programmation git

Git

Git est un logiciel de gestion de versions décentralisé. Il est conçu pour être efficace tant avec les petits projets, que les plus importants.

Git a spécialement été créé pour le développement du noyau linux.

Ce projet a débuté en 2005, Linus Torvalds voulait créer une alternative au logiciel propriétaire BitKeeper. Depuis, Git a beaucoup évolué et est utilisé par de nombreux projets.

Contrairement à des outils comme SVN ou CVS, Git fonctionne de façon décentralisée, c'est-à-dire que le développement ne se fait pas sur un serveur centralisé, mais chaque personne peut développer sur son propre dépôt. Git facilite ensuite la fusion (*merge*) des différents dépôts.

1. Installation

Pour pouvoir utiliser Git, il suffit d'installer le paquet **git (apt://git)**.

Git dispose également de nombreux plugins, les plus utilisés sont dans les dépôts :

- git-svn (apt://git-svn) : Gestion des dépôts SVN ;
- git-cvs (apt://git-cvs) : Gestion des dépôts CVS ;
- git-track (apt://git-track) ;
- etc.

Une fois l'installation réalisée, il faut impérativement définir le paramètre [user]. Pour cela, éditez le fichier caché `.gitconfig` se trouvant dans votre dossier personnel :

```
[user]
   email = p.nom@ubuntu-fr.org
   name = nom_programmeur

[alias]
   ci = commit
   co = checkout
   st = status
   br = branch
```

Les alias quant à eux permettent de raccourcir les commandes, exemple : `git st` au lieu de `git status`

2. Utilisation basique

2.1 Gérer les dépôts

Avant de commencer à utiliser Git, il faut lui demander de créer un nouveau dépôt.

```
mkdir nom_depot
cd nom_depot
git init
```

Dans le répertoire `nom_depot`, vous aurez alors un dossier caché `.git`, c'est dans ce dossier que Git stockera les différentes révisions et informations du projet.

Vous pouvez aussi récupérer un dépôt déjà existant et travailler à partir de celui-ci en faisant

```
git clone git://<dépot>
```

ou

```
git clone https://<dépot>
```

Vous pourrez ensuite (si vous avez les droits suffisants sur le dépôt distant), envoyer vos changements avec `git push`.

2.2 État du dépôt

```
git diff
git diff <commit1> <commit2>
```

Permet de comparer 2 versions. Vous pourrez ainsi voir les changements effectués. Si vous avez des changements pas encore *commités*, la commande `git diff` affichera les modifications effectuées depuis le dernier *commit*.

```
git status
```

Savoir tout ce qui n'a pas encore été validé

```
git log
```

Liste les *commits* effectués dans le dépôt. Vous pourrez ainsi voir les modifications qui ont été faites.

2.3 Gestion des fichiers

```
git add <nom_fichier_ou_dossier>
```

Cette commande indique à Git que le fichier (ou dossier) nommé "nom_fichier_ou_dossier" devra être versionné.

```
git add --all *
```

Permet d'ajouter tout le contenu (fichier et dossier) du dossier.

```
git rm <nom_fichier>
```

Supprime le fichier de votre ordinateur, ainsi que du dépôt Git.

```
git mv <nom_fichier> <nouvelle_destination>
```

Déplace le fichier.

2.4 Gestion des commits

```
git pull
```

Met à jour votre dépôt local (à faire avant de commencer à modifier des fichiers pour être sûr de travailler sur leurs dernières versions et avant tout *commit* pour éviter les éventuels conflits avec des modifications effectuées par d'autres utilisateurs entre temps).

```
git commit <fichier1> <fichier2>
```

Crée un *commit* contenant fichier1 et fichier2. Ces fichiers auront dû être au préalable ajoutés au dépôt avec la commande `git add` .

("Il s'agit de la validation d'une transaction. Après avoir commis la transaction, les informations traitées par cette transaction seront disponibles pour les autres sessions, c'est-à-dire pour toute autre transaction éventuelle."http://fr.wikipedia.org/wiki/Commit (http://fr.wikipedia.org/wiki/Commit))

```
git commit -a
```

Crée un nouveau *commit* contenant tous les changements effectués sur les fichiers (`git add` n'est donc pas nécessaire avant un `commit -a`).

```
git push origin master
```

Envoie le *commit* dans la branche principale "master" du dépôt "origin".

2.5 Commandes d'annulation

Git dispose de commandes permettant d'annuler des changements effectués. **Attention, ces annulations ne sont pas réversibles** !

