Timothé Belcour

TP GIT

1. Mise en place et configuration

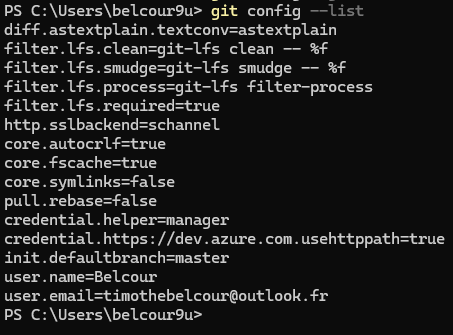
Vérifier la version



Configurer son identité

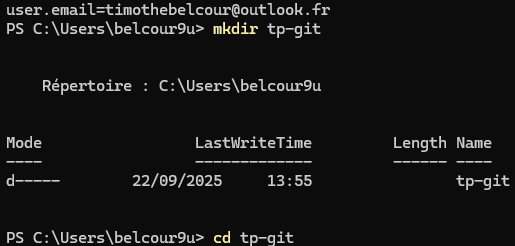


Afficher la configuration

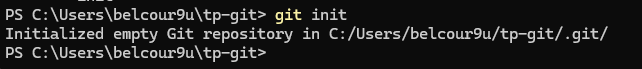


1. Initialiser un dépôt

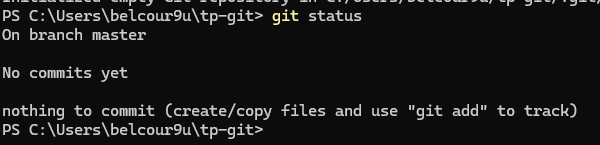
Créer un dossier de travail



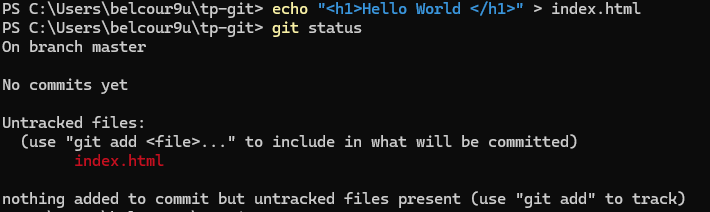
Initialiser Git

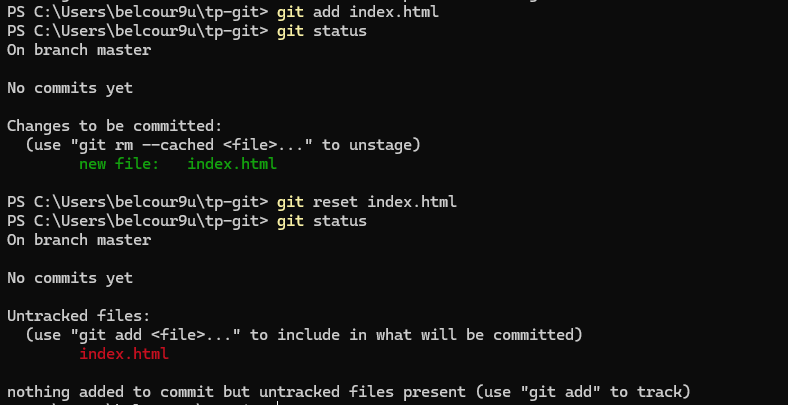


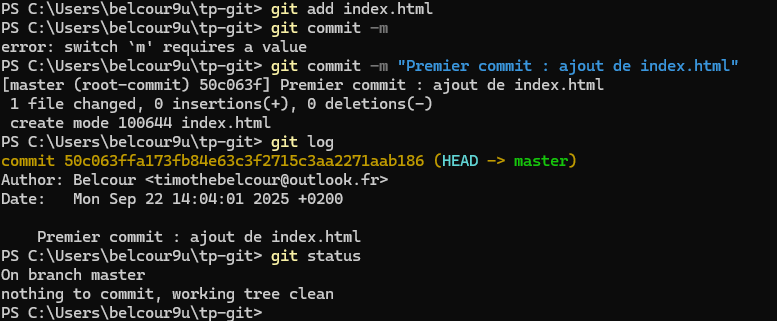
Vérifier l’état du dépôt

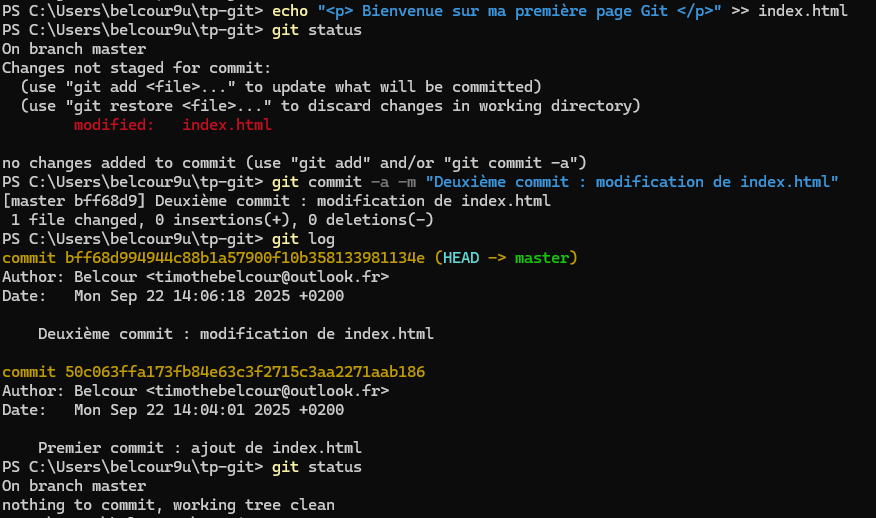


1. Les zones de Git
2. Premier commit









Questions : Répondre avec des captures d’écran et des explications :

1. On peut observer que pour git add le status affiche que le fichier à été ajouter au Git mais sans commit.  
   Pour le git reset le status affiche que le fichier devient une untracked file c’est-à-dire qui n’est plus relier au git.

Pour le git commit le status affiche qu’un commit a été ajouter au fichier index.html.

1. Les informations qui sont affichés après l’exécution git log après chaque commit sont :

Le nom de qui a fait le ou les commit et à l’heure à laquelle ils ont été réaliser et le numéro du commit et dans quel fichier ils ont été ajouter. On a aussi l’ID du commit.

1. L’option -m dans la commande git commit sert à rentrer le commentaire du commit.
2. La différence entre -m et le -m -a c’est que -m passe de la zone index au dépôt local alors que pour -m -a on passe directement du dossier de travail au dépôt local.
3. Non car ça ne fonctionne que pour les fichiers déjà suivis par le Git.
4. 1 : git commit -a -m

