TutorialinFrench

From JRLWiki

Les commandes usuelles de git

Note préliminaire: A l'exception des commandes d'inialisation git init et git clone, toutes les commandes de git peuvent être lancées à partir de n'importe quel sous dossier du répertoire principal

Contents

- 1 Installation et configuration de git
  - 1.1 Installation de git
    - 1.1.1 Windows
    - 1.1.2 Unix
  - 1.2 Initialiser sa configuration git
- 2 Initaliser le répertoire git
  - 2.1 Initialiser le suivi de version
  - 2.2 Creer à partir d'un repertoire distant (clone)
- 3 Parametrer git
- 4 Versionner les fichiers
  - 4.1 Ajouter des fichiers a l'arborescence de git
  - 4.2 Supprimer des fichiers de l'arborescence de git
- 5 Afficher des infos sur la version courante
- 6 Gestion des branches
  - 6.1 Creer une branche
  - 6.2 Fusionner des branches
  - 6.3 Merge et conflit
- 7 Gestion des répertoires distants
  - 7.1 Creer le repertoire sur le serveur distant
  - 7.2 Creer une branche sur le repertoire distant
  - 7.3 Effacer une branche distante
- 8 FAQ / Problèmes rencontrés
  - 8.1 Impossible d'utiliser git avec pageant (Win32)
  - 8.2 The server's host key is not cached in the repository
- 9 References externes

Installation et configuration de git

Installation de git

Windows

télécharger git sur http://git-scm.com/
Lors de l'installation, choisir les options
Adjusting your PATH environment
Option 2: "Use Git Bash only"

Choosing the SSH executable
Option 2: "Use OpenSSH"

Choosing CR/LF behavior
Option 1: "Use Unix"

Unix

Initialiser sa configuration git

Obligatoire: Définir son "profil" git (ce qui sera indiqué dans les logs)

```
> git config --global user.name "Nom Prenom"
> git config --global user.email "mon_email"
```

Optionnel: ajouter la coloration pour les git diff, status et branch

```
> git config --global color.diff auto
> git config --global color.status auto
> git config --global color.branch auto
```

Optionnel: choisir l'editeur appele pour ecrire les messages de commit, il faut mettre a jour la variable d'environement EDITOR.

ex : dans le .bashrc

```
# Git
export EDITOR=emacs
```

Initaliser le répertoire git

Initialiser le suivi de version

```
> cd path
```

Creer le dossier

```
> mkdir local_rep && cd local_rep
```

Puis initialiser le versionning

```
> git init
```

Creer à partir d'un repertoire distant (clone)

```
> cd path
```

Puis cloner le fichier (/!\ le dossier n'a pas été créé cette fois-ci)

```
> git clone <remote_repository>
ex :  > git clone git+ssh://login@machine/path/monprojet
```

pour connaitre la liste des clone

```
> git remote
```

pour ajouter un lien vers un clone

```
> git remote add name <url>
```

Parametrer git

Pour ne jamais versionner certains fichiers ou repertoires, specifier leur path dans un fichier **path/monprojet/.gitignore**

```
> less .gitignore
*.dat
*.so
*.a
Makefile
CMakeCache.txt
CMakeFiles
cmake_install.cmake
.cdtproject
.cproject
.project
.settings
include/*
lib/*
```

Versionner les fichiers

Ajouter des fichiers a l'arborescence de git

Se placer dans le repertoire du projet a versionner Puis ajouter les fichiers/repertoires a suivre

```
> git add monFichier/monRepertoire
```

cette commande liste les fichier a mettre a jour

```
> git commit
```

met a jour les fichier precedemment listes par la commande add dans la version local du .git ou

```
> git commit -a
```

met a jour tous les fichiers versionnes qui ont une modification par rapport a la version precedente

```
> git push
```

met a jour tous les repertoires distants

```
> git pull
```

met a jour le repertoire local

Supprimer des fichiers de l'arborescence de git

```
> git reset HEAD [monfichier]
```

Supprime tout la liste ou uniquement [monfichier] de la liste construite avec la commande *add*

```
> git checkout [monfichier]
```

Annule les modifications faites au fichier [monfichier]

```
> git reset --hard
```

Supprime toutes les modifications et revient à la dernière version commitée

```
> git reset commit_number
```

Supprime tous les logs de commit_courant jusqu'à commit_number mais ne

modifie pas les fichiers dans le répertoire local. Si les fichiers ont été sujets à modification entre le commit commit_number et commit_courant, ces modifications seront visibles grâce à un dit diff.

```
> git rm monfichier
> git mv monfichier
```

Realise la commande (sans le git) ET applique la modification dans le .git

Afficher des infos sur la version courante

```
> git log
```

Affiche la liste des commits

```
> git status
```

Affiche l'état du répertoire git (fichiers modifiés, fichiers non versionnés)

```
> git diff
```

