Pour les logiciels libres dans l'administration publique

Définir et mettre en oeuvre votre stratégie logiciels libres dans l'administration publique

L'équipe de la mission logiciels libres

Table des matières

1 Présentation 1

2 Utiliser des logiciels libres 2

2.1 Trouver des logiciels libres répondant à vos besoins 2
2.1.1 Identifier : diverses sources 2
2.1.2 Analyser : vérifier la licence d'un logiciel 2
2.1.3 Analyser : les autres aspects 3
2.2 Trouver de l'expertise sur une solution libre 4
2.3 Faire évoluer une solution libre 4
2.4 Se repérer dans l'écosystème des logiciels libres 4
2.4.1 Dans l'administration publique 4
2.4.2 Hors administration publique 5
2.5 Quelle attention porter aux modèles économiques des entreprises ? 5
2.5.1 Modèles économiques des entreprises du numérique libre ... 6
2.5.2 *Openwashing* 7
2.6 Le marché public pour le logiciel libre 8
2.6.1 Marchés interministériels support et expertise à l'usage des logiciels libres 8
2.6.2 Plus de détail sur le marché public pour le logiciel libre 8

3 Publier un logiciel libre 10

3.1 Cadre juridique et appui ministériel 10
3.1.1 Régime juridique du logiciel 10
3.1.2 Pour qu'un code source soit communicable 11
3.1.3 Licences applicables à la publication d'un code source 11
3.1.4 Licences : les indispensables à connaître 11
3.1.5 Doit-on prioriser le choix d'une licence permissive ? 15
3.1.6 Guide juridique interactif 16
3.2 Quels degrés d'ouverture pour les codes sources ? 16
3.3 Quels logiciels ouvrir à quel degré ? 16
3.4 Responsabilité de l'administration publique 17
3.4.1 Responsabilité en cas de produits défectueux 17
3.4.2 Responsabilité en cas de contrefaçon 17
3.5 Où et comment publier votre code source ? 18
3.5.1 Sur quelle forge et dans quel compte publier votre code source ? 18
3.5.2 Bonnes pratiques de nommage des organisations/groupes et dépôts 18
3.5.3 Comment faire connaître et valoriser votre logiciel libre ? .. 18
3.6 Promouvoir votre projet de logiciel libre 18

4 Contribuer à un logiciel libre 19
4.1 TL;DR 19
4.2 En savoir plus 19

5 Comprendre les licences libres 20

6 Comprendre les modèles économiques du libre 21

7 Comprendre les logiques communautaires du libre 22

8 Monter un Open Source Programme Office 23

9 Participer au mouvement BlueHats 24

10 Foire aux questions 25
10.1 Généralités sur les logiciels libres 25
10.1.1 Qu'est-ce qu'un logiciel libre ? 25
10.1.2 Qu'est-ce qu'un fork ou une « dérivation » ? 25
10.1.3 Quelle différence entre « algorithme public »et « code source » ? 25
10.1.4 Quelle est la différence entre GitHub, GitLab, SourceHut ? 26
10.1.5 En tant que citoyen, puis-je exiger d'un organisme public qu'il publie un code source ? 26
10.1.6 En tant qu'agent, ai-je le droit de contribuer à un projet libre ? 26
10.1.7 Qu'est-ce qu'un Administrateur Ministériel des Données, des Algorithmes et des Codes sources ? 26
10.1.8 Comment contacter la mission logiciels libres ? 26
10.2 Utiliser des logiciels libres 27
10.2.1 Qu'est-ce que le socle interministériel de logiciels libres ? 27
10.2.2 Comment mesurer la maturité d'un logiciel libre ? 27
10.2.3 Existe-t-il des formations aux logiciels libres dans l'administration ? 27
10.2.4 Comment m'assurer que le titulaire d'un marché me livre les codes sources ? 27
10.2.5 Puis-je exiger un logiciel libre dans un marché public ? ... 27
10.2.6 Qu'est-ce que le Software Bill Of Materials (SBOM) ? ... 28
10.2.7 Où trouver les codes sources publiés par mon ministère ? 28
10.3 Publier un logiciel libre 28

10.3.1 Comment vérifier qu'un logiciel libre n'est pas déjà développé ? ... 28
10.3.2 Comment faire connaître le logiciel libre que mon administration développe ? ... 28
10.3.3 Comment construire une communauté open source autour de son projet ? ... 28
10.3.4 Quelles sont les bonnes pratiques de développement d'un logiciel libre ? ... 29
10.3.5 Quel processus de contribution mettre en place pour mon projet libre ? ... 29
10.3.6 Quelle gouvernance mettre en place pour un logiciel libre ? ... 29
10.3.7 Quelles langues utiliser pour mon code source et ma documentation ? ... 29
10.3.8 Où trouver des entreprises capables de développer un logiciel libre ? ... 29
10.3.9 Quels points vérifier avant d'ouvrir un code source existant ? ... 30
10.3.10 Qui peut m'aider à publier les codes sources de mon organisme public ? ... 30
10.3.11 Est-il interdit de publier ses codes sources sur github.com ou gitlab.com ? ... 30
10.3.12 Quelle forge dois-je choisir pour publier mes codes sources ? ... 30
10.3.13 Suis-je obligé de permettre la contribution sur mes dépôts ? ... 31
10.3.14 Puis-je publier un code que je ne maintiens plus ? ... 31
10.3.15 Le prestataire doit-il m'envoyer le code source qu'il a développé pour moi ? ... 31
10.3.16 Existe-t-il une forge interministérielle publique ? ... 31
10.3.17 Pouvez-vous m'aider à utiliser Git ? ... 31
10.3.18 Comment détecter et effacer des secrets dans mon historique Git ? ... 31
10.3.19 Deux administrations développent la même chose, que faire ? ... 32
10.3.20 À quoi sert la plateforme ecosystem.code.gouv.fr ? ... 32
10.4 Contribuer à un logiciel libre ... 32
10.4.1 Une administration peut-elle « sponsoriser »un logiciel libre ? ... 32
10.4.2 En tant qu'agent de l'État, puis-je contribuer à un logiciel libre existant ? ... 32
10.4.3 Comment mettre l'interface d'un logiciel libre au système de design de l'État ? ... 32
10.5 Comprendre les licences libres ... 33
10.5.1 Quelle licence libre utiliser pour publier des codes sources de l'administration ? ... 33
10.5.2 Le code source hérite-t-il de la licence du langage de programmation ? ... 33

10.5.3 Puis-je créer une marque pour protéger mon logiciel libre ? 33
10.5.4 Puis-je interdire la réutilisation commerciale des codes sources publiés ? 33
10.5.5 À qui appartiennent les droits d'auteur d'un logiciel développé par une administration ? 34
10.5.6 Qu'est-ce qu'un Copyright License Agreement (CLA) ? .. 34
10.5.7 Qu'est-ce qu'un Developer Certificate of Origin (DCO) ? 34
10.6 Comprendre les modèles économiques du libre 34
10.7 Comprendre les logiques communautaires du libre 34
10.8 Monter un Open Source Programme Office 34
10.8.1 Qu'est-ce qu'un Open Source Program Office (OSPO) ? .. 34
10.8.2 Une administration peut-elle faire de l'« inner source » ? 34
10.8.3 Quelle gouvernance open source mettre en place dans un organisme public ? 34
10.8.4 À quoi sert le site code.gouv.fr ? 35
10.9 Participer au mouvement BlueHats 35

11 Glossaire 36

11.1 Algorithme 36
11.2 Algorithme public 36
11.3 Bibliothèque 36
11.4 Codes sources 36
11.5 Commit 36
11.6 Commun numérique 36
11.7 Dépendances logicielles 36
11.8 Dépôt de code source 37
11.9 Étoiles (dans GitHub ou GitLab) 37
11.10 Forge 37
11.11 Fork 37
11.12 Génie logiciel 37
11.13 Intégration continue 37
11.14 Licence 37
11.15 Logiciel 37
11.16 Logiciel libre 38
11.17 Organisation et groupe (dans GitHub ou GitLab) 38
11.18 Pull/merge request 38
11.19 Réutilisations 38
11.20 Secteur public 38
11.21 Socle interministériel de logiciels libres 38
11.22 Software Heritage 38
11.23 Tag 39
11.24 Ticket 39

12 Concept Index 40

1 Présentation

Cette documentation est publiée par la mission logiciels libres de la direction interministérielle du numérique.

Vous pouvez la lire en ligne et en PDF.

Elle s'adresse à plusieurs audiences :

- les missions logiciels libres ou *Open Source Programme Offices* des administrations publiques (voir cette liste) ou les administrations qui souhaitent en déployer ;
- les équipes métier et chefs de projet qui veulent *publier* des codes sources sous licence libre ;
- les équipes métier et chefs de projet qui souhaitent *contribuer* à des projets libres existants ;
- les DSI qui cherchent à renforcer *l'utilisation* des logiciels libres dans leurs systèmes d'information ;

Tout agent public peut proposer des modifications en suivant ces instructions.

Cette documentation ne porte pas sur les logiciels non libres. Elle ne porte pas non plus sur les bonnes pratiques de développement, à moins que celles-ci soit propres au libre ou inspiré de pratiques issues des communautés open source.

Si ces sujets sont nouveaux pour vous, un Chapitre 11 [glossaire], page 36 est disponible.

2 Utiliser des logiciels libres

2.1 Trouver des logiciels libres répondant à vos besoins

Trouver un logiciel libre peut se décomposer en deux étapes :

- Identifier des projets existants
- Analyser et comparer les divers projets identifiés

2.1.1 Identifier : diverses sources

Lorsqu'il s'agit de trouver un logiciel libre, la multiplication des pistes est une bonne option. Les deux pistes les plus courantes sont internet et les communautés. Elles sont bien évidemment complémentaires.

1. Internet

   La mission logiciels libres maintient plusieurs sources disponibles en ligne :

   - code.gouv.fr/awesome liste les logiciels libres développés par des administrations, à fort potentiel de réutilisation et à forte valeur ajoutée pour tout le secteur public ;
   - code.gouv.fr/sources liste les données de tous les dépôts de codes sources publiés par l'administration publique sur des forges ;
   - code.gouv.fr/sill liste les logiciels libres utilisés par des organismes publics en recommandant une version minimale et en permettant de contacter les référents et les utilisateurs du logiciel.

   L'utilisation des moteurs de recherche peut aussi aider. Avec les mots clefs « open source » et « logiciel libre », les résultats sont en général satisfaisants. Néanmoins, des recherches plus abouties sont nécessaires : aller sur le site web principal du logiciel, vérifier si la licence est indiquée ; mieux encore, trouver le dépôt de code source du logiciel et y vérifier quelle licence est apposée sur le code.

   Enfin, il est aussi possible d'aller sur des forges comme github.com, gitlab.com, sr.ht et des plateformes de distribution dédiées à un écosystème comme pypi.org (Python) ou npmjs.com (JavaScript) pour trouver des modules logiciels. Dans les deux cas, vérifiez bien que la licence est libre, c'est-à-dire approuvée telle par la Free Software Foundation et l'Open Source Initiative.

2. Les communautés

   Quel que soit votre contexte, vous pouvez demander à vos pairs s'ils ont connaissance de logiciels libres adaptés à votre domaine et vos besoins.

