

# Git Cheat Sheet



## GIT BASICS

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

## UNDOING CHANGES

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

## REWRITING GIT HISTORY

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

## GIT BRANCHES

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

## REMOTE REPOSITORIES

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