How To work with GIT

B. Mouginot, N. Thiolliere

17 janvier 2017

Table des matières

1 Introduction

Ce document vise à fournir les bases minimales pour l'utilisation et le développement des resources disponibles sur le serveur git de l'in2p3, relatives aux études systèmes et scénarios. Pour une documentation complète et de très bonne qualité sur GIT, vous pouvez vous rendre à l'adresse https://git-scm.com/book/fr/v2

2 Connexion au gitlab in2p3

La gestion du code CLASS et d'autres outils ou resources est actuellement géré par un dépot GIT situé à l'adresse suivante : https://gitlab.in2p3.fr. Ce site web fournit une interface graphique qui permet de naviguer sur le projet SENS (Systems for ElectroNuclear Scenarios) et ces sous projets associés. Chaque utilisateur a le droit de créer son propre sous projet et d'inviter des participants.

2.1 Se connecter à un dépot GIT en ligne via ligne de commande

La méthode la plus simple pour gérer et utiliser un dépot GIT se fait via console. Placez vous à la location souhaitée pour brancher le serveur GIT. Pour la première utilisation, vous devez cloner le dépot distant avec la commande suivante :

```
1        git clone adresse_du_depot
```

Les adresses des dépots GIT existants sont listés ci dessous :

— Dépot principal de CLASS : git@gitlab.in2p3.fr:sens/CLASS.git
— Dépot pour les documents en partage : git@gitlab.in2p3.fr:sens/Notes_et_Papiers.git
— Dépot pour la configuration zsh : git@gitlab.in2p3.fr:sens/zsh_config.git

Vous pouvez alors naviguer dans l'arborescence du dépot, et charger les branches disponibles.

2.2 Gestion via interface graphique

3 Gestion des branches

3.1 Commandes de bases

Les commandes suivantes permettent de visualiser les branches :

```
git branch
git branch -a
git show-branch
```

La première commande permet de simplement voir les branches existantes sur votre dépot git. La deuxième commande suivante permet de visualiser les branches locale et distante (remote). La troisième écrit également les commits réalisés.

3.2 Création de branch

Une branche permet de travailler sur un programme sans affecter la version courante, supposée fonctionnelle. Usuellement, on peut créer une branche pour tout nouveau développement ou pour toute nouvelle fonctionnalité du code. Pour créer votre branche, relative à vos développements, ou tout simplement pour corriger un simple bug, la commande suivante peut être utilisée :

```
git checkout -b NewBranch
```

L'option -b permet de basculer directement dessus. En deux étapes distinctes, cela revient à faire ceci :

```
git branch NewBranch
git checkout NewBranch
```

La première commande crée un clone de la branche master appelé NewBranch. La deuxième commande switch sur la nouvelle branche.

4 Modification et commit du contenu d'une branches de travail

4.1 Rajouter un nouveau fichier ou dossier au dépot

Travaillez simplement comme vous le faite dans un répertoire standard. Ajoutez ou modifiez les fichiers et répertoires que vous souhaitez. Pour synchroniser les modifications sur votre branche, vous devez ajouter les nouveaux fichiers pour qu'ils soit suivis par GIT :

```
git add nom_des_fichiers
```

Une fois les nouveaux fichiers ẗrackés, vous pouvez faire un commit sur votre branche. Cette action mets votre dépot local à jour mais pas le dépot distant.

```
git commit -m "Commentaire␣decrivant␣votre␣commit"
```

Pour mettre à jour votre version avec le dépot distant, vous devez p̈ousserv̈otre commit :

```
git push
```

5 Merge / fusion d'une branch avec la branche principale / master

Lorsque votre travail sur votre branche est satisfaisant, qu'il a été testé un nombre incalculable de fois et que vous voulez proposer votre modification à la communauté, vous pouvez fusionner votre branche avec la branche master. Pour ce faire, il faut switcher sur la branche master et faire un merge :

```
1    git checkout master
2    git merge YourBranchToMerge
```

6 Divers

6.1 Commandes importantes

De manière générale, vous pouvez toujours faire un status de votre dépot local avec la commande :

```
1    git status
```

Cette commande est très importante car elle fournit les commandes que vous devez probablement effectuer pour mettre votre dépot à jour, en fonction des modifications apportées.

6.2 Du bon usage de la gestion des branches

La bonne habitude de travail consiste à générer sa branche en fonction de la fonctionnalité que vous souhaitez apporter à partir de la branche principale (master). Lorsque vos modifications sont faites, vous pouvez proposer une merge request afin de valider vos modifications. De cette manière, la branche master avance régulièrement en fonction des modifications réalisées. Même si c'est relativement tentant au début, il convient d'éviter de créer une branche personnelle, sauf si c'est une branche de test qui n'a pas vocation à être fusionnée dans la branche master.