**Stashing and Cleaning:**

Often, when you’ve been ***working on part*** of your project, ***things*** are in a ***messy state*** and you want to ***switch branches*** for a ***bit to work on something else***. The problem is, you ***don’t want to do a commit*** of ***half-done work*** just so you can get back to this point later. The answer to this issue is the ***git stash command***.

**Stashing Your Work**

To demonstrate stashing, you’ll go into your project and start working on a couple of files and possibly stage one of the changes. If you run git status, you can see your dirty state:

* **git status**

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

modified: index.html

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

modified: lib/simplegit.rb

Now you want to switch branches, but you don’t want to commit what you’ve been working on yet, so you’ll stash the changes. To push a new stash onto your stack, run git stash or git stash push:

* **git stash**

Saved working directory and index state \

"WIP on master: 049d078 added the index file"

HEAD is now at 049d078 added the index file

(To restore them type "git stash apply")

You can now see that your working directory is clean:

* **git status**

# On branch master

nothing to commit, working directory clean

At this point, you can switch branches and do work elsewhere; your changes are stored on your stack. To see which stashes you’ve stored, you can use git stash list:

* **git stash list**

stash@{0}: WIP on master: 049d078 added the index file

stash@{1}: WIP on master: c264051 Revert "added file\_size"

stash@{2}: WIP on master: 21d80a5 added number to log

In this case, ***two stashes were saved previously***, so you have ***access to three different stashed works***. You can ***reapply the one you just stashed*** by using the command shown in the help output of the original stash command: **git stash apply**. If you want to apply one of the older stashes, you can specify it by naming it, like this: git stash apply stash@{2}. If you don’t specify a stash, Git assumes the most recent stash and tries to apply it:

* **git stash apply**

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

modified: index.html

modified: lib/simplegit.rb

no changes added to commit (use "git add" and/or "git commit -a")

You can see that Git re-modifies the files you reverted when you saved the stash. In this case, you had a clean working directory when you tried to apply the stash, and you tried to apply it on the same branch you saved it from. Having a clean working directory and applying it on the same branch aren’t necessary to successfully apply a stash. You can save a stash on one branch, switch to another branch later, and try to reapply the changes. You can also have modified and uncommitted files in your working directory when you apply a stash — Git gives you merge conflicts if anything no longer applies cleanly.

The changes to your files were reapplied, but the file you staged before wasn’t restaged. To do that, you must run the git stash apply command with a --index option to tell the command to try to reapply the staged changes. If you had run that instead, you’d have gotten back to your original position:

* **git stash apply --index**

On branch master

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

modified: index.html

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

modified: lib/simplegit.rb

The apply option only tries to apply the stashed work — you continue to have it on your stack. ***To remove it, you can run git stash drop*** with the name of the ***stash to remove***:

* **git stash list**

stash@{0}: WIP on master: 049d078 added the index file

stash@{1}: WIP on master: c264051 Revert "added file\_size"

stash@{2}: WIP on master: 21d80a5 added number to log

* **git stash drop stash@{0}**

Dropped stash@{0} (364e91f3f268f0900bc3ee613f9f733e82aaed43)

You can also run git stash pop to apply the stash and then immediately drop it from your stack.