Cours Logiciels Libres

Démarrer un projet de logiciel libre

Théo Zimmermann

Télécom Paris, Institut Polytechnique de Paris

Vendredi 17 février 2023

Référence

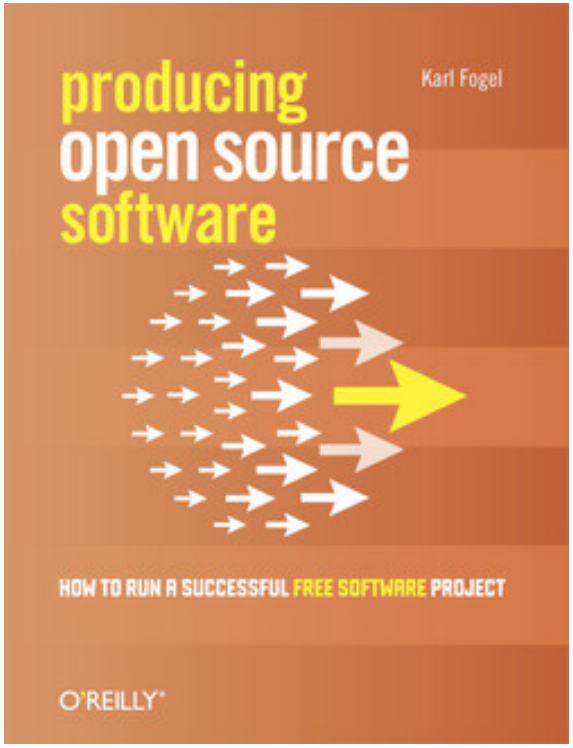

Figure 1: Producing Open Source Software, par Karl Fogel

- Première édition (2005).
- Nouvelle édition (2021) sur https://producingoss.com/.

Dans quel contexte ?

Vous avez codé / commencé à coder :

Dans quel contexte ?

Vous avez codé / commencé à coder :

- Une application pour votre **usage personnel** / un but **non-commercial** (recherche, éducation, projet associatif, etc.).
- Des fonctions génériques qui pourraient former une **bibliothèque réutilisable** dans d'autres projets écrits dans le même langage de programmation.
- Un **logiciel utile en appui** à votre entreprise (hors de son cœur de métier). Exemple : GitHub https://tom.preston-werner.com/2011/11/22/open-source-everything.html
- Un logiciel pour un client qui n'a **pas besoin d'en avoir l'usage exclusif**.
- Le principal **produit logiciel** de votre entreprise (moins évident mais c'est possible aussi, exemple : RedHat).

Pourquoi faire ?

Pour :

Pourquoi faire ?

Pour :

- En faire bénéficier le plus grand monde (**générosité**).
- Pouvoir vous-même **réutiliser** le code dans d'autres projets.
- Rendre le code **auditable**.
- **Satisfaire** les développeurs de votre entreprise / aider au recrutement / **améliorer l'image** de votre entreprise.
- Recevoir des **contributions** / améliorer plus vite le logiciel.
- Créer une **alternative libre** à un logiciel propriétaire / réduire les coûts / créer un **standard** commun.
- Satisfaire la **demande d'un client**.

Comment faire ?

Comment faire ?

- Vérifier si ce genre de logiciel libre **n'existe pas** déjà.

Comment faire ?

- Vérifier si ce genre de logiciel libre **n'existe pas** déjà.
- Ouvrir le code **dès le début**.

Comment faire ?

- Vérifier si ce genre de logiciel libre **n'existe pas** déjà.
- Ouvrir le code **dès le début**.
- Utiliser un gestionnaire de version (git) et une forge (par exemple GitHub).
- Inclure la documentation dans le **même dépôt** que le code.
- Utiliser des **outils standards** (compilation, tests, etc.).

Comment faire ?

- Vérifier si ce genre de logiciel libre **n'existe pas** déjà.
- Ouvrir le code **dès le début**.
- Utiliser un gestionnaire de version (git) et une forge (par exemple GitHub).
- Inclure la documentation dans le **même dépôt** que le code.
- Utiliser des **outils standards** (compilation, tests, etc.).
- Présenter quelque chose d'**utile** (ou au moins de convaincant).
- Présenter l'information de manière **approchable**.
- Choisir quand **annoncer** le projet.

