Système de Gestion de Version

1. Introduction

Qu'est-ce que Git ?

Git est un logiciel de gestion de versions créé par Linus Torvalds en 2005. Il permet de conserver un historique des modifications apportées à un projet et facilite la collaboration entre les développeurs. Git est distribué, ce qui signifie que chaque utilisateur dispose d'une copie complète du projet. Il offre également des fonctionnalités avancées telles que la gestion des branches, la détection des conflits et une intégrité des données garantie. Git est largement utilisé dans le développement logiciel et est considéré comme un outil essentiel pour tout développeur.

2. A quoi sert concrètement un système de gestion de version ?

Un système de gestion de version permet de coordonner le travail entre plusieurs personnes en conservant un historique des modifications effectuées sur des fichiers.
Git, en particulier, permet aux développeurs d'accéder à l'historique des changements, de savoir qui a fait quelles modifications et de gérer les différentes versions des fichiers.

3. Les deux modèles des logiciels de gestion de version : modèle centralisé vs modèle décentralisé

Les logiciels de gestion de version sont basés sur deux modèles : le modèle centralisé et le modèle décentralisé.
Dans le modèle centralisé, le code source est centralisé sur un serveur distant, tandis que dans le modèle décentralisé, chaque utilisateur télécharge et héberge le code source complet sur sa propre machine.
Le modèle décentralisé, popularisé par Git, offre une flexibilité de travail et une sécurité accrues, permettant aux utilisateurs de travailler hors ligne et d'utiliser leurs copies locales comme sauvegardes en cas de problème avec le serveur central.

4. Qu'est-ce que GitHub ?

GitHub est un service en ligne qui permet d'héberger des dépôts Git, qui sont des copies des fichiers d'un projet et de leurs versions. En tant que plus grand hébergeur de dépôts Git au monde, GitHub facilite le partage et la collaboration sur les projets.
La plupart des dépôts sur GitHub sont publics, ce qui signifie que n'importe qui peut accéder au code et contribuer à son développement

5. Utiliser Git : ligne de commande, console et interface graphique

Pour utiliser Git, on peut choisir entre la ligne de commande et une interface graphique. Dans ce cours, nous privilégions la ligne de commande pour avoir accès à toutes les commandes et garantir une uniformité dans l'apprentissage.

La maîtrise de la ligne de commande permet également de s'adapter à n'importe quelle interface graphique. Sur Windows, on peut ouvrir la ligne de commande en recherchant "cmd" dans la barre de recherche

6. Installation de Git

Pour installer Git, il suffit de se rendre sur le site officiel http://git-scm.com/downloads, télécharger la dernière version et suivre les instructions à l'écran en laissant les valeurs par défaut.

7. Paramétrage de Git

Pour paramétrer Git, utilisez les commandes suivantes dans la console:

— git config --global user.name "Votre nom"
— git config --global user.email "Votre adresse email"
— Remplacez "Votre nom" par votre nom réel et "Votre adresse email" par votre adresse email réelle.
Vous pouvez vérifier si les informations ont été enregistrées en utilisant les commandes git config user.name et git config user.email.

8. Démarrer un dépôt Git

Pour démarrer un dépôt Git, vous avez deux options :

— Importer un répertoire existant dans Git.
— Cloner un dépôt Git déjà existant.
La suite du cours vous expliquera en détail comment réaliser ces opérations. Avant cela, il est important de comprendre le fonctionnement général de Git et sa gestion des informations.

9. La gestion des informations selon Git

Git gère les données sous forme d'instantanés ou de "snapshots". À chaque enregistrement de l'état d'un projet, Git prend un instantané du contenu de l'espace de travail et crée une référence à cet instantané. Les instantanés sont stockés localement dans une base de données sur notre propre machine. Cette approche permet d'effectuer la plupart des opérations de Git localement, sans avoir besoin d'une connexion à un serveur central distant, ce qui rend les opérations plus rapides et le travail plus agréable

10. Les états des fichiers

Les fichiers dans Git peuvent être soit "suivis" soit "non suivis". Lorsqu'on démarre un dépôt, les fichiers sont non suivis et doivent être indexés et validés. Les fichiers suivis peuvent être dans l'état "modifié", "indexé" ou "validé".
Les fichiers modifiés doivent être indexés avant d'être validés. Une fois validés, les fichiers font partie de l'instantané et le cycle peut recommencer.

