使用します。プロジェクトを中心としたコミュニティの構築を支援し、ユーザーがヘルプを入手して他のユーザと接続するための簡単な方法を提供します。
7. 検索とアクセスが容易なヘルプ センターを設定します:これは、ユーザーが一般的な質問に対する回答を検索し、サポート チケットを送信し、コア開発者に連絡できる一元化された場所です。
8. 非アクティブなリポジトリをアーカイブして、ユーザーと開発者がリポジトリがもはやサポートを受けていないことを知るようにします。
9. FUNDING.md ファイルを追加して、プロジェクトのサポート方法をユーザーに知らせることを検討します。

これらのプラクティスに従うことで、GitHub でホストされているオープンソース プロジェクトは、より良いサポートを提供し、より成功した持続可能なプロジェクトをもたらすことができます。

セキュリティ

GitHub の組織のセキュリティ対策を実装することは、組織のプロジェクトのコードとデータを保護するために不可欠です。以下では、GitHub の組織のセキュリティ対策に焦点を当てたいくつかの推奨事項を提供します。

1. 2 要素認証 (2FA)：すべての組織メンバーに対して 2FA を有効にすることで、組織のアカウントにセキュリティ レイヤーを追加できます。
2. アクセス コントロール：ロールやチームなどの GitHub の組み込みアクセス コントロール機能を使用して、組織のリポジトリへのアクセス権を持つユーザーと、そのユーザーが実行できるアクションを制限します。
3. GitHub Branch Protection Rules と CODEOWNERS ファイルを使用して、リポジトリに対するすべての変更が適切な担当者によってレビューされていることを確認します。
4. セキュアなコード レビュー：静的コード解析などのセキュリティ チェックを含むコード レビュー プロセスを実装して、コード内の潜在的な脆弱性を特定し、修正します。
5. 継続的インテグレーションとデプロイメント (CI/CD)：GitHub Actions や CircleCI などの CI/CD ツールを使用して、コードのビルド、テスト、デプロイを自動化し、セキュリティ問題の迅速な特定と修正を可能にします。
6. GitHub Code Scanning を使用して、コード内の脆弱性を見つけます。
7. 依存性セキュリティ スキャン ツールを使用する：GitHub の Dependabot や Snyk などのツールを実行して、パッケージ、ライブラリ、その他のサードパーティ コードの脆弱性を検出します。
8. 従業員のセキュリティ トレーニング：組織が提供するセキュリティ プラクティス トレーニングに登録します。
9. オープンソース プロジェクトの OpenSSF Best Practices Badge を取得し、維持します。OpenSSF Best Practices Badge は、セキュリティおよび脆弱性管理のベストプラクティスに従うオープンソース プロジェクトを認識し、推奨します。バッジを取得したプロジェクトは、セキュリティ ポリシー、脆弱性報告プロセス、および報告された脆弱性を処理する方法があることを証明します。バッジを取得することで、プロジェクトはセキュリティを真剣に受け止め、ユーザーと貢献者はプロジェクトが責任を持って脆弱性に対処することを信頼できることを示します。
10. プロジェクトの誰がセキュリティ問題を処理するかを特定します (チームの場合もあります)。SECURITY.md ファイルを作成し、セキュリティの脆弱性に関するアラートを受信するための電子メール アカウントを設定します。
11. GitHub Security Advisories を使用して、プロジェクトで見つかった脆弱性への対応を追跡、管理、公開します。
12. GPG を使用してローカルでのコミットに署名し、GitHub で検証済みとしてマークできるようにします。他の人は、変更が信頼できるソースからのものであることを確信できます。
13. OpenSSF scorecard は、メンテナンス担当者がセキュリティのベストプラクティスを改善するのに役立ち、オープンソースの使用者はソフトウェアの依存関係が安全かどうかを判断します。scorecard は、ソフトウェア セキュリティに関連するいくつかの経験則を評価し、各チェックに 0 ～ 10 のスコアを割り当てます。これらのスコアを使用して、特定の領域を理解し、プロジェクトのセキュリティ体制を強化できます。

これらのプラクティスを実装することで、組織や個人はコードやデータを保護し、GitHub 上のプロジェクトを安全かつ安全に保つことができます。セキュリティは継続的なプロセスであり、セキュリティ対策は継続的に更新し、強化する必要があります。

