

Git Cheat Sheet



Git Basics

| | | |
|--|---|---|
| git init <directory> | Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository. | |
| git clone <repo> | Clone repo located at <repo> onto local machine. Original repo can be located on the local filesystem or on a remote machine via HTTP or SSH. | |
| git config user.name <name> | Define author name to be used for all commits in current repo. Devs commonly use --global flag to set config options for current user. | + |
| git add <directory> | Stage all changes in <directory> for the next commit. Replace <directory> with a <file> to change a specific file. | |
| git commit -m "<message>" | Commit the staged snapshot, but instead of launching a text editor, use <message> as the commit message. | |
| git status | List which files are staged, unstaged, and untracked. | |
| git log | Display the entire commit history using the default format. For customization see additional options. | + |
| git diff | Show unstaged changes between your index and working directory | + |

Undoing Changes

| | | |
|--|---|---|
| git revert <commit> | Create new commit that undoes all of the changes made in <commit>, then apply it to the current branch. | |
| git reset <file> | Remove <file> from the staging area, but leave the working directory unchanged. This unstages a file without overwriting any changes. | + |
| git clean -n | Shows which files would be removed from working directory. Use the -f flag in place of the -n flag to execute the clean. | |

Rewriting Git History

| | | |
|--------------------------------|--|---|
| git commit --amend | Replace the last commit with the staged changes and last commit combined. Use with nothing staged to edit the last commit's message. | |
| git rebase <base> | Rebase the current branch onto <base>. <base> can be a commit ID, a branch name, a tag, or a relative reference to HEAD. | + |
| git reflog | Show a log of changes to the local repository's HEAD. Add --relative-date flag to show date info or --all to show all refs. | |

Git Branches

| | | |
|---|---|--|
| git branch | List all of the branches in your repo. Add a <branch> argument to create a new branch with the name <branch>. | |
| git checkout -b <branch> | Create and check out a new branch named <branch>. Drop the -b flag to checkout an existing branch. | |
| git merge <branch> | Merge <branch> into the current branch. | |

Remote Repositories

| | | |
|--|---|---|
| git remote add <name> <url> | Create a new connection to a remote repo. After adding a remote, you can use <name> as a shortcut for <url> in other commands. | |
| git fetch <remote> <branch> | Fetches a specific <branch>, from the repo. Leave off <branch> to fetch all remote refs. | |
| git pull <remote> | Fetch the specified remote's copy of current branch and immediately merge it into the local copy. | + |
| git push <remote> <branch> | Push the branch to <remote>, along with necessary commits and objects. Creates named branch in the remote repo if it doesn't exist. | + |

Additional Options +



git config

| | |
|---|---|
| <code>git config --global user.name <name></code> | Define the author name to be used for all commits by the current user. |
| <code>git config --global user.email <email></code> | Define the author email to be used for all commits by the current user. |
| <code>git config --global alias.<alias-name> <git-command></code> | Create shortcut for a Git command. E.g. <code>alias.glog "log --graph --oneline"</code> will set "git glog" equivalent to "git log --graph --oneline" |
| <code>git config --system core.editor <editor></code> | Set text editor used by commands for all users on the machine. <editor> arg should be the command that launches the desired editor (e.g., vi). |
| <code>git config --global --edit</code> | Open the global configuration file in a text editor for manual editing. |

git log

| | |
|---|---|
| <code>git log -<limit></code> | Limit number of commits by <limit> . E.g. "git log -5" will limit to 5 commits |
| <code>git log --oneline</code> | Condense each commit to a single line. |
| <code>git log --stat</code> | Include which files were altered and the relative number of lines that were added or deleted from each of them. |
| <code>git log -p</code> | Display the full diff of each commit. |
| <code>git log --author="<pattern>"</code> | Search for commits by a particular author. |
| <code>git log --grep="<pattern>"</code> | Search for commits with a commit message that matches <pattern>. |
| <code>git log <since>..<until></code> | Show commits that occur between <since> and <until>. Args can be a commit ID, branch name, HEAD, or any other kind of revision reference. |
| <code>git log -- <file></code> | Only display commits that have the specified file. |
| <code>git log --graph --decorate</code> | --graph flag draws a text based graph of commits on left side of commit msgs. --decorate adds names of branches or tags of commits shown. |

git diff

| | |
|--------------------------------|--|
| <code>git diff HEAD</code> | Show difference between working directory and last commit. |
| <code>git diff --cached</code> | Show difference between staged changes and last commit. |

git reset

| | |
|--|---|
| <code>git reset</code> | Reset staging area to match most recent commit, but leave the working directory unchanged. |
| <code>git reset --hard</code> | Reset staging area and working directory to match most recent commit and overwrites all changes in the working directory. |
| <code>git reset <commit></code> | Move the current branch tip backward to <commit>, reset the staging area to match, but leave the working directory alone. |
| <code>git reset --hard <commit></code> | Same as previous, but resets both the staging area & working directory to match. Deletes uncommitted changes, and all commits after <commit>. |

git rebase

| | |
|---|---|
| <code>git rebase -i <base></code> | Interactively rebase current branch onto <base>. Launches editor to enter commands for how each commit will be transferred to the new base. |
|---|---|

git pull

| | |
|---|---|
| <code>git pull --rebase <remote></code> | Fetch the remote's copy of current branch and rebases it into the local copy. Uses git rebase instead of merge to integrate the branches. |
|---|---|

git push

| | |
|--|---|
| <code>git push <remote> --force</code> | Forces the <code>git push</code> even if it results in a non-fast-forward merge. Do not use the <code>--force</code> flag unless you're absolutely sure you know what you're doing. |
| <code>git push <remote> --all</code> | Push all of your local branches to the specified remote. |
| <code>git push <remote> --tags</code> | Tags aren't automatically pushed when you push a branch or use the <code>--all</code> flag. The <code>--tags</code> flag sends all of your local tags to the remote repo. |