📖

Introduction à Git

1. Introduction à Git et au Contrôle de Version

1.1. Introduction au contrôle de version

- Problèmes rencontrés sans système de versioning (perte de code, duplication).
- Notion de versioning centralisé vs décentralisé.

1.2. Git : c'est quoi ?

Git est un **système de gestion de version distribué**, conçu pour suivre les modifications apportées aux fichiers dans un projet, collaborer efficacement et gérer l'historique des changements.

1.2.1. Définition

Git permet de :

- **Suivre les modifications** apportées aux fichiers d'un projet, comme un code source.
- **Revenir à une version précédente** du projet en cas de besoin.
- **Travailler en équipe** sur le même projet, même si les collaborateurs sont dispersés géographiquement.

1.2.2. Origines

- Créé en 2005 par **Linus Torvalds**, le créateur de Linux.
- Développé pour gérer le code source de Linux avec des performances élevées et une grande fiabilité.

1.2.3. Caractéristiques principales

1. **Distribué** :
   - Contrairement à des outils centralisés comme SVN, chaque utilisateur dispose d'une copie complète du dépôt (l'ensemble des fichiers et leur historique).
   - Pas besoin d'une connexion constante à un serveur.
2. **Rapide** :
   - Les opérations (comme les commits, les fusions) sont rapides car elles sont effectuées localement.
3. **Historique sécurisé** :
   - Les données et l'historique sont stockés de manière immuable grâce à des mécanismes de hachage cryptographique (SHA-1).
   - Impossible de modifier l'historique sans que cela soit détecté.
4. **Branches flexibles** :
   - Les branches permettent de travailler sur des fonctionnalités ou corrections de bugs sans affecter la version principale.
   - Fusionner les branches est facile et efficace.

1.2.4. Pourquoi utiliser Git ?

1. **Travail collaboratif** :
   - Permet à plusieurs développeurs de travailler simultanément sur le même projet.
   - Les plateformes comme GitHub ou GitLab facilitent la gestion collaborative.
2. **Suivi des modifications** :
   - Historique complet des changements, avec les auteurs et les dates.
   - Identification et correction rapide des erreurs.
3. **Expérimentation** :
   - Créez des branches pour tester de nouvelles idées sans risquer de casser le projet principal.

1.3. Installation de Git

1.3.1. Téléchargement de Git

Sous Windows :

1. Aller sur le site officiel de Git : https://git-scm.com/.
2. Télécharger l'installeur pour Windows (version stable recommandée).
3. Lancer l'installeur et suivre les étapes :
   - **Choisir les options par défaut**.
   - Configurer l'éditeur par défaut (recommandé : Visual Studio Code ou Vim).
   - Sélectionner « Git from the command line and also from 3rd-party software ».
4. Terminer l'installation.

Sous Linux :

1. Ouvrir le Terminal.
2. Installer Git avec la commande adaptée à votre distribution :
   - **Debian/Ubuntu** :

     ```
     sudo apt update && sudo apt install git
     ```

   - **Fedora** :

     ```
     sudo dnf install git
     ```

   - **Arch Linux** :

     ```
     sudo pacman -S git
     ```

3. Vérifier l'installation avec `git --version`.

1.3.2. Configuration initiale de Git

1. Configurer le nom d'utilisateur global :

   ```
   git config --global user.name "your Name"
   ```

2. Configurer l'adresse email associée :

```
git config --global user.email "your.email@example.com"
```

3. Vérifier la configuration :

```
git config --list
```

1.3.3. Tester l'installation

1. Créer un dossier pour le test :

```
mkdir test-git && cd test-git
```

2. Créer un fichier vide :

```
echo "Hello world :)" > README.md
```