**GIT**

Git est un système de gestion de versions décentralisé largement utilisé pour suivre les modifications apportées aux fichiers et aux projets informatiques. Il permet aux développeurs de collaborer efficacement sur le code source en maintenant un historique des modifications et en facilitant le suivi des différentes versions. Voici les concepts de base pour comprendre Git :

**Dépôt (repository) :** Un dépôt Git est un espace de stockage qui contient l'historique complet des modifications apportées à un projet. Il peut être situé localement sur votre machine (dépôt local) ou sur un serveur distant (dépôt distant).

**Clone :** Cloner un dépôt signifie créer une copie locale complète du dépôt distant sur votre machine. Cela vous permet de travailler sur le projet localement.

**Commit :** Un commit représente une modification ou un ensemble de modifications apportées à un fichier ou à un groupe de fichiers. Chaque commit est accompagné d'un message qui explique la raison de la modification.

**Branch (branche) :** Une branche est une ligne de développement distincte dans un dépôt Git. Les branches permettent de travailler sur des fonctionnalités ou des correctifs de manière isolée, sans perturber la branche principale (généralement appelée "master" ou "main").

**Merge (fusion) :** La fusion est le processus de combinaison des modifications d'une branche dans une autre. Par exemple, vous pouvez fusionner une branche de fonctionnalité dans la branche principale une fois que la fonctionnalité est terminée.

**Pull Request (demande de fusion) :** Une demande de fusion est une demande formelle pour fusionner des modifications d'une branche vers une autre. C'est souvent utilisé dans le contexte de développement collaboratif, notamment sur des plates-formes telles que GitHub.

**Conflict (conflit) :** Un conflit survient lorsque Git ne peut pas résoudre automatiquement les différences entre deux branches à fusionner. Les conflits doivent être résolus manuellement en choisissant quelle version des modifications doit être conservée.

**Push (pousser) :** Pousser signifie téléverser les commits locaux vers un dépôt distant. Cela permet de partager vos modifications avec d'autres collaborateurs.

**Pull (tirer) :** Tirer signifie récupérer les commits depuis un dépôt distant pour mettre à jour votre copie locale. Cela vous permet d'obtenir les dernières modifications des autres membres de l'équipe.

**Versionnage :** Git maintient un historique complet de toutes les modifications apportées à un projet, ce qui permet de suivre l'évolution du code source au fil du temps.

**Ignore (ignorer) :** Vous pouvez spécifier des fichiers ou des répertoires à ignorer dans Git en utilisant un fichier .gitignore. Cela est utile pour exclure des fichiers de build, des fichiers temporaires, etc., du suivi de version.

**Log (journal des commits) :** Le journal des commits affiche l'historique des commits dans un dépôt Git. Vous pouvez voir qui a apporté quelles modifications et quand.

Git est un outil puissant qui offre un contrôle complet sur le processus de développement logiciel et facilite la collaboration entre les membres d'une équipe de développement. Il est largement utilisé dans l'industrie du logiciel pour gérer le code source de projets de toutes tailles. Pour commencer avec Git, vous pouvez installer Git sur votre machine et suivre un tutoriel ou une documentation pour apprendre les commandes de base.

Installation de GIT

Téléchargez Git : Rendez-vous sur le site officiel de Git à l'adresse <https://git-scm.com/download/win>.

2. Configuration de Git :

Après l'installation, configurez Git avec votre nom d'utilisateur et votre adresse e-mail. Cela permet d'associer vos modifications à votre identité. Utilisez les commandes suivantes en remplaçant "Votre Nom" et "votre@email.com" par vos informations :

git config --global user.name "Votre Nom"

$ git config --global user.name "hermann paula"

git config --global user.email [votre@email.com](mailto:votre@email.com)

$ git config --global user.email "fotiepaula@gmail.com"

3. Création d'un nouveau dépôt Git :

Vous pouvez initialiser un nouveau dépôt Git dans un répertoire en utilisant la commande suivante :

git init

4. Clonage d'un dépôt existant :

Si vous travaillez sur un projet existant, vous pouvez le cloner à partir d'un dépôt distant (comme GitHub) en utilisant la commande :

git clone URL\_du\_dépôt

$ git clone https://gitlab.com/eazytrainingfr/alpinehelloworld.git

5. Ajout de fichiers au suivi de version :

Pour ajouter des fichiers à la zone de suivi de version (staging area), utilisez la commande :

git add nom\_du\_fichier

$ git add dsm\ 14\ sept\ 2023.txt

6. Création d'un commit :

Une fois les fichiers ajoutés à la zone de suivi de version, vous pouvez créer un commit avec un message explicatif :

git commit -m "Message du commit"

$ git commit -m "adjout du fichier dsm"

7. Consultation de l'historique des commits :

Pour voir l'historique des commits, utilisez :

git log

8. Création et gestion de branches :

Vous pouvez créer une nouvelle branche avec la commande :

git branch nom\_de\_la\_branche

$ git branch fpaula

Pour basculer vers une branche existante, utilisez :

git checkout nom\_de\_la\_branche

$ git checkout fpaula

9. Fusion de branches :

Pour fusionner une branche dans une autre (par exemple, pour intégrer une fonctionnalité), utilisez :

git merge nom\_de\_la\_branche

$ git merge fpaula

10. Récupération de mises à jour :

Pour récupérer les dernières mises à jour d'un dépôt distant, utilisez :

git pull

$ git pull https://gitlab.com/eazytrainingfr/alpinehelloworld.git

11. Envoi de commits vers un dépôt distant :

Pour pousser vos commits vers un dépôt distant, utilisez :

git push

$ git push https://gitlab.com/eazytrainingfr/alpinehelloworld.git

12. Résolution de conflits :

Si des conflits surviennent lors de la fusion de branches, vous devrez les résoudre manuellement en éditant les fichiers en conflit, puis en créant un nouveau commit.

Ceci constitue une introduction de base à Git. Pour approfondir vos connaissances, consultez la documentation officielle de Git (https://git-scm.com/docs) et explorez des tutoriels en ligne. La pratique régulière est essentielle pour maîtriser Git, alors n'hésitez pas à créer un compte sur une plate-forme d'hébergement de dépôts Git (comme GitHub, GitLab ou Bitbucket) pour collaborer avec d'autres et gérer vos projets.