Table des matières

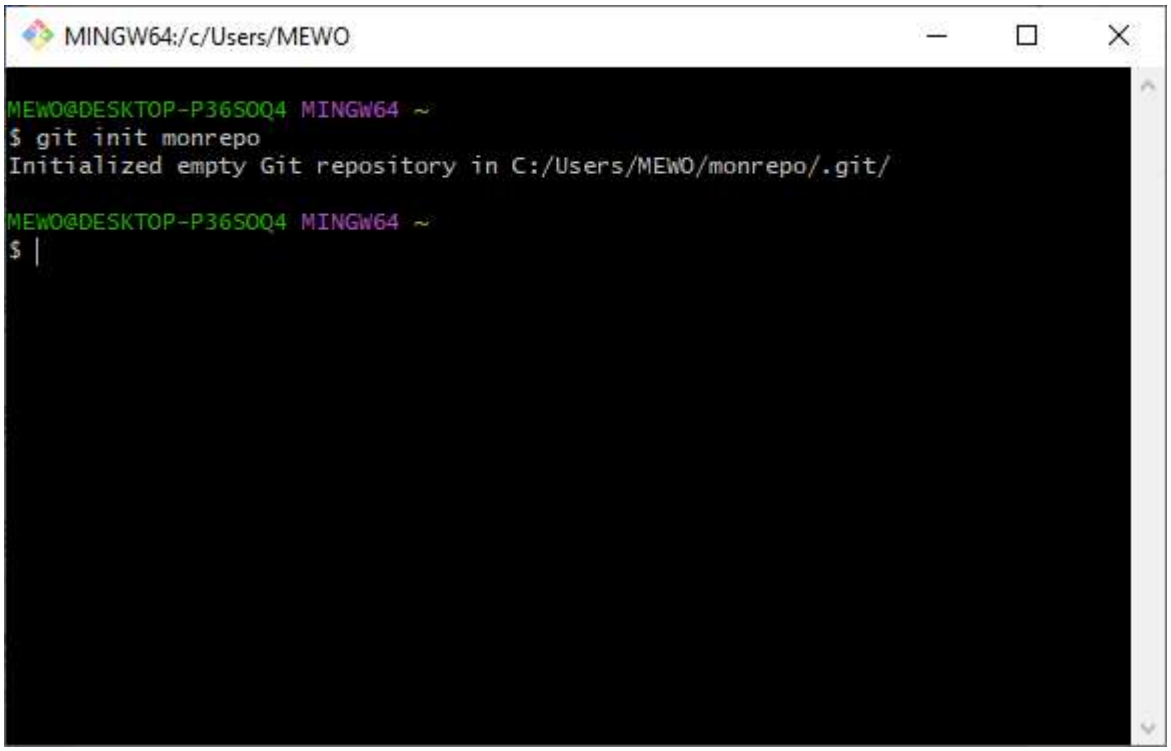
Quesque Git

Git est un logiciel open source qui permet aux développeurs de sauvegarder leurs projets au fil du temps dans un répertoire local.

Git Hub et une version web avec les mêmes fonctionnalités mais qui permet de travaillé en collaboration avec d'autres personnes

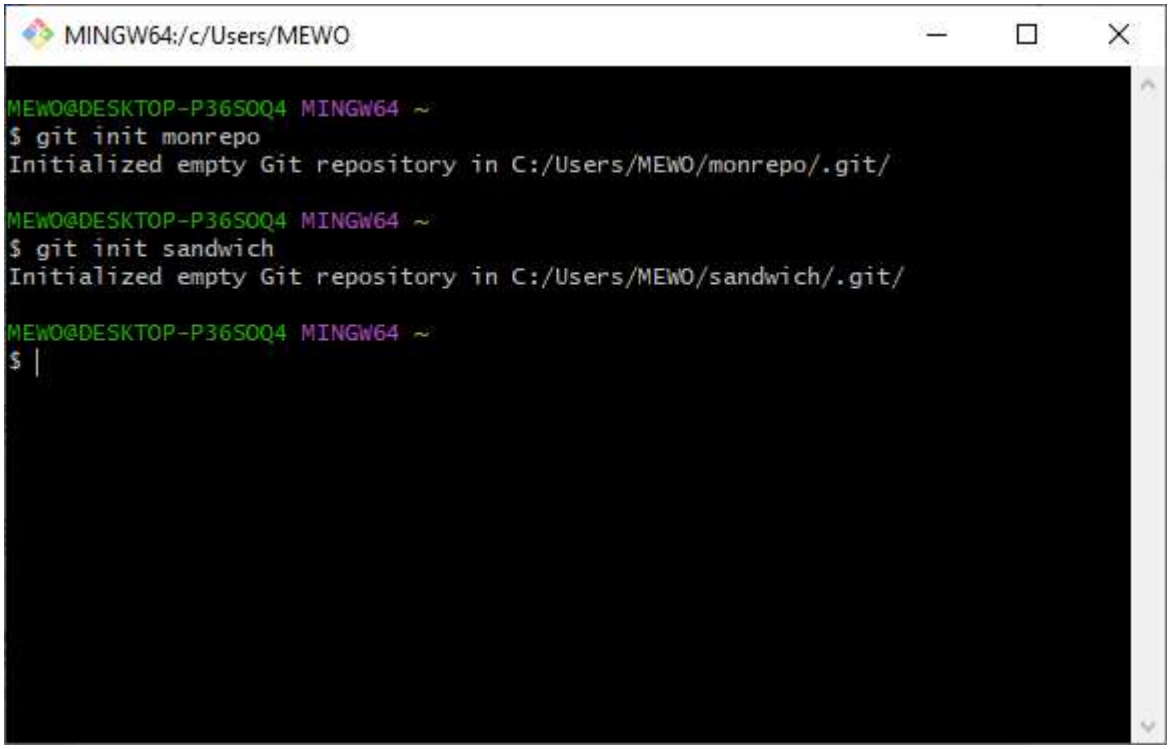
Question 2.0) Création d’un nouveau dépôt

Afin de comprendre la création de dépôt et d’avoir un dossier ou l’on peut sauvegarder notre travail, on va créer un dépôt avec le nom monrepo grâce à la commande **git init monrepo**

