Introduction à Git

Git est un outil de gestion de version qui permet de naviguer dans l'historique de votre projet. À la base créé par Linus Torvalds pour gérer les sources du noyau Linux, Git est désormais le logiciel de gestion de version le plus utilisé au monde. Les systèmes de contrôle de version aident les utilisateurs à analyser plus facilement les modifications et les contributions apportées à du code collaboratif.

Utilité

Les gestionnaires de version comme git présente un intêret certain pour favoriser le FAIR et l'open data et de façon plus générale la reproductibilité en informatqiue et en science des données

- sauvegarder les modification ou *commits* representant différentes étapes d'elaboration d'un projet de code, de document, de données;
- consulter l'historique de vos commits à travers le temps ;
- récupérer des codes ou documents correspondant à une date ou version donnée ;
- partager votre version d'un projet entre collaborateurs ou le public ;
- résoudre les conflits de versions entre modifications incompatibles ;

git est à la base un outil en ligne de commande. Des intrefaces graphiques peuvent s'y ajouter. Des site web comme des forges ou github, gitlab ou bitbucket ajoute à git des fonctions collaboratives et de gestion de projet.

Git est un système de gestion de version **décentralisé**.

Les données du dépôt ne se trouvent pas seulement un serveur distant mais également sur votre machine.

- vous pouvez travailler sans réseau;
- nombre de depots limité uniquement limité pat lespace de stockage.

Installation de GIT

Linux:

Debian / Ubuntu :

```
sudo apt-get update
sudo apt-get install git
```

Centos/Red hat/Fedora :

```
sudo yum install git
```

Windows:

https://git-for-windows.github.io

MacOS:

https://sourceforge.net/projects/git-osx-installer/files/

Vérifier la version:

```
git --version
```

Utilisation

Le workflow git de base est le plus souvent:

- git config
- git init/clone
- créer/modifier fichier
- git add
- git commit
- git push

Nous allons étudier ces commandes principales et bien d'autres

Configuration initiale: config

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

git config --global user.email "myemail@zmail.fr"
```

Recopier un dépôt existant: clone

git clone vous permet de créer une copie du dépôt pour le modifier en local.

```
git clone https://github.com/fjrmoreews/umt-reproducibility.git
```

Créer un nouveau dépôt: init

Pour initialiser un dépôt, il suffit de se placer dans le répertoire des sources, et :

```
git init
```

Ajouter des fichiers : add, commit

```
git add .

git commuit -m "first commit"

#git commit -a
```

Ces commandes initialisent un nouveau dépôt(repository), indiquent à git d'y placer le répertoire courant, et de lancer un commit.

Supposons que j'aie modifié un fichier README.md à la racine de mon répertoire. Je vais donc commiter :

```
git add   README.md
git commit
```

Pour que la commande prenne en compte automatiquement les fichiers modifiés , Ils doivent être ajouté avec la commande ‘add’

```
git add <file,directory>
```

Etat d’un dépôt existant: status

L’état (status) d’un dépôt décrit les différences avec le dé »pot de référence, si il existe, les nouveaux fichiers et plus générakement le smodification dans l’arborescence. Pour afficher l’état du dépot, on utilise la commande :

```
git status
```

Variable Head

Git posséde une variable de référence nommée HEAD qui pointe vers le dernier commit dans la branche d'extraction courante. Vous pouvez imaginer HEAD comme étant la "branche commist actuelle".

Si nous faisons un nouveau commit dans le dépôt, alors HEAD va référencer le nouveau commit.

Mise à jour d’un dépôt distant: push

Pour appliquer vos modificatiosn dans le dépôt distant de référence ::

```
git push origin master
```

Ici, c’est la branche master qui sera impactée.

les modifications dans **HEAD** sont envoyés au dépôt distant

Lier un dépôt à un dépôt distant

On doit etablir une connexion entre un dépôt local et un serveur distant, sauf si on a cloner le dépôt.

C’est noamment le cas, pur un depot créer avec init ou une copie issue d’archive par exemple

commande pour lier les dépôts :

```
git remote add origin <url_repository>
```

On a ainsi défini l’origine du dépôt local (origin)

Afficher l'historique des modifications: log, diff, show

Pour afficher l'historique des commits sur la branche en cours :

```
git log
```

example de sortie :

```
commit dab43fcd6f75d4d54362c022b3cada2fb64563da
Author: MORELS Franz <fmorels21@zmail.fr>
Date:   Mon Jun 7 16:20:52 2021 +0200
    paper intro improvement

commit cea94a43390be305001b56bd08df50496be36360
Author: MORELS Franz <fmorels21@zmail.fr>
Date:   Fri Jun 4 12:42:07 2021 +0200
```

```
    SQL fix
```

un commit par ligne :

```
git log --pretty=oneline
```

les commits d’un utilisateur, :

```
git log --author =username
```

Remplacement des modifications locales :

```
git checkout -- <file name>
```

Cet commande remplace les changements d’arbre de travail par les dernières données présentes dans le HEAD . Les changements qui ont été ajoutés à l’index (commit) ne seront pas écrasés.

Suppression de tous les changements locaux / réinitialisation

```
git fetch origin git reset --hard origin/master
```

tous les changements / commits locaux vont être supprimés et que la branche master locale va pointer vers le commit le plus récent du serveur,:

Afficher différences et modifications

Pour voir la liste des différences depuis le dernier commit :

```
git diff
```

Pour générer la liste des fichiers modifiés depuis le premier commit :

```
git log
git diff --name-only
```

Pour voir un commit spécifique :

```
git show <commit_id>
```

(Note <*commit*> est une branche, un tag ou un identifiant de la forme e73316b2e..., qu'on récupère dans les logs.

Modifier, annuler des modifications/commit:

Annuler le dernier commit :

```
git reset --hard HEAD^
```

l'option –hard force la modification local

Utilisez cette option si vous êtes certain de vouloir tout effacer après le commit choisi.

Annuler un commit quelconque :

Obtention du commit_id:

```
git log
```

Apres avoir identifier le commit_id, taper :

```
git reset --hard <commit_id>
```

Vous pouvez alors appliquer vos changements dans le dépôt distant. Pour cela, ajouter l'option

–force (-f) à la command push :

git push -f

Créer des versions stables : tag

Parfois, on veut marquer un état précis de l'arbre du développement, pour créer, par exemple, une version stable.

```
git tag <mon_tag>
```

Pour voir tous les tags existants :

```
git tag -l
```

Créer sa version d'un code communautaire : checkout , branch

Git peut gérer plusieurs branches de développement. Pour créer une nouvelle branche à partir de la branche courante :

```
git checkout -b  <my_branch>
```

Pour lister les branches existantes :

```
git branch
```

Pour passer d'une branche à l'autre :

```
git checkout <my_branch>
```

Revenir à la branche master principale :

```
git checkout master
```

Supprimer une branche localement :

```
git branch -d  my_branch
```

Supprimer une branche distante:

```
git push origin --delete  my_branch
```

Mise à jour et fusion:

Dans le cas où vous souhaitez mettre à jour votre dépôt local, on peut utiliser :

git pull

Pour fusionner (merge) une autre branche dans celle actuellement active, utilisez:

git merge my_other_branch .

Avec push ou merge, GIT gére les conflits automatiquement. Si ce n'ets pas possible, il y a un conflit. Il faut alors résoudre le conflit soi-même :

par exemple, afficher les differences :

git diff ou git status,

modifier les fichiers,

puis git add myfile