Versioning Git

Mounir BENDAHMANE : mnbdpro@gmail.com

Introduction :

Peer to peer = système d'information décentralisé (stockée à plusieurs endroits à la fois)

→ exemple : blockchain

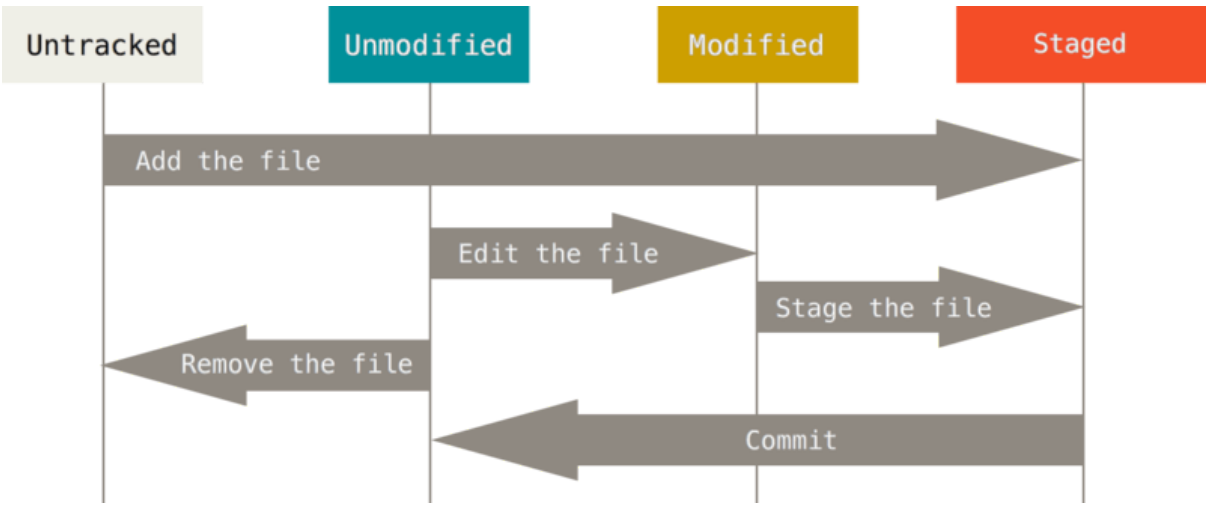
But de git : travailler efficacement à plusieurs