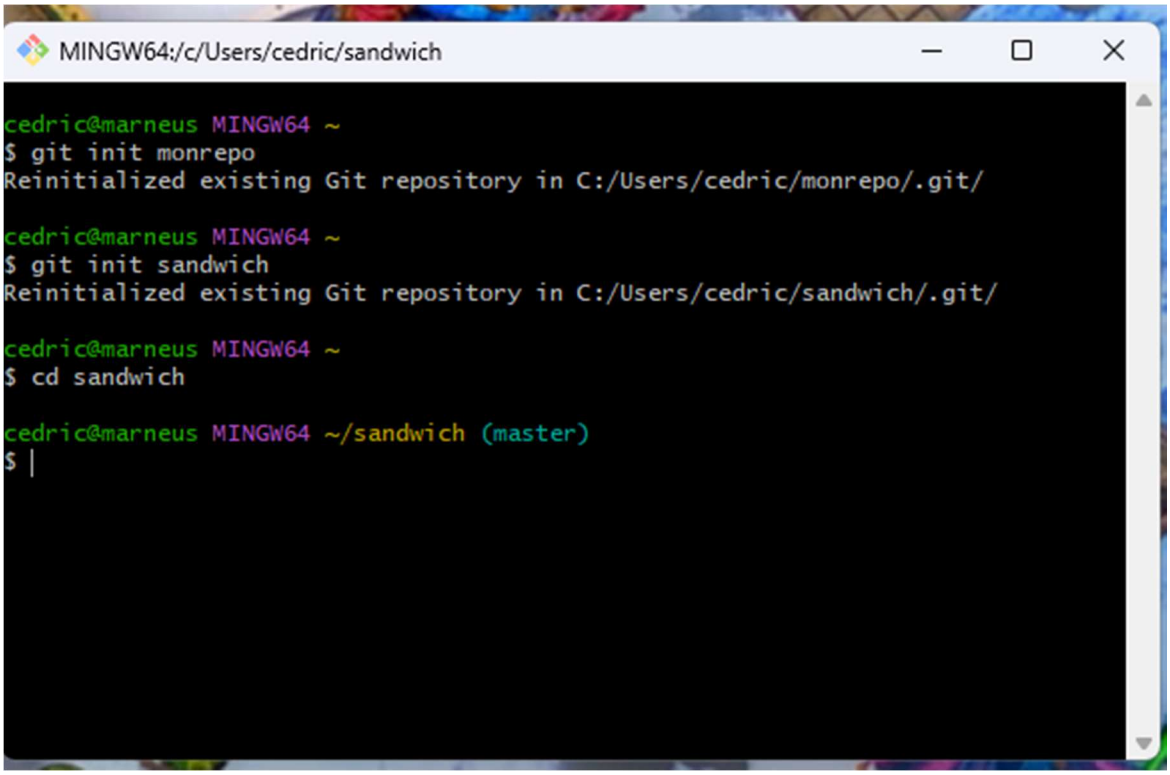


Question 2.1) Initialiser un nouveau dépôt Git dans un répertoire sandwich, et créez le fichier burger.txt qui contient la liste des ingrédients d'un burger, un ingrédient par ligne.

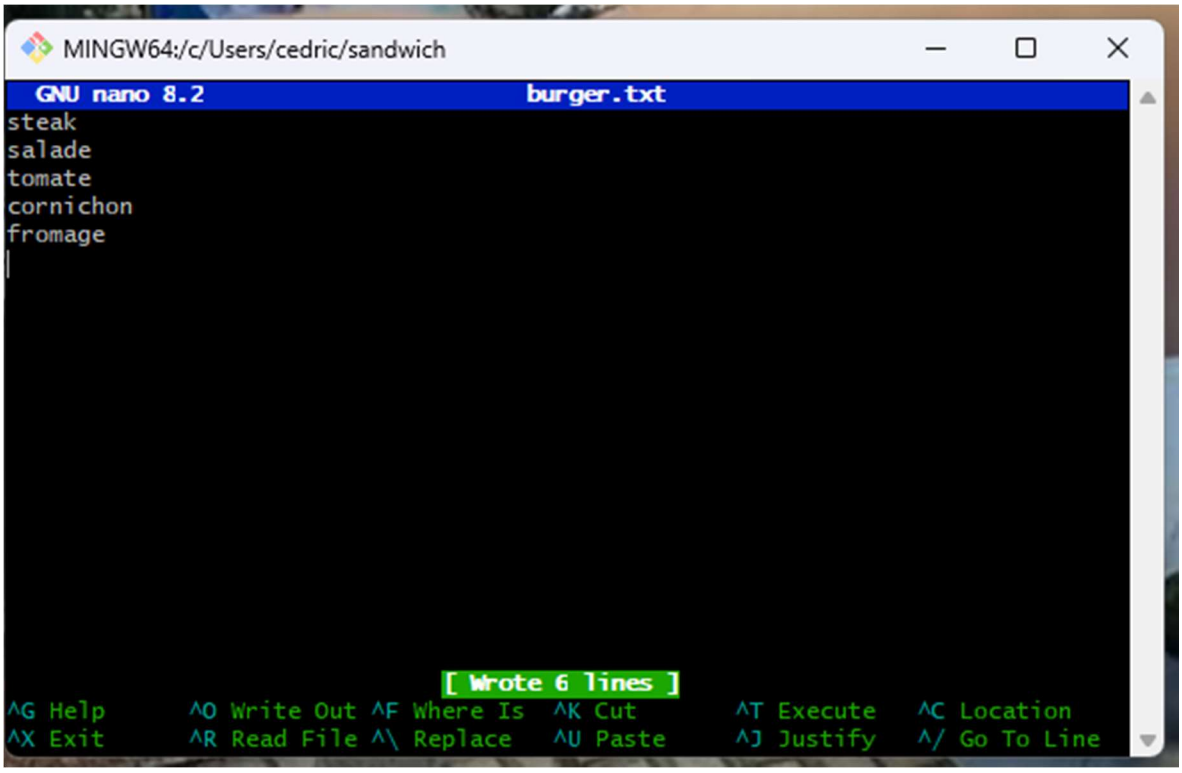
Afin de pouvoir faire le reste du TP je vais créer un dépôt sandwich



Une fois le dépôt j'utilise la commande **cd** afin de rentré dans le dépôt, puis je crée le fichier **burger.txt** à l'aide de la commande **nano**

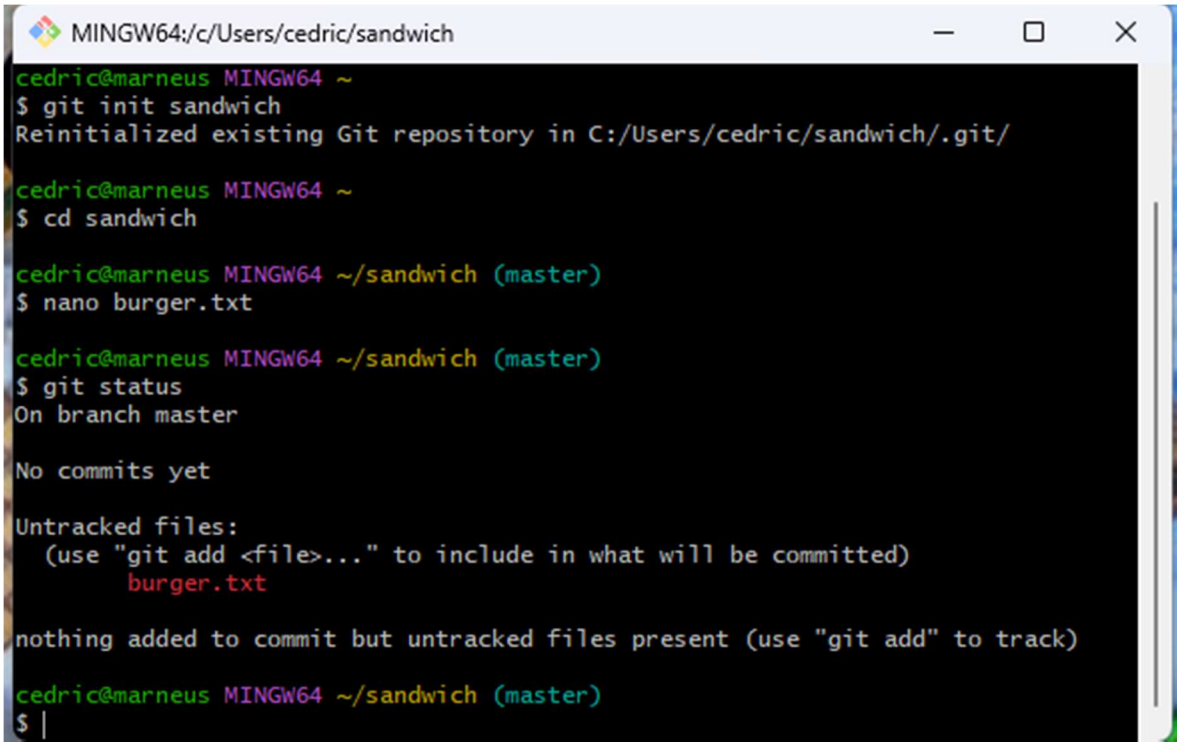


J'ajoute les ingrédients dans **burger.txt,** puis je sauvegarde avec **ctrl O** puis **ctrl X** pour sortir



