**Définition**

**Git** est un outil en lignes de commandes gratuit et open source développé par des utilisateurs Linux pour faciliter le développement informatique.

**Terminologie**

**Remote:** Un remote est un serveur à distance(serveur distant) qui utilise la technologie Git. Elle permet d’héberger son code, de le partager avec d’autres utilisateurs et aussi de collaborer simultanément sur un même projet à distance.

Exemples: Github, GitLab, BitBucket

**Dépot Git:** Un projet, un dépôt git est un dossier ou répertoire qui utilise Git

**Dépôt local:** C’est un dépôt git qui se trouve sur une machine locale(un simple ordinateur )

**Dépôt remote:** c’est un dépôt git qui n’est pas local, qui se trouve plutôt sur un serveur git distant

**Commit:**  Dans un dépôt git, un commit est un point du projet ou on peut retourner en cas d’erreur, de bug ou de changement dans le projet

**Branche :** C’est une référence, un pointeur léger et mobile vers l’un des commits du dépôt

Au fur et à mesure des validations, la branche master pointe vers le dernier des commits réalisés. À chaque validation, le pointeur de la branche master avance automatiquement.

**Commandes Git**

* Installation de Git

sudo apt install git

* Vérification de la version de Guit

git –version

* Initialisation d'un dépôt git

créer un répertoire nommé GuitTraining

naviguer dans le répertoire créé

exécuter la commande

git init

Cette commande permet d’initialiser un dossier pour en faire un dépôt git local

* Vérifier le status des dossiers et fichiers de son dépôt

git status

Cela permet de suivre la trace des différents fichiers et répertoires, autrement dit, de voir s'il y en a, les nouveaux fichiers ou dossiers créés et non ajoutés au dépôt, et aussi les changements opérés qui ne sont pas encore commités.

* Ajouter des fichiers ou dossiers existants à son dépôt

git add <file\_or\_folder\_name>

Cette commande ajoute file\_or\_folder\_name au dépôt.

Pour ajouter tous les fichiers et répertoires au dépôt, utiliser la commande:

git add .

ou

git add –all

ou encore

git add -A

C faisant le ou les fichiers et répertoires concernés sont staged

Vous pouvez utiliser ensuite git status pour vérifier

* Faire un Commit

git commit -m <commit\_message>

La commande git commit utilisé comme option m qui permet de spécifier le message du commit

Exécuter la commande

git commit -m “Ceci est mon premier commit”

* Configurer son git

Affichage des informations de configuration

git config –list

Cette commande affiche les informations de configuration du dépot

On peut filtrer ces informations pour afficher uniquement celle concernant l’utilisateur du dépôt

git config –list | grep user

qui équivaut à

git config –list | findstr user

Modification des informations de configuration

Les commandes

git config user.name <username>

et

git config user.email <email>

permet de modifier ou de définir respectivement le nom d’utilisateur et l’email pour le dépôt courant

Pour une configuration globale du nom d’utilisateur pour tous les dépôts locaux git, il faut ajouter l’option global :

git config –global user.name <username>

git config –global user.email <email>

git config –global –list | findstr user

* Remote

git remote

Cette commande permet de lister le nom des différents remote qui sont associés à son dépôt local

git remote -v

Avec l’option v(--verbose au complet), on peut afficher en plus, entre autres l’url des remotes associés au dépôt

git remote show <remote\_name>

Cette commande permet de montrer les informations concernant une branche donnée, dont l’url, la branche principale, etc

git config –get remote.<remote\_name>.url

ou encore

git remote get-url <remote\_name>

Cette commande permet d'afficher l’url d’un remote

git remote add <remote\_name> <remote\_url>

Cette commande permet de relier un remote dont l’url et le nom sont donnés à son dépôt local. En effet, on peut associer plusieurs remotes à un même dépôt local

