




# Guide d'utilisation de Git

## WIZARDS & DICE

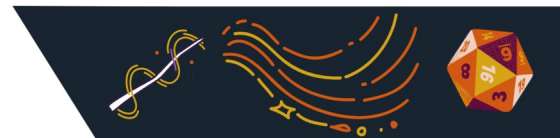
 Créateur : Hugo CLAMOND – Responsable infrastructure et système

 Date de Création : 02/12/2024

 Dernier modificateur : Axel MOURILLON – Chef de projet

 Date de modification : 06/01/2025

 Version : 2.0



# Table des matières

<b>Table des figures.....</b>	<b>2</b>
<b>I. Introduction.....</b>	<b>3</b>
<b>II. Première mise en place et suivi des fichiers.....</b>	<b>4</b>
A) Cloner le repo.....	4
B) Configurer le repo.....	4
C) Ajouter un fichier au suivi.....	4
1/ Dans SourceGit.....	4
2/ Avec l'outil CLI.....	5
D) Créer un commit.....	6
1/ Dans SourceGit.....	6
2/ Avec l'outil CLI.....	6
E) Supprimer un fichier suivi par Git en le conservant localement.....	6
1/ Avec l'outil CLI (uniquement).....	6
<b>III. Travailler avec l'historique.....</b>	<b>7</b>
A) Visualiser l'historique.....	7
1/ Dans SourceGit.....	7
2/ Avec l'outil CLI.....	7
B) Afficher les détails d'un commit spécifique.....	8
1/ Dans SourceGit.....	8
2/ Avec l'outil CLI.....	8
<b>IV. Branches.....</b>	<b>9</b>
A) Créer une nouvelle branche.....	9
1/ Dans SourceGit.....	9
2/ Avec l'outil CLI.....	10
B) Lister les branches.....	10
1/ Dans SourceGit.....	10
2/ Avec l'outil CLI.....	10
C) Changer de branche active.....	10
1/ Dans SourceGit.....	10
2/ Avec l'outil CLI.....	11
D) Fusionner une branche souhaitée dans la branche active.....	11
1/ Dans SourceGit.....	11
2/ Avec l'outil CLI.....	11
E) Récupérer un commit spécifique d'une branche et le copier sur la branche active.....	11
1/ Dans SourceGit.....	12
2/ Avec l'outil CLI.....	12
F) Résoudre des commits de fusion post-merge ou pull.....	12
1/ Dans SourceGit.....	13
2/ Avec l'outil CLI.....	15
<b>V. Travailler avec un repo distant.....</b>	<b>16</b>
A) Tirer les changements du repo distant (pull).....	16
1/ Dans SourceGit.....	16
2/ Avec l'outil CLI.....	17
B) Récupérer l'historique de commits de toutes branches sans les télécharger (fetch).....	17

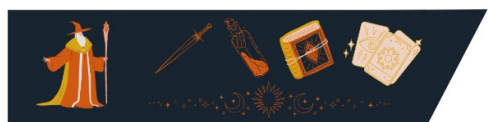




1/ Dans SourceGit.....	17
2/ Avec l'outil CLI.....	17
C) Pousser les changements sur le repo distant (push).....	18
1/ Dans SourceGit.....	18
2/ Avec l'outil CLI.....	18
<b>VI. Workow général et bonnes pratiques.....</b>	<b>19</b>

## Table des figures

Figure 1 : Étape 1 de l'ajout d'un fichier au suivi.....	5
Figure 2 : Étape 2 de l'ajout d'un fichier au suivi.....	5
Figure 3 : Création d'un commit.....	6
Figure 4 : Ajout d'une entrée à l'historique de commit.....	6
Figure 5 : Historique des commit affiché en graph via le CLI.....	7
Figure 6 : Détails d'un commit.....	8
Figure 7 : Créer une nouvelle branche.....	9
Figure 8 : Branches locales et distantes du repo.....	10
Figure 9 : Exemple de "cherry-pick".....	12
Figure 10 : Fusion post-merge SourceGit – contexte.....	13
Figure 11 : Fusion post-merge SourceGit – conflit.....	13
Figure 12 : Fusion post-merge SourceGit – outil de fusion externe.....	13
Figure 13 : Fusion post-merge SourceGit – résolution de conflit.....	14
Figure 14 : Fusion post-merge SourceGit – confirmation des changements.....	14
Figure 15 : Fusion post-merge SourceGit – finalisation.....	15
Figure 16 : Fusion post-merge SourceGit – historique.....	15
Figure 17 : Fusion post-merge CLI – message d'erreur de fusion.....	15
Figure 18 : Fusion post-merge CLI – historique après le merge.....	15
Figure 19 : Tirer les changements du repo distant.....	16
Figure 20 : Paramètre du "pull".....	16
Figure 21 : Récupérer uniquement l'historique des données.....	17
Figure 22 : Pousser des changements sur le repo distant.....	18





# I. Introduction

Cette fiche récapitulative a pour but de présenter l'essentiel des opérations que l'on sera amenés à faire durant le PTUT. Elles seront présentées avec l'outil GUI SourceGit, couplées à leur équivalent avec l'outil CLI.

Je conseille de travailler avec l'interface de SourceGit en anglais, car certains termes francisés des différents process Git rendent les choses plus complexes à retenir. De plus, si vous voulez aller chercher de la documentation sur le sujet, vous la trouverez le plus souvent en anglais (comme tout...).





## II. Première mise en place et suivi des fichiers

### Introduction de la section :

Cette section traite du déploiement, de la configuration du repo et du suivi des fichiers

- Cloner le repo : détaille comment cloner le repo
- Configurer le repo : détaille comment configurer le repo
- Ajouter un fichier au suivi : détaille comment ajouter un fichier au suivi
- Créer un commit : détaille comment enregistrer une étape de suivi
- Supprimer un fichier suivi par Git en le conservant localement : détaille comment supprimer un fichier du suivi

### A) Cloner le repo

Pour cloner le repo, je conseille d'utiliser l'outil CLI de git ; cela peut éviter les soucis étant donné que l'on clone avec une adresse SSH.

Pour cloner le repo, taper ceci :

```
git clone git@github.com:Greta-Ardeche-Drome/wizards-n-dices.git
```

### B) Configurer le repo

Configurer le repo cloné veut dire ici la spécification de l'utilisateur et de l'adresse e-mail poussés avec un commit.

```
git config user.name "Prénom NOM"  
git config user.email "adresse@mail.org"
```

### C) Ajouter un fichier au suivi

#### 1/ Dans SourceGit

Cliquer sur "Changes", vous basculez dans une nouvelle fenêtre où vous voyez la liste des changements (voir figure ci-après).



