Git & Repositories

Introduction à Git et aux repositories

- Git : système de gestion de versions distribué
- Repository : espace où le projet et son historique sont stockés
- Utilité : suivre l'évolution du code, collaborer, revenir en arrière

Qu’est-ce qu’un repository ?

- Contient les fichiers du projet
- Stocke l’historique complet des modifications
- Peut être local (ordinateur) ou distant (GitHub, GitLab, Bitbucket)

Structure d’un repository

- Dossier .git : configuration + historique
- Zone de travail : fichiers en cours d’édition
- Staging area : zone de préparation des commits
- Branches : différentes versions du projet

Git : fonctionnement général

- Git enregistre des snapshots (photos de l'état des fichiers)
- Chaque modification devient un commit
- Identifiant unique (SHA) pour suivre chaque version

Les commandes essentielles

- `git init` : créer un repository
- `git clone` : copier un repository distant
- `git add` : ajouter au staging
- `git commit` : enregistrer des modifications
- `git push` / `git pull` : synchroniser

Branches et fusion

- Une branche = ligne de développement parallèle
- `git branch`, `git checkout`, `git switch`
- `git merge` pour réunir les changements

Ce qu'on trouve dans un repository

- Fichiers du projet (code, docs…)
- Historique complet des commits
- Branches et tags
- Fichiers de configuration : .gitignore, workflows CI/CD, README

Plateformes associées : GitHub, GitLab, Bitbucket

- Hébergement de repositories distants
- Collaboration : issues, pull requests, commentaires
- Automatisation : CI/CD, documentation, gestion de projet

Bonnes pratiques

- Faire des commits clairs et fréquents
- Éviter de commit du code non fonctionnel
- Travailler sur des branches
- Écrire des messages de commit explicites
- Utiliser .gitignore correctement

Conclusion

- Git : outil indispensable du développement moderne
- Repositories : base de la collaboration et du suivi de versions
- Approche structurée = qualité et traçabilité