Tutoriel git

Antoine Busque && Benjamin Brodeur Mathieu
Hiver 2014
Version 1.1

Section 0 : Qu'est-ce que git

Git est un système de contrôle de versions distribué. L'objectif d'un système de contrôle de version est de garder un historique de toutes les modifications apportées aux fichiers d'un projet donné. L'aspect distribué de git signifie seulement que cet historique n'a pas nécessairement besoin d'être stocké sur un serveur central. D'autres exemples de systèmes de contrôle de versions sont CVS, SVN (qui sont tous deux centralisés), ou encore mercurial ou bazaar (distribués).

0.1) Installation

Pour l'installation, il suffit de télécharger la version de git correspondant à votre système d'exploitation sur la page suivante : http://git-scm.com/downloads. Ce tutoriel portera uniquement sur l'utilisation de git en ligne de commande. Nous favorisons cette utilisation, car la ligne de commande est un outil universel dont l'utilisation est identique peu importe le système d'exploitation et cette approche donne un meilleur contrôle et une meilleure compréhension de git.

0.2 Configuration

La première chose à faire après l'installation de git est de configurer le nom et l'adresse courriel puisque ceux-ci seront utilisés pour identifier toutes les modifications apportées aux projets sur lesquels vous travaillerez. Pour ce faire, il suffit d'utiliser la commande `git config`, qui comme son nom le suggère sert à configurer toutes les options de git. On lui passe un paramètre `--global` pour que les changements soient valides pour tous les projets sur lesquels vous utiliserez git. Les commandes sont :

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

Pour éviter d'entrer constamment le nom d'utilisateur et le mot de passe github, il faut créer une clé SSH et l'ajouter au compte utilisateur. Pour créer la clé SSH, il faut utiliser la commande `ssh-keygen`. Les valeurs par défauts sont les bonnes, il suffira d'appuyer sur la touche entrée lorsqu'on vous demandera de fournir des informations. Une fois cela fait, une paire de clés publique et privée aura été créée. Il faudra ajouter la clé publique à votre compte github via cette page : https://github.com/settings/ssh. Il faut faire Add SSH Key, lui donner un titre significatif (e.g. le nom de votre ordinateur), puis copier et coller le contenu du fichier ~/.ssh/id_rsa.pub dans la section Key.

Section 1 : Créer un projet partir de zéro utilisant github

Sur votre compte github cliquez sur le bouton de création de repo à droite du nom d'utilisateur et entrez votre nom de repo. Choisir si vous voulez un repository public ou privé et cochez l'initialisation avec README. L'option gitignore est recommandée, car elle permet d'éviter l'ajout de fichiers indésirables au projet. Il suffit de sélectionner le langage de programmation utilisé dans le menu déroulant. Une fois cela fait il suffit de suivre les étapes de la section suivante 1.0.1.

1.0.1Cloner un projet existant

Utilisez Bash pour se rendre à l'endroit où vous voulez que se trouve votre projet, puis utilisez la commande : `git clone git@github.com:username/nomdurepo.git`. Cette commande crée un nouveau dossier au nom du projet dans lequel se trouvent les fichiers.

1.1 Ajouter des fichiers

Lorsqu'un nouveau fichier est ajouté à un projet, il faut dire à git de suivre les modifications de ce fichier, sans quoi il l'ignorera. Pour ce faire, on utilise la commande `git add chemin/versLe/ficher`. Vous pouvez faire utilisation de l'astérisque pour ajouter plusieurs fichiers simultanément.

Pour ajouter un nouveau sous-répertoire, le principe est le même. Notez toutefois que l'ajout ne fonctionnera que si ce sous-répertoire est non-vide.

1.2 retirer des fichiers

Pour que git arrête de suivre les modifications d'un fichier, on utilise la commande `git rm chemin/versLe/fichier`.

Section 2 : commit

Un commit avec git permet d'enregistrer les modifications faites sur un ou plusieurs fichiers tout en lui associant un message. Le but d'un commit est simplement de regrouper des modifications liés entre elles logiquement afin de facilement pouvoir les retrouver dans l'historique et éventuellement les partager avec les autres contributeurs du projet.

Une fois des modifications effectuées, il faut ajouter les fichiers que l'on veut avoir dans le commit avec la commande `git add`, puis on commit ces fichiers avec la commande `git commit -m "Message Significatif"`.

Afin de facilement voir ce qui a été modifié ou ajouté en vue d'être mis dans un commit, on peut utiliser la commande `git status`, qui retournera la liste des fichiers modifiés et de ceux qui sont prêt à être mis dans un commit.

Désormais, le commit est enregistré localement. On peut maintenant le propager vers le repository central.

2.1 push/pull

Une fois le ou les commit(s) effectué(s), on utilise la commande `git push` pour propager les changements sur github.

Afin de récupérer les changements des autres contributeurs au projet, on utilise la commande `git pull`. Il est possible que des conflits surviennent lors de l'utilisation de cette commande. Pour la résolution de ces conflits voir la section suivante 2.2.

