GIT - Introduction git en tant qu'outil de versionning

<https://www.youtube.com/watch?v=V6Zo68uQPqE>

# Configuration initiale

git config –global user.name "Luc DUPONT"

git config –global user.email [luc@dupont.fr](mailto:luc@dupont.fr)

git -l

ex)

JPM@JPM-PC MINGW64 ~/test (master)

$ git config --global user.name "Jean-Paul MEIRSMAN"

JPM@JPM-PC MINGW64 ~/test (master)

$ git config --global user.email "jean-paul@meirsman.fr"

# Connaitre la version de git

git --version

which git

ex)

JPM@JPM-PC MINGW64 ~/test (master)

$ git version

git version 2.7.2.windows.1

# Démarrer git ou le réinitialiser (commencement du versionning)

git init

JPM@JPM-PC MINGW64 ~/test (master)

$ git init

Reinitialized existing Git repository in C:/Users/JPM/test/.git/

# Connaitre l'état des modifications du projet

git status

JPM@JPM-PC MINGW64 ~/test (master)

$ git status

On branch master

Initial commit

nothing to commit (create/copy files and use "git add" to track)

JPM@JPM-PC MINGW64 ~/test (master)

$ notepad aaa

JPM@JPM-PC MINGW64 ~/test (master)

$ git status

On branch master

Initial commit

Untracked files:

(use "git add <file>..." to include in what will be committed)

aaa.txt

nothing added to commit but untracked files present (use "git add" to track)

# Commit - pour prendre un "cliché/snapshot" de l'état des fichiers

(permet de tracker les fichiers souhaités) - le git status montre les fichiers en rouge ceux non trackés et en vert ceux qui le sont.

git add nomfichier

git status

JPM@JPM-PC MINGW64 ~/test (master)

$ git add aaa.txt

JPM@JPM-PC MINGW64 ~/test (master)

$ git status

On branch master

Initial commit

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: aaa.txt

JPM@JPM-PC MINGW64 ~/test (master)

$ git commit --message "first commit"

[master (root-commit) 9f46050] first commit

1 file changed, 3 insertions(+)

create mode 100644 aaa.txt

JPM@JPM-PC MINGW64 ~/test (master)

$ git status

On branch master

nothing to commit, working directory clean

JPM@JPM-PC MINGW64 ~/test (master)

$ notepad aaa.txt

JPM@JPM-PC MINGW64 ~/test (master)

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

modified: aaa.txt

no changes added to commit (use "git add" and/or "git commit -a")

## Connaitre les modifications

JPM@JPM-PC MINGW64 ~/test (master)

$ git diff

**diff --git a/aaa.txt b/aaa.txt**

**index 081ffc7..277974c 100644**

**--- a/aaa.txt**

**+++ b/aaa.txt**

@@ -1,3 +1,4 @@

zaerzareazrzraze

azerze

-rz

\ No newline at end of file

+rz

+sdfgsdfg

JPM@JPM-PC MINGW64 ~/test (master)

$ git add aaa.txt

JPM@JPM-PC MINGW64 ~/test (master)

$ git commit --message "second commit"

[master 3c7cd11] second commit

1 file changed, 2 insertions(+), 1 deletion(-)

JPM@JPM-PC MINGW64 ~/test (master)

on peut utiliser

git diff fichier/répertoire

# Lister tous les commit réalisés

JPM@JPM-PC MINGW64 ~/test (master)

$ git log

commit 3c7cd1159b952a8c9a98d7aeeca1f60d846663f4

Author: Jean-Paul MEIRSMAN <jean-paul@meirsman.fr>

Date: Tue Mar 8 23:22:01 2016 +0100

second commit

commit 9f46050a411078a1ba5befd50a82f54a93dffb2a

Author: Jean-Paul MEIRSMAN <jean-paul@meirsman.fr>

Date: Tue Mar 8 23:11:28 2016 +0100

first commit

GIT - Introduction - Les branches

# Créer une branche

Cela est utile pour travailler sur un des aspects du projet

JPM@JPM-PC MINGW64 ~/test (master)

$ git branch MaBranche

# Afficher les branches

La branche active est précédée d'une astérisque et est en vert

JPM@JPM-PC MINGW64 ~/test (master)

$ git branch

MaBranche

\* master

# Changer de branche

JPM@JPM-PC MINGW64 ~/test (master)

$ git checkout MaBranche

Switched to branch 'MaBranche'

JPM@JPM-PC MINGW64 ~/test (MaBranche)

$ git branch

\* MaBranche

master

Le changement de branche nous fournit, dans notre répertoire, la version des fichiers de la branche.

Regardez le nombre d'octets du fichier aaa.txt pour comprendre le changement.

