TP Git :

0.Prérequis :

- Installer Git

Vous pouvez installer Git via le site https://git-scm.com/downloads

1.Introduction :

Git est un logiciel de gestion de versions décentralisé. C'est un logiciel libre créé par Linus Torvalds, auteur du noyau Linux, et distribué selon les termes de la licence publique générale GNU version 2. En 2016, il s'agit du logiciel de gestion de versions le plus populaire qui est utilisé par plus de douze millions de personnes.

— https://fr.wikipedia.org/wiki/Git

Lors cette première séance du TPs, nous allons nous initier à l'utilisation du Git et puis à la maintenance d'un répertoire de travail de manière structurée et ordonné.

2.Premiers pas avec Git :

Git [4] est un logiciel de contrôle de versions, il permet de sauvegarder l'historique du contenu d'un répertoire de travail. Pour ce faire l'utilisateur doit régulièrement enregistrer (en créant une révision ou commit) les modifications apportées au répertoire, il pourra ensuite accéder à l'historique de toutes les modifications et inspecter l'état du dossier à chaque révision.

Git à la particularité de permettre de créer une copie d'un répertoire de travail, working copy, et de synchroniser entre eux plusieurs copies du même répertoire, permettant la décentralisation du travail.

De plus, Git permet d'utiliser une ou plusieurs branches de développement et de fusionner entre elles ces branches.

2.1. Création d’un nouveau dépôt :

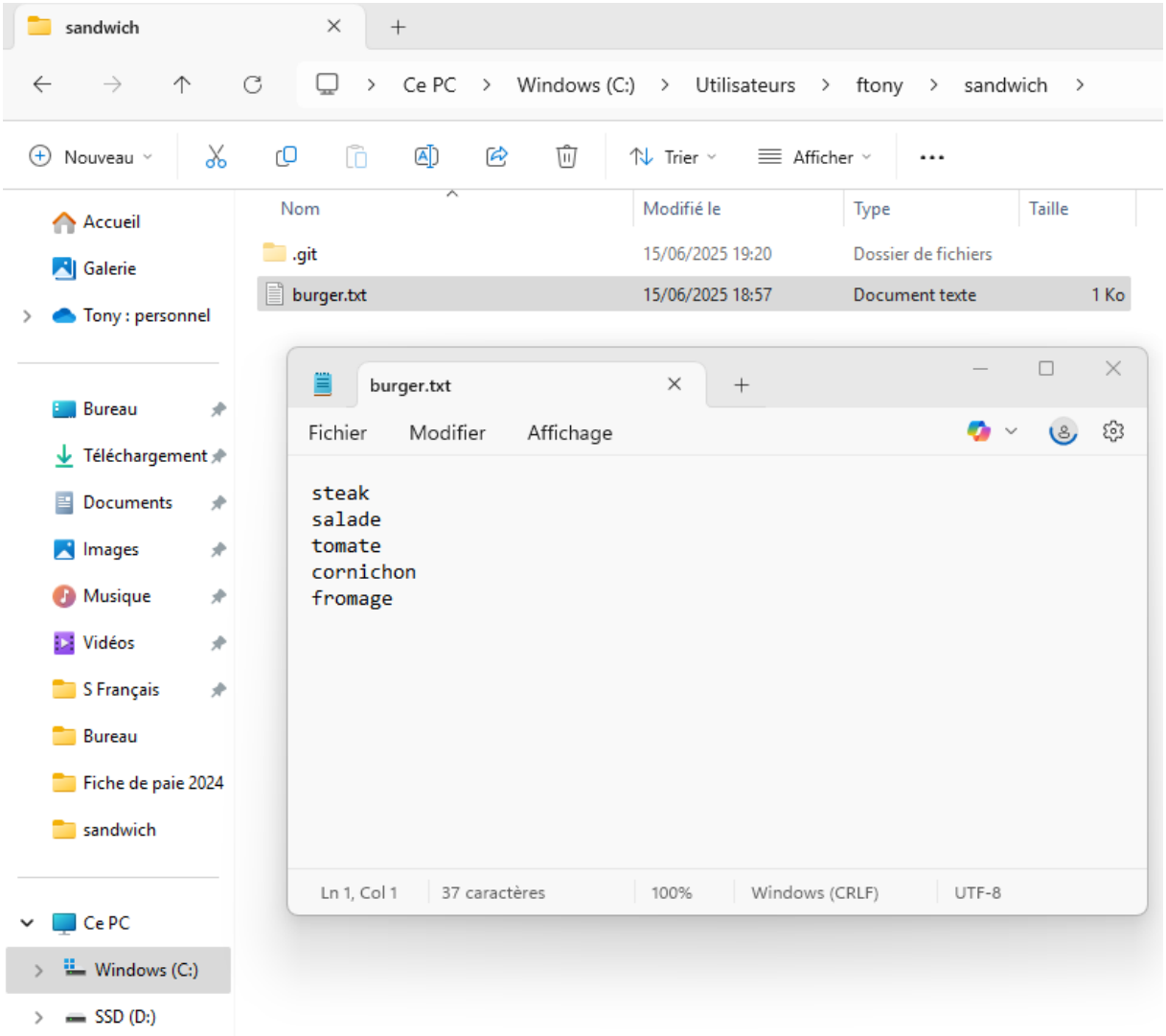
Question 2.1.

