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)

- Ce chapitre va se focaliser sur les bonnes attitudes à avoir pour gérer l'information d'un projet:
  - **techniques** : nécessaires car les logiciels de gestion de l'information nécessitent d'être configurés, maintenus et ajustés en continu !
  - **relationnelles** : nécessaires pour maintenir la communauté, e.g. encourager les contributeurs à faire ce qu'il faut pour que les infos soient bien organisées

Note : les réponses aux questions sur l'infrastructure sont très subjectives et impliquent un équilibrage :

- commodité entre producteur – consommateur d'informations
- temps requis à la configuration d'un outil VS bénéfices apportés au projet

Avez-vous des exemples d’outils / pratiques pour l’infrastructure technique d’un projet ?

Avez-vous des exemples d’outils / pratiques pour l’infrastructure technique d’un projet ?

- l’utilisation d’un wiki dans un projet :
  - fait vivre le projet (vue d'ensemble et avancement en temps réel)
  - partage les connaissances (limite le travaille en silo)
  - améliore la coordination (élimine la perte d'information, accès aux dernières données à jour)
  - accessible et facile d'utilisation (pas réservé aux spécialistes web)
- la pratique des pull / merge request comme moyen pour packager les proposition de contributions

Liste d’outils standards à mettre en place pour gérer l'information:

- **Site web** : centralisation des informations de manière unidirectionnelle, i.e. du projet au publique
- **Liste de diffusion / forum de discussions** : outil du communication très actif + "moyen d'enregistrement"
- **Gestionnaire de modifications** : gestion du code + visibilité des modifications de codes, e.g. Git, SVN, Mercurial, ...
- **Traqueur de bogues** :
  - gestion de l'état des bogues + reproducteur;
  - gestion de tâches, version, nouvelles fonctionnalités, ...
- **Autres** : salons de discussions en temps réel, wiki , Q&A forum, réseau sociaux, ...

III – Infrastructure technique

III.1 – Site web

III.2 - Liste de diffusion & Forum

III.3 – Gestionnaire de versions

III.4 - Gestionnaire de bogues

III.5 - Autres

III.1 – Site web

Site web

À votre avis, qu'est ce qu'un bon site web :

- Du point de vue du projet ?
- Du point de vue d'un visiteur ?

Site web

À votre avis, qu'est ce qu'un bon site web :

- Du point de vue du projet ?
  Des pages dédiées à l'aide aux personnes souhaitant participer au projet
- Du point de vue d'un visiteur ?
  Des pages répondant aux attentes du lecteur

⚠️ Il est important d'avoir des pages spécifiquement destinées aux utilisateurs et aux contributeurs

💭 Ne pas essayer d'avoir un site deux en un (mauvaise page web pour tous)
💭 Favoriser les liens-croisés ! Passage facile d'un type d'audience à l'autre et premier point d'entrée pour les deux !

Exemple de bon site web

Exemple de bon site web : https://www.libreoffice.org/

- Page principale orientée pour les utilisateurs
  → liens vers la page de téléchargement
- Contributeur : recherche d'un lien avec un mot clef "Développeurs" / "S'impliquer" / ...
  - lien pour tout type de contributeur ("Improve it / Join us")
    → développeur, documentariste, testeur, marketeur, donneur de fonds, concepteur d'interface, support ...
  - liens spécifiques à chaque contributeur ("Improve it / ...")
  - lien pour identifier comment aider ("Improve it / What can you do for Libre Office")

Site web – info en vrac

Est-il nécessaire d'avoir un site web au début d'un projet ? Si oui, doit-on avoir plusieurs passerelles pour les contributeurs, i.e. une développeur et une plus générale ?

Site web – info en vrac

Est-il nécessaire d'avoir un site web au début d'un projet ? Si oui, doit-on avoir plusieurs passerelles pour les contributeurs, i.e. une développeur et une plus générale ?

💭 Pas nécessaire au début mais c'est bien d'en avoir un (si projet mature, un lien vers un site d'hébergement de projet est probablement insuffisant)
💭 Les passerelles ne sont probablement pas nécessaire au début (utilisez votre jugement pour déterminer si la sous-division est appropriée)

Point de vue technique : le problème d'hébergement est facile à résoudre

Principal fonction du site Web : présenter une vue d'ensemble claire et accueillante du projet + relier les outils de collaboration entre eux.

Site d'hébergement

- Service en ligne mettant à disposition des outils de collaboration nécessaire pour faire tourner un logiciel libre / open source !
- Qu’est ce qu’on retrouve dans un site d’hébergement et qui est utile à un projet?

Site d'hébergement

- Service en ligne mettant à disposition des outils de collaboration nécessaire pour faire tourner un logiciel libre / open source !
- Qu’est ce qu’on retrouve dans un site d’hébergement et qui est utile à un projet?
  - gestionnaire de contrôle de version
  - traqueur de bogues
  - un espace wiki
  - hébergement de listes de diffusion
  - services d'intégration continue ...

Solution orientée développeur, pratique pour beaucoup de projet

Avantages d'un site d'hébergement

Quels sont les avantages (non présentés précédemment) d’un site d’hébergement ?

Avantages d'un site d'hébergement

Quels sont les avantages (non présentés précédemment) d'un site d'hébergement ?

