

TP 4 : GIT

SOMMAIRES

PREMIERS PAS AVEC GIT	3
Question 2.1 :	3
ADD ET COMMIT	4
Question 2.2 :	4
Question 2.3 :	4
Question 2.4 et question 2.5 :	5
Question 2.6 :	6
Question 2.7 :	7
Question 2.8 :	8
VOYAGE DANS LE TEMPS	10
Question 2.9 :	10
Question 2.10 :	11
Question 2.11 :	11
Question 2.12 :	12
CONCLUSION	12

PREMIERS PAS AVEC GIT

Question 2.1 :

Pour initialiser un nouveau répertoire, je vais effectuer la commande :

- **Git init** nom du répertoire

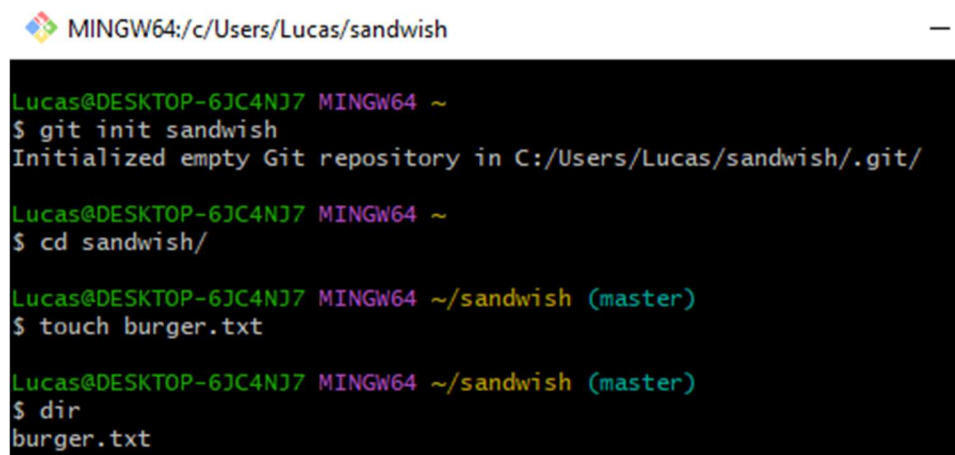
Je vais ensuite me déplacer dans mon nouveau dossier, ici « sandwich » qui est le master.

- **Cd** sandwich/

Puis je vais créer un fichier dans ce répertoire.

- **Touch** nom du fichier.extension

Grace à la commande **DIR**, je peux voir les éléments présents dans mon dossier.



```
MINGW64:/c/Users/Lucas/sandwich

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~
$ git init sandwich
Initialized empty Git repository in C:/Users/Lucas/sandwich/.git/

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~
$ cd sandwich/

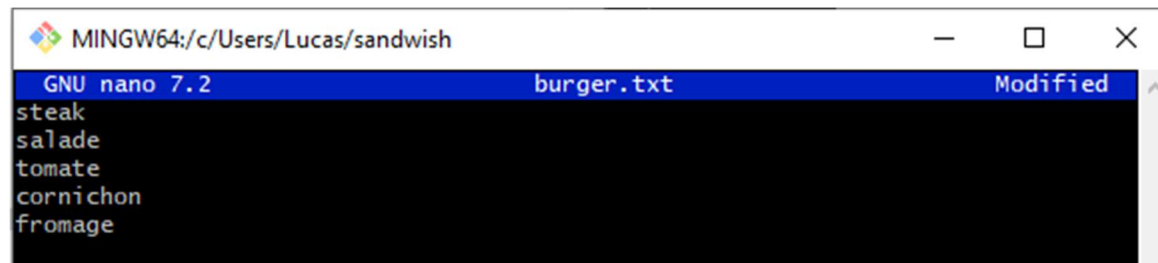
Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwich (master)
$ touch burger.txt

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwich (master)
$ dir
burger.txt
```

Pour modifier mon fichier et ajouter les ingrédients, je vais utiliser :

- **Nano** nom du fichier.extension

Sans oublier d'enregistrer avec CTRL+X puis Y et entrée.



