Environnement de développement Web 2

#3 - Git (les bases)

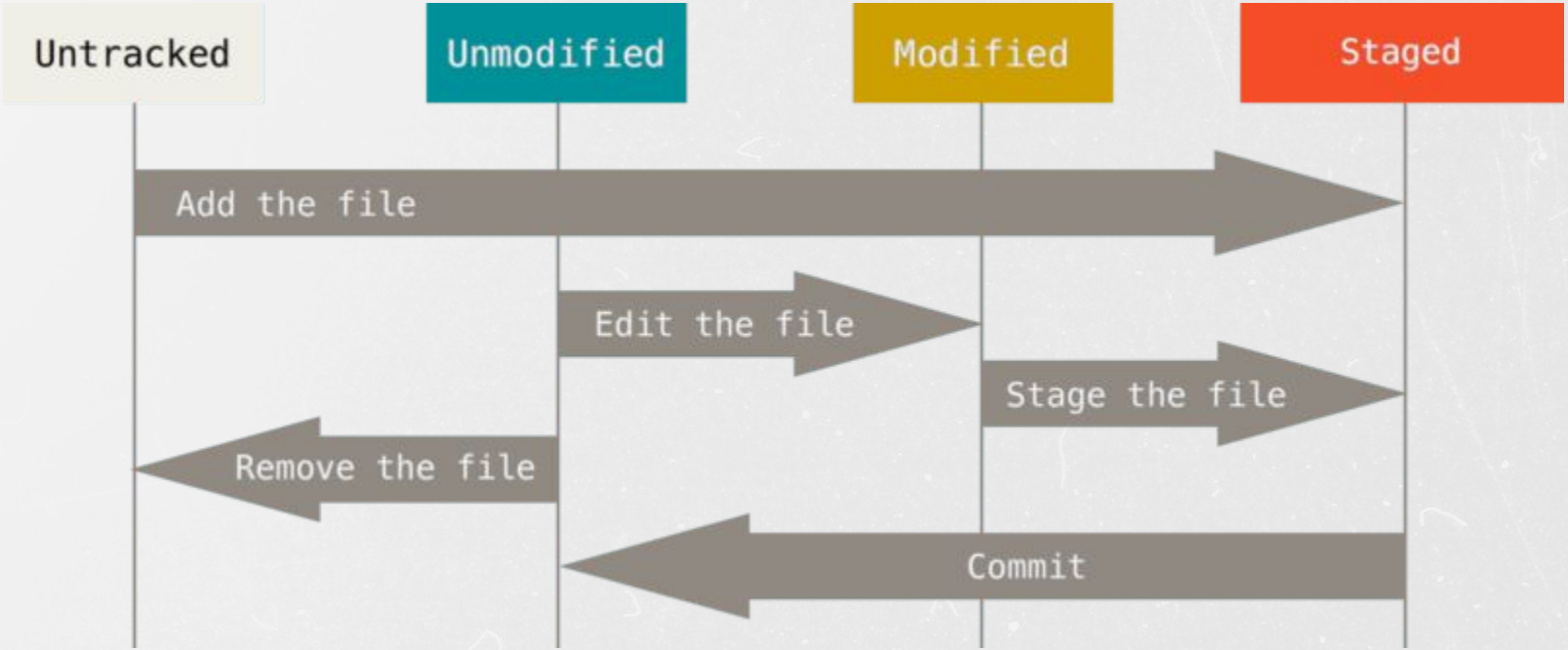
Par : Lilia Ould Hocine
Collège de Maisonneuve, été 2024

Sommaire

01

Le suivi des fichiers

EDW4 - 1


Untracked
Unmodified
Modified
Staged
Add the file
Edit the file
Stage the file
Remove the file
Commit