Je créer donc un nouveau dépôt Git dans un répertoire sandwich avec la commande : **git init sandwich**

Par défaut le dépôt se créer à la racine de mon dossier utilisateur.

```
ftony@Omen-Shadow MINGW64 ~
$ git init sandwich
Initialized empty Git repository in C:/Users/ftony/sandwich/.git/
```

Puis dans mon nouveau répertoire, je viens créer un fichier texte burger.txt qui contient la liste des ingrédients d’un burger.



2.2 Add et Commit :

Question 2.2.

Je me rends dans mon répertoire fraichement créer via la commande : **cd sandwich**

Puis je fais la commande : **git status**

Je peux observer grâce à **git status** que mon fichier burger.txt existe mais n'est pas inclus dans l'index de mon dépôt.

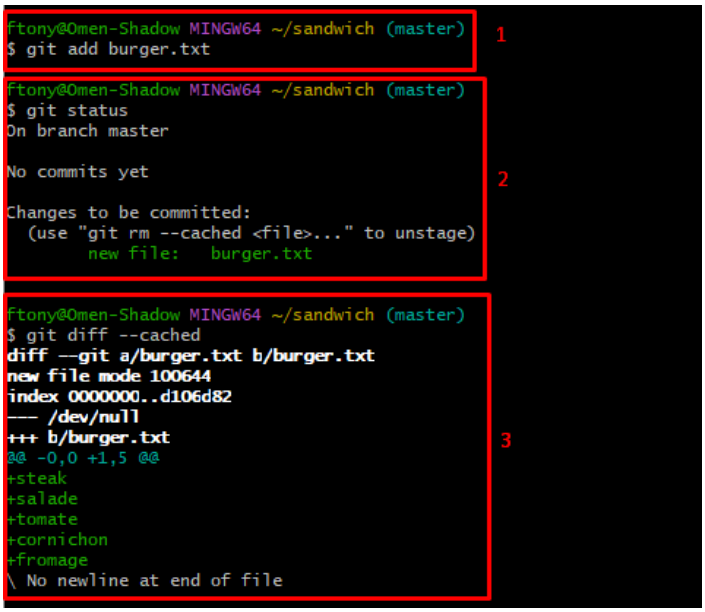
```
ftony@Omen-Shadow MINGW64 ~
$ cd sandwich

ftony@Omen-Shadow MINGW64 ~/sandwich (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.

