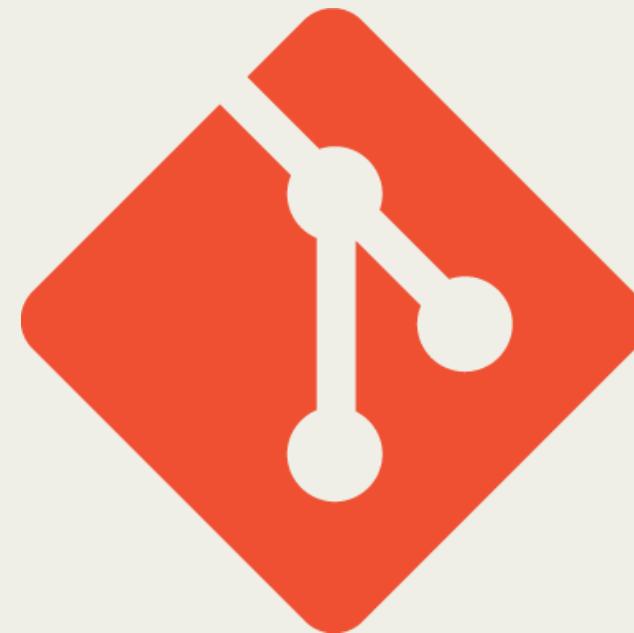


Git & Github



git



À quoi sert Git ?

permet de :

- Sauvegarder l'évolution d'un projet dans le temps
- Travailler à plusieurs sans écraser le travail des autres
- Revenir à une version précédente du code
- Expérimenter dans des "branches" indépendantes

=> outil de versionning

Problématique avant Git

Avant Git :

- Chaque développeur travaillait sur sa copie du projet.
- Fusionner les modifications était manuel et risqué.
- Beaucoup de fichiers “versionnés à la main” :
 - code_final_v3_definitif_2_bis.py
 - rapport_final_final_VRAI.docx

Linus Torvalds

- Crateur du noyau Linux
 - en 2005, ge des milliers de contributions au code source
 - recevait des patchs par e-mail de milliers de contributeurs dans le monde
 - Cre git en 17 jours



Git ≠ GitHub

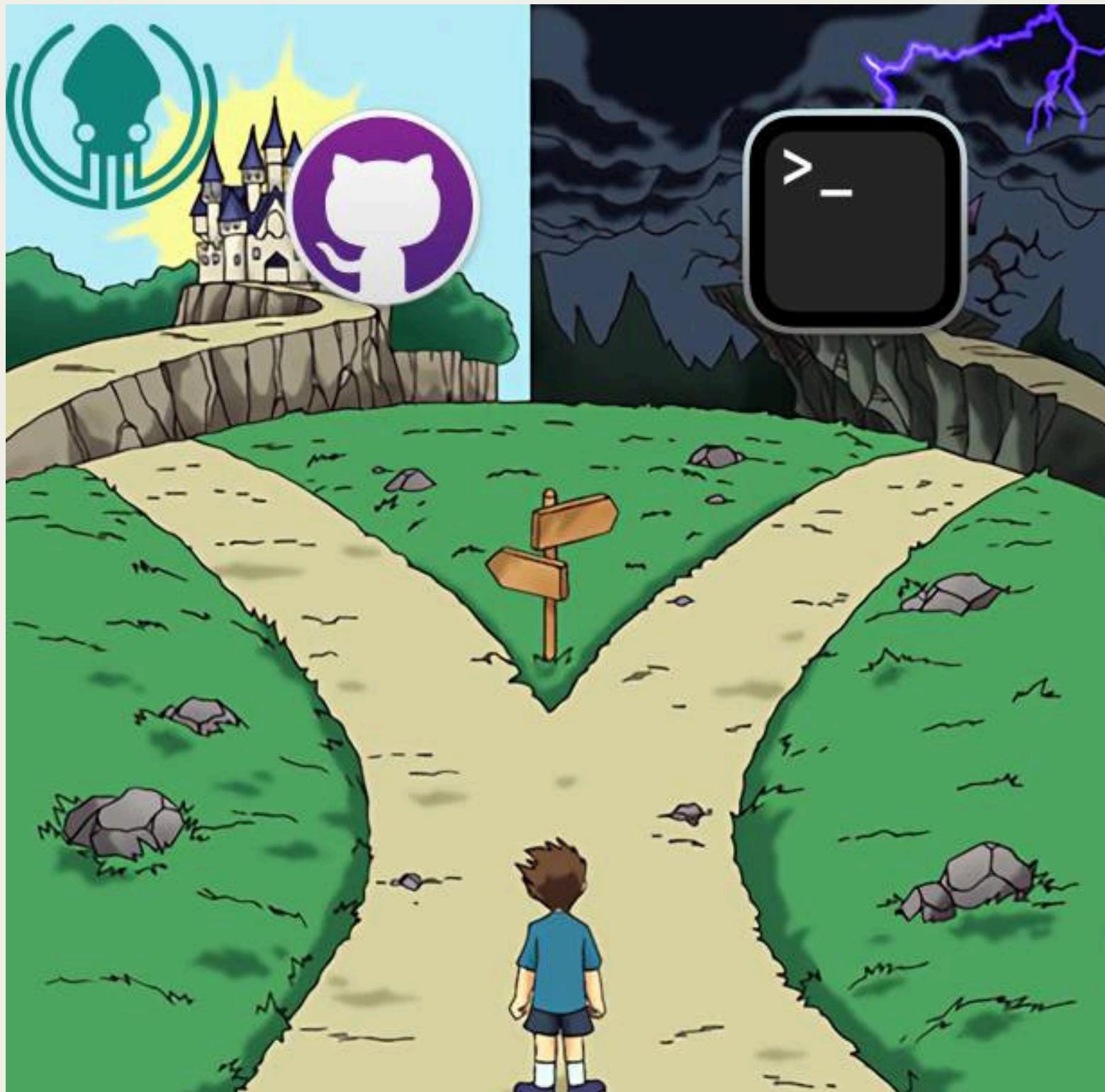
Git

- Git est un logiciel que tu installes sur ton ordinateur.
- Il sert à suivre l'évolution de tes fichiers dans le temps.
- Tu peux l'utiliser sans Internet, tout seul, juste en local.
- Il garde un historique complet de ton projet.

GitHub

- site web
- héberge des projets Git en ligne
- permet de collaborer à plusieurs via
 - les Pull Requests
 - les Issues
 - les discussions

Comment utiliser git ?



