TP GIT

Règle n°1 : Vous n'êtes pas obligé de suivre le TP à la lettre
Règle n°2 : Testez des trucs
Règle n°3 : Posez-moi des questions (je suis là pour ça)

1. Configurer GIT

1. Regarder la configuration actuelle avec `git config --list`
2. Modifier en global la configuration du :
    a. `name`
    b. `email`
    c. `push.default` à `simple` (demandez-moi pourquoi ;))

2. Initialiser un dépôt

1. Créer un dossier et se placer dedans
2. Initialiser un dépôt GIT
3. Créer un premier commit VIDE (attention il y a une option à utiliser dont je vous ai pas parlé !)
4. Demander au prof pourquoi :)

3. 1er commit

1. Pour pouvoir faire un commit, il faut des modifications de fichiers, ajoutez quelques fichiers dans votre dossiers et écrire des trucs dedans (vous pouvez récupérer des fichiers avec du code)
2. Regardez le `status` de votre dépôt local
3. Ajoutez les fichiers à votre espace de staging
4. Regardez le `status` de votre dépôt local
5. Faites un commit
6. Regardez le `status` de votre dépôt local
7. Regardez l'historique de votre dépôt (`git log --decorate --all --oneline --graph`)

Vous devriez avoir quelque-chose comme ça

```
* 4fb59b7 (HEAD -> master) 1er commit
* f049158 Initial commit
(END)
```

4. Créer des branches

1. Créer une branche (`branch1`)
2. Se positionner sur la branche
3. Faire des modifications dans votre dépôt (Ajouter un fichier par exemple)
4. Commiter ces modifications
5. Regardez l'historique de votre dépôt

Vous devriez avoir quelque-chose comme ça

```
* a830285 (HEAD -> branch1) 2eme commit
* 4fb59b7 (master) 1er commit
* f049158 Initial commit
(END)
```

5. Merger avec fast-forward

1. Se positionner sur `master`
2. Merger votre `branch1` dans `master`
3. Vérifier dans la trace de la commande qu'il y a bien eu un `fast-forward`
4. Regardez l'historique de votre dépôt

Vous devriez avoir quelque-chose comme ça

```
* a830285 (HEAD -> master, branch1) 2eme commit
* 4fb59b7 1er commit
* f049158 Initial commit
(END)
```

6. Merger avec un commit de merge

Le but est de merger 2 branche sur lesquelles il y a eu des commit et voir le commit de merge

1. Se positionner sur `master`
2. Créer une nouvelle branche (`branch2`) et se positionner dessus
3. Faire des modifications dans votre dépôt et les commiter
4. Se positionner sur `master`
5. Faire des modifications dans votre dépôt et les commiter
6. Merger votre `branch2` dans `master`
7. Regardez l'historique de votre dépôt

Vous devriez avoir quelque-chose comme ça

```
*   9e73cf0 (HEAD -> master) Merge branch 'branch2'
|\
| * 1207794 (branch2) 3eme commit
* | 3a3cd41 4eme commit
|/
* a830285 (branch1) 2eme commit
* 4fb59b7 1er commit
* f049158 Initial commit
(END)
```

7. Se déplacer dans le graphe

Le but est de bouger dans le graphe et voir un peu ce qui se passe

1. Se déplacer entre les branches `branch1`, `branch2` (et autres)
2. Constater les changements dans l'espace de travail
3. Kiffer

Quand on en est là, on gère bien la base déjà !

8. Comparer

Le but est d'arriver à voir les différences entre l'espace de travail, le staging, des branches…

1. Faire des modifications dans votre espace de travail (ajouter des lignes dans un fichier)
2. Afficher les modifications faites :
3. Mettre les modifications dans le staging
4. Comparer les modifications qui sont en staging avec celles commitées (même résultat que précédemment)
5. Commiter
6. Regarder le contenu du dernier commit (même résultat que précédemment)
7. Comparer une branche avec une autre

9. Tagger

Le but est de créer un tag (qui restera fixe contrairement à une branche)

1. Créer un tag `V1.0`
2. Faire des modifications et les commiter
3. Regarder les logs pour voir où est le tag

4. Revenir sur le tag et constater l'état du dépôt

10. Rebase de branche

1. Se positionner sur `master`
2. Créer une nouvelle branche `branch3`
3. Faire des modifications et commiter
4. Revenir sur `master`
5. Faire des modifications et commiter
6. Etat Attendu :

7. Revenir sur `branch3`

8. Rebaser `branch3` sur `master`

11. Gérer les conflits lors d'un merge

Le but et de voir le comportement de Git en cas de conflit

1. Se positionner sur `master`
2. Créer une nouvelle branche `branch4`
3. Modifier une ligne d'un fichier existant et commiter
4. Revenir sur `master`
5. Modifier la même ligne du même fichier et commiter
6. Merger `branch4` sur `master`
7. Résoudre le conflit

12. Ajouter une remote

1. Créer un compte Github/Gitlab/Bitbucket (ou prendre son compte à soi)
2. Créer un dépôt sur Github/Gitlab/Bitbucket
3. Ajouter le dépôt Github/Gitlab/Bitbucket en tant que remote
4. Pousser les modifications locales sur github
5. Naviguer dans github
6. Ajouter un commit depuis l'interface Github
7. Récupérer le commit en local
8. Cloner le dépôt Github dans un autre dossier

13. Annuler le dernier

1. Faire plusieurs commit et pousser sur le dépôt distant
2. Annuler le dernier commit en utilisant `git reset [....]`
3. Pousser sur le dépôt distant (attention, y'a un piège !)

14. Nettoyer le graph

Nettoyer, balayer, astiquer !C'est important de garder un bel historique !

1. Squasher des commits
    a. Faire plein de commits !
    b. Utiliser un rebase interactif pour squasher les commits (avec `squash` ou `fixup`)
    c. Vérifier que l'on a plus qu'un seul commit
    d. Pousser !

2. Déplacer des commits
   Lors d'un rebase interactif, il est possible aussi de changer l'ordre des commits
   Faites plusieurs commits et modifiez l'ordre (il est très probable d'avoir des conflits ! se reporter à l'étape suivante)
   Pousser !

3. Gérer des conflits
   Si vous avez pas eu de conflits à l'étape précédente, on va déplacer des commit en veillant à modifier les mêmes lignes dans les fichiers
   On résout les conflits et on continue avec `git rebase --continue`
   On peut aussi annuler le rebase avec `git rebase --abort`
   Pousser !
4. Supprimer des commits
   Pareil mais cette fois, on efface un commit avec `drop`

15. Faire une Pull Request

1. Faire une branche et la pousser sur le dépôt distant
2. Créer une pull request sur `master`
3. Commenter
4. Merger
5. Tirer (aka `pull`) les modifications

16. Encore des trucs à tester en vrac !

1. Ajouter un fichier, l'ignorer et voir comment se comporte le dépôt
2. Essayer de commiter un dossier **VIDE**
3. Faire un `cherry-pick`