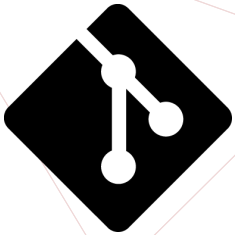


Maxime Chupin
chupin@ceremade.dauphine.fr
www.ceremede.dauphine.fr/~chupin

CNRS, CEREMADE, Université Paris-Dauphine

27 juin 2023

Introduction à Git

logiciel de gestion de versions

inspiré de la présentation de D. Giorgi pour Infomath

Des documents qui évoluent

Divers stratégies

Nommage de fichier

```
19-11-07-doc.tex
19-11-07-doc_initialEdits.tex
19-11-20-doc.tex
20-03-24-doc_PNASsubmitted.tex
20-04-16-doc_PLOSrevision.tex
20-05-08-doc_PLOSpublished.tex
```

Overleaf

etc.

Seul·e… ou à plusieurs

> *« Je travaillais avec un collègue et il a écrasé ma dernière version »*

> *« J'ai deux dernières versions et je ne sais pas comment les fusionner correctement »*

> *« Je n'avais pas de sauvegarde et mon ordinateur a planté/a été volé »*

Systèmes de contrôle de version...

Définition

Système qui permet de stocker un ensemble de fichiers en conservant la chronologie de toutes les modifications qui ont été effectuées dessus. Il permet notamment de retrouver les différentes versions d'un lot de fichiers.

Plusieurs systèmes possibles :

- Git ;
- SVN ;
- Mecurial ;
- etc.

Utilisation

- Développement de code ;
- Écriture de document LaTeX ;
- etc.

Les contributeurs et contributrices peuvent consulter l'historique du projet pour retrouver :

- Quels changements ont été apportés ?
- Qui a effectué ces changements ?
- Quand ces modifications ont-elles été apportées ?
- Pourquoi des changements étaient-ils nécessaires ?

Définition (Git)

Git est un logiciel de gestion de versions décentralisé. C'est un logiciel libre et gratuit, créé en 2005 par Linus Torvalds, auteur du noyau Linux, et distribué selon les termes de la licence publique générale GNU version 2.

Décentralisé/distribué

- Les systèmes de contrôle de version distribués n'ont pas besoin d'être constamment connectés à un dépôt central.
- Au lieu de copies de dépôts à distance, on travaille avec des dépôts locaux.

Dépôt local/distant

- Git peut s'utiliser simplement localement
- Souvent préférable d'avoir un dépôt d'hébergement distant avec interface web

Interface graphique (locale)

Deux outils que je recommande

- Gittyup `https://murmele.github.io/Gittyup/`, libre, multiplateforme
- Sublimemerge `https://www.sublimemerge.com/`, propriétaire, mais utilisable gratuitement, multiplateforme

Plugins en tout genre

https://git-scm.com/downloads

Configuration de base (indépendante des projets)

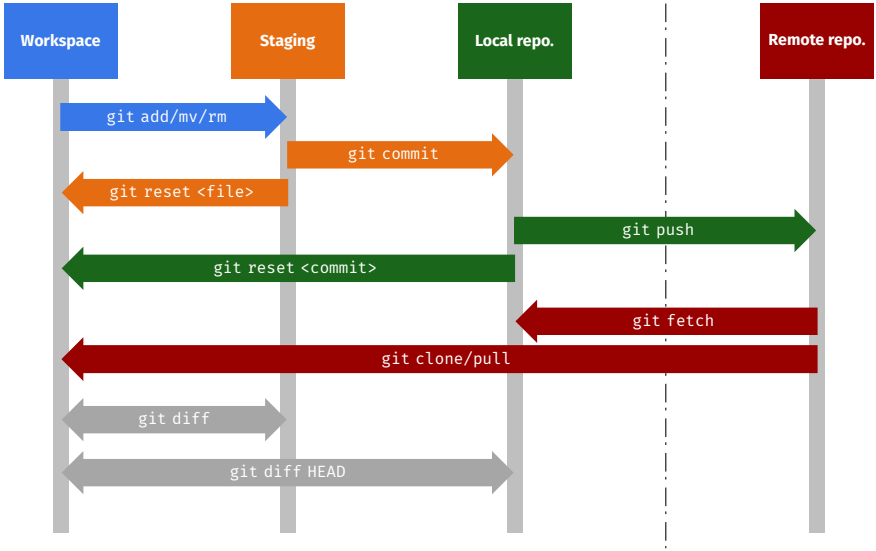
```
user $> git config --global user.name "Mon nom"
user $> git config --global user.email "mon.nom@mail.fr"
user $> git config --global core.editor "vscode" # optionnel
```

Possible aussi avec Gittyup et Sublimemerge.

