Introduction à Git

Introduction

Pourquoi utiliser Git ?

- **Mise en situation : un projet logiciel qui évolue dans le temps et sur lequel une équipe de plusieurs membres travaille.**
  - Actuellement, les développeurs modifient le code dans un répertoire partagé en réseau.
  - Le code sur le serveur est toujours le plus récent (pas d'historique).

Pourquoi utiliser Git ? (suite)

- **Difficultés possibles :**
  - Retour en arrière suite à un bogue ou retrait d'une modification.
  - Pas de méthode claire pour gérer plusieurs versions en parallèle (p.ex. prochaine relâche vs. correctifs en production).
  - Conflits de modifications (p.ex. fichiers écrasés si deux personnes travaillent dans le même module).
  - Pas d'historique : aucune information sur quand et par qui quelque chose a été changé.

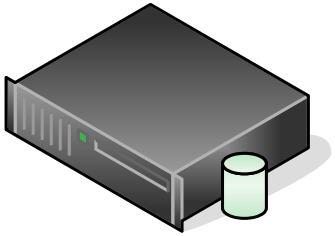
Qu’est-ce qu’un SCV?

- **Un système de contrôle de versions (SCV) sert à gérer le versionnement d’un ensemble de fichiers et de répertoires.**
  - Garde l’historique des modifications.
  - Permet de revenir à une version antérieure.
  - Gère les modifications concurrentes.

Architecture des SCV

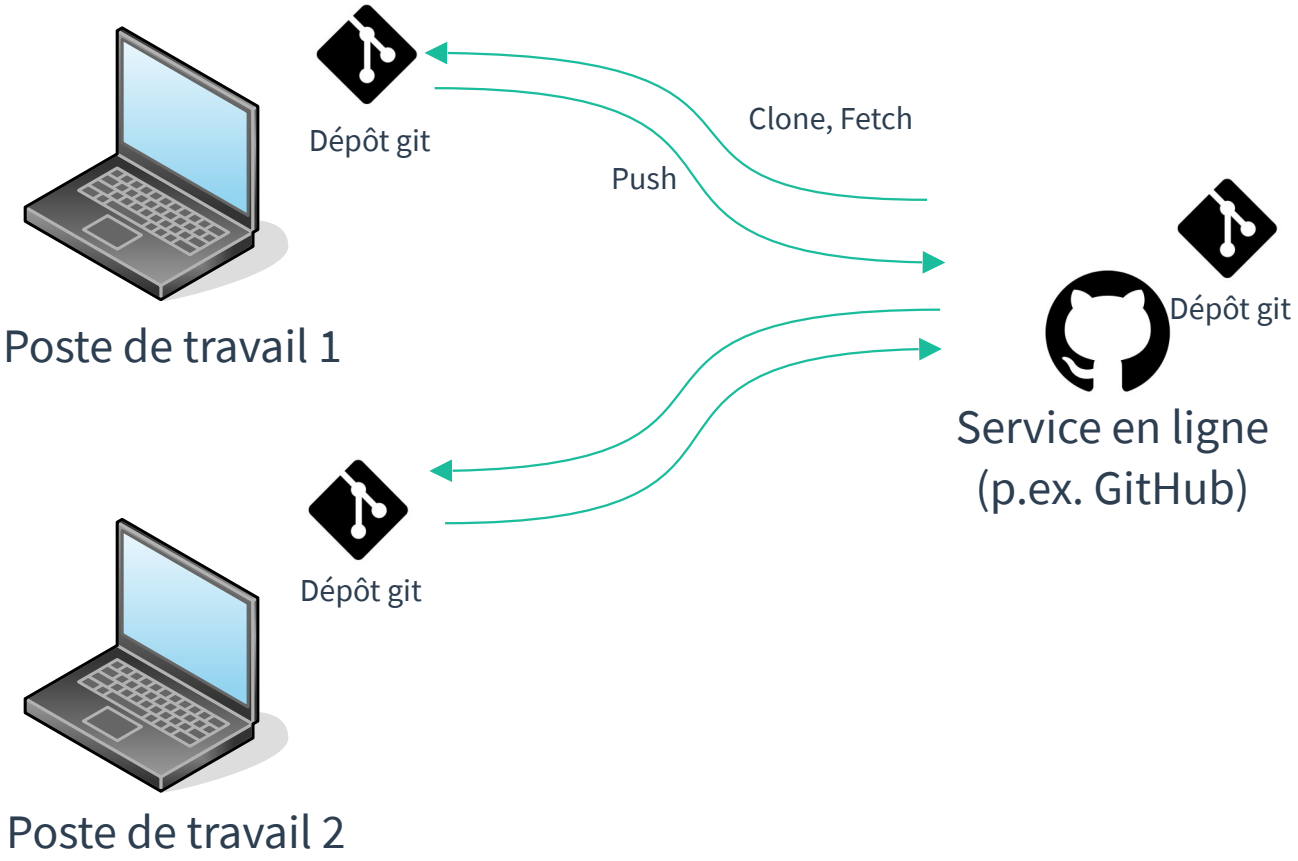
- **Traditionnellement, architecture client-serveur :**
  - CVS, Subversion, Perforce, MS SourceSafe

Copie locale
(une version/branche seulement)



Dépôt centralisé
(serveur)

Git : un SCV distribué



Git : quelques faits

- **Version initiale conçue en 2005 par Linus Torvalds, pour les fins du développement de Linux.**
- **Caractéristiques souhaitées :**
  - Vitesse
  - Conception simple
  - Support pour développements non linéaires (branches)
  - Complètement distribué
  - Capacité à gérer les projets d’envergure

Git : fondements

- **Git est avant tout un logiciel (en ligne de commande) qui roule en local**
  - La plupart des opérations sont locales et opèrent sur un dépôt dans un format spécial (dans le répertoire .git).
  - Certaines opérations (clone, fetch, push) permettent de synchroniser un dépôt distant.
  - Git permet de prendre une image instantanée d'un ensemble de fichiers et d'en faire une version.

Git : fondements

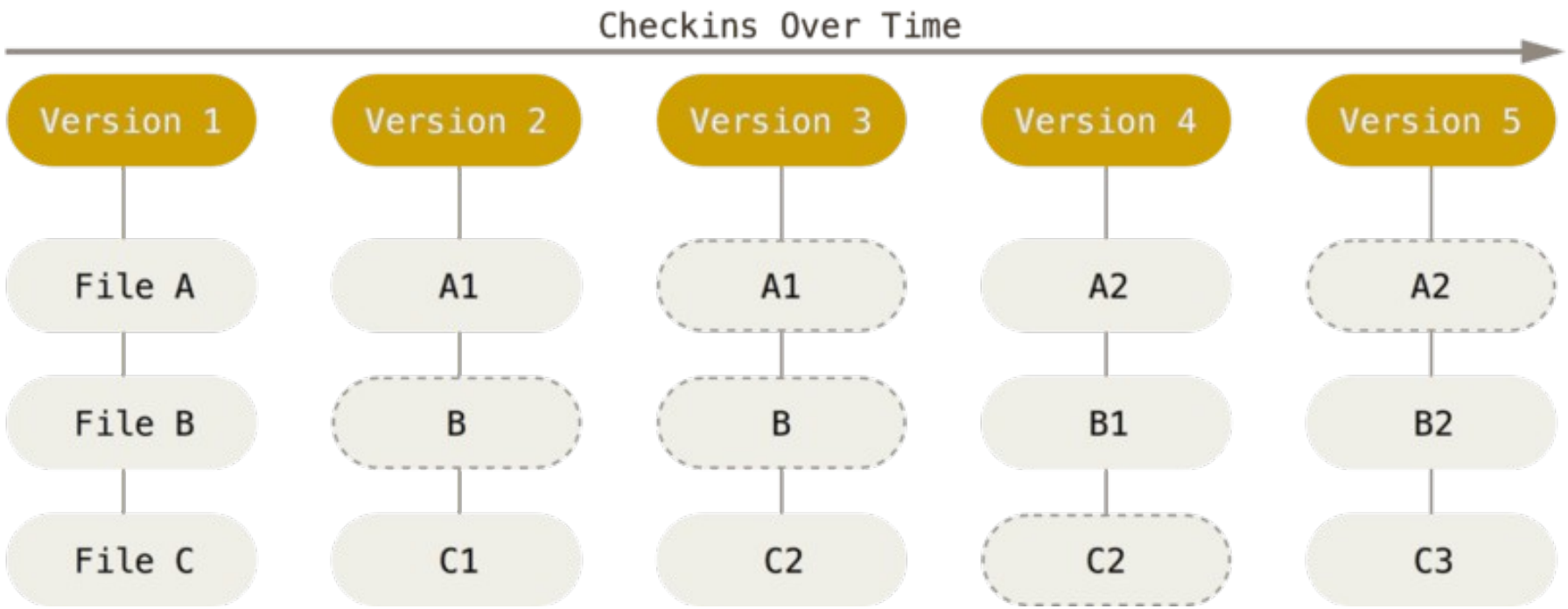


Illustration tirée du livre en ligne Pro Git : https://git-scm.com/book/en/v2

Git : commandes de base

Git : création d'un dépôt

- **Une fois Git installé, on peut créer un dépôt dans un répertoire (que celui-ci soit vide ou non) avec la commande *git init*.**

```
D:\formations\git_repo>git init
Initialized empty Git repository in D:/formations/git_repo/.git/
```

- **Cela crée un répertoire .git qui contient quelques fichiers et répertoires.**
  - Seul le fichier config est éditable (pour les paramètres spécifiques au dépôt).
  - Les autres fichiers servent au stockage du dépôt et ne doivent pas être altérés. Il s'agit en quelque sorte d'une « base de données » contenant les fichiers versionnés dans le dépôt.

Git : état du dépôt

- Le dépôt ainsi créé existe seulement sur le disque local.
- On peut voit l'état du dépôt avec *git status* :

```
D:\formations\git_repo>git status
On branch main

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

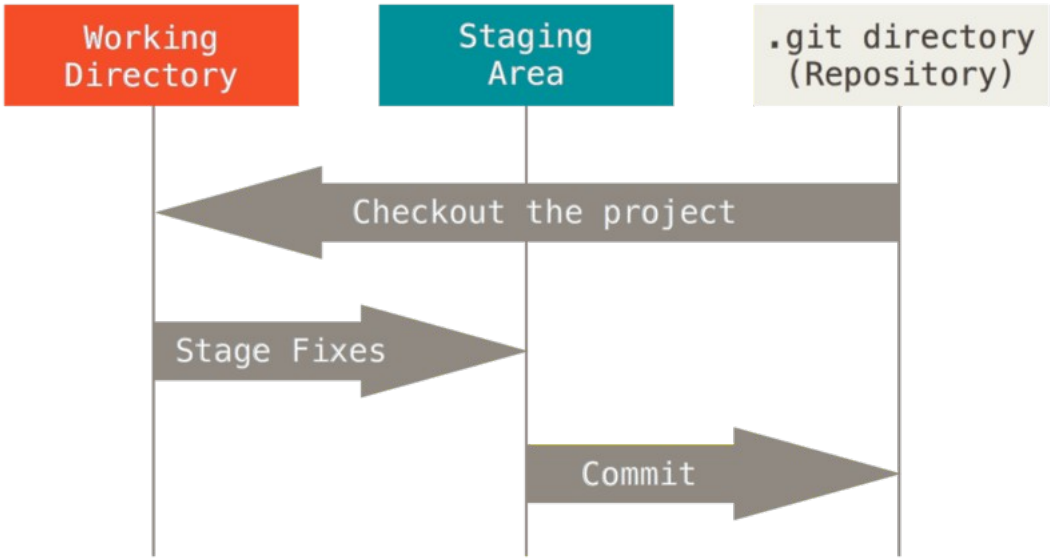
- On peut voir qu'on se situe dans la branche *main*, qui est la branche de travail par défaut.
  - Nous verrons le concept de branche plus loin.
- On peut voir aussi qu'il n'y a aucun commit pour le moment.

Git : états des fichiers

- **Pour travailler avec git, on utilise différentes commandes qui agissent sur les fichiers locaux (sur disque) et le dépôt.**
- **Un fichier peut résider dans trois états dans git :**
  - Modifié : le fichier a été modifié par rapport au dépôt.
  - Indexé *(staged)* : un fichier modifié (ou ajouté) a été marqué pour faire partie du prochain commit.
  - Validé *(commited)* : le fichier est stocké dans le dépôt.

Git : états des fichiers (suite)

- **Ces états correspondent aux trois « sections » d'un projet git :**
  - Le dépôt (contenu du répertoire .git)
  - La zone d'index (*staging area*)
  - Le répertoire de travail



Ajouter des fichiers à l'index

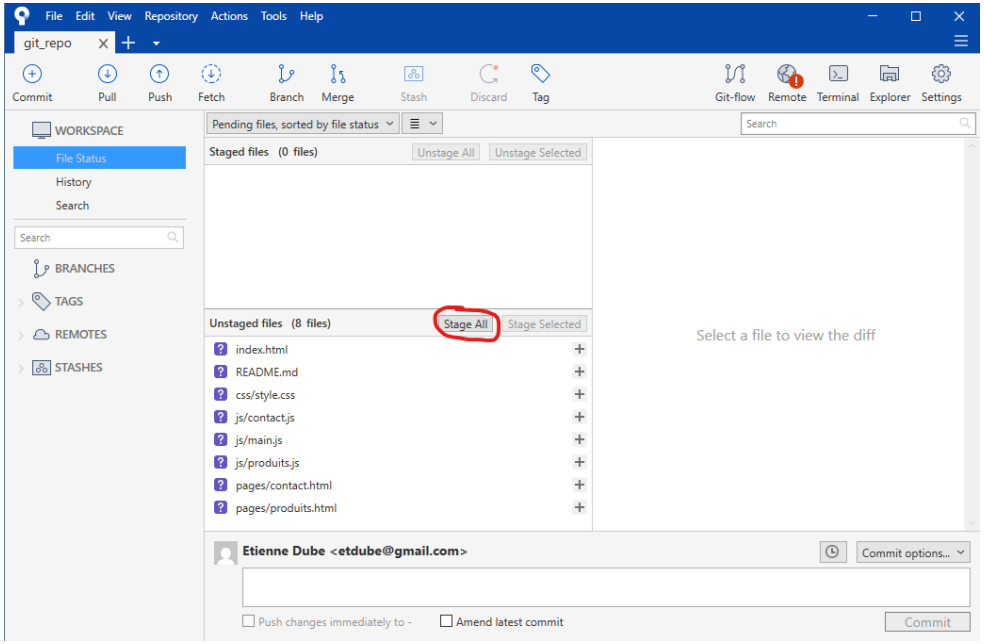
- **Commande : git add**
  - Ajouter un seul fichier : git add <fichier>
  - Ajouter tous les fichiers : git add .

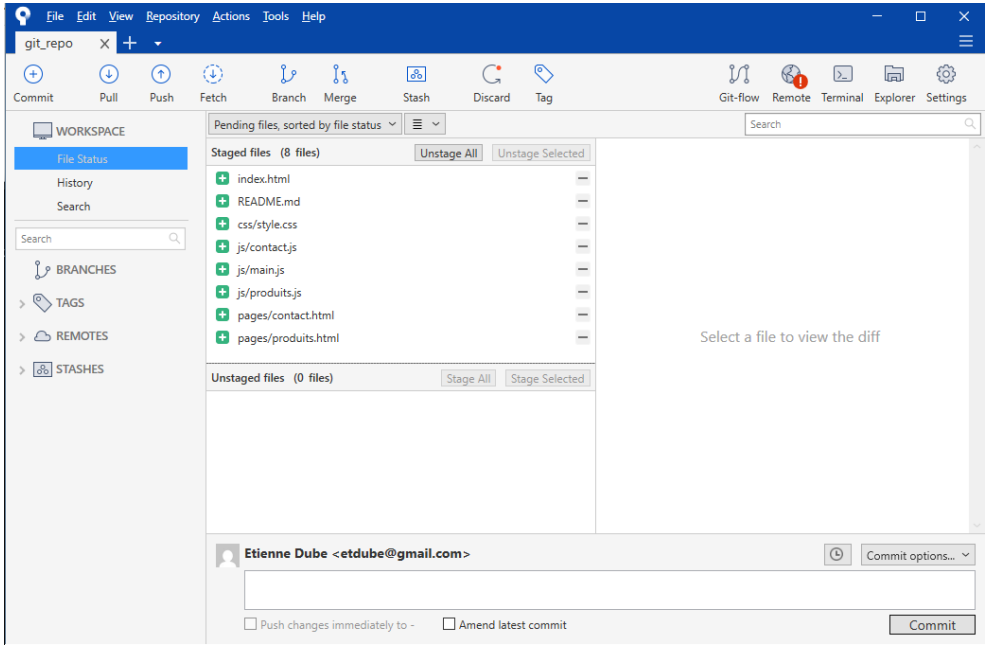
```
D:\formations\git_repo>git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md
        css/
        index.html
        js/
        pages/

nothing added to commit but untracked files present (use "git add" to track)

