Chouchoutez vos contributeurs et contributrices !

Le groupe Framalang a traduit l'article de Julien, qui a listé tous les moyens de se tirer une balle dans le pied quand on coordonne un projet libre.

Apprenez à les éviter !

Halte à la stratégie de l'échec !

Conduite de projets Open Source : 10 erreurs à éviter

Article original : https://julien.danjou.info/blog/2016/foss-projects-management-bad-practice

Auteur : Julien Danjou(CC-BY-SA)
Traduction : lyn., KoS, AlienSpoon, David 5.1, Simon, goofy, Slimane Aguercif, Fred, galadas, terhemis, gégé

Il y a quelques semaines, lors de l'*OpenStack Summit* (réunion annuelle des contributeurs à *OpenStack*, NDT), j'ai eu l'occasion de discuter de mon expérience de la conduite de projets *Open Source*. Je me suis rendu compte qu'après avoir fait partie de plusieurs communautés et beaucoup contribué pendant des années, je pourrais faire profiter les nouveaux venus de conseils et d'avis expérimentés.

Il existe une foule de ressources disponibles expliquant comment conduire un projet *Open Source*. Mais aujourd'hui, je voudrais aborder ce sujet sous un angle différent, en insistant sur ce qu'il convient de ne pas faire avec les personnes qui y participent. Je tire cette expérience des nombreux projets *Open Source* auxquels j'ai participé ces dernières années. Je vais décrire, sans ordre particulier, quelques mauvaises pratiques que j'ai constatées, en les illustrant par des exemples concrets.

1. Considérer les contributeurs comme une nuisance

Quand les informaticiens sont au travail, qu'ils soient chargés du développement ou de la maintenance d'un logiciel, il est une chose dont ils n'ont pas besoin : du travail supplémentaire. Pour la plupart d'entre eux, la première réaction à toute contribution externe est : « Zut, du travail en plus ». Et c'est effectivement le cas.

Par conséquent, certains développeurs essayent d'éviter ce travail supplémentaire : ils déclarent ne pas vouloir de contributions, ou font en sorte que les contributeurs se sentent indésirables. Cela peut prendre de nombreuses formes, comme les ignorer ou être désagréable avec eux. Ainsi, les développeurs évitent d'avoir à traiter le surplus de travail qu'on leur met sur le dos.

C'est une des plus grandes erreurs que l'on peut commettre, et une vision fausse de l'*Open Source*. Si des gens vous apportent du travail supplémentaire, faites tout ce que vous pouvez pour bien les accueillir afin qu'ils continuent à travailler avec vous. Il s'agit peut-être de ceux qui prendront votre relève d'ici peu. Préparez votre future retraite.

Prenons le cas de mon ami Gordon, que j'ai vu démarrer en tant que contributeur sur *Ceilometer* en 2013. Il examinait très bien le code, si bien qu'il me donnait en fait davantage de travail : non seulement en détectant les anomalies dans mes contributions, mais aussi en me faisant vérifier les siennes. Au lieu de m'en prendre à lui pour qu'il arrête de me faire retravailler mon code et vérifier ses correctifs, j'ai proposé qu'on lui fasse davantage confiance en l'ajoutant officiellement à l'équipe des correcteurs sur un des projets.

Et s'ils ne font pas cette première contribution, ils ne feront pas non plus la seconde. En fait, ils n'en feront aucune ; et ces projets perdraient alors leurs futurs correcteurs.

2. Ne confier aux autres que le sale boulot

Lorsque de nouveaux contributeurs se présentent pour travailler sur un certain projet, leurs motivations peuvent être très diverses. Certains sont des utilisateurs, d'autres veulent simplement voir ce que c'est de contribuer. Ressentir le frisson de la participation, comme un simple exercice ou avec la volonté d'apprendre afin de contribuer en retour à l'écosystème qu'ils utilisent.

En général, les développeurs confient le sale boulot à ces personnes, c'est à dire des tâches sans intérêt, à faible valeur ajoutée, et qui n'auront sans doute aucun impact direct sur le projet.

Pour certains, ce n'est pas grave, pour d'autres, ça l'est. Certains seront vexés de se voir confier du travail sans grand intérêt, alors que d'autres seront heureux de le faire pourvu qu'on leur témoigne de la reconnaissance. Soyez sensible à cela, et félicitez ces personnes. C'est la seule manière de les garder dans le projet.

3. Mépriser les petites contributions

Quand le premier patch d'un nouveau contributeur est une correction d'orthographe, qu'en pensent les développeurs ? Qu'ils s'en fichent, que vous gâchez de leur temps précieux avec votre petite contribution. Et personne ne s'intéresse à la perfection grammaticale d'une documentation, n'est-ce pas ?

