**Les commande de git :**

commande git init « nom\_projet » est utilisée pour initialiser un dépôt Git dans un répertoire appelé nom\_projet.

*1. touch <nom\_fichier>*

La commande touch est utilisée pour créer un fichier vide. Si le fichier n'existe pas, touch va le créer. Par exemple, touch mon\_fichier.txt va créer un fichier vide nommé mon\_fichier.txt dans le répertoire courant.

*2. git status*

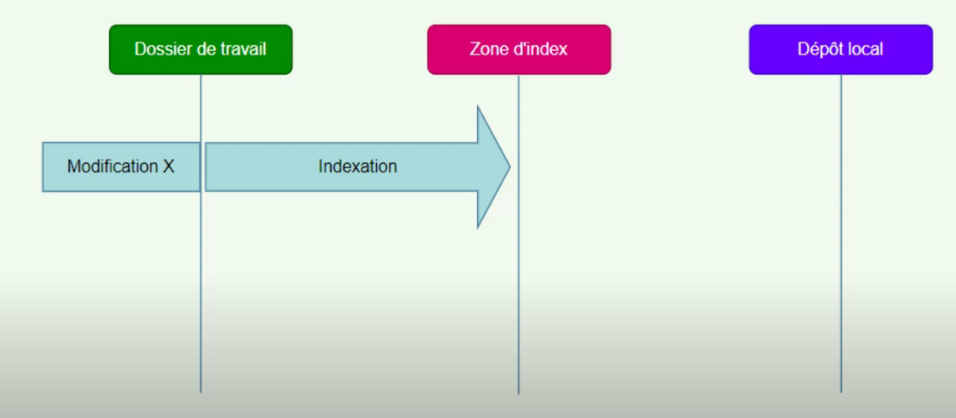
La commande git status permet de voir l'état actuel du dépôt Git. Elle vous indique :

\* Les fichiers modifiés qui n'ont pas encore été ajoutés à l'index (zone de staging).

\* Les fichiers qui sont dans l'index (en attente de commit).

\* Les fichiers non suivis (fichiers qui ne sont pas encore dans le dépôt Git).

\* Le branchement actuel.

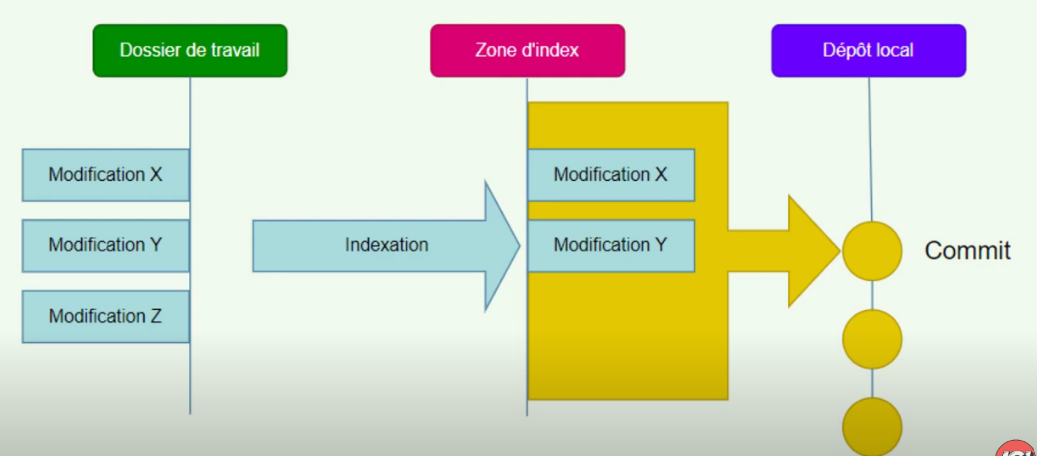


*. \*git add "nomfile"\* :*

Cette commande ajoute un fichier spécifique, nommé ici "nomfile", à l'index de Git (la zone de staging). Cela signifie que vous indiquez à Git que vous souhaitez inclure ce fichier dans le prochain commit. Le fichier n'est pas encore sauvegardé dans l'historique du projet, mais il est prêt à l'être.