Exécuter les commandes :

git remote remove <remote\_name>

Permet de supprimer un remote dont le nom est spécifié

git remote rename <old\_remote\_name> <new\_remote\_name>

Permet de renommer un remote

git remote get-url <remote\_name>

Cette commande permet d’obtenir l’url du remote dont le nom est spécifié

git remote set-url <remote\_name> <new\_remote\_url>

Cette commande permet de modifier l’URL d’un remote dont le nom est spécifié

* Travailler sur des branches

La branche git par défaut est master et on peut avoir avoir plusieurs braches dans un même dépôt git

git branch

Cette commande permet de lister les différentes branches du git et de marquer(en vert et avec une astérisque devant) la branche courante

Créer une nouvelle branche

git branch <branch\_name>

Permet de créer une nouvelle branche git avec le nom spécifié

Renommer ou déplacer une branche

git branch -m [<old\_branch\_name>] [<new\_branch\_name>]

Permet de renommer ou de déplacer une branche. Si le premier paramètre <old\_branch\_name> n’est pas spécifié, la branche courante est renommée avec le nouveau nom spécifié

Copier une branche

git branch -c [<old\_branch\_name>] [<new\_branch\_name>]

Permet de copier une branche dans une autre

Supprimer une branche

git branch -d [<branch\_name>]

Permet de supprimer la branche dont le nom est spécifié

NB

* Avec c et m l’opération concernée n’est pas effectuée lorsque la branche <new branch name> existe
* Pour forcer l’opération(copie, suppression, modification, déplacement), il faut utiliser la lettre majuscule correspondante.Par exemple:
* C et M permettent de forcer l’opération
* d et D suppriment la branche même si celle ci n'est pas fusionnée
* Pousser un remote

git push -u [<remote\_name>] [<branch\_name>]

Permet de pousser les commits sur un remote donné

Les arguments <remote\_name> et <branch\_name> sont optionnels et spécifient respectivement le nom du remote et le nom de la branche à pousser

La commande git push peut être également utilisée pour supprimer une branche d’un remote .Pour cela, utiliser:

git push <remote\_name> –delete <bracnh\_name>

ou simplement

git push <remote\_name>:<branch\_name>

* Cloner un dépôt git distant(remote)

Cloner dans le répertoire courant

git clone <remote\_url>

Cette commande permet de créer une copie locale du dépôt remote dont l’url est spécifié dans le répertoire courant

Exécuter la commande

Cloner dans un répertoire donné

git clone <remote\_url> chemin

Ex

git clone

* De l’aide

git <command> –help

* Historique des commits:git log, git shortlog et git show

git log

Cette commande affiche les détails sur les commits effectués sur le dépôt, à savoir le nom de l’utilisateur, son email, le message du commit, la date du commit, la branche concernée, l’identifiant du commit …

git log -n

Donne la liste des n derniers commits

git shortlog

Groupe les commits par noms d’utilisateur et par nombre de commits

log et shortlog peuvent être utilisés avec certaines options.

L’option –numbered(-n en abrégé) regroupe les commits par nom d’utilisateur et par ordre décroissant de nombre de commits

L’option –summary(-s en abrégé) donne les utilisateurs ayant fait des commits par nombre de commits effectués

L’option –email(-e en abrégé) regroupe les commits par nom d’utilisateur, email et par nombre de commits

L’option –patch(-p en abrégé) donne en plus les changements intervenus dans les fichiers concernés par les commits

L’option –start donne le nombre de fichiers ayant changés dans le commit ainsi que le nombre d’insertions et de suppressions concernant chacun de ces fichiers

NB:On peut combiner ces options

EX

Filtrer l’historique des commits

L'option after permet de filtrer les commits par date

git log –after “3 years ago”

git shortlog –after “2 months ago”

git log –after “2022-11-29”

git log –after “7 days ago”

git log –after ‘2 hours ago’

git shortlog –after ‘16 minutes ago’

git shortlog –after ‘20 seconds ago’

L’option author permet de s'intéresser seulement aux commits d’un utilisateur donné

git log –author <username>

L’option grep permet de s'intéresser uniquement aux commits contenant une chaîne de caractères donnée

git log [option(s)] –grep <my\_str>

git shortlog [option(s)] –grep <my\_str>

Ex

git log –grep “implementation”

git shortlog -n –grep “copie”

