POLYTECH SORBONNE

Automne 2020

Plus de commandeset git

TP 2

Objectifs

- ✯ Utiliser Git.
- ✯ Manipuler de nouvelles commandes.
- ✯ Composer des commandes.

Rendu

- ✏ Le TP est à rendre en mettant vos fichiers dans votre dépôt *github*.
- ✏ Les fichiers doivent être ajoutés au commit, comités et enfin pushés sur le serveur

```
git add fichiers_ajoutés_ou_modifiés_pendant_le_tp
git commit -m''un message''
git push
```

- ✏ Les fichier doivent être comités au fur et à mesure du TP

Les fichier à rendre vont nous servir à savoir si vous avez compris, n'hésitez pas à y ajouter des explications.

1 Git

Git est un gestionnaire de version décentralisé le plus populaire actuellement. Git trouve ses origines dans le développement du noyau Linux et a été fondé par Linus Torvalds. Il est aujourd'hui utilisé par un grand nombre de projet open source, comme par exemple Android, Eclipse . . ..

Pour illustrer l'utilisation du gestionnaire de version git, nous allons utiliser le serveur *github.com*. Il est tout à fait possible d'utiliser d'autre serveur git comme par exemple *gitlab.org*, *framagit.org* ou en auto-hébergement. Git en soit n'a pas besoin de serveur pour fonctionner, l'idée est de gérer l'évolution de fichiers et répertoires. Le serveur permet de pouvoir échanger facilement les fichiers gérés entre plusieurs ordinateurs.

Préparation : Créer un compte sur GitHub.com

— Créez un compte sur `https://github.com`

— Vous allez recevoir une invitation *github/classroom* pour ce TP.

Préparation : Créer un répertoire de travail

— Créer un répertoire *TP2* dans votre répertoire de travail habituel.

— Pour chaque exercices créer un répertoire différent si besoin.

— Les réponses pour les exercices sur git sont à écrire dans un fichier `git.txt`

Exercice 1 Travailler seul.e avec Git (TP2/ex1)

1. Nous allons commencer par configurer Git en vous identifiant. Cela n'est à faire qu'une seule fois.

```
git config  --global user.name "votre_login"
git config  --global user.email  "votre_mail"
git config  --global core.editor emacs # ou votre éditeur préféré
git config  --global color.ui  auto
git config  --global alias.lola 'log --graph --decorate --oneline --all'
git config  --global alias.lop 'log --pretty=format:"%h - %an, %ar : %s"'
```

2. Créer un répertoire de dépot git, c'est à dire créer un répertoire normalement, puis s'y déplacer et enfin utiliser la commande

```
git init
```

qui va donc initialiser le répertoire courant comme étant maintenant sous le contrôle d'un gestionnaire de version.
Quels répertoires ont été créés ?

3. Créer un fichier dans ce répertoire.
4. Tapez la commande

```
git status
```

et comprendre le message.

5. Tapez la commande

```
git add votrefichier
```

, quel est maintenant le statut de votre fichier ?

6. Vous pouvez maintenant commiter vos changements

```
git commit -m"Un superMessage"
```

7. Quel est maintenant le statut de votre répertoire ?
8. Expliquer maintenat le résultat de la commande

```
git log -p
```

9. Modifier maintenant votre fichier et comprendre le résulat de la commande

```
git diff
```

10. Vous allez maintenant lié votre dépôt local à un dépôt sur le serveur `https://github.com`.
    - Votre espace se trouve à l'adresse suivante `https://epudx.polytech.upmc.fr/git/epu-opi-ex1-votrelogin` où `votrelogin` est votre login.
    - Dans votre répertoire appliquer la procédure suivante :

```
git remote add origin https://github.com/polytech-sorbonne-opi-2020/tp2-ex1-votrelogin
git push -u origin main
```

11. Que voyez-vous dans votre dépôt sur le serveur github `https://github.com/polytech-sorbonne-opi-2020/tp2-ex1-votrelogin` ?
12. Procédure pour mettre à jour les modifications sur le serveur :
    - Modifier votre fichier.
    - Ajouter le fichier pour mettre à jour ce qui sera commiter

```
git add filename
```

— Commiter votre fichier

```
git commit -m"Un message"
```

— Pusher les commits

```
git push
```

Vérifier que vos commits sont bien arrivés sur le serveur.

