

TP1 – Git

Brahim HAMDI

Initialisation et premier commit

1. Initialiser un nouveau dépôt Git sous le répertoire *tpgit*, puis entrer dans ce répertoire de travail :

```
brahim@Training:~/TP1$ git init tpgit
astuce: Utilisation de 'master' comme nom de la branche initiale. Le nom de la branche
astuce: par défaut peut changer. Pour configurer le nom de la branche initiale
astuce: pour tous les nouveaux dépôts, et supprimer cet avertissement, lancez :
astuce:
astuce:     git config --global init.defaultBranch <nom>
astuce:
astuce: Les noms les plus utilisés à la place de 'master' sont 'main', 'trunk' et
astuce: 'development'. La branche nouvellement créée peut être renommée avec :
astuce:
astuce:     git branch -m <nom>
Dépôt Git vide initialisé dans /home/brahim/TP1/tpgit/.git/
brahim@Training:~/TP1$ cd tpgit
brahim@Training:~/TP1/tpgit$ 
brahim@Training:~/TP1/tpgit$ 
```

2. Il est important (surtout en équipe) de savoir qui a fait quoi. Pour identifier vos futurs commits, ajouter les paramètres de configuration suivants :

- Votre Nom
- Votre Email
- Votre éditeur de texte préféré (nano par exemple).

```
brahim@Training:~/TP1/tpgit$ git config --global user.name "Brahim HAMDI"
brahim@Training:~/TP1/tpgit$ git config --global user.email "brahim.hamdi.consult@gmail.com"
brahim@Training:~/TP1/tpgit$ 
brahim@Training:~/TP1/tpgit$ git config --global core.editor nano
brahim@Training:~/TP1/tpgit$ 
brahim@Training:~/TP1/tpgit$ git config --list --global
user.name=Brahim HAMDI
user.email=brahim.hamdi.consult@gmail.com
core.editor=nano
brahim@Training:~/TP1/tpgit$ 
brahim@Training:~/TP1/tpgit$ 
```

3. P
o
u
r

votre premier commit, vous allez créer un fichier README.md à la racine avec le contenu de votre choix.

```
brahim@Training:~/TP1/tpgit$ ls README.md
README.md
brahim@Training:~/TP1/tpgit$ cat README.md
Doc sur la demo GIT
brahim@Training:~/TP1/tpgit$ 
brahim@Training:~/TP1/tpgit$ 
```

- Regarder l'état du dépôt.

```

brahim@Training:~/TP1/tppgit$ git status
Sur la branche master

Aucun commit

Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)
    README.md

aucune modification ajoutée à la validation mais des fichiers non suivis sont présents (utilisez "git add" pour les suivre)
brahim@Training:~/TP1/tppgit$
```

4. Mettre le fichier sans l'index, puis vérifier son état de nouveau

```

brahim@Training:~/TP1/tppgit$ git add README.md
brahim@Training:~/TP1/tppgit$ git status
Sur la branche master

Aucun commit

Modifications qui seront validées :
  (utilisez "git rm --cached <fichier>..." pour désindexer)
    nouveau fichier : README.md

brahim@Training:~/TP1/tppgit$
```

5. Pour valider le nouveau fichier et, ainsi, créer votre premier commit:

```

brahim@Training:~/TP1/tppgit$ git commit -m "initialisation du projet"
[master (commit racine) bbca000] initialisation du projet
  1 file changed, 1 insertion(+)
   create mode 100644 README.md
brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
rien à valider, la copie de travail est propre
brahim@Training:~/TP1/tppgit$
brahim@Training:~/TP1/tppgit$
```

Trois états du fichier

6. Créez un nouveau fichier hello.py avec comme contenu:

print "Hello world!"	brahim@Training:~/TP1/tppgit\$ ls hello.py README.md brahim@Training:~/TP1/tppgit\$ cat hello.py print "Hello world!" brahim@Training:~/TP1/tppgit\$ brahim@Training:~/TP1/tppgit\$
----------------------	--

- Ajouter ce fichier dans l'index puis commitez-le.

```

brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)
    hello.py

aucune modification ajoutée à la validation mais des fichiers non suivis sont présents (utilisez "git add" pour les suivre)
brahim@Training:~/TP1/tppgit$ git add hello.py
brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
Modifications qui seront validées :
  (utilisez "git restore --staged <fichier>..." pour désindexer)
    nouveau fichier : hello.py

brahim@Training:~/TP1/tppgit$ git commit -m "Add hello.py"
[master 63c5fef] Add hello.py
  1 file changed, 1 insertion(+)
   create mode 100644 hello.py
brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
rien à valider, la copie de travail est propre
brahim@Training:~/TP1/tppgit$
```

7. Ajouter la fonction 'ecrire()', pour obtenir le contenu suivant :

```
def ecrire(chaine):
    print chaine
print "Hello world!"
```

```
brahim@Training:~/TP1/tppgit$ ls
hello.py  README.md
brahim@Training:~/TP1/tppgit$ cat hello.py
def ecrire(chaine):
    print chaine

    print("Hello world!")
brahim@Training:~/TP1/tppgit$
```

- Regardez l'état du dépôt:

```
brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
Modifications qui ne seront pas validées :
  (utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git restore <fichier>..." pour annuler les modifications dans le répertoire de travail)
  modifié :          hello.py

aucune modification n'a été ajoutée à la validation (utilisez "git add" ou "git commit -a")
brahim@Training:~/TP1/tppgit$
```

- Visualisez les modification apportées avec la commande `git diff`:

```
brahim@Training:~/TP1/tppgit$ git diff
diff --git a/hello.py b/hello.py
index ed708ec..bdd3a36 100644
--- a/hello.py
+++ b/hello.py
@@ -1 +1,4 @@
-print "Hello world!"
+def ecrire(chaine):
+    print chaine
+
+print("Hello world!")
brahim@Training:~/TP1/tppgit$
```

- Ajoutez la modification à l'index sans la valider pour l'instant (pas de commit), puis visualisez la différence une autre fois.

```
brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
Modifications qui ne seront pas validées :
  (utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git restore <fichier>..." pour annuler les modifications dans le répertoire de travail)
  modifié :          hello.py

aucune modification n'a été ajoutée à la validation (utilisez "git add" ou "git commit -a")
brahim@Training:~/TP1/tppgit$ git add hello.py
brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
Modifications qui seront validées :
  (utilisez "git restore --staged <fichier>..." pour désindexer)
  modifié :          hello.py

brahim@Training:~/TP1/tppgit$ git diff
```

Modifier le programme pour utilisez la fonction `ecrire()` plutôt que `print()`, de la façon suivante :

```
def ecrire(chaine):
    print chaine
ecrire("Hello world!")
```

```
brahim@Training:~/TP1/tppgit$ ls
hello.py  README.md
brahim@Training:~/TP1/tppgit$ cat hello.py
def ecrire(chaine):
    print chaine

    ecrire("Hello world!")
brahim@Training:~/TP1/tppgit$
```

- Affichez l'état courant.

On

```

brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
Modifications qui seront validées :
  (utilisez "git restore --staged <fichier>..." pour désindexer)
    modifié :           hello.py

Modifications qui ne seront pas validées :
  (utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git restore <fichier>..." pour annuler les modifications dans le répertoire de travail)
    modifié :           hello.py

brahim@Training:~/TP1/tppgit$ -

```

remarque ici 2 modifications sur notre fichier : une modification indexée (l'ajout de la fonction) et une autre non (l'utilisation de la fonction)

- comparer l'espace de travail avec le dernier commit du dépôt (HEAD)

```

brahim@Training:~/TP1/tppgit$ git diff HEAD
diff --git a/hello.py b/hello.py
index ed708ec..320c425 100644
--- a/hello.py
+++ b/hello.py
@@ -1 +1,4 @@
-print "Hello world!"
+def ecrire(chaine):
+    print chaine
+
+ecrire("Hello world!")
brahim@Training:~/TP1/tppgit$ -

```

Renommage et suppression

9. Ajouter un fichier non vide **fichier1** et commitez-le

```

brahim@Training:~/TP1/tppgit$ touch fichier1
brahim@Training:~/TP1/tppgit$ ls
fichier1  hello.py  README.md
brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
Modifications qui seront validées :
  (utilisez "git restore --staged <fichier>..." pour désindexer)
    modifié :           hello.py

Modifications qui ne seront pas validées :
  (utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git restore <fichier>..." pour annuler les modifications dans le répertoire de travail)
    modifié :           hello.py

Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)
    fichier1

brahim@Training:~/TP1/tppgit$ git add fichier1
brahim@Training:~/TP1/tppgit$ git commit -m "Add fichier1"
[master 0c54fd2] Add fichier1
 2 files changed, 4 insertions(+), 1 deletion(-)
  create mode 100644 fichier1
brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
Modifications qui ne seront pas validées :
  (utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git restore <fichier>..." pour annuler les modifications dans le répertoire de travail)
    modifié :           hello.py

aucune modification n'a été ajoutée à la validation (utilisez "git add" ou "git commit -a")
brahim@Training:~/TP1/tppgit$ -

```

10. Obs

ervez ce que vous dit **git status** pour chacune des étapes suivantes :

```
mv fichier1 fichier2
```

```

git rm fichier1
git add fichier2

brahim@Training:~/TP1/tppgit$ mv fichier1 fichier2
brahim@Training:~/TP1/tppgit$ git status
brahim@Training:~/TP1/tppgit$ git add fichier2
brahim@Training:~/TP1/tppgit$ ls
fichier2 hello.py README.md
brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
Modifications qui seront validées :
  (utilisez "git restore --staged <fichier>..." pour désindexer)
    renommé :      fichier1 -> fichier2

Modifications qui ne seront pas validées :
  (utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git restore <fichier>..." pour annuler les modifications dans le répertoire de travail)
    modifié :      hello.py

brahim@Training:~/TP1/tppgit$
brahim@Training:~/TP1/tppgit$ 
Sur la branche master
Modifications qui seront validées :
  (utilisez "git restore --staged <fichier>..." pour désindexer)
    supprimé :      fichier1

Modifications qui ne seront pas validées :
  (utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git restore <fichier>..." pour annuler les modifications dans le répertoire de travail)
    modifié :      hello.py

Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)
    fichier2

brahim@Training:~/TP1/tppgit$ ls
fichier2 hello.py README.md
brahim@Training:~/TP1/tppgit$
```

Git détecte que les 2 fichiers (1 supprimé, 1 ajouté) ont le même contenu. Il en déduit que c'est un renommage. Le renommage peut se faire aussi plus directement avec `git mv`.

