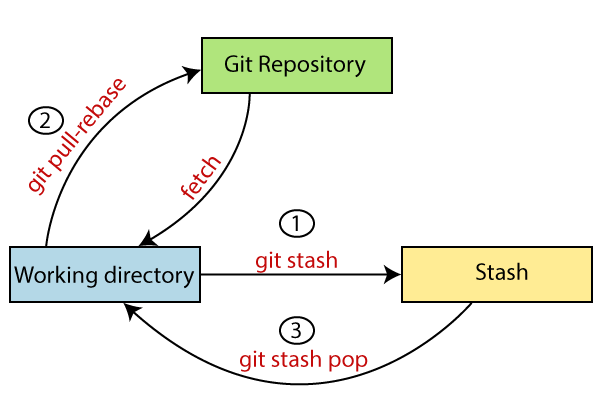
**Assignment no. 1**

**Git Stash**

Sometimes you want to switch the branches, but you are working on an incomplete part of your current project. You don't want to make a commit of half-done work. Git stashing allows you to do so. The **git stash command** enables you to switch branches without committing the current branch.

The below figure demonstrates the properties and role of stashing concerning repository and working directory.



Generally, the stash's meaning is "**store something safely in a hidden place**." The sense in Git is also the same for stash; Git temporarily saves your data safely without committing.

Stashing takes the messy state of your working directory, and temporarily save it for further use. Many options are available with git stash. Some useful options are given below

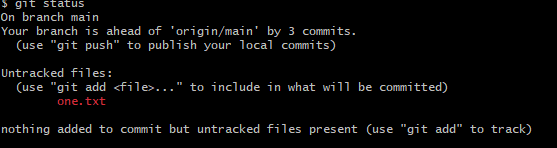
* **Git stash**
* **Git stash save**
* **Git stash list**
* **Git stash apply**
* **Git stash changes**
* **Git stash pop**
* **Git stash drop**
* **Git stash clear**
* **Git stash branch**

## **Stashing Work**

## Let's understand it with a real-time scenario. I have made changes to my project GitExample2 in two files from two distinct branches. I am in a messy state, and I have not entirely edited any file yet. So I want to save it temporarily for future use. We can stash it to save as its current status. To stash, let's have a look at the repository's current status. To check the current status of the repository, run the git status command. The git status command is used as:

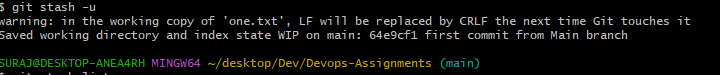
**Syntax**

**$ Git status**

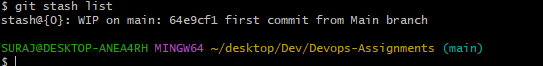


**Note-From the above the output using git status command we can see status of un -tracked file**

**$ Git stash**

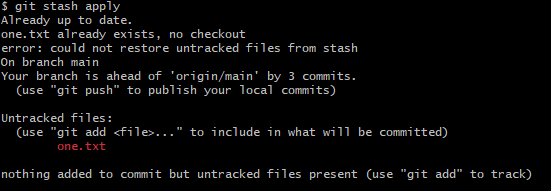
**Note-we can use Git stash command for saving the temporary file.**

**$ Git stash list**

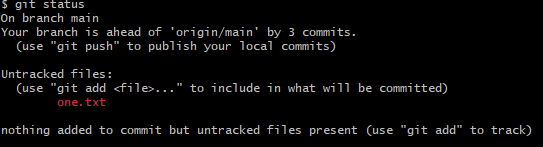


**Note:- To check the stored stashes.**

**$ Git stash apply**



The above output restores the last stash. Now, if you will check the status of the repository, it will show the changes that are made on the file. Consider the below **output:**

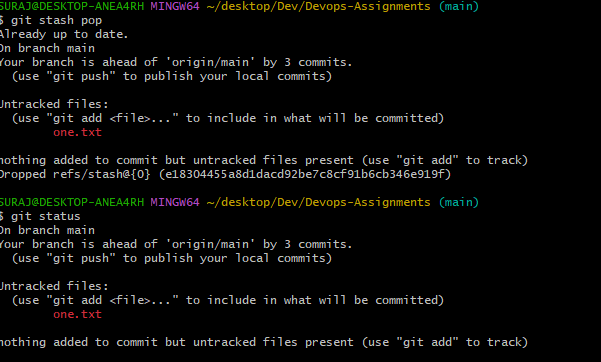


**Git Stash Pop (Reapplying Stashed Changes)**

Git allows the user to re-apply the previous commits by using stash pop command. The popping option removes the changes from stash and applies them to your working file.

The stash pop command is quite similar to stash apply. The main difference between both of these commands is stash pop command that deletes the stash from the stack after it is applied.

**$ git stash pop**



## **Git Stash Drop (Unstash)**

The **git stash drop** command is used to delete a stash from the queue. Generally, it deletes the most recent stash. Caution should be taken before using stash drop command, as it is difficult to undo if once applied.

The only way to revert it is if you do not close the terminal after deleting the stash. The stash drop command will be used as:

**$ git stash drop**

**Output:**



In the above output, the most recent stash **(stash@{0})** has been dropped from given three stashes. The stash list command lists all the available stashes in the queue.



**Git Stash Clear**

The **stash clear** command allows deleting all the available stashes at once. To delete all the available stashes, operate below command:

**$ git stash clear**

it will delete all the stashes that exist in the repository.