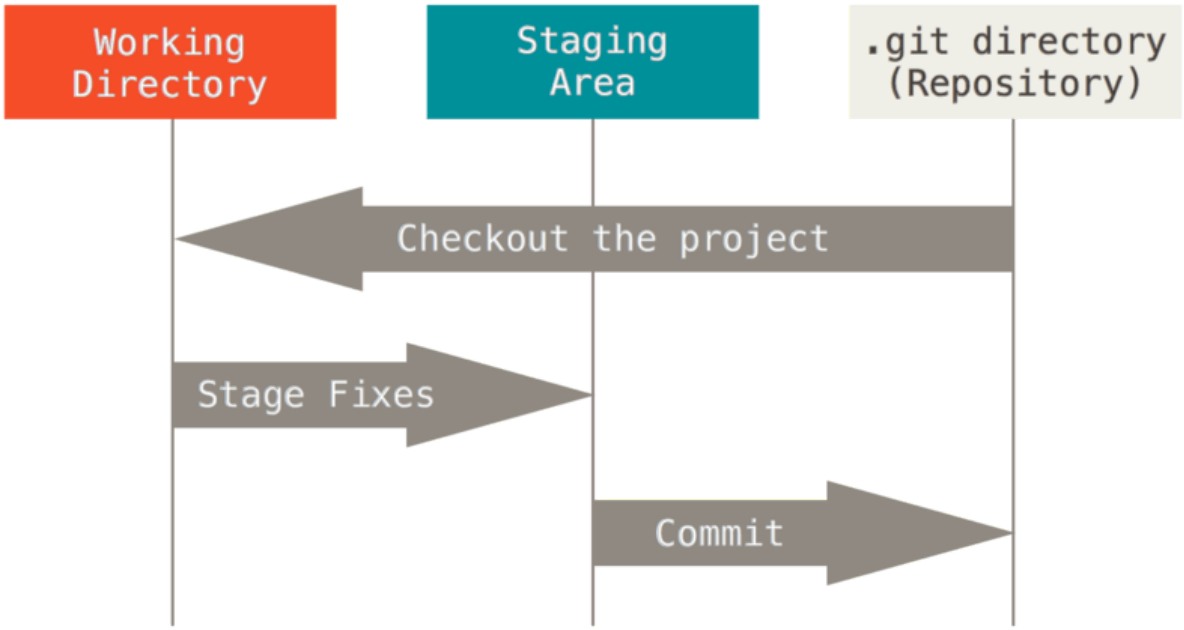
git-scm.org → documentation git

Github fournit une interface de gestion d'un repo, un espace de stockage distant pour le repo, …

Les fichiers ont trois états dans git:

- Modifié (modified) en modifiant simplement un fichier
- Indexé (staged) avec "git add"
- Validé (commited) avec git commit"





A RETENIR :

commit - pull - push (toujours git pull avant un commit pour ne pas avoir de pb de conflits)

Commandes git de base :

git init → ajouter git (.git) à un répertoire

git add → ajouter un fichier du répertoire à git

git remote add (/set-url) origin (url ssh ou http) → ajouter ou modifier la connexion à distance

git commit -m “message” → enregistrer les modifications du répertoire dans sa branche locale git (“-m” pour mettre un message ensuite)

git pull → récupérer les changements de la branche distante vers la branche locale

git push -u origin main → envoyer le code sur origin (repo distant), la branche main locale est synchronisée avec celle de origin

git branch -m Mai

git status → statut du repo en cours (fichiers non-indexés, indexés, changés, …)

git diff → voir les différences entre les changements effectués et les fichiers indexés

conflits APRES un pull (fichier distant modifié aux mêmes endroits que le fichier local)

→ git insère des informations dans le fichier (code) qui indique quels sont les changements distants, et quels sont les changements locaux, afin de pouvoir résoudre le conflit en décidant de n'en garder qu'une partie, les deux ou un mélange des deux. On fait ensuite un nouveau commit pour résoudre le conflit.

(Malt → proposer du freelance programmation)

ssh-keygen (voir doc github) → générer une nouvelle clé ssh

git commit -a → ajouter et indexer les fichiers de son repo à la volée

git commit -m → pour mettre le message en (“”)

rm -rf .git/ → supprimer le dossier caché git

git clone (url) → récupérer un repo distant et le copier dans son repo local

touch (nomfichier) → créer un fichier

git rm --cached (nomfichier) → retirer un fichier du suivi git

ou

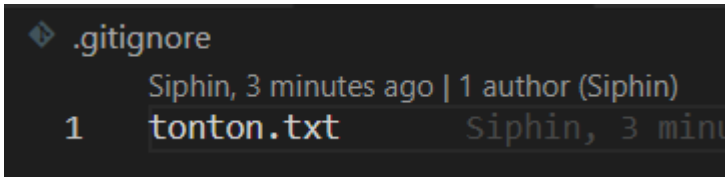
git restore --staged (nomfichier)

ou

git reset HEAD testsuivi.txt

Les fichiers de build (executables) / dépendances, ne devraient pas être dans le suivi git (ce qui ne constitue pas le code source), car ils peuvent changer en fonction de la machine (pas le code source).

