Génie logiciel, Web, AI, Conception, Projet

Cours 1.1 - Introduction à Git

Table des matières

Définition

Git est un système de contrôle de version distribué permettant de suivre les changements (versions) des fichiers.

Son domaine d'application est généralement pour le développement logiciel, mais on peut y stocker des fichiers quelconques (qui ne sont pas binaires, comme des .exe, vidéos, etc.).

Son utilité

Git est utile puisque dans un contexte réel, le développement est fait en parallèle par plusieurs personnes et le code de production doit être sécurisé contre les changements non autorisés.

Son utilité

Bénifices :

- Protection de la branche principale de production (main, lorsque activé par l'administrateur du GitHub)
- Collaboration avec une équipe de développement fluide et contrôlée
- Permet de retracer les contributeurs et leur code source poussé dans le projet vis-à-vis les différentes versions du code
  - Le mauvais code est associé à un nom
  - On peut retourner dans une version antérieure du code pour régler un problème rapidement
- Offre un contrôle de la qualité (Pull Requests) entre collaborateurs avant de "merge" les changements dans la branche de production (main)
- Sécurise l'accès au projet et au code, seuls les comptes ajoutés avec les permissions nécessaires peuvent appliquer leurs droits (c.à.d. lire, écrire, maintenir, admin, etc.)

Son utilité

Inconvénients :

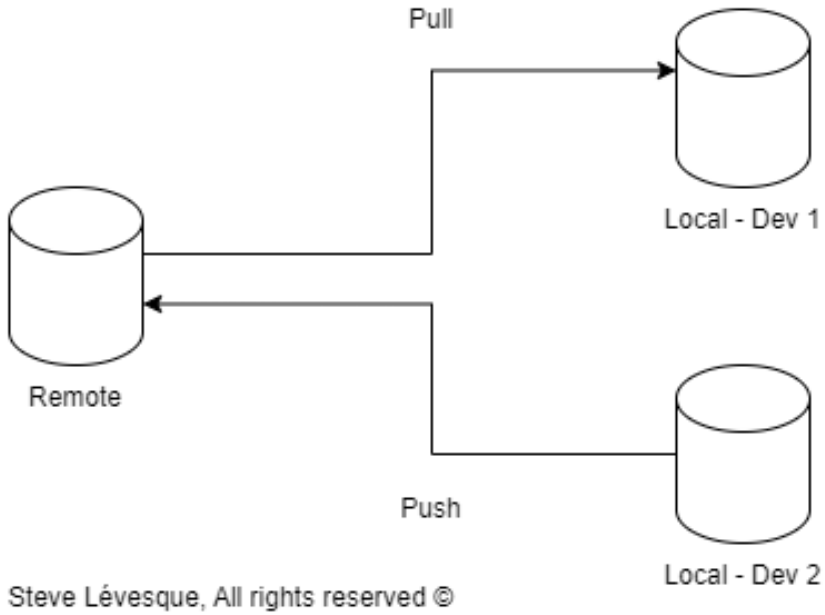
- Plus complexe que de sauvegarder, par exemple, dans OneDrive
- Demande des connaissances techniques plus avancées

Local vs Remote

Le code peut être local (dans l'ordinateur) ou remote (sur le serveur GitHub).

Il est important de pousser (push) le code régulièrement pour plusieurs raisons :

- Être à l'abri si votre ordinateur à un problème
- Montrer des avancements constants à votre gestionnaire

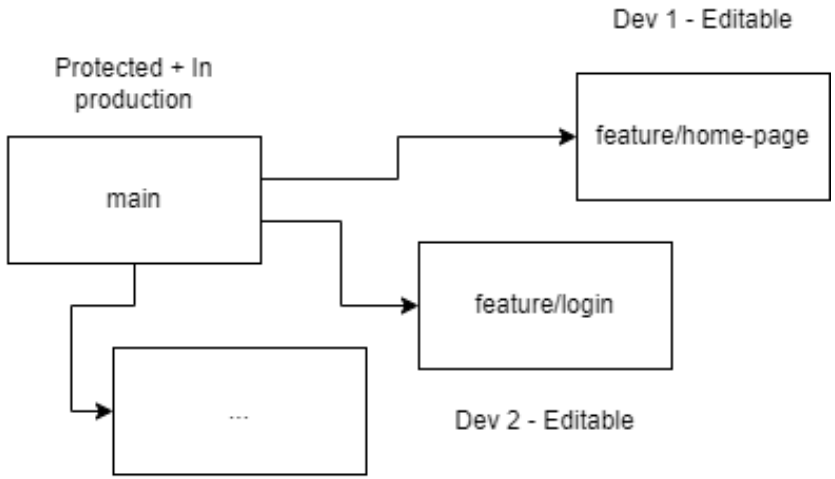


Main et les autres branches

Main (ou master) est la branche (branch) de production de manière standard.

Un groupe organisé se respectant ne va jamais laisser les développeurs modifier/ajouter du nouveau code dans main sans avoir fait une révision de celui-ci.

Les responsables (architectes, leads) peuvent généralement en pratique se permettre de “push” sur main directement pour éviter du temps perdu de leur côté puisqu’ils connaissent le projet par coeur. Ceci-dit, c’est une pratique qui a des risques.



Son utilité

Nous utilisons GitHub dans le cadre du cours : `https://github.com/`.

Outil très populaire en entreprise qui prends en confiance depuis quelques années.

Avait été acheté par Microsoft en 2018 environ.

