- **git push --all origin**
  - Push all the branches to the remote repository.
- **git checkout <filename>**
  - If you had changed one file and have not committed yet, then running this command will erase everything you updated and make it same as the last committed version.
  - It doesn't effect any untracked file. (as git doesn't know about this file)
  - If you have already staged the file (using git add) then you are executing this command, then it doesn't effect this as the changes are already staged.
  - *Hence, if the file is not untracked & the changes are unstaged, then only the **git checkout <filename>** will reset this file to the last commit version.*
- **git restore <filename>**
  - Same as *git checkout <filename>*.
  - But it is recommended to use instead of *git checkout <filename>*.
  - git checkout should be used for switching among the branches only.
  - It has some additional features like you can rollback to a specific commit in the past.
    - ***--source=<commit>***

```
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git status
On branch main
nothing to commit, working tree clean

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ ls
dir1/  dir2/  test.txt  test2.txt

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ cat test.txt
Absdfaf
a;faf
 ahfsaf

  lkhfas'fas
  ;hhlhfasf

ahfhffas
1safjh


alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ echo "some some" > test.txt

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git status
On branch 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:   test.txt

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

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ cat test.txt
some some

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git restore test.txt

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git status
On branch main
nothing to commit, working tree clean
g
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ cat test.txt
Absdfaf
a;faf
 ahfsaf

  lkhfas'fas
  ;hhlhfasf

ahfhffas
1safjh
```

  - ***git restore --staged <filename>*** : It unstaged the file from staging area.
    - Let the file is already committed, now you have changed (added or deleted) some lines in side the file, and staged it (git add command). Now you are using this command *git restore --staged <filename>*, it'll only unstage the current updation (means the added/deleted lines) as it the file was pushed to tracking stage(git add) in the previous commit.

- Ex:
  - The file is committed.
  - Now changed come lines. And executed *git add .*
  - Now executed *git restore --staged <filename>*
  - Now the changed in side the fill will be there, but unstaged.
  - Now execute *git restore <filename>*
  - Now the changes done in this file will be no more.

```
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git status
On branch main
nothing to commit, working tree clean

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ ls
dir1/  dir2/  test.txt  test2.txt

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ cat test.txt
anything

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ echo "some some" > test.txt

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git add .
warning: in the working copy of 'test.txt', LF will be replaced by CRLF the next time Git touches it

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test.txt

ca
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ cat test.txt
some some

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git restore test.txt

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test.txt


alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ cat test.txt
some some

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git restore --staged test.txt

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git status
On branch 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:   test.txt

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

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ cat test.txt
some some

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git restore test.txt

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ cat test.txt
anything

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git status
On branch main
nothing to commit, working tree clean

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
```

- **git diff**
  - It shows the differences between *working directory* and *staging area*.

```
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git status
On branch main
nothing to commit, working tree clean

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ ls
dir1/  dir2/  test.txt  test2.txt

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ echo "some some hola hola" > test.txt

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git status
On branch 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:   test.txt

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

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git diff
warning: in the working copy of 'test.txt', LF will be replaced by CRLF the next time Git touches it
diff --git a/test.txt b/test.txt
index 1ba4650..9b2dba4 100644
--- a/test.txt
+++ b/test.txt
@@ -1 +1 @@
-anything
+some some hola hola

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ touch "hhhhhhhhhhhhheeeeeeeeeeeelllllllllloooooooooooooooooooooo" > test3.txt

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git status
On branch 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:   test.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        hhhhhhhhhhhhheeeeeeeeeeeelllllllllloooooooooooooooooooooo
        test3.txt

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

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git diff
warning: in the working copy of 'test.txt', LF will be replaced by CRLF the next time Git touches it
diff --git a/test.txt b/test.txt
index 1ba4650..9b2dba4 100644
--- a/test.txt
+++ b/test.txt
@@ -1 +1 @@
-anything
+some some hola hola

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ it'll not show about the test3.txt as it was not in the staged area.
```

