Des routes et des ponts (4) – la gratuité pour changer le monde

*Nous poursuivons la lecture du livre Des routes et des ponts de **Nadia Eghbal** que le groupe Framalang vous traduit au fil des semaines. Après nous avoir expliqué en termes simples de quoi sont constitués les logiciels (n'hésitez pas à reprendre les épisodes précédents, si par exemple vous avez oublié ce qu'est un* framework *ou une bibliothèque), elle nous explique en quoi l'accès libre et gratuit à ces composants a révolutionné l'industrie du logiciel : son fonctionnement, son financement, mais aussi la formation des professionnels.*

Comment la gratuité des logiciels a transformé la société

par **Nadia Eghbal**

Traduction Framalang : Luc, urlgaga, Penguin, Mika, Asta, Edgar Lori, Julien / Sphinx, flo, xi, Bromind, goofy, salade, lyn. et 3 anonymes.

La première réflexion qui vient à l'esprit est : *« Pourquoi ces développeurs ont-ils rendu leur logiciel gratuit ? Pourquoi ne pas le faire payer ? »*
Les arguments en faveur du logiciel public reposent sur sa riche histoire politique et sociale. Mais d'abord, regardons la vérité en face : notre société ne serait pas là où elle est aujourd'hui si des développeurs n'avaient pas rendu le logiciel libre et gratuit.

Avec le logiciel libre, la production de logiciel est plus simple et considérablement moins chère

Uber, un service de transport de personne, a annoncé récemment que des développeurs avaient créé un système permettant de réserver une voiture en utilisant *Slack* (une application de développement collaboratif) et non l'application mobile *Uber*. Le projet a été bouclé en 48 heures par une équipe de la *App Academy*, une école de programmation.
Uber a constaté que l'équipe avait été capable d'achever le projet rapidement car elle « *avait utilisé des bibliothèques ouvertes telles que* rails, geocoder *et* unicorn *pour accélérer le développement tout en travaillant sur une base solide.* »
En d'autres termes, la quantité de code que l'équipe a dû écrire par elle-même a été fortement réduite car elle a pu utiliser des bibliothèques libres créées par d'autres.
Ruby Geocoder, par exemple, est une bibliothèque réalisée en 2010 et maintenue par Alex Reisner, un développeur indépendant. Geocoder permet à une application de chercher facilement des noms de rues et des coordonnées géographiques.
Unicorn est un serveur datant de 2009, il est administré par une équipe de sept contributeurs (leurs noms sont visibles sur le site web d'Unicorn) encadrés par Eric Wong, un développeur.
Créer un nouveau logiciel n'a jamais été aussi simple, car il existe de plus en plus de portions de code « prêtes à l'emploi » dont on peut se servir. Pour en revenir à la métaphore de l'entreprise de bâtiment, il n'est plus

nécessaire pour construire un immeuble de fabriquer soi-même tout ce dont on a besoin, il est plus simple d'acheter du « préfabriqué » et d'assembler fondation, structure porteuse et murs comme des Legos.
Du coup, il n'est plus nécessaire de savoir comment construire un logiciel à partir de zéro pour être qualifié de développeur. le service des statistiques sur le travail des USA (Bureau of Labor Statistics) estime que l'emploi des développeurs va augmenter de 22% entre 2012 et 2022, soit bien plus rapidement que la moyenne dans les autres professions.

Le logiciel libre est directement responsable de la renaissance actuelle des startups

Les coûts de lancement d'une entreprise ont énormément baissé depuis la première bulle internet de la fin des années 90. Le capital-risqueur et ex-entrepreneur Mark Suster évoquait son expérience dans un billet de blog de 2011 :

> *Quand j'ai monté ma première entreprise, en 1999, l'infrastructure coûtait 2,5 millions de dollars, simplement pour commencer, et il fallait y ajouter 2,5 millions de dollars de plus pour payer l'équipe chargée de coder, lancer, gérer, démarcher et vendre notre logiciel. […]*