ライセンス

ライセンスに関する一般的な推奨事項

オープンソース プロジェクトのライセンスは、コードを使用、コピー、変更、および配布する権利を決定します。ライセンスの選択は、プロジェクトのオープン性を決定する上で重要です。オープンソース プロジェクトは、Open Source Initiative によって承認され、Free Software Foundation によって "free/libre" として認められたライセンスのみを使用することをお勧めします。このようなライセンスは、ソフトウェアを自由に使用、変更、および共有することを可能にします。Open Source Initiative によって承認されるためには、ライセンスが Open Source Definition(OSD) を満たしていることを確認するために、ライセンス レビュー プロセスを通過する必要があります。OSD と互換性のない他の多くのライセンスに遭遇する可能性もあるでしょう。これらのライセンスのほとんどは、一般的にソフトウェアの使用と配布に関する制限または制限を含む「ソース利用可能」ライセンスです。これらの制限は、多くの場合、ライセンスを OSD と互換性のないものにします。

正確なライセンス情報を提供するためにオープンソース プロジェクトが採用できる推奨プラクティスのリストを以下に示します。

1. 慣習的な場所 (リポジトリのルートなど) に `LICENSE.md` ファイルを含めます：このファイルには、ソフトウェアの使用、配布、および変更に関する条件を指定する必要があります。
2. OSI-approved open source licenses を使用する：これらのライセンスは広くレビューされ、使用されているため、開発者が理解し、準拠するためによりアクセスしやすくなっています。さらに、GitHub には、リポジトリを作成するときに最も一般的な OSI 承認ライセンスを簡単に使用できる便利な「ライセンスを選択する」機能があります。
3. `README.md` ファイルにライセンス情報を含める：`README.md` ファイルに、プロジェクトのライセンスを指定し、`LICENSE.md` ファイルへのリンクを提供する注記を含めます。
4. サードパーティのコードをマークする：プロジェクトにサードパーティのコードが含まれている場合は、そのコードを明確にマークし、サードパーティのコードのライセンスに関する情報を含めます。
5. `EADME.md` ファイルでライセンス バッジを使用する：`README.md` ファイルにライセンス バッジを含めると、プロジェクトで使用されているライセンスを簡単に確認できます
6. Software Package Data Exchange (SPDX) の短い形式の識別子を、各ソース コード ファイルの先頭ヘッダーのコメントに含めます。SPDX は、ソフトウェア パッケージに関連付けられたコンポーネント、ライセンス、および著作権を伝達するための標準形式です。オープンソース プロジェクトでは、SPDX を使用して、ライセンスおよび著作権情報をよりアクセスしやすく、機械で読み取りやすくしています。
7. GitHub がライセンスを認識していることを確実にする：選択したライセンスは、GitHub UI リポジトリのサマリー セクションに表示されます。表示されない場合は、GitHub がライセンスを検出または理解できなかったことを意味しています。残念ながら、それは他の人も同様に認識できないことを意味する可能性が高いでしょう。
8. ライセンス情報を定期的に確認および更新して、最新かつ正確な状態に保ちます。
9. 特にプロジェクトで DCO または CLA を使用する場合は、ライセンスの観点からプロジェクトに貢献するための明確なガイドラインを提供します (詳細については次のセクションを参照)。

これらのベストプラクティスに従うことで、GitHub 上のオープンソース プロジェクトは、正確なライセンス情報を提供し、開発者が法的要件を遵守するのを支援し、開発者がソフトウェアを理解して使用することを容易にします。これらのプラクティスは、法的またはコンプライアンスの問題を回避するのにも役立ちます。

DCO と CLA

DCO と CLA は、GitHub のオープンソース開発で使用されている 2 つの概念です。

DCO は Developer Certificate of Origin の略で、開発者がプロジェクトへの貢献が独自の作業であること、またはコードを登録するために必要な権限を持っていることを証明する方法です。これは、各コミット メッセージの最後に "Signed-off-by" 行を追加することによって行われます。この行は、開発者が DCO に同意し、開発者がコードの作成者であるか、またはコードを登録する権限を持っていることを示します。たとえば、Linux カーネル開発プロセスでは、すべての貢献者がコードにこの追加をする必要があります。これは、貢献者が DCO で概説されているようにコードを証明することを示します。署名は、貢献者が適切なオープンソース ライセンスの下で貢献を作成または受け取ったことを伝達します。これにより、ファイルに示されているライセンスの下でプロジェクトのコード ベースに貢献を組み込むことができます。DCO は、プロジェクトへの貢献のライセンスと出所に責任を負う一連の人々を確立します。

