GIT ET LA GESTION DE VERSIONS 1/3

Auteur : Sébastien Inion

Gestion de versions centralisée et décentralisée

- Gestion de versions décentralisée
  - Git
  - Mercurial
  - Bazaar

GIT

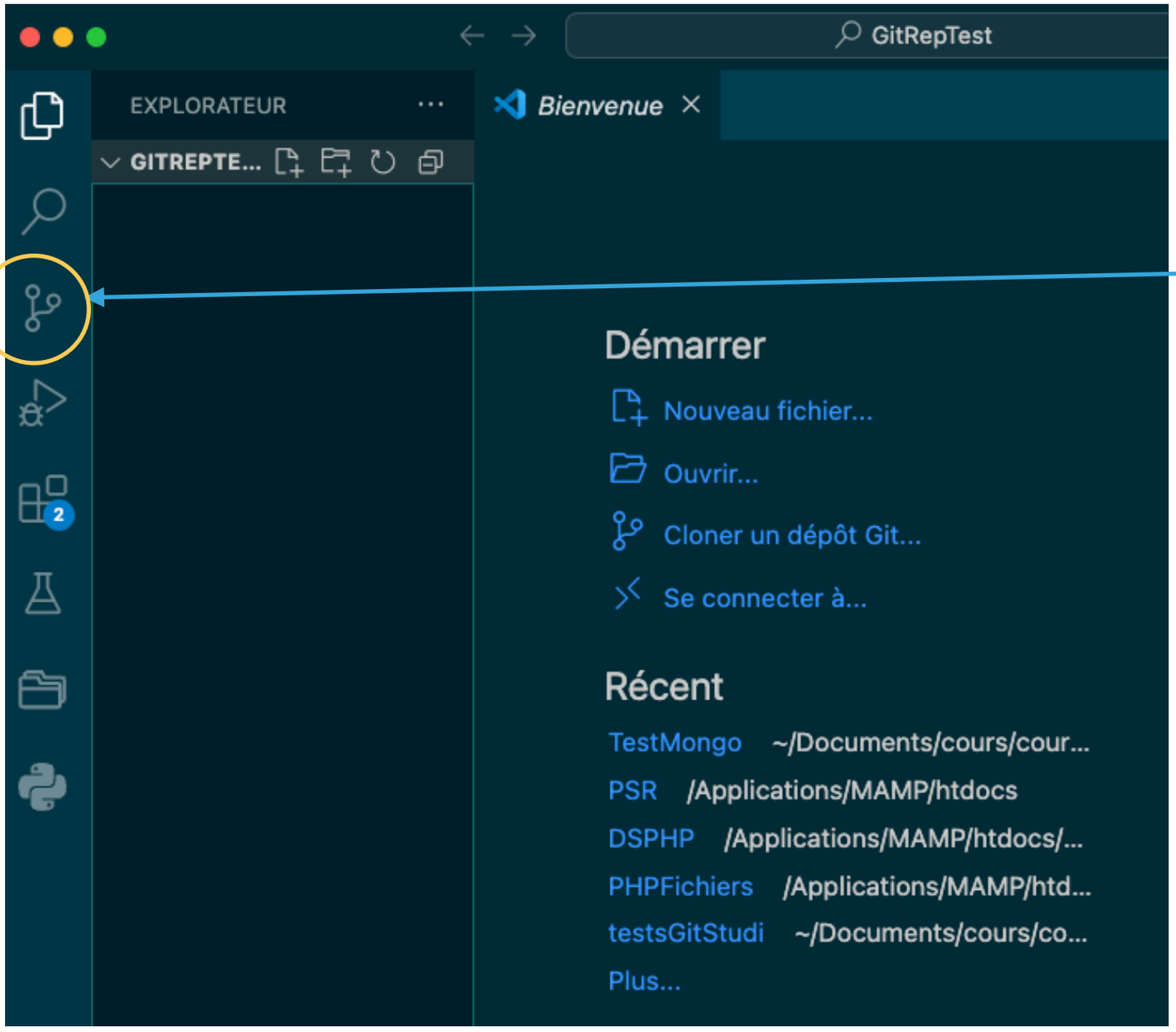
- Git est désynchronisée (tout est sur votre ordinateur en local : fichiers et historique)

- Un moment vous pouvez avoir besoin de vous synchroniser et cela se fera avec un serveur : dépôt (ex : Github)

- Sur ces serveurs on trouve des projets, ou dépôts ou repository…

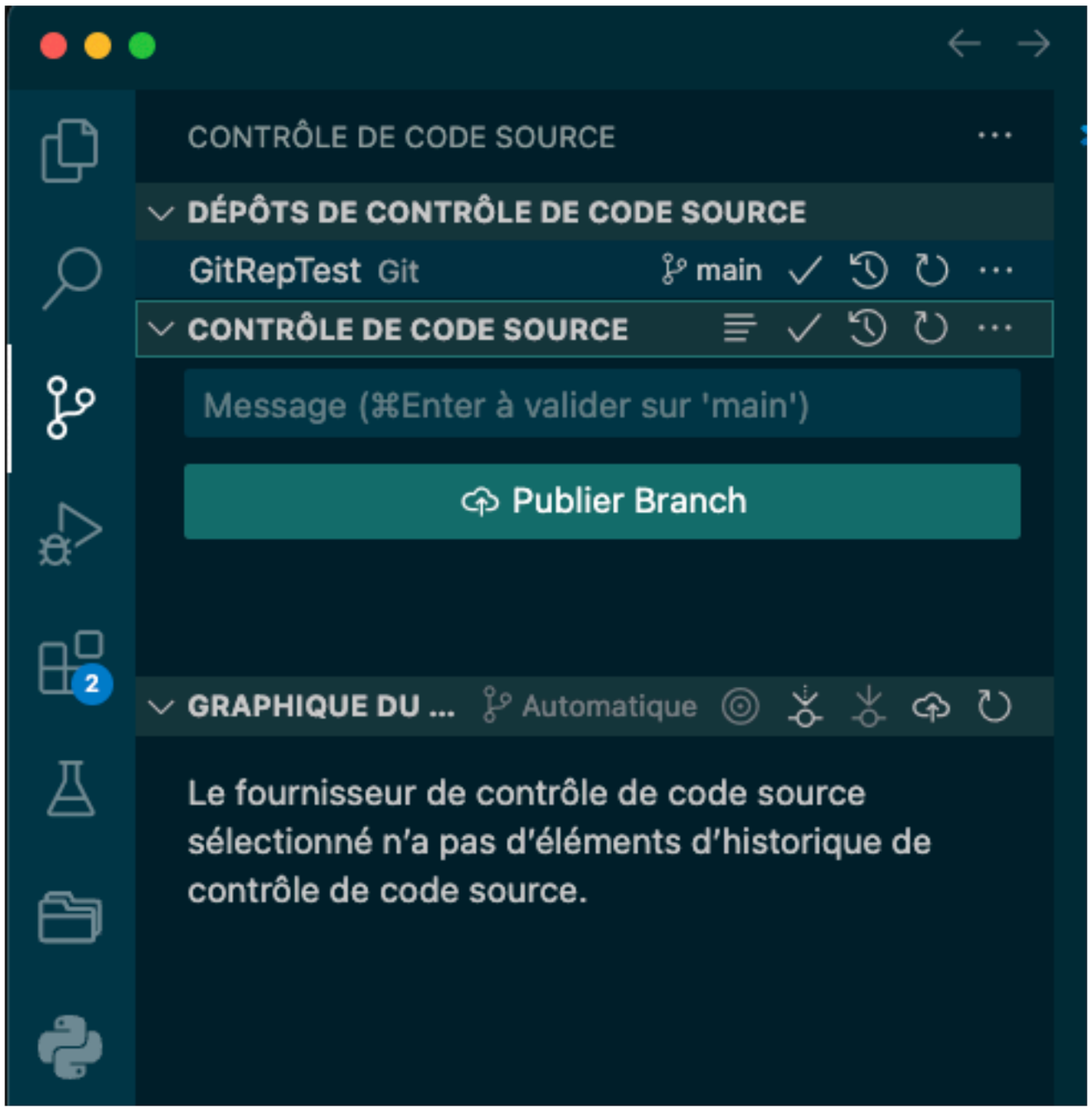
GIT ET LA GESTION DE VERSIONS

Sur VSC

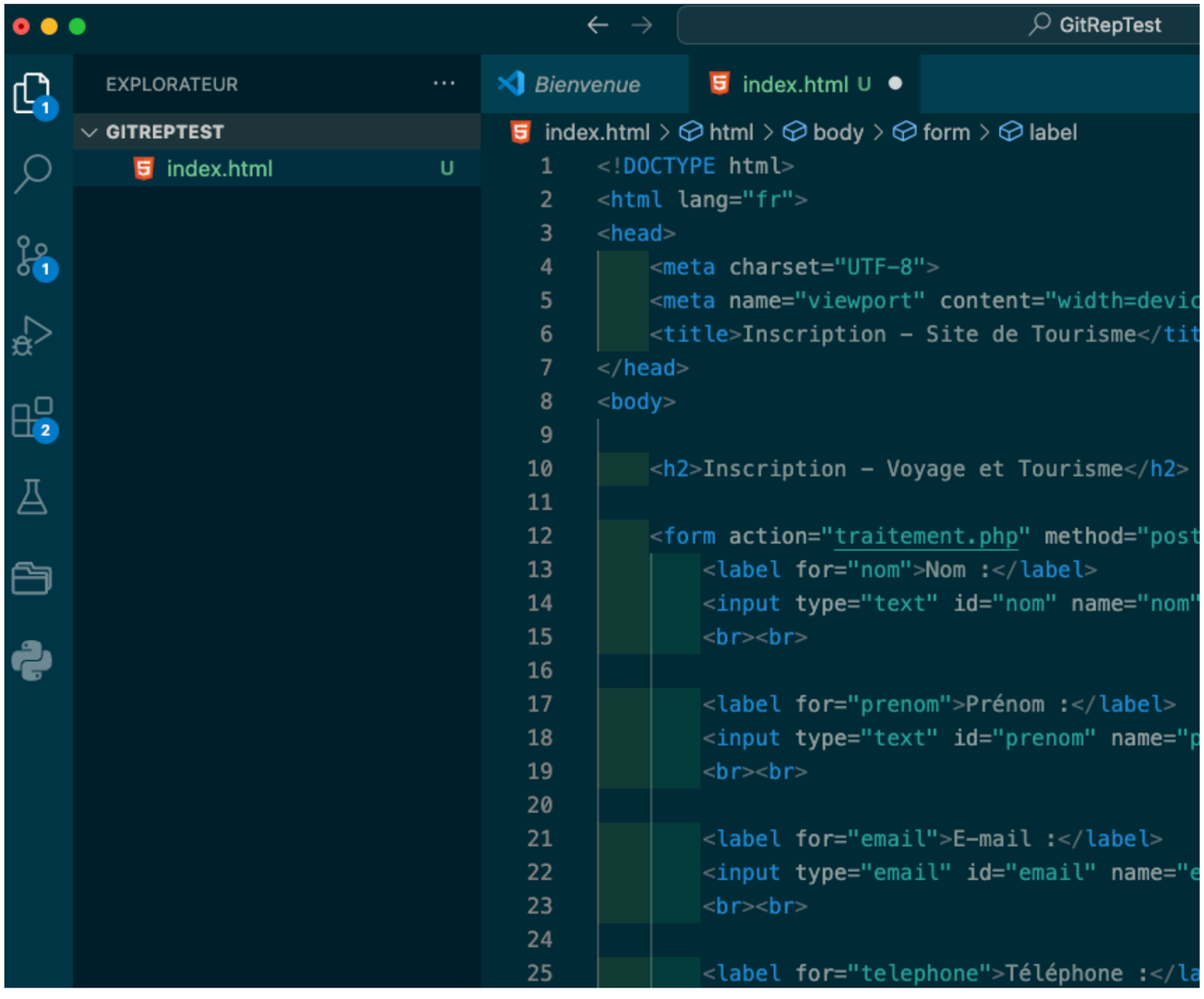


Initialisation d'un dépôt

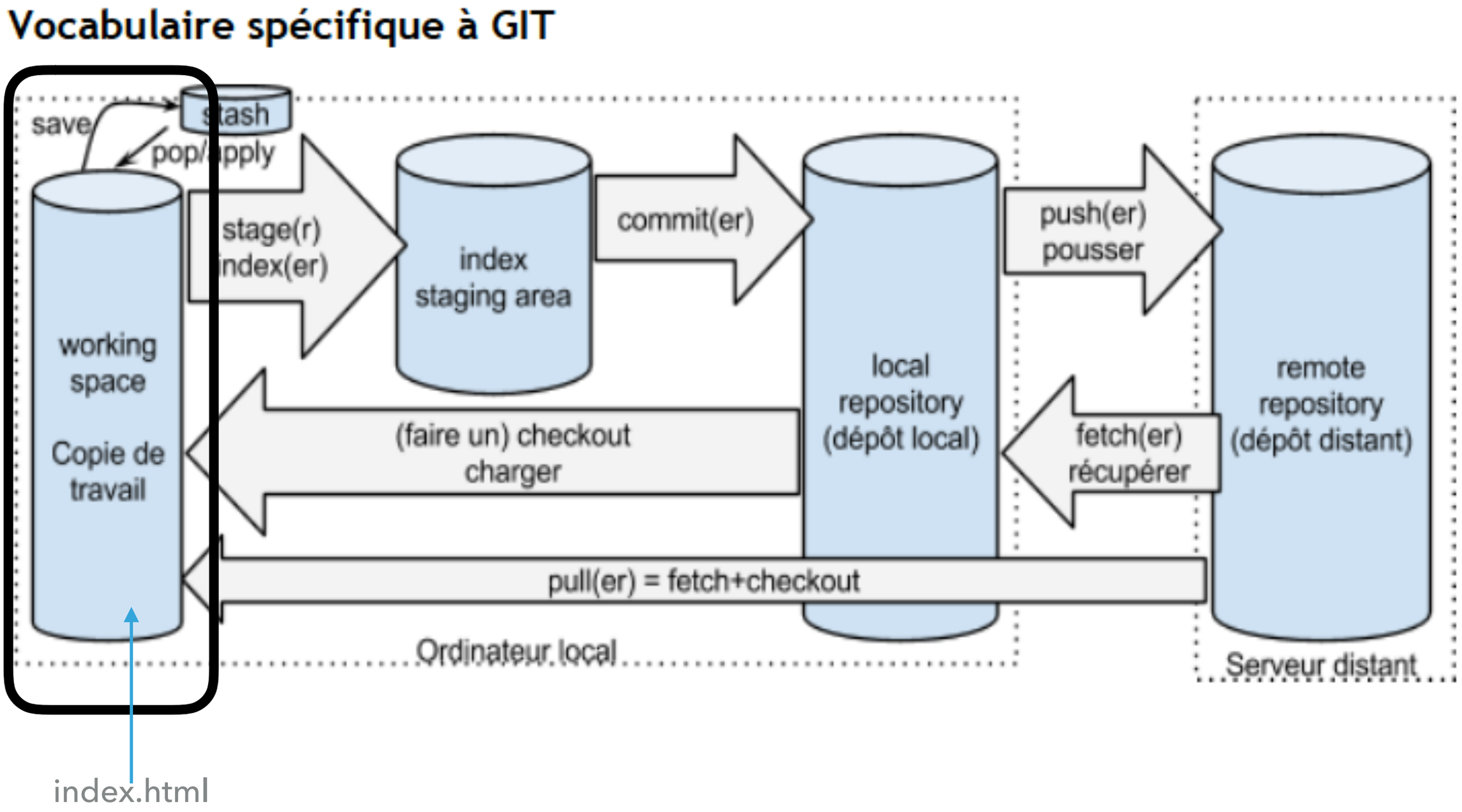
Sur VSC


CONTRÔLE DE CODE SOURCE
DÉPÔTS DE CONTRÔLE DE CODE SOURCE
GitRepTest Git
main
CONTRÔLE DE CODE SOURCE
Message (⌘Enter à valider sur 'main')
Publier Branch
2
GRAPHIQUE DU ...
Automatique
Le fournisseur de contrôle de code source sélectionné n'a pas d'éléments d'historique de contrôle de code source.

