

# Contenu et objectifs du cours

Présenter comment faire correctement (et mal) de l'open source

## Objectifs:

- reconnaître et corriger les problèmes tôt;
- éviter les pièges communs;
- gérer la croissance et maintenance d'un projet.

Focalisation sur les problèmes associés au développement de logiciel libre.

⚠ Les logiciel libre / open-source n'ont pas le monopole des exigences irréalistes, des spécifications vagues, de l'ignorance des retours utilisateurs, ...

# Organisation du cours

**Cours:** 7 séances de 3 heures (pas de TD / TP)

**Planning:** 13/09 + 20/09 + 27/09 + 04/10 + 11/10 + 18/10 + 25/10

**Exercice pratique:** contribution à [https://github.com/SMoraisDev/DLL\\_2023](https://github.com/SMoraisDev/DLL_2023)

**Examen:** 14/11

-> questions de cours & mock de contribution open source

# Plan du cours

- I – Introduction à l'open source et au libre
- II – Démarrer un projet
- III – Infrastructure technique
- IV - Infrastructure sociale
- V – Communication
- VI – Packaging, publication et développement quotidien
- VII – Divers (licences, modèle économique, gestion des contributeurs)

# Références

- <https://www.gnu.org/> (système d'exploitation et mouvement du logiciel libre)
- <https://producingoss.com> (livre sur le développement open source)
- <https://openhatch.org> (archive: organisation aidant à contribuer dans l'open source)
- <http://teachingopensource.org/> (actualité / blog posts sur l'enseignement open source)
- <http://open-advice.org/> (conseils de développeurs open source)
- <http://oss-watch.ac.uk> (conseil sur l'utilisation, le développement et les licences open source)

# Qui êtes vous ?

- Nom / prénom
- Activités en entreprise ?
- Langage de programmation pour l'examen (Python / Java) ?
- Rapport à l'open source ?

# **I – Introduction aux logiciels libres / open source**

## **I.1 - Introduction**

## **I.2 - Un peu d'histoire**

## **I.3 - Échange oral / début : l'open source**

# I.1 - Introduction

? Dans quelle mesure sont visible / présent les logiciels libres / open source ?



# Dans quelle mesure sont visible / présent les logiciels libres / open source ?

- Très largement **invisible** et cela même aux personnes travaillant dans la tech  
-> la nature de l'open source est de **passer inaperçu** sauf par ceux dont le travail le touche directement
- Un **pilier des technologies de l'information modernes** qu'on trouve dans la téléphonie (Pine64), les ordinateurs portables / bureaux (Linux, Firefox), l'automobile (Genivi)
- Particulièrement présent sur les serveurs fournissant des services en ligne sur Internet (e.g. Facebook: Cassandra, MySQL, Tornado)

# Qu'attendre du passage à l'open source?

# Qu'attendre du passage à l'open source?

- Une erreur commune consiste à avoir des attentes irréalistes sur les avantages de l'open source lui-même:
  - **ne garanti pas** des hordes de développeurs actifs;
  - **ne guérit pas** tous les maux d'un projet.
- Ouvrir un projet peut ajouter une nouvelle complexité et coûter cher à court terme (pure surcharge au début):
  - organiser le code pour qu'il soit compréhensible par des inconnus;
  - rédiger une documentation de développement;
  - mettre en place des forums de discussion / outils de collaboration;
  - répondre aux questions des développeurs intéressés, ...

# Qu'attendre du passage à l'open source?

- Une autre erreur commune consiste à minimiser l'importance de la présentation et le packaging du projet:
  - le site web associé au projet doit être agréable;
  - la compilation et l'installation du logiciel doivent être automatisées autant que possible.

Activités essentielles pour éliminer les distractions et barrières cognitives !

⚠ Ce travail est souvent traité comme d'importance secondaire:

1. avantages peu visibles pour ceux qui développent le projet;
2. compétences différentes de celles d'écrire du code.

# Qu'attendre du passage à l'open source?

- Une dernière erreur consiste à penser que peu / pas de gestion de projet est requise OU que les pratiques "internes" peuvent être utilisées
- ✗ La gestion de projet open source est rarement visible mais **toujours présent** dans les projets réussis !
- Que se passerait-il si des développeurs *aléatoires* travaillaient ensemble sans supervision ?