EDW4 - 1

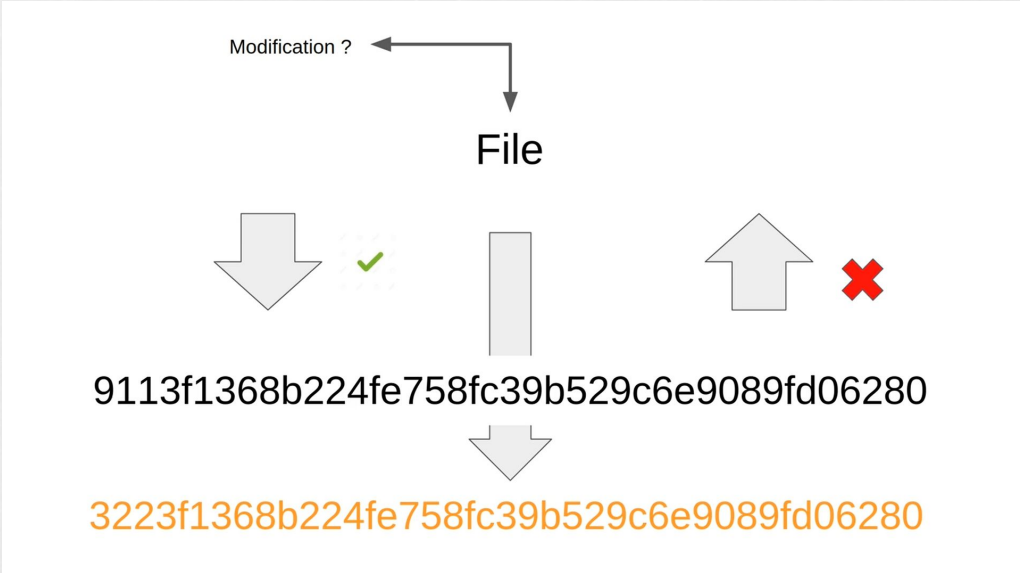
- (th) Créer un fichier (dans le projet “initié”)
  - État suivi
  - État non suivi
- (th) Créer une version
  - Étape intermédiaire (indexation)
  - Fichier suivi
- (thd) Créer un fichier dans notre dossier de projet
  - ***git status***
  - ***git add / git add .***
  - ***git status***
  - ***git ls-files --stage***
  - ***git commit -m “first commit”***

02

État et index

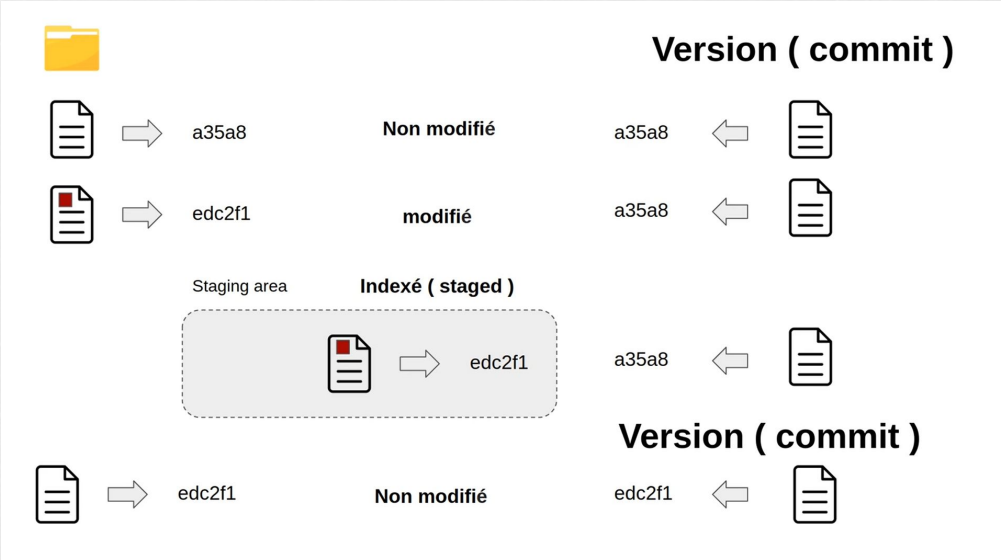
EDW4 - 2

- (th) Algorithme de hachage
  - ***SHA1***
  - Fichier => ***SHA1*** => code (40 caractères)



EDW4 - 2

- (thp) sur le terminal :
  - ***git status***
  - Modifier fichier
  - ***git commit***



EDW4 - 2

Important :

- Un fichier non indexé, ne fera pas partie du commit.