   Vous pouvez également solliciter les Bluehats sur leurs canaux d'échanges.

2.1.2 Analyser : vérifier la licence d'un logiciel

Avant d'utiliser un logiciel, il est nécessaire de vérifier la licence afin d'éviter d'utiliser un logiciel sans en avoir le droit.

Une recherche sur internet permet de localiser le site de présentation et de mise à disposition du logiciel. Une fois l'URL identifiée, une recherche du mot « license » limitée au site permet de localiser une page indiquant sous quelle licence le logiciel est mis à disposition. Par exemple, pour le projet LibreOffice :

- une recherche de '`LibreOffice`' donne l'url du site projet : `http://www.libreoffice.org` ;
- une recherche de '`license site:https://www.libreoffice.org/`' donne la page `https://www.libreoffice.org/about-us/licenses` qui contient effectivement les éléments d'informations relatifs à la licence.

S'il n'est pas possible de trouver une information directe et fiable sur la licence d'un logiciel avec la technique précendente, il faut se rendre sur le dépôt de code source du logiciel. La licence doit se trouver soit à la racine du dépôt soit dans un répertoire '`LICENSES`'.

2.1.3 Analyser : les autres aspects

Si la licence est l'élément indispensable que l'on vérifie avant d'aller plus loin, il en est trois autres à ne pas négliger : les fonctionnalités, la sécurité et les communautés.

1. Fonctionnalités

   Il n'est pas rare qu'un logiciel ne couvre pas entièrement les besoins fonctionnels visés. Et c'est là que la magie opère : vous pouvez au projet et ainsi participer à l'effort collectif, récompenser la communauté.

   Il est donc utile de distinguer au sein des fonctionnalités attendues, celles qui sont couvertes et celles qui ne le sont pas. Parmi ces dernières, il est utile de mettre en balance leur criticité pour vous et leur moyen d'intégration dans le projet (ex. "Est-ce facile à intégrer via des plugins ou cela nécessite-t-il une modification du coeur ?").

2. Sécurité

   La sécurité est un sujet que l'on ne peut plus ignorer. Cet aspect d'un projet peut être analysé sur la base de divers critères. On pourra analyser deux facettes :

   - Les dépendances : Est-ce que le projet fait appel à des dépendances à jour ? Est-ce que le projet utilise des outils pour assurer une mise à jour régulière ?
   - Le code lui-même : Est-ce que le code contient des vulnérabilités connues ?

   Des outils existent et permettent d'analyser le code automatiquement comme le score-card de la OpenSSF.

   Cette dernière dresse la liste des critères qu'elle utilise pour labelliser des projets. C'est une base de départ solide pour découvrir ce sujet.

3. Communautés

   Il existe a minima deux types de communautés : celles qui utilisent et celles qui développent le projet. Les unes comme les autres peuvent être plus ou moins grandes.

   Bien évidemment, la présence et la taille des communautés ne devraient pas freiner l'utilisation d'un projet. Dans tous les cas, nous vous conseillons d'analyser la situation afin de savoir dans quoi vous vous engagez.

   Voici quelques exemples :

   - Un projet entouré d'une grande et forte communauté de développement nécessite peu d'investissement. Par contre, il est plus compliqué d'influencer sa trajectoire.
   - Un projet entouré d'une petite communauté pourrait demander plus d'investissement mais en contrepartie, vous aurez la liberté de choisir quand ou où investir.

Pour analyser l'aspect communautaire d'un projet, il existe plusieurs métriques intéressantes. Le projet CHAOSS en répertorie de nombreuses et les documente bien.

2.2 Trouver de l'expertise sur une solution libre

Le site code.gouv.fr/sill est un bon point de départ : non seulement vous pouvez contacter le référent ou la référente du logiciel libre qui, de fait, le connaît et l'utilise dans son administration, mais le site recense aussi des prestataires sur le logiciel spécifique capables de vous fournir de l'expertise.

Les marchés interministériels support et expertise à l'usage des logiciels libres est aussi une bonne ressource.

De façon plus générale, il existe plusieurs consortiums et ensembles d'organisations qui tentent de rassembler l'expertise sur les solutions libres. Ces structures (CNLL et Hub Open Source) participent au conseil logiciels libres.

2.3 Faire évoluer une solution libre

Vous êtes libre de prendre le code source d'un logiciel libre et de l'adapter à vos besoins spécifiques tant que vous respectez la licence. Vous contribuez ainsi à son amélioration pour le bénéfice de tous.

2.4 Se repérer dans l'écosystème des logiciels libres

Pour se repérer dans cet écosystème complexe, voici quelques liens :

- Connaître et comprendre les fondements du logiciel libre
- Connaître et comprendre Section 3.1.4 [les indispensables des licences libres], page 11
  - Explorer les licences :
    - Utiliser l'outil de comparaison des licences de l'UE
    - Utiliser l'outil de code.gouv.fr/sources pour explorer les licences les plus utilisées et créées par l'administration.
- Explorer les communautés des différents logiciels ou écosystèmes qui ont chacune des façons différentes d'interagir, de communiquer, de participer (par exemple la constitution de la communauté Debian)
- Suivre l'actualité du logiciel libre (gazette BlueHats, Linux Magazine, LinuxFr.org, lwn.net, les sites d'organisations et associations sur le fediverse, sur l'instance fosstodon, par exemple, ou encore les lettres d'informations de Framasoft, de l'April, etc.)

2.4.1 Dans l'administration publique

Dans l'administration publique, il existe la communauté BlueHats, qui rassemble les agents publics qui s'intéressent/utilisent/contribuent aux logiciels libres dans/par/pour l'administration publique, en France et ailleurs.

Initiée par la DINUM fin 2018, elle est animée par la mission logiciels libres qui organise ou accueille des ateliers et des rencontres. Les administrations sont invitées à prendre part à ce mouvement et peuvent solliciter la mission pour co-organiser des ateliers ou des rencontres.

2.4.2 Hors administration publique

En dehors de l'administration publique, l'écosystème du logiciel libre est animé par des associations et entreprises du libre.

On notera les associations fondatrices du mouvement logiciel libre par la *Free Software Foundation*, et de l'open source avec l'*Open Source Initiative*.

Il y a des fondations structurantes de l'écosystème des logiciels libres orientées commerce, industrie et/ou grand public :

- Linux Foundation, un consortium à but non lucratif visant à protéger et standardiser le noyau Linux en procurant les ressources pour concurrencer les autres systèmes d'exploitation.
- OW2, un consortium visant à développer une base de logiciel d'infrastructure open source.
- Apache Software Foundation, dont le projet emblématique est le serveur HTTP Apache et sa licence, est une communauté de développeurs open source.
- La Mozilla Foundation, dont le projet emblématique est Firefox et sa licence MPL, vise à promouvoir un internet sûr et ouvert pour tous en suivant son manifeste.

D'autres fondations et associations soutiennent un projet libre en particulier :

- The Document Foundation portant le projet LibreOffice et le format ouvert ODF.
- GNOME Foundation portant le projet GNOME, un environnement de bureau entièrement libre.
- La Fondation Matrix portant le projet Matrix, un protocole ouvert pour des communications décentralisées et sécurisées.

Des associations sont plus spécifiquement ancrées géographiquement :

- Free Software Foundation Europe, promouvant le logiciel libre au niveau de l'Union européenne.
- Framasoft, en France, promouvant le logiciel libre, et une société libre et décentralisée.
- L'AFUL, l'Association Francophone des Utilisateurs de Logiciels Libres.
- L'April, en France, promouvant le logiciel libre pour une société libre.
- L'ADULLACT, soutenant l'action des Administrations et Collectivités territoriales dans le but de « promouvoir, développer et maintenir un patrimoine de logiciels libres utiles aux missions de service public. »

Cette liste ne prétend pas être exhaustive mais donne une idée de la structuration de l'écosystème, de sa taille, et de sa diversité. Une liste plus complète a été rédigée sur le _langages.

2.5 Quelle attention porter aux modèles économiques des entreprises ?

Nous abordons ici les modèles économiques des entreprises du logiciel libre dans la mesure où ces modèles exigent une attention particulière de la part des administrations publiques.

2.5.1 Modèles économiques des entreprises du numérique libre

Notamment, elles doivent prendre en compte les Chapitre 4 [CLA et DCO], page 19 mis en perspective avec les modèles économiques des entreprises avant de contribuer à leur projet.

Une attention particulière doit être portée au CLAs. Par exemple, l'entreprise Element (derrière le protocole Matrix et l'application Tchap) fait signer un CLA avec une exception à l'AGPL pour pouvoir vendre du code source contribué par des auteurs extérieurs à Element sous une licence propriétaire (Article 2 du CLA).

Lorsque vous souhaitez utiliser du logiciel libre dans votre parc d'infrastructure, plusieurs entreprises du libre peuvent répondre à vos différents besoins, chacune avec des modèles différents, qui ne sont pas mutuellement exclusifs.

La liste suivante n'est pas exhaustive. Pour plus de détail, nous vous redirigeons vers ces documents :

- Le livret bleu du CNLL
- Le dossier de l'Aful
- Cette étude, revue par les pairs, de Nicolas Jullien et Robert Viseur, en particulier le tableau page 23 qui identifie 8 modèles économiques en fonction des différents modes de captation de valeur et des types d'activités.

1. Services de déploiements

   L'un des modèles est de valoriser des logiciels libres via une offre SaaS (*Software as a Service*) : l'entreprise fournit un service de déploiement de logiciel libre managé dans le *cloud*. Par « SaaS » ou « managé » on entend que tout est pris en charge : la maintenance et les mises à jour des machines et de toute la pile logicielle. En général, cela vient avec une garantie de disponibilité, un *Service Level Agreement* (SLA).

2. Intégrateur logiciel

   L'intégrateur logiciel propose des services pour exploiter le logiciel libre sur la totalité de son cycle de vie. Il réemploie le code source communautaire existant et accompagne ses clients dans le déploiement du logiciel, que ce soit sur site, sur le cloud, ou simplement sur les postes de travail. Il personnalise aussi en fonction des attentes de ses clients (personnalisation graphique, mais aussi ajout de fonctionnalités spécifiques, etc.).

   Suivant la licence du logiciel de base, l'intégrateur peut être en mesure d'ajouter des couches propriétaires si le client l'exige. Néanmoins, cela n'est généralement ni dans l'intérêt du client, ni dans l'intérêt de l'intégrateur puisqu'ils s'éloigneraient des bénéfices de la mutualisation des efforts ; il est plus intéressant de fournir les ajouts sous licence libre.

   L'intégrateur tire profit de l'intégration de la solution logiciel dans l'environnement du client, mais aussi dans les conseils qu'il peut lui apporter, et dans la maintenance applicative.

3. Éditeur logiciel

   L'éditeur logiciel libre édite et distribue des produits sous une licence libre. De là, on peut distinguer trois façons de faire du profit.

   1. Le modèle *Open Core*

      Le modèle *Open Core* consiste à éditer un logiciel de base sous licence libre et vendre des extensions propriétaires, ou vendre des outils de développement propriétaires au-dessus du logiciel. Dans ce modèle la version libre est souvent appelée

la « version communautaire », ou « CE » pour *Community Edition* en opposition à « EE » pour *Entreprise Edition*.