On peut également s'intéresser aux différents changements(insertions, suppression, etc) dans les fichiers concernés par un commit donné. Pour cela, il faut utiliser la commande git show. C’est le diminutif de git log –patch

git show

Sans argument, les informations concernant le dernier commit sont affichées

git show <commit\_id>

Cette commande permet d’afficher ces mêmes informations concernant le commit dont l’id est spécifié

NB: Les commandes shortlog et show peuvent utiliser toutes les options de la commande log

Ex

* checkout

git checkout

Cette commande permet de vérifier certaines informations sur la branche courante ainsi que les commits qui ne sont pas encore poussés

* traquer

git add -p

ou encore

git add –patch

Permet de savoir si tous les changements du dépôt sont suivis, ie sont ajoutés à ce qui va etre commité et dans le cas contraire lister les éléments(fichiers et répertoires pas encore suivis)

La différence avec git status est que git status affiche en plus les changements ajoutés et pas encore commités

git add –interactive

ou encore en abrégé

git add -i

git diff –cached

Cette commande permet de voir les fichiers ajoutés et prêt à être commiter

Cette commande permet de faire certaines opérations sur le dépôt de façon interactive en présentant un menu

git rm –cached <filename>

Cette commande permet de supprimer le fichier dont le chemin est spécifié, des fichiers mis en scène et ajoutés au dépôt git. Le fichier n’est pas effacé du disque

* Ignorer des fichiers et répertoires

Dans un projet, on peut ne pas vouloir suivre certains fichiers et répertoires. Pour cela, git utilise un ou plusieurs fichier(s) .gitignore contenant un lien vers des fichiers:

* contenant des informations de configurations que l’on ne veut pas partager avec d’autres développeurs
* contenant des informations secrètes comme le login, le mot de passe, des clés de chiffrements,etc
* temporaires,intermédiaires, comme les caches, les fichiers contenant du code source compilés

Ces liens sont des noms de fichiers ou dossier et peuvent utiliser les règles des expressions globales

Ex

On peut également ignorer des fichiers toujours en éditant ces mêmes liens mais cette fois ci dans le fichier .git/info/exclude

git clean -xn

Permet d’afficher la liste de tous les fichiers ignorés dans son dépôt

git clean -xf

Cette commande permet de supprimer les fichiers ignorés

git clean -Xn

Permet d’afficher la liste de tous les fichiers ignorés dans son dépôt en plus des fichier non suivis

git clean -Xf

Cette commande permet de supprimer les fichiers ignorés et les fichiers non suivis

On peut aussi vérifier si un fichier est ignoré avec la commande

git check-ignore <file>

La commande

git status –ignored permet de lister les fichiers non suivis et ceux ignorés

git rm –cached <file>

Cette commande permet d'arrêter de suivre un fichier traqué(déjà ajouté au dépôt) tout en conservant l’historique des commits précédents qui concernent ce fichier(changements).

Notons que le fichier peut être à tout moment ajouté à nouveau (avec la commande git add)

En omettant l’option –cached, le fichier est aussi supprimé du projet

Keep in mind that if anyone else pulls from the repository after you removed the file from the index, their copy will be physically deleted.

git update-index -skip-worktree

Après création et ajout d’un fichier à l’index, ou après un commit donné, on peut décider d’ignorer les changements intervenus dans un fichier ou un chemin donné. La commande ci-dessous est utilisée à cet effet. La commande git status ne listera donc pas le(s) fichier(s) concerné(s) parmi les fichiers à ajouter dans l’index

git update-index -no-skip-worktree

Cette commande annule l’effet de la précédente. Ce qui permettra de suivre les changements.

git update-index --assume-unchanged <file\_path>

ignorer les futurs changements

git update-index –no-assume-unchanged <file\_path>

Par défaut, dans git on ne peut pas traquer les dossiers vides. Pour contourner ce fait, on peut créer un fichier nommé .gitkeep ou dummy.txt et le placer dans le répertoire concerné

–untracked-files=all