Principales commandes Git Bash

Voici une liste plus complète des commandes Git courantes, classées par catégorie, utilisables dans Git Bash (ou tout autre terminal où Git est installé).

Note : Cette liste n'est pas exhaustive, mais elle couvre la grande majorité des commandes utilisées au quotidien et pour des opérations plus avancées.

Catégorie	Commande	Description Principale
Configuration Initiale	git config	Configurer les options Git (nom utilisateur, email, alias, couleurs, éditeur par défaut, etc.) au niveau local ou global.
Création et Récupération de Projets	git init	Initialiser un nouveau dépôt Git vide dans le répertoire courant.
	git clone <url>	Copier (cloner) un dépôt Git distant existant sur votre machine locale.
Workflow de Base (Snapshotting)	git status	Afficher l'état des fichiers du répertoire de travail et de la zone d'index (staging area).
	git add <fichier/dossier> ou git add .	Ajouter des modifications de fichiers à la zone d'index (staging) pour le prochain commit.
	git commit -m "message"	Enregistrer les modifications indexées (snapshot) dans l'historique du dépôt avec un message descriptif.
	git commit -am "message"	Combine git add pour les fichiers déjà suivis et git commit en une seule étape.
	git rm <fichier>	Supprimer un fichier du répertoire de travail ET de l'index.
	git mv <ancien_nom> <nouveau_nom>	Renommer ou déplacer un fichier et indexer le changement.
Gestion des Branches et Fusions	git branch	Lister, créer ou supprimer des branches.
	git branch <nom_branche>	Créer une nouvelle branche.
	git checkout <nom_branche> ou git switch <nom_branche>	Changer de branche active (se déplacer sur une autre branche). (switch est plus récent et recommandé)
	git checkout -b <nom_branche> ou git switch -c <nom_branche>	Créer une nouvelle branche et s'y déplacer immédiatement. (switch -c est plus récent)
	git checkout <commit/fichier> ou git restore <fichier>	Restaurer des fichiers du répertoire de travail à partir d'un commit ou de l'index. (restore est plus récent)
	git merge <nom_branche>	Fusionner l'historique d'une autre branche dans la branche courante.
	git tag <nom_tag> ou git tag -a <nom_tag>	Créer un marqueur (tag) pour référencer un commit spécifique (ex: version v1.0).
Partage et Mise à Jour (Dépôts Distants)	git remote -v	Lister les dépôts distants configurés (souvent nommés origin).
	git remote add <nom> <url>	Ajouter une connexion vers un dépôt distant.
	git fetch <nom_distant>	Récupérer les objets et références (commits, branches, tags) d'un dépôt distant sans fusionner.
	git pull <nom_distant> <branche>	Récupérer les modifications d'un dépôt distant ET les fusionner (Workspace + merge).
	git push <nom_distant> <branche>	Envoyer vos commits locaux vers une branche d'un dépôt distant.
Inspection et Comparaison	git log	Afficher l'historique des commits de la branche courante.

	git log --oneline --graph --decorate --all	Variante utile de git log pour une vue concise et graphique de tout l'historique.
	git diff	Afficher les différences entre le répertoire de travail et l'index.
	git diff --staged ou git diff --cached	Afficher les différences entre l'index et le dernier commit (HEAD).
	git diff <commit1> <commit2> ou git diff <branche1> <branche2>	Afficher les différences entre deux commits ou deux branches.
	git show <commit/tag/objet>	Afficher les informations détaillées sur un objet Git spécifique (commit, tag, blob, tree).
	git blame <fichier>	Afficher qui a modifié quelle ligne d'un fichier et lors de quel commit.
Modification Avancée de l'Historique (Attention!)	git rebase <branche_base>	Réappliquer les commits de la branche courante sur une autre branche de base (modifie l'historique).
	git reset <mode> <commit>	Annuler des commits en déplaçant HEAD. Modes: --soft, --mixed (défaut), --hard (potentiellement destructeur).
	git revert <commit>	Créer un nouveau commit qui annule les modifications introduites par un commit précédent (sécuritaire).
	git cherry-pick <commit>	Appliquer les modifications d'un commit spécifique existant sur la branche courante.
	git stash	Mettre temporairement de côté les modifications non commitées du répertoire de travail.
	git stash pop ou git stash apply	Récupérer les dernières modifications mises de côté.
Administration et Nettoyage	git gc	Garbage Collect : Nettoyer les objets inutiles et optimiser le dépôt local.
	git fsck	File System Check : Vérifier l'intégrité de la base de données Git.

Conseil : Pour obtenir de l'aide détaillée sur une commande spécifique, utilisez `git help <nom_commande>` (par exemple, `git help commit`).