*git add .*

Cette commande est utilisée pour ajouter tous les fichiers modifiés (y compris les nouveaux fichiers non suivis et les fichiers déjà suivis) à l'index (ou staging area) de Git.



\*git commit -m « message »\* :

Après avoir ajouté des fichiers avec git add, la commande git commit enregistre ces changements dans l'historique du dépôt. Un commit crée un instantané du projet à un moment donné, avec un message décrivant les modifications apportées avec l’utilisation l'option -m pour spécifier le message directement en ligne de commande (par exemple : git commit -m "Message du commit").

*git log* : Affiche l'historique des commits dans le projet.

**De qoi est constitué une commit :**

=>SHA-1

=>Ensemble des modification

=>Un message qui decrivants la commit

=>Information auteur

=>date creation

*git diff* : Affiche les différences entre les fichiers modifiés dans le répertoire de travail et ceux dans la zone de staging.

*git diff "nomfile" :* Affiche les différences spécifiques pour un fichier donné.

*\*git checkout sha-1\* :*

La commande \*git checkout sha-1\* permet de \*basculer vers un commit spécifique\* en utilisant l'ID unique du commit (appelé \*SHA-1\*). L'ID SHA-1 est une chaîne de caractères hexadécimaux qui identifie de manière unique chaque commit dans l'historique du dépôt.

*\*git show SHA-1\* :*

La commande \*git show SHA-1\* affiche les \*détails d'un commit\* spécifique, en utilisant son ID SHA-1. Elle montre des informations détaillées sur ce commit, telles que :

- La date et l'auteur du commit.

- Les modifications apportées (ajouts, suppressions, modifications de fichiers).

*\*git log\** :

La commande \*git log\* affiche l'historique des commits dans le projet. Elle montre tous les commits effectués, avec des informations détaillées sur chaque commit, comme :

- L'ID du commit (SHA-1).

- L'auteur du commit.

- La date du commit.

- Le message de commit.

\*git log --oneline\* :

La commande \*git log --oneline\* affiche un résumé des commits, chaque commit étant affiché sur une seule ligne avec un identifiant court (SHA-1 abrégé) et le message de commit.

- Cette commande est utile pour avoir une vue d'ensemble rapide de l'historique des commits.

*\*git log -n 1\* ou \*git log --max-count=1\** affiche \*le dernier commit\* effectué.

*\*git log --author "baghbagh"\** :

La commande \*git log --author "baghbagh"\* permet de filtrer l'historique des commits et d'afficher uniquement ceux réalisés par un auteur spécifique. Dans cet exemple, cela affiche tous les commits effectués par un auteur dont le nom (ou une partie de son nom) est \*"baghbagh"\*.

- Cela est utile pour examiner les commits d'un utilisateur particulier dans un projet.

.gitignore\* :

Le fichier \*.gitignore\* est un fichier spécial dans un dépôt Git qui permet de \*spécifier les fichiers ou répertoires que Git doit ignorer\*. Il est utilisé pour éviter de suivre certains fichiers qui ne doivent pas être inclus dans le dépôt.

La commande cat .gitignore est utilisée dans un terminal pour afficher le contenu du fichier .gitignore dans un dépôt Git. Le fichier .gitignore contient une liste de fichiers et de répertoires que Git doit ignorer lors du suivi des modifications dans le dépôt.

\*git mv name\_oldfile name\_newfile\* :

La commande \*git mv\* permet de \*renommer un fichier ou de déplacer un fichier\* dans un dépôt Git. Elle est équivalente à une combinaison de commandes mv dans le terminal et git add. Cette commande indique à Git que vous avez déplacé ou renommé un fichier, et elle met à jour la zone de staging pour que vous puissiez commettre ce changement.

\*git commit --amend -m "message"\* :

La commande \*git commit --amend\* permet de \*modifier le dernier commit\*. Cela est utile si vous avez oublié d'ajouter un fichier au dernier commit, si vous voulez modifier le message de commit, ou si vous souhaitez ajuster d'autres aspects du commit.