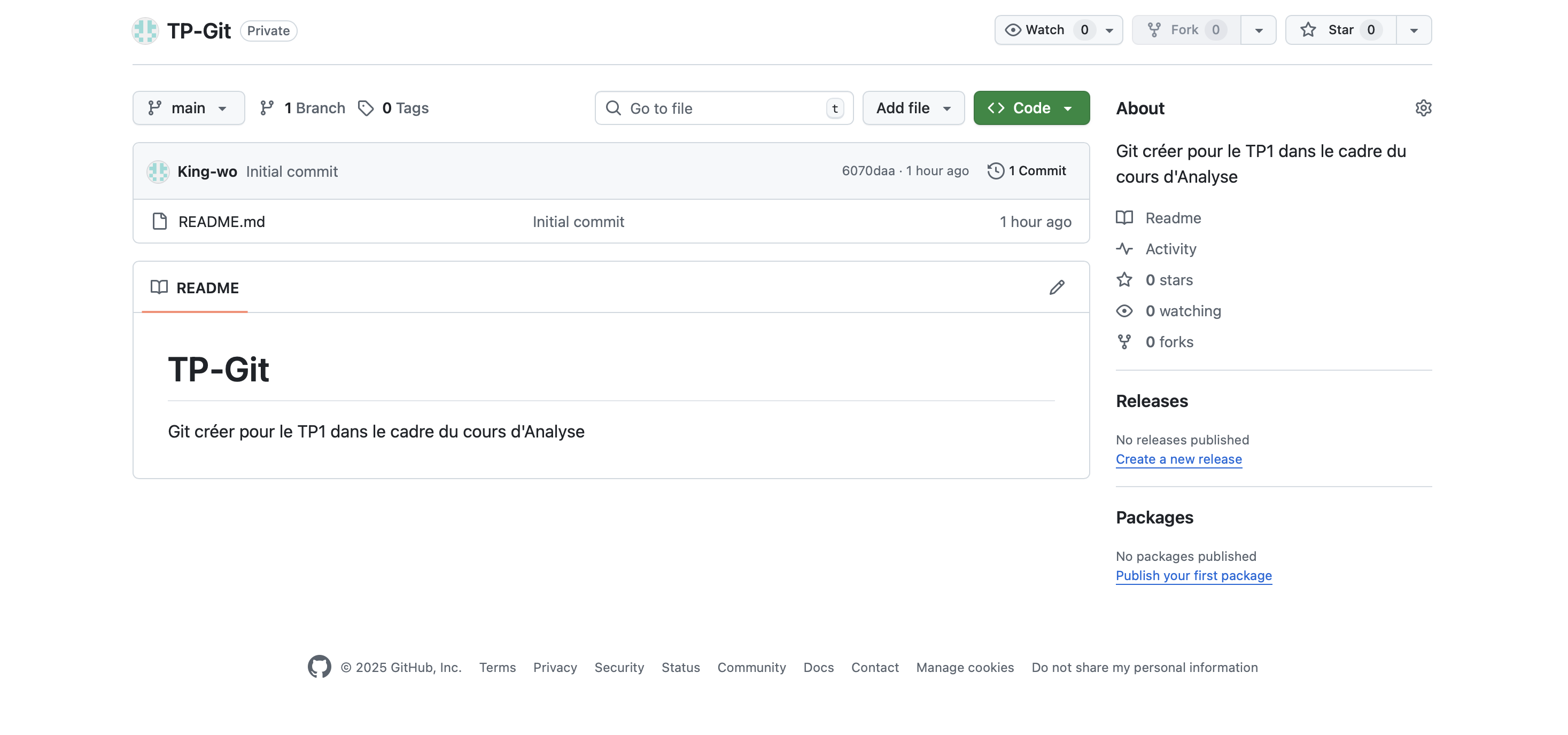
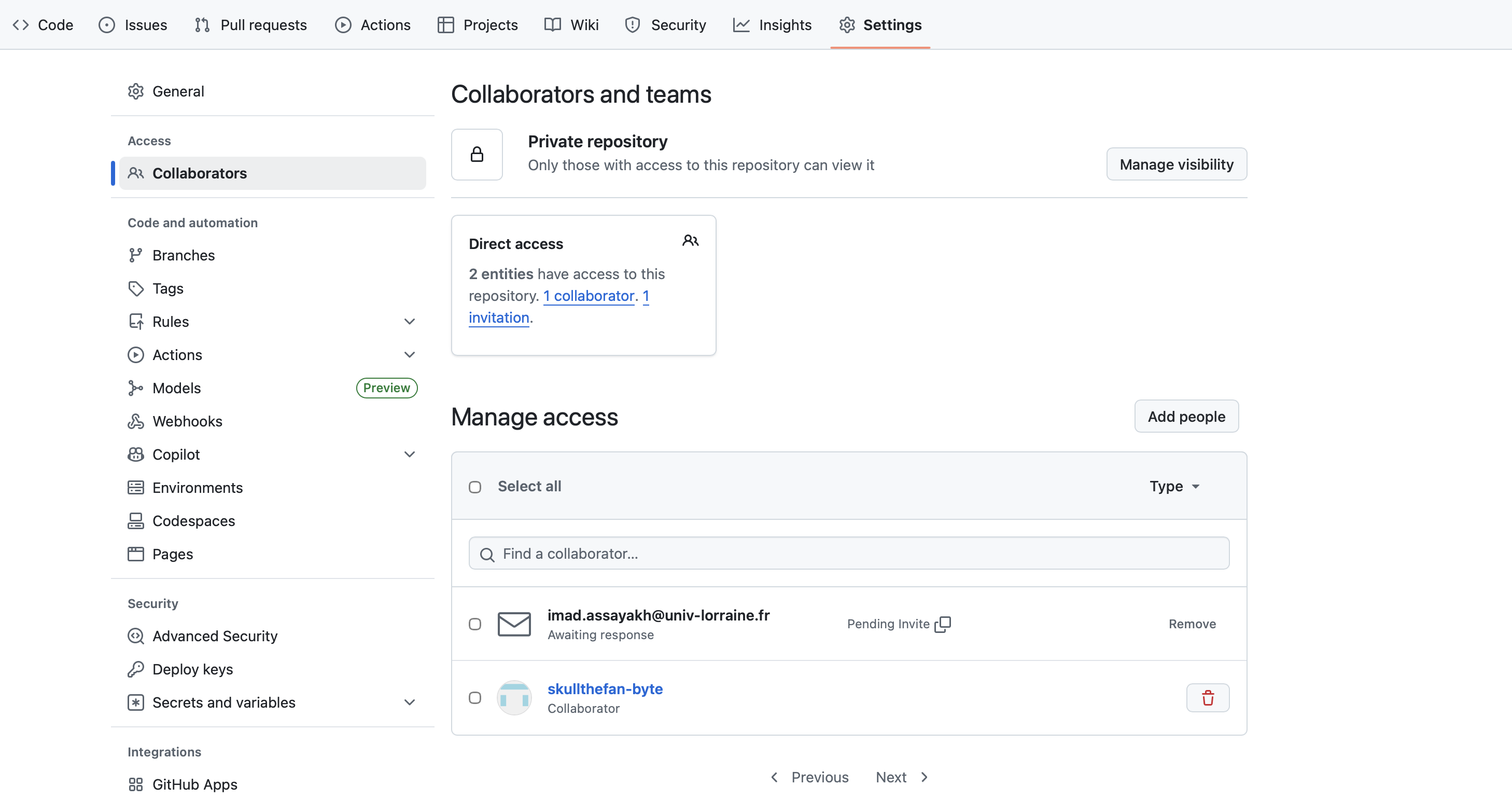
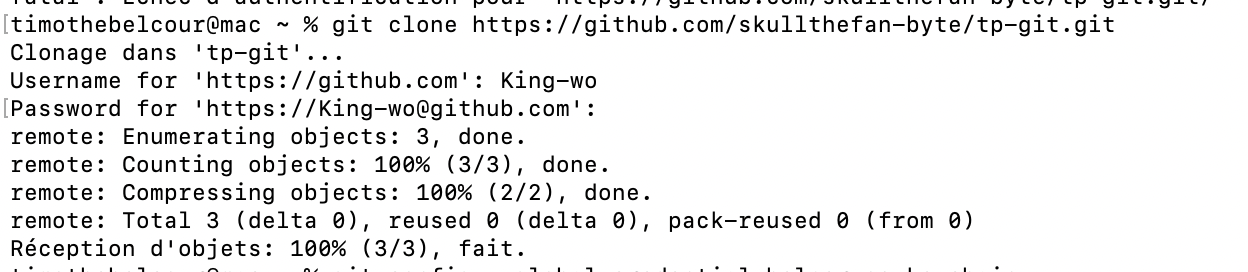
2 : git add