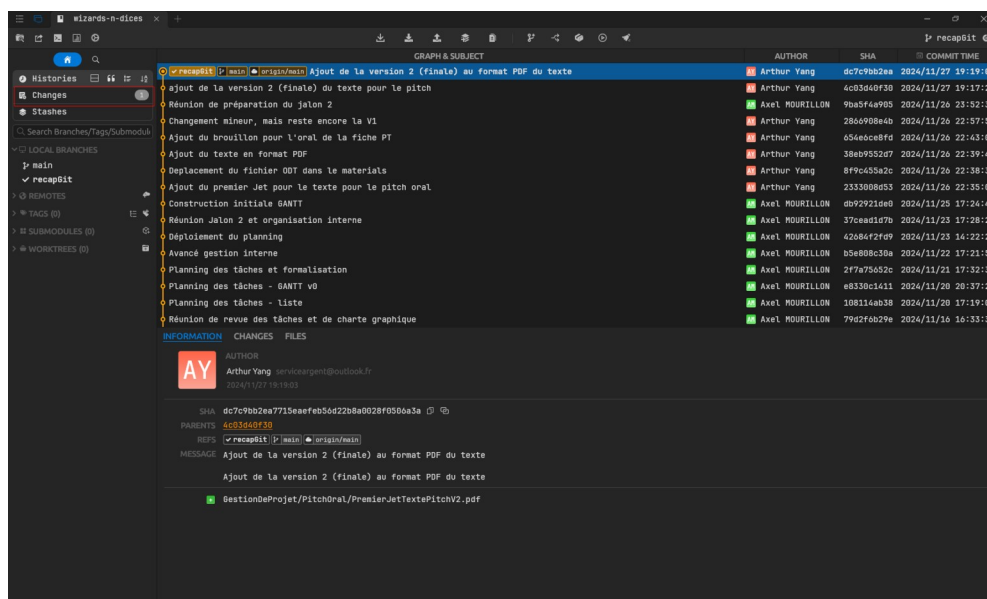
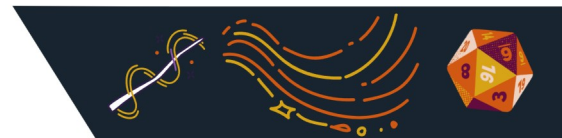


Figure 1 : Étape 1 de l'ajout d'un fichier au suivi

On ajoute les changements que l'on veut à l'index en cliquant dessus ou en cliquant sur l'icône mise en évidence ci-dessous, ce qui ajoute tous les changements non indexés à l'index. En anglais, l'index s'appelle le "stage".

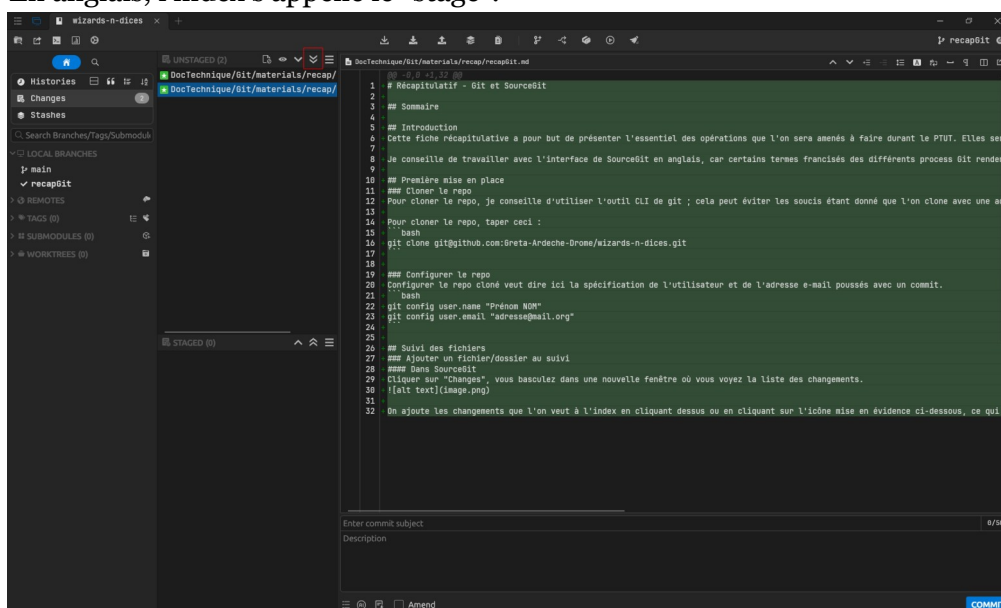


Figure 2 : Étape 2 de l'ajout d'un fichier au suivi

Rappelez-vous qu'on ne peut pas ajouter un dossier vide à l'index, il est obligatoire de créer un fichier dans le dossier en question pour qu'il apparaisse dans le repo distant.

## 2/ Avec l'outil CLI

Ajouter un certain fichier au suivi :

```
git add chemin/vers/le/fichier
```

Ajouter tous les changements non indexés à l'index :

```
git add .
```



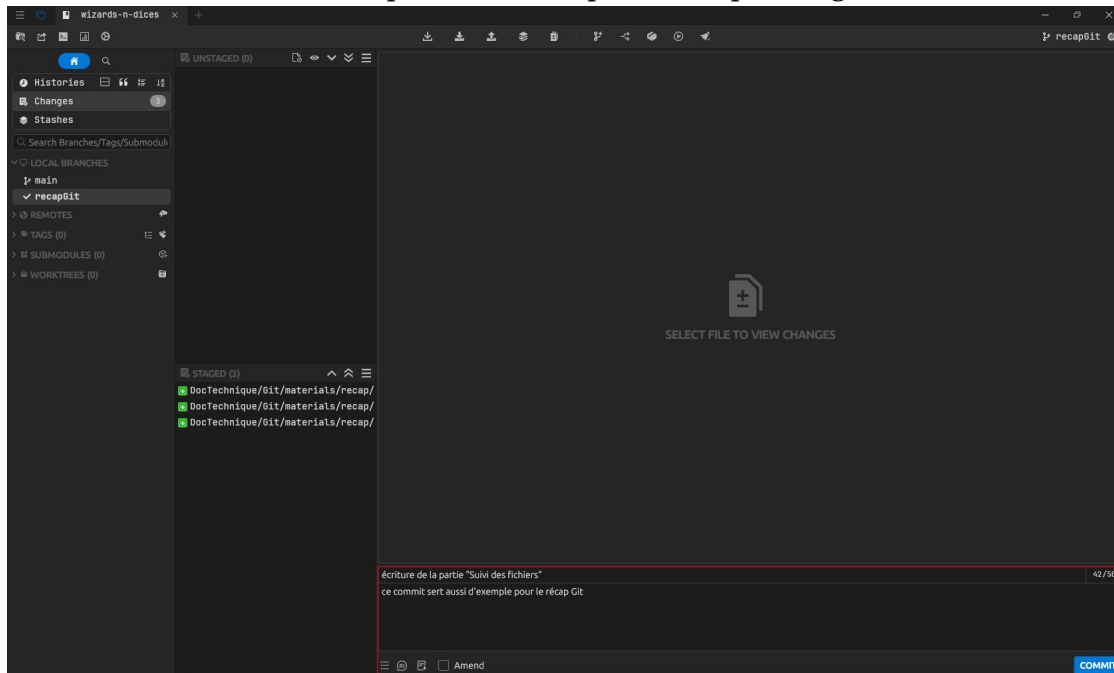




## D) Créer un commit

### 1/ Dans SourceGit

Une fois les changements indexés, on crée un commit pour sauvegarder leur état dans l'historique du repo. Dans la fenêtre "Changes", il y a un espace pour créer un message de commit avec titre et description. La description n'est pas obligatoire.



