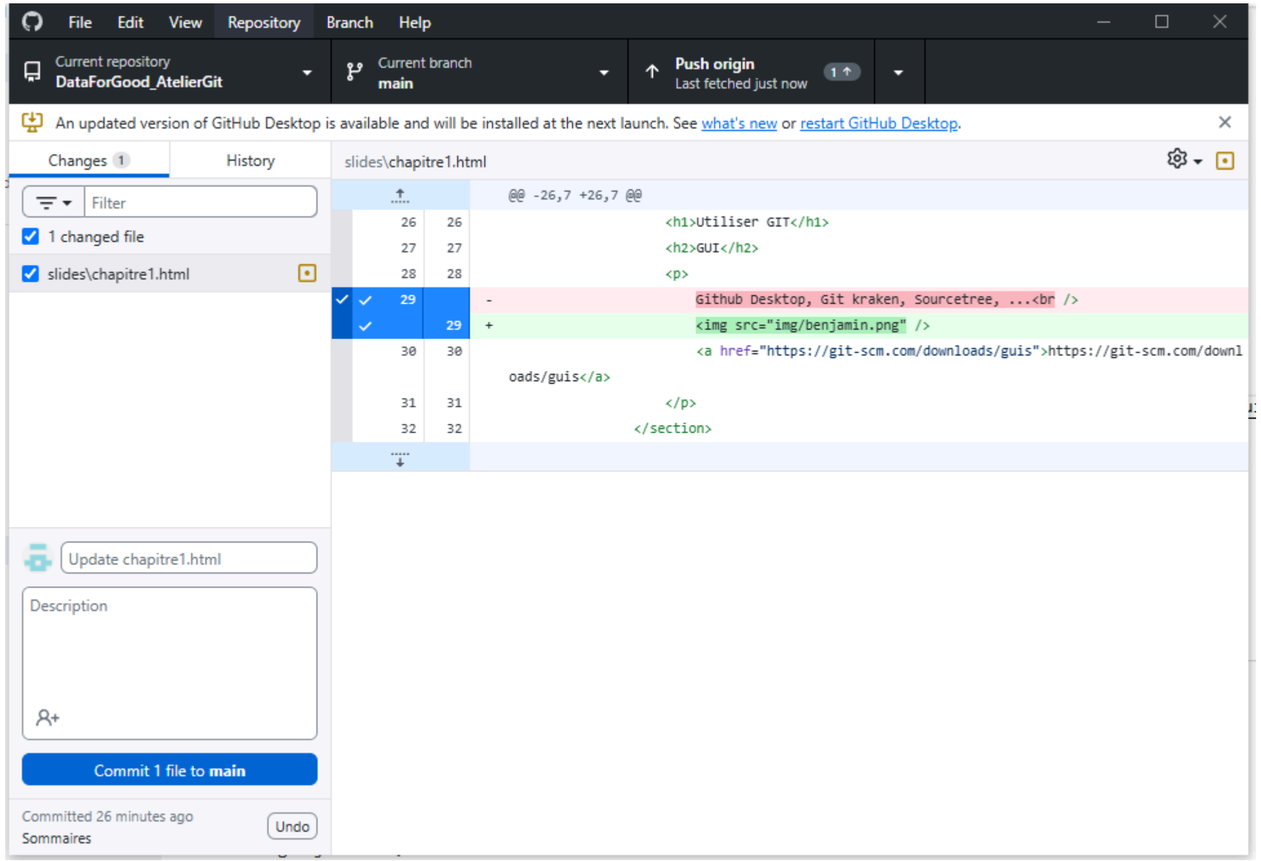
UTILISER GIT

LIGNE DE COMMANDE

```
> git ...
```

UTILISER GIT

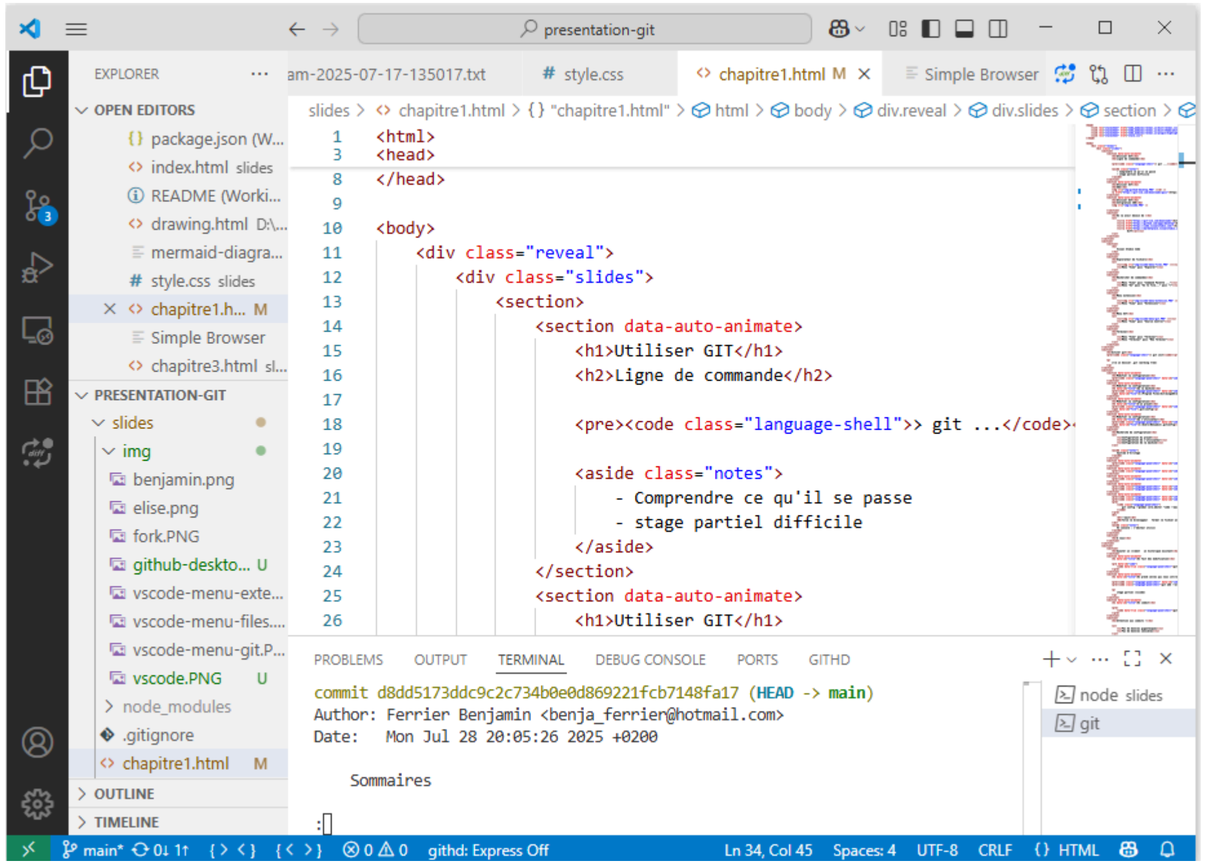
GUI



https://git-scm.com/downloads/guis

UTILISER GIT

INTÉGRATION IDE



ON VA AVOIR BESOIN DE

- Git
- Github Desktop
- Visual Studio Code
- Git History Diff

ACTIVER GIT

```
git init
```

crée un dossier .git (working tree)

MODIFIER LA CONFIGURATION

```
git config
```

MODIFIER LA CONFIGURATION

DE LA MACHINE

```
git config --system
```

C:\Program Files\Git\mingw64\etc\gitconfig

MODIFIER LA CONFIGURATION

D'UN PROJET

```
git config
```

.git/config

MODIFIER LA CONFIGURATION

DE L'UTILISATEUR

```
git config --global
```

C:\Users\Benjamin\.gitconfig

RECHERCHE DE CONFIGURATION

1. Configuration du projet
2. Configuration de l'utilisateur
3. Configuration de la machine

```
git config --global
```

```
git config --global user.name
```

```
git config --global user.name "Ferrier Benjamin"
```

```
git config --global user.name "Ferrier Benjamin"
```

```
git config --global user.email
```

```
git config --global user.name "Ferrier Benjamin"
```

```
git config --global user.email
```

```
git config --global core.editor "code --wait"
```

--wait

force le développeur à fermer le fichier avant de continuer

A VOUS

AJOUTER UN ÉLÉMENT À UN HISTORIQUE EXISTANT

ON FAIT DES MODIFICATION

```
git status
```

ON PREND CELLES QUI NOUS INTÉRESSES (STAGE)

```
git add "nom/chemin du fichier"
```

```
git add --all
```

stage partiel (vscode)

ON COMMIT

```
git commit -m "message"
```

ATTENTION AUX COMMITS !

- Pas de données gigantesques
- Pas de données sensibles

QUELLE TAILLE DE COMMITS ?

- Trop gros: difficile à comprendre le commit
- Trop petit: difficile de comprendre l'historique

QUELLE TAILLE DE COMMITS ?

