NOM: PRÉNOM: CLASSE:

MR ROLIN 2025

[DEV] Git & Repositories

Objectifs
À la fin de ce cours, l'apprenant doit être capable de :

- Expliquer ce qu'est Git et son rôle dans la gestion de version.
- Décrire la structure et le fonctionnement d'un repository.
- Effectuer les principales opérations Git : init, clone, add, commit, push, pull.
- Comprendre la logique des branches et de la fusion.
- Identifier les éléments que l'on trouve habituellement dans un repository.
- Utiliser Git pour organiser et suivre un projet professionnel.

Introduction

Dans les métiers du numérique, la gestion de versions est devenue indispensable. Les projets informatiques évoluent rapidement, impliquent souvent plusieurs personnes et nécessitent une traçabilité rigoureuse. Git est devenu l'outil de référence dans ce domaine. Il permet de sauvegarder l'évolution d'un projet, de collaborer efficacement et de revenir facilement à une version précédente en cas d'erreur.

Les repositories, ou dépôts, sont au cœur de cette technologie. Ils constituent l'espace de stockage où se trouve non seulement le code, mais aussi tout l'historique des modifications. Comprendre leur fonctionnement est une compétence essentielle pour tout professionnel travaillant sur des projets techniques, qu'il s'agisse de développement logiciel, d'intégration, d'administration systèmes ou même de documentation.

Git : un système de gestion de versions distribué

Git est un logiciel créé par Linus Torvalds pour assurer le suivi de versions du noyau Linux. Sa particularité est d'être distribué : chaque utilisateur possède une copie complète du repository et de son historique. Cela rend le travail plus rapide et sécurisé, car aucune opération essentielle ne dépend d'un serveur externe.

Contrairement à d'autres outils qui enregistrent uniquement les différences entre chaque version, Git prend des snapshots, c'est-à-dire des "photos" complètes de l'état des fichiers. Cette approche facilite la relecture de l'historique et accélère les opérations internes.

Qu'est-ce qu'un repository et à quoi sert-il ?

Un repository est un espace où Git stocke un projet et toutes les informations relatives à son évolution. Il peut exister sous deux formes :

- Local, sur l'ordinateur de l'utilisateur.
- Distant, hébergé sur une plateforme comme GitHub ou GitLab.

Lorsque l'utilisateur travaille, il modifie des fichiers dans son espace de travail. Lorsqu'il souhaite sauvegarder ces changements, Git lui permet de les ajouter à une zone intermédiaire (le staging), puis de les enregistrer dans l'historique sous forme de commit.

Le repository contient donc à la fois les fichiers du projet, leur historique, les métadonnées nécessaires au suivi, et éventuellement des branches supplémentaires.

Structure interne du repository

À l'intérieur du repository, on distingue trois zones principales :

- Working Directory (ou zone de travail) : c'est là que se trouvent les fichiers que l'utilisateur modifie.
- Staging Area (ou index) : zone de préparation où les fichiers sont placés avant d'être enregistrés. Elle permet de choisir précisément ce que le commit va contenir.

- Repository (.git) : dossier caché contenant toute l'histoire du projet, les commits, les branches et la configuration.

Cette organisation garantit une grande flexibilité, en offrant la possibilité de préparer soigneusement ses enregistrements avant de les valider dans l'historique.

Les commandes essentielles

Pour interagir avec Git, quelques commandes sont indispensables.
`git init` crée un nouveau repository vide.
`git clone` permet de récupérer une copie d'un repository distant existant.
Lorsqu'un fichier a été modifié, `git add` l'ajoute au staging.
`git commit` enregistre définitivement les changements.
Enfin, `git push` et `git pull` servent à synchroniser le repository local avec un repository distant, respectivement pour envoyer ou récupérer des modifications.

Ces opérations constituent le flux de travail standard de la plupart des projets.

Branches et fusion

Les branches sont des lignes de développement parallèles. Elles permettent de travailler sur des fonctionnalités, des corrections ou des tests sans perturber la version principale du projet. La branche principale est souvent appelée main ou master.

Lorsque le travail sur une branche est terminé, il est possible de la fusionner (merge) dans la branche principale. Cette opération réunit les évolutions et peut parfois générer des conflits si deux personnes ont modifié les mêmes parties du code. Git fournit alors des outils pour résoudre ces conflits manuellement.

Ce que contient un repository

Outre les fichiers du projet et les commits, un repository contient plusieurs éléments importants :

- Les branches, qui représentent différentes versions en parallèle.
- Les tags, souvent utilisés pour marquer les versions stables (ex : v1.0).
- Le fichier .gitignore, qui indique à Git quels fichiers ne doivent pas être suivis (exécutables, fichiers temporaires…).
- Un fichier README, qui sert de documentation d'accueil pour comprendre le projet.
- Des dossiers de configuration pour l'intégration continue (CI/CD), des workflows automatisés ou des dépendances.

Ces éléments contribuent à structurer le projet, à faciliter la collaboration et à garantir une organisation claire.

Plateformes de repositories distants

Des plateformes comme GitHub, GitLab ou Bitbucket proposent des services supplémentaires : gestion des issues, pull requests, suivi de tâches, hébergement de pages web, automatisation de tests et de déploiements. Elles jouent un rôle central dans le travail collaboratif.

Les pull requests ou merge requests, par exemple, permettent de proposer des modifications, de les faire relire par d'autres membres de l'équipe et de les valider dans un processus transparent et professionnel.

Conclusion

Git et les repositories sont des outils incontournables dans les environnements professionnels modernes. Ils permettent à la fois d'assurer un suivi précis de l'évolution d'un projet, d'améliorer la

collaboration et d’éviter les pertes de données. Leur fonctionnement, fondé sur la notion de snapshots, de branches et de commits, offre une grande flexibilité et une traçabilité complète.

Pour bien utiliser Git, il est important d’adopter de bonnes pratiques : faire des commits réguliers et clairs, travailler sur des branches dédiées, documenter le travail et maintenir un repository bien structuré. Une fois ces réflexes acquis, Git devient un allié puissant pour mener à bien n’importe quel projet technique.