D:\formations\git_repo>git add index.html

D:\formations\git_repo>git status
On branch main

No commits yet

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

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md
        css/
        js/
        pages/
```

```
D:\formations\git_repo>git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md
        css/
        index.html
        js/
        pages/

nothing added to commit but untracked files present (use "git add" to track)

D:\formations\git_repo>git add .

D:\formations\git_repo>git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md
        new file:   css/style.css
        new file:   index.html
        new file:   js/contact.js
        new file:   js/main.js
        new file:   js/produits.js
        new file:   pages/contact.html
        new file:   pages/produits.html
```

Ajouter des fichiers à l'index (suite)

- Avec Sourcetree





Retirer les fichiers de l'index

- Commande : git rm -r --cached .

```
D:\formations\git_repo>git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md
        new file:   css/style.css
        new file:   index.html
        new file:   js/contact.js
        new file:   js/main.js
        new file:   js/produits.js
        new file:   pages/contact.html
        new file:   pages/produits.html


D:\formations\git_repo>git rm -r --cached .
rm 'README.md'
rm 'css/style.css'
rm 'index.html'
rm 'js/contact.js'
rm 'js/main.js'
rm 'js/produits.js'
rm 'pages/contact.html'
rm 'pages/produits.html'

D:\formations\git_repo>git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md
        css/
        index.html
        js/
        pages/

nothing added to commit but untracked files present (use "git add" to track)
```

Les commits dans Git

- **Un *commit* ajoute une version aux fichiers dans le dépôt, à partir des fichiers dans l'index.**
  - Peut contenir des ajouts, modifications et suppressions de fichiers.
- **Un commit a un identifiant unique sous la forme d'un *hash* SHA-1.**
  - p.ex. : 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
- **Un commit comporte aussi certaines informations : auteur, date et commentaire.**

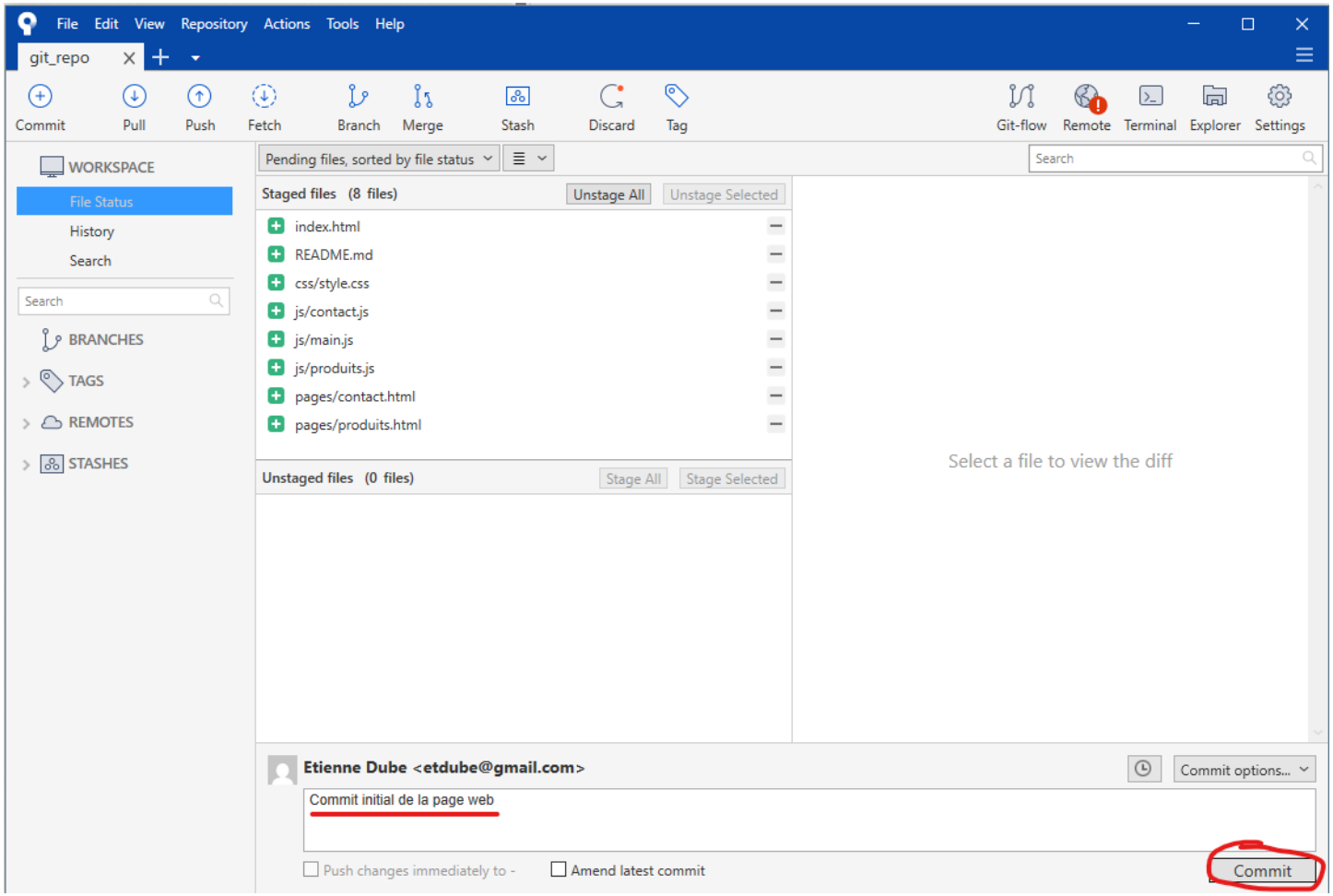
Les commits dans Git (suite)

- Commande : git commit

```
D:\formations\git_repo>git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md
        new file:   css/style.css
        new file:   index.html
        new file:   js/contact.js
        new file:   js/main.js
        new file:   js/produits.js
        new file:   pages/contact.html
        new file:   pages/produits.html


D:\formations\git_repo>git commit
[main (root-commit) ef54344] Commit initial de la page web
 8 files changed, 555 insertions(+)
 create mode 100644 README.md
 create mode 100644 css/style.css
 create mode 100644 index.html
 create mode 100644 js/contact.js
 create mode 100644 js/main.js
 create mode 100644 js/produits.js
 create mode 100644 pages/contact.html
 create mode 100644 pages/produits.html

D:\formations\git_repo>git status
On branch main
nothing to commit, working tree clean
```

Les commits dans Git (suite)

- Avec Sourcetree



Les commits dans Git (suite)

- Lorsqu'on fait des modifications dans le répertoire de travail, on doit faire un *commit* afin de les persister dans le dépôt git.
- Exemple : modification et ajout de fichiers.

```
D:\formations\git_repo>git status
On branch main
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:   index.html
        modified:   pages/contact.html
        modified:   pages/produits.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        css/nouveaustyle.css
        pages/soutien_technique.html

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

D:\formations\git_repo>git add .

D:\formations\git_repo>git commit
[main 2b10e9a] Nouvelle page Soutien technique
 5 files changed, 46 insertions(+)
 create mode 100644 css/nouveaustyle.css
 create mode 100644 pages/soutien_technique.html
```

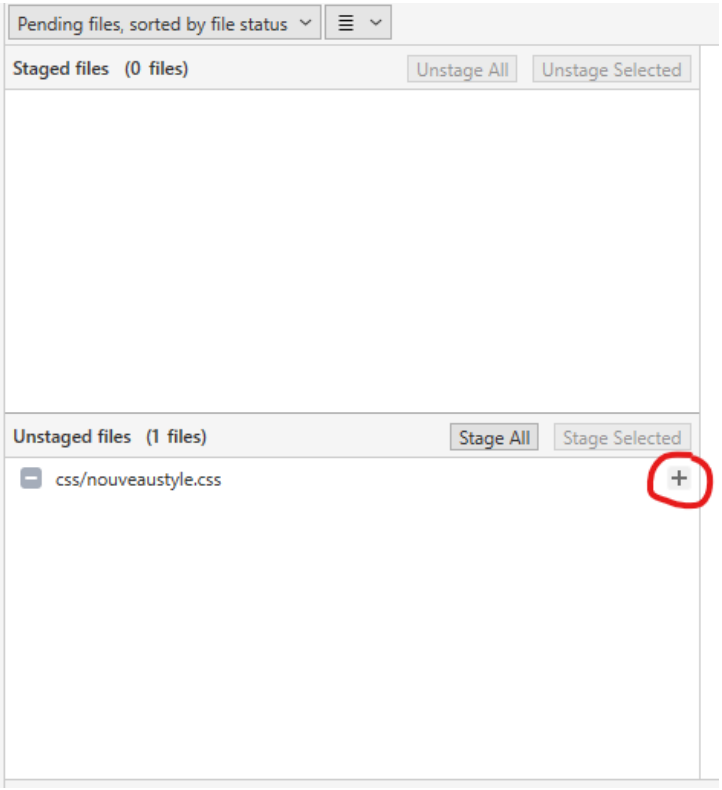
Les commits dans Git (suite)

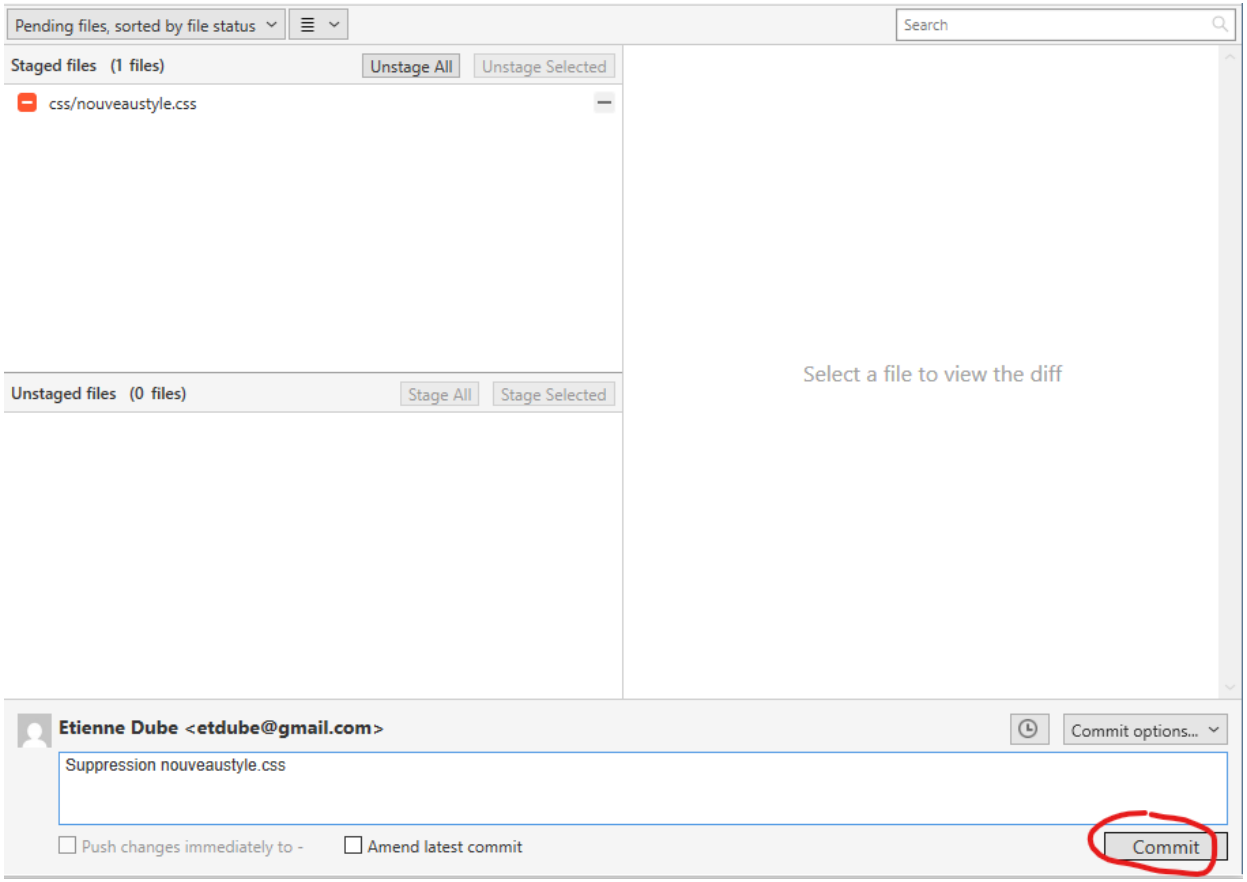
- **Pour supprimer un fichier du dépôt, on doit « ajouter la suppression » dans un commit.**
- **Commande : git rm <fichier>**
  - Supprime le fichier du répertoire local et « ajoute sa suppression » dans l'index

```
D:\formations\git_repo>git rm css\nouveaustyle.css
rm 'css/nouveaustyle.css'

