SCIENCES SORBONNE UNIVERSITÉ

Travaux Pratiques

UE : MU5EES05

Lab2 :
Gestion du développement logiciel
Git

Git est un logiciel de gestion de versions décentralisé. C'est un logiciel libre créé par Linus Torvalds, auteur du noyau Linux, et distribué selon les termes de la licence publique générale GNU version 2. Il s'agit du logiciel de gestion de versions le plus populaire. Vous trouverez en annexe un résumé des commandes git les plus populaires.

1. Installation des git packages :
   - First, let's install software packages that we will need throughout the practical labs :

     sudo apt install git gitk git-email

     - **Lancer un terminal, (ctl + alt + t)**

2. Git configuration :
   - After installing git on a your machine, the first thing to do is to let git know about your name and e-mail address (Ce qui est entre crochets est à remplacer en supprimant les [ ])

     git config --global user.name "[votre_prenom]"

     git config --global user.email [me@mydomain.net]

Such information will be stored in commits. It is important to configure it properly when the time comes to generate and send patches, in particular.

Vérifier que ces informations ont été bien pris en compte en utilisant les commandes suivantes : **git config --global user.name** et **git config --global user.email**

3. Créer un répertoire que vous appellerez test (commande **mkdir**).

4. Dans ce répertoire, créer un fichier 'nom de famille' et écrivez dans ce fichier le texte suivant : hello M2 syscom.

5. Mentionner que ce répertoire sera un git repository avec la commande **git init**

6. Utiliser la commande **git status** pour voir l'état du fichier test.

7. Utiliser la commande **git add** pour rajouter le fichier et retappez la commande **git status** pour vérifier si le fichier test à été rajouter à l'index ou pas.

8. Utiliser la commande : ***git commit -m "[Mettre votre message initial]"***

9. Créer un compte sur le git HUB et un nouveau repository. Ne rajoutez pas de fichier d'initialisation de fichier README dans votre repository. Laissez le vide pour éviter toute erreur. Vous pouvez le faire ultérieurement.

10. Connecter votre local repository au remote repository avec la commande git remote (copiez les deux commandes se trouvant dans votre compte git Hub et mettez à jour votre repository distant avec le contenu de votre répertoire se trouvant dans votre machine locale.

    Commandes :

    git remote add origin [https://github.com/votrecompte/toto.git]

    git push -u origin master

Note importante :

En tapant la commande git push –u origin master, vous allez être amenés à donner votre username sur github et votre mot de passe. Sachez que Git Hub et pour des raisons de sécurité, demande à chaque fois un code que vous devez générer vous-même à partir de votre compte GitHub et qui permet de sécuriser l'accès au repository. Pour avoir ce code, vous

devez aller dans setting puis **dans developer setting -> personnel access** Token puis Generate new token. N'oublier de cocher repo dans la section repo. Copier ce code dans un fichier .txt et utiliser le comme mot de passe pour pouvoir faire des push/pull.

11. Pour des raisons pratiques, nous souhaitons mémoriser le username et le code pour faire des push/pulls sans avoir à rentrer le mot de passe. Pour ceci avant de lancer une commande de push/pull, lancer la commande suivante :

git config credential.helper store

Vérifier que vous n'avez pas à fournir le username/password et que vous arrivez à modifier les fichiers dans votre repository et à tout envoyer sur gitHub sans avoir à fournir vos identifiants sur GitHub.

12. Créer un nouveau fichier README.md sur votre compte git HUB. Mettre un commentaire quelconque dans ce fichier.

Info : ***Markdown est un langage de balisage léger créé en 2004 par John Gruber, avec l'aide d'Aaron Swartz, dans le but d'offrir une syntaxe facile à lire et à écrire en l'état dans sa forme non formatée. Markdown est principalement utilisé dans des blogs, des sites de messagerie instantanée, des forums et des pages de documentation de logiciels. Les CR des labs sont à écrire en faisant appel à ce langage.***

13. A partir de votre repository local, utilisez la commande git pull pour mettre à jour votre répertoire locale. Vérifier que le nouveau fichier README.md est bien présent en local sur votre machine.

14. Aller vers votre répertoire /home/« username »/Downloads et créer un clone du répertoire que vous avez créé et vérifier que vous arrivez à récupérer tout le répertoire se trouvant sur git HUB.

Commande : **git clone [URL-to-git-HUB].**

15. Nous vous proposons maintenant de créer votre espace dépôt de l'UE MU5EES05. Créer sur votre bureau un répertoire qui s'appelle MU5EES05. Dans ce répertoire, créer 5 sous répertoire ayant comme nom : partie_1, partie_2, partie_3, partie_4, partie_5 et partie_6. Dans le répertoire partie_1, créer les sous répertoires lab2 et lab3 et lab4. Créer dans le répertoire lab2, un md file que vous appellerez cr_lab2.md. Rajouter dans ce fichiers les commandes git que vous avez tapez lors de ce lab (Réponses aux questions 1 à 14).

Note :

Sachez que vous êtes amené à créer un compte rendu pour chaque lab que vous appellerez cr_lab3.md, cr_lab4.md, cr_lab5.md. Le compte rendu contient les commandes utilisées pour répondre aux questions numérotées. Vous devez également inclure dans le répertoire en question correspondant au lab, vos commentaires et les codes sources que vous avez modifiés/développés.

16. Lancer la commande : **git init** Vérifier que le répertoire .git a été bien crée dans MU5EES05 (caché).

17. Créez un nouveau dépôt MU5EE05 sur GitHub.com. Pour éviter les erreurs, n'initialisez pas le nouveau dépôt avec des fichiers README ou licence. Vous pouvez rajouter ces fichiers après que votre projet MU5EES05 ait été déposé sur GitHub.

18. Suivez les étapes mentionnées précédemment (add to the **staging** area, commit) et lancer un push pour synchroniser votre repository local avec celui crée sur Git Hub.

19. Le repository que vous venez de créer sur GitHub devrait être privée. Si ce n'est pas le cas, modifier les droits pour que le repository devient désormais privée. Envoyer au user « khachicha » une invitation via GitHub pour lui autoriser l'accès. Vérifier avec votre encadreur de TP la bonne réception de l'invitation.

git cheat sheet

learn more about git the simple way at *rogerdudler.github.com/git-guide/*
cheat sheet created by Nina Jaeschke of *ninagrafik.com*

create & clone

create new repository	*git init*
clone local repository	*git clone /path/to/repository*
clone remote repository	*git clone username@host:/path/to/repository*

add & remove

add changes to INDEX	*git add <filename>*
add all changes to INDEX	*git add **
remove/delete	*git rm <filename>*

commit & synchronize

commit changes	*git commit -m "Commit message"*
push changes to remote repository	*git push origin master*
connect local repository to remote repository	*git remote add origin <server>*
update local repository with remote changes	*git pull*

branches

create new branch	*git checkout -b <branch>* *e.g. git checkout -b feature_x*
switch to master branch	*git checkout master*
delete branch	*git branch -d <branch>*
push branch to remote repository	*git push origin <branch>*

merge

merge changes from another branch	*git merge <branch>*
view changes between two branches	*git diff <source_branch> <target_branch>* *e.g. git diff feature_x feature_y*

tagging

create tag	*git tag <tag> <commit ID>* *e.g. git tag 1.0.0 1b2e1d63ff*
get commit IDs	*git log*

restore

replace working copy with latest from HEAD	*git checkout -- <filename>*