Screenshot of a GitHub interface showing a pull request. The top bar shows "File", "Edit", "View", "Repository", "Branch", and "Help". The repository is set to "desktop" and the branch is "#17192". The status bar says "Pull origin Last fetched 8 minutes ago".

The main area shows a list of changes across three files:

- app\src\ui\lib\list\section-list.tsx
- app\src\ui\lib\tooltip.tsx
- app\src\ui\...\tooltipped-content.tsx

Line 140 is highlighted with a blue selection bar, containing code related to row focus handling. A tooltip for "onRowKeyboardFocus" is visible in the center of the screen.

Screenshot of a Windows Command Prompt window titled "C:\Windows\System32\cmd.exe". The command "git status" is run, showing an initial commit with changes to be committed:

```
D:\GFG>git status
On branch master
Initial commit

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

    new file:  Basic Code/C++ Project.cbproj
    new file:  Basic Code/README.md
    new file:  Basic Code/bin/Debug/C++ Project.exe
    new file:  Basic Code/data.in
    new file:  Basic Code/data.out
    new file:  Basic Code/main.cpp
    new file:  Basic Code/obj/Debug/main.o
    new file:  DS/stackPar.cpp
    new file:  DS/stackPar.exe
    new file:  DS/stackPar.o
    new file:  DS/stlsort.cpp
    new file:  DS/structcar.cpp
    new file:  DS/triplesum.cpp
    new file:  Debugging/Debugging.cbproj
    new file:  Debugging/Debugging.depend
    new file:  Debugging/Debugging.layout
    new file:  Debugging/bin/Debug/Debugging.exe
    new file:  Debugging/bin/Release/Debugging.exe
    new file:  Debugging/main.cpp
    new file:  Debugging/obj/Debug/main.o
    new file:  Debugging/obj/Release/main.o
```

Connexion à GitHub

```
git --version
```

```
git config --global user.name "Ton Nom"
```

```
git config --global user.email "ton.email@example.com"
```

```
git config --global credential.helper manager
```

(active le Git Credential Manager, inclus avec Git sur Windows.)

```
git push -u origin main
```

Commande de base: git init

git init

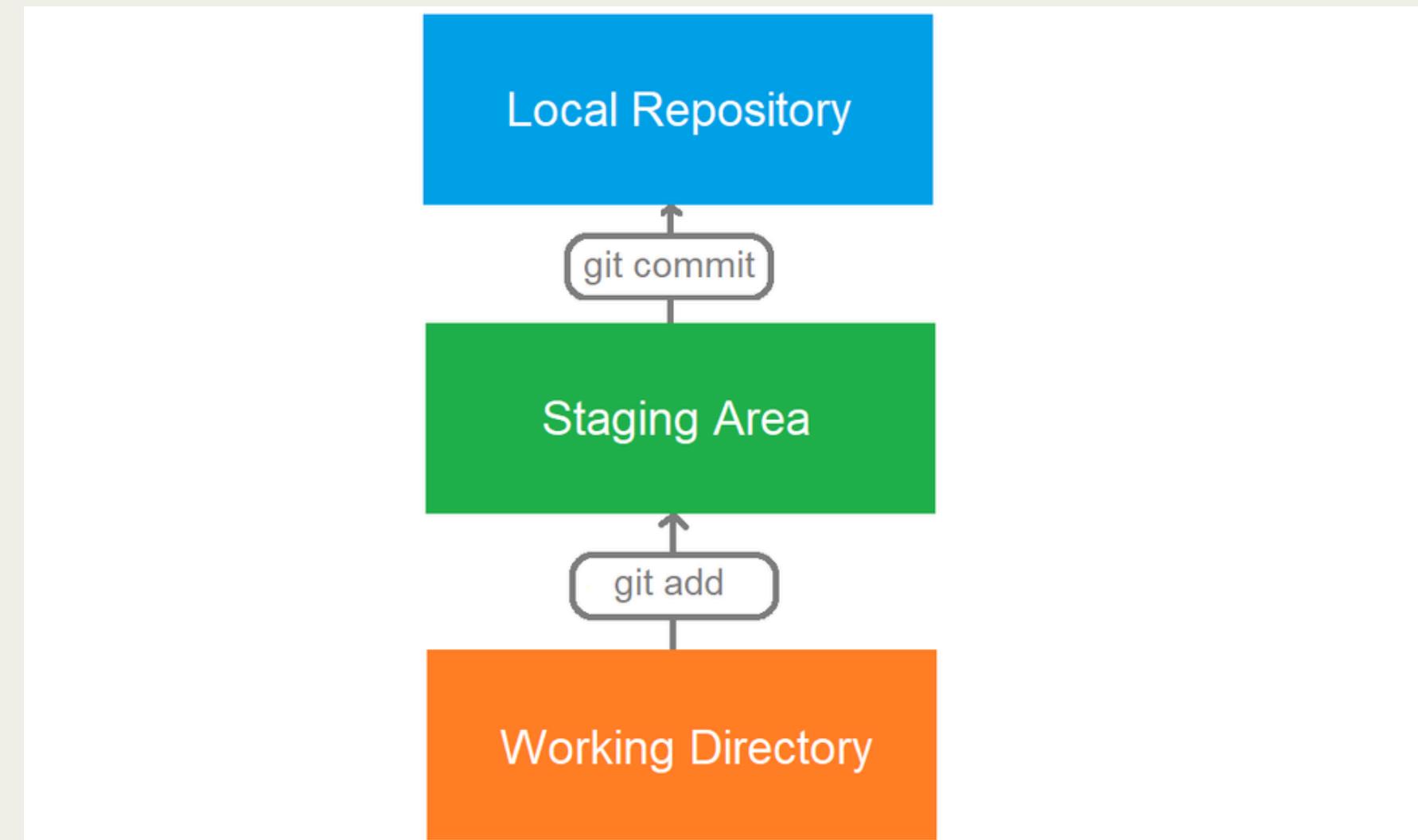
- ➔ Initialise un nouveau dépôt Git dans le dossier courant.
- ➔ Crée un dossier caché .git/ qui contient tout l'historique du projet.

Commande de base: git add

git add nom_du_fichier

git add .

- Ajoute les fichiers modifiés à la zone de préparation (staging area).
- Le . ajoute tous les fichiers modifiés à la fois.