# Qu'attendre du passage à l'open source?

- Une dernière erreur consiste à penser que peu / pas de gestion de projet est requise OU que les pratiques "internes" peuvent être utilisées
- ✗ La gestion de projet open source est rarement visible mais **toujours présent** dans les projets réussis !
- Que se passerait-il si des développeurs *aléatoires* travaillaient ensemble sans supervision ?

Sauf miracle, effondrement du projet ou séparation des programmeurs !

=> Maintenir l'idée que *"un groupe d'individus peu faire plus ensemble qu'individuellement"*

# Qu'attendre du passage à l'open source?

Concrètement:

- l'effort requis pour le passage à l'open source dépend du projet et sera le plus perceptible au début;
- les **bénéfices** apparaîtront au fur et à mesure de l'avancement du projet;
- permet au développeur d'exposer son travail au grand jour et d'être reconnu par ses pairs (un contributeur peu rester actif même après changement de situation personnelle);
- une interface à travers laquelle les développeurs / managers sont exposés à des personnes et des idées nouvelles.

=> Mêmes bénéfices qu'une conférence sans les frais associés.

## I.2 Un peu d'histoire



? A votre avis, comment était considéré le logiciel et le partage de logiciel ?

# Le début du partage de logiciel

- Le partage de logiciels existe depuis aussi longtemps que le logiciel lui-même
- Au début, les fabricants se focalisaient sur l'innovation matérielle
- **Le logiciel n'était pas considéré comme un actif commercial:**
  - les clients initiaux étaient des scientifiques / techniciens capables de modifier et étendre le logiciel;
  - les clients redistribuaient parfois leurs correctifs au fabricants et autres clients.
- Cette approche était tolérée et encouragée car elle rendait le matériel plus attrayant à d'autres clients potentiels.

# Le partage de cette époque diffère de la culture du logiciel libre actuelle

Première différence : **la raison du partage**

- Il était important pour un constructeur que le code et les connaissances spécifiques à une machine soit diffusés **le plus largement possible**
- ⚠ Pas de standardisation du matériel et diversité d'architectures informatiques  
=> un logiciel écrit pour une machine ne fonctionnait pas sur une autre !

# Le partage de cette époque diffère de la culture du logiciel libre actuelle

- Gain d'expertise pour le développeur :
  - **avant** : une architecture ou une famille d'architectures particulière
  - **maintenant** : un langage de programmation ou une famille de langages

=> L'accumulation d'expertise avait pour effet de rendre une architecture plus attrayante pour l'expert et ses collègues

# Le partage de cette époque diffère de la culture du logiciel libre actuelle

Seconde différence : **les restrictions techniques**

- Internet n'existait pas !
- Le partage avec le monde se faisait via:
  - envoie de disques / bandes par courrier terrestre;
  - échange avec le fabricant lui même.

=> Peu de partage à cause des contraintes physiques (souvent proportionnelles à la distance à parcourir) et pas à cause des contraintes légales

# Le démarrage des logiciels propriétaires

Avec le temps:

- les gagnants de la course au matériel ont été **clairement identifiés**  
-> diminution des différences de performances entre ordinateurs
- les langages de programmation "*haut niveau*" émergent  
-> **écriture du programme une seule fois** (exécution possible sur plusieurs architectures)

Le logiciel devient le différenciateur (et non plus la puissance de calcul)

=> Modification de la stratégie commerciale: les logiciels font partie intégrante des ventes de matériel !


# Le démarrage des logiciels propriétaires

Les fabricants appliquent plus **strictement le droit d'auteur** :

- limiter la capacité des utilisateurs à **réimplémenter** et à avoir de **nouvelles fonctionnalités**  
-> nouvelle *valeur ajoutée* par le fabricant !
- éviter le partage du code aux fabricants concurrents

Les fournisseurs de logiciels renforcent leur position en:

- refusant aux utilisateurs d'accéder au code source
- exigeant des accords de non-divulgence interdisant le partage

 Ironique car cela s'est passé quand Internet a commencé à décoller...