D:\formations\git_repo>git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    css/nouveaustyle.css


D:\formations\git_repo>git commit
[main f586d9a] Supprimé nouveaustyle.css
 1 file changed, 5 deletions(-)
 delete mode 100644 css/nouveaustyle.css
```

Les commits dans Git (suite)

- Suppression avec Sourcetree





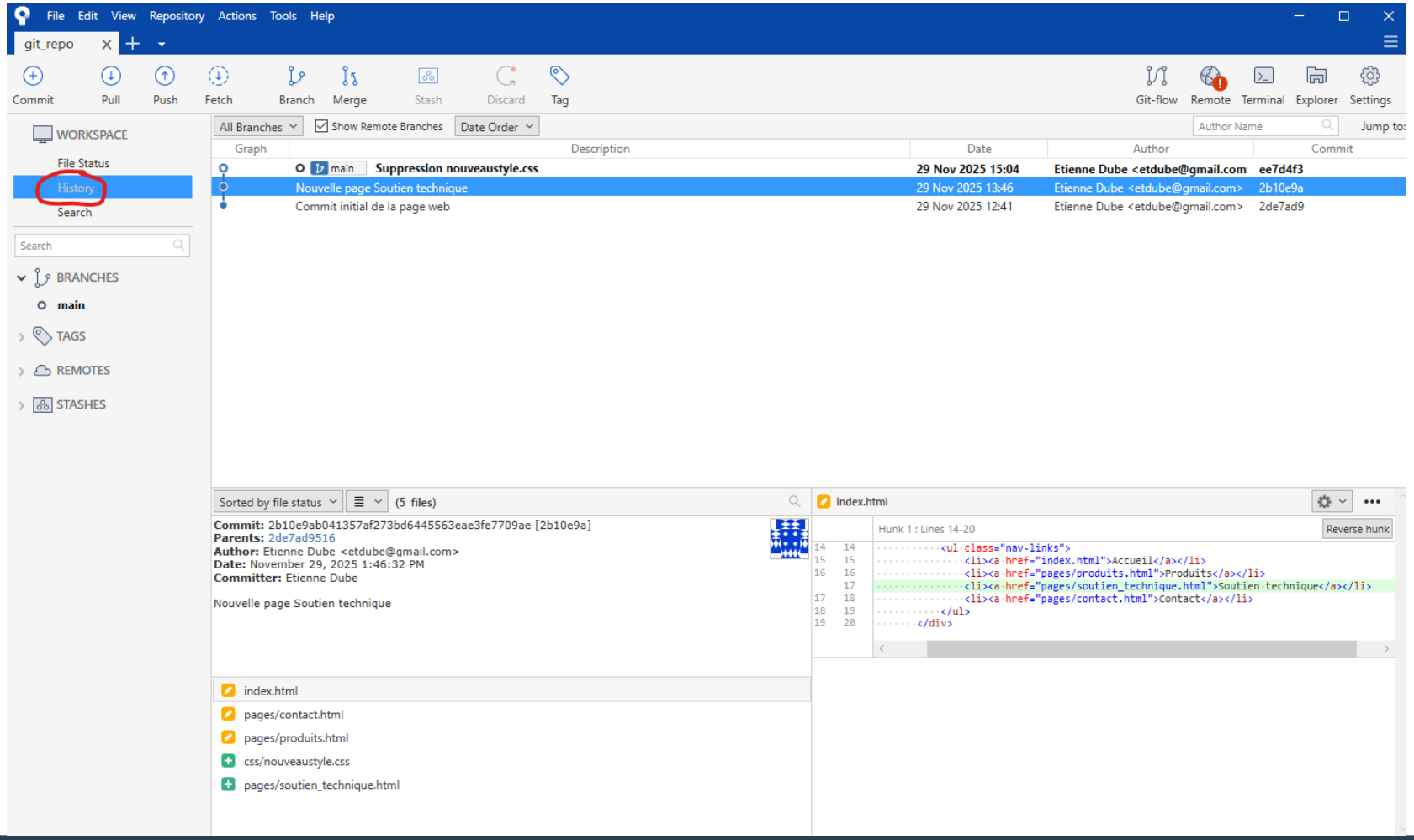
Voir historique des commits

- Commande : git log

```
D:\formations\git_repo>git log
commit ee7d4f35c0b58a2756a5372d425ef7860dae6b61 (HEAD -> main)
Author: Etienne Dube <etdube@gmail.com>
Date:   Sat Nov 29 15:04:30 2025 -0500

    Suppression nouveaustyle.css

commit 2b10e9ab041357af273bd6445563eae3fe7709ae
Author: Etienne Dube <etdube@gmail.com>
Date:   Sat Nov 29 13:46:32 2025 -0500

    Nouvelle page Soutien technique

commit 2de7ad9516af646d623edc46d717740c6cfbbe21
Author: Etienne Dube <etdube@gmail.com>
Date:   Sat Nov 29 12:41:46 2025 -0500

    Commit initial de la page web
```

Voir l’historique des commits (suite)

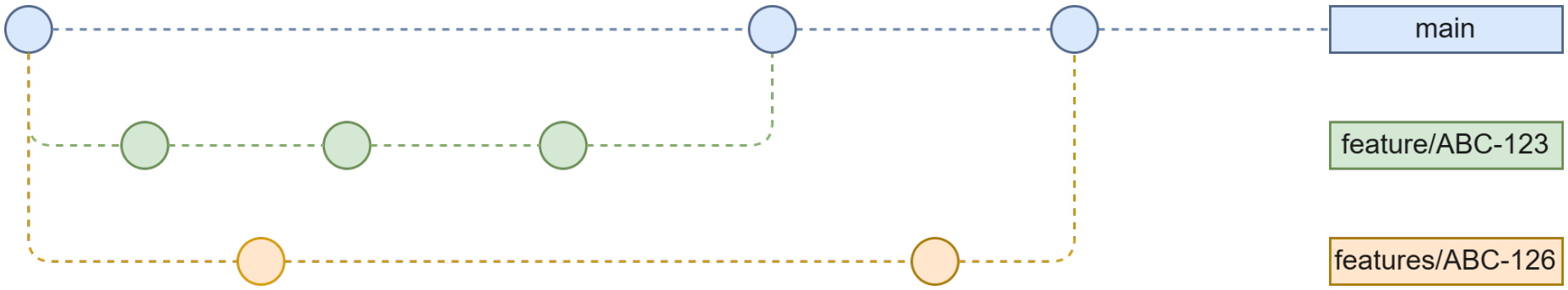
- Dans Sourcetree :



Les branches

- **Une branche permet de faire du développement en parallèle, sur des fonctionnalités différentes.**
  - Crée un « chemin » différent pour les commits.
  - Permet à deux personnes (ou plus) de travailler sur des fonctionnalités différentes.
  - On réintègre ou fusionne (*merge*) une branche de travail vers la branche principale quand le travail sur la fonctionnalité est terminé.

Les branches (suite)



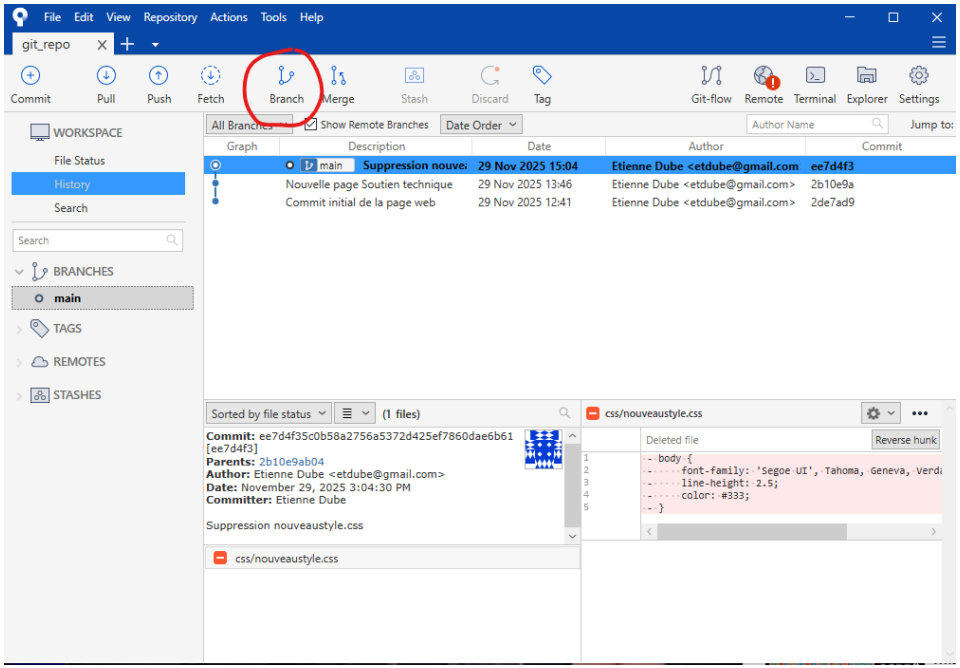
Créer une branche, changer de branche

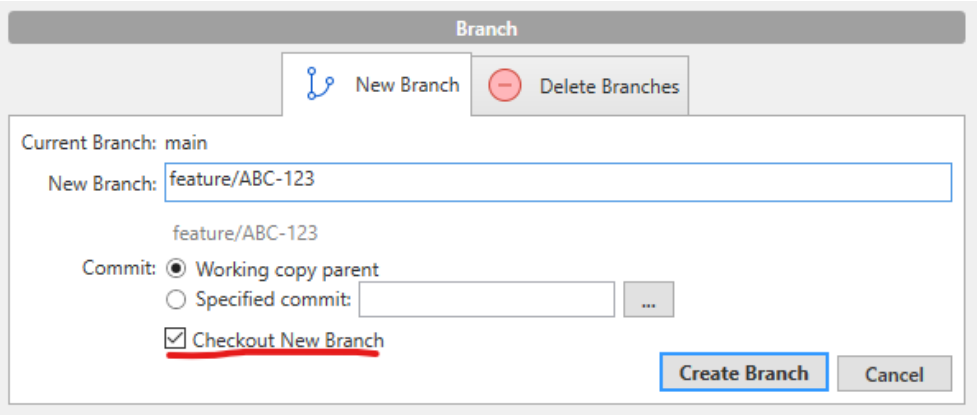
- Créer une branche : git branch <nom>
- Changer de branche : git checkout <nom>

```
D:\formations\git_repo>git branch feature/ABC-123

D:\formations\git_repo>git status
On branch main
nothing to commit, working tree clean

D:\formations\git_repo>git checkout feature/ABC-123
Switched to branch 'feature/ABC-123'

D:\formations\git_repo>git status
On branch feature/ABC-123
nothing to commit, working tree clean
```

Créer une branche, changer de branche (suite)

- Avec Sourcetree :





Travailler dans une branche

- Faisons quelques modifications et *commits* dans la nouvelle branche feature/ABC-123...

Graph	Description	Date	Author	Commit
	feature/ABC-123 Changement libellés	**30 Nov 2025 12:49**	**Etienne Dube <etdube@gmail.com**	**e00e571**
	Ajout produit disque dur externe	30 Nov 2025 12:44	Etienne Dube <etdube@gmail.com>	24f700c
	main Suppression nouveaustyle.css	29 Nov 2025 15:04	Etienne Dube <etdube@gmail.com>	ee7d4f3
	Nouvelle page Soutien technique	29 Nov 2025 13:46	Etienne Dube <etdube@gmail.com>	2b10e9a
	Commit initial de la page web	29 Nov 2025 12:41	Etienne Dube <etdube@gmail.com>	2de7ad9

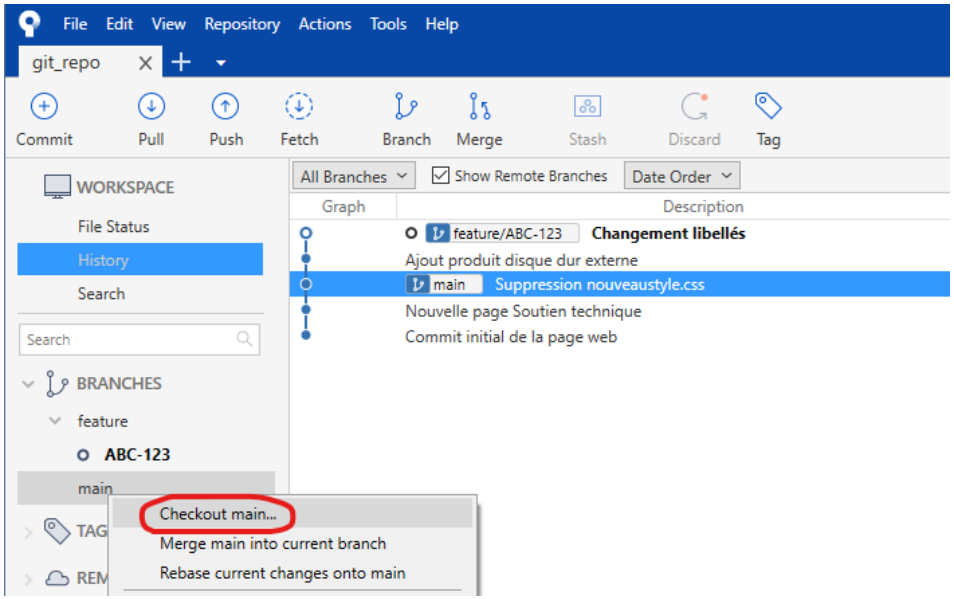
Fusionner les changements d'une branche

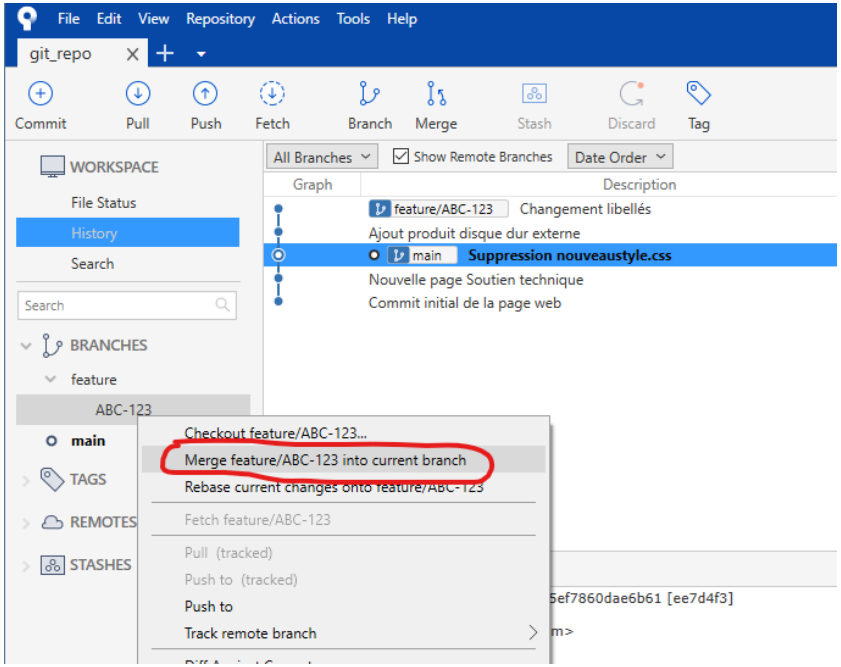
- **Commande *merge***
  - On doit être dans la branche de destination pour ce faire.
  - *git merge <nom branche à intégrer>*
  - Bonne idée de supprimer la branche fusionnée par après : git branch -d <branche>