Commande de base: git status

git status

→ Montre les fichiers :

- modifiés mais non ajoutés
- ajoutés mais pas encore commités
- prêts à être commités

```
$ git status
on branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   file1.txt
    new file:   file2.txt

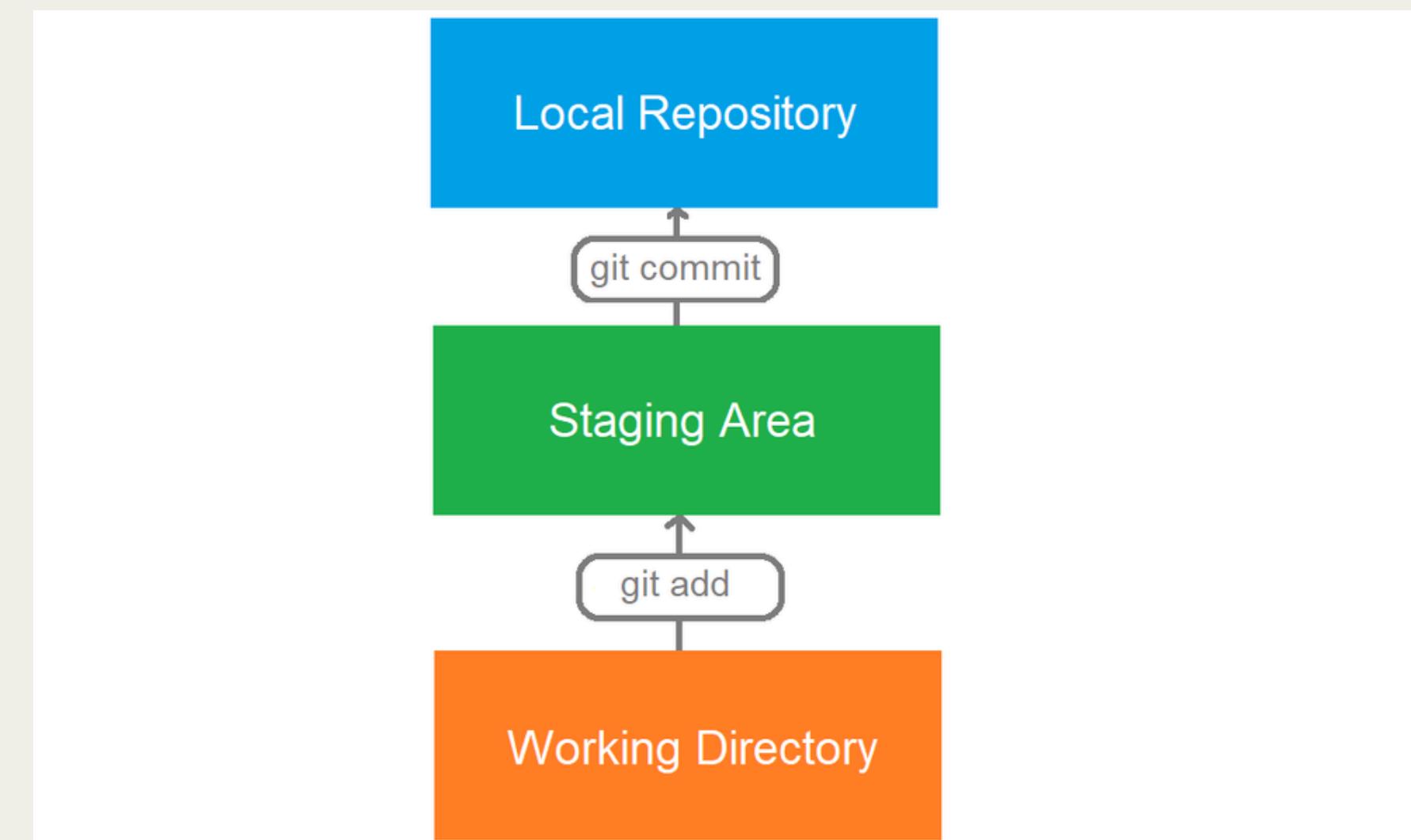
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    file3.txt
```

Commande de base: git commit

`git commit -m "description"`

- Enregistre une photo de l'état du projet à ce moment-là.
- Le message doit décrire ce que tu viens de faire.

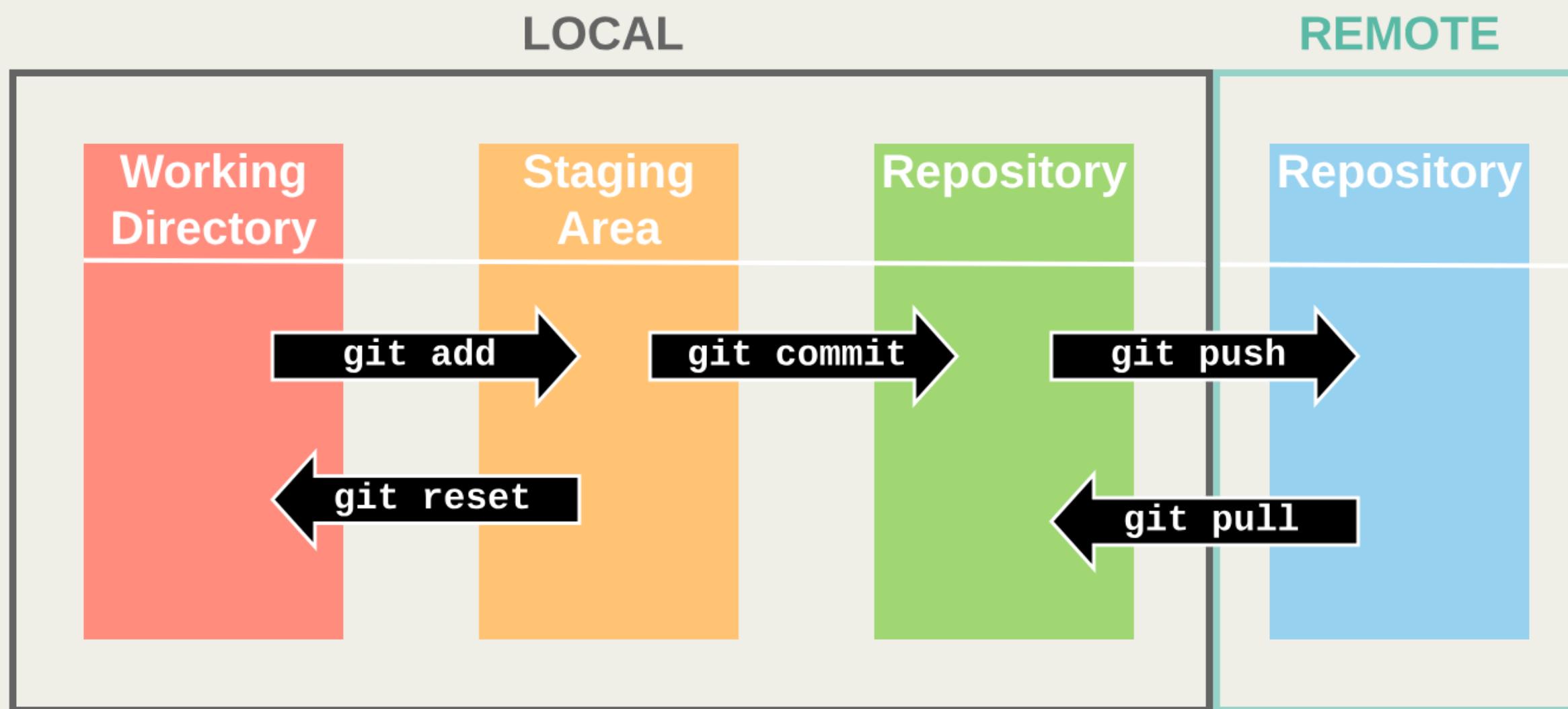
Chaque commit = une version du projet.