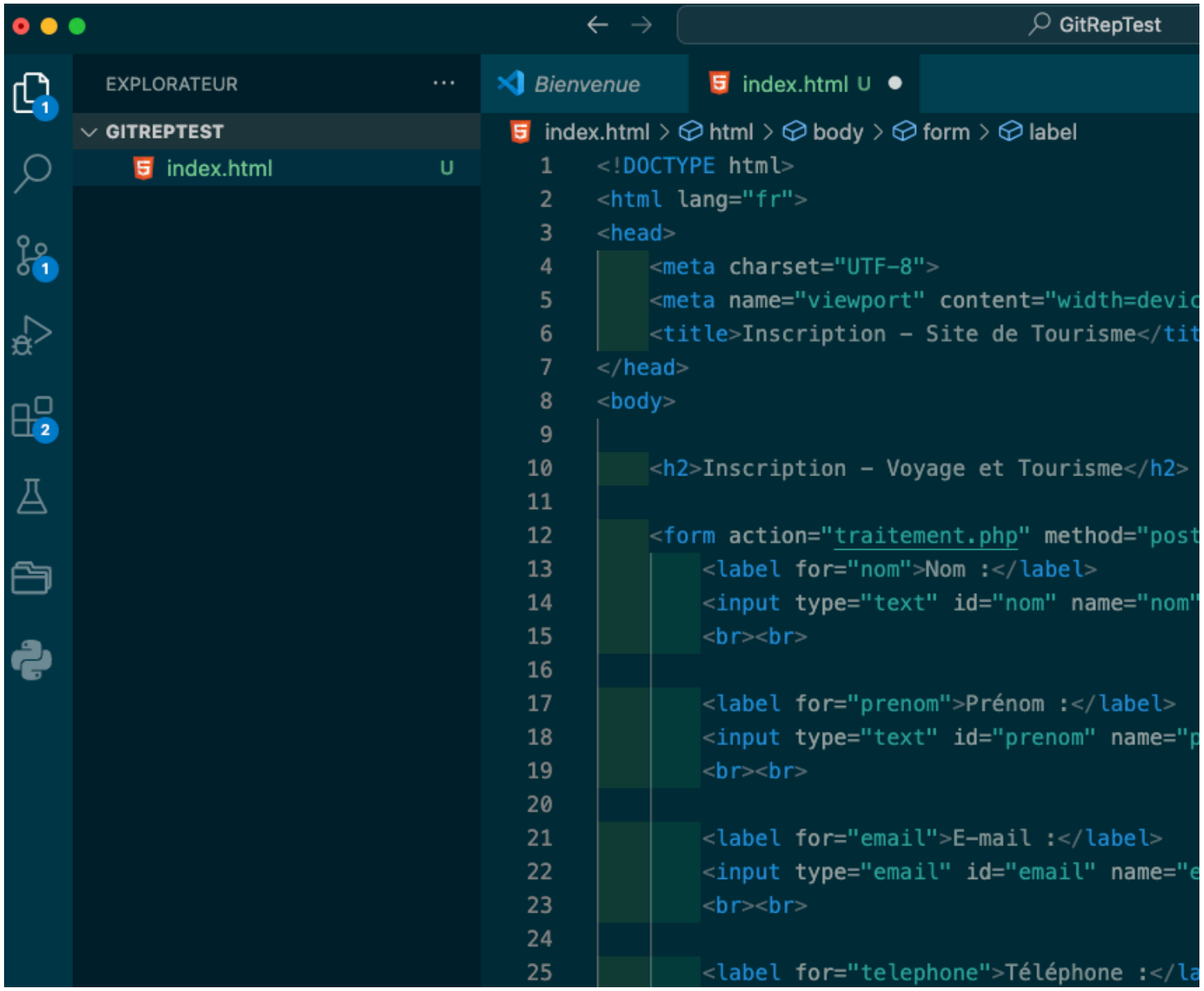
Sur VSC - Ajout d’un fichier

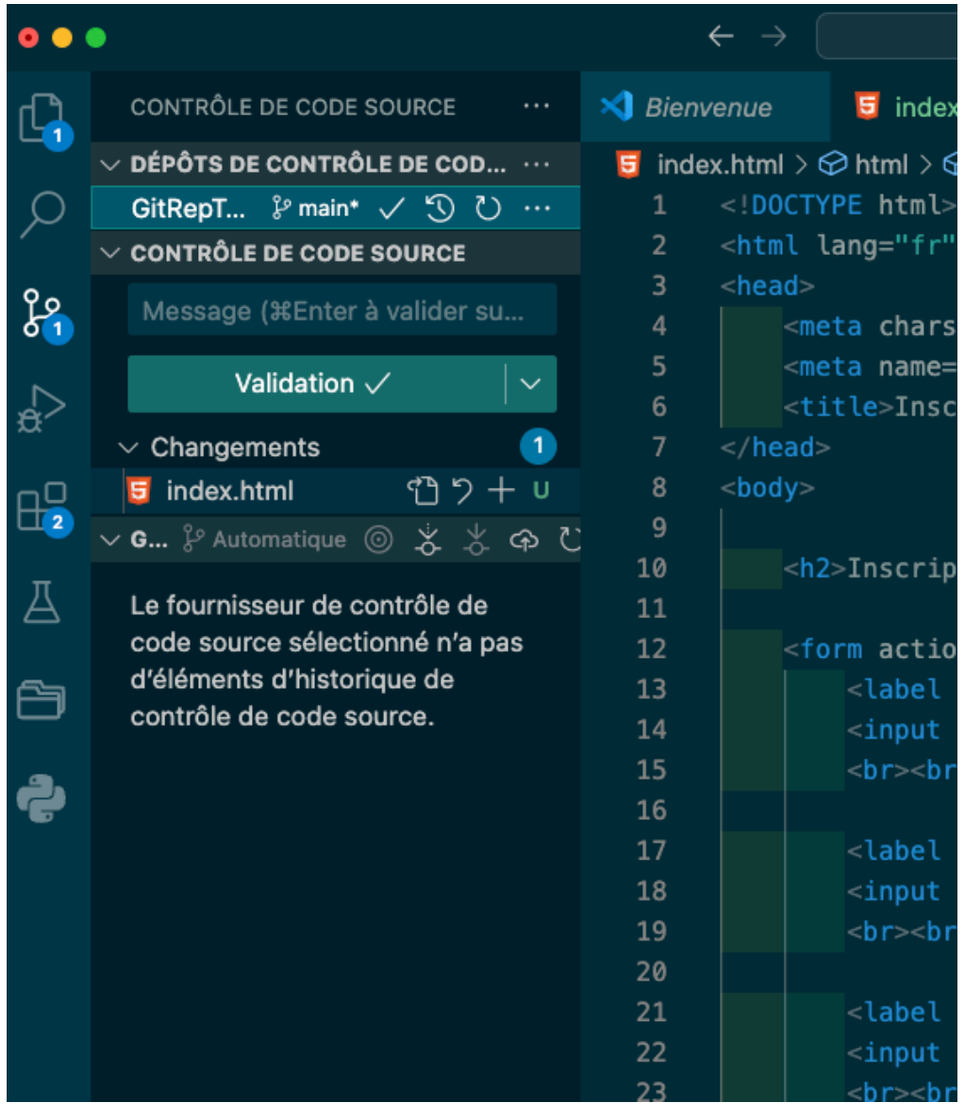


Vocabulaire spécifique à Git



Sur VSC - Commit

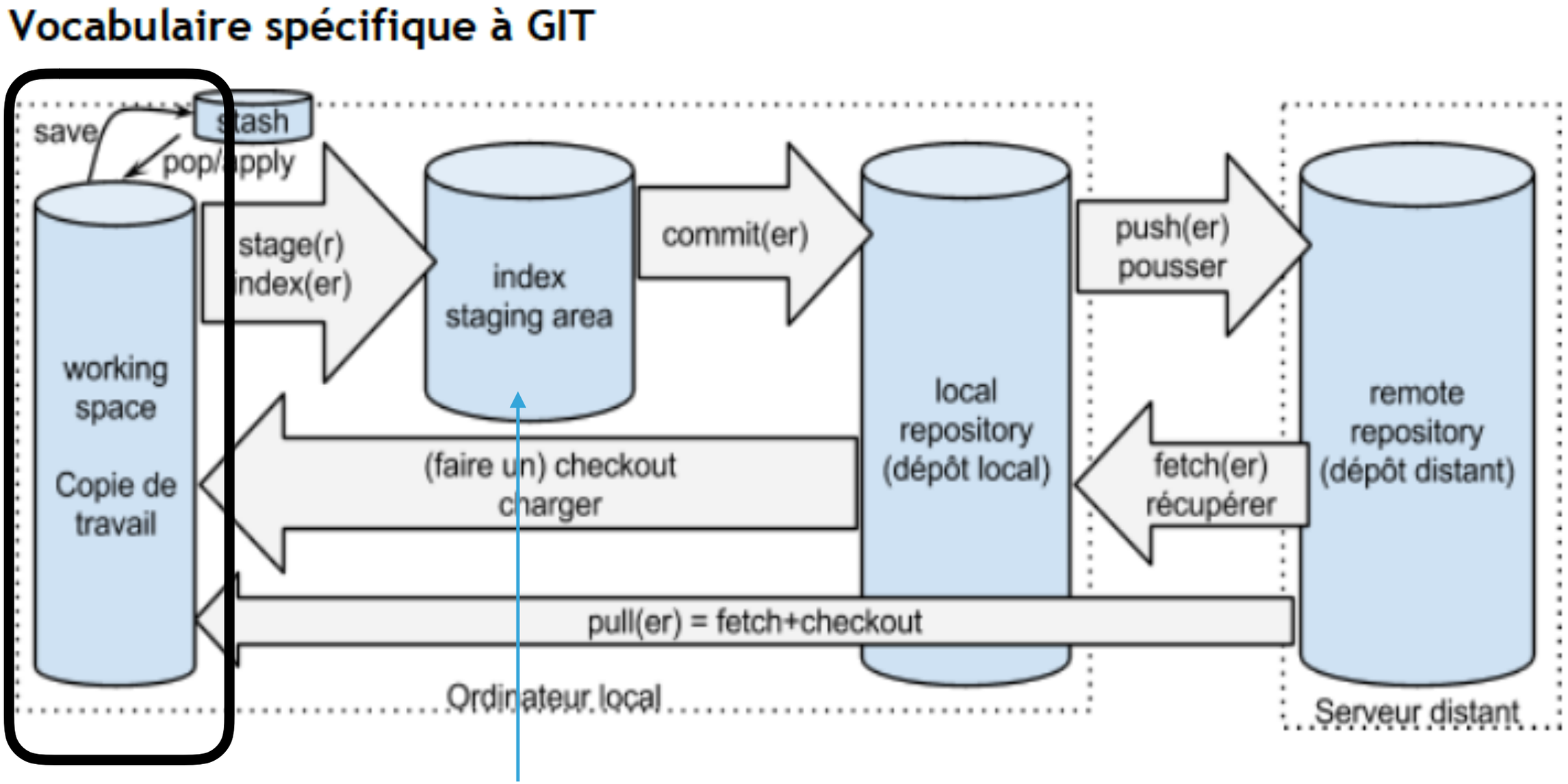




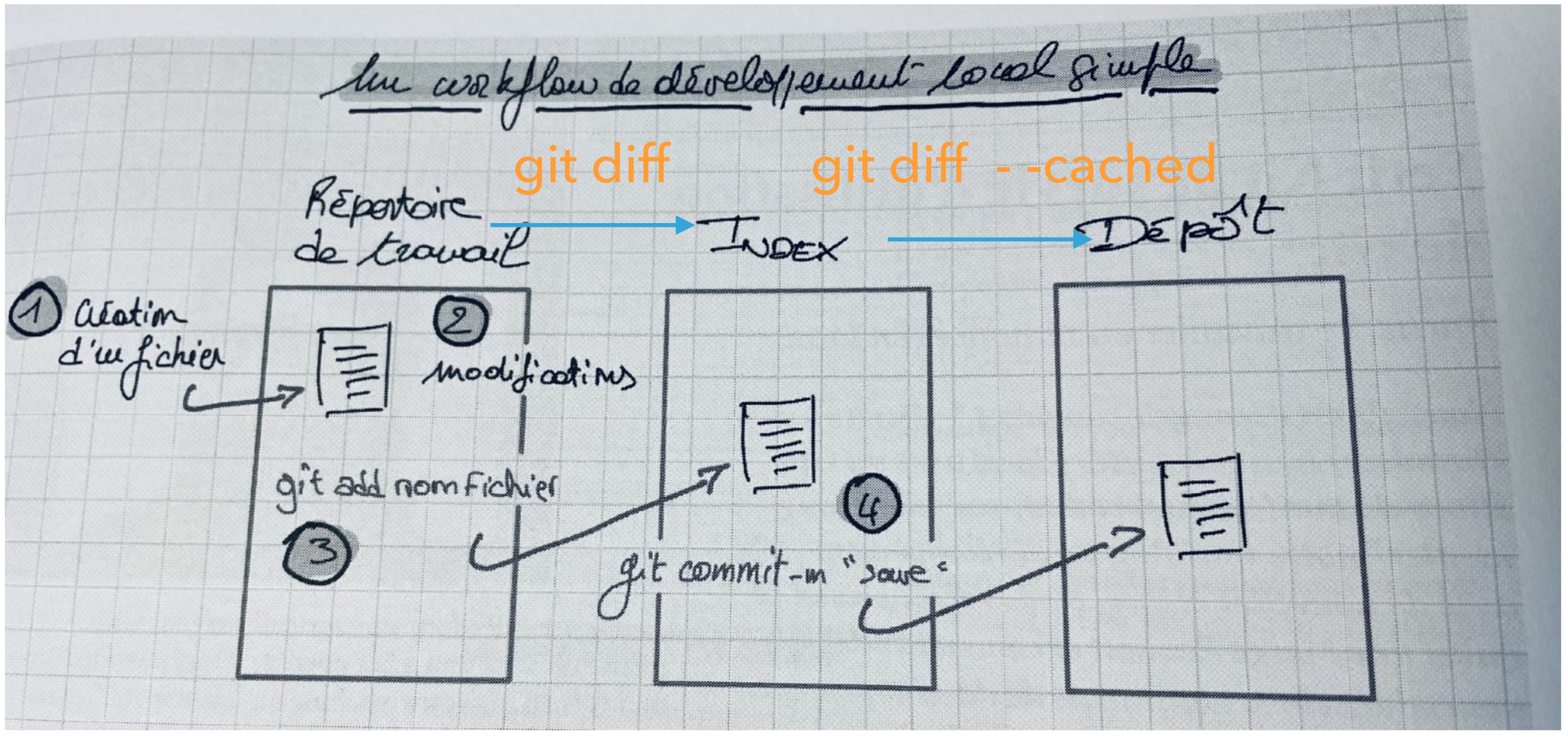
On peut **commiter** car on a fini la partie HTML de mon formulaire.

Rappel : Un commit n'est pas une sauvegarde mais une validation d'une étape dans la réalisation de votre projet !

Vocabulaire spécifique à Git



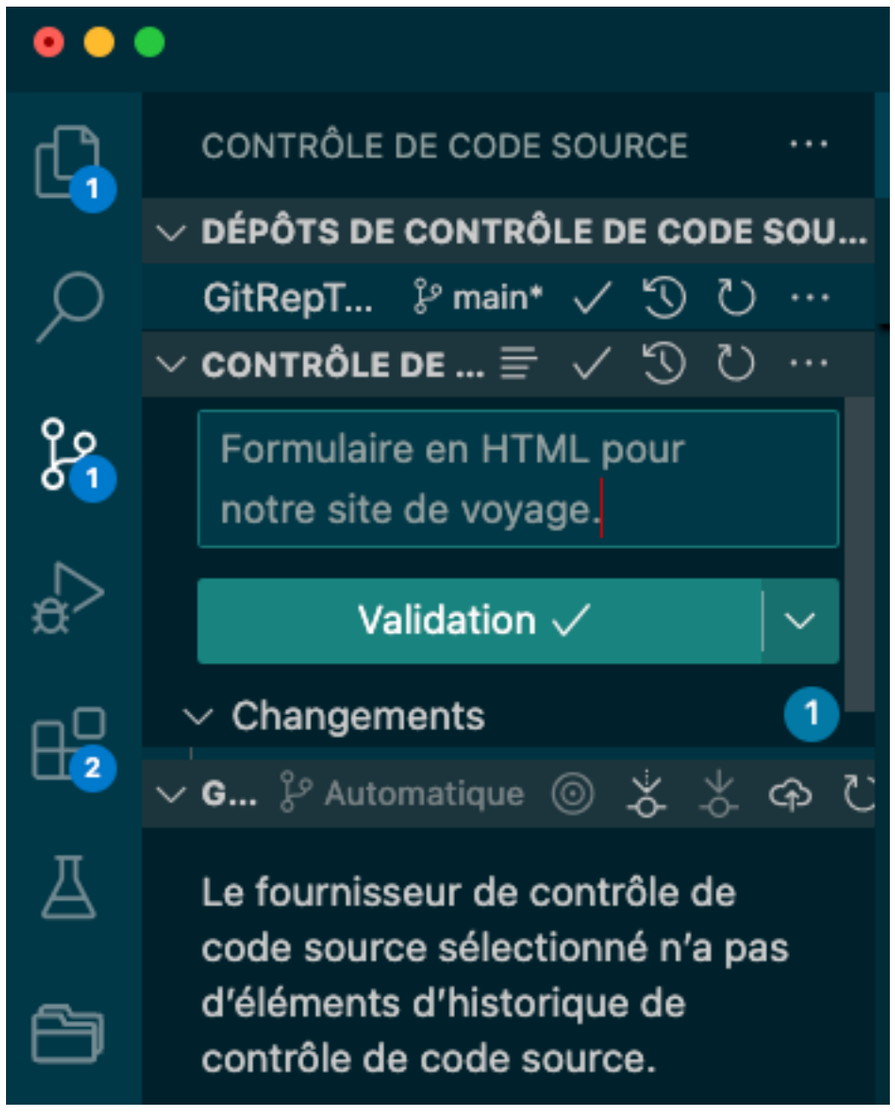
EN LOCAL



git status : a faire régulièrement pour savoir où vous en êtes.

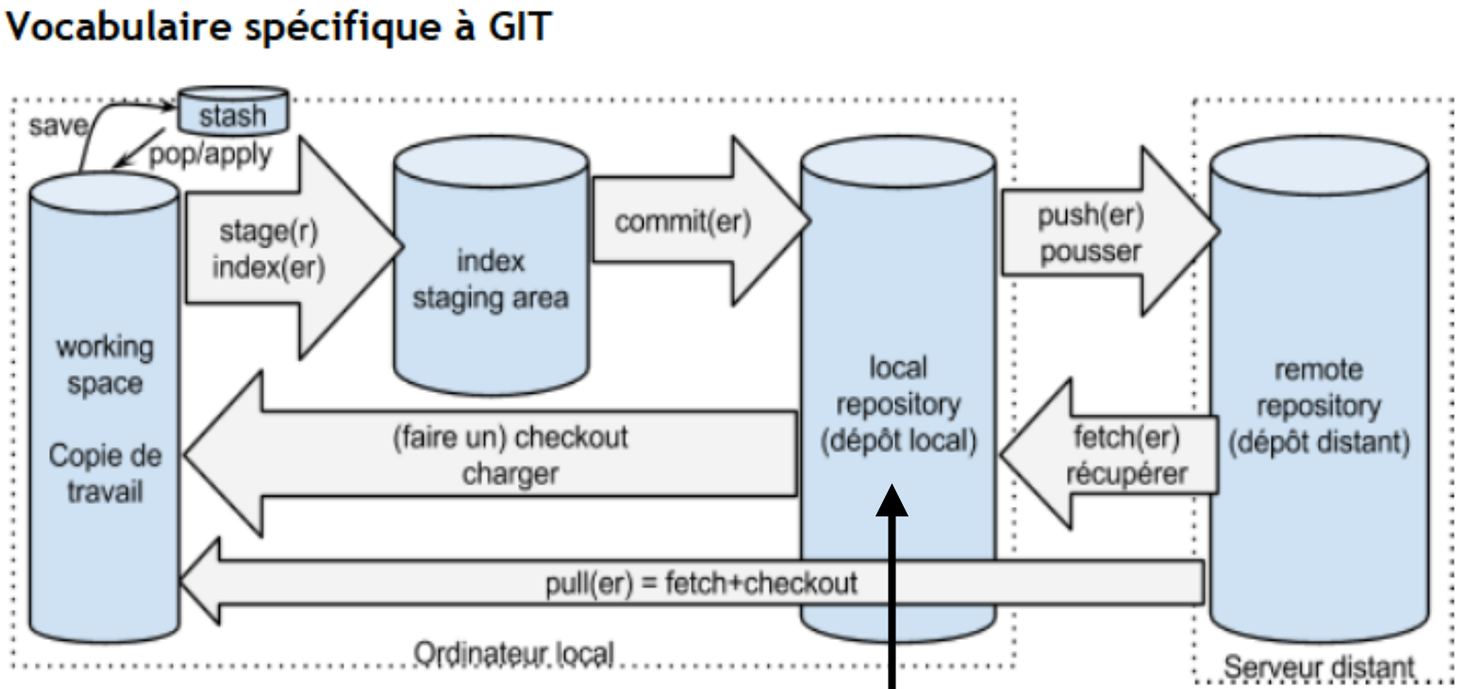
Étape 1 : Création ou modification d'un fichier dans le répertoire de travail.
Étape 2 : Ajout des modifications au staging area (ou index) avec git add nom_fichier.
Étape 3 : Enregistrement des modifications dans le dépôt local avec git commit -m "message".
Étape 4 : Les modifications sont désormais sauvegardées dans l'historique du dépôt Git.

Sur VSC - On commit



Il est important de bien remplir ce champ message !

Ajouter un fichier à l'Index



index.html

On teste notre projet

Inscription - Voyage et Tourisme

Nom :

Prénom :

E-mail :

Téléphone :

Destination préférée : Paris, France

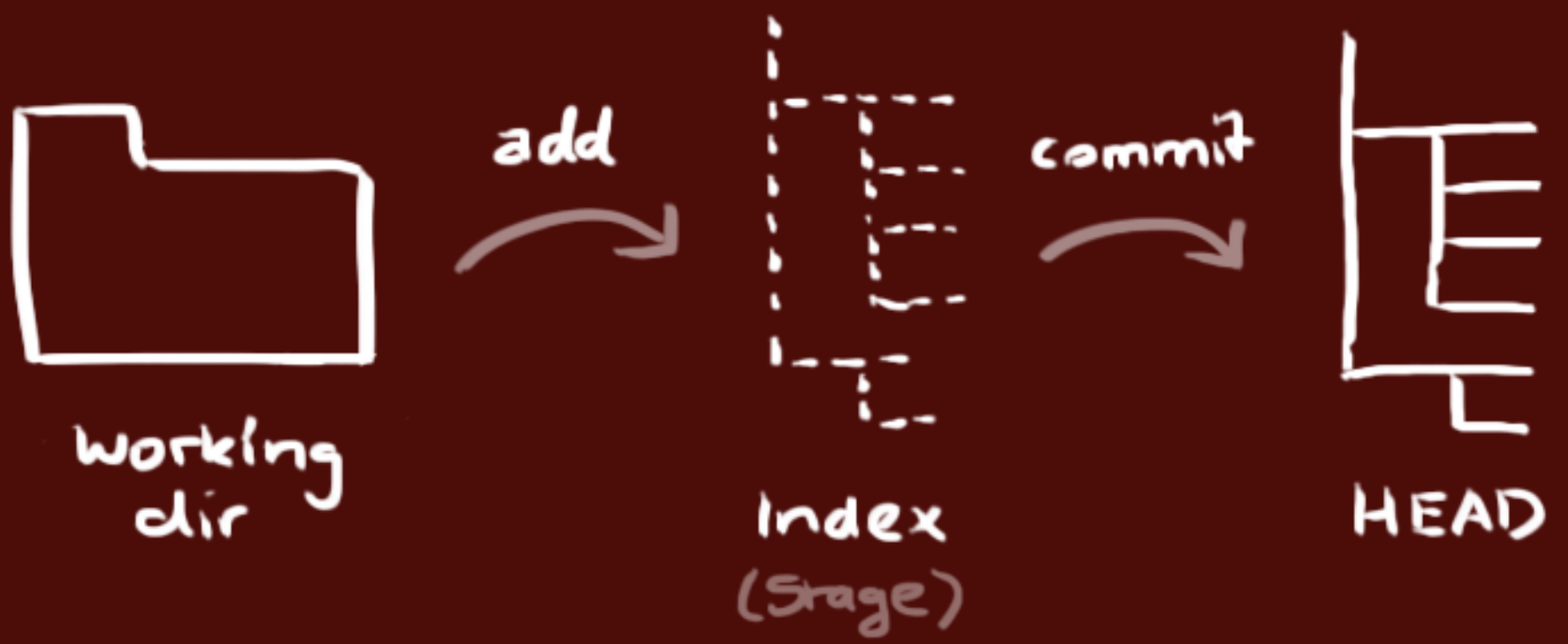
Pourquoi souhaitez-vous voyager ?