```
D:\formations\git_repo>git status
On branch feature/ABC-123
nothing to commit, working tree clean

D:\formations\git_repo>git checkout main
Switched to branch 'main'

D:\formations\git_repo>git merge feature/ABC-123
Updating ee7d4f3..e00e571
Fast-forward
 index.html          | 2 +-
 js/main.js          | 6 ++++++
 pages/contact.html | 2 +-
 3 files changed, 8 insertions(+), 2 deletions(-)

D:\formations\git_repo>git branch -d feature/ABC-123
Deleted branch feature/ABC-123 (was e00e571).
```

Fusionner les changements d'une branche (suite)

- Avec Sourcetree :

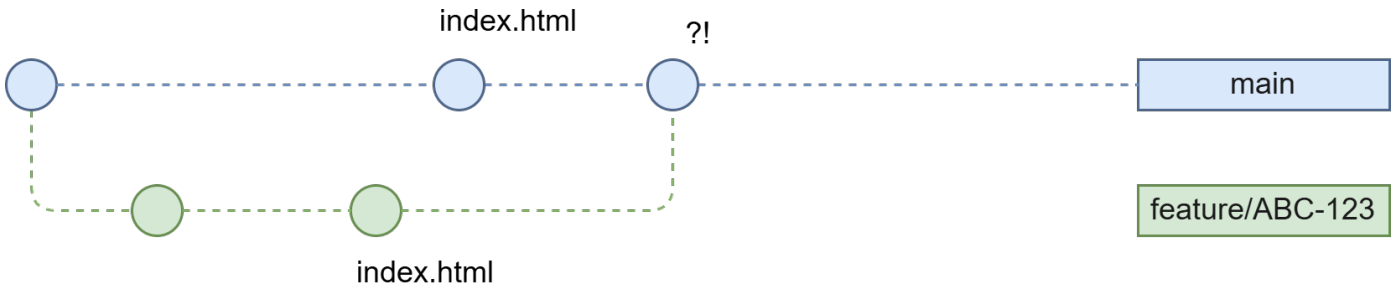




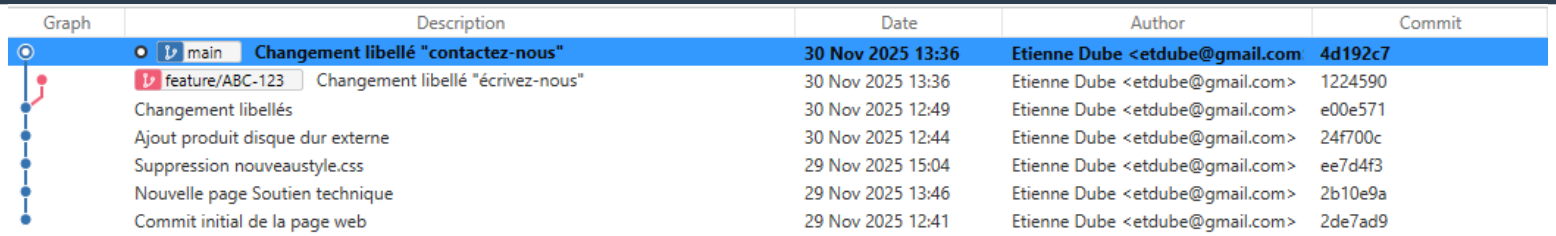
Graph	Description	Date	Author	Commit
	main feature/ABC-123 **Changement libellés**	**30 Nov 2025 12:49**	**Etienne Dube <etdube@gmail.com**	**e00e571**
	Ajout produit disque dur externe	30 Nov 2025 12:44	Etienne Dube <etdube@gmail.com>	24f700c
	Suppression nouveaustyle.css	29 Nov 2025 15:04	Etienne Dube <etdube@gmail.com>	ee7d4f3
	Nouvelle page Soutien technique	29 Nov 2025 13:46	Etienne Dube <etdube@gmail.com>	2b10e9a
	Commit initial de la page web	29 Nov 2025 12:41	Etienne Dube <etdube@gmail.com>	2de7ad9

Gestion des conflits

- Qu’arrive-t-il à la fusion lorsque deux branches contiennent des changements pour un même fichier ?



Gestion des conflits (suite)



Graph	Description	Date	Author	Commit
	main **Changement libellé "contactez-nous"**	**30 Nov 2025 13:36**	**Etienne Dube <etdube@gmail.com>**	**4d192c7**
	feature/ABC-123 Changement libellé "écrivez-nous"	30 Nov 2025 13:36	Etienne Dube <etdube@gmail.com>	1224590
	Changement libellés	30 Nov 2025 12:49	Etienne Dube <etdube@gmail.com>	e00e571
	Ajout produit disque dur externe	30 Nov 2025 12:44	Etienne Dube <etdube@gmail.com>	24f700c
	Suppression nouveaustyle.css	29 Nov 2025 15:04	Etienne Dube <etdube@gmail.com>	ee7d4f3
	Nouvelle page Soutien technique	29 Nov 2025 13:46	Etienne Dube <etdube@gmail.com>	2b10e9a
	Commit initial de la page web	29 Nov 2025 12:41	Etienne Dube <etdube@gmail.com>	2de7ad9

```
D:\formations\git_repo>git status
On branch main
nothing to commit, working tree clean

D:\formations\git_repo>git merge feature/ABC-123
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.

D:\formations\git_repo>git status
On branch main
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   index.html

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

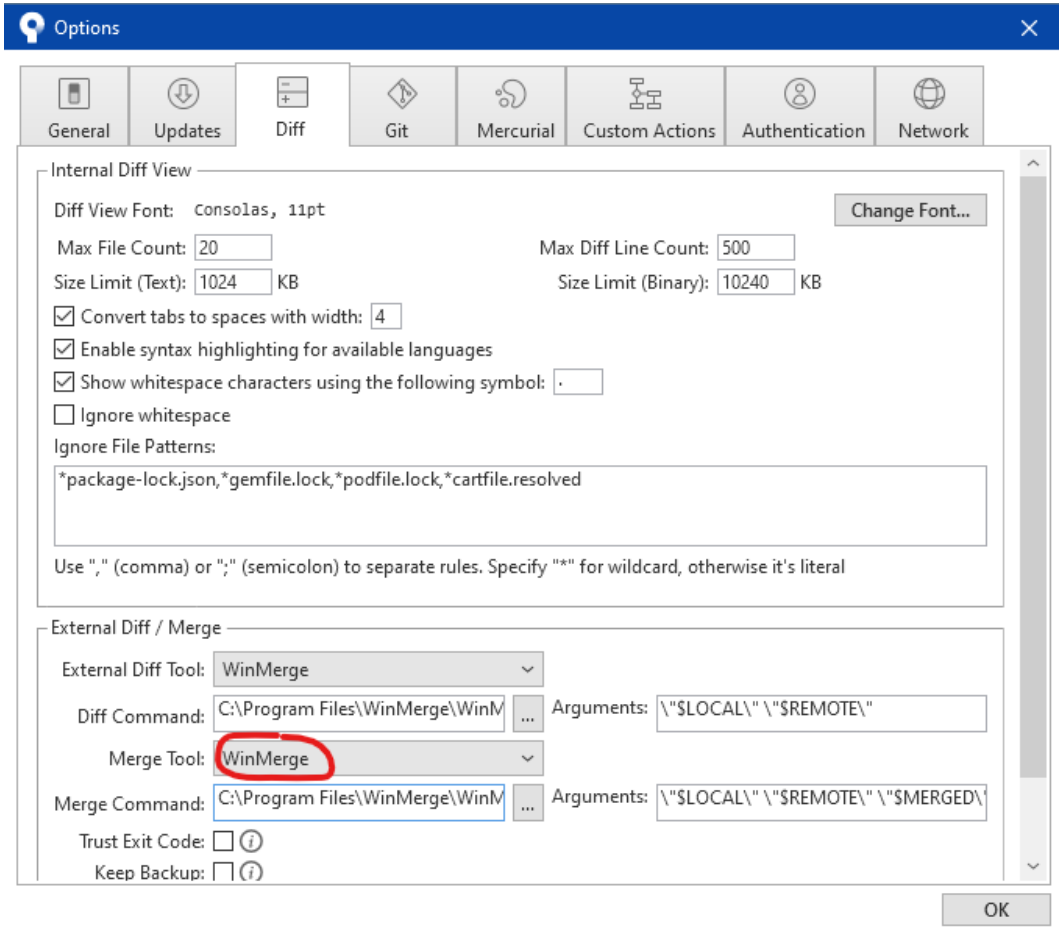
```
17                  <li><a href="pages/soutien_technique.html">Soutien technique</a></li>
18  <<<<<<< HEAD
19                  <li><a href="pages/contact.html">Contactez-nous</a></li>
20  =======
21                  <li><a href="pages/contact.html">Écrivez-nous</a></li>
22  >>>>>>> feature/ABC-123
23              </ul>
24          </div>
```

Gestion des conflits (suite)

```
D:\formations\git_repo>git status
On branch main
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   index.html

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

D:\formations\git_repo>git add index.html

D:\formations\git_repo>git status
On branch main
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:
        modified:   index.html


D:\formations\git_repo>git commit
[main dd990b1] Merge branch 'feature/ABC-123'
```

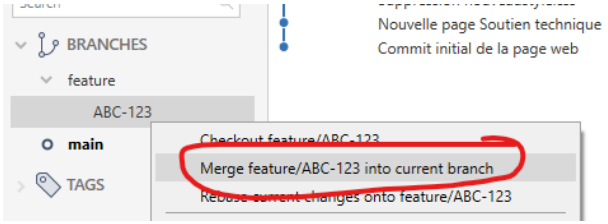
Gestion des conflits (suite)

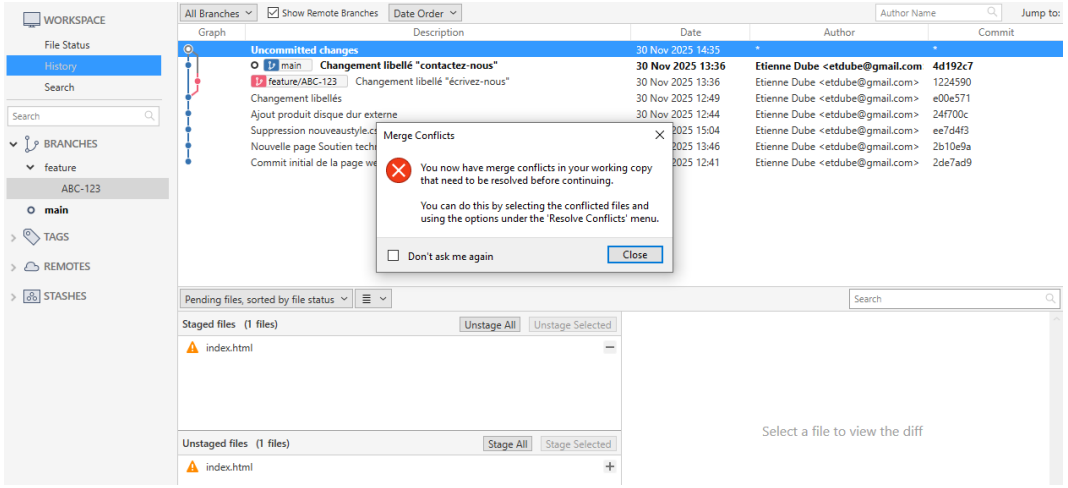
- Avec Sourcetree : on peut configurer un éditeur de conflits, p.ex. WinMerge



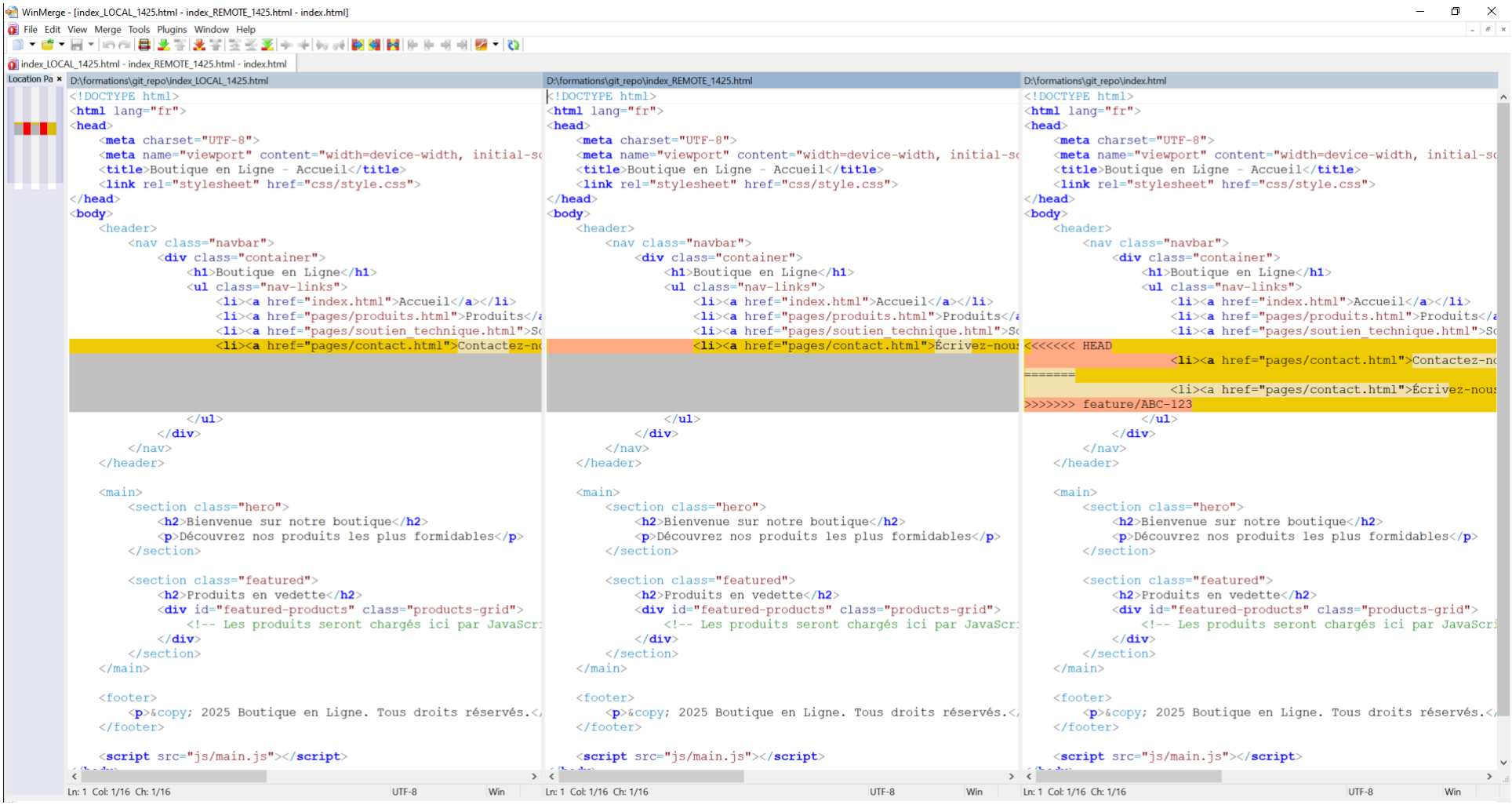
Gestion des conflits (suite)