**Figure 3 : Création d'un commit**

Une fois le message écrit, il suffit de cliquer sur "Commit". En revenant dans la fenêtre "Histories", le commit apparaît dans l'historique.



**Figure 4 : Ajout d'une entrée à l'historique de commit**

### 2/ Avec l'outil CLI

Exécuter la commande suivante :

```
git add .
```

## E) Supprimer un fichier suivi par Git en le conservant localement

### 1/ Avec l'outil CLI (uniquement)

Par soucis de simplicité, ce genre d'opération sera à effectuer en mode CLI uniquement.

```
git rm --cached <chemin-du-fichier>
```





## III. Travailler avec l'historique

### Introduction de la section :

Cette section traite de l'historique des commits

- Visualiser l'historique : détaille comment voir l'historique des commits
- Afficher les détails d'un commit spécifique : détaille comment voir les informations d'un commit

### A) Visualiser l'historique

#### 1/ Dans SourceGit

L'interface de SourceGit, tout comme n'importe quel autre outil GUI, a pour fenêtre principale l'historique en "graph" et synthétise les infos de chaque commit en une ligne.

#### 2/ Avec l'outil CLI

Affichage classique :

```
git log
```

Affichage plus compact :

```
git log --oneline
```

Affichage en "graph" :

```
git log --graph
```

Affichage en "graph" plus compact (plus proche de l'interface SourceGit) :

```
git log --graph --oneline
```

```
hugoc@Legion-5-15ACH6H.../wizards-n-dices$ git log --graph --oneline
* 9e754d0 HEAD -> recapGit écriture de la partie "Suivi des fichiers"
* dc7c9b0 (origin/main, origin/HEAD, main) Ajout de la version 2 (finale) au format PDF du texte
* 4c83d40 ajout de la version 2 (finale) du texte pour le pitch
* 9b25f4a Réunion de préparation du jalon 2
* 23b69b0 Changement mineur, mais reste encore la V1
* 654e6ce Ajout du brouillon pour l'oral de la fiche PT
* 38eb955 Ajout du texte en format PDF
* 8f8e455 Déplacement du fichier GDT dans le materials
* 2333980 Ajout du premier jet pour le texte pour le pitch oral
* d092921 Construction initiale GANTT
* 37ceead Réunion Jalon 2 et organisation interne
* 42e6412 Déploiement du planning
* b5e889c Avancé gestion interne
* 27fa756 Planning des tâches et formalisation
* e8338c1 Planning des tâches - GANTT v0
* 108114a Planning des tâches - liste
* 79d2f6b Réunion de revue des tâches et de charte graphique
* 2a3c3ae ajout du dispo de la présentation de la fiche PT
* d51feb3 Réunion de répétition pour l'oral de la fiche PT
* 76d2846 Finalisation fiche PT - rendu
* 0821257 Réunion de préparation de la présentation de la fiche PT
* 2777630 Réunion gestion de projet et préparation du jalon 2
* e3ee662 Merge branch 'fiche-pt'
*
* 8a0e23e TG-3 #closed TG-4 #in-progress
* 83d63c8 TG-2 #closed
* 8233225 Merge remote-tracking branch 'origin/main'
*
* 68bc7fa Fichier de synthèse en 2 pages pour fiche PTUT
* 7d8457a Réunion de workflow et de communication interne
*
* 4fe4c42 Réunion de relecture de la fiche PT
* 1114156 Réunion d'avancement de la fiche PT
*
* 7732a20 TG-2 #ready-for-test
* 1386596 ajout de .gitignore
* 99f3b6e Ajout du premier document de cadrage
* 81665a5 Réunion Fiche PT version 0
* 2a32f44 Fiche PTUT brouillon
* 9776021 Réunion de fin de préparation de la fiche PT
* 4c2b9a8 double
* 37c8db5 Répertoire de stockage des documents de références
* 792b158 Réunion Verification Git et Fiche PT
* 8a8e229 Préparation additionnelles de la réunion de samedi 19/10/2024
* 0c92637 Préparation de la réunion de samedi 19/10/2024
* 7284929 Réunion d'organisation et conduite du projet
* 26f4538 Test Merge fusion manuelle
* 4641922 Mise au propre
* be6e52f Merge remote-tracking branch 'origin/main'
*
* 82bb8ef Merge remote-tracking branch 'origin/main'
```

Figure 5 : Historique des commit affiché en graph via le CLI







## B) Afficher les détails d'un commit spécifique

### 1/ Dans SourceGit

Cliquer sur un commit permet de voir plusieurs informations, séparées en trois onglets : les infos générales et les changements.