- **git diff --cached**
  - It shows the difference between *staging area* and *last commit (HEAD)*.
- **git revert** and **git stash:**
  - Let you created one file test.txt, insert some texts i.e."Hello", staged it, committed it and pushed it to main branch.

```
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ echo "Hello" > dir1/test.txt

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git add .
warning: in the working copy of 'dir1/test.txt', LF will be replaced by CRLF the
gi
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git commit -m "created dir1/test.txt"
[main 9ad4ec9] created dir1/test.txt
 1 file changed, 1 insertion(+)
 create mode 100644 dir1/test.txt

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git push origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 363 bytes | 363.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Alok905/test_git.git
   a878090..9ad4ec9  main -> main

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ cat dir1/test.txt
Hello
```

- Now let you appended one more line i.e. “bye” inside that test.txt file. (now content is: *Hello \nBye*)
- Now you executed the command: *git revert HEAD. It’ll give some error saying..*

```
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ echo "Bye" >> dir1/test.txt

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ cat dir1/test.txt
Hello
Bye

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git log --oneline
9ad4ec9 (HEAD -> main, origin/main) created dir1/test.txt
a878090 did
e6ff520 Reapply "undoing test.txt"
9e464c6 Revert "test.txt created"
9ac4b8b test.txt created
eabc214 test.txt deleted
1c00548 Revert "checkinf revert"
4636705 test.txt created
048127d commit done
c074b29 Reapply "commit 1: new cmt"
e758be8 Revert "commit 1: Sm chng hd bn dn bt i dnt wnt amr"
a00142a commit 1
7293f76 some changes
e46b729 changd
7e5b6d9 (origin/sprint1, sprint1) dir deleted and dir2 created
17e6097 some test files have been created.

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git revert HEAD
error: Your local changes to the following files would be overwritten by merge:
        dir1/test.txt
Please commit your changes or stash them before you merge.
Aborting
fatal: revert failed
```

```
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git add .
warning: in the working copy of 'dir1/test.txt', LF will be replaced by CRLF the
g
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   dir1/test.txt


alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git commit -m "added one more line"
[main b6b81e7] added one more line
 1 file changed, 1 insertion(+)
g
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 364 bytes | 364.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Alok905/test_git.git
   9ad4ec9..b6b81e7  main -> main

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git log --oneline
b6b81e7 (HEAD -> main, origin/main) added one more line
9ad4ec9 created dir1/test.txt
a878090 did
e6ff520 Reapply "undoing test.txt"
9e464c6 Revert "test.txt created"
9ac4b8b test.txt created
eabc214 test.txt deleted
1c00548 Revert "checkinf revert"
4636705 test.txt created
048127d commit done
c074b29 Reapply "commit 1: new cmt"
e758be8 Revert "commit 1: Sm chng hd bn dn bt i dnt wnt amr"
a00142a commit 1
7293f76 some changes
e46b729 changd
7e5b6d9 (origin/sprint1, sprint1) dir deleted and dir2 created
17e6097 some test files have been created.
```

-
  - I'll undo this commit afterwards (bcs If I would undo the previous commit then the file would have been deleted)
- I appended a new line to the file and staged it.

```
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ cat dir1/test.txt
Hello
Bye

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ echo "Hola Hola" >> dir1/test.txt

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ cat dir1/test.txt
Hello
Bye
Hola Hola

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git add .
warning: in the working copy of 'dir1/test.txt', LF will be replaced by CRLF the

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   dir1/test.txt
```

-
- I appended one more line to the file "Bola Bola" but didn't stage it.

```
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ echo "Bolaa Bolaa" >> dir1/test.txt

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ cat dir1/test.txt
Hello
Bye
Hola Hola
Bolaa Bolaa

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   dir1/test.txt

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:   dir1/test.txt
```