11. Les zones de travail

Un projet Git comprend trois zones de travail : le répertoire de travail, la zone d'index et le répertoire Git. Le répertoire de travail contient les fichiers extraits pour être utilisés ou modifiés. La zone d'index stocke les informations sur les fichiers qui feront partie du prochain instantané. Le répertoire Git est l'endroit où sont stockées les méta-données et la base de données du projet. On travaille sur les fichiers dans le répertoire de travail, puis on peut choisir de les indexer avant de les valider dans le répertoire Git.

12. Créer un dépôt Git à partir d'un répertoire existant

Pour créer un dépôt Git à partir d'un répertoire existant, utilisez la commande git init pour initialiser le dépôt. Ensuite, utilisez la commande git status pour voir l'état des fichiers.
Pour indexer des fichiers, utilisez la commande git add, suivi du nom du fichier ou du répertoire.
Enfin, utilisez la commande git commit pour valider les fichiers et les ajouter en base avec un message de commit. Vérifiez l'état avec git status pour confirmer que tous les fichiers sont suivis et enregistrés en base.

13. Cloner un dépôt Git

Pour cloner un dépôt Git déjà existant localement, utilisez la commande git clone suivie de l'URL du dépôt.
Vous pouvez créer un nouveau dépôt sur GitHub en suivant les étapes suivantes :

- Accédez à la page d'accueil de GitHub (https://github.com) et connectez-vous à votre compte.
- Cliquez sur le bouton vert "New" (Nouveau) pour créer un nouveau dépôt.
- Donnez un nom à votre dépôt et choisissez les options de visibilité et d'initialisation appropriées.
- Cliquez sur le bouton "Create repository" (Créer le dépôt) pour créer le dépôt sur GitHub.
- Sur la page du dépôt nouvellement créé, vous trouverez l'URL du dépôt (sous la forme https://github.com/utilisateur/nom-du-depot.git).
- Pour cloner ce dépôt localement, ouvrez votre terminal ou votre interface Git et exécutez la commande git clone <URL-du-dépôt>. Cela créera une copie locale complète du dépôt sur votre machine.

14. Ajouter ou modifier des fichiers dans un projet et actualiser notre dépôt Git

Pour ajouter ou modifier des fichiers dans votre projet et actualiser votre dépôt Git :

- Utilisez la commande git add <nom-du-fichier> pour ajouter un fichier spécifique à la zone d'index.
- Utilisez la commande git add . pour ajouter tous les fichiers modifiés et déjà sous suivi à la zone d'index.
- Utilisez la commande git commit -m "message-du-commit" pour valider les modifications et enregistrer l'état actuel des fichiers dans le dépôt Git.
- Assurez-vous d'ajouter un message descriptif pour chaque commit.
- N'oubliez pas d'exécuter régulièrement les commandes git add et git commit pour mettre à jour votre dépôt Git avec les derniers changements de votre projet.

15. Consulter l'historique des modifications Git

Pour consulter l'historique des modifications Git, utilisez la commande git log.
Elle affiche la liste des commits réalisés, avec les informations telles que la somme de contrôle SHA-1, le nom et l'e-mail de l'auteur, la date et le message du commit.
Cette commande affiche les commits du plus récent au plus ancien.

16. Annuler des modifications apportées à un fichier

Pour annuler des modifications apportées à un fichier dans Git, vous pouvez utiliser la commande générale git checkout -- nom-du-fichier ou la commande spécialisée git restore. Ces commandes vous permettent de revenir à l'état précédent du fichier, en annulant les modifications non validées.

17. Qu'est-ce qu'une branche ?

En Git, une branche est simplement un pointeur vers un commit spécifique. Contrairement à d'autres systèmes de contrôle de version, Git ne crée pas une copie physique complète du répertoire de travail lors de la création d'une branche. Au lieu de cela, une branche est un fichier contenant l'empreinte SHA-1 du commit sur lequel elle pointe. La branche par défaut dans Git s'appelle "master", mais elle peut être renommée. En résumé, créer une nouvelle branche dans Git signifie simplement créer un nouveau pointeur vers un commit sans copier l'ensemble du répertoire de travail.