Commande de base: git push

git push

→ Envoie les commits locaux vers le dépôt distant (GitHub).



Commande de base: git remote

Objectif: Relier ton dépôt local (sur ton PC) à ton dépôt distant (sur GitHub).

- Crée ton dépôt sur GitHub (<https://github.com/new>)

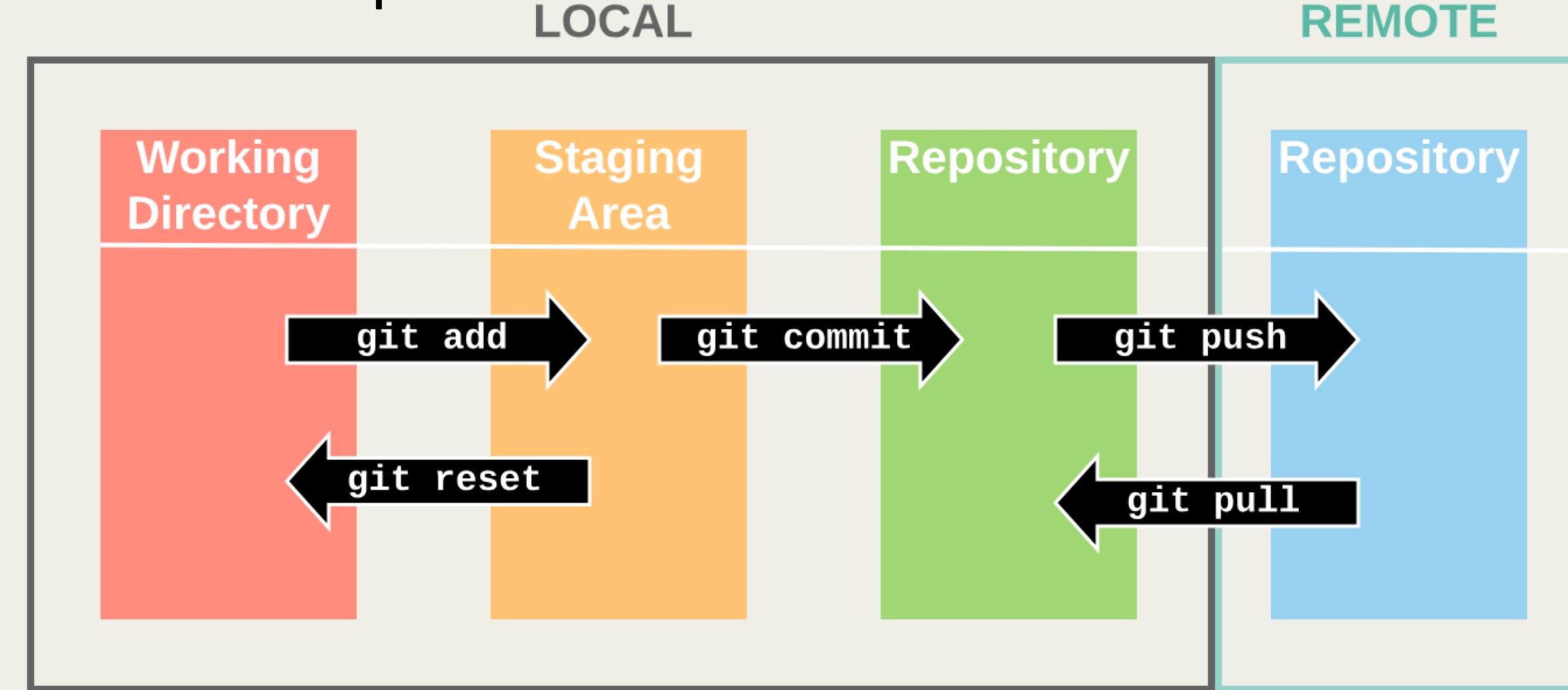
```
git remote add origin https://github.com/ton-pseudo/demo-git.git
```

Exercice: Premier commit

Commande de base: git pull

git pull

- Récupère les dernières modifications du dépôt GitHub vers ton dossier local.
- Combine automatiquement avec ta version actuelle.



Exercice: Premier pull

Commande de base: Récapitulatif

Action	Commande	Description
Initialiser	git init	Crée un dépôt Git local
Préparer	git add .	Ajoute les fichiers à commiter
Sauvegarder	git commit -m "..."	Crée une version
Vérifier	git status	Montre l'état des fichiers
Historique	git log	Liste les commits
Envoyer	git push	Envoie vers GitHub
Mettre à jour	git pull	Récupère les modifs
Copier un projet	git clone	Télécharge un dépôt existant

ignorer les fichiers inutiles: gitignore

Empêcher Git de suivre les fichiers temporaires, volumineux ou générés automatiquement.

=> On garde uniquement le code source et les assets importants.