JPM@JPM-PC MINGW64 ~/test (master)

$ git checkout MaBranche

M aaa.txt

Switched to branch 'MaBranche'

JPM@JPM-PC MINGW64 ~/test (MaBranche)

$ notepad aaa.txt

JPM@JPM-PC MINGW64 ~/test (MaBranche)

$ git add aaa.txt

JPM@JPM-PC MINGW64 ~/test (MaBranche)

$ git commit --message "modif3"

[MaBranche 6497227] modif3

1 file changed, 7 insertions(+)

JPM@JPM-PC MINGW64 ~/test (MaBranche)

$ ls -l

total 1

-rw-r--r-- 1 JPM 197121 **72** mars 8 23:44 aaa.txt

JPM@JPM-PC MINGW64 ~/test (MaBranche)

$ git checkout master

Switched to branch 'master'

JPM@JPM-PC MINGW64 ~/test (master)

$ ls -l

total 1

-rw-r--r-- 1 JPM 197121 **40** mars 8 23:46 aaa.txt

# Modifications simultanées dans la branche master et une autre

git checkout master

git diff master..MaBranche

git merge MaBranche

Le merge va créer un nouveau commit comportant tous les changements de tout le monde.

Si la même ligne est modifiée dans deux branches différentes il se produit un conflit lors du merge. En éditant le texte on voit la ligne en conflit avec des <<<<<<<<<<<<<< ============== et >>>>>>>>>>>>>>>> montrant les deux versions. il faut donc qu'un humain prenne la décision de modifier le fichier en question. Un git status nous montrera que le fichier a été modifié deux fois puis faire un add puis un commit.

# Suppression d'une branche

git branch -d NomDeLaBranche

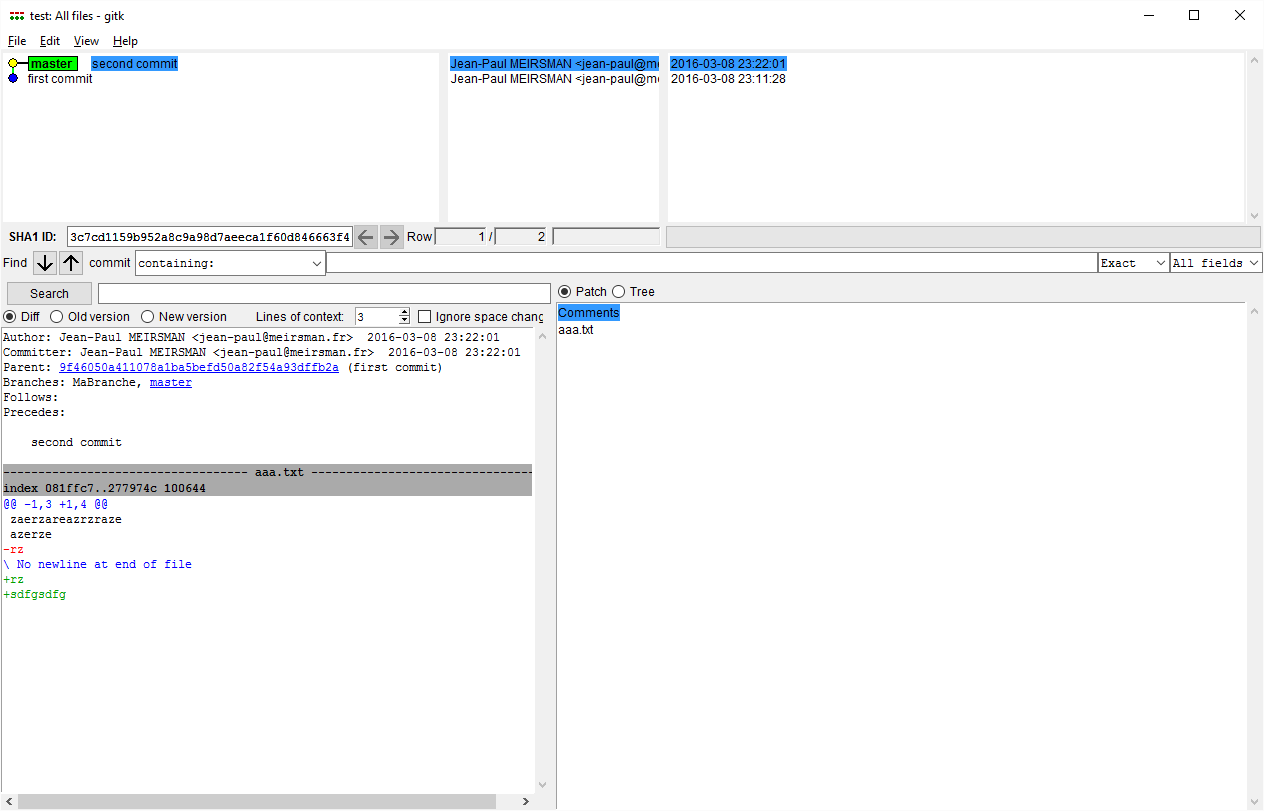
C'est une bonne pratique une fois que la branche n'a plus de raison d'être.

Si une branche avait des opérations non encore mergées dans master, la demande de suppression de branche sera refusée. Si l'on veut vraiment la supprimer en force il faut utiliser l'option -D au lieu de -d.

