Des routes et des ponts (7) – pour une infrastructure durable

Une question épineuse pour les projets libres et *open source* est leur maintenance à long terme, et bien évidemment les ressources, tant financières qu'humaines, que l'on peut y consacrer. Tel est le sujet qu'aborde ce *nouveau chapitre de l'ouvrage de* ***Nadia Eghbal*** Des routes et des ponts *que le groupe Framalang vous traduit semaine après semaine (si vous avez raté les épisodes précédents)*

Elle examine ici une série de cas de figures en fonction de l'origine de l'origine projet.

Comment les projets d'infrastructure numérique sont-ils gérés et maintenus ?

TraductionFramalang : Diane, Penguin, Asta, Rozmador, Lumibd, jum-s, goofy, salade, AFS, Théo

Nous avons établi que les infrastructures numériques sont aussi nécessaires à la société moderne que le sont les infrastructures physiques. Bien qu'elles ne soient pas sujettes aux coûts élevés et aux obstacles politiques auxquels sont confrontées ces dernières, leur nature décentralisée les rend cependant plus difficiles à cerner. Sans une autorité centrale, comment les projets *open source* trouvent-ils les ressources dont ils ont besoin ?

En un mot, la réponse est différente pour chaque projet. Cependant, on peut identifier plusieurs lieux d'où ces projets peuvent émaner : au sein d'une entreprise, via une startup,

de développeurs individuels ou collaborant en communauté.

Au sein d’une entreprise

Parfois, le projet commence au sein d’une entreprise. Voici quelques exemples qui illustrent par quels moyens variés un projet *open source* peut être soutenu par les ressources d’une entreprise :

Go, le nouveau langage de programmation évoqué précédemment, a été développé au sein de Google en 2007 par les ingénieurs Robert Griesemer, Rob Pike, et Ken Thompson, pour qui la création de Go était une expérimentation. Go est *open source* et accepte les contributions de la communauté dans son ensemble. Cependant, ses principaux mainteneurs sont employés à plein temps par Google pour travailler sur le langage.

React est une nouvelle bibliothèque JavaScript dont la popularité grandit de jour en jour.
React a été créée par Jordan Walke, ingénieur logiciel chez Facebook, pour un usage interne sur le fil d’actualités Facebook. Un employé d’Instagram (qui est une filiale de Facebook) a également souhaité utiliser React, et finalement, React a été placée en *open source*, deux ans après son développement initial.
Facebook a dédié une équipe d’ingénieurs à la maintenance du projet, mais React accepte aussi les contributions de la communauté publique des développeurs.

Swift, le langage de programmation utilisé pour iOS, OS X et les autres projets d’Apple, est un exemple de projet qui n’a été placé en *open source* que récemment. Swift a été développé par Apple en interne pendant quatre ans et publié en tant que langage propriétaire en 2014. Les développeurs pouvaient utiliser Swift pour écrire des programmes pour les appareils d’Apple, mais ne pouvaient pas contribuer au développement du cœur du langage. En 2015, Swift a été rendu *open source* sous la licence Apache 2.0.

Pour une entreprise, les incitations à maintenir un projet *open source* sont nombreuses. Ouvrir un projet au public peut signifier moins de travail pour l'entreprise, grâce essentiellement aux améliorations de la collaboration de masse.
Cela stimule la bonne volonté et l'intérêt des développeurs, qui peuvent alors être incités à utiliser d'autres ressources de l'entreprise pour construire leurs projets.

Disposer d'une communauté active de programmeurs crée un vivier de talents à recruter. Et parfois, l'ouverture du code d'un projet aide une entreprise à renforcer sa base d'utilisateurs et sa marque, ou même à l'emporter sur la concurrence. Plus une entreprise peut capter de parts de marché, même à travers des outils qu'elle distribue gratuitement, plus elle devient influente. Ce n'est pas très différent du concept commercial de « produit d'appel ».

Même quand un projet est créé en interne, s'il est *open source*, alors il peut être librement utilisé ou modifié selon les termes d'une licence libre, et n'est pas considéré comme relevant de la propriété intellectuelle de l'entreprise au sens traditionnel du terme. De nombreux projets d'entreprise utilisent des licences libres standard qui sont considérées comme acceptables par la communauté des développeurs telles que les licences Apache 2.0 ou BSD. Cependant, dans certains cas, les entreprises ajoutent leurs propres clauses. La licence de React, par exemple, comporte une clause additionnelle qui pourrait potentiellement créer des conflits de revendications de brevet avec les utilisateurs de React.

En conséquence, certaines entreprises et individus sont réticents à utiliser React, et cette décision est fréquemment décrite comme un exemple de conflit avec les principes de l'*open source*.