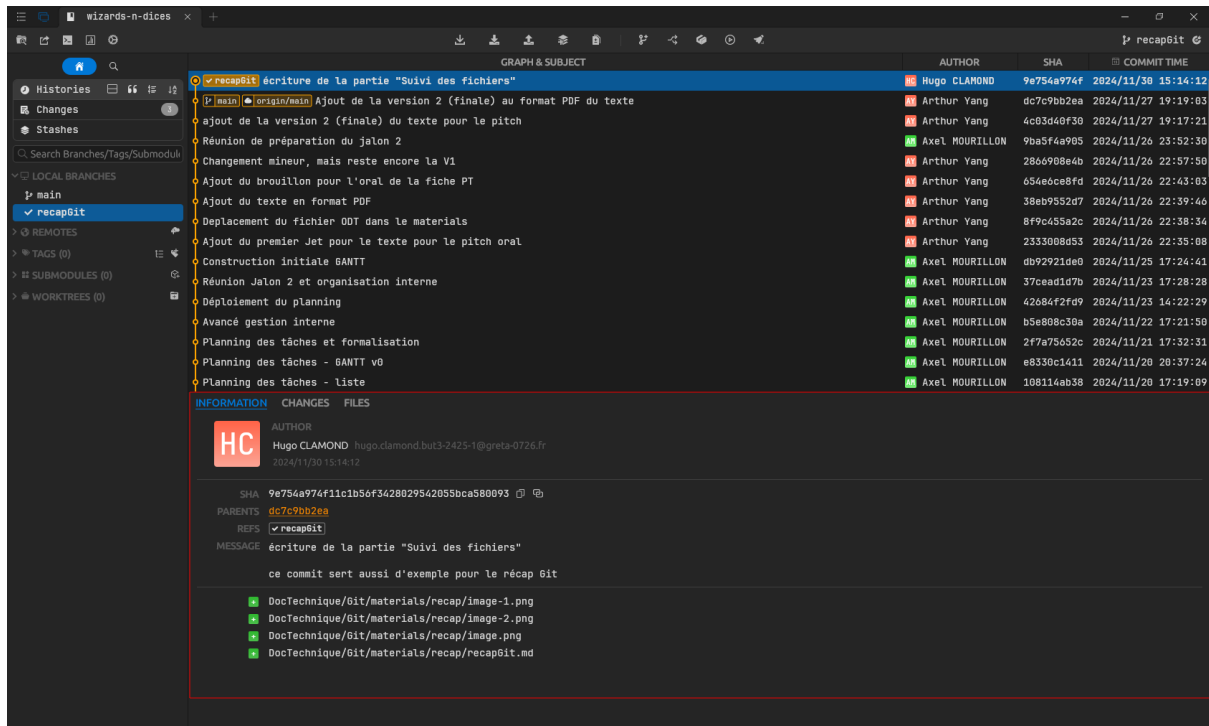


Figure 6 : Détails d'un commit

Les infos générales regroupent la plupart des infos que l'on souhaite récupérer d'un coup d'œil, à savoir :

- Les infos sur l'auteur du commit,
- Les branches sur lesquelles est le commit,
- Le message de commit,
- Les changements des fichiers liés au commit.

### 2/ Avec l'outil CLI

Une commande, parmi d'autres, d'obtenir des infos sur un commit en particulier, est la suivante :

```
git show <commit-id>
```

où "<commit-id>" correspond au hash du commit visé.





## IV. Branches

### Introduction de la section :

Cette section aborde le fonctionnement des branches

- Créer une nouvelle branche : détaille comment créer une branche
- Lister les branches : détaille comment lister les branches
- Changer de branche active : détaille comment changer de branche
- Fusionner une branche souhaitée dans la branche active : détaille comment fusionner deux branches
- Récupérer un commit spécifique d'une autre branche et le copier sur la branche active : détaille comment récupérer un commit spécifique
- Résoudre des commits de fusion post-merge ou pull : détaille quoi faire en cas de problème pendant une fusion

### A) Créer une nouvelle branche

#### 1/ Dans SourceGit

Cliquer sur un commit pour l'avoir en surbrillance dans l'historique, cliquer sur l'icône mise en évidence :

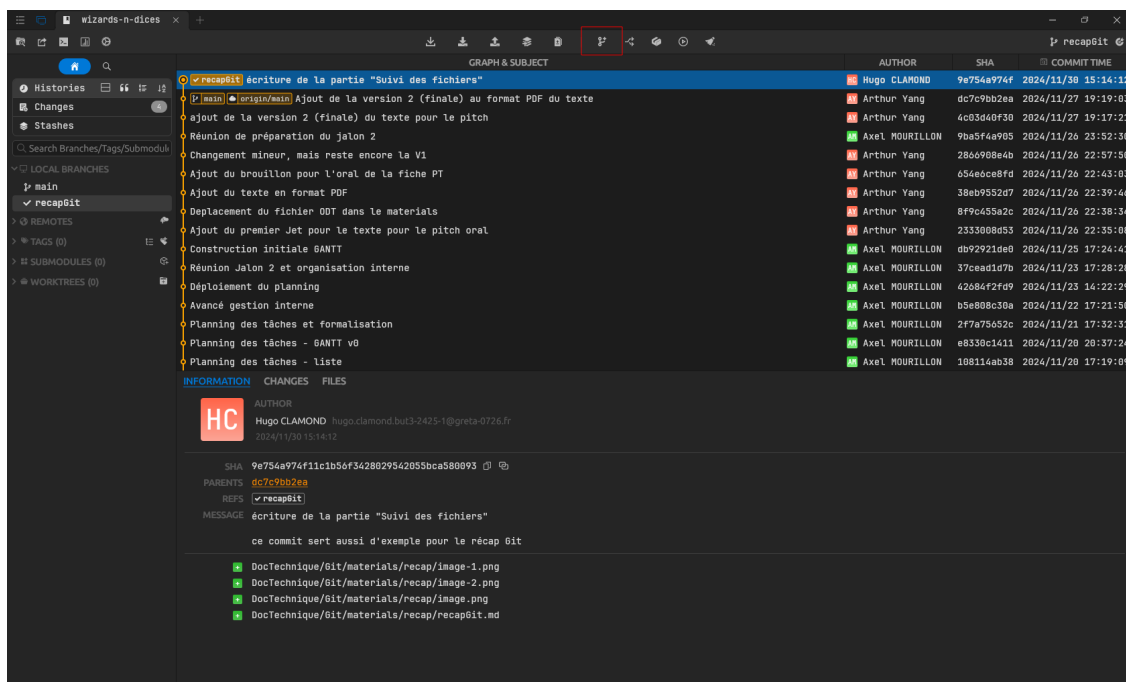


Figure 7 : Créer une nouvelle branche

#### ATTENTION :

Commitez vos changements avant de "checkout"(= vous placer) sur votre branche nouvellement créée. Git vous donnera un message d'erreur vous disant que vous ne pourrez pas forcément changer de branche avant d'avoir mis vos changements dans un commit ou autre.





## 2/ Avec l'outil CLI

Pour seulement créer une branche, saisir la commande suivante ...

```
git branch <nom-de-la-branche>
```

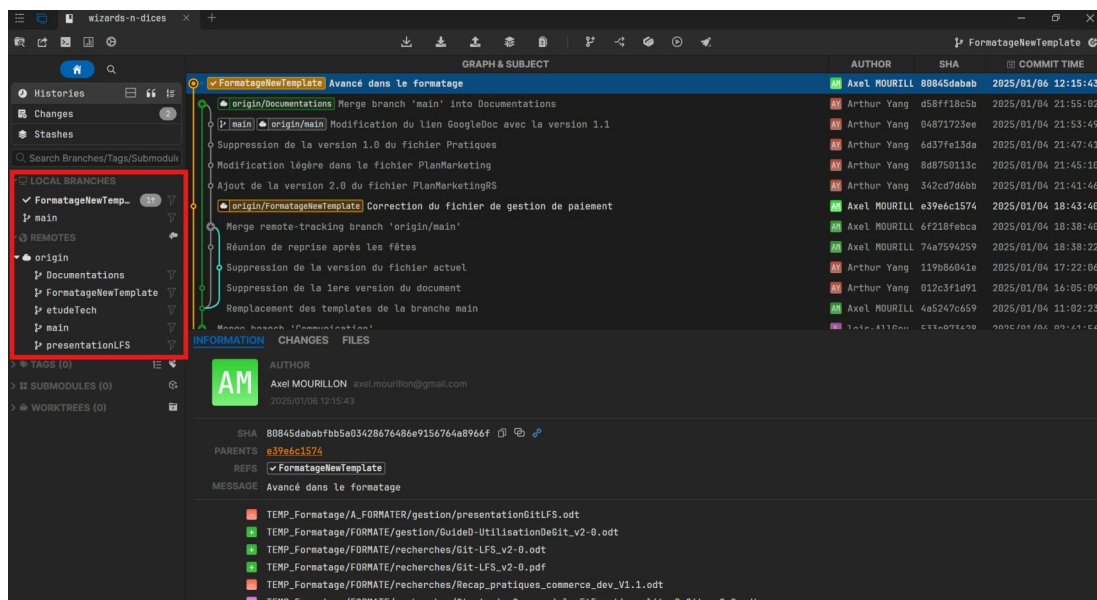
... ou créer une branche et basculer dessus immédiatement :

```
git checkout -b <nom-de-la-branche>
```

