FAQ pour les établissements publics

Leo Vivier, Bastien Guerry

2020-11-20

Comment opérer sa transition vers l'ouverture ?

La transition vers le libre amène tout d'abord une évolution des vocables : on ne parle plus de « redevance » mais de « co-participation aux coûts de maintenance et de fonctionnement ».

Elle permet aussi de faire évoluer les services à valeur ajoutée pour les établissements et notamment d'accompagner d'autres communautés d'usage à l'international en assurant la pérennité de ces solutions.

Enfin, faire le choix du libre, c'est s'obliger à maintenir un niveau d'excellence tout en rassurant ses membres/clients qu'ils ne seront pas captifs.

Quels bénéfices attendre de l'ouverture des codes sources ?

1. **En contributions** : Ouvrir des codes sources, c'est permettre à d'autres de les tester et de proposer des corrections ou des améliorations.
2. **En confiance** : Si vous mettez en place un espace d'échange avec vos utilisateurs, les dépôts de code source ouverts peuvent permettre de remonter les demandes et d'organiser une **feuille de route ouverte**, augmentant ainsi la confiance de ces utilisateurs.
3. **En décloisonnement** : En ouvrant des codes sources, vous faites que vos développeurs travaillent plus efficacement *entre eux* et vous favorisez les contacts avec d'autres développeurs de l'ESR qui s'inscrivent dans la même démarche (du côté de Renater, d'Esup Portail, etc.)
4. **En attractivité RH** : En mettant en avant des pratiques de développement s'appuyant sur les pratiques du logiciel libre et de l'open source, vous augmentez l'attractivité de votre établissement.

Comment éviter la concurrence de services réutilisant nos logiciels ?

Le déploiement d'une solution en SaaS (Software as a Service) présente de nombreux intérêts pour l'utilisateur, dont celui de ne plus avoir à faire « tourner » les logiciels de son côté.

C'est aussi une démarche intéressante pour le producteur de logiciels, qui gère plus facilement l'évolution de la solution qu'il propose (il ne doit pas gérer plusieurs versions déployées), qui a une vue plus centralisée des usages et des besoins, etc.

En interne, cela implique une responsabilité accrue des équipes de déploiement : le service doit être fiable, disponible en permanence. Cela a souvent pour conséquence un rapprochement entre les

compétences des administrateurs système et celles des développeurs, rapprochement qui peut faire émerger des profils « devops » : qu'il y ait ou non des devops, l'idée centrale est que toutes ces compétences concourent à la fiabilité et à la l'évolutivité de la solution proposée.

Lorsqu'il place le code source de son service en open source, un organisme signifie que la valeur du service proposé repose dans la qualité de l'équipe technique qui l'assure, et dans la fiabilité technique du service fourni – plutôt que dans les « secrets » de fabrication d'un code source qui, de toutes façons, ne transite plus sur les ordinateurs des uns et des autres.

Certes, d'autres peuvent tenter de récupérer le code source du service proposé pour en déployer un autre : mais le coût associé à la mise en concurrence est aussi élevé que les attentes des utilisateurs du service original.

Dans ce contexte, l'ouverture est un gage de confiance supplémentaire plutôt qu'une aubaine pour la concurrence.

Comment choisir ses licences ?

- Les licences doivent être choisies parmi celles du décret : https://www.data.gouv.fr/fr/licences
- Les licences permissives doivent être être considérées en priorité.
- Si vous hésitez entre plusieurs licences permissives, choisissez en fonction des pratiques de l'écosystème : MIT, Apache ou BSD.
- Si vous pouvez argumenter et dire qu'il serait dangereux pour l'intérêt général d'utiliser une licence permissive au lieu d'une licence à réciprocité, alors vous pouvez choisir de la GPL, LGPL ou AGPL ou MPL qui est à copyleft faible.
- Le https://guide-juridique-logiciel-libre.etalab.gouv.fr peut aussi aider à choisir entre licences permissives et à réciprocité.
- Attention à ne pas tomber dans l'illusion consistant à croire que c'est la licence qui fait la communauté : il n'y a pas adéquation systématique entre une licence à réciprocité forte (type AGPL) et l'existence d'une communauté (cf. les dépôts de beta.gouv.fr, qui sont friands de cette licence).

A noter que la licence CeCILL est peu connue.

Il y a deux critères à prendre en compte:

- Choisir une licence copyleft ou pas. A priori pour un logiciel métier, il faut faire normalement le choix du copyleft. En revanche pour des bibliothèques logicielles de bas niveau, plus techniques, il est parfois utile de prendre une licence permissive pour qu'elle s'impose encore davantage dans tous les univers informatiques.

- Dans le cas du cloud, il y a des licences qui imposent la publication du code modifié (habituellement on n'est pas tenu de le faire), et qui peuvent être utilisées, telle l'affero:

https://www.gnu.org/licenses/why-affero-gpl.fr.html

Le passage en *cloud* peut, en effet, avoir une incidence sur le choix des licences, mais il présente des avantages significatifs pour le constructeur.

De manière générale, le *cloud* permet de lever un certain nombre d'inquiétudes relatives à la maintenance des solutions logicielles. En effet, plutôt que d'avoir à maintenir plusieurs versions concurrentes en production dans différents établissements, le *cloud* permet de concentrer la maintenance et l'assistance sur une seule version. C'est donc un vecteur de simplification.

Doit-on permettre des restrictions à la réutilisation ?

Non : le code est soit ouvert, soit fermé.