- Now If I do git revert HEAD it'll give error.

```
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git log --oneline
b6b81e7 (HEAD -> main, origin/main) added one more line
9ad4ec9 created dir1/test.txt
a878090 did
e6ff520 Reapply "undoing test.txt"
9e464c6 Revert "test.txt created"
9ac4b8b test.txt created
eabc214 test.txt deleted
1c00548 Revert "checkinf revert"
4636705 test.txt created
048127d commit done
c074b29 Reapply "commit 1: new cmt"
e758be8 Revert "commit 1: Sm chng hd bn dn bt i dnt wnt amr"
a00142a commit 1
7293f76 some changes
e46b729 changd
7e5b6d9 (origin/sprint1, sprint1) dir deleted and dir2 created
17e6097 some test files have been created.

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git revert HEAD
error: your local changes would be overwritten by revert.
hint: commit your changes or stash them to proceed.
fatal: revert failed
```

- You need to ***stash*** it now.
  - Stash means the changes that have been done (both staged & unstaged) will be saved in a stash stack so that the working directory will be clean.

```
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git stash list

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git stash
Saved working directory and index state WIP on main: b6b81e7 added one more line

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        dir1.test.txt

nothing added to commit but untracked files present (use "git add" to track)

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git stash list
stash@{0}: WIP on main: b6b81e7 added one more line
```

- Now you can **revert** it.

```
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git revert HEAD
```

```
Revert "TRYING TO COMMIT AFTER STASH"

This reverts commit b6b81e76100c90d3a3e58a912b6a794688fe7f9a.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch main
# Changes to be committed:
#       modified:   dir1/test.txt
#
# Untracked files:
#       dir1.test.txt
```

```
[main b7c1161] Revert "TRYING TO COMMIT AFTER STASH"
 1 file changed, 1 deletion(-)

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        dir1.test.txt

nothing added to commit but untracked files present (use "git add" to track)

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ cat dir1/test.txt
Hello

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git log --oneline
b7c1161 (HEAD -> main) Revert "TRYING TO COMMIT AFTER STASH"
b6b81e7 (origin/main) added one more line
9ad4ec9 created dir1/test.txt
a878090 did
e6ff520 Reapply "undoing test.txt"
9e464c6 Revert "test.txt created"
9ac4b8b test.txt created
eabc214 test.txt deleted
1c00548 Revert "checkinf revert"
4636705 test.txt created
048127d commit done
c074b29 Reapply "commit 1: new cmt"
e758be8 Revert "commit 1: Sm chng hd bn dn bt i dnt wnt amr"
a00142a commit 1
7293f76 some changes
e46b729 changd
7e5b6d9 (origin/sprint1, sprint1) dir deleted and dir2 created
17e6097 some test files have been created.
```

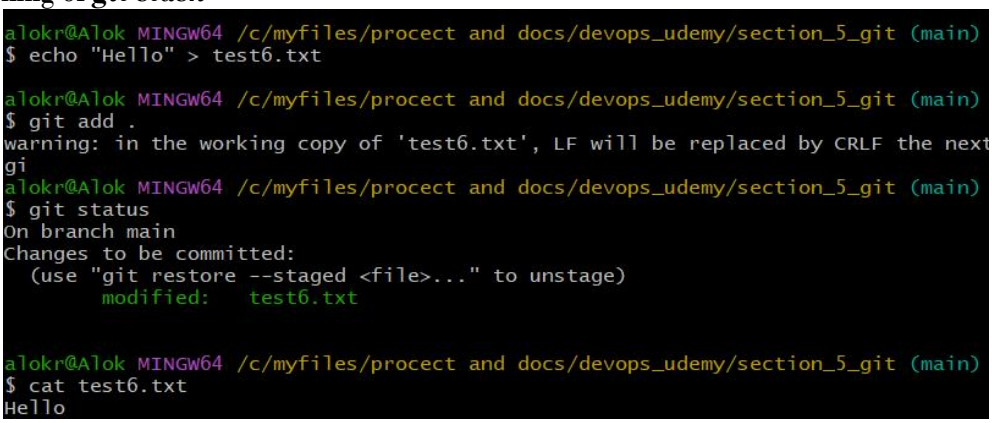
- The last commit was previously ***b6b81e7***, which was for added the line "Bye". Now the latst commit is ***b7c1161*** which deleted that line "Bye". Only "Hello" is there.