## B) Lister les branches

### 1/ Dans SourceGit

On peut voir les branches du repo, locales et distantes, dans la partie mise en évidence de la capture d'écran ci-dessous



**Figure 8: Branches locales et distantes du repo**

## 2/ Avec l'outil CLI

Pour afficher les branches, entrer la commande :

```
git branch
```

## C) Changer de branche active

### 1/ Dans SourceGit

Pour basculer sur une branche souhaitée, il suffit de double cliquer sur la branche souhaitée dans la partie de l'interface où l'on peut les visualiser (cf. section précédente).





### ATTENTION :

Committez vos changements avant de basculer sur une branche. Git vous donnera un message d'erreur vous disant que vous ne pourrez pas forcément changer de branche avant d'avoir mis vos changements dans un commit ou autre.

## 2/ Avec l'outil CLI

Pour basculer sur une branche, entrer l'une de ces deux commandes :

```
git checkout <nom-de-la-branche>  
# OU #  
git switch <nom-de-la-branche>
```

## D) Fusionner une branche souhaitée dans la branche active

### 1/ Dans SourceGit

Faites un clic droit sur la branche que vous souhaitez fusionner dans la branche actuelle, puis "Merge <branche-visée> into"

### 2/ Avec l'outil CLI

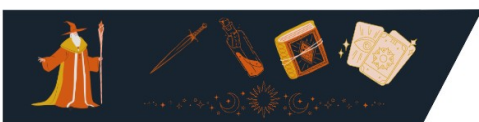
Pour fusionner une branche voulue dans la branche active, entrer la commandes :

```
git merge <nom-de-la-branche-souhaitée>
```

## E) Récupérer un commit spécifique d'une branche et le copier sur la branche active

Cette manipulation s'appelle le cherry-pick. Bien qu'il puisse s'agir d'une manipulation plutôt avancée, je choisis d'en parler car il peut être intéressant de récupérer un commit spécifique d'une autre branche ayant évolué en parallèle de votre branche actuelle, sans pour autant faire un merge de toute l'autre branche.

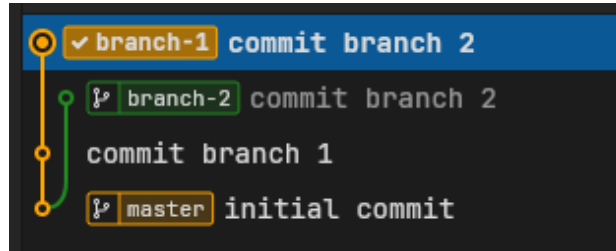
Cela peut être utile si, par exemple, en étant sur votre branche actuelle, qu'on va nommer "A", vous voulez récupérer le commit n°1 de la branche B, mais que le commit n°2 introduit un bug que vous ne voulez pas.





## 1/ Dans SourceGit

En étant dans votre branche actuelle, sélectionnez un commit d'une autre branche ayant évolué en parallèle. Faites clic droit, puis "Cherry-pick this commit". La capture d'écran ci-dessous montre, sur un repo de démonstration, que le cherry-pick a copié le commit de la branch "branch-2" dans la branche "branch-1".



**Figure 9 : Exemple de "cherry-pick"**

Note :

Si vous faites cherry-pick d'un commit qui a été réalisé avant votre dernier commit présent sur votre branche actuelle, il se place alors avant votre dernier commit sur votre branche. Par ailleurs, si vous prenez un commit d'une autre branche qui a déjà plusieurs commits avant lui, alors il copiera également les commits qui ont été réalisés avant, dans votre branche actuelle.

### ATTENTION :

Un cherry-pick peut, au même titre qu'un merge, provoquer des conflits de fusion. Voir la section IV. F) Résoudre des commits de fusion post-merge ou pull

## 2/ Avec l'outil CLI

Tout d'abord, récupérer le hash du commit souhaité. Ensuite :

```
git cherry-pick <commit-id>
```

où "<commit-id>" correspond au hash du commit visé.

## F) Résoudre des commits de fusion post-merge ou pull

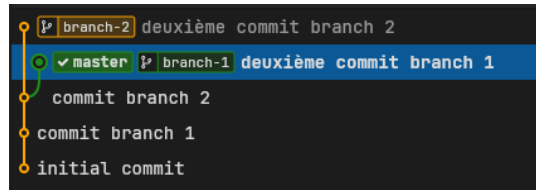
Je vais montrer une situation simple de conflit de fusion, où deux branches ont réalisé le même changement, ici un readme.md. Les deux branches, "branch-1" et "branch-2", ont modifié la ligne 2 du fichier. On va voir comment résoudre le conflit. Ici, on veut récupérer les changements des deux côtés. Il existe des cas où l'on ne voudra récupérer que les changements d'un côté, où quelques changements de l'un et de l'autre.





## 1/ Dans SourceGit

Dans notre exemple (toujours sur un repo de démonstration), on se place sur la branche "master", puis on commence par fusionner les changements de la branche "branch-1" dans la branche "master".



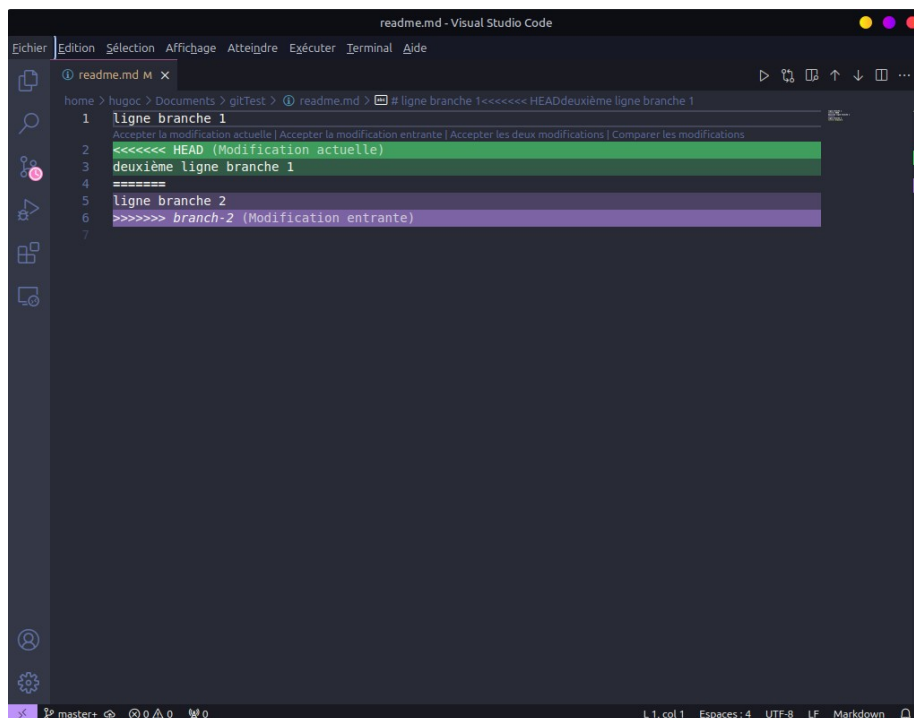
**Figure 10 : Fusion post-merge SourceGit – contexte**

