Environnement de développement Web 2

#4 - Git (Inspection)

Par : Lilia Ould Hocine
Collège de Maisonneuve, été 2024

Sommaire

1. Visualiser l'historique d'un projet Git avec ***git log***
2. Visualiser l'historique d'un fichier avec ***git blame***
3. L'onglet ***Source Control*** de ***VS Code***
4. L'extension ***Git Lens***
5. ***master*** et ***HEAD***
6. ***git checkout***

01

Git log

EDW4 - 01

- git log
- git log -p (patch text - similaire à git diff)
- git help
- git help log
- git log --oneline
- git log --pretty=oneline
- git log -(nombre de commit) / commit -n (nombre de commit)
- git log --author=”lilia”
- git shortlog
- git log --grep=”test”
- git log --stat
- git log --since=1.weeks
- git log -S methodeModifiee (tous les commits qui ont touchés à cette méthoe)

02

Historique

EDW4 - 02

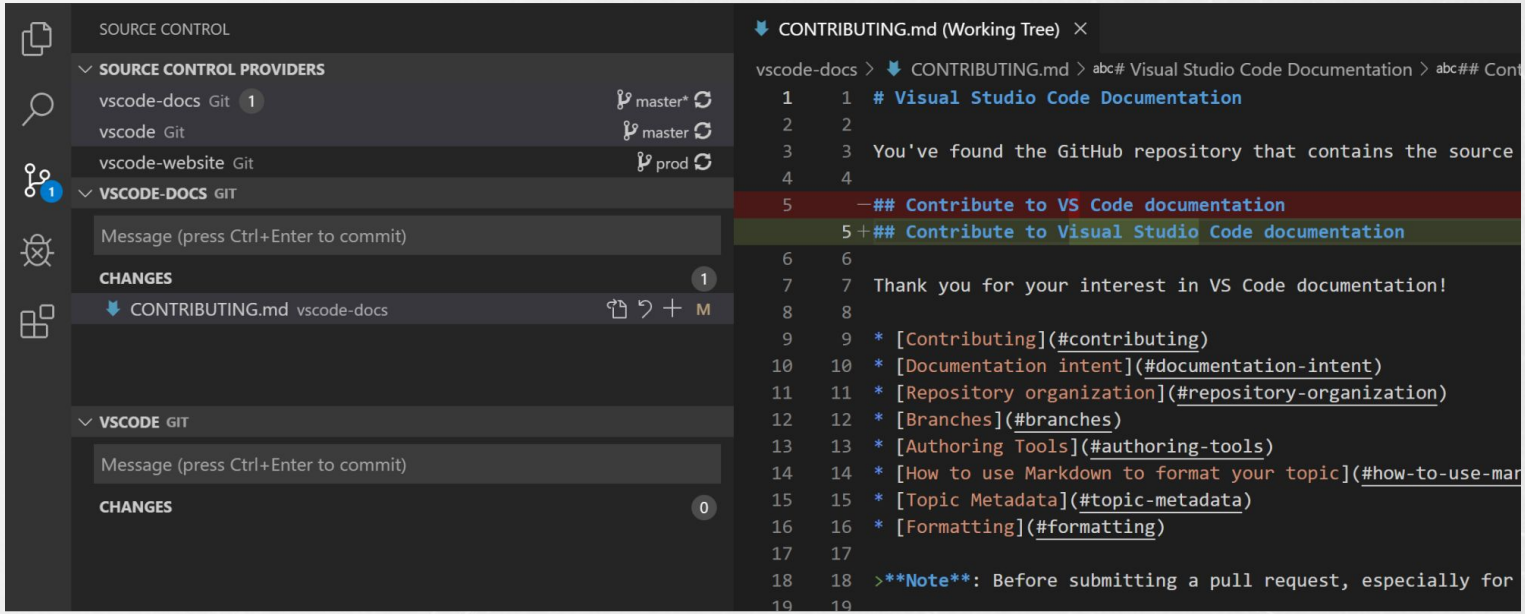
- git blame (plus dure que ça l'est) / git help blame
- git blame index.html
- git blame index.html -b (enlever le hash)
- git blame index.html -L 1,3
- git blame index.html -C

03

Code Source VS Code

EDW4 - 03

Par défaut, Visual Studio permet d'utiliser très facilement miscellaneous grâce à l'onglet ***Source Control*** :



EDW4 - 3

VS Code sépare les fichiers modifiés en trois groupes :

- Premièrement, les fichiers sous ***CHANGES*** : à savoir les fichiers qui ont au moins une modification et qui ne sont pas indexés.
- Deuxièmement, les fichiers sous ***STAGED CHANGES*** : ce sont les fichiers qui sont indexés dans la zone de transit.
- Troisièmement, les fichiers sous ***MERGE CHANGES*** : nous verrons plus tard cette catégorie.

EDW4 - 3

- Lorsque vous avez des fichiers modifiés et sauvegardés, vous pouvez les indexer avec ***git add***. Il est également possible d'utiliser le bouton ***+***, intitulé ***Stage changes***, lorsque vous passez la souris.

  Il a exactement le même effet que de faire ***git add fichier***.

- Pour les fichiers indexés, qui se trouvent dans le groupe ***STAGED CHANGES***, vous verrez que vous avez un bouton ***-***. Il permet d'annuler l'indexation.
- Comme vous avez des changements indexés et que vous voulez les sauvegarder définitivement dans une version, nous avons vu que vous pouviez faire ***git commit -m "Message d'enregistrement"***. Sur VS Code, vous pouvez de plus remplir la case ***Message*** avec le message de validation, puis cliquer sur la flèche de validation.