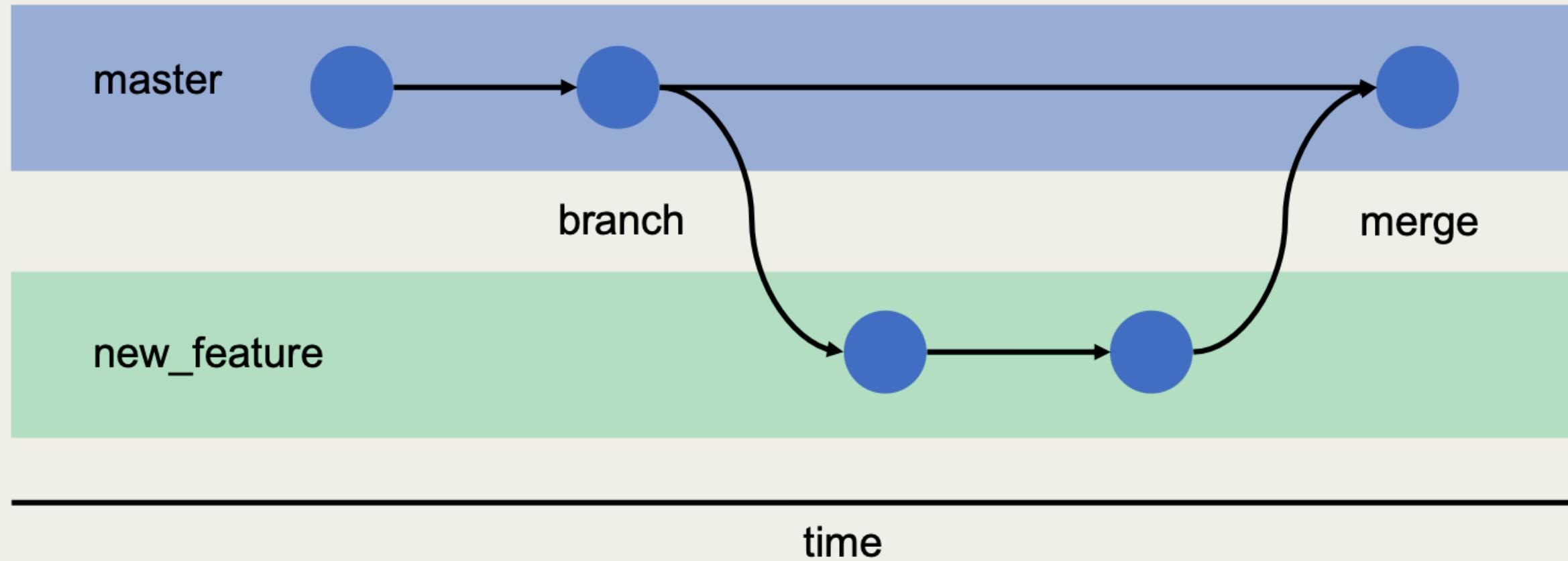
Générer un .gitignore automatiquement

The image shows a screenshot of the .gitignore.io website. The main heading is ".gitignore.io" in a large, bold, blue font. Below it is the subtext "Créez des fichiers .gitignore utiles à votre projet". At the bottom of the page, there is a search bar containing the text "Unity" and a green button labeled "Créer". At the very bottom, there are links for "Code Source" and "Documentation".

Exercice: Cree un gitignore & push le dernier projet

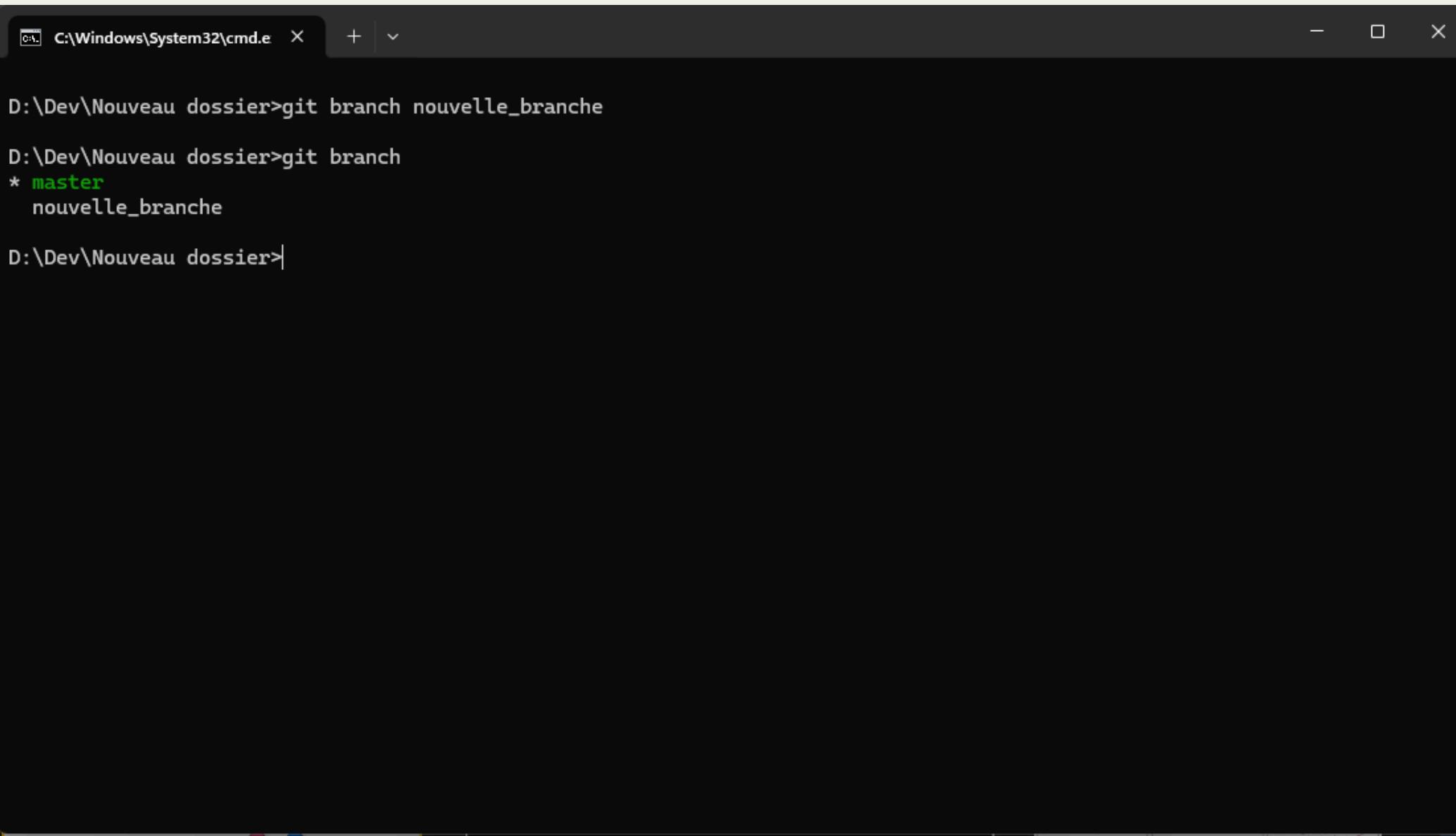
Les branches

Objectif : travailler sur une nouvelle idée sans casser le projet principal.