Puis on réalise un autre merge, de la "branch-2" vers la "master". Un bandeau s'affiche pour indiquer qu'il y a un conflit. On clique alors sur "Resolve".

	AUTHOR	SHA	COMMIT TIME
initial commit	Hugo CLAROND	392f56416b	2024/12/02 09:17:56
commit branch 1	Hugo CLAROND	7884d8e6ae	2024/12/02 09:17:31
commit branch 2	Hugo CLAROND	a59a034d82	2024/12/02 09:06:20
commit branch 1	Hugo CLAROND	3d88d3a4d4	2024/12/02 09:03:59
commit branch 2	Hugo CLAROND	7b950085b8	2024/12/02 09:02:21

**Figure 11 : Fusion post-merge SourceGit – conflit**

D'ici, on fait un clic droit sur le changement en conflit, représenté sur SourceGit par une icône rouge contenant un point d'exclamation. Puis on sélectionne "Open external merge tool". Si vous avez bien configuré votre SourceGit, ça devrait ouvrir VSCode sur le fichier en question.



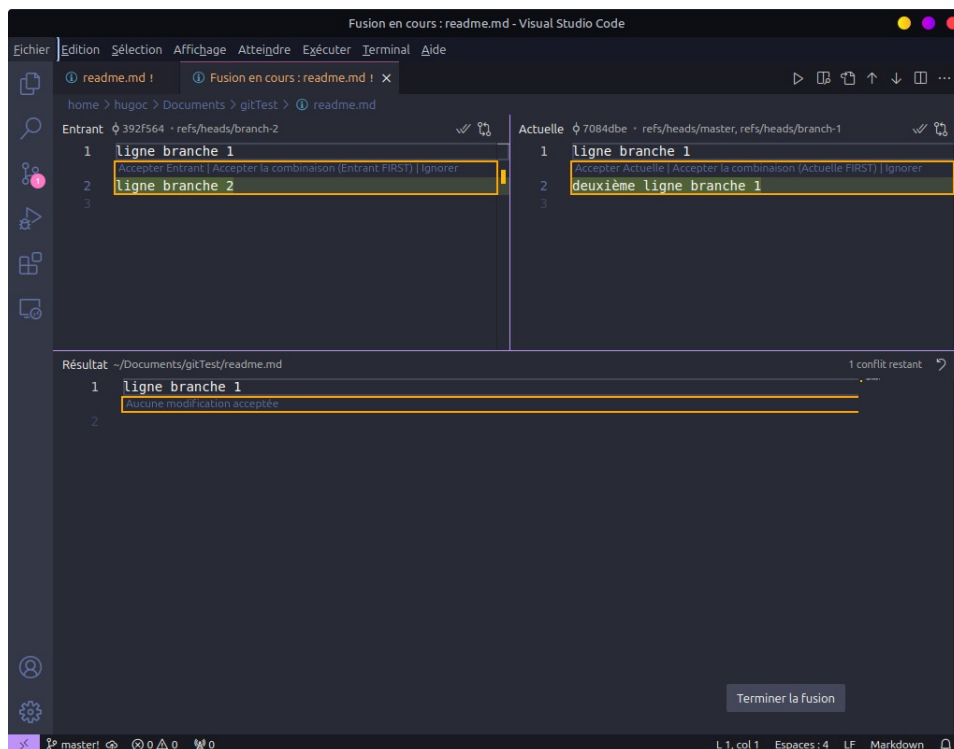
**Figure 12 : Fusion post-merge SourceGit – outil de fusion externe**





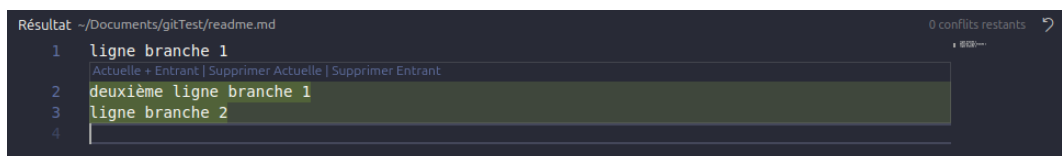


D'ici, on clique sur "Résoudre dans l'éditeur de fusions", et on arrive sur la capture d'écran suivante



**Figure 13 : Fusion post-merge SourceGit – résolution de conflit**

Vu qu'on souhaite récupérer les changements des deux côtés, on clique sur le texte cliquable "Accepter Actuelle", puis sur "Accepter Entrante". Le résultat est le suivant :

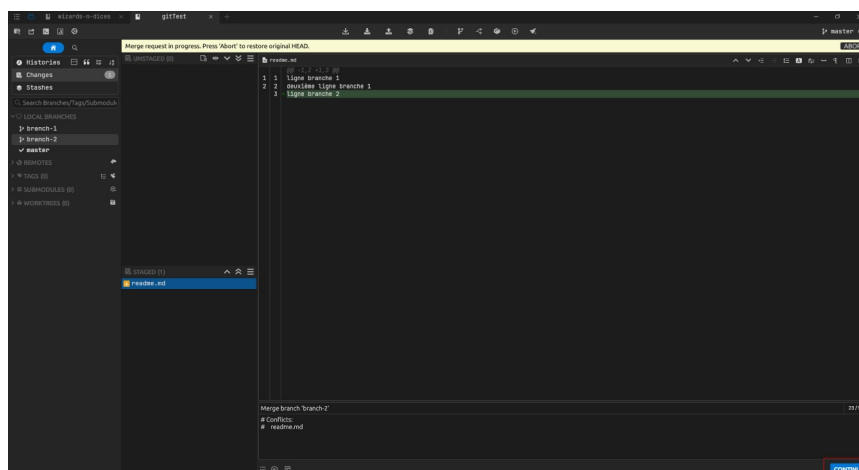


**Figure 14 : Fusion post-merge SourceGit – confirmation des changements**

On clique sur le bouton "Terminer la fusion", puis on ferme la fenêtre de VSCode qui s'est ouverte. Cette étape est très importante pour que SourceGit détecte que l'on a fini de travailler sur le conflit.

