**Git Stash**

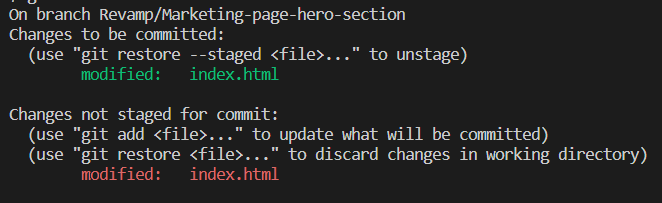
Let's say you're working on a serious feature of a branch in Git – like revamping the hero section of your marketing page. You'll want to start doing experiments in your Revamp/Marketing-page-hero-section branch without screwing up master or main branch.

Then suddenly you get call from your coworker to fix some bugs on the login-page branch. It's a serious issue. So, you try to switch to the login-page branch using git switch login-page or git checkout login-page.

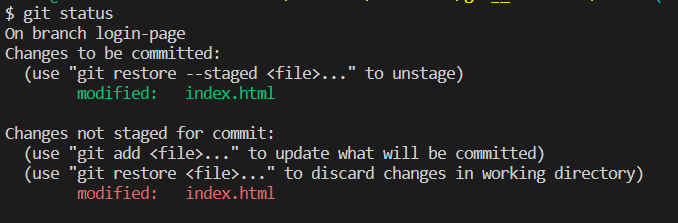
If you're switching branches with staged and unstaged changes, you might encounter any of the following scenarios:

First, while switching to the login-page branch, the staged and unstaged changes of the Revamp/Marketing-page-hero-section branch will come with you to the login-page branch.

The Revamp/Marketing-page-hero-section branch contains some staged and unstaged changes on index.html.

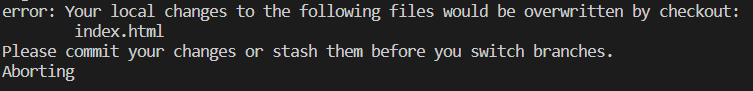


If we switch to the login-page branch, the staged and unstaged changes on Revamp/Marketing-page-hero-section branch come to login-page also.



This muddies the login-page branch.

Next, sometimes Git doesn't allow you to switch branches without committing those changes. This is because you might lose the changes you made in your current branch or they may conflict with the destination branch (login-page). Whatever the reason, we can't switch the branch without committing or stashing the changes.



At the same time, you can't commit the half-done feature branch.

To help with this, you can use **Git stashing**.Stash means to store (changes) safely in a hidden place (the stash stack).

Stashing the current working directory's staged or unstaged changes or untracked files and then storing them in the stash stack reverts the current working directory to the last commit.

**How to stash your changes 🤐**

You can use any one of the below commands to **stash your staged and unstaged changes in the stash stack**. It undoes things to the latest commit and doesn't delete the changes, which are stored in the stash stack.

git stash

or

git stash save

**How to stash untracked files❗**

Want to stash your untracked files in the stash stack? Just use the --include-untracked flag at the end of the command.

git stash --include-untracked

or use -u at the end of the command:

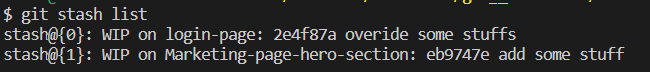
git stash -u

**How to list stashes 📃**

Use the below command to list out all the stashes stored in the stash stack:

git stash list

Listing the stashes like below,



* The latest stashes (stash@{0}) will be at the top of the stack.
* The older stashes (stash@{1}) will be at the bottom of the stack.

**Understanding the format of stash**

The stash command lists the stashes in the below format:

A single stash format

If you're not exactly clear on what this is saying 😅, it's completely fine. Let's explore some of the terms on the stash listing:

* **Stash@{0}** – this is just a stash reference. It refers to the particular stash. By default, Stash@{0} is always the latest stash.

*Note: Higher numbered stashes like stash@{3} are older stashes. The latest stashes always have the lowest number.*

* **WIP On fake –** *fake* is just a branch name like any other branch and WIP stands for Work In Progress. "*WIP on fake"* means that stash@{0} was created on the branch "fake".
* **fc99b30 add head line** – *fc99b30* is aCommit hash and *add head line* is acommit message. At that time of stash creation, *fc99b30 add head line* is the latest commit.

**How to show the latest stash 📺**

Maybe you have multiple stashes in your stash stack and you're not able to tell which stash reference holds which changes.

So, before you apply stashes on the current working branch, you can confirm and show the changes recorded in the stash with the below command:

git stash show

By default git stash show shows the changes recorded in the **latest** stash (stash@{0}) in the --stat format.

The --stat format shows only how many lines you've added and deleted on each of the modified files.

readme.md | 2 +-

1 file changed, 1 insertion(+), 1 deletion(-)

If you want to show the recorded changes of the latest stash on **patch view**, use the -p flag at the end of the command, like this:

git stash show -p

If you also want to show **untracked files**, use the -u flag.

git stash show -u

or you can use the --include-untracked flag like this:

git stash show --include-untracked

You can show untracked files with the **patch format**:

git stash show -p -u

You can also show **only** untracked files with the patch format like this:

git stash show -p --only-untracked

Wow, you've already learned 10+ stash commands 🎉🎉

Let's look at some more so you can get the most out of stash.

**How to show an individual stash 📺**

You can show the recorded changes of an individual stash by using **stash reference**.

git stash show stash@{1}

For the patch format, you guessed it right 👏🏻👏🏻 – use the -p flag.

git stash show stash@{1} -p

Want to show a stash with untracked files? Use this command:

git stash show stash@{1} -u

or this one:

git stash show stash@{1} --include-untracked

Whereas you can do this to show untracked files only:

git stash show stash@{1} --only-untracked

**How to apply the stash 🖊️**

To apply the recorded changes of your latest stash on the current working branch as well as remove that stash from the stash stack, run this command:

git stash pop

*Note: We can apply stashes on any branch. It's not specific to the branch where the stash was created.*

You can also apply the latest stash without removing the stash from the stash stack like this:

git stash apply

You can apply an earlier stash by using the stash reference:

git stash apply stash@{3}

**How to delete a stash ☠️**

Want to clear all the stashes from stash stack? Use this command:

git stash clear

Want to delete a particular stash? Yes! you are right – you use the stash reference:

git stash drop stash@{2}

**How to create a branch from stash**

Yes, you can create a new branch from your latest stash. Just use this command:

git stash branch <branch\_name>

For instance,

git stash branch demo

If you want to create a branch from an earlier stash, that's also possible by using stash reference:

git stash branch <branch\_name> stash@{revision}

For instance,

git stash branch purple stash@{3}