Pack gratuit pour les étudiants : `https://education.github.com/pack`

Applications pour utiliser Git

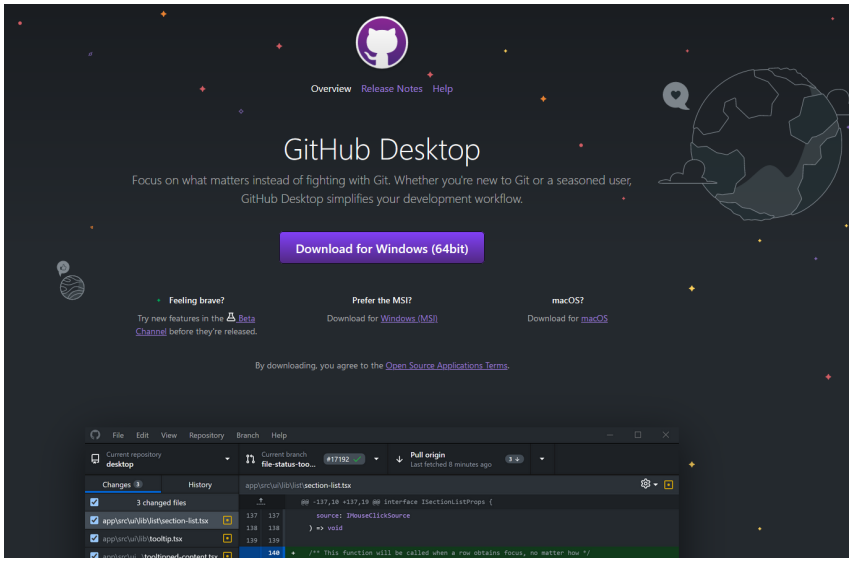


Figure: `https://desktop.github.com/`

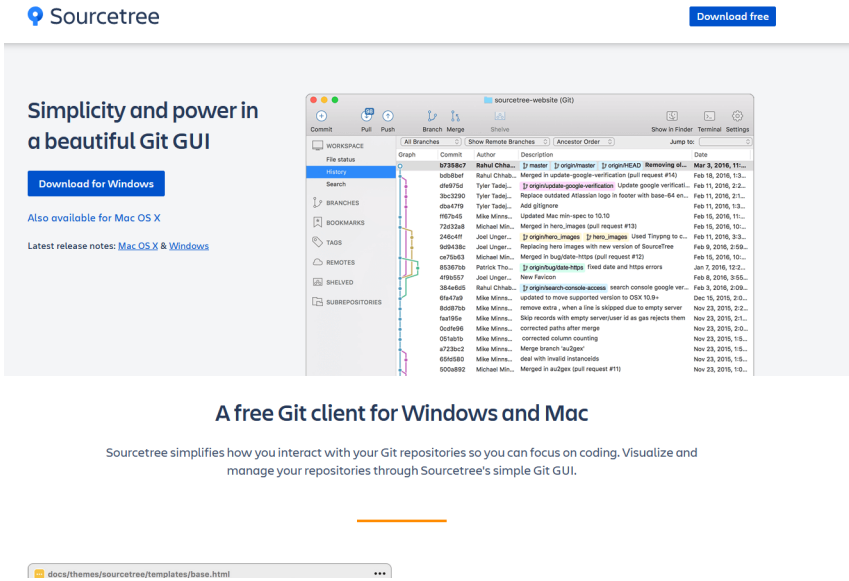


Figure: `https://www.sourcetreeapp.com/`

Créer une nouvelle branche (local + remote)

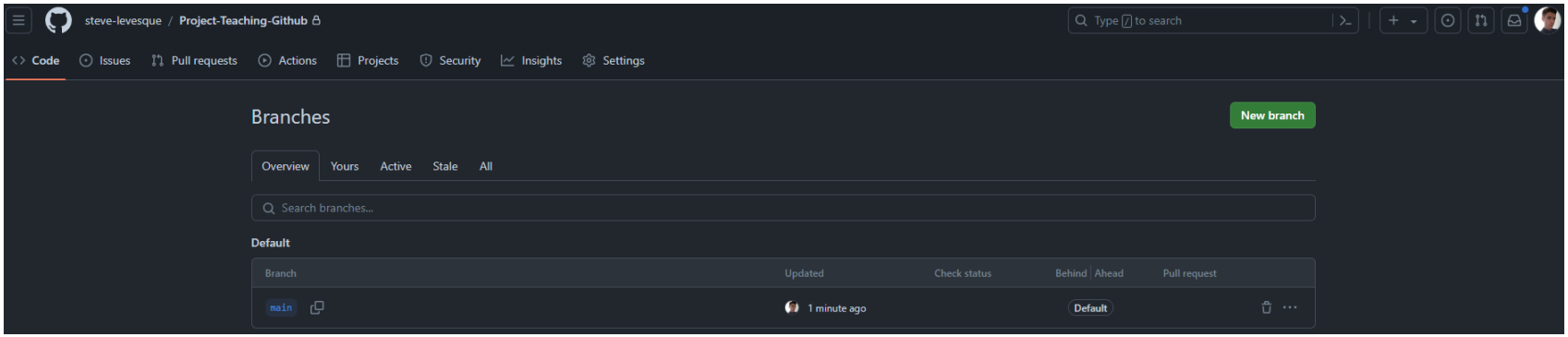


Figure: Créer une branch remote directement (avec le site Web)

Sinon à partir des applications, il faut créer la “branch” en local et ensuite “push” pour qu’elle soit sauvegardée en “remote”.

Créer une nouvelle branche (local + remote)

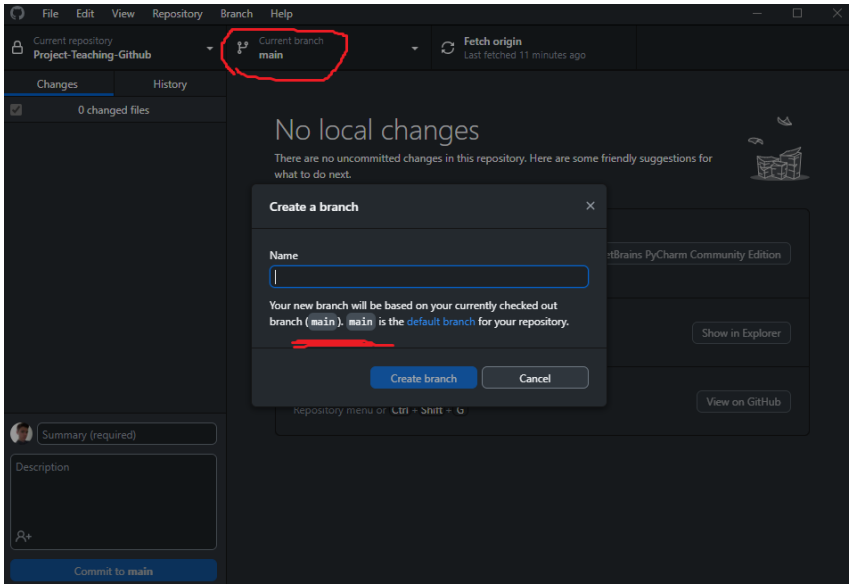


Figure: Créer la branche à partir de main

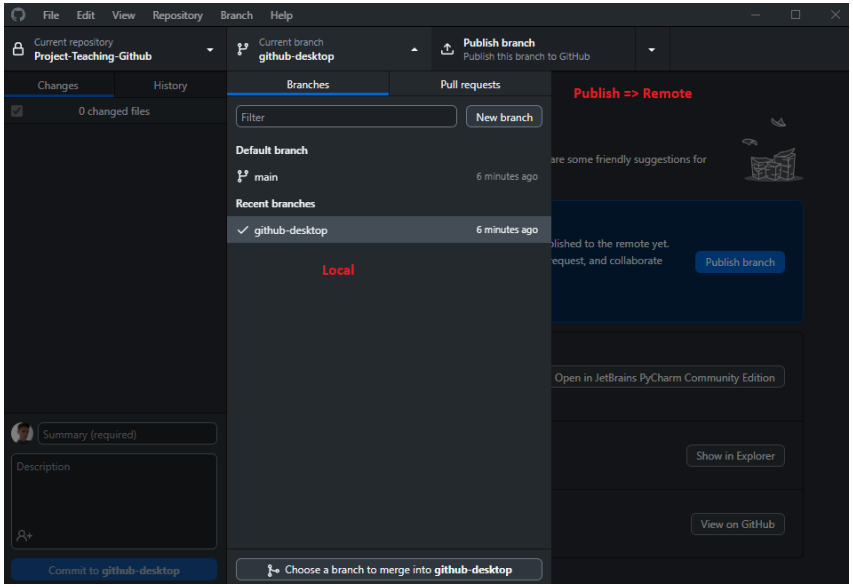


Figure: La branche créée est en local pour le moment

Créer une nouvelle branche (local + remote)