Commande	**Effet**
`git reset –hard HEAD`	Annule les changements effectués depuis le dernier *commit*.
`git reset –hard HEAD^`	Supprime le dernier *commit*. Cette action peut être répétée autant de fois que vous le désirez.
`git revert commit`	Restaure le dépôt tel qu'il l'était lors du *commit* spécifié.
	Pour que cette commande fonctionne, il faut que toutes les modifications soient *commitées*

(ou annulées avec `git reset`).

3. Utilisation avancée

Généralement, quand on utilise Git, on ne travaille pas seul mais en équipe. Voici les principales commandes qui vous aideront à utiliser Git dans de telles situations. Ici, nous supposerons qu'on travaille avec deux dépôts Git distincts nommés respectivement moi et bob. Le dépôt moi est votre dépôt, bob est celui d'une autre personne, admettons qu'il est situé à l'adresse git://github.com/bob

3.6 Gestion des branches

Git permet une gestion efficace des branches, et des *merges* (fusion de branches). Les branches permettent d'avoir simultanément plusieurs versions de votre programme dans votre dépôt Git. C'est très utile, par exemple pour développer une nouvelle fonctionnalité, tout en gardant la branche principale intacte. Ainsi, vous pouvez toujours faire des changements dans la branche principale (corrections de bugs par exemple), tout en développant en parallèle une nouvelle fonctionnalité.

```
git branch test
```

Crée une nouvelle branche nommée "test".

```
git branch
```

Liste les branches présentes dans le dépôt. Le nom de la branche courante est précédé du caractère "*".

```
git checkout <nom_branche>
```

Change de branche. Après cette commande, vous aurez alors accès aux fichiers présents dans la branche nommée "nom_branche". Avant de changer de branche, il faut impérativement que tous les changements effectués soient *commités* ou annulés, sinon Git refusera de changer de branche.

```
git merge <nom_branche>
```

Fusionne la branche courante avec la branche nommée "nom_branche". Il se peut qu'il y ait des conflits et que Git ne soit pas capable de les résoudre tout seul. Les conflits apparaissent lors de changements divergents au même endroit dans un fichier. Si il y a des conflits, Git laissera un marquage directement dans le fichier, contenant le code de la branche courante, et celui de la branche que vous voulez fusionner. Vous devrez alors corriger le problème manuellement. Une fois corrigé, vous devez *commiter* les changements pour finaliser le *merge*.

Si jamais vous décidez d'abandonner la fusion, alors exécutez la commande :

```
git reset --hard HEAD
```

3.7 Récupération des changements

Imaginons que Bob ait implémenté une nouvelle fonctionnalité. Vous voulez naturellement l'intégrer à votre dépôt.

```
git remote add bob git://github.com/bob
git fetch bob
git merge bob/master
```

La première commande crée un alias qui fait pointer bob vers l'adresse du dépôt. Ça permet d'éviter d'avoir à taper l'adresse complète à chaque fois. La deuxième commande récupère les changements que bob a effectués. La troisième commande fusionne les changements de bob avec votre branche courante. Il existe une commande qui a le même effet que `git fetch` suivi de `git merge` :

```
git pull bob
```

4. Voir aussi

4.8 Interfaces graphiques

- **(en)** Giggle (https://live.gnome.org/giggle) (GTK+)
- **(en)** git-age (https://github.com/krig/git-age), interface graphique pour la commande git blame (http://www.alexgirard.com/git-book/5_trouver_les_probl%25C3%25A8mes_-_git_blame.html) (PyGTK)
- **(en)** git-cola (http://git-cola.github.com/) (PyQt)
- **(en)** GitEye (http://www.collab.net/giteyeapp) (propriétaire ?)
- **(en)** GitForce (https://github.com/gdevic/GitForce) (Mono)
- gitg (apt://gitg) (GTK+)
- git-gui (apt://git-gui) et gitk (apt://gitk), interfaces graphiques (Tcl/Tk) livrées avec Git
- qgit (apt://qgit) (Qt)
- **(en)** qgrit (https://github.com/qgrit/qgrit/wiki), interface graphique pour la commande git rebase -i (http://www.alexgirard.com/git-book/4_recombinaison_interactive.html) (Qt)
- **(en)** RabbitVCS (http://www.rabbitvcs.org/) (PyGTK)
- **(en)** SmartGit (http://www.syntevo.com/smartgithg/) (propriétaire ?)
- **(en)** StupidGit (https://github.com/gyim/stupidgit/wiki) (wxPython)
- **(en)** Team Git (http://www.devslashzero.com/teamgit) (Qt)
- **(en)** Tig (http://jonas.nitro.dk/tig/), interface en mode texte
- **(en)** Ungit (https://github.com/FredrikNoren/ungit) Interopérable (Linux, Mac, Windows) et moderne
- **(en)** GitKraken (https://www.gitkraken.com/) Linux, Windows et Mac