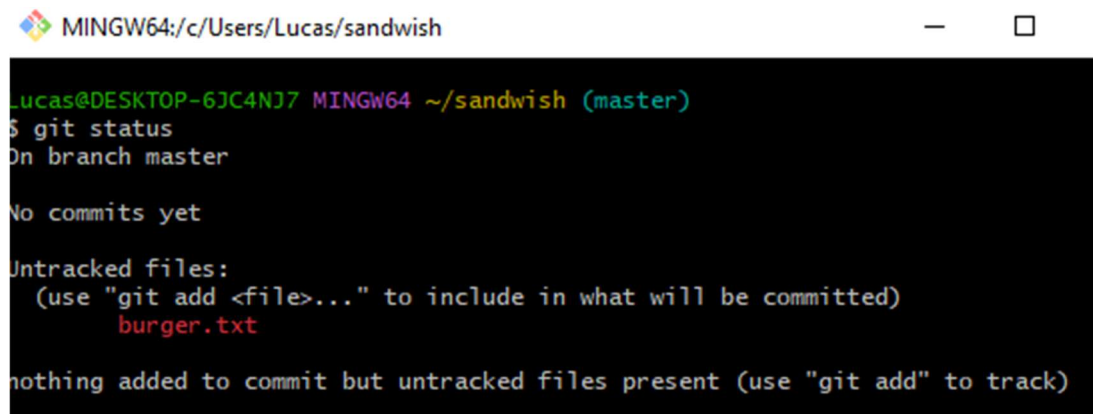
```
MINGW64:/c/Users/Lucas/sandwich

GNU nano 7.2 burger.txt Modified
steak
salade
tomate
cornichon
fromage
```

ADD ET COMMIT

Question 2.2 :

Grace à **git status** je vais pouvoir voir dans quel état est ma modification. Précédemment j'ai créé un fichier burger.txt. On voit donc sur la capture d'écran qu'on est dans la brancher MASTER, qu'il n'y a pas eu de commit et qu'il y a un fichier non déposé. (Section : Untracked files). Il est donc seulement présent dans la « working copy » ce qui équivaut à ma fenêtre de travail.



```
MINGW64:/c/Users/Lucas/sandwish

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwish (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        burger.txt

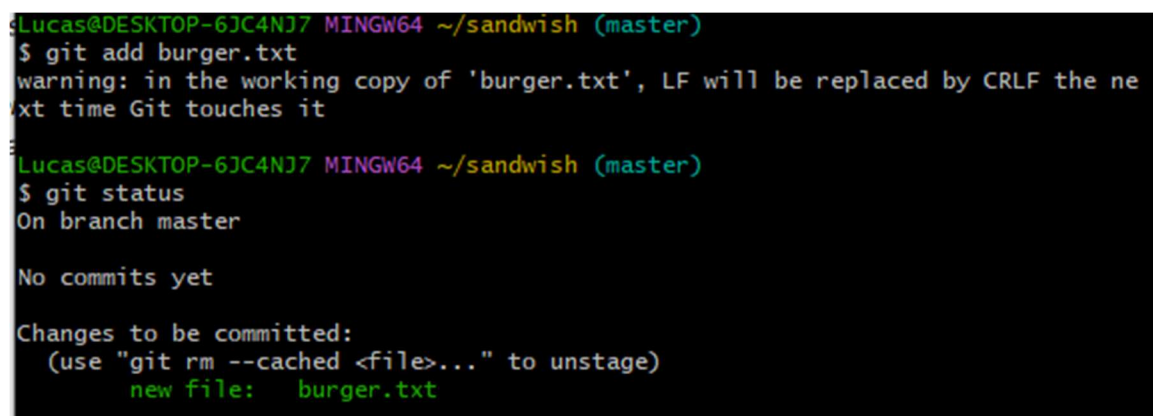
nothing added to commit but untracked files present (use "git add" to track)
```

Question 2.3 :

Je vais ici « envoyer » mon burger.txt dans l'index grâce à la commande :

- **Git add** *nom du fichier.extension*

Avec **git status**, on voit qu'un changement a été effectué (section : Changes to be committed : new files). Cette fois le fichier est en attente d'être « commité » donc d'être sauvegardé finalement.



```
MINGW64:/c/Users/Lucas/sandwish

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwish (master)
$ git add burger.txt
warning: in the working copy of 'burger.txt', LF will be replaced by CRLF the next time Git touches it

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwish (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   burger.txt
```

