M1 SC, Université de Lorraine Sept. 25, 2020

Introduction à Git

Chuyuan Li

Consignes d'exercices
- Ce TD comprend deux parties : Pour la première partie, nous allons regarder ensemble une exemple de Git. Ensuite vous commencez les exercices basiques ci-dessus.
- Pour ceux qui n'ont jamais utilisé Git, pour le système Windows télécharger depuis : `https://gitforwindows.org`. Pour Mac et Linux, voir ici `https://www.atlassian.com/git/tutorials/install-git`. Ensuite configurer votre username et email avec :

```
git config --global user.name "votre non et prénom"
git config --global user.email votre-email@example.com
```

- Pour ceux qui connaissent déjà bien Git, après avoir fini les exercices en-bas, vous pouvez avancer sur les exercises sur ce site : `https://learngitbranching.js.org`, ou apprendre sur celui-ci : `https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud`.

1 git init, git add, git commit

Git commit

1. Ouvrir un terminal et basculer dans un répertoire pour les exercices de ce cours.
2. Initialiser un nouveau répertoire git avec la commande `git init depot`.
3. Observer les nouveaux fichiers créés, qu'est-ce qui est dans `.git/` ?
4. Utiliser `git status` pour regarder sur quelle branche vous êtes ?
5. Qu'est-ce que `git log` vous donne ?
6. Créer un fichier vide, et le nommer `file1`.
7. A quoi `git status` ressemble-t-il ?
8. Changer le contenu du fichier `file1`.
9. A quoi `git status` ressemble-t-il ?
10. Ajouter à l'index le changement que vous venez de faire avec `git add file1`.
11. A quoi `git status` ressemble-t-il ?
12. Faire un commit avec `git commit -m "message"`.
13. A quoi `git status` ressemble-t-il ?
14. Changer encore le contenu du fichier, ajouter et commiter avec un message.
15. Tester avec `git log -n 2`, `git log --oneline` et `git log --oneline --graph`.

2 git status, git log

Ce sont les deux commandes utiles pour suivre le processus de votre travail. Chaque fois quand vous faites un changement dans le répertoire (créer un nouveau fichier, modifier le contenu d'un fichier, ajouter à l'index et commiter, etc.), vous pouvez lancer ces commandes.

1. git commit -m "commit message"
2. git log
3. git log -n 5
4. git log –oneline
5. git log –oneline –graph

3 git diff, git reset, git checkout

Git staging (Indexation)

1. Afficher le contenu du fichier `file1.txt` avec `cat`.
2. Écraser le contenu dans le fichier avec `echo 2 > file1.txt`. (Ou bien `sc file1.txt '2'` dans PowerShell).
3. Qu'est-ce que `git diff` affiche ? Et `git diff --staged` ?
4. Executer `git add file1.txt` pour indexer les changements.
5. Qu'est-ce que `git diff` affiche ? Et `git diff --staged` ?
6. Écraser le contenu dans le fichier avec `echo 3 > file1.txt`. (Ou bien `sc file1.txt '3'` dans PowerShell).
7. Qu'est-ce que `git diff` affiche ? Et `git diff --staged` ?
8. Expliquer ce qui s'est passé.
9. Executer `git status` et observer que le fichier est apparu deux fois.
10. Executer `git reset HEAD file1.txt` pour désindexer le changement.
11. Qu'est-ce que `git status` affiche ?
12. Indexer le changement et faire un commit.
13. Observer le log.
14. Écraser le contenu dans le fichier avec `echo 4 > file1.txt`. (Ou bien `sc file1.txt '4'` dans PowerShell).
15. Qu'est-ce que le contenu du fichier ?
16. Observer `git status`.
17. Executer `git checkout file1.txt`. Que contient le fichier ?
18. Observer `git status`.

4 git branch, git checkout, git merge

Git branching

1. Utiliser `git branch` pour observer les branches présentées dans le répertoire.
2. Créer une nouvelle branche avec `git branch [nom_de_la_branche]`.
3. Sur quelle branche êtes-vous positionnés ?
4. Utiliser `git checkout [nom_du_branch]` pour basculer sur une autre branche.
5. Observer les changement du `git status` quand vous basculez d'une branche à l'autre.
6. Positionnez-vous sur la nouvelle branche et créer un fichier `file2.txt`. Ajouter et commiter ce changement.
7. Utiliser `git log --oneline --graph` pour visualiser.
8. Basculer sur la branche `master` et visualiser avec `git log --oneline --graph`. Qu'est-ce qui change ?
9. Créer un deuxième fichier `file2.txt`, ajouter et commiter.
10. Utiliser `git log --oneline --graph --all` pour observer les deux branches.
11. Basculer sur votre branche. Le fichier `file2.txt` existe-t-il?
12. Utiliser `git diff [nom_du_branch] master` pour voir la différence entre ces deux branches.
13. Basculer sur la branche master. Merger votre propre branche dans master avec `git merge [nom_de_la_branche]`.
14. Utiliser `git log --oneline --graph` pour visualiser.

5 git clone, git fetch, git pull, git push

Git collaboration avec binôme

1. Ouvrir un compte sur Github. Vous pouvez suivre le guide `https://guides.github.com/activities/hello-world/`.
2. Créer un nouveau répertoire sur votre github, rendez-le "public".
3. Cloner ce répertoire git avec `git clone [https]`.
4. Sur votre machine, créer une branche A et un nouveau fichier. Ajouter, commiter et pusher ces changement sur votre branche A avec `git push`.
5. Sur la machine de votre binôme, créer une branche B et un nouveau fichier. Ajouter, commiter et pusher ces changement sur la branche B.
6. Récupérer sur votre machine les changements pushés par votre binôme avec `git fetch`. Pareil pour votre binôme.
7. Si tout se passe bien, synchroniser votre répertoire local en utilisant `git pull`. Pareil pour votre binôme.
8. Votre binôme modifie le fichier sur la branche A et pusher la branche A. De votre côté, modifier le même fichier et sans le pusher. Quand vous faites `git pull`, qu'est-ce qui se passe ? Régler le conflit en suivant les consignes.
9. Une fois le conflit réglé, pusher ce fichier de votre côté.