Affiche les modifications par rapport au dernier commit

```
> git blame [fichier]
```

Indique l'auteur de chacune des lignes de fichier

```
> gitk    #ou bien
> qgit    #(only available on unix)
```

Lance une interface graphique contenant tous le log détaillé de la branche courante (git log + git diff)

Gestion des branches

Creer une branche

```
> git branch maNouvelleBranche
```

crée la branche

```
> git checkout maNouvelleBranche
```

permet de passer sur la nouvelle branche

```
> git checkout -b maNouvelleBranche
```

Exécute les deux commandes précédentes

```
> git branch [-a]
```

Affiche la liste des branches existantes sur le répertoire local [-a : et sur les répertoires distant]

NB: il est possible de faire un checkout sur une ancien commit de numéro abcdef...

```
> git checkout abcdef
```

Attention : pour que les modifications de la branches maNouvelleBranche soient conservees, penser a commiter

```
> git commit
```

avant

```
> git checkout master
```

Fusionner des branches

Par exemple pour fusionner la branche maNouvelleBranche a la branche master

On se place sur la branche master :

```
> git checkout master
```

On fusionne en utilisant la commande merge

```
> git merge maNouvelleBranche
```

Merge et conflit

Le merge est normalement automatique. Il ne peut pas être réalisé sur le répertoire distant, uniquement sur le répertoire local.

S'il y a conflit pour certains fichiers, il est indiqué par une balise de type "<<<<<<<< v1 ===== v2 >>>>>>>>>" Une fois le conflit résolu, il est nécessaire de faire explicitement

```
> git add fichier_en_conflit
```

(> git commit -a ne marche pas)

Gestion des répertoires distants

Creer le repertoire sur le serveur distant

site de référence (http://toolmantim.com/articles /setting_up_a_new_remote_git_repository)

```
> mkdir rep.git && cd rep.git
> git --bare init --shared=group
```

Initialized empty Git repository in /var/git/myapp.git
--shared=group autorise le groupe a modifier le .git (donc a faire des push)
--shared=all autorise tout le monde

Add the remote repository to your existing local git repo and push:

```
> cd  local_folder/rep.git
> git remote add origin <remote_repository>/rep.git
> git push origin master
```

Creer une branche sur le repertoire distant

```
> git push origin <local branch name>
```

Crée une branche du meme nom que la branche locale dans le repertoire distant

```
> git branch -f <local branch name> origin/<remote branch name>
```

Forcer l'association de ta branche à la branche à distance, autrement dit, tu pourras MAINTENANT faire des pull et push dessus

Effacer une branche distante

```
> git push <remote_repository>  :heads/<remote branch name>
```

Ex: git push origin :heads/name_branch

FAQ / Problèmes rencontrés

Impossible d'utiliser git avec pageant (Win32)

Problème : Dans une console git (Lancée avec "C:/Program Files/Git/bin/sh.exe --login -i"), je n'arrive pas à utiliser une clef ajoutée à pageant : il me redemande le mot de passe. Pourtant, la clé est fonctionnelle : il est possible de se loger avec putty, winscp (...) sans utiliser de mot de passe.

Traitement : Vérifiez que la variable **GIT_SSH** existe bien dans l'environnement

```
> echo $GIT_SSH
c:\WINDOWS\system32\plink.exe
```

Si la variable n'existe pas, il faut l'ajouter aux variables d'environnement, dans un premier temps uniquement dans la console courante, afin de vérifier que cette modification corrige le problème

```
> export GIT_SSH=\c\WINDOWS\system32\plink.exe  #Path vers plink.exe
> ssh login@server #test de la clef
```

Ajoutez la ensuite aux variables d'environnement de Windows (Click-Droit Poste de Travail > Avancé > Variable d'environnement)

Note La variable **GIT_SSH** est normalement initialisée durant l'installation de Git. Cependant, certains installeur qui ne proposent pas l'écran *Choosing the SSH executable* ne le font pas.

The server's host key is not cached in the repository

Problème : Dans une console git, je ne peux pas me connecter au server distant, l'erreur soulevée est: *The server's host key is not cached in the repository*

Traitement : Connectez vous à ce compte avec putty dans un premier temps, il vous permettra d'ajouter cette clé dans le cache.

References externes

http://markelikalderon.com/blog/2008/08/26/pushing-and-tracking-remote-

branches-with-git/
http://toolmantim.com/articles/setting_up_a_new_remote_git_repository

Retrieved from "http://jrlserver.a01.aist.go.jp/mediawiki/index.php/TutorialinFrench"

- This page was last modified 03:10, 4 November 2009.