- Résolution de conflit avec Sourcetree et WinMerge :

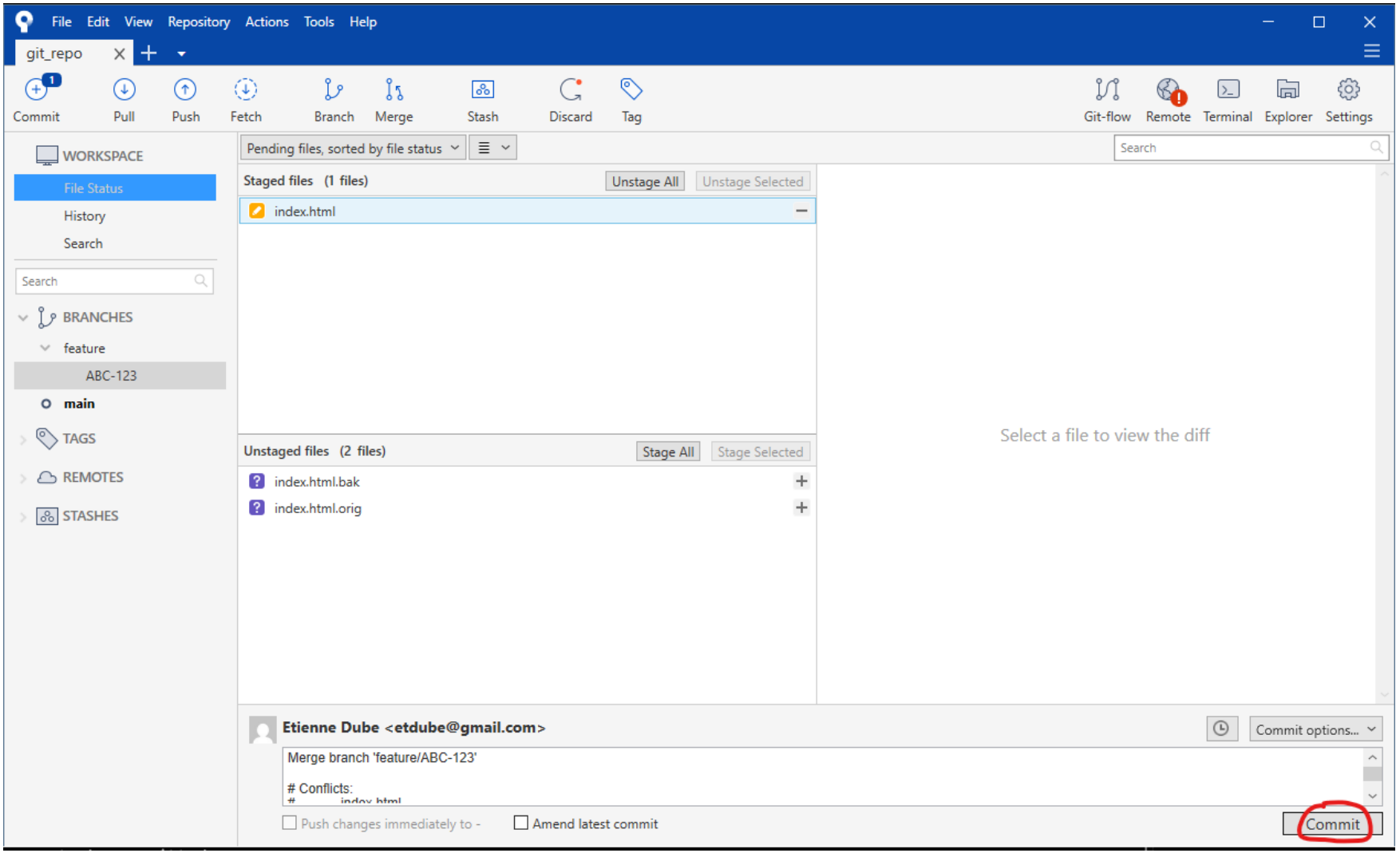




Gestion des conflits (suite)



Gestion des conflits (suite)



Gestion des conflits (suite)

- Résultat de la fusion :

Graph	Description	Date	Author	Commit
	main **Merge branch 'feature/ABC-123'**	**30 Nov 2025 14:44**	**Etienne Dube <etdube@gmail.com**	**edf0454**
	Changement libellé "contactez-nous"	30 Nov 2025 13:36	Etienne Dube <etdube@gmail.com>	4d192c7
	feature/ABC-123 Changement libellé "écrivez-nous"	30 Nov 2025 13:36	Etienne Dube <etdube@gmail.com>	1224590
	Changement libellés	30 Nov 2025 12:49	Etienne Dube <etdube@gmail.com>	e00e571
	Ajout produit disque dur externe	30 Nov 2025 12:44	Etienne Dube <etdube@gmail.com>	24f700c
	Suppression nouveaustyle.css	29 Nov 2025 15:04	Etienne Dube <etdube@gmail.com>	ee7d4f3
	Nouvelle page Soutien technique	29 Nov 2025 13:46	Etienne Dube <etdube@gmail.com>	2b10e9a
	Commit initial de la page web	29 Nov 2025 12:41	Etienne Dube <etdube@gmail.com>	2de7ad9

Gestion des conflits (suite)

- **Un commit de fusion (*merge commit*) est créé.**
  - N'arrive pas que lorsqu'il y a des conflits, mais dans toute situation où les deux branches ont chacune des commits différents.
  - Autrement, le merge peut être *fast-forward* (sans commit de fusion).
- **Bonne pratique : fusionner la branche commune (p.ex. main ou develop) vers la branche de travail et résoudre les conflits *avant* de réintégrer celle-ci.**
  - On peut faire ceci régulièrement afin d'éviter que la branche de travail diverge trop par rapport à la branche commune.

Git : dépôts distants

Les dépôts distants

- Jusqu'à maintenant, toutes les commandes Git montrées s'effectuaient en local.
- Pour collaborer avec d'autres, on peut utiliser un dépôt distant (*remote* dans la terminologie Git).
- Typiquement, les dépôts distants sont hébergés par des services en ligne, p.ex. : GitHub, GitLab, Bitbucket, etc.

Les dépôts distants (suite)

- Voir les dépôts distants configurés : *git remote -v*
- Ajouter un dépôt distant :
  git remote add origin <url>
  (note : le dépôt doit exister sur le serveur à l'URL indiqué)

```
D:\formations\git_repo>git remote -v

D:\formations\git_repo>git remote add origin https://github.com/etdube/formation_git_exemples.git

D:\formations\git_repo>git remote -v
origin  https://github.com/etdube/formation_git_exemples.git (fetch)
origin  https://github.com/etdube/formation_git_exemples.git (push)
```

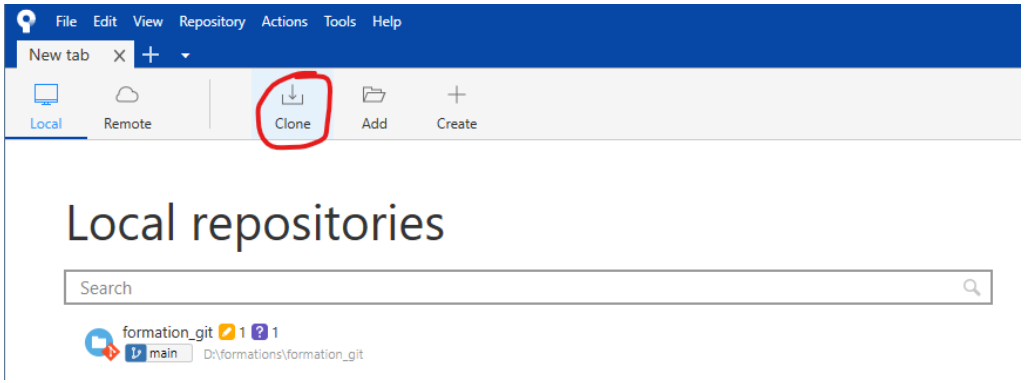
Cloner un dépôt distant

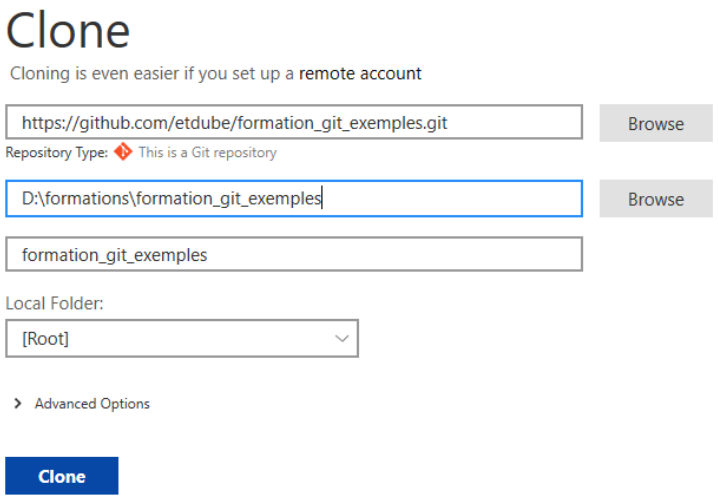
- Généralement, le dépôt distant existe déjà et on veut récupérer son contenu sur notre poste de travail.
- Pour ce faire, on doit utiliser la commande *clone* :
  - git clone <url>

```
D:\formations>git clone https://github.com/etdube/formation_git_exemples.git
Cloning into 'formation_git_exemples'...
remote: Enumerating objects: 40, done.
remote: Counting objects: 100% (40/40), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 40 (delta 18), reused 40 (delta 18), pack-reused 0 (from 0)
Receiving objects: 100% (40/40), 7.50 KiB | 7.50 MiB/s, done.
Resolving deltas: 100% (18/18), done.

D:\formations>cd formation_git_exemples

D:\formations\formation_git_exemples>git remote -v
origin  https://github.com/etdube/formation_git_exemples.git (fetch)
origin  https://github.com/etdube/formation_git_exemples.git (push)
```

Cloner un dépôt distant (suite)

- Avec Sourcetree :





Les branches de suivi à distance

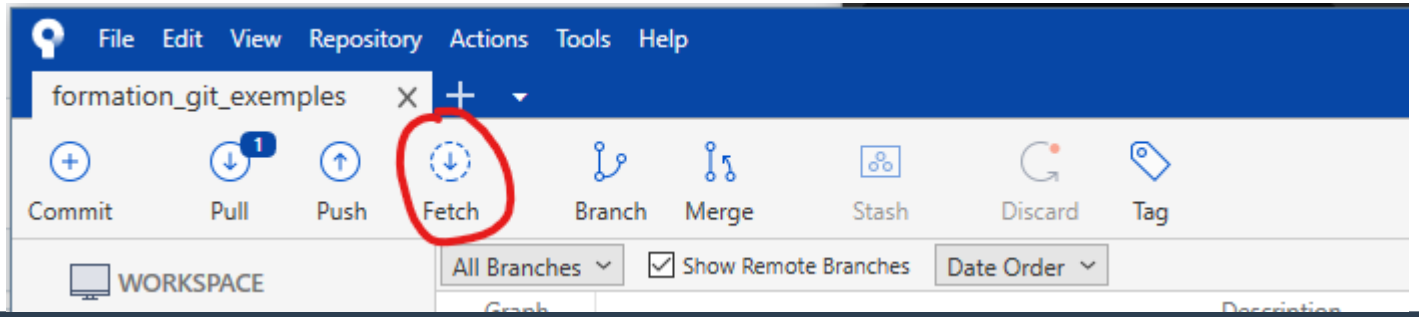
- **Les branches de suivi à distance (*remote tracking branches*) sont des références vers les branches du dépôt distant configuré.**
  - p.ex. origin/main → fait référence à la branche *main* du dépôt distant *origin*.
- **On ne fait pas de commit directement dans ces branches, on utilise plutôt leur copie locale pour ce faire.**

Les branches de suivi à distance (suite)

- Pour synchroniser les branches de suivi à distance depuis le dépôt distant, on utilise la commande *git fetch*

```
D:\formations\formation_git_exemples>git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 1.13 KiB | 231.00 KiB/s, done.
From https://github.com/etdube/formation_git_exemples
   edf0454..3663dcc  main       -> origin/main

D:\formations\formation_git_exemples>git status
On branch main
Your branch is behind 'origin/main' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

nothing to commit, working tree clean
```

- Avec Sourcetree :



Les branches de suivi à distance (suite)

- Que signifie ceci ?

```
Your branch is behind 'origin/main' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)
```

- La branche locale *main* est en retard d'un commit sur la branche distante *origin/main*.
- La commande *git pull* permet de mettre à jour la branche locale p.r. à la branche distante.
- pull = fetch + merge

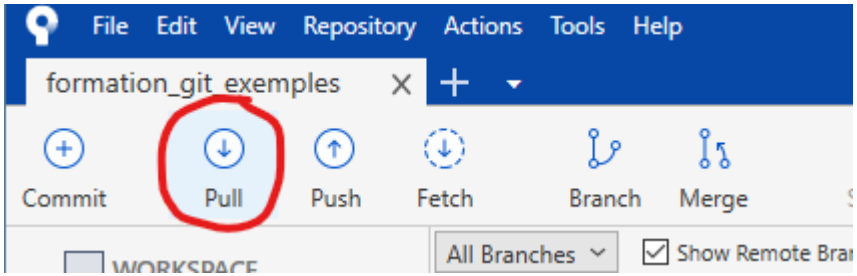
Les branches de suivi à distance (suite)

```
D:\formations\formation_git_exemples>git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 1.13 KiB | 231.00 KiB/s, done.
From https://github.com/etdube/formation_git_exemples
   edf0454..3663dcc  main       -> origin/main

D:\formations\formation_git_exemples>git status
On branch main
Your branch is behind 'origin/main' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

nothing to commit, working tree clean

D:\formations\formation_git_exemples>git merge origin/main
Updating edf0454..3663dcc
Fast-forward
 README.md | 15 +++++++-------
 1 file changed, 8 insertions(+), 7 deletions(-)
```


```
D:\formations\formation_git_exemples>git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 1.16 KiB | 595.00 KiB/s, done.
From https://github.com/etdube/formation_git_exemples
   edf0454..e0a4b20  main       -> origin/main
Updating edf0454..e0a4b20
Fast-forward
 README.md | 15 +++++++-------
 1 file changed, 8 insertions(+), 7 deletions(-)
```