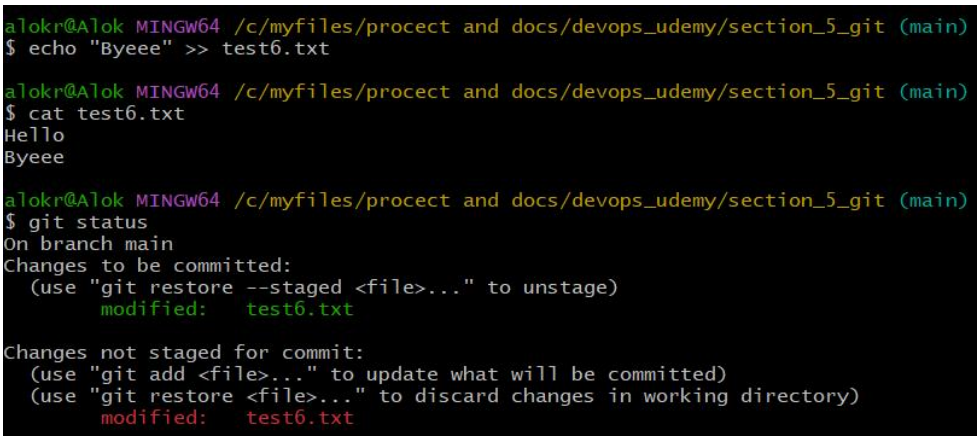
- Now, to get the ***stashed*** changes, you need to execute ***git stash pop***

```
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git stash pop
Auto-merging dir1/test.txt
CONFLICT (content): Merge conflict in dir1/test.txt
On branch main
Unmerged paths:
  (use "git restore --staged <file>..." to unstage)
  (use "git add <file>..." to mark resolution)
        both modified:   dir1/test.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        dir1.test.txt

no changes added to commit (use "git add" and/or "git commit -a")
The stash entry is kept in case you need it again.

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git status
On branch main
Unmerged paths:
  (use "git restore --staged <file>..." to unstage)
  (use "git add <file>..." to mark resolution)
        both modified:   dir1/test.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        dir1.test.txt

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

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ cat dir1/test.txt
Hello
<<<<<<< Updated upstream
=======
Bye
Hola Hola
>>>>>>> Stashed changes
```

- ***git reset***
  - Unlike *git stash* this command i.e. ***git reset*** doesn't create another commit.
  - It means, let you have 4 commits: c1, c2, c3, c4 and you want to go back to commit c1.
  - Then it doesn't create any other commit c5 and roll back to c1, it'll delete c2, c3, c4 and go to c1.

```
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git log --oneline
cb7587b (HEAD -> main) Revert "prev commit of deleting files"
5eda11a (origin/main) created def.txt, def2.txt
16a8471 Created some files abc.txt, abc1.txt
e354153 abcd
4130337 dddddd
5183c96 afsdfsf
72cdd6d Revert "aa"
840bbd2 aa
0ad0909 done
b7c1161 Revert "TRYING TO COMMIT AFTER STASH"
b6b81e7 added one more line
9ad4ec9 created dir1/test.txt
a878090 did
e6ff520 Reapply "undoing test.txt"
9e464c6 Revert "test.txt created"
9ac4b8b test.txt created
eabc214 test.txt deleted
1c00548 Revert "checkinf revert"
4636705 test.txt created
048127d commit done
c074b29 Reapply "commit 1: new cmt"
e758be8 Revert "commit 1: Sm chng hd bn dn bt i dnt wnt amr"
a00142a commit 1
7293f76 some changes
e46b729 changd
7e5b6d9 (origin/sprint1, sprint1) dir deleted and dir2 created
17e6097 some test files have been created.

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git reset --hard b7c1161
HEAD is now at b7c1161 Revert "TRYING TO COMMIT AFTER STASH"

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git log --oneline
b7c1161 (HEAD -> main) Revert "TRYING TO COMMIT AFTER STASH"
b6b81e7 added one more line
9ad4ec9 created dir1/test.txt
a878090 did
e6ff520 Reapply "undoing test.txt"
9e464c6 Revert "test.txt created"
9ac4b8b test.txt created
eabc214 test.txt deleted
1c00548 Revert "checkinf revert"
4636705 test.txt created
048127d commit done
c074b29 Reapply "commit 1: new cmt"
e758be8 Revert "commit 1: Sm chng hd bn dn bt i dnt wnt amr"
a00142a commit 1
7293f76 some changes
e46b729 changd
7e5b6d9 (origin/sprint1, sprint1) dir deleted and dir2 created
17e6097 some test files have been created.
```

