GIT et GitHub

Par Vera DOS SANTOS
2024

Introduction au contrôle de version et au versioning :

Le contrôle de version est un système qui enregistre les modifications apportées à un fichier ou un ensemble de fichiers au fil du temps afin de pouvoir revenir à une version spécifique plus tard.

Importance du versioning :

Permet de suivre l'historique des modifications, collaborer efficacement, revenir à une version précédente en cas de problème, et maintenir différentes versions d'un projet simultanément.

Git :

Git est un système de contrôle de version distribué, conçu pour gérer les projets de développement avec rapidité et efficacité.

Les concepts de base :

Dépôt (repository) : Un espace de travail Git où le code est stocké.

Commit : Un instantané du projet à un moment donné, avec un message de commit descriptif.

Branche (branch) : Une ligne de développement indépendante.

Merge : Fusionner les modifications d'une branche dans une autre.

Git Bash :

Un émulateur de terminal qui permet d'utiliser Git en ligne de commande ; il permet la gestion des fichiers comme : Ajouter, modifier, supprimer et suivre les fichiers dans un dépôt Git.

Commandes Git de base :

git init : Initialiser un nouveau dépôt Git.

git clone : Cloner un dépôt existant.

git add : Ajouter des fichiers au staging area.

git commit : Enregistrer les modifications dans le dépôt.

git status : Vérifier l'état du dépôt.

git log : Afficher l'historique des commits.

git branch : Lister, créer ou supprimer des branches.

git checkout : Passer à une autre branche.

GitHub :

Une plateforme de collaboration et de gestion de code basée sur Git, permettant aux développeurs de travailler ensemble sur des projets.

Dépôts GitHub :

Création d'un dépôt : Comment créer un nouveau dépôt sur GitHub.

Gestion des dépôts : Ajouter des fichiers, créer des branches, configurer des dépôts.

Collaboration sur GitHub :

Pull requests : Proposer des modifications à un projet.

Forks : Copier un dépôt pour y apporter des modifications indépendantes.

Issues : Suivre les bugs et les tâches.

Utilisation de GitHub Actions : Automatiser les workflows de développement.

Bonnes pratiques de versioning :

Stratégies de branchement :

Branche principale (main) : Branche stable contenant le code prêt pour la production.

Branches de fonctionnalité : Branches temporaires pour développer de nouvelles fonctionnalités.

Branches de correction de bug : Branches pour corriger des bugs spécifiques.

Messages de commit clairs et informatifs : Rédiger des messages de commit descriptifs qui expliquent clairement les modifications apportées.

Gestion des conflits de merge : Techniques pour résoudre les conflits lors de la fusion de branches.

Utilisation des releases et des tags : Marquer des versions spécifiques du code avec des tags et des releases pour faciliter la gestion des versions.