Les branches: Créer une branche

git branch nom_de_branch



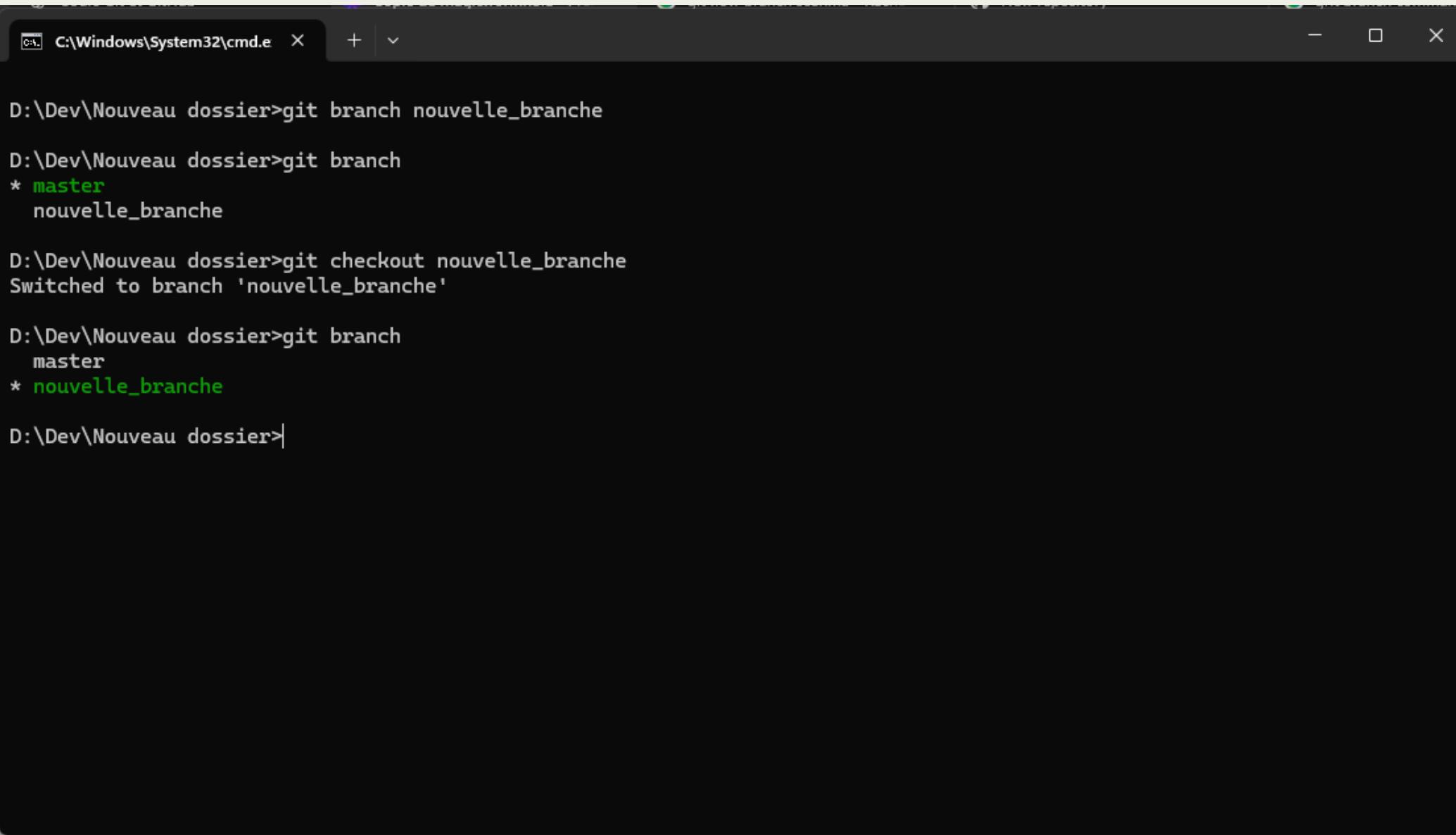
A screenshot of a Windows Command Prompt window titled "C:\Windows\System32\cmd.e". The window shows the following command-line session:

```
D:\Dev\Nouveau dossier>git branch nouvelle_branche
D:\Dev\Nouveau dossier>git branch
* master
  nouvelle_branche
D:\Dev\Nouveau dossier>
```

The output shows that a new branch named "nouvelle_branche" has been created, in addition to the existing "master" branch.

