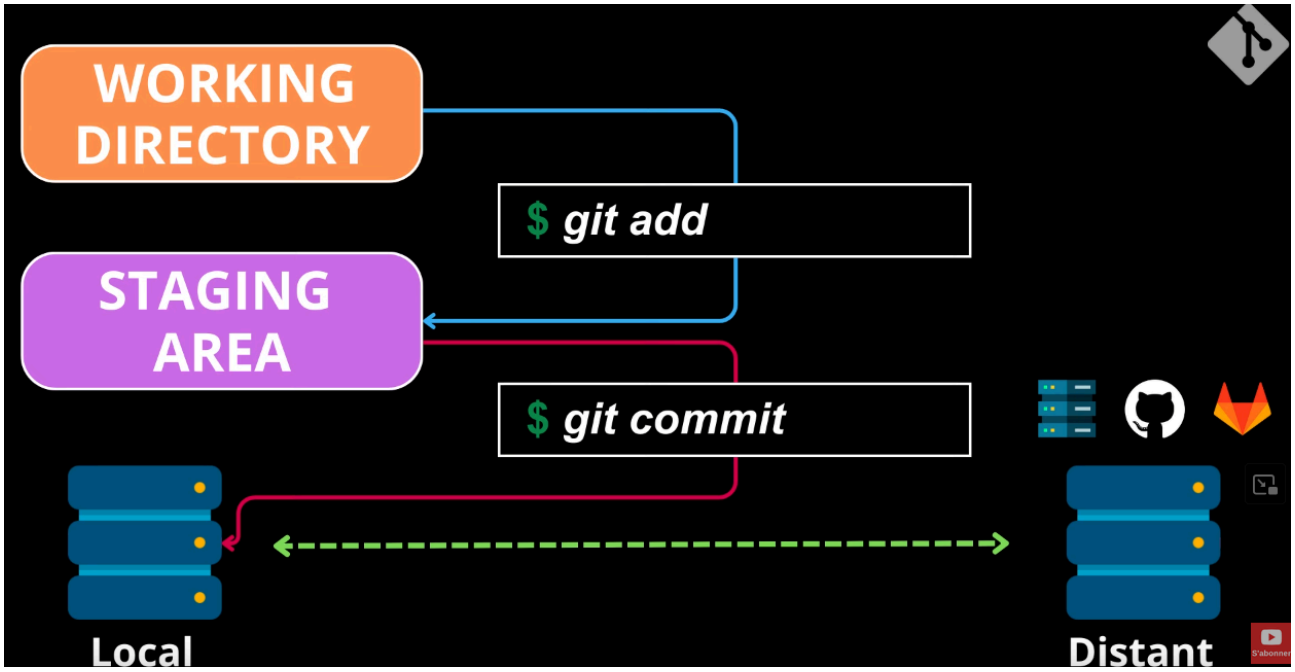
Fonctionnement :
Git permet de suivre les modifications sur les fichiers au fil du temps (versioning, on garde l'historique). Il y a un dépôt local et distant (github, gitlab,…).

Ensuite avec un commit on créer un point de sauvegarde en local
Une fois que l'on a fait les modifications que l'on voulait sur un fichier on prend une photographie des fichiers (un snapshot) de l'état des fichiers à un moment précis cela se fait lorsque l'on met le fichier dans le **staging area** assurant ainsi un historique et la possibilité de revenir à un état antérieur

Working directory : endroit où les fichiers résident et sont modifiés (dans l'explorateur de fichier)



Commandes utiles :

git add : permet d'ajouter un ou des fichiers à la staging area (add . : permet d'ajouter tous les fichiers)
git clone : permet de copier un dépôt distant sur un dépôt local
git status : montre l'état des fichiers dans le working directory (« changes not staged for commit ») et dans le staging area (« changes to be commited ») => très utile pour voir les fichiers qui sont trackés ou non

Les branches permettent d'isoler des modifications du code (développement d'autres fonctionnalités ou debug, ou test) une fois le travailler terminer sur ces branches on peut fusionner les branches

git pull : permet de récupérer les modifications faite sur un dépôt distant et de les fusionnées localement
git commit -m “message pour décrire le commit”
git push <dépôt_distant> <nom_branche> : permet d’envoyer les modifs locales vers le dépôt distant
exemple : git push -u origin main (pousse ce que l’on a en local sur github en spécifiant le nom de la branche distante grâce à l’option -u), il faut spécifier la branch sur laquelle on push si on veut push sur une autre branch

git init : créer un répertoire git vide (pour lancer le projet)
git add <nom_fichier> ou <.> : pour ajouter les fichiers dans le dossier dans lequel on est dans la staging area
git remote add origin <lien_du_ dépôt_distant> : ajoute notre dépôt disant

git checkout <nom_branche> : permet de changer de branche
git merge <nom_branche> : la branche sur laquelle on est va être mise à jour avec les commits de la branch spécifier en commande
git branch : pour lister/créer/supprimer des branches (option -a : affiche toutes les branches - en vert/blanc branches locals et rouge les branches distantes, la branche surlaquelle on se trouve aura une étoile (*))
git merge : permet de fusionner des branches

Les branches permettent d’isoler des modifications du code (développement d’autres fonctionnalités ou debug, ou test) une fois le travailler terminer sur ces branches on peut fusionner les branches

git log (permet de voir tous les commits faits sur le projet du plus récent au plus ancien)

Utiliser un fichier **gitignore** permet de lister tous les fichiers que git doit ignorer dans les commit

Par convention :
- le nom du répertoire git doit être le même en local et remote
- le répertoire doit contenir un readme (infos général sur le dépôt
- la branche principal se nomme “main” ou “master” (git branch -M main ; renomme la branche sur laquelle on est en “main”)

PS : utilisé les lignes de commandes est intéressant pour débuter/apprendre à utiliser git

ressources : https://www.youtube.com/watch?v=gGKZLfPYORs&t=302s, https://www.youtube.com/watch?v=A5_kJps4qjc, https://git-scm.com/docs