Un exemple du premier cas est Gitlab ou Odoo. Un exemple du second cas est Zend qui vend son environnement de développement Zend Studio PHP.

2. Le modèle double licence

   Un modèle à double licence signifie qu'un code source est disponible sous deux licences, en général une libre et une autre propriétaire. L'utilisateur choisit l'une ou l'autre licence. L'idée est souvent de proposer une licence de type copyleft et une licence non libre (ou « commerciale »), cette dernière préférée par les utilisateurs ou les entreprises voulant éviter les contraintes de réciprocité des licences copyleft.

   Il est aussi possible qu'une solution logicielle ne soit pas sous double licence par défaut, mais qu'il y ait un changement au cours du temps. Par exemple :

   1. une licence propriétaire chronodégradable en licence libre ;
   2. une licence propriétaire comportant une clause de réversibilité en licence libre si, par exemple, l'entreprise est amenée à disparaître.

   Attention : ce modèle à double licence ne doit pas être confondu avec le fait, pour un dépôt de code source, de publier des éléments sous des licences distinctes. Par exemple, un dépôt peut publier le code sous licence GPL-3.0-or-later et la documentation sous FDL-1.3. Dans ce cas, l'utilisateur doit accepter les deux licences.

3. L'open source professionnel

   L'open source professionnel (terme employé par le CNLL dans son livret bleu) désigne les autres moyens qu'une entreprise peut tirer profit à partir d'un logiciel libre.

   Cela peut venir du support, de la maintenance, de la documentation, du conseil, de formations, etc. Pour avoir des revenus récurrents, une entreprise peut facturer du support forfaitaire, des garanties juridiques et de fonctionnement.

2.5.2 *Openwashing*

Le logiciel libre domine le marché des serveurs et autres utilités de développement comparé aux logiciels propriétaires. Vu comme plus éthique, beaucoup d'entreprises se vendent comme étant « open source » alors qu'elles ne publient pas de code libre.

Openwashing, est dérivé du mot *greenwashing* (et tous les autres mots-valises en *-washing*). Le mot *fauxpen* signifie la même chose :

> Description d'un logiciel qui prétend être open source, mais qui ne dispose pas de toutes les libertés requises par la définition de l'Open Source Initiative [ou de la FSF].

La question fondamentale à se poser pour savoir si c'est un projet est libre : la licence garantit-elle les quatre libertés fondamentales (étudier, copier, modifier, redistribuer) ou répond-elle aux critères de la définition de l'OSI ?

Si oui, c'est un logiciel libre. Si non, ce n'est pas un logiciel libre.

Pour vous faciliter la vie, l'OSI maintient une liste de licences acceptées.

2.6 Le marché public pour le logiciel libre

Une administration publique peut passer commande sur la prestation de services sur des logiciels libres *explicitement nommés*. Tant que cela ne relève pas de la marque du logiciel, le nommer explicitement ne contrevient pas au principe d'égalité de traitement des candidats.

De plus, une administration peut demander le développement de logiciels libres *spécifiquement* en prévoyant la dévolution des droits d'auteurs. L'article 46 du CCAG-TIC prévoit cette dévolution des droits, permettant la préservation d'une mutualisation sous licence libre.

2.6.1 Marchés interministériels support et expertise à l'usage des logiciels libres

La Direction Générale des Finances Publiques (DGFIP) pilote deux marchés interministériels à l'usage des logiciels libres : le marché support et le marché d'expertise.

Ces deux marchés ont pour objet de couvrir l'ensemble du cycle de vie d'un logiciel libre au sein d'un système d'information, en constant échange avec les communautés de ces logiciels libres. Notamment, les marchés obligent la redistribution de tous les correctifs issus de ses activités.

Pour en savoir plus sur les marchés, rendez-vous ici.

2.6.2 Plus de détail sur le marché public pour le logiciel libre

À défaut des logiciels privatifs, un logiciel libre peut être utilisé, copié, modifié, par n'importe qui, y compris des entreprises concurrentes proposant des services autour d'un logiciel libre. Dans ce cadre-là, exiger un logiciel libre précis ne déroge en rien aux principes de libertés d'accès et d'égalité de traitement du Code de la commande publique. Le logiciel libre, *par définition*, garantit le principe d'égalité.

La commande publique, en revanche, ne sera pas passée sur *l'acquisition* d'un logiciel libre, mais sur la *prestation* de service autour de ce logiciel libre. Sauf rare exception, on n'acquiert pas un logiciel libre puisque l'on en dispose librement. Dans ce cas, l'appropriation du logiciel libre échappe aux règles de la commande publique.

Une administration, dans le cadre d'un marché public, **peut inclure dans les clauses contractuelles l'exigence d'une solution numérique basée sur des logiciels libres**.

En effet, l'aspect libre d'un logiciel, déterminé par sa *licence libre*, est une caractéristique juridique. Rien ne s'oppose à ce que la commande publique requiert des solutions logicielles avec comme caractéristiques juridiques la possibilité de les étudier, copier, modifier, et redistribuer.

En revanche, une commande publique portant sur le développement d'un logiciel libre est un cas particulier à prendre en compte. Deux points d'attention :

D'abord la dévolution des droits de propriété intellectuelle doit être prévue par une clause spécifique. L'article 46 du CCAG-TIC prévoit cette dévolution des droits permettant la préservation d'une mutualisation sous licence libre.

Ensuite, vient la question de l'égalité de traitement des candidats. Ce cas est plus délicat lorsqu'une entreprise est déjà engagée dans la gouvernance d'un logiciel libre que l'administration pourrait être amenée à passer commande. Néanmoins, cela ne saurait remettre en cause le principe d'égalité de traitement des candidats, puisque le logiciel étant

libre, chacun est libre de créer un *fork* et d'avoir droit de *commit* par défaut. La décision du Conseil d'État du 30 septembre 2001 va dans ce sens.

Aussi, certains textes de lois priorisent les logiciels libres, comme l'article 9 de la loi n° 2013-660 du 22 juillet 2013 relative à l'enseignement supérieur et à la recherche modifiant l'article L123-4-1 du Code de l'éducation

3 Publier un logiciel libre

3.1 Cadre juridique et appui ministériel

Toute entité chargée d'une mission de service public doit publier tout document produit ou reçu dans le cadre de cette mission, quelle qu'en soit la date, le lieu de conservation et le support. Les codes sources, en tant que documents administratifs, relèvent de cette obligation (voir l'avis CADA du 8 janvier 2015 n ° 20144578).

Les codes sources concernés sont, au même titre que n'importe quelle autre donnée administrative publiable en open data, celles « dont la publication présente un intérêt économique, social, sanitaire ou environnemental. »

Pour les licences, voir les articles L323-2 et D323-2-1 du Code des relations entre le public et les administrations.

Au sein des ministères, les AMDACs (Administrateurs Ministériels des Données, des Algorithmes et des Codes sources) sont chargés de faire appliquer ce cadre légal et de mettre en place les politiques de publication des codes sources que leurs ministères et opérateurs développent ou font développer.

3.1.1 Régime juridique du logiciel

Le logiciel, comme oeuvre de l'esprit est couvert automatiquement (sans formalité particulière) par le droit d'auteur.

Le droit d'auteur est constitué des **droits patrimoniaux** ou droits d'exploitations (équivalent au copyright anglo-saxon) et de **droits moraux**.

Toute personne utilisant, copiant, modifiant ou diffusant le logiciel sans autorisation explicite du détenteur des droits patrimoniaux est **coupable de contrefaçon et passible de trois ans d'emprisonnement et de 300 000 € d'amende** (Art. L. 335-2 du CPI)

Concernant le logiciel, le droit d'utilisation ouvre de manière encadrée (Art. L122-6-1 du CPI), les possibilités de :

- Corriger des erreurs (sauf si l'auteur s'en réserve le droit dans une licence)
- Réaliser une copie de sauvegarde si celle-ci est nécessaire à la préservation de l'utilisation du logiciel
- Analyser le fonctionnement externe du logiciel
- Reproduire et traduire du code dans un but d'inter-opérabilité avec d'autres applicatifs

La protection au titre des droits patrimoniaux est limitée dans le temps (Pour la France, 70 ans après le décès de l'auteur (personne physique) ou de la première publication (personne morale). Au delà, le logiciel, pour une version donnée, **s'élève dans le domaine public**, il est utilisable par quiconque sans aucune restriction.

Les droits moraux, quant à eux, sont inaliénables. Pour le logiciel, cela se résume au respect du nom des auteurs ayant travaillé sur logiciel.

Néanmoins, lorsque des «logiciels et leur documentation [sont] créés par un ou plusieurs employés dans l'exercice de leurs fonctions ou d'après les instructions de leur employeur» (Art. L113-9 du Code de la propriété intellectuelle), les droits patrimoniaux sont automatiquement dévolus à l'employuer. Dans le cadre des adminisrtations publiques, lorsrque

les développements sont réalisés sur le temps professionnel et son en lien direct avec les instructions de l'employeur, au sens de l'article L113-9, les droits patrimoniaux sont alors automatiquement dévolus à l'État. De plus, l'auteur voit ses droit moraux diminués selon l'article L121-7 du Code de la propriété intellectuelle et perd donc son droit de se repentir ou de retrait, et son droit au respect de l'œuvre.

3.1.2 Pour qu'un code source soit communicable

- L'obligation de communicabilité porte sur les collectivités de plus de 3500 habitants et les organismes publics de plus de 50 agents.
- L'organisme public ouvrant le code source doit en avoir la propriété intellectuelle.
- Le code source doit être « achevé » : dès lors qu'une version du code est mise en oeuvre dans l'administration, cette version est considérée comme « achevée ». Notamment une version dite bêta ou inférieure à 1.0, si elle est effectivement utilisée, est bien achevée et communicable.
- Sa communication ne doit pas porter atteinte :
  - au secret commercial et industriel ;
  - à la sûreté de l'État, à la sécurité publique, à la sécurité des personnes ou à la sûreté des systèmes d'information des administrations ;
  - à la recherche et à la prévention, par les services compétents, d'infractions de toute nature.

En dehors de ces limites, toute personne ou toute administration peut demander la communication d'un code source.

3.1.3 Licences applicables à la publication d'un code source

Afin d'éviter la prolifération des licences, la loi pour une République numérique a prévu la création d'une liste, fixée par décret, de licences qui peuvent être utilisées par les administrations pour la réutilisation à titre gratuit (Art. D.323-2-1 du CRPA).

Cette liste est accessible ici.

3.1.4 Licences : les indispensables à connaître

Une licence logicielle est un contrat passé entre les auteurs d'un logiciel et ses réutilisateurs. Les licences libres accordent aux utilisateurs le droit d'étudier, copier, modifier, redistribuer le code source d'un logiciel.

L'utilisation d'une licence libre permet de sécuriser et simplifier la relation entre le ou les auteurs et les utilisateurs explicitant leurs droits, prévenant les litiges, et la contractualisation individuelle pour chaque utilisateur.

Une fois en possession du logiciel, à titre onéreux ou gratuit, l'utilisateur a l'obligation de se conformer à la licence l'accompagnant, sachant que **tout ce qui n'est pas explicitement autorisé est interdit**.