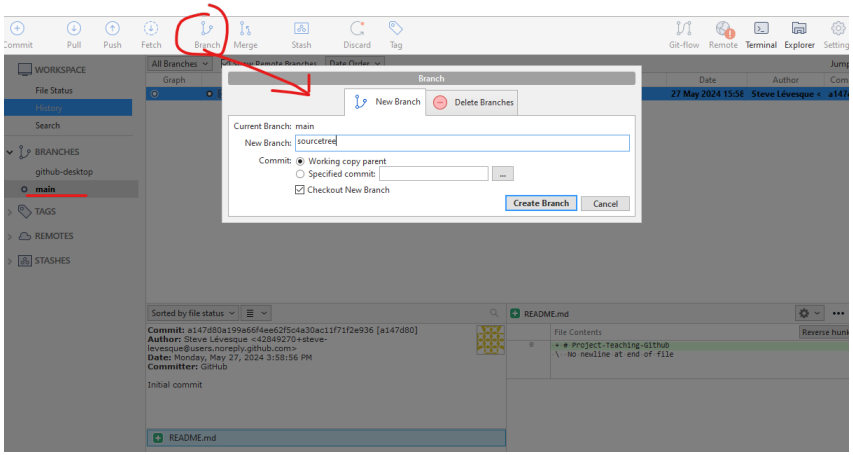


Figure: Créer la branche à partir de main

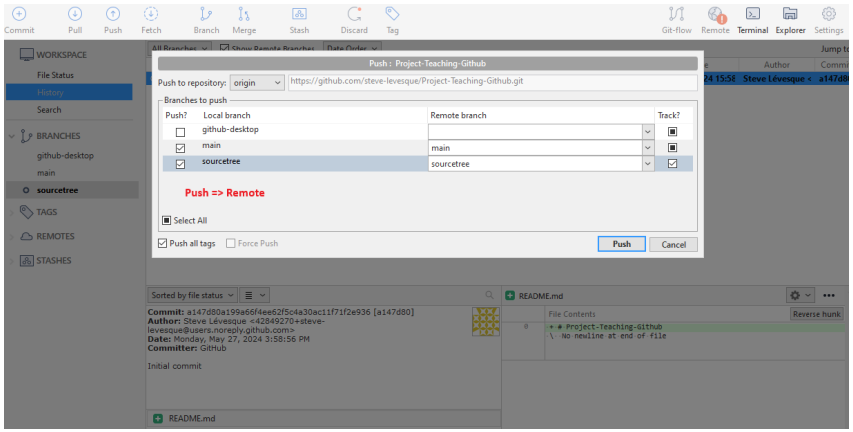


Figure: La rendre “remote”

Créer une nouvelle branche (local + remote)

