Git

Table des matières

1 Installation

```
# apt-get install git

Les paquets supplémentaires suivants seront installés :
  git-man liberror-perl rsync

Paquets suggérés :
  git-daemon-run git-daemon-sysvinit git-doc git-el git-email
  git-gui gitk gitweb git-arch git-cvs git-mediawiki git-svn

# apt-cache search cgit
cgit - hyperfast web frontend for git repositories written in C
ticgit - ticketing system built on Git
ticgitweb - web interface to ticgit
```

Naviguer dans le manuel : /usr/share/doc/git-doc/ and look for examples.

```
# apt-get install git-doc
$ man gittutorial
```

1.1 git-config

```
$ git config --global color.diff auto
$ git config --global color.status auto
$ git config --global color.branch auto
$ git config --global user.name "nroche"
$ git config --global user.email nroche@narval.tk
$ git config --global push.default simple
$ cat ~/.gitconfig
[color]
        diff = auto
        status = auto
        branch = auto
[user]
        name = nroche
        email = nroche@prologue.fr
[core]
        editor = vi
[alias]
        ci = commit
        co = checkout
        st = status
        br = branch
```

1.2 Nouveau dépôt central

```
# addgroup git
# addgroup nroche git
```

1.2.1 Créer le dépôt central

J'ai l'impression que le périmètre de Git est le projet et non plus le dépôt (avec des modules). Le dépôt dit “centralisé” ne contient pas les projets mais uniquement les méta données.

```
# mkdir -p /gitroot/MON_PROJET
# cd !$
# git init --bare
# find . -exec chgrp git {} \;
# find . -type d -exec chmod g+w {} \;
# find . -ls
```

1.2.2 Importer un projet

```
$ cd ~/git
$ cp -r MON_PROJET ~/git
$ cd MON_PROJET
$ find . -type d -name "CVS" -exec rm -fr {} \;
$ find . -type f -name '.cvsignore' -exec bash -c 'mv $0 ${0/.cvsignore/.gitignore}' {} \;
$ git init
$ git status
$ git add -A
$ git commit
$ git log
$ git log -p
$ git log --stat
```

```
$ git push ssh://evo/gitroot/MON_PROJET master
$ cd ..
$ rm -fr MON_PROJET
```

1.2.3 Checkout du projet

```
$ mkdir ~/git/
$ cd !$
$ git clone ssh://evo/gitroot/MON_PROJET
$ cd MON_PROJET
$ cat .git/config
...
[remote "origin"]
url = ...
fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
remote = origin
merge = refs/heads/master
```

Git-clone renseigne le dépôt central utilisé dans la configuration.

2 Utilisation

2.0.1 Status

```
$ git status
```

2.0.2 Commit

```
$ git status
Modifications qui ne seront pas validées :
modifié :         FICHIER
aucune modification n'a été ajoutée à la validation

$ git add FICHIER
$ git commit
$ git status
Votre branche est en avance sur 'origin/master' de 1 commit.

$ git push
```

Comme avec svn, un push est irréversible.

2.0.3 Update

```
$ git pull --no-edit

 // ou en 2 temps :
$ git fetch (update sans merge)
$ git merge origin/master
```

2.0.4 Update avec conflits

```
$ LANG=C git pull
Updating 51c7b19..3ef777b
```

```
error: Your local changes to the following files would be overwritten by merge:
README
bin/Uip/Search.pm
bin/uipIndexM.pl
etc/useitPortal/useitPortal.conf
Please commit your changes or stash them before you merge.
Aborting
```

Dans ce cas là, il n'y a pas le diff inclus dans les fichiers indiqués. Pour le générer :

```
$ git stash
$ git pull
$ git stash pop
Fusion automatique de etc/useitPortal/useitPortal.conf
CONFLIT (contenu) : Conflit de fusion dans etc/useitPortal/useitPortal.conf
Fusion automatique de bin/uipIndexM.pl
CONFLIT (contenu) : Conflit de fusion dans bin/uipIndexM.pl
Fusion automatique de bin/Uip/Search.pm
CONFLIT (contenu) : Conflit de fusion dans bin/Uip/Search.pm
Fusion automatique de README
```

2.0.5 Restore

```
$ git checkout b045067dc8923d20f83d3c05e3042053ae2cbd6e page.tex
```

3 Travailler avec les branches

3.1 Branches locales

- Voir les branches :

  ```
  $ git branch
  ```

- Créer une branche :

  ```
  $ git branch test1
  ```

- Changer de branche :

  ```
  $ git co test1
  ```

  Pour changer de branche il faut d'abord commiter les changements, ou utiliser `git stash/git stash apply`.

- Mercher une branche :

  ```
  $ git co master
  $ git merge test1
  ```

- Supprimmer une branche :

  ```
  $ git branch -d test1

  // ou, pour forcer s'il y a des commits
  $ git branch -D test1
  ```

3.2 Branches partagées

- Voir les branches :

```
$ git branch -r
```

- Créer une branche :

```
$ git push origin origin:refs/heads/test2
```

- Supprimer une branche :

```
$ git push origin :heads/test2
```

  La branche ne sera pas supprimée localement sur les clients.

- Checkout de la branche :

```
$ git branch --track test2 origin/test2
```

4 Tags

- Tagger (sur un commit) :

```
$ git log
$ git tag intialRelease 6cf157489508dd8f9dea037f8d399071afaf1f7b
$ git push --tags
$ git tag
```

- Basculer temporairement sur une version taggée :

```
$ git tag
$ git co initialRelease
$ git branch
* (détaché de intialRelease)
  master
$ git co master
```

5 Recherche

- dans les sources actuels :

```
$ git grep REGEX
```

- dans les révisions :

```
$ git log -S MOT
commit 8aa01699090906fec4b30494256ff1806b8a59ba
$ git diff 8aa01699090906fec4b30494256ff1806b8a59ba
```

6 Rendre le dépôt publiv

Git supports this natively. You'll need an HTTP server, of course.

Put your (bare) repository in a folder that can be accessed by your web server. In that directory, run the following commands :

```
$ git --bare update-server-info
$ mv hooks/post-update.sample hooks/post-update
```

The first command provides extra information so the web server knows what to do with the repository. The second command makes sure that the information gets updated any time someone pushes to the repository.

You can find that information here

7 Astuces

7.1 Avec Emacs

A priori ne gère pas les push/pull.

```
# apt-get install git.el
```

Redémarer le client apache.

```
M-x git-status
C-h mode
```

7.2 Connexion à un dépôt cvs

- Checkout

  ```
  $ git config --global cvsimport.r cvs
  $ git config --global cvsimport.d $CVSROOT

  // configure update
  $ cd ~/git
  $ git cvsimport -C netSpeed2 -k netSpeed
  $ cd netSpeed2
  $ git config cvsimport.module netSpeed

  // configure commit
  $ cd ~/cvs
  $ cvs co netSpeed
  $ cd -
  $ git config cvsexportcommit.cvsdir /home/nroche/cvs/netSpeed

  $ cat .git/config
  ```

- Update

  ```
  $ git cvsimport
  ```

- Commit

  ```
  $ git-cvsexportcommit -cup HEAD
  ```

  Remarque : penser à travailler sur une branche car les commit-cvs ne se font que depuis un seul commit-git à la fois.

7.3 CGit

IHM comme viewvc :

- Install

```
# apt-get install cgit
```

- */etc/cgitrc*

```
scan-path=/home/nroche/git
```