- Trop gros: difficile à comprendre le commit
- Trop petit: difficile de comprendre l'historique

- Modification qui intéragissent ensemble
- Apporte un changement "visible"

AFFICHER L'HISTORIQUE

```
git log
```

Commandes du man (linux)

- / + N: recherche textuelle
- q: pour quitter

GIT HISTORY DIFF

```
> GitHd: View Entire History
```

AFFICHER UN COMMIT

```
git show "id du commit"
```

GIT HISTORY DIFF

```
> GitHd: Input ref
```

```
> id
```

A VOUS

BRANCHES

- Pouvoir abandonner une évolution entamée
- Travailler sur plusieurs sujets en parallèle

BRANCHES

1. **On va sur la branche principal (master ou main)**
2. On crée une nouvelle branche
3. On va dessus
4. Les nouveaux commit sont uniquement sur notre branche
5. **Quand on a terminé, on récupère les nouveautées de la branche principal**
6. **Puis on ajoute les nouveaux commits sur la branche principal**

CRÉER UNE BRANCHE

```
git branch nom_branch
```

CHANGER DE BRANCHE

```
git switch nom_branch
```

LISTER LES BRANCHES

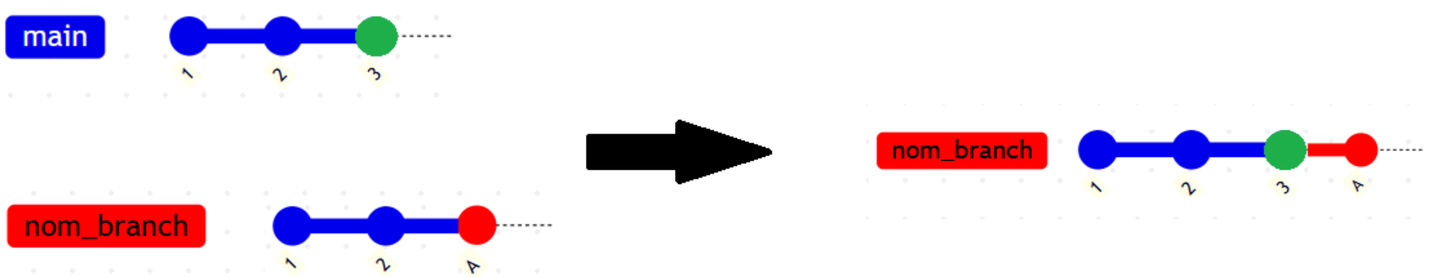
```
git branch
```

SUPPRIMER UNE BRANCHE

```
git branch -d nom_branch
```

RÉCUPÈRE LES NOUVEAUTÉ DE MASTER

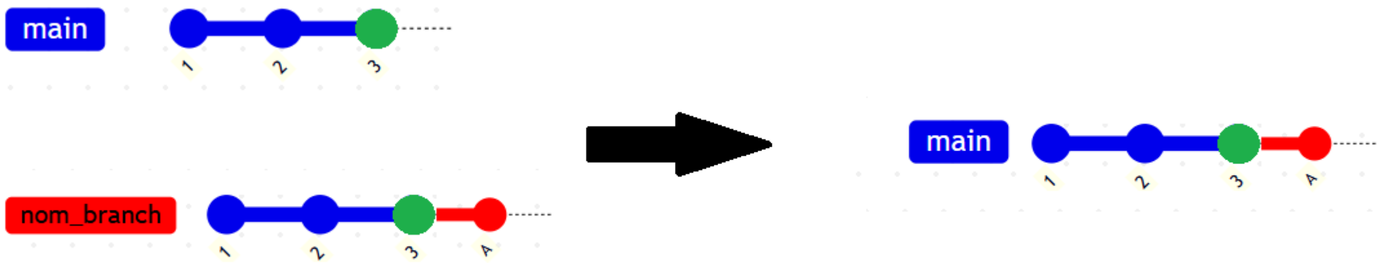
```
git rebase master nom_branch
git rebase --continue
git rebase --abort
```



AJOUTER NOS COMMIT À MASTER

FAST FORWARD

```
git switch master
git merge nom_branch
```



EN INTERNE

BRANCHES

- Les branches sont dans .git\refs\heads
- Chaque ficher branche contient l'identifiant du dernier commit
- .git\HEAD permet de connaitre la branche courante

EN INTERNE

RETRACER L'HISTORIQUE ?

Les commits sont stockés dans .git/object

```
git cat-file -p "commit id"
```

EN INTERNE

CONTENU D'UN COMMIT

- **tree** Id du fichier contenant les modifications
- **parent** Id du commit précédent
- **author** user.name <user.email> date_commit
- **message**

EN INTERNE

AFFICHER L'HISTORIQUE

1. Récupérer le premier commit dans .git/HEAD
2. Afficher le contenu du fichier
3. Récupérer l'identifiant du parent
4. Afficher le contenu du fichier parent
5. ...

Optimisation: .git\logs

EN INTERNE

ID DU COMMIT = HASH

UN HASH:

- Transforme un contenu en une clée unique
- Si le contenu change, le hash change

SOMMAIRE