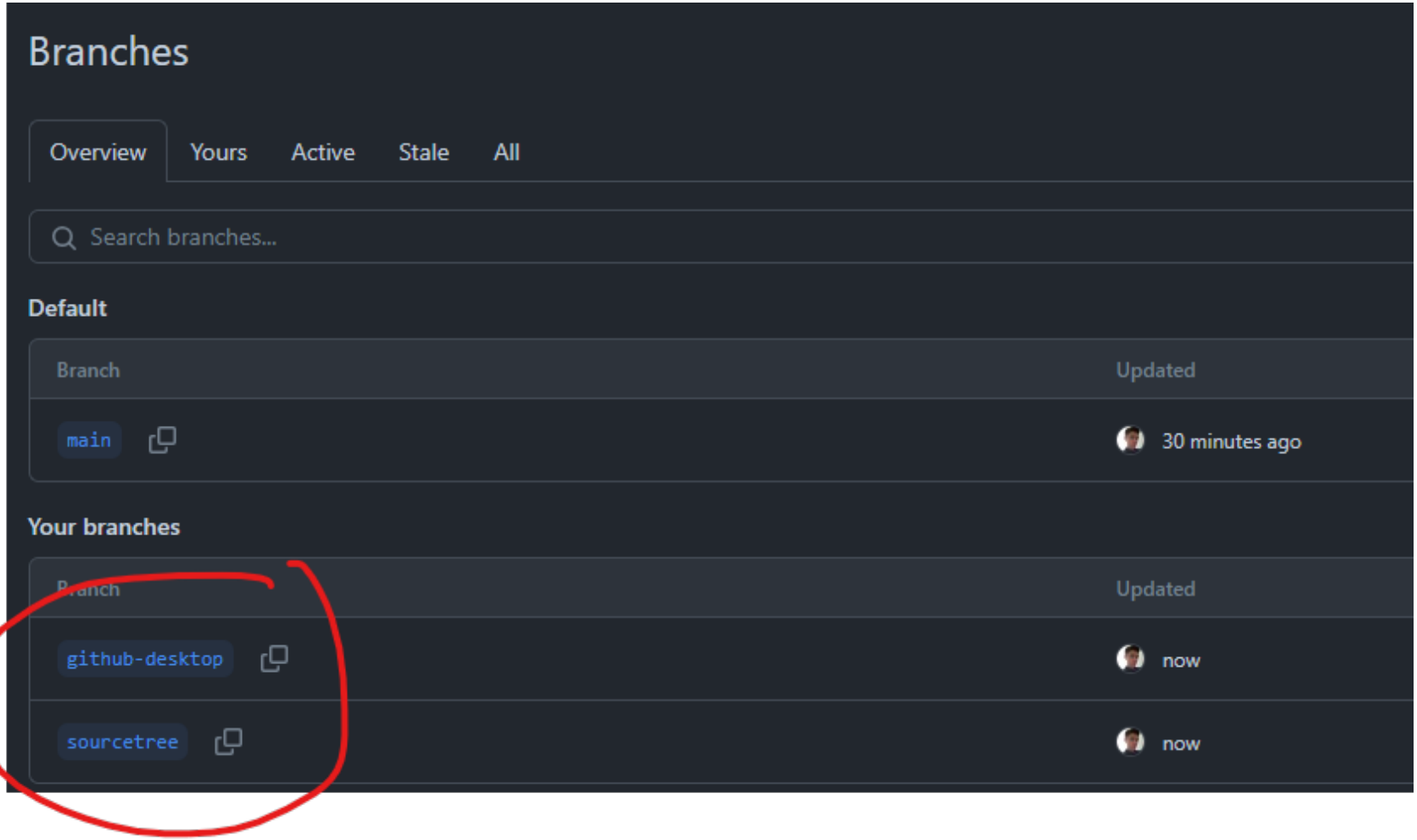


Figure: La branche en "remote" est disponible sur l'interface GitHub

Avoir en local une branch existante sur remote

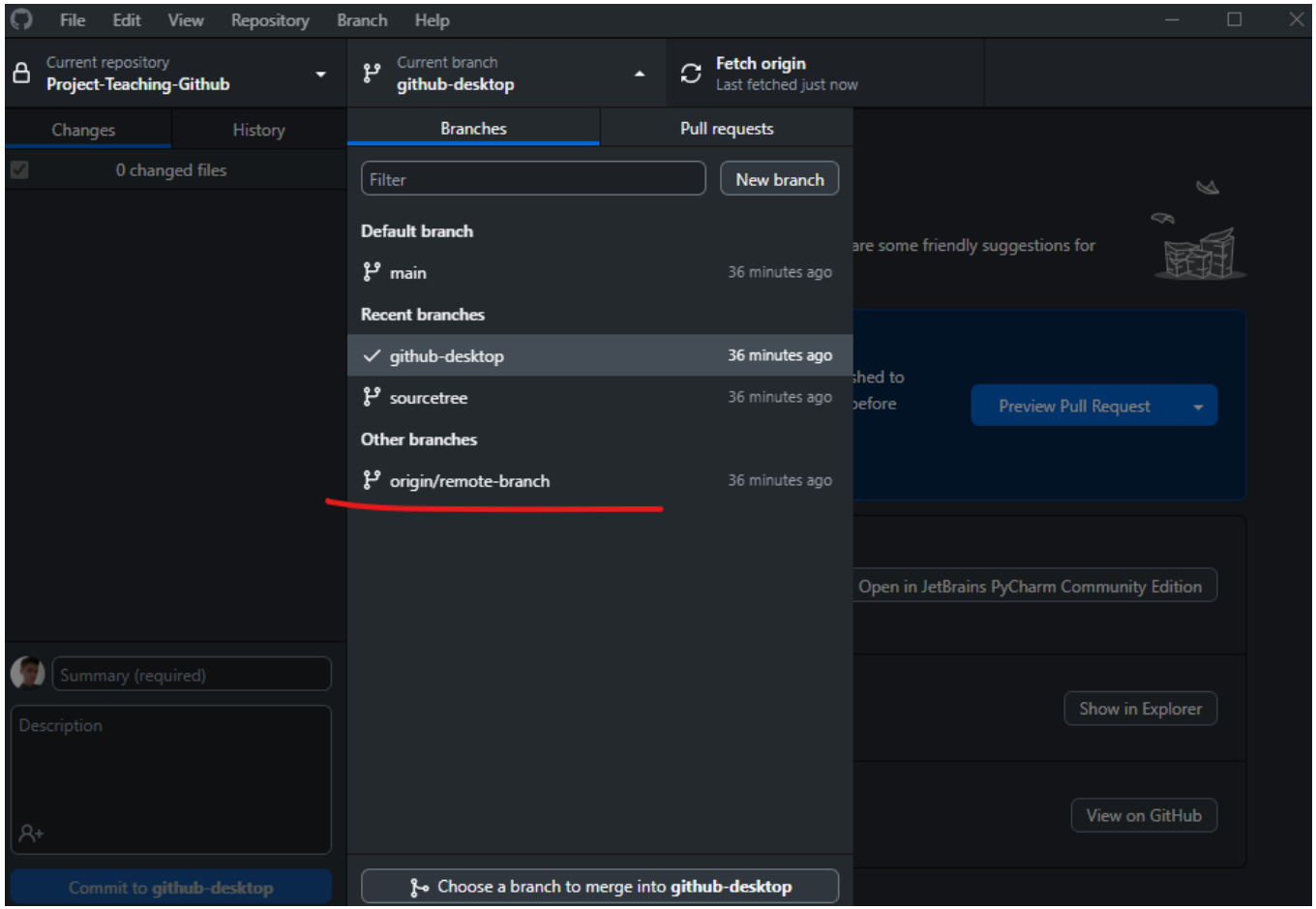


Figure: Les branches en "remote" uniquement peuvent être téléchargées (pulled) en local

Avoir en local une branch existante sur remote

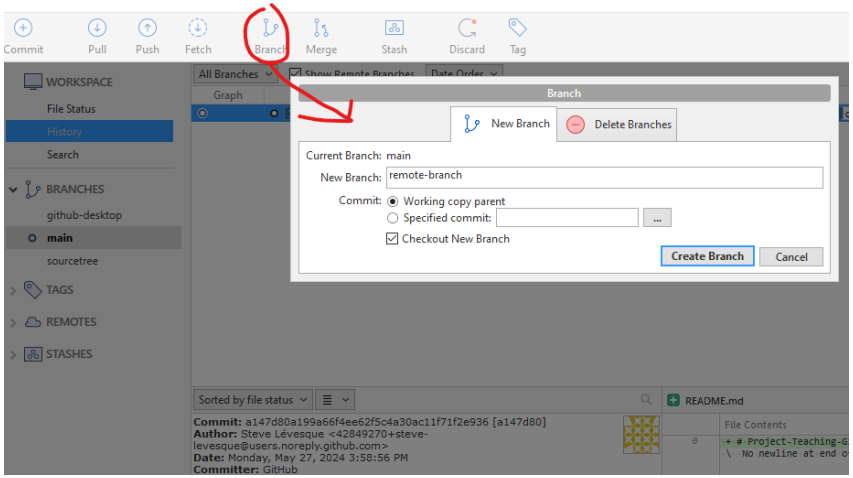


Figure: Créer la branche en local **avec le même nom exact que celle en remote**

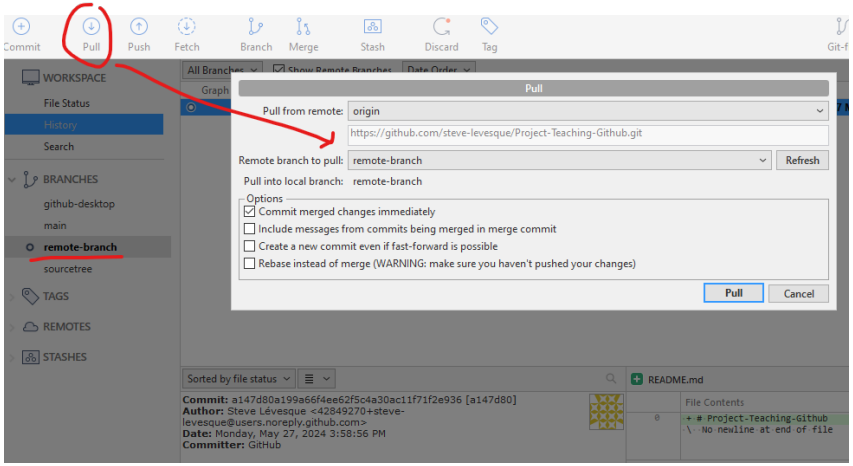


Figure: Pull la branche remote, elle va télécharger le contenu dans la branche locale portant le même nom

Commit/push des avancements sur une branch

Plus simple, il suffit de mettre un titre, descritpion (facultatif), et de “Commit” dans la “branch”.

