CodeKraft

# I Git! Stashing Explained

*by* ABDULFATTAAH POPOOLA *on* JUNE 5, 2014JUNE 7, 2014

Imagine these scenarios when working in Git:

1. You do not want to commit your unfinished work but need to switch to a new branch to start some new work immediately.

2. A git pull results in conflicts however you do not want to commit local changes yet.

3. You realize you have worked in the wrong branch and want to move your uncommitted work to the right branch.

The Git stash command would help in these situations – it takes all the changes in your working directory and puts them in a stack you can always access later.

The command does two things:

1. It saves the working directory and index to a safe temporary place (the latest stash is usually at *.git/refs/stash*).

2. Then, it restores the working directory and index to the most recent commit (i.e. the commit pointed to by HEAD).

Thus, you can go ahead and switch to a new branch or complete the pull after a stash.

```
1 | git stash
```

It might fail if you do not have any commit in the repository. You need to have at least a single revision as shown below:

```
1 | git init
2 | git stash
3 | //fatal: Bad revision 'HEAD'
4 | //fatal: Bad revision 'HEAD'
5 | //fatal: Needed a single revision
6 | //You do not have the initial commit yet
```

Now, let's try again after making some commits to the repository

```
echo "Temp" > temp
git add temp
git commit -m "stash first commit"
git stash
//No local changes to save
```

Next, lets stash some temporary changes; note that stashing does not save changes that are being tracked (more on tracking vs staging areas in an upcoming post insha Allaah).

```
echo "staging" >> temp

#Add changes to staging area
git add temp
git status
#Should show staged changes to temp

echo "tracking" > tracking
git status
#shows staged and tracked changes

git stash
#Saved working directory and index state
#WIP on master: 2038ddd stash

git status
#clean staging area
#Tracked changes unmodified
```

**Using stashed work**

The stash is a stack and can contain multiple changes. The following commands show how to interact with it.

```
#list all stashed changes
git stash list

#apply the topmost stashed changes
git stash apply

#Show applied changes
git status

#Stash still has applied stash
git stash list

#Pop the stashed change at ref 2
git stash pop stash@{2}

#Show applied changes
git status

#Verify stash@{2} was removed
git stash list

#delete stash at ref 3
git stash drop stash@{3}

#Delete all stash entries
git stash clear
```

The difference between apply and pop is simple: apply is non-destructive, it preserves the stashed entry on the stack while pop will remove it. The pop command can be seen as an *apply* + *drop* combo. And yes, it is possible to create a new branch from a stashed entry.

```
# create a new branch from stash
git stash branch newBranchForStash
```

It automatically checks out the branch too; but make sure the branch name is unique!

Done! Happy Stashing!!

Published In Software Development Tagged

## 2 Comments · LEAVE A REPLY

This site uses Akismet to reduce spam. Learn how your comment data is processed.