

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 mettez des conneries dedans
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 repositionner 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 repositionner sur `master`
2. Créer une nouvelle branche (`branch2`) et se positionner dessus
3. Faire des modifications dans votre dépôt et les commiter
4. Se repositionner 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

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 :

```
diff --git a/fichier1 b/fichier1
index 071a87e..302855b 100644
--- a/fichier1
+++ b/fichier1
@@ -1,4 +1,7 @@
    bla
    bla
    bla
+  plop
+  plop
+  plop
(END)
```

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

```
diff --git a/fichier1 b/fichier1
index 302855b..071a87e 100644
--- a/fichier1
+++ b/fichier1
@@ -1,7 +1,4 @@
  bla
  bla
  bla
- plop
- plop
- plop

diff --git a/fichier3 b/fichier3
deleted file mode 100644
index b3405a1..0000000
--- a/fichier3
+++ /dev/null
@@ -1,4 +0,0 @@
- zoeihf
- zefz
- efze
- f

diff --git a/fichier4 b/fichier4
deleted file mode 100644
index e69de29..0000000
(END)
```

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

```
* 9f9fc5b (HEAD -> master) 6eme commit
* e7d3df3 (tag: v1.0) 5eme commit
* 9e73cf0 Merge branch 'branch2'
| \
| * 1207794 (branch2) 3eme commit
* | 3a3cd41 4eme commit
| /
* a830285 (branch1) 2eme commit
* 4fb59b7 1er commit
* f049158 Initial commit
(END)
```

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

10. Rebase

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 :

```
* 700c63c (HEAD -> master) 8eme commit
| * 3bd63f3 (branch3) 7eme Commit
| /
* 9f9fc5b 6eme commit
* e7d3df3 (tag: v1.0) 5eme commit
* 9e73cf0 Merge branch 'branch2'
| \
| * 1207794 (branch2) 3eme commit
* | 3a3cd41 4eme commit
```

7. Revenir sur `branch3`

8. Rebaser `branch3` sur `master`

```
* 6c7804f (HEAD -> branch3) 7eme Commit
* 700c63c (master) 8eme commit
* 9f9fc5b 6eme commit
* e7d3df3 (tag: v1.0) 5eme commit
* 9e73cf0 Merge branch 'branch2'
| \
| * 1207794 (branch2) 3eme commit
* | 3a3cd41 4eme commit
| /
```

11. Ajouter une remote

1. Créer un compte Github/Bitbucket (ou prendre son compte à soi)
2. Créer un dépôt sur Github/Bitbucket
3. Ajouter le dépôt Github/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

12. Gérer les conflits

Le but est 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
8. Faire la même chose avec un rebase au lieu d'un merge^