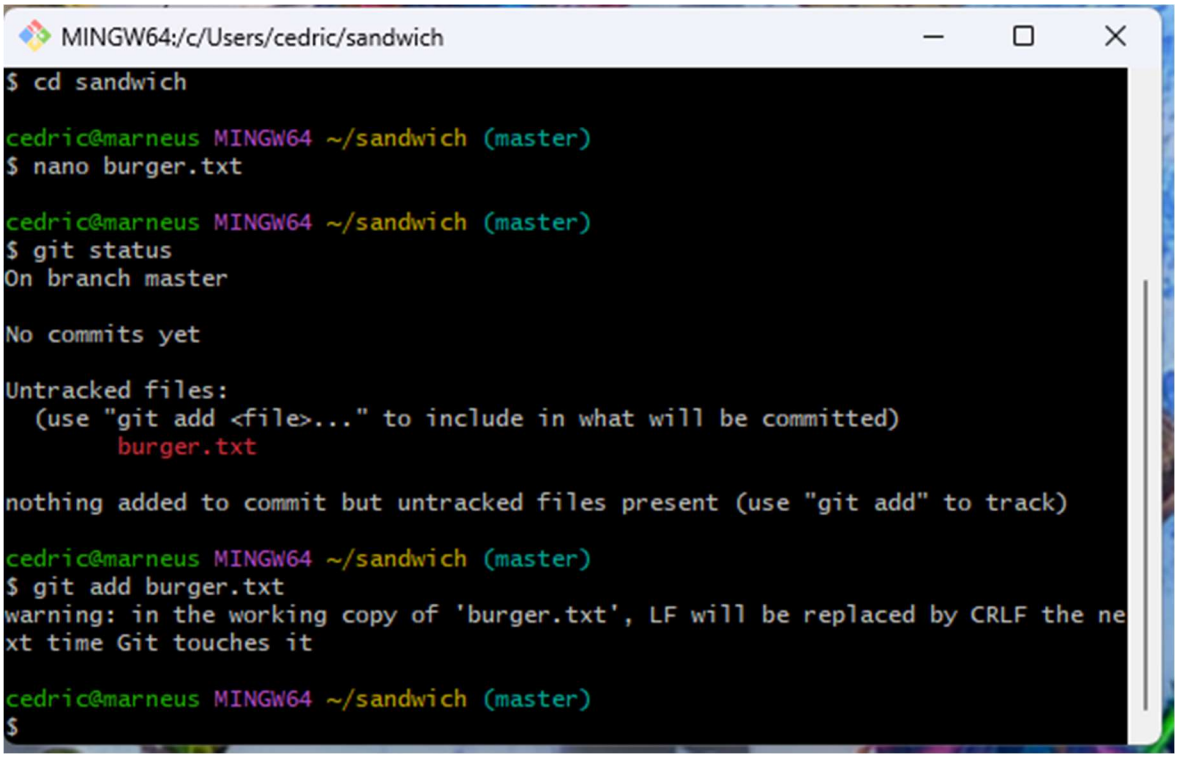
Question 2.2) Vérifiez avec git status l'état dans lequel se trouve votre dépôt. Vos modifications (l'ajout du fichier burger.txt) devraient être présentes seulement dans la copie de travail

Je regarde le statut du fichier via la commande **Git status** et l'on aperçoit qu'il est en copie de travail

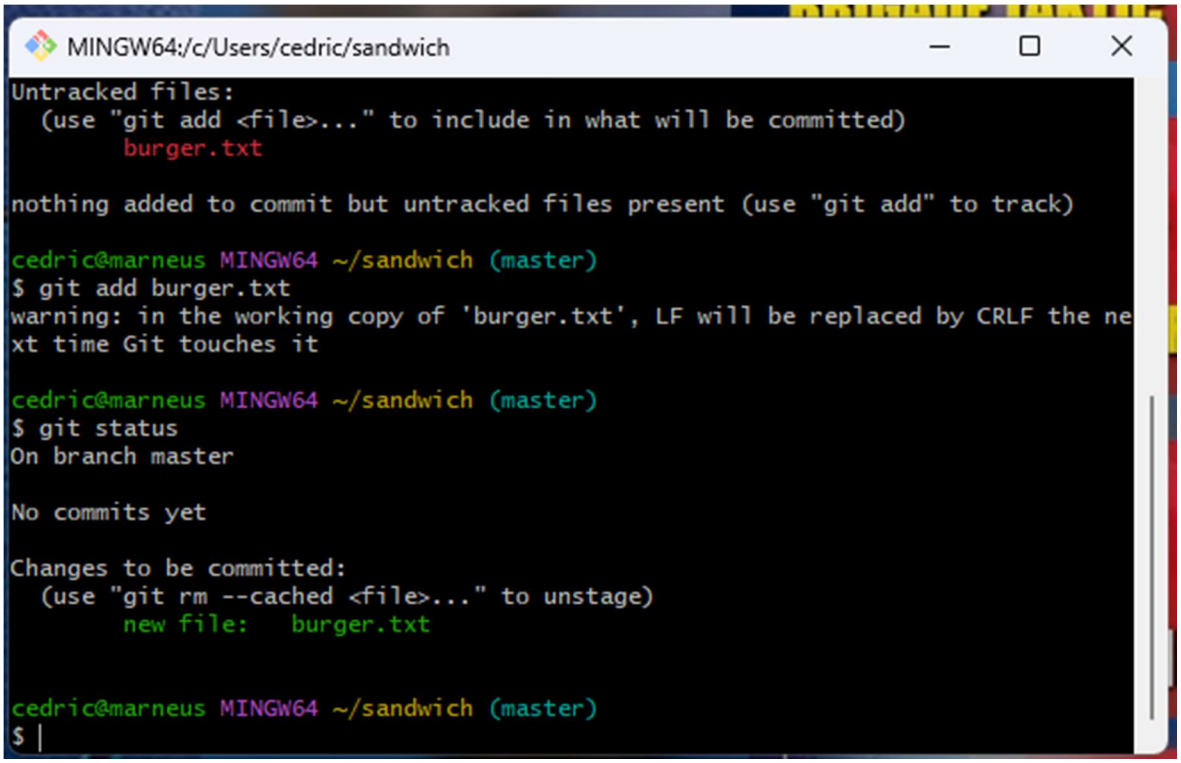


Question 2.3) Préparez burger.txt pour le commit avec git add burger.txt. Utilisez git status à nouveau pour vérifier que les modifications ont bien été placées dans l'index. Puis, utilisez git diff --cached pour observer les différences entre l'index est la dernière version présent dans l'historique de révision (qui est vide)