- Commitez le renommage

```

brahim@Training:~/TP1/tppgit$ git commit -m "renommage fichier1 --> fichier2"
[master 6cd1bd4] renommage fichier1 --> fichier2
1 file changed, 0 insertions(+), 0 deletions(-)
 rename fichier1 => fichier2 (100%)
brahim@Training:~/TP1/tppgit$ 
brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
Modifications qui ne seront pas validées :
  (utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git restore <fichier>..." pour annuler les modifications dans le répertoire de travail)
    modifié :      hello.py

aucune modification n'a été ajoutée à la validation (utilisez "git add" ou "git commit -a")
brahim@Training:~/TP1/tppgit$
```

Consulter l'historique

11. utiliser `git log` pour afficher l'historique des commits.

```

brahim@Training:~/TP1/tppgit$ git log
commit 6cd1bd4ea6ee132e0c6fdc67be40ce967bbcca78 (HEAD -> master)
Author: Brahim HAMDI <brahim.hamdi.consult@gmail.com>
Date:   Sun Jan 7 13:57:44 2024 +0100

  renommage fichier1 --> fichier2

commit 0c54fd22ea614b11eb14ccf2410c4708268fbceb
Author: Brahim HAMDI <brahim.hamdi.consult@gmail.com>
Date:   Sun Jan 7 13:38:58 2024 +0100

  Add fichier1

commit 63c5fef9cb29a044d70858bb14b485159460cc9
Author: Brahim HAMDI <brahim.hamdi.consult@gmail.com>
Date:   Sun Jan 7 13:13:28 2024 +0100

  Add hello.py

commit bbca00092d76105fe427053a7d88e45ebbae151d
Author: Brahim HAMDI <brahim.hamdi.consult@gmail.com>
Date:   Sun Jan 7 13:09:04 2024 +0100

  initialisation du projet
brahim@Training:~/TP1/tppgit$
```

- Pour un affichage plus concis, utiliser l'option `--oneline` :

```
brahim@Training:~/TP1/tppgit$ git log --oneline
6cd1bd4 (HEAD -> master) renommage fichier1 --> fichier2
0c54fd2 Add fichier1
63c5fef Add hello.py
bbc000 initialistion du projet
brahim@Training:~/TP1/tppgit$
```

12. Utiliser `git show` suivi de l'identifiant du dernier commit pour avoir plus de détail sur ce commit.

```
brahim@Training:~/TP1/tppgit$ git log --oneline
6cd1bd4 (HEAD -> master) renommage fichier1 --> fichier2
0c54fd2 Add fichier1
63c5fef Add hello.py
bbc000 initialistion du projet
brahim@Training:~/TP1/tppgit$
brahim@Training:~/TP1/tppgit$ git show 0c54fd2
commit 0c54fd22ea614b11eb14ccf2410c4708268fbceb
Author: Brahim HAMDI <brahim.hamdi.consult@gmail.com>
Date:   Sun Jan 7 13:38:58 2024 +0100

        Add fichier1

diff --git a/fichier1 b/fichier1
new file mode 100644
index 0000000..e69de29
diff --git a/hello.py b/hello.py
index ed708ec..bdd3a36 100644
--- a/hello.py
+++ b/hello.py
@@ -1 +1,4 @@
-print "Hello world!"
+def ecrire(chaine):
+    print chaine
+
+print("Hello world!")
brahim@Training:~/TP1/tppgit$ █
```

Ignorer des fichiers/répertoires

13. Afficher l'état de votre projet et valider tous les changements

```
brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
Modifications qui ne seront pas validées :
  (utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git restore <fichier>..." pour annuler les modifications dans le répertoire de travail)
    modifié :          hello.py

aucune modification n'a été ajoutée à la validation (utilisez "git add" ou "git commit -a")
brahim@Training:~/TP1/tppgit$ git commit -am "modification de hello.py"
[master 1d3431d] modification de hello.py
 1 file changed, 2 insertions(+), 2 deletions(-)
brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
rien à valider, la copie de travail est propre
brahim@Training:~/TP1/tppgit$
```

14. Compiler votre programme python avec la commande : python3 -c "import hello"

```
brahim@Training:~/TP1/tppgit$ python3 -c "import hello"
Hello world!
brahim@Training:~/TP1/tppgit$
```

- Regardez l'état de votre projet :

```
brahim@Training:~/TP1/tppgit$ python3 -c "import hello"
Hello world!
brahim@Training:~/TP1/tppgit$ ls
fichier2 hello.py __pycache__ README.md
brahim@Training:~/TP1/tppgit$ brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)
  __pycache__/
aucune modification ajoutée à la validation mais des fichiers non suivis sont présents (utilisez "git add" pour les suivre)
brahim@Training:~/TP1/tppgit$ brahim@Training:~/TP1/tppgit$ ls __pycache__/
hello.cpython-310.pyc
brahim@Training:~/TP1/tppgit$ █
```

Le dossier `__pycache__` (résultat de la compilation) contient un fichier binaire généré par la compilation `hello.cpython-310.pyc`. Vous n'avez aucun besoin de suivre l'état de ce fichier qui peut être généré à nouveau à chaque compilation.

D'autre part, comme tout VCS, Git n'est réellement adapté qu'au suivi de fichiers textes. Il ne faut donc intégrer `*.pyc` ni à l'index ni au dépôt.

Pour l'oublier une fois pour toute, vous pouvez l'ajouter à la liste des fichiers à ignorer par Git.

- Créer un fichier `.gitignore` à la racine de votre répertoire de travail, qui contiendra une liste de fichiers à ignorer (une ligne par fichier). Dans notre cas, mettez simplement la `*.pyc` pour ignorer les fichiers avec cette extension.

```
brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)
  __pycache__/
aucune modification ajoutée à la validation mais des fichiers non suivis sont présents (utilisez "git add" pour les suivre)
brahim@Training:~/TP1/tppgit$ brahim@Training:~/TP1/tppgit$ touch .gitignore
brahim@Training:~/TP1/tppgit$ brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)
  .gitignore
  __pycache__/
aucune modification ajoutée à la validation mais des fichiers non suivis sont présents (utilisez "git add" pour les suivre)
brahim@Training:~/TP1/tppgit$ brahim@Training:~/TP1/tppgit$ echo *.pyc >> .gitignore
brahim@Training:~/TP1/tppgit$ brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)
  .gitignore
aucune modification ajoutée à la validation mais des fichiers non suivis sont présents (utilisez "git add" pour les suivre)
brahim@Training:~/TP1/tppgit$ █
```

15. Une autre solution qui consiste à mettre le nom du repertoires. Remplacez `*.pyc` par `__pycache__` dans le fichier `.gitignore`

```

brahim@Training:~/TP1/tppgit$ cat .gitignore
*.pyc
brahim@Training:~/TP1/tppgit$ echo __pycache__ > .gitignore
brahim@Training:~/TP1/tppgit$ cat .gitignore
__pycache__
brahim@Training:~/TP1/tppgit$
brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)
.gitignore

aucune modification ajoutée à la validation mais des fichiers non suivis sont présents (utilisez "git add" pour les suivre)
brahim@Training:~/TP1/tppgit$ -

```

16. Valider la création de .gitignore

```

brahim@Training:~/TP1/tppgit$ git add .gitignore
brahim@Training:~/TP1/tppgit$ git commit -m "ajout de .gitignore"
[master 79a7550] ajout de .gitignore
 1 file changed, 1 insertion(+)
 create mode 100644 .gitignore
brahim@Training:~/TP1/tppgit$
brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
rien à valider, la copie de travail est propre
brahim@Training:~/TP1/tppgit$ 

```

Revenir en arrière

17. Supprimer le fichier hello.py en utilisant la commande `git rm`. Observez le status du projet après la suppression.

```

brahim@Training:~/TP1/tppgit$ ls
fichier2 hello.py __pycache__ README.md
brahim@Training:~/TP1/tppgit$ brahim@Training:~/TP1/tppgit$ git rm hello.py
rm 'hello.py'
brahim@Training:~/TP1/tppgit$ brahim@Training:~/TP1/tppgit$ ls
fichier2 __pycache__ README.md
brahim@Training:~/TP1/tppgit$ brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
Modifications qui seront validées :
  (utilisez "git restore --staged <fichier>..." pour désindexer)
    supprimé :          hello.py
brahim@Training:~/TP1/tppgit$ 

```

- Committe
z cette

suppression.

```

brahim@Training:~/TP1/tppgit$ git commit -m "Suppression de hello.py"
[master 1070124] Suppression de hello.py
 1 file changed, 4 deletions(-)
 delete mode 100644 hello.py
brahim@Training:~/TP1/tppgit$ brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
rien à valider, la copie de travail est propre
brahim@Training:~/TP1/tppgit$ brahim@Training:~/TP1/tppgit$ ls
fichier2 __pycache__ README.md
brahim@Training:~/TP1/tppgit$ brahim@Training:~/TP1/tppgit$ 