# Le démarrage des logiciels libres

Contre-réaction aux restrictions sur l'échange de code : **début du logiciel libre**

1970 : **Richard Matthew Stallman** (RMS) membre du laboratoire d'intelligence artificielle du Massachusetts Institute of Technology

Le laboratoire AI avait une orientation "*hacker*" : les chercheurs devaient partager toutes les améliorations qu'ils apportaient

Autorisation à d'autres université / entreprise de :

- porter / utiliser le code
- accéder au code source
- lire, modifier et « piller » des parties de code



# Le démarrage des logiciels libres

En 1980, effondrement de cette communauté, rattrapée par l'industrie:

- membres du laboratoire recrutés pour travailler sur un O.S. sous licence exclusive
- nouveau équipements avec O.S. propriétaire

=> Résistance de RMS qui démissionne et lance **le projet GNU** et la **Free Software Foundation** (FSF) !

**Objectif** : développer un O.S. et un ensemble de logiciels entièrement libres et ouverts

-> les utilisateurs pourraient pirater et partager leurs modifications

# Le démarrage des logiciels libres

En parallèle, RMS travaille sur une licence de droit d'auteur garantissant la liberté perpétuelle de son code !

? Comment auriez-vous fait pour cela ?

# Le démarrage des logiciels libres

En parallèle, RMS travaille sur une licence de droit d'auteur garantissant la liberté perpétuelle de son code !

Idée : Empêcher quiconque, même l'auteur, de limiter la diffusion d'une œuvre

## Licence GNU (GPL) :

- le code peut être copié et modifié sans restriction
- les copies et les travaux dérivés, i.e. versions modifiées, doivent être distribués sous la même licence que les originaux, sans restrictions supplémentaire !

# Le démarrage des logiciels libres

Cette restriction est ingénieuse et plus forte que de *simplement* mettre le code dans le domaine public !

Voyez-vous pourquoi ?

# Le démarrage des logiciels libres

Cette restriction est ingénieuse et plus forte que de *simplement* mettre le code dans le domaine public !

Voyez-vous pourquoi ?

Toute copie ne peut être incorporée dans un programme propriétaire !

# Le démarrage des logiciels libres

Le projet GNU a attiré beaucoup de programmeurs :

- certains partageant l'idéologie de RMS;
- d'autres voulant simplement voir du code source libre.

=> Publication des premiers remplacement gratuit des composants d'O.S

**Exemple de réussite** : éditeur de texte GNU (**Emacs**) ou le compilateur C (**GCC**)

1990 : GNU avait presque un O.S. libre complet (exception du noyau)

1991 : Linus Torvalds (et d'autres) créèrent le noyau Linux

=> Apparition du premier O.S. totalement libre !

# Le démarrage des logiciels libres

Apparition d'autres *choses* sur la scène du logiciel libre **mais sans l'idéologie exacte du projet GNU de RMS**

La **Berkeley Software Distribution** (BSD) : ré-implémentation de l'O.S. Unix par des chercheurs de l'Université de Californie à Berkeley.

Groupe ne prônant pas le regroupement et le partage mais le pratiquant.

=> Un des premiers projets de développement de logiciel libre non idéologique

Autre exemple : "TeX" (composition de document) de Donald Knuth

- Publication permettant à quiconque de modifier et distribuer le code
- MAIS, la modification ne pouvait être appelée "TeX" que si elle passait un ensemble de tests de compatibilité


# En résumé: des divergences de motivations

Dans les années 1980: beaucoup de logiciels libres sous diverses licences.

→ Diversité reflétant différentes motivations !

Typiquement, certains développeurs :

- utilisent la licence GNU GPL sans être motivé par l'idéologie associée;
- ne considèrent pas les logiciels propriétaires comme un mal;
- sont motivés à débarrasser le monde des logiciels propriétaires !

 Ces divergences de motivations n'ont pas interagie de manières destructrices.



# Les entreprises s'intéressent à l'open source

? A votre avis, pourquoi les entreprises se sont-elles mises à s'intéresser aux logiciels open source ?

# Les entreprises s'intéressent aux logiciels libres

Le monde du logiciel libre attire les entreprises car il :

- peut produire du code de très bonne qualité;  
☁ dans certains cas, de meilleure qualité technique que le propriétaire !  
⚠ la tendance à produire du bon code n'était pas universelle !
- est moins cher à l'acquisition;
- ne court pas le risque que son propriétaire fasse faillite.