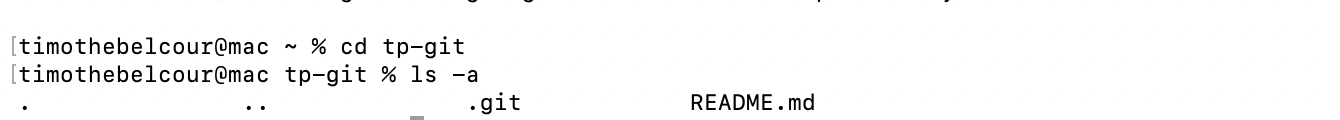
3 : git commit -m

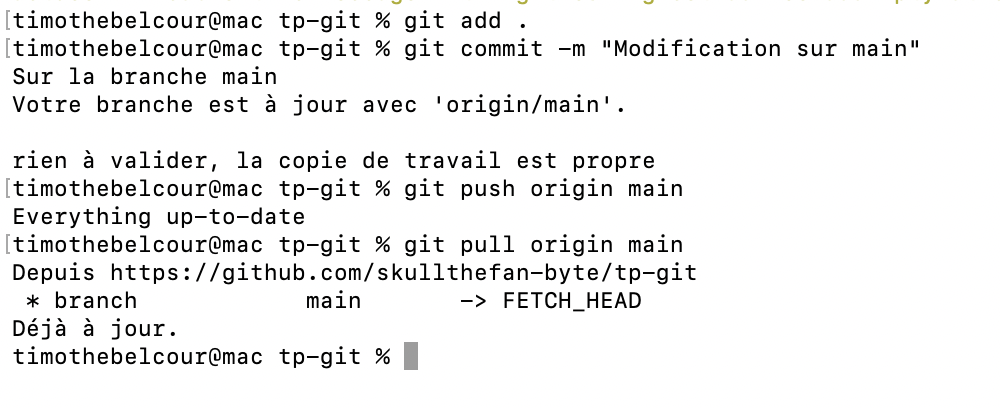