1/ Pour inclure notre fichier dans l'index nous allons utiliser la commande : **git add burger.txt**

2/ Nous pouvons vérifier que le fichier est bien inclus dans l'index à l'aide de : **git status**

3/ A l'aide de la commande : **git diff --cached** je peux observer la différence entre le fichier actuel dans l'index et la version commit dans le repository (qui n'existe pas encore).



Question 2.4. & 2.5. :

1/ Je peux maintenant commit à l'aide de la commande : **git commit -m « MESSAGE »**

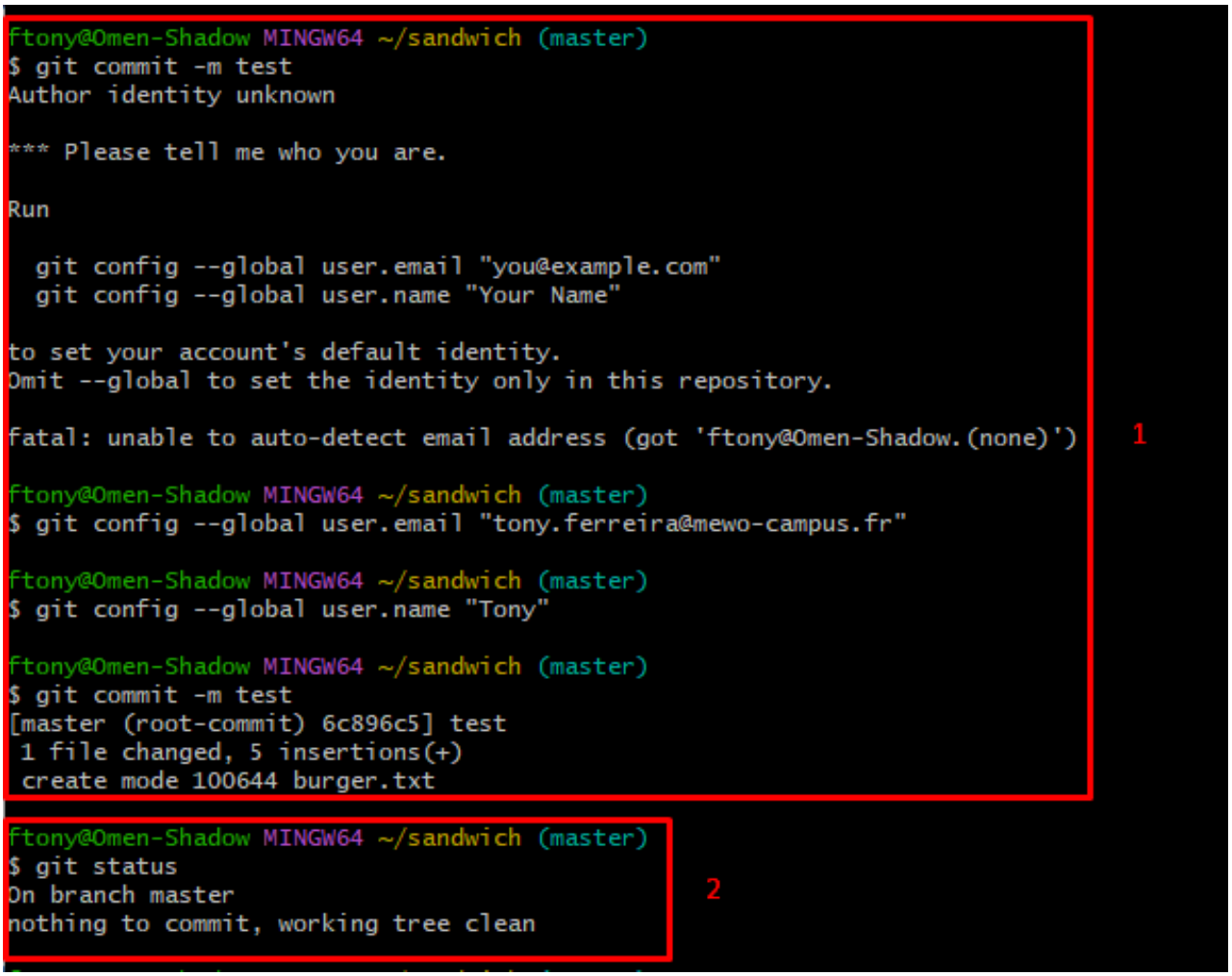
Lors de mon 1er commit, git ne sait pas qui je suis, je dois alors me déclarer avec les commandes :

git config --global user.email "MonEmail"

git config --global user.name "MonNom"

Puis je peux créer mon 1er commit avec : **git commit -m test**

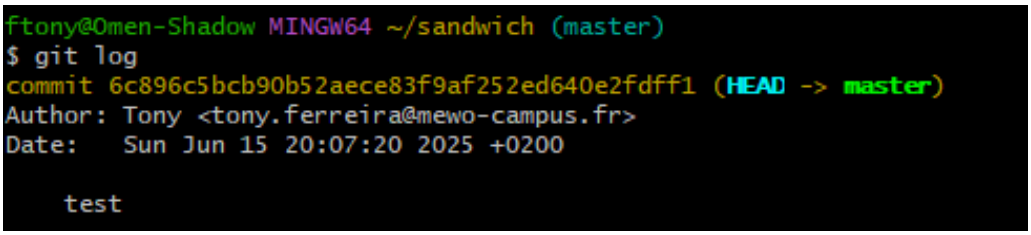