EDW4 - 3

- Dans un fichier, ***VS Code*** a un code couleur qui va vous indiquer les lignes ajoutées, modifiées et supprimées.
    - Les rouges correspond à la suppression
    - Le vert correspond à l'ajout
    - Le bleu correspond à la modification.

- Lorsque vous cliquez sur le nom d'un fichier dans l'onglet ***Source Control***, l'outil de comparaison vous permet de visualiser les changements exacts.

04

Git lens

EDW4 - 4

- L’extension ***GitLens*** de ***VS Code*** permet principalement de visualiser directement dans l’ éditeur, les informations fournis par ***git blame***.
- La fonctionnalité principale est de pouvoir voir quelles lignes ont été modifiées, quand et par qui directement dans ***VS Code***.
- En haut de tous les fichiers, vous aurez maintenant également une nouvelle ligne, elle indique, qui a modifié le fichier en dernier et quand.
- Aussi, combien de personnes ont touché au fichier (***x authors***) et l’auteur principal (celui qui a le plus de lignes écrites.
- Si vous cliquez, vous aurez un espace pour chaque ligne du dernier commit l’ayant modifiée et la date de la modification.
- Si vous cliquez sur un ***commit*** dans la colonne de gauche, les lignes modifiées seront surlignées.
- Plus de notions sont disponibles, que nous verrons au fur et à mesure que nous avançons sur ***Git***.

05

Master et HEAD

EDW4 - 5

Dans ce chapitre, nous allons nous intéresser en détails à défaire des changements qui sont soit dans le répertoire de travail, soit indexés.

Nous allons apprendre comment faire pour annuler des modifications, et revenir à l'état souhaité, sans perdre ou en perdant ces modifications.

master :

Nous verrons bientôt les branches de manière plus approfondie, mais aujourd'hui, nous entamons le sujet.

Nous avons appris qu'un commit contient notamment une référence à un ***tree***, et ce dernier contient des références vers des ***blobs*** ou d'autres ***tree*** et ainsi de suite. L'ensemble constitue une version du projet, ça nous permet d'avoir la structure du projet ainsi que le contenu en binaire.

Tous les ***commits*** (sauf le premier) ont une référence à leur ***commit*** parent. Nous verrons qu'un ***commit***, peut avoir plusieurs parents, en cas de fusion.

EDW4 - 5

Une branche est donc tout simplement un pointeur, également appelé référence, vers un ***commit***.

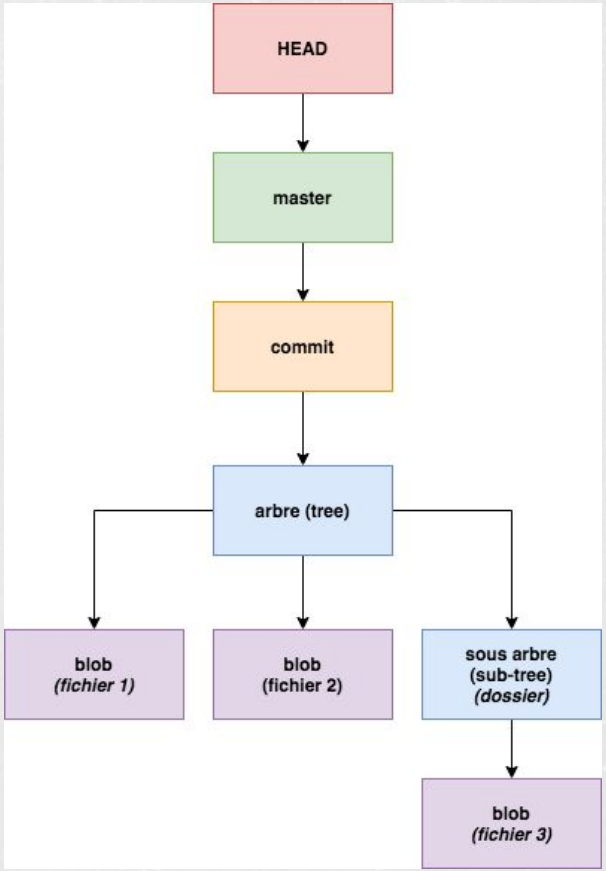
master est la branche principale. Créée par défaut pat Git lors d'un ***git init***.

Au fur et à mesure que vous créez des commits sur une branche, la référence de la branche se déplace automatiquement au dernier ***commit***.

Qu'est-ce que ***HEAD*** :

HEAD est une référence, ou un pointeur, vers le commit, la branche, ou le tag sur lequel vous vous trouvez actuellement. Autrement dit, ça pointe directement ou indirectement vers un commit.

EDW5 - 5


HEAD
master
commit
arbre (tree)
blob
(fichier 1)
blob
(fichier 2)
sous arbre
(sub-tree)
(dossier)
blob
(fichier 3)

06

git checkout

EDW4 - 6

La commande ***git checkout*** est l'une des plus utilisées dans Git. Elle permet de se déplacer vers une autre branche, ou vers un tag (comme nous le verrons), vers un autre commit, ou de restaurer la version indexée.

Il faut vraiment voir la commande ***checkout*** comme une commande de navigation de ***HEAD***. Autrement dit, elle permet de déplacer ***HEAD*** où vous souhaitez.