Contributor License Agreement(CLA) は、開発者またはその雇用者と、プロジェクトの所有者または保守担当者との間の法的合意です。CLA は、コードをプロジェクトに提供するための両当事者の権利と責任などの条件の概要を示します。オープンソース プロジェクトは、CLA を使用して、サードパーティによる貢献から生じる可能性のある法的問題から自身を保護します。また、企業や組織がプロジェクトを開発している場合や、外部の貢献者からの貢献の要請がある場合にも使用されます。

推奨事項：

1. DCO のコピーまたは参照を `CONTRIBUTING.md` ファイルに含めます。
2. 各コミットで "Signed-off-by:" タグを強制するようにボットを設定します。たとえば、GitHub DCO app をインストールして、この側面を管理することができます。
3. The Linux Foundation やその傘下の財団でホストされているプロジェクトの場合は、The Linux Foundation EasyCLA ツール (https://lfcla.com/) を使用して、コントリビューションを受け入れる前に署名付き CLA を強制します。
4. 創業者によってホストされ、管理されているプロジェクトの場合は、CLA Assistant の使用を検討してください。従業員は、CLA を扱うための最も適切な方法について、企業の顧問と相談することが推奨されます。

言語

The Linux Foundation では、世界中のオープンソース プロジェクトと協力しています。一部のプロジェクトは、英語が公用語ではない国からホスティングのために The Linux Foundation に参加しています。GitHub にあるプロジェクトのドキュメントの多くは、英語で提供されていない場合があります。GitHub で公開されているグローバルなユーザーを対象としたコンテンツには、常に英語を使用することをお勧めしています。ソフトウェア開発およびオープンソース コミュニティでは、英語が最も一般的な言語です。プロジェクトで英語のドキュメントが利用可能になれば、より多くのユーザーが簡単に理解し、利用できるようになります。

オープンソースの原則の採用

オープンソース開発とは、誰もがソース コードを自由に使用、変更、および配布できるソフトウェア開発への共同アプローチです。開発プロセスの決定的な特徴は、設計からリリースまでのライフ サイクル全体にわたるオープン性です。設計、計画、実装、テスト、およびリリースを含む開発プロセスのすべての側面は透明であり、より広範なコミュニティからの貢献に対して開かれています。これにより、個人および組織が専門知識、知識、およびリソースを共有して、多様なユーザーのニーズを満たす高品質のソフトウェアを作成できる、ソフトウェア開発への共同アプローチが可能になります。

大規模なコラボレーションを可能にするオープンソース開発のコア コンセプトには、ピア レビュー、早期リリース、頻繁なリリース、継続的なテストと統合などがあります。

- ピア レビューは、他の開発者がコードをレビューして批評することであり、ソフトウェアの品質を向上させるのに役立ちます。
- 早期リリースと頻繁なリリースは、ソフトウェアをできるだけ早くユーザーが利用できるようにし、頻繁な更新と改善を行うための戦略です。
- 継続的なテストと統合は、新しいコードの変更を絶えずテストし、統合することであり、開発プロセスの初期段階でバグを特定し、修正するのに役立ちます。

これらの概念は、ソフトウェア開発に対する協力的で、透過的で、効率的なアプローチを作成するために連携します。次のサブ セクションでは、これらの概念の大規模な実装を可能にするさまざまな機能を通じて、GitHub がこれらの中核的なオープンソース開発概念をどのようにサポートするかについて説明します。

