# Comment utiliser Git en ligne de commande pour gÃ©rer mes projetsÂ ?

Git est un puissant systÃ¨me de contrÃ´le de version qui permet aux dÃ©veloppeurs de suivre les modifications apportÃ©es Ã  leur code au fil du temps. Il est largement utilisÃ© dans le dÃ©veloppement de logiciels et est essentiel pour collaborer sur des projets avec d'autres dÃ©veloppeurs. Bien qu'il existe de nombreuses interfaces utilisateur graphiques (GUI) disponibles pour Git, l'utilisation de la ligne de commande offre plusieurs avantages, notamment une plus grande flexibilitÃ©, une meilleure efficacitÃ© et un meilleur contrÃ´le.

### Premiers pas avec Git en ligne de commande

Pour commencer Ã  utiliser Git en ligne de commande, vous devez installer Git sur votre systÃ¨me. Les instructions d'installation pour Windows, macOS et Linux sont disponibles sur le site Web de Git.

Une fois Git installÃ©, vous pouvez le configurer en dÃ©finissant votre nom d'utilisateur et votre adresse e-mail. Vous pouvez Ã©galement gÃ©nÃ©rer des clÃ©s SSH, qui vous permettront de vous connecter en toute sÃ©curitÃ© aux dÃ©pÃ´ts Git distants.

### Commandes Git de base en ligne de commande

Une fois Git configurÃ©, vous pouvez commencer Ã  utiliser des commandes de base pour gÃ©rer vos projets.

### Initialisation

Pour initialiser un nouveau dÃ©pÃ´t Git, utilisez la commande `git init`. Cela crÃ©era un rÃ©pertoire `.git` dans le rÃ©pertoire de votre projet, qui contiendra toutes les mÃ©tadonnÃ©es Git.

### Ã‰tapes des modifications

Pour ajouter des modifications Ã  la zone de prÃ©paration, utilisez la commande `git add`. Cela marquera les modifications comme prÃªtes Ã  Ãªtre validÃ©es dans le dÃ©pÃ´t.

### Validation des modifications

Pour valider les modifications de la zone de prÃ©paration vers le dÃ©pÃ´t local, utilisez la commande `git commit`. Cela crÃ©era un nouvel instantanÃ© de votre projet Ã  ce moment prÃ©cis.

### Affichage des modifications

Pour afficher l'Ã©tat de l'arborescence de travail et de la zone de prÃ©paration, utilisez la commande `git status`. Cela vous montrera quels fichiers ont Ã©tÃ© modifiÃ©s, ajoutÃ©s ou supprimÃ©s.

Pour afficher les diffÃ©rences entre l'arborescence de travail et la zone de prÃ©paration ou entre deux validations, utilisez la commande `git diff`.

### Branchement et fusion

Git vous permet de crÃ©er et de basculer entre des branches, qui sont des lignes de dÃ©veloppement indÃ©pendantes. Cela peut Ãªtre utile pour travailler sur diffÃ©rentes fonctionnalitÃ©s ou corrections de bogues sans affecter la branche principale de votre projet.

### CrÃ©ation et changement de branches

Pour rÃ©pertorier toutes les branches, utilisez la commande `git branch`. Pour passer Ã  une branche spÃ©cifiÃ©e, utilisez la commande `git checkout`.

Pour crÃ©er une nouvelle branche, utilisez la commande `git branch <branch-name>`.

### Fusion des branches

Pour fusionner une branche spÃ©cifiÃ©e dans la branche actuelle, utilisez la commande `git merge <branch-name>`.

## DÃ©pÃ´ts distants

Git vous permet de stocker votre projet dans un dÃ©pÃ´t distant, tel que GitHub ou GitLab. Cela vous permet de collaborer avec d'autres dÃ©veloppeurs et de partager votre code avec le monde.

## Ajout d'un dÃ©pÃ´t distant

Pour ajouter un dÃ©pÃ´t distant, utilisez la commande `git remote add <remote-name> <remote-url>`.

## Pousser et tirer les modifications

Pour pousser les modifications locales vers un dÃ©pÃ´t distant, utilisez la commande `git push <remote-name> <branch-name>`. Pour extraire les modifications d'un dÃ©pÃ´t distant, utilisez la commande `git pull <remote-name> <branch-name>`.

## Collaboration avec Git

Git fournit plusieurs fonctionnalitÃ©s qui facilitent la collaboration avec d'autres dÃ©veloppeurs.

## Bifurquer un dÃ©pÃ´t

Bifurquer un dÃ©pÃ´t vous permet de crÃ©er votre propre copie d'un projet sur GitHub ou d'autres plateformes d'hÃ©bergement Git. Cela vous permet d'apporter des modifications au projet sans affecter le dÃ©pÃ´t d'origine.

## Clonage d'un dÃ©pÃ´t

Le clonage d'un dÃ©pÃ´t vous permet de crÃ©er une copie locale d'un dÃ©pÃ´t distant. Cela vous permet de travailler sur le projet hors ligne et de renvoyer vos modifications vers le dÃ©pÃ´t distant lorsque vous avez terminÃ©.

## RÃ©solution des conflits de fusion

Lorsque vous fusionnez deux branches, Git peut rencontrer des conflits de fusion. Cela se produit lorsque le mÃªme fichier a Ã©tÃ© modifiÃ© dans les deux branches. Pour rÃ©soudre les conflits de fusion, vous devrez modifier manuellement le fichier et rÃ©soudre les conflits.

## Commandes Git avancÃ©es

Git propose une large gamme de commandes avancÃ©es qui peuvent Ãªtre utilisÃ©es pour effectuer des tÃ¢ches plus complexes.

## Cacher les modifications

La commande `git stash` vous permet d'enregistrer temporairement les modifications dans l'arborescence de travail. Cela peut Ãªtre utile si vous devez passer Ã  une autre branche ou travailler sur une autre tÃ¢che.

## Ignorer les fichiers

La commande `git add -f <file-name>` vous permet de forcer l'ajout d'un fichier Ã  la zone de prÃ©paration. Cela peut Ãªtre utile pour ignorer les fichiers que vous ne souhaitez pas suivre dans Git.

## Annuler les modifications

La commande `git reset HEAD <file-name>` vous permet d'annuler la prÃ©paration d'un fichier dans la zone de prÃ©paration. La commande `git checkout -- <file-name>` vous permet de restaurer un fichier Ã  son dernier Ã©tat validÃ©.

Git est un outil puissant qui peut Ãªtre utilisÃ© pour gÃ©rer des projets de toutes tailles. En apprenant les bases de Git en ligne de commande, vous pouvez amÃ©liorer votre productivitÃ© et votre collaboration avec d'autres dÃ©veloppeurs.

Pour en savoir plus sur Git, je vous encourage Ã  explorer la documentation officielle de Git et d'autres ressources disponibles en ligne.

https://fr.commandline.wiki/how-can-i-use-commandline-git-to-manage-my-projects/