```

- Modifiez le fichier README.md pour expliquer la suppression et commitez le.

```

brahim@Training:~/TP1/tppgit$ ls
fichier2 __pycache__ README.md
brahim@Training:~/TP1/tppgit$ cat README.md
Doc sur la demo GIT

On va faire mieux!
On procède à une refonte totale du projet.

brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
Modifications qui ne seront pas validées :
  (utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git restore <fichier>..." pour annuler les modifications dans le répertoire de travail)
    modifié :          README.md

aucune modification n'a été ajoutée à la validation (utilisez "git add" ou "git commit -a")
brahim@Training:~/TP1/tppgit$ git commit -am "Explication suppression de hello.py sur README.md"
[master 501fefc] Explication suppression de hello.py sur README.md
  1 file changed, 5 insertions(+)
brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
rien à valider, la copie de travail est propre
brahim@Training:~/TP1/tppgit$ █

```

l'historique des commits. Notez l'id du commit la suppression de hello.py

```

brahim@Training:~/TP1/tppgit$ git log --oneline
501fefc (HEAD -> master) Explication suppression de hello.py sur README.md
1070124 Suppression de hello.py
79a7550 ajout de .gitignore
1d3431d modification de hello.py
6cd1bd4 renommage fichier1 --> fichier2
0c54fd2 Add fichier1
63c5fef Add hello.py
bbc0000 initialisation du projet
brahim@Training:~/TP1/tppgit$ █

```

- Récupérer le fichier hello.py à partir du commit noté précédemment, en utilisant la commande `git checkout`

```

brahim@Training:~/TP1/tppgit$ git checkout 79a7550 hello.py
1 chemin mis à jour depuis 904346f
brahim@Training:~/TP1/tppgit$ brahim@Training:~/TP1/tppgit$ ls
fichier2 hello.py __pycache__ README.md
brahim@Training:~/TP1/tppgit$ brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
Modifications qui seront validées :
  (utilisez "git restore --staged <fichier>..." pour désindexer)
    nouveau fichier : hello.py
brahim@Training:~/TP1/tppgit$ █

```

- Valider ce changement et visualiser l'historique des commits

```

brahim@Training:~/TP1/tppgit$ git commit -m "Récupération du fichier hello.py après suppression"
[master 0064c70] Récupération du fichier hello.py après suppression
  1 file changed, 4 insertions(+)
  create mode 100644 hello.py
brahim@Training:~/TP1/tppgit$ cat hello.py
def ecrire(chaine):
    print (chaine)

ecrire("Hello world!")
brahim@Training:~/TP1/tppgit$ git log --oneline
0064c70 (HEAD -> master) Récupération du fichier hello.py après suppression
501fefc Explication suppression de hello.py sur README.md
1070124 Suppression de hello.py
79a7550 ajout de .gitignore
1d3431d modification de hello.py
6cd1bd4 renommage fichier1 --> fichier2
0c54fd2 Add fichier1
63c5fef Add hello.py
bbc0000 initialisation du projet
brahim@Training:~/TP1/tppgit$ █

```

Vous devez récupérer l'ancien contenu du fichier REAME.md aussi.

- On va utiliser une autre solution avec la commande `git revert`.

Afficher le contenu actuel du fichier README.md

- Chercher dans l'historique le bon commit à « reverter » pour récupérer l'ancien contenu de README.md

```
brahim@Training:~/TP1/tppgit$ git log --oneline
0064c70 (HEAD -> master) Récupération du fichier hello.py après suppression
501fefc Explication suppression de hello.py sur README.md
1070124 Suppression de hello.py
79a7550 ajout de .gitignore
1d3431d modification de hello.py
6cd1bd4 renommage fichier1 --> fichier2
0c54fd2 Add fichier1
63c5fef Add hello.py
bbca000 initialistion du projet
brahim@Training:~/TP1/tppgit$ █
```

o L
a
c
e
r

le « revert » du commit, et vérifier de nouveau le contenu de README.md

```
brahim@Training:~/TP1/tppgit$ git revert 501fefc
[master af94cd4] Revert "Explication suppression de hello.py sur README.md"
 1 file changed, 5 deletions(-)
brahim@Training:~/TP1/tppgit$ 
brahim@Training:~/TP1/tppgit$ cat README.md
Doc sur la demo GIT
brahim@Training:~/TP1/tppgit$ git log --oneline
af94cd4 (HEAD -> master) Revert "Explication suppression de hello.py sur README.md"
0064c70 Récupération du fichier hello.py après suppression
501fefc Explication suppression de hello.py sur README.md
1070124 Suppression de hello.py
79a7550 ajout de .gitignore
1d3431d modification de hello.py
6cd1bd4 renommage fichier1 --> fichier2
0c54fd2 Add fichier1
63c5fef Add hello.py
bbca000 initialistion du projet
brahim@Training:~/TP1/tppgit$ █
```

19. Supprimer le fichier fichier2 sans valider par un commit

```
brahim@Training:~/TP1/tppgit$ git rm fichier2
rm 'fichier2'
brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
Modifications qui seront validées :
  (utilisez "git restore --staged <fichier>..." pour désindexer)
    supprimé :      fichier2
brahim@Training:~/TP1/tppgit$ █
```

o Désindex

er la suppression de fichier2

```
brahim@Training:~/TP1/tppgit$ ls
hello.py  __pycache__  README.md
brahim@Training:~/TP1/tppgit$ git restore --staged fichier2
brahim@Training:~/TP1/tppgit$ git restore --staged fichier2
brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
Modifications qui ne seront pas validées :
  (utilisez "git add/rm <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git restore <fichier>..." pour annuler les modifications dans le répertoire de travail)
    supprimé :      fichier2
aucune modification n'a été ajoutée à la validation (utilisez "git add" ou "git commit -a")
brahim@Training:~/TP1/tppgit$ █
```

Le

fichier n'existe pas dans le répertoire de travail bien que la désindexation a passé avec succès. Il faut le récupérer à partir de la base de Git.

- Utiliser la commande `git checkout` pour copier fichier2 à partir du dernier commit.

```

brahim@Training:~/TP1/tppgit$ git checkout HEAD fichier2
1 chemin mis à jour depuis 904346f
brahim@Training:~/TP1/tppgit$ ls
fichier2 hello.py __pycache__ README.md
brahim@Training:~/TP1/tppgit$ $
$ : commande introuvable
brahim@Training:~/TP1/tppgit$ git status
Sur la branche master
rien à valider, la copie de travail est propre
brahim@Training:~/TP1/tppgit$
brahim@Training:~/TP1/tppgit$ git log --oneline
af94cd4 (HEAD -> master) Revert "Explication suppression de hello.py sur README.md"
0064c70 Récupération du fichier hello.py après suppression
501fefc Explication suppression de hello.py sur README.md
1070124 Suppression de hello.py
79a7550 ajout de .gitignore
1d3431d modification de hello.py
6cd1bd4 renommage fichier1 --> fichier2
0c54fd2 Add fichier1
63c5fef Add hello.py
bbca000 initialistion du projet
brahim@Training:~/TP1/tppgit$ 

```

20. Dans le dernier commit (celui de revert), on a accepté le message par défaut alors qu'on a du mettre un message plus clair.

Amender le dernier commit en modifiant le message comme suit : « *Annulation de la suppression du fichier hello.py* »

```

brahim@Training:~/TP1/tppgit$ git log --oneline
af94cd4 (HEAD -> master) Revert "Explication suppression de hello.py sur README.md"
0064c70 Récupération du fichier hello.py après suppression
501fefc Explication suppression de hello.py sur README.md
1070124 Suppression de hello.py
79a7550 ajout de .gitignore
1d3431d modification de hello.py
6cd1bd4 renommage fichier1 --> fichier2
0c54fd2 Add fichier1
63c5fef Add hello.py
bbca000 initialistion du projet
brahim@Training:~/TP1/tppgit$ 
brahim@Training:~/TP1/tppgit$ git commit --amend
[master 9082484] Annulation de la suppression du fichier hello.py
  Date: Sun Jan 7 15:16:48 2024 +0100
  1 file changed, 5 deletions(-)

```

message a été bien modifié

21. Vérifier que le

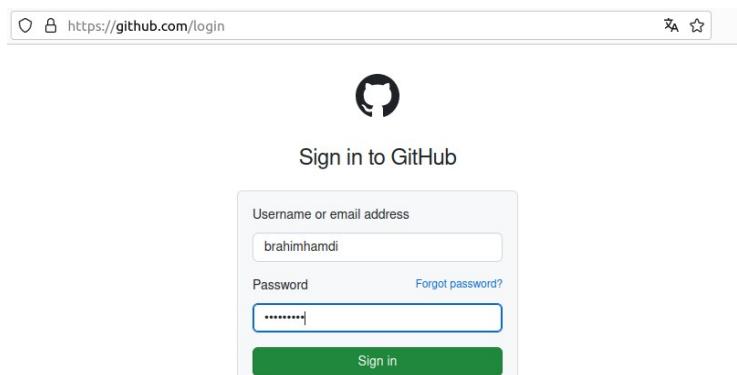
```

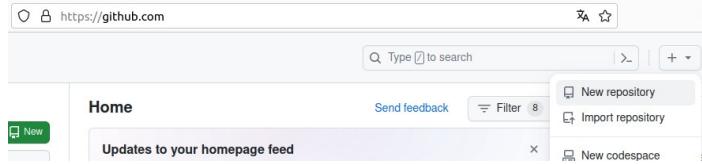
brahim@Training:~/TP1/tppgit$ git log --oneline
9082484 (HEAD -> master) Annulation de la suppression du fichier hello.py
0064c70 Récupération du fichier hello.py après suppression
501fefc Explication suppression de hello.py sur README.md
1070124 Suppression de hello.py
79a7550 ajout de .gitignore
1d3431d modification de hello.py
6cd1bd4 renommage fichier1 --> fichier2
0c54fd2 Add fichier1
63c5fef Add hello.py
bbca000 initialistion du projet
brahim@Training:~/TP1/tppgit$ 

```

Utiliser un dépôt distant

22. Vous allez d'abord créer un dépôt distant en utilisant le serveur GitHub.com





Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * Repository name *

brahimhamdi / tpgit
tpgit is available.

