Git & Github

Git is a version control system.
Git helps you keep track of code changes.
Git is used to collaborate on code.

👉 **Git** :
est un gestionnaire de versions. Vous l'utiliserez pour créer un dépôt local et gérer les versions de vos fichiers.

👉 **GitHub** :
est un service en ligne qui va héberger votre dépôt. Dans ce cas, on parle de dépôt distant puisqu'il n'est pas stocké sur votre machine.

Git est un gestionnaire de versions tandis que GitHub est un service en ligne qui héberge les dépôts Git. On parle alors de dépôt distant.

```
git config --global user.name "w3schools-test"
git config --global user.email "test@w3schools.com"
```

Cette commande définit le nom d'utilisateur associé à votre compte Git. Le nom d'utilisateur est utilisé pour identifier l'auteur des modifications dans les opérations Git telles que les commits.

git init :
initialise un nouveau dépôt Git dans un répertoire existant, permettant ainsi de commencer à utiliser Git pour gérer le code source de ce projet.

git add :
La commande git add est utilisée pour ajouter des modifications de fichiers à la zone de staging dans Git. La zone de staging est un espace temporaire où vous préparez vos modifications avant de les valider avec un commit.

Une fois que vous avez ajouté vos modifications à la zone de staging à l'aide de git add, vous pouvez ensuite créer un commit pour enregistrer ces modifications dans l'historique Git à l'aide de la commande git commit.

$ git rm --cached index.htm
rm 'index.htm'

est utilisée pour supprimer le fichier index.htm de la zone de staging

.gitignore :
Le fichier .gitignore est utilisé pour spécifier intentionnellement des fichiers et des répertoires que vous ne souhaitez pas suivre avec Git. Ces fichiers et répertoires spécifiés dans .gitignore seront ignorés lors des opérations de suivi des modifications, telles que git add et git commit.

git commit : git commit -m "Ajout de nouvelles fonctionnalités" :
git commit est utilisé pour enregistrer les modifications de fichiers ajoutées à la zone de staging dans l'historique Git, avec un message de commit qui décrit les changements effectués.

git commit -a -m "updated temp for baking instructions" :
La commande git commit -a -m "updated temp for baking instructions" effectue un commit dans Git en ajoutant automatiquement tous les fichiers modifiés et en fournissant un message de commit spécifié. (a : add to staging zone)

git diff :
La commande git diff est utilisée pour afficher les différences entre les modifications non encore ajoutées à la zone de staging (c'est-à-dire les modifications dans le répertoire de travail) et la dernière version enregistrée dans l'historique Git (c'est-à-dire la version dans la zone de staging ou dans le dernier commit).

git restore - -staged index.htm :
La commande git restore --staged index.htm est utilisée pour retirer les modifications d'un fichier de la zone de staging (également appelée index) dans Git. Cela signifie que les modifications qui ont été précédemment ajoutées à la zone de staging pour le fichier index.htm seront retirées de la zone de staging, mais les modifications resteront inchangées dans votre répertoire de travail.

git commit -a -m "updated text to free range" :
la commande git commit -a -m "updated text to free range" ajoute automatiquement toutes les modifications des fichiers déjà suivis par Git à la zone de staging, puis crée un nouveau commit avec le message de commit spécifié.

git restore "secret recipe.htm" :
La commande git restore "secret recipe.htm" est utilisée pour restaurer un fichier spécifique dans son état précédent dans le répertoire de travail à partir du dernier

commit enregistré dans Git. Cela signifie que toutes les modifications apportées au fichier depuis le dernier commit seront annulées et le fichier sera ramené à son état tel qu'il était lors du dernier commit.

git log --oneline :
La commande git log --oneline est utilisée pour afficher l'historique des commits de manière condensée, en affichant uniquement l'identifiant abrégé (hash) de chaque commit ainsi que son message de commit sur une seule ligne.
ccf48ed (HEAD -> master) changed the file name of an image
119f2bd updated text to free range
fea296e first commit - committing all files to the repository

git commit --amend -m "changed file name to Primary Logo.png" :
La commande git commit --amend -m "changed file name to Primary Logo.png" est utilisée pour modifier le message du dernier commit et, en même temps, mettre à jour les modifications apportées à ce commit.

git log -p :
La commande git log -p est utilisée pour afficher l'historique des commits avec les différences introduites par chaque commit. Cela signifie que pour chaque commit, cette commande affiche non seulement le message de commit et les informations de l'auteur, mais aussi les modifications spécifiques qui ont été apportées dans ce commit.

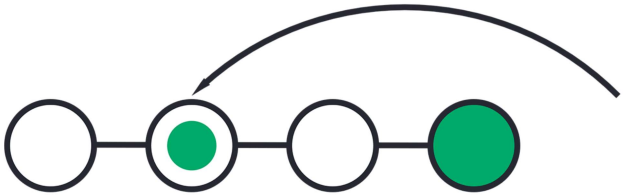
👉 Q for exit/quit

git rebase :
La commande git rebase est utilisée pour la réapplication des commits sur une autre branche. Par exemple:
git rebase master

La commande git rebase -i --root permet de démarrer une opération interactive de rebase en partant du tout premier commit de l'historique (la racine de l'historique). Cela signifie que vous pouvez réécrire ou modifier l'historique de tous les commits depuis le début du projet.

git reset :
est la commande que nous utilisons lorsque nous voulons déplacer le dépôt vers un commit précédent, en ignorant toutes les modifications apportées après ce commit.