Ouvrir le code dès le début. Pourquoi ?

Ouvrir le code dès le début. Pourquoi ?

- Plus on attend, plus c'est difficile (**nettoyage à prévoir**).
- Le projet ne sera **jamais suffisamment prêt**.
- Vous pourriez recevoir des contributions **plus tôt que prévu**.
- Ouvrir le code **n'oblige pas à** :
  - écrire de la documentation,
  - être disponible pour répondre aux questions,
  - corriger les bugs.

Ouvrir le code dès le début. Comment ?

Ouvrir le code dès le début. Comment ?

- Choisir une **licence** :
  - **Avant de recevoir des contributions**, donc dès le début.
  - Si vous ne savez pas quelle licence choisir, commencez par une licence permissive (MIT) : il est **plus facile de changer** de permissif vers moins permissif que dans l'autre sens (MIT $\rightarrow$ Apache 2.0 $\rightarrow$ MPL 2.0 $\rightarrow$ LGPL 3.0 $\rightarrow$ GPL 3.0 $\rightarrow$ AGPL 3.0).

Ouvrir le code dès le début. Comment ?

- Choisir une **licence** :
  - **Avant de recevoir des contributions**, donc dès le début.
  - Si vous ne savez pas quelle licence choisir, commencez par une licence permissive (MIT) : il est **plus facile de changer** de permissif vers moins permissif que dans l'autre sens (MIT $\rightarrow$ Apache 2.0 $\rightarrow$ MPL 2.0 $\rightarrow$ LGPL 3.0 $\rightarrow$ GPL 3.0 $\rightarrow$ AGPL 3.0).
- Choisir un **nom** :
  - Vous pouvez toujours dire que c'est un **nom de code**.
  - Vous pourrez en changer mais n'attendez pas trop longtemps (par exemple, première sortie de version).

Ouvrir le code dès le début. Comment ?

- Choisir une **licence** :
  - **Avant de recevoir des contributions**, donc dès le début.
  - Si vous ne savez pas quelle licence choisir, commencez par une licence permissive (MIT) : il est **plus facile de changer** de permissif vers moins permissif que dans l'autre sens (MIT $\rightarrow$ Apache 2.0 $\rightarrow$ MPL 2.0 $\rightarrow$ LGPL 3.0 $\rightarrow$ GPL 3.0 $\rightarrow$ AGPL 3.0).
- Choisir un **nom** :
  - Vous pouvez toujours dire que c'est un **nom de code**.
  - Vous pourrez en changer mais n'attendez pas trop longtemps (par exemple, première sortie de version).
- Utiliser une forge (de type GitHub) : pour le code et pour la documentation (même dépôt).

Ouvrir le code dès le début. Comment ?

- Choisir une **licence** :
  - **Avant de recevoir des contributions**, donc dès le début.
  - Si vous ne savez pas quelle licence choisir, commencez par une licence permissive (MIT) : il est **plus facile de changer** de permissif vers moins permissif que dans l'autre sens (MIT $\rightarrow$ Apache 2.0 $\rightarrow$ MPL 2.0 $\rightarrow$ LGPL 3.0 $\rightarrow$ GPL 3.0 $\rightarrow$ AGPL 3.0).
- Choisir un **nom** :
  - Vous pouvez toujours dire que c'est un **nom de code**.
  - Vous pourrez en changer mais n'attendez pas trop longtemps (par exemple, première sortie de version).
- Utiliser une forge (de type GitHub) : pour le code et pour la documentation (même dépôt).
- Si vous êtes plusieurs :
  - Tenir toutes les discussions en public.
  - Pratiquer la revue de code (pull requests).