2/ J'exécute à nouveau ma commande **git status** et je peux observer que mon travail est à jour.



Question 2.6.

J'utilise maintenant la commande **git log** afin d'afficher la liste des changements effectués dans ce dépôt.

On peut observer 1 changement (1 commit) fait par « Tony » ainsi que son ID sous format SHA1 6c896c5bcb90b52aece83f9af252ed640e2fdff1

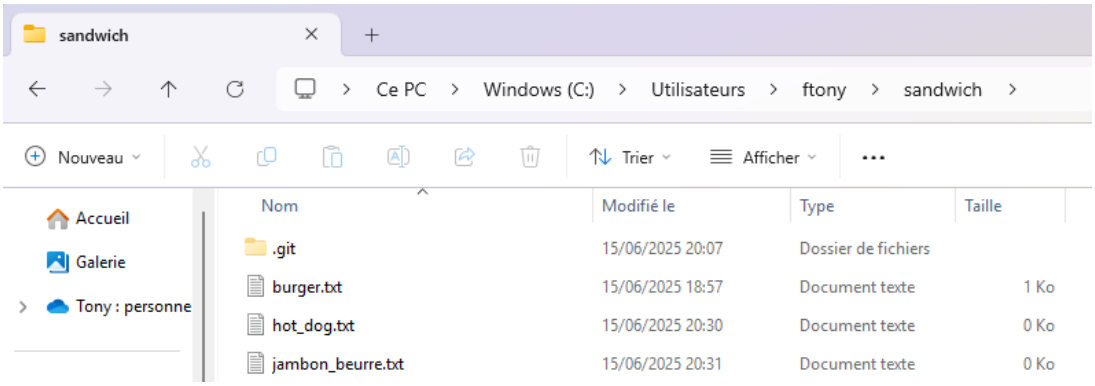


Question 2.7.

Je vais créer 2 autres sandwiches et modifier la composition de burger.txt, à la toute fin j'ajouterai aussi un ingrédient supplémentaire dans hot_dog.txt.

Bien évidemment je fais 1 commit à chaque fois que je créer / modifie un fichier.

Cela me donnera un total de 5 commits de ce répertoire.



Ici je peux utiliser la commande **git diff** avant de faire **git add** pour voir ce que je vais ajouter à l'index.

