**git stash**

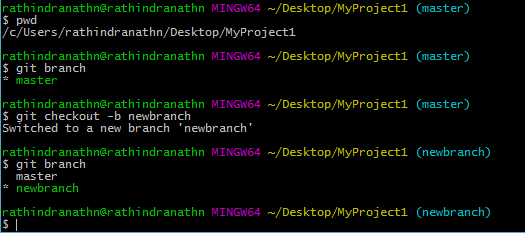
**Using 'git stash' to Store Changes for Later**

**By Short and Sweet Courses**

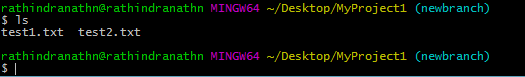
<https://www.youtube.com/watch?v=FfKSu1kfK2Q>

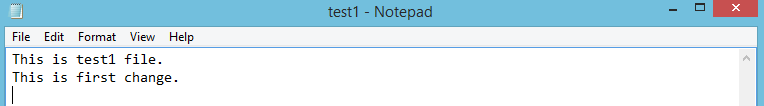
Here we shall talk about git stash. What is git stash? We can think git stash a temporary storage box where we keep the changes which we are not ready to commit.

Suppose we are in master branch and everything is going file there. Now we want to make few changes in a file in our current project folder. For that we prefer to create a branch and make the changes there instead of doing the same in master branch.



Let’s make change to the file test1.txt.

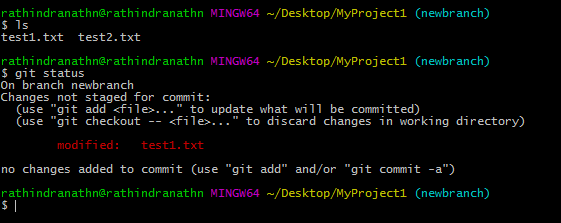




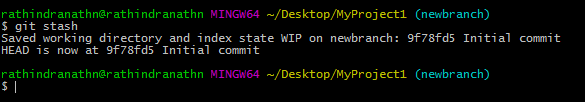
A new line is added.

When we do the changes, suddenly we receive the notification that something is wrong in master branch. So we can’t keep on working on the present change, rather we shall switch our attention to the issue, which our project is facing elsewhere. For that we need to checkout master branch, where actual production copy of the project resides. But we can’t lose the changes, which we have done so far in another branch. Since we are in the middle of the change – in other words the change is incomplete, we can’t commit the change. What can I do at this juncture – we can neither discard the change done so far nor commit it? Git has come with an excellent solution to overcome such situation and it is “git stash”. Using this facility of git we can temporarily store whatever change we made so far. After that we can switch the branch where our project resides and facing major issue. After fixing the issue we can again come back to the previous branch and finish our incomplete work.

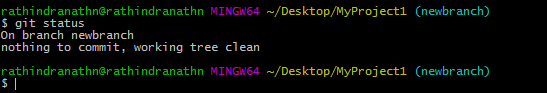
Now we shall close the file test1, which contains incomplete change (adding a new line).

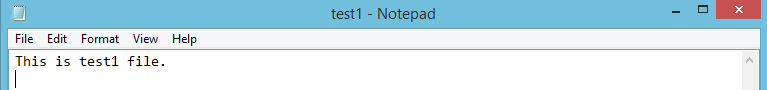


Now we shall execute the command “git stash”.



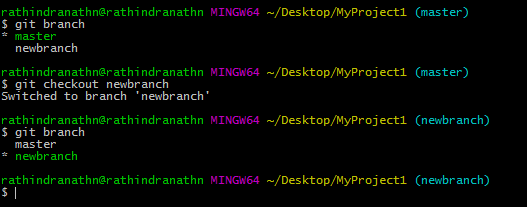
From the output of the command it is clear that changes made to the file test1 is saved in “newbranch” branch. If we execute “git status” command, current branch will be shown as clean. Also if we check the file teat1 we can’t see the newly added line.



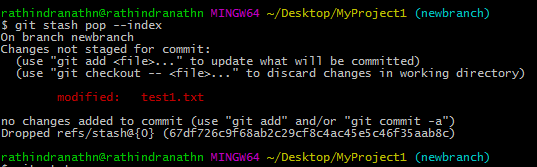


It seems no modification is made to the file. In fact all the changes have moved to the temporary storage box.

Now we can switch back to our master branch and start working on the project kept there to deal with the problem it faces. After our working is done we can again go back to our previous branch and resume our work there.

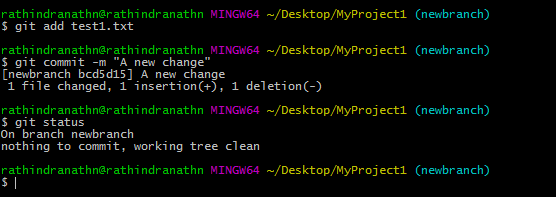


Now we are in our newly created branch. Let’s unbox our changes – get the changes out of the box. Following command we need to execute,



What does the “—option” do is if we have staged any change before putting the changes in the box it will pull back those changes and everything will be as before. In our case we have not staged any change. Therefore this option is useless here.

Above output tells that our previous change to the file in this branch is back again. We can continue our change. After doing the changes we can move them to the staging area and then commit.



If we stash any change and then come back to the concerned branch, it is advisable to pull those changes back and then work on it (if needed) and then commit before making any change in the same branch. It helps us to avoid merge conflict.