Les branches de suivi à distance (suite)

- Avec Sourcetree :



- On peut voir que *main* et *origin/main* sont à la même place :

Graph	Description	Date	Author	Commit
	main origin/main **Mettre à jour structure fichiers dans README.md**	**30 Nov 2025 18:1**	**etdube <etdube@**	**e0a4b20**
	Merge branch 'feature/ABC-123'	30 Nov 2025 14:44	Etienne Dube <etc	edf0454
	Changement libellé "contactez-nous"	30 Nov 2025 13:36	Etienne Dube <etc	4d192c7
	Changement libellé "écrivez-nous"	30 Nov 2025 13:36	Etienne Dube <etc	1224590
	Changement libellés	30 Nov 2025 12:49	Etienne Dube <etc	e00e571
	Ajout produit disque dur externe	30 Nov 2025 12:44	Etienne Dube <etc	24f700c
	Suppression nouveaustyle.css	29 Nov 2025 15:04	Etienne Dube <etc	ee7d4f3
	Nouvelle page Soutien technique	29 Nov 2025 13:46	Etienne Dube <etc	2b10e9a
	Commit initial de la page web	29 Nov 2025 12:41	Etienne Dube <etc	2de7ad9

Les branches de suivi à distance (suite)

- **Lors d'un *pull*, il peut y avoir des conflits de fusion !**
  - Cela arrive si la branche distante contient des commits avec modifications sur des fichiers qui ont aussi été modifiés dans la branche locale.
- **Dans ce cas, on doit résoudre les conflits de la même manière que lors d'un *merge* (voir section précédente).**

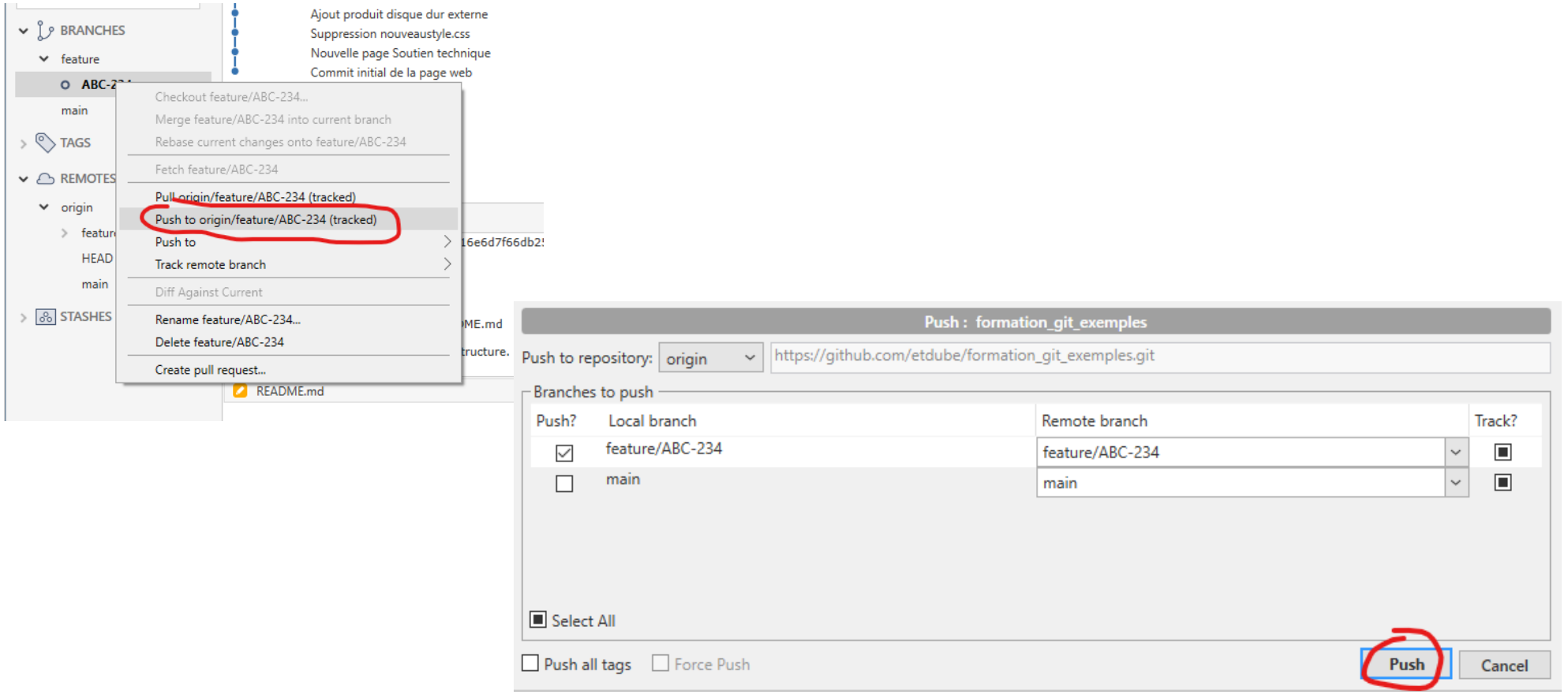
Publier une branche locale

- **Pour publier une branche locale vers le dépôt distant, on utilise la commande *push* :**
  - git push -u origin <branche>

```
D:\formations\formation_git_exemples>git branch feature/ABC-234

D:\formations\formation_git_exemples>git checkout feature/ABC-234
Switched to branch 'feature/ABC-234'

D:\formations\formation_git_exemples>git push -u origin feature/ABC-234
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature/ABC-234' on GitHub by visiting:
remote:      https://github.com/etdube/formation_git_exemples/pull/new/feature/ABC-234
remote:
To https://github.com/etdube/formation_git_exemples.git
 * [new branch]      feature/ABC-234 -> feature/ABC-234
branch 'feature/ABC-234' set up to track 'origin/feature/ABC-234'.

D:\formations\formation_git_exemples>git status
On branch feature/ABC-234
Your branch is up to date with 'origin/feature/ABC-234'.

nothing to commit, working tree clean
```

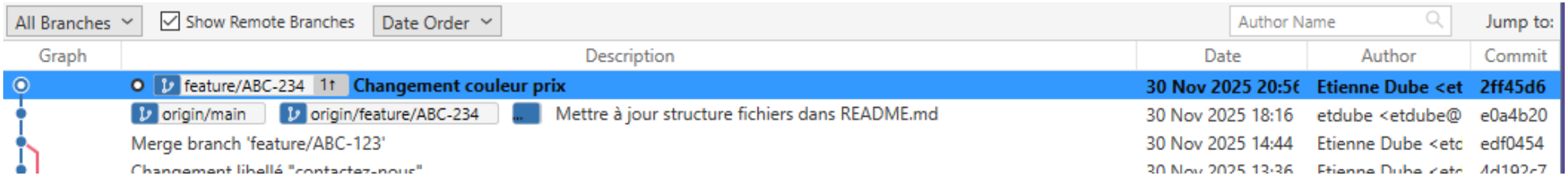
Publier une branche locale (suite)

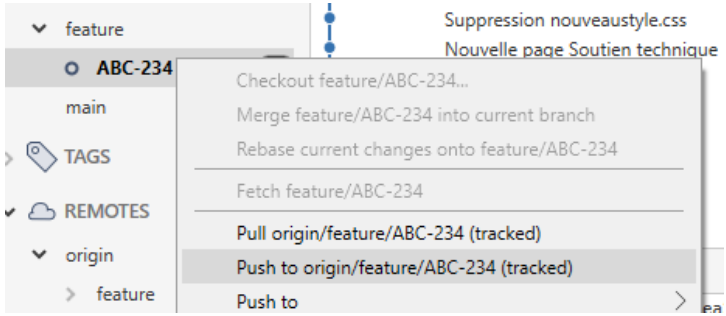
- Avec Sourcetree :

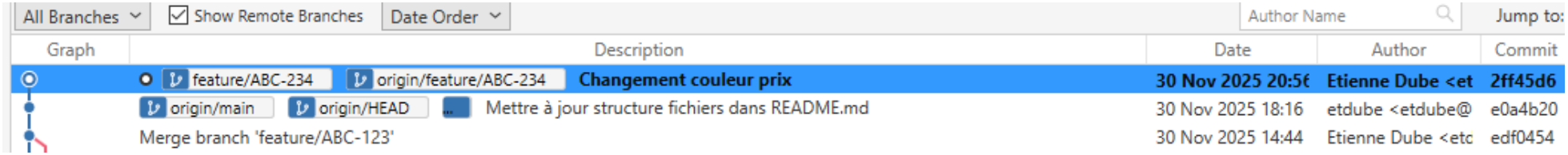


Publier une branche locale (suite)

- Après avoir fait des commits, on utilise également *push* pour mettre à jour la branche distante.

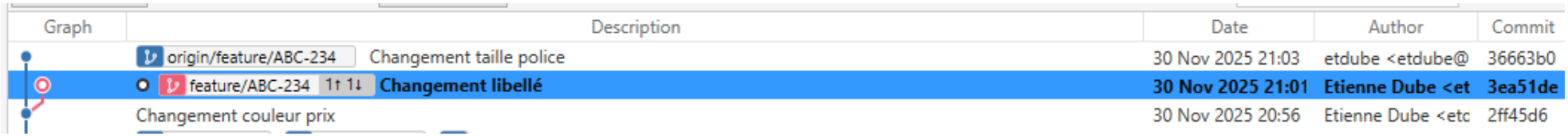


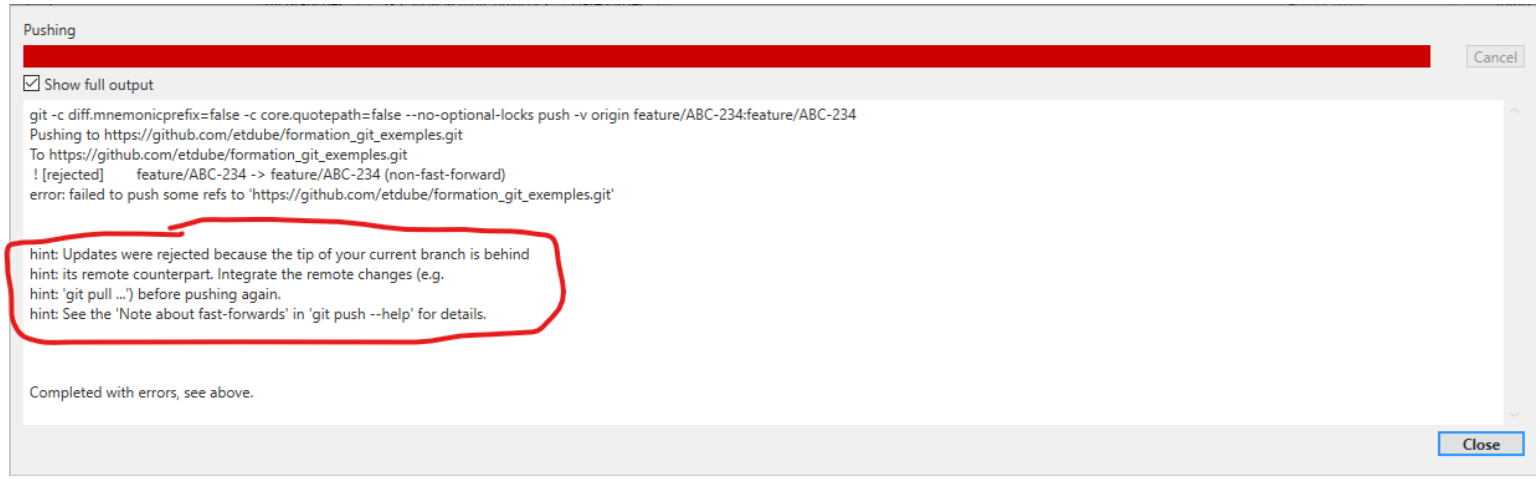




Publier une branche locale (suite)

- S’il y a des commits dans la branche distante qui ne sont pas encore dans la branche locale, il est impossible de faire un *push* :





- Solution : faire un *pull* avant de faire *push*.

Publier une branche locale (suite)

- **Il est possible de forcer le remplacement d'une branche distante avec *git push --force***
  - Attention : c'est une opération dangereuse et à éviter dans la plupart des cas !
  - Les commits distants qui ne sont pas dans la branche locale seront perdus.

Git : notions avancées

Fichiers .gitignore

- Les fichiers .gitignore permettent d'ignorer des fichiers ou des répertoires, afin qu'ils ne soient pas automatiquement ajoutés à l'index.
- On le met généralement à la racine du dépôt, mais il peut être ailleurs (s'applique à partir du répertoire dans lequel il est).
- Documentation : https://git-scm.com/docs/gitignore

Fichiers .gitignore (suite)

- Exemple

```
# Ignore all .log files
*.log

# Ignore a specific file
my_secret_key.pem

# Ignore a directory and its contents
temp_data/
build/

# Ignore all files with a specific extension in a directory
docs/*.txt

# Ignore node_modules directory, common in JavaScript projects
node_modules/

# Ignore Python bytecode files
__pycache__/
*.pyc

# Ignore environment variables files
.env

# Ignore IDE-specific files (e.g., VS Code)
.vscode/

# Negation: Ignore all .tmp files except important.tmp
*.tmp
!important.tmp

# Ignore all files in the 'output' directory at any depth
output/**

# Ignore common OS-generated files
.DS_Store
Thumbs.db
```

Autres opérations de branches

- *Cherry-pick* : copier un commit d'une branche à une autre.
- *Rebase* : met à jour une branche par rapport à une autre en rejouant l'historique des commits par rapport à l'autre branche.