- **NOTE: *git revert* will only undoes the perticular commit and doesn't effect later commits but *git reset* will remove all the later commits.**
- there is a already committed file: test.txt
  - I added one line "First Line" and committed. (C1)
  - I added one more line "Second Line" and committed. (C2)
  - I added one more line "Third Line" and committed. (C3)
  - I added one more line "Fourth Line" and committed. (C4)
  - I added one more line "Fifth Line" and committed. (C5)
- *Now I did git revert C2. Will it give any error?*
- It usually won't cause an error unless:
  - The line added in C2 ("Second Line") was modified in later commits (C3–C5).
  - For example, if C3 edited "Second Line" instead of just appending "Third Line", Git may throw a merge conflict during the revert.
- Working of ***git stash***
  - 

  - 

```
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test6.txt

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:   test6.txt


alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git stash list

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git stash
warning: in the working copy of 'test6.txt', LF will be replaced by CRLF the next
Saved working directory and index state WIP on main: 0ad0909 done

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git log --oneline
0ad0909 (HEAD -> main, origin/main) done
b7c1161 Revert "TRYING TO COMMIT AFTER STASH"
b6b81e7 added one more line
9ad4ec9 created dir1/test.txt
a878090 did
e6ff520 Reapply "undoing test.txt"
9e464c6 Revert "test.txt created"
9ac4b8b test.txt created
eabc214 test.txt deleted
1c00548 Revert "checkinf revert"
4636705 test.txt created
048127d commit done
c074b29 Reapply "commit 1: new cmt"
e758be8 Revert "commit 1: Sm chng hd bn dn bt i dnt wnt amr"
a00142a commit 1
7293f76 some changes
e46b729 changd
7e5b6d9 (origin/sprint1, sprint1) dir deleted and dir2 created
17e6097 some test files have been created.

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git stash list
stash@{0}: WIP on main: 0ad0909 done

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ cat test6.txt

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git status
On branch main
nothing to commit, working tree clean
```

```
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git stash pop
On branch 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:   test6.txt

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (b053f84a7017164c14aa6a05f4afe30ddd95245c)

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git status
On branch 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:   test6.txt

no changes added to commit (use "git add" and/or "git commit -a")
g
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git stash list

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ cat test6.txt
Hello
Byeee
```

Steps:

- I added one line in a already created file “Hello”.
- Staged it
- Added one more line “Bye”
- Then ***stashed*** it
- Now the staged and unstaged changes were stashed and working tree became clean.
- Then *unstashed*
- Got everything in side the file.
- ***Note: The staged modification will be now in unstaged state.*** As git can’t guarantee that the unstaged changes are still valid or not.

