Notes Git / GIT Hub - Versioning

Le SCM : Source Code Management

Logiciel qui permet le suivi des modifications apportées au projet courant ( GIT)

Les SCM :

Interets :

Il permet une maitrise totale du code :

* Tracabilité –
* Collaboration –
* Peu de conflits et facilité de gestion –
* Code fonctionnel et développement simultané –
* Création de workflow –

Les principaux SCM :

* GIT
* AWS CodeCommit
* Azure DevOps Server
* Subversion (apache, appelé SVM)
* Mercurial
* CVS
* Etc…

GIT :

Les mauvaises Pratiques :

* Projet Web sans SCM
* Utilization de methods dépassé ou risqués: OneDrive, réseaux Sociaux, courriers électroniques, disque dur
* Très difficile de travailler à plusieurs
* Mise en commun ; source d’erreur
* Perte de temps considérable

Notions Fondamentales :

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

Gestion des différences vs Snapshots

A chaque version git sauvegarde l’intégralité des fichiers au moment où la version est établie : snapshot permettant ainsi de revenir facilement à une version antérieure

Peer2peer

Git permet de travailler de manière collaborative peu importe le langage et la techno

Système décentralisé : peer2peer (par exemple : torrent)

SVM est un système entièrement centralisé

Avantages des systèmes décentralisé, très robuste car la data est stocké sur plein d’autres machines. Là ou système centralisé très fragile.

Repository : espace disque dans lequel seras stocké chaque version du projet.

MarkDown : langage interprété par les navigateurs

Check out : récupérer le projet

HEAD : pointe vers le commit le plus récent

Merge

Commit

Pull

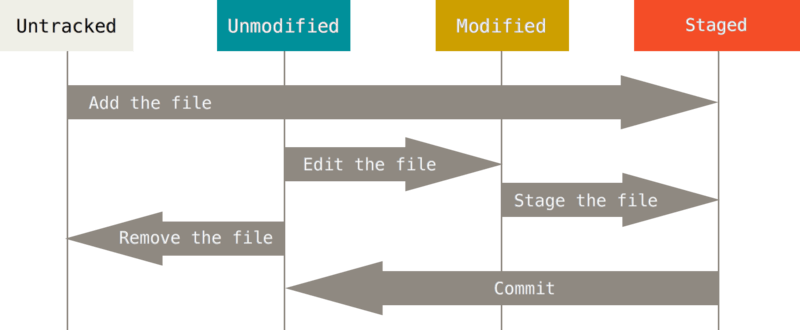
Push

Différents états d’un fichier avec git :

Indexé (staged): avec “git add”

Modifié (modified) : modifier le fichier

Validé ( commited): avec “git commit”



Une fois la modification du fichier faite il faut :

* Git add nomdufichier ( index le fichier)
* Git commit –m « message » ( commit le fichier en local )
* Git push –u origin main ( envoie du fichier sur le repo distant )

Création d’une clé ssh :

* Créer la clé localement :

ssh keygen –t ed25516 –c “mon@mail.com” ( création de clés pour s’authentifier)

passphrase : ( Mot de passe, pas obligatoire)

verify passphrase :

* Ajouter la clé sur GitHub :

Clip < ~/.ssh/id\_ed25519.pub (copie de la clé pour ajouter à git hub)

* Push sur le repo distant :

Git push –u origin main

Live Coding prof

Création d’un repository :

Git hub

Créer un repository

Mettre un nom

Public

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

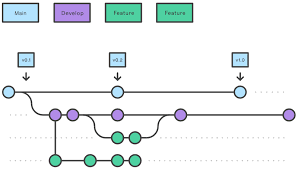
Touch : créer un fichier

Vérifier si un fichier est modifié : git status

La notion de branche :

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

un pointeur vers un commit de données



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

Lorsque l’on push vers le repo distant il est possible de créer des conflits. On peut créer des conflits en travaillant sur le même fichier et en l’envoyant en même temps. Afin d’éviter ça il faut toujours pull avant de push afin de récupérer les modifications faites par d’autres et se mettre à jour (d’où commit pull push). Quand on a un conflit git on a le choix entre accepter les changements qui se trouvent sur le repo distant ou accepter les notres ou les deux ( pas conseiller car ca mets les deux a la suite).