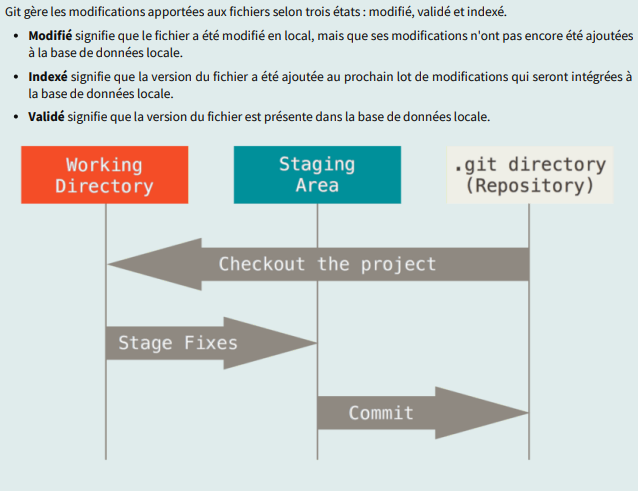
01 INTRO

Git est un DVCS Distributed Version Control System



Git repose sur un système de gestion des instantanés ; il sauvegarde tjrs le fichier entier ds sa base locale

$ git version

$ git config - -global - -list

$ git config - -global user.name «John Doe »

$ git config - -global user.email « jd@example.com »

$ git config --global help.autocorrect 1

$ echo -e "# blabla Fichier README.md\n\n Il ne contient pas grand-chose." > README.md

$ git status

$ git add README.md

$ git commit m « first commit »

02 BASES

Initialisation d’un nouveau projet :

$ git init

Clonage d’un projet existant :

$ git clone [url] [nouveau\_nom(facultatif)]

Exemple :

$ git clone <https://github.com/PHPMailer/PHPMailer.git> myPhpMailer

**Cycle de vie fichiers :**

2 états possibles d'un fichier :

- tracked -> sous suivi de version, déjà présent lors du dernier état des lieux du système

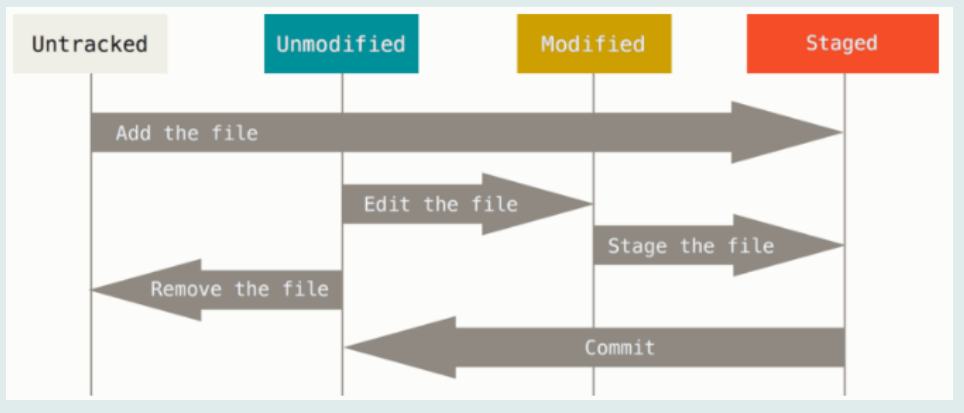
- untracked : l'inverse

Lorsqu'un fichier est suivi (tracked), il a 3 états possibles :

- unmodified

- modified

- staged (indexé) : le fichier a été modifié et ces modifs ont été indexées, mais pas encore enregistrées depuis le dernier état des lieux. Si on effectue un commit, càd une sauvegarde, seuls les fichiers indexés seront modifiés



$ git status

Une image contenant texte

Description générée automatiquement

$ git restore file => permet de restaurer un fichier modifié et pas encore passé dans la zone de staging (indexation)

$ git restore - -staged file => permet de sortir un fichier ou rep de la zone de staging (les modifs apportées au fichier ne seront pas perdues pour autant)

Pour déplacer ou renommer un fichier ou répertoire :

$ git mv docs nouveauxdocs

$ git rm file permet de supprimer un fichier de l’espace de travail et de l’index de git

$ git rm file - -cached permet de supprimer de l’index de git (staging) sans pour autant supprimer de l’espace de travail

Memo :

La commande git restore permet de rétablir l'état d'origine d'un fichier.