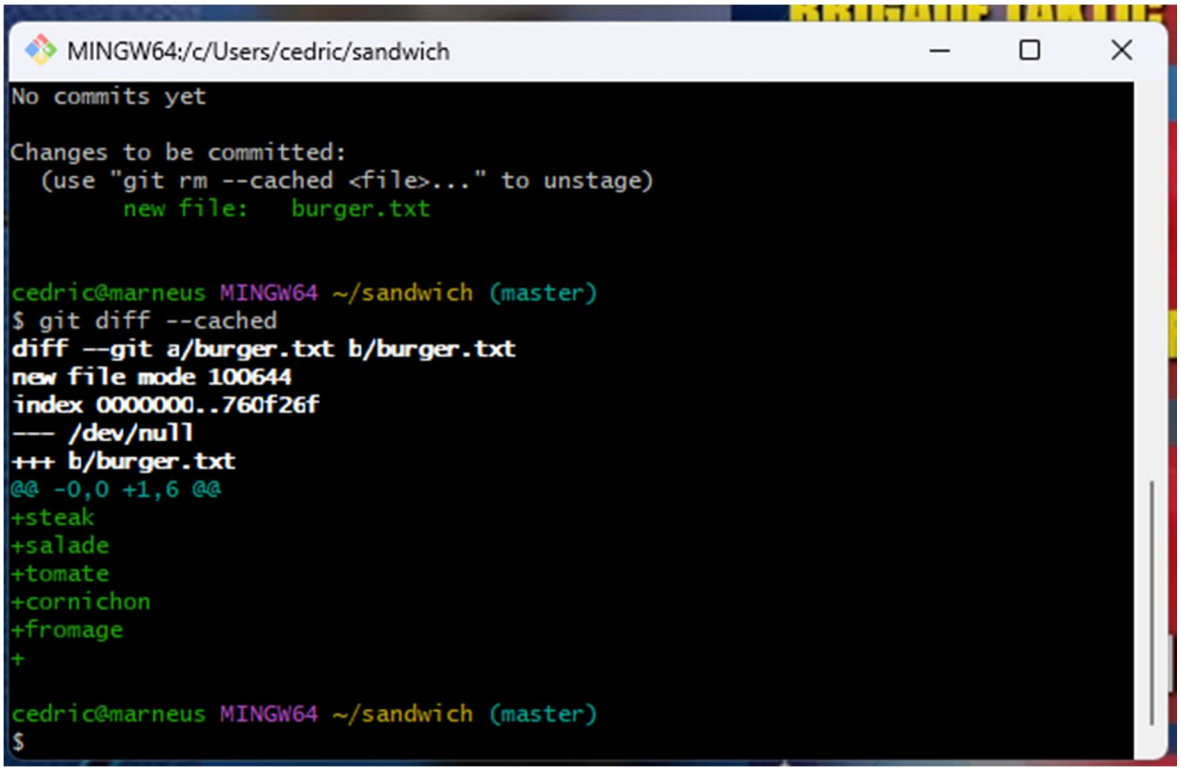
Je place le fichier dans l'index grâce à la commande **git add**, il passe du statut de copie de travail à l'index



Puis je contrôle le statut du fichier, on peut voir qu'il est prêt à être commitez



Avec la commande **git diff –cached** j'ai accès à plus de détails ainsi que le contenu du fichier **burger.txt**

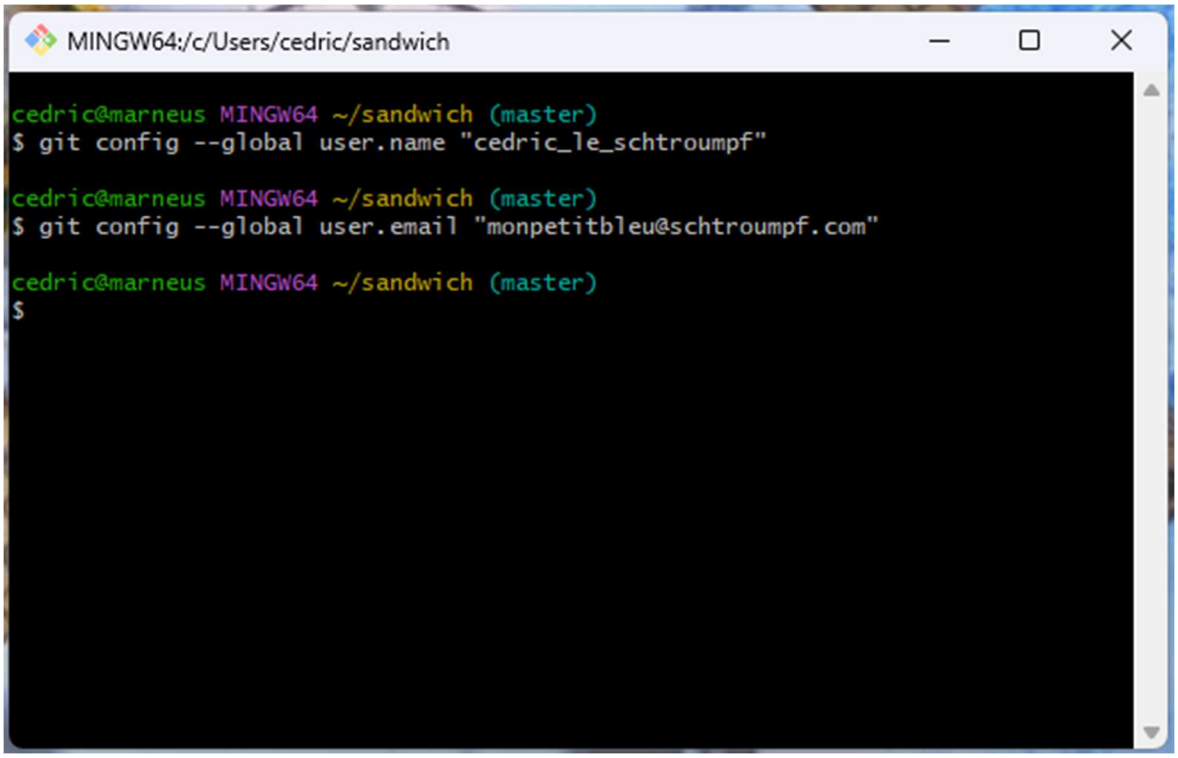


Question 2.4) Commitez votre modification avec git commit -m "". Le message entre guillemets doubles décrira la nature de votre modification (généralement ≤ 65 caractères).

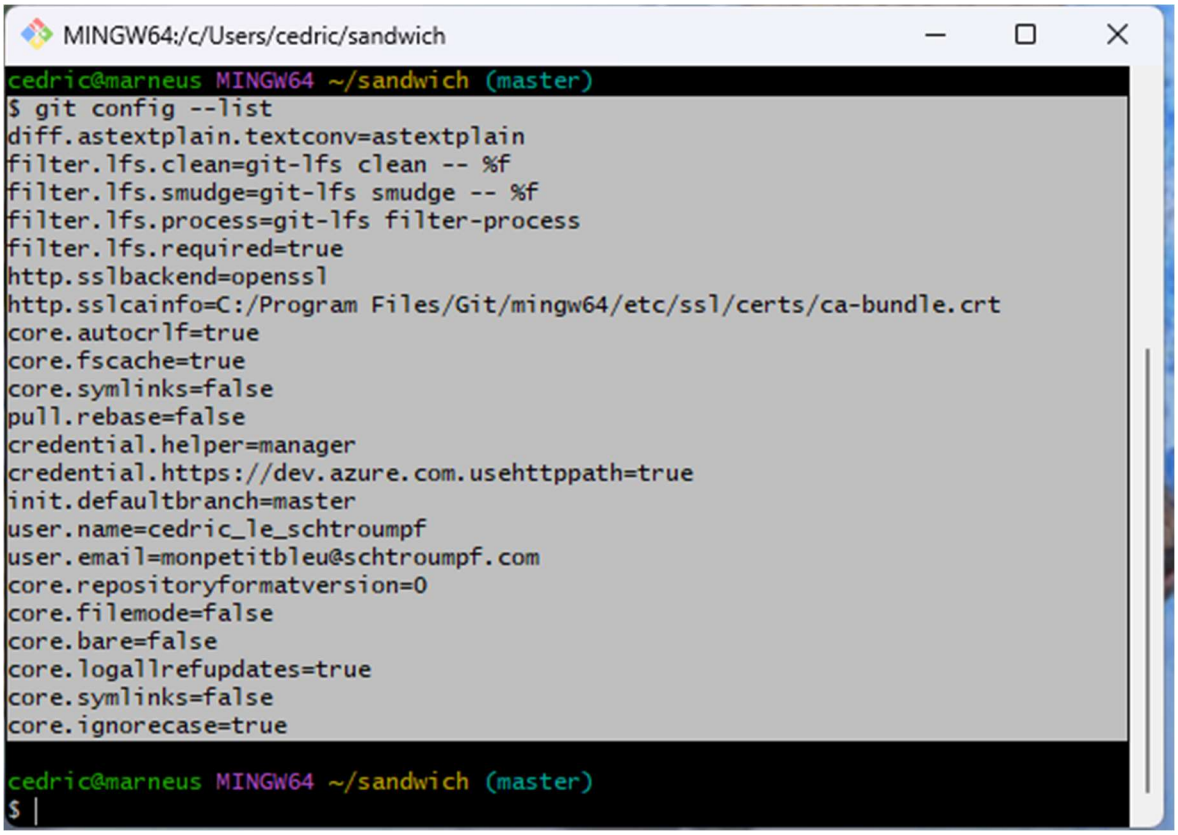
Afin de commitez je vais devoir me crée une **authentification** afin de savoir qui a crée ou modifier le fichier grâce aux commandes