ピア レビュー

ピア レビュー プラクティスは、スタイルのバリエーションを減らし、価値のある会話を促し、プロジェクトの品質基準を維持します。GitHub は、PR 機能を通じてそれを管理します。PR は、開発者がプロジェクトのコードに加えた変更を他の共同作業者によるレビューのために提出する方法です。PR の作成により、他の共同作業者は、コードがプロジェクトのメイン ブランチにマージされる前に、コードの変更をレビューし、コメントを残し、承認または変更を要求することができます。ピア レビューを確実にするために、PR を要求するように GitHub 組織の所有者が設定を調整し branch protection rules（組織または特定のレポートのいずれか）を有効化することをお勧めします。

早期かつ頻繁なリリース

「早期かつ頻繁なリリース」は、プロジェクトの頻繁かつ段階的なリリースを提唱する OSS 開発哲学です。GitHub は、ブランチとリリースを通じてオープンソース プロジェクトにおけるこの開発アプローチをサポートします。

進行中の作業には開発ブランチを使用し、安定版やプロダクション用のコードには別のリリース ブランチを使用することで、開発者は開発ブラ

ンチで新機能とバグ修正に取り組み、定期的に変更をリリース ブランチにマージします。

もう 1 つの方法は、開発者が作業中の機能やバグ修正ごとに新しいブランチを作成するフィーチャー ブランチを使用することです。フィーチャー ブランチの作業が完了すると、開発ブランチへのマージ、最終的にはリリース ブランチへのマージが行われます。

さらに、GitHub にはリリースと呼ばれる機能があり、ソフトウェア バージョンのパッケージング、タグ付け、配布が可能です。この機能は、コードの特定のバージョンをリリースとしてマークし、ユーザーがリリース バージョンをダウンロードして使用する方法を提供します。ユーザーはリリース ノートを表示し、各リリースで行われた変更を確認することもできます。

継続的なテストと統合

GitHub での継続的なテストと統合は、GitHub Actions やその他の CI ／ CD ツールなどのツールを組み合わせて管理されます。

GitHub Actions は、ブランチや PR へのプッシュなど、特定のイベントによってトリガーされるカスタム アクションを作成することで、開発者がビルドとテストのワーク フローを自動化できる機能です。GitHub Actions を使用すると、開発者は、特定のイベントが発生したときに実行される自動テストと統合ステップを設定できます。たとえば、PR の作成では、開発者は GitHub Actions と Status Checks を設定して、コード変更に対して一連のテストを実行し、メイン ブランチにマージする前に期待どおりに動作することを確認できます。

継続的なテストと統合を管理するもう 1 つの方法は、CircleCI、TravisCI、Jenkins などのサードパーティの CI/CD ツールを使用することです。これらのツールは GitHub と統合され、開発者はリポジトリにプッシュされたコード変更があるたびに自動的に実行されるテストと統合ステップのパイプラインを設定できます。たとえば、開発ブランチにプッシュされたコード変更がある場合、CI ／ CD ツールは自動的にテストを実行し、コードをビルドしてステージング環境にデプロイし、次に追加のテストを実行して、本番環境にデプロイする前にすべてが期待どおりに動作することを確認できます。

さらに、多くのサードパーティ ツールでは、ユニット テスト、統合テスト、エンド ツー エンド テストなど、さまざまなテスト オプションが提供されているため、アプリケーションのさまざまな側面をテストできます。

GitHub Actions とサードパーティの CI ／ CD ツールの両方の設計により、継続的なテストと統合プロセスが可能な限りシームレスで効率的になります。

結論

GitHub は、コード コラボレーション、コミュニティ構築、プロジェクト管理のための一元化されたプラットフォームを提供することによって、オープンソース開発において不可欠な役割を果たし、開発者がオープンソース プロジェクトで協力し、それらを実現することを容易にします。これは、開発者がオープンソース エコシステムへの貢献を共有、協力、および向上できるようにする重要なツールです。