Fonctionnalités fournies par GitHub

Fonctionnalités fournies par GitHub

- **Dépôt** sous git avec **accès** web + système de **pull requests**.
- Collaborateurs (plusieurs niveaux de permissions).
- Bug tracker (“**Issues**”) avec liens inter-projets *optionnel*.
- Modèles pour les tickets et les pull requests *optionnels*.
- Forum + système de questions / réponses (“**Discussions**”) *optionnel* et *nouveau*.
- **Wiki** *optionnel*.
- Système de publication de nouvelles versions (“**Releases**”).
- Site web (“**GitHub Pages**”) *optionnel*.
- Système d'intégration continue / déploiement continu (“**GitHub Actions**”) *optionnel*.
- **API** permettant d'ajouter des services / de l'automatisation.
- **Organisations** pour regrouper plusieurs dépôts *optionnel*.
- **Équipes** de mainteneurs *optionnel*.
- Discussions privées pour les équipes *optionnel*, **à éviter**.
- ~~Messagerie instantanée~~ (IRC, Discord, Slack, Gitter, Zulip).

Choix technologiques

Une règle clé : **réduire les besoins d'apprentissage**.

Pourquoi ?

Choix technologiques

Une règle clé : **réduire les besoins d'apprentissage**.

Pourquoi ?

Moins de choses à apprendre =

- plus de **contributeurs potentiels**,
- qui sont plus vite **productifs**.

Choix technologiques

Une règle clé : **réduire les besoins d'apprentissage**.

Pourquoi ?

Moins de choses à apprendre =

- plus de **contributeurs potentiels**,
- qui sont plus vite **productifs**.

Donc :

- Viser les choix les plus **standards** possibles (gestionnaire de version, langage, gestionnaire de paquets, compilation, tests).
- **Sans se forcer** à utiliser des technologies que vous n'aimez pas ou qui vous rendent moins productifs.
- **Éviter les solutions "maison"** sans réelle justification.

Documentation

Écrire de la documentation vous sera aussi **utile personnellement**.

C'est normal que ce soit toujours un **travail en cours**.

Une partie de la documentation sera écrite **en réaction** à des besoins (par exemple de nouveaux contributeurs).

Documentation

Écrire de la documentation vous sera aussi **utile personnellement**.

C'est normal que ce soit toujours un **travail en cours**.

Une partie de la documentation sera écrite **en réaction** à des besoins (par exemple de nouveaux contributeurs).

- Approchable = différents **niveaux de présentation** :
  - mission en **une phrase**,
  - résumé en **un paragraphe**,
  - démo courte (vidéo, captures d'écran, **exemples**),
  - liste des **fonctionnalités** disponibles,
  - liste des fonctionnalités à venir / idéales.

Documentation

Écrire de la documentation vous sera aussi **utile personnellement**.

C'est normal que ce soit toujours un **travail en cours**.

Une partie de la documentation sera écrite **en réaction** à des besoins (par exemple de nouveaux contributeurs).

- Approchable = différents **niveaux de présentation** :
  - mission en **une phrase**,
  - résumé en **un paragraphe**,
  - démo courte (vidéo, captures d'écran, **exemples**),
  - liste des **fonctionnalités** disponibles,
  - liste des fonctionnalités à venir / idéales.
- Types de documentation :
  - pour les utilisateurs (essentiel),
  - pour les développeurs (idéal).

Annoncer le projet (progressivement)

- Si vous ne parlez à personne de votre projet, il est peu probable que quiconque le remarque.

Annoncer le projet (progressivement)

- Si vous ne parlez à personne de votre projet, il est peu probable que quiconque le remarque.
- Vous n'êtes **pas obligés d'attendre** que le projet soit suffisamment avancé pour commencer à en parler.
  - Vous pouvez en parler à des connaissances qui pourraient être intéressées.
  - Vous pouvez en parler à des inconnus (sur des forums) s'il répond à un **besoin exprimé**.

Annoncer le projet (progressivement)

- Si vous ne parlez à personne de votre projet, il est peu probable que quiconque le remarque.
- Vous n'êtes **pas obligés d'attendre** que le projet soit suffisamment avancé pour commencer à en parler.
  - Vous pouvez en parler à des connaissances qui pourraient être intéressées.
  - Vous pouvez en parler à des inconnus (sur des forums) s'il répond à un **besoin exprimé**.
- **Annonce officielle** (sur des forums, des mailing lists, etc.) :
  - Peut avoir lieu à l'occasion de la sortie de la **première version**.
  - Ou bien à l'occasion de la première **version stable**.
  - Ou bien cela peut être l'annonce du **démarrage du projet** (avant que la première version soit prête) si vous pensez pouvoir recruter des contributeurs dès le début.