Grace à la commande :

- **Git diff --cached**

Je peux voir les modifications qui vont être ajoutée de l'index au répertoire final. Ici ce sera les ingrédients, écrit à la première étape.

```
Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwish (master)
$ git diff --cached
diff --git a/burger.txt b/burger.txt
new file mode 100644
index 0000000..3788869
--- /dev/null
+++ b/burger.txt
@@ -0,0 +1,5 @@
+steak
+salade
+tomate
+cornichon
+fromage
```

Question 2.4 et question 2.5 :

Finalement avec la commande :

- **Git commit -m « <mon message> »**

Je vais ici grâce à cette commande envoyé mes modifications de l'index vers mon répertoire, en ajoutant un message pour pouvoir retrouver mes changements ultérieurement.

```
Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwish (master)
$ git commit -m "<j'ai ajoute les ingredients du sandiwsh>"
[master (root-commit) bd7793a] <j'ai ajoute les ingredients du sandiwsh>
1 file changed, 5 insertions(+)
create mode 100644 burger.txt

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwish (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Avec **git status**, je vais vérifier si je n'ai plus de modification en attente. Ici c'est le cas.

Question 2.6 :

Avec la commande :

- **Git log**

On peut voir tous les changements effectués dans mon répertoire. Ici on voit qu'il y a eu un commit par Lucas, le 20/02/2024 à 17h21 « j'ai ajoute les ingrédients du sandwich ». Il y a donc une seule modification et son ID en SHA1 c'est

- « Bd7793a0e9166e495c7d74c5f81f925182156409 »

```
Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwish (master)
$ git log
commit bd7793a0e9166e495c7d74c5f81f925182156409 (HEAD -> master)
Author: Lucas <test@test.fr>
Date: Tue Feb 20 17:21:53 2024 +0100

    <j'ai ajoute les ingredients du sandiwsh>
```

Question 2.7 :

Je vais créer un fichier kebab.txt dans le répertoire sandwich.

Grace à **git status**, je peux voir que mon fichier kebab.txt est en attente d'être inclus à l'index.

Je vais donc **git add** et pour vérifier une nouvelle fois grâce à **git status**, on voit qu'il est actuellement en attente d'être commité.

```
MINGW64/c/Users/Lucas/sandwich

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwich (master)
$ touch kebab.txt

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwich (master)
$ nano kebab.txt

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwich (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    kebab.txt

nothing added to commit but untracked files present (use "git add" to track)

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwich (master)
$ git add kebab.txt

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwich (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   kebab.txt

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwich (master)
$ git commit -m "<fichier kebab>"
[master 91a76fd] <fichier kebab>
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 kebab.txt

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwich (master)
$ git log
commit 91a76fdaa9837ca13825e511c20c823bb9be03ce (HEAD -> master)
Author: Lucas <test@test.fr>
Date:   Tue Feb 20 17:35:38 2024 +0100

    <fichier kebab>

commit d0dbac20e41e6ccad96a939fb09398fa3658a191
Author: Lucas <test@test.fr>
Date:   Tue Feb 20 17:28:41 2024 +0100

    hot_dog.txt

commit bd7793a0e9166e495c7d74c5f81f925182156409
Author: Lucas <test@test.fr>
Date:   Tue Feb 20 17:21:53 2024 +0100

    <j'ai ajoute les ingredients du sandiwsh>
```

Une fois commité, grâce à **git log**, on peut voir que mon fichier kebab a été ajouter dans mon répertoire.

Question 2.8 :

Grace a **git status** on voit qu'il n'y a actuellement rien à commit.

