

# Additional Options +



## git config

<code>git config --global user.name &lt;name&gt;</code>	Define the author name to be used for all commits by the current user.
<code>git config --global user.email &lt;email&gt;</code>	Define the author email to be used for all commits by the current user.
<code>git config --global alias.&lt;alias-name&gt; &lt;git-command&gt;</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 &lt;editor&gt;</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 -&lt;limit&gt;</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="&lt;pattern&gt;"</code>	Search for commits by a particular author.
<code>git log --grep="&lt;pattern&gt;"</code>	Search for commits with a commit message that matches <pattern>.
<code>git log &lt;since&gt;..&lt;until&gt;</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 -- &lt;file&gt;</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 <b>overwrites all changes</b> in the working directory.
<code>git reset &lt;commit&gt;</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 &lt;commit&gt;</code>	Same as previous, but resets both the staging area & working directory to match. <b>Deletes</b> uncommitted changes, and <b>all commits after</b> <commit>.

## git rebase

<code>git rebase -i &lt;base&gt;</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 &lt;remote&gt;</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 &lt;remote&gt; --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 &lt;remote&gt; --all</code>	Push all of your local branches to the specified remote.
<code>git push &lt;remote&gt; --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.