- Capacité du serveur et bande passante !
  💭 Peu de chances de manquer d'espace disque ou de saturer la connexion réseau
- Simplicité d'utilisation : requiert une *simple* inscription et met à disposition:
  - des outils de collaborations déjà sélectionnés et configurés;
  - un système d'authentification;
  - un système de sauvegarde des données.

Avantages d'un site d'hébergement

- Intégration d'une myriade de services qu'il serait difficile de reproduire:
  - les redirections associées aux hash des commits / branches;
  - les mails automatique lors de la notification dans une discussion;
  - la possibilité de répondre par mail;
  - la possibilité de relire et modifier le code depuis le navigateur (intègre de puissant outil d'autocomplétion, ...)

=> Beaucoup de solution pratiques et éprouvées sont proposée par les site d'hébergement !

Inconvénients d'un site d'hébergement

Quels sont les inconvénients d’un site d’hébergement ?

Inconvénients d'un site d'hébergement

Quels sont les inconvénients d’un site d’hébergement ?

- Pas de choix sur les outils et leurs configurations
- Pas de contrôle à grain fin sur les paramètres, e.g. la gestion des droits à un projet, son expiration, …
- Possible difficultés pour réaliser des actions sous *anonymat*, e.g. remplir un rapport de bogue ou réaliser des actions en lectures seules pourraient ne pas nécessiter de s'enregistrer

Réflexion sur les sites d'hébergement

Beaucoup de choix possible de nos jours : Gitlab, Github, Bitbucket, SourceForge, Phabricator, SourceHut, ...

⚠️ Le choix doit être réalisé selon les choix de fonctionnalités / outils associés au projet. Par exemple, utiliser Mercurial comme système de contrôle de modification n'est pas compatible avec Gitlab / Github.

💭 Si vous n'avez pas de contraitnes, choisissez Github, Gitlab ou Bitbucket. Ces solutions sont très populaires depuis plusieurs années & beaucoup de développeurs sont déjà familiarisés avec ses solutions d'hébergement.

⚠️ Il est important de se laisser une porte de sortie pour toujours pouvoir passer sur un site différent plus tard !

Réflexion sur les sites d'hébergement

💭 Si possible, faites en sorte que l'adresse principale du projet ne change pas !

⚠️ Les sites d'hébergement peuvent utiliser une stack open source ET du code propriétaire !
=> Partir d'un hébergeur ne permet pas de partir avec toute son infrastructure

Site listant les différences entre site d’hébergement :
https://en.wikipedia.org/wiki/Comparison_of_open_source_software_hosting_facilities

III.2 – Liste de diffusion & Forum

Liste de diffusion / Forum de discussion

L'utilisation d'une liste de diffusion / d'un forum de discussion est-il nécessaire à tout les projets ?

Liste de diffusion / Forum de discussion

L'utilisation d'une liste de diffusion / d'un forum de discussion est-il nécessaire à tout les projets ?

💭 Non, cela va **dépendre de la taille / complexité du projet**:

- Petite / moyen / simple : pas forcément nécessaire, possible utilisation du gestionnaire de bogues comme substitue
- Large / comple : Nécessaire pour les discussion qui ne sont pas spécifiques à un bogue (laisser le gestionnaire de bogues focaliser sur les bogues)

💭 Les moyens de discussion modernes permettent l'accessibilité croisée des forums et des listes de diffusions (et vice versa) ! Exemple de service l'ayant établi Google Groups et Discourse

Liste de diffusion / Forum de discussion

Quelles fonctionnalités retrouve-t-on dans une liste de diffusion ou un forum moderne ?

Liste de diffusion / Forum de discussion

Quelles fonctionnalités retrouve-t-on dans une liste de diffusion ou un forum moderne ?

- discussions liées entre elles via fils de discussions;
- possibilité de souscrire à un fils de discussions;
- recherche d'archive d'ancien messages;
- interaction possible via courrier / navigateur internet.

⚠️ Il est important d'exposer une description évidente des forums publics !
Objectif : guider les nouveaux contributeurs vers le bon forum

Liste de diffusion / Forum de discussion

Exemple de description :

- message d’accueil indiquant :
  - pas besoin de souscrire pour poster
  - validation par un modérateur pour le premier post
- message indiquant que le projet possède les listes suivantes (abonnement):
  - utilisateur@projet.org (discussion du projet, de l'API, d'améliorations, ...)
  - developpeur@projet.org (discussion sur les développements du projet)
  - annonces@projet.org (nouvelles distributions, éléments d'intérêt, ...)

💭 Pour chaque description, pensez à proposer des lien vers les archives d'un fils de discussion et un lien d'abonnement !

Comment choisir le bon logiciel de management de forum?

Comment choisir le bon logiciel de management de forum?

Exemple de fonctionnalités des logiciels de management de forums modernes :

- **Accès via courrier & navigateur internet** : abonnement à un forum par courrier et le lecture via navigateur
- **Capacité de modération** : vérification des messages pour éviter les courriers indésirables, retirer les adresses obsolètes (e.g. changement d'entreprise)
- **Archivage** : messages stockés et accessible via le web
- **Manipulation d'entête** : permettre la configuration de règles de filtrage / réponse à des courriers
- ...

Comment choisir le bon logiciel de management de forum?