Git config --global user.name

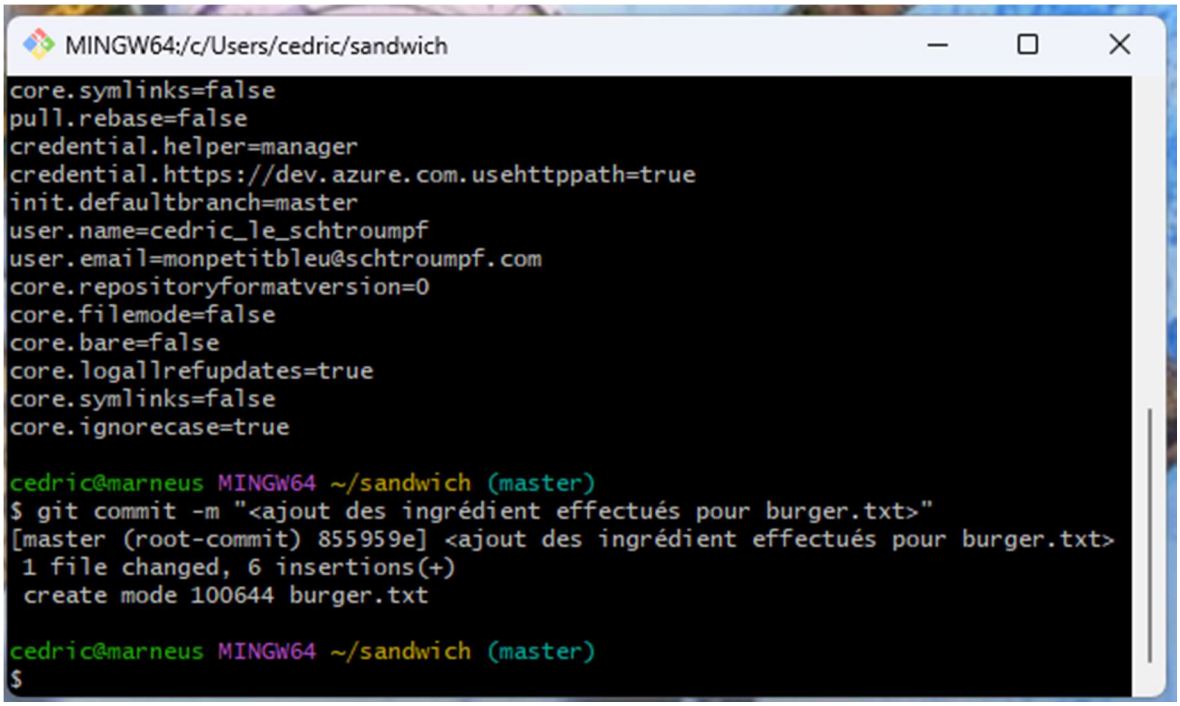
Git config –global user.email



Afin de voir si le compte a bien été créé, je fais la commande **gitconfig --list**, on peut voir que le compte a bien été créé

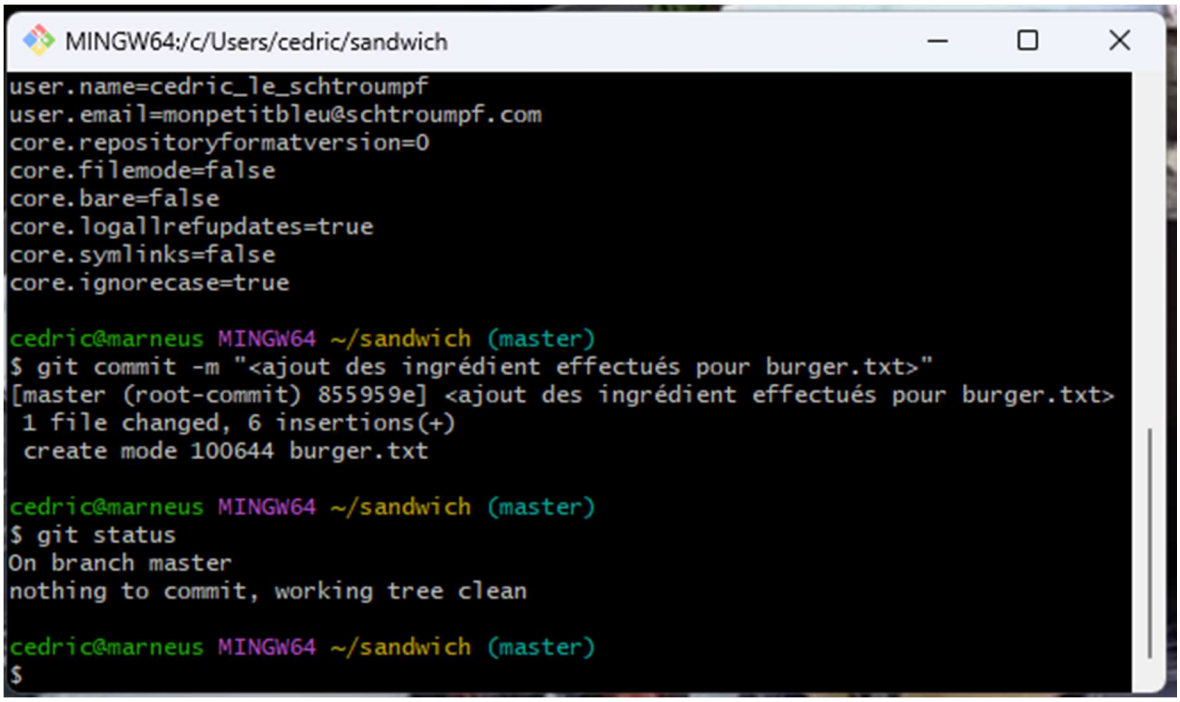


Une fois identifié je vais pouvoir commit le fichier texte, et je rajoute mon texte personnalisé, en l'occurrence <ajout des ingrédients effectués pour …..>