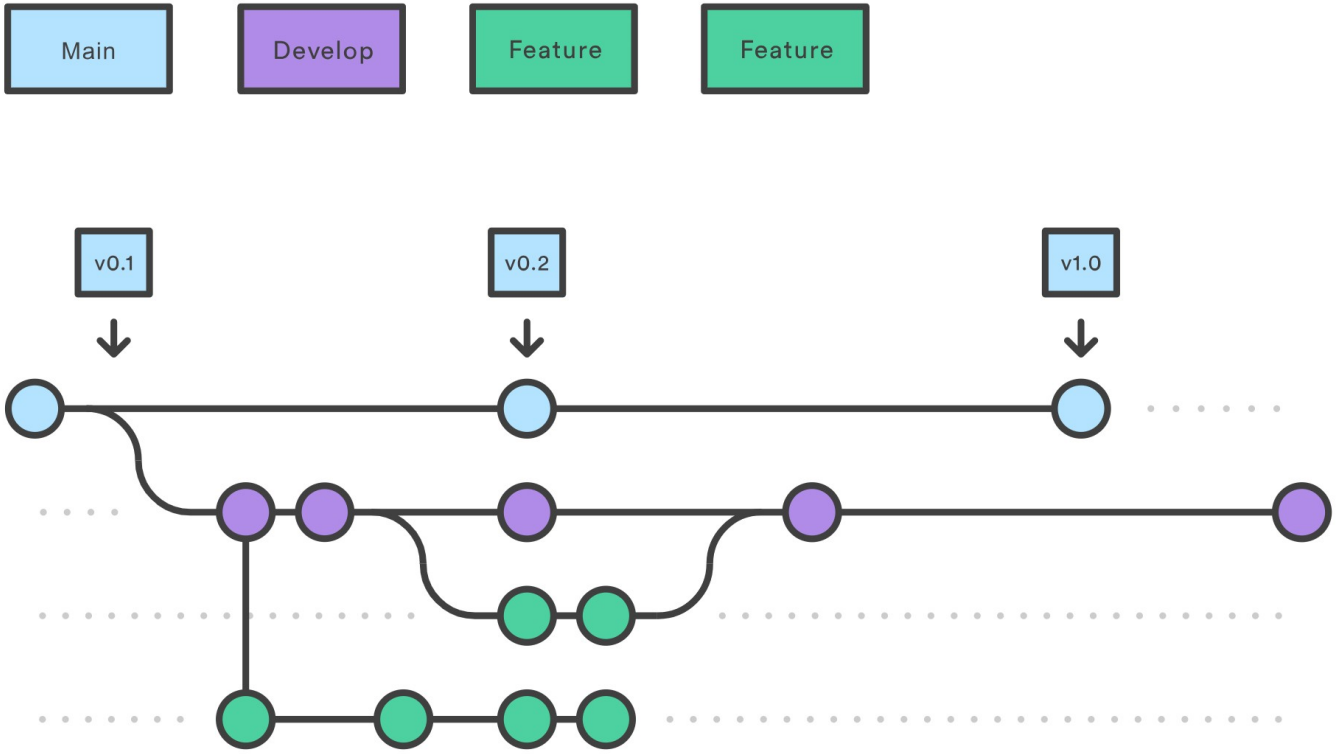
Les étiquettes

- Les étiquettes (*tags*) identifient un commit avec un nom facile à retenir.
- On s'en sert par exemple pour identifier la version d'une relâche déployée en production.
- Commande :
  - git tag <nom_étiquette>
- Une étiquette doit être poussée sur le dépôt distant pour être visible pour tout le monde :
  - git push origin <nom_étiquette>

```
D:\formations\formation_git_exemples>git tag v1.0.0

D:\formations\formation_git_exemples>git push origin v1.0.0
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/etdube/formation_git_exemples.git
 * [new tag]         v1.0.0 -> v1.0.0
```

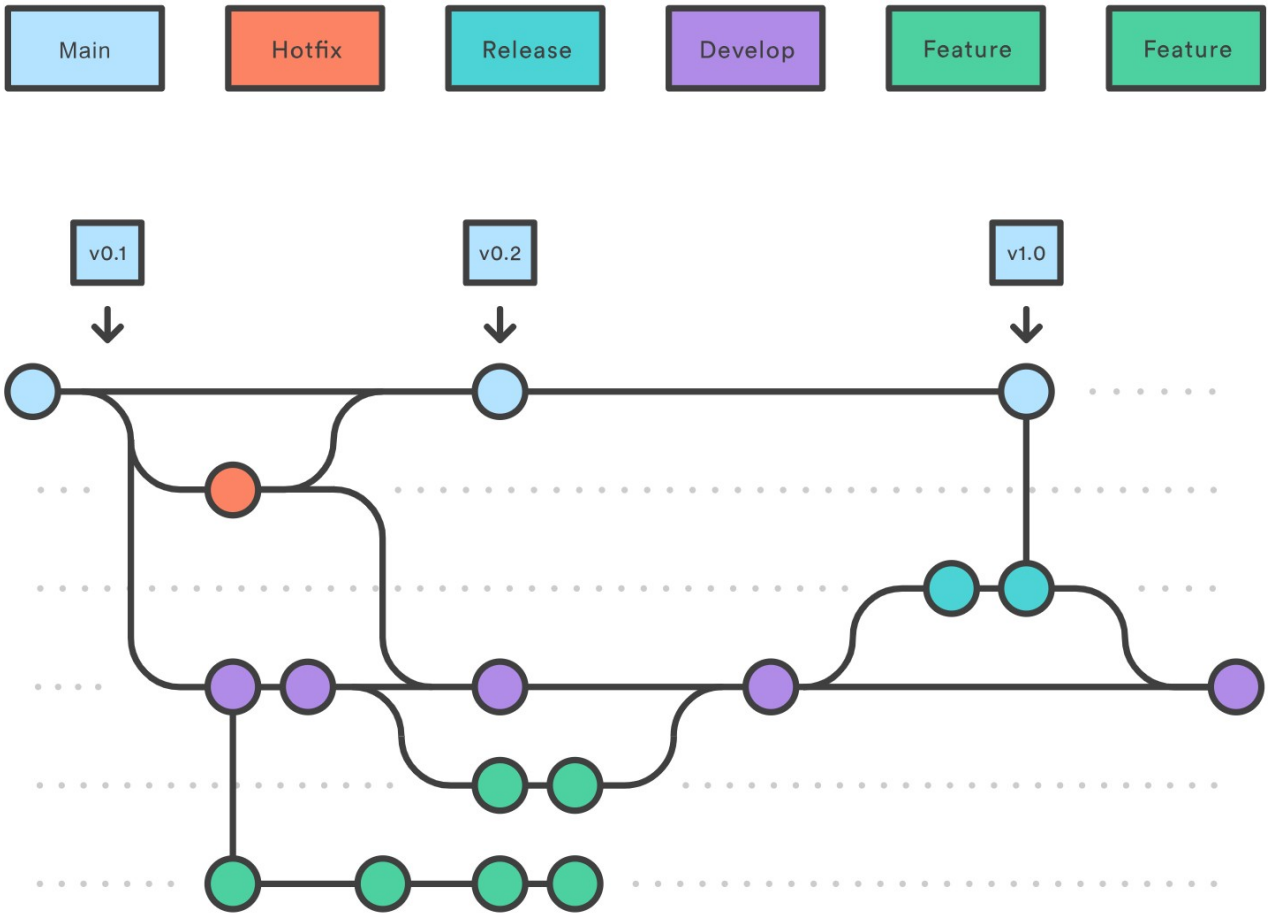
Stratégie de gestion des branches GitFlow

- **GitFlow propose une manière de gérer les branches d'un projet :**
  - main : version en production
  - develop : branche d'intégration de la prochaine version (en développement)
  - feature/… : branches de fonctionnalités, dérivées de develop et fusionnées vers celle-ci



Stratégie de gestion des branches GitFlow (suite)

- Branche « hotfix » pour correctifs en production
- Branche « release » lorsqu'on est en phase de stabilisation (pré-MEP)

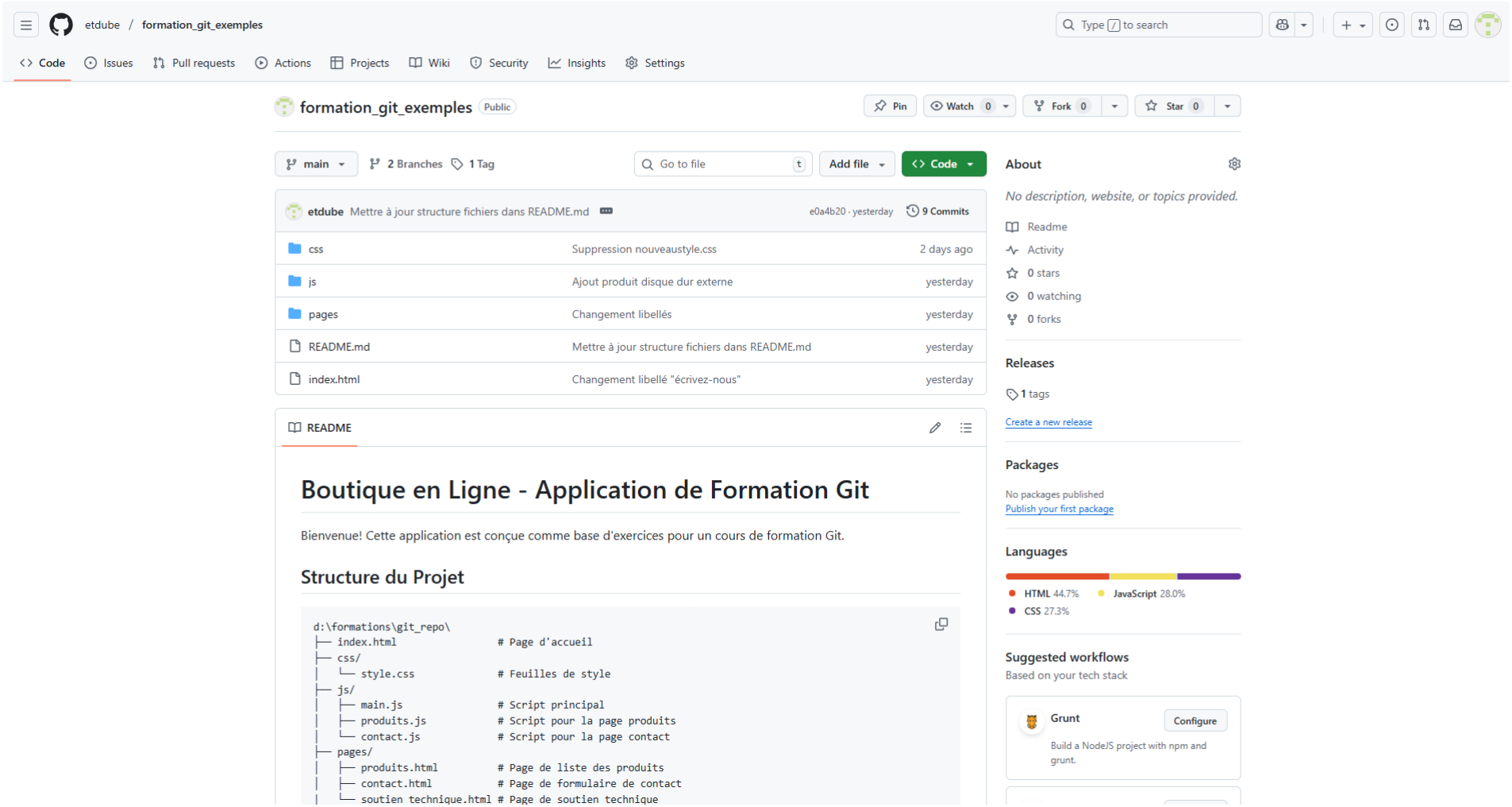


GitHub

Qu’est-ce que GitHub ?

- **GitHub est un service d’hébergement pour Git qui offre également des fonctionnalités de gestion de code.**
- **Offre gratuite (dépôts publics et privés)**
- **Offre pour entreprises : GitHub Enterprise**
  - Gestion d’accès (SSO, équipes, organisations)
  - Dépôts privés partagés dans l’organisation

Survol de l'interface utilisateur

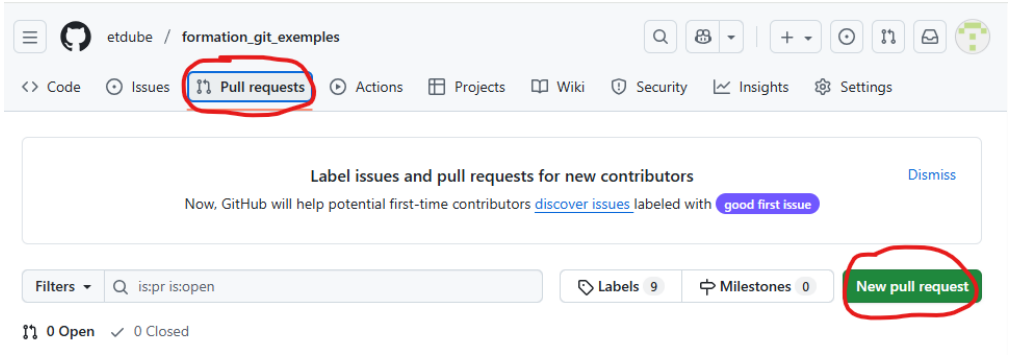


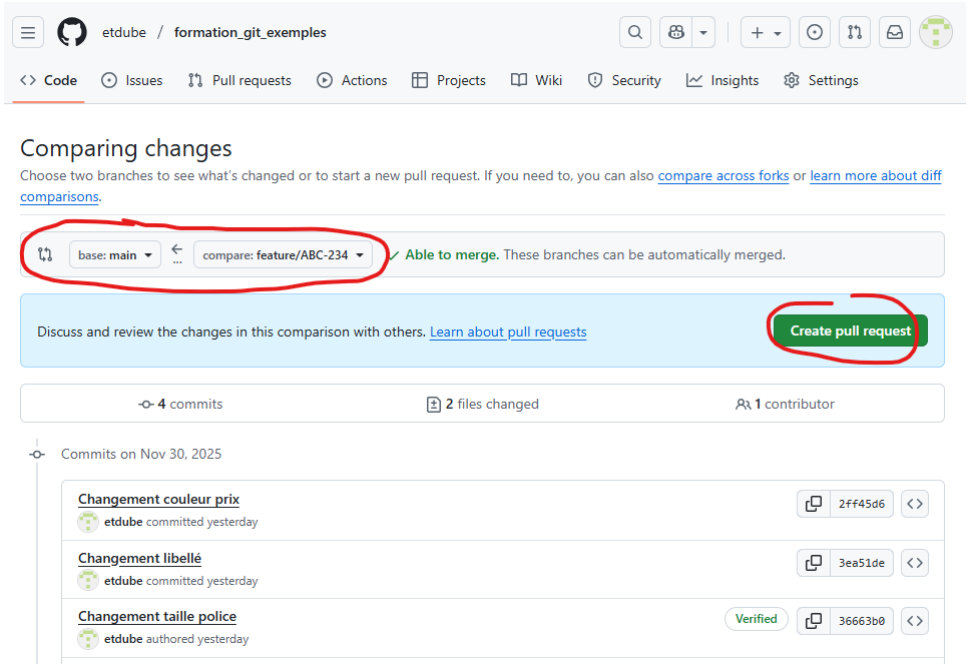
Les *Pull Request*

- Les *Pull Requests* (PR) dans GitHub permettent de solliciter une revue de code avant de fusionner une branche de travail vers une branche commune (p.ex. *main*).
- Les approbateurs peuvent commenter, demander des modifications et approuver la PR.
- Selon les réglages du dépôt, un certain nombre d'approbations peut être requis avant de fusionner une PR.
- Une fois tous les critères satisfaits, on peut fusionner la PR, ce qui revient à fusionner la branche.

