Des routes et des ponts (11) – Les défis de la maintenance

Dans ce onzième chapitre de son ouvrage Des routes et des ponts que l'équipe Framalang vous traduit semaine après semaine (si vous avez raté le début), Nadia Eghbal aborde le problème endémique des infrastructures numériques *open source*, leur manque de ressources humaines pérennes : entre ceux qui s'y consacrent à corps perdu et s'arrêtent au bord du *burnout*, les entreprises qui profitent des ressources sans jamais contribuer en retour, ceux qui se lancent ingénument dans le développement sans notion bien nette de sécurité, ceux qui ne peuvent contribuer que sur un temps libre limité… Des témoignages sur ces situations et d'autres encore ont été recueillis dans ce chapitre.

Comme le souligne l'autrice, cette situation bancale a pour conséquence un coût important en termes d'argent et de sécurité pour l'ensemble de l'industrie numérique qui puise abondamment dans l'infrastructure numérique *open source*.

Négliger les infrastructures a un coût caché

Traduction Framalang : Piup, jums, Penguin, serici, xi, Asta, Diane, glissière de sécurité, Luc, goofy, lyn

Comme nous l'avons vu, l'infrastructure numérique est un constituant essentiel du monde actuel. Notre société repose sur les logiciels, et ces logiciels s'appuient de plus en plus sur une infrastructure qui utilise des méthodologies *open source*. Dans la mesure où nous prenons peu d'initiatives pour comprendre et pérenniser notre infrastructure numérique, que

mettons-nous en péril ?
Ne pas réinvestir dans l'infrastructure numérique présente des dangers que l'on peut classer en deux catégories : les coûts directs et les coûts indirects.

Les coûts directs

Les coûts directs sont les bogues non détectés et les vulnérabilités de sécurité qui peuvent être exploitées à des fins malveillantes, ou qui mènent à des interruptions imprévues dans le fonctionnement des logiciels. Ces coûts sont fortement ressentis et causent des problèmes qui doivent être résolus immédiatement.

Les coûts indirects

Les coûts indirects se traduisent par exemple par la perte de main-d'œuvre qualifiée, ainsi qu'une croissance faible et peu d'innovation. Même si ces coûts ne sont pas immédiatement perceptibles, ils représentent une valeur sociale difficile à évaluer.

Bogues, failles de sécurité et interruptions du service

L'introduction de ce rapport relatait les détails de la faille de sécurité Heartbleed, qui a été découverte en avril 2014 dans une bibliothèque logicielle appelée OpenSSL. Du fait de son usage généralisé, notamment pour le fonctionnement de nombreux sites web majeurs, Heartbleed a largement attiré l'attention du public sur le problème des failles de sécurité des logiciels.

En septembre 2014, une autre faille majeure a été découverte dans un autre outil essentiel appelé Bash. Bash est inclus dans des systèmes d'exploitation populaires tels que Linux et Mac OS, ce qui fait qu'il est installé sur 70 % des machines connectées à internet.

L'ensemble de bugs de sécurité, surnommés « ShellShock »,

peuvent être exploités pour permettre un accès non autorisé à un système informatique. Les vulnérabilités étaient restées non-détectées pendant au moins une décennie. Bash a été créé à l'origine par un développeur appelé Brian Fox en 1987, mais depuis 1992 il est maintenu par un seul développeur, Chet Ramey, qui travaille comme architecte technique senior à la Case Western University dans l'Ohio.

Un autre projet, OpenSSH, fournit une suite d'outils de sécurité dont l'usage est largement répandu sur internet. Des développeurs ont trouvé de multiples failles dans son code qui ont pu être prises en charge et corrigées, y compris celle de juillet 2015, qui permettait aux attaquants de dépasser le nombre maximal d'essais sur les mots de passe, et celle de janvier 2016, qui laissait fuiter les clefs de sécurité privées.

L'un des aspects du problème est que beaucoup de projets sont des outils anciens, développés au départ par un ou plusieurs développeurs passionnés, qui ont par la suite manqué de ressources pour gérer le succès de leur projet. Avec le temps, les contributions diminuent et les acteurs restants se lassent et s'en vont, mais pour autant le projet est toujours activement utilisé, avec seulement une ou deux personnes tâchant de le maintenir en vie.