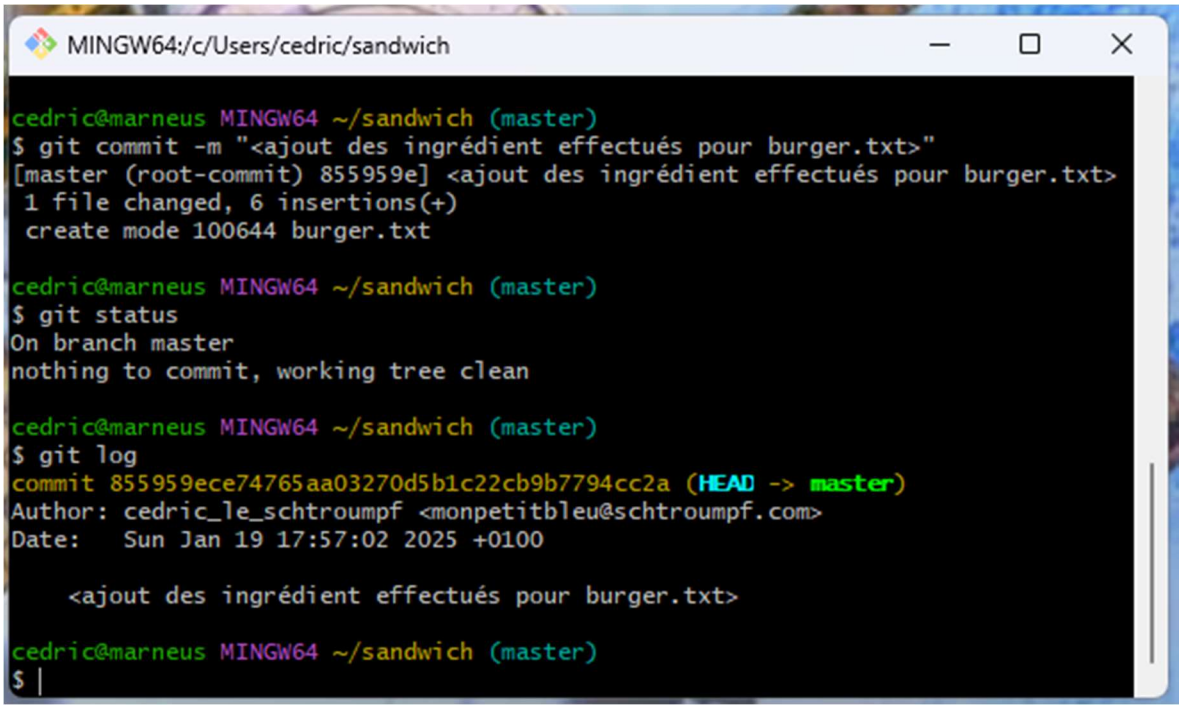
Question 2.5) Exécutez à nouveau git status, pour vérifier que vos modifications ont bien été commités.

Une fois le fichier commitez je contrôle le statut et je peux voir que le fichier et bien été déplacé car il m'indique qu'il n'y a plus rien dans l'index



Question 2.6) Essayez à présent la commande git log pour afficher la liste des changements effectués dans ce dépôt ; combien y en a-t-il ? Quel est le numéro (un hash cryptographique en format SHA1) du dernier commit effectué ?

J'essaye la commande git log, je peux voir qu'il y a 1 fichier dans le dépôt, son numéro (SHA1) et : **855959ece74765aa03270d5b1c22cb9b7794cc2a**



Question 2.7) Créez quelques autres sandwiches hot_dog.txt, jambon_beurre.txt. . .et/ou modifiez les compositions de sandwiches déjà créés, en commitant chaque modification séparément. Chaque commit doit contenir une et une seule création ou modification de fichier. Effectuez au moins 5 modifications différentes (et donc 5 commits différents). À chaque étape essayez les commandes suivantes :

— git diff avant git add pour observer ce que vous allez ajouter à l'index ; (Effectué avec la recette croque madame)

— git diff --cached après git add pour observer ce que vous allez committer. (Effectué avec la recette croque madame)

Je vais créer maintenant 5 autres recettes au format **.txt**

Voici la deuxième recette

```
MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ nano curry_wurst.txt

MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ git add curry_wurst.txt
warning: in the working copy of 'curry_wurst.txt', LF will be replaced by CRLF the next time Git touches it

MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   curry_wurst.txt
```

```
MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ git commit -m "<ajout des ingrédients effectués pour curry_wurst.txt>"
[master 0119212] <ajout des ingrédients effectués pour curry_wurst.txt>
 1 file changed, 6 insertions(+)
 create mode 100644 curry_wurst.txt
```

Je contrôle que la deuxième recette et bien enregistré

```
MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ git log
commit 011921253634f4a58b322f64eb0406e30e28afdf (HEAD -> master)
Author: cedric_le _schtroumpf <monpetitbleu@schtroumpf.com>
Date:   Tue Jan 21 09:35:29 2025 +0100

    <ajout des ingrédients effectués pour curry_wurst.txt>

commit 0566cd1d3c3e7eabce4ded8a30a8a40e099b849a
Author: cedric_le _schtroumpf <monpetitbleu@schtroumpf.com>
Date:   Tue Jan 21 09:28:38 2025 +0100

    <ajout des ingrédients effectués pour burger.txt>
```

Voici la troisième recette

```
MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ nano croque_madame.txt

MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ git add croque_madame.txt
warning: in the working copy of 'croque_madame.txt', LF will be replaced by CRLF the next time Git touches it

MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ git commit -m "<ajout des ingrédients effectués pour croque_madame.txt>"
[master 8a7314b] <ajout des ingrédients effectués pour croque_madame.txt>
 1 file changed, 6 insertions(+)
 create mode 100644 croque_madame.txt
```

Voici ce que l'on voit lorsque j'utilise la commande **git diff** avant git add on peut voir que la modification apparait en vert

```
MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ git diff
diff --git a/croque_madame.txt b/croque_madame.txt
index 8341bfe..900f64c 100644
--- a/croque_madame.txt
+++ b/croque_madame.txt
@@ -1,5 +1,5 @@
 tranches de pain de mie
-jmabon cuit
+jambon cuit
 tranches de fromage
 oeufs
 beurre
```

Lorsque l'on utilise la commande **git diff –cached**, ce sont les mêmes informations que celle d'avant

```
MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ git diff --cached
diff --git a/croque_madame.txt b/croque_madame.txt
index 8341bfe..900f64c 100644
--- a/croque_madame.txt
+++ b/croque_madame.txt
@@ -1,5 +1,5 @@
 tranches de pain de mie
-jmabon cuit
+jambon cuit
 tranches de fromage
 oeufs
 beurre
```

Voici la quatrième recette

```
MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ nano poulet_et_choucroute.txt

MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ git add poulet_et_choucroute.txt
warning: in the working copy of 'poulet_et_choucroute.txt', LF will be replaced by CRLF the next time Git touches it

MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ git commit -m "<ajout des ingrédients effectués pour poulet_et_choucroute.txt>"
[master de88711] <ajout des ingrédients effectués pour poulet_et_choucroute.txt>
 1 file changed, 5 insertions(+)
 create mode 100644 poulet_et_choucroute.txt
```

Voici la cinquième recette

```
MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ nano bratwurst.txt

MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ git add bratwurst.txt
warning: in the working copy of 'bratwurst.txt', LF will be replaced by CRLF the next time Git touches it

MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ git commit -m "<ajout des ingrédients effectués pour bratwurst.txt>"
[master 686a807] <ajout des ingrédients effectués pour bratwurst.txt>
 1 file changed, 5 insertions(+)
 create mode 100644 bratwurst.txt
```

Et la dernière recette

```
MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ nano triangle_au_saumon.txt

MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ git add triangle_au_saumon.txt
warning: in the working copy of 'triangle_au_saumon.txt', LF will be replaced by CRLF the next time Git touches it

MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ git commit -m "<ajout des ingrédients effectués pour triangle_au_saumon.txt>"
[master 54eeadb] <ajout des ingrédients effectués pour triangle_au_saumon.txt>
 1 file changed, 8 insertions(+)
 create mode 100644 triangle_au_saumon.txt
```

Question 2.8) Regardez à nouveau l'historique des modifications avec git log et vérifiez avec git status que vous avez tout commité. Git offre plusieurs interfaces, graphiques ou non, pour afficher l'historique. Essayez les commandes suivantes (gitg et gitk ne sont pas forcément installés) :

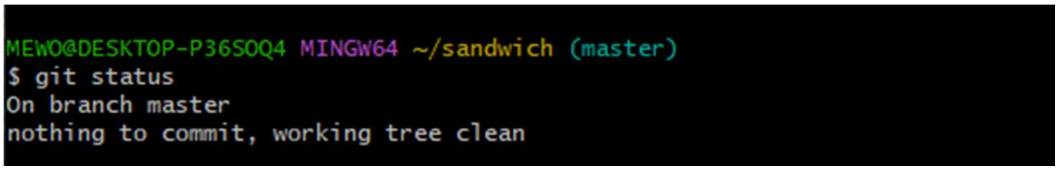
— git log

— git log --graph --pretty=short

— gitg

— gitk

Je vais contrôler que tous les fichiers ont bien été commitez, grâce à la commande **git status**, et l'on peut voir que l'index est vide et que tous les fichiers sont dans le dépôt.