> *Nous avons à peine perçu le premier changement d'ampleur dans notre industrie. Il a été porté par l'introduction du logiciel libre et plus précisément par ce que l'on a appelé la pile LAMP. Linux (au lieu de UNIX), Apache (un logiciel de serveur web), MySQL (à la place d'Oracle) et PHP. Il y a bien sûr eu des variantes – nous préférions PostgreSQL à MySQL et beaucoup de gens utilisaient d'autres langages de programmation que PHP.*

> *Le libre est devenu un mouvement, un état d'esprit. Soudain, les logiciels d'infrastructure étaient presque gratuits. Nous avons payé 10% du tarif normal pour l'achat des logiciels et le reste de l'argent est allé dans le support. Un tel effondrement de 90% des coûts engendre de l'innovation, croyez-moi.*

La disponibilité actuelle des composants logiciels libres et gratuits (associée à des services d'hébergement moins chers comme Amazon Web Services et Heroku) permet à une *startup* technologique de se lancer sans avoir besoin de millions de dollars. Les entrepreneurs peuvent tout à fait sortir un produit et trouver un marché sans dépenser un seul dollar, la levée de fonds auprès de capital-risqueurs se faisant seulement après avoir montré la viabilité de leur projet.
Alan Schaaf, qui a fondé Imgur, un site populaire de partage d'images faisant partie des 50 sites les plus consultés au monde, a justement déclaré que les sept dollars nécessaires à l'achat du nom de domaine représentaient la seule dépense indispensable au démarrage de son entreprise. Imgur était rentable et avant de lever 40 millions de dollars en 2014 auprès de l'entreprise de capital-risque Andreessen Horowitz, Schaaf n'a eu recours à aucun fond extérieur pendant 5 ans (source).
Les capital-risqueurs ainsi que les autres acteurs de l'investissement ont, à leur tour, commencé à investir des montants moindres, développant ainsi de nouvelles formes de fond d'investissement dont voici trois exemples.

Fonds spécialisés dans le capital d'amorçage : sociétés de capital-risque préférant financer la première levée de fond, plutôt que de participer à une augmentation de capital ultérieure.

Fonds de micro capital-risque : une définition assez large sous laquelle on regroupe les sociétés de capital-risque disposant de moins de 50 millions de dollars d'actifs.

> **Accélérateurs de *startup*** : des sociétés qui financent de petites sommes, souvent inférieures à 50 000 dollars, et qui également conseille et parraine les toutes jeunes entreprises..

Aujourd'hui, avec 10 millions de dollars, on peut financer cent entreprises contre seulement une ou deux dans les années 90.

Le logiciel libre a simplifié l'apprentissage de la programmation, rendant la technologie accessible à tous, partout dans le monde.

Si aujourd'hui vous voulez apprendre à coder chez vous, vous pouvez commencer par étudier *Ruby on Rails*. *Rails* est le nom d'un framework et *Ruby* est un langage de programmation. N'importe qui disposant d'un accès internet peut installer gratuitement ces outils sur n'importe quel ordinateur. Parce qu'ils sont libres et gratuits, ils sont également très populaires, ce qui signifie qu'il existe énormément d'informations en ligne permettant de bien démarrer, du simple tutoriel au forum d'aide. Cela montre qu'apprendre comment coder est aussi accessible que d'apprendre à lire et écrire l'anglais ou le français.
Pour comparer, l'utilisation de *frameworks* et de langages non *open source* impliquaient : de payer pour y avoir accès, d'utiliser un système d'exploitation et des logiciels spécifiques, et d'accepter des contraintes de licence susceptibles d'entraver le dépôt d'un brevet pour un logiciel construit sur la base de ce framework. Aujourd'hui il est difficile de trouver des exemples de *frameworks* qui ne sont pas publics. L'un des plus célèbres exemples de *framework* propriétaire est le .NET, développé et sorti en 2002. En 2014, Microsoft a annoncé la sortie d'une version publique de .NET, appelée .NET Core.

Audrey Eschright, une développeuse, a décrit comment les logiciels *open source* l'ont aidée à apprendre la programmation à la fin des années 90.

