Flata-C Git Guidelines

Maxime Gaudin *mai 2011*

Résumé

Afin de faciliter le travail de tout le monde, voici un petit guide sur l'utilisation cordiale et productive du fameux gestionnaire de version de notre amis Linus : Git ! Pour toute questions sur l'utilisation de Git, veuillez consulter [Com11] et [Cha09]. Bon courage !

1 Listes des branches

Le découpage en branches permet une séparation sémantique du matériel et doit être respecté à la lettre afin que chacun puisse s'y retrouver. N'hésitez pas à vous y reporter à la moindre hésitation :

- `master` : Elle contient le code et la documentation de la dernière *release*. C'est elle que l'utilisateur final doit consulter pour trouver tout ce qu'il cherche sur `Flata-C` .
- `minors` : Cette branche contient l'ensemble des versions mineures du programmes. Chacune de ces versions est contenue dans un dossier et *bundlée* avec la documentation qui lui est associée.
- `dev` : C'est la branche de développement. Elle contient la dernière version *compilable* du code source.
- `backup` : C'est aussi une branche de développement. Elle contient la dernière version du code. **C'est la seule branche où Git tolèrera le commit d'une version non compilable.**
- `doc` : Cette branche contient la dernière version de la documentation. Elle est en partie générée automatiquement à partir du code source de la branche *dev*.
- `papers` : C'est la branche qui contient l'ensemble des papiers relatifs au projet. Attention, toutes les sources LaTeX déposées sur cette branche doivent compiler.

2 Best-practices

De par sa nature collaborative, l'utilisation de Git impose à ses utilisateurs de respecter certaines règles. En plus de suivre la nomenclature des branches définie ci-dessus, il est primordiale de :

1. **Commiter souvent** : Plus vous commitez, plus il sera facile de retrouver une situation favorable après une erreur de programmation, et moins vous perdrez de travail.
2. **Commiter clair** : Git impose de d'associer un message à chaque commit. Soyez exhaustif et pensez aux autres qui n'auront pas à relire tout le code pour comprendre vos modifications. Inutile de dire que lors d'une erreur de programmation, il est incroyablement plus simple de retrouver un commit nommé « Portage de la fonction XXX à Frama-C Carbon » plutôt que « djfn ».
3. **L'utlisation des branches est gratuite :** Il ne faut pas hésiter, lorsque l'on s'aventure sur un terrain que l'on sait glissant, à créer une branche temporaire afin de faire des tests, puis, si tout c'est bien passé, à merger par la suite.
4. **N'ayez pas peur de garder le code en local** : Il ne faut pas confondre *sauvegarder* et *versionner*. Un push est donc plus une publication qu'une sauvegarde (qui elle correspond à un commit).

5. **Ne restez jamais sur une branche autre que celle de backup** : L'idée de Git est de passer 99% de son temps sur la branche de développement, et de la merger avec une autre si les conditions s'y prêtent.

6. **Le problème est souvent entre la chaise et la clavier** : Par mesure de précaution, Git rompt souvent le *workflow* avec des messages d'erreur ou de conseil. Sachez qu'il n'y est pour rien et qu'en plus de vous arrêter avant de casser quelque chose, il vous explique gentiment le problème.

Glossaire

COMMIT

Action d'enregistrer dans l'arbre Git local un ou plusieurs fichiers. [Page(s) 1]

MAJEURE

Forme raccourcie de « Numéro de version majeur » [Page(s) 3]

MERGER

Action de fusionner deux branches Git. [Page(s) 1, 2]

MINEURE

Forme raccourcie de « Numéro de version mineur » [Page(s) 3]

PUSH

Action d'enregistrer dans l'arbre Git distant un ou plusieurs commit. [Page(s) 1]

Références

[Cha09] Scott Chacon. *Pro Git*. Apress, Berkely, CA, USA, 1st edition, 2009.

[Com11] Community. The git community book. http://book.git-scm.com/book.pdf, 2011.

Annexe

Versions

L'IDENTIFICATION de la version d'un programme se fait grâce à deux (parfois plus) entiers :

- Le numéro de version majeur
- Le numéro de version mineur

Ils sont liés de la manière suivante : `Flata-C` [MAJEUR].[MINEUR]

Il n'y a pas de consensus pour incrémenter ces numéros mais on peut citer les règles suivantes :

- La publication d'une version ou le portage sur la nouvelle version d'une de ses bibliothèques incrémente la majeure.
- L'ajout d'une FEATURE augmente la mineure.
- La correction de bug ne modifie pas les numéros de version.
- L'optimisation de l'existant ne modifie généralement pas la version sauf si elle apporte une utilisation différente du logiciel.

Table des matières