```
ftony@Omen-Shadow MINGW64 ~/sandwich (master)
$ git diff
diff --git a/burger.txt b/burger.txt
index d106d82..5b7dbe5 100644
--- a/burger.txt
+++ b/burger.txt
@@ -2,4 +2,6 @@ steak
 salade
 tomate
 cornichon
-fromage
\ No newline at end of file
+fromage
+poivre
+sel
\ No newline at end of file
```

La commande **git diff --cached** est à utiliser après **git add** pour voir ce que je peux commit.

```
ftony@Omen-Shadow MINGW64 ~/sandwich (master)
$ git diff --cached
diff --git a/burger.txt b/burger.txt
index d106d82..5b7dbe5 100644
--- a/burger.txt
+++ b/burger.txt
@@ -2,4 +2,6 @@ steak
 salade
 tomate
 cornichon
-fromage
\ No newline at end of file
+fromage
+poivre
+sel
\ No newline at end of file
```

Question 2.8.

Avec la commande **git log** je peux vérifier mes 5 commits.

Puis avec **git status** je peux vérifier qu'il ne me reste rien à sauvegarder.

```
ftony@Omen-Shadow MINGW64 ~/sandwich (master)
$ git status
On branch master
nothing to commit, working tree clean

ftony@Omen-Shadow MINGW64 ~/sandwich (master)
$ git log
commit 931987e2f73e6ff0576c333a0522aa1171543348 (HEAD -> master)
Author: Tony <tony.ferreira@mewo-campus.fr>
Date:   Sun Jun 15 20:56:49 2025 +0200

    Ajout moutarde a hot_dog.txt

commit 38f7624cb4a382d3302a32c8d6770f6de5a5cced
Author: Tony <tony.ferreira@mewo-campus.fr>
Date:   Sun Jun 15 20:54:36 2025 +0200

    Creation jambon_beurre.txt

commit e4b39a106ac3fae7bd801ccf9e01154c0a4197c9
Author: Tony <tony.ferreira@mewo-campus.fr>
Date:   Sun Jun 15 20:53:53 2025 +0200

    Creation hot_dog.txt

commit a74eb1d73277c6fd4f4a35eab951037cac3de6fb
Author: Tony <tony.ferreira@mewo-campus.fr>
Date:   Sun Jun 15 20:47:55 2025 +0200

    Ajout ingredients burger.txt

commit 6c896c5bcb90b52aece83f9af252ed640e2fdff1
Author: Tony <tony.ferreira@mewo-campus.fr>
Date:   Sun Jun 15 20:07:20 2025 +0200

    test

ftony@Omen-Shadow MINGW64 ~/sandwich (master)
$
```

2.3 Voyage dans le temps :

Question 2.9.

Nous allons faire un test et ajouter un ingrédient à notre **jambon_beurre.txt** puis nous allons **git add jambon_beurre.txt**

Notre modification se trouve alors à mi-chemin, elle a été ajouter à l'index mais pas sauvegarder dans le repository je peux le vérifier à l'aide de **git status**

Maintenant je me retrouve à l'étape du **git commit** mais je ne souhaite pas faire cette modification et surtout pas la sauvegarder.

Je fais alors un **git reset jambon_beurre.txt** et je vérifie à l'aide de **git status**.

Voilà je n'ai plus rien a commit car la modification de mon **jambon_beurre.txt** n'as jamais été **git add** à mon index.

```
ftony@Omen-Shadow MINGW64 ~/sandwich (master)
$ git add jambon_beurre.txt

ftony@Omen-Shadow MINGW64 ~/sandwich (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   jambon_beurre.txt


ftony@Omen-Shadow MINGW64 ~/sandwich (master)
$ git reset jambon_beurre.txt
Unstaged changes after reset:
M       jambon_beurre.txt

ftony@Omen-Shadow 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:   jambon_beurre.txt