Une personne disposant d'un code libre (qu'elle l'ait acheté ou non) peut le donner à qui il veut « pour aider son voisin ». Bien qu'il existe des moyens de dissuasion aux allures de chantage (« c'est dans votre intérêt de ne pas distribuer le code »), il n'est pas concevable de limiter la distribution d'un code source puisque cela s'opposerait à la 3ème liberté garantie par les licences libres (voir).

Paradoxalement, une licence de type *copyleft* fort, c'est-à-dire une ouverture franche qui rende impossible de refermer le code, est un bon moyen de défense dans une communauté naturellement ouverte comme le monde universitaire.

Plutôt que de restreindre l'utilisation, il faut se concentrer sur les services autour du logiciel (voir) : si un établissement n'a pas reçu ces services, il ne pourra pas facilement utilisé la solution.

L'absence de documentation, de guide d'installation, etc., fait partie des mauvaises pratiques qui visent à empêcher un logiciel libre d'être libre. De façon absurde, ces stratégies donnent chez les libristes une mauvaise image des souches que l'on cherche à « valoriser ».

Est ce que mon établissement conserve la propriété du code produit en passant en licence libre ?

Oui. On reste propriétaire du code. Mais on ne peut empêcher personne de l'utiliser, pour tous usages. Comme le théorème de Pythagore!

Il n'y a pas de contradiction, au contraire: cela la renforce.
En cas de contestation de propriété sur un "logiciel non libre" (notamment si on est "volé"), il faut apporter la preuve qu'on était bien le propriétaire. La preuve est à la charge du propriétaire.

Mais en cas de logiciel libre, il n'y a pas de vol: c'est de la contrefaçon. En cas de contestation, la charge de la preuve est à la charge de celui qui est accusé de contrefaçon de prouver que ce n'est pas le cas. Dans ce cas, on notera que la forge fait bien preuve et d'autre part, la charge de la preuve est inversée (le "voleur" doit prouver qu'il est propriétaire).

On remarquera que cette notion de propriété intellectuelle est particulièrement bien protéger par nos lois françaises. C'est grâce à notre industrie du luxe qui a depuis très longtemps à faire à des contrefaçons...

Aussi, le signe le plus tangible de cette propriété, c'est la maîtrise de la feuille de route du logiciel: les décisions d'arbitrages sur le code qui, même en ligne, est ouvert à la participation. Mais n'importe qui ne décide pas!

Quel est le meilleur moment pour libérer le code de ses logiciels ?

Dès la 1ere ligne de code !

Travailler ensemble, et s'installer sur une forge n'a rien de naturel. Ne pas le faire dès le début impliquerait de refaire le travail une seconde fois, doublant ainsi les coûts de recherche et de développement.

Toutefois, si le projet est nativement co-construit, modulaire, mutualisé et installé dans une forge, cet effort est limité.

La libération du code doit être faite le plus tôt possible. Mais en fait cela ne sert à rien si on ne peut pas en tirer parti. Or la modularité, la documentation du code, et la capacité à animer la "communauté" est nécessaire à cela.

Cependant, il est cependant vain de partager du code qui n'est pas modulaire, c'est-à-dire du code qui ne peut pas être réutilisés dans d'autres applications. C'est notamment le cas de Lutèce dont la seconde version, en s'attachant à la modularisation, a su conquérir davantage de développeurs que la première.

Comment protéger les parties sensibles d'un code source ?

S'il s'agit de données sensibles qui doivent rester confidentielles (pour des raisons de sécurité, de respect des données personnelles ou de respect des droits de propriété intellectuelle tiers), celles-ci ne doivent pas être publiée, et une réflexion sur l'architecture du logiciel doit permettre d'éviter cette publication tout en ouvrant le reste du code.

S'il s'agit de codes sources sur lesquels un tiers détient des droits de propriété intellectuelle en interdisant l'ouverture (par exemple un module sous licence propriétaire), alors vous ne devrez pas publier ces parties et vous assurer qu'elles sont modulaires et n'empêchent pas la publication des autres parties du code source.

Notez qu'à tout moment vous avez le choix, fichier par fichier, de la licence que vous souhaitez voir appliquée aux codes souces qui vous appartiennent. Il existe des moyens juridiques de faire cohabiter du code *copyleft* fort, du code *copyleft* faible et du code propriétaire.

Peut-on publier des paramétrages de solutions propriétaires ?

Oui, les paramétrages en tant que tels, qu'ils s'expriment ou non sous forme de lignes de code (un fichier de configuration par exemple) vous appartiennent.

Comment s'assurer qu'un établissement utilise la version « garantie » ?

Cela peut se faire par des éléments techniques, notamment en s'assurant que la version en production sur un établissement est bien celle garantie pour le support et l'assistance.

Il est également possible de considérer une exploitation mutualisée (ou *cloud* qui permettrait de savoir, pour celui qui exploite et offre le service d'assistance, que la version utilisée est bien celle garantie.

Pourquoi et comment ouvrir le code de solutions anciennes ?

L'ouverture de solutions anciennes présente les intérêts de l'ouverture en général (confiance, décloisonnement, attractivité) moins l'avantage des contributions, qui ne sont plus sollicitées, à moins qu'elles ne portent sur des considérations d'architecture ou de fonctionnalités.

La méthode pour ouvrir ces solutions anciennes est la même que pour ouvrir les solutions récentes : vous pouvez les publier sur une forge si elles ont été développées en utilisant un système de gestion de versions. Sinon, un dépôt d'archive (.tar.xz, .tar.gz, .zip) peut être suffisant.

Dans tous les cas, la publication d'une version ancienne (legacy) n'a de sens que si vous pouvez publier en même temps la documentation développeur qui l'accompagne.