**Learn Stash Command - Learn Git and Github - Part #4**

**By Shoaib Bhimani**

<https://www.youtube.com/watch?v=ibo_wlpmcsE>

Here we shall discuss about trash command. Suppose I am working in a project and suddenly my boss comes to me tells me to work in a different task. In that case I need to engage in another task leaving the current one. In the current task I have worked on few features, which I don’t want to discard. Again, I can’t commit my work since it has not been completed yet. In such scenario “git trash” comes to our rescue.

Following are the commands related to “git trash”,

git stash save “<Comment/Description>”

git stash list

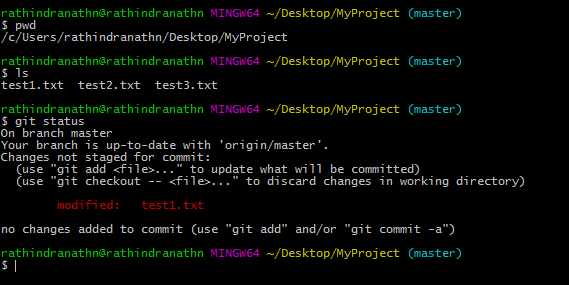
git stash apply {uniqueID}

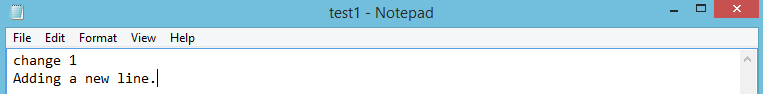
git stash drop {uniqueID}

git stash pop

git stash clean

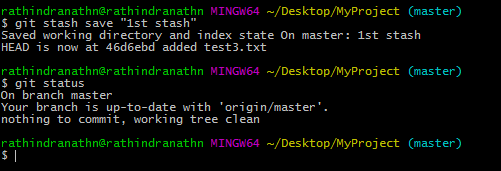
When we execute the command “git stash” the changes are saved in a temporary storage area. Let’s do some changes in a file test1 in our project directory MyProject. We have added one line.

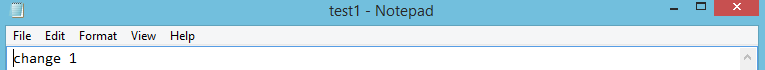




“git status” command output shows that the file test1 is modified.

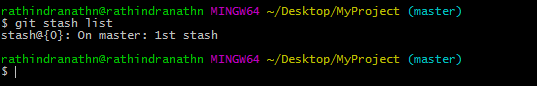
Now we shall stash the change,





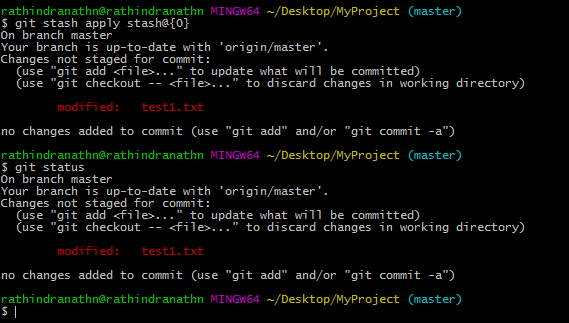
After running “git sash” command our working directory is clean and there is nothing to commit. The added line is also deleted from test1 file.

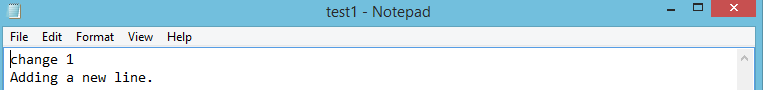
To see the list of stash we need to execute following command,



One stash id is generated, which is “stash@{0}”. Each stash also bear a comment or description which is given at the time creating the stash.

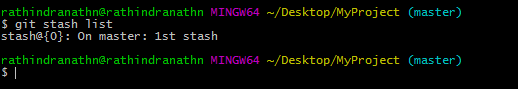
If we want to get back the changes and work on it we need to execute following command,



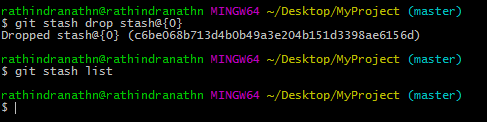


“git status” command output shows that test1 is modified and also newly added line has appeared.

Let’s do the listing of the stashes.



Still the stash “stash@{0}” exists even if it is applied. We can delete it explicitly and for that we need to execute the following command,



If we have multiple stash and we want to delete all of them at one go we need to execute following command,

git stash clean

**Git Stash**

**By TechJini University**

<https://www.youtube.com/watch?v=kPnm1Lc_490>

“git stash pop” command deletes the stash after restoring the change.

We can apply stash in any branch irrespective of the branch in which it is created.

If we don’t want to delete a stash after we restore it we need to execute the command “git stash apply <stash id>”

Stash is actually useful when we are in middle of something and got to know that there are upstream changes which are relevant to what we are doing. In such a scenario we can simply stash the changes we made and restore those changes later.

**Git Tutorial: Using the Stash Command**

**By Corey Schafer**

<https://www.youtube.com/watch?v=KLEDKgMmbBI>

**006 Git Stash**

**By Dan Gitschooldude**

<https://www.youtube.com/watch?v=Fk-Gb4MfTd8>