Prévention des courriers indésirables :

- éviter les courriers indésirables dans la liste de diffusion
- éviter que les adresses de la liste de diffusion soit utilisées par des *spamers*

Connaissez-vous des techniques pour éviter les courriers indésirables ?

Comment choisir le bon logiciel de management de forum?

Prévention des courriers indésirables :

- éviter les courriers indésirables dans la liste de diffusion
- éviter que les adresses de la liste de diffusion soit utilisées par des *spamers*

Connaissez-vous des techniques pour éviter les courriers indésirables ?

- Liens du forum en mode *"no follow"* : robot de moteurs de recherche ne vont pas "suivre" le lien
- Utiliser des plugins anti-spam
- Désactiver l'édition de commentaires après quelques minutes

Comment choisir le bon logiciel de management de forum?

- Mettre à disposition à la communauté des boutons "signaler", ...
- Enregistrement incluant CAPTCHA
- Demander une confirmation courrier

💭 Les points deux derniers points contraignent à être enregistrer pour soumettre des messages

Approche alternative : autoriser automatiquement les publications des abonnés de la liste (faible coût additionnel d'administration et permet de gérer via modérateur les publications anonymes)

Comment choisir le bon logiciel de management de forum?

Comment faire pour éviter l'utilisation d'une liste de diffusion par des *spamers* ?

Comment choisir le bon logiciel de management de forum?

Comment faire pour éviter l'utilisation d'une liste de diffusion par des *spamers* ?

Une approche *simple* consiste à obscurcir les adresses courriers :

- monadresse@gmail.com → monadresse_AT_gmail.com
- l'idée associée est que l'encodage est évident pour un humain et inutilisable pour les *spamers*

⚠️ Technique n'empêchant pas le spam de la liste de diffusion elle même, juste de ses membres !

💭 Controversée car nécessite le dés-encodage des adresses (pour l'humain) et n'empêche pas, en théorie, le dés-encodage à cause des motifs récurrents

Utilisation et gestion des en-têtes

Connaissez-vous des exemples d’en-têtes utilisable ? Qu’elle utilisation peut-on en avoir ?

Utilisation et gestion des en-têtes

Connaissez-vous des exemples d'en-têtes utilisable ? Qu'elle utilisation peut-on en avoir ?

Exemples d'en-têtes utilisables : "De: ...", "A: ...", "Sujet: ...", "Date: ...", ...

Exemple de **classement automatique** à l'aide d'en-têtes :

- placement d'un courrier dans un répertoire spécifique à un projet / sous-projet
- association d'une étiquette au courrier : "travail", "prioritaire", "extra", ...
- identifier les courriers à lire et / ou répondre !

Idée pour simplifier le filtrage : ajouter un préfixe prédéfini à l'en-tête "Sujet"

Archivage des discussions

L’archivage des données est-il nécessaire ? Si oui, qu’est ce qui doit être pris en compte lors du choix du moyen utilisé pour archiver ?

Archivage des discussions

L'archivage des données est-il nécessaire ? Si oui, qu'est ce qui doit être pris en compte lors du choix du moyen utilisé pour archiver ?

Chaque forum de discussion doit être **entièrement archivé** ! Cela permet de référencer d'ancienne discussion ET fournit un historique et un contexte pour les nouveaux contributeurs !

Pour choisir un moyen d'archiver, prenez en compte :

- mise à jour rapide : permet de référencer des messages archivés et récent
- stabilité : le lien vers un message archivé doit toujours être accessible
- support : possibilité de passer du message au fils de discussion
- recherche : recherches par clefs en spécifiant l'en-tête associée

III.3 – Gestionnaire de versions

Qu'est ce qu'un gestionnaire de versions ?

Qu'est ce qu'un gestionnaire de versions ?

Mélange de technologies et pratiques pour suivre et contrôler les modifications au code source, à la documentation, pages web, ...

Intérêts du gestionnaires de versions: communications entre développeurs; stabilité du code; gestion des versions; gestions des bogues; développement expérimentale; attribution et autorisation des modifications

Cœur du contrôle de version : identifier les modifications, les annoter avec des métadonnées et permettre de rejouer ces modifications si c'est souhaité

⚠️ Un prérequis pour qu'un projet soit pris au sérieux et **un mécanisme de communication où le changement est l'unité de base de l'information**

Connaissez-vous des termes consacrés aux outils de gestion de version ?

Vocabulaire d'outil de gestion de version

Commit :

Vocabulaire d'outil de gestion de version

Commit : apporter une modification au projet

- *plus formellement* : enregistrement d'un changement dans la base de donnée de l'outil de gestion des versions
- plusieurs contexte d'utilisation :
  - comme un verbe : action d'enregistrer une modification
  - comme un nom : référence à une modification
    → exemple : « je viens de corriger rapporté par Hubert, peux-tu vérifier le commit et t'assurer qu'il est bien correcte ? »

Vocabulaire d'outil de gestion de version

Push :

Vocabulaire d'outil de gestion de version

Push : publier un commit dans un référentiel publique

- Opération de routine
- Commit rendu accessible pour que d'autres l'incorpore dans leur copie du code
- Si le référentiel n'est pas cité, c'est généralement le référentiel principale (*maître*) du projet

💭 Selon l'outil utilisé, les commits sont automatiquement poussés vers le référentiel central (Subversion) ou alors sont poussés quand le développeur le décide (Git et Mercurial)

Vocabulaire d'outil de gestion de version

Pull :

Vocabulaire d'outil de gestion de version

Pull : récupérer les modifications réalisées par d'autres développeurs dans votre copie du code

- Opération de routine : généralement, les développeurs mettent à jour leur code plusieurs fois par jour
- Permet de s'assurer que l'on utilise sensiblement le même code que les autres

💭 Détection d'un bug permet d'être *"certain"* que celui-ci n'a pas été corrigé

⚠️ Si cette action n'est pas faite régulièrement, vous pourriez remonter un bug déjà corrigé depuis quelques jours, semaines, ...