03

Git Ignorer

EDW4 - 3

- (th) Les fichiers qu'on ne veut pas suivre.
- (thd) Créer un fichier perso.json
- (thd) Ajouter le fichier ***.gitignore***
- (thd) Indexer le fichier .gitignore
- (thd) Le voir dans le ***staged***

04

Les différences

EDW4 - 4

- (th) Le ***working directory***
- (th) L'état d'un fichier dans l'***index***
- (th) L'état du fichier tel qu'il est présent dans le dernier ***commit***
- (thd) ***git diff*** (?)
    - Modifier le fichier
    - ***git diff***
    - ***git add leFichier***
    - ***git diff --staged***

Exercice

- Avec les lignes de commandes :
  - Créez un répertoire ***exo1***
  - Entrez dans le répertoire
  - Créez un fichier ***exo1.txt***
  - Inscrivez la date d'aujourd'hui dans ***exo1.txt*** (dans le format que vous désirez)
  - Initialisez ***Git*** dans votre répertoire.
  - Affichez le status de votre environnement de travail
  - Ajoutez ***exo1.txt*** dans l'environnement ***stagged***
  - Faites votre premier ***commit***
  - Créez un dossier nommé ***confidentiel***
  - Créez un fichier nommé ***acces.json*** dans le répertoire ***confidentiel***
  - Ajoutez les étapes nécessaires pour que ***Git*** ignore votre répertoire ***confidentiel*** et son contenu
  - Faites le commit de votre nouvelle version du répertoire ***exo1***.

05

Git commit...

EDW4 - 05

- Un ***commit*** c'est :
  - Un nom (***hash***)
  - Un Auteur
  - Une date
  - Un message
  - Liste de ***hash*** de tous les fichiers.
- Un ***commit*** est un fichier, qui a comme non un ***hash***.
  - Tree
  - Parent
  - Author
  - Committer
  - Message
- ***Tree*** est un fichier, aussi représenté par un ***hash***.
  - ***blob***

EDW4 - 05

- (thd) Il existe trois types d'objet ***Git*** :
  - Les ***blob***
  - Les ***commits***
  - Les ***trees***

```
Find .git/objects -type f
```

**** Find*** est une commande permettant de rechercher des fichiers et des dossiers.
****** Nous lui donnons les options ***-type f*** pour indiquer que nous cherchons des fichiers réguliers.
**** .git/objects*** permet de préciser que nous cherchons uniquement les fichiers dans ce dossier.

- Chaque objet a un ***hash SHA-1*** qui sera la clé si on considère Git comme une base de données clés / valeurs.
- Dans le dossier ***.git/objects***, chaque objet est inclus dans un dossier qui a pour nom les deux premiers caractères de son ***hash***, et le nom du fichier contient les 38 caractères restants de son ***hash***.
- Le contenu de ces objets est compressé avec la librairie ***zlib*** qui utilise l'algorithme de compression ***defalte***.

EDW4 - 05

- Il est possible de voir ce que contient n'importe quel objet Git avec la commande suivante :
  - ***git cat-file -p "hash"***
- Il est aussi possible de voir le type d'objet représenté par le hash :
  - ***git cat-file -t "hash"***
- Tout contenu en ***Git*** est soit un arbre, soit un ***blob***.
  - Les ***blobs*** (***Binary Large Objects***) sont des objets binaires qui peuvent être n'importe quoi : des fichiers de code, des images, des vidéos...
  - Les arbres (ou trees) correspondent à un répertoire et permettent à Git de stocker plusieurs fichiers ensemble.
  - Un arbre contient une ou plusieurs entrées, chacune étant le ***hash*** d'un ***blob*** ou d'un autre arbre avec ses droits d'accès, son type et son nom de fichier associé.

EDW4 - 05

Exercice (dirigé) :

- À partir d'un répertoire vide, ne faisant pas partie d'un projet Git.

```
echo 1 > fichier1.txt
echo 2 > fichier2.txt
mkdir dossier
echo 3 > dossier/fichier3.txt
```

- Nous indexerons tous les objets, puis nous afficherons les objets créés :

```
git add.
find .git/objects -type f
```

- Que se passe-t-il si nous créons notre premier ***commit*** ?
  - Nous déterminons, en faisant ***git cat-file -t "hash"*** pour chaque, que nous avons toujours trois ***blobs***, mais également deux ***trees*** et un ***commit***.
- Pour voir le contenu du ***commit*** :
  - ***git cat-file -p master***

06

Suppression

EDW4 - 06

- Effacer des fichiers avec ***Git*** :

```
rm fichier.txt
git status
git add fichier.txt
git status
```

- Raccourci :

```
git rm test.txt
git status
```

- Effacer de l'environnement ***stagged*** seulement :

```
git rm --cached fichier
```