> *Je voulais apprendre à programmer mais je n'avais pas d'argent. Pas la version « étudiante fauchée » : ma famille était pauvre mais également dans une situation chaotique…. Cela peut sembler étrange aujourd'hui, mais à l'époque il y avait en fait deux options pour quelqu'un qui voulait écrire de véritables logiciels : on pouvait utiliser un ordinateur avec Windows et payer pour les coûteux outils de développement de Microsoft, ou on pouvait avoir accès a un système Unix et utiliser [le compilateur] gcc…. Mon but devint donc d'avoir accès à des systèmes Unix pour pouvoir apprendre à programmer et faire des trucs sympas.*

Jeff Atwood, un développeur .NET de longue date, a expliqué sa décision d'utiliser *Ruby* pour un nouveau projet, *Discourse*, en 2013 :

> *Quand on habite en Argentine, au Népal ou en Bulgarie par exemple, il est vraiment très difficile de démarrer en programmation avec les outils fournis par Microsoft. Les systèmes d'exploitation, les langages et les outils open source permettent de mettre tout le monde au même niveau, ils constituent le socle sur lequel travaillera, partout dans le monde, la prochaine génération de programmeurs, celle qui nous aidera à changer le monde.*

Le nombre de *startups* a explosé et dans leur sillage sont apparues de nombreuses initiatives pour enseigner la programmation aux gens : aux enfants et aux adolescents, mais aussi aux membres de communautés défavorisées, aux femmes ou aux personnes en reconversion professionnelle. Parmi ces initiatives on retrouve *Women Who Code*, *Django Girls*, *Black Girls Code*, *One Month* et *Dev Bootcamp*.
Certaines de ces organisations offrent leurs services

gratuitement, tandis que d'autres les font payer. Toutes se reposent sur des logiciels libres et gratuits dans leur enseignement. Par exemple, *Django Girls* a appris à coder à plus de 2000 femmes dans 49 pays. Bien que l'organisation n'ait pas développé *Django* elle-même, elle a le droit d'utiliser *Django*, que les étudiantes téléchargent et utilisent gratuitement dans leur programme d'apprentissage.

Django Girls hackathon à Rome — Photo Django Girls CC-BY-2.0

Dev Bootcamp apprend à programmer aux personnes qui veulent changer de carrière, et prépare n'importe qui, du professeur d'anglais au vétéran, à devenir développeur professionnel. Le programme coûte entre 12 et 14 000 dollars. *Dev Bootcamp* enseigne entre autres Ruby, JavaScript, Ruby on Rails et *SQL*. Les étudiants peuvent télécharger et utiliser tous ces outils gratuitement, et *Dev Bootcamp* n'a pas besoin de payer pour les utiliser. *Dev Bootcamp* a été acheté par Kaplan en 2014 pour un prix inconnu.

Si des logiciels aussi importants n'étaient pas gratuits,

beaucoup de gens seraient dans l'incapacité de participer à la renaissance technologique actuelle. Il existe encore de nombreux obstacles économiques et sociaux qui empêchent qu'ils soient encore plus nombreux à participer, comme le prix du matériel nécessaire pour avoir un ordinateur portable et une connexion Internet, mais les outils de programmation eux-mêmes ne coûtent rien.

Mes données dans un nuage ? — Oui mais le mien

Plutôt que de se résigner à l'usage de services en ligne n'offrant aucune garantie réelle de confidentialité, Frank Karlitschek a décidé de ne pas se contenter de prêcher la bonne parole mais de passer à l'acte en élaborant (avec d'autres) un projet qui remporte un succès grandissant : un logiciel libre et open source *de stockage de données. En revenant sur l'historique du projet ownCloud, il nous rappelle au passage les clés de la réussite (ne perdons pas de vue la proportion importante de projets open source qui n'aboutissent jamais) : développement collaboratif du code ouvert, prenant appui sur des outils et choix techniques ayant déjà une large base de développeurs, flexibilité, compatibilité multi-plateforme…*

*Cet article donne quelques indications plus précises sur les technologies mises en œuvre qui peuvent laisser perplexe le lecteur non développeur, mais la démarche et la philosophie de l'*open source *y apparaîtront pour tous avec clarté. L'enjeu, c'est de rendre à l'utilisateur le contrôle de ses données.*

Au fait, Framasoft dispose depuis un an de son propre ownCloud

[1], *pourquoi pas vous ?*

Pourquoi j'ai créé OwnCloud et l'ai rendu *open source*

par **Frank Karlitschek**, fondateur de ownCloud et mainteneur de l'architecture globale du projet.