Vocabulaire d'outil de gestion de version

Message de commit / message de log :

Vocabulaire d'outil de gestion de version

Message de commit / message de log : commentaire associé à un commit

- Décrit la nature et le but du commit
- Un des *document* les plus important d'un projet car c'est la passerelle entre l'aspect très technique (modification du code) et le monde visible pour l'utilisateur, e.g. correction de bug, ajout de fonctionnalité, amélioration, refactoring, ...

Vocabulaire d'outil de gestion de version

```
Title: Summary, imperative, start upper case, don't end with a period
# No more than 50 chars. #### 50 chars is here:  #

# Remember blank line between title and body.

# Body: Explain *what* and *why* (not *how*). Include task ID (Jira issue).
# Wrap at 72 chars. ################################ which is here:  #

# At the end: Include Co-authored-by for all contributors.
# Include at least one empty line before it. Format:
# Co-authored-by: name <user@users.noreply.github.com>
#
# How to Write a Git Commit Message:
# https://chris.beams.io/posts/git-commit/
#
# 1. Separate subject from body with a blank line
# 2. Limit the subject line to 50 characters
# 3. Capitalize the subject line
# 4. Do not end the subject line with a period
# 5. Use the imperative mood in the subject line
# 6. Wrap the body at 72 characters
# 7. Use the body to explain what and why vs. how
```

Vocabulaire d'outil de gestion de version

Dépôt (repository) :

Vocabulaire d'outil de gestion de version

Dépôt (repository) : base de données où les modifications sont stockées et à partir de laquelle les modifications sont publiées

- Système centralisé : un seul référentiel qui stock toutes les modifications du projet et chaque développeur travaille avec le *dernier résumé* sur sa propre machine
- Système décentralisé : chaque développeur avec son propre référentiel, modifications échangeables de manière arbitraire entre référentiel

Vocabulaire d'outil de gestion de version

Clône :

Checkout :

Vocabulaire d'outil de gestion de version

Clône : obtention de son propre référentiel de développement
💭 En général c'est une copie du référentiel central du projet

Checkout :

- Dans une discution : *checkout* fait en général référence à quelque chose (comme clône)
- Système centralisé : obtention d'une copie de travail (fait intervenir le réseau)
- Système décentralisé : obtention d'une copie de travail à partir d'un référentiel particulier (ne fait pas intervenir le réseau car le référentiel est local)

Vocabulaire d'outil de gestion de version

Copie de travail :

Vocabulaire d'outil de gestion de version

Copie de travail : arborescence privée du développeur contenant le code source

- Contient des métadonnées relatives au contrôle des modifications, e.g. de quel dépôt la copie provient, qu'elle branche elle représente, …
- Dans les système décentralisés, ce terme n'est pas très utilisé.
  💭 On parlera plutôt de *mon clone*, *ma copie*, *mon fork*

Vocabulaire d'outil de gestion de version

Diff :

Vocabulaire d'outil de gestion de version

Diff : représentation textuelle d'un changement

- Montre les lignes modifiées et les modifications associées
- Inclut généralement quelques lignes de contexte d’un côté et de l’autre des modifications

💭 Pour les développeurs familiarisés au code, peux suffire à déterminer l’impact des modifications et les bogues potentiels

Vocabulaire d'outil de gestion de version

Tag :

Vocabulaire d'outil de gestion de version

Tag : étiquette associée à un état particulier du projet, à un moment donné dans le temps

- Utilisé pour exposer des instantannés (snapshots) intéressant du projet
- Exemple : un tag est généralement associé à chaque version publique
  → version_1.2.0, livrable_2_2_1, ...

Vocabulaire d'outil de gestion de version

Branche :

Vocabulaire d'outil de gestion de version

Branche : copie isolée du projet

- Modifications apportées à la branche principale, n'impactant pas les autres branches
- La branche principale est usuellement appelée *master* ou plus récemment *main* (https://github.com/github/renaming)
  💭 Les utilisateurs de SVN parlent de *tronc*
- Facilite les développements expérimentaux
- Possibilité de fusionner les modifications d'une branche à l'autre

Vocabulaire d'outil de gestion de version

Fusion (merge) :

Vocabulaire d'outil de gestion de version

Fusion (merge) : Déplacer des modifications d'une branche à une autre

- Fréquent entre une branche de développement et la branche principale
  💭 Plus rarement entre des branches de développement
- En plus de l'idée *active* d'ajouter les modifications d'une branche à l'autre, *merge* fait aussi référence à ce que fait le gestionnaire de versions sur les modifications non chevauchante
- Modification automatique fusionnées
- Peut conduire à des conflits