Puis je vais regarder quand les fichiers ont été enregistrés et par qui grâce à la commande **git log**

```
MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ git log
commit 54eeadb38c2e0aad5e48fe51189e70ae3f45606a (HEAD -> master)
Author: cedric_le _schtroumpf <monpetitbleu@schtroumpf.com>
Date:   Tue Jan 21 10:01:53 2025 +0100

    <ajout des ingrédients effectués pour triangle_au_saumon.txt>

commit 686a80776e48a87ab6e59b8eff46adc4a2c51c99
Author: cedric_le _schtroumpf <monpetitbleu@schtroumpf.com>
Date:   Tue Jan 21 09:50:16 2025 +0100

    <ajout des ingrédients effectués pour bratwurst.txt>

commit de887110729b9b2ea18e26e78a18ddcda024e409
Author: cedric_le _schtroumpf <monpetitbleu@schtroumpf.com>
Date:   Tue Jan 21 09:43:48 2025 +0100

    <ajout des ingrédients effectués pour poulet_et_choucroute.txt>

commit 8a7314b4dae065a2456a6c5272e0c50382c17065
Author: cedric_le _schtroumpf <monpetitbleu@schtroumpf.com>
Date:   Tue Jan 21 09:40:24 2025 +0100

    <ajout des ingrédients effectués pour croque_madame.txt>

commit 011921253634f4a58b322f64eb0406e30e28afdf
Author: cedric_le _schtroumpf <monpetitbleu@schtroumpf.com>
Date:   Tue Jan 21 09:35:29 2025 +0100

    <ajout des ingrédients effectués pour curry_wurst.txt>

commit 0566cd1d3c3e7eabce4ded8a30a8a40e099b849a
Author: cedric_le _schtroumpf <monpetitbleu@schtroumpf.com>
Date:   Tue Jan 21 09:28:38 2025 +0100

    <ajout des ingrédients effectués pour burger.txt>
```

Avec la commande **git log –graph –pretty=short**, on n’a moins d’informations,

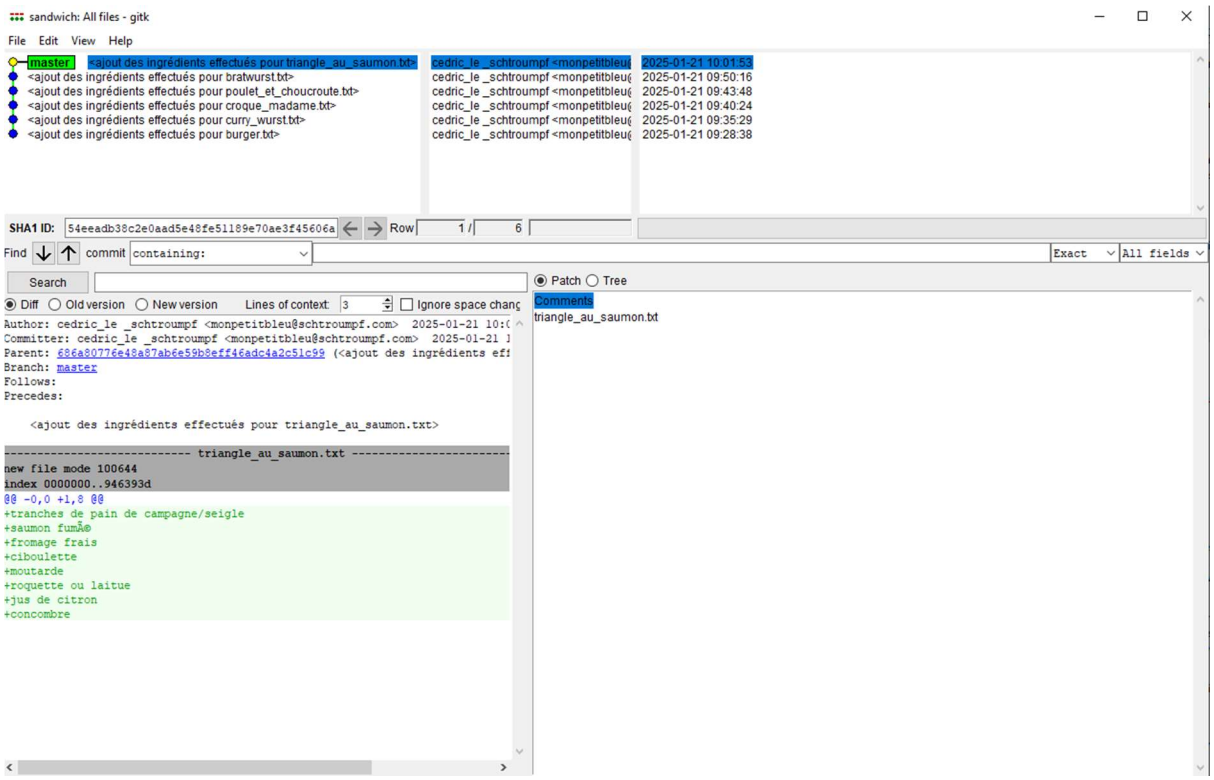
La différence entre les deux c’est que l’on n’a pas la date et l’heure

```
MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ git log --graph --pretty=short
* commit 54eeadb38c2e0aad5e48fe51189e70ae3f45606a (HEAD -> master)
| Author: cedric_le _schtroumpf <monpetitbleu@schtroumpf.com>
|
|     <ajout des ingrédients effectués pour triangle_au_saumon.txt>
|
* commit 686a80776e48a87ab6e59b8eff46adc4a2c51c99
| Author: cedric_le _schtroumpf <monpetitbleu@schtroumpf.com>
|
|     <ajout des ingrédients effectués pour bratwurst.txt>
|
* commit de887110729b9b2ea18e26e78a18ddcda024e409
| Author: cedric_le _schtroumpf <monpetitbleu@schtroumpf.com>
|
|     <ajout des ingrédients effectués pour poulet_et_choucroute.txt>
|
* commit 8a7314b4dae065a2456a6c5272e0c50382c17065
| Author: cedric_le _schtroumpf <monpetitbleu@schtroumpf.com>
|
|     <ajout des ingrédients effectués pour croque_madame.txt>
|
* commit 011921253634f4a58b322f64eb0406e30e28afdf
| Author: cedric_le _schtroumpf <monpetitbleu@schtroumpf.com>
|
|     <ajout des ingrédients effectués pour curry_wurst.txt>
|
* commit 0566cd1d3c3e7eabce4ded8a30a8a40e099b849a
  Author: cedric_le _schtroumpf <monpetitbleu@schtroumpf.com>

      <ajout des ingrédients effectués pour burger.txt>
```

La commande **gitg** n’a pas fonctionné,

```
MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ gitg
bash: gitg: command not found

MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ gitk
```

Par contre la commande **gitk** fonctionne et permet d'affiché les informations grâce a un navigateur graphique.