Article original : Why I Built OwnCloud and Made It Open Source Traduction Framalang : Asta, r0u, KoS, Wan, Omegax, goofy, Diab

Il y a 4 ans, j'étais au CampKDE à San Diego, je donnais une conférence sur la protection des données, mettant en garde le public sur les risques pour leur vie privée auprès des fournisseurs de *cloud* – en particulier Dropbox. « – Eh bien fais-le toi-même », m'a-t-on dit. Bien sûr, j'avais déjà créé des choses dans le passé, alors bien sûr, j'ai dit que j'allais le faire. Et c'est là que j'ai commencé mon odyssée, en premier lieu pour me protéger moi-même, mes amis et mes collègues de l'espionnage des gouvernements et d'autres méchants, et plus tard – quand j'ai vu l'intérêt croître dans le monde – pour concevoir un projet concret et efficace.

> *je n'avais pas envie d'envoyer mes données à un service tiers pour qu'il les stocke on ne sait où*

Évidemment, je devais décider d'un certain nombre de choses avant de commencer, notamment ce que je voulais que fasse le logiciel, quelle plateforme de développement utiliser, comment le structurer et bien sûr il fallait que je lui trouve un nom : ownCloud (NdT : littéralement, « le nuage qu'on possède »).

Mes amis et moi avions besoin d'un moyen de synchroniser nos images, nos documents et même nos vidéos en passant d'un appareil à l'autre (au lieu d'utiliser une clé USB), nous

voulions aussi partager ces fichiers avec nos amis et nos proches. À l'époque, Dropbox devenait très populaire, mais je n'avais pas envie d'envoyer mes données à un service tiers pour qu'il les stocke on ne sait où. Je voulais créer une plateforme que mes amis puissent utiliser sur les espaces de stockage qu'ils avaient déjà, à la différence du *cloud*, pas seulement pour synchroniser et partager, mais aussi une plateforme assez flexible pour qu'on puisse y créer des applications.

Bien sûr ownCloud allait être *open source*.

Je faisais déjà partie de la communauté *open source*, mais ce n'est pas la seule raison. En faisant de l'*open source* je concevais un code qui serait complètement transparent (et donc aurait peu de risques de comporter des « portes dérobées » pour entrer dans mes données). De plus je pouvais compter sur un grand nombre de personnes animées des mêmes convictions pour m'aider à créer ownCloud, je n'étais donc pas tout seul. Et je pouvais réutiliser les technologies d'autres projets. Comme SABREDAV, qui est le framework que nous utilisons pour la communication WebDAV du serveur (CalDAV, CardDAV et WebDAV sont tous utilisés par ownCloud), et nous utilisons aussi jQuery. Nous avons également utilisé csync pour les capacités de synchronisation bi-directionnelle du client de bureau et Qt pour l'interface utilisateur multi-plateforme. Je n'ai pas eu à réinventer la roue une fois de plus, je n'ai eu qu'à assembler ce qui existait déjà pour que tout fonctionne.

Mais comme je l'ai déjà dit, je savais ce que je voulais : ownCloud devait être plus qu'une « app ». Bien sûr, stocker les données d'une manière sûre et sécurisée est une chose importante. Mais en fin de compte, les gens veulent faire quelque chose de leurs données, alors j'ai voulu ajouter davantage de fonctionnalités à travers les applications ownCloud. Les applications sont des extensions qui peuvent implémenter des fonctionnalités telles que la détection de virus, la journalisation des accès et des changements de

fichiers, le *versionnage*, le chiffrement, l'édition de fichiers et bien d'autres choses. Ce genre d'intégration du stockage de fichiers avec d'autres services est essentiel pour le développement futur.