Vocabulaire d'outil de gestion de version

Conflict :

Vocabulaire d'outil de gestion de version

Conflict : modifications réalisées par deux personnes au même endroit du code
💭 Des modifications sur une même ligne de fichiers peuvent être marquées comme un conflit sans pour autant toucher aux mêmes variables / mots

- Détecté automatiquement et notification par le gestionnaire de modifications
- Résolution humaine et communication de la solution à l'outil de gestion des modifications

Vocabulaire d'outil de gestion de version

Inverser (revert) :

Vocabulaire d'outil de gestion de version

Inverser (revert) : action d’annulation de modifications validées par le gestionnaire de modifications

- L’événement de *revert* est versionné par le gestionnaire
- En général, action réalisée en passant par le gestionnaire des modifications (pas à la main)

Vocabulaire d'outil de gestion de version

Verrou (lock) :

Vocabulaire d'outil de gestion de version

Verrou (lock) : modification exclusive d'un fichier / répertoire particulier

- Pas systématiquement proposée par tous les outils
- Contraire à l'idéal du développement parallèle et simultané
  💭 Pas nativement présent sur Git mais disponible sur SVN / ClearCase
- Accessible via « git LFS » https://git-lfs.github.com/ (supporté par des hébergeurs tel que Github) et l'utilisation de *git lfs lock*

Vocabulaire d'outil de gestion de version

Log :

Vocabulaire d'outil de gestion de version

Log : liste l’ensemble des commits accessibles

- S'il n'y a pas de commit de référence : l'ensemble des commits sont obtenus à partir du commit courant
- S'il y a plusieurs commit : ensemble des commits accessible depuis chacun des commits

Utilisations de l'outil de gestion de version

A votre avis, que devez-vous versionner dans projet ?

Utilisations de l'outil de gestion de version

A votre avis, que devez-vous versionner dans projet ?

Versionnez TOUT (ou presque) :

- Si c'est écrit, cela mérite d'être versionné
- Le code source, les pages webs, la doc, la FAQ, ... (tout ce qui pourrait être modifié)

💭 Tout versionner dans un seul endroit permet d’avoir un seul mécanisme pour soumettre les modifications (simplicité pédagogique)

Utilisations de l'outil de gestion de version

A votre avis, que ne devez-vous pas versionner ?

Utilisations de l'outil de gestion de version

A votre avis, que ne devez-vous pas versionner ?

- Ce qui ne change pas doit être archivé et pas versionné (exemple: courrier)
- Ce qui est généré car pas vraiment modifiable (mais versionner les modèles)
  Exemple : certains systèmes de build créent un fichier de configuration *configure* généré à partir de *configure.in*, pour changer le premier il faut modifier le second.

Raison : versionner les fichiers générés risque de mener à des commits où l'auteur n'a pas régénéré les fichiers après avoir modifié le template

Utilisations de l'outil de gestion de version : historisation

Naviguer dans l'historique

- Voir les dernières modifications du projet (sur un certain référentiel)
- Remonter dans le temps et voir les modifications antérieures
- Voir les différences entre les modifications
- Lire les messages associés aux commits des modifications

⚠️ Importance de la navigation en ligne : pouvoir réaliser rapidement les actions précédentes (pas d'installation de l'outil)

Utilisations de l'outil de gestion de version : les branches

Permet de convaincre d'une idée après que les développements soient réalisés

- Workflow associé : ouverture d'une branche, développement, MR / PR
- Si le résultant est convaincant : validation et merge des modifications
  💭 Suppression des branches lorsqu'elles ne sont plus nécessaires

Utilisation des branches:

- Evite les goulots d'étranglement
- Permet de développer sans casser le code de tiers et sans suivre un référentiel en mouvement
- Aide d'examiner et commenter les développements

Utilisations de l'outil de gestion de version : singularité de l’information

- Ne pas faire entrer une modification deux fois
- Le commit contenant un modification est son unique identifiant !

Quelles différences avec un changement textuellement identique ?

Utilisations de l'outil de gestion de version : singularité de l’information

- Ne pas faire entrer une modification deux fois
- Le commit contenant un modification est son unique identifiant !

Quelles différences avec un changement textuellement identique ?

- Pas de différence au niveau du code (qui est identique)
- Rendrait difficile le fait de *tenir les comptes* et la gestion des releases
- Facilite la recherche de la source d’un changement (lors de la consultation des notes de commits, s’il n’y a pas de référence à une fusion alors la branche courante est responsable du changement)
- Simplifie l’annulation d’une modification

sebastien.namaneh@protonmail.com

Utilisations de l'outil de gestion de version : singularité de l’information

⚠️ L’effet pratique du conseil précédent diffère d’un outil de gestion à l’autre !

- Dans certains outils les fusions sont des événements particuliers et distincts des commits (possèdent leurs propres métadonnées)
- Dans d’autres, les merges sont enregistrées comme des commits

Comment distinguer une merge request d’un commit *classique* dans ce cas ?

Utilisations de l'outil de gestion de version : singularité de l’information

⚠️ L’effet pratique du conseil précédent diffère d’un outil de gestion à l’autre !