Les branches: Changer de branche

git checkout nom_de_branch



```
C:\Windows\System32\cmd.e D:\Dev\Nouveau dossier>git branch nouvelle_branch
D:\Dev\Nouveau dossier>git branch
* master
  nouvelle_branch

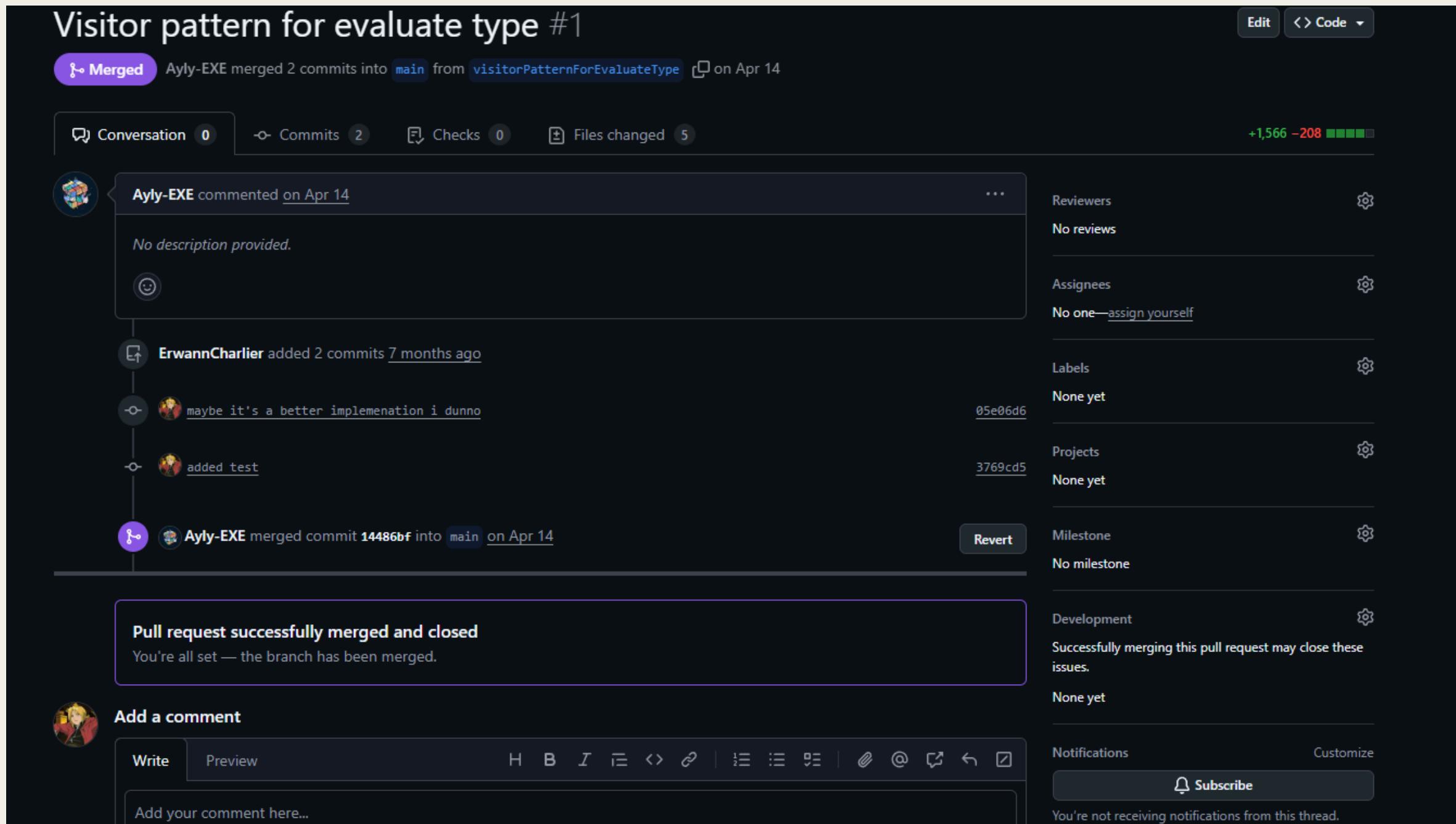
D:\Dev\Nouveau dossier>git checkout nouvelle_branch
Switched to branch 'nouvelle_branch'

D:\Dev\Nouveau dossier>git branch
  master
* nouvelle_branch

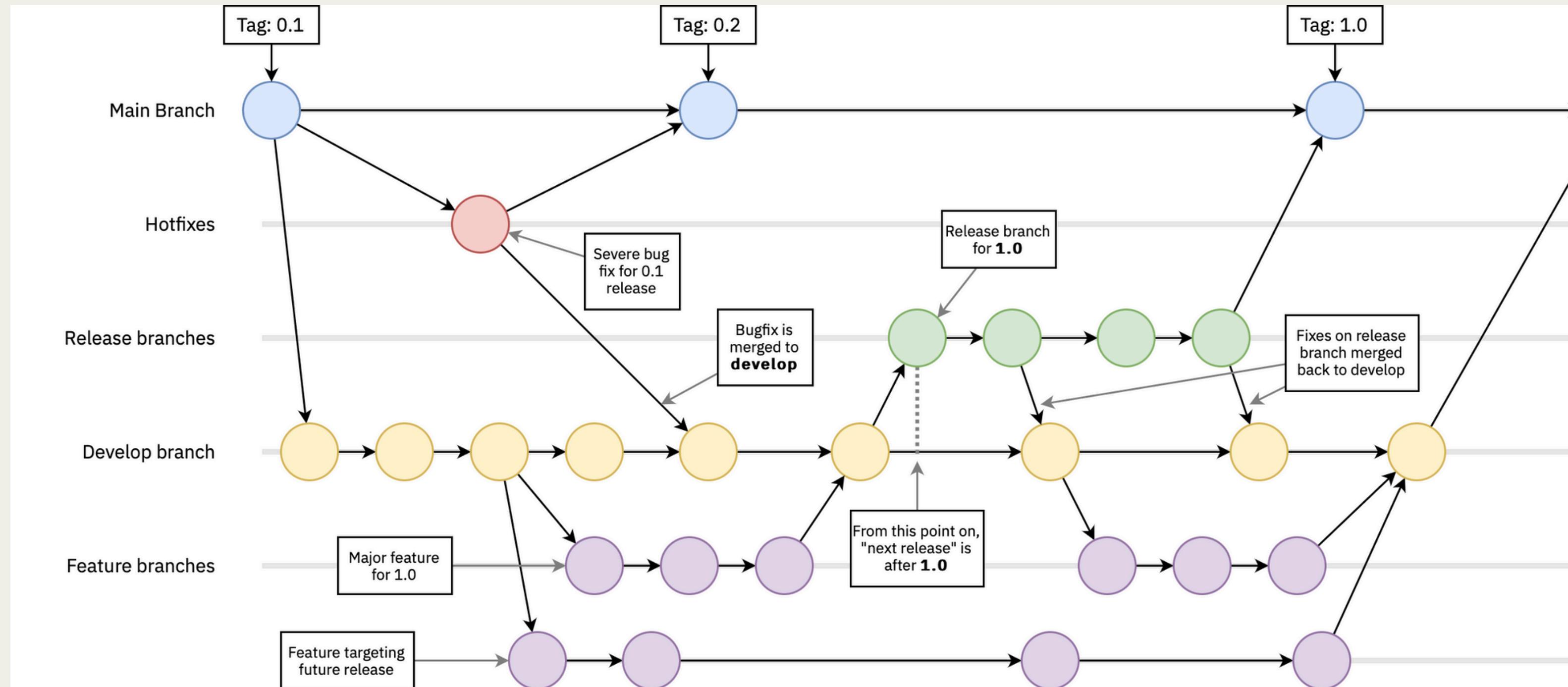
D:\Dev\Nouveau dossier>
```