En ligne de commande

- Quelques commandes de base (à utiliser après `git`) :
  - `init`
  - `add`
  - `commit`
  - `reset`
  - `push`
  - `pull`
  - `branch`
  - `checkout`
  - `merge`

Structure


Workspace
Staging
Local repo.
Remote repo.
git add/mv/rm
git commit
git reset <file>
git push
git reset <commit>
git fetch
git clone/pull
git diff
git diff HEAD

Description

Voir l'index (*staging area*) comme un aperçu de votre prochaine soumission (*commit*)

Avantages

- Séparer un grand changement en plusieurs
- Examiner les changements
- Les fusions (*merge*) sont régulièrement source de conflits
- etc.

Création d'un répertoire Git

- Depuis un dépôt local (création d'un répertoire `.git` dans le répertoire de travail)

```
user $> cd path/to/directory
user $> git init [project-name]
```

- Depuis un dépôt distant

```
user $> git clone [url]
```

Après avoir édité un fichier

- Lister les nouveaux fichiers ou les modifications faites

```
user $> git status
```

- Montrer les différences

```
user $> git diff [optional filename/repository]
```

Premières commandes (3)

- Ajouter ou enlever un fichier de l'index (*staging area*)

```
user $> git add [filename]
user $> git reset [filename] # unstages
user $>
   git rm --cached [filename] # unstages and untracks
```

- Enregistrer les changements dans l'historique des versions

```
user $> git commit -m "[message]"
```

Ignorer des fichiers

Premières commandes (4)

- Fichier `.gitignore` (à ajouter au dépôt) :

```
*.aux
images/toto.jpg
```

- Exemples pour différents types de projets ici :

https://github.com/github/gitignore

Visualisation et historique

- Historique à partir du HEAD

```
user $> git log
```

- Graphe

```
user $> git log --oneline --graph
```

De nombreux autres outils de visualisation

Avec Gittyup

Dépôt distant

- Github, Gitlab, ou PLMlab, instance Gitlab de l'INSMI du CNRS (mathrice)
- Création du dépôt avec le navigateur web
- Utilisation des commandes `clone, fetch, merge, pull` et `push`
- Fichiers `README.md` (*markdown*) permet de produire une page web d'information sur le projet,
- Publier ses codes de calculs des papiers
- Travailler à plusieurs
- Intégration continue
- Signalement de bugs, demande de fonctionnalités, proposition d'ajout, etc. via les tickets

Travailler avec un répertoire distant (1)

- Cloner

```
user $> git clone [url]
```

- Télécharger l'historique distant et fusionner avec le répertoire local

```
user $> git fetch
user $> git merge
```

Travailler avec un répertoire distant (2)

- Télécharger et fusionner

```
user $> git pull
```

- Envoyer sur le répertoire distant

```
user $> git push
```

Demo PLMlab

Les branches (1)

- Travailler sur une « copie » du dépôt, de manière isolée
- Une fois les modifications voulues terminées, on peut les fusionner avec la branche principale (ou non)
- La branche par défaut créée par Git s'appelle `master` ou `main`

Commandes

- Voir la branche courante

```
user $> git branch
```

Les branches (2)

- Créer une branche

```
user $> git switch -c mabranche
user $> # ou
user $> git checkout -b mabranche
```

- Changer de branche

```
user $> git switch mabranche
```

Les branches (3)

- Supprimer une branche

```
user $> git branch -d mabranche
```

- Fusionner deux branches

```
user $> git switch master # (ou main) pour se mettre dans
    la branche destinataire
user $> git merge mabranche
```

On peut aussi tout faire avec Gittyup ou Sublimemerge

Références

- Exposé de Daphné Giorgi au séminaire infomath : https://infomath.pages.math.cnrs.fr/talk/2022-2023/git/git_2022.pdf
- Tutoriel de Romain Theron : https://plmlab.math.cnrs.fr/theron/formation_git/
- Manuel de référence : https://git-scm.com/docs
- Le livre de référence : https://git-scm.com/book/en/v2
- *Cheatsheet* interactive : https://ndpsoftware.com/git-cheatsheet.html
- *Cheatsheet* de Github : https://training.github.com/downloads/github-git-cheat-sheet.pdf

Merci !