**Git Cheat Sheet**

Last Updated : 30 Jul, 2025

**Git Cheat Sheet** is a concise, well-structured guide for developers and DevOps engineers ideal for both beginners and experienced users. It covers everything from Git installation (Linux, Windows, macOS) to configuration, core commands, branching, merging, history management, and collaboration. Perfect for quick reference during development and deployment.

**Git Installation Commands**

Here are the Git installation commands for different operating systems:

| **Commands** | **Description** |
| --- | --- |
| Git for Windows stand-alone installer. | For more Details [Read Here](https://www.geeksforgeeks.org/installation-guide/how-to-install-git-on-windows-command-line/) |
| brew install git | Install Git with **Homebrew** on Mac OS |
| sudo port selfupdate | Install Git with **MacPorts** on Mac OS |
| sudo apt-get install git | Install Command for Linux |
| git --version | Shows the current version of your Git |

**Git Configuration & Setup**

Here are Git configuration and setup commands:

| **Commands** | **Description** |
| --- | --- |
| git config --global user.name "Your Name" | Set your Git username globally for all repositories. |
| git config --global user.email "youremail@example.com" | Set your Git email address globally. |
| git config --global color.ui auto | Set to display colored output in the terminal for better readability. |
| git config --global alias. <alias-name> <git-command> | Create a custom alias for a Git command to save time. |
| git config --list | List all Git configuration settings (global, system, local). |
| git config --get <key> | Retrieve the value of a specific configuration key (e.g., user.name). |
| git help | Display the main help documentation, showing a list of commonly used Git commands. |

**Initializing a Repository**

Here are the Git initializing a repository commands:

| **Commands** | **Description** |
| --- | --- |
| git init | Initializes a new Git repository in the current directory. |
| git init <directory> | Creates a new Git repository in the specified directory. |
| git clone <repository\_url> | Clone a repository from a remote server to your local machine. |
| git clone --branch <branch\_name> <repository\_url> | Clones a specific branch from a remote repository. |

**Basic Git Commands**

Here are some basic Git commands:

| **Commands** | **Description** |
| --- | --- |
| git add <file> | Adds a specific file to the staging area. |
| git add . or git add --all | Adds all modified and new files to the staging area. |
| git status | Shows the current state of your repository, including tracked and untracked files, modified files, and branch information. |
| git status --ignored | Displays ignored files in addition to the regular status output. |
| git diff | Shows the changes between the working directory and the staging area (index). |
| git diff <commit1> <commit2> | Displays the differences between two commits. |
| git diff --staged or git diff --cached | Displays the changes between the staging area (index) and the last commit. |
| git diff HEAD | Display the difference between the current directory and the last commit |
| git commit | Creates a new commit with the changes in the staging area and opens the default text editor for adding a commit message. |
| git commit -m "<message>" or git commit --message "<message>" | Creates a new commit with the changes in the staging area and specifies the commit message inline. |
| git commit -a or git commit --all | Commits all modified and deleted files in the repository without explicitly using git add to stage the changes. |
| git notes add | Creates a new note and associates it with an object (commit, tag, etc.). |
| git restore <file> | Restores the file in the working directory to its state in the last commit. |
| git reset <commit> | Moves the branch pointer to a specified commit, resetting the staging area and the working directory to match the specified commit. |
| git reset --soft <commit> | Moves the branch pointer to a specified commit, preserving the changes in the staging area and the working directory. |
| git reset --hard <commit> | Moves the branch pointer to a specified commit, discarding all changes in the staging area and the working directory, effectively resetting the repository to the specified commit. |
| git rm <file> | Removes a file from both the working directory and the repository, staging the deletion. |
| git mv <old> <new> | Moves or renames a file or directory in your Git repository. |

**Git Commit (Updated Commands)**

Here are some of the updated commands for Git commit:

| **Commands** | **Description** |
| --- | --- |
| git commit -m "feat: message" | Create a new commit in a Git repository with a specific message to indicate a new feature commit in the repository. |
| git commit -m "fix: message" | Create a new commit in a Git repository with a specific message to fix the bugs in codebases |
| git commit -m "chore: message" | Create a new commit in a Git repository with a specific message to show routine tasks or maintenance. |
| git commit -m "refactor: message" | Create a new commit in a Git repository with a specific message to change the code base and improve the structure. |
| git commit -m "docs: message" | Create a new commit in a Git repository with a specific message to change the documentation. |
| git commit -m "style: message" | Create a new commit in a Git repository with a specific message to change the styling and formatting of the codebase. |
| git commit -m "test: message" | Create a new commit in a Git repository with a specific message to indicate test-related changes. |
| git commit -m "perf: message" | Create a new commit in a Git repository with a specific message to indicate performance-related changes. |
| git commit -m "ci: message" | Create a new commit in a Git repository with a specific message to indicate the continuous integration (CI) system-related changes. |
| git commit -m "build: message" | Create a new commit in a Git repository with a specific message to indicate the changes related to the build process. |
| git commit -m "revert: message" | Create a new commit in a Git repository with a specific message to indicate the changes related to revert a previous commit. |