Great repository names are short and memorable. Need inspiration? How about [probable-eureka](#) ?

Description (optional)

Public

Anyone on the internet can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

ⓘ You are creating a public repository in your personal account.

[Create repository](#)

23. Par la suite,

copier l'url du dépôt distant.

Quick setup — if you've done this kind of thing before

or [HTTPS](#) [SSH](#) <https://github.com/brahimhamdi/tpgit.git>

- Il faut indiquer à Git de créer un *lien* du dépôt local vers le dépôt géré par github.com.

```
brahim@Training:~/TP1/tpgit$ git remote add origin https://github.com/brahimhamdi/tpgit.git
brahim@Training:~/TP1/tpgit$ git remote -v
origin https://github.com/brahimhamdi/tpgit.git (fetch)
origin https://github.com/brahimhamdi/tpgit.git (push)
brahim@Training:~/TP1/tpgit$
```

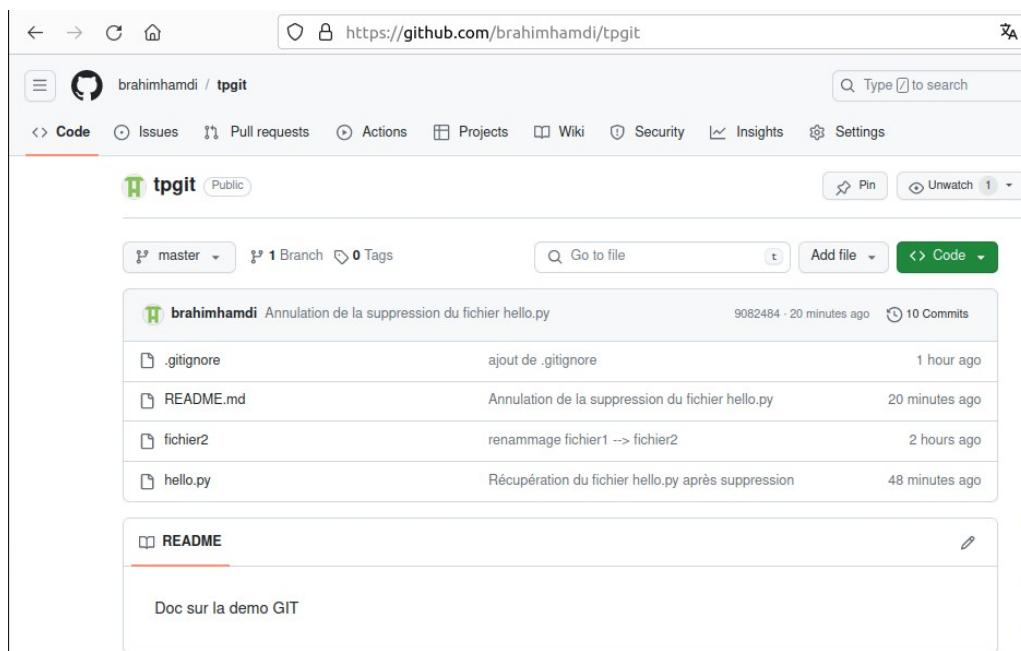
Le nom 'origin' est celui donné usuellement, vous pouvez en choisir un autre si vraiment vous le désirez.

La commande `git remove -v` est utilisée pour vérifier les liens distants enregistrés.

24. Envoyer la version actuelle de votre historique sur le dépôt distant

```
brahim@Training:~/TP1/tpgit$ git push -u origin master
Username for 'https://github.com': brahimhamdi
Password for 'https://brahimhamdi@github.com':
Énumération des objets: 26, fait.
Décompte des objets: 100% (26/26), fait.
Compression par delta en utilisant jusqu'à 8 fils d'exécution
Compression des objets: 100% (21/21), fait.
Écriture des objets: 100% (26/26), 2.44 Kio | 831.00 Kio/s, fait.
Total 26 (delta 7), réutilisés 0 (delta 0), réutilisés du pack 0
remote: Resolving deltas: 100% (7/7), done.
To https://github.com/brahimhamdi/tpgit.git
 * [new branch]      master -> master
La branche 'master' est paramétrée pour suivre la branche distante 'master' depuis 'origin'.
brahim@Training:~/TP1/tpgit$
```

- Une fois cette commande terminée, vous pouvez vérifier avec l'interface Web du serveur github que votre projet contient bien votre historique.



Nous allons *mettre de côté* le répertoire 'tpgit' pour la suite, en le renommant

```
brahim@Training:~/TP1/tpgit$ cd ..
brahim@Training:~/TP1$ mv tpgit tpgit-local
brahim@Training:~/TP1$ ls
tpgit-local
brahim@Training:~/TP1$
```

Maintenant que votre dépôt est enregistré sur le serveur, il est possible à tous les membres de votre projet d'en créer une copie locale, un *clone*.

- Imaginons que vous êtes un nouveau collaborateur au projet. Vous devez récupérer le dépôt hébergé sur le serveur.

Retournez dans votre répertoire racine, et utilisez une commande `git clone` pour récupérer tous le dépôt distant

```
brahim@Training:~/TP1$ git clone https://github.com/brahimhamdi/tpgit tpgit-remote
Clonage dans 'tpgit-remote'...
remote: Enumerating objects: 26, done.
remote: Counting objects: 100% (26/26), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 26 (delta 7), reused 26 (delta 7), pack-reused 0
Réception d'objets: 100% (26/26), fait.
Résolution des deltas: 100% (7/7), fait.
brahim@Training:~/TP1$
```

- Sous le dépôt cloné, vérifiez qu'un lien vers le dépôt distant a été créé :

```
brahim@Training:~/TP1$ cd tpgit-remote/
brahim@Training:~/TP1,tpgit-remote$ git remote -v
origin  https://github.com/brahimhamdi/tpgit (fetch)
origin  https://github.com/brahimhamdi/tpgit (push)
brahim@Training:~/TP1,tpgit-remote$
```

`git clone` crée par défaut un lien nommé 'origin'.

Les branches

26. Dans le dépôt tpgit-local, On va commencer par un retour en arrière vers la validation de la dernière version de hello.py

```
brahim@Training:~/TP1,tpgit-local$ git log --oneline
9082484 (HEAD -> master, origin/master) Annulation de la suppression du fichier hello.py
0064c70 Récupération du fichier hello.py après suppression
501fefc Explication suppression de hello.py sur README.md
1070124 Suppression de hello.py
79a7550 ajout de .gitignore
1d3431d modification de hello.py
6cd1bd4 renommage fichier1 --> fichier2
0c54fd2 Add fichier1
63c5fef Add hello.py
bbc0000 initialistion du projet
brahim@Training:~/TP1,tpgit-local$
brahim@Training:~/TP1,tpgit-local$ git reset --hard 63c5fef
HEAD est maintenant à 63c5fef Add hello.py
brahim@Training:~/TP1,tpgit-local$
brahim@Training:~/TP1,tpgit-local$ git log --oneline
63c5fef (HEAD -> master) Add hello.py
bbc0000 initialistion du projet
brahim@Training:~/TP1,tpgit-local$
```

27. Créer la nouvelle branche wip.

```
brahim@Training:~/TP1,tpgit-local$ git branch
* master
brahim@Training:~/TP1,tpgit-local$ git branch wip
brahim@Training:~/TP1,tpgit-local$ git branch
* master
  wip
brahim@Training:~/TP1,tpgit-local$
```

- Visualisez l'historique des commits.

```
brahim@Training:~/TP1,tpgit-local$ git log --oneline
63c5fef (HEAD -> master, wip) Add hello.py
bbc0000 initialistion du projet
brahim@Training:~/TP1,tpgit-local$
```

Notez bien que les HEAD pointe sur la branche master.

28. Basculer sur la nouvelle branche wip et visualiser l'historique.

```
brahim@Training:~/TP1,tpgit-local$ git switch wip
Basculement sur la branche 'wip'
brahim@Training:~/TP1,tpgit-local$ git branch
  master
* wip
brahim@Training:~/TP1,tpgit-local$ 
brahim@Training:~/TP1,tpgit-local$ git log --oneline
63c5fef (HEAD -> wip, master) Add hello.py
bbc0000 initialistion du projet
brahim@Training:~/TP1,tpgit-local$
```

La référence **HEAD** pointe maintenant effectivement sur **wip**. Résultat : tout nouveau commit sera ajouté après **wip** et non plus après **master**.