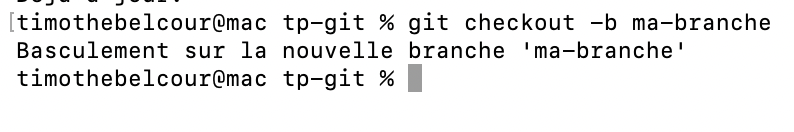
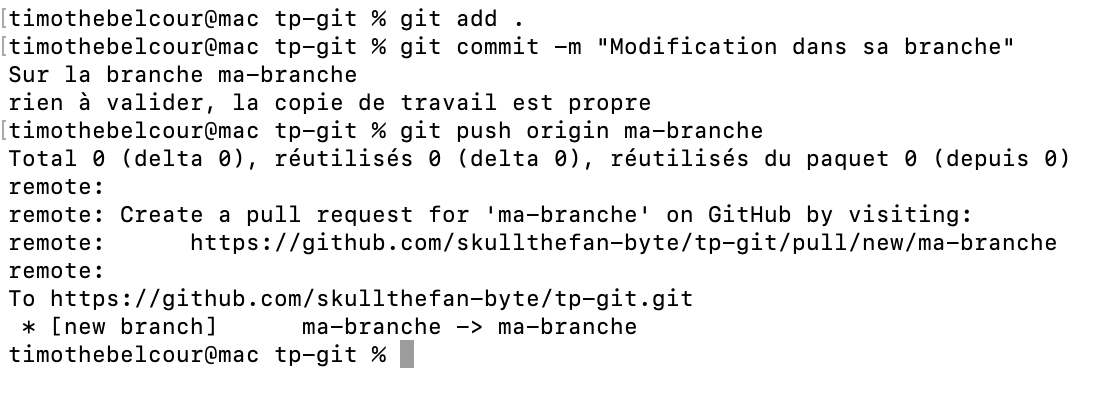
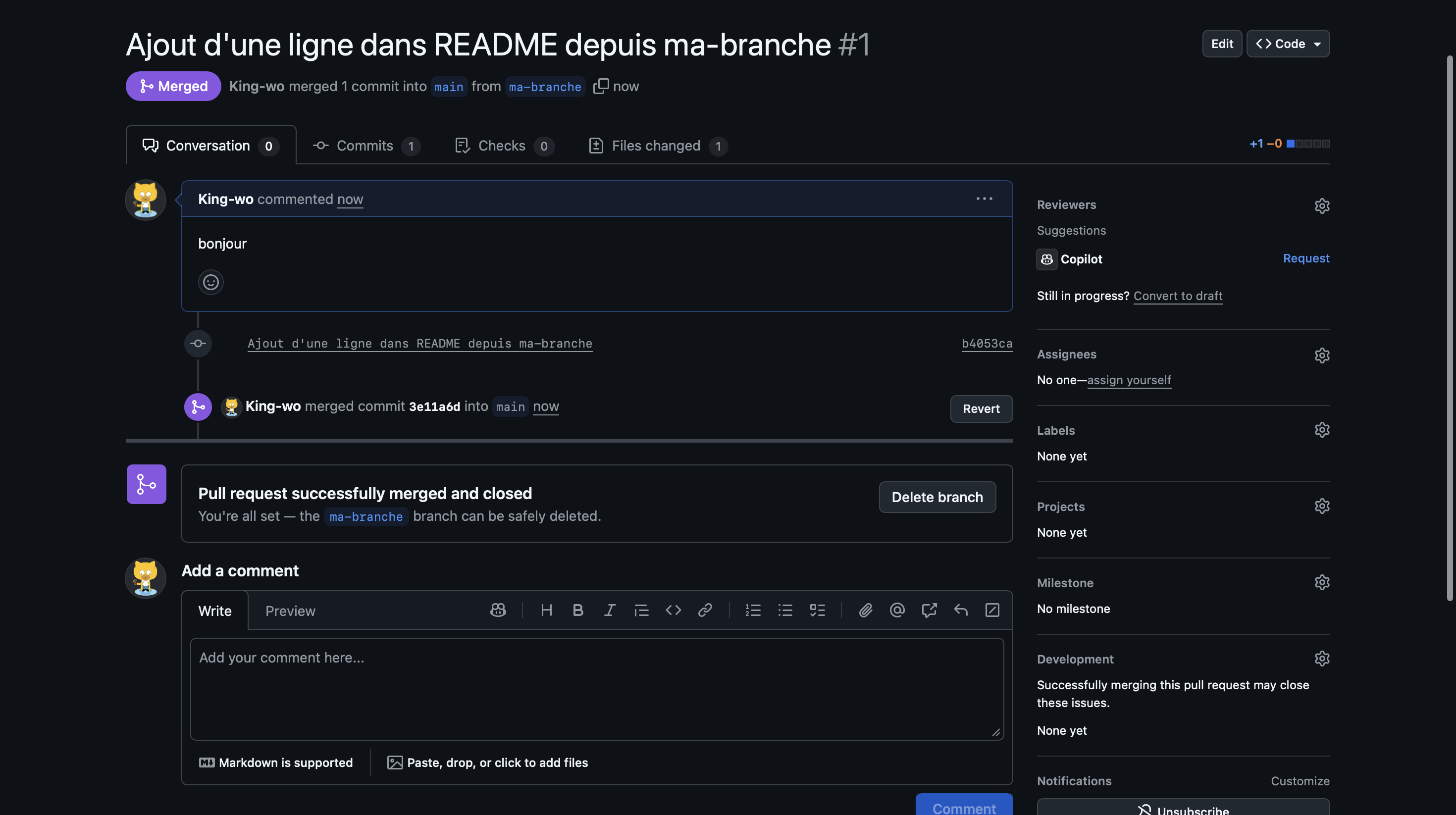
4 : git reset