```
Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwish (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Plusieurs interfaces sont disponibles, les voici ;

L'interface avec git log :

```
MINGW64:/c/Users/Lucas/sandwish

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwish (master)
$ git log
commit 91a76fdaa9837ca13825e511c20c823bb9be03ce (HEAD -> master)
Author: Lucas <test@test.fr>
Date: Tue Feb 20 17:35:38 2024 +0100

    <fichier kebab>

commit d0dbac20e41e6ccad96a939fb09398fa3658a191
Author: Lucas <test@test.fr>
Date: Tue Feb 20 17:28:41 2024 +0100

    hot_dog.txt

commit bd7793a0e9166e495c7d74c5f81f925182156409
Author: Lucas <test@test.fr>
Date: Tue Feb 20 17:21:53 2024 +0100

    <j'ai ajoute les ingredients du sandiwsh>

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwish (master)
$ |
```

L'interface avec git log --graph --pretty=short ;

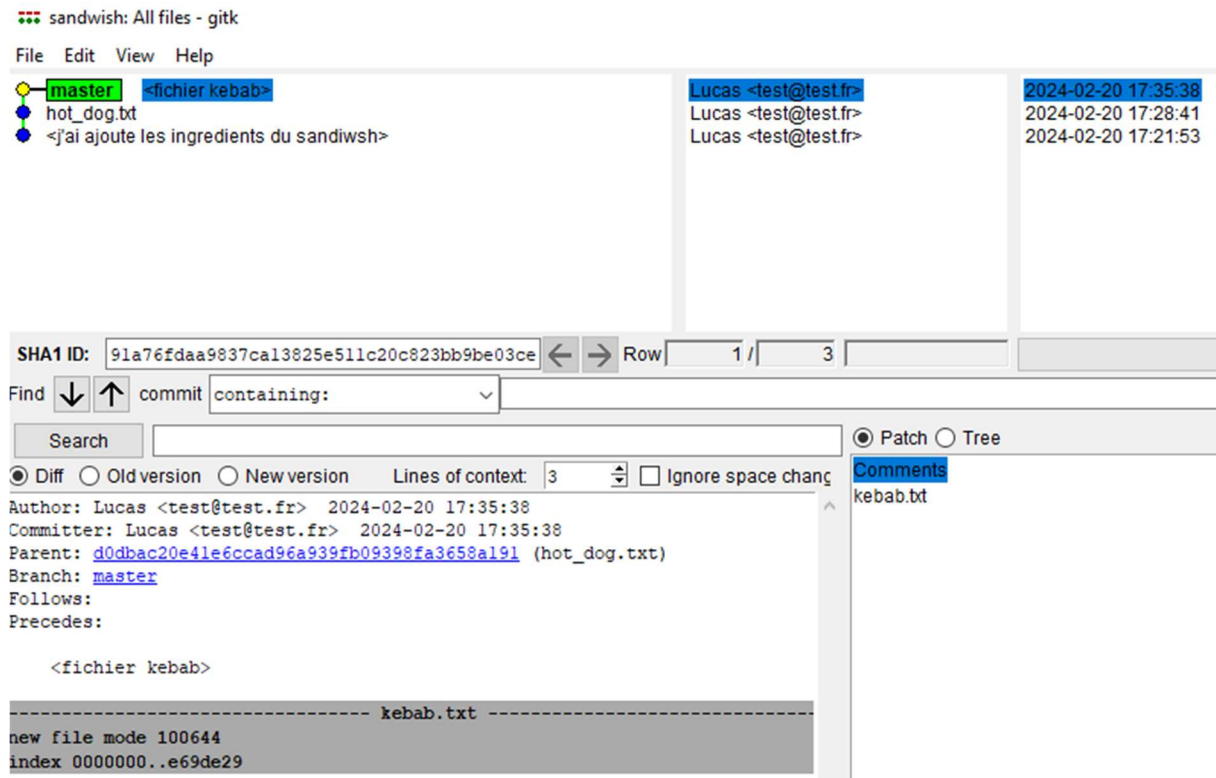
Il n'y a pas grande différence.

```
MINGW64:/c/Users/Lucas/sandwish

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwish (master)
$ git log --graph --pretty=short
* commit 91a76fdaa9837ca13825e511c20c823bb9be03ce
| Author: Lucas <test@test.fr>
|
|    <fichier kebab>
|
| * commit d0dbac20e41e6ccad96a939fb09398fa3658a191
| | Author: Lucas <test@test.fr>
| |
| |    hot_dog.txt
| |
| * commit bd7793a0e9166e495c7d74c5f81f925182156409
| | Author: Lucas <test@test.fr>
| |
| |    <j'ai ajoute les ingredients du sandiwsh>
```


L'interface avec gitk ;

Qui est une interface graphique



Quant à gitg, la commande n'a pas été trouvée, je suppose que c'est donc un téléchargement supplémentaire.