2.2 Conflits

Les conflits surviennent lorsque vous et un autre contributeur avez modifiés la même section d'un même fichier. Lorsqu'un conflit survient git spécifie le ou les fichier(s) en conflits. Il suffit d'ouvrir les fichiers dans un éditeur de texte et de sélectionner laquelle des deux versions vous garderez. Celles-ci sont délimitées par `<<<<<<< HEAD` votre version `=======` leur version `>>>>>>> nomBranche.`

Il est possible de conserver une combinaison des deux versions. Une fois le code en conflit édité, il suffit de sauvegarder le fichier et de faire un commit le contenant, ce qui mettra fin au conflit. Le push devrait se dérouler sans problème.

Section 3 : Branchs

Par défaut les travaux effectués et versionnés avec git sont placés dans une branche “master”. Il est toutefois possible de créer une ou plusieurs branches séparées de celle-ci, permettant ainsi de faire des modifications sans affecter directement la version commune du code situé sur master. Par exemple, si l’on veut tester une nouvelle fonctionnalité sans être certain de vouloir l’inclure définitivement dans le projet, il suffit de créer une branche spécifique à cette fonctionnalité et de faire les changements sur celle-ci. Lorsqu’une branche est créée, l’état du code dans celle-ci est le même que dans la branche initiale (master), mais les modifications subséquentes ne s’appliquent qu’à cette branche. Une fois les modifications faites, il est possible soit de supprimer la branche ou de la fusionner avec la version principale sur master. Voici donc quelques exemples de cas d’utilisation des branches sous git :

Cas #1 : Création d’une branche

Pour créer une nouvelle branche, il suffit d’utiliser la commande `git branch nomDeLaBranche` . Après cette commande vous vous trouverez toujours sur la branche master. Pour travailler sur la branche que vous venez de créer, il suffit d’entrer : `git checkout nomDeLaBranche.` À ce moment, il est possible de travailler comme vous le feriez normalement sans que cela affecte la branche master. N’importe quand, il est possible de revenir sur la branche master avec la commande `git checkout nomDeLaBranche.` Les fichiers qui se trouvent dans le répertoire de votre projet git reflètent toujours la branche sur laquelle vous vous trouvez, mais toutes les versions des autres branches sont conservées. Ainsi, en changeant de branche à n’importe quel moment, l’état de celle-ci sera conservé et restauré lorsque vous y reviendrez. Cette méthode crée seulement des branches locales. Pour envoyer cette branche sur le serveur utilisez la commande : `git push origin nomDeLaBranche.`

Cas #2 Fusion des branches (merge)

Une fois des changements accomplis sur une branche spécifique, il peut être désirable d’ajouter ces modifications à une autre branche en les fusionnant. Pour fusionner des branches il suffit de se déplacer sur la branche dans laquelle vous voulez fusionner (généralement master) avec la commande `git checkout nomDeLaBranche` suivie de `git merge nomDeLaBrancheAInclure` .¸Une fois la fusion faite[1], la branche sur laquelle la fusion a été accomplie contiendra les modifications de la branche fusionnée. Cette dernière restera inchangée.

Cas #3 Suppression de branches

Pour supprimer une branche, il faut d’abord se déplacer sur une autre branche que celle qu’on veut supprimer. À ce moment, il suffit d’utiliser la commande `git branch -d nomDeLaBranche`. Le `-d` permet de supprimer la branche seulement si elle a été fusionnée précédemment. Si ce n’est pas le cas, il est possible d’utiliser `-D` pour forcer la suppression. Ces

[1] Note : Il est possible de rencontrer des conflits au moment de la fusion de branches, ceux-ci sont réglés de la même manière que les conflits de commits.

commandes suppriment uniquement la branche locale. Pour supprimer une branche sur le serveur utiliser la commande `git push origin :nomdelaBranche`.

3.1 Flux de travaux

Un exemple de flux de travaux pourrait inclure les éléments suivants :

- Branche master (release)

- Branche « develop »

- Branches pour chacune des fonctionnalités

Le travail doit être effectué sur les différentes branches relatives aux fonctionnalités développées. Une fois une fonctionnalité implémentée, partiellement ou complètement, il faut la fusionner dans la branche « develop ».

La branche master sert uniquement pour les versions stables du logiciel. On « merge » sur master uniquement lorsqu'on obtient une version stable. Une version sur le master doit être associée à un tag qu'il est possible d'ajouter à un commit avec `git tag -a v1.4 -m 'my version 1.4'`.

Pour plus de détails vous référez à l'article suivant : http://nvie.com/posts/a-successful-git-branching-model/

Annexe :

Petits trucs Bash :

Commande	Effet
Pwd	Print working directory : affiche le chemin du répertoire courant
cd -	Revient au dernier répertoire dans lequel vous vous trouviez
Cd	Retour au répertoire usager
Ctrl + a	Revient en début de ligne
Ctrl + e	Fin de ligne
Ctrl + k	Couper ce qui est après le curseur
Ctrl + y	Coller
Ctrl + r	Recherche interactive dans l'historique des commandes
Ctrl + g	Annule la recherche
!!	Exécute la commande précédente (pratique avec : sudo !!)

Nous contacter :

Benjamin Brodeur Mathieu : benjaminbrodeur@gmail.com

Antoine Busque : antoinebusque@gmail.com