S'abonner à notre newsletter

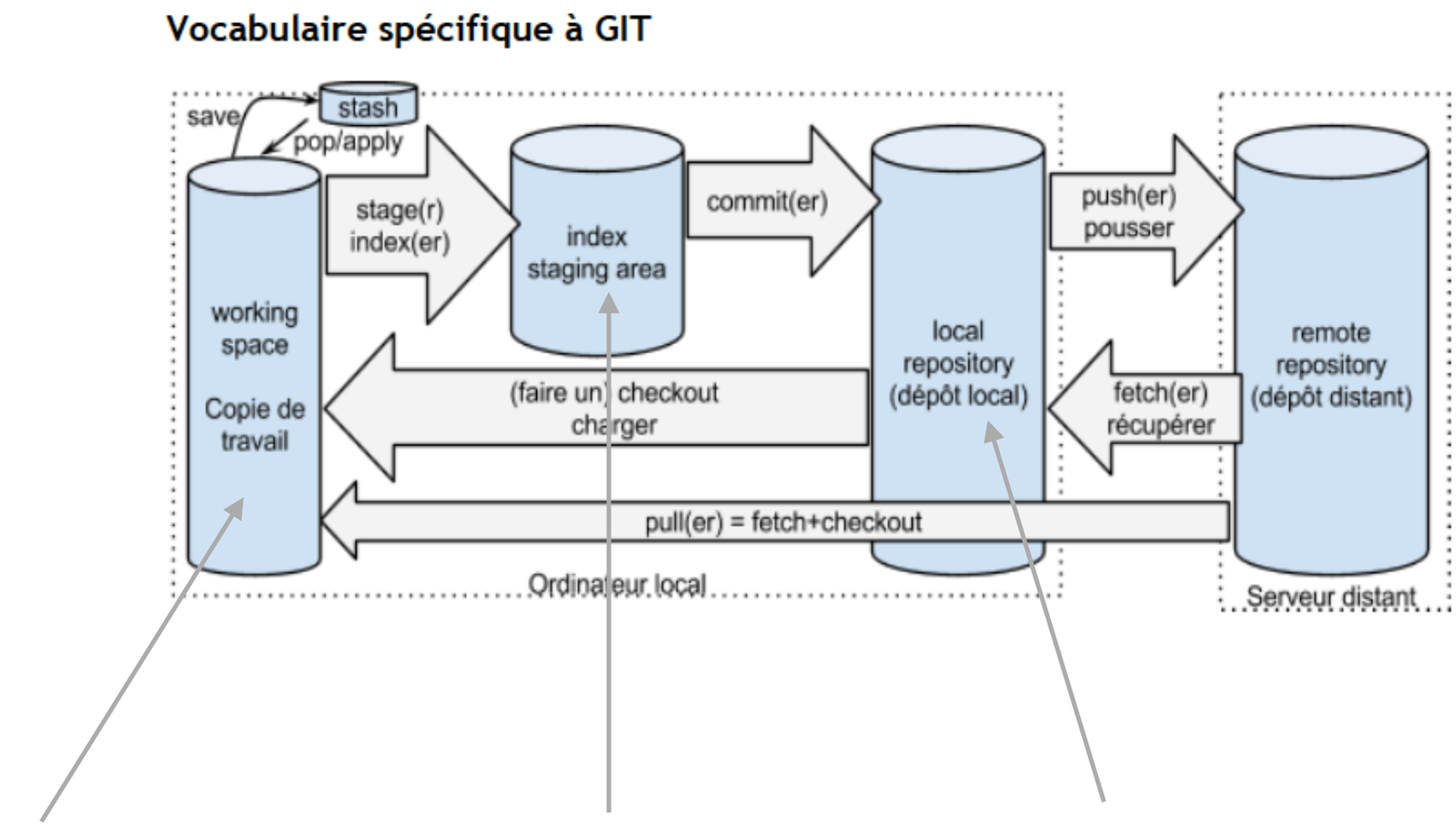
S'inscrire

arbres

votre dépôt local est composé de trois "arbres" gérés par git. le premier est votre `espace de travail` qui contient réellement vos fichiers. le second est un `Index` qui joue un rôle d'espace de transit pour vos fichiers et enfin `HEAD` qui pointe vers la dernière validation que vous ayez fait.



Lister les fichiers



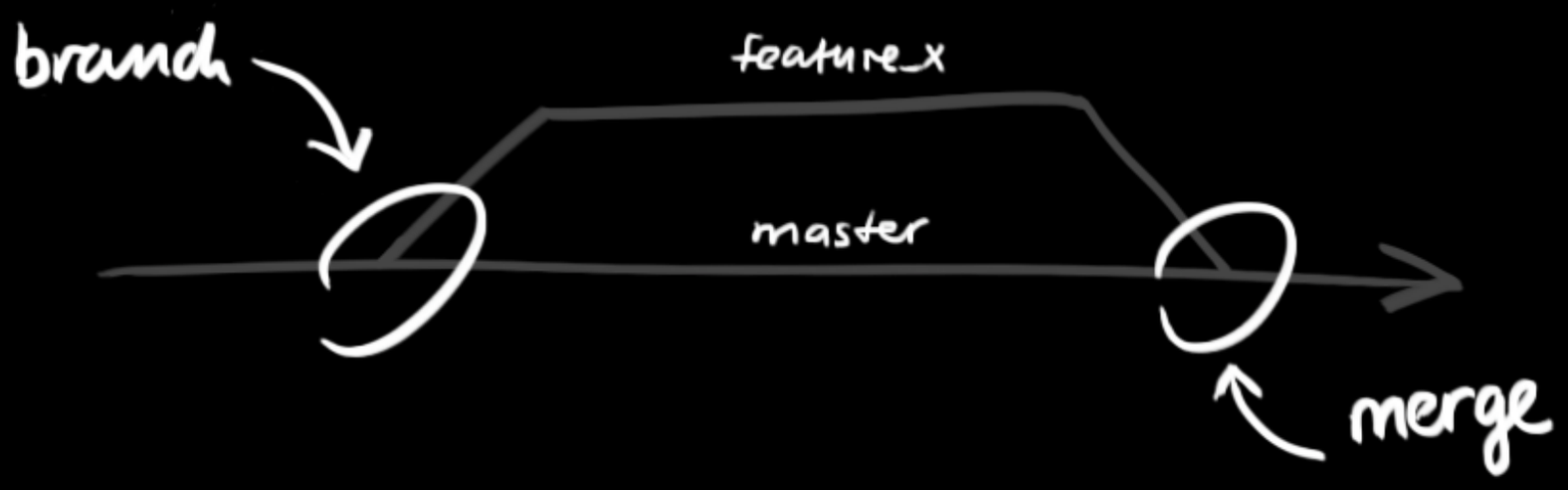
ls **git ls-files —stage** **git ls-tree -r HEAD**

LE FLUX DE TRAVAIL

Branches

branches

Les branches sont utilisées pour développer des fonctionnalités isolées des autres. La branche *master* est la branche par défaut quand vous créez un dépôt. Utilisez les autres branches pour le développement et fusionnez ensuite à la branche principale quand vous avez fini.



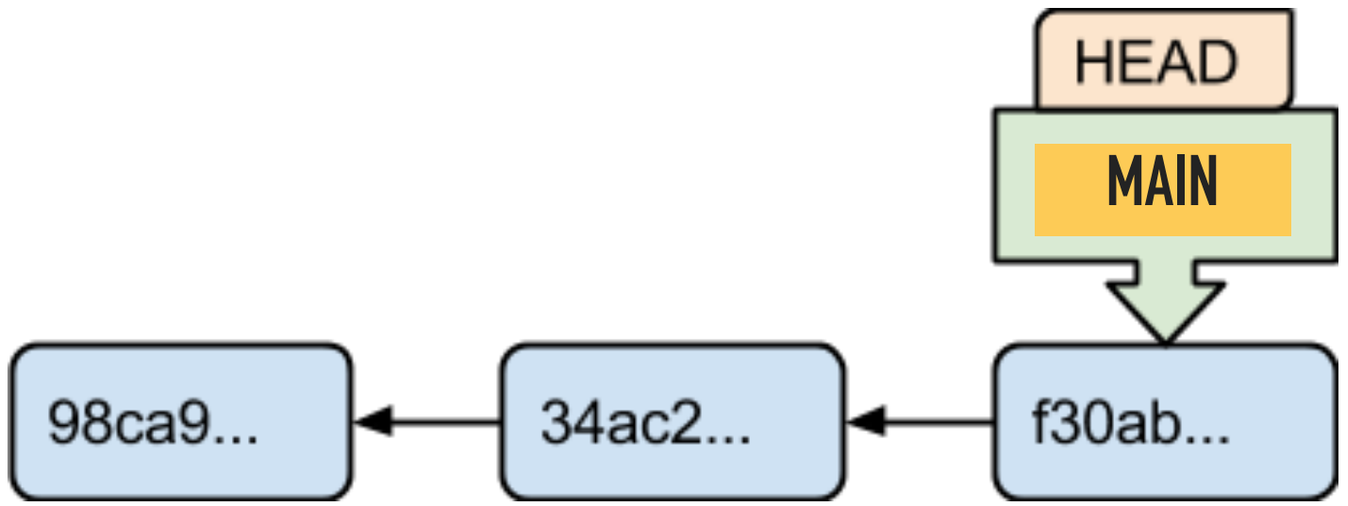
Git et les branches

Première branche : **main**

Une branche pointe sur un commit

Git et les branches

Un pointeur spécial nommé HEAD pointe vers la branche courante de travail !

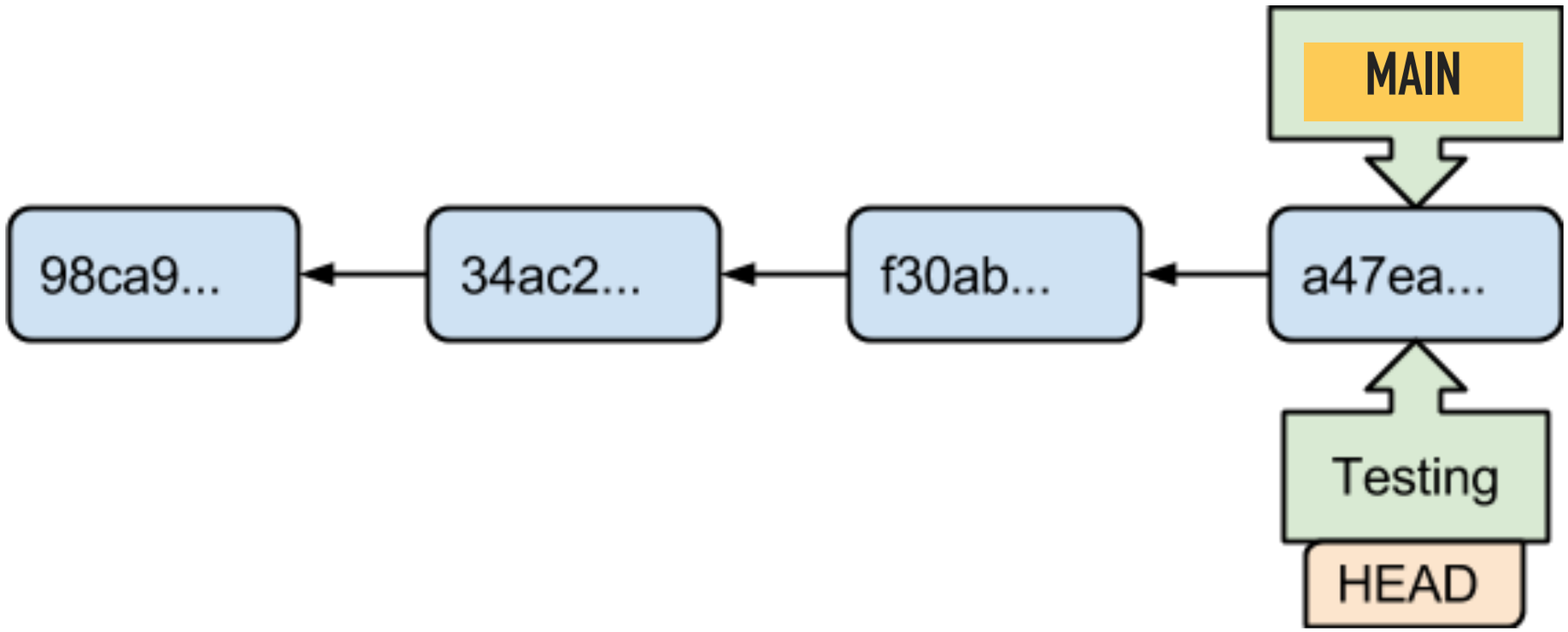


Git et les branches

Création d'une nouvelle branche

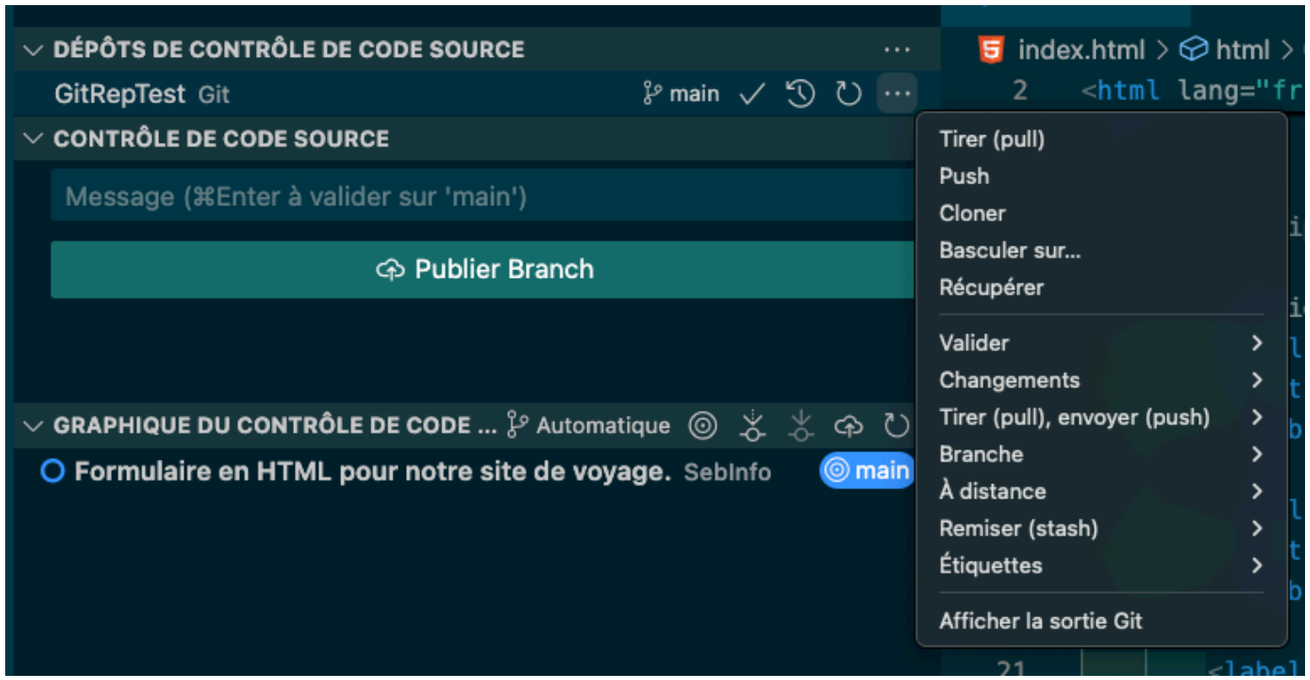
$git branch Testing

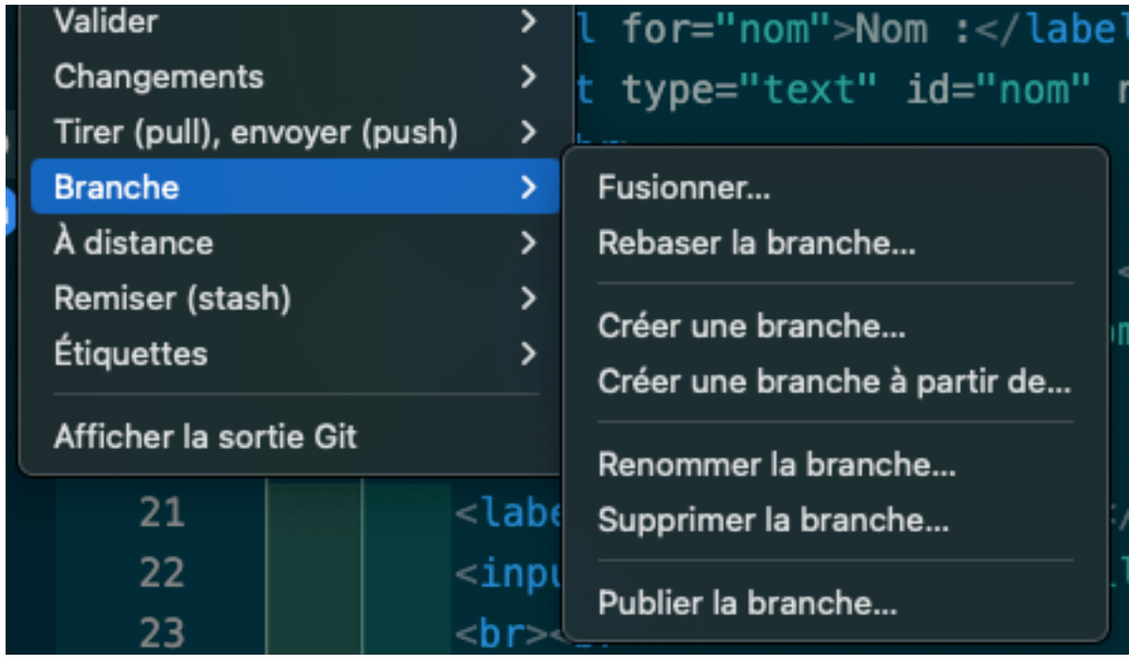
$git checkout Testing



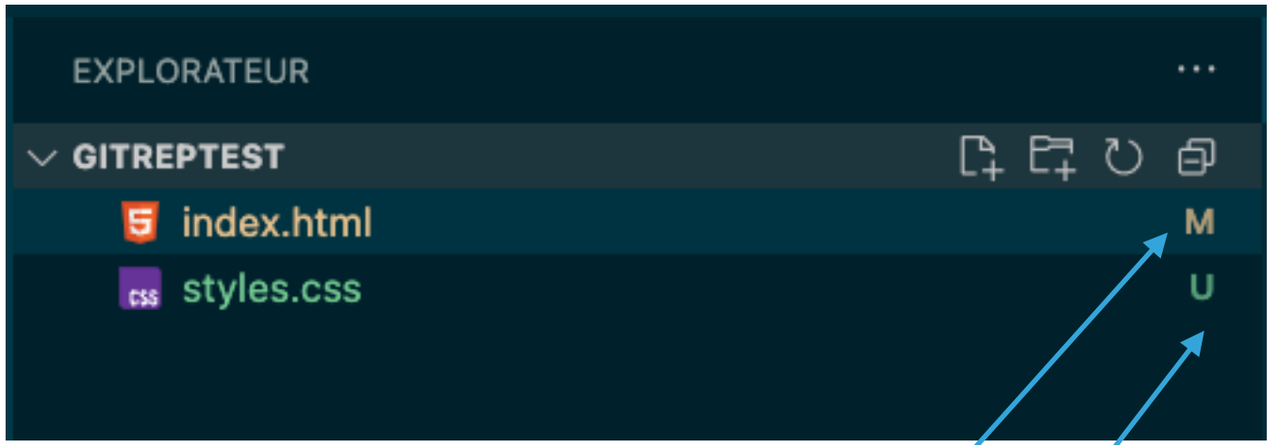
Les branches

On va créer une branche pour tester notre projet en ajoutant du CSS





Les branches

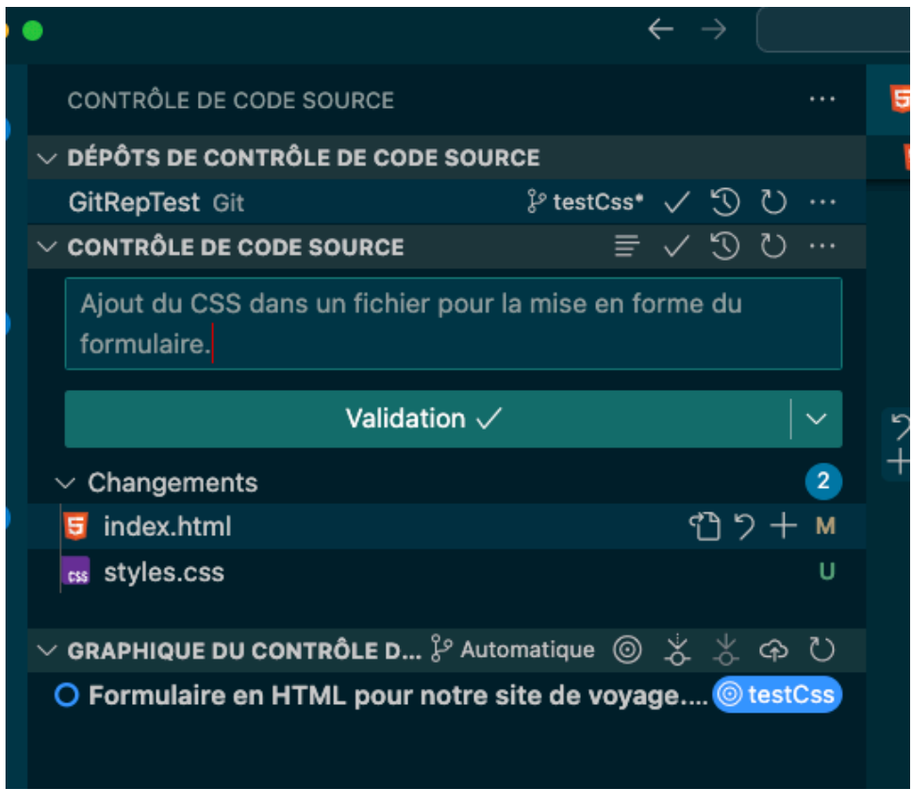


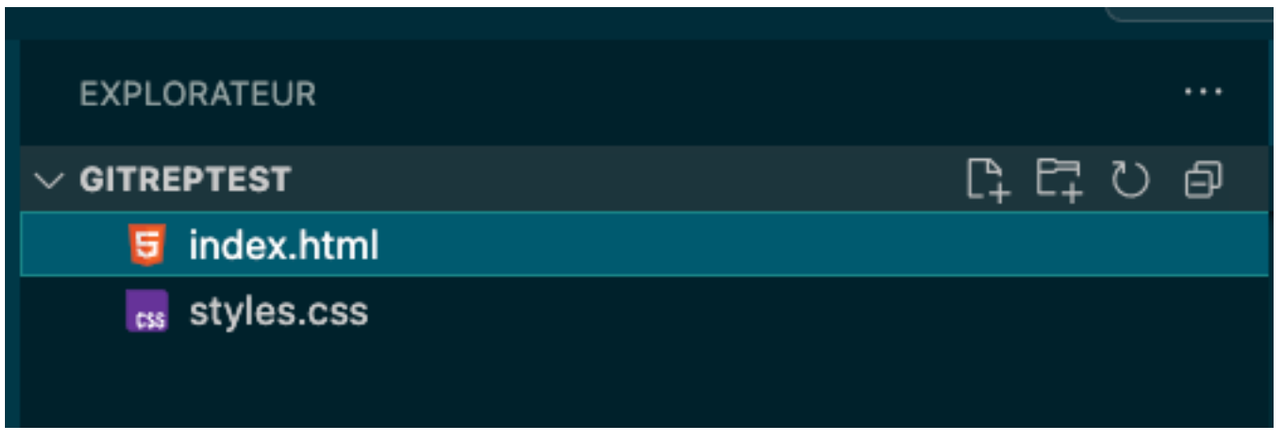
index.html (M - Modified) → Ce fichier a été modifié, mais les changements n’ont pas encore été committés.

styles.css (U - Untracked) → Ce fichier est nouveau et non suivi par Git, ce qui signifie qu'il n'a jamais été ajouté à l'index.

