Git - Tuto Marion

Tuesday, 17 March 2020 15:41

GIT (local versioning)

https://openclassrooms.com/fr/courses/2342361-gerez-votre-code-avec-git-et-github/

BASICS

Create a repo
git init —> Tells to git that this repo. (folder) is a Git repo
Create a file in that repo (not in the command lines)
git add <myfile> —> Tells Git to it needs to look after that file, it actually adds that file to the INDEX
git commit myfile -m "My message" —> Tells git to register the changes on that file
git push --> pushes the commit to the global git

Every time you do a change, you're supposed to **git add** the file again, even if the file has already been added
=> In order to avoid that, you can directly commit using **git commit -a -m "My message"**

The -a option means « All files that you have in the index, add them again if they've been modified »

How to check commits history ?
git log

How to check the status ?
git status

How do I go back to a previous commit ? (= a previous version of my code)
git checkout commit_sha (sha is the « ID » of the commit, you get when doing git log)
You get back to the previous version of your code
Getting back to the last commit ?
git checkout master

Delete a commit (= a version of your code)

- I haven't committed yet, but I want to cancel all changes I haven't committed

git reset --hard

- I haven't pushed yet and want to change the name of the commit

git commit --amend -m "Votre nouveau message"

BRANCHES

BRANCHES

Check on which branch you are and branches available : **git branch**
Create a new branch : **git branch my_branch**
Change branch : **git checkout my_branch**
Merge branches : **git merge my_branch** (merges my_branch with the branch you're on)
Delete branch : **git branch -d my_branch**

GITHUB/GITLAB (online servers for remote versioning)

Online server adapted to git repo's, GitHub and GitLab are just concurrents

git@git.immc.ucl.ac.be:windTurbineERC/lofiipc.git

git push --set-upstream git@gitlab.example.com:namespace/nonexistent-project.git master
git push --set-upstream git@git.immc.ucl.ac.be:windTurbineERC/hifiipc.git master

REMOTE

The remote is the repo that is saved on the server, it is referred to as « origin »

Getting an existing GitLab repository on your local device :

- Through https

git clone https://gitlab.com/machflu/learning_git.git

- Through ssh

ssh-keygen -t rsa generate your ssh key ON THE CLUSTER & upload the public key (~/.ssh/id_rsa.pub) ON THE GITLAB SERVER
git clone git@gitlab.com:machflu/learning_git.git

git status

Sending code to GitLab
git add (suivi du nom du fichier) ou git add . (ajoute tout dans le dossier qui est untracked atm)
git commit
git push origin master origin is the remote repo (here: GitLab), master is the branch on which we want to push
Getting the last commits locally
git pull origin master

In case you get the following error
Delta compression using up to 24 threads.
fatal: unable to create thread: Resource temporarily unavailable
error: pack-objects died with strange error
Do
git config --global pack.windowMemory "100m"
git config --global pack.packSizeLimit "100m"

git config --global pack.threads "1"

Sometimes, Git is just going to mess up with you and won't be OK to pull telling you there are conflicts.
If you solve the conflicts, it tells you you need to commit the changes… which you don't want to do as you just wanted to update your version of the codes.

BRANCHES

You have two « kinds » of branches: the local ones and the remote ones. You have to make things such that my_branch (local) is connected to origin/my_branch.

Get new remote branches and new changes from remote branches : **git fetch**
See all branches (local and remote) : **git branch -av**

You can face recurrent situations:

1. You have created a new branch locally and want to push it on the remote, making the local track the remote
   Use the -u flag (upstream) when you make your first push: **git push -u origin my_branch**
2. There is a new branch on the remote and you want to get a local tracked version of it
   git checkout --track origin/my_branch tracks the remote branch and make a local one with the same name

(https://www.git-tower.com/learn/git/faq/checkout-remote-branch)

3. You already have a branch and you want to track a remote branch (quite similar)
   git branch --set-upstream-to origin/my_branch
4. Rename a local and remote branch in git
   1. Rename your local branch: **git branch -m new-name**
   2. Delete the old-name remote branch and push the new-name local branch: **git push origin :old-name new-name**
   3. Reset the upstream branch for the new-name local branch: **git push origin -u new-name**

If you want to change stuff on branchA, that is present both locally and remotely (the two are linked), you should make your modifications locally (**git checkout branchA**),
then you push that on the remote (**git push origin branchA**)

If you want to save some modifications before doing a pull and than get them back
(This typically happens when you want to push yet you're —not up to date => first you need to get up to date and only then you can push)
git stash —> allows you to keep your modifications without committing
you then go back to the last commit
git stash pop —> you get your modifications back (you will probably have to deal with conflicts at that time, see below)

If you want to force pull
git pull force
git fetch --all
git reset --hard origin/master

Delete local branch : **git branch -d my_branch**
Delete remote branch : **git branch -rd origin/my_branch** OR **git push origin —delete my_branch**

To get the changes that were made oh the remote (not pull them, just know they were made)
git remote update OR **git fetch**
Then you can **git status** and see what happened
If you want to see what's going on in the whole remote repo
git remote show origin

CONFLICTS

Dealing with conflicts

<<<<<<< HEAD —> where I am
Code
==========
Code
>>>>>>>>>> my_branch —> the branch I'm trying to merge into where I am
Chose the one you want (delete all the >><< HEAD/modif and code you don't want to keep)

git status
>> « unmerged paths »
You need to tell Git you solved the conflicts !
git add conflict_file
git commit (no message, just commit => knows you've fixed the conflicts - you can actually add a message if needed)

OTHERS

Checking who changed what line
git blame my_file
git log (find the sha of the commit)
git show (shows you exactly what was changed at the commit)

Ignoring some files —> .gitignore file
gitignore is a file just like the others

Find the commit reference
git rev-parse HEAD
>> d58e9d552fb57c7877c1946db22c5573f53e5951

Pour apprendre un peu comment utiliser le terminal, tu peux commencer avec ça (slides 22 à 36).

Et après, si tu veux vraiment devenir ultra geek, tu peux apprendre à éditer des fichiers de texte directement depuis le terminal en utilisant *vim* (https://www.linux.com/tutorials/vim-101-beginners-guide-vim/).