→ On peut créer un fichier “.gitignore” qui fait en sorte que git ignore du suivi les fichiers choisi (récursif) :



Au début d'un projet on peut récupérer un gitignore pré-fait en fonction du projet / des technos utilisées (ex: gitignore pour les logs Java, etc)

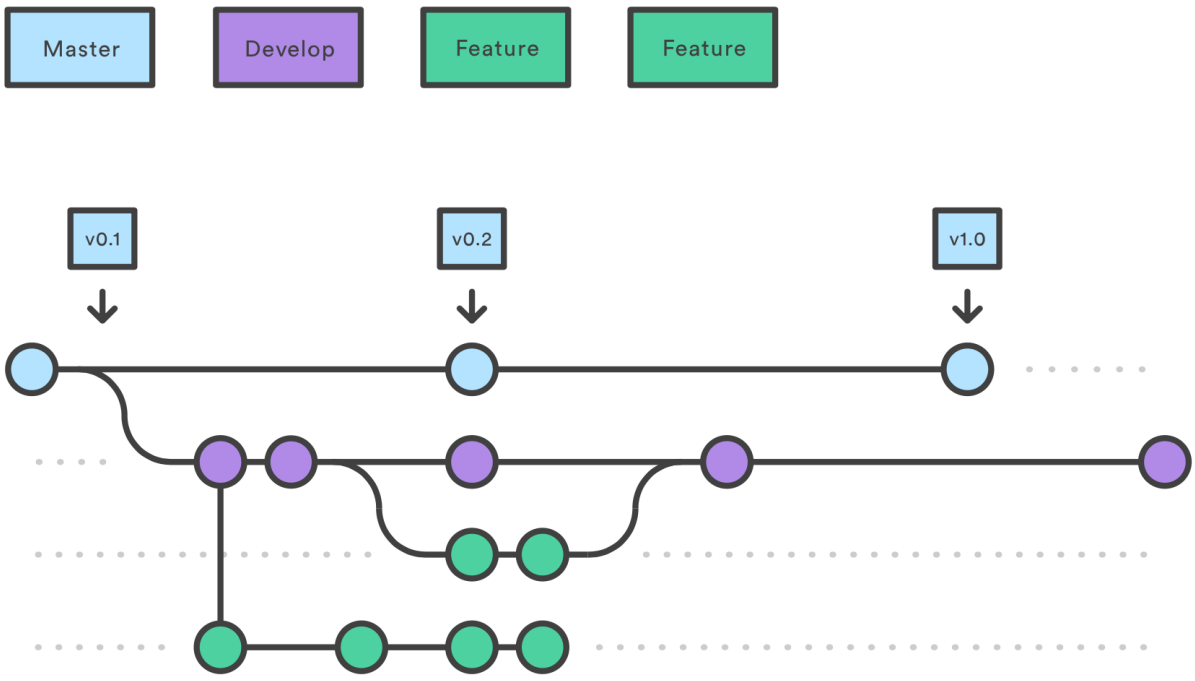
git mv (nomfichier) (nouveaunomfichier) → déplacement d'un fichier (on peut le renommer de cette manière)

HEAD = pointeur ver le commit le plus récent, de la branche de travail en cours

git log --pretty=oneline (ou --p) → logs du repo (changements) ligne par ligne (une ligne par log)

git log --graph --abbrev-commit --decorate --format=format:'%C(bold blue)%h%C(reset) - %C(bold green)(%ar)%C(reset) %C(white)%s%C(reset) %C(dim white)- %an%C(reset)%C(bold yellow)%d%C(reset)' --all → affichage des logs personnalisé (moment du changement, utilisateur…)

Branches :



Quand on crée une branche, on crée une copie du projet à partir d'un commit donné. En travaillant ensuite sur cette branche, cela n'influera pas sur le projet de base (sur la branche d'origine)

Quand on fait un commit, git se déplace sur le nouveau commit

→ une branche est un pointeur vers un commit donné

git branch (nombranche) → créer une nouvelle branche

git branch → list des branches

git checkout (nombranche) → changer de branche (HEAD pointera vers le dernier commit de la branche choisie)

git push --set-upstream origin (nombranche) → push une branche

git branch -d (nombranche) → supprimer une branche

Pour fusionner des branches : on se place sur la branche principale (ou celle qui reçoit les changements), puis,

git merge (branchesecondaire)

Lien convention des commits :

https://les-enovateurs.com/conventional-commits-details-exemples-pratiques