Il faut apporter des changements aux fichiers pour que le commit soit possible.

Avec les Mises à jours des logiciels, cette opération est généralement de plus en plus conviviale.

Commit/push des avancements sur une branch

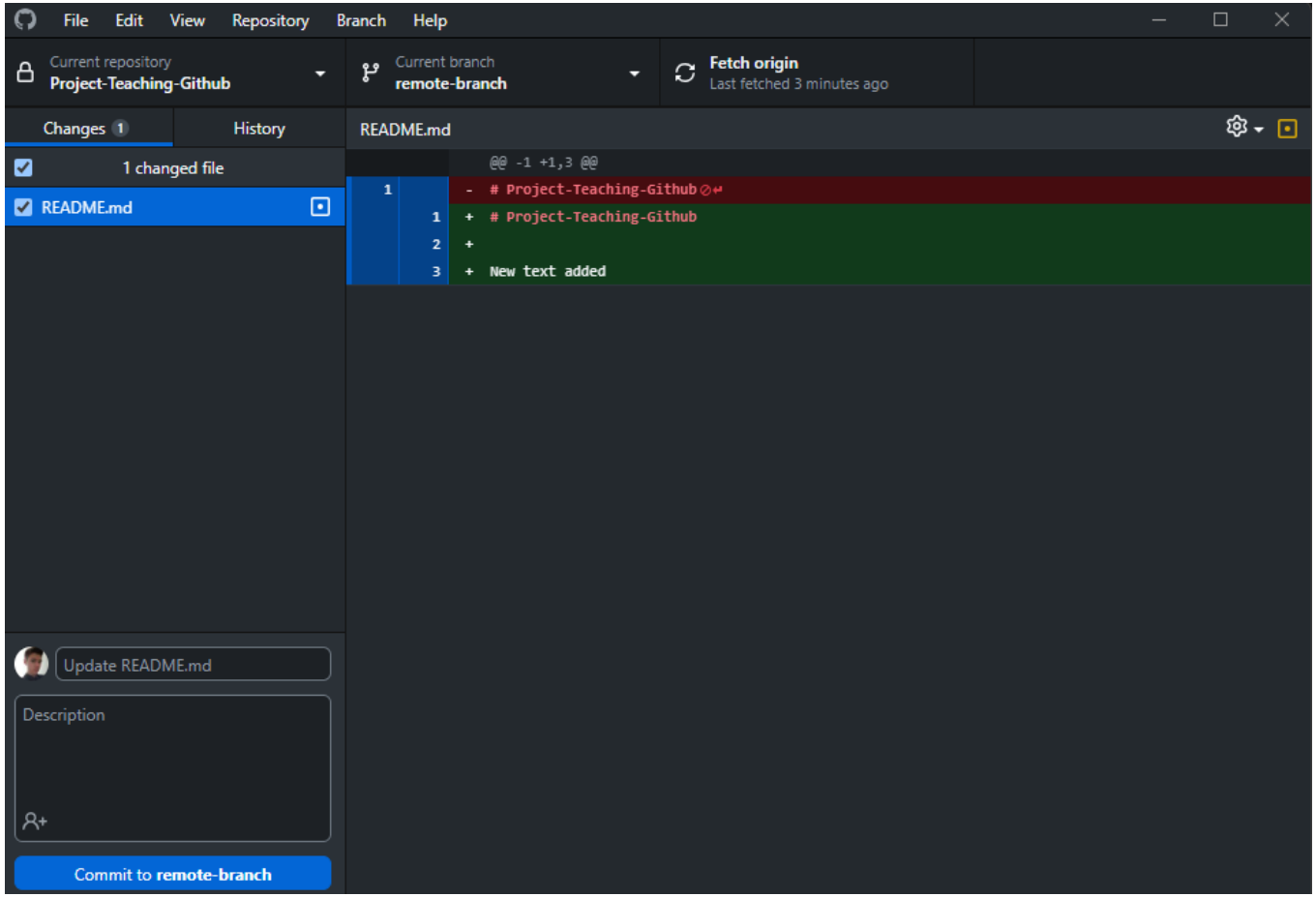


Figure: Faire un commit avant de push le tout en remote

Commit/push des avancements sur une branch

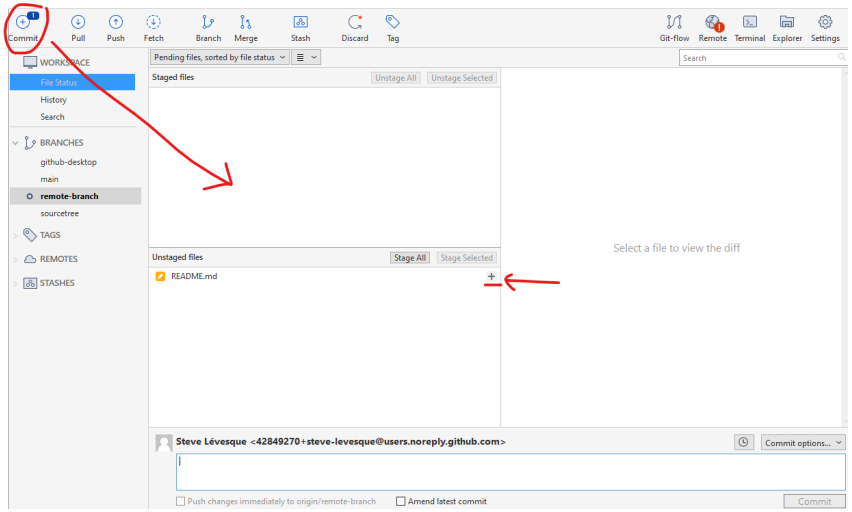


Figure: Commit les fichiers (votre code en général)