- Ajouter 2 commits sur la branche wip

```
brahim@Training:~/TP1/tppgit-local$ ls
hello.py  __pycache__  README.md
brahim@Training:~/TP1/tppgit-local$ echo "__pycache__" > .gitignore
brahim@Training:~/TP1/tppgit-local$ git add .
brahim@Training:~/TP1/tppgit-local$ git commit -m "Ajout .gitignore + ignorer le dossier __pycache__"
[wip a6fc769] Ajout .gitignore + ignorer le dossier __pycache__
1 file changed, 1 insertion(+)
create mode 100644 .gitignore
brahim@Training:~/TP1/tppgit-local$ echo "On doit ignorer tous les fichier .pyc" >> README.md
brahim@Training:~/TP1/tppgit-local$ git commit -am "Modification de README.md"
[wip 8a2104e] Modification de README.md
1 file changed, 1 insertion(+)
brahim@Training:~/TP1/tppgit-local$
```

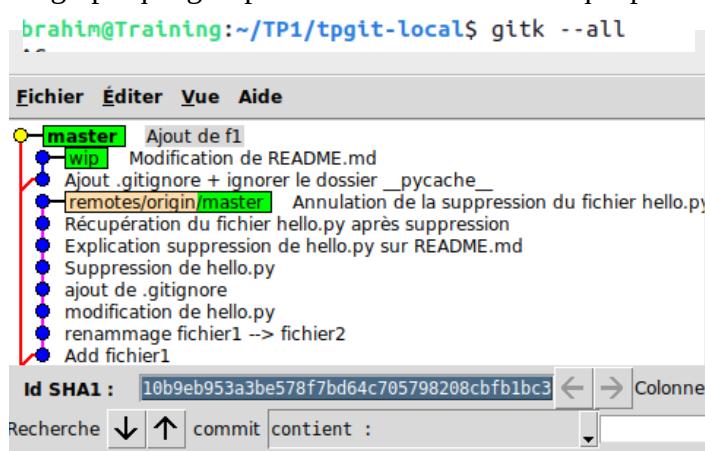
- Basculer une autre fois sur master et y ajouter un nouveau commit

```
brahim@Training:~/TP1/tppgit-local$ git switch master
Basculement sur la branche 'master'
Votre branche est en retard sur 'origin/master' de 8 commits, et peut être mise à jour en avance rapide.
  (utilisez "git pull" pour mettre à jour votre branche locale)
brahim@Training:~/TP1/tppgit-local$ touch f1
brahim@Training:~/TP1/tppgit-local$ git add .
brahim@Training:~/TP1/tppgit-local$ git commit -m "Ajout de f1"
[master 10b9eb9] Ajout de f1
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 __pycache__/hello.cpython-310.pyc
create mode 100644 f1
brahim@Training:~/TP1/tppgit-local$
```

- Pour visualiser l'arbre des commits, vous pouvez utiliser `git log` avec l'option `--graph`. Et pour voir toutes les branches, vous pouvez utiliser l'option `--all`. Visualisez l'historique de toutes les branches du projet en mode graphique

```
brahim@Training:~/TP1/tppgit-local$ git log --oneline --graph --all
* 10b9eb9 (HEAD -> master) Ajout de f1
| * 8a2104e (wip) Modification de README.md
| * a6fc769 Ajout .gitignore + ignorer le dossier __pycache__
|
| * 9082484 (origin/master) Annulation de la suppression du fichier hello.py
| * 0064c70 Récupération du fichier hello.py après suppression
| * 501fefc Explication suppression de hello.py sur README.md
| * 1070124 Suppression de hello.py
| * 79a7550 ajout de .gitignore
| * 1d3431d modification de hello.py
| * 6cd1bd4 renommage fichier1 --> fichier2
| * 0c54fd2 Add fichier1
|
* 63c5fef Add hello.py
* bbca000 initialisation du projet
brahim@Training:~/TP1/tppgit-local$
```

- Utiliser l'outil graphique `gitk` pour faire la même chose que précédemment.



Fusion des branches

29. Vérifiez que vous êtes sur la branche 'wip', à l'aide de la commande `git branch`, sinon basculez sur cette branche.

```
brahim@Training:~/TP1/tppgit-local$ git branch
* master
  wip
brahim@Training:~/TP1/tppgit-local$ git switch wip
Basculement sur la branche 'wip'
brahim@Training:~/TP1/tppgit-local$ git branch
  master
* wip
brahim@Training:~/TP1/tppgit-local$ █
```

- Ajoutez la fonction suivante à **hello.py**

```
def ecrireXfois(x, chaine):
    for i in range(x):
        ecrire(chaine)

brahim@Training:~/TP1/tppgit-local$ ls
hello.py  README.md
brahim@Training:~/TP1/tppgit-local$ git status
Sur la branche wip
Modifications qui ne seront pas validées :
  (utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git restore <fichier>..." pour annuler les modifications dans le répertoire de travail)
    modifié :          hello.py

aucune modification n'a été ajoutée à la validation (utilisez "git add" ou "git commit -a")
brahim@Training:~/TP1/tppgit-local$
```

- Commitez la modification.

30. Retournez sur master et afficher l'historique complet du projet

```
brahim@Training:~/TP1/tppgit-local$ git log --oneline --graph --all
* 7609899 (wip) modif hello.py - ajout fonction ecrireXfois
* 8a2104e Modification de README.md
* a6fc769 Ajout .gitignore + ignorer le dossier __pycache__
| * 10b9eb9 (HEAD -> master) Ajout de f1
|
| * 9082484 (origin/master) Annulation de la suppression du fichier hello.py
| * 0064c70 Récupération du fichier hello.py après suppression
| * 501fefc Explication suppression de hello.py sur README.md
| * 1070124 Suppression de hello.py
| * 79a7550 ajout de .gitignore
| * 1d3431d modification de hello.py
| * 6cd1bd4 renommage fichier1 --> fichier2
| * 0c54fd2 Add fichier1
|
* 63c5fef Add hello.py
* bbca000 initialisation du projet
brahim@Training:~/TP1/tppgit-local$
```

31. Regardez le contenu du fichier **hello.py**. Que remarquez-vous ?

```
brahim@Training:~/TP1/tppgit-local$ cat hello.py
print "Hello world!"
brahim@Training:~/TP1/tppgit-local$
brahim@Training:~/TP1/tppgit-local$ git branch
* master
  wip
brahim@Training:~/TP1/tppgit-local$ █
```

La fonction `ecrireXfois()` n'est plus là ! Par ce que on a fait la modification sur la branche `wip` et pas la branche `master`. On doit fusionner `wip` dans `master` pour mettre à jour version sur cette branche.

- Fusionner la branche `wip` dans `master`, puis vérifier de nouveau l'historique et le contenu de `hello.py`

```
brahim@Training:~/TP1/tppgit-local$ git branch
* master
  wip
brahim@Training:~/TP1/tppgit-local$ git merge wip
Merge made by the 'ort' strategy.
.gitignore | 1 +
README.md   | 1 +
hello.py    | 9 ++++++-
 3 files changed, 10 insertions(+), 1 deletion(-)
create mode 100644 .gitignore
brahim@Training:~/TP1/tppgit-local$ brahim@Training:~/TP1/tppgit-local$ cat hello.py
def ecrire(chaine):
    print chaine

ecrire("Hello world!")

def ecrireXfois(x, chaine):
    for i in range(x):
        ecrire(chaine)
brahim@Training:~/TP1/tppgit-local$ brahim@Training:~/TP1/tppgit-local$ git log --oneline --graph --all
* ee4c4c4 (HEAD -> master) Merge branch 'wip'
|\ 
| * 7609899 (wip) modif hello.py - ajout fonction ecrireXfois
| * 8a2104e Modification de README.md
| * a6fc769 Ajout .gitignore + ignorer le dossier __pycache__
* | 10b9eb9 Ajout de f1
|| 
| * 9082484 (origin/master) Annulation de la suppression du fichier hello.py
| * 0064c70 Récupération du fichier hello.py après suppression
| * 501fefc Explication suppression de hello.py sur README.md
| * 1070124 Suppression de hello.py
| * 79a7550 ajout de .gitignore
| * 1d3431d modification de hello.py
| * 6cd1bd4 renommage fichier1 --> fichier2
| * 0c54fd2 Add fichier1
|| 
* 63c5fef Add hello.py
* bbcba000 initialisation du projet
brahim@Training:~/TP1/tppgit-local$
```

Fusion avec un conflit

Nous allons maintenant provoquer volontairement un conflit, pour voir comment se comporte Git dans ce cas au moment de la fusion. On va modifier la même ligne du fichier `hello.py` dans les 2 branches (master et wip)

32. Dans `master`, modifier la fonction `écrire()` pour qu'elle affiche en majuscules et faire un commit

```
brahim@Training:~/TP1/tpgit-local$ ls
f1 hello.py __pycache__ README.md
brahim@Training:~/TP1/tpgit-local$ cat hello.py
def ecrire(chaine):
    print chaine.upper()

ecrire("Hello world!")

def ecrireXFois(x, chaine):
    for i in range(x):
        ecrire(chaine)
brahim@Training:~/TP1/tpgit-local$ git commit -am "Afficher en masjuscule!"
[master c508d3b] Afficher en masjuscule!
 1 file changed, 1 insertion(+), 1 deletion(-)
brahim@Training:~/TP1/tpgit-local$
```

33. Dans `wip`, effectuez un commit dans lequel la fonction `écrire()` affiche en minuscules.

```
brahim@Training:~/TP1/tppgit-local$ cat hello.py
def ecrire(chaine):
    print chaine.lower()

ecrire("Hello world!")

def ecrireXFois(x, chaine):
    for i in range(x):
        ecrire(chaine)
brahim@Training:~/TP1/tppgit-local$ brahim@Training:~/TP1/tppgit-local$ git commit -am "Afficher en minuscule !"
[wip d6fa075] Afficher en minuscule !
 1 file changed, 1 insertion(+), 1 deletion(-)
brahim@Training:~/TP1/tppgit-local$
```