Step 1: Find the previous commit:



Step 2: Move the repository back to that step:

```
[user@localhost] $ git log --oneline
e56ba1f (HEAD -> master) Revert "Just a regular update, definitely no accid
52418f7 Just a regular update, definitely no accidents here...
9a9add8 (origin/master) Added .gitignore
81912ba Corrected spelling error
3fdaa5b Merge pull request #1 from w3schools-test/update-readme
836e5bf (origin/update-readme, update-readme) Updated readme for GitHub Bra
daf4f7c (origin/html-skeleton, html-skeleton) Updated index.html with basic
facaeae (gh-page/master) Merge branch 'master' of https://github.com/w3scho
e7de78f Updated index.html. Resized image
5a04b6f Updated README.md with a line about focus
d29d69f Updated README.md with a line about GitHub
```

We want to return to the commit: 9a9add8 (origin/master) Added .gitignore, the last one before we started to mess with things.

git reset 9a9add8

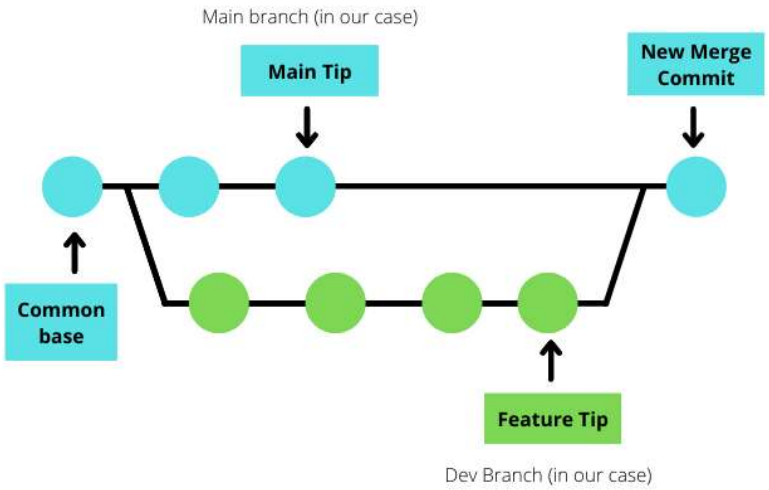
Even though the commits are no longer showing up in the log, it is not removed from Git.

👉 If you know the commit hash you can reset to it:

branch

In Git, a branch is a new/separate version of the main repository.

La fusion de branches est une opération dans Git qui combine les modifications de deux branches différentes en une seule. C'est utile lorsque vous avez travaillé sur une fonctionnalité dans une branche distincte et que vous souhaitez intégrer ces modifications dans une autre branche, comme la branche principale (par exemple, master ou main).



git merge -m "Merge fixtemp back to main" FixTemp :
La commande git merge -m "Merge fixtemp back to main" FixTemp fusionne la branche FixTemp dans la branche actuelle

git branch -d FixTemp :
supprimer la branche

git switch -c UpdateText :
(c : create) créer et switcher vers cette branche

git remote add origin URL :
Cette commande signifie que vous ajoutez un référentiel distant à votre dépôt Git local. L'argument origin est un nom d'alias conventionnel qui est souvent utilisé pour désigner le dépôt distant principal, mais vous pouvez choisir un autre nom si vous le souhaitez.

L'argument URL est l'adresse du référentiel distant que vous ajoutez à votre dépôt local. Cette adresse spécifie où votre dépôt local peut envoyer et récupérer des modifications.

En résumé, en utilisant git remote add origin URL, vous établissez une liaison entre votre dépôt local et un dépôt distant spécifique (dans ce cas, le dépôt spécifié par URL). Cela vous permet de partager votre code avec d'autres collaborateurs et de synchroniser vos modifications avec le dépôt distant.

la configuration initiale d'un dépôt Git local pour qu'il soit lié à un dépôt distant sur GitHub

1. git remote add origin https://github.com/elatlati24/KevinCookieCompany.com.git:

- Cette commande ajoute un nouveau référentiel distant à votre dépôt Git local.

- L'alias "origin" est utilisé pour se référer à ce référentiel distant. C'est un nom couramment utilisé, mais vous pouvez en choisir un autre si vous le souhaitez.

- L'URL spécifiée (`https://github.com/elatlati24/KevinCookieCompany.com.git`) est l'emplacement du dépôt distant sur GitHub.

2. git branch -M main :

- Cette commande renomme la branche principale locale de votre dépôt de "master" à "main".