- ***Github ssh login:***

```
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_u
$ cat .git/config
[core]
        repositoryformatversion = 0
        filemode = false
        bare = false
        logallrefupdates = true
        symlinks = false
        ignorecase = true
[remote "origin"]
        url = https://github.com/Alok905/test_git.git
        fetch = +refs/heads/*:refs/remotes/origin/*
```

  - Here the authentication is based on http so you need the password. There is a chance of loosing or leaking of the password.

```
alokr@Alok MINGW64 ~
$ pwd
/c/Users/alokr

alokr@Alok MINGW64 ~
$ ls .ssh
known_hosts  known_hosts.old

alokr@Alok MINGW64 ~
$ rm -rf .ssh/*

alokr@Alok MINGW64 ~
$ ls -al .ssh
total 24
drwxr-xr-x 1 alokr 197609 0 May 27 20:21 ./
drwxr-xr-x 1 alokr 197609 0 May 27 20:09 ../

alokr@Alok MINGW64 ~
$ ssh-keygen.exe
Generating public/private ed25519 key pair.
Enter file in which to save the key (/c/Users/alokr/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/alokr/.ssh/id_ed25519
Your public key has been saved in /c/Users/alokr/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:ShDSiHFSkqMoM3p2cfgZ4Aivbk+iCfBHfeKk6XKyZUk alokr@Alok
The key's randomart image is:
+--[ED25519 256]--+
|+=+o.            |
|=+..o.           |
|o+ o.o           |
|* o +oo          |
|++  E+=oS        |
|+.oo.Bo+         |
|o+.oB o          |
|o+==.            |
|+ o*.            |
+----[SHA256]-----+

alokr@Alok MINGW64 ~
$ ls .ssh
id_ed25519  id_ed25519.pub

alokr@Alok MINGW64 ~
$
```

  - 2 keys are there: *id_ed25519 (private key), id_ed25519.pub (public key)*

```
alokr@Alok MINGW64 ~
$ cat .ssh/id_ed25519
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5[redacted]AAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
QyNTUxOQAAACAWOKMoCP8i0IPOOmN[redacted]x6FUIgGOU5BrfCVgqB1cg
CgAAAAtzc2gtZWQyNTUxOQAAAC[redacted]P8i0IPOOmNtmx5V[redacted]
AAAECTFvh4y7YbonQ08F0ZHCDJoQ[redacted]YCqU+YF8WfC9RY4oygI/yLQg/Q6Y22bHlVf
HoVQiAbRTkGt8JWCoHVyAAAACmFsb2t[redacted]Fsb2sBAgM=
-----END OPENSSH PRIVATE KEY-----

alokr@Alok MINGW64 ~
$ cat .ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC[redacted]ygI/yLQg/Q6Y22bHlVfHoVQiAbRTkGt8JWCoHVy alokr@Alok
```

- Add the public key in your GitHub account (not GitHub repo). *Settings > SSH and GPG keys > add ssh key*

```
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git clone git@github.com:Alok905/test_git.git ./
Cloning into '.'...
remote: Enumerating objects: 70, done.
remote: Counting objects: 100% (70/70), done.
remote: Compressing objects: 100% (51/51), done.
Receiving objects: 100% (70/70), 435.80 KiB | 670.00 KiB/s, done.(from 0)

Resolving deltas: 100% (22/22), done.

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ ls
abc.txt  abc1.txt  def.txt  def2.txt  dir1/  dir2/

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$
```

Git Tags, Semantic Versioning

```
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ echo "First line added. v1.0.0" > t.txt

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git tag -a v1.0.0 -m "Release 1.0.0"

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git add .
warning: in the working copy of 't.txt', LF will be replaced by CRLF the next tim
g
alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git commit -m "v1.0.0 commit"
[main c7e5b0d] v1.0.0 commit
 1 file changed, 1 insertion(+)

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 282 bytes | 282.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:Alok905/test_git.git
   5a49e26..c7e5b0d  main -> main

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git push --tags origin main
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 169 bytes | 169.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Alok905/test_git.git
 * [new tag]         v1.0.0 -> v1.0.0

alokr@Alok MINGW64 /c/myfiles/procect and docs/devops_udemy/section_5_git (main)
$ git tag
v1.0.0
```