- #### ○ Visualisez l'historique

```
brahim@Training:~/TP1/tpgit-local$ git log --oneline --graph --all
* d6fa075 (HEAD -> wip) Afficher en minuscule !
| * c508d3b (master) Afficher en masjuscule!
| * ee4c4c4 Merge branch 'wip'
| |\ \
| | |
| |
* | 7609899 modif hello.py - ajout fonction ecrireXFois
* | 8a2104e Modification de README.md
* | a6fc769 Ajout .gitignore + ignorer le dossier __pycache__
* | 10b9eb9 Ajout de f1
|
* | 9082484 (origin/master) Annulation de la suppression du fichier hello.py
* | 0064c70 Récupération du fichier hello.py après suppression
* | 501fecf Explication suppression de hello.py sur README.md
* | 1070124 Suppression de hello.py
* | 79a7550 ajout de .gitignore
* | 1d3431d modification de hello.py
* | 6cd1bd4 renommage fichier1 --> fichier2
| * 0c54fd2 Add fichier1
|
* 63c5fef Add hello.py
* bbc000 initialistion du projet
brahim@Training:~/TP1/tpgit-local$
```

- Fusionner la branche wip dans master. Que remarquez vous ?

```

brahim@Training:~/TP1/tppgit-local$ git switch master
Basculement sur la branche 'master'
Votre branche et 'origin/master' ont divergé,
et ont 6 et 8 commits différents chacune respectivement.
    (utilisez "git pull" pour fusionner la branche distante dans la vôtre)
brahim@Training:~/TP1/tppgit-local$ git merge wip
Fusion automatique de hello.py
CONFLICT (contenu) : Conflit de fusion dans hello.py
La fusion automatique a échoué ; réglez les conflits et validez le résultat.
brahim@Training:~/TP1/tppgit-local$
```

34. E
d
i
t
e
z

le fichier **hello.py** comme Git vous y invite et résoudre le conflit (déjà marqué dans le fichier)

```

def ecrire(chaine):
<<<<< HEAD
    print chaine.upper()
=====
    print chaine.lower()
>>>> wip

    ecrire("Hello world!")

def ecrireXfois(x, chaine):
    for i in range(x):
        ecrire(chaine)
```

- Supprimer le balisage

de conflit et la modification que vous ne voulez pas garder. Committer la version à conserver.

```

brahim@Training:~/TP1/tppgit-local$ vim hello.py
brahim@Training:~/TP1/tppgit-local$ git commit -am "Résolution du conflit --> continuer le merge !"
[master 38aca7e] Résolution du conflit --> continuer le merge !
brahim@Training:~/TP1/tppgit-local$
```

○
○

Visualiser l'historique de tout le projet

```

brahim@Training:~/TP1/tppgit-local$ git log --oneline --graph --all
* 38aca7e (HEAD -> master) Résolution du conflit --> continuer le merge !
|\ \
| * d6fa075 (wip) Afficher en minuscule !
| * c508d3b Afficher en masjuscule!
| * ee4c4c4 Merge branch 'wip'
|\ \
| * 7609899 modif hello.py - ajout fonction ecrireXfois
| * 8a2104e Modification de README.md
| * a6fc769 Ajout .gitignore + ignorer le dossier __pycache__
| * 10b9eb9 Ajout de f1
|/
| * 9082484 (origin/master) Annulation de la suppression du fichier hello.py
| * 0064c70 Récupération du fichier hello.py après suppression
| * 501fefc Explication suppression de hello.py sur README.md
| * 1070124 Suppression de hello.py
| * 79a7550 ajout de .gitignore
| * 1d3431d modification de hello.py
| * 6cd1bd4 renommage fichier1 --> fichier2
| * 0c54fd2 Add fichier1
|/
* 63c5fef Add hello.py
* bbca000 initialisation du projet
brahim@Training:~/TP1/tppgit-local$
```

Rebaser

Nous allons maintenant utiliser **git rebase** dans plusieurs cas réels.

35. En étant placé sur la branche 'master', annulez le commit de fusion

```

brahim@Training:~/TP1/tppgit-local$ git checkout master
Déjà sur 'master'
Votre branche et 'origin/master' ont divergé,
et ont 8 et 8 commits différents chacune respectivement.
(utilisez "git pull" pour fusionner la branche distante dans la vôtre)
brahim@Training:~/TP1/tppgit-local$ git reset --hard HEAD^
HEAD est maintenant à c508d3b Afficher en masjuscule!
brahim@Training:~/TP1/tppgit-local$
```

- En étant placé sur la branche 'wip', on supprime le commit qui est source de conflit

```

brahim@Training:~/TP1/tppgit-local$ git checkout wip
Basculement sur la branche 'wip'
brahim@Training:~/TP1/tppgit-local$ git reset --hard HEAD^
HEAD est maintenant à 7609899 modif hello.py - ajout fonction ecrireXfois
brahim@Training:~/TP1/tppgit-local$
```

- Retournez sur la branche master et visualisez l'historique du projet

```

brahim@Training:~/TP1/tppgit-local$ git switch master
Basculement sur la branche 'master'
Votre branche et 'origin/master' ont divergé,
et ont 6 et 8 commits différents chacune respectivement.
(utilisez "git pull" pour fusionner la branche distante dans la vôtre)
brahim@Training:~/TP1/tppgit-local$
brahim@Training:~/TP1/tppgit-local$ git log --oneline --graph --all
* c508d3b (HEAD -> master) Afficher en masjuscule!
* ee4c4c4 Merge branch 'wip'
|\ 
| * 7609899 (wip) modif hello.py - ajout fonction ecrireXfois
| * 8a2104e Modification de README.md
| * a6fc769 Ajout .gitignore + ignorer le dossier __pycache__
* | 10b9eb9 Ajout de f1
|/
| * 9082484 (origin/master) Annulation de la suppression du fichier hello.py
| * 0064c70 Récupération du fichier hello.py après suppression
| * 501fefc Explication suppression de hello.py sur README.md
| * 1070124 Suppression de hello.py
| * 79a7550 ajout de .gitignore
| * 1d3431d modification de hello.py
| * 6cd1bd4 renommage fichier1 --> fichier2
| * 0c54fd2 Add fichier1
|/
* 63c5fef Add hello.py
* bbca000 initialisation du projet
brahim@Training:~/TP1/tppgit-local$
```

36. Avant de faire la fusion de la branche 'wip' sur la branche 'master', nous allons donc déplacer la base de wip *au bout* de la branche master.

Placez-vous dans wip et effectuez le rebasage :

```

brahim@Training:~/TP1/tppgit-local$ git checkout wip
Basculement sur la branche 'wip'
brahim@Training:~/TP1/tppgit-local$ git rebase master
Rebasage et mise à jour de refs/heads/wip avec succès.
brahim@Training:~/TP1/tppgit-local$
```

- Visualisez l'historique complet, et comparer avec l'historique avant le rebasage

```

brahim@Training:~/TP1/tppgit-local$ git log --oneline --graph --all
* c508d3b (HEAD -> wip, master) Afficher en masjuscule!
* ee4c4c4 Merge branch 'wip'
| \
| * 7609899 modif hello.py - ajout fonction ecrireXfois
| * 8a2104e Modification de README.md
| * a6fc769 Ajout .gitignore + ignorer le dossier __pycache__
| | 10b9eb9 Ajout de f1
| /
| * 9082484 (origin/master) Annulation de la suppression du fichier hello.py
| * 0064c70 Récupération du fichier hello.py après suppression
| * 501fefc Explication suppression de hello.py sur README.md
| * 1070124 Suppression de hello.py
| * 79a7550 ajout de .gitignore
| * 1d3431d modification de hello.py
| * 6cd1bd4 renommage fichier1 --> fichier2
| * 0c54fd2 Add fichier1
| /
* 63c5fef Add hello.py
* bbca000 initialisation du projet
brahim@Training:~/TP1/tppgit-local$ █

```

- Retourner dans 'master' et faire un 'merge fast-forward'

```

brahim@Training:~/TP1/tppgit-local$ git switch master
Basculement sur la branche 'master'
Votre branche et 'origin/master' ont divergé,
et ont 6 et 8 commits différents chacune respectivement.
    (utilisez "git pull" pour fusionner la branche distante dans la vôtre)
brahim@Training:~/TP1/tppgit-local$ 
brahim@Training:~/TP1/tppgit-local$ git merge wip
Déjà à jour.
brahim@Training:~/TP1/tppgit-local$ 
brahim@Training:~/TP1/tppgit-local$ git log --oneline --graph --all
* c508d3b (HEAD -> master, wip) Afficher en masjuscule!
* ee4c4c4 Merge branch 'wip'
| \
| * 7609899 modif hello.py - ajout fonction ecrireXfois
| * 8a2104e Modification de README.md
| * a6fc769 Ajout .gitignore + ignorer le dossier __pycache__
| | 10b9eb9 Ajout de f1
| /
| * 9082484 (origin/master) Annulation de la suppression du fichier hello.py
| * 0064c70 Récupération du fichier hello.py après suppression
| * 501fefc Explication suppression de hello.py sur README.md
| * 1070124 Suppression de hello.py
| * 79a7550 ajout de .gitignore
| * 1d3431d modification de hello.py
| * 6cd1bd4 renommage fichier1 --> fichier2
| * 0c54fd2 Add fichier1
| /
* 63c5fef Add hello.py
* bbca000 initialisation du projet
brahim@Training:~/TP1/tppgit-local$ █

```

37. Pour provoquer un conflit avant de rebaser, nous allons maintenant modifier la même ligne du README.md dans les 2 branches.

