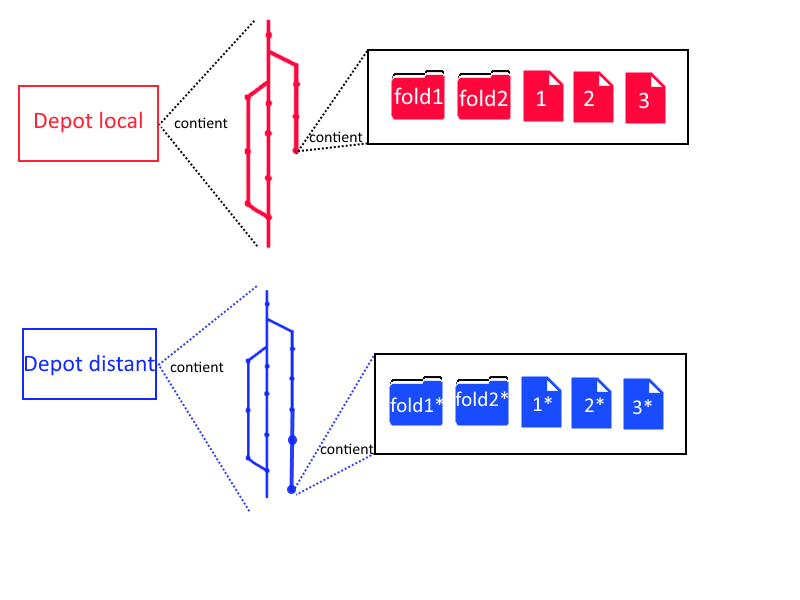
# Tuto pratique git

## Introduction



Dans git, votre dépôt local coexiste avec un dépôt distant.

Les deux dépôts sont indépendants mais peuvent être synchronisés (par exemple en utilisant git pull et git push).

Chaque dépôt contient des versions organisées en arborescence.

Les versions correspondent à aux versions du répertoire (dans lequel vous avez créé votre dépôt) ainsi que de son contenu.

## Manipuler des dépôts

### git init

Créer un dépôt local dans le répertoire courant. Cela se traduira par la création par git d’un fichier nommé « .git » dans le répertoire courant qui contiendra toutes les infos liées à votre dépôt.

### git clone

Clone le dépôt distant indiqué dans le répertoire local. C’est-à-dire créé un dépôt local (pas besoin de git init donc) identique au dépôt distant indiqué

### git remote

Consulter/Editer la liste des dépôts distants enregistrés

Git remove –v : affiche la liste des dépôts distants ainsi que leur URL

Git remote add <nom> <url> : enregistre un dépôt distant présent à <url> sous le nom indiqué <nom>

Git remote remove <nom> : supprime le dépôt <nom> de la liste

### git fetch

Renvoie les différences entre votre dépôt local et le dépôt distant SANS EFFECTUER DE MODIFICATIONS (à contrario de git pull).

Git fetch <branch> : différence entre la dernière version sur de la branche indiquée sur le dépôt local et celle sur le dépôt distant

### git pull

Fusionne le dépôt distant vers votre dépôt local. Des conflits peuvent se produire.

Git pull <branch> : git pull pour une seule branche

### git push

Fusionne le dépôt local vers le dépôt distant. Attention à bien faire en sorte de disposer d’une version à jour (en faisant un git pull) avant de push.

Git push <remoteName> <branchName> : Ne push que la branche indiquée

## Manipulation de l’arborescence

### git log

Afficher l’historique des commit

### git branch

Afficher les branches du dépôt local

### git checkout

Permet de revenir à un ancien commit.

### git merge

Permet de fusionner deux branches.

## Manipulation de versions

### Git checkout

Git checkout permet aussi de revenir à une version antérieure d’un fichier.

Ex : git checkout file.txt : Fait revenir file.txt à son état lors du dernier commit

### Git rm

Permet de supprimer un fichier et de commit sa suppression.

En effet, si on supprime un fichier sans utiliser git rm, celui-ci continue d’exister dans le dépôt.

On doit alors utiliser git rm sur le fichier que l’on vient de supprimer.

A noter qu’en utilisant Git commit –a, git supprimera automatiquement du dépôt les fichiers supprimés par l’utilisateur.

Git rm –r : Permet de supprimer un dossier et son contenu.

### git status

Affiche le status du prochain commit en répartissant les fichiers en plusieurs catégories :

-untracked : pas encore suivi par git (par exemple nouveau fichier)

-unstaged : fichier dont les modifications n’ont pas été ajoutées au prochain commit

-staged : fichiers ajoutés au commit

### git add

Ajoute un fichier au commit.

### git reset

Permet de retirer des fichiers du prochain commit.

### git commit

Créer sur la branche courante du dépôt local une nouvelle version contenant les modifications appliquées aux fichiers dans le commit.

### git diff

Affiche les différences entre deux versions

Git diff –cached : différence entre le commit « en cours » et le dernier commit effectué

Git diff HEAD : (HEAD = mot clé pour désigner la branche courante) Différences entre l’état du répertoire actuel et le dernier commit de la branche courante

Git diff <branche> : Différence entre le répertoire actuel et le dernier commit de la branche indiquée.

Mise en situation

https://try.github.io/levels/1/challenges/1