git

Les commandes essentielles pour utiliser et manipuler Git

Versionner • Collaborer • Garder l'historique de votre code

Git, c'est simple quand on connait les bonnes commandes !

1 Configuration

Définir votre identité et vos préférences.

`git config --global user.name "Votre Nom"`	Définit votre nom d'utilisateur
`git config --global user.email "vous@exemple.com"`	Définit votre email
`git config --global core.editor "code --wait"`	Choisit l'éditeur par défaut
`git config --list`	Affiche la configuration actuelle

2 Initialiser un projet

Créer un dépôt Git dans un dossier existant ou nouveau.

`git init`	Initialise un dépôt Git (dans le dossier courant)
`git clone <url>`	Clone un dépôt distant (ex. git clone https://github.com/user/projet.git)

3 Explorer l'état du dépôt

Voir ce qui a changé, où vous en êtes.

`git status`	Affiche l'état des fichiers
`git log`	Affiche l'historique des commits
`git log --oneline`	Historique simplifié
`git diff`	Montre les différences non indexées
`git diff --staged`	Montre les différences indexées

4 Travailler avec les fichiers

Ajouter, retirer et valider vos modifications.

`git add <fichier>`	Ajoute un fichier à l'index
`git add .`	Ajoute tous les fichiers modifiés
`git rm <fichier>`	Supprime un fichier du dépôt
`git mv <ancien> <nouveau>`	Renomme un fichier
`git commit -m "message"`	Crée un commit avec un message
`git commit --amend`	Modifie le dernier commit

5 Gérer les branches

Créer, basculer et fusionner des branches.

`git branch`	Liste les branches
`git branch <nom>`	Crée une nouvelle branche
`git checkout <nom>`	Change de branche
`git switch <nom>`	(Alternative plus simple à checkout)
`git merge <branche>`	Fusionne une branche
`git branch -d <nom>`	Supprime une branche

6 Travailler avec un dépôt distant

Pousser et récupérer vos modifications.

`git remote -v`	Affiche les dépôts distants
`git remote add origin <url>`	Ajoute un dépôt distant
`git fetch`	Récupère les modifications (sans fusion)
`git pull`	Récupère et fusionne les modifications
`git push`	Envoie vos commits
`git push -u origin <branche>`	Pousse et définit la branche de suivi

7 Annuler et revenir en arrière

Corriger une erreur ou revenir à un état précédent.

`git reset <fichier>`	Retire un fichier de l'index
`git reset --hard`	Annule tous les changements (attention !)
`git revert <commit>`	Crée un commit qui annule un autre commit
`git checkout <commit>`	Revient à un commit précédent (en lecture seule)
`git reflog`	Voir l'historique des positions de la tête

8 Astuces utiles

Quelques commandes bonus pour aller plus loin.

`git stash`	Met en pause vos modifications
`git stash pop`	Récupère les modifications en pause
`git show <commit>`	Affiche le contenu d'un commit
`git tag <nom>`	Crée un tag
`git blame <fichier>`	Montre qui a modifié chaque ligne
`git clean -fd`	Supprime les fichiers non suivis

Le workflow Git en un coup d'œil

Modifiez → Ajoutez → Validez → Partagez

À retenir

- `git status` : toujours le bon réflexe.
- Faites des commits clairs et fréquents.
- Utilisez les branches pour travailler en parallèle.
- Pull avant de pousser (et résoudre les conflits si besoin).
- N'ayez pas peur de refaire un `git reset` ou un `git revert`.

Un bon usage de Git, c'est un projet plus serein !