- README.md sur la branche master

```

brahim@Training:~/TP1/tppgit-local$ git branch
* master
  wip
brahim@Training:~/TP1/tppgit-local$ vim README.md
brahim@Training:~/TP1/tppgit-local$ git commit -am "modif ligne2 sur branche master"
[master d55628f] modif ligne2 sur branche master
 1 file changed, 1 insertion(+), 1 deletion(-)
brahim@Training:~/TP1/tppgit-local$ 

```

- README.md sur la branche wip

```

brahim@Training:~/TP1/tppgit-local$ git switch wip
Basculement sur la branche 'wip'
brahim@Training:~/TP1/tppgit-local$ vim README.md
brahim@Training:~/TP1/tppgit-local$ git commit -am "modif ligne2 sur branche wip"
[wip a141412] modif ligne2 sur branche wip
 1 file changed, 1 insertion(+), 1 deletion(-)
brahim@Training:~/TP1/tppgit-local$ 
brahim@Training:~/TP1/tppgit-local$ 

```

- Visualiser l'historique

```
brahim@Training:~/TP1/tppgit-local$ git log --oneline --graph --all
* a141412 (HEAD -> wip) modif ligne2 sur branche wip
| * d55628f (master) modif ligne2 sur branche master
|/
| * c508d3b Afficher en masjuscule!
| * ee4c4c4 Merge branch 'wip'
| \
| * 7609899 modif hello.py - ajout fonction ecrireXfois
| * 8a2104e Modification de README.md
| * a6fc769 Ajout .gitignore + ignorer le dossier __pycache__
* | 10b9eb9 Ajout de f1
|/
| * 9082484 (origin/master) Annulation de la suppression du fichier hello.py
| * 0064c70 Récupération du fichier hello.py après suppression
| * 501fefc Explication suppression de hello.py sur README.md
| * 1070124 Suppression de hello.py
| * 79a7550 ajout de .gitignore
| * 1d3431d modification de hello.py
| * 6cd1bd4 renommage fichier1 --> fichier2
| * 0c54fd2 Add fichier1
|/
| * 63c5fef Add hello.py
* bbca000 initialistion du projet
```

38. Tentez

un rebasage de 'wip' sur 'master'

```
brahim@Training:~/TP1/tppgit-local$ git switch wip
Déjà sur 'wip'
brahim@Training:~/TP1/tppgit-local$ git rebase master
Fusion automatique de README.md
CONFLIT (contenu) : Conflit de fusion dans README.md
error: impossible d'appliquer a141412... modif ligne2 sur branche wip
astuce: Resolve all conflicts manually, mark them as resolved with
astuce: "git add/rm <conflicted_files>", then run "git rebase --continue".
astuce: You can instead skip this commit: run "git rebase --skip".
astuce: To abort and get back to the state before "git rebase", run "git rebase --abort".
Impossible d'appliquer a141412... modif ligne2 sur branche wip
brahim@Training:~/TP1/tppgit-local$
```

Un conflit a donc été détecté dans README.md. On est dans un cas similaire à une fusion avec conflit, que vous devez régler de la même façon : éditez le fichier README.md qui contient un marquage de conflit

- Conservez ce que vous voulez.
- **git status** vous rappelle ce que vous avez à faire : utilisez **git add** pour indexer la résolution du conflit

```
brahim@Training:~/TP1/tppgit-local$ vim README.md
brahim@Training:~/TP1/tppgit-local$ git status
rebasage interactif en cours ; sur d55628f
Dernière commande effectuée (1 commande effectuée) :
  pick a141412 modif ligne2 sur branche wip
Aucune commande restante.
Vous êtes en train de rebaser la branche 'wip' sur 'd55628f'.
(réglez les conflits puis lancez "git rebase --continue")
(utilisez "git rebase --skip" pour sauter ce patch)
(utilisez "git rebase --abort" pour extraire la branche d'origine)

Chemins non fusionnés :
  (utilisez "git restore --staged <fichier>..." pour désindexer)
  (utilisez "git add <fichier>..." pour marquer comme résolu)
    modifié des deux côtés : README.md

aucune modification n'a été ajoutée à la validation (utilisez "git add" ou "git commit -a")
brahim@Training:~/TP1/tppgit-local$
brahim@Training:~/TP1/tppgit-local$ git add .
```

- Le nouveau **git status** vous indique comment poursuivre le rebasage : **git rebase --continue**

```

brahim@Training:~/TP1/tppgit-local$ git rebase --continue
[HEAD détachée a78bb72] modif ligne2 sur branche wip
  1 file changed, 1 insertion(+), 1 deletion(-)
Rebasage et mise à jour de refs/heads/wip avec succès.
brahim@Training:~/TP1/tppgit-local$ git status
Sur la branche wip
rien à valider, la copie de travail est propre
brahim@Training:~/TP1/tppgit-local$ █

```

- Constatez que votre arbre de commit est linéaire

```

brahim@Training:~/TP1/tppgit-local$ git log --oneline --graph --all
* a78bb72 (HEAD -> wip) modif ligne2 sur branche wip
* d55628f (master) modif ligne2 sur branche master
* c508d3b Afficher en masjuscule!
* ee4c4c4 Merge branch 'wip'
|\ 
| * 7609899 modif hello.py - ajout fonction ecrireXFois
| * 8a2104e Modification de README.md
| * a6fc769 Ajout .gitignore + ignorer le dossier __pycache__
* | 10b9eb9 Ajout de f1
|/
| * 9082484 (origin/master) Annulation de la suppression du fichier hello.py
| * 0064c70 Récupération du fichier hello.py après suppression
| * 501fefc Explication suppression de hello.py sur README.md
| * 1070124 Suppression de hello.py
| * 79a7550 ajout de .gitignore
| * 1d3431d modification de hello.py
| * 6cd1bd4 renommage fichier1 --> fichier2
| * 0c54fd2 Add fichier1
|/
* 63c5fef Add hello.py
* bbca000 initialistion du projet
brahim@Training:~/TP1/tppgit-local$ █

```

Vous pouvez retourner dans master pour exécuter un 'merge fast-forward' : `git merge wip`

```

brahim@Training:~/TP1/tppgit-local$ git switch master
Basculement sur la branche 'master'
Votre branche et 'origin/master' ont divergé,
et ont 7 et 8 commits différents chacune respectivement.
  (utilisez "git pull" pour fusionner la branche distante dans la vôtre)
brahim@Training:~/TP1/tppgit-local$ 
brahim@Training:~/TP1/tppgit-local$ git merge wip
Mise à jour d55628f..a78bb72
Fast-forward
 README.md | 2 ++
 1 file changed, 1 insertion(+), 1 deletion(-)
brahim@Training:~/TP1/tppgit-local$ 
brahim@Training:~/TP1/tppgit-local$ git log --oneline --graph --all
* a78bb72 (HEAD -> master, wip) modif ligne2 sur branche wip
* d55628f modif ligne2 sur branche master
* c508d3b Afficher en masjuscule!
* ee4c4c4 Merge branch 'wip'
|\ 
| * 7609899 modif hello.py - ajout fonction ecrireXFois
| * 8a2104e Modification de README.md
| * a6fc769 Ajout .gitignore + ignorer le dossier __pycache__
* | 10b9eb9 Ajout de f1
|/
| * 9082484 (origin/master) Annulation de la suppression du fichier hello.py
| * 0064c70 Récupération du fichier hello.py après suppression
| * 501fefc Explication suppression de hello.py sur README.md
| * 1070124 Suppression de hello.py
| * 79a7550 ajout de .gitignore
| * 1d3431d modification de hello.py
| * 6cd1bd4 renommage fichier1 --> fichier2
| * 0c54fd2 Add fichier1
|/
* 63c5fef Add hello.py
* bbca000 initialistion du projet
brahim@Training:~/TP1/tppgit-local$ 

```

Étiquettes

39. Etiqueter le commit (commit courant où pointe le HEAD)

```
brahim@Training:~/TP1/tppgit-local$ git branch
* master
  wip
brahim@Training:~/TP1/tppgit-local$ git tag v1.0
brahim@Training:~/TP1/tppgit-local$
```

40. étiqueter un ancien commit (commit 'après-coup')

```
brahim@Training:~/TP1/tppgit-local$ git tag v0.4 7609899
brahim@Training:~/TP1/tppgit-local$
```

41. Lister les étiquettes de votre projet avec la commande `git tag` sans paramètre

```
brahim@Training:~/TP1/tppgit-local$ git tag
v0.4
v1.0
brahim@Training:~/TP1/tppgit-local$
```

42. Afficher les étiquettes associées à un commit.

```
brahim@Training:~/TP1/tppgit-local$ git log --oneline --decorate
a78bb72 (HEAD -> master, tag: v1.0, wip) modif ligne2 sur branche wip
d55628f modif ligne2 sur branche master
c508d3b Afficher en masjuscule!
ee4c4c4 Merge branch 'wip'
7609899 (tag: v0.4) modif hello.py - ajout fonction ecrireXFois
10b9eb9 Ajout de f1
8a2104e Modification de README.md
a6fc769 Ajout .gitignore + ignorer le dossier __pycache__
63c5fef Add hello.py
bbca000 initialistion du projet
brahim@Training:~/TP1/tppgit-local$
```

43. Une étiquette n'est pas automatiquement transmise sur le dépôt distant lorsque le commit associé est poussé. Il faut le faire *manuellement* à l'aide de `git push`
Poussez l'étiquette créée précédemment vers le dépôt distant

```
brahim@Training:~/TP1/tppgit-local$ git remote -v
origin https://github.com/brahimhamdi/tppgit.git (fetch)
origin https://github.com/brahimhamdi/tppgit.git (push)
brahim@Training:~/TP1/tppgit-local$ git push origin v1.0
Username for 'https://github.com': brahimhamdi
Password for 'https://brahimhamdi@github.com':
Énumération des objets: 28, fait.
Décompte des objets: 100% (28/28), fait.
Compression par delta en utilisant jusqu'à 8 fils d'exécution
Compression des objets: 100% (22/22), fait.
Écriture des objets: 100% (25/25), 2.51 Kio | 856.00 Kio/s, fait.
Total 25 (delta 9), réutilisés 0 (delta 0), réutilisés du pack 0
remote: Resolving deltas: 100% (9/9), done.
To https://github.com/brahimhamdi/tppgit.git
 * [new tag]           v1.0 -> v1.0
brahim@Training:~/TP1/tppgit-local$
```

