Pourquoi je contribue et ne contribue pas au logiciel libre

Pour un débutant participer au logiciel libre peut être si intimidant qu'on n'hésite pas à évoquer « un aquarium à requins » pour qualifier la communauté !

Un blogueur explique pourquoi il ne contribue pas au logiciel libre (alors qu'au fond de lui il le souhaite sincèrement). Un autre lui répond, en théorie mais aussi en pratique en s'appuyant sur GitHub (qui a le vent en poupe actuellement chez les développeurs). Telle est la petite passe d'armes que nous vous proposons traduite ci-dessous[1].

En ce qui nous concerne, c'est aussi pour cela que l'on a publié notre framabook Produire du logiciel libre. Afin de participer à ce qu'il y ait de plus en plus de développeurs francophones, notamment parmi les plus jeunes qui ne reçoivent pour le moment aucune sensibilisation ou formation pendant leur cursus scolaire.

Pourquoi je ne contribue toujours pas à l'open source

Why I still don't contribute to open source

The Daily Flux – 3 mai 2011 – Brandonhays.com
(Traduction Framalang : Pandark)

Je suis tellement hypocrite. Il y a quelques mois, je me demandais dans un billet comment surmonter ma peur de

contribuer aux logiciels open source ?[1]?.

Depuis, je n'ai toujours pas vraiment participé. Sur Twitter, j'ai écrit que les FOSS ressemblent à un aquarium de requins pour les newbies, et il faut que je le confirme.

Le fait est que je contribue activement d'une manière ou d'une autre à plusieurs projets open source. Cependant, je me sens toujours extérieur au projet, car mes contributions ne sont généralement pas liées au code. Alors pourquoi est-que je ne m'implique pas complètement dans le FOSS (et je pense, beaucoup d'autres comme moi) ?

Au risque de prêter aux autres mon ressenti personnel, j'aimerais vous faire part des obstacles qui peuvent, selon moi, intimider les nouveaux devs qui voudraient contribuer à des logiciels open source.

I**l n'y a pas de certification, de cérémonie ou de badge du mérite** disant « Tu es prêt à contribuer au FOSS ». (Il y en a cependant un pour après)

Il n'est pas évident de savoir par où commencer. D'après ce que j'entends, beaucoup de contributions aux FOSS surviennent parce que quelqu'un a besoin d'une fonctionnalité qui n'existe pas dans un logiciel, ou trouve un bug. Il peut proposer une procédure de test reproductible, voire un patch. Dans mon utilisation quotidienne, je ne croise pas beaucoup de ces situations. Il n'y a pas beaucoup de devs qui agitent les bras en demandant spécifiquement de l'aide sur un projet, et encore moins qui voudraient prendre un nouveau développeur sous leur aile.

Les règles de participation (guidelines) rendent souvent la vie d'un mainteneur plus facile, et compliquent la mienne. Oui, maintenir un projet open source est une tâche ardue et ingrate. Cependant, j'ai vu des règles/lignes de conduite pour contribuer qui transformaient une simple idée de correction en un mur de brique bureaucratique digne de Microsoft. La page

d’accueil aux contributions accompagnée d’un tutoriel vidéo de Wayne Seguin est une exception remarquable à cela.

L’open source est pour les gens qui sont meilleurs que moi. J’ai bien conscience que c’est une excuse pour ne pas me lancer, mais je ne suis simplement pas à l’aise de me retrouver à un endroit où je pourrais publier des logiciels suffisamment bons pour que de véritables *développeurs* les utilisent.

Essayer de contribuer et échouer me donne le sentiment d’être stupide. J’ai déjà soumis plusieurs requêtes de pull et aucune n’a été acceptée, sans commentaire expliquant pourquoi. C’est comme si l’univers confirmait que oui, je suis un idiot, et mon « aide » n’est pas utile. Quelle perte de temps profondément déprimante !

J’ai pas le temps. J’ai des enfants, un nouveau boulot, et un nombre grandissant de responsabilités. Cela me prend entre 3 et 10 fois plus de temps pour écrire du code qu’un développeur plus expérimenté. Maintenant, mes contributions non liées au code mangent le temps que je passais à coder. Oui, tout le monde a la même excuse, du genre qui se dissipe si les autres excuses disparaissent, mais ça vaut le coup de le mentionner.

C’est une activité solitaire. Je pense que la plupart des gens comprennent ces choses par eux-même, et que ce serait donc un peu trop demander que d’attendre d’être pris par la main. Mais est-ce vraiment une démarche spirituelle où personne ne peut vous accompagner, de crainte que vous n’appreniez rien ?

Donc oui, le FOSS peut sembler intimidant, voire autant qu’un aquarium de requins. Je n’ai pas toutes les réponses à ces problèmes, mais je voudrais voir plus de mainteneurs cherchant des contributions avec une certaine spécificité, et répondant ensuite aux requêtes de pull, un appel pour des cas de tests supplémentaires, des corrections de bugs et, oui, de la documentation.