Un autre problème croissant dans le paysage des logiciels actuel, où l'on voit tant de jeunes développeurs inexpérimentés, c'est que les concepts de sécurisation ne sont pas enseignés ou pas considérés comme prioritaires. Les nouveaux développeurs veulent simplement écrire du code qui marche. Ils ne savent pas faire un logiciel sécurisé, ou pensent à tort que le code public qu'ils utilisent dans la sécurité de leurs programmes a été vérifiée. Même les bonnes pratiques de divulgation sécurisée ou de gestion des failles ne sont généralement pas enseignées ni comprises. La sécurité ne devient un enjeu que lorsque le code d'un développeur a été compromis.

Christopher Allen a coécrit la première version du protocole de transfert sécurisé TLS (Transport Layer Security), dont les versions successives sont devenues un standard utilisé quasiment universellement en ligne, y compris sur des sites comme Google, Facebook ou YouTube. Bien que le protocole soit devenu un standard, Christophe parle ainsi de ses origines :

> *« En tant que co-auteur de TLS, je n'aurais pas prédit que 15 ans plus tard la moitié d'Internet utiliserait une implémentation de TLS maintenue par un ingénieur à quart-temps. C'est ce manque de maintenance qui a conduit au bug tristement célèbre de Heartbleed. Je raconte aujourd'hui cette anecdote à mes collègues qui travaillent sur les crypto-monnaies pour les avertir que leur chiffrage, ultra moderne aujourd'hui, pourrait être 'dépassé' dans 10 ans et subir le même sort, le projet n'étant plus aussi passionnant, et leur travail acharné risquerait d'être compromis. »*

En définitive, la stabilité de nos logiciels repose sur la bonne volonté et la coopération de centaines de développeurs, ce qui représente un risque significatif. La fragilité de notre infrastructure numérique a récemment été démontrée par un développeur nommé Azer Koçulu.
Azer, un développeur Node.js, hébergeait un certain nombre de bibliothèques sur un gestionnaire de paquets nommé npm. Après un conflit avec npm sur la propriété intellectuelle d'un de ses projets, Azer, mécontent de l'issue du conflit, décida de supprimer toutes les publications qu'il avait pu faire sur npm.

L'une de ces bibliothèques, left-pad, avait été réutilisée dans des centaines d'autres projets. Même s'il ne s'agissait que de quelques lignes de code, en supprimant le projet left-pad, Azer a « cassé » les algorithmes d'innombrables protocoles logiciels développés par d'autres. La décision d'Azer a provoqué tant de problèmes que npm a pris la décision sans précédent de republier sa bibliothèque, contre la volonté

d’Azer, afin de restaurer les fonctionnalités offertes par le reste de l’écosystème.

Npm a aussi revu sa politique pour qu’il soit plus difficile pour les développeurs de retirer leurs bibliothèques sans avertissement, reconnaissant ainsi que les actions d’un individu peuvent en affecter négativement beaucoup d’autres.

Les logiciels ne reçoivent pas la maintenance nécessaire dont ils ont besoin

Construire une infrastructure numérique de façon désorganisée implique que tout logiciel sera construit plus lentement et moins efficacement. L’histoire de l’infrastructure Python en fournit un bon exemple.

L’un des importants projets d’infrastructure pour les développeurs Python se nomme Setuptools. Setuptools fournit un ensemble d’outils qui rendent le développement en Python plus simple et plus standardisé.
Setuptools a été écrit en 2004, par le développeur PJ Eby. Pendant les quatre années qui ont suivi, l’outil a été largement adopté. Néanmoins, Setuptools était difficile à installer et à utiliser, et Eby était très peu réceptif aux contributions et aux corrections apportées par d’autres, car il voulait, en tant que concepteur, avoir le dernier mot sur Setuptools. En 2008, un groupe de développeurs conduits par Tarek Ziade a décidé de *forker* le projet pour obliger Eby à faire des améliorations. Ils ont appelé le nouveau projet « Distribute ». En 2013, les deux projets ont fusionné dans Setuptools.
Ce long désaccord a néanmoins souligné à la fois l’état douteux des outils de l’infrastructure de Python, et la difficulté de mettre en œuvre des améliorations, notamment parce que personne ne se consacrait aux problèmes de la communauté ni ne désirait s’en occuper.

Les outils de Python ont commencé à s'améliorer quand le groupe de travail Python Packaging Authority (PyPA) s'est formé pour se consacrer spécifiquement à définir de meilleurs standards pour le paquetage. Un développeur, Donald Stufft, fit des outils de paquetage Python le cœur de son travail et fut engagé par HP (devenu HPE) en mai 2015 pour poursuivre son travail (son parcours sera évoqué plus tard dans ce rapport).