C'est faux. Voyez mes premières contributions à *home-assistant* et *Postmodern* : j'ai corrigé des fautes d'orthographe dans la

documentation.

J’ai contribué au projet Org-mode pendant quelques années. Mon premier patch corrigeait simplement une chaîne de caractères du code. Ensuite, j’ai envoyé 56 patchs, corrigeant des bogues et ajoutant de nouvelles fonctionnalités élégantes, et j’ai aussi codé quelques extensions. À ce jour, je suis toujours seizième dans la liste des plus gros contributeurs d’*Org-mode*, qui en contient 390. Certainement pas ce qu’on appellerait un petit contributeur, donc. Je suis sûr que la communauté est bien contente de n’avoir pas méprisé ma première correction dans la documentation.

4. Mettre la barre trop haut pour les nouveaux arrivants.

Quand de nouveaux contributeurs arrivent, leurs connaissances du projet, de son contexte, et des technologies sont très variables. L’une des erreurs que les gens font le plus souvent consiste à demander aux nouveaux contributeurs des choses trop compliquées, dont ils ne pourront pas venir à bout. Cela leur fait peur (surtout les timides ou les introvertis) et ils risquent de disparaître, se croyant trop stupides pour aider.

Avant de faire le moindre commentaire, vous ne devriez avoir strictement aucun *a priori* sur leur niveau. Cela permet d'éviter ce genre de situations. Il faut également mettre beaucoup de délicatesse quand vous estimez les compétences des nouveaux, car certains pourraient se vexer si vous les sous-estimez trop.

Quand son niveau est bien estimé (un petit nombre d'échanges devrait suffire), vous devez former votre contributeur en ne le guidant ni trop ni trop peu, afin qu'il puisse s'épanouir. Il faut du temps et de l'expérience pour y parvenir, et vous allez probablement perdre certains contributeurs avant de bien maîtriser ce processus, mais c'est un chemin que tous ceux qui gèrent des projets doivent suivre.

Façonner ainsi les nouveaux arrivants est au cœur de la gestion de vos contributeurs, et ce quel que soit votre

projet. Et je suis quasiment sûr que cela s'applique à tout projet, même en dehors du logiciel libre.

5. Exiger des gens qu'ils fassent des sacrifices sur le plan personnel

C'est un point qui dépend beaucoup du projet et du contexte, mais il est très important. Dans le logiciel libre, où la plupart des gens vont contribuer par bonne volonté et parfois sur leur temps libre, vous ne devez surtout pas exiger d'eux qu'ils fassent des sacrifices personnels importants. Ça ne passera pas.

L'une des pires manières de faire cette erreur est de fixer un rendez-vous à l'autre bout du monde pour discuter du projet. Cela place les contributeurs qui vivent loin dans une situation injuste, car ils ne peuvent pas forcément quitter leur famille pour la semaine, prendre l'avion ou un autre moyen de transport, louer une chambre d'hôtel… Ce n'est pas bon : tout devrait être mis en œuvre pour que les contributeurs se sentent partie prenante du projet et intégrés à la communauté, sans que l'on exige cela de leur part. Entendons-nous bien, cela ne veut pas dire qu'il faut s'interdire toute rencontre ou activité sociale, au contraire. Pensez simplement à n'exclure personne lorsque vous discutez d'un projet.

Il en va de même pour les moyens de communication susceptibles de compliquer la vie des participants : se retrouver sur IRC (il est difficile pour certaines personnes de réserver une heure, surtout lorsqu'il faut tenir compte des différents fuseaux horaires), faire une visioconférence (en particulier quand on n'utilise pas de logiciel libre et gratuit), etc.

En gros, tout ce qui impose de se synchroniser en temps réel avec l'avancement du projet est contraignant pour certains contributeurs.

C’est pourquoi le meilleur moyen de communiquer reste le courriel, mais tous les outils de communication asynchrones (pisteurs de bogues, etc.) feront l’affaire dans la mesure où ils permettent à chacun de travailler à son rythme et à l’heure qui lui convient.

6. Ne pas avoir de code de conduite

À une époque où de plus en plus de communautés s’ouvrent à un public plus large que celui auquel elles étaient habituées (ce qui est fantastique), les codes de conduite semblent être un sujet à la mode, mais aussi délicat.

En réalité, toutes les communautés ont un code de conduite, qu’il soit inscrit noir sur blanc ou suivi inconsciemment par chacun. Sa forme dépend de la taille et de la culture de la communauté.