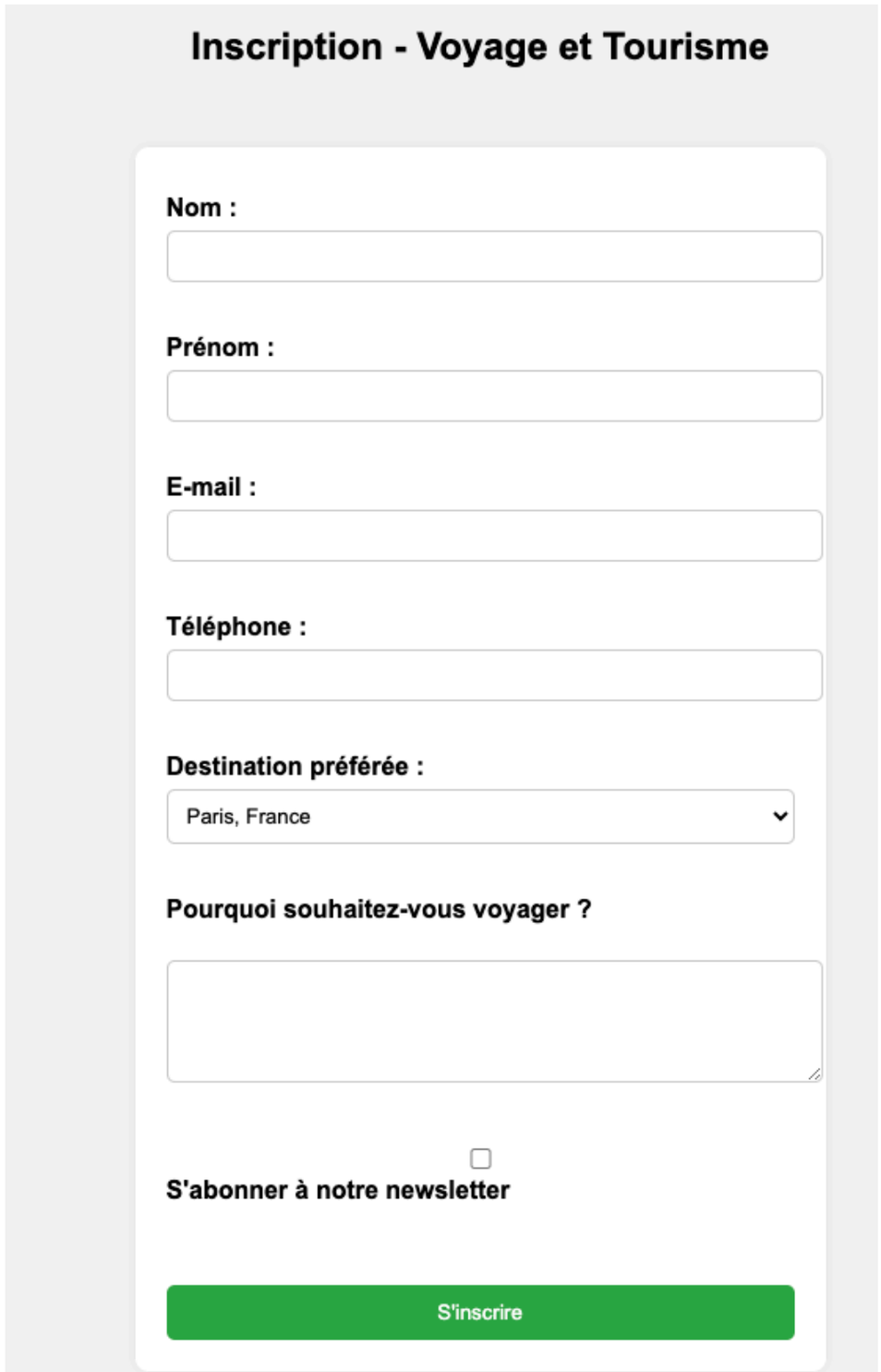
Les branches

On commit dans cette nouvelle branche

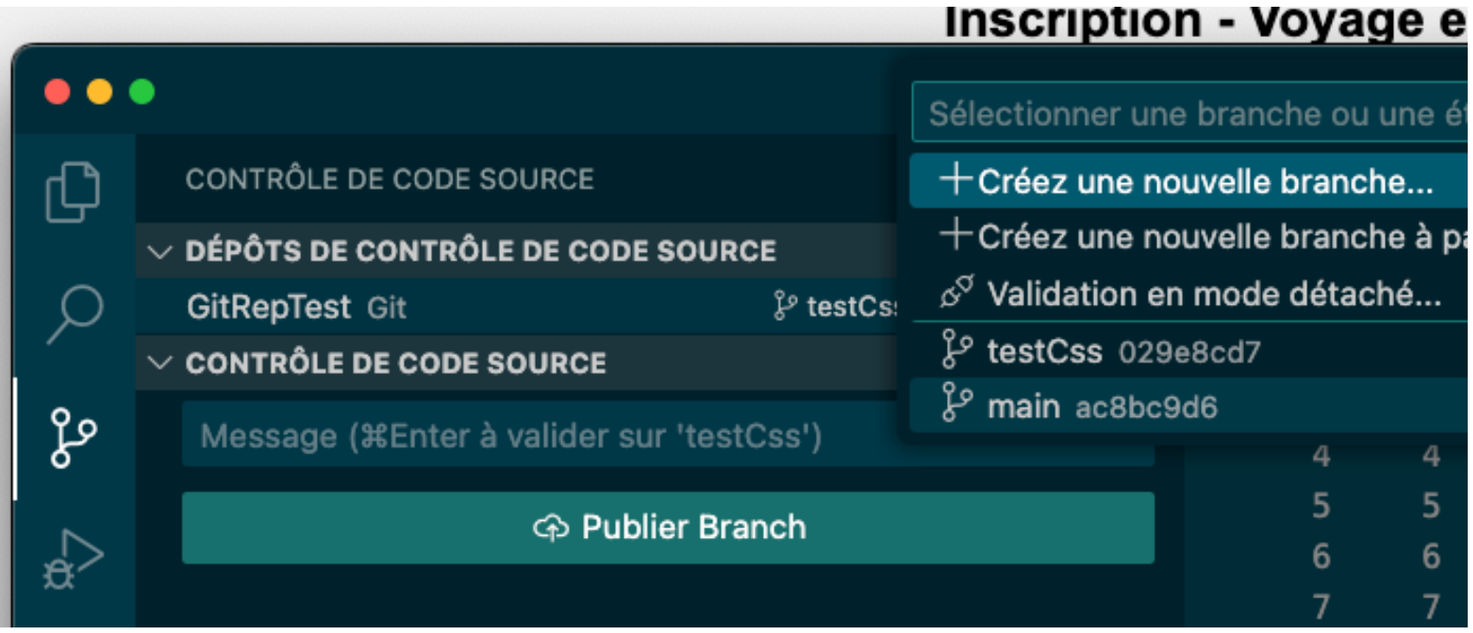


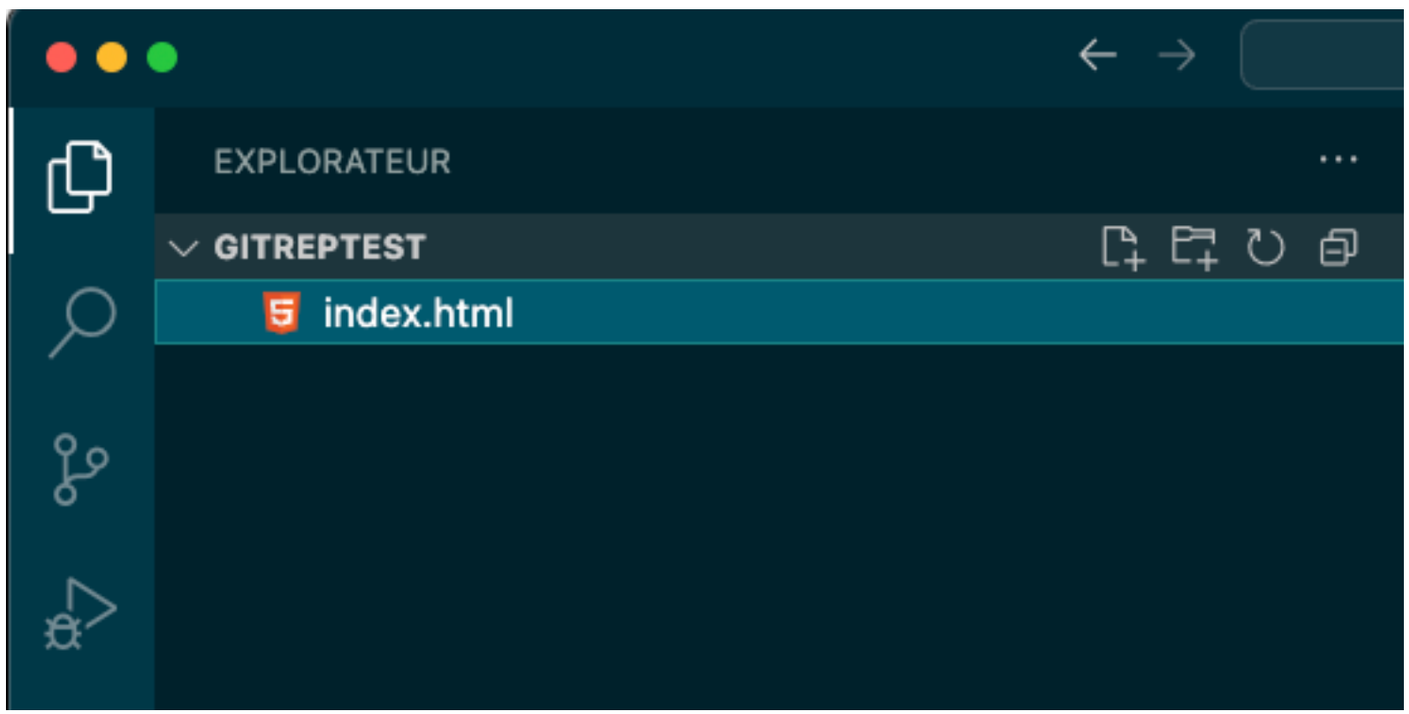


On test la branche testCss



On repasse sur la branche main





Inscription - Voyage et Tourisme

Nom :

Prénom :

E-mail :

Téléphone :

Destination préférée : Paris, France

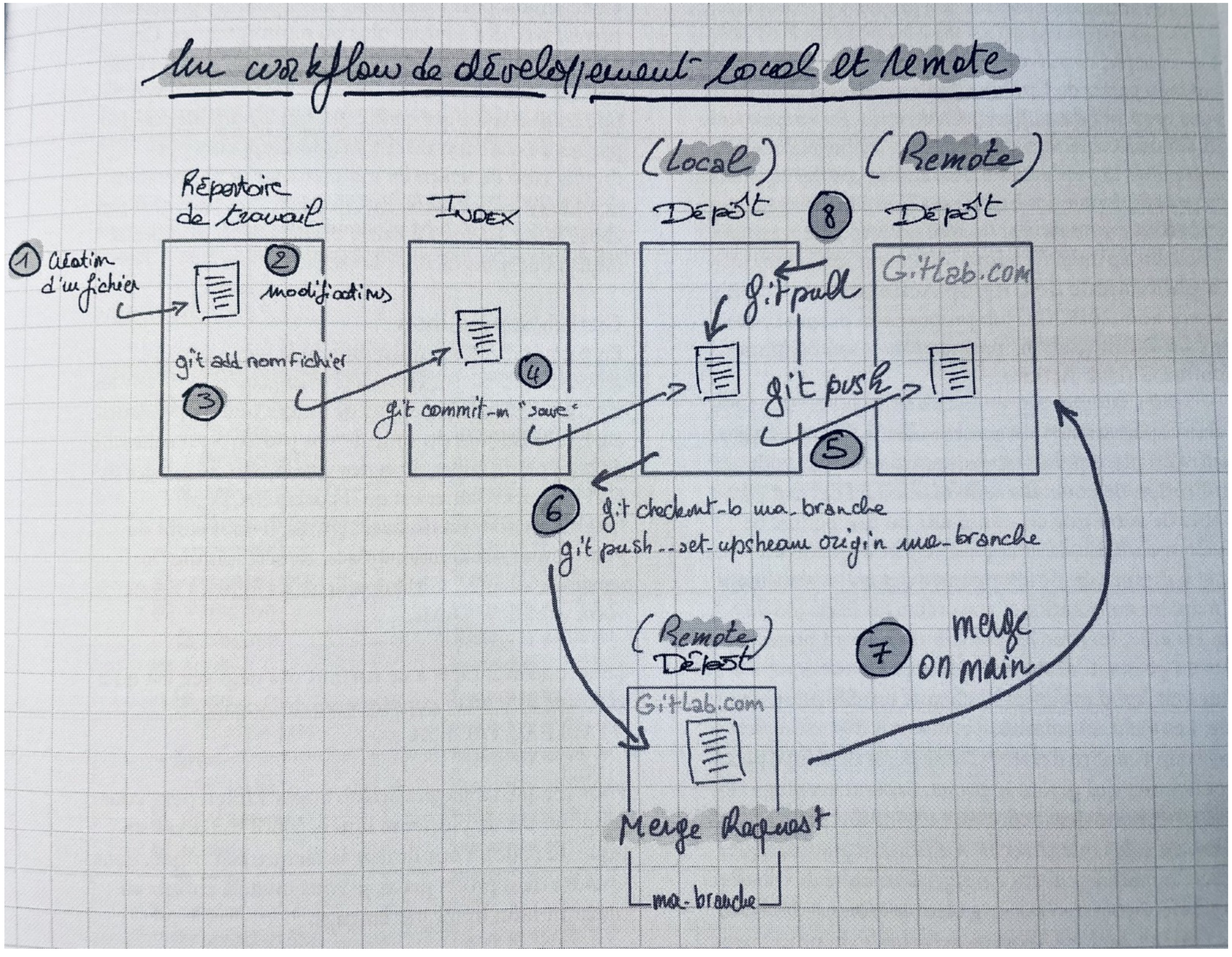
Pourquoi souhaitez-vous voyager ?

S'abonner à notre newsletter

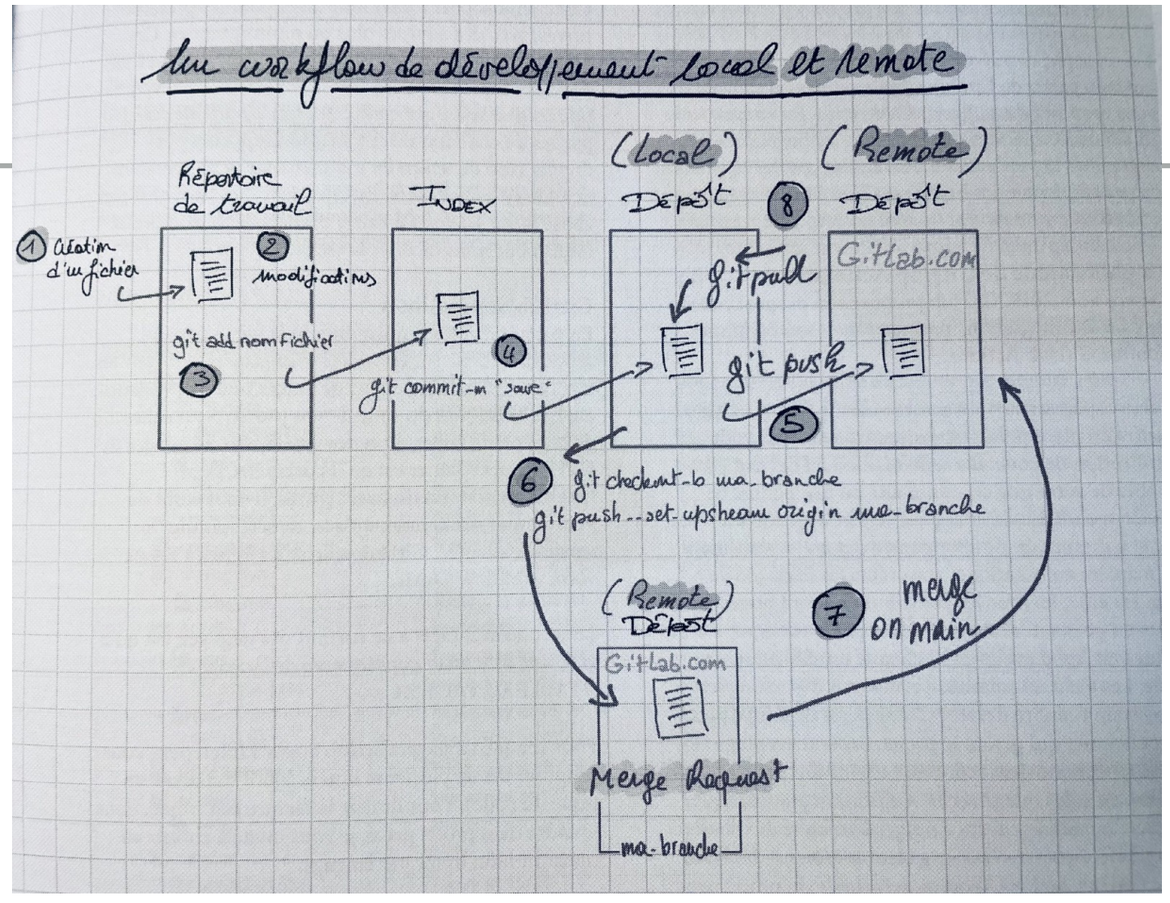
S'inscrire

PASSER PAR UN DÉPÔT

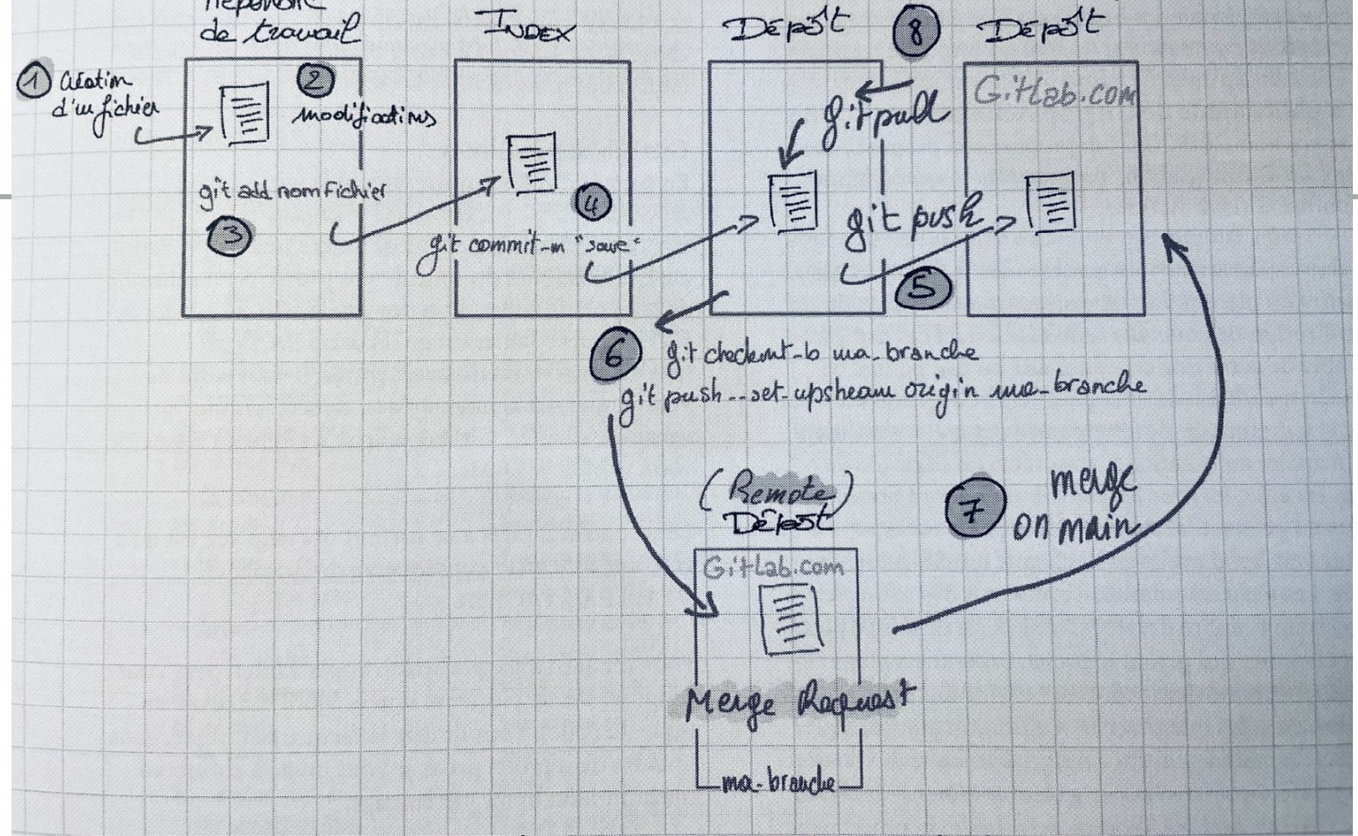
FLUX DE TRAVAIL



FLUX DE TRAVAIL



Étape 1 : Création ou modification d'un fichier dans le répertoire de travail.
Étape 2 : Ajout des modifications au staging area (ou index) avec la commande git add nom_fichier.
Étape 3 : Enregistrement des modifications dans le dépôt local avec git commit -m "message".
Étape 4 : Création d'une branche locale avec git checkout -b ma_branche et association au dépôt distant av
git push --set-upstream origin ma_branche.
Étape 5 : Envoi des modifications sur le dépôt distant (ex. GitLab ou GitHub) avec git push. Après avoir créé
une **nouvelle branche** et fait des modifications dessus, il faut l'envoyer sur le dépôt distant (ex: **GitHub/ GitLab**) pour que d'autres développeurs puissent la voir et l'examiner.
Étape 6 : Création d'une merge request dans l'interface du dépôt distant pour proposer la fusion des changements.
Étape 7 : Validation et fusion de la branche sur la branche principale (ex. main).
Etape 8 : permet de récupérer les dernières modifications du dépôt distant vers le dépôt local.

