Git Commands Reference Manual

A Comprehensive Guide to Git Commands

Created by Grok, Powered by xAI

July 25, 2025

This document provides a complete list of Git commands, from basic to advanced, with detailed descriptions for effective version control.

1 Introduction

This document serves as a comprehensive reference for Git commands, covering essential operations for version control. Whether you're a beginner or an advanced user, this guide provides clear descriptions of commands for repository management, branching, merging, collaboration, and more. The commands are presented in a structured table with separating lines for easy reference.

2 Git Commands and Descriptions

The following table lists Git commands along with their descriptions, with horizontal lines separating each command for clarity.

Command	**Description**
git init	Initializes a new Git repository in the current directory, creating a .git subdirectory to track changes.
git clone <repo$_{U}RL >$	Downloads a copy of a remote repository to your local machine.
git clone <repo$_{U}RL >$ $-b < branch_{n}ame >$	Clones a specific branch from a remote repository.
git status	Displays the current state of the working directory and staging area, showing modified, staged, and untracked files.
git add <file$_{n}ame >$	Adds a specific file to the staging area for the next commit.
git add -A	Adds all modified and untracked files in the current directory to the staging area.
git commit -m "message"	Commits staged changes with a descriptive message to the local repository.
git commit -a -m "message"	Stages and commits all modified files in a single command.
git log	Shows the commit history for the current branch.
git log –summary	Displays detailed commit history, including additional metadata.
git log –oneline	Shows a condensed commit history with one line per commit.
git log –graph –pretty=oneline	Displays commit history with a visual graph of branches.
tableheader **Command**	**Description**

Command	Description
git diff	Shows changes between the working directory and the staging area.
git diff <source$_{b}ranch><$ $target_{b}ranch>$	Compares changes between two branches.
git diff <commit$_{i}d1><$ $commit_{i}d2>$	Compares changes between two specific commits.
git branch	Lists all branches in the repository, highlighting the current branch.
git branch <branch$_{n}ame>$	Creates a new branch with the specified name.
git checkout <branch$_{n}ame>$	Switches to the specified branch.
git checkout -b <branch$_{n}ame>$	Creates a new branch and switches to it immediately.
git merge <branch$_{n}ame>$	Merges the specified branch into the current branch.
git merge <source$_{b}ranch><$ $target_{b}ranch>$	Merges the source branch into the target branch.
git pull	Fetches and merges changes from the remote repository to the current branch.
git pull –rebase	Fetches remote changes and rebases the current branch on top of them.
git push	Uploads local branch commits to the remote repository.
git push origin <branch$_{n}ame>$	Pushes the specified branch to the remote repository.
git push –force	Force-pushes local changes to the remote repository, overwriting remote history (use with caution).
git remote add origin <repo$_{U}RL>$	Links a local repository to a remote repository.
git remote -v	Lists all remote repositories associated with the local repository.
git fetch	Downloads objects and refs from the remote repository without merging.
tableheader **Command**	**Description**

Command	**Description**
git reset <file$_name>$	Unstages a file while preserving its changes in the working directory.
git reset –hard <commit$_id>$	Resets the repository to a specific commit, discarding all changes after it.
git revert <commit$_id>$	Creates a new commit that undoes the changes introduced by the specified commit.
git stash	Temporarily saves uncommitted changes to allow switching branches.
git stash pop	Applies the most recent stashed changes and removes them from the stash.
git stash clear	Removes all stashed changes.
git stash apply	Applies the most recent stashed changes without removing them from the stash.
git rm <file$_name>$	Removes a file from the working directory and stages the deletion.
git rm -r <directory$_name>$	Recursively removes a directory and its contents, staging the deletion.
git config –global user.name ”Your Name”	Sets the global username for Git commits.
git config –global user.email ”your.email@example.com”	Sets the global email for Git commits.
git config –list	Displays all Git configuration settings.
git rebase <branch$_name>$	Reapplies commits from the current branch onto the specified branch, creating a linear history.
git rebase -i <commit$_id>$	Starts an interactive rebase to edit, squash, or reorder commits starting from the specified commit.
git cherry-pick <commit$_id>$	Applies the changes from a specific commit to the current branch.
git tag <tag$_name>$	Creates a lightweight tag at the current commit.
git tag -a <tag$_name>$ $-m$”$message$”	Creates an annotated tag with a message.
git push origin <tag$_name>$	Pushes a specific tag to the remote repository.
tableheader **Command**	**Description**

Command	**Description**
git push origin –tags	Pushes all tags to the remote repository.
git blame <file$_name$>	Shows who last modified each line of a file and in which commit.
git bisect start	Begins a binary search to find a commit that introduced a bug.
git bisect good <commit$_id$>	Marks a commit as good during a bisect session.
git bisect bad <commit$_id$>	Marks a commit as bad during a bisect session.
git bisect reset	Ends a bisect session and returns to the original branch.
git clean -f	Removes untracked files from the working directory.
git clean -fd	Removes untracked files and directories from the working directory.
git reflog	Shows a log of all reference changes, useful for recovering lost commits.
git archive –format=zip –output=<file$_name$> $.zip < branch_name >$	Creates a zip archive of the specified branch.

Table 1: Git Commands and Their Descriptions

tableheader **Command** **Description**

3 Conclusion

This reference manual provides a complete guide to Git commands, enabling users to manage version control effectively. Each command is separated by a horizontal line for improved readability. For further details, consult the official Git documentation or reach out to the community for support.

Official Git Documentation