Guide d'utilisation de Git

Introduction

Git est un système de contrôle de version distribué qui permet de gérer le code source et son historique de modifications. Contrairement aux systèmes centralisés, Git permet à chaque développeur de travailler sur une branche indépendante avant de fusionner ses modifications avec le projet principal.

Pourquoi utiliser Git ?

- Historique des modifications → Chaque changement est enregistré.
- Travail collaboratif → Plusieurs développeurs peuvent travailler en parallèle.
- Sauvegarde et partage → Facilite le stockage et la récupération du code via GitHub.

1. Installation de Git

Avant de commencer, installe Git for Windows :

https://git-scm.com/downloads

Git for Windows permet d'utiliser :

- Git Bash → Terminal avec commandes Linux (ls, cd, touch…).
- Invite de commande (cmd) → Utilisation de Git sous Windows.

2. Comment déposer un projet sur Git

Étape 1 : Créer un projet Git

- Créer un dossier et l'initialiser avec Git

  Ouvre Git Bash ou l'invite de commande, et exécute :

  mkdir Exercice (on peut créer le dossier manuellement)

  cd Exercice

  git init

  git init permet d'activer Git dans le dossier Exercice.

- Si tu crées un dépôt sur GitHub, NE coche PAS "Add a README file" si tu as déjà initialisé Git en local.

Étape 2 : Ajouter les fichiers et faire un commit

- Ajoute un fichier (ex: document.pdf ou script.linq) dans le dossier Exercice, puis exécute

git add . → Ajoute tous les fichiers modifiés.

git commit -m "Ajout des fichiers PDF et LinqPad" → Enregistre les modifications avec un message clair.

Étape 3 : Lier le projet à GitHub

- Créer un dépôt GitHub

Va sur GitHub

Clique sur "New Repository"

Nom du dépôt : Exercice

Clique sur "Create Repository"

- Lier le dépôt local à GitHub

git remote add origin https://github.com/TonNomUtilisateur/Exercice.git

Étape 4 : Définir master comme branche par défaut (cette étape se fait une seule fois)

git branch -M master

git push -u origin master

Explication :

git branch -M master → Renomme la branche actuelle en master.

git push -u origin master → Envoie les fichiers sur GitHub et définit master comme branche principale.

Étape 5 : Mettre à jour GitHub après une modification locale

- À chaque modification d'un fichier en local, suis ces étapes :

git status # Vérifier les changements (optionnel)

git add .

git commit -m "Description des modifications"

git push

- Si la modification a été faite en ligne (GitHub), il faut synchroniser en local avec :

git pull

Travailler en équipe (binôme)

- Si ton binôme veut récupérer le projet, il doit :

Cloner le dépôt GitHub sur son PC

git clone https://github.com/TonNomUtilisateur/Exercice.git

- Modifier et envoyer ses changements

Il suit ensuite les mêmes étapes :

git add .

git commit -m "Modification par binôme"

git push

- Récupérer les modifications de son binôme

git pull

Assurez-vous de toujours faire git pull avant git push pour éviter les conflits.

Conclusion

- Git facilite le travail collaboratif et la gestion des versions.
- GitHub permet de sauvegarder et partager le projet en ligne.
- Les bases : git add, git commit, git push, git pull.