Pour les licences libres, la liberté d'utiliser et de modifier le logiciel est inconditionnelle, aucune limitation ou contrainte ne pèse sur l'utilisateur tant que le logiciel reste à l'intérieur de son organisation. En revanche, en cas de redistribution à l'extérieur de son organisation, les obligations de licences doivent être respectées au risque d'être coupable de contrefaçon.

1. Licences permissives

   La redistribution d'un logiciel sous licence permissive avec ou sans modification peut se faire sous une autre licence. Par exemple, des composants du système d'exploitation FreeBSD sous licence libre BSD sont utilisés pour réaliser le système d'exploitation Mac OS X. L'ensemble est redistribué sous une licence propriétaire.

   Exemple de licences permissives autorisé pour les administrations par décret :

   - Licence Ouverte version 2.0 (etalab-2.0)
   - Apache License 2.0 (Apache-2.0)
   - BSD 3-Clause "New" or "Revised" License (BSD-3-Clause)
   - CeCILL-B Free Software License Agreement (CECILL-B)
   - MIT License (MIT)

2. Le « copyleft »

   Le mot « copyleft » est un jeu de mots avec le mot « copyright » (le droit d'auteur aux États-Unis). Ce terme est révélateur du mouvement du logiciel libre qui, au lieu de se battre contre le *copyright*, a utilisé ses mécanismes de protection des œuvres pour garantir les libertés essentielles des utilisateurs.

   Le *copyleft* va plus loin que de simplement donner les quatre libertés aux logiciels : il oblige la **réciprocité** en interdisant l'ajout de restrictions sur les libertés utilisateurs. Ce sont des licences dites à réciprocité ou « diffusives ».

   La licence GPL est l'exemple paradigmatique d'une licence copyleft. D'autres sont :

   - GNU Affero General Public License v3.0 or later (AGPL-3.0-or-later)
   - Mozilla Public License 2.0 (MPL-2.0)
   - European Union Public License 1.2 (EUPL-1.2)

   Les licences copyleft se distinguent des licences permissives qui, elles, autorisent l'ajout de restrictions au code redistribué.

   Les obligations des licences copyleft diffèrent selon que la licence est à Section 10.4 [copyleft faible ou fort], page 32.

   Légère précision sur un malentendu régulier :

   L'ajout de restrictions ne se fait pas sur la copie du logiciel originel. La copie d'un logiciel X publiée sous une licence libre, **le restera pour toujours** (à condition que l'auteur détienne les droits et l'originalité pour revendiquer ses droits d'auteur).

   Le code source Y ajouté au code source X (sur une autre copie du code X) publié avec une licence permissive, peut être re-distribué sous une licence plus restrictive, voire, propriétaire. Cependant, rien ne changera la copie originel du code source X restant sous sa licence permissive, à condition que le ou les auteurs ne changent pas sa licence.

   1. Différence entre copyleft faible et fort

      La notion de copyleft *faible* ou *fort* se réfère aux obligations plus ou moins fortes appliquées aux personnes voulant redistribuer une œuvre.

      Le copyleft *fort* exige que la redistribution de l'œuvre, qu'elle soit modifiée ou non, ainsi que les logiciels liés, soit effectuée sous la même licence, (ou une licence à copyleft fort compatible).

A contrario, le copyleft *faible* n'impose pas les logiciels liés à être distribués sous la même licence, mais impose toute redistribution du logiciel à l'être sous la même licence (ou une licence compatible).

Une image vaut mille mots :

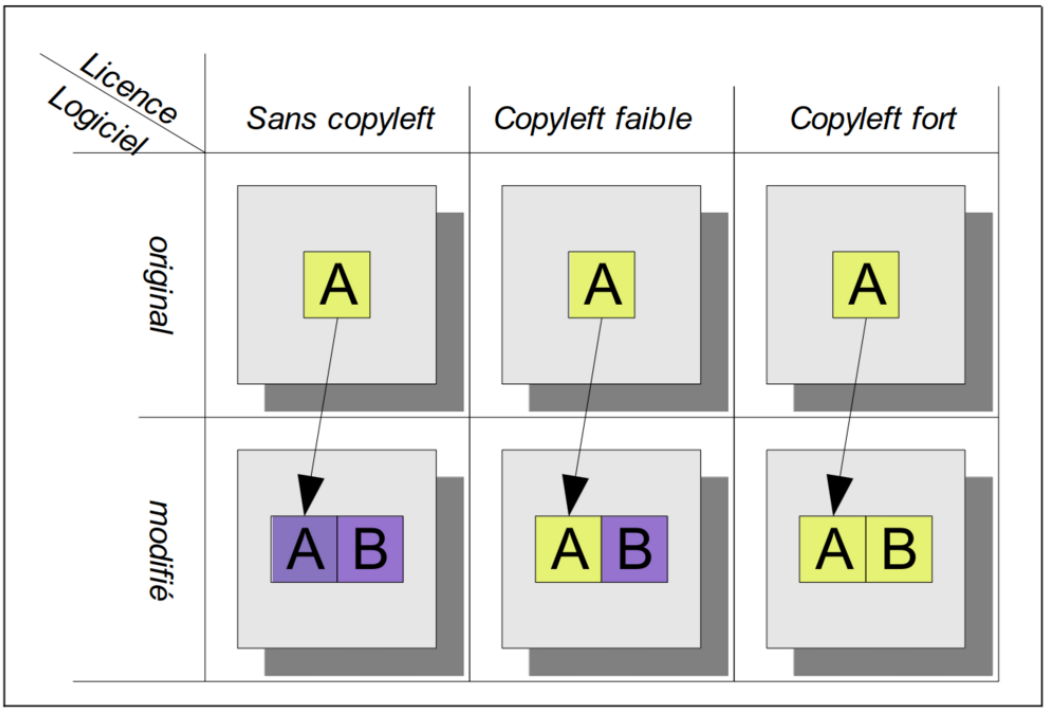

Figure 3.1: Diffusivité des différents types de licence (la couleur correspond à la licence)

Un logiciel lié désigne tout composant assemblé avec le logiciel final lors de l'édition de lien. En générale, ce sont des bibliothèques logicielles, qui, seules, n'ont pas de grande utilité, répondant à des fonctions de bases, mais nécessaires au fonctionnement d'un logiciel complet.

Le copyleft faible est souvent utilisé pour les bibliothèques logicielles permettant une réutilisation plus simple de la bibliothèque et l'ajout de composants logiciels sous différentes licences, potentiellement privatrices.

3. Compatibilité entre licences libres

La compatibilité des licences libres est une questions qui a été étudié par Benjamin Jean dans son livre *Option libre* (9782953918748. hal-04136860), duquel nous en tirons la table de compatibilité entre licences suivante (page 316) :

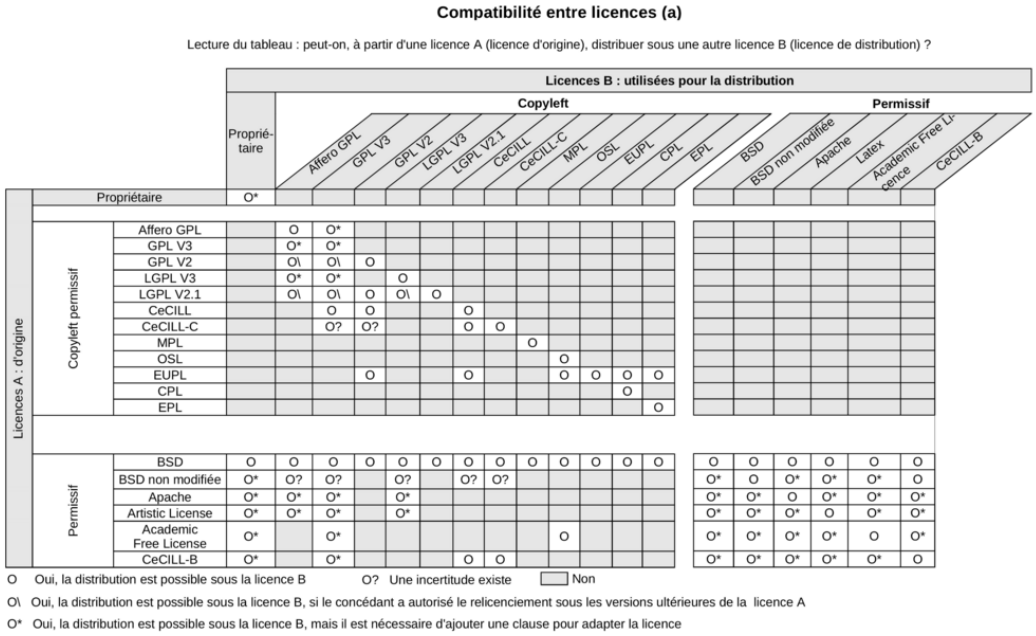

Figure 3.2: Table de compatibilité entre licences

Aussi, il existe aussi le Joinup Licensing Assistant de l'UE qui est un outil simple pour déterminer en fonction de la licence du projet ou du bout de code qu'une administration publique souhaiterait intégrer à son projet.

Un élément important à remarquer est que **la compatibilité a un sens** : un composant sous licence A peut être compatible **vers** une licence B, mais la réciproque n'est pas nécessairement vraie.

Par exemple, un composant sous licence EUPL peut-être redistribué sous licence GPL v2. En revanche, un composant sous licence GPL v2 ne peut pas être redistribué sous licence EUPL.

Le principe général est que la licence du logiciel ne peut pas conférer plus de droits et moins d'obligations que les licences de chacun des composants ; on parle de compatibilité logique.

Illustrons ce principe avec l'exemple d'une application que l'on souhaite publier sous GPL V2 et intégrant un composant sous licence Apache. L'ensemble des droits accordés sur le composant au titre de la licence Apache est intégralement repris par la GPL V2. Par contre certaines obligations de la licence Apache, ne sont pas exigées par la licence GPL V2, en matière de brevet particulièrement. Il n'est donc pas possible d'utiliser un composant sous licence Apache dans une application publiée sous GPL V2. Avec la nouvelle GPL V3 cette incompatibilité n'existe plus.

Cependant, une incompatibilité logique peut être levée par un accord spécifique auprès du détenteur des droits patrimoniaux du composant que l'on souhaite intégrer. Cela suppose de prendre contact avec la communauté en charge du composant. Il est probable qu'un accord sera trouvé sous la forme d'une exception spécifique. Il arrive même qu'une clause d'exception adjointe à la licence du composant règle l'incompatibilité.

La question de la compatibilité n'existe véritablement que lorsque l'on publie un logiciel sous une licence de type copyleft fort, soit par choix soit parce qu'un composant du logiciel est déjà sous copyleft fort. Le tableau montre, au moyen du triangle, la zone d'influence ou la licence GPL s'impose. Au delà il y a incompatibilité. Par exemple la présence d'un composant sous licence EPL est incompatible dans un logiciel sous GPL (ou sous CeCILL V2).