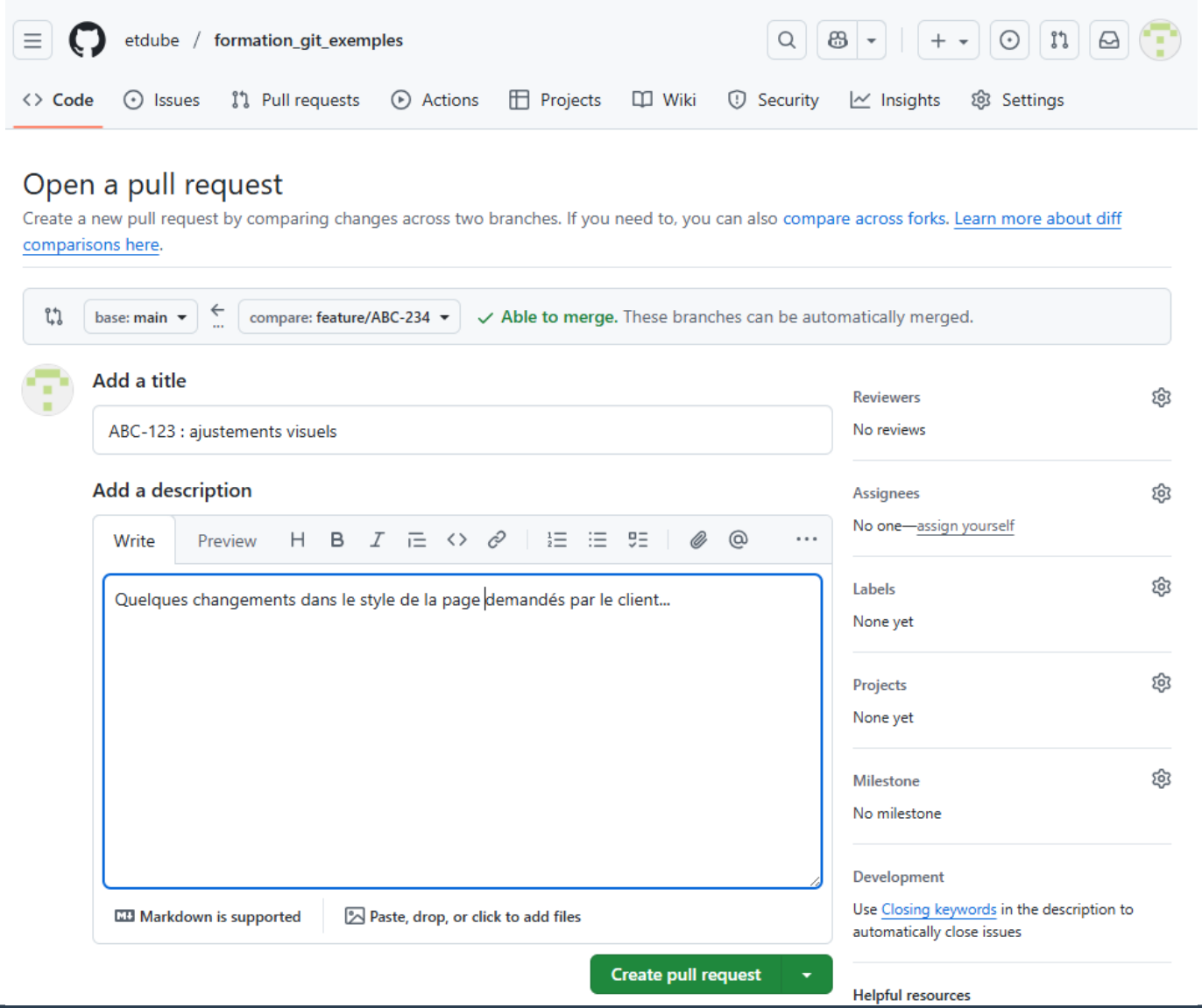
Exemple de PR

- Création à partir d'une branche



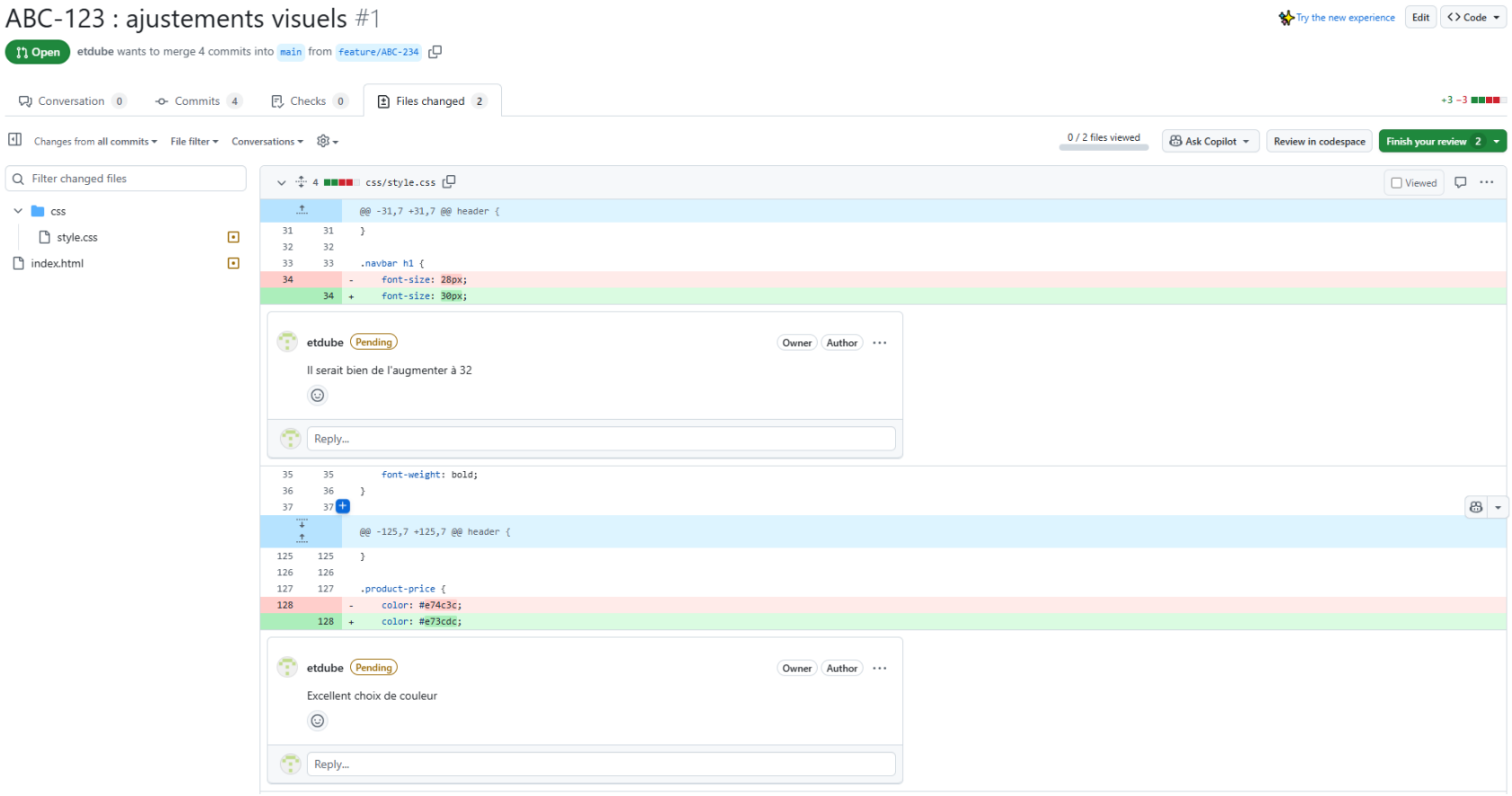


Exemple de PR (suite)



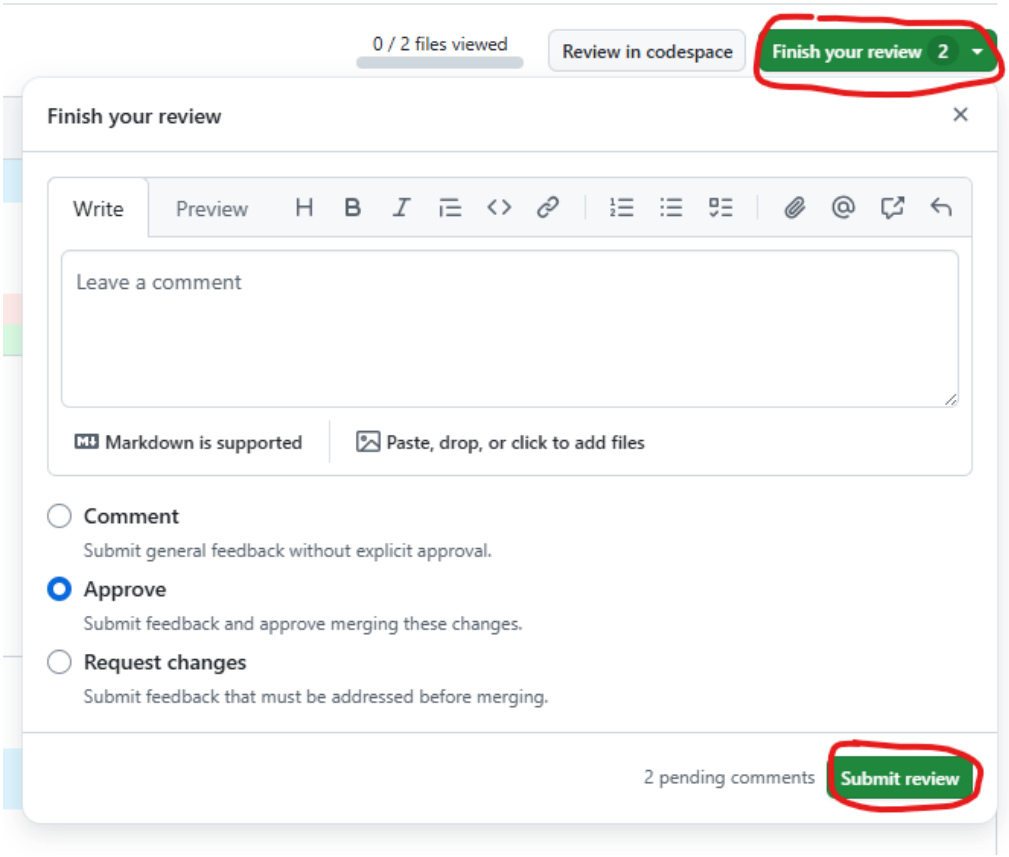
Exemple de PR (suite)

- Revue du code



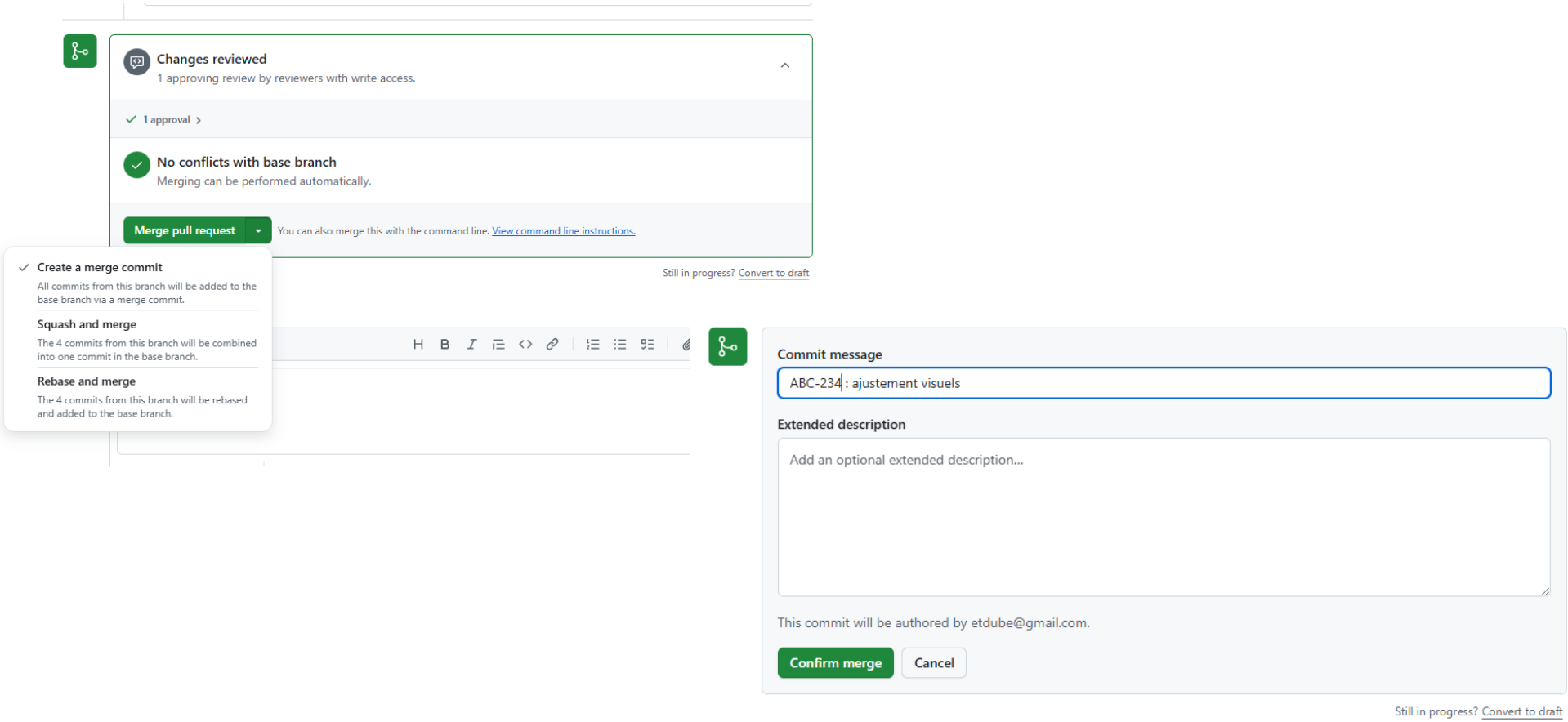
Exemple de PR (suite)

- Approbation



Exemple de PR (suite)

- Fusion de la PR



Options pour la fusion de PR

- « Create a merge commit » : conserve l'historique de tous les commits de la branche de travail et crée un commit de fusion dans la branche commune.
- « Squash and merge » : crée un seul commit dans la branche commune, en combinant tous les changements de la branche de travail (l'historique de celle-ci n'est pas conservé).
- « Rebase and merge » : effectue une opération *rebase* sur la branche de travail, à partir de la branche commune. Cela a pour effet de préserver l'historique de la branche, mais de manière linéaire par rapport à la branche commune.

Fin.

Questions ?