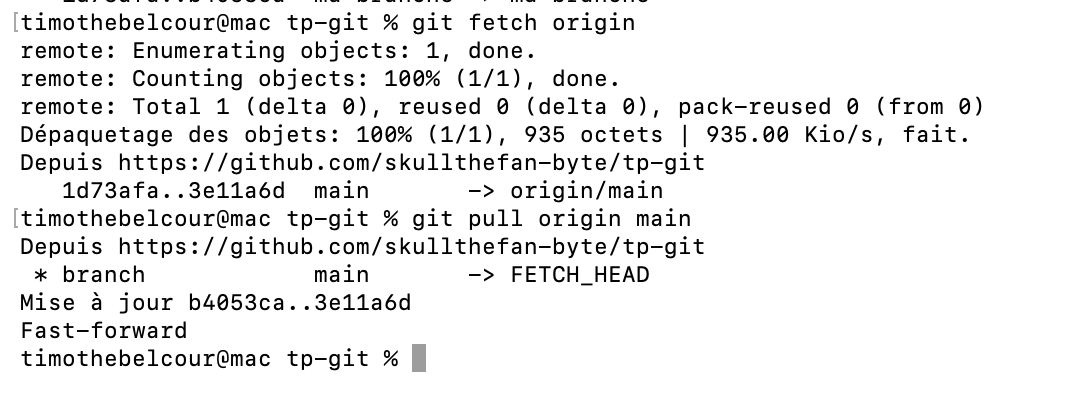
1. La commande git diff quand elle est exécutée avant git add montre toutes les différentes modifications faites au fichier.
2. La commande git diff –cached après avoir exécuté git add montre les modifications que j’ai ajoutées mais n’ont pas encore été intégrées dans l’historique de mon dépôt. Cela me permet de vérifier les différences entre les modifications que j’ai ajoutées et celles que j’ai déjà enregistrer dans l’historique.
3. Git diff montre les modifications non ajoutées à l’index, tandis que git diff --cached affiche celles déjà prêtes à être commitées.

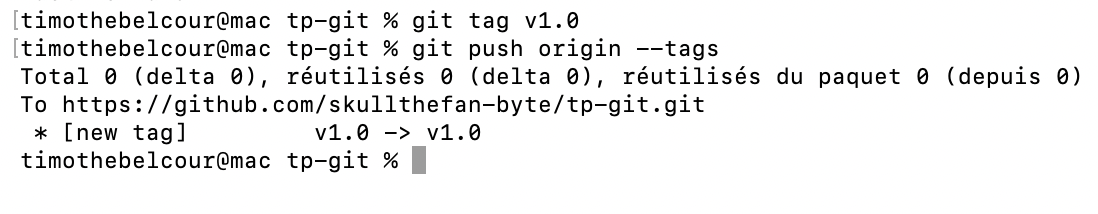
5° Accès distant (GitHub)

1. 
2. 
3. 



1. 
2. 
3. 
4. 8.  
   

9.  


10. 

Questions :

1. Directement sur main les commits vont tout de suite dans la branche principale, donc risque d’erreur et pas de relecture.

Branche + Pull Request on travailles isolé, puis je proposes la fusion via un PR, donc relecture, discussion, historique propre.

1. git branch nom : crée une branche, ne change pas de branche.

git checkout -b nom : crée et bascule immédiatement dessus.

git branch : liste les branches locales et met un \* sur celle courante.

1. git push origin ma-branche : quand je push une branche spécifique.

git push : suffit quand la branche locale a déjà une branche amont configurée (tracking), donc la destination est connue.

1. git fetch : récupère les nouveaux objets/refs du remote sans fusionner.

git pull : fait fetch puis intègre (merge ou rebase) dans ma branche courante.

1. git tag : crée un tag local (n’envoie rien).

git push : push la branche courante (pas les tags, sauf configuration spéciale).

git push --tags : push tous les tags locaux manquants vers le remote.

1. À marquer un point précis de l’historique.
2. 5= git push

6=git fetch

7=git checkout

8=git restore

9=git pull