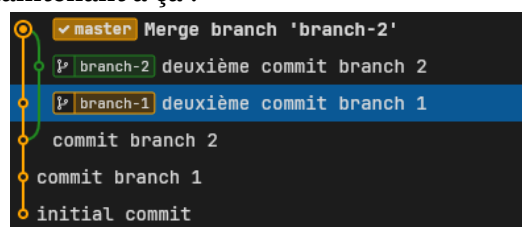
De retour dans SourceGit, on clique sur "Continue" (voir capture d'écran ci-après) :





**Figure 15 : Fusion post-merge SourceGit – finalisation**

L'historique ressemble maintenant à ça :



**Figure 16 : Fusion post-merge SourceGit – historique**

## 2/ Avec l'outil CLI

Pour initier la fusion, entrer la commande suivante :

```
git merge <nom-de-la-branche-souhaitée>
```

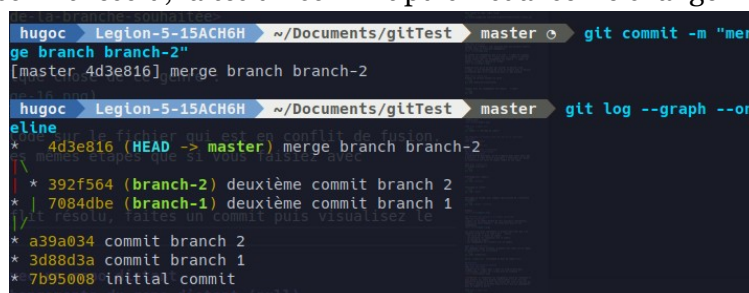
Vous verrez quelque chose de ce genre :



**Figure 17 : Fusion post-merge CLI – message d'erreur de fusion**

Ouvrez alors VSCode sur le fichier qui est en conflit de fusion. De là, suivez les mêmes étapes que si vous faisiez avec SourceGit.

Une fois le conflit résolu, faites un commit puis visualisez le changement.



**Figure 18 : Fusion post-merge CLI – historique après le merge**





## V. Travailler avec un repo distant

### Introduction de la section :

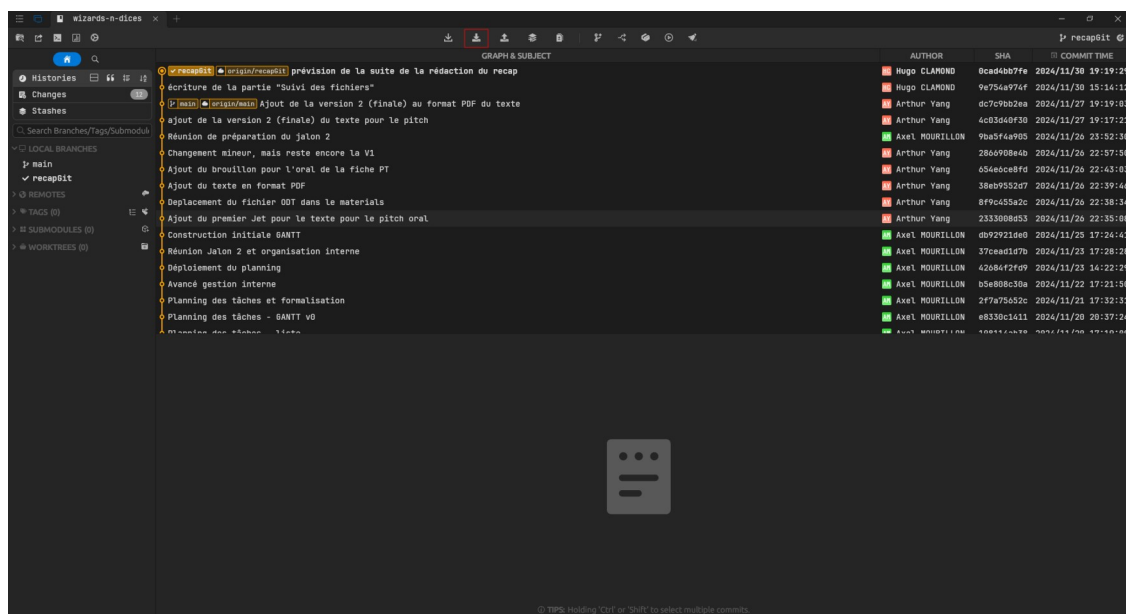
Cette section aborde les interactions entre le repo local et le repo distant

- Tirer les changements du repo distant (pull) : détaille comment tirer les changements
- Récupérer l'historique des commits de toutes les branches sans les télécharger (fetch) : détaille comment récupérer l'historique des commits
- Pousser les changements sur le repo distant (push) : détaille comment pousser des changements

### A) Tirer les changements du repo distant (pull)

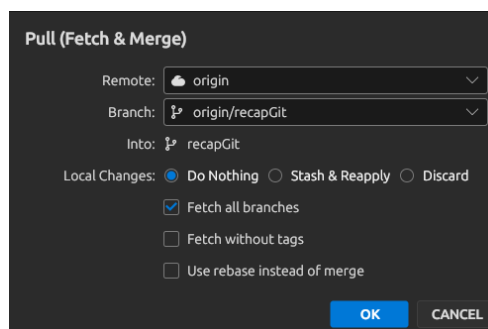
#### 1/ Dans SourceGit

Cliquez sur l'icône mise en évidence sur la capture d'écran :



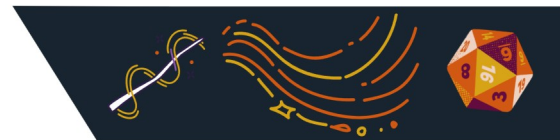
**Figure 19 : Tirer les changements du repo distant**

Cette fenêtre s'ouvre, faites bien attention à ce que les réglages soient les mêmes que sur la capture d'écran :