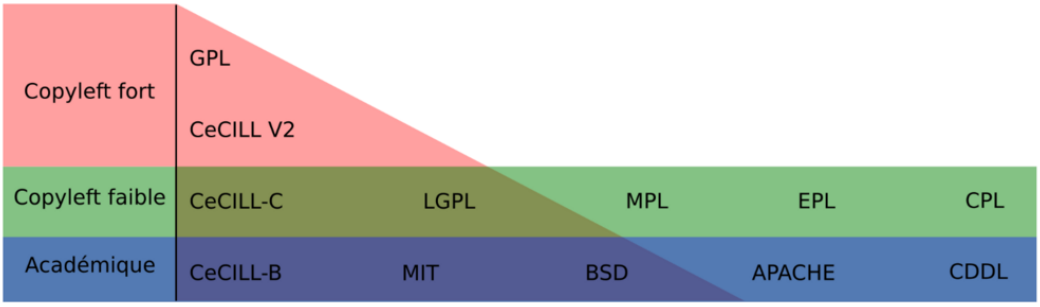

Figure 3.3: Compatibilités entre licences libres populaires avec du copyleft fort

Un logiciel composé de briques sous licences de type copyleft faible est possible. Ce n'est pas forcément facile à gérer car chaque composant va garder sa licence propre. Il faudra respecter chacune d'entre elles. Si cela est possible, on pourra re-licencier chaque composant sous une licence globale compatible, c'est-à-dire garantissant l'ensemble des droits conférés par chacune et respectant les obligations de chacune.

3.1.5 Doit-on prioriser le choix d'une licence permissive ?

Oui. La doctrine de la DINUM sur les licences à utiliser pour la publication des codes sources est d'utiliser des licences permissives. Les libertés octroyées par ces licences permettent en tout temps à n'importe quel acteur de réutiliser le code produit par des agents publics, et ce, même à des fins lucratives et/ou d'intégration dans un logiciel propriétaire.

Si la réutilisation et l'intégration d'un code source dans un logiciel propriétaire est considéré comme une menace avérée pour l'intérêt général, alors un choix de licence à réciprocité (ou « copyleft ») est conseillé. L'évaluation des risques pesant sur « l'intérêt général » est laissée à l'appréciation des administrations.

Par exemple, une mission de service public finance le développement d'un logiciel A, publie son code source, et en fait un service pour les autres administrations. Ensuite, une entreprise privée prend ce code source A, l'améliore en code source B, et vend un service SaaS (*Software as a Service*) basé sur B aux administrations. L'État aura alors payé deux fois le service, la mission de service public n'aura plus de raison d'exister, et les améliorations faites par l'entreprise ne seront pas redistribuées. Dans ce cas de figure, mettre le code source A sous la licence AGPL (qui oblige la redistribution des contributions sous la même licence même lorsque le logiciel est distribué en SaaS) est fortement conseillé.

Pour plus de détails sur le copyleft, se référer à [cette section], page 12. Attention, le copyleft n'empêche pas la vente des codes sources.

3.1.6 Guide juridique interactif

Pour savoir si le code source d'un logiciel développé et utilisé par votre organisme public est communicable, nous vous invitons à tester ce guide juridique interactif.

3.2 Quels degrés d'ouverture pour les codes sources ?

- Niveau A - contributif : Le code source est publié, les contributions extérieures sont activement recherchées et traitées.
- Niveau B - ouvert : Le code source est publié, les contributions extérieures sont traitées mais non activement recherchées.
- Niveau C - publié : Le code source est publié mais les contributions extérieures ne sont pas traitées.
- Niveau D - non-communicable : Le code source n'est pas communicable au public.

Au début du fichier `README.md` d'un dépôt, vous pouvez ajouter l'un de ces badges pour prévenir vos utilisateurs :

```
https://img.shields.io/badge/code.gouv.fr-contributif-blue.svg
https://img.shields.io/badge/code.gouv.fr-ouvert-mediumseagreen.svg
https://img.shields.io/badge/code.gouv.fr-publi%C3%A9-orange.svg
```

Si votre fichier `README` est écrit en markdown, vous pouvez ajouter le badge avec un lien vers cette documentation :

```
[![img](https://img.shields.io/badge/code.gouv.fr-contributif-blue.svg)](https://code.
```

3.3 Quels logiciels ouvrir à quel degré ?

Tous les logiciels développés par un organisme public n'ont pas vocation à être ouverts au même degré. Pour définir votre stratégie et adopter le bon degré d'ouverture, nous vous proposons ces questions :

1. Le logiciel est-il **un module utile à d'autres logiciels libres** (vs un logiciel « monolithique » sans utilité pour d'autres logiciels libres) ?
2. Le logiciel répond-il a un **besoin générique** (vs à un besoin spécifique à l'organisme qui le produit) ?
3. Le logiciel doit-il bientôt être **maintenu et développé par d'autres** (vs votre administration s'engage sur du long terme) ?
4. L' **utilisateur final** du logiciel a-t-il un **profil technique** (développeur, datascientiste ou designer vs un utilisateur non-technique) ?

Le niveau A est recommandé pour les logiciels répondant à au moins deux critères ; le niveau B est recommandé pour ceux répondant à au moins un critère ; le niveau C pour ceux ne répondant à aucun de ces critères (par ex. un logiciel métier très spécifique, dont aucune partie ne peut être réutilisée ailleurs, qui n'a pas vocation à être repris par d'autres et dont les utilisateurs ne sont pas du tout des contributeurs potentiels.)

Pour les logiciels ne répondant à aucun de ces critères, le niveau D est admissible, tant qu'aucun citoyen n'exige la communication du code source en question, selon le cadre juridique défini dans la loi pour une République numérique.

Bien sûr, ces critères sont **relatifs** : la modularité, la généricité, le besoin de reprise par d'autre et le potentiel de contribution des utilisateurs ne s'évaluent pas *in abstracto*. Ces notions sont proposées pour aider à **prioriser les ouvertures logicielles**. Le but est de **canaliser votre énergie** sur les logiciels qui ont un bon potentiel contributif et **de communiquer clairement** sur la posture de l'administration dans le cas des publications simples.

3.4 Responsabilité de l'administration publique

3.4.1 Responsabilité en cas de produits défectueux

Quelle est la responsabilité engagée par une collectivité publique (État ou collectivité locale) qui met à disposition un logiciel sous licence de logiciel libre ?

Généralement licences libres et licences propriétaires de logiciel rejettent toutes responsabilités quant aux dommages directs et indirects que pourraient causer l'utilisation du logiciel. Une telle clause est-elle compatible avec le droit français ?

En droit français, la limitation, voire l'exonération de responsabilité, est autorisée en matière contractuelle. La protection du consommateur suppose néanmoins que l'exclusion totale de responsabilité ne soit pas admise quand le contrat est passé avec un consommateur (art. L.132-1 du code de la consommation).

Il en est de même pour les produits défectueux, l'article 1386-15 du code civil ne permettant pas que soit écartée par voie contractuelle la responsabilité de ce fait, sauf entre professionnels.

Dans la mesure où le logiciel s'adresse manifestement à des professionnels et des informaticiens, et c'est le cas des applications portées par les administrations, l'exclusion de responsabilité pour les dommages directs est ainsi admise.

3.4.2 Responsabilité en cas de contrefaçon

Concernant la responsabilité de l'administration en matière de contrefaçon, le risque existe même lorsque le logiciel n'est pas diffusé comme logiciel libre ; mais une diffusion large expose plus facilement à ce risque.

Contrefaçon en matière de droit d'auteur : le logiciel diffusé inclut un composant ou même un bout de code source pour lequel l'administration n'a pas les droits de diffusion. La responsabilité de l'administration est engagée. Toutefois si le logiciel a été produit dans le cadre d'un marché public, il conviendra de rechercher la responsabilité du prestataire coupable de négligence ou même plagiaire sur les développements spécifiques dans le **rapport de conformité**.

Le risque de différends entre l'administration engagée dans une démarche de mutualisation et les acteurs du logiciel libre est très faible et devrait se résoudre à l'amiable tant les objectifs des uns et des autres convergent.

Contrefaçon en matière de marque : une marque est un signe distinctif (logo), un mot ou un groupe de mots servant de reconnaissance légale pour un produit, une société, etc. Il est de la responsabilité de l'administration, de s'assurer que la mise à disposition du logiciel ne contrefait pas une marque déposée. En particulier concernant le nom du logiciel, il faudra vérifier qu'il n'empiète pas sur une marque déposée. D'une façon générale, la mutualisation d'un logiciel doit se faire en marque blanche, sans signe distinctif autre que celui de l'administration.

Contrefaçon en matière de brevet : Les brevets logiciels en tant que tels, en France et en Europe n'ont pas de reconnaissance juridique. La Convention sur le brevet européen (CBE) l'indique clairement dans son article 52.

3.5 Où et comment publier votre code source ?

3.5.1 Sur quelle forge et dans quel compte publier votre code source ?

TBD.

3.5.2 Bonnes pratiques de nommage des organisations/groupes et dépôts

Un bon nom de dépôt décrit la finalité du code source du dépôt.

Un bon nom d'organisation décrit l'équipe qui porte les dépôts.

Il vaut mieux plusieurs organisations avec des noms stables que peu d'organisations avec des mauvais noms.

Le nom d'organisation doit être explicite et minimaliste :

- évitez les acronymes correspondant à une entité administrative, sauf si vous êtes certain que cet acronyme va perdurer dans le temps ;
- éviter de préfixer ou suffixer un nom d'organisation avec un acronyme administratif.

Exemple de mauvais nom : `https://github.com/DISIC/` car il était prévisible que l'acronyme ne serait plus d'actualité.

Exemple de bon nom : `https://github.com/etalab/` car la marque perdure.

3.5.3 Comment faire connaître et valoriser votre logiciel libre ?

Si votre logiciel libre gagne à être connu et déployé par d'autres organismes publics, vous pouvez envisager de proposer qu'il rejoigne la liste Awesome code.gouv.fr.

3.6 Promouvoir votre projet de logiciel libre

4 Contribuer à un logiciel libre

4.1 TL;DR

Une administration **peut contribuer** à un logiciel libre. Un point d'attention doit être porter sur comment les droits d'auteurs sont gérés par le projet auquel l'administration veut contribuer.

Si le projet est géré par un **DCO** (*Developer Certificate of Origin*), c'est simple : chaque contributeur doit avoir l'accord de sa hierarchie, et signer avec un simple *sign-off* chacun de ses *commits*.

Si le projet est géré par un **CLA** (*Contributor Licence Agreement*), le service juridique de l'administration devra lire, signer, et garder le CLA de chaque contributeur.

4.2 En savoir plus

La contribution de l'administration à un logiciel libre, qu'il soit communautaire ou édité par une entreprise privée, requiert, dans certains cas, un DCO ou un CLA.

Ces contrats ou ces *agreement* sont un moyen, plus ou moins simple, de donner un accord d'utilisation des contributions des développeurs à l'entité gérant le projet et de lui permettre d'utiliser et de distribuer ces contributions sous sa licence.

Le **CLA**, *Contributor Licence Agreement*, est un document légal devant être signé par le contributeur clarifiant les termes et conditions de sa contribution, établissant qu'il a le droit de contribuer (le contenu lui appartient, son employeur a donné l'accord, etc.) *et* que le projet a le droit d'utiliser ce contenu (changer de licence sur le contenu, le redistribuer). Cela permet au projet de se protéger contre de potentielles attaques en justice en lien avec le droit d'auteur des contributions.