Pour retirer un fichier de la zone de transit, on peut ajouter l'option --staged à la commande git restore afin de rétablir son statut d'origine (modifié/non-suivi) sans perdre les modifications effectuées.

Pour déplacer ou supprimer un fichier, il est préférable d'utiliser les commandes git mv ou git rm.

Syntaxes à adopter pour .gitignore :

* # permet d'ajouter un commentaire
* \* permet d'ignorer tous les éléments répondant à un certain pattern, par exemple \*.docx ignore tous les documents Word
* ! permet d'ajouter une exception à l'une des règles préalablement écrites, par exemple !documentation.docx
* /fichier-temporaire.tmp permet d'ignorer le fichier fichier-temporaire.tmp situé à la racine du projet
* build/ ignore tous les éléments situés dans le répertoire **build**
* **exemples ici :** <https://github.com/github/gitignore>

# On ignore les fichiers .txt

\*.txt

# On conserve important.txt

!important.txt

# On ignore le répertoire /vendors

vendors/

# On ignore les fichiers pdf du répertoire uploads

uploads/\*\*/\*.pdf

NB : si on travaille sur des projets similaires qui doivent ignorer tjrs les m répertoires, alors possibilité d’avoir un rep commun (ex : rep utilisateur) et de le déclarer au niveau de git config :

git config --global core.excludesfiles ~/.gitignore

Sauvegarder les fichiers volumineux :

<https://git-lfs.github.com/>

Comparer des versions de fichiers avec git diff :

attention la comparaison concerne une version du fichier committé, et une version en cours de modification mais pas encore ajoutée à l’index.  
Si toutefois on veut comparer avec une version déjà à l’index, il suffit d’ajouter - -staged ($ git diff - -staged)

$ git diff --color-words permet de visualiser les éléments modifiés mot à mot, et non par ligne

$ git diff nom\_fichier -> pr cibler un fichier précis

Remettre un répertoire à son état d’origine :

git reset --hard origin/master (ou main)

On peut consulter le log des versions enregistrées dans le repository via

$ git log

03 DEPOT DISTANT (repository)

<https://docs.github.com/en/authentication/connecting-to-github-with-ssh>

SYNCHRONISER UN PROJET LOCAL AVEC UN DEPOT DISTANT

($ git remote -v permet de lister les dépôts)

Une fois créé le dépôt sur github, récupérer son adresse (ssh ou https) et utiliser git remote add pour l’ajouter localement

$ git remote add origin [git@github.com:patrickcharda/GITraining.git](mailto:git@github.com:patrickcharda/GITraining.git)

Origin est par convention le nom donné au dépôt distant

$ git remote -v -> on voir maintenant le dépôt référencé

$ git remote show origin -> détails d’un dépôt particulier (origin ou autre)

* Travailler avec un dépôt Git distant implique de toujours récupérer la dernière version du projet grâce aux commandes git fetch et git merge ou git pull, avant de pousser ses propres modifications locales vers le dépôt distant grâce à la commande git push.
* Lorsque l'on travaille sur un projet existant, il faut utiliser la commande git clone pour récupérer le projet localement : de cette manière, un dépôt local automatiquement lié au dépôt distant est créé.

GIT FETCH

La commande git fetch compare l'état du projet local et récupère tous les commits manquants de la branche courante depuis le dépôt distant.

Cette commande ne télécharge pas les fichiers modifiés, mais uniquement les metadata permettant de connaître les différences existantes.

De cette manière, il est possible d'être informé des modifications à venir et de ne décider de les intégrer, avec la commande git merge, qu'après s'être assuré qu'elles n'auront pas d'impact sur les développements en cours.

GIT MERGE

La commande git merge va quant à elle intégrer localement les modifications présentes sur le dépôt distant.

GIT PULL

La commande git pull permet d'effectuer les commande git fetch et git merge en une seule fois.

Ainsi, lorsque git pull est effectuée, les modifications présentes sur le dépôt distant sont immédiatement intégrées au projet local.

GIT PUSH

Une fois les développements d'une branche terminés, il faut mettre à jour le dépôt distant afin que les autres membres de l'équipe puissent récupérer les modifications apportées.

Pour cela, il va falloir pousser les commits réalisés en local vers le dépôt distant grâce à la commande git push.

Cette commande prend en paramètres le nom du dépôt distant suivi du nom de la branche cible à mettre à jour.