Squasher

44. Nettoyer le répertoire de travail par un `git reset --hard HEAD`.

```
brahim@Training:~/TP1/tppgit-local$ git branch
* master
  wip
brahim@Training:~/TP1/tppgit-local$ git reset --hard HEAD
HEAD est maintenant à a78bb72 modif ligne2 sur branche wip
brahim@Training:~/TP1/tppgit-local$ git log --oneline --graph --all
* a78bb72 (HEAD -> master, tag: v1.0, wip) modif ligne2 sur branche wip
* d55628f modif ligne2 sur branche master
* c508d3b Afficher en masjuscule!
* ee4c4c4 Merge branch 'wip'
\\
| * 7609899 (tag: v0.4) modif hello.py - ajout fonction ecrireXFois
| * 8a2104e Modification de README.md
| * a6fc769 Ajout .gitignore + ignorer le dossier __pycache__
* | 10b9eb9 Ajout de f1
|
| * 9082484 (origin/master) Annulation de la suppression du fichier hello.py
| * 0064c70 Récupération du fichier hello.py après suppression
| * 501fefc Explication suppression de hello.py sur README.md
| * 1070124 Suppression de hello.py
| * 79a7550 ajout de .gitignore
| * 1d3431d modification de hello.py
| * 6cd1bd4 renommage fichier1 --> fichier2
| * 0c54fd2 Add fichier1
|
* 63c5fef Add hello.py
* bbca000 initialistion du projet
brahim@Training:~/TP1/tppgit-local$
```

45. On veut re-écrire le message de l'avant-avant dernier commit (id=c508d3b ici) de « Afficher en masjuscule! » vers « Afficher en minuscule ! »

```
* d55628f modif ligne2 sur branche master
* c508d3b Afficher en masjuscule!
* ee4c4c4 Merge branch 'wip'
\\
| * 7609899 (tag: v0.4) modif hello.py - ajout fonct
```

Pour revenir avant ce commit, copier l'id du commit qui lui précède (id=ee4c4c4 ici) et mettre-le en argument de la commande `git rebase -i`

```
brahim@Training:~/TP1/tppgit-local$ git rebase -i ee4c4c4
```

- Git ouvre alors l'éditeur de texte avec une liste de *chooses à faire*.

Vous allez lui demander d'interrompre le processus, en remplaçant la commande **pick** par une commande **edit**, puis vous sauvegardez la liste et vous quittez

```
edit c508d3b Afficher en masjuscule!
pick d55628f modif ligne2 sur branche master
pick a78bb72 modif ligne2 sur branche wip

# Rebasage de ee4c4c4..a78bb72 sur ee4c4c4 (3 commandes)
#
# Commandes :
# p, pick <commit> = utiliser le commit
# r, reword <commit> = utiliser le commit, mais reformuler son message
# e, edit <commit> = utiliser le commit, mais s'arrêter pour le modifier
# s, squash <commit> = utiliser le commit, mais le fusionner avec le précédent
# f, fixup [-C | -c] <commit> = comme "squash", mais en ne gardant que le message
#                   du commit précédent, à moins que -C ne soit utilisé, auquel cas, conserv
#                   ne conserver que le message de ce commit ; -c est identique à -C mais ou
#                   un éditeur
# x, exec <commit> = lancer la commande (reste de la ligne) dans un shell
# b, break = déraccorder ici (on peut continuer ensuite avec 'git rebase --continue')
```

- Git vous indique alors la marche à suivre pour la suite

```

brahim@Training:~/TP1/tppgit-local$ git rebase -i ee4c4c4
Arrêt à c508d3b... Afficher en masjuscule!
Vous pouvez corriger le commit maintenant, avec

    git commit --amend

après avoir réalisé vos modifications, lancez

    git rebase --continue
brahim@Training:~/TP1/tppgit-local$
```

- Le commit a été appliqué, et nous pouvons l'*amender* à l'aide de `git commit --amend`.

```

brahim@Training:~/TP1/tppgit-local$ git commit --amend
[HEAD détachée 69aab4c] Afficher en minuscule !
  Date: Sun Jan 7 16:59:13 2024 +0100
  1 file changed, 1 insertion(+), 1 deletion(-)
brahim@Training:~/TP1/tppgit-local$
```

- Utilisez `git status` qui vous indique ce que vous pouvez faire maintenant

```

brahim@Training:~/TP1/tppgit-local$ git status
rebasage interactif en cours ; sur ee4c4c4
Dernière commande effectuée (1 commande effectuée) :
  edit c508d3b Afficher en masjuscule!
Prochaines commandes à effectuer (2 commandes restantes) :
  pick d55628f modif ligne2 sur branche master
  pick a78bb72 modif ligne2 sur branche wip
  (utilisez "git rebase --edit-todo" pour voir et éditer)
Vous êtes actuellement en train d'éditer un commit pendant un rebasage de la branche 'master' sur 'ee4c4c4'.
  (utilisez "git commit --amend" pour corriger le commit actuel)
  (utilisez "git rebase --continue" quand vous avez effectué toutes vos modifications)

rien à valider, la copie de travail est propre
brahim@Training:~/TP1/tppgit-local$ brahim@Training:~/TP1/tppgit-local$ git rebase --continue
Rebasage et mise à jour de refs/heads/master avec succès.
brahim@Training:~/TP1/tppgit-local$
```

Vous pouvez donc continuer à corriger le commit, ou continuer le processus de rebasage.

- Visualiser l'historique en vérifiant que le message du commit a bien changé.

```

brahim@Training:~/TP1/tppgit-local$ git log --oneline --graph --all
* 92da939 (HEAD -> master) modif ligne2 sur branche wip
* c655abe modif ligne2 sur branche master
* 69aab4c Afficher en minuscule !
| * a78bb72 (tag: v1.0, wip) modif ligne2 sur branche wip
| * d55628f modif ligne2 sur branche master
| * c508d3b Afficher en masjuscule!
|/
| * ee4c4c4 Merge branch 'wip'
| \
| * 7609899 (tag: v0.4) modif hello.py - ajout fonction ecrireXfois
| * 8a2104e Modification de README.md
| * a6fc769 Ajout .gitignore + ignorer le dossier __pycache__
| * 10b9eb9 Ajout de f1
|/
| * 9082484 (origin/master) Annulation de la suppression du fichier hello.py
| * 0064c70 Récupération du fichier hello.py après suppression
| * 501fefc Explication suppression de hello.py sur README.md
| * 1070124 Suppression de hello.py
| * 79a7550 ajout de .gitignore
| * 1d3431d modification de hello.py
| * 6cd1bd4 renommage fichier1 --> fichier2
| * 0c54fd2 Add fichier1
|/
| * 63c5fef Add hello.py
* bbca000 initialisation du projet
brahim@Training:~/TP1/tppgit-local$
```

Cherry-pick

46. Créer une nouvelle branche feature1

```
brahim@Training:~/TP1/tppgit-local$ git branch feature1
brahim@Training:~/TP1/tppgit-local$ git branch
  feature1
* master
brahim@Training:~/TP1/tppgit-local$
```

47. Ajouter 2 commits à la branche master.

```
brahim@Training:~/TP1/tppgit-local$ touch f2
brahim@Training:~/TP1/tppgit-local$ git add f2
brahim@Training:~/TP1/tppgit-local$ git commit -m "ajout de f2 après création de la branche feature1"
[master 622ce47] ajout de f2 après création de la branche feature1
 1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 f2

brahim@Training:~/TP1/tppgit-local$ echo ligne2 > f2
brahim@Training:~/TP1/tppgit-local$ git add f2
brahim@Training:~/TP1/tppgit-local$ git commit -m "modif de f2 après création de la branche feature1"
[master 87fad06] modif de f2 après création de la branche feature1
 1 file changed, 1 insertion(+)
brahim@Training:~/TP1/tppgit-local$
```

48. Basculer sur la branche feature1 et copier le dernier commit de la branche master

```
brahim@Training:~/TP1/tppgit-local$ git log --oneline master
87fad06 (master) modif de f2 après création de la branche feature1
622ce47 ajout de f2 après création de la branche feature1
a78bb72 (HEAD -> feature1, tag: v1.0) modif ligne2 sur branche wip
d55628f modif ligne2 sur branche master
c508d3b Afficher en masjuscule!
ee4c4c4 Merge branch 'wip'
7609899 (tag: v0.4) modif hello.py - ajout fonction ecrireXfois
10b9eb9 Ajout de f1
8a2104e Modification de README.md
a6fc769 Ajout .gitignore + ignorer le dossier __pycache__
63c5fef Add hello.py
bbc0000 initialistion du projet
brahim@Training:~/TP1/tppgit-local$
```

```
brahim@Training:~/TP1/tppgit-local$ git cherry-pick 87fad06
CONFLIT (modification/suppression) : f2 supprimé dans HEAD et modifié dans 87fad06 (modif de f2 après création de la branche feature1). Version 87fad06 (modif de f2 après création de la branche feature1) de f2 laissée dans l'arbre.
error: impossible d'appliquer 87fad06... modif de f2 après création de la branche feature1
astuce: Après résolution des conflits, marquez-les avec
astuce: "git add/rm <spéc-de-réf>", puis lancez
astuce: "git cherry-pick --continue".
astuce: Vous pouvez aussi sauter ce commit avec "git cherry-pick --skip".
astuce: Pour arrêter et revenir à l'état antérieur à "git cherry-pick",,
astuce: lancez "git cherry-pick --abort".
```

- Résoudre le conflit et continuer comme indiquer dans le message d'erreur. Vérifier le résultat en affichant l'historique.