



Les commandes essentielles pour utiliser et manipuler Git

Versionner • Collaborer • Garder l'historique de votre code

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

1 Configuration



Définir votre identité et vos préférences.

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

2 Initialiser un projet



Créer un dépôt Git dans un dossier existant ou nouveau.

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

3 Explorer l'état du dépôt



Voir ce qui a changé, où vous en êtes.

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

4 Travailler avec les fichiers



Ajouter, retirer et valider vos modifications.

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

5 Gérer les branches



Créer, basculer et fusionner des branches.

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

6 Travailler avec un dépôt distant



Pousser et récupérer vos modifications.

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

7 Annuler et revenir en arrière



Corriger une erreur ou revenir à un état précédent.

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

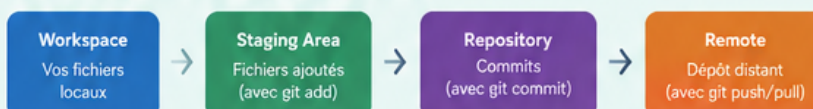
8 Astuces utiles



Quelques commandes bonus pour aller plus loin.

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