Les branches: fusion de branches

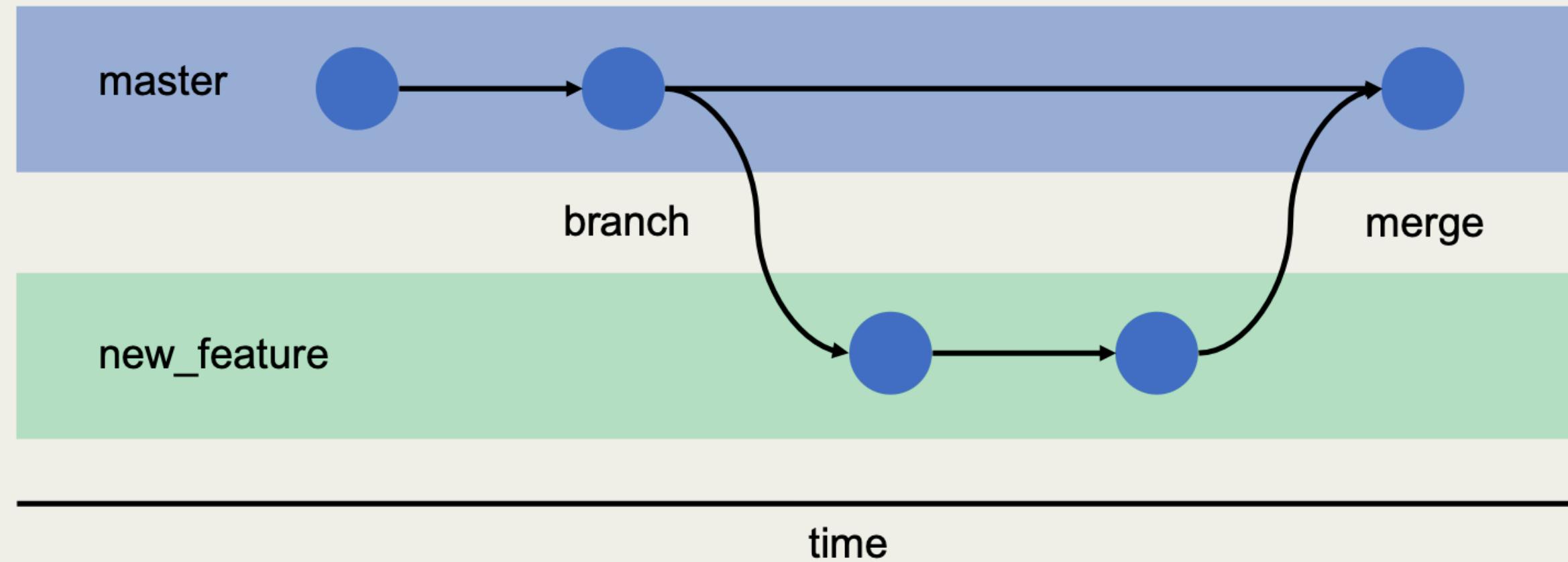
via pull request sur github



Les branches: git flow

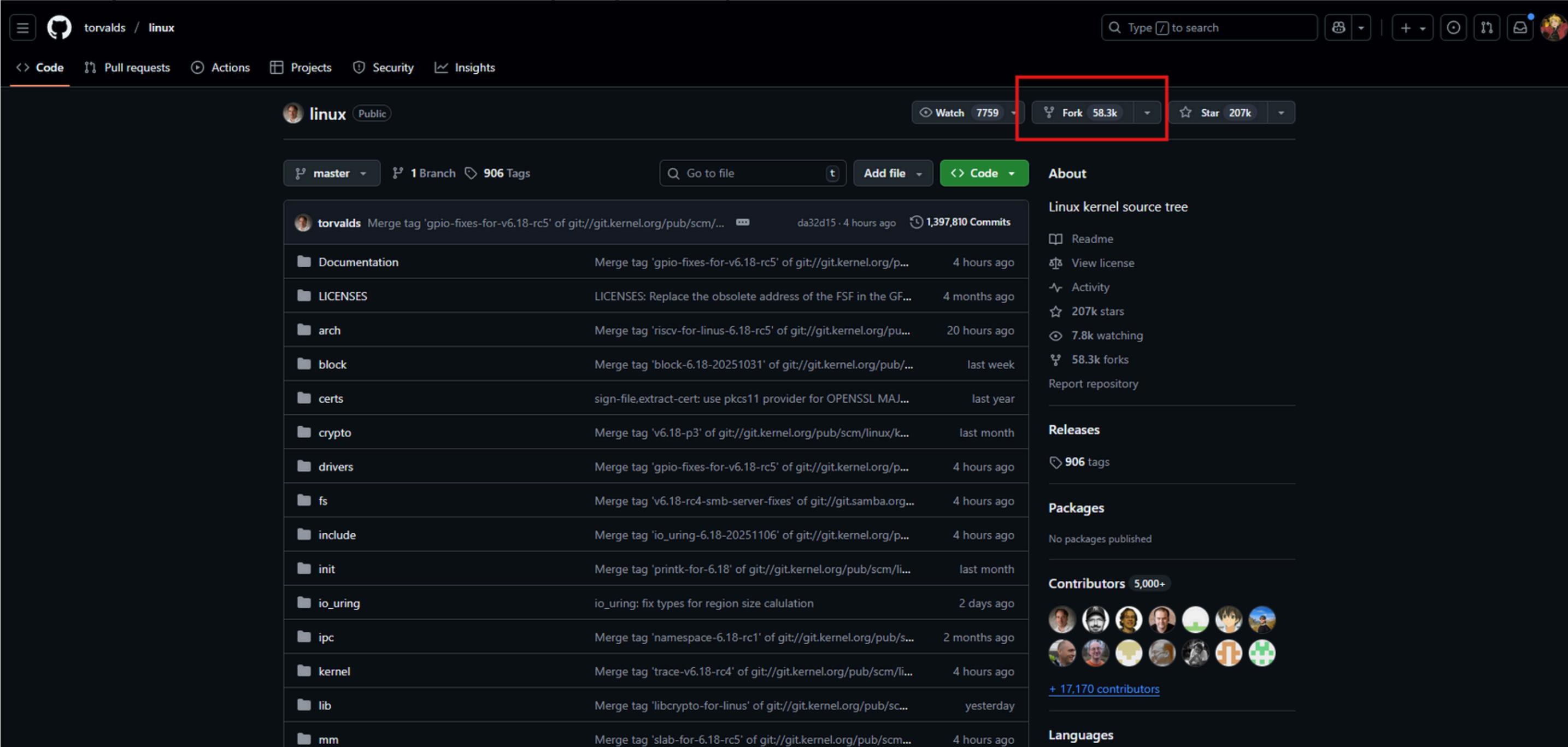


Exercice: Cree une nouvelle branche et la fusionner dans master



Qu'est-ce qu'un “fork” ?

- Un fork = une copie personnelle d'un dépôt GitHub existant.
- Il te permet de tester, modifier, ou améliorer un projet sans toucher à l'original.
- Utilisé pour contribuer à des projets open-source.



“Je duplique le projet → je fais mes modifs → je propose mes améliorations.”

Pourquoi forker ?

- Corriger un bug ou améliorer une fonctionnalité
- Traduire ou documenter un projet
- Proposer de nouvelles idées au créateur du repo

Tu n'as pas besoin d'être membre du projet pour contribuer
=> c'est tout l'intérêt des forks.

Exercice: Forker un projet, le modifier puis faire une pull Request