EPU GB5 BIMB **Génie Logiciel et UML** septembre 2023

TD n^o 1 : Prise en main du gestionnaire de version `git`

La première étape consiste à installer `git` sur vos machines.

— sous Linux, les dépôts de paquets contiennent `git`, se laisser guider par votre gestionnaire de paquets.

— sous Windows, télécharger `git` en allant sur `https://gitforwindows.org/`.

Exercice 1 : (**Création d'un dépôt local `git`**)

1. Créer un répertoire.
2. S'y rendre.
3. Initialiser le dépôt.
4. Créer plusieurs petits fichiers.
5. Interroger l'état du dépôt.
6. Ajouter à l'index l'ensemble des fichiers.
7. Réinterroger l'état du dépot.
8. Valider les modifications. (suivant les architectures, il faudra peut-être définir les nom et email de l'utilisateur, voir 1 et 2 de l'exercice suivant)
9. Visualiser le journal des modifications.
10. Modifiez un de vos fichiers, validez les modifications.
11. Visualiser le journal des modifications.

Exercice 2 : (**Configuration**)

1. Donnez un nom d'utilisateur.
2. Donnez l'adresse mail de l'utilisateur.
3. Changez l'éditeur pour les `commit`.
4. Testez en mettant à jour un fichier, et en le validant.
5. La commande `git log` est assez verbeuse. Si on veut moins d'information, on peut taper la commande :
   ```
   git log --pretty='%h %ad | %s%d [%an]' --graph --date=short
   ```
   qui est difficile à retenir. On peut créer un alias, qu'on appellera `git hist` :
   ```
   git config --global alias.hist \
     "log --pretty=format:'%h %ad | %s%d [%an]' --graph --date=short"
   ```
   Testez la commande `git hist`.
6. Allez voir les modifications du fichier `.gitconfig` du `$HOME`.
7. Modifiez le message du dernier `commit`.

Exercice 3 : (**Suppression de fichiers**)

1. Créez un nouveau fichier. Ajoutez-le à l'index. Validez-le.
2. Supprimez le fichier créé. Examinez l'état du dépot. Que constatez-vous ?
3. Récupérez le fichier supprimé en suivant l'indication donnée par la commande `git status`.
4. Vérifiez l'état du dépôt ainsi que son contenu. Que constatez-vous ?
5. Utilisez la commande `git rm` afin de supprimer ce fichier.
6. Vérifiez l'état du dépôt ainsi que son contenu. Que constatez-vous ?
7. Enregistrez un instantané de votre dépôt. Indiquez un message bien explicite.

Exercice 4 : (**Les fichiers à ignorer**)

Certains fichiers ne doivent pas être indexés. Pour cela, on va spécifier les noms des fichiers à ignorer dans le fichier `.gitignore`.

1. Créez le fichier pour ignorer tous les fichiers dont le nom se termine par « ~ », ainsi que tous les fichiers du sous-répertoire `tests`.
2. Créez un répertoire `tests`, mettez-y un nouveau fichier, testez la validation des changements.

3. Modifiez éventuellement le fichier `.gitignore`.

Exercice 5 : (**Annuler des actions**)

1. Pour compléter l'index (en cas d'oubli d'un fichier dans l'index par exemple), on peut utiliser la commande `git add` puis charger le nouveau fichier dans l'ancien `commit` sans créer un nouveau `commit`. On utilise la commande : `git commit --amend`.
   — Créez 2 fichiers `test1.txt` et `test2.txt`.
   Ajouter `test1.txt` à l'index et faites un `commit`.
   Vérifiez l'état du dépôt.
   — Ajoutez `test2.txt` à l'index.
   — Amendez le dernier `commit` pour prendre en compte le fichier `test2.txt`.
   Vérifiez l'état du dépôt et de l'historique.
2. Pour enlever un fichier de l'index :
   — Créez 2 fichiers `test3.txt` et `test4.txt`.
   Ajoutez les 2 fichiers à l'index.
   — Interrogez l'état du dépôt.
   — Utilisez `git reset HEAD test4.txt` pour enlever de l'index `test4.txt`.
   Vérifiez l'état du dépôt.
3. Pour annuler toutes les modifications locales faites à un fichier : `git checkout -- <nomfichier>` ou alors `git restore <nomfichier>`. Testez ces commandes sur un fichier créé puis modifié.
4. Annulez le dernier `commit`, pour cela :
   — Vérifiez l'état du dépôt et de l'historique.
   — Testez la commande `git revert HEAD`
   — Vérifiez l'état du dépôt et de l'historique.

Exercice 6 : (**Etiqueter une version stable / se déplacer dans l'historique**)

On peut mettre une étiquette sur un instantanné particulier. `git` utilise pour cela le mot `tag`.

1. Etiquetez la version courante grâce à la commande `git tag version-0.1`.
2. Listez l'historique.
3. Faites plusieurs modifications sur différents fichiers.
   Validez les modifications.
4. Revenez à la version « `version-0.1` » grâce à la commande `git checkout version-0.1`.
5. Revenez à la dernière version grâce à la commande `git checkout master`.
6. Faites de même avec une version non étiquetée.