- Dans certains outils les fusions sont des événements particuliers et distincts des commits (possèdent leurs propres métadonnées)
- Dans d’autres, les merges sont enregistrées comme des commits

Comment distinguer une merge request d’un commit *classique* dans ce cas ?

Utilisation des messages de commit : ne pas répéter le message du commit de modification; indiquer que c’est un merge et pointer vers le commit

Utilisations de l'outil de gestion de version : gestion des autorisations

Deux tendances sur la gestion des droits :

- Contrôle fin des droits d'un individu :
  - autorisation de push des commits à certains endroits uniquement
- Contrôle grossier des droits d'un individu :
  - officiellement : push uniquement dans certains endroits
  - officieusement : autorisation de push partout
  - pas de risque réel de problème car si push au mauvais endroit : notification à l'auteur qu'il s'est trompé et annulation simple à réaliser

Utilisations de l'outil de gestion de version : gestion des autorisations

Qu’elle approche serait préférable ?

Utilisations de l'outil de gestion de version : gestion des autorisations

Qu’elle approche serait préférable ?

💭 Dans le cadre d'un logiciel libre / open source , il est intéressant de choisir la deuxième approche car elle :

- Encourage la confiance et le respect : fait passer le message *"nous reconnaissons que vous avez de l’expertise"* contre la première approche *"nous reconnaissons une limite sur votre expertise et doutons de vos intentions"*

Utilisations de l'outil de gestion de version : gestion des autorisations

- Simplifie le temps associé à l'administration des droits : un contributeur qui étend son périmètre de développement progressivement et dont on a pas besoin de mettre à jours les droits
- Extension progressive et auto gérée : un contributeur qui étend son périmètre en commençant par poster des patchs avec demande de revue et qui sera invité à commit directement par la suite (gain supplémentaire au niveau des métadonnées author / commiter)

⚠️ Un commit devrait toujours être revue par ceux qui travaillent sur la partie du code concerné. Autrement, le soucis de l’autorisation d’un individu ne devrait pas être le point le plus inquiétant !

Utilisations de l'outil de gestion de version : Pull / Merge request

- Demande d'un contributeur au projet pour introduire des changements
- Action unidirectionnelle : envoie de changement du contributeur au projet
  💭 Si le projet a évolué, cela peut être ignoré pour simplifier la discussion
- Présentation du changement : différences entre source et modifications
- En pratique : action réalisable via le site d'hébergement
- Autre possibilité : contribution via un outil de révision de code collaboratif tel que Gerrit ou [ReviewBoard] (https://www.reviewboard.org/)

💭 Certains projets préfèrent encore travailler par envoie de patchs par email

Utilisations de l'outil de gestion de version : notifications

💭 Bonne manière de se tenir informé des modifications intéressantes (pour nous). En général, la notification inclut : nom du contributeur, date, fichiers et répertoires concernés, changements réalisés, ...

Forme commune de notification de commit : envoie de courrier à une liste de notification à laquelle un contributeur peut souscrire.

Utilisation de *"hooks"* : tâches automatisées en réaction à certains événements
Par exemple : notification par courriers, affichage dans un channel de discussion en temps réel, rebase automatique, ...
Margebot : https://github.com/smarkets/marge-bot
Dependabot : https://github.com/dependabot (pyaedt #3701)

Utilisations de l'outil de gestion de version : notifications

Exemples :

1. https://gitlab.com/gitlab-org/gitlab/-/issues/244345#note_1462352182
2. https://gitlab.com/gitlab-org/gitlab-runner/-/issues/28121#note_1525037572
3. https://github.com/python-gitlab/python-gitlab/issues/1647

Mail associé à 3) :

Expéditeur : ****
À: python-gitlab/python-gitlab python-gitlab@noreply.github.com
Cc: SébastienM author@noreply.github.com
Répondre à: python-gitlab/python-gitlab reply+AJZUITNOS6ERAL4HUC6W6457PRSHFEVBNHHD255RXY@reply.github.com

The GitLab API does not officially support Kerberos for authentication so this will not be done with the library natively.

However it seems like you can kind of hack it to get a session cookie: https://gitlab.com/gitlab-org/gitlab/-/issues/1469 (4 years old, but might still apply)

You can then provide a custom Session when instantiating Gitlab as described here:
https://python-gitlab.readthedocs.io/en/stable/api-usage.html#advanced-http-configuration

—
You are receiving this because you authored the thread. Reply to this email directly, view it on GitHub, or unsubscribe. Triage notifications on the go with GitHub Mobile for iOS or Android.

III.4 - Gestionnaire de bogues

Le gestionnaire de bogues

Les outils de gestion de version peuvent intégrer des gestionnaires de bogues, e.g. Github, Gitlab, ... **mais** il est possible d'utiliser d'autres gestionnaires.
Par exemple : https://docs.gitlab.com/ee/integration/jira/

⚠️ Le terme *"gestionnaire de bogues"* est trompeur car il ne se limite pas aux bogues. Il est utilisé pour suivre tout ce qui a un début, une fin et des états de transitions entre les deux !

Ticket (issue) : élément de la base de données du gestionnaire

💭 Terme utilisé pour différencier le comportement rencontré par l'utilisateur, de l'ensemble d'informations associées : discussion, résolution, ...