ICLA et **CCLA** sont des déclinaisons plus spécifiques du CLA, *Individual Contributor Licence Agreement* et *Corporate Contributor Licence Agreement* respectivement. Le ICLA concerne les individus contribuant en leur nom propre en dehors de toute organisation ou employeur. Le CCLA concerne la contribution d'une entreprise sur le projet d'une autre entreprise. En général, ces documents légaux sont basés sur la CLA de la fondation Apache.

Certains CLA permettent de sous-licencier des contributions sous des licences propriétaires. Par exemple, l'entreprise Element (derrière le protocole Matrix et l'application Tchap) fait signer un CLA avec une exception à l'AGPL pour pouvoir vendre du code source contribué par des auteurs extérieurs à Element sous une licence propriétaire (Article 2 du CLA d'Element)

Parce que les CLAs sont des documents légaux, le département juridique doit se charger de les signer et de garder une trace de ces éléments, rendant le processus lourd.

Par conséquent, la fondation Linux, et plusieurs autres organisations qui ont suivi, sont passées au **DCO**, *Developer Certificate of Origin*. Celui-ci n'est pas un contrat légal, mais un mécanisme plus simple indiquant qu'un contributeur a le droit de contribuer son code et qu'il donne son accord pour que ses contributions soient utilisées et redistribuées sous la licence libre choisie par le projet. Un DCO requiert simplement de signer (*sign-off*) chaque commit.

5 Comprendre les licences libres

6 Comprendre les modèles économiques du libre

7 Comprendre les logiques communautaires du libre

8 Monter un Open Source Programme Office

Un Open Source Programme Office (ou une mission logiciels libres) est une entité au sein d'une organisation en charge de définir et de mettre en oeuvre sa stratégie Open Source (ou "logiciels libres").

Une stratégie logiciels libres présente la façon dont l'organisation **utilise** des logiciels libres, **développe** des nouveaux logiciels libres et **contribue** à l'écosystème open source.

La DINUM encourage les administrateurs ministériels de données, algorithmes et codes sources à soutenir la mise en place, au sein de leurs ministères et des opérateurs qui en dépendent, des Open Source Programme Offices. Ceux-ci permettront d'institutionnaliser et de pérenniser les moyens mis pour respecter le cadre juridique actuel et tirer le meilleur parti des logiques de mutualisation à l'oeuvre dans l'écosystème open source.

Voir notre proposition de définition détaillée d'un OSPO et la page où sont listés les OSPOs d'organismes publics français.

9 Participer au mouvement BlueHats

10 Foire aux questions

Vous pouvez aussi naviguer dans cette FAQ depuis code.gouv.fr/faq.

Si vous avez des questions que vous voulez voir figurer dans cette FAQ, écrivez à '`floss@numerique.gouv.fr`'.

10.1 Généralités sur les logiciels libres

10.1.1 Qu'est-ce qu'un logiciel libre ?

Un logiciel est dit libre si son code source est publié sous l'une des licences reconnue libre soit par la Free Software Foundation soit par l'Open Source Initiative. Une licence libre octroie quatre libertés :

- la liberté d'utiliser le logiciel ;
- la liberté de copier le logiciel ;
- la liberté d'étudier le logiciel ;
- la liberté de modifier le logiciel et de redistribuer les versions modifiées.

Voir spdx.org/licenses pour la liste des licences et de leur validation par l'OSI ou la FSF.

10.1.2 Qu'est-ce qu'un fork ou une « dérivation » ?

Il y a deux notions distinctes pour qualifier un "fork". Une notion technique qui a été popularisée par GitHub consistant à faire une copie du code source d'un projet sur lequel des personnes peuvent contribuer sans être dépendantes des mainteneurs du projet originel.

Soit B le fork du code source A : le fork B (ou la « dérivation » B) est une nouvelle version de A dont les versions successives (B2, B3, etc.) s'écarteront des versions successives de A (A2, A3, etc.)

Il y a aussi une notion plus orientée projet. Dans ce cas, un fork est généralement créé lorsque les contributeurs d'un projet sont en désaccord et qu'une partie des contributeurs décide de créer une version divergente.

10.1.3 Quelle différence entre « algorithme public » et « code source » ?

L'expression « algorithme public » désigne de façon relâchée les algorithmes définis et utilisés par une administration et qui relèvent des obligations d'open data. Vous pouvez consulter ce guide d'Etalab à leur sujet. Ces « algorithmes » ne sont pas systématiquement exprimés sous forme de code source.

Un code source est la version lisible par un humain d'un programme informatique : une partie relève de l'algorithmique, d'autres de la documentation, de la gestion de données, etc.

Les obligations de publication des algorithmes publics et les obligations de publication des codes sources ne se confondent pas.

10.1.4 Quelle est la différence entre GitHub, GitLab, SourceHut ?

Il faut d'abord distinguer le logiciel et le service en ligne : github.com et gitlab.com sont les services en ligne délivrés par les entreprises Github et Gitlab Inc. Ces services en ligne sont des SaaS (Software as a Service).

La principale différence entre GitHub et Gitlab se trouve alors dans la licence et le modèle économique.

GitHub propose son service via un logiciel propriétaire; le code n'est pas visible. GitLab Inc. propose son service en partie via un logiciel open source, sous la licence MIT, et en partie via un logiciel *source available* (source lisible, une licence propriétaire). Cela signife que l'on peut voir et étudier le code source, sans pour autant pouvoir le réutiliser librement.

GitHub a un modèle économique classique : c'est une platforme basée sur un logiciel propriétaire. GitLab a un modèle dit *open core* : la version du logiciel libre communautaire (GitLab CE), et une version plus complète avec des fonctionnalités supplémentaires propriétaires payantes disponible sous une licence *source available*.

SourceHut est le nom du projet qui déploie des services autour du nom de domaine `sr.ht`. Ce service utilise uniquement des logiciels entièrement libre. Parmi les forges dont le code source est entièrement libre, SourceHut est la seule qui propose à la fois de l'intégration continue et des listes de discussion. Si vous voulez contribuer à un projet, vous n'avez pas besoin de créer de compte sur SourceHut : il suffit d'une adresse de courriel pour envoyer des correctifs et proposer des idées. SourceHut et son service sr.ht ne collecte aucune donnée de ses utilisateurs.

10.1.5 En tant que citoyen, puis-je exiger d'un organisme public qu'il publie un code source ?

Oui, si la publication de ce code source entre bien dans les obligations de l'administration. Ce guide juridique donne les liens vers les textes pertinents.

10.1.6 En tant qu'agent, ai-je le droit de contribuer à un projet libre ?

Oui, si votre responsable est d'accord, il n'y a aucun obstacle à ce que vous puissiez contribuer à des logiciels libres sur votre temps de travail.

10.1.7 Qu'est-ce qu'un Administrateur Ministériel des Données, des Algorithmes et des Codes sources ?

AMDAC est l'acronyme de « Administrateur Ministériel des Données, des Algorithmes et des Codes sources ». Les AMDACs veillent à appliquer le principe d'ouverture par défaut des données publiques, incluant les codes sources des administrations.

Vous trouverez la liste des AMDACs sur data.gouv.fr.

10.1.8 Comment contacter la mission logiciels libres ?

Vous pouvez nous écrire à '`floss@numerique.gouv.fr`'.

Pour entrer en contact avec d'autres agents publics libristes, voir les espaces de communication entre BlueHats.

10.2 Utiliser des logiciels libres

10.2.1 Qu'est-ce que le socle interministériel de logiciels libres ?

Le socle interministériel de logiciels libres (SILL) est le catalogue des logiciels libres recommandés pour toutes les administrations publiques.

Il est publié par la mission logiciels libres sur code.gouv.fr/sill et tout agent public est invité à s'y créer un compte pour déclarer ses usages de logiciels ou se proposer comme référent d'un logiciel.

Tous les logiciels libres présentés dans le SILL sont déjà en cours d'utilisation, soit par la DSI d'un organisme public soit par un agent public dans le cadre de ses fonctions.

Voir code.gouv.fr/sill/readme pour plus de détails.

10.2.2 Comment mesurer la maturité d'un logiciel libre ?

La fondation OW2 propose un outil de mesure de la maturité Open Source d'un projet, le Market readiness level.

Une autre structure propose une variante, l'Open Source Readiness.

Le projet chaoss.community propose des métriques sur la « santé » d'un projet libre pouvant s'apparenter et/ou compléter ces outils.

10.2.3 Existe-t-il des formations aux logiciels libres dans l'administration ?

Si vous êtes agent public avec un accès à la plateforme Mentor, vous pouvez consulter une capsule introductive produite par la DINUM.

Le site code.gouv.fr liste des offres de formation logiciels libres existantes, mais qui ne ciblent pas spécifiquement les agents publics.

10.2.4 Comment m'assurer que le titulaire d'un marché me livre les codes sources ?

Vous pouvez l'exiger dans votre marché.

En pratique, vous pourrez l'exiger sur tout ou partie du système que vous souhaitez développer et exploiter.

Si vous prévoyez d'ouvrir un code source développé pour vos besoins, vous devez exiger que la propriété de ce code vous soit cédée et qu'il vous soit livré.

Voir l'Arrêté du 30 mars 2021 portant approbation du cahier des clauses administratives générales des marchés publics de techniques de l'information et de la communication.

10.2.5 Puis-je exiger un logiciel libre dans un marché public ?

En tant qu'organisme public, vous avez le droit de publier un marché exigeant un logiciel libre et/ou des services autour d'un logiciel libre.

Si le nom du logiciel est le même que le nom d'une marque portée par une entreprise éditrice, veillez bien à préciser que c'est le logiciel libre qui est exigé, indépendamment de son éditeur.

Voir la section 5.6 du livre Droit des logiciels de Pellegrini et Canevet qui porte sur ce sujet.

10.2.6 Qu'est-ce que le Software Bill Of Materials (SBOM) ?

Un SBOM (*Software Bill of Materials*) est une liste décrivant les composants (libres ou non) dont dépend un logiciel.

Maintenir une telle liste avec les métadonnées associées à chaque composant permet d'avoir des éléments objectifs pour renforcer la conformité légale (le respect des licences) du produit et en vue d'améliorer les pratiques de sécurité internes au projet.

Les deux formats principaux de SBOM sont celui du projet CycloneDX, maintenu par la fondation OWASP, et celui du projet SPDX, maintenu par la fondation Linux.

Il existe de nombreux outils pour générer des SBOMs en respectant l'un ou l'autre de ces formats.

10.2.7 Où trouver les codes sources publiés par mon ministère ?

Vous pouvez chercher sur code.gouv.fr/sources l'organisation qui correspond à votre direction ou, plus largement, à votre ministère.

Vous pouvez aussi parcourir la liste Awesome code.gouv.fr qui présente des logiciels libres à fort potentiel de réutilisation.

10.3 Publier un logiciel libre

10.3.1 Comment vérifier qu'un logiciel libre n'est pas déjà développé ?

Vous pouvez vérifier dans l'inventaire des codes sources et dans le Socle Interministériel des Logiciels Libres.

Vous pouvez aussi vérifier au-delà, sur les forges « grand public ».

10.3.2 Comment faire connaître le logiciel libre que mon administration développe ?

- Si votre logiciel est d'intérêt pour d'autres organismes publics, faites-le référencer dans Awesome code.gouv.fr.
- Consulter cette présentation BlueHats qui propose des pistes.