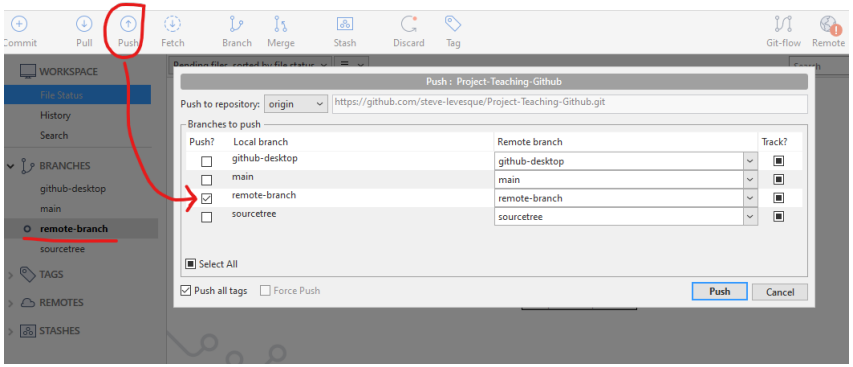


Figure: Push en remote tous vos commits

Merge main dans la branch courrante de dev pour avoir les changements

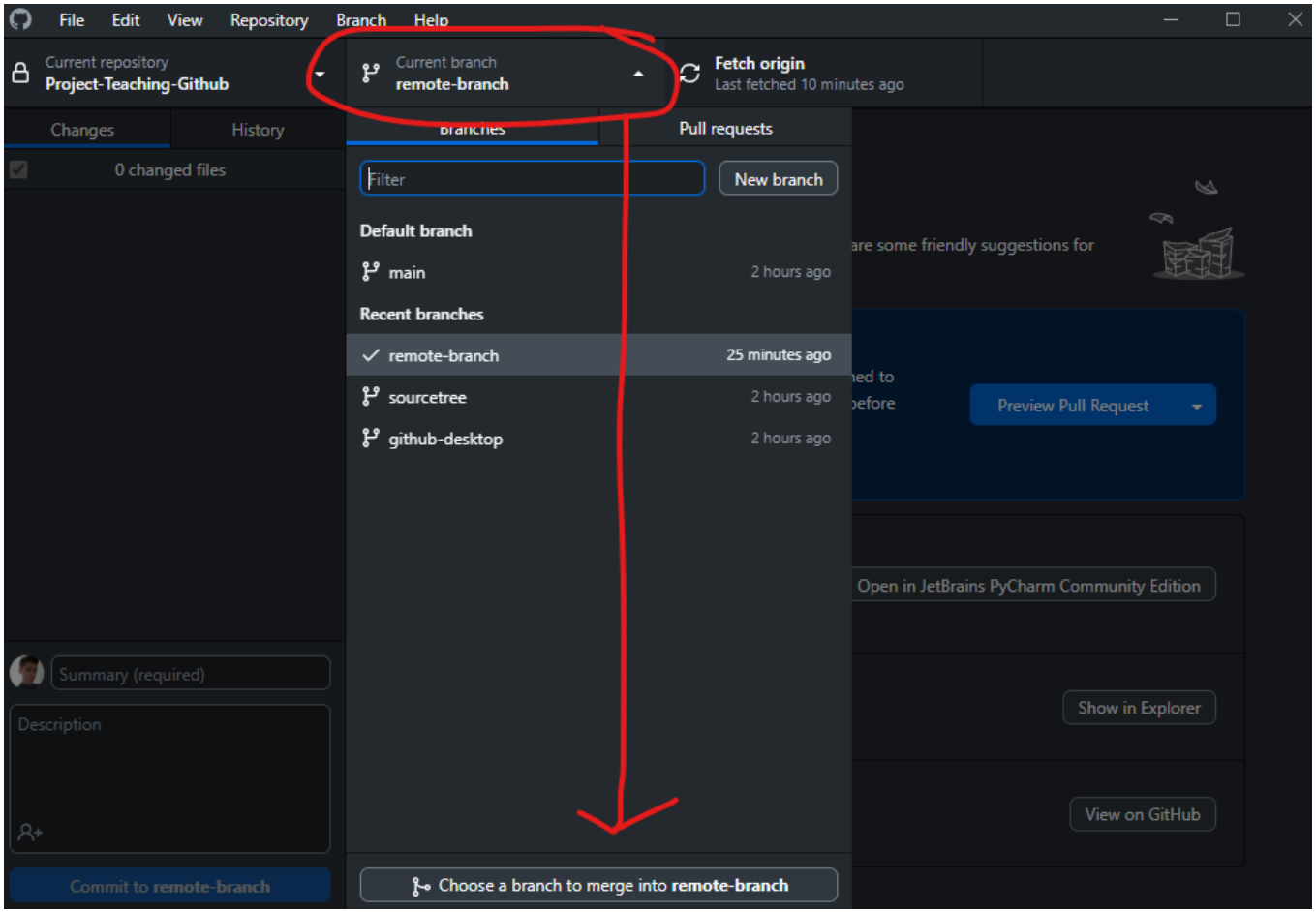


Figure: Lorsqu'une branche importante comme **main** est à jour avec des nouveaux avancements, il faut mettre notre branche auxiliaire à jour en faisant un merge de branches