**Figure 20 : Paramètre du "pull"**





## 2/ Avec l'outil CLI

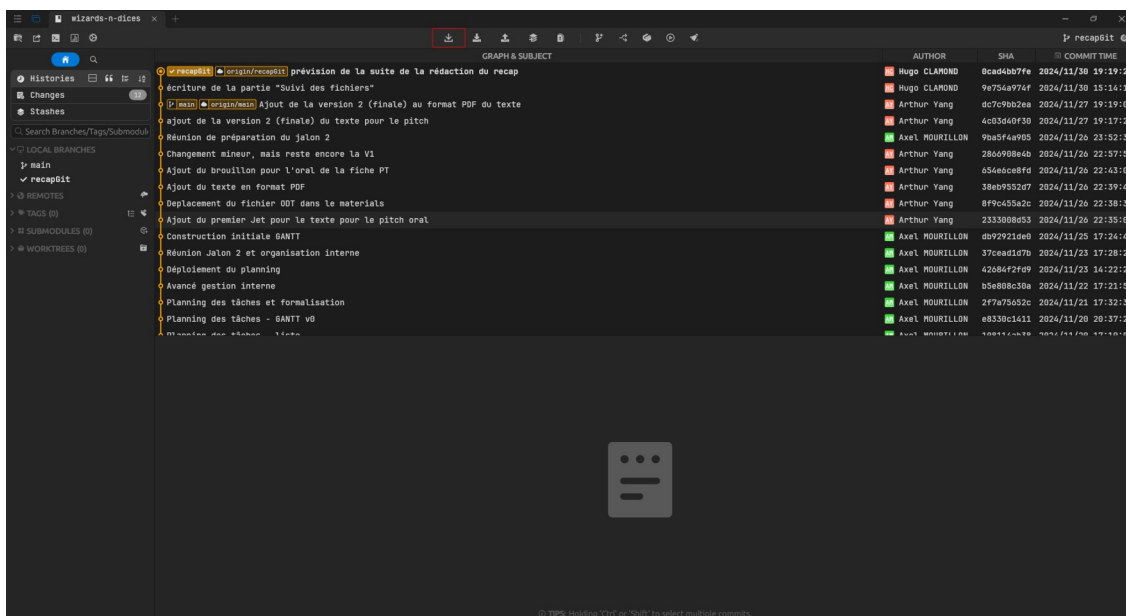
Pour tirer les changements, effectuer la commande suivante :

```
git pull
```

## B) Récupérer l'historique de commits de toutes branches sans les télécharger (fetch)

### 1/ Dans SourceGit

Cliquez sur l'icône mise en évidence sur la capture d'écran :



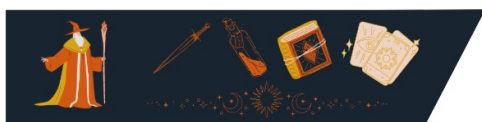
**Figure 21 : Récupérer uniquement l'historique des données**

Une autre fenêtre s'ouvre : laissez tout par défaut

## 2/ Avec l'outil CLI

Pour récupérer uniquement l'historique des commits de toutes les branches, effectuer la commande suivante :

```
git fetch --all
```



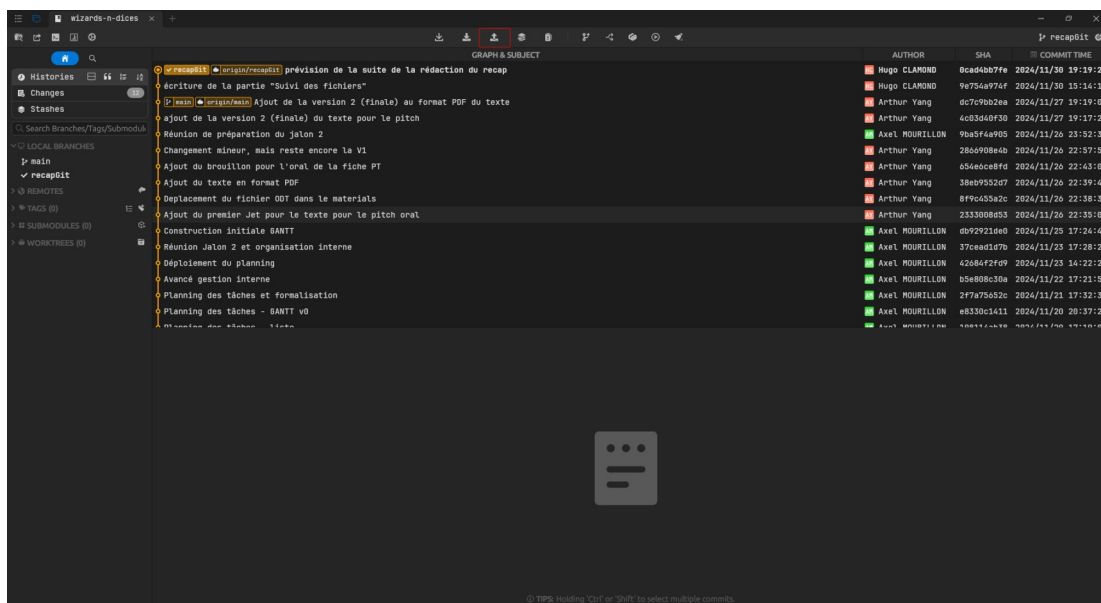


## C) Pousser les changements sur le repo distant (push)

Quoi qu'il arrive, vous devez effectuer cette opération après avoir fait un pull ou un fetch, et vous devez avoir réalisé un commit.

### 1/ Dans SourceGit

Cliquez sur l'icône mise en évidence sur la capture d'écran :



**Figure 22 : Pousser des changements sur le repo distant**

**ATTENTION :** Ne **JAMAIS** cocher la case *Force push*.

### 2/ Avec l'outil CLI

Pour pousser des changements sur le repo distant, effectuer la commande suivante :

`git push`





## VI. Workow général et bonnes pratiques

### Workow général :

- Lorsque l'on travaille sur un nouveau sujet, on crée une branche nommée d'après le dit sujet. Une fois le travail terminé et prêt à être rendu dans la branche stable, on fait un merge vers la branche "main". Une fois le merge terminé, on supprime la branche.  
Clic droit sur la branche > "Delete "  
Cocher "Also remove remote branch "
- Lorsque l'on veut commencer à travailler sur un nouveau sujet, on part du dernier commit de la branche "main" et on crée la branche à partir de ce dernier commit.
- On évite de travailler directement sur la main, sauf pour quelques **exceptions n'impactant pas la stabilité** du projet (compte-rendus de réunion, planning organisationnel, etc.).

### Bonnes pratiques :

- Je fais régulièrement des commits durant mon avancée sur mon travail.
- Je garde les messages de commit clairs et concis. On doit pouvoir savoir directement ce sur quoi j'ai travaillé en lisant le message de commit.
- Je supprime les branches obsolètes après un merge pour garder le repo propre.
- J'utilise des noms de branche explicites, par exemple :
  - *fix-affichage-images*
  - *feature-ajout-tableau*
  - *doc-update-recap-git*
- Avant de pousser, je m'assure que le code fonctionne et a été testé. Si des tests automatisés sont en place, je vérifie qu'ils passent tous. Si je sais que le code ne fonctionne pas mais que je dois pousser malgré tout, je le marque dans mon message de commit.
- Si des conflits de fusion se présentent, je prends le temps de les résoudre avec attention, pour éviter les pertes de données accidentelles.
- Si je veux garder un fichier dans le repo mais que je ne veux pas que Git le prenne en compte, je le mets dans le dossier *wizards-n-dices/\_fichiers\_temporaires*.  
Si le besoin s'en fait sentir, je propose au responsable technique l'ajout des dossiers ou des fichiers à ignorer dans *wizards-n-dices/.gitignore*.
- Contacter le responsable technique avant d'éditer le fichier *wizards-n-dices/.gitignore*.
- Contacter le responsable technique en cas de doute sur l'utilisation de Git persistant après la lecture de ce document.