Cycle de vie d'un ticket

1. Ouverture d'un ticket : rempli les informations demandées par le traqueur
   Par exemple: résumé, description, recette de reproduction de bogue, ..
   💭 A l'ouverture, le ticket est en *zone d'attente*, i.e. ouvert mais pas encore intégré dans la *conscience du projet*
2. Lecture du ticket par des tiers
   💭 Réaction à celui-ci : commentaires, demandes d'éclaircissement si nécessaire, ...
3. Reproduction du bogue (si déclaration associée à un bogue)
   Validation par un tiers de l'existence du bogue
   💭 Confirmation à l'émetteur qu'il a contribué au projet en signalant un bogue avéré

Cycle de vie d'un ticket

4. Diagnostique du bogue : identification de la cause et estimation de l'effort nécessaire à la résolution. Il est aussi possible d'attribuer le ticket à quelqu'un et/ou de définir un niveau de priorité à associer au ticket.
   💭 Notez ces informations dans le ticket !
5. Ordonnancer la résolution du ticket (étape optionnelle si la résolution est rapide)
   💭 Définir s'il est bloquant pour certaines version et, potentiellement, décider dans quelle prochaine version le bogue devra être corrigé
   ⚠️ Ne définit pas nécessairement une date butoir

Cycle de vie d'un ticket

6. Correction du bogue : fermeture du ticket
   💭 Les changements sont accessibles même après la fermeture du ticket. Ceci est d'autant plus intéressant que certains ticket sont réouvert quand un bogue se réitère (on parle alors de **régression**) !

Exemples :

https://gitlab.com/gitlab-org/gitlab/-/issues/372545

https://github.com/pylint-dev/pylint/issues/8947

Caractéristiques techniques utiles

Le gestionnaire de bogue étant un **élément centrale** de la vie d'un projet libre / open source, il doit posséder certaines fonctionnalités :

- Gestionnaire connecté au courrier électronique :
  - Une modification apportée à un ticket déclenche un courrier de notification (e.g. commentaire, status, ...)
  - Possibilité de créer de ticket / changer le status d'une ticket
- Formulaire de ticket standardisé : avoir un endroit où enregistrer l'adresse courrier du rapporteur ou des informations de contacts
  💭 Ne devrait pas requérir une adresse mail ou un identifiant (anonymat)

Caractéristiques techniques utiles

- Gestionnaire ayant une API :
  - Moyen de personnaliser le comportement du traqueur, e.g. l'étendre avec des logiciels tiers, ...
  - Possibilité de récupération des tickets (utile pour la migration vers un autre outil)

https://python-gitlab.readthedocs.io/en/stable/gl_objects/issues.html
https://pygithub.readthedocs.io/en/stable/examples/Issue.html

💭 Intérêt de l'abonnement : intégration facile du trafic sur le ticket dans le flux de messagerie quotidien !
=> **Centralisation de l'information**

Points sensibles

Problème qui peut rapidement survenir : **grosse charge** dûe à des **tickets doublons** et/ou des **tickets invalides**

Quelles contre mesures pouvez vous mettre en place ?

Points sensibles

Problème qui peut rapidement survenir : **grosse charge** dûe à des **tickets doublons** et/ou des **tickets invalides**

Quelles contre mesures pouvez vous mettre en place ?

- Avis bien visible détaillant comment savoir si c'est bien un bogue et si ce n'est pas un ticket doublon (i.e. comment chercher)
  Exemple : https://gitlab.com/gitlab-org/gitlab/-/issues/new
- (Humain) surveiller et fermer les tickets pour doublons / invalidité
  💭 Approche utilisée universellement et pouvant s'appuyer sur la redirection automatique des tickets vers des personnes dédiées à certains type de tickets

Points sensibles

- (Humain) Encourager la confirmation du bogue par un tiers
  💭 Prise de contact via forum, IRC, ... pour confirmer l'existence du bogue avant l'ouverture du ticket
  💭 Approche facilitant la détection de ticket doublon

Dans le cas d'une ouverte sans confirmation par un tiers il est important de :

- Répondre avec retenue, remercier et encourager le rapporteur à faire confirmer son bogue
- Valider le ticket s'il est clairement valide et n'est pas un doublon
- Sinon, fermer le ticket et demander au rapporteur de le rouvrir après confirmation

III.5 - Autres

IRC / Salon de discussion en temps réel

Intérêt : échanges entre utilisateurs et développeurs en temps réel !

IRC (Internet Relay Chat) : interface basée sur du texte
💭 Existe depuis longtemps et est un élément clef dans la communication de nombreux projets open source

Systèmes de discussion en temps réel plus récents (basés sur le web) :
– Mattermost (open source) https://mattermost.com/
– Element (open source) https://element.io/
– Microsoft Teams (propriétaire) https://www.microsoft.com/fr-fr/microsoft-teams/
– Slack (propriétaire) https://slack.com/intl/fr-fr/
– Discord (propriétaire) https://discord.com/
– Talkspirit (propriétaire) <-- Français ! https://www.talkspirit.com/
– ....