Un autre exemple intéressant est celui de RubyGems.org, un site web utilisé par la plupart des développeurs Ruby pour héberger leurs bibliothèques Ruby. Ruby a été utilisé pour bâtir des sites web majeurs comme Twitter, AirBnB, YellowPages et GitHub lui-même. En 2013, une faille de sécurité de RubyGems.org a été découverte, mais elle ne fut pas réparée avant plusieurs jours, parce que RubyGems.org était entièrement maintenue par des bénévoles. Les bénévoles pensaient régler le problème le week-end, mais pendant ce temps, quelqu'un d'autre a découvert la faille et a piraté le serveur de RubyGems.org. Après l'attaque, les serveurs ont dû être entièrement reconfigurés. Plusieurs bénévoles ont pris sur leur temps de travail, et certains ont même pris des jours de congé, afin de remettre RubyGems.org en état de marche le plus vite possible.

Comme RubyGems.org est un élément d'infrastructure critique, la faille de sécurité affectait par rebond beaucoup de développeurs et d'entreprises. L'incident a mis en lumière le fait qu'un travail fondé uniquement sur la base du volontariat limitait les garanties de sécurité et de fiabilité que l'on pouvait offrir sur une infrastructure logicielle importante. Des dizaines de développeurs se mobilisèrent de façon « bénévole » pendant l'incident parce que le problème affectait directement leur emploi salarié.

Malheureusement, aucun d'entre eux n'avait l'expérience requise pour être utile, et aucun d'entre eux n'a continué à offrir son aide une fois les serveurs réparés. En 2015, une organisation nommée Ruby Together a été formée pour aider à

financer la maintenance et le développement de l'infrastructure Ruby, entre autres RubyGems.org, en sollicitant des entreprises comme sponsors.

La perte de main-d'œuvre qualifiée

Comme dans beaucoup de communautés de bénévoles, le « *burnout* » est commun parmi les contributeurs *open source*, qui se retrouvent à répondre aux requêtes d'utilisateurs individuels ou d'entreprises, pour un travail sans compensation. Beaucoup de développeurs ont des anecdotes où des entreprises les sollicitaient pour du travail gratuit. Daniel Roy Greenfield, développeur Django et Python, a écrit :

> *« J'ai personnellement eu des demandes pour du travail non-payé (les discussions pour payer le travail n'aboutissent jamais) par des entreprises à haut profit, grandes ou petites, pour [mes projets]. Si je ne réponds pas dans les temps convenus, si je n'accepte pas une pull request merdique, on va me mettre une étiquette de connard. Il n'y a rien de pire que d'être face à des développeurs du noyau Python/PyPA travaillant pour Redhat [sic], qui exigent de toi un travail non payé tout en critiquant ce qu'ils considèrent comme les insuffisances de ton projet, pour te pourrir ta journée et plomber ta foi en l'open source. »*

(Red Hat est une multinationale du logiciel avec un revenu annuel excédant les 2 milliards d'euros, qui vend des logiciels open source à des clients d'entreprise. Du fait de la nature de leur entreprise, les employés de Red Hat utilisent et contribuent à de nombreux projets open source : en un sens, Red Hat est devenu la tête d'affiche de l'open source dans le monde de l'entreprise. Nous reparlerons de son succès financier plus tard dans ce rapport).

Read the Docs, service d'hébergement de documentation technique précédemment mentionné, annonce explicitement sur

son site qu'il ne s'occupe pas de l'installation personnalisée dans les entreprises ou chez les particuliers.

L'un des mainteneurs, Eric Holscher, va jusqu'à faire ce commentaire :
« Je suis à peu près sûr que Read the Docs n'a aucun intérêt à être *open source*, vu que les utilisateurs privés ne contribuent jamais en retour, et se contentent de demander une assistance gratuite. »
Maquess, le contributeur OpenSSL, a tenu un discours acerbe à propos des requêtes récurrentes sur ses posts qui parlent de financement :