10.3.3 Comment construire une communauté open source autour de son projet ?

1. Développez un logiciel qui sera utile à d'autres
2. Choisissez une licence libre utile à ces « autres »
3. Identifiez les utilisateurs potentiels
4. Identifiez les contributeurs potentiels
5. Ajoutez un CONTRIBUTING.md pour les guider
6. Allez chercher des utilisateurs
7. Allez chercher des contributeurs
8. Communiquez de façon publique et prédictible
9. Allez chercher votre première contribution

10. Améliorez en itérant sur toutes ces étapes

Vous pouvez faciliter les contributions en publiant un fichier '`CONTRIBUTING.md`' à la racine de votre dépôt ou vous expliquerez aux potentiels contributeurs le moyen de vous aider.

10.3.4 Quelles sont les bonnes pratiques de développement d'un logiciel libre ?

- Choisir une licence : choosealicense.com et le guide DINUM
- Rédiger un fichier README : www.makeareadme.com
- Publier un fichier de ChangeLog : keepachangelog.com
- Utiliser le versionnement sémantique : semver.org
- Rédiger des messages de commit : conventionalcommits.org
- Ajouter un code de conduite

En plus de ces conventions assez consensuelles :

- Utiliser gitmoji pour les commits
- Afficher les contributeurs avec allcontributors.org

10.3.5 Quel processus de contribution mettre en place pour mon projet libre ?

Vous pouvez exiger la signature d'un Developer's Certificate of Origin et/ou un Contributor License Agreement.

La convention est de décrire les modalités de contribution en anglais dans un fichier '`CONTRIBUTING.md`' à la racine du dépôt.

10.3.6 Quelle gouvernance mettre en place pour un logiciel libre ?

Pour mettre en place une gouvernance open source dans un projet, vous pouvez vous référer à ce guide en anglais de la fondation Eclipse.

10.3.7 Quelles langues utiliser pour mon code source et ma documentation ?

Le code source est écrit dans un langage de programmation (par exemple en Javascript). Les commentaires dans le code source sont considérés comme faisant partie du code et doivent être écrits en anglais.

Si le code source est développé en lien avec un référentiel, alors les noms de variable et de fonction doivent reprendre ce référentiel. Par exemple, si le référentiel est en français, les noms de variable et de fonction seront en français.

Le manuel destiné au développeur du projet ou à une personne qui va réutiliser le projet (l'intégrer, le déployer, etc.) doit être écrit en français.

Le manuel destiné à l'utilisateur final doit être écrit en français.

10.3.8 Où trouver des entreprises capables de développer un logiciel libre ?

Il n'y a pas de catalogue centralisé exhaustif, mais des initiatives existent. Notamment, le CNLL regroupe les principales associations et entreprises de l'écosystème open source en France.

Plusieurs entreprises du libre se sont rassemblées pour créer un guichet unique : Open source experts (OSE)

10.3.9 Quels points vérifier avant d'ouvrir un code source existant ?

Au niveau juridique :

- Les licences des dépendances appelées par votre code source.
- Les licences des codes sources modifiés et/ou améliorés par votre code.
- Quelles licences pouvez/voulez-vous utiliser pour votre code ?
- Vos licences choisies sont-elles bien déclarées dans votre code (cf. les conventions de reuse.software) ?

Au niveau de la sécurité :

- Est-ce que l'historique Git de votre dépôt contient des données sensibles ?
- Avez-vous testé les éléments de sécurité de votre code ?

Pour ce qui est de la documentation :

- Avez-vous une documentation pour l'utilisateur final ?
- Avez-vous une documentation pour l'administrateur système ?
- Avez-vous une documentation pour les contributeurs ?

10.3.10 Qui peut m'aider à publier les codes sources de mon organisme public ?

Vous pouvez interroger vos collègues et votre direction pour savoir si vous disposez d'une forge et/ou de comptes d'organisation dédiés où publier vos codes sources.

À défaut de réponse, vous pouvez solliciter l'Administrateur Ministériel des Données, des Algorithmes et des Codes sources de votre ministère. Voir la liste des AMDACs.

Vous pouvez enfin solliciter directement la mission logiciels libres en écrivant à floss@numerique.gouv.fr.

Dès que vous publiez un code développé par votre administration, assurez-vous que la forge et l'organisation via laquelle vous publiez sont référencés sur code.gouv.fr/sources : si ce n'est pas le cas, écrivez-nous pour que nous procédions à ce référencement.

10.3.11 Est-il interdit de publier ses codes sources sur github.com ou gitlab.com ?

Non, il n'y a pas d'obstacle légal à la publication des codes sources d'une administration sur github.com ou gitlab.com.

10.3.12 Quelle forge dois-je choisir pour publier mes codes sources ?

Vous pouvez vérifier sur cette liste si votre organisme public déploie une forge et si oui, contacter les personnes en interne qui pourront vous aider à y publier vos codes sources.

Si vous êtes une administration centrale et souhaitez publier sur une forge interministérielle, vous pouvez contacter les responsables de la forge gitlab.mim-libre.fr.

Si vous souhaitez publier sur une forge hébergée en France via le partenariat que la DINUM a avec l'ADULLACT, vous pouvez contacter les responsables de la forge gitlab.adullact.net.

Sinon, vous pouvez publier votre code sur la forge de votre choix, par exemple gitlab.com, github.com ou SourceHut.

10.3.13 Suis-je obligé de permettre la contribution sur mes dépôts ?

Non. Vous pouvez consulter à ce sujet nos propositions sur les degrés d'ouverture.

10.3.14 Puis-je publier un code que je ne maintiens plus ?

Oui. Dans ce cas, indiquez bien dans le fichier `README.md` que le code source n'est plus maintenu.

Si vous le souhaitez, vous pouvez préciser dans ce `README.md` qu'un nouveau mainteneur est recherché.

10.3.15 Le prestataire doit-il m'envoyer le code source qu'il a développé pour moi ?

Si le contrat prévoit que le prestataire cède ses droits patrimoniaux sur le code source développé pour une administration, il est obligé de vous mettre à disposition ces codes sources.

Nous recommandons d'exiger que ces codes sources soient mis à disposition sur une forge gérée par l'administration dès le premier commit : attendre le versement d'un code source après la fin d'une prestation est une mauvaise pratique.

10.3.16 Existe-t-il une forge interministérielle publique ?

À ce jour, gitlab.mim-libre.fr fait office de forge interministérielle.

Pour les projets des administrations centrales qui ne sont pas ouverts, il existe une forge GitLab privée gérée par la DGFiP.

10.3.17 Pouvez-vous m'aider à utiliser Git ?

Vous trouverez de l'aide en contactant l'un des membres de la communauté BlueHats.

10.3.18 Comment détecter et effacer des secrets dans mon historique Git ?

Adopter les bonnes pratiques dès la création du dépôt git est crucial. Ces bonnes pratiques sont nombreuses, mais notamment utiliser des variables d'environnements pour les secrets plutôt que de les écrire noir sur blanc dans les fichiers commités est un bon réflexe.

Néanmoins, si l'erreur a été faite il existe certains outils :

- TruffleHog sous licence AGPL
- Gitleaks sous licence MIT
- Detect Secrets sous licence Apache 2
- Gitgardian sous licence MIT

10.3.19 Deux administrations développent la même chose, que faire ?

Si vous avez identifié les porteurs de ces projets, envoyez leur un mail pour les mettre en contact en ajoutant '`floss@numerique.gouv.fr`' en copie.

10.3.20 À quoi sert la plateforme ecosystem.code.gouv.fr ?

ecosystem.code.gouv.fr déploie le logiciel libre ecosyste.ms pour collecter des données sur les forges où sont publiés des dépôts d'organismes publics.

À terme, ce sont les données exposées via ecosystem.code.gouv.fr qui seront utilisées pour l'interface d'exploration des codes sources code.gouv.fr/sources/.

10.4 Contribuer à un logiciel libre

10.4.1 Une administration peut-elle « sponsoriser » un logiciel libre ?

Une administration soutenir des structures privées (association, fondation, entreprise) qui développent des logiciels libres et/ou contribuent à des logiciels libres :

- en subventionnant des structures ;
- en adhérant comme membre à des associations ;
- en passant commande.

Ces démarches doivent respecter le cadre légal de la subvention, de l'adhésion et de la commande publiques.

10.4.2 En tant qu'agent de l'État, puis-je contribuer à un logiciel libre existant ?

Si votre hiérarchie est d'accord, oui.

Le document qui acte de cette possibilité de contribuer à des logiciels libres existants est la politique de contribution open source de 2018.

Les droits d'exploitation sur le code source contribué sont transmis de plein droit à l'administration qui vous emploie (Art. L. 131-3-1 du Code de la propriété intellectuelle) : vous devez en tenir compte au moment où vous faites votre contribution. Le projet auquel vous contribuez se trouve dans l'une de ces trois situations :

- Il exige un "Corporate Contributor License Agreement" : dans ce cas, vous devez demander à votre administration de le signer pour que le projet commence à accepter vos contributions.
- Il exige un "Contributor License Agreement" inviduel : dans ce cas, vous devez demander à votre hiérarchie s'il vous est permis de le signer.
- Il exige la signature d'un DCO ou il n'exige rien de particulier : dans ces deux cas, le seul accord de votre hiérarchie suffit.

10.4.3 Comment mettre l'interface d'un logiciel libre au système de design de l'État ?

Vous avez le droit de modifier ou d'enrichir le code source d'un logiciel libre pour que son interface web utilise le système de design de l'État (ci-après "DSFR").

Notez que l'utilisation de ce système de design est strictement encadrée. En cas de doute, référez-vous à ses conditions d'utilisation et prenez contact avec le Service d'information du gouvernement.

Pour trouver des bibliothèques de code implémentant le DSFR, vous pouvez chercher "dsfr" dans l'inventaire des codes sources publics.

Une implémentation du DSFR pour React largement réutilisée est le dépôt react-dsfr.

10.5 Comprendre les licences libres

10.5.1 Quelle licence libre utiliser pour publier des codes sources de l'administration ?

Si vous êtes un agent public ou un organisme public et que vous publiez un logiciel sous licence libre, vous devez utiliser les licences listées sur cette page.

Toutes sont valables en droit français, même si elles ne sont pas toutes rédigées en français.

Si vous tenez absolument à utiliser une licence rédigée en français, vous pouvez utiliser la licence EUPL 1.2 ou l'une des licences CeCILL.

En tant que mission de service public, la loi pour une République numérique exige la publication des codes sources sous l'une des licences référencées à l'article D323-2-2 du Code des Relations entre le Public et les Administrations.

Le portail data.gouv.fr présente ces licences de réutilisations, pour les données comme pour les logiciels.

10.5.2 Le code source hérite-t-il de la licence du langage de programmation ?

Un langage de programmation ne relève pas du droit d'auteur, tout comme la langue française ne relève pas du droit d'auteur. En revanche, l'implémentation d'un langage de programmation relève du droit d'auteur, et une licence peut s'appliquer. Par exemple, le langage Python est sous licence PSFL.

Le code source, qu'il soit compilé ou exécuté, n'est pas une œuvre dérivée du langage de programmation. Le code source n'hérite donc pas de la licence du langage de programmation.