# Les entreprises s'intéressent aux logiciels libres

Les sociétés commencent à avoir un rôle plus actif dans l'open source :

- investissement de temps et d'équipement
- financement de programmes libres

## Raisons :

- large utilisation d'un produit libre
- peut rapporter plusieurs fois l'investissement
- paie quelques experts et récolte les travaux de bénévoles et des développeurs payés par d'autres entreprises

# Libre vs Open Source : Un besoin de clarification inévitable

Le succès du logiciel dans le monde des affaires a rendu ce besoin inévitable !

⚠ Ambiguïté entre *gratuit* et libre, liée à l'anglais "*free*"

Les logiciels libres sont à coût nul MAIS tous les logiciels gratuits ne sont pas libre !

→ Il peuvent ne pas être libre de partage / modification à n'importe quelle fin

**Exemple** : Internet Explorer

- navigateur Web accessible gratuitement mais code source inaccessible
- si accédé, interdiction d'être partagé et/ou modifié !

# Libre vs Open Source : Un besoin de clarification inévitable

Divergence entre deux grands mouvement idéologiques:

- **RMS** : la liberté de partager et modifier est le plus important  
-> arrêter de parler de *libre* revient à exclure le plus important
- **Les autres** : le logiciel est le plus important
  - ne voient pas le logiciel propriétaire fondamentalement mauvais
  - certains pensent que l'auteur devrait pouvoir contrôler les conditions de distribution

1998 : *open source* comme alternative à *libre* (variété d'opinions très vaste)

# Libre vs Open Source : Un besoin de clarification inévitable

Naissance de l'**Open Source Initiative** (O.S.I) un programme de commercialisation des logiciels libres:

- change l'image associée au logiciel libre dans le monde de la finance  
-> bon == viable pour les affaires;
- offre un vocabulaire pour parler des logiciels libres avec un plan de développement et une stratégie commerciale;
- clarifie la divergence avec la croisade idéologique du logiciel libre

 L'O.S.I a fondamentalement changée le paysage du logiciel libre

# Libre vs Open Source : Un terrain d'entente

Les deux mouvements ont dû trouver un terrain d'entente car :

- programmeurs appartenant aux deux mouvements
- participants n'appartenant pas clairement à un camp

**Entente** : la contribution l'emporte sur le contributeur !

-> Les développeurs ne doutent pas des motivations des autres (morales, paie par l'employeur, étoffe de CV, ...)

=> On met l'accent sur la production de bon code partagé

# De nos jours ...

Il faut être conscient des divergences entre logiciel libre et open source  
-> Évitez de tenir des propos déplacés envers les participants d'un projet

Le logiciel libre est une culture par choix !

L'utilisation de la force / menace dans cet environnement ne fonctionne pas  
-> Cela conduit simplement les participants vers d'autres projets !



# Petit récap : logiciel libre

Qu'est ce que le logiciel libre ?

# Petit récap : logiciel libre

Qu'est ce que le logiciel libre ?

Définition GNU : Logiciel qui respecte les libertés suivantes :

- la **liberté d'exécuter** le programme comme vous le souhaitez, dans n'importe quel but;
- la **liberté d'étudier** le fonctionnement du programme et de le modifier pour qu'il effectue les choses comme vous le souhaitez;
- la **liberté de redistribuer** des copies;
- la **liberté de distribuer** des copies de vos versions modifiées à d'autres.

# Petit récap : logiciel libre

**Le logiciel libre fait référence à la liberté et pas au prix.**

Il est possible de payer pour obtenir une copie d'un logiciel libre ou il est possible de l'avoir obtenu gratuitement !

Peu importe la manière dont vous avez obtenu le logiciel, vous avez toujours la liberté de le copier et de le modifier, voir même d'en vendre des copies

Exemple de logiciel gratuit mais pas libre : **Google Chrome**

Exemple de projet open source payant : **Red Hat Enterprise Linux**

-> disponible gratuitement mais convertir le code en exécutable demande des connaissances, du temps et des serveurs !

=> entreprises paient une souscription pour pouvoir l'utiliser sans avoir à le reconstruire à partir du code source !