Question 2.9) Vous voulez changer d'avis entre les différents états de la Figure 1 ? Faites une modification d'un ou plusieurs sandwiches, ajoutez-la à l'index avec git add (vérifiez cet ajout avec git status), mais ne la commitez pas. Exécutez git reset sur le nom de fichier (ou les noms de fichiers) que vous avez préparés pour le commit ; vérifiez avec git status le résultat.

J'ai changé d'avis, et je décide de modifié le fichier croque madame je modifie via nano puis je sauvegarde le **txt**. On peut voir lorsque l'on utilise la commande status, on voit que le fichier a été modifier « **modified** »

```
MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ nano croque_madame.txt

MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ git add croque_madame.txt
warning: in the working copy of 'croque_madame.txt', LF will be replaced by CRLF the next time Git touches it

MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   croque_madame.txt
```

Finalement je décidé de ne pas modifier le texte, j'utilise la commande **git reset** plus le nom afin d'annulé ce que j'ai fait.

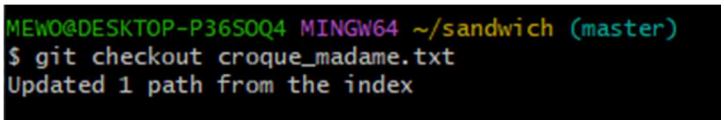
```
MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ git reset croque_madame.txt
Unstaged changes after reset:
M       croque_madame.txt
```

Vérification avec **git status**

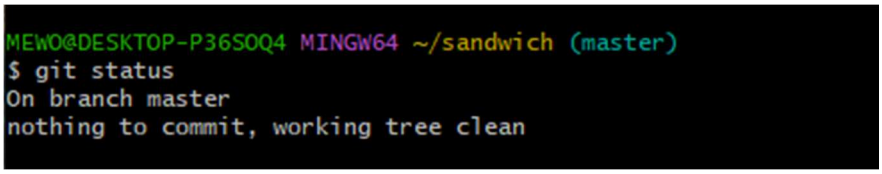
```
MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Question 2.10) Votre modification a été « retirée » de l'index. Vous pouvez maintenant la jeter à la poubelle avec la commande git checkout sur le ou les noms des fichiers modifiés, qui récupère dans l'historique leurs versions correspondant au tout dernier commit. Essayez cette commande, et vérifiez avec git status qu'il n'y a maintenant plus aucune modification à commiter

Lors du **git checkout** je m'aperçois qu'il y a plus de fichier dans l'index



Vérification avec **git status**



Question 2.11) Regardez l'historique de votre dépôt avec git log ; choisissez dans la liste un commit (autre que le dernier). Exécutez git checkout COMMITID où COMMITID est le numéro de commit que vous avez choisi. Vérifiez que l'état de vos sandwiches est maintenant revenu en arrière, au moment du commit choisi. Que dit maintenant git status ? git log n'affiche plus les commits postérieurs à l'état actuel, sauf si vous ajoutez l'option --all. Attention, avec git checkout les fichiers de votre copie de travail sont modifiés directement par Git pour les remettre dans l'état que vous avez demandé. Si les fichiers modifiés sont ouverts par d'autres programmes (e.g. un éditeur de texte comme Emacs), il faudra les réouvrir pour observer les modifications.

Je vais utiliser la commande **commitid** sur une recette de la liste

En l'occurrence «**curry_wurst.txt**» à l'aide de la commande **git checkout commitid**

011921253634f4a58b322f64eb0406e30e28afdf

```
MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ git checkout 011921253634f4a58b322f64eb0406e30e28afdf
Note: switching to '011921253634f4a58b322f64eb0406e30e28afdf'.

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 0119212 <ajout des ingrédients effectués pour curry_wurst.txt>
```

On peut s'apercevoir que avec la commande **git status** le fichier et détaché du commit à partir de la **0119212**, qui sont les initiales du commit **011921253634f4a58b322f64eb0406e30e28afdf**

```
MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich ((0119212...))
$ git status
HEAD detached at 0119212
nothing to commit, working tree clean
```

En revenant en arrière on aperçoit que les fichiers crées après le **0119212** ne sont affichés via la commande **git log**

```
MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich ((0119212...))
$ git log
commit 011921253634f4a58b322f64eb0406e30e28afdf (HEAD)
Author: cedric_le _schtroumpf <monpetitbleu@schtroumpf.com>
Date:   Tue Jan 21 09:35:29 2025 +0100

    <ajout des ingrédients effectués pour curry_wurst.txt>

commit 0566cd1d3c3e7eabce4ded8a30a8a40e099b849a
Author: cedric_le _schtroumpf <monpetitbleu@schtroumpf.com>
Date:   Tue Jan 21 09:28:38 2025 +0100

    <ajout des ingrédients effectués pour burger.txt>
```

On peut voir que avec la commande **git log –all** tous les fichiers commitez apparaissent, même ceux postérieures a au **checkout**.

```
MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich ((0119212...))
$ git log --all
commit 54eeadb38c2e0aad5e48fe51189e70ae3f45606a (master)
Author: cedric_le _schtroumpf <monpetitbleu@schtroumpf.com>
Date:   Tue Jan 21 10:01:53 2025 +0100

    <ajout des ingrédients effectués pour triangle_au_saumon.txt>

commit 686a80776e48a87ab6e59b8eff46adc4a2c51c99
Author: cedric_le _schtroumpf <monpetitbleu@schtroumpf.com>
Date:   Tue Jan 21 09:50:16 2025 +0100

    <ajout des ingrédients effectués pour bratwurst.txt>

commit de887110729b9b2ea18e26e78a18ddcda024e409
Author: cedric_le _schtroumpf <monpetitbleu@schtroumpf.com>
Date:   Tue Jan 21 09:43:48 2025 +0100

    <ajout des ingrédients effectués pour poulet_et_choucroute.txt>

commit 8a7314b4dae065a2456a6c5272e0c50382c17065
Author: cedric_le _schtroumpf <monpetitbleu@schtroumpf.com>
Date:   Tue Jan 21 09:40:24 2025 +0100

    <ajout des ingrédients effectués pour croque_madame.txt>

commit 011921253634f4a58b322f64eb0406e30e28afdf (HEAD)
Author: cedric_le _schtroumpf <monpetitbleu@schtroumpf.com>
Date:   Tue Jan 21 09:35:29 2025 +0100

    <ajout des ingrédients effectués pour curry_wurst.txt>

commit 0566cd1d3c3e7eabce4ded8a30a8a40e099b849a
Author: cedric_le _schtroumpf <monpetitbleu@schtroumpf.com>
Date:   Tue Jan 21 09:28:38 2025 +0100

    <ajout des ingrédients effectués pour burger.txt>
```

Question 2.12) Vous pouvez retourner à la version plus récente de votre dépôt avec git checkout master. Vérifiez que cela est bien le cas. Que dit maintenant git status ?

Pour retourner à la version a la récente du dépôt j'utilise le commande **git checkout master**, on peut apercevoir que le statut de l'index et vide et que a côté du nom de dépôt on n'ait passé de ((**0119212**)) a (**master**) ce qui prouve aussi que l'on n'ait dans la bonne version

```
MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich ((0119212...))
$ git checkout master
Previous HEAD position was 0119212 <ajout des ingrédients effectués pour curry_wurst.txt>
Switched to branch 'master'
```

Vérification avec **git status**

```
MEWO@DESKTOP-P36SOQ4 MINGW64 ~/sandwich (master)
$ git status
On branch master
nothing to commit, working tree clean
```