Ce processus est valable de façon plus générale : les droits d'auteur d'un logiciel ne s'appliquent pas aux résultats (ou sorties) dudit logiciel.

Par exemple, lorsque vous convertissez un fichier .odt en PDF via LibreOffice, la licence LibreOffice ne s'applique ni au fichier .odt ni au fichier PDF, n'étant pas des œuvres dérivées du logiciel.

10.5.3 Puis-je créer une marque pour protéger mon logiciel libre ?

Oui.

10.5.4 Puis-je interdire la réutilisation commerciale des codes sources publiés ?

Non, toutes les licences libres que vous pouvez utiliser pour publier votre code source autorisent la réutilisation commerciale de ce code.

10.5.5 À qui appartiennent les droits d'auteur d'un logiciel développé par une administration ?

S'il est développé par des agents de cette administration, les droits patrimoniaux appartiennent à l'administration.

S'il est développé par un prestataire et si le contrat a précisé que l'administration récupère les droits patrimoniaux du logiciel, alors ils appartiennent à l'administration.

10.5.6 Qu'est-ce qu'un Copyright License Agreement (CLA) ?

- https://contributoragreements.org
- https://www.harmonyagreements.org

10.5.7 Qu'est-ce qu'un Developer Certificate of Origin (DCO) ?

Le *Developer Certificate of Origin* est un texte que les contributeurs d'un projet libre sont invités à accepter *avant* de contribuer: il donne la garantie au projet que le contributeur a fait toutes les vérifications nécessaires au sujet de sa contribution.

Voir developercertificate.org qui est le texte du DCO pour le noyau Linux.

Il est d'usage que la signature des commits (avec '`git commit -s`') signifie que le contributeur accepte le DCO déclaré par le projet.

10.6 Comprendre les modèles économiques du libre

10.7 Comprendre les logiques communautaires du libre

10.8 Monter un Open Source Programme Office

10.8.1 Qu'est-ce qu'un Open Source Program Office (OSPO) ?

C'est une entité dans une entreprise ou une administration dédiée à la définition et à la mise en oeuvre d'une stratégie open source pour cette entreprise ou administration.

Voir notre entrée de blog au sujet des OSPOs et la liste des OSPOs du secteur public.

10.8.2 Une administration peut-elle faire de l'« inner source » ?

La notion d'*innersource* désigne l'adoption des pratiques de développement logiciels open source au sein d'une organisation, sans partager les codes publiquement.

Si vous n'êtes pas obligés de publier certains codes sources, vous pouvez les développer via des organisations ou des dépôts privés ou via une forge privée.

La démarche d'*innersource* suppose néanmoins une **visibilité partagée** sur ce qui est développé par les uns et les autres et un encouragement à contribuer aux dépôts partagés.

Pour aller plus loin, vous pouvez lire le livre "Understanding the InnerSource Checklist" publié en 2017 chez O'Reilly Media par Silona Bonewald.

10.8.3 Quelle gouvernance open source mettre en place dans un organisme public ?

Pour mettre en place une gouvernance open source dans une organisation, vous pouvez vous référer à la Good Governance Initiative développée et promue par la fondation OW2. Vous pouvez consulter cet outil permettant de la mesurer, et le déployer.

10.8.4 À quoi sert le site code.gouv.fr ?

Le site code.gouv.fr est le site de présentation de l'ensemble des activités et produits de la mission logiciels libres de la DINUM.

Il donne notamment accès au socle interministériel de logiciels libres et à la liste des codes sources publiés par des administrations.

10.9 Participer au mouvement BlueHats

11 Glossaire

11.1 Algorithme

Un algorithme est la description d'une suite d'étapes permettant d'obtenir un résultat à partir d'éléments fournis en entrée (cf. définition de la CNIL).

En informatique, cette suite d'étape est une suite d'opérations formelles traitant et produisant des informations.

11.2 Algorithme public

Un algorithme *public* est un suite opératoire (formelle ou non, informatisée ou non, automatisée ou non) sollicitée pour une décision administrative individuelle envers des personnes physiques ou morales, de droit public ou privé nommément désignées.

Voir le guide des algorithmes publics à l'usage des administrations.

11.3 Bibliothèque

Dans code.gouv.fr, une bibliothèque est un ensemble de fonctions distribuées sous forme de paquetage via une plateforme dédiée, par exemple `https://npmjs.com`.

Pour ajouter une bibliothèque dans code.gouv.fr, il suffit que le compte d'organisation depuis lequel vous publiez cette bibliothèque soit ajouté à ce fichier.

Vous pouvez écrire à floss@numerique.gouv.fr pour nous indiquer un compte à ajouter.

11.4 Codes sources

Le code source d'un programme informatique est ce qu'écrit une programmeuse ou un programmeur. Il peut s'agir de programmes complexes ou de quelques lignes. Ce code source peut être partagé sous licence libre pour permettre aux autres programmeurs de l'étudier, de le modifier, de le diffuser et de partager leurs améliorations.

11.5 Commit

Unité de modification.

11.6 Commun numérique

Un commun numérique est une ressource disponible sous format numérique, gérée par une communauté qui définit, pour cette ressource, des règles d'utilisation et de contribution, et pour la communauté, des règles de participation.

11.7 Dépendances logicielles

Un logiciel intègre souvent des briques logicielles publiées sous licence libre. Celles-ci sont appelées « dépendances ». Ce site permet de parcourir la liste des dépendances de *mise en production*, non les dépendances de *développement* ; d'autre part, seules sont comprises les dépendances sollicitées par au moins deux dépôts.

Les dépendances listées dans code.gouv.fr sont automatiquement identifiées à partir des dépôts référencés sur cette même plateforme. Ne sont prises en compte que les dépendances de premier niveau.

11.8 Dépôt de code source

Un « dépôt » est un espace dans lequel sont publiés les fichiers de code source. C'est ce que vous voyez lorsque vous visitez un lien vers un code source hébergé sur une forge. C'est aussi ce que vous pouvez copier sur votre machine pour l'explorer localement.

Pour ajouter un dépôt dans code.gouv.fr, envoyez-nous le compte d'organisation GitHub ou le groupe GitLab depuis lequel vous le publiez, nous l'ajouterons dans ce fichier.

Vous pouvez écrire à floss@numerique.gouv.fr pour nous indiquer un compte à ajouter.

11.9 Étoiles (dans GitHub ou GitLab)

Les « étoiles » (« stars » en anglais) sont un moyen pour les utilisateurs des plates-formes de mettre un dépôt en favori. Pour l'instant, nous collectons cette information sur GitHub, GitLab et les instances de GitLab. Ce n'est pas une mesure de la qualité du code source.

11.10 Forge

Outil de développement logiciel collaboratif.

11.11 Fork

Un dépôt « forké » en franglais est un dépôt de code source qui a été développé à partir d'un autre.

11.12 Génie logiciel

Champ de l'informatique s'intéressant à la gestion et au cycle de vie des projets logiciels.

11.13 Intégration continue

Capacité pour une forge de permettre la construction automatique du logiciel depuis l'ensemble de ses sources et en fonction de certains paramètres.

11.14 Licence

Une licence logicielle est un contrat passé entre les auteurs d'un logiciel et ses réutilisateurs. Les licences dites « libres » accordent aux utilisateurs le droit de réutiliser le code source d'un logiciel.

11.15 Logiciel

Un logiciel est un ensemble de séquences d'instructions interprétables par une machine. À la différence d'un code source qui est aussi *un ensemble de séquence d'instructions* (mais lisible par l'humain), les instructions sont en code objet, généralement en binaire.

11.16 Logiciel libre

Un logiciel libre est un logiciel dont le code source est publié sous l'une des licences reconnues libres par la Free Software Foundation ou "open source" par l'Open Source Initiative.

Ces licences ont toutes en commun d'octrayer aux utilisateurs quatre libertés : celle d'*utiliser* le programme informatique comme on le souhaite, pour toute finalité ; celle d'*étudier et de modifier* le programme à loisir ; celle de redistribuer des copies du programme à d'autres ; celle de redistribuer des versions modifiées du programme à d'autres.

11.17 Organisation et groupe (dans GitHub ou GitLab)

GitHub permet d'avoir des comptes personnels pour y héberger du code et des « comptes d'organisation ». Un « groupe » est la notion plus ou moins équivalent sur les instance de GitLab. Un organisme remplissant une mission de service public peut avoir un ou plusieurs organisations et/ou groupes sur une ou plusieurs forges. p Pour ajouter une organisation dans code.gouv.fr, il suffit que le compte d'organisation GitHub ou le groupe GitLab soit ajouté dans ce fichier.

Vous pouvez écrire à '`floss@numerique.gouv.fr`' pour nous indiquer un compte à ajouter.

11.18 Pull/merge request

Proposition de révision. *Merge request* est l'expression utilisée sur GitLab. *Pull request* est l'expression utilisée sur les autres forges.

11.19 Réutilisations

GitHub permet de connaître le nombre de dépôts qui en utilisent un autre : le nombre de ces dépôts est présenté ici dans la colonne "Réutilisations" de la liste des dépôts.

11.20 Secteur public

Les codes sources développés dans le cadre de missions de service public ont vocation à être publiés, dans certains conditions. Ce site propose de chercher dans l'ensemble des codes sources aujourd'hui identifiés comme provenant d'un organisme remplissant une mission de service public. Il a été développé par Etalab.

11.21 Socle interministériel de logiciels libres

Le socle interministériel de logiciels libres (SILL) est le catalogue de référence des logiciels libres recommandés par l'Etat pour toute l'administration.

Voir le site du SILL.

11.22 Software Heritage

Initiative internationale visant à conserver pour l'Histoire les codes source des logiciels dont le code source est public.

11.23 Tag

Dans un dépôt de code source géré avec Git, un tag est un label associé à un commit. Ce label peut être annoté ou non. Un tag correspond en général à une nouvelle version du logiciel.

11.24 Ticket

Déclaration en ligne d'un incident ou d'un dysfonctionnement, ou proposition d'amélioration du logiciel.

12 Concept Index

A

algorithme ... 36
algorithme, public ... 25, 36
AMDAC ... 26

B

bibliothèque ... 36

C

CLA ... 29
code source ... 36
code.gouv.fr ... 32, 35
commit ... 36
commun numérique ... 36
communauté ... 28, 29
contribuer ... 29, 31, 32
contributeurs ... 28

D

DCO ... 29
dépendance ... 36
dépôt ... 37
dérivation ... 25
dsfr ... 32

F

forge ... 30, 37
fork ... 25, 37
formation ... 27

G

GitHub ... 26
github.com ... 30
GitLab ... 26
gitlab.com ... 30
gouvernance ... 29, 34

I

innersource ... 34
intégration continue ... 37

L

licence ... 33, 37
logiciel ... 37
logiciel libre ... 25, 38

M

maintenance ... 31
marché public ... 27
maturité ... 27
mission ... 26
MR, PR ... 38

O

OSPO ... 34

P

PR, MR ... 38
publier ... 30

S

SBOM ... 28
secteur public ... 38
sill ... 27, 38
socle interministériel de logiciels libres ... 38
Software Heritage ... 38
SourceHut ... 26

T

tag ... 39
ticket ... 39