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

ftony@Omen-Shadow MINGW64 ~/sandwich (master)
$
```

Question 2.10.

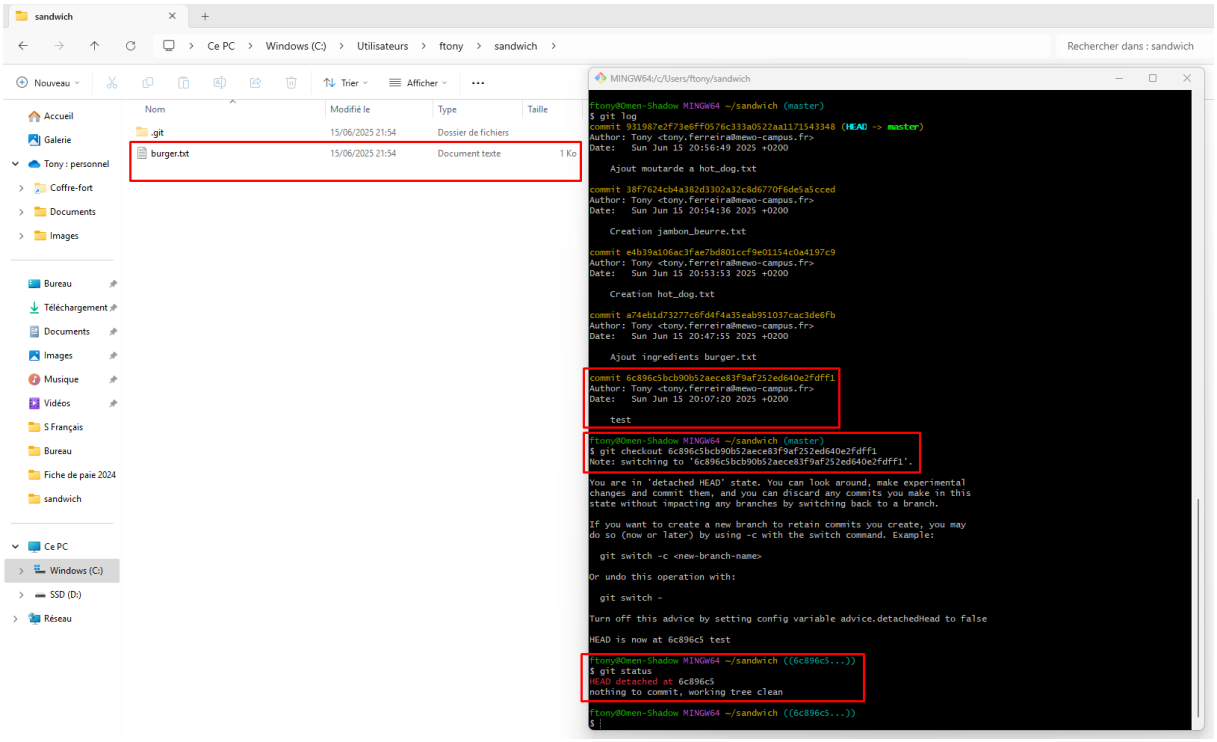
Ma modification a donc été retirer de l'index mais mon fichier est toujours modifié actuellement, pour remédier à ça nous pouvons alors utiliser la commande : **git checkout**

Elle va nous permettre de faire revenir notre fichier **jambon_beurre.txt** au même état que son tout dernier commit.

```
ftony@Omen-Shadow 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:   jambon_beurre.txt

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

ftony@Omen-Shadow MINGW64 ~/sandwich (master)
$ git checkout jambon_beurre.txt
Updated 1 path from the index

