Versionning avec Git

Rapport écrit sur les Systèmes de Gestion de Versions et Git

Villeret Ugo
Duc Phung Mai

Introduction

Les systèmes de gestion de versions (VCS) sont des outils essentiels dans le développement logiciel. Ils permettent de suivre les modifications apportées au code source, de collaborer efficacement et de gérer les différentes versions d'un projet. Ce rapport explore l'historique des systèmes de gestion de versions, les principes fondamentaux de Git, ses fonctionnalités avancées ainsi que des plateformes comme GitHub.

Table des matières

Historique et Utilité des Systèmes de Gestion de Versions

Les systèmes de gestion de versions (VCS) ont évolué à travers plusieurs étapes clés.

Les premières solutions étaient centralisées comme SCCS et RCS dans les années 1970.

Avec le CVS dans les années 1980, une gestion efficace de l'intégration simultanée a été introduite.

Subversion (SVN) a remplacé CVS dans les années 2000 avec des améliorations notables.

L'émergence des systèmes distribués comme Git a marqué une nouvelle ère dans la gestion de versions.

Utilité des VCS

Suivi des modifications : Chaque commit enregistre un instantané du projet.
Collaboration efficace : Permet à plusieurs développeurs de travailler sans interférence.
Gestion des versions : Facilite le travail sur plusieurs versions en parallèle.
Résolution des conflits : Intègre des outils pour faciliter l'intégration des modifications.
Automatisation des tests et déploiements : Intègre les processus CI/CD.
Sécurité : Chaque développeur a une copie complète de l'historique.
Amélioration de la qualité du code : Favorise les relectures et les bonnes pratiques.

Principes Fondamentaux de Git

Dépôts : Un dépôt peut être local ou distant, chaque développeur a une copie complète.
Branches : Facilitent le développement parallèle et évitent les conflits.
Commits : Correspondent à des instantanés du projet à un moment donné.
Fusion (Merge) : Combine les modifications de différentes branches.
Rebase : Réapplique une branche sur une autre, modifiant l'historique.
Gestion des conflits : Outils pour résoudre les conflits entre branches.
Historique : Commandes comme git log pour examiner les modifications.

Fonctionnalités Avancées de Git

Rebasing : Pour maintenir un historique propre et linéaire.
Cherry-Pick : Applique des commits spécifiques d'une branche à une autre.
Stash : Sauvegarde temporaire des modifications sans commit.
Hooks : Scripts automatiques pour différents événements Git.
Bisect : Outil pour retrouver l'origine d'un bug efficacement.
Submodules : Intègre des dépôts externes dans un projet.

Présentation de GitHub et de ses Alternatives

GitHub : Plateforme la plus populaire pour le développement open-source.
GitLab : Solution complète avec CI/CD intégré et modèle open-source.
Bitbucket : Utilisé par les entreprises pour son intégration avec Jira.

Cas d'Utilisation dans des Projets Collaboratifs

Développement open source : Permet des contributions de développeurs du monde entier.
Gestion d’équipes : Coordonne les fonctionnalités et correctifs.
Intégration et déploiement continus (CI/CD) : Automatise les processus.

Conclusion

Git et les systèmes de gestion de versions ont transformé le développement logiciel, offrant des outils puissants pour la collaboration et l'automatisation. Maîtriser Git est essentiel pour les développeurs modernes.

Merci de votre attention!