Commencer avec Git

Par le mangeur masqué

August 29, 2018

1 Obtenir le pouvoir de Git

Git est un logiciel local qui sert à sauvegarder, gérer et partager du texte. Github est une plateforme web qui héberge des repository git.

1.1 installation

Un simple sudo-apt-get install git devrait fonctionner. Pour vérifier sa bonne installation, il suffit de taper le nom du programme seul dans le terminal. Si git répond, c'est bon. Si le terminal dit que la commande n'est pas trouvée, ce n'est pas bon !

2 Introduction au repository git

Un dossier est un espace virtuel qui contient : des fichiers et d'autres dossiers. Un repo git est un dossier normal, qui répond en plus à des commandes git et possède un dossier caché '.git' qui stock des informations utiles.

2.1 Créer un repo git en local

Rappel de commandes de base pour le terminal : ls = list. Afficher le contenu du dossier courant. cd (nom de dossier) = change directory. Faire de (nom de dossier) le nouveau dossier courant. mkdir (nom de dossier) = make directory. Crée un nouveau dossier dans le dossier courant nommé (nom de dossier). touch (nom de fichier) = Crée un nouveau fichier dans le dossier courant nommé (nom de fichier).

Maintenant tu vas pouvoir entrer des commande sous ma guidance sacrée. La marche prévue est de taper la commande donnée entre apostrophe, puis de comparer ce qu'il se passe avec le petit texte de description ! Tu devrai vite saisir ce qu'il se passe.

Suite d'instructions pour créer un premier repo git (tu peux à tout moment faire 'ls' pour observer ce qui se passe) :

'cd' Sans argument, ramène l'utilisateur dans home, le dossier racine de l'utilisateur.

‘cd Document’ Transporte dans Document.

‘mkdir lab’ Crée le dossier lab.

‘cd lab’. ‘mkdir test’. ‘cd test’. Rien de nouveau pour ces trois commandes.

‘ls -a’ ls normal sauf que l’option -a permet de voir les fichiers et dossiers cachés. Normalement seuls un . et un .. apparaissent, qui sont les pointeurs respectifs sur le dossier courant et son dossier racine.

‘git init’ Convertit le dossier courant en repo git. Un nouveau ls -a fait apparaitre le .git dont il était question plus haut. L’utilisateur n’a en principe pas besoin de s’aventurer dedans mais il est possible de le faire.

2.2 Sauvegarder du code

Voilà, on est dans un repo git. Les commandes essentielles (ne pas taper): ‘git status’ permet de voir la situation de chaque dossiers et fichiers par rapport à git (non déclaré, ajouté et commis). ‘git log’ permet de voir l’historique des commits. ‘git add (file-name)’ déclare un fichier si ce n’est pas fait et prend note d’une modification éventuelle de son contenu. ‘git commit -m ”(message)”‘ sauvegarde un snapshot des dossiers ajoutés qu’il est ensuite possible de voir en faisant git log.

Note : à un moment ou un autre, git refusera d’executer une commande car il voudra pouvoir t’identifier avec un login et une addresse mail. Ce n’est pas grave, tu n’aura qu’à copier les commandes qu’il te donnera pour ce faire, puis relancer la commande.

Exemple de suite d’instruction pour sauvegarder du code : Toujours dans le repository test, faire :

‘touch main’, (Pour créer un fichier main. Il n’y a rien dedans mais ce n’est pas important, il pourrait aussi bien y avoir du code dedans).

‘git status’ (main est en rouge, état non pris en note)

’git add main’ (git prend note de main pour son futur commit).

’git status’ (on voit que main est passé en vert, il a été add)

‘git commit -m ”I just created the main in this snapshot.”‘

‘git status’ (git affiche all clear, rien a commit !),

‘git log’ (le journal affiche effectivement le commit avec son message, ce qui permet de savoir quel snapshot correspond à quoi ! Mieux vaut éviter de faire des messages de commit de type ”fuck that !” ou ”mdr” pour garder un journal exploitable XD.

Voilà la base de la sauvegarde dans git, bravo si tu es arrivé jusqu'ici ! Maintenant, répondons à la question originale : comment intéragir avec un repo sur github, qui ne nous appartient pas qui plus est.

3 Le Grand Jeu

3.1 Importer un repo git depuis l'exterieur.

Il est temps de dire adieux au repo test ! Pour travailler avec d'autres personnes sur un repo commun, voici comment les choses se passent : étape 1 : obtenir ta copie du repo en local.

Dans virtual box, dans le navigateur web, va à cette adresse : https://github.com/overetou/synthe_alea Clique sur le bouton vert ”Clone or download” puis, dans la fenêtre qui s'est ouverte, sur l'icone prise de note avec une flèche. Ca copie l'url du repo. Normalement si tu colle n'importe ou, l'addresse que je t'ai donné apparait, avec un .git à la fin.

Dans la console, dans le dossier qui te plaira, genre Document, tape : ‘git clone (colle l'adresse) (nom que tu veux donner au repo, qui n'apparaitra que pour toi. Ou rien, auquel cas le repo s'appelera comme sur github.)’

Git télécharge... Et le dossier apparait, avec tout le contenu à l'intérieur. Tu peux maintenant t'en servir exactement comme d'un repo local, faire tes modifs avec un editeur de texte, faire des add, des commits, ect... Tous ces changements resteront locaux, jusqu'à ce que tu les pousse sur le serveur avec une commande que nous n'avons pas encore vu.

3.2 Pousser une modif sur le serveur

Pour ce faire, tu dois commit une modification (sachant qu'il est nécessaire de faire des adds pour cela, qui définiront ce qu'il y a dans le commit.) Par exemple, crée un dossier doc dans lequel on pourra mettre de la documentation, add le et fait un commit ! Ne t'inquiète surtout pas, tu ne risque pas de déranger un équilibre fragile et je serai là pour tout remettre en ordre si c'est un champs de guerre :D. Ensuite, tape dans le terminal :

‘git push origin master’ Ce qui pousse le commit sur la branche principale (et pour l'instant unique) du projet. Il te demandera alors de donner tes identifiants github, pour vérifier que je t'ai autorisé à modifier mon repo, ce qui sera le cas car j'ai ajouté ton

compte en contributeur (tu dois encore accepter l'invit sur github au moment ou j'écris). Et voilà, tes changements et ton commit apparaitront sur le repo commun !

Par la suite, si je push des modifs sur github, tu n'aura qu'à faire 'git pull' dans le repertoire pour qu'elle apparaissent dans ton repo local. Si tu décide alors de faire un nouveau push, tu n'aura qu'à entrer 'git push', car le suffixe origin master n'est utile que pour le premier push, pour éviter des errreurs.

Te voila désormais armé pour survivre avec git ! Je te montrerai plus tard comment gérer certains cas comme les conflits et gestion de branches, mais saches que la doc est gratuite et excellente sur le site même de git ! Si tu veux faire quelque chose mais que tu ignore comment, tu trouvera ton bonheur dessus !