Merge main dans la branch courrante de dev pour avoir les changements

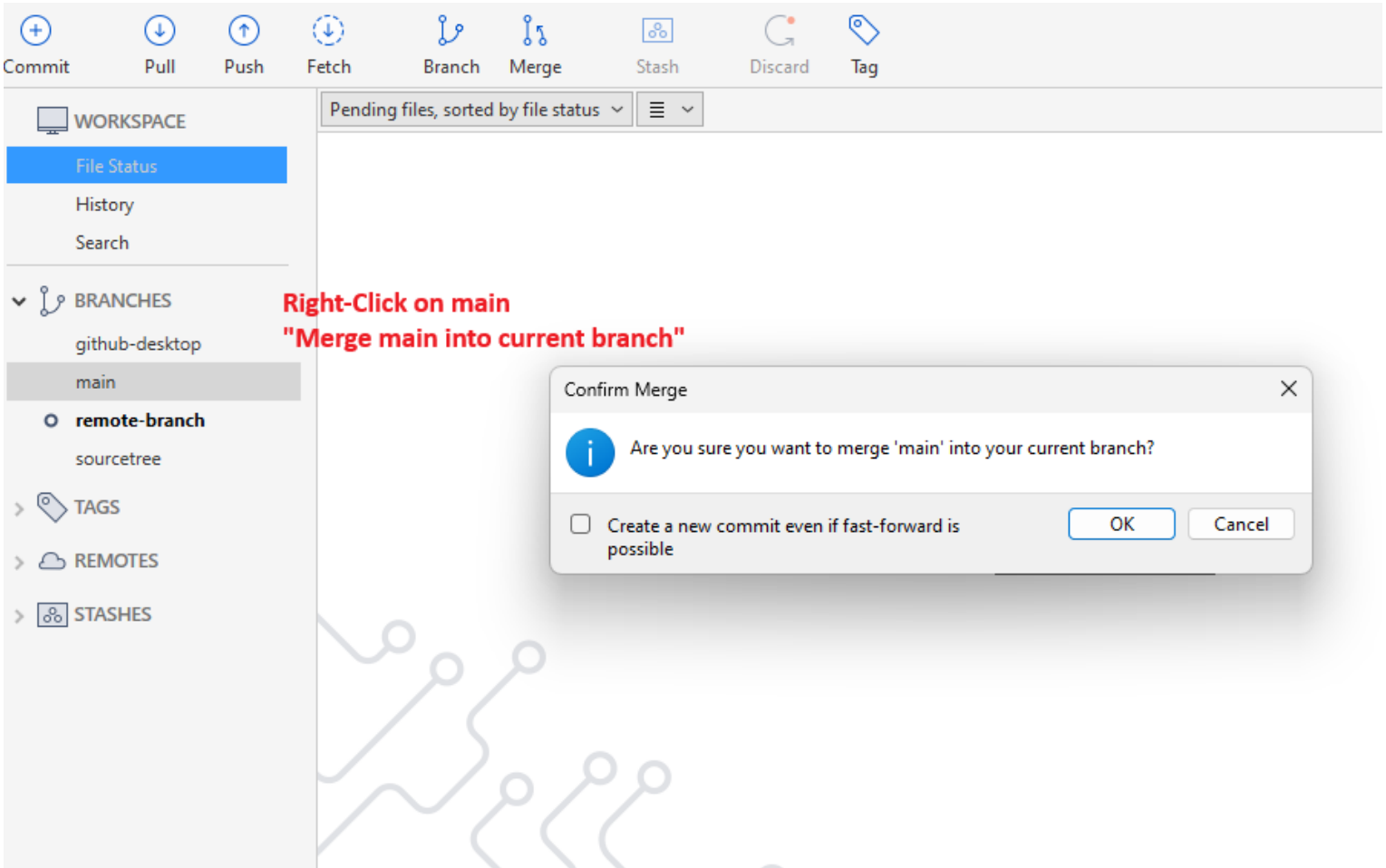


Figure: Un merge de branches avec Sourcetree

Création de “Pull Requests” (PR) à partir de GitHub

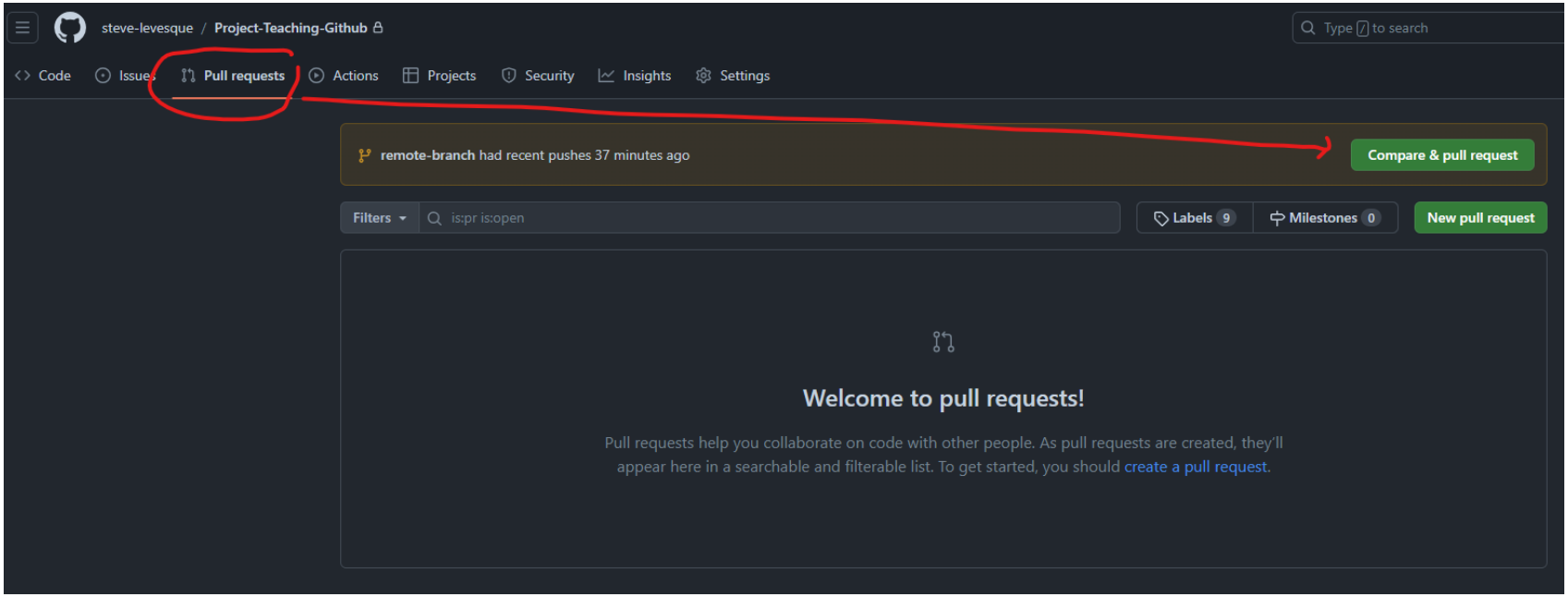


Figure: Lorsqu’un code est prêt à être dans main, il est possible de passer par un processus d’assurance qualité en faisant une PR (Pull request)

