

🚩 Current Skill Stashing

Stashing

Sometimes you are working with certain files and do not want to add and commit them, but you do not want to discard them either. For example, maybe you're right in the middle of working on some feature when a huge update gets pushed to your remote upstream, and you need to pull the changes in right away to be sure that the code you're writing still works.

When you try to `pull` or `merge` code and you have changes in your working directory, `git` won't let the `pull` or `merge` to go through. In other words, you can't `merge` code into the branch you're working on **unless your working directory is clean**. So, what should you do if you're working directory isn't clean, but you aren't ready to commit yet? This is a perfect example of where `stashing` can help. You can think of stashing as a temporary way of remembering changes without making an official commit.

Here are the commands you can use when `stashing`:

`git stash` - stash your commits (same as `git stash save`)

`git stash list` - show the list of stashed changes

`git stash apply` - move the latest stashed change back into the working directory, but keep it on the list

`git stash pop` - move the latest stashed change back into the working directory and **remove** it from the list

`git stash show` - show the latest stash

`git stash show stash@{number}` - show a specific stashed change

You can also use `git stash pop/apply stash@{number}` to retrieve a specific stashed change.

Stashing is quite useful when you are not ready to commit something, but need a clean working directory. You can learn more about `git stash` [here](#) and [here](#).



CLONE

Repository

ANGELO BREWING

Code Changes

PULL

git stash

Local

Code
Added

git POP



< Previous

next >