**Branching and Merging**

Here are some Git branching and merging commands:

| **Commands** | **Description** |
| --- | --- |
| git branch | Lists all branches in the repository. |
| git branch <branch-name> | Creates a new branch with the specified name. |
| git branch -d <branch-name> | Deletes the specified branch. |
| git branch -a | Lists all local and remote branches. |
| git branch -r | Lists all remote branches. |
| git checkout <branch-name> | Switches to the specified branch. |
| git checkout -b <new-branch-name> | Creates a new branch and switches to it. |
| git checkout -- <file> | Discards changes made to the specified file and revert it to the version in the last commit. |
| git merge <branch> | Merges the specified branch into the current branch. |
| git log | Displays the commit history of the current branch. |
| git log <branch-d | Displays the commit history of the specified branch. |
| git log --follow <file> | Displays the commit history of a file, including its renames. |
| git log --all | Displays the commit history of all branches. |
| git stash | Stashes the changes in the working directory, allowing you to switch to a different branch or commit without committing the changes. |
| git stash list | Lists all stashes in the repository. |
| git stash pop | Applies and removes the most recent stash from the stash list. |
| git stash drop | Removes the most recent stash from the stash list. |
| git tag | Lists all tags in the repository. |
| git tag <tag-name> | Creates a lightweight tag at the current commit. |
| git tag <tag-name> <commit> | Creates a lightweight tag at the specified commit. |
| git tag -a <tag-name> -m "<message>" | Creates an annotated tag at the current commit with a custom message. |

**Remote Repositories**

Here are some Git remote repositories commands:

| **Commands** | **Description** |
| --- | --- |
| git fetch | Retrieves change from a remote repository, including new branches and commit. |
| git fetch <remote> | Retrieves change from the specified remote repository. |
| git fetch --prune | Removes any remote-tracking branches that no longer exist on the remote repository. |
| git pull | Fetches changes from the remote repository and merges them into the current branch. |
| git pull <remote> | Fetches changes from the specified remote repository and merges them into the current branch. |
| git pull --rebase | Fetches changes from the remote repository and rebases the current branch onto the updated branch. |
| git push | Pushes local commits to the remote repository. |
| git push <remote> | Pushes local commits to the specified remote repository. |
| git push <remote> <branch> | Pushes local commits to the specified branch of the remote repository. |
| git push --all | Pushes all branches to the remote repository. |
| git remote | Lists all remote repositories. |
| git remote add <name> <url> | Adds a new remote repository with the specified name and URL. |
| git remote rm <remote> | Remove a connection to a remote repository (e.g., origin). |
| git remote rename <old\_name> <new\_name> | Rename an existing remote connection. |

**Git Comparison**

Here are some Git comparison commands:

| **Commands** | **Description** |
| --- | --- |
| git show | Shows the details of a specific commit, including its changes. |
| git show <commit> | Shows the details of the specified commit, including its changes. |

**Git Logging and Reviewing Commands**

Here are some Git Logging and Reviewing commands:

| **Command** | **Description** |
| --- | --- |
| git log | Displays the commit history of the current branch. |
| git log --oneline | Shows commits in a compact format (1 line per commit). |
| git log --graph | Displays an ASCII graph of the branch history alongside log output. |
| git log --all | Shows commit logs for all branches. |
| git log --author="Name" | Shows commits made by a specific author. |
| git log --since="2 weeks ago" | Filters commits made in the last 2 weeks. |
| git log --until="2024-12-31" | Shows commits made **before** a specific date. |
| git log <file> | Shows the commit history of a specific file. |
| git log --follow <file> | Tracks file history including renames. |
| git show | Displays the full details of a specific commit (diff + metadata). |
| git show <commit> | Shows information about the given commit hash. |
| git blame <file> | Shows which commit last modified each line of a file. |
| git diff | Compares working directory and staging area (unstaged changes). |
| git diff --staged | Compares staged changes with the last commit. |
| git diff <commit1> <commit2> | Shows the difference between two commits. |

**Git Managing History**

Here are some Git managing history commands:

| **Commands** | **Description** |
| --- | --- |
| git revert <commit> | Creates a new commit that undoes the changes introduced by the specified commit. |
| git revert --no-commit <commit> | Undoes the changes introduced by the specified commit, but does not create a new commit. |
| git rebase <branch> | Reapplies commits on the current branch onto the tip of the specified branch. |

**Git Reflog – Recovering Lost Commits**

Here are some commands for recovering lost commits:

| **Command** | **Description** |
| --- | --- |
| git reflog | Show history of HEAD changes (including resets, rebases, checkouts) |
| git checkout HEAD@{n} | Restore a previous HEAD state |
| git reset --hard HEAD@{n} | Hard reset to a previous state |