VOYAGE DANS LE TEMPS

Question 2.9 :

Je vais modifier mon fichier *kebab.txt* puis l'ajouter dans l'index avec **git add**. On voit avec **git status** qu'il est bien en attente.

```
Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwich (master)
$ dir
burger.txt hot_dog.txt kebab.txt

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwich (master)
$ nano kebab.txt

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwich (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   kebab.txt

no changes added to commit (use "git add" and/or "git commit -a")

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwich (master)
$ git add kebab.txt
warning: in the working copy of 'kebab.txt', LF will be replaced by CRLF the next time Git touches it

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwich (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   kebab.txt
```

Ici je vais réaliser la commande :

- **Git reset** nom du fichier.extension

Grace a cette commande je vais pouvoir passer de l'index à mon working copy. Avec **git status**, je confirme donc que mon fichier a bien été déplacé de l'index à ma working copy.

```
Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwich (master)
$ git reset kebab.txt
Unstaged changes after reset:
M      kebab.txt

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwich (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   kebab.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Question 2.10 :

Avec la commande :

- **Git checkout** *nom du fichier.extension*

Je vais pouvoir aller récupérer l'historique d'une ancienne version correspondant au dernier commit réaliser.

```
Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwish (master)
$ git checkout kebab.txt
Updated 1 path from the index

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwish (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Question 2.11 :

Avec la commande :

- **Git checkout** *COMMITID*

Je vais pouvoir me déplacer dans la branche de ce commit. Je suis donc plus dans le master.

Je peux donc effectuer des modifications puis commit pour enregistrer dans mon répertoire.

```
Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwish (master)
$ git log
commit 91a76fdaa9837ca13825e511c20c823bb9be03ce (HEAD -> master)
Author: Lucas <test@test.fr>
Date: Tue Feb 20 17:35:38 2024 +0100

    <fichier kebab>

commit d0dbac20e41e6ccad96a939fb09398fa3658a191
Author: Lucas <test@test.fr>
Date: Tue Feb 20 17:28:41 2024 +0100

    hot_dog.txt

commit bd7793a0e9166e495c7d74c5f81f925182156409
Author: Lucas <test@test.fr>
Date: Tue Feb 20 17:21:53 2024 +0100

    <j'ai ajoute les ingredients du sandiwh>

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwish (master)
$ git checkout ^C
Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwish (master)
$ git checkout d0dbac20e41e6ccad96a939fb09398fa3658a191
Note: switching to 'd0dbac20e41e6ccad96a939fb09398fa3658a191'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at d0dbac2 hot_dog.txt

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwish ((d0dbac2...))
$ git log
commit d0dbac20e41e6ccad96a939fb09398fa3658a191 (HEAD)
Author: Lucas <test@test.fr>
Date: Tue Feb 20 17:28:41 2024 +0100

    hot_dog.txt

commit bd7793a0e9166e495c7d74c5f81f925182156409
Author: Lucas <test@test.fr>
Date: Tue Feb 20 17:21:53 2024 +0100

    <j'ai ajoute les ingredients du sandiwh>
```

Question 2.12 :

Pour revenir dans le master j'effectue la commande :

- **Git checkout master**

Je vais donc me déplacer du COMMITID à master.

```
Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwich ((d0dbac2...))
$ git checkout master
Previous HEAD position was d0dbac2 hot_dog.txt
Switched to branch 'master'

Lucas@DESKTOP-6JC4NJ7 MINGW64 ~/sandwich (master)
$ .....
```

CONCLUSION

Au début de ce TP, j'ai honnêtement eu du mal à voir le potentiel de cette technique de travail, et surtout du mal à comprendre le fonctionnement fondamental de GIT. En avançant et en discutant autour de moi, j'ai pu appréhender et l'utiliser assez fluidement. Je me doute bien que nous avons survolé de très haut le potentiel de GIT avec ce TP. Mais ça pousse forcément à en savoir davantage.