Cependant, en fonction de la taille de votre communauté et de la façon dont ce code s’applique, vous auriez peut-être intérêt à l’écrire dans un document, comme l’a par exemple fait Debian.

Ce n’est pas parce que vous aurez rédigé un code de conduite que tous les membres de votre communauté vont soudain se transformer en adorables Bisounours le suivant à la lettre, mais il fournit des règles que vous pouvez citer en cas de besoin. Il peut être utile de le transmettre à certains pour leur faire comprendre que leur comportement ne convient pas, et cela peut aider si une exclusion devient nécessaire, même s’il ne sert que rarement à cela, puisqu’en général personne ne veut aller aussi loin.

Je pense qu’on peut très bien se passer d’un tel document sur de petits projets. Mais vous devez garder à l’esprit qu’un code de conduite implicite découlera de votre comportement. La manière dont vos membres les plus influents communiqueront avec les autres installera l’ambiance à travers tout le

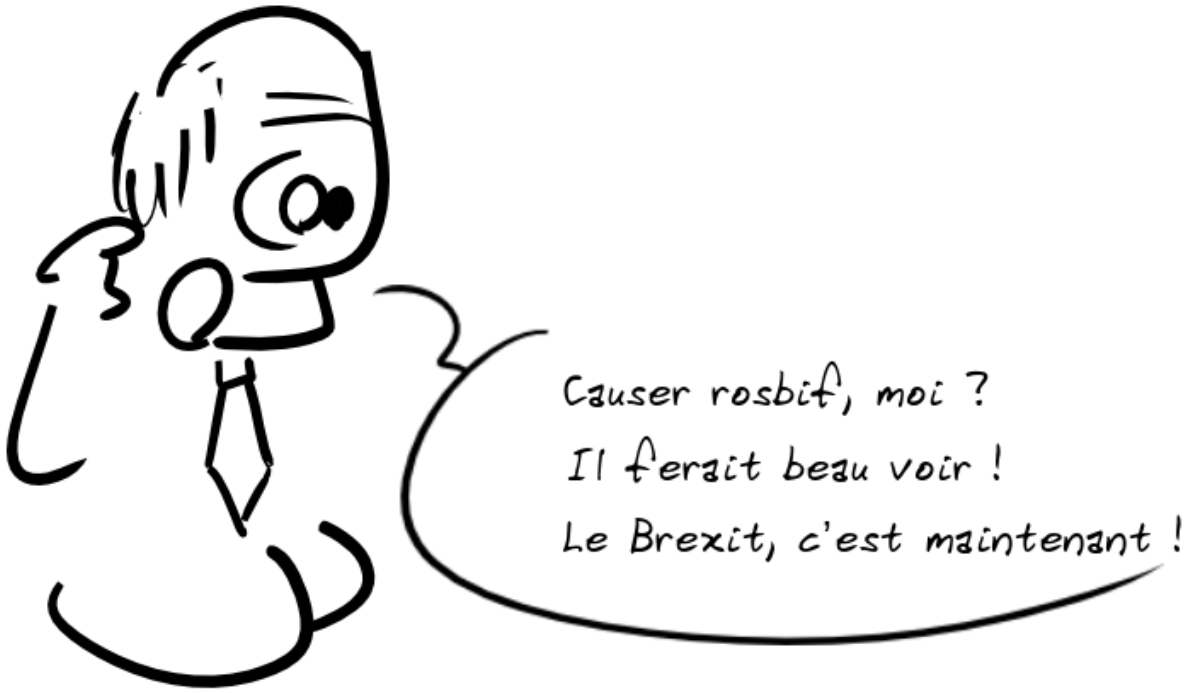
réseau. Ne sous-estimez pas cela.

Quand nous avons commencé le projet *Ceilometer*, nous avons suivi le code de conduite du projet *OpenStack* avant même qu'il ait été écrit, et probablement avons-nous même placé la barre un peu plus haut. En étant agréables, accueillants et ouverts d'esprit, nous avons réussi à obtenir une certaine mixité, avec plus de 25% de femmes dans notre équipe permanente – bien au dessus de la moyenne actuelle d'*OpenStack* et de la plupart des projets de logiciel libre !

7. Mettre les non-anglophones à l'écart

Il est important d'être conscient que la grande majorité des projets de logiciel libre utilisent l'anglais en tant que langue de communication principale. C'est logique. C'est une langue répandue, et qui semble remplir ce rôle correctement.

Mais une grande partie des codeurs n'ont pas l'anglais pour langue maternelle. Beaucoup ne le parlent pas couramment. Cela signifie que le rythme auquel ils peuvent converser peut-être très lent, ce qui peut en frustrer certains, notamment ceux qui sont nés en terre anglophone.