- Ceci est souvent fait en réponse à un changement de convention dans la communauté Git pour éliminer l'utilisation de termes qui pourraient être perçus comme racialement insensibles, et pour refléter plus fidèlement la nature de la branche principale comme étant le "main" point d'ancrage du projet.

- **-M :** Cette option est utilisée pour forcer le renommage d'une branche même si une branche avec le nouveau nom existe déjà.

3. git push -u origin main :

- Cette commande pousse vos modifications locales vers le dépôt distant spécifié (utilisant l'alias "origin").

- La branche que vous poussez est "main" (renommée à partir de "master" dans l'étape précédente).

- L'option `-u` ou `--set-upstream` configure la branche locale "main" pour suivre la branche distante "main". Cela signifie que lors de futurs `git push` ou `git pull`, Git saura où pousser ou tirer les modifications pour cette branche.

En résumé, ces trois commandes sont souvent utilisées ensemble pour lier un dépôt Git local à un dépôt distant sur GitHub, renommer la branche principale locale pour refléter une convention de nommage plus moderne, et pousser les modifications locales vers le dépôt distant nouvellement créé.

git push --all :
La commande git push --all est utilisée pour pousser toutes les branches locales vers le dépôt distant. Cela inclut toutes les branches, y compris la branche principale et toutes les branches de fonctionnalités que vous avez créées localement.

Une Pull Request (PR) :
dans GitHub est une fonctionnalité qui permet à un contributeur de proposer des modifications à un dépôt hébergé sur GitHub. Voici un aperçu de la façon dont fonctionne une Pull Request :

1. **Création d'une branche :** Le contributeur crée une branche à partir de la branche principale du dépôt (généralement `main`). Cette branche sera utilisée pour implémenter les modifications ou ajouts de fonctionnalités.

2. **Implémentation des modifications :** Le contributeur implémente les modifications nécessaires dans cette nouvelle branche. Ces modifications peuvent inclure l'ajout de nouveaux fichiers, la modification de fichiers existants, ou toute autre tâche pertinente.

3. **Soumission de la Pull Request :** Une fois que les modifications sont prêtes, le contributeur ouvre une Pull Request. Cela se fait généralement à partir de l'interface GitHub, où il sélectionne la branche contenant ses modifications et indique la branche de destination vers laquelle les modifications doivent être fusionnées (généralement `main`).

4. **Examen et discussion :** Les autres membres de l'équipe ou les collaborateurs peuvent examiner la Pull Request, commenter les modifications proposées, poser des questions ou suggérer des améliorations. Cela permet une collaboration transparente et un feedback constructif.

5. **Tests automatiques et manuels :** Si des tests automatiques sont configurés dans le dépôt, ils seront exécutés pour s'assurer que les modifications ne cassent pas le code existant. Les tests manuels peuvent également être effectués pour vérifier le fonctionnement des fonctionnalités.

6. **Fusion de la Pull Request :** Une fois que la Pull Request a reçu l'approbation des examinateurs et que toutes les questions ou problèmes ont été résolus, elle peut être fusionnée dans la branche de destination. Cela intègre les modifications proposées dans la base de code principale du projet.

7. **Nettoyage :** Après la fusion, la branche utilisée pour la Pull Request peut être supprimée si elle n'est plus nécessaire.

Les Pull Requests sont une composante essentielle de la collaboration dans le développement logiciel moderne, car elles permettent un processus structuré pour l'examen et l'intégration des contributions de divers contributeurs à un projet.

Release :
Dans GitHub, un "release" (ou une "version") est une étape spécifique dans le cycle de vie d'un logiciel où une version spécifique du projet est marquée comme étant prête pour une utilisation générale ou une distribution

git fetch :
La commande git fetch est utilisée pour récupérer les modifications depuis le dépôt distant vers votre dépôt local, mais sans fusionner ces modifications dans votre branche actuelle.

git pull :
La commande git pull récupère les modifications depuis le dépôt distant et fusionne ces modifications dans votre branche locale.

La commande **git pull** combine deux opérations en une : **git fetch** pour récupérer les modifications depuis le dépôt distant et **git merge** pour fusionner ces modifications dans votre branche locale.

Pour changer le remote d'un dépôt Git, vous pouvez utiliser la commande git remote set-url. Voici comment faire :

Affichez les remotes actuels pour vérifier le nom du remote que vous souhaitez modifier :

- **git remote -v**

Par exemple, pour changer l'URL du remote nommé origin vers https://github.com/nouveau_nom/mon_projet.git, vous utiliserez :

- **git remote set-url origin https://github.com/nouveau_nom/mon_projet.git**

Pour pousser un commit spécifique vers le remote, vous pouvez utiliser la référence du commit avec la commande git push. Voici comment faire :

Affichez les commits de votre dépôt pour trouver le commit que vous souhaitez pousser. Vous pouvez utiliser **git log**

Repérez l'ID du commit que vous souhaitez pousser.

Utilisez la commande git push avec l'option commit_ID:branch_name pour pousser ce commit spécifique vers le remote :

git push origin commit_ID:branch_name