La bonne pratique est qu'une branche existe pour un objectif précis d'une durée courte (ex : 2 jours). Ensuite elle doit être mergée puis supprimée. Les branches trop anciennes pourraient induire des difficultés pour le merging car le projet aurait trop divergé et l'on ne saurait plus comment accepter la bonne version de modification.

# Outil graphique pour visualiser l'historique des commit

Il suffit juste de taper gitk en mode console et une fenêtre graphique s'ouvre sous Windows.



Un site pour visualiser les commit

<http://www.wei-wang.com/ExplainGitWithD3/>

# Placer son travail sur github.com

Créer un compte sur <http://github.com>

Github vous indique quelles lignes à entrer

git remote add origin https://github.com/jpmeirsman/test.git

git push -u origin master

ex)

JPM@JPM-PC MINGW64 ~/test (master)

$ git remote add origin <https://github.com/jpmeirsman/test.git>

JPM@JPM-PC MINGW64 ~/test (master)

$ git remote -v

origin https://github.com/jpmeirsman/test.git (fetch)

origin https://github.com/jpmeirsman/test.git (push)

Pour vérifier si l'on a bien une connexion distante

JPM@JPM-PC MINGW64 ~/test (master)

$ git push -u origin master

Counting objects: 6, done.

Delta compression using up to 8 threads.

Compressing objects: 100% (2/2), done.

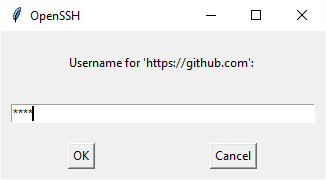
Writing objects: 100% (6/6), 498 bytes | 0 bytes/s, done.

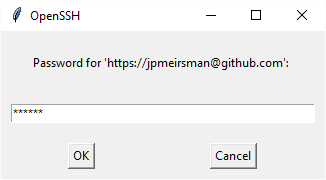
Total 6 (delta 0), reused 0 (delta 0)

To https://github.com/jpmeirsman/test.git

\* [new branch] master -> master

Branch master set up to track remote branch master from origin.





Pour votre nom de compte et password

Lors du push, origin représente le serveur github et master la branche.

Dans github, l'outil commit permet de voir tous les commit et l'outil graph permet de voir toutes les branches. Le survol avec la souris sur les différents points montre les commit.

# Récupérer les modifications des autres utilisateurs

git pull <server> <branch>

git pull origin master

C'est utile aussi après un merge effectué par d'autres utilisateurs

# Applications graphiques

<http://mac.github.com>

<http://windows.github.com>

Permet de surveiller les modifications apportées aux fichiers et de faciliter les commit.

Il faut faire commit et publish afin que les modifs soient effectives.

# Contrôle du code par les autres contributeurs

On demande à d'autres contributeurs de valider le travail (Code Review)

Sur github faire un "Compare et Pull Request", mettre un message et enfin "Create pull request

Hello @user... dans le message envoie un mail à user

On peut voir le code en vue splittée dans l'onglet "Files changed"

Possibilité de feedback ligne par ligne and "Merge pull request" à la fin du Code review puis Confirm et finalement effacer la branche.

Possibilité d'ajouter du code avec ``` ``` (caractère 96 en ascii ou AltGr-7)

ex)

```css

body {

code: red;

}

```

puis Update ou Crayon pour rééditer

git fetch rapatrie une autre branche sans faire les avances de commit

git branch -a liste toutes les branches

Mais la bonne pratique d'être seul sur une branche (un peu comme dans la vraie vie :) ) sinon il y a des risques de collision.

Le pull c'est fetch et un merge.

Le Fork permet de recopier un repository d'un autre utilisateur. Ensuite il faut le cloner. Cela ne fait pas de nous un contributeur.

git clone git@github.com:myuser/repository.git

myuser est notre compte

Le clonage va importer également tout l'historique du repository.

git commit --amend -m "nouveau message" permet de modifier le commit

on push

git push origin mabranche

Un pull request proposera la modification aux développeurs du projet et non plus sur notre master

# Publier son site Web

git branch gh-pages

git checkout gh-pages

voir dans settings de github pour avoir le lien où est publié le site web

jekyll est un bon outil pour faire un site web avec des pages statiques

le fichier CNAME permet de créer un nom de domaine à la place de celui proposé par github mais il faut gérer son dns

Il existe un dossier .git dans votre dossier git contenant la configuration de votre git local

git add .

ajoute tous les éléments

git tag 1.0 permet de revenir sur une version antérieure?

ungit permet de sortir du git

# Resynchronisation d'un repo externe

permet de bénéficier des commits des autres branches tout en continuant à travailler sur sa branche

git remote upstream <https://github.com/user/repo.git>

git fetch upstream

permet de mettre à jour notre fork

git checkout Unebranche

git merge upstream

git push origin

pour envoyer les modifs

Pour copier un fichier du staging vers le working directory (donc supprimer les modifications en cours), on utilise git checkout.

Pour visualiser les modifications entre le répertoire de travail et la zone de staging, on utilisegit diff.

Pour visualiser les modifications entre la zone de staging et le dernier commit, on utilise git diff --cached.