# Petit récap : logiciel libre

Le logiciel libre s'oppose au logiciel propriétaire :

- Logiciel libre : l'utilisateur contrôle le programme et ce qu'il fait pour lui
- Logiciel propriétaire : l'utilisateur ne contrôle pas le programme

Le logiciel libre doit être utilisable, développable et distribuable dans un cadre commercial !

💡 Un des objectifs du logiciel libre est de remplacer les programmes propriétaires comparables

⚠️ Possible si et seulement si les entreprises ont l'autorisation de l'utiliser !

# Petit récap : logiciel open source

Parmi tous les programmes qui sont open source, seule une minuscule fraction n'est pas libre

☁ Certaines licences open source sont trop restrictives pour être libre  
Exemple : *Open Watcom* est non libre car sa licence ne permet pas d'en faire une version modifiée et de l'utiliser en privé

Une idéologie différente du logiciel libre : "*plus pragmatique*" et prône la collaboration efficace en matière de développement logiciel

# Petit récap : logiciel open source

⚠ Le choix d'une licence n'est pas à prendre à la légère !

Exemples de licences de logiciel libre incompatible avec la GPL

- Licence Apache version 1.1 <https://directory.fsf.org/wiki/License:Apache-1.1>  
Exigences incompatibles, e.g. interdiction d'utiliser des noms en rapport avec Apache
- Licence BSD originale <https://directory.fsf.org/wiki/License:BSD-4-Clause>  
Clause publicitaire BSD : mention du copyright dans toute publicité ou document fourni avec le logiciel

# Petit récap : logiciel open source

- Jabber Open Source Licence <https://directory.fsf.org/wiki/License:JOSL-1.0>  
Liste exhaustive des possibilités de redistribution sous certaines licences
- Mozilla Public Licence (MPL) version 1.1  
<https://directory.fsf.org/wiki/License:MPL-1.1>  
Incompatibilité juridique interdisant de lier un module couvert par la GPL et un module couvert par la MPL

## **I.3 Échange oral / débat : contribution au libre / à l'open source**



# Pourquoi contribuer à un logiciel libre / open source ?

# Pourquoi contribuer à un logiciel libre / open source ?

- Améliorer les logiciels que vous utilisez
  - patcher les bugs qui vous bloquent / ajouter des fonctionnalités
- Trouver des mentors et/ou enseigner
  - échanges sur votre manière de faire et celle des autres
  - fait travailler sa capacité à transmettre / s'approprier une connaissance
- Améliorer les compétences existantes
  - lieux propice à la pratique

# Pourquoi contribuer à un logiciel libre / open source ?

- Rencontrer des personnes avec des intérêts similaires
  - communautés accueillantes et chaleureuses
  - amitiés qui perdurent toute la vie
- Visibilité
  - libre / open source == public
    - => exemples gratuits démontrant vos capacités
  - contact direct par des tiers à propos du projet
    - => visibilité & reconnaissance

# Pourquoi contribuer à un logiciel libre / open source ?

- Apprendre des compétences relationnelles (gestion / leadership)
  - résolutions de conflits
  - organisation d'équipes
  - hiérarchisation du travail
- Valorisant à titre personnel
  - sensation d'appartenir à une communauté
  - sensation de contribuer à l'intérêt général

# Qu'est ce que veut dire contribuer au libre / open source ?

# Qu'est ce que veut dire contribuer au libre / open source ?

Idée fausse la plus répandue : contribuer au libre / à l'open source signifie contribuer au code

## Raisons :

- en tant que développeurs, c'est la première (voir unique) manière de contribuer qui vient à l'esprit;
- en général, c'est la partie qui est la moins négligée / ignorée

⚠ Il est important de contribuer sur d'autres parties que le code !  
→ intéressant pour le projet et bon moyen de rencontrer la communauté

# Qu'est ce que veut dire contribuer au libre / open source ?

- Vous aimez le **design** ?
  - retravailler la mise en page;
  - identifier les utilisateurs pour retravailler la navigation / les menus du projet;
  - définir un guide de style pour avoir une conception visuelle cohérente;
  - créer des illustrations / logos / ...