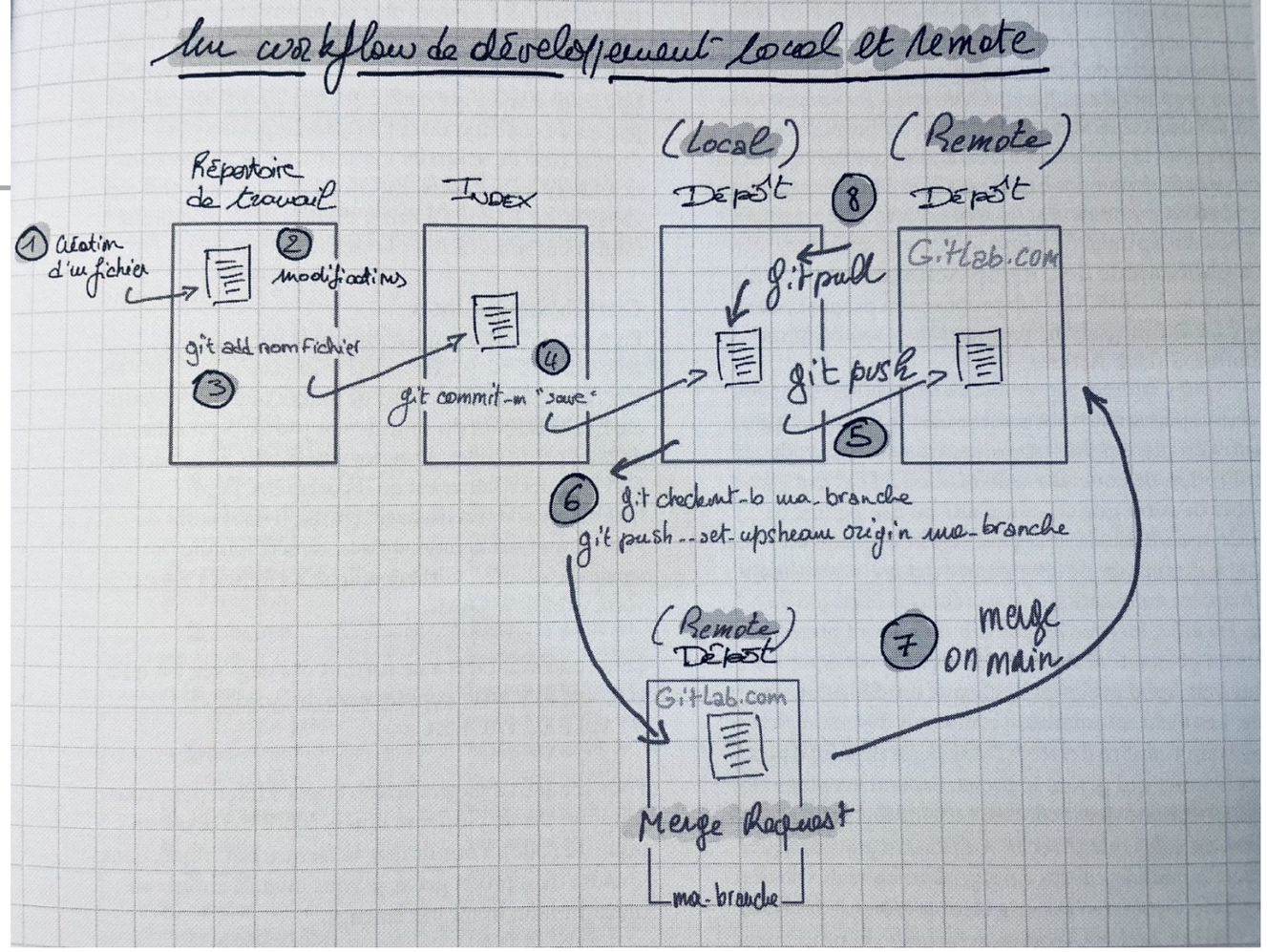


L'étape 5 est importante car elle permet d'envoyer les commits **du dépôt local vers le dépôt distant** (ex: GitHub/ GitLab).
Rendre les modifications **accessibles aux autres développeurs**.
Prépare le terrain pour une Merge Request (Pull Request).

L'étape 6 : **Demander une revue de code avant d'intégrer la branche dans `main`.**
Permettre aux autres développeurs de vérifier et commenter le code.
Éviter d'introduire des erreurs ou bugs dans `main`.
Comment faire sur GitHub/GitLab ?

1. Aller sur **GitHub/GitLab**.
2. Sélectionner la branche `ma-branche`.
3. Cliquer sur "**New Merge Request**" **(GitLab) / "New Pull Request" (GitHub)**.
4. Comparer avec `main` et ajouter un **titre + description**.
5. Assigner des **reviewers** pour examiner le code.

Le code est **prêt à être fusionné dans `main`**, mais attend une validation.



L'étape 7 : Fusion de la branche avec `main` (`merge`)

📌 **Rôle** :

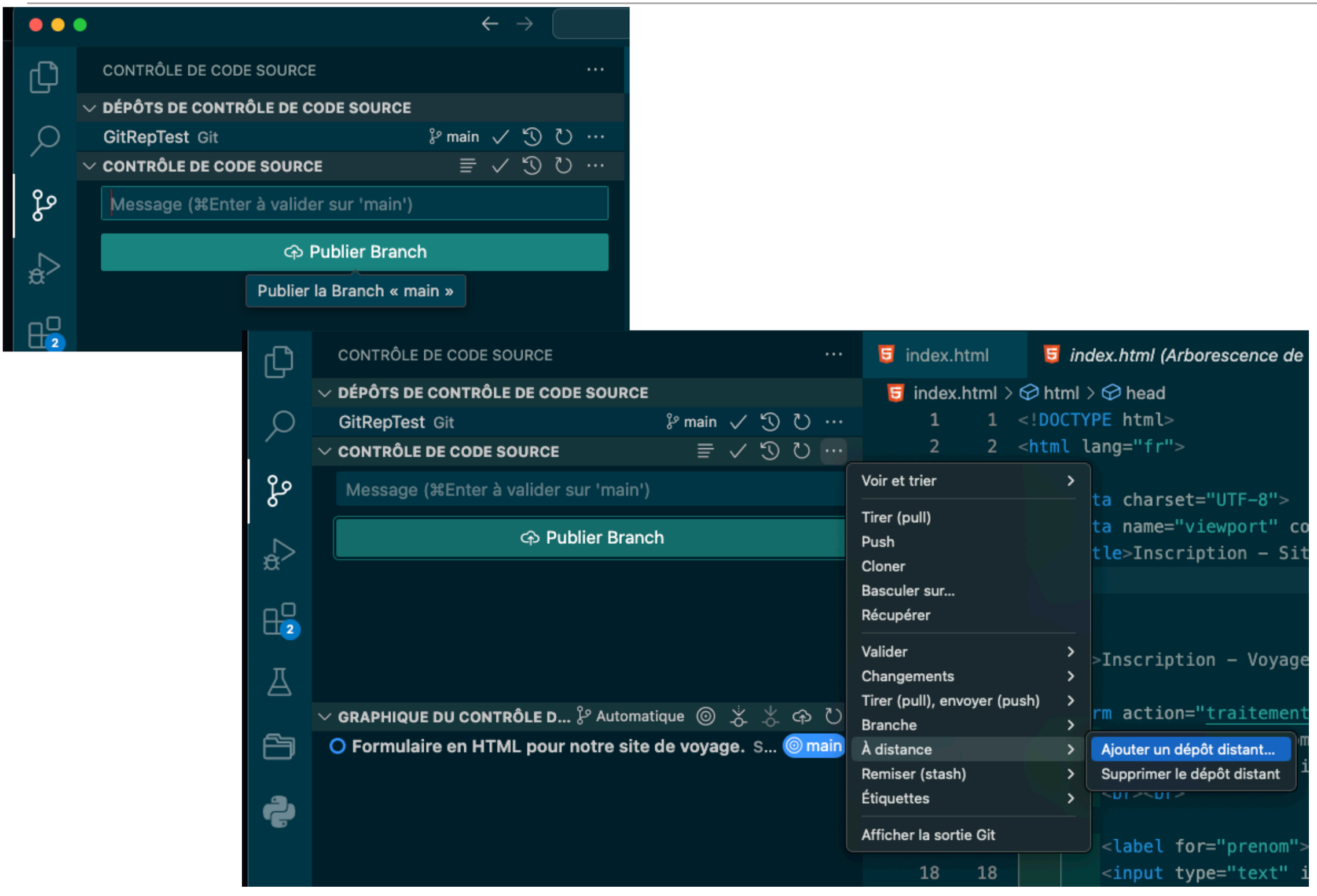
- **Intégrer les modifications dans `main`** une fois validées.
- Finaliser la mise à jour du code.

L'étape 8 :Mise à jour du dépôt local avec `git pull`

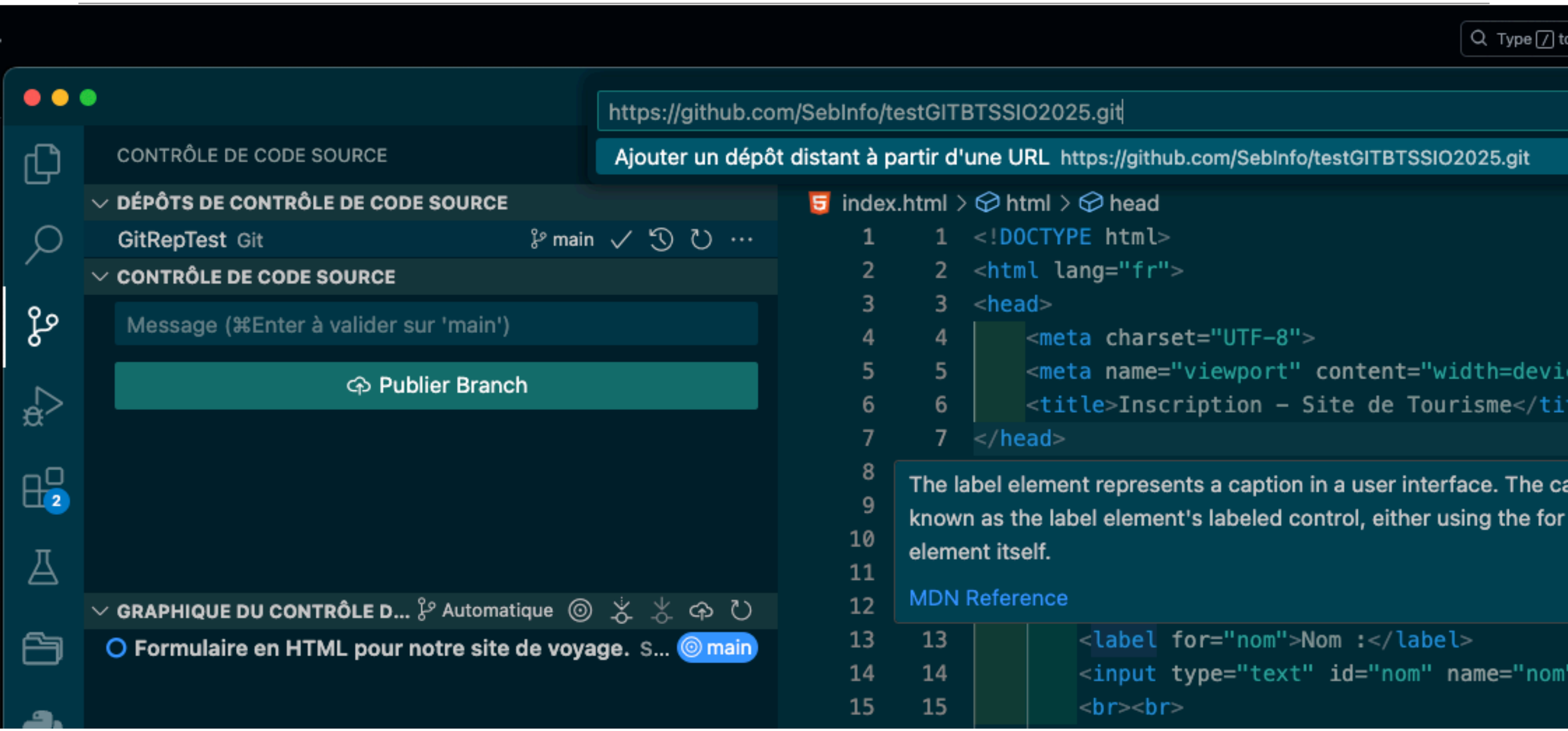
📌 **Rôle** :

- **Synchroniser le dépôt local avec les dernières modifications de `main`.**
- Éviter les conflits lors du prochain développement.