Github a beau être on ne peut plus ouvert, il n'y a pas de système type Quora/StackExchange qui permette de savoir quel projets ont besoin de quelque chose qui correspond à ce que vous pouvez faire. Ça pourrait être une bonne fonctionnalité.

Toi (oui, toi !), tu devrais contribuer à l'open source

You (yes, you!) should contribute to open source

Steve Klabnik – 10 mai 2011 – TheChangelog.com
(Traduction Framalang : Pandark)

Si vous lisez ce blog, vous vous souciez évidemment de l'open source. Si vous n'avez jamais contribué à un projet open source, cependant, vous êtes peut-être frileux à ce propos. Donc, inspiré par le concours de documentation Ruby 1.9.3, j'ai écrit un billet pour mon blog à propos de la manière de contribuer à la documentation de Ruby. J'ai reçu des retours comme celui-ci :

@steveklabnik Hé, c'est génial. Il est temps pour moi de m'engager et de me mettre au boulot. Merci pour la motivation supplémentaire !

Je me suis donc dit que quelque chose de plus général pourrait vous encourager à vous impliquer dans n'importe lequel des projets open source que vous utilisez, même si ce n'est pas en Ruby. Tout projet peut avoir besoin d'un coup de main supplémentaire, en particulier les petits.

Un petit aparté à propos du fait d'être frileux.

Si vous ne contribuez pas parce que vous pensez que vous n'êtes pas prêt, ne vous inquiétez pas pour ça ! Je sais que c'est plus facile à dire qu'à faire, mais vraiment, vous êtes prêt. Un de mes amis a publié un article à propos des raisons pour lesquelles il ne contribue pas, et je suis sûr que beaucoup de personnes partagent ce genre de peurs. Greg Brown a répondu et a dissipé certaines de ses inquiétudes, mais la plupart des gens auxquels j'ai parlé s'y refusent principalement pour deux raisons :

- C'est trop dur
- Je ne suis pas assez bon pour contribuer
- Je n'ai pas le temps

Parlons de chacun de ces points dans l'ordre inverse. C'est vrai, vous pouvez avoir une vie remplie. Je ne connais pas votre emploi du temps personnel. Cependant, je suis sûr que vous pouvez trouver une heure ou deux, peut-être un week-end ? Il n'en faut pas plus pour commencer. La plupart des projets sont construits sur la base de milliers de minuscules commits. Vous n'avez pas besoin de faire une grosse contribution, même les petites sont importantes.

Si vous avez peur que la qualité de votre code ne soit pas suffisante, eh bien la seule manière de vous améliorer est de pratiquer. Alors lancez-moi cet éditeur et soumettez un patch ! En général, si quelque chose ne va pas dans votre soumission, il y aura une discussion à son propos sur GitHub et tout le monde peut y apprendre quelque chose.

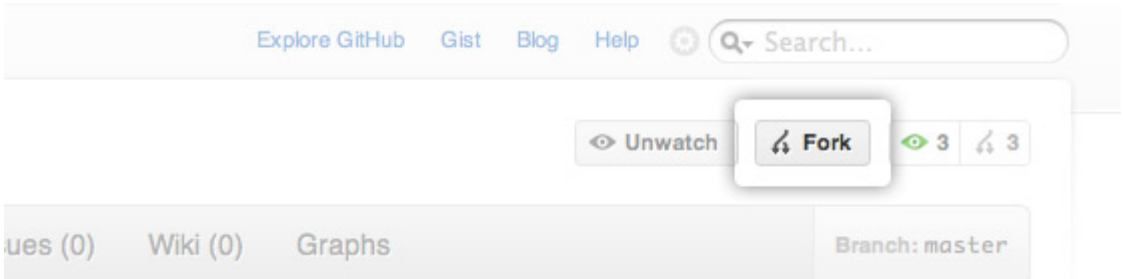
Prenez cette demande de pull, par exemple. À l'origine, Colin a soumis un patch qui faisait un lien vers la mauvaise url ; wilkie l'a mentionné, et Colin a mis le code à jour. Cela sera intégré dès que j'aurai fini d'écrire ce billet pour The Changelog. Mais c'est généralement ce qui arrive si votre première proposition est un peu inexacte. N'ayez pas peur ! C'est comme ça que l'on a tous appris, les uns des autres.