4.9 Autres outils autour de Git

- **(en)** EGit (http://www.eclipse.org/egit/), extension ajoutant le support de Git dans Eclipse
- **(fr)** Gitolite: outil de gestion de dépôts Git avec utilisateurs, droits, etc.
- **(fr)** Gogs (https://gogs.io/): serveur de gestion de dépôts comme Gitolite avec interface http(s) inclus par défaut.
- **(en)** Magit (http://philjackson.github.com/magit/), intégration de Git dans Emacs.
- **(en)** Fugitive (http://www.vim.org/scripts/script.php?script_id=2975), Intégration de Git dans Vim.
- Gource, Visualiser l'activité de votre dépôt dans une animation qui traverse le temps.
- **(en)** thunar-vcs-plugin (http://goodies.xfce.org/projects/thunar-plugins/thunar-vcs-plugin), permet l'accès aux dépôts Git et Subversion via Thunar. Le paquet thunar-vcs-plugin (apt://thunar-vcs-plugin) est disponible pour **Ubuntu versions 12.04 & +**.
- kdesdk-dolphin-plugins (apt://kdesdk-dolphin-plugins) permet l'utilisation de l'overlay des dépôt de versionning (pas uniquement git) dans dolphin. (activable par services dans les paramètres)
- **(en)** Git-flow (https://github.com/nvie/gitflow), gestion avancées des branches dans une logique projet : **(en)**Un modèle de branches concluant pour Git (http://nvie.com/posts/a-successful-git-

branching-model/). Le paquet git-flow (apt://git-flow) est disponible pour **Ubuntu versions 12.04 & + (dépôts "universe")**.
- **(fr)** GITg (http://actual-it.info/2013/gitg/), GIT en mode graphique - Tutoriel d'utilisation de GITg
- **(en)** git-lfs Large File Storage (https://git-lfs.github.com/), gestion des fichiers lourds associés au projet: fichiers audio, vidéo, jeux de données (test…), graphiques… qui sont déposés sur un serveur à part.

4.10 Principaux sites concernant Git

- **(en)** Le site officiel de git maintenu par Scott Chacon (http://git-scm.com/) ;
- **(fr)** le livre en français sur le site officiel (http://git-scm.com/book/fr);
- **(en)** Le site officiel de git maintenu par Petr Baudis (http://git.or.cz/index.html) ;
- **(en)** Manuel utilisateur Git (http://www.kernel.org/pub/software/scm/git/docs/user-manual.html) : Une référence très complète ;
- **(fr)** Le livre communautaire (libre et gratuit;) entièrement dédié à Git (http://www.alexgirard.com/git-book/index.html) ;
- **(fr)** Git immersion en français (http://gitimmersion.fr), un des guides références anglais traduit en Français

4.11 Principaux dépôts Git

- GitHub (http://github.com) : très bon dépôt gratuit. Une bonne gestion du travail en équipe, avec la possibilité de forker un dépôt, pour contribuer au développement sur son propre dépôt, tout en gardant un contact étroit avec le dépôt original.
- Gitorious (http://www.gitorious.org/) : utilisé par de gros projets comme le framework Qt. A noter que le site web gitorious est un logiciel libre et donc installable sur son propre serveur à l'inverse de GitHub qui est propriétaire. Les fonctionnalités communautaires des 2 me semblent assez proches.
- BitBucket (https://bitbucket.org/) : concurrent de Github. Propose de faire des dépôts privés gratuits (jusqu'a **5 utilisateurs** sur le même dépôt. Pas d'autre restriction).
- GitLab (https://github.com/gitlabhq/gitlabhq) : logiciel open-source de dépôt, à installer sur sa machine Manuel d'installation (https://github.com/gitlabhq/gitlabhq/blob/master/doc/install/installation.md).
- FramaGit (https://framagit.org) : dépôt fonctionnant sous GitLab géré par FramaSoft, une alternative aux dépôts propriétaires comme GitHub.

4.12 Autres

- **(fr)** Discussion «Conversion d'un dépôt bzr en un dépôt git» (http://forum.ubuntu-fr.org/viewtopic.php?id=692171) sur le forum ubuntu-fr.
- Création rapide de serveur git avec tutoriel (https://forum.ubuntu-fr.org/viewtopic.php?id=1310241)
- **(fr)** Aide-mémoire interactif (http://blogs.media-tips.com/bernard.opic/2013/12/01/initiez-vous-a-git-avec-laide-memoire-interactif-dandrew-peterson/)
- **(fr)** Aide mémoire GIT au format PDF (http://www.chtiland.fr/page.php?id=informatique:developpement:start#docs)

Contributeurs principaux : geenux.