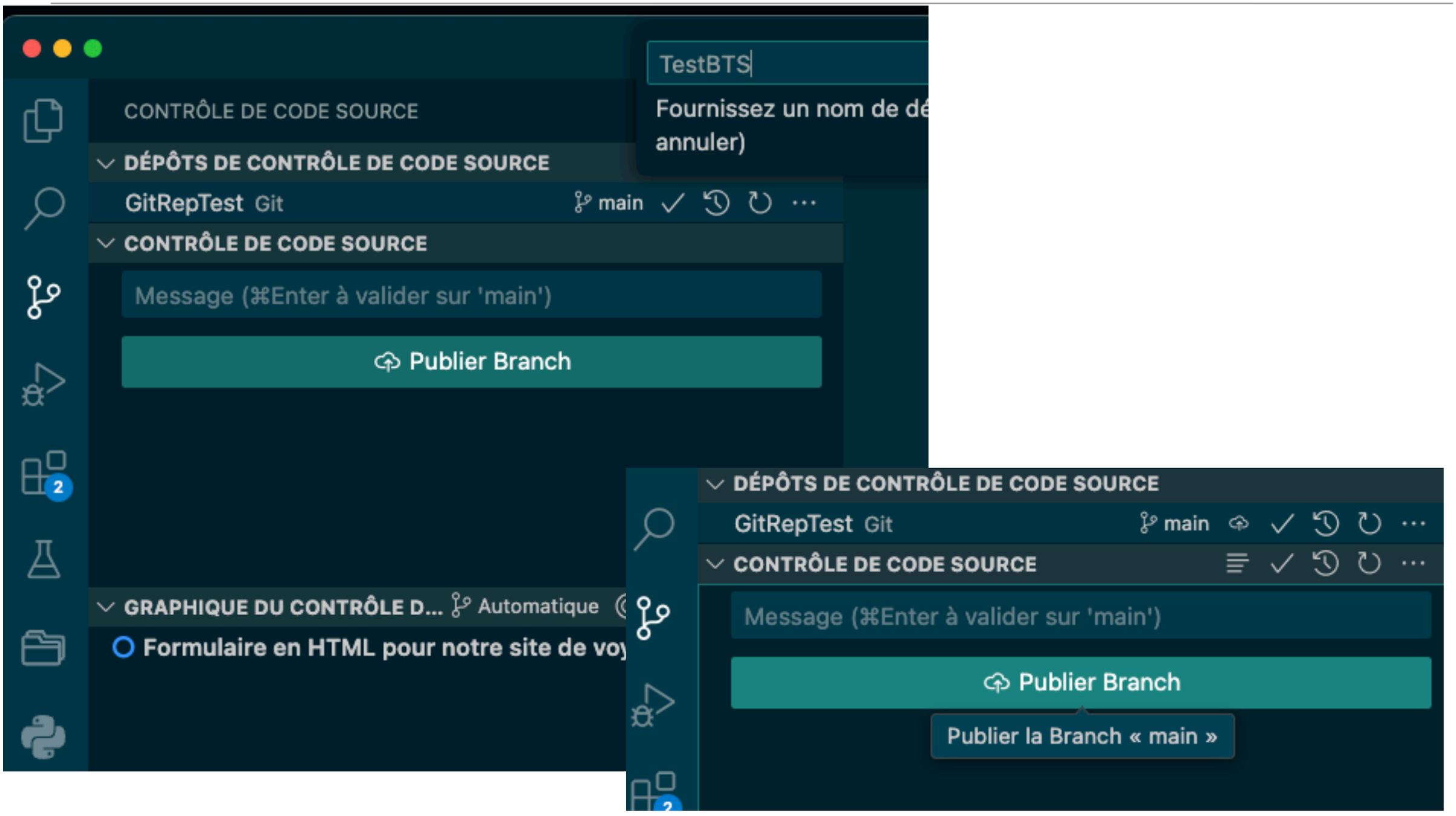
Ajouter un dépôt distant



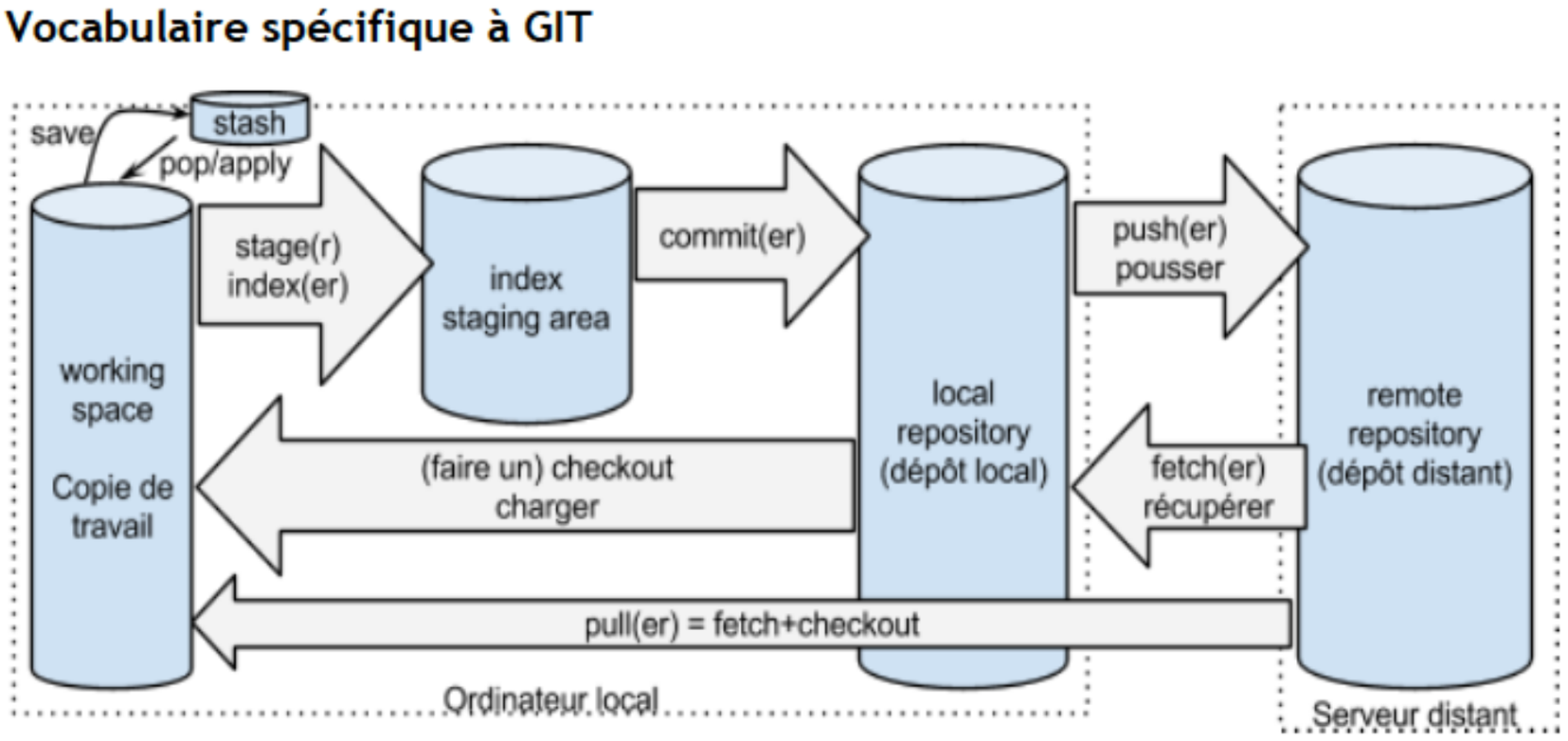
On envoie la branche main sur le dépôt



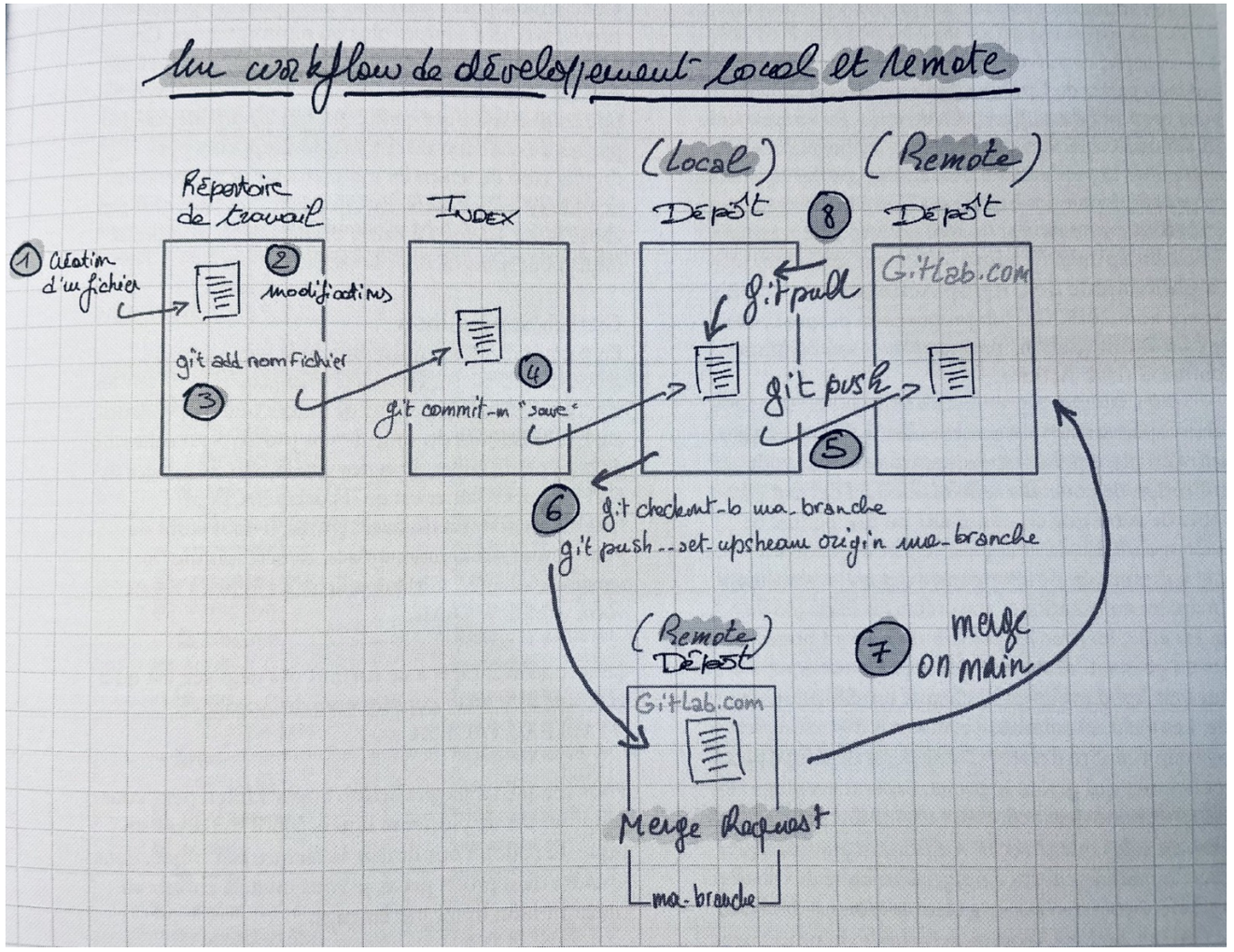
On envoie la branche main sur le dépôt



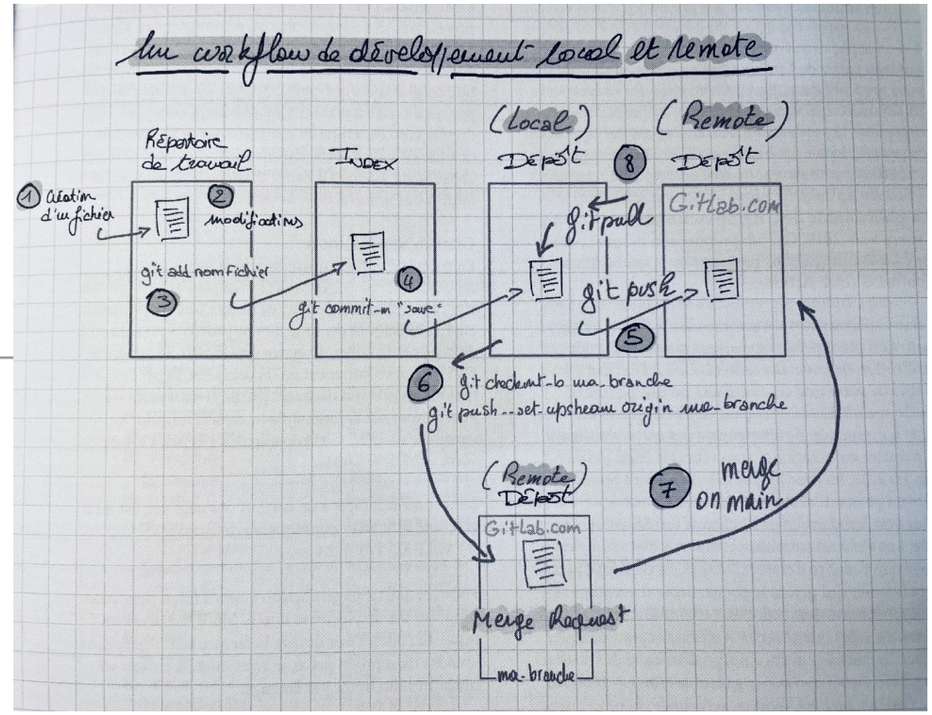
On envoie la branche avec le push

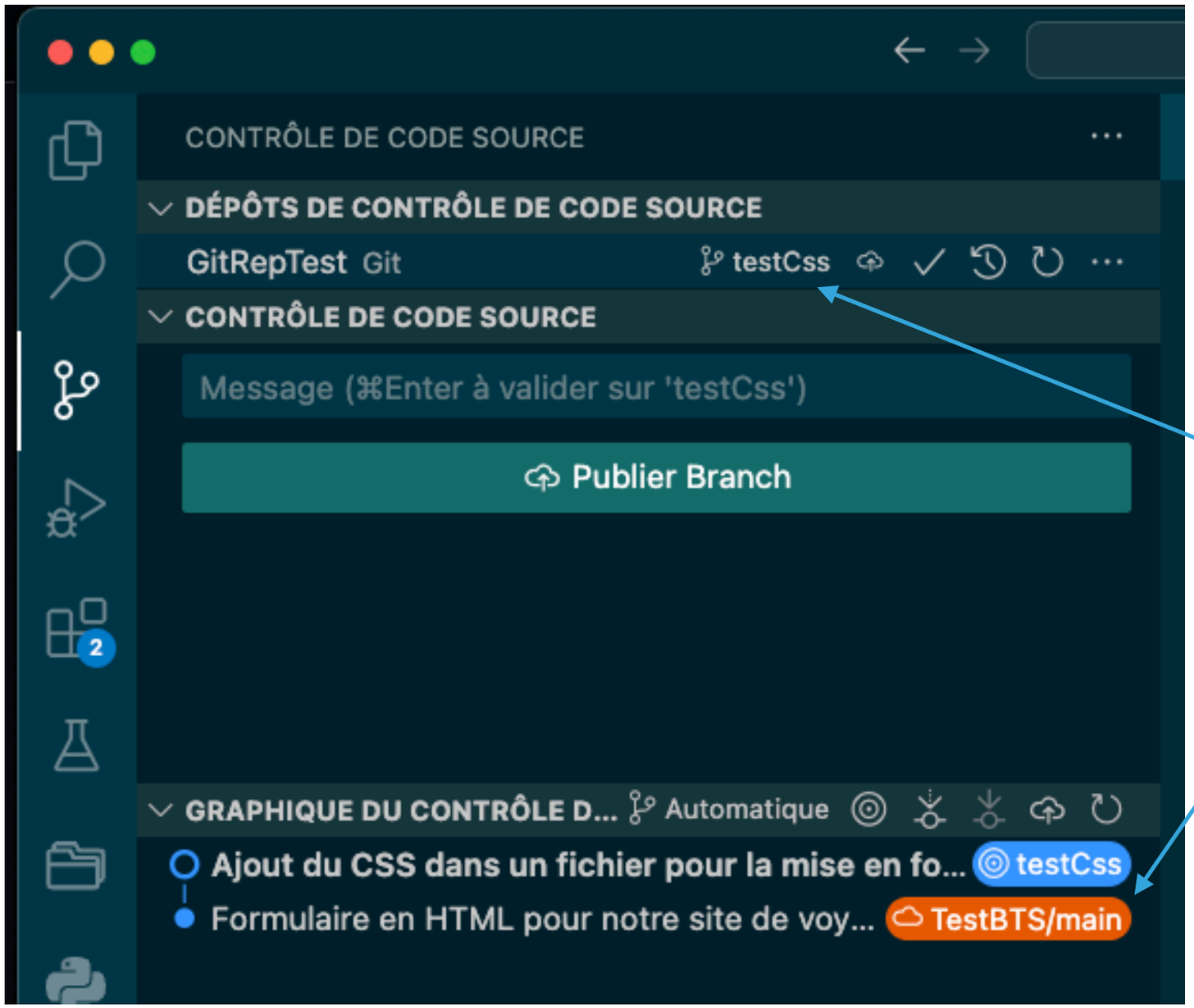


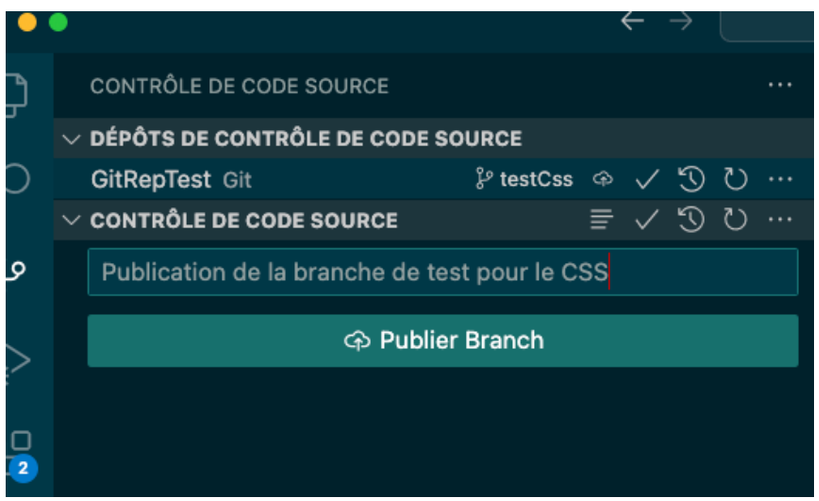
On va maintenant envoyer la branche testCss

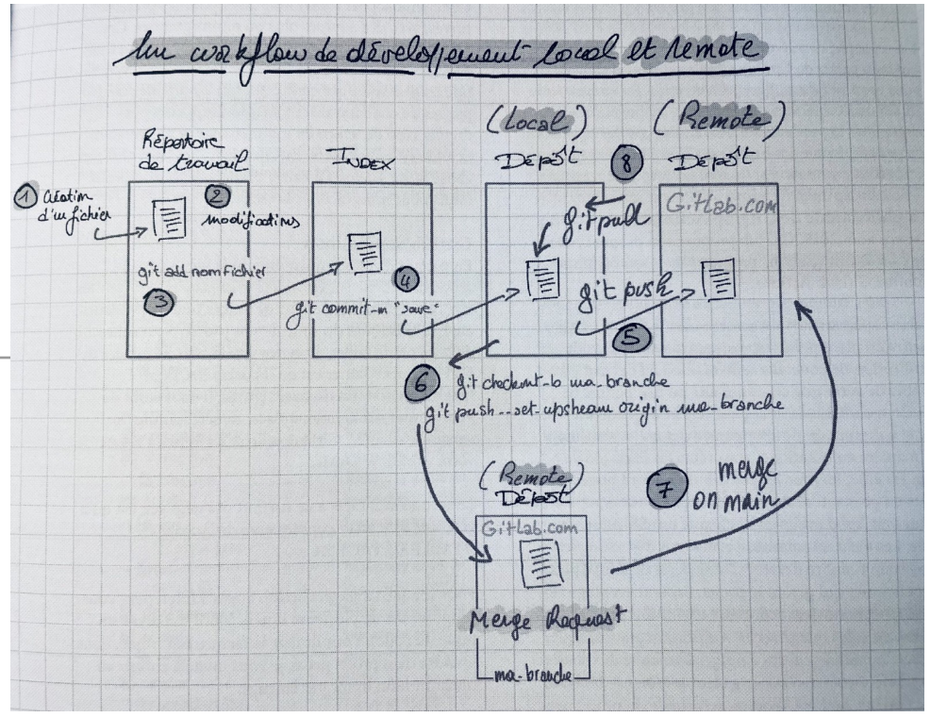


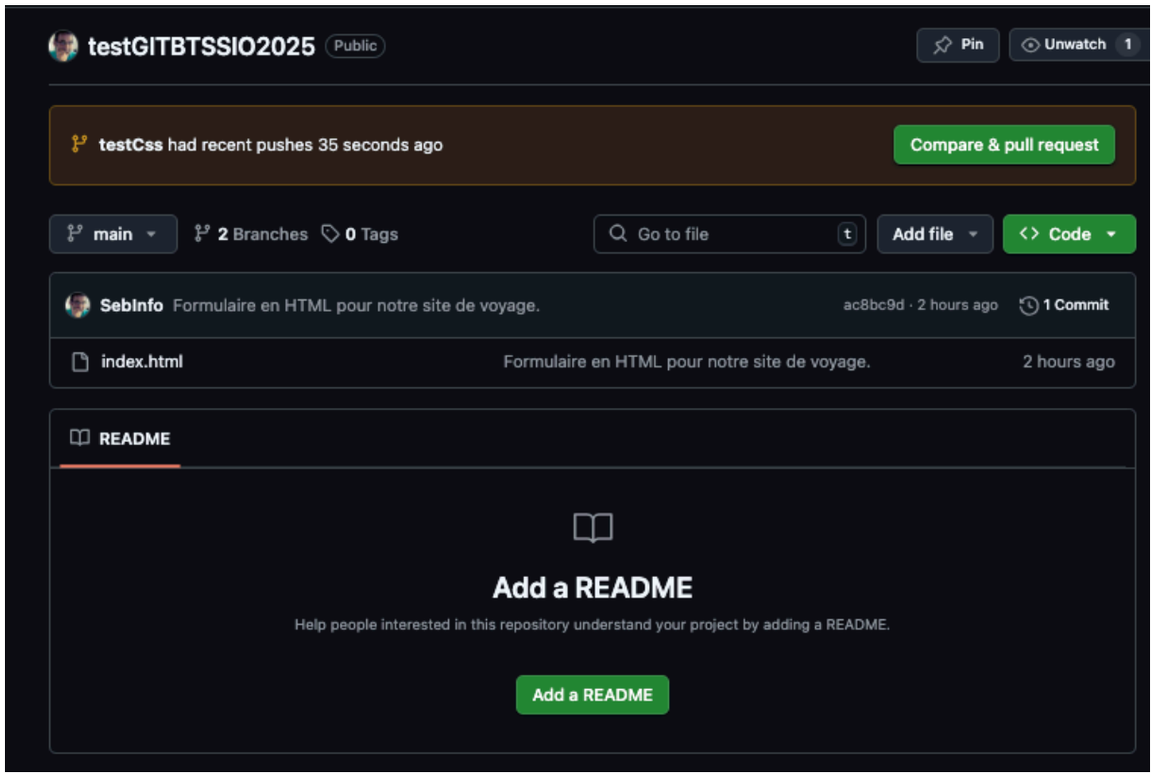
On va repartir de l'étape 5 avec notre branche testCss que nous allons pusher aussi !


Un workflow de développement local et remote
Répertoire de travail
Index
(Local) Dépôt
(Remote) Dépôt
GitLab.com
1 Création d'un fichier
2 modifications
git add nomFichier
3
git commit -m "save"
4
git pull
git push
5
8
6 git checkout -b ma_branche
git push --set-upstream origin ma_branche
(Remote) Dépôt
GitLab.com
Merge Request
ma_branche
7 merge on main


CONTRÔLE DE CODE SOURCE
DÉPÔTS DE CONTRÔLE DE CODE SOURCE
GitRepTest Git
testCss
CONTRÔLE DE CODE SOURCE
Message (⌘Enter à valider sur 'testCss')
Publier Branch
GRAPHIQUE DU CONTRÔLE D...
Automatique
Ajout du CSS dans un fichier pour la mise en fo...
testCss
Formulaire en HTML pour notre site de voy...
TestBTS/main


CONTRÔLE DE CODE SOURCE
DÉPÔTS DE CONTRÔLE DE CODE SOURCE
GitRepTest Git
testCss
CONTRÔLE DE CODE SOURCE
Publication de la branche de test pour le CSS
Publier Branch

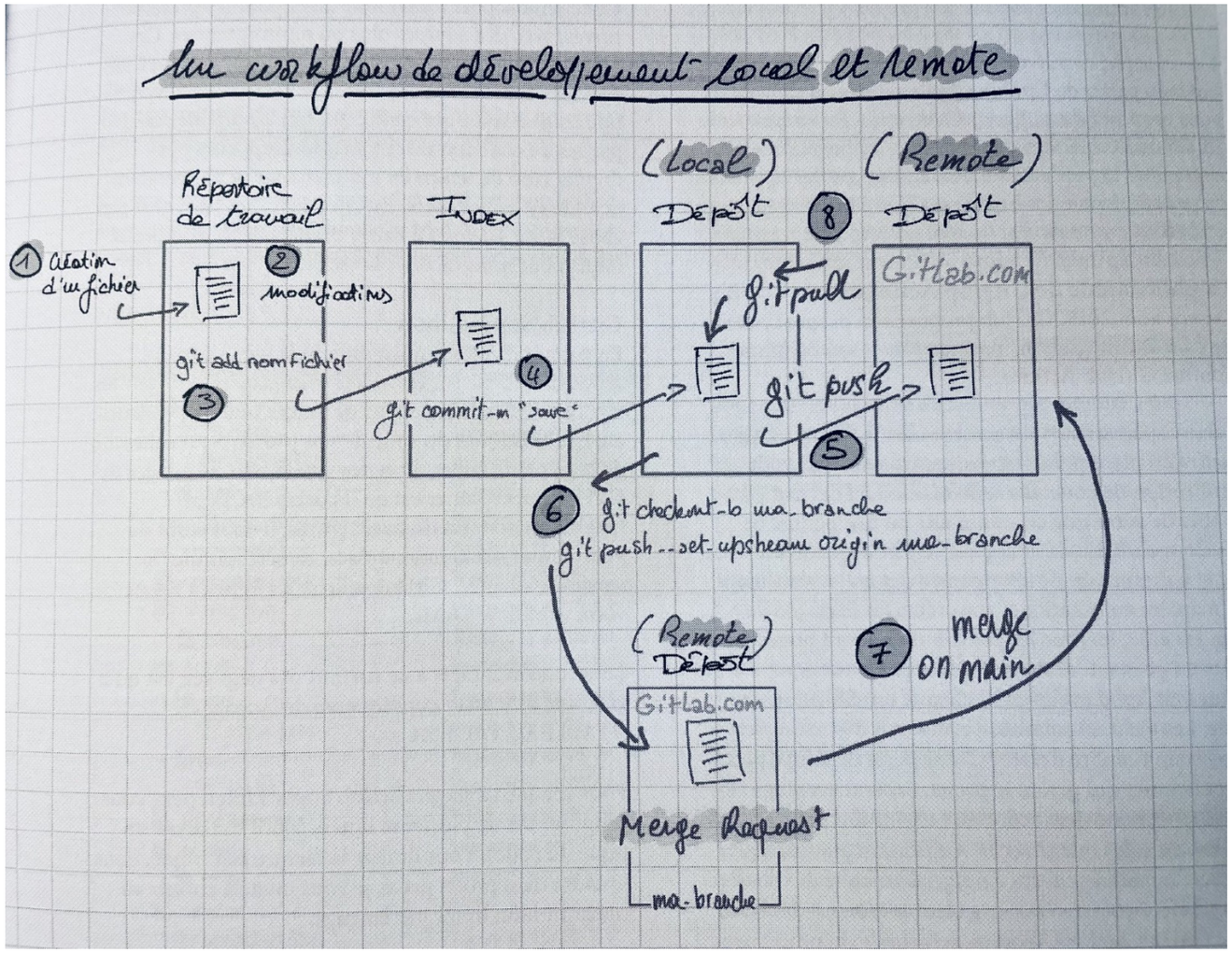




Les **Pull Requests (PR)** et **Merge Requests (MR)** sont deux concepts **quasiment identiques** utilisés dans Git pour **demander la fusion d'une branche dans une autre**, généralement de **feature-branch → main**.