Création de “Pull Requests” (PR) à partir de GitHub

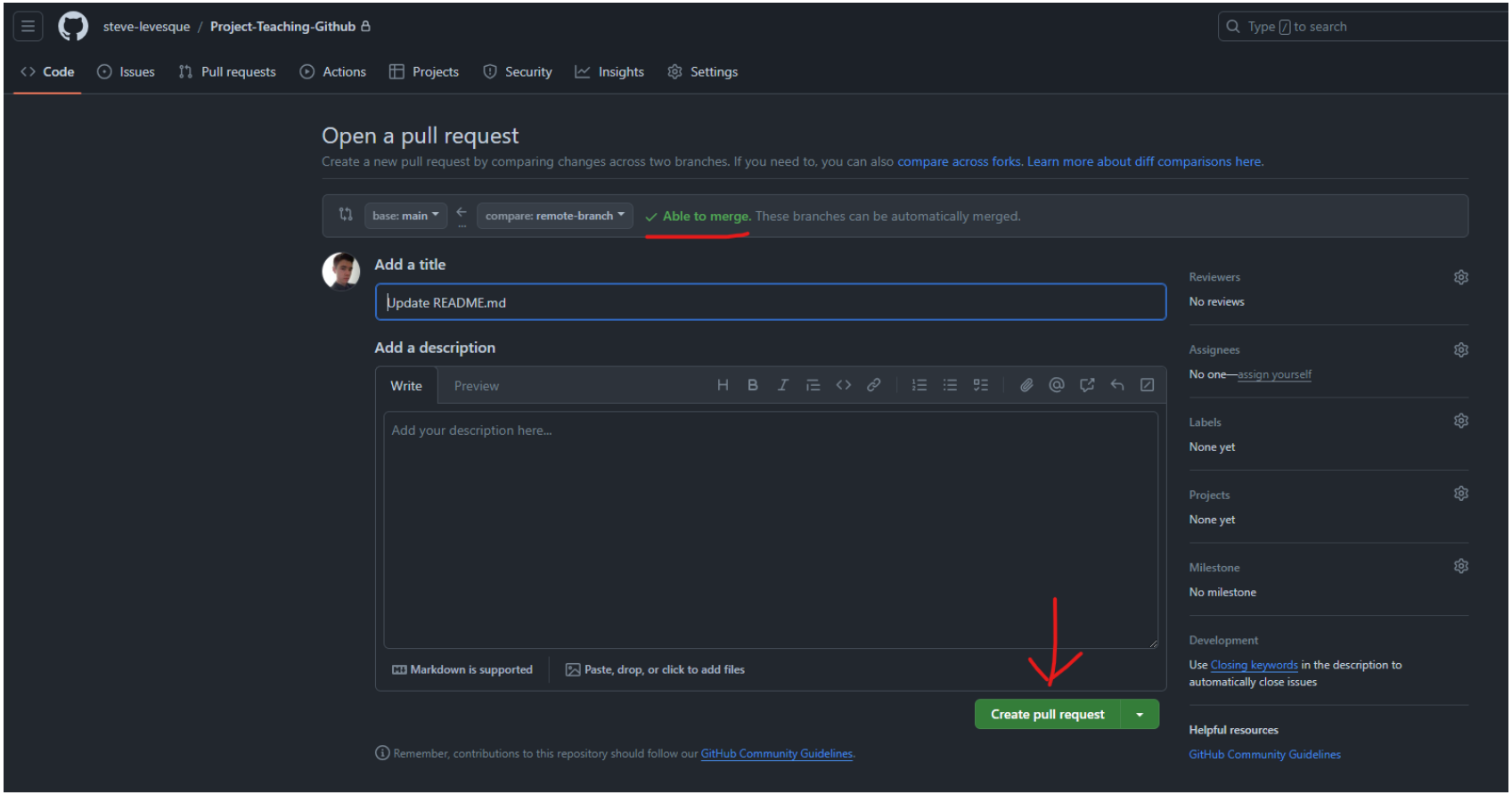


Figure: Il est possible de créer la PR lorsque la branche est faire à partir de main (ou d’une branche basée sur notre cible, mais généralement c’est main)

Création de “Pull Requests” (PR) à partir de GitHub

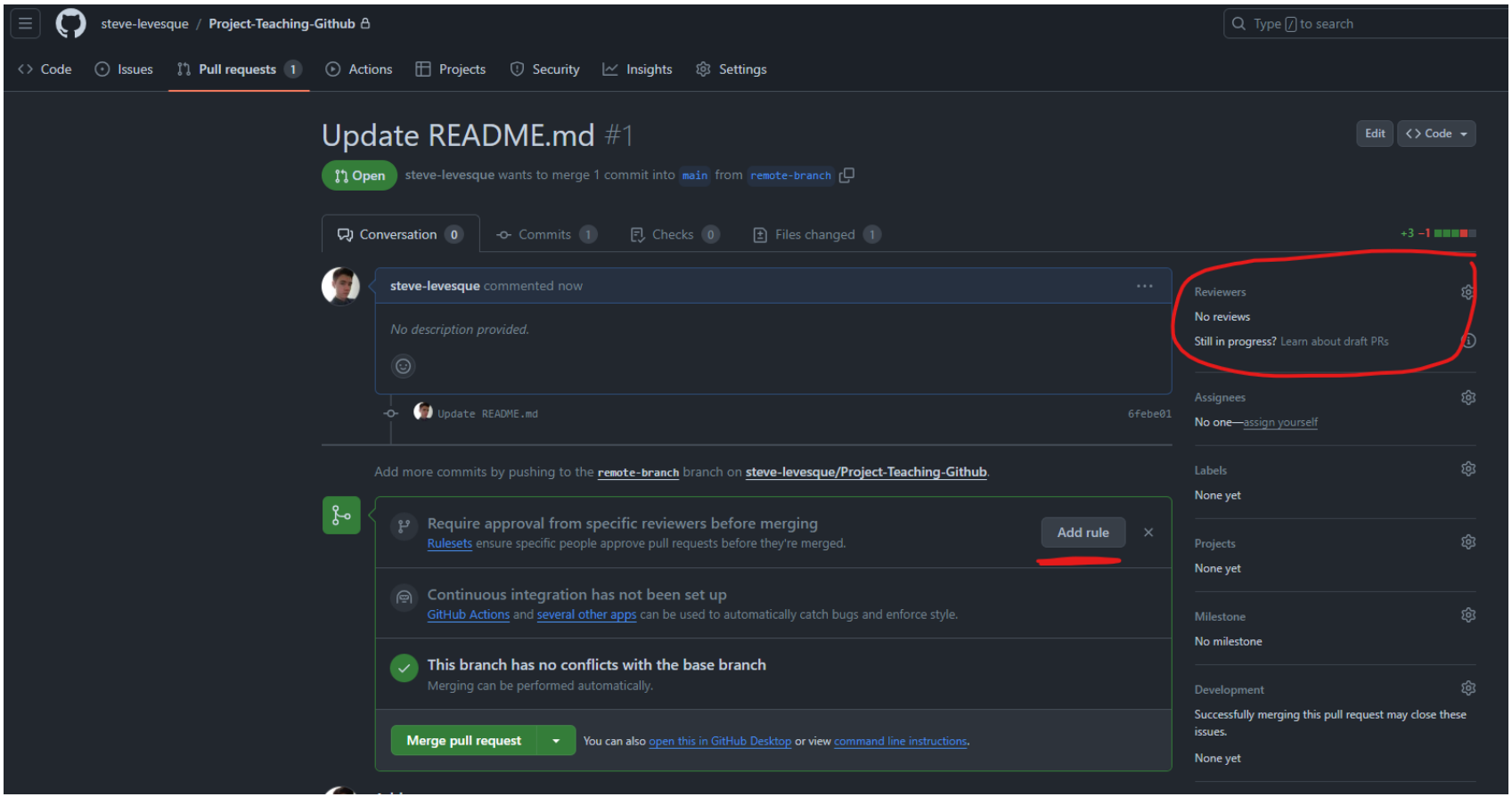


Figure: Le gestionnaire ou développeur sénior peut assigner des “Reviewers” pour aider à réviser le code de la branche auxiliaire du contributeur respectif

Merge conflict

Il est possible de rencontrer un merge conlict si votre branche à un changement vis-à-vis main qui n'est pas un ajout.

Par exemple, si vous changer une ligne de code de main, il va y avoir un conflit puisque il y a deux états pour une même ligne.

Merge conflict

Il faut **supprimer** les annotations (<<, >>, ==) et choisir le code que l'on veut garder. Cela dépends de la situation, alors faites des décisions réfléchies.

```
1        <<<<<<< main
2  [1]   console.log("main")
3        =======
4  [1]   console.log("New code at the same line of old code...")
5        >>>>>>> new_branch
```

Bibliographie

- https://github.com/
- https://desktop.github.com/download/
- https://www.sourcetreeapp.com/