GitHub でオープンソースの開発活動を始めるには、いくつかのステップを踏むことができます：

1. 関心のあるプロジェクトを見つけることから始めます。透過的な開発プロセスと OSI 承認ライセンスを持つプロジェクトを探します。
2. プロジェクトのドキュメントとガイドラインを読みます。プロジェクトの目標と組織を理解していることを確認します。
3. GitHub でプロジェクトをフォークして、作業して変更を登録できるプロジェクトのコピーを作成します。
4. 小さな変更を行い、それらの上流のプロジェクトへの PR を作成します。このプロセスにより、プロジェクトの管理者はあなたの作業を確認し、フィード バックを提供できます。
5. フィードバックに対してオープンであり、変更を行うことをいとわないでください。オープンソース開発は協力的であり、優れたコミュニケーションが不可欠です。
6. 質問に答え、サポートを提供し、新しいコントリビューターを支援することで、プロジェクトのコミュニティに貢献します。
7. メンターシップの機会を探しましょう；あなたが尊敬する仕事をしているメンテナを見つけたら、彼らに連絡を取り、あなたをメンターしてくれるかどうか尋ねてください。
8. フォークを元のリポジトリと定期的に同期させることで、フォークをメイン プロジェクトに合わせて最新の状態に保ちます。

オープンソース開発に GitHub を使用するためのベストプラクティスの概要を提供する上で、この資料が役立つことを願っています。オープンソース開発の作業を開始したい場合は、GitHub にアクセスしてアカウントを設定し、そして開始しましょう！

脚注

1 TODO Group は、The Linux Foundation がホストする実践者のオープン コミュニティです。TODO Group の目的は、オープンソースの取り組みやプログラム オフィスを成功させ、効果的に運営するために、知識の創造と共有、プラクティスやツールなどに関する協業を行うことです。

謝辞

Hilary Carter 氏、Jason Perlow 氏、Melissa Schmidt 氏、Barry Hall 氏に感謝の意を表します。序文に貢献してくれた Jeff McAffer 氏と、この論文の改善に重要な役割を果たしてくれた彼の洞察と詳細なフィードバックに特に感謝します。皆さんの時間と努力に感謝します。また、著者は LinkedIn を通じて Sumanta Mukhopadhyay 氏、Phil Coval 氏、Andreas Fehlner 氏から受けた貢献にも感謝の意を表します。

Linux Foundation リソース

- E-book: A Road Map to Improve the Effectiveness and Impact of Enterprise Open Source Development
- E-book: A Deep Dive into Open Source Program Offices: Structure, Roles, Responsibilities, and Challenges
- E-book: A Guide to Enterprise Open Source
- E-book: Open Source Audits in Merger and Acquisition Transactions
- Linux Foundation Enterprise Guides
- Linux Foundation Open Source Compliance Program
- TODO Group
- The Software Package Data Exchange®
- Linux Foundation Training & Certification
- Linux Foundation Events

フィードバック

著者は、いかなるスペル ミスまたは可能性のあるミスについても事前に謝罪し、改善のための訂正および提案を受けることに感謝します。

著者について

Dr.Ibrahim Haddad は、Linux Foundation AI & Data および PyTorch Foundation のエグゼクティブ ディレクターです。オープンソース AI プラットフォームを進化させるためのベンダー ニュートラルな環境の促進に重点を置いています。その経歴を通じて、Ericsson Research、Open Source Development Labs、Motorola、Palm、Hewlett-Packard、Samsung Research、および Linux Foundationで技術およびポートフォリオ管理の役割を果たしてきました。Haddad は、Concordia University(カナダ、モントリオール) でコンピューター サイエンスの博士号を取得し、優等で卒業しました。

LinkedIn: @ibrahimhaddad　Twitter: @IbrahimAtLinux　Website: IbrahimAtLinux.com

この文書は、以下のレポートの参考訳です。
Ibrahim Haddad, Ph.D., “Recommended Practices for Hosting and Managing Open Source Projects on GitHub,” foreword by Jeff McAffer, The Linux Foundation, March 2023

翻訳協力：橋本修太

THE LINUX FOUNDATION | Research

2021 年に設立された Linux Foundation Research は、オープンソース コラボレーションの規模の拡大を調査し、新しいテクノロジー トレンド、ベストプラクティス、オープンソース プロジェクトの世界的な影響についての洞察を提供します。プロジェクトのデータベースとネットワークを活用し、定量的・定性的な方法論のベストプラクティスに取り組むことで、Linux Foundation Research は、世界中の組織のために、オープンソースの洞察を得るための最適なライブラリーを構築しています。

この著作物を引用する場合は、以下のように記載してください。
Ibrahim Haddad, Ph.D., “Recommended Practices for Hosting and Managing Open Source Projects on GitHub,” foreword by Jeff McAffer, The Linux Foundation, March 2023