Wiki

Barrière la plus basse possible pour contribuer à un projet (click & edit)

⚠️ Besoin d'un effort centralisé pour être maintenu :

- organisation et éditions cohérentes;
- assurer la clarté d'une page / section vis-à-vis de l'audience cible;
- mise en page simple et agréable (facilite l'intégration d'éditions);
- documenter les normes d'édition du wiki !
  Wikipedia: documentation sur l'écrire de nouvelles notes, maintenir un certain point de vue, quel type d'édition réaliser / éviter, un processus de résolution de conflit d'édition, ...

Choix du wiki :

- si utilisation d'un site d'hébergement : utiliser le wiki associé !
- sinon: choisir en fonction de vos critères (e.g. ACL, ...) : https://www.wikimatrix.org/ ou https://en.wikipedia.org/wiki/Comparison_of_wiki_software

Forum de questions - réponses

Composante attendue et destinée aux utilisateurs

Similaire à une FAQ (Frequently Asked Questions) avec des mise à jours en temps réel
Exemples de forum de questions-réponses connus : **StackOverflow** / **Reddit**

Quand pensez-vous qu'il est bien de le démarrer ?

Forum de questions - réponses

Composante attendue et destinée aux utilisateurs

Similaire à une FAQ (Frequently Asked Questions) avec des mise à jours en temps réel
Exemples de forum de questions-réponses connus : **StackOverflow** / **Reddit**

Quand pensez-vous qu'il est bien de le démarrer ?

💭 **Pas de moment précis**, sauf si votre projet apparaît souvent dans StackOverflow / Reddit !

⚠️ Si vous vous retrouvez dans l'état de fait où il existe une étiquette / fils de discussion au nom de votre projet c'est qu'il est même tard pour s'en occuper !

Services de réseaux sociaux

Utilisation de services de microblog pour échanger avec la communauté du projet : blagues courtes; annonces qui peuvent être facilement partagées et / ou répondues.

Exemple de site :

- Mastodon : https://mastodon.social/about
- Twitter : https://twitter.com/

⚠️ Evitez d'utilisation d'autres types de services (médias sociaux grand publique comme Facebook / Instagram)
💭 Investissement trop important (en temps et attention) pour les retombées associées

Infrastructure de traduction

L'activité autours de la traduction inclus : la documentation, l'interface utilisateur d'exécution, messages d'erreur, ...

💭 Pas forcément besoin d'avoir une plateforme séparée du dépôt du projet mais cela peut être intéressant car :

- les traducteurs ne sont pas tous des développeurs et ne sont pas toujours familiés des logiciels de gestion de version / site d'hébergement de projet
- le processus de traduction peut être réalisé plus efficacement avec des outils dédiés

Exemple de plateformes de traductions en ligne : http://zanata.org/ et https://translatewiki.net/

Git attributes

L'utilisation de **différents OS** (Windows, Linux, MacOs) ou **éditeurs de code** (Visual Studio Code, Sublime Text, CLion) peuvent mener à des conflits.

Exemple du saut de ligne (EOL):

- sur une machine Windows : par défaut Carriage Return Line Feed (CRLF)
- sur une machine Linux/MacOS : Line Feed (LF)

Problématique : outil de formattage avec une propriété de fin de ligne définie

Solution : gitattributes pour utiliser Git et modifier le comportement par défaut de certaines opérations sur les fichiers et répertoires.
Exemple : https://github.com/ansys/pyaedt/blob/main/.gitattributes
Documentation : https://www.git-scm.com/docs/gitattributes

Ignorer des révisions Git

Contexte : utilisation de `git blame` pour récupérer des informations sur le dernier changement ayant impacté une ligne de code.

Problème : L'utilisation de formatteur automatique...

Exemple en direct avec `git blame --ignore-revs-file .git-blame-ignore-vers` (twinbuilder L129) en CLI.
Intégré dans Visual Studio Code avec Gitlens v13.4 et dans Github !

Precommit

Idée générale : **analyser le code pour imposer des conventions avant commit** !

Exemple de solution : https://pre-commit.com/

Applications possibles :

- Convention pour les messages de commits, e.g. https://www.conventionalcommits.org/fr/v1.0.0/
- Formatter du code automatiquement, e.g. https://pypi.org/project/black/
- Valider le code au regard d'une norme, e.g. https://pypi.org/project/flake8/
- Corriger les fautes d'orthographe courantes dans les fichiers texte, e.g. https://github.com/codespell-project/codespell

Exemple : https://github.com/ansys/pyaedt/blob/main/.pre-commit-config.yaml

CI & Codecov

💭 Rappel du cours de TECHNOLOG de Master 1 ... (https://github.com/errata-ai/vale-action, https://github.com/hadolint/hadolint-action)

Exemple d'outil mettant à dispositions des informations sur la couverture du code dans votre flux de travail : **Codecov** (https://about.codecov.io/)

💭 Solution utilisée dans beaucoup de projet libre / open source, permettant :

- d'analyser rapidement le taux de couverture et le risque associé à une PR;
- de bloquer certaines PR n'atteignant pas un seuil;
- de séparez et catégorisez vos rapports de couverture en fonction des tests et des fonctionnalités de votre projet;
- ...

Extra (découverte)

Pour info, je suis tombé sur cela récemment :

https://github.com/figify/gh-metrics

Description rapide : une CLI pour calculer des métriques relatives aux PR / issues d'un projet Github.