Via une *startup*

Certains projets d'infrastructures empruntent la voie traditionnelle de la *startup*, ce qui inclut des financements en capital-risque. Voici quelques exemples :
Docker, qui est peut-être l'exemple contemporain le plus connu, aide les applications logicielles à fonctionner à l'intérieur d'un conteneur. (Les conteneurs procurent un environnement propre et ordonné pour les applications logicielles, ce qui permet de les faire fonctionner plus facilement partout). Docker est né en tant que projet interne chez dotCloud, une société de Plate-forme en tant que service (ou PaaS, pour *platform as a service* en anglais), mais le projet est devenu si populaire que ses fondateurs ont décidé d'en faire la principale activité de l'entreprise. Le projet Docker a été placé en *open source* en 2013. Docker a collecté 180 millions de dollars, avec une valeur estimée à plus d'1 milliard de dollars.

Leur modèle économique repose sur du support technique, des projets privés et des services. Les revenus de Docker pour l'année 2014 ne dépassaient pas 10 millions de dollars.

Npm est un gestionnaire de paquets sorti en 2010 pour aider les développeurs de Node.js à partager et à gérer leurs projets. Npm a collecté près de 11 millions de dollars de financements depuis 2014 de la part de True Ventures et de Bessemer Ventures, entre autres. Leur modèle économique se concentre sur des fonctionnalités payantes en faveur de la vie privée et de la sécurité.

Meteor est un *framework* JavaScript publié pour la première fois en 2012. Il a bénéficié d'un programme d'incubation au sein de Y Combinator, un prestigieux accélérateur de *startups* qui a également été l'incubateur d'entreprises comme AirBnB et Dropbox. À ce jour, Meteor a reçu plus de 30 millions de dollars de financements de la part de firmes comme Andreessen Horowitz ou Matrix Partners. Le modèle économique de Meteor se

base sur une plateforme d'entreprise nommée Galaxy, sortie en Octobre 2015, qui permet de faire fonctionner et de gérer les applications Meteor.

L'approche basée sur le capital-risque est relativement nouvelle, et se développe rapidement.
Lightspeed Venture Partners a constaté qu'entre 2010 et 2015, les sociétés de capital-risque ont investi plus de 4 milliards de dollars dans des entreprises *open source*, soit dix fois plus que sur les cinq années précédentes.

Le recours aux fonds de capital-risque pour soutenir les projets *open source* a été accueilli avec scepticisme par les développeurs (et même par certains acteurs du capital-risque eux-mêmes), du fait de l'absence de modèles économiques ou de revenus prévisibles pour justifier les estimations. Steve Klabnik, un mainteneur du langage Rust, explique le soudain intérêt des capital-risqueurs pour le financement de l'*open source* :

> *« Je suis un investisseur en capital-risque. J'ai besoin qu'un grand nombre d'entreprises existent pour gagner de l'argent… J'ai besoin que les coûts soient bas et les profits élevés. Pour cela, il me faut un écosystème de logiciels open source en bonne santé. Donc je fais quoi? … Les investisseurs en capital-risque sont en train de prendre conscience de tout ça, et ils commencent à investir dans les infrastructures. […]*
> *Par bien des aspects, le matériel open source est un produit d'appel, pour que tu deviennes accro…puis tu l'utilises pour tout, même pour ton code propriétaire. C'est une très bonne stratégie commerciale, mais cela place GitHub au centre de ce nouvel univers. Donc pour des raisons similaires, a16z a besoin que GitHub soit génial, pour servir de tremplin à chacun des écosystèmes open source qui existeront à l'avenir… Et a16z a suffisamment d'argent pour en «gaspiller » en finançant un projet sur lequel ils ne récupéreront pas de bénéfices directs, parce qu'ils sont suffisamment*

> *intelligents pour investir une partie de leurs fonds dans le développement de l'écosystème. »*

GitHub, créé en 2008, est une plateforme de partage/stockage de code, disponible en mode public ou privé, doté d'un environnement ergonomique. Il héberge de nombreux projets *open source* populaires et, surtout, il est devenu l'épicentre culturel de la croissance explosive de l'*open source* (dont nous parlerons plus loin dans ce rapport).
GitHub n'a reçu aucun capital-risque avant 2012, quatre ans après sa création. Avant cette date, GitHub était une entreprise rentable. Depuis 2012, GitHub a reçu au total 350 millions de dollars de financements en capital-risque.

Image par Nick Quaranto (CC BY-SA 2.0)

Andreessen Horowitz (alias a16z), la firme d'investissement aux 4 millards de dollars qui a fourni l'essentiel du capital de leur première levée de fonds de 100 millions de dollars, a déclaré qu'il s'agissait là de l'investissement le plus important qu'elle ait jamais fait jusqu'alors.
En d'autres termes, la théorie de Steve Klabnik est que les sociétés de capital-risque qui investissent dans les infrastructures *open source* promeuvent ces plateformes en tant que « produit d'appel », même quand il n'y a pas de modèle économique viable ou de rentabilité à en tirer, parce que cela permet de faire croître l'ensemble de l'écosystème. Plus GitHub a de ressources, plus l'*open source* est florissant.