Cette lamentation « c'est trop dur » débouche souvent sur un « je ne suis pas assez bon ». Cela peut aussi arriver si vous essayez de contribuer à un gros projet ayant beaucoup de règles. Les lignes de conduite pour contribuer, obligation de relecture du code, mise à jour des fichiers AUTHORS et CHANGELOG… les gros projets doivent avoir des procédures pour gérer le grand nombre de contributeurs, mais cela peut certainement créer une barrière à l'entrée pour les nouveaux venus. Si ces procédures vous intimident, j'ai une suggestion : commencez petit ! Les petits projets ont généralement peu, voire pas du tout de procédure. De plus, vous vous sentirez incroyablement bien. Pensez à ça : Python reçoit un tas de patchs tous les jours, mais si vous avez un petit outil que vous avez écrit sur GitHub, et que soudainement vous recevez un courriel « Hé, quelqu'un a un patch pour vous, » je parie que vous en serez rudement content.

Le B.A BA

Lorsque l'on contribue à un projet open source sur GitHub, il y a un processus que presque tous les projets suivent. Trois étapes : fork, commit, demande de pull.

GitHub rend l'étape du fork très simple. Cliquez simplement sur le bouton « fork » trouvé sur la page de n'importe quel projet. Utilisons Ruby comme exemple. La page du projet est ici. Vous pouvez voir le bouton fork en haut à droite. Il ressemble à ceci :



Cliquez dessus, et vous verrez certaines « hardcore forking action, » puis vous serez dans votre propre fork ! C'est votre

propre version du projet, et elle apparait sur votre page GitHub. Par exemple, voici mon fork de Ruby. Vous verrez une URL sur la page, qui vous permettra de cloner ce projet lui-même.

```
$ git clone git@github.com:steveklabnik/ruby.git
```

Cela crée un répertoire « ruby » avec tout le code à l'intérieur. Ensuite, ajouter un lien vers le projet parent pour pouvoir suivre les modifications qu'il fait.

```
$ cd ruby
$ git remote add upstream https://github.com/ruby/ruby.git
$ git fetch upstream
```

À partir de maintenant, à n'importe quel moment, nous pouvons récupérer les modifications du dépôt Ruby principal en faisant un rebase :

```
$ git rebase upstream/master
```

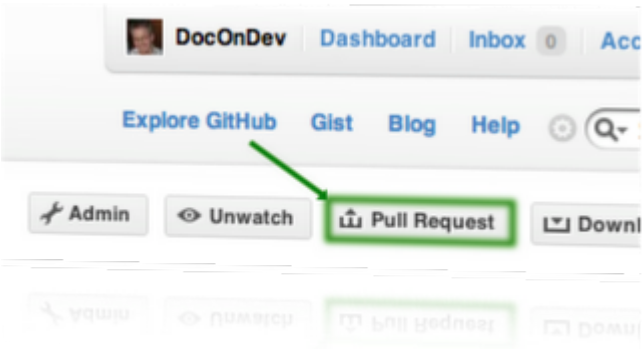
Une petite remarque : ruby continue d'utiliser à la fois svn subversion et git, ils appellent donc leur branche maîtresse trunk. Si vous faites cela pour Ruby, vous devrez faire git rebase upstream/trunk. Maintenant que vous avez cloné, vous pouvez faire votre boulot ! J'aime travailler dans des branches par fonctionnalités, parce que cela rend les choses plus propres et jolies, et que je peux travailler sur deux fonctionnalités à la fois.

```
$ git checkout -b feature/super-cool-feature
$ vim something
$ git add something
$ git commit -m "Fixed something in something"
```

Une fois que vous avez obtenu des commits qui fixent votre problème, envoyez les (faites un push) sur GitHub :

```
$ git push origin feature/super-cool-feature
```

Ensuite, vous cliquez sur le bouton pull request :



Choisissez votre branche, modifiez la description comme vous le souhaitez, et vous êtes prêt à vous lancer ! Le mainteneur du projet y jettera un coup d'œil et vous aurez peut-être droit à une discussion, et bientôt vous aurez quelque chose accepté quelque part !

À quoi devrais-je contribuer ?

Le meilleur moyen de contribuer est d'aider un projet que vous utilisez effectivement. De cette manière, vous arriverez à tirer profit du fruit de votre labeur. Vous serez plus motivé, vous comprendrez déjà le projet et ce qu'il fait, ce qui vous rendra tout ça plus facile.

Si vous ne voulez pas ou ne pouvez pas trouver comment fonctionne quelque chose que vous utilisez, le deuxième meilleur moyen est de commencer à utiliser de nouveaux logiciels ! Continuez à lire The Changelog et choisissez un projet qui a l'air intéressant, utilisez le quelques semaines, puis contribuez !

Nous sommes tous dans le même bateau

J'espère que ceci vous encouragera à vous salir les mains, vous retrousser les manches, et contribuer. Même le plus petit des patchs est important, alors s'il vous plaît, trouvez un moment dans votre emploi du temps, choisissez un projet et faites un essai. Mais attention, vous pourriez vite vous retrouver accro !

Notes

[1] Crédit photo : Yasuhiro (Creative Commons By)