Versioning-Git Notes :

Git est un outil qui permet de partager la code-base et de partagé les infos.

SCM : Source Code Management

Git permet de sauvegarder son code et toutes ses versions (arborescence du projet). Cela permet une traçabilité en temps réelle.

Facilite le travail collaboratif, avec peu de conflits et facilite la gestion. De plus il offre :

1. Code fonctionnel et développement simultané
2. Création de workflow

Points importants :

1. Une machine qui exécute git possède son propre dépôt local
2. Chaque dépôt local enregistre la totalité des changements apportés
3. Développement de features possible en parallèle

P2P = on parle de système où l’information est décentralisée (qui sera renforcé si on l’utilise).

Par exemple la block chain est un système décentralisé.

((Voir umbru))

Une image contenant texte

Description générée automatiquement

Repository : espace disque ou sera stocké chaque version de notre projet git

MarkDown : langage interprété par les navigateurs

Checkout : récupérer le fichier

HEAD : pointe vers le commit le plus récent

Merge

Commit : sauvegarde les changements

Git add nomdufichier ( index le fichier)

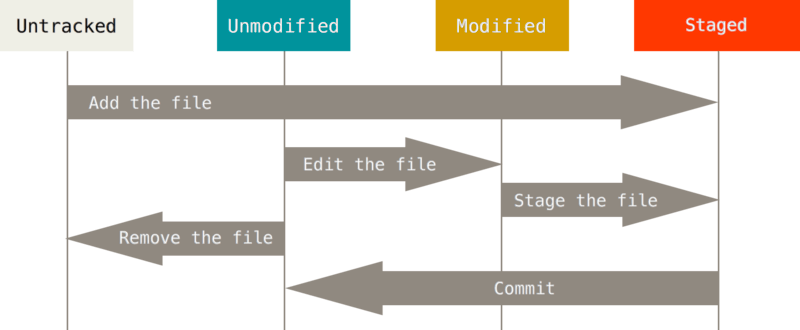
Remote add : désigne un alias

Git push -u origin main : envoie du fichier sur le repo distant

git c’est le logiciel de versioning

github c’est un site web

Voir différence git status/git diff



Une branche est une copie d’un projet à partir d’un commit, elle permet de travailler dessus sans impacter le projet

Commande pour créer une branche :

git branch nombranche

lister les branches :

git branch

changer de branche :

git checkout nombranche ( checkout –v permet de créer la branche)

Fusionner les branches :

Il faut se placer sur la branche sur laquelle on veut merge une autre branche (souvent la branche main/master). On utilise « git merge nombranche ».

Supprimer une branche :

Git branche –d nombranche

(Un camarade m’a aidé pour les notes suivantes car j’étais absents tout le jeudi).

Suivre les commandes :

Echo “# nomdurepo” >> README.md - Créer un readme.md

Git init – initialization de git

Git add README.md – ajout d’un dossier README.md

Git commit –m “first commit”

Git branch –M main – creation de la branche main

Git remote add origin [git@github.com:monmail/nomdurepo.git](mailto:git@github.com:monmail/nomdurepo.git) - ajout d’un repo distant et ajout d’un alias “origin”

Git push –u origin main – push de ma branche local sur le repos distant

Ouvrir le dossier avec vs code

Ouvrir le terminal

Ajouter des collaborateurs :

Settings 🡪 Collaborators 🡪

Récupérer un un repo git :

Copier l’url du repo sur git hub

Cmd : git clone url

Supprimer un fichier du projet :

Rm nomdufichier.format ( supprimer le fichier )

Git add monfichier.format ( il faut indexer et commit la suppression )

Git commit

Exclure un fichier du suivi git sans le supprimer du disque :

Git rm --cached nomdufichier.format

Puis commit

L’interet d’enlever le suivi sur des fichiers est car certains fichier ne sont pas voué à changer donc on ne les fait pas suivre pour sauvegarder le minimum de choses.

.gitignore 🡪 fichier dans lequel on décrit quels fichiers nous ne souhaitons pas suivre.

Retirer un fichier de la staging area :

Apres le git add il faut

Git reset HEAD nomdufichier.format

Ou

Git restore –staged nomdufichier.format

Renome un fichier avec git :

Git mv nomfichier.format nouveaunom.format ( mv déplace le fichier et on en profite pour le renommer)

Commandes utilisées :

Git diff ( affiche la différence entre l’état du fichier avant la modification et après)

Git log : historique des commit, est capable de prendre des arguments pour modifier la sortie des données

Git log –pretty=oneline permet d’fficher tout les comit sur une seule ligne

--abbrev-commit

--decorate

--format

Git commit –a : ajoute et commit

Touch : créer un fichier

Vérifier si un fichier est modifié : git status