# Qu'est ce que veut dire contribuer au libre / open source ?

- Vous aimez **écrire** ?
  - rédiger et améliorer la documentation du projet;
  - écrire des exemples sur l'utilisation du projet;
  - démarrer un bulletin d'information pour le projet;
  - rédiger des tutoriels;
  - traduire le projet dans d'autres langues;



# Qu'est ce que veut dire contribuer au libre / open source ?

- Vous aimez **organiser** ?
  - relier une issue à son duplicata (si il existe);
  - proposer étiquettes pour les issues;
  - fermer les anciennes questions ouvertes;
  - participer à l'échange autours d'une issue;  
=> faire avancer la discussion
  - modérer le forum / canaux de discussions;

# Qu'est ce que veut dire contribuer au libre / open source ?

- Vous aimez coder ?
  - résoudre une issue ouverte;
  - écrire de nouvelles fonctionnalités;
  - automatiser la configuration du projet;
  - améliorer les tests / les outils utilisés;
  - relire le code d'autres contributeurs;

# Qu'est ce que veut dire contribuer au libre / open source ?

- Vous aimez planifier des événements ?
  - organiser des ateliers / rencontres autour du projet
  - planifier une conférence sur le projet (si elle existe)
- Vous aimez aider les gens ?
  - répondre aux questions sur le projet, e.g. sur StackOverflow, Reddit, ...
  - répondre à des questions ouvertes dans les issues
  - proposer d'être le mentor de quelqu'un

# Comment bien s'intégrer dans un projet ?

# Comment bien s'intégrer dans un projet ?

**Situation analogue** : s'intégrer en soirée quand on ne connaît personne !

- comprendre les émotions & pensées des participants  
=> augmente les chances d'être remarqué et entendu
- identifier le vocabulaire, les normes et le style utilisés

# Comment bien s'intégrer dans un projet ?

Structure organisationnelle classique :

- auteur (author) : personne / organisation ayant créée le projet;
- propriétaire (owner) : droit administratif sur l'organisation ou le dépôt;
- mainteneur (maintainer) : responsable de la vision et de l'organisation du projet;
- contributeur (contributor) : personne ayant contribué au projet;
- utilisateur (user) : utilisateur du projet.

 Potentielle existence d'équipes dédiées à certaines tâches

# Comment bien s'intégrer dans un projet ?

Lire les fichiers de documentation du projet :

- **LICENCE** : fichier définissant la licence du projet
- **README** : fichier de bienvenue
  - ☁ Explique pourquoi le projet est utile et comment faire ses premiers pas
- **CONTRIBUTING** : similaire au README mais orienté développeur
  - ☁ Décrit la manière de contribuer et les contributions nécessaires
- **CODE\_OF\_CONDUCT** : fichier fixant les règles de comportement
  - ☁ Définit un environnement convivial et accueillant, e.g.  
<https://www.contributor-covenant.org/>

☁ **[Option]** CONTRIBUTING et CODE\_OF\_CONDUCT attestent d'une volonté

# Comment bien s'intégrer dans un projet ?

Autres endroits pour *"tâter l'ambiance"*:

- **gestionnaire de bogues** : discussions de problèmes (ou autres) autours du projet;
- **pull/merge requests** : discussions et études des changement en cours;
- **forums ou listes de diffusions** : discussions plus générales autours du projet;
- **canaux de discussions en temps réel** : discussions informelles, collaborations et échanges rapides.



# Comment trouver un projet auquel contribuer ?

# Comment trouver un projet auquel contribuer ?

Ne pas trop réfléchir et éviter de se brider pour sa première contribution !


- Se tourner vers les projets que vous utilisez / comptez utiliser
  - quelque chose pourriez être mieux ? différent ?
  - un lien du README ne fonctionne plus ?
  - regarder la page « contribute » du projet (ex: url\_github + « /contribute »)

# Comment trouver un projet auquel contribuer ?

- <https://github.com/explore/> ou <https://github.com/search/advanced>
- <https://goodfirstissue.dev>
- <https://github.com/despo/issuehub.io> : language + issue label
- <https://up-for-grabs.net/> : projets avec tâches pour nouveaux contributeurs
- <https://ovio.org/how-it-works> : projets/issues selon compétences/intérêts
- <https://www.codetriage.com/> : mail journalier avec une issue différente
- <https://firstcontributions.github.io/> : première contribution en 5 minutes
- <https://codetribute.mozilla.org/> : guide pour premières contributions
- <https://gauger.io/contrib/#/language/javascript>