Je voulais que mon projet soit flexible, de sorte que les gens puissent s'appuyer sur ownCloud (et beaucoup l'ont fait, avec une application type « Google News », un *streamer* de vidéos, un lecteur de musique, un calendrier – et plus encore) et que ownCloud puisse s'intégrer dans de nombreux environnements. Par exemple, n'importe quel client WebDAV devait pouvoir accéder à ownCloud dès le départ et le concept d'applications internes est là aussi depuis le début du projet.

Bien entendu, nous sommes plus avancés à présent – il y a des API de partage et d'administration, des API internes pour les applications utilisant OCS, il existe des bibliothèques pour mobile (que nous avons rendues *open source*) et qui permettent l'intégration à d'autres applications mobiles, une base de données clés-valeurs pour un usage général de stockage de données, de synchronisation, et davantage encore. Ensuite, il y a l'intégration de systèmes de stockage externe comme FTP, S3, SWIFT, CIFS, iRODS et beaucoup d'autres. Mais même à l'époque où nous avons commencé, les intentions étaient claires – construire quelque chose d'assez flexible pour que les gens puissent créer des solutions auxquelles nous n'avions pas pensé.

Et c'est justement ça, la puissance de l'*open source*.

Nous (ma communauté grandissante et moi) avons évalué différentes options pour trouver la bonne technologie qui pourrait tourner sur chaque plateforme, du micro serveur jusqu'à des clusters de serveurs, qui aurait toutes les fonctionnalités et serait connue d'un grand nombre de développeurs. C'est pourquoi nous avons opté pour PHP et JS pour la partie serveur, C++ pour la synchronisation des Clients, Objective-C pour iOS et Java pour Android.

Il y avait plusieurs critères architecturaux à remplir dès le départ : multiplateforme, facilité d'extension, support des infrastructures, haute disponibilité basée sur les composants les plus largement utilisés. Donc, nous avons choisi PHP, pour cibler la pile « LAMP » (Linux / Apache / MySQL / PHP) qui est la plus répandue et éprouvée des plateformes permettant tout cela.

C'est également un projet *open source* et PHP est disponible gratuitement, facile à trouver, et multiplateforme (variantes Windows et Linux, IIS, Apache et autres serveurs Linux). Il bénéficie d'une communauté massive de développeurs dont beaucoup sont très expérimentés. Enfin, c'est un langage facilement accessible pour la communauté. Avec tout ça, c'était une évidence.

> *« L'open source est la seule solution pour un stockage de données réellement sécurisé »*

Comme j'ai commencé ce projet par une conférence sur la sécurité et la confidentialité, il était essentiel d'avoir la meilleure sécurité possible pour les API. J'ai choisi un chiffrement SSL fort pour toutes les API WebDAV et REST. L'authentification est faite via la méthode basique, qui est très simple et facile à gérer. On peut également utiliser SAML, fourni au travers de son implémentation Shibboleth. En complément OAuth et l'authentification à deux facteurs sont disponibles, et nous profitons même de la flexibilité de

ownCloud pour intégrer un backend personnalisé, en utilisant des jetons à la place des mots de passe standards.

Je suis convaincu que le stockage de fichiers n'est pas seulement un service web ou une infrastructure informatique de plus. C'est là où les gens et les entreprises stockent et gèrent leurs données les plus importantes. C'est pourquoi il est essentiel de le rendre aussi sécurisé que possible. Avec un logiciel propriétaire, vous ne pouvez jamais être sûr qu'il n'y a pas une porte dérobée ou d'autres problèmes de sécurité. L'open source est la seule solution pour un stockage de données réellement sécurisé. Voilà ce que j'ai fait et pourquoi je l'ai fait. J'ai mis à ce travail toute ma passion pour l'*open source* et il a aussi demandé beaucoup de soin !

- Le site du projet ownCloud
- Contribuer au développement d'ownCloud

Notes

[1] Tiens par exemple, vous voulez de quoi imprimer de chouettes posters qui expliquent ce qu'est le logiciel libre ? C'est par là