Plus l'*open source* est florissant, et mieux se portent les startups. À lui seul, l'intérêt que portent les sociétés d'investissement à l'*open source*, particulièrement quand on considère l'absence de véritable retour financier, est une preuve du rôle-clé que joue l'*open source* dans l'écosystème plus large des startups.
Par ailleurs, il est important de noter que la plateforme GitHub en elle-même n'est pas un projet *open source*, et n'est donc pas un exemple de capital-risque finançant directement l'*open source*. GitHub est une plateforme à code propriétaire qui héberge des projets *open source*. C'est un sujet controversé pour certains contributeurs *open source*.

Par des personnes ou un groupe de personnes

Enfin, de nombreux projets d'infrastructures numériques sont intégralement développés et maintenus par des développeurs indépendants ou des communautés de développeurs. Voici quelques exemples :

Python, un langage de programmation, a été développé et publié par un informaticien, Guido van Rossum, en 1991.
Van Rossum déclarait qu'il « était à la recherche d'un projet de programmation « passe-temps », qui [le] tiendrait occupé pendant la semaine de Noël. »
Le projet a décollé, et Python est désormais considéré comme l'un des langages de programmation les plus populaires de nos jours.

Van Rossum reste le principal auteur de Python (aussi connu parmi les développeurs sous le nom de « dictateur bienveillant à vie » et il est actuellement employé par Dropbox, dont les logiciels reposent fortement sur Python.

Python est en partie géré par la Python Software Foundation

(NdT: Fondation du logiciel Python), créée en 2001, qui bénéficie de nombreux sponsors commerciaux, parmi lesquel Intel, HP et Google.

RubyGems est un gestionnaire de paquets qui facilite la distribution de programmes et de bibliothèques associés au langage de programmation Ruby.
C'est une pièce essentielle de l'infrastructure pour tout développeur Ruby. Parmi les sites web utilisant Ruby, on peut citer par exemple Hulu, AirBnB et Bloomberg. RubyGems a été créé en 2003 et est géré par une communauté de développeurs. Certains travaux de développement sont financés par Ruby Together, une fondation qui accepte les dons d'entreprises et de particuliers.

Twisted, une bibliothèque Python, fut créée en 2002 par un programmeur nommé Glyph Lefkowitz. Depuis lors, son usage s'est largement répandu auprès d'individus et d'organisations, parmi lesquelles Lucasfilm et la NASA.
Twisted continue d'être géré par un groupe de volontaires. Le projet est soutenu par des dons corporatifs/commerciaux et individuels ; Lefkowitz en reste l'architecte principal et gagne sa vie en proposant ses services de consultant.

Comme le montrent tous ces exemples, les projets *open source* peuvent provenir de pratiquement n'importe où. Ce qui est en général considéré comme une bonne chose. Cela signifie que les projets utiles ont le plus de chances de réussir, car ils évitent d'une part les effets de mode futiles inhérents aux startups, et d'autre part la bureaucratie propre aux gouvernements. La nature décentralisée de l'infrastructure numérique renforce également les valeurs de démocratie et d'ouverture d'Internet, qui permet en principe à chacun de créer le prochain super projet, qu'il soit une entreprise ou un individu.
D'un autre côté, un grand nombre de projets utiles proviendront de développeurs indépendants qui se trouveront tout à coup à la tête d'un projet à succès, et qui devront

prendre des décisions cruciales pour son avenir. Une étude de 2015 menée par l'Université fédérale de Minas Gerai au Brésil a examiné 133 des projets les plus activement utilisés sur Github, parmi les langages de programmation, et a découvert que 64 % d'entre eux, presque les deux tiers, dépendaient pour leur survie d'un ou deux développeurs seulement.

Bien qu'il puisse y avoir une longue traîne de contributeurs occasionnels ou ponctuels pour de nombreux projets, les responsabilités principales de la gestion du projet ne reposent que sur un très petit nombre d'individus.

Coordonner des communautés internationales de contributeurs aux avis arrêtés, tout en gérant les attentes d'entreprises classées au Fortune 500 qui utilisent votre projet, voilà des tâches qui seraient des défis pour n'importe qui. Il est impressionnant de constater combien de projets ont déjà été accomplis de cette manière. Ces tâches sont particulièrement difficiles dans un contexte où les développeurs manquent de modèles clairement établis, mais aussi de soutien institutionnel pour mener ce travail à bien. Au cours d'interviews menées pour ce rapport, beaucoup de développeurs se sont plaints en privé qu'ils n'avaient aucune idée de qui ils pouvaient solliciter pour avoir de l'aide, et qu'ils préféreraient « juste coder ».

Pourquoi continuent-ils à le faire ? La suite de ce rapport se concentrera sur pourquoi et comment les contributeurs de l'*open source* maintiennent des projets à grande échelle, et sur les raisons pour lesquelles c'est important pour nous tous.