How to Git Stash Your Work [the Correct Way]

Nesha Zoric May 15 '18 · 3 min read

#git

Imagine that you are working on a part of a project and it starts getting messy. There has been an urgent bug that needs your immediate attention. It is time to **save** your changes and switch branches. *The problem is, you don't want to do a commit of half-done work.* The solution is `git stash`.

> Stashing is handy if you need to quickly switch context and work on something else but you're mid-way through a code change and aren't quite ready to commit. By Bitbucket

Stashing

Let's say you currently have a couple of local modifications. Run `git status`, to check your current state:

```
$ git status
# 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)
#      modified:   assets/stylesheets/styles.css
```

You need to work on that urgent bug. First, you want to save out unfinished work changes without committing them. This is where `git stash` comes as a savior:

```
$ git stash
Saved working directory and index state WIP on master:
  bb06da6 Modified the index page
HEAD is now at bb06da6 Modified the index page
(To restore them type "git stash apply")
```

Your **working directory** is now clean and all uncommitted local changes have been saved! At this point, you're free to make new changes, create new commits, switch branches, and perform any other Git operations.

> By default, stashes are identified as "WIP" – work in progress, on top of the branch and commit they are created from.

Re-applying Your Stash

Git stash is a temporary storage. When you're ready to continue where you left off, you can restore the saved state easily: `git stash pop`.

Popping your stash removes the changes from your stash and reapplies the last saved state. If you want to **keep the changes** in the stash as well, you can use `git stash apply` instead.

Additional Tips and Tricks

There are a couple of other things you can do with a stash. Let's take a look!

- **Saving stashes**
  Save a stash with a **message**: `$ git stash save <message>`.

  > Try this out by adding CSS-line high to your styles and stash it with a nice comment.

- **Stashing untracked files**
  This is the only way to save **untracked files**: `$ git stash -u` or `$ git stash --include-untracked`
- **List multiple stashes**
  When you `git stash` or `git stash save`, Git will create a Git commit

Nesha Zoric
@neshaz
Founder/CTO at Kolosek. Building tech starups. Starting project? Contact at https://kolosek.com/startproject/
+ FOLLOW
work
CEO
education
MA in Computer Science
joined
Feb 28, 2018

object with a name and then save it in your repo. You can view the **list of stashes** you made at any time! `$ git stash list`.

```
$ git stash list
stash@{0}: On master: Modified the index page
stash@{1}: WIP on master: bb06da6 Initial Commit
```

- **Partial stashes**
  You can choose to **stash** just a single file, a collection of files, or individual changes from within files: `$ git stash -p` or `$ git stash --patch`.

  > RSpec tests are a must in the Ruby on Rails projects, but they might not be always complete. Stash only the part that is ready to go!

- **Viewing stash diffs**
  There are two ways to view a stash: to view the **full diff** of a stash **-** `$ git stash show -p` or view only the **latest stash -** `$ git stash show`.

```
$ git stash show
index.html | 1 +
style.css | 2 ++
2 files changed, 3 insertions(+)
```

- **Creating a branch from the stash**
  Create a **new branch** to apply your stashed changes to, and then pop your stashed changes onto it: `$ git stash branch <branch_name> <stash_id>`.

  > This is another way to save your stash before moving on with the project.

- **Remove your stash**
  Use it with caution, it maybe is difficult to revert. The only way to revert it is if you didn't close the terminal after deleting the stash.
  If you no longer need a **particular stash**, you can delete it with: `$ git stash drop <stash_id>`. Or you can **delete all** of your stashes from the repo with: `$ git stash clear`.

Hope this article helped you to get a better understanding how stashing works. Be sure to test it out!

This article is originally published on Kolosek Blog.

Nesha Zoric + FOLLOW
Founder/CTO at Kolosek. Building tech starups. Starting project? Contact at https://kolosek.com/startproject/
@neshaz nebojsaz kolosek.com

Add to the discussion

PREVIEW SUBMIT

Mateen Apr 17

Thanks for the info!

♡ 1 REPLY

code of conduct - report abuse

Classic DEV Post from Jan 16

Is 2019 the year of TypeScript?

Nick Taylor

Also published at iamdeveloper.com on Jan 16, 2019 Photo by NordWood Themes on...

100 68

Nick Taylor

+ FOLLOW

Another Post You Might Like

Git Bisect is Easy (How to Initiate the Robot Uprising)

Jacob Herrington

A simple guide to git bisect

107 14

Jacob Herrington

+ FOLLOW

Another Post You Might Like

How To Use Git: A Reference Guide

Lisa Tagliaferri

A handy reference guide on how to use git.

288 4

DigitalOcean

+ FOLLOW

Using Kentico CI and Git to help with releases
mattnield - Oct 24

Launch git-fork app from WSL
Lucas Dasso - Oct 25

Github: SAML, Okta, and Github Enterprise Cloud – Organization SSO configuration
Arseny Zinchenko - Oct 21

Day1 : Drum Kit in Vanilla JS
Abhishek Naidu - Oct 28

Home About Privacy Policy Terms of Use Contact Code of Conduct