- 🔷 **Pull Request (PR)** → Terme utilisé sur **GitHub** et **Bitbucket**.
- 🔷 **Merge Request (MR)** → Terme utilisé sur **GitLab**.

On passe à l'étape 6 !

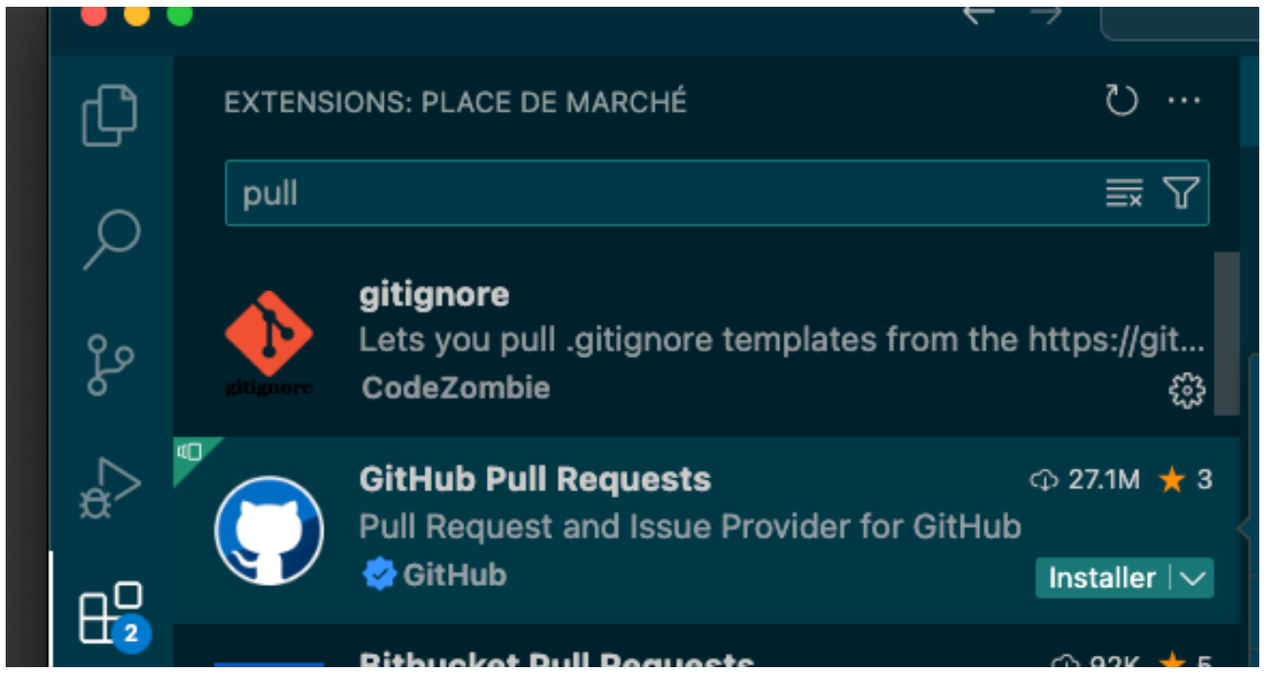


L'étape suivante avec le Merge Request ou Pull Request puisqu'on est sur Github !

On passe à l’étape 6 !

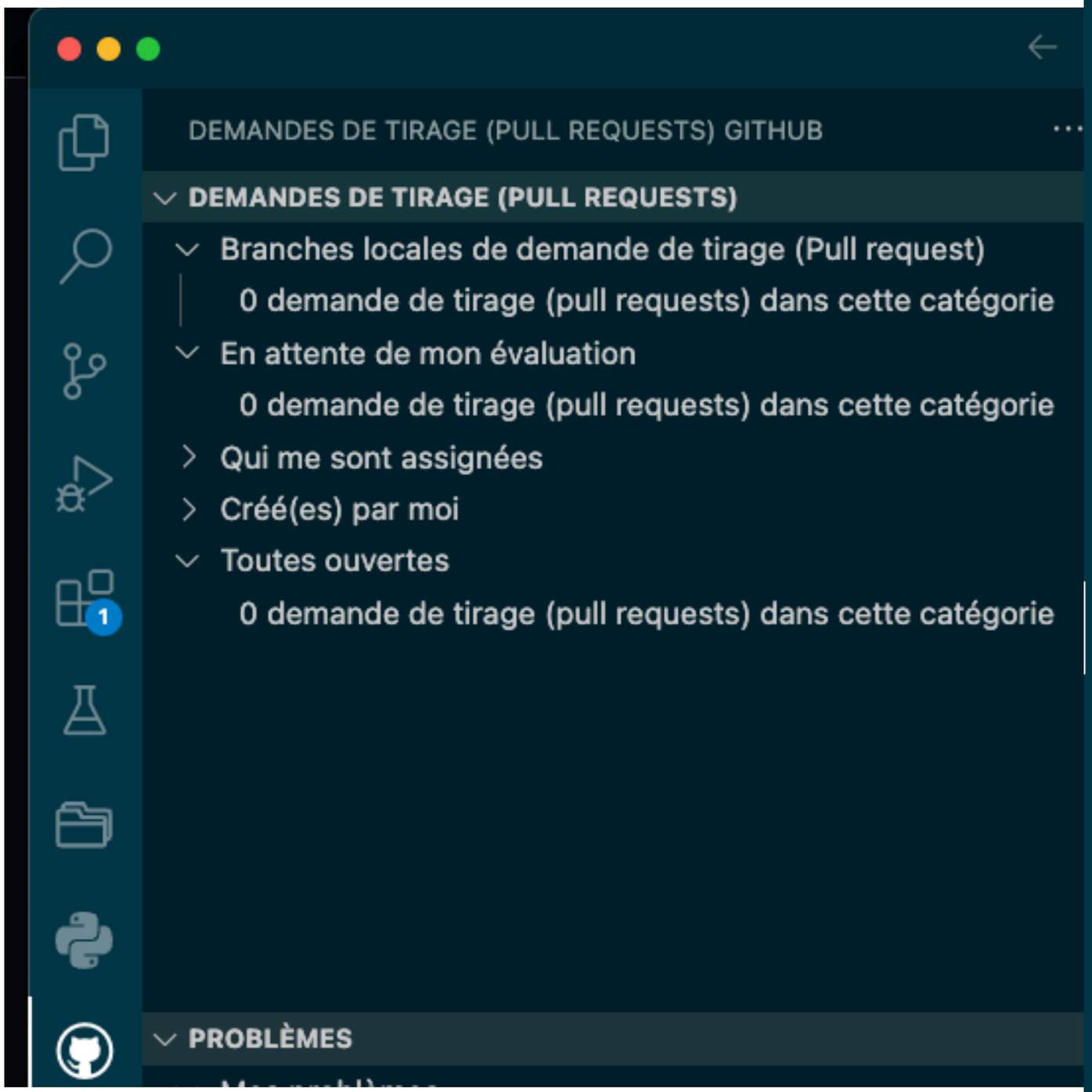
L’étape 6 : On peut le faire depuis Github. Mais si on veut le faire depuis VSC c’est possible !

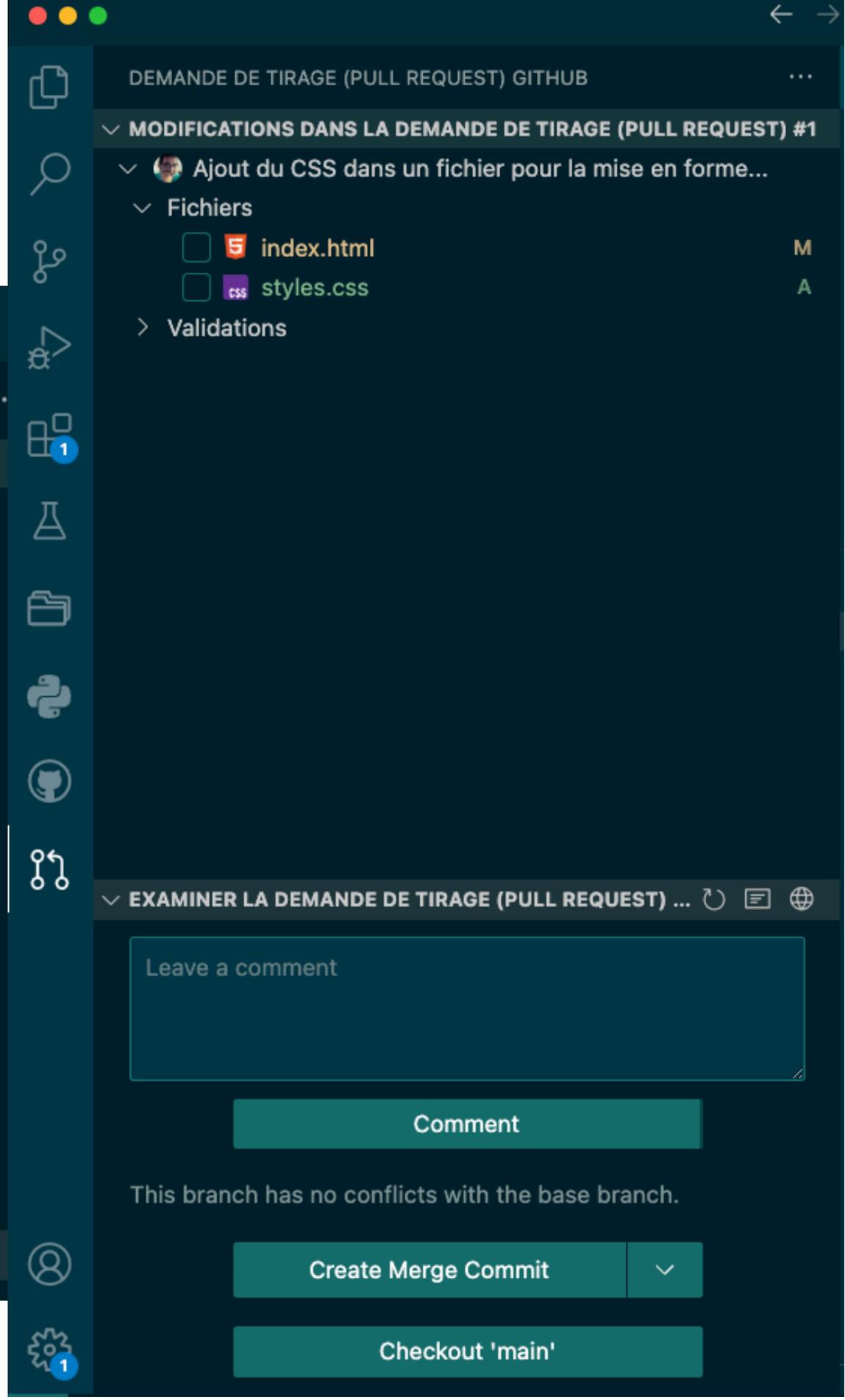
Il faut installer l'extension "GitHub Pull Requests and Issues"



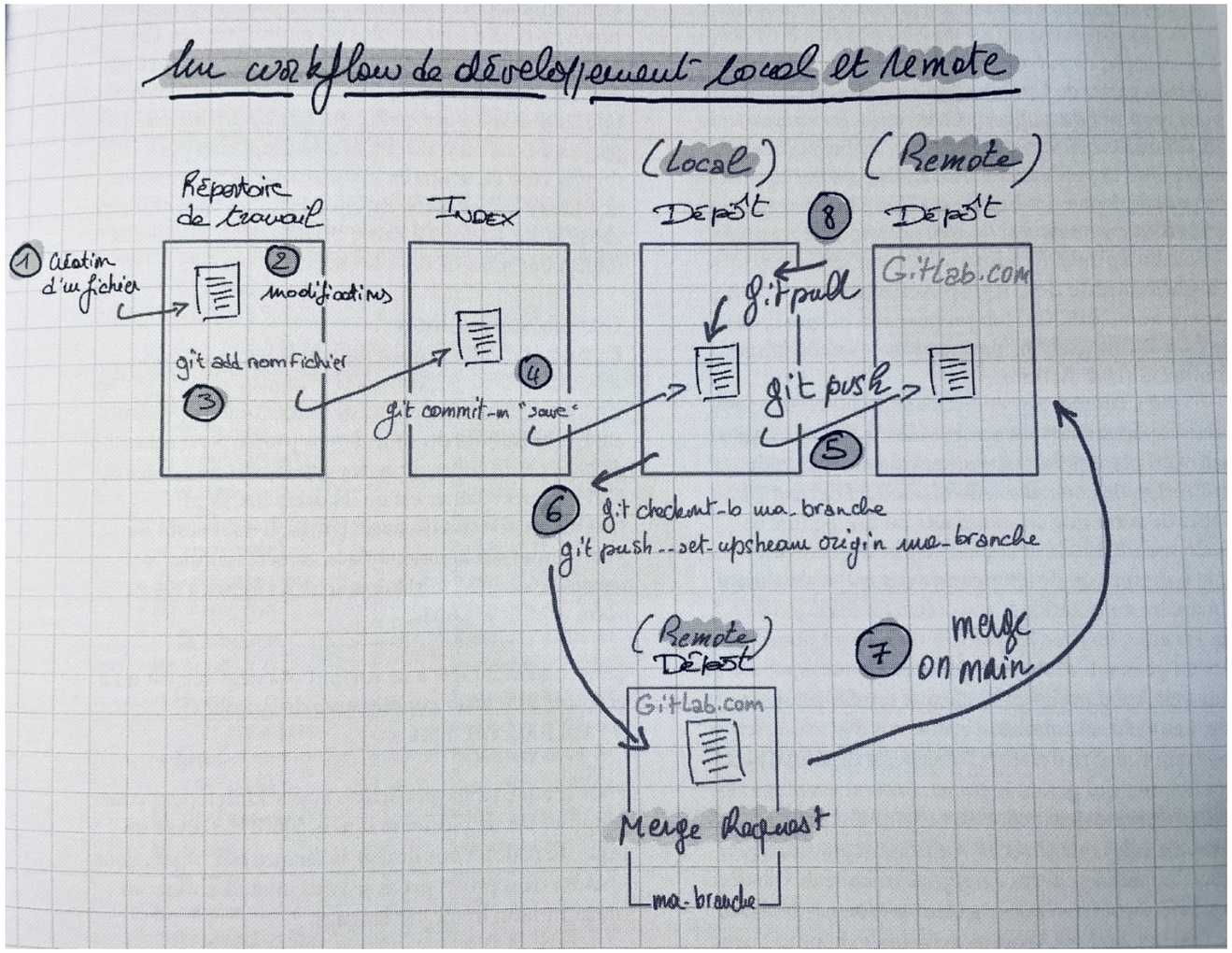
J’ai du renommer mon host pour la compatibilité
git remote rename TestBTS origin

On passe à l’étape 6 !

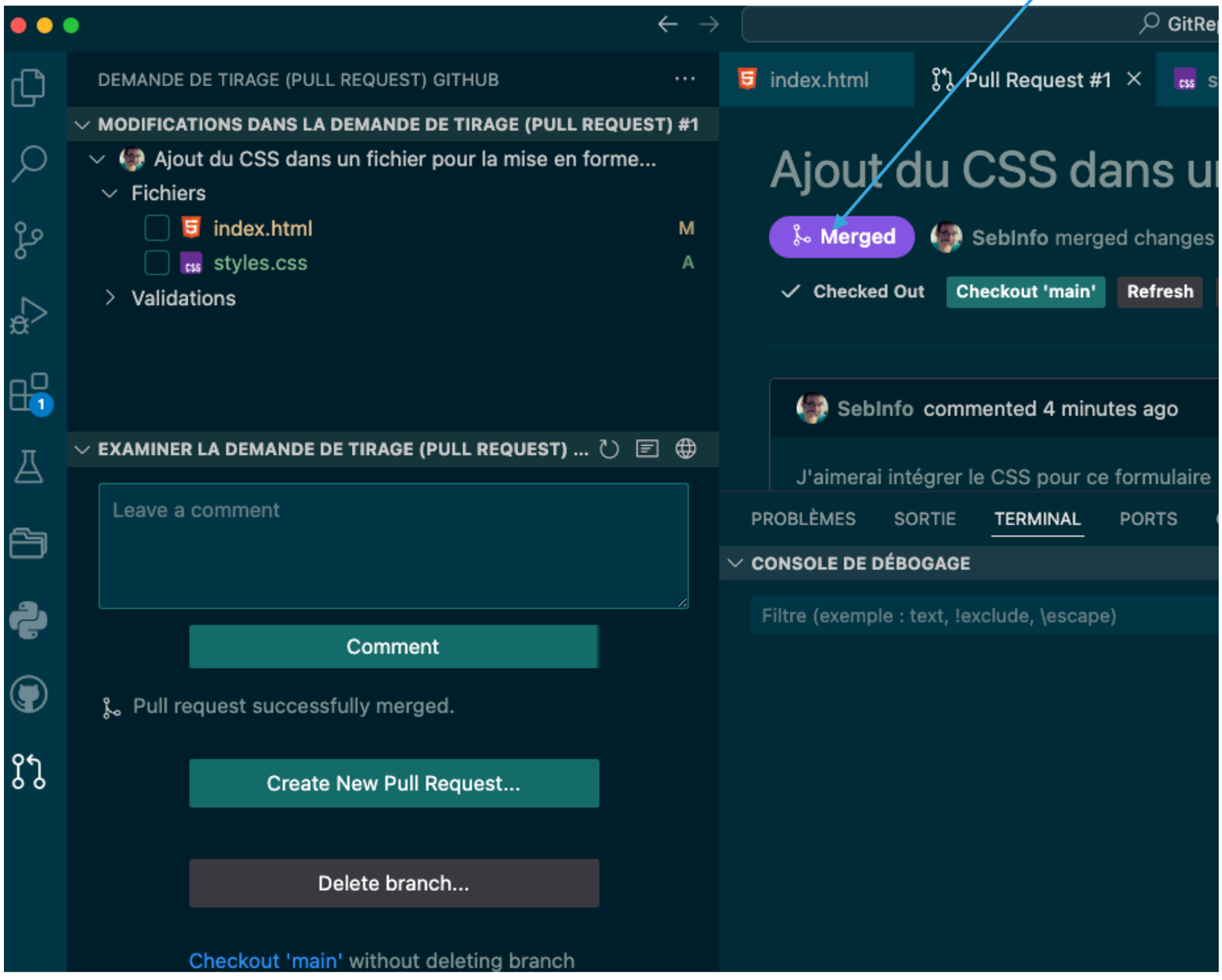




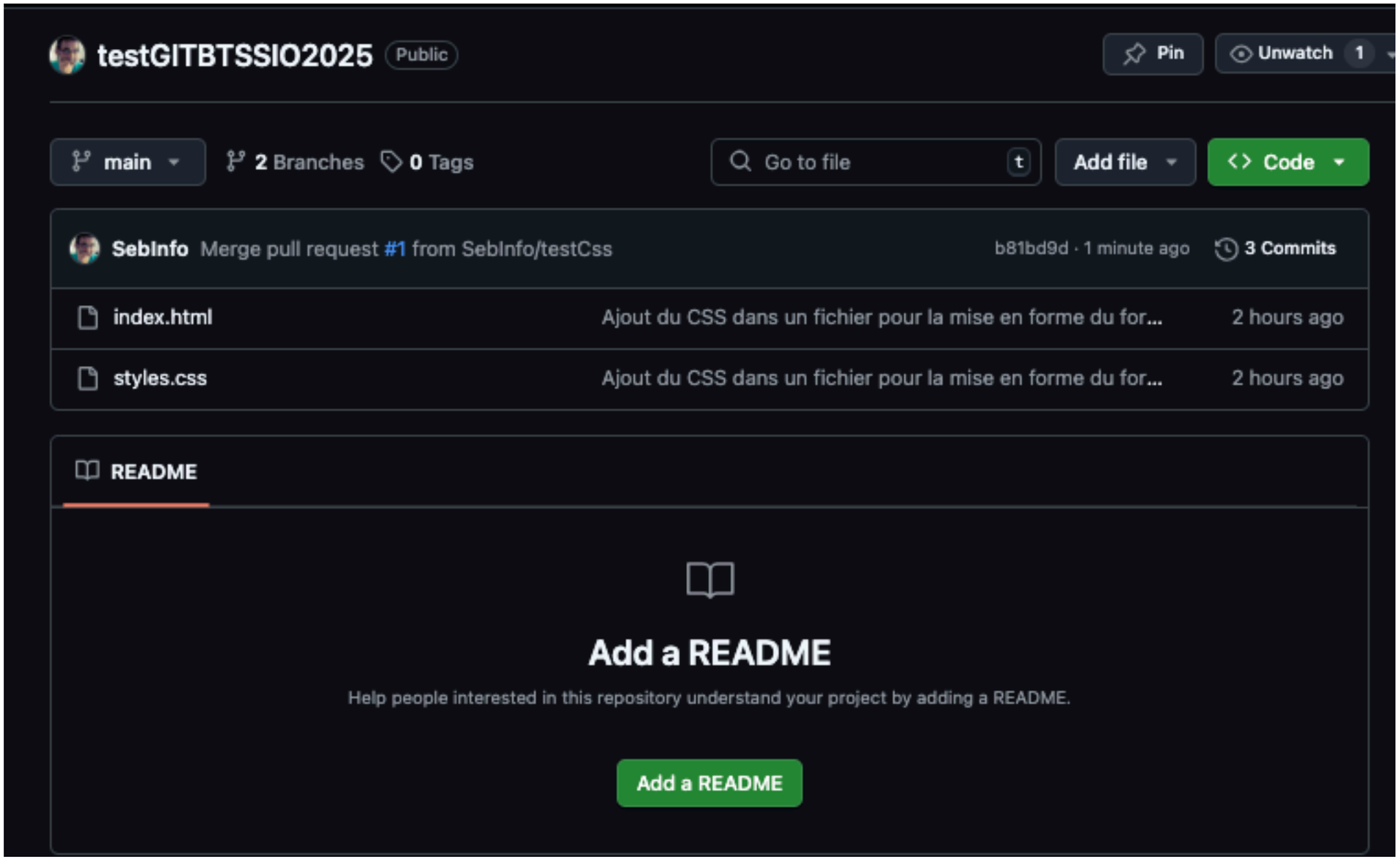
On passe à l'étape 7 !



