Git débutant

Café développeur LIRIS

Dorian Goepp & Françoise Conil

Quelques mots sur Git

- Outil très utile
  - Pour la gestion de version
  - Permet “facilement” le travail collaboratif
  - Partage et gestion de vos fichiers TEXTES vers un serveur distant
- Limites
  - SUPPORTE MAL LES FICHIERS BINAIRES !
  - parfois complexe
- Installation : https://git-scm.com/downloads
- Ce n’est pas le seul outil de cette famille !

Les commandes Git : configuration initiale

Deux paramètres utiles pour identifier l'auteur des commit :

- `git config --global user.name "<your name>"`
- `git config --global user.email "<your@mail.com>"`

Concepts de base

- Le “commit”
- Fonctionnement en trois espaces (+ si affinités)
  - « workspace », espace de travail
  - « index » / « staging area », la zone de transit, en préparation à un commit
  - « local repository », dépôt local

Pour en savoir plus : http://ndpsoftware.com/git-cheatsheet.html

Créer un dépôt (repository)

- Quelques forges :
  - GitHub
  - GitLab public
  - le GitLab du LIRIS : gitlab.liris.cnrs.fr
  - Renater (gforge)
- Un dépôt pour notre projet, sur le Gitlab du CNRS: git-presentation

Commencer à versionner

- Cas 1 : le projet est vierge
  - `git clone https://gitlab.liris.cnrs.fr/dgoepp/nouveau-depot.git`
  - `cd nouveau-depot`
  - *cycle de travail local*
- Cas 2 : nous avons déjà des fichiers pour ce projet
- Cas 3 : reprendre un projet en cours

Commencer à versionner

- Cas 1 : le projet est vierge
- Cas 2 : nous avons déjà des fichiers pour ce projet
  - `cd <dossier existant>`
  - `git init`
  - `git add .`
  - `git commit -m "Import de la présentation"`
  - `git remote add origin https://gitlab.liris.cnrs.fr/dgoepp/presentation-git.git`
  - `git push -u origin master`
- Cas 3 : reprendre un projet en cours

Commencer à versionner

- Cas 1 : le projet est vierge
- Cas 2 : nous avons déjà des fichiers pour ce projet
- Cas 3 : reprendre un projet en cours
  - `git clone https://gitlab.liris.cnrs.fr/behaviors-ai/april_messages.git`
  - *cycle de travail local*

Cycle de travail en local

- Ajouter un fichier
  - `git status` : observer qu'il n'est pas suivi
  - `git add` : ajouter le fichier au suivi
  - `git commit`
- Modifier un fichier
  - `git diff` : voir les modifications
  - `git add`
  - `git commit`
- Supprimer un fichier
  - `git rm` ou
  - `git checkout -- <nom_du_fichier_supprimé>`
- Déplacer un fichier
  - `git mv` ou
  - git rm & git add

Cycle de travail avec le serveur

- Le cycle de base
  a. `git pull origin`
  b. cycle de travail local
  c. *`git pull origin` : vérifier si de nouveaux changements ont été faits côté serveur*
  d. `git push origin master`
- Avec quel serveur distant travaillons-nous ?
  a. `git remote -v`
- Gestion des conflits : modifications concurrentes

Un peu plus loin : Historique

- En ligne de commande
  - `git log`
  - `git log --oneline --decorate --graph`
  - `tig`
- Avec un outil graphique
  - `gitg`
  - `gitk --all`
  - interface web de Gitlab ou Github
  - l’IDE

Un peu plus loin : ne pas versionner certains fichiers

- Fichiers de logs, binaires, mots de passe, etc. n'ont pas lieu d'être versionnés ou partagés
- On procède ainsi
  - touch .gitignore
  - echo "*.out" > .gitignore
  - git add .gitignore
  - git commit
- On peut aussi ignorer des dossiers
  - `echo "log/" >> .gitignore`
  - …
- Limitation : les fichiers déjà existants sont toujours suivis

Un peu plus loin : revenir sur ce qu'on a fait

"Oups ! J'ai fait une bourde, comment annuler ?"