> *« C'est à vous que je pense, entreprises du Fortune 1000. Vous qui incluez OpenSSL dans vos firewall/dispositifs/cloud/produits financiers ou de sécurité, tous ces produits que vous vendez à profit, ou que vous utilisez pour vos infrastructures et communications internes. Vous qui n'avez pas besoin de financer une équipe interne de programmeurs pour se débattre avec du code crypté, puis qui nous harcelez pour obtenir une assistance gratuite quand vous réalisez que vous êtes incapables de l'utiliser. Vous qui n'avez jamais levé le petit doigt pour contribuer à la communauté open source qui vous a fait ce cadeau. Les concernés se reconnaîtront. Certains développeurs choisissent d'arrêter de maintenir leurs projets parce qu'ils n'ont plus assez de temps à y consacrer, et espèrent que quelqu'un d'autre prendra le relais. Pendant ce temps, les entreprises, les gouvernements et les individus dépendent de ces bibliothèques pour leur bon fonctionnement, inconscients des enjeux sous-jacents. »*

David Michael Ross, ingénieur manager dans une agence web, a écrit au sujet de son expérience :

> *« Pour moi, c'est là que le bât blesse. […] On sait qu'on a créé quelque chose gratuitement, par passion, et on voit ce*

> *flux infini de personnes qui crient « plus ! encore plus ! » et qui se mettent en colère quand on ne traite pas leur cas particulier.*
> *Il y avait mon numéro de téléphone sur l'un de mes sites personnels pour que mes amis puissent me joindre. Je l'ai enlevé au bout d'une semaine parce que des gens m'appelaient à toute heure de la journée pour de l'assistance sur les plugins, alors qu'il y a un forum consacré à ça. Il n'y a rien de fondamentalement méchant là-dedans, c'est juste que c'est usant. On se met à avoir peur de vérifier ses mails ou de répondre au téléphone. »*

Ryan Bigg, qui écrit de la documentation technique pour le *framework* Ruby on Rails, a annoncé en novembre 2015 qu'il renonçait à tout travail *open source* :

> *« Je n'ai plus le temps ni l'énergie de m'investir dans l'open source. Je ne retire strictement aucun revenu de mon travail open source, ce qui veut dire que le travail que je fais là, c'est du temps que je pourrais consacrer à des activités perso, ou à écrire. Ce n'est pas justifié d'attendre de moi que je travaille encore, en dehors de mon emploi salarié, sans que je sois honnêtement rétribué pour ça (en temps ou en argent). C'est aussi une recette qui a de bonnes chances de me conduire au burnout ou de me rendre juste globalement aigri.*

Par ailleurs, la perte de main-d'œuvre qualifiée dans l'*open source*, ce n'est pas seulement les contributeurs qui démissionnent, c'est aussi ceux qui n'ont jamais contribué du tout. »

Il existe très peu de statistiques sur la démographie des contributeurs open source, ce qui est déjà révélateur en soi. Une analyse récente de GitHub a révélé que seulement 5,4% des contributeurs *open source* étaient des femmes, qui occupent pourtant environ 15 à 20% des postes techniques dans

l’ensemble des entreprises de logiciels.

L’une des raisons qui font que les contributeurs *open source* sont un groupe remarquablement plus homogène que le secteur de la technologie dans son ensemble, c’est qu’ils ont besoin de temps et d’argent pour apporter dans un premier temps des contributions significatives. Ces contraintes empêchent des contributeurs par ailleurs qualifiés d’entrer dans cet espace. David Maclver, créateur de Hypothésis, une bibliothèque Python qui sert à tester des applications logicielles, explique pourquoi il a pu passer autant de temps sur le projet :

> *« J’ai pu le faire seulement parce que j’avais le temps et l’argent pour le faire. J’avais le temps parce que j’étais obsessionnel, je n’avais personne à charge, et je n’avais pas d’emploi. Je pouvais me permettre de ne pas avoir d’emploi parce que j’avais de l’argent. J’avais de l’argent parce que pendant la dernière moitié de l’année passée, je touchais un salaire deux fois plus élevé que d’habitude, en dépensant deux fois moins que d’habitude, et je traversais une dépression qui me rendait trop borderline pour avoir envie de dépenser mon argent dans quoi que ce soit d’intéressant. Ce ne sont pas des conditions qu’on peut raisonnablement exiger de quelqu’un. […] Est-ce qu’on pourrait produire un logiciel de qualité en moins de temps que ça, en ne travaillant que sur du temps libre ? J’en doute. »*

Photo par hiroo yamagata (CC BY 2.0)

Cory Benfield, un développeur pour les fonctions de base de Python, écrit :

> *« De manière générale, les personnes qui ne sont pas des hommes **cisgenres**, hétérosexuels, blancs, de classe moyenne, et anglophones sont moins susceptibles de pouvoir assumer les risques financiers accrus associés à l'absence d'emploi stable. Cela signifie que ces personnes ont vraiment besoin d'un salaire régulier pour pouvoir contribuer le plus efficacement possible. Et nous avons **besoin** de leur contribution : des équipes diversifiées font un meilleur travail que des équipes homogènes. »*