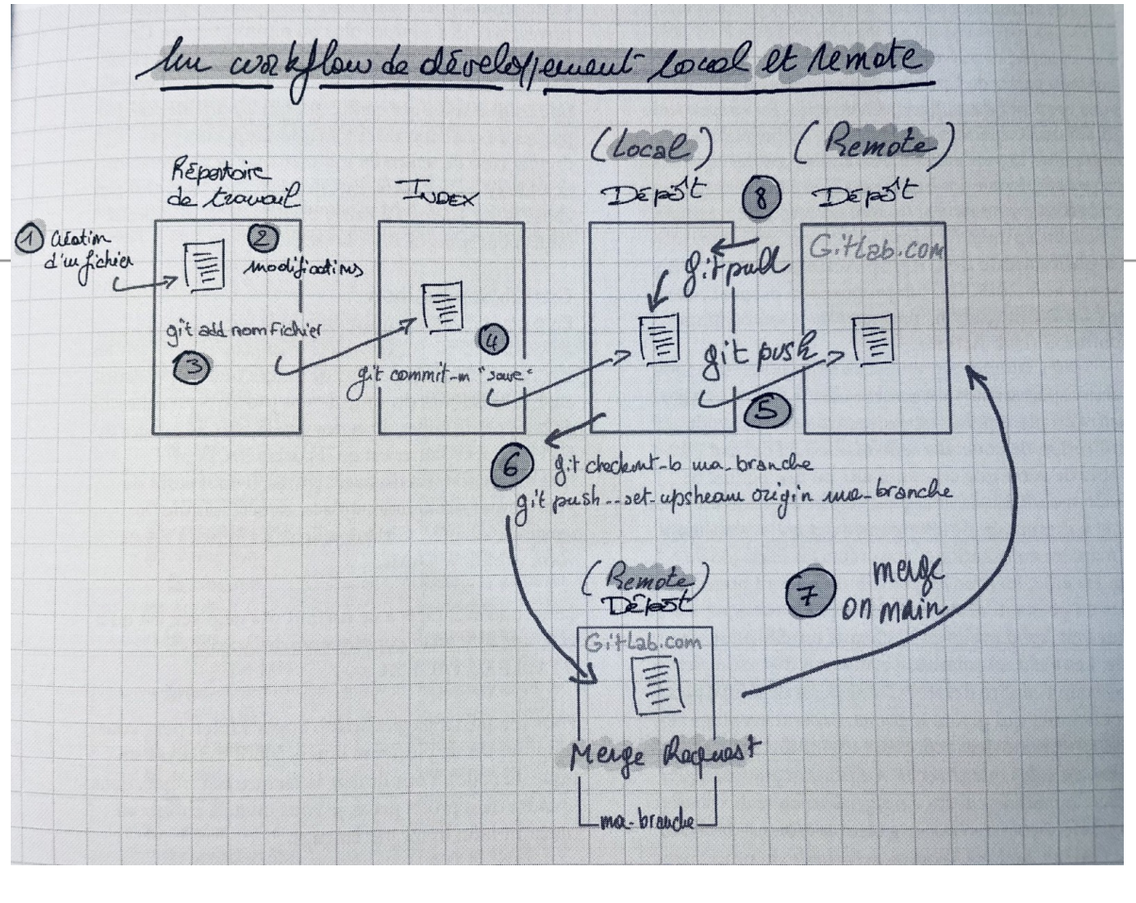
On passe à l’étape 6 !



On a bien une fusion puisque ici main contient le css



On passe à l'étape 8

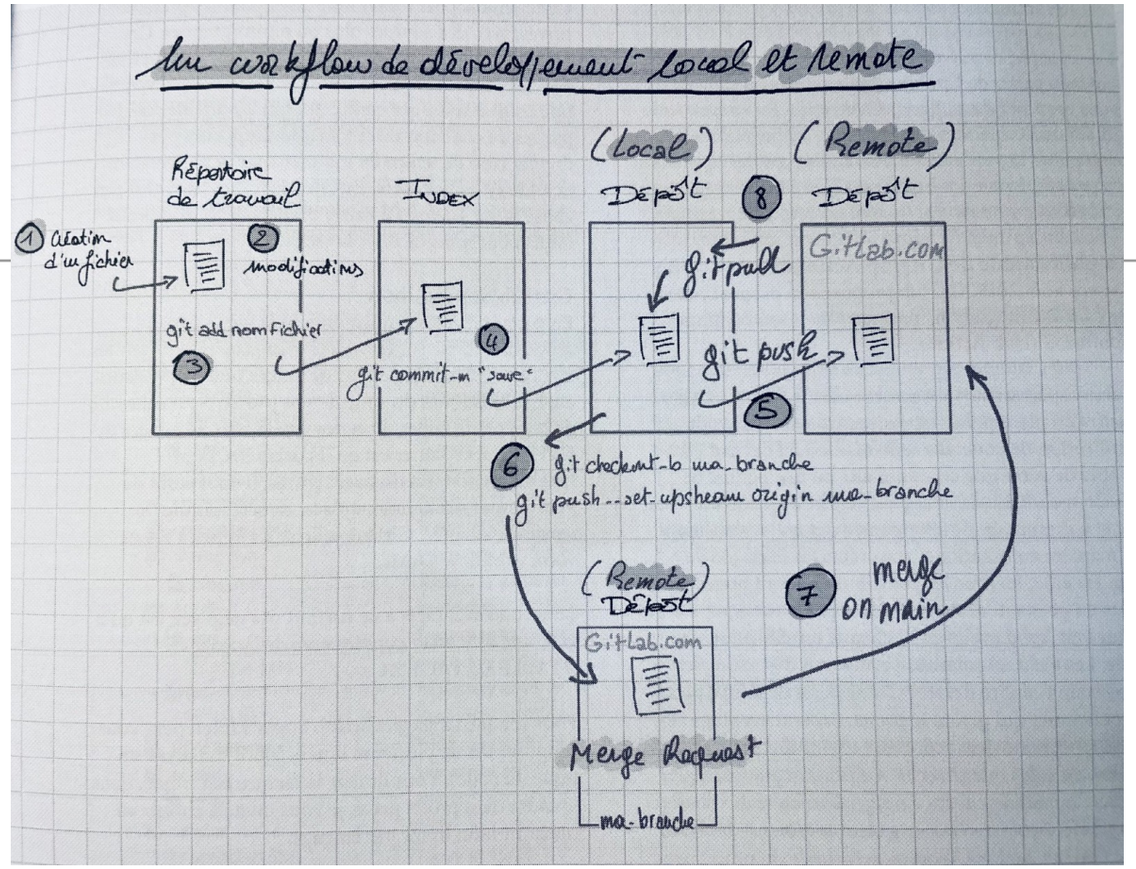


Etape 8 : Permet de récupérer les dernières mises à jour du dépôt distant (`main`) vers ton dépôt local.Important cette étape 8 car sinon le main local et distant sera pas le même.

Pourquoi `git pull` est important ?

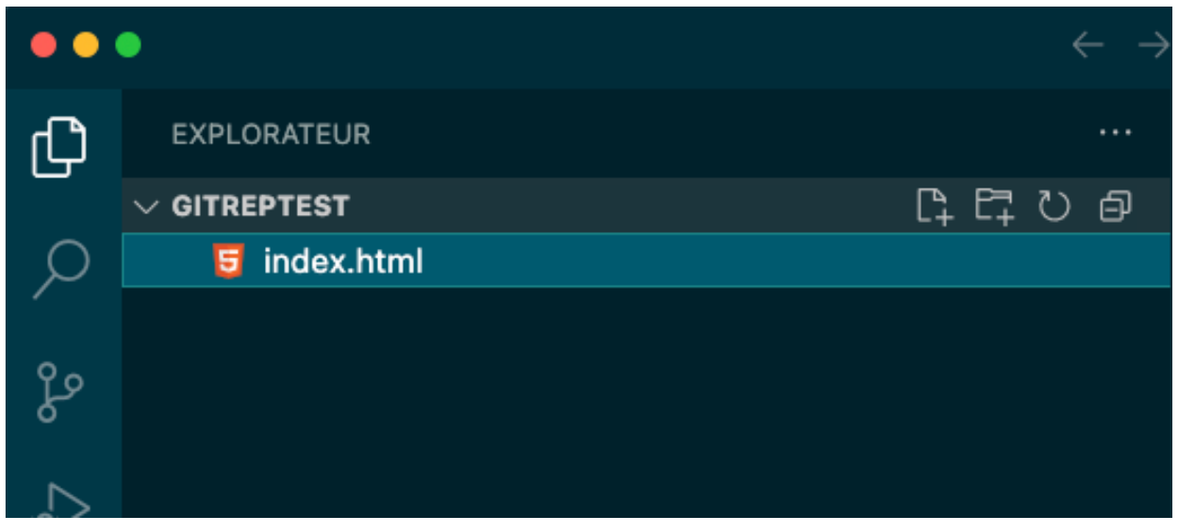
✅ **Synchronise ton code local avec GitHub**.
✅ **Évite les conflits** en s'assurant que tu as bien la dernière version de `main`.
✅ **Permet de travailler sur du code à jour** avant d'ajouter de nouvelles modifications.

On passe à l'étape 8



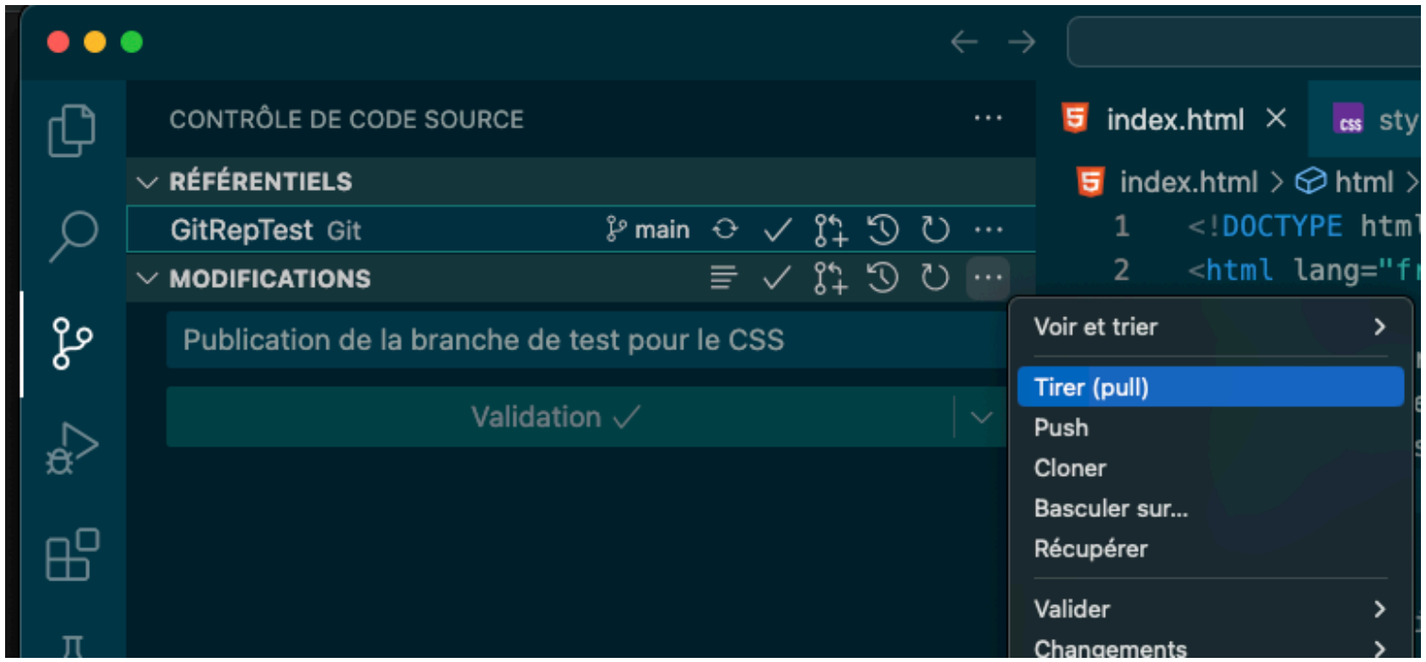
Sur VSC placer vous sur la branche main

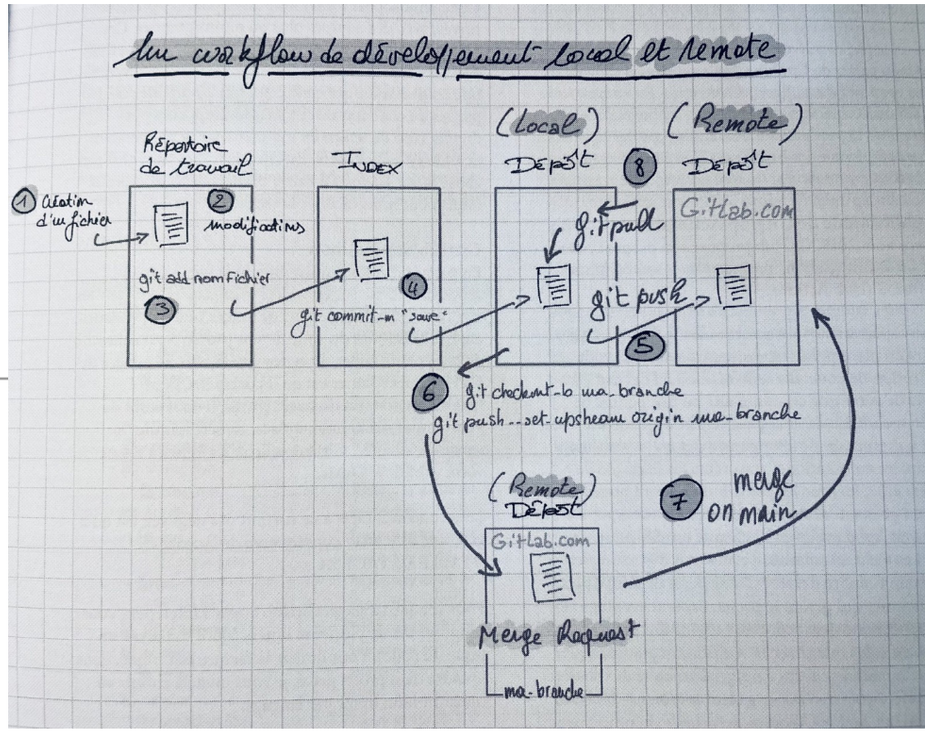
Pas encore le CSS

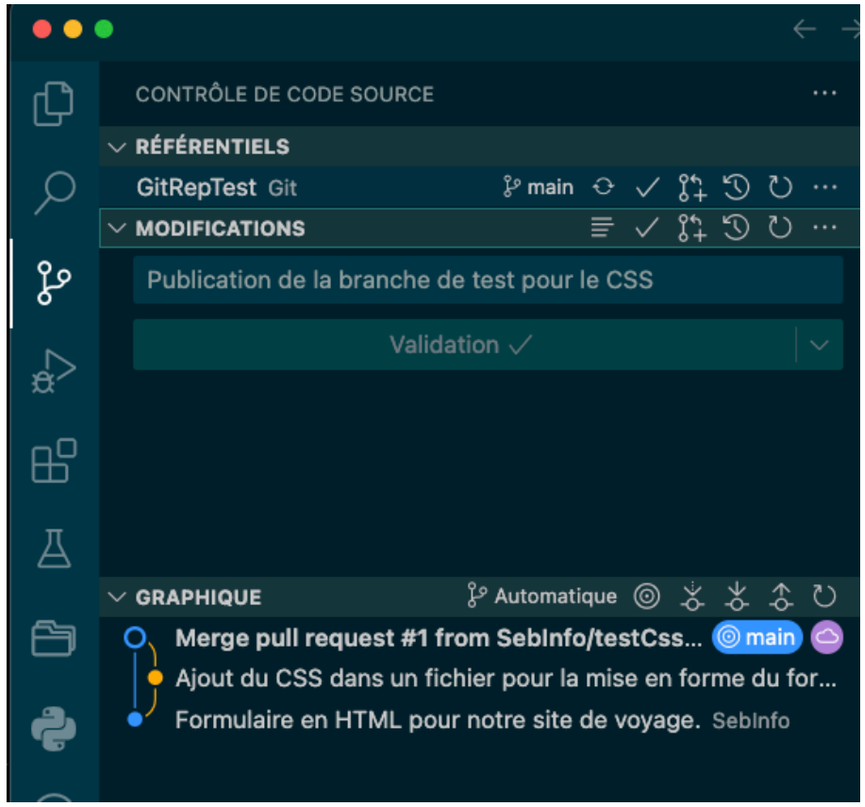


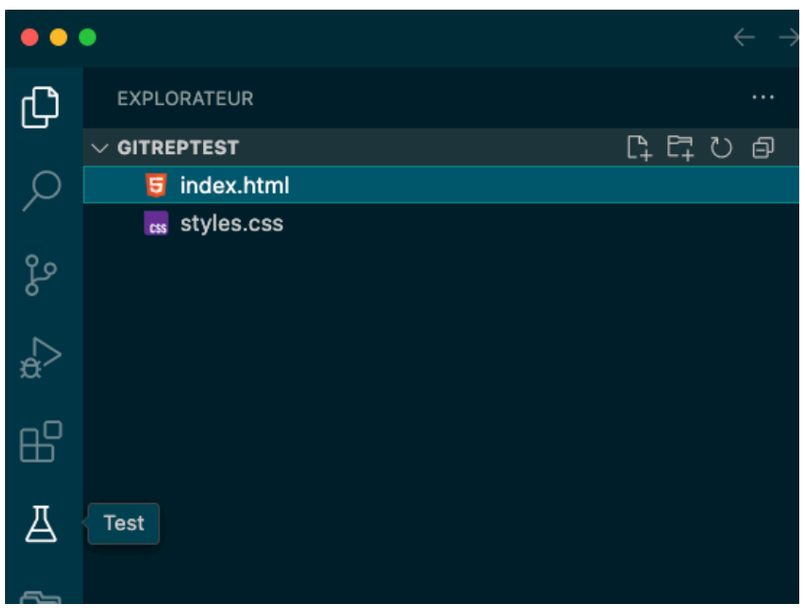
On passe à l'étape 8

Puis on fait le git pull









Git et les bonnes pratiques

- Commiter des modifications en relation les unes des autres
  -> deux aspects = deux commit
- Tester le code avant de comiter
- Utiliser les branches
- Commiter souvent
- Ecrire de bonne description dans les commit
- Ne pas commiter un travail à moitié fait
- Git n’est pas un système de Sauvegarde/Backup

MERCI POUR VOTRE ATTENTION