Exercice 2 Travailler à plusieurs (TP2/ex2)

1. Aller dans un autre répertoire que la question précédente.
2. Clonez le répertoire `https://github.com/Polytech-Sorbonne-OPI-2020/TP2-EX2`

```
$ git clone https://github.com/Polytech-Sorbonne-OPI-2020/TP2-EX2
```

   Quels répertoires ont été créés ?
3. Déplacez vous dans le répertoire versionné, que retourne la commande **`git`** `status`.
4. Créer un fichier `votre_nom.txt` où `votre_nom` est votre nom ou votre login.
5. Ajouter votre fichier sur le dépôt distant (voir la procédure question précédente).
6. Que se passe-t-il lors du push ? (si vous n'arrivez pas à pusher, envoyer un mail à `cecile.braunstein@lip6.fr` en indiquant votre login github).
7. Mettre à jour votre copie locale (le répertoire que vous avez cloner) avec la commande `git pull`
8. Réessayer de *pusher* (comme beaucoup d'élèves vont pusher en même temps il va falloir mettre à jour plusieurs fois votre copie locale avant de pouvoir pusher).
9. Modifier la première ligne de votre fichier et celle de votre voisin.
10. Commiter (*add/commit*) vos changements et envoyer les sur le serveur (*push*).
11. Que se passe-t-il ?
12. Régler le conflit sur votre fichier (choisissez la ou les modifications que vous voulez garder), commiter et pusher.
13. Regarder l'historique des changements avec la commande **`git`** `lola` et **`git`** `lop` ou l'interface graphique `gitg` ou sur redmine.

2 Fichiers

Dans un nouveau répertoire , clonez le répertoire `https://github.com/Polytech-Sorbonne-OPI-2020/TP2-EX3-votre\_login`.

Les réponses devront être écrite dans un fichier `fichier.txt`.

Exercice 3 Lecture de fichiers (`cat`, `head`, `tail`)

Placez-vous dans le répertoire que vous venez de cloner. Qu'affichent les commandes suivantes ?

```
cat essai1
cat -n essai1
head essai1
head -n 2 essai1
head -v essai1 essai2 essai3
tail essai1
tail -n 3 essai1
```

Exercice 4 Écriture de fichiers avec `cat`

La syntaxe **`cat`** `> nom_fichier` permet de créer le fichier `nom_fichier` (s'il n'existe pas déjà), et d'écrire dans ce fichier.

1. Placez-vous dans votre répertoire `TP2`, et tapez la commande **`cat`** `> tmp1.txt`. Vous pouvez maintenant écrire du texte, qui sera sauvegardé dans le fichier tmp1.txt. Entrez quelques lignes, puis appuyez sur `C-d` pour terminer.
2. Une des options très utiles de **`tail`** est l'option `-f` : on se sert en général de cette option pour examiner à la volée des fichiers de traces (`log`). Entrez plusieurs lignes de texte avec **`cat`** dans un fichier `tmp.txt`, et observez entre temps **dans un autre terminal** le résultat de `tail -f tmp.txt`.

Exercice 5 Recherche dans un fichier avec `grep`

1. Faites un alias de **`grep`** `--color` en **`grep`** dans votre fichier `.bashrc` (n'oublier pas de *sourcer* le fichier).
2. Afficher les lignes des films tournés dans le 3ème arrondissement.
3. Afficher les lignes avec leur numéro (option `-n`) des films tournés dans une impasse.
4. Combien de films ont été tournés dans une impasse ?
5. Afficher les lignes des films tournés dans une impasse ou dans le 11ème arrondissement.
6. Aller dans le répertoire `/usr/include/X11`, quels sont les fichiers qui ont besoin de la bibliothèque `stdio.h`
7. Avec l'option `-R` trouver aussi les fichiers plus bas dans l'arborescence.

Flux et pipe

Les réponses devront être écrite dans un fichier `flux_pipe.txt`.

Exercice 6 Redirections (>,>> et <)

1. Quel est le contenu des fichiers `file1` et `file2` après exécution des commandes suivantes :

```
echo "Redirection1" > file1
echo "Redirection2" >> file2
```

   Et après avoir relancé ces 2 commandes ?
2. Créez un fichier `noread` avec la commande `touch noread`. Enlevez les droits en lecture avec la commande `chmod -r noread`.
   Qu'affiche la commande

```
grep Redirection *
```

   et que se passe-t-il si on ajoute `> res1` à la fin ? Même question avec `2> /dev/null` à la fin.
3. Que contient le fichier `res` après l'exécution de la commande

```
grep Redirection *  > res 2>&1
```

4. Compiler le programme `moyenne.c` et essayer le.
   Préparer un fichier contenant un ensemble de note (1 note par ligne) et essayer le programme avec le fichier : `moyenne.exe < notes`.

Exercice 7 Pipes

1. Qu'affiche la commande **ls** essai* | **wc** -w ?
2. Comment afficher les 3 dernière commandes entrées sur le terminal ?
3. La commande ifconfig eth0 donne des informations sur l'interface réseau eth0 (qui correspond à ethernet). Voici un exemple d'exécution depuis une des machines des salles étudiants :

```
$ ifconfig
eth0      Link encap:Ethernet  HWaddr 14:b3:1f:1d:bc:5e
          inet addr:134.157.104.8  Bcast:134.157.105.255  Mask:255.255.254.0
          inet6 addr: fe80::16b3:1fff:fe1d:bc5e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:16209145 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1206538 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:24498902285 (24.4 GB)  TX bytes:85284319 (85.2 MB)
          Interrupt:20 Memory:f7200000-f7220000

...
```

   Afficher la ligne contenant l'adresse IP de l'interface eth0.
4. Comment obtenir automatiquement le fichier le plus gros du répertoire courant en une ligne de commande ?