ftony@Omen-Shadow MINGW64 ~/sandwich (master)
$ git status
On branch master
nothing to commit, working tree clean
```

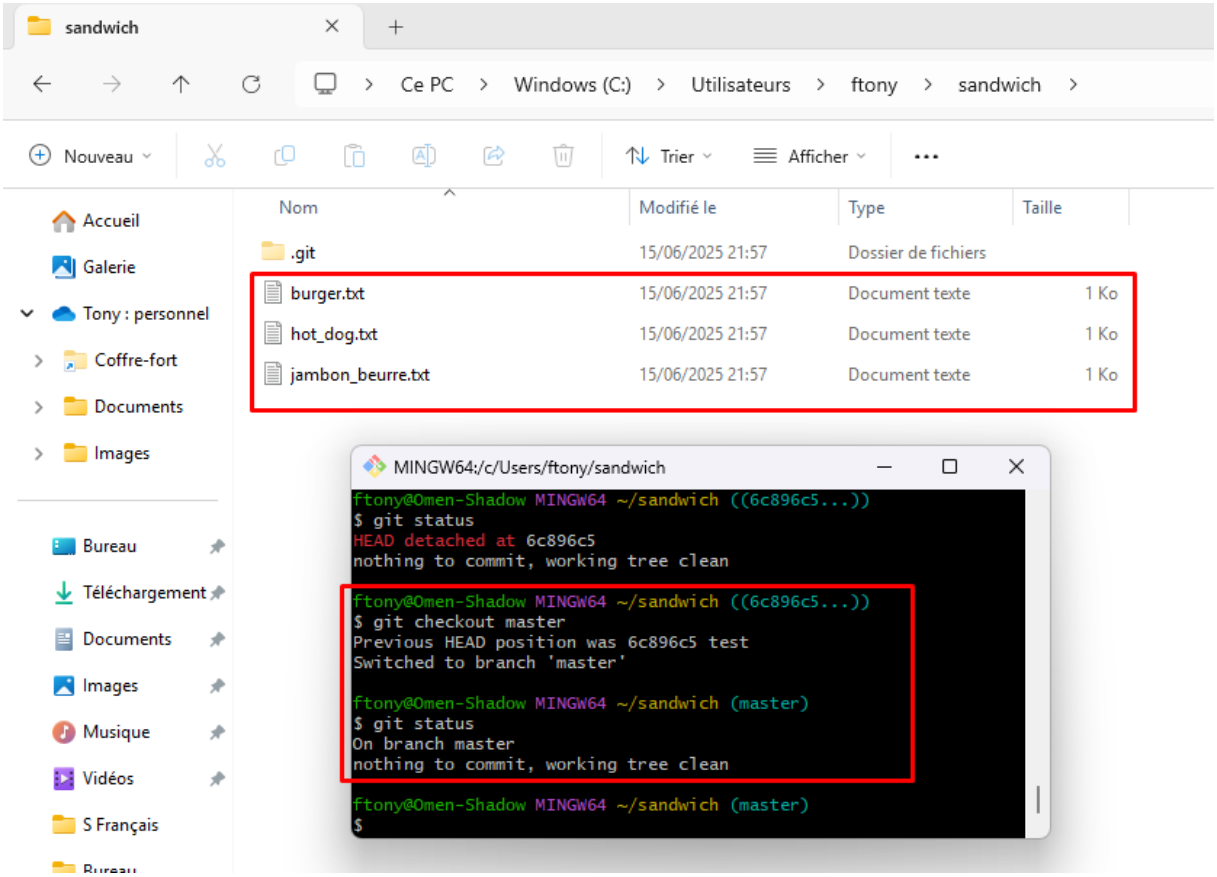
Question 2.11.

Avec la commande **git checkout « ID de mon commit en format SHA1 »** je peux faire un bond dans le temps au moment du commit choisis.



Question 2.12.

Pour revenir au « présent » il me suffit de faire un **git checkout master**



Références :

[1] git cheat sheet. https://services.github.com/on-demand/downloads/github-git-cheat-sheet. pdf.

[2] git cheat sheet interactif. http://ndpsoftware.com/git-cheatsheet.html.

[3] git livre. https://git-scm.com/book/en/v2.

[4] git page d'accueil. https://git-scm.com/.

[5] git tutoriel. https://git-scm.com/docs/gittutorial.