# Comment trouver un projet auquel contribuer ?

Une fois un projet sélectionné, vérifier qu'il est propice à de nouvelles contributions :

- **Répond-il à la définition d'open source** ? (e.g. il a une licence ?)
- Nombre de contributeurs, d'issues et de Pull/Merge requests (PRs/MRs) ?
- **Quel est le niveau d'activité du projet** ?
  -  Date de dernier commit dans la branche principale ? fréquence de commit ? issues et PR/MR récentes ? temps de réponse aux issues et PR/MR ? discussions actives dans les issues et PRs/MRs ? ancienneté de la dernière issue fermée et PR/MR fusionnée ?

# Comment trouver un projet auquel contribuer ?

- Le projet est-il accueillant ?
  - Réponses utiles dans les issues ?
  - Echanges amicaux dans les questions ? forums ? chats ?
  - MRs / PRs sont revues ?
  - Remerciement pour contributions ?

# Comment soumettre une contribution ?

# Comment soumettre une contribution ?

**Communiquer efficacement** : une des compétences les plus importantes et développée dans l'open source !

- **donner du contexte** pour une compréhension rapide:
  - erreur rencontrée : expliquer ce que vous vouliez faire et comment la reproduire
  - fonctionnalité : expliquer pourquoi elle serait utile
- **être concis** :
  - augmente les chances que quelqu'un vous aide
- **converser publiquement** :
  - la discussion peut en elle-même être une contribution

# Comment soumettre une contribution ?

**Ouvrir une issue** dans les situations suivantes :

- signaler une erreur que vous ne pouvez résoudre;
- discuter d'un sujet / d'une idée de haut niveau;
- proposer une nouvelle fonctionnalité.

Conseils pour communiquer dans les issues :

- **notifiez que vous souhaitez participer** (évite les doublons)
- **demandez confirmation** avant de démarrer une action
  - ⚠ problème ayant déjà été résolu directement ou par effet de bord



# Comment soumettre une contribution ?

💡 Ne pas hésiter à ouvrir une issue, la commenter et la fermer pour informer autours d'un problème pour lequel vous avez trouvé une solution

# Comment soumettre une contribution ?

**Ouvrir une PR / MR** dans les situations suivantes:

- soumettre un correctif mineur (typo / lien mort / erreur évidente)
- travailler sur une contribution déjà demandée / discutée dans une issue

Quand ?

- de préférence **dès le début**: permet d'être observé et d'avoir des retours
  - 💡 Notez *WIP* dans le titre du commit pour signaler d'un travail en cours

# Comment soumettre une contribution ?

Comment ?

- fork le dépôt d'origine-> **clôner votre fork localement**  
⚠ pensez à synchroniser votre fork régulièrement et/ou à configurer la synchronisation de votre dépôt local avec le dépôt d'origine !
- **créer une branche** et démarrer vos modifications
- **tester vos modifications**, i.e. créer les tests associés à vos modifications et passer les tests existant
- **contribuer avec le style du projet** (linting, commentaires, ...)
- soumettre une PR/MR et utiliser des **références vers les issues associées** (e.g. *Closes #42*), la documentation utile, ...

# Que se passe-t-il après l'envoi de votre contribution ?

# Que se passe-t-il après l'envoi de votre contribution ?

## Pas de réponse

- vous avez bien regardé que le projet est encore actif ?
- si vous avez en tête la personne qui devrait revoir votre contribution : @LeLaReviewer
- si toujours pas de réponse ... passez à autre chose !

# Que se passe-t-il après l'envoi de votre contribution ?

## Demande de modifications

- exemple : le périmètre de l'idée, changements nécessaires dans le code, ...
- prendre le temps de répondre !
- si vous ne pouvez pas faire les modifs (connaissances/temps) prévenez la personne demandant les modifications

# Que se passe-t-il après l'envoi de votre contribution ?

## Rejet

- demande de clarification sur la raison / des retours du pourquoi
- fork possible si c'est souhaité !

## Acceptation !

👉 Savourez votre contribution à l'open source