Charlotte Spencer, qui contribue au framework logiciel Hoodie et au système de bases de données PouchDB, fait écho à cette opinion :

> *« Toutes mes contributions sont purement bénévoles. Je n'en retire pas d'argent, même si j'aimerais beaucoup pouvoir. J'ai demandé à des vétérans de l'open source s'ils étaient payés et ce n'est pas le cas, ce qui m'a découragé d'essayer quoi que ce soit (si ces gens-là ne sont pas payés, pourquoi*

> *le serais-je ?). J'y consacre la plus grande partie de mon temps libre, mais j'essaie d'en faire moins parce que ça envahissait trop ma vie. »*

Jessica Lord, développeuse, a contribué activement à l'*open source* tout en travaillant à Code for America, une organisation à but non-lucratif qui soutient la technologie dans le secteur public. Urbaniste de formation, elle insiste sur le fait qu'elle n'avait « pas de diplôme en informatique, pas d'expérience formelle en programmation, mais un portfolio GitHub ».

Ses contributions régulières attirèrent l'attention de la plate-forme GitHub elle-même, pour qui elle travaille désormais. Cependant, Jessica note qu'elle a pu contribuer à l'*open source* grâce à un concours de circonstances « particulier » : elle a accepté une baisse de salaire pour travailler à Code for America, utilisé toutes ses économies, travaillé « presque non-stop » sur des projets *open source*, et bénéficié d'une communauté de soutiens.

À propos du manque de diversité dans l'*open source*, Jessica écrit:

> *« La valeur des savoirs communs ne peut être surestimée. Nous devons faire mieux. Nous avons besoin des idées de tout le monde. C'est le but que nous devrions chercher à atteindre. Il est nécessaire que l'open source soit ouvert à tous. Pas seulement aux privilégiés ou même aux seuls développeurs. »*

Ce dernier point soulevé par Jessica Lord est révélateur : permettre à des profils plus divers de participer à l'open source peut aider à pérenniser l'open source en elle-même. D'un point de vue fonctionnel, la grande majorité des contributeurs *open source* sont des développeurs, mais beaucoup d'autres rôles sont nécessaires pour gérer les projets d'ampleur, comme la rédaction, la gestion de projet ou la

communication. Les projets *open source* ne sont pas différents des autres types d'organisations, y compris les *startups* où l'on a besoin de personnes se chargeant de l'administration, du marketing, du design, etc., qui sont autant de fonctions nécessaires au fonctionnement de base d'une structure. C'est en partie parce que la culture *open source* repose principalement sur les développeurs que la pérennité financière est si rarement l'objet de discussions et d'actions concrètes.

Enfin, l'homogénéité des contributeurs *open source* impacte les efforts en faveur de la diversité dans le monde de la technologie au sens large, puisque l'*open source* est étroitement lié à l'embauche. En effet, comme nous l'avons remarqué plus haut, beaucoup d'employeurs utilisent les contributions *open source*, notamment les profils GitHub, pour découvrir leurs futurs employés potentiels ou pour vérifier les qualifications d'un candidat. Les employeurs qui se fient ainsi essentiellement aux preuves de contributions *open source* ne recrutent que parmi un vivier de candidats extrêmement restreint.

Ashe Dryden, dans un essai important intitulé *L'Éthique du travail non payé et la Communauté OSS*, expliquait :

> *« Juger que quelqu'un est un bon programmeur en se basant uniquement sur le code qu'il rend disponible publiquement, c'est exclure bien plus que les gens marginaux. C'est aussi exclure quiconque n'est pas autorisé à publier son code publiquement pour des raisons de licence ou de sécurité. Cela concerne également un grand nombre de travailleurs freelance ou de contractuels qui, pour des raisons légales, n'ont pas le droit de revendiquer publiquement leur participation à un projet (par exemple s'ils ont signé un accord de confidentialité). Dans une industrie où on lutte pour dénicher assez de talents, pourquoi limitons-nous artificiellement le spectre des candidats ? »* (source)

Comment atténuer ou éviter certains des coûts qui s’imposent aux personnes qui participent à l’élaboration d’infrastructures numériques aujourd’hui ? Pour commencer, nous analyserons comment les projets d’infrastructure sont actuellement financés.