

Git : comment faire ?

Benoît Charroux

1. Création d'un compte et d'un projet sous Gitlab

Les projets contenant du code dont on veut garder plusieurs versions doivent être créés via le site de Gitlab :

<https://gitlab.com/>

Il faut au préalable créer un compte. On peut ensuite créer un projet en lui donnant un simple nom.

2. Installation du client git

Pour qu'un dossier sur votre machine puisse être synchronisé avec un dépôt git sur le serveur, il est nécessaire d'avoir installé un client git sur votre machine.

Avant toute chose il faut vérifier si un client Git n'a pas déjà été installé sur votre machine. Pour cela ouvrir une fenêtre de commande et saisissez la commande git.

Si nécessaire, procédez à l'installation comme suit : plusieurs clients git sont disponibles, dont un qui dispose d'une interface graphique sous Windows facilitant l'utilisation de git. Cependant, pour ne pas dépendre d'une plateforme, c'est le client en ligne de commande (mode bash) qui est utilisé dans ce tutoriel.

3. Config globale

A faire une seule fois après l'installation afin de paramétrer une fois pour toute l'accès au serveur. Gitlab

```
git config --global user.name "chacoux" git
```

```
config --global user.email chacoux@free.fr
```

Si on est derrière un proxy : git config -- global

```
http.proxy http://proxy.efi.fr:3128
```

Ne plus être derrière un proxy :

```
git config --global --unset http.proxy
```

4. Clonage local d'un projet existant sur le serveur

cd d : (optionnellement pour changer de drive, d ici)

mkdir RMC3 cd RMC3

git init

git clone <http://github.com/chacoux/MonProjet.git> Indiquez ici l'adresse d'un projet existant sur Github (le votre par exemple) cd monprojet

git init permet d'initialiser localement un dépôt git. Git init peut être fait dans un dossier vierge ou contenant déjà des fichiers sources ou dossier d'un projet existant. git clone permet de récupérer localement une copie d'un projet existant sur le serveur.

pwd permet de connaître dans quel dossier on est, ls permet de lister le contenu d'un dossier et cd .. de remonter dans l'arborescence des dossiers.

5. Ajout de fichiers

Il ne suffit pas de mettre des nouveaux fichiers dans le dossier local sous git car il faut aussi l'indiquer à git via la commande :

git add ./WebContent/WEB-INF/jsp/jobconf.jsp git add peut aussi se faire sur un dossier contenant des sous-dossiers et des fichiers.

Ne pas oublier de faire un git commit : git

commit -a -m 'un commentaire'

et git push ensuite. git push permet d'envoyer vers le serveur le contenu du dépôt local.

Contrairement à git push, git fetch permet de télécharger le contenu du dépôt distant vers le dépôt local de votre machine. git rm permet de supprimer un fichier, et git mv de le déplacer.

6. Créer un branche

Une branche est une copie du code qu'on peut modifier sans affecter la version originale. On utilise typiquement les branches quand on veut implanter une nouvelle fonctionnalité pour une application sans prendre le risque de corrompre le code existant.

git branch nomDUneNouvelleBranche

Aller sur la nouvelle branche git

checkout nomDUneNouvelleBranche

Vérifier en affichant la liste des branches

`git branch`

La branche en cours d'utilisation s'affiche avec une *

On peut travailler normalement sur une branche (voir « ajout de fichiers »).

Ce qui est remarquable avec les branches c'est que seuls les fichiers propres à une branche s'affichent. Par exemple, le fichier `essai.txt` a été ajouté à git avec un `git add` sur une branche. Ainsi on voit apparaître ce fichier dans le dossier local :

 application\src\ProjetRMC\src\essai.txt	06/09/2012 10:11	Document texte	1 Ko
 essai	27/03/2013 15:52	Document texte	1 Ko
 local\essai.txt	06/09/2012 10:11	Fichier PROJET RMC	1 Ko

Et si on change de branche, et que dans cette nouvelle branche on y ajoute un fichier `bidon.txt` (toujours avec un `git add`), on voit maintenant le fichier `bidon.txt` et on ne voit plus le fichier `essai.txt` alors qu'on est toujours dans le même dossier !

 application\src\ProjetRMC\src\essai.txt	06/09/2012 10:11	Document texte	1 Ko
 bidon	27/03/2013 15:56	Document texte	1 Ko
 local\essai.txt	06/09/2012 10:11	Fichier PROJET RMC	1 Ko

Pour pousser la nouvelle branche vers le serveur :

`git push origin nomDUneNouvelleBranche`

Après avoir travaillé sur une branche, on peut la supprimer ou la fusionner avec l'originale (merge).

Pour merger la branche avec la branche dont elle est issue, il faut se placer dans la branche à merger et faire :

`git merge nomDUneNouvelleBranche`

7. Mise en évidence et résolution des conflits

Un conflit apparaît dès que git cherche à fusionner deux versions d'un même fichier contenant des différences aux mêmes numéros de lignes de code.

Pour mettre en évidence un conflit, on peut créer dans deux branches le même fichier et en y ajoutant un contenu différent pour des mêmes numéros de ligne. Lors d'une tentative de merge, git vous indique le conflit :

```
$ git merge site_thematique
Auto-merging ProjetRMC/src/essai.txt
CONFLICT (add/add): Merge conflict in ProjetRMC/src/essai.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Il y a plusieurs façons de résoudre les conflits mais la procédure la plus simple consiste à éditer le fichier en cause, de choisir la version à conserver (en effaçant celle à supprimer) et de faire un nouveau `git commit`. Par exemple, nous avons un conflit sur la première ligne du fichier :

```
<<<<<< HEAD
Homer dit : cool mec !
=====
Homer dit : d'oh !
>>>>>> site_thematique
```

On ne conserve que la deuxième ligne :

```
Homer dit : d'oh !
```

8. Contribuer à un projet

Pour contribuer à un projet, il faut commencer par récupérer une copie du code source séparée de l'original. Cette opération est appelée fork. Elle se fait dans l'interface de GitHub en cliquant sur fork

Shopify / dashing

The exceptionally handsome dashboard framework
in Ruby and Coffeescript.

★ 3,164

🔗 202

Forks

Il faut ensuite cloner le projet (récupérer le code source sur sa propre machine) : git

clone <http://github.com/chacoux/MonProjet.git>

Votre copie sur GitHub s'appelle origin, et elle ne pointe pas vers le code original que vous avez forké. Pour suivre les évolutions du code original vous devez ajouter un autre non que origin : upstream :

git remote add upstream <http://github.com/chacoux/MonProjet.git> Enfin

pour récupérer les derniers changements fait sur le code original :

git fetch upstream

Si vous voulez pousser du code vers votre copie sur le serveur git

push origin master

Après avoir travaillé sur votre copie, vous voulez parfois fusionner (merge) votre travail avec des modifications qui ont été faites entre temps sur l'original par d'autres personnes :

git fetch upstream git

merge upstream/master

9. Suppression de fichiers

Il faut indiquer à git quel fichier supprimer : git

rm ./WebContent/WEB-INF/jsp/jobconf.jsp Ne

pas oublier de faire un git commit et git push
ensuite.

10. Annulation d'un commit

Pour afficher les commit :

git log

Puis q pour quitter Pour

effacer un commit :

git reset HEAD

- HEAD : dernier commit ;
- HEAD^ : avant-dernier commit ;
- HEAD^^ : avant-avant-dernier commit ;
- HEAD~2 : avant-avant-dernier commit (notation équivalente) ;

11. Référence sur git

Le site <http://gitref.org/> présente la

plupart des commandes git.