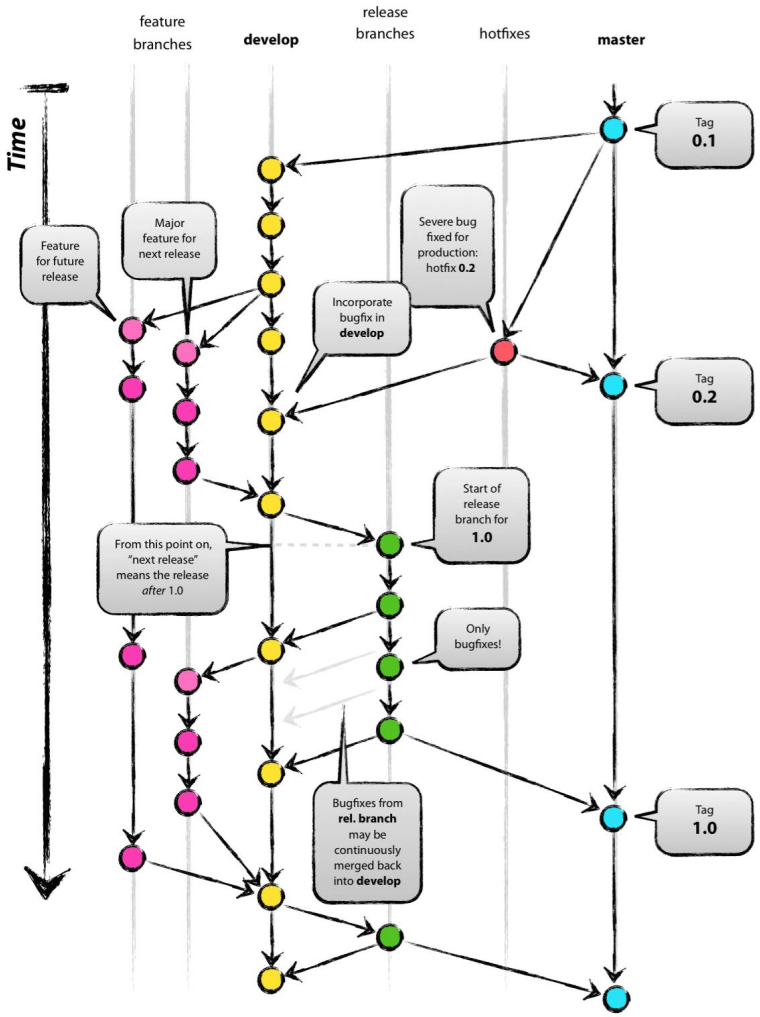
- Les modifications d'un fichier qui n'auraient pas dû être
  - `git checkout -- <fichier>` (fichier entier)
  - `git checkout -p -- <fichier>` (interactif, bouts du fichier)
- Vider l'index
  - `git reset`
- Un commit de trop
  - effacer un commit de l'historique : `git reset HEAD^` (n'annule pas les modifications)
  - `git reset <commit>`
  - (DANGEREUX) pour annuler aussi les changements des fichiers `--hard`
  - `git revert <commit>`

Gitlab du LIRIS

- Documentation : https://liris.cnrs.fr/intranet/documentation/ressources-et-services#gitlab
- Il est possible de créer des comptes pour des extérieurs au LIRIS

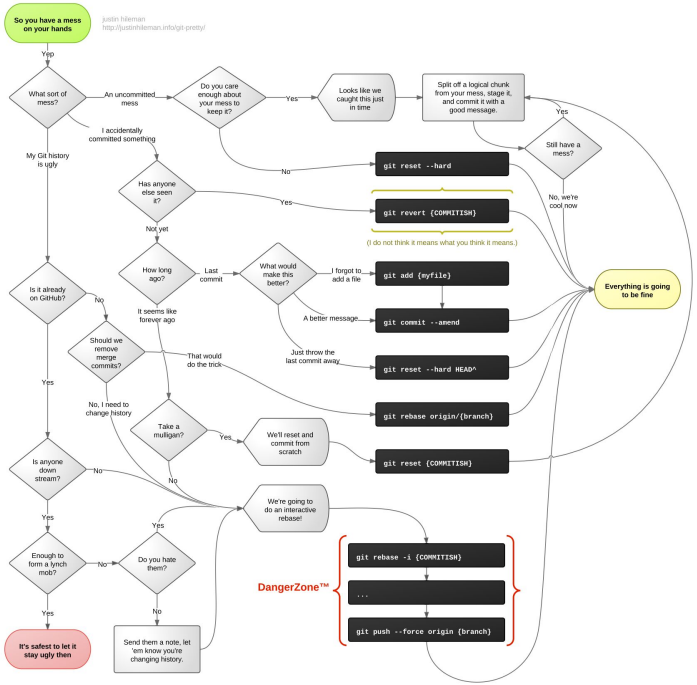
Encore plus loin

- Git permet aussi de
  - gérer des branches : réalités alternatives
  - gérer plusieurs serveurs : remotes
  - avec git-annex, ou Large File System, gérer des gros binaires (git-annex vs git-lfs https://lwn.net/Articles/774125/)
  - déclencheurs avant commit, e.g. tests unitaires, linting, formateurs
- Bonnes pratiques
  - “git workflow”, pour bien gérer ses branches : http://nvie.com/posts/a-successful-git-branching-model/ (à droite)
  - dans un dépôt public : Readme et Licence



Quelques liens

- Se sortir d'une situation difficile : http://justinhileman.info/article/git-pretty/git-pretty.png (**à droite**)
- De SVN à Git (très complet) : http://people.irisa.fr/Anthony.Baire/git/git-for-svn-users-handout.pdf
- Tutoriels d'introduction à Git
  - https://learngitbranching.js.org/
  - https://onlywei.github.io/explain-git-with-d3/
- CheatSheet interactive illustrant Workspace, Index, Local Repository, Upstream et Stash : http://ndpsoftware.com/git-cheatsheet.html
- Doc officielle : https://git-scm.com/book/fr/v2



https://xkcd.com/1597/