On peut observer ce phénomène dans les rencontres de codeurs, lors de conférences par exemple. Quand les gens débattent, il peut être très difficile pour certains d'expliquer leurs pensées en anglais et de communiquer à un rythme correct, ce qui ralentit la conversation et la transmission des idées. La pire chose qu'un anglophone puisse faire dans ce cas est de leur couper la parole, ou de les ignorer, pour la seule raison qu'ils ne parlent pas assez vite. Je comprends que cela puisse être frustrant, mais le problème n'est pas la façon de parler des non-anglophones mais les outils de communication utilisés qui ne mettent pas tout le monde au même niveau en privilégiant des conversations orales.

La même chose s'applique, à un moindre degré, aux rencontres sur IRC, qui sont relativement synchrones. Les médias complètement asynchrones ne pâtissent pas de ce défaut, et c'est pourquoi ils faudrait, à mon avis, les privilégier.

8. Pas de vision, aucune délégation des tâches

C’est une autre grosse erreur de gestion si le responsable ne parvient pas à gérer la croissance du projet alors que des gens sont disponibles et prêts à aider.

Évidemment, lorsque le flux des contributeurs commence à grossir, ajoutant de nouvelles fonctionnalités, demandant des retours et des instructions à suivre, certains responsables se retrouvent la tête sous l’eau et ne savent pas comment répondre. Ce qui a pour conséquences de frustrer les contributeurs, qui vont simplement partir.

Il est important d’avoir une vision de votre projet et de la communiquer. Dites clairement à vos contributeurs ce que vous voulez, et ne voulez pas, dans votre projet. Exprimer ces informations de manière claire (et non-agressive !) permet de minimiser les frictions entre vos contributeurs. Ils vont vite savoir s’ils veulent rejoindre ou non votre navire et quoi en attendre. Donc, soyez un bon capitaine.

S’ils choisissent de travailler avec vous et de contribuer, vous devez rapidement commencer à croire en eux et à leur déléguer certaines de vos responsabilités. Cela peut être n’importe quelle tâche que vous faites d’habitude vous-même : vérifier les patchs de certains sous-systèmes, traquer les bogues, écrire la documentation… Laissez les gens s’approprier entièrement une partie du projet car ils se sentiront responsables et ils y mettront autant de soin que vous. Faire le contraire en voulant tout contrôler vous-même est la meilleure façon de vous retrouver seul avec votre logiciel *open source*.

Aucun projet ne va gagner en taille et en popularité de cette manière.

En 2009, quand Uli Schlachter a envoyé son premier patch à

awesome, cela m'a donné plus de travail. J'ai du vérifier son patch, et j'étais déjà bien occupé pour sortir la nouvelle version d'*awesome* sur mon temps libre en dehors de mon travail ! Le travail d'Uli n'était pas parfait, et j'ai eu à gérer les bogues moi-même. Plus de travail. Alors qu'ai-je fait ? Quelques minutes plus tard, je lui ai répondu en lui envoyant un plan de ce qu'il devait faire et de ce que je pensais de son travail.

En retour, Uli envoya d'autres patchs et améliora le projet. Savez-vous ce que fait Uli aujourd'hui ? Il est responsable du gestionnaire des fenêtres *awesome* à ma place depuis 2010. J'ai réussi à transmettre ma vision, déléguer, puis à quitter le projet en le laissant dans de bonnes mains !

9. Ignorer certains types de contributions

Les gens contribuent de différentes manières, et pas toujours en codant. Il y a beaucoup de choses autour d'un projet de logiciel libre : la documentation, le tri de bogues, le support, la gestion de l'expérience utilisateur, la communication, les traductions…

Par exemple, il a fallu du temps pour que *Debian* songe à donner le statut de Développeur Debian à leurs traducteurs. *OpenStack* prend la même direction en essayant de reconnaître les contributions autres que techniques.

Dès lors que votre projet commence à récompenser certaines personnes et à créer différents statuts dans la communauté, vous devez faire très attention à n'oublier personne, car c'est le meilleur moyen de perdre des contributeurs en chemin.

10. Oublier d'être reconnaissant

Cette liste est le fruit de nombreuses années de bidouillages *open source* et de contributions à des logiciels libres.

L’expérience et le ressenti de chacun sont différents, et les mauvaises pratiques peuvent prendre différentes formes : si vous connaissez ou si vous avez vous-même rencontré d’autres obstacles dans vos contributions à des projets open-source, n’hésitez pas à compléter la liste dans les commentaires. C’est une forme de contribution.