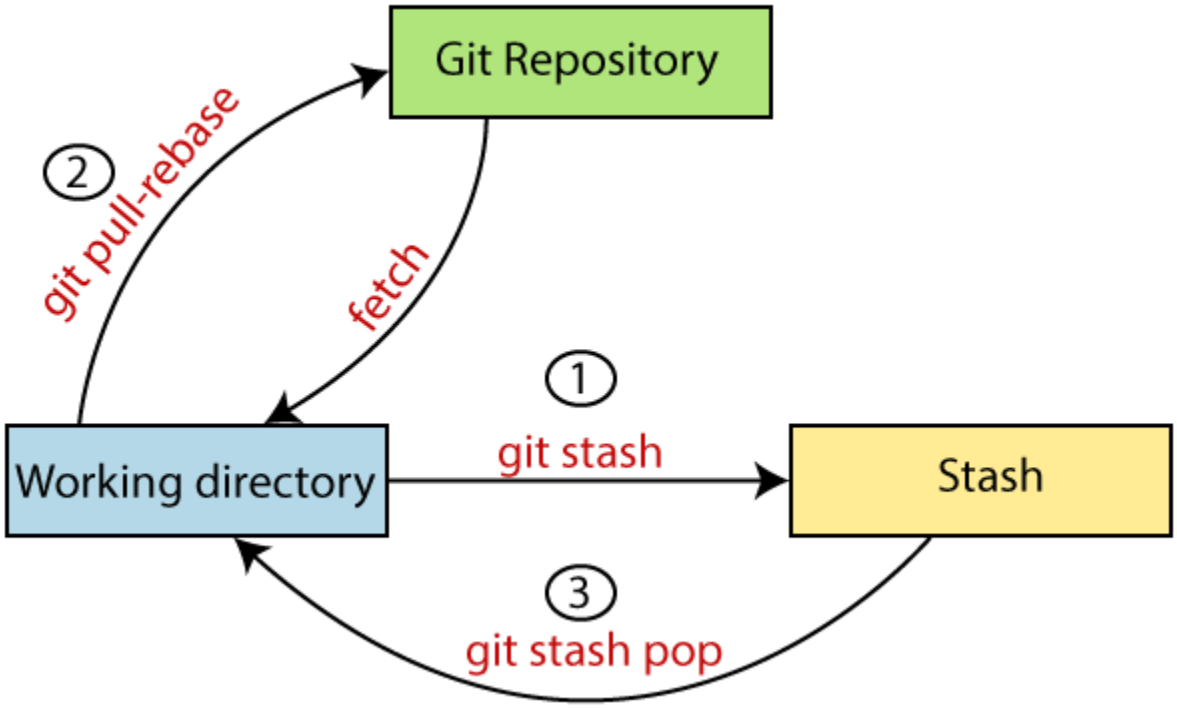
**Assignment no. 1**

## 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.

## 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:**

1. $ git status

```
HP@DESKTOP-5SO29UL MINGW64 /c/git_exmp/assignments (main)
$ vi demo.txt

HP@DESKTOP-5SO29UL MINGW64 /c/git_exmp/assignments (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   demo.txt
```

**Note: - From the above output, you can see the status that there are one untracked file demo.txt available in the repository. To save it temporarily, we can use the stash command.**

```
HP@DESKTOP-5SO29UL MINGW64 /c/git_exmp/assignments (main)
$ git stash
Saved working directory and index state WIP on main: 5ceed62 Data_Set Added!
```

We can check the status of the repository.

```
HP@DESKTOP-5SO29UL MINGW64 /c/git_exmp/assignments (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

**Note: - This point, you can switch between branches and work on them.**

**Git Stash Save (Saving Stashes with the message):**

To stash a change with a message, run the below command:

$ git stash save "<Stashing Message>"

```
HP@DESKTOP-5SO29UL MINGW64 /c/git_exmp/assignments (main)
$ git stash save "Edited demo.txt file!"
Saved working directory and index state On main: Edited demo.txt file!
```

**Git Stash List (Check the Stored Stashes)**

To check the stored stashes, run the below command:

```
$ git stash list
```

If we have more than one stash, then It will display all the stashes respectively with different stash id. It will show all the stashes with indexing as **stash@{0}: stash@{1}:** and so on.

```
HP@DESKTOP-5SO29UL MINGW64 /c/git_exmp/assignments (main)
$ git stash list
stash@{0}: On main: Edited demo.txt file!
stash@{1}: WIP on main: 5ceed62 Data_Set Added!
stash@{2}: WIP on main: 5ceed62 Data_Set Added!
```

**Git Stash Apply**

You can re-apply the changes that you just stashed by using the stash command. To apply the commit, use the stash command, followed by the apply option.

$ git stash apply

```
HP@DESKTOP-5SO29UL MINGW64 /c/git_exmp/assignments (main)
$ git stash apply
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   demo.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

**Note: -** 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.

$ git status

```
HP@DESKTOP-5SO29UL MINGW64 /c/git_exmp/assignments (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   demo.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

**Note: - From the above output, you can see that the repository is restored to its previous state before stash. It is showing output as "Changes not staged for commit."**

**In case of more than one stash, you can use "stash apply" command followed by stash index id to apply the particular commit.**

$ git stash apply <stash id>

```
HP@DESKTOP-5SO29UL MINGW64 /c/git_exmp/assignmer
$ git stash list stash@{1}
stash@{0}: On main: Edited demo.txt file!
stash@{1}: WIP on main: 5ceed62 Data_Set Added!
stash@{1}: WIP on main: 5ceed62 Data_Set Added!
stash@{2}: WIP on main: 5ceed62 Data_Set Added!
stash@{2}: WIP on main: 5ceed62 Data_Set Added!
```

**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)**

This 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.

$ git stash drop

```
HP@DESKTOP-5SO29UL MINGW64 /c/git_exmp/assignments (main)
$ git stash list
stash@{0}: On main: Edited demo.txt file!
stash@{1}: WIP on main: 5ceed62 Data_Set Added!
stash@{2}: WIP on main: 5ceed62 Data_Set Added!

HP@DESKTOP-5SO29UL MINGW64 /c/git_exmp/assignments (main)
$ git stash drop
Dropped refs/stash@{0} (613e495aaa2e34d90323c000fc7d1ac689b2ca94)
```

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.

We can also delete a particular stash from